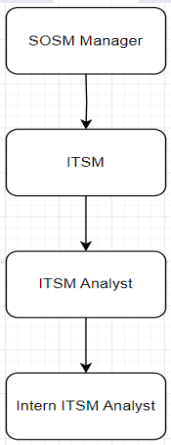
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Posisi pada pelaksanaan program kerja magang di PT Kalbe Farma Tbk, mahasiswa ditempatkan pada posisi *IT Service Management (ITSM)* yang bertujuan memperoleh pengalaman praktik kerja secara menyeluruh. Khususnya bagi mahasiswa Sistem Informasi dengan peminatan *Big Data Analytics*, peran mahasiswa berfokus pada analisis data serta melakukan pemrosesan data guna mendukung berbagai kebutuhan perusahaan dalam melakukan analisis data dan pemrosesan informasi, sehingga memerlukan pengerjaan tugas sesuai dengan kebutuhan perusahaan. Secara spesifik, posisi *intern* dalam divisi *SOSM* memiliki beberapa tugas utama seperti pengelolaan data dan analisis informasi, termasuk penyusunan laporan bulanan berdasarkan data yang tersedia di *dashboard* yang telah dibuat sebelumnya. Selain itu, mahasiswa magang juga berperan dalam analisis tren *tiket* yang mengalami permasalahan, pengembangan *dashboard* sesuai kebutuhan tim *IT* lainnya, serta analisis grafik yang dibuat untuk mendukung proses pengambilan keputusan secara lebih efektif. Berikut merupakan struktur dari kedudukan dan koordinasi pada *SOSM*.



Gambar 3.1 Struktur Kedudukan SOSM Kalbe Farma [18].

Pada tim SOSM dipegang oleh Bapak Bambang dengan jabatan *CIT Support & Services Manager*, yang melakukan analisis pekerjaan *intern* yang sudah dibuat, dan melakukan pemberitahuan terkait project baru. pada tim *ITSM* analyst supervisi *intern* adalah Bapak Agus Hermawan sebagai Office Data Analyst *(ITSM)*.

3.2 Tugas dan Uraian Kerja Magang

Dalam melakukan praktik magang selama 6 bulan dari pada 13 Januari 2025 hingga 20 Juni 2025, terdapat berbagai tugas yang dikerjakan berdasarkan proyek dilakukan dan disesuaikan dengan kemampuan mahasiswa magang. Berikut merupakan proyek utama yang dikerjakan berdasarkan timeline pada tabel 3.1 Periode Pelaksanaan Magang.

Tabel 3.1 Periode Pelaksanaan Magang

No	Kegiatan magang	Waktu Pengerjaan	Hasil Kinerja
	Pengenalan perusahan dan tugas/project yang dilakukan	Minggu 1-2	Memahami stuktur pada perusahaan Kalbe Farma
Pengelolahan Data Melalui Python			
1	Melakukan *data preprocessing* pada *file json (Project Monitoring)*	Minggu 3	Menjadi Data yang terstruktur serta merubah menjadi format file excel
SQL dan Optimasi Query & Pembuatan Dashboard & Penggunaan Metabase			
2	Membuat Query SQL pada metabase dari *data preprocessing* project *Monitoring* dan pembuatan *dashboard Project Monitoring*	Minggu 4 - 5	Menjadi *dashboard* yang sesuai kebutuhan sekaligus melakukan analisis.
3	Pembuatan *Dashboard Service Request* pada BU Perusahaan B7 (Metabase)	Minggu 5 - 6	Hasil *dashboard* yang sesuai dengan *ITSM*.
Interaksi dengan API & Pengelolahan Data dengan Python			
4	Melakukan *data scraping* menggunakan *API Microsoft* pada data bitlocker 365 (python)	Minggu 6 - 7	Hasil data dapat ketarik melalui *API* dan dimasukan ke database.
5	Melakukan *data scraping* menggunakan API Microsoft pada data teams (python)	Minggu 7 - 8	Hasil data dapat ketarik melalui *API* dan dimasukan ke database.
6	Melakukan *Data preprocessing* dari data *Microsoft teams (python)*	Minggu 9 - 12	Data menjadi lebih terstruktur dan siap untuk buat visualisasi

No	Kegiatan magang	Waktu Pengerjaan	Hasil Kinerja
Pembuatan Laporan dan Dashboard			
7	Pembuatan *Dashboard* dari *data Microsoft Teams* dengan *API*	Minggu 12 - 14	Menjadi Sebuah dashboard yang interaktif dan informatif bagi *audience*
8	Data *Cleanising* dan *comparison* pada *data kav* serta *purchasing (python)*	Minggu 12 - 13	Menjadi sebuah *Summary* Data perbandingan untuk dilakukan laporan kepada team purchasing dan tim Microsoft Admin
Penggunaan Metabase			
9	*Engineer resolution Analysis* melalui metabase serta excel dan Analysis pada *Service katalog*	Minggu 13	Menjadi Sebuah *report* operational yang sederhana dan informatif.
10	Komparasi *Data User* dan *Bitlocker 365*	Minggu 14	Menjadi data komparasi yang dibutuhkan oleh tim *Microsoft Admin*
11	Data Mail *Cleansing, merge*, dan *compare persentase*	Minggu 14	Menjadi data komparasi yang dibutuhkan oleh tim *Microsoft Admin*
12	Pembuatan *Report* bulanan pada *service ITSM*, dengan perkembangan *incident, service Request* dan *summary* (Maret 2025).	Minggu 14 - 15	Hasil tersebut menjadi laporan bulanan pada operational pharma maret 25.
13	Penarikan *data Sign In log* menggunakan *API microsoft*.	Minggu 15-17	Hasil data dapat ketarik dan diberikan tim *microsoft admin*.
14	Pembuatan *Report* bulanan pada *service ITSM*, dengan perkembangan *incident, service Request* dan *summary* (Apr 2025).	Minggu 17 - 18	Hasil *dashboard* tersebut menjadi laporan bulanan pada operational pharma april 25.
Pembuatan Model Prediksi			
15	Pembuatan Model pada *Waiting Reasons*	Minggu 15-20	Menjadi model yang siap untuk di implementasi dalam Operational Kalbe *Group*
16	Pembuatan *Report bulanan* pada *service ITSM*, dengan	Minggu 20	Hasil *Report* yang dapat menganalisis tiket Service Request dan Incident

No	Kegiatan magang	Waktu Pengerjaan	Hasil Kinerja
	perkembangan *incident, service Request* dan *summary* (Mei 2025).		

Pada pelaksanaan tugas maupun proyek untuk mahasiswa magang di bagian *ITSM*, mahasiswa magang juga diajarkan untuk menggunakan berbagai *tools* pendukung yang relevan dengan peran sebagai *IT Service Management Analyst*. Penggunaan *tools* menjadi peran penting dalam meningkatkan proses pembelajaran sebagai *Analyst*, sekaligus memberikan pengalaman praktis dalam penggunaan perangkat lunak. Mahasiswa akan memanfaatkan perangkat lunak tersebut mulai dari pengolahan data, pembuatan laporan, analisis kinerja *software*, hingga mengembangkan sebuah *dashboard* interaktif untuk mendukung tim *SOSM*. Berikut merupakan beberapa *tools* yang digunakan pada kebutuhan divisi *ITSM*:

a. Python

Python merupakan bahasa pemrograman pada tingkat lanjut yang mengalami pertumbuhan pesat dari jumlah pengguna dalam beberapa tahun terakhir [10]. *Python* diciptakan pada tahun 1991 oleh seorang bernama Guido van Rossum yang berasal dari Belanda, dan kini *Python* menjadi salah satu bahasa pemrograman terbaik di dunia, terutama digunakan dalam industri *data analysis*. *Python* tidak hanya dapat digunakan untuk pengembangan aplikasi modern, tetapi juga sangat mendukung teknik *machine learning*, karena memiliki pustaka yang melimpah serta kemudahan dalam penggunaannya. Hal ini menjadikannya populer di kalangan ilmuwan, akademisi, analis data, maupun pengembang aplikasi [10]. Untuk tim *IT Service Management*, aplikasi *Python* digunakan dalam berbagai keperluan seperti pembacaan data dari berbagai sumber, penarikan data dari berbagai *API*, *data preprocessing*, *data cleansing*, hingga penjadwalan (*scheduling*) data secara *real-time*.

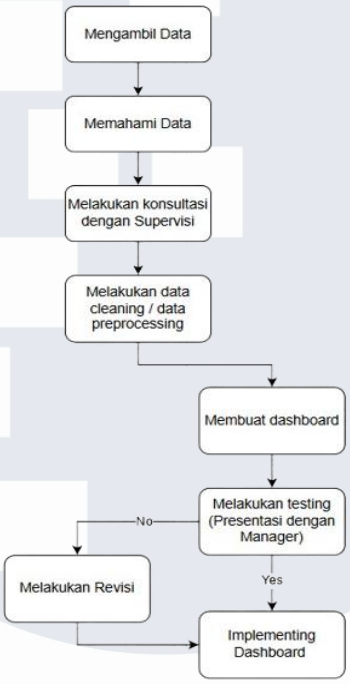
b. SQL Server Management Studio

SQL Server Management Studio merupakan sistem manajemen basis data relasional (*Relational Database Management System* / *RDBMS*) yang dikembangkan oleh *Microsoft*, yang dirancang untuk menyimpan sekaligus mengolah data dari skala kecil hingga besar [11]. Alat ini banyak digunakan pada beberapa sektor, salah satunya sektor kesehatan seperti pada perusahaan *Kalbe Farma*. Bahasa utama yang digunakan untuk melakukan *kueri* atau perintah dalam *SQL Server* yaitu *Transact-SQL*. *Transact-SQL* merupakan implementasi standar dari *SQL ANSI/ISO* yang dikembangkan secara khusus oleh *Microsoft* dan *Sybase* [11]. Pada tim *ITSM Analyst* di *Kalbe Farma*, alat ini digunakan untuk keperluan seperti pembacaan data, *filtering*, serta pembuatan *database* baru.

c. Metabase

Metabase merupakan platform yang berjenis *open-license*, yang memungkinkan karyawan perusahaan membuat pertanyaan dan mempelajari data turunan dari sumber data yang tersedia. Alat ini dapat melakukan *filter* dan *grouping* pada data sesuai dengan kebutuhan pengguna, tanpa memerlukan penggunaan bahasa *SQL*, meskipun jika diperlukan, *Metabase* juga menyediakan dukungan *SQL* bagi pengguna [12]. Terdapat beberapa jenis *license* selain yang gratis, yaitu tiga kategori utama: pertama, *license* gratis; kedua, *Premium Embedding License* berbayar (yang mencakup opsi *labeled embedding*); dan ketiga, *Metabase Commercial License* berbayar dengan fitur yang lebih lengkap dibandingkan lisensi sebelumnya [12]. Pada tim *ITSM Analyst*, *Metabase* digunakan untuk pembuatan *dashboard* dari data yang telah melalui proses *filtering* dan *grouping*, serta langsung dibuat visualisasi sesuai dengan kebutuhan pengguna.

Pengerjaan proyek di perusahaan Kalbe Farma memerlukan alur kerja yang terstruktur agar setiap tugas dapat diselesaikan secara efisien dan tepat sasaran. Berikut ini merupakan gambaran umum alur kerja dari tugas atau proyek yang dikerjakan selama masa magang di perusahaan Kalbe Farma pada Gambar 3.5:



Gambar 3.5 Alur Pekerjaan pada Kalbe Farma

Pada awal pengerjaan, terdapat proyek dari *supervisor*/tim yang bersangkutan, yang biasanya disertai data mentah yang diperlukan untuk analisis. Data yang diberikan bisa langsung dari tim, atau diperoleh melalui proses penarikan data menggunakan *API*. Setelah data diperoleh, langkah berikutnya adalah memahami struktur dan isi data secara menyeluruh. Untuk memastikan pemahaman data yang sesuai dengan kebutuhan tim/proyek, diperlukan konsultasi dan diskusi dengan *supervisor*. Tahapan ini sangat penting karena bertujuan untuk menjelaskan konteks data serta menentukan bagian mana dari data tersebut yang relevan untuk dianalisis.

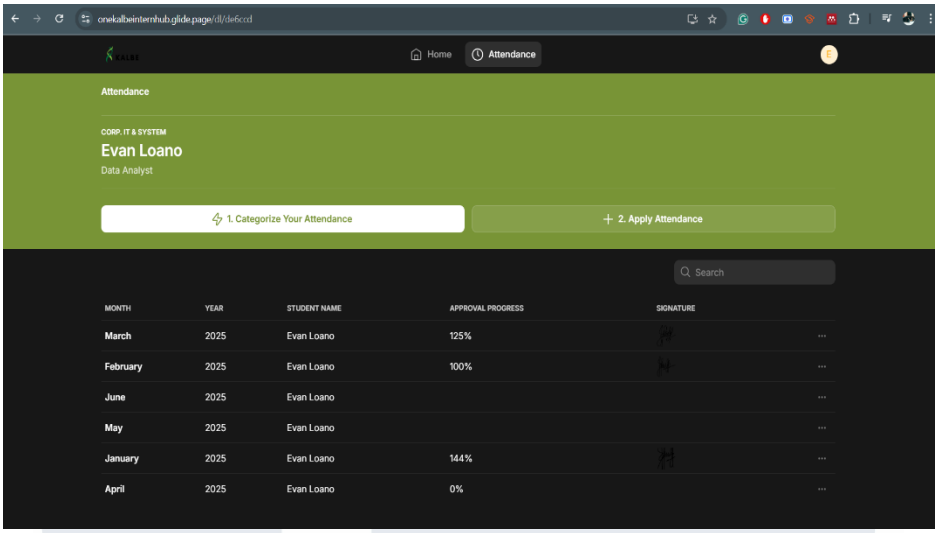
Setelah itu, dilakukan proses *cleaning*/*preprocessing* data agar data menjadi lebih siap dan terstruktur untuk divisualisasikan. Langkah selanjutnya adalah melakukan pembuatan visualisasi dalam bentuk *dashboard* yang interaktif sesuai dengan kebutuhan tim. Salah satu contohnya adalah data penggunaan *Microsoft*

Teams; dari *dashboard* yang dibuat, kita akan melakukan presentasi kepada *manager* terkait visualisasi yang telah disusun.

Setelah *dashboard* selesai dibuat, hasilnya akan dipresentasikan kepada *manager* atau pihak terkait untuk mendapatkan saran dan evaluasi. Apabila *dashboard* tersebut telah sesuai dengan kebutuhan dan kriteria, maka akan dilanjutkan ke tahap implementasi. Namun, jika masih terdapat kekurangan dan perlu penyesuaian, maka revisi akan dilakukan hingga *dashboard* siap diterapkan.

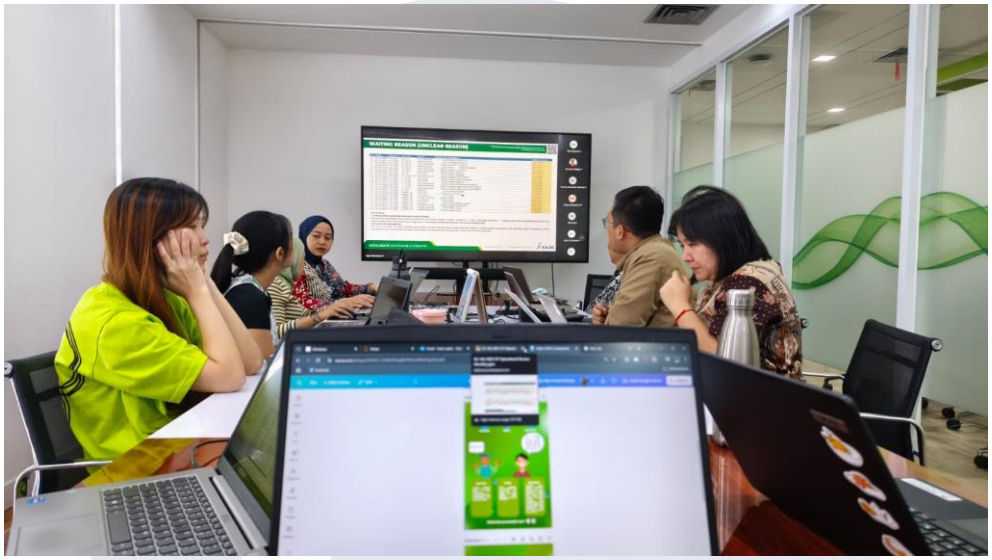
3.2.1 Pengenalan Lingkungan

Saat masuk, diawali dengan pengenalan lingkungan kerja dan anggota tim, lalu mempelajari masing-masing divisi yang ada di *CIT* serta tugas-tugasnya. Mahasiswa juga diperkenalkan dengan lini bisnis pada perusahaan *Kalbe Farma*, termasuk sejarah, *business unit*, dan proses produksi barang. Setelah itu, terdapat rapat mengenai perkenalan proyek yang akan dikerjakan oleh tim *ITSM Analyst*, yaitu pembuatan *dashboard* bagi tim internal maupun anak perusahaan. Tugas-tugas tersebut akan dieksekusi menggunakan *tools* seperti *Python*, *Microsoft SQL Server*, dan *Metabase* yang digunakan dalam lingkungan kerja di *Kalbe Farma*. Pada minggu pertama, mahasiswa ditawari penggunaan laptop yang disediakan oleh kantor, serta diminta untuk mengaktifkan *VPN FortiClient* (untuk mendukung *Work From Home*/*WFH*), mengatur *Outlook* dan *Teams*, serta melakukan aktivasi akun pada website *One Kalbe* (Gambar 3.6) untuk keperluan absensi harian. Selain itu, juga diberikan akses ke akun *helpdesk* (dengan alamat *email* berdomain *kalbe*), sebagaimana ditunjukkan pada gambar berikut.



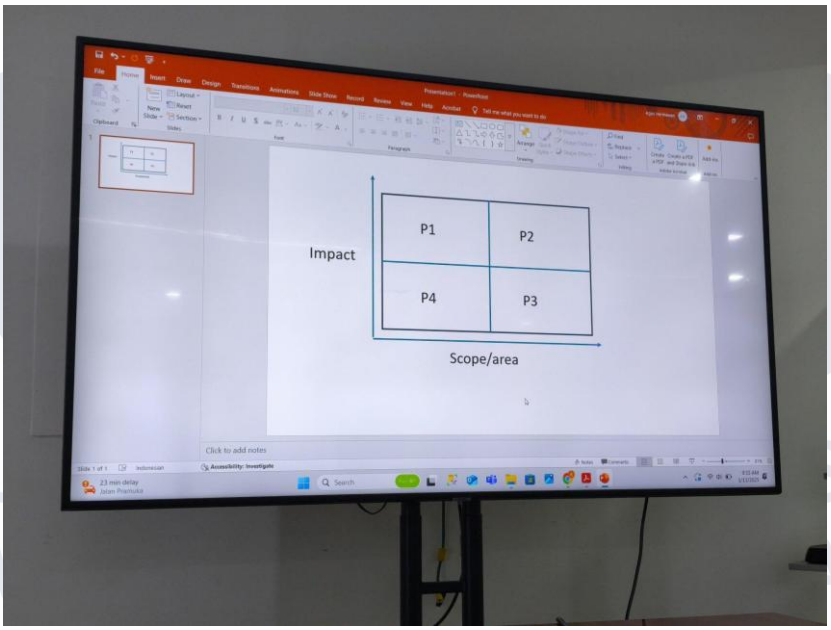
Gambar 3.6 *Website* Absensi *One Kalbe*

Gambar 3.7 Ruangan *CIT*



Gambar 3.8 Menghadiri *Meeting*

Selain itu terdapat beberapa meeting yang perlu hadir untuk memahami terkait *ITSM* pada gambar 3.8, lalu melakukan laporan bulanan mengenai kinerja dukungan IT pada pengguna. melakukan meeting dengan departemen atau divisi lain terkait *dashboard* atau proyek yang diperlukan.



Gambar 3.9 Penjelasan dari area *ITSM*

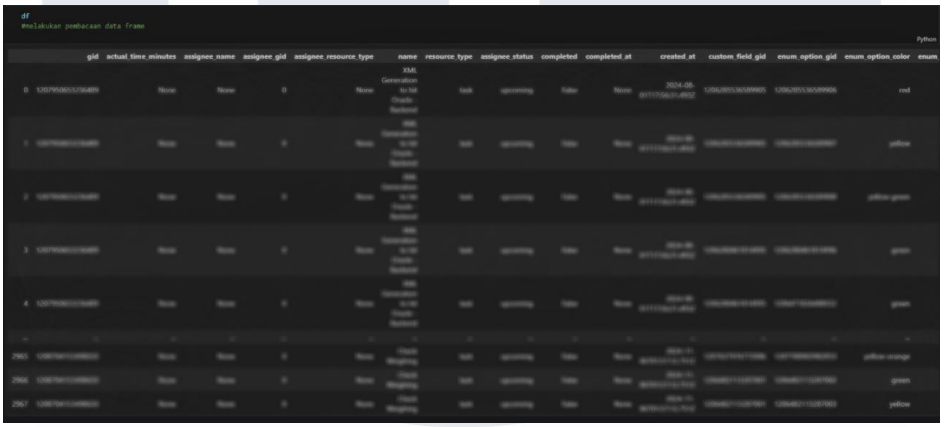
Gambar 3.9 merupakan penjelasan terkait area dan *impact ITSM* yang dimana semakin besar *impact* dan area yang berdampak maka akan dikerjakan harus dengan sesegera mungkin, *P1* merupakan area yang dimana berdampak sangat signifikan seperti permasalahan keseluruhan *server down*, *Dirut* yang mengalami masalah pada *device*-nya, dan kejadian lainnya, lalu untuk *P2* merupakan area yang dimana berdampak tapi tidak mengganggu pada kinerja bisnis sepenuhnya, seperti kasus *email* yang mengalami proses lambat walaupun bisa dikirim, *internet* atau *jaringan* yang *down* performanya tapi tetap berjalan koneksinya, dan lainnya, yang berikutnya *P3* merupakan area yang masalah berdampak sedang, seperti contoh beberapa pengguna tidak bisa *reset password* ketika lupa/ingin mengganti *password*, *request aplikasi* yang paling terbaru / melakukan *install apps*, dan lainnya, yang terakhir *P4* yang dimana area yang masalah kecil / tidak urgensi, seperti pada melakukan *email* baru pada karyawan baru ketika masih belum bekerja, pengguna ingin *font* baru pada *tools* seperti *Word*, dan lainnya.

3.2.2 Proyek/tugas yang dikerjakan

3.2.2.1 *Preproccesing data Project Management (PPM CDT)*

Gambar 3.10 Parsing Json File.

Gambar 3.10. Pada minggu ke-3, aktivitas pertama yaitu melakukan membaca file *JSON*, *parsing JSON file* lalu mengubah menjadi *DataFrame* dan luaran-nya menjadi *Excel*. dan kode ini melakukan penanganan file *JSON*, melakukan pengambilan *key data* pada *JSON*, melakukan ekstraksi pada file yang kompleks lalu menyusun data menjadi *list dictionaries* lalu mengubah menjadi *DataFrame* struktur seperti pada *Excel*, dan menangani *error* seperti format *JSON* tidak valid atau data tidak ditemukan.



Gambar 3.11 Membaca data dan Mengecek kolom.

Pada gambar 3.11, Setelah melakukan proses *data parsing* pada file *JSON* berhasil dilakukan, langkah berikutnya membaca data yang telah disimpan pada variabel *DataFrame*, untuk mengetahui kualitas dan struktur data dilakukan dengan menggunakan *info()* yang dimana terdapat 15 *columns* yaitu *gid*, *actual_time_minutes*, *assignee_name*, *assignee_gid*, *assignee_resource_type*, *name*, *resource_type*, *assignee_status*, *completed*, *completed_at*, *created_at*, *custom_field_gid*, *enum_option_gid*, *enum_option_color*, *enum_option_enabled*.

Fungsi *info()* juga dapat menampilkan data yang tidak kosong (*non-null*) pada masing-masing kolom, tipe data yang digunakan, dan memori keseluruhan. Tahap ini cukup penting apakah terdapat nilai yang kosong atau memiliki *values*, ketika kosong maka akan perlu ditangani.

Gambar 3.12 Memasukan data ke *SQL database*.

Setelah data tersebut dilakukan pengecekan dengan *Python*, berikutnya data (pada gambar 3.12) tersebut akan dimasukan ke *database* departemen *CIT* dan nama *tabel* yaitu *PPMCDTSample*. Setelah proses penyimpanan selesai, selanjutnya pada *tabel* tersebut dilakukan pembuatan visualisasi, serta *dashboard* yang interaktif dengan *Metabase* agar mempermudah tim dalam mengecek aktivitas, tugas, hingga status dan performa dari *operational* berdasarkan data yang sudah diolah.

3.2.2.2 *Dashboard Project Management (PPM CDT)*

ReportingHelpdesk

```
1  select *,
2      datediff(day, [created_at], [completed_at]) as Completed_Duration,
3      datediff(day, [created_at], getdate()) as Running_Duration,
4      CASE WHEN assignee_status = 'inbox' and completed = 'true' THEN 'Completed'
5          WHEN assignee_status = 'upcoming' and completed != 'false' and completed != 'true' THEN 'Upcoming'
6          ELSE 'In Progress' END AS CalcStatus
7  from PPMCDTSample
8
```

Gambar 3.13 Melakukan Read data dan Query SQL pada metabase.

Gambar 3.13 pada minggu ke 4-5 , Ketika data tersebut sudah masuk ke *database SQL*, maka melakukan *read data* pada *Metabase* lalu terdapat beberapa *query* seperti melakukan penghitungan waktu penyelesaian *project*, *project* yang masih berjalan, dan melakukan penggunaan *case* terkait kolom baru di *CalcStatus*. Berikut merupakan penjelasan *code* tersebut. Pertama melakukan *select* pada semua kolom di *tabel PPMCDTSample*, berikutnya penggunaan *DATEDIFF* untuk melakukan perhitungan selisih hari dari tanggal pembuatan *proyek* hingga selesai

proyek dan *proyek* yang masih berjalan/belum selesai. Berikutnya membuat *case* untuk membuat kolom baru pada *CalcStatus* dengan kondisi seperti *assignee_status* adalah *inbox* dan *completed* adalah *true*, maka *project* tersebut menjadi *completed*, dll.

Gambar 3.14 Hasil dari Query SQL Metabase.

Hasil dari *query* tersebut menambahkan beberapa kolom baru pada *tabel* dari gambar 3.14, seperti *Completed_Duration*, *Running_Duration*, dan *CalcStatus*. Penambahan kolom-kolom ini bertujuan untuk memberikan informasi tambahan yang lebih bermakna, khususnya terkait dengan durasi penyelesaian tugas dan status terkini dari setiap *task*. Dengan adanya kolom-kolom tersebut, proses analisis data menjadi lebih mudah dan efisien. Selain itu, data yang telah diperkaya ini juga lebih siap untuk divisualisasikan dan digunakan dalam pembuatan *dashboard* pada platform seperti *Metabase*, sehingga dapat mendukung proses pengambilan keputusan yang lebih baik.

Gambar 3.15 ***Dashboard Project Monitoring Summary***

Dashboard pada gambar 3.15 merupakan hasil visualisasi pada data proyek *IT Service Management* pada *Kalbe Farma* yang telah diolah melalui pendekatan *Big Data Analytics* ditampilkan melalui *tools Metabase* yang dapat membantu dalam melakukan *monitoring* status proyek secara *real-time*. Tujuan utama pada *dashboard* ini untuk menyajikan ringkasan keseluruhan status proyek secara komprehensif seperti kinerja masing-masing *engineer*, durasi pengerjaan pada proyek, dan beban kerja yang ditangani setiap individu atau dari tim masing-masing proyek. *Top metrics* pada *dashboard* ini terdapat 6 yang utama dari pertama yaitu *Total Project* (*number visualization*) yang dimana visualisasi ini berisikan pada total keseluruhan *project* yang sudah dibuat, seperti proyek dari tim *developer* ingin melakukan perubahan dari sistem, dll. *Upcoming Project* (*number visualization*) visualisasi *metrik* ini berisikan terkait proyek yang akan datang yang belum dilakukan eksekusi. *In Progress Project* (*number visualization*) merupakan visualisasi *metrik* untuk proyek yang dikerjakan dan masih dalam *progress* pengerjaan. *Completed Project* (*number visualization*) merupakan visualisasi *metrik* yang berisikan terkait proyek yang sudah selesai dikerjakan dan dapat digunakan hasil dari pengerjaan proyek tersebut. *Perc. of Completed Project* (*number visualization*) visualisasi *metrik* ini merupakan hasil dari persentase total *project* yang sudah diselesaikan. Visualisasi *metrik* yang terakhir yaitu *Actual Effort per Unit* (*number visualization*) merupakan visualisasi *metrik* untuk rata-rata waktu kerja secara aktual yang dibutuhkan per proyek. Visualisasi ini merupakan *Pie Chart* yang dimana diagram pada *Total Effort by Calculation Status* menunjukan

distribusi pada status proyek berdasarkan hasil perhitungan status (*CalcStatus*), dan hasil dari diagram lingkaran yaitu terdapat 3 status *In Progress*, *Completed*, *Upcoming*. *In Progress* untuk proyek yang dikerjakan dan masih dalam *progress* pengerjaan. Visualisasi ini merupakan *Bar Chart* yang dimana diagram pada *Total Project per Engineer* menunjukan sebaran pada proyek dari beberapa nama *engineer* dan status pada proyek yang sedang ditangani (*In Progress*, *Completed*, *Upcoming*). Visualisasi ini merupakan *Bar Chart* yang dimana diagram pada *Project Completion by Engineer* menunjukan tingkatan penyelesaian proyek yang dikerjakan dari beberapa nama *engineer*. Visualisasi ini merupakan *Bar Chart* yang dimana diagram pada *Total Effort by Engineer* menunjukan tingkatan dari proyek yang sudah selesai dikerjakan oleh para *Engineer*. Pada visualisasi paling terakhir merupakan berbentuk *tabel* dengan nama *Project Running Table* yang menampilkan secara detail pada masing-masing proyek secara keseluruhan, yang memiliki beberapa *column* untuk dimunculkan pada *tabel* tersebut seperti *Project Name*, *Total Engineer*, *Engineer*, *Created Date*, *Total Running*, *Upcoming*, *In Progress*, *Completed*, *Completed Date*, *Total Effort*, *CalcStatus*.

3.2.2.3 *Dashboard* Service *Request* B7

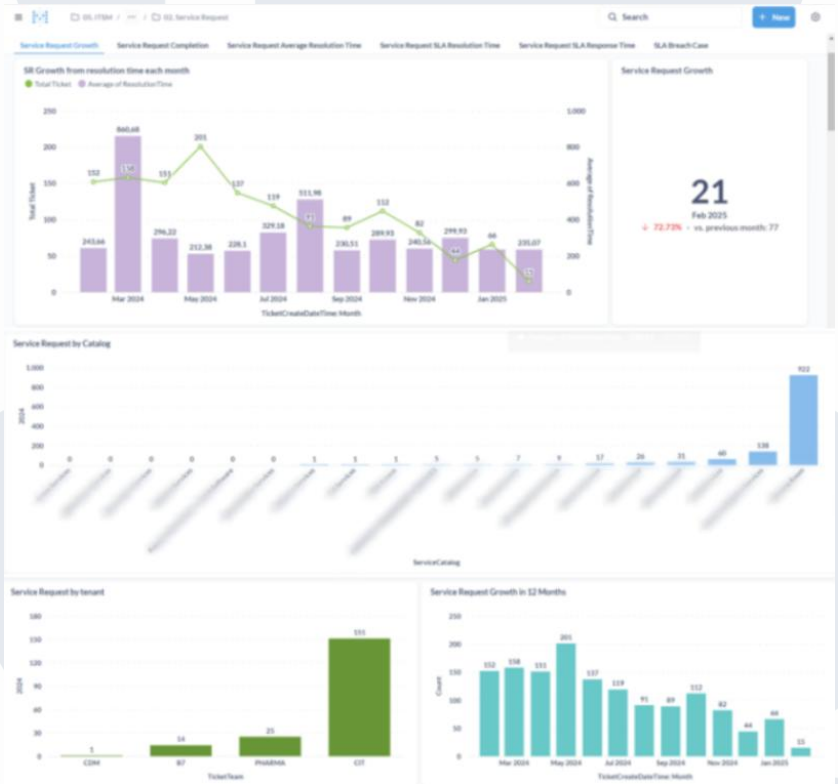
Gambar 3.16 *Query of Service Request B7*

Pada minggu ke-5 hingga ke-6, pembuatan dashboard ini bertujuan untuk mengubah data mentah dari business unit B7 menjadi informasi yang bermakna dan mudah dipahami oleh tim IT B7. Melalui dashboard ini, pengguna—baik dari tim IT B7 maupun pihak terkait lainnya—dapat memperoleh gambaran mengenai aktivitas *service request* yang terjadi, seperti jenis layanan yang paling sering diminta, waktu *response time* dalam penanganan setiap tiket, serta tingkat penyelesaian permintaan berdasarkan *Service Level Agreement* (SLA) yang telah ditentukan.

Hal pertama yang dilakukan ketika ingin membuat *dashboard* B7 yaitu melakukan pemrosesan data melalui *query* yang sudah disusun secara khusus pada gambar 3.16. *Query* ini dapat melakukan pengelompokan status *tiket*, menghitung sisa waktu sebelum melewati *SLA (Service Level Agreement)*, mengukur durasi *approval* pada berbagai *level*, dan dapat melakukan *monitor progress tiket* berdasarkan waktu. *Query* ini diawali dengan mengambil seluruh data dari suatu *tabel* yang berisi informasi *tiket layanan*. Setelah itu, dilakukan pengelompokan ulang terhadap status *tiket* agar lebih mudah dibaca dan dipahami. Setiap status asli diubah menjadi format baru yang diberi kode urut agar proses analisis menjadi lebih terstruktur, dimulai dari status paling awal hingga status akhir seperti *tiket* yang dibatalkan.

Selanjutnya, *query* ini juga menghitung sisa waktu sebelum *tiket* tersebut melampaui batas waktu yang ditentukan dalam *SLA*. Jika *tiket* tersebut belum memiliki batas waktu tertentu, maka akan dianggap belum memiliki tenggat, dan diberikan nilai *default* yang sangat tinggi untuk menunjukkan bahwa tidak ada batas waktu yang aktif. Namun jika batas waktu tersedia, maka sistem akan menghitung selisih waktu antara waktu saat ini dengan batas waktu tersebut. Hasil perhitungan ini akan ditampilkan dalam format hari, jam, dan menit. Jika waktunya sudah lewat, maka *tiket* tersebut dianggap telah melanggar *SLA*. Jika masih ada waktu tersisa, sistem akan menunjukkan dengan jelas berapa lama waktu yang tersisa sebelum batas waktu terlewati.

Query ini juga menghitung lama waktu *tiket* aktif sejak dibuat hingga saat ini, dengan menampilkan hasil dalam format durasi yang mudah dibaca. Untuk *tiket* yang masih dalam proses *persetujuan* dari atasan langsung, sistem menghitung berapa hari telah berlalu sejak pengajuan dilakukan. Selain itu, untuk menghitung durasi secara lebih realistis, sistem juga menyediakan perhitungan berdasarkan hari kerja saja dengan mengabaikan hari Sabtu dan Minggu. Dalam logikanya, setiap akhir pekan akan dikeluarkan dari perhitungan, dan sistem juga mempertimbangkan jika tanggal awal jatuh pada hari Sabtu atau tanggal akhir berada di hari Minggu. Terakhir, salah satu *kolom* yang sebelumnya berfungsi sebagai penanda waktu pengajuan, diubah penamaannya agar lebih merepresentasikan waktu sebenarnya ketika *tiket* dibuat.



Gambar 3.17 *Dashboard Service Request Growth* B7

Dashboard pada gambar 3.17 merupakan *summary visual* yang menyajikan informasi dari *Service Request (SR)* pada *business unit* B7. Data yang ditampilkan berasal dari hasil *query* yang dijalankan dengan *Metabase*, kemudian diolah dan dilakukan *visualisasi* dalam bentuk grafik serta *tabel interaktif* agar mudah dipahami dan dianalisis oleh tim IT terutama B7. Dengan *dashboard* tersebut, *pengguna* (tim IT B7 atau *users*) dapat melihat dan mendapatkan informasi perkembangan *tiket Service Request* dari waktu ke waktu, termasuk jumlah *tiket* yang masuk dan status penyelesaiannya. Selain itu, *dashboard* juga menyajikan informasi mengenai rata-rata waktu penyelesaian *tiket*, yang memberikan gambaran seberapa cepat tim IT dalam merespons dan menyelesaikan permintaan layanan. *Visualisasi* lainnya menampilkan sebaran *Service Request* berdasarkan jenis *katalog layanan* yang digunakan, sehingga dapat diketahui layanan mana yang paling sering diminta oleh *pengguna*. Selain itu, informasi juga disajikan berdasarkan *tenant* atau unit kerja *pengguna*, sehingga tim IT dapat memahami distribusi permintaan layanan dari masing-masing bagian organisasi.

Dashboard ini juga menampilkan aktivitas *Service Request* berdasarkan hari dalam seminggu dan jam dalam sehari, lengkap dengan informasi waktu *respons* yang dilakukan oleh tim IT. Dengan begitu, pola waktu sibuk atau waktu-waktu dengan beban kerja tinggi dapat diidentifikasi dan dikelola dengan lebih baik. Terakhir, *dashboard* juga mencakup pengembangan persyaratan kategori layanan yang paling sering dibuat, memungkinkan tim untuk fokus pada peningkatan layanan yang dibutuhkan *pengguna*. Keseluruhan *visualisasi* dalam *dashboard* ini

membantu tim IT *business unit* B7 dalam memahami kondisi operasional layanan, memantau kinerja, serta mendukung pengambilan keputusan berbasis data untuk meningkatkan efisiensi dan kualitas layanan IT.

Gambar 3.18 *Dashboard Service Request Completion* B7

Dashboard pada gambar 3.18 merupakan ringkasan visual dari capaian penyelesaian *tiket layanan* atau *Service Request (SR) Completion* pada *Business Unit* B7. Tujuan utama dari *dashboard* ini adalah untuk memberikan gambaran

menyeluruh mengenai bagaimana progres penyelesaian *tiket* berlangsung dari waktu ke waktu, serta sejauh mana *tiket-tiket* tersebut dapat diselesaikan sesuai dengan target waktu yang telah ditetapkan dalam *SLA* (*Service Level Agreement*). *Visualisasi* pertama di bagian atas menunjukkan tingkat penyelesaian *tiket* secara keseluruhan dalam satu bulan tertentu. Grafik ini berfungsi untuk menilai apakah proses penanganan *tiket* telah berjalan optimal atau masih terdapat kendala seperti keterlambatan atau pelanggaran terhadap batas waktu *SLA*. Selanjutnya, grafik batang berjudul *"Service Completions in Months"* menggambarkan tren penyelesaian *tiket* dari bulan ke bulan selama dua tahun terakhir. *Visualisasi* ini berguna untuk memantau kestabilan kinerja tim dari waktu ke waktu dan mengidentifikasi periode-periode tertentu di mana penyelesaian *tiket* menurun.

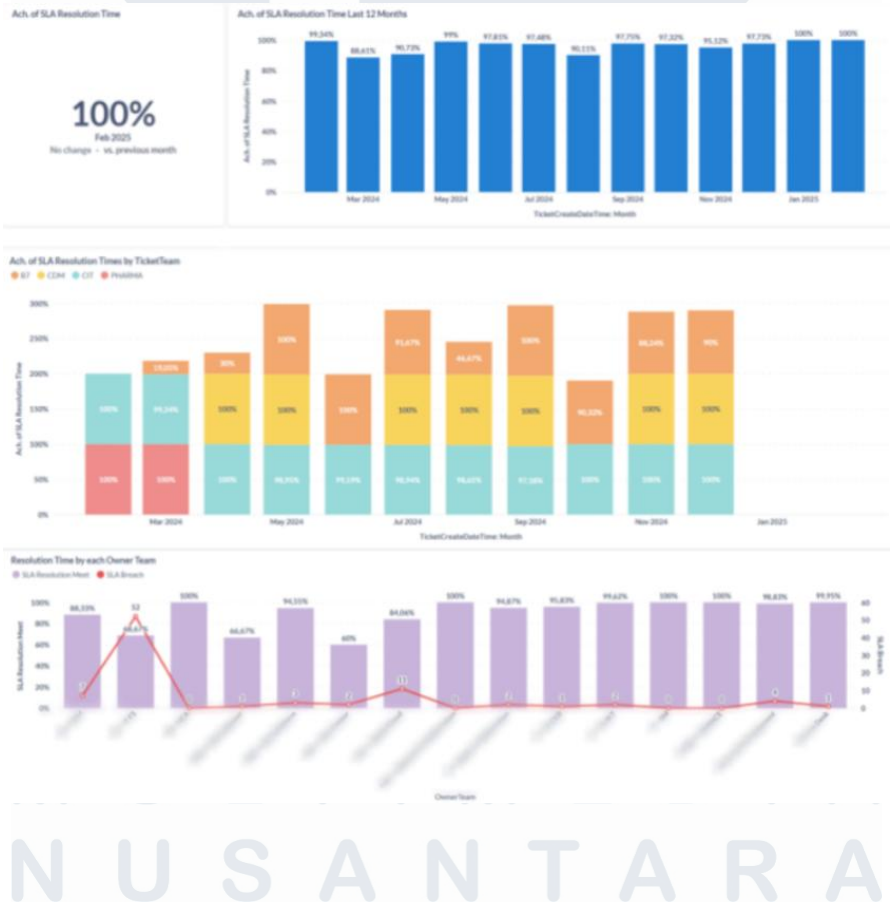
Visualisasi "Service Request Completion by Tenant Each Month" memberikan informasi mengenai kontribusi masing-masing *tenant* atau divisi dalam menyelesaikan *tiket* setiap bulannya. Grafik ini membantu mengetahui *tenant* mana yang memiliki kinerja konsisten serta yang perlu mendapat perhatian lebih terkait penyelesaian *tiket*. Berikutnya, grafik *"Achievement SR Completion by Owner Team"* memperlihatkan performa tim pemilik *tiket* dalam menyelesaikan tugasnya. *Visualisasi* ini bermanfaat untuk menilai tanggung jawab dan efektivitas tiap tim dalam menangani *tiket* yang menjadi bagian dari lingkup kerjanya. Diagram setelahnya menampilkan capaian penyelesaian berdasarkan tim pelaksana *tiket*. Tujuan dari grafik ini adalah untuk mengevaluasi kinerja masing-masing tim dalam memenuhi target *SLA* setiap bulan, sekaligus melihat konsistensi antar tim dalam menjaga standar penyelesaian. Paling bawah pada *dashboard* dilengkapi dengan *tabel* yang menyajikan informasi rinci dari tiap *tiket layanan*. *Tabel* ini mencakup elemen-elemen seperti identitas *tiket*, waktu pembuatan, waktu penyelesaian, status persetujuan, dan waktu tertunda. Informasi ini sangat berguna untuk melakukan penelusuran lebih lanjut terhadap proses dari setiap *tiket* secara individu dan membantu dalam evaluasi detail.

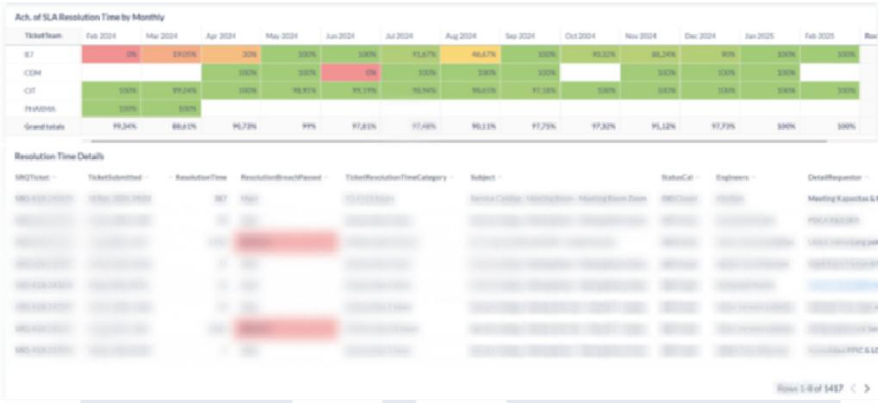
Gambar 3.19 ***Dashboard Service Request Average Resolution Time B7***

Dashboard pada gambar 3.19 merupakan rangkuman dari *Service Request Average Resolution Time* di perusahaan B7. Tujuan utama dari *dashboard* ini adalah untuk menunjukkan berapa lama rata-rata waktu yang dibutuhkan untuk menyelesaikan *tiket layanan*. Pada *visualisasi* pertama, yaitu *Average of Resolution Time (Trend Metrics)*, ditampilkan tren waktu penyelesaian secara keseluruhan. *Visualisasi* selanjutnya, *Average of Resolution Time Each Monthly*, menampilkan

kombinasi antara rata-rata waktu *resolusi* dan jumlah *tiket* yang ditangani setiap bulan. Lalu, *visualisasi Average Resolution Time Monthly by Service Catalog* menyajikan perbandingan rata-rata waktu penyelesaian berdasarkan jenis layanan.

Selanjutnya, pada *visualisasi Average of Resolution Time by Each Owner Team*, diperlihatkan rata-rata waktu penyelesaian berdasarkan tim yang menangani *tiket*. *Tabel Average of Resolution Time Monthly by Ticket Team* memberikan gambaran lebih rinci mengenai rata-rata waktu penyelesaian berdasarkan masing-masing tim penangan setiap bulan. Warna-warna dalam *tabel* tersebut membantu menyoroti performa; misalnya, *sel* berwarna merah menunjukkan bahwa waktu penyelesaian lebih tinggi dari rata-rata. Sebagai penutup, *visualisasi* terakhir menyajikan detail *tiket* yang memiliki waktu penyelesaian paling lama. Informasi ini penting untuk evaluasi lebih lanjut, guna mengetahui penyebab keterlambatan serta mengidentifikasi bagian mana yang perlu diperbaiki ke depannya.





Gambar 3.20 ***Dashboard Service Request SLA Resolution Time***

Dashboard pada gambar 3.20 merupakan *visualisasi* dari data *Service Request SLA Resolution Time* yang dirancang untuk menampilkan persentase penyelesaian *tiket layanan* sesuai dengan *SLA (Service Level Agreement)*. Tujuan dari *dashboard* ini adalah untuk memberikan gambaran apakah dalam periode tertentu terdapat *tiket* yang penyelesaiannya melebihi waktu yang ditentukan dalam *SLA*, yang berarti terjadi *breach*. *Visualisasi* pertama berupa *Trend Metrics* menunjukkan capaian *SLA* secara keseluruhan. *Visualisasi* berikutnya berbentuk *bar chart* yang memperlihatkan persentase pencapaian *SLA* dari bulan ke bulan selama dua belas bulan terakhir.

Selanjutnya, *visualisasi Ach. of SLA Resolution Time by Ticket Team* juga disajikan dalam bentuk *bar chart*. Grafik ini memperlihatkan performa setiap tim dalam mencapai *SLA*. *Visualisasi* keempat adalah *Resolution Time by Each Owner Team*, yang memberikan informasi detail mengenai jumlah *tiket* yang mengalami *breach* dan persentase capaian *SLA* pada setiap tim pemilik *tiket*. Berikutnya, *tabel Ach. of SLA Resolution Time by Monthly* merinci pencapaian *SLA* setiap tim per bulan. *Tabel* ini berguna untuk melihat konsistensi performa masing-masing tim dan mengidentifikasi jika ada pola atau penurunan capaian dalam bulan-bulan tertentu. Dan yang terakhir merupakan detail terkait *Resolution Time Detail* yang dimana informasi secara mendetail pada *tabel* dan masing-masing *tiket* tersebut.

Gambar 3.21 ***Dashboard Service Request SLA Response Time***

Dashboard pada gambar 3.21 merupakan rangkuman *visual* dari *metrik Service Request SLA Response Time* yang bertujuan untuk menampilkan informasi terkait waktu tanggapan atas *tiket layanan*, apakah sudah sesuai dengan target *SLA* atau

tidak. Seluruh *visualisasi* dalam *dashboard* ini memberikan gambaran mengenai performa *response time* dalam berbagai sudut pandang. *Visualisasi* pertama yang ditampilkan adalah *Ach. of SLA Response Time* dalam bentuk angka dan persentase. Kemudian, *visualisasi bar chart Ach. of SLA Response Time Last 12 Months* memperlihatkan tren pencapaian *response time* selama satu tahun terakhir. *Visualisasi* berikutnya yaitu *Ach. of SLA Response Time by Ticket Team*, masih dalam bentuk *bar chart*, memperlihatkan bagaimana masing-masing tim menangani *tiket* mereka.

Selanjutnya, *visualisasi Average Response Time by Each Owner Team* disajikan dalam bentuk *line chart* yang menampilkan rata-rata waktu tanggapan serta jumlah *tiket* per tim pemilik. *Line chart* selanjutnya yaitu *Average Response Time by Each Service Catalog*, yang memperlihatkan rata-rata waktu *respon* per jenis layanan. Terakhir, terdapat *tabel Ach. of SLA Response Time by Monthly* yang memuat detail capaian *response time* berdasarkan tim dan bulan. *Tabel* ini sangat berguna untuk menelusuri lebih jauh performa masing-masing tim dari waktu ke waktu dan dapat membantu dalam analisis lebih lanjut untuk peningkatan layanan ke depan.

Gambar 3.22 ***Dashboard Service Request SLA Breach Case***

Dashboard pada gambar 3.22 dirancang untuk memberikan gambaran menyeluruh mengenai *tiket-tiket layanan* yang mengalami keterlambatan penyelesaian, atau dikenal dengan *SLA Breach*. Fokus utamanya adalah mengidentifikasi bagian mana saja yang belum memenuhi standar waktu tanggapan atau penyelesaian berdasarkan ketentuan *SLA*, serta mengetahui pola keterlambatan yang terjadi baik dari sisi jenis layanan maupun tim penanggung jawabnya. *Visualisasi* pertama menampilkan *metrik* tren terkait waktu tanggapan *tiket* (*SLA Breach Response Time*). *Metrik* ini memberikan informasi apakah tim mampu memberikan *respon awal* terhadap *tiket* tepat waktu atau masih terdapat keterlambatan. Melalui *visualisasi* ini, pengguna bisa melihat pergerakan performa dari waktu ke waktu, apakah mengalami perbaikan atau justru penurunan.

Berikutnya, *metrik* yang menampilkan *SLA Breach Resolution Time* fokus pada proses penyelesaian *tiket*. *Visualisasi* ini menunjukkan sejauh mana tim berhasil menyelesaikan *tiket* sesuai batas waktu yang telah ditentukan dalam *SLA*. Ini penting untuk mengukur konsistensi kinerja dalam menuntaskan tugas sampai akhir. Selanjutnya, *bar chart* dengan judul *Breach Case in the Service Catalog* menyajikan informasi mengenai jumlah kasus keterlambatan berdasarkan kategori layanan. Dari sini, pengguna bisa mengetahui layanan mana yang paling sering mengalami *breach*, sehingga menjadi titik awal untuk perbaikan atau evaluasi lebih lanjut pada kategori layanan tertentu.

Visualisasi lainnya yaitu *Total Task of Ticket Breach by Each Owner Team* menunjukkan distribusi jumlah *tiket* bermasalah yang ditangani oleh masing-masing tim pemilik. Ini berguna untuk melihat beban dan tanggung jawab tiap tim dalam menangani *tiket* yang tidak selesai tepat waktu, serta menyoroti tim-tim yang mungkin perlu dukungan tambahan. Grafik berikutnya berjudul *Growth of Ticket Breach Each Month by Resolution Time* menggambarkan perkembangan kasus *breach* dari bulan ke bulan. *Visualisasi* ini membantu melihat tren dan fluktuasi jumlah *tiket* yang melewati batas waktu penyelesaian, serta mengidentifikasi periode-periode tertentu yang mengalami lonjakan kasus. *Dashboard* juga dilengkapi dengan *pivot table* berjudul *SLA Breach Each Service Catalog*, yang menyajikan data secara lebih terperinci berdasarkan jenis layanan dan waktu kejadian. *Tabel* ini sangat berguna untuk menganalisis frekuensi *breach* berdasarkan layanan tertentu dalam rentang waktu bulanan. Bagian paling bawah, *visualisasi* dalam bentuk *tabel SLA Breach Details* menampilkan rincian paling lengkap dari setiap *tiket* yang mengalami *breach*. Informasi yang ditampilkan meliputi waktu pembuatan *tiket*, siapa yang menangani, status penyelesaian, hingga tanggal penyelesaian akhir. *Tabel* ini menjadi sumber penting untuk penelusuran kasus per kasus dan mendukung proses evaluasi serta peningkatan sistem layanan secara menyeluruh.

3.2.2.4 *Data Scraping* dengan *API* pada *Bitlocker 365*

Pada gambar 3.23. Dalam proses pengambilan data *Bitlocker 365* menggunakan *Microsoft Graph API*, ada beberapa langkah yang perlu dilakukan. Pertama, harus memastikan bahwa akses sudah diberikan melalui *Microsoft Admin Center*. Hal ini dilakukan bersama tim yang memiliki otoritas agar bisa mendapatkan izin untuk menarik data yang dibutuhkan. Setelah akses diberikan, tahap berikutnya adalah mencari tahu bagaimana cara mengambil data melalui *API*. Untuk itu, digunakan website *Microsoft Graph Explorer* sebagai tempat uji coba dan eksplorasi *endpoint* yang sesuai. Dari sana, dapat menemukan *API* yang tepat untuk digunakan dalam pengambilan data *Bitlocker*.

Setelah *endpoint* yang sesuai ditemukan, proses dilanjutkan dengan membuat kode di *Python*. Kode ini dimulai dengan pembuatan fungsi untuk menangani *pagination*, agar data bisa diambil secara menyeluruh tanpa terpotong. Kemudian, data perangkat yang berkaitan dengan *Bitlocker* ditarik, dan hanya kolom-kolom yang relevan saja yang dipilih agar lebih efisien. Langkah selanjutnya adalah menambahkan informasi pengguna dari masing-masing perangkat. Data perangkat dan data pengguna ini kemudian digabungkan agar dapat diketahui siapa pemilik atau pengguna dari setiap perangkat yang terdaftar. Setelah semua proses selesai, data yang telah terkumpul dan digabungkan dikonversi ke dalam bentuk *dataframe*.

01. CORP

```
app = msal.ConfidentialClientApplication(
    client_id,
    authority=authority,
    client_credential=client_secret
)

result = app.acquire_token_for_client(scopes=scopes)

if "access_token" in result:
    print("Token diperoleh!")
else:
    print("Gagal mendapatkan token:", result.get("error"))
    print(result.get("error_description"))

import requests
import pandas as pd

def call_api_with_pagination(url, headers):
    results = []
    page = 1
    while url:
        response = requests.get(url, headers=headers)
        if response.status_code == 200:
            data = response.json()
```

Gambar 3.23 Penarikan data *Bitlocker 365* dengan *API Microsoft Graph* melalui *Python*

Pada minggu 6 hingga minggu 7, setelah data *Bitlocker 365* berhasil ditarik melalui *Microsoft Graph API* dan sudah disusun dalam bentuk *dataframe*, pada gambar 3.24 langkah selanjutnya adalah memasukkan data tersebut ke dalam *database* perusahaan. Tujuan dari penyimpanan ini adalah agar data bisa diakses dengan lebih mudah dan digunakan sebagai sumber utama dalam pembuatan visualisasi di *Metabase*. Dengan data yang sudah tersimpan di *database*, tim dapat memantau dan menganalisis informasi *Bitlocker* secara lebih efisien melalui *dashboard* yang interaktif dan *real-time*.

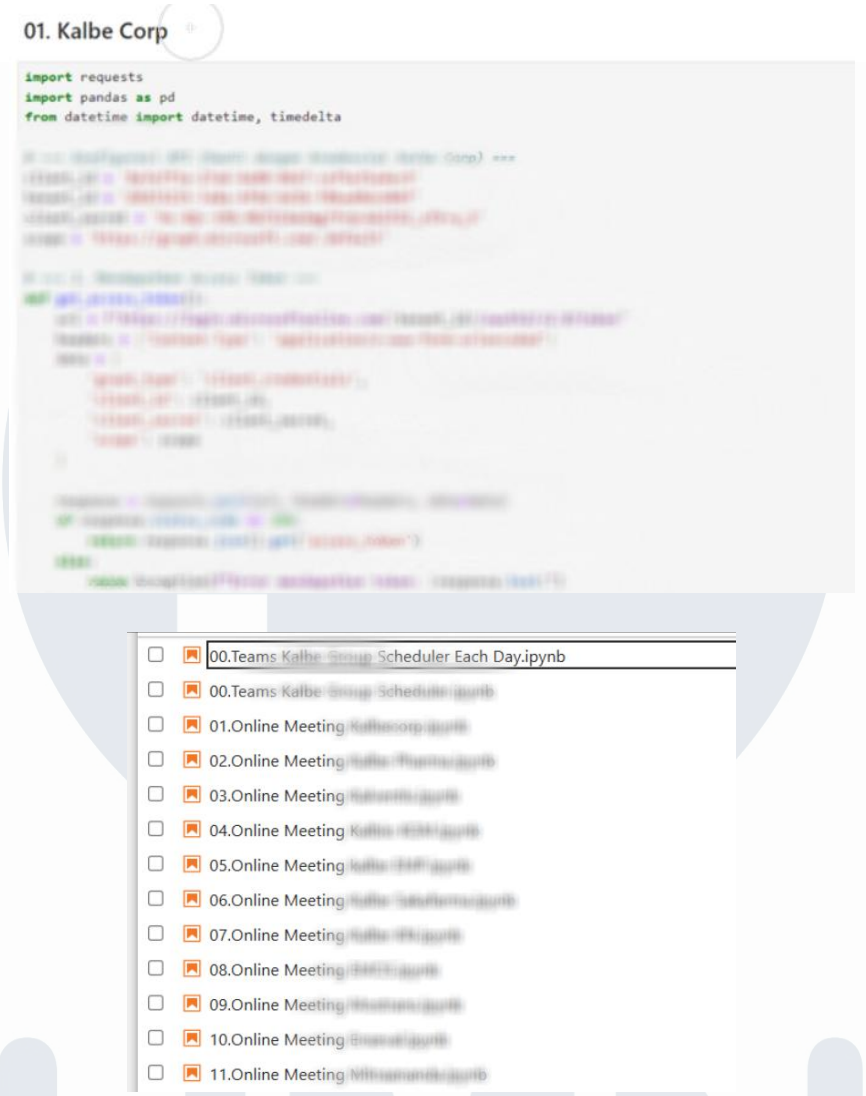
Gambar 3.24 *Output* Penarikan data lalu memasukan ke database

3.2.2.5 *Data Scraping Microsoft Teams Usage*

Langkah awal dalam pembuatan visualisasi penggunaan *Microsoft Teams* dimulai dengan proses pengambilan data. Pada gambar 2.25, perlu dipahami

terlebih dahulu bagaimana cara menarik data dari *Microsoft Teams*. Untuk itu, digunakan *API* sebagai jembatan dalam pengambilan data tersebut. Sebelum menulis kode, terlebih dahulu berkoordinasi dengan tim yang memiliki akses ke *Microsoft Admin* untuk membantu dalam proses pemberian izin akses pada *Microsoft Azure permission* yang perlu pada penarikan data *Teams* adalah *Calendars.Read*, akses ini bertujuan untuk membaca semua *event* pada *kalender* pengguna, namun akses ini hanya sekedar membaca *event* tersebut *Calendars.ReadWrite*, akses ini berguna sama seperti sebelumnya namun ini bisa melakukan menulis *kalender* pengguna *OnlineMeetings.Read*, akses ini bertujuan untuk membaca data pada *online meetings* pengguna dan *OnlineMeetings.ReadWrite*, akses ini juga mirip dengan *OnlineMeetings.Read* namun kita membuat *Teams online meetings*. Setelah akses berhasil diberikan, langkah selanjutnya adalah mencari referensi kode yang dapat digunakan untuk menarik data melalui *API*.

Kemudian menyusun kode dengan tahapan awal membuat skrip untuk memperoleh *access token*, yang diperlukan agar sistem dapat terhubung dan mengakses data dari *Microsoft Teams*. Setelah berhasil mendapatkan akses, dilakukan pemanggilan data berupa daftar *email* pengguna melalui *endpoint Users API*. Selanjutnya, dibuat fungsi untuk menarik data aktivitas berupa *event* dari *kalender Microsoft Teams* masing-masing pengguna. Data yang diambil juga difilter berdasarkan rentang tanggal tertentu agar hanya *event* yang relevan yang ditampilkan. Setelah data berhasil ditarik, seluruh informasi tersebut kemudian dikonversi ke dalam bentuk *dataframe* agar lebih mudah dibaca dan diolah menggunakan *Python*. Data ini nantinya menjadi dasar dalam pembuatan visualisasi yang menggambarkan penggunaan *Microsoft Teams* secara lebih informatif dan terstruktur.



Gambar 3.25 Penarikan data *Teams* dengan *API Microsoft Graph* melalui *Python.*

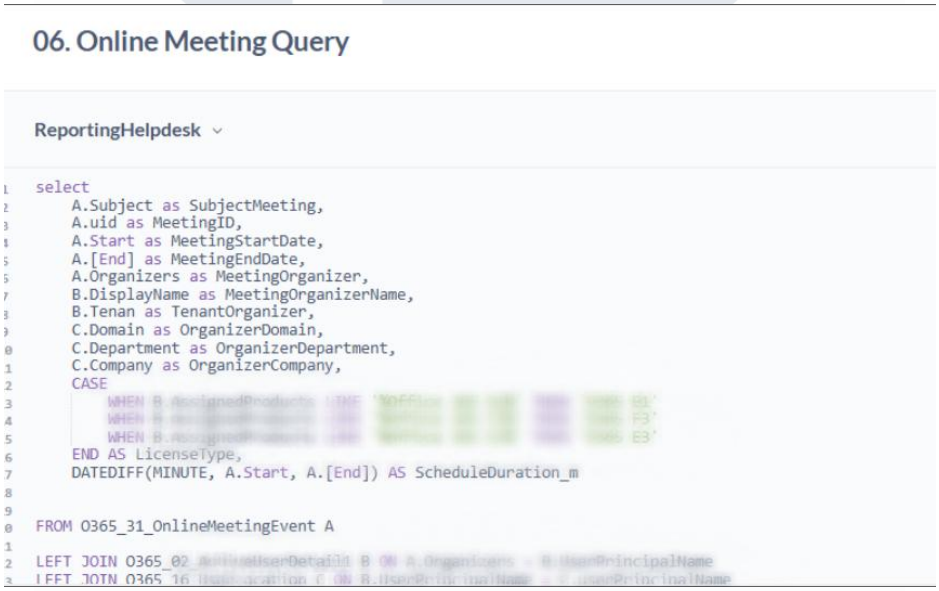
Setelah data berhasil diambil menggunakan kode yang telah dibuat sebelumnya, langkah selanjutnya pada gambar 2.26 yaitu melakukan proses *data preprocessing*. Tahapan ini penting agar data memiliki struktur yang lebih rapi dan mudah untuk dianalisis. Hal ini dilakukan karena terdapat beberapa kolom dalam data yang berisi lebih dari satu nilai dalam satu sel, sehingga perlu dipisahkan dan disesuaikan formatnya.

Selain itu, dilakukan juga proses *data cleansing* atau pembersihan data, seperti menghilangkan karakter yang tidak diperlukan—misalnya tanda seru atau simbol lain yang bisa mengganggu proses analisis. Setelah seluruh data dibersihkan

dan sudah tersusun dengan baik, data yang telah siap digunakan kemudian dimasukkan ke dalam *database* sebagai tahap akhir dari proses pengolahan data setelah *cleaning* data tersebut menjadi lebih *300000 rows* dari tahun 2023 hingga saat ini.

Gambar 3.26 *Output* Penarikan melalui *API* data memasukan ke *database*

3.2.2.6 *Data Preprocessing & Query Microsoft Teams Usage*

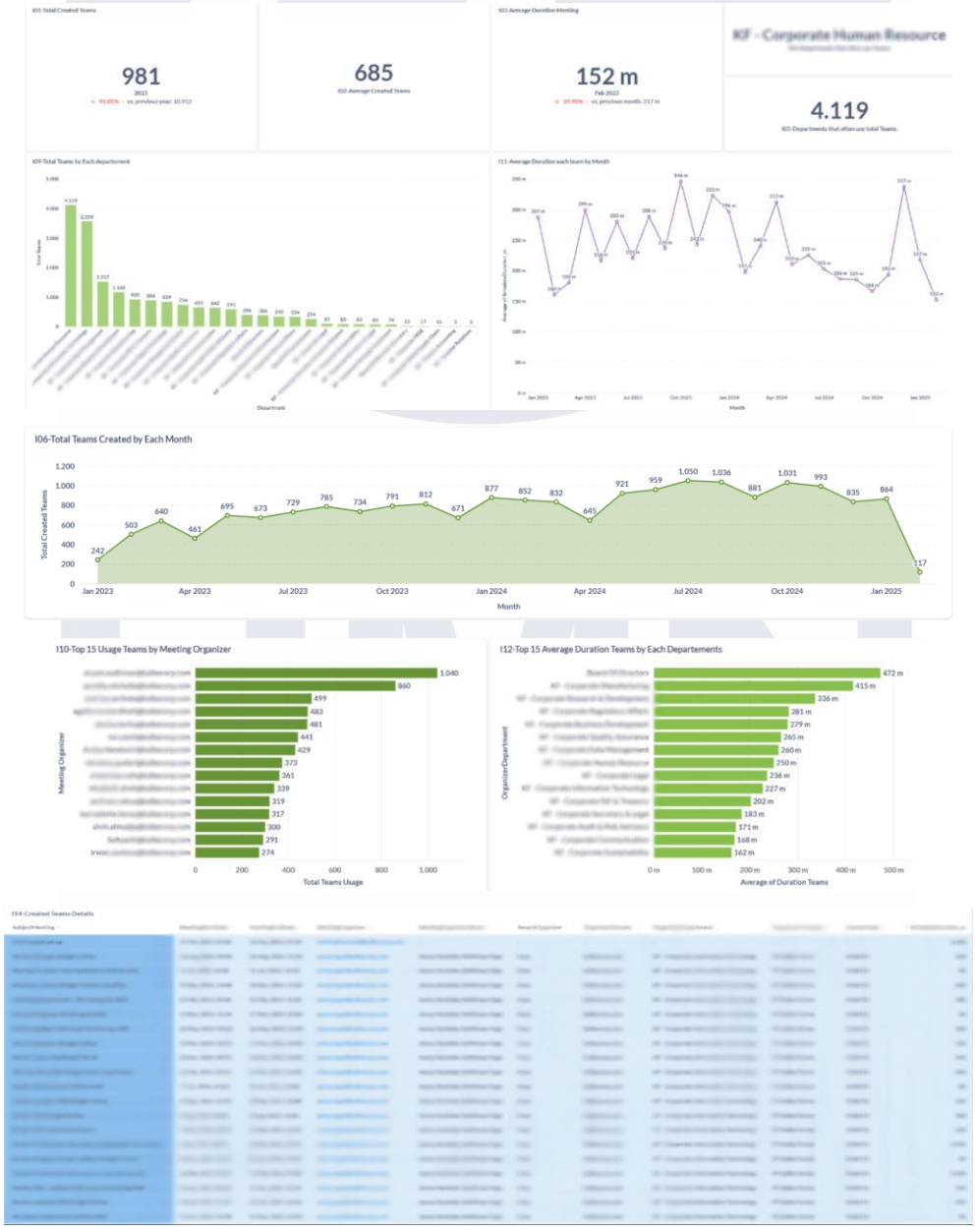


Gambar 3.27 Melakukan *Query* pada *metabase* untuk memanggil Data pada *DB*.

Setelah data dimasukkan ke dalam *database*, langkah berikutnya adalah menyusun *query* untuk keperluan visualisasi pada gambar 3.27. Namun, sebelum data dapat divisualisasikan, perlu dilakukan beberapa tahapan *query* tambahan untuk mempersiapkan datanya. Salah satu langkah yang dilakukan adalah

menambahkan logika *case* untuk memudahkan proses *filter* berdasarkan jenis lisensi *Microsoft* yang digunakan oleh pengguna. Selain itu, data juga digabungkan (*join*) dengan data pengguna dari *Office 365*, agar bisa diketahui informasi lebih lengkap, seperti siapa yang membuat tim di *Microsoft Teams*. Setelah semua tahapan *query* ini selesai dan data sudah dalam bentuk yang sesuai, barulah proses pembuatan *dashboard* dapat dilakukan untuk menghasilkan *insight* yang bermanfaat dari data tersebut.

3.2.2.7 *Dashboard Microsoft Teams Usage*



Gambar 3.28 ***Dashboard Microsoft Teams Summary Usage***

Berikut adalah penjelasan terkait pembuatan *dashboard* penggunaan *Microsoft Teams* yang bertujuan untuk memantau keseluruhan penggunaan *Teams*, durasi penggunaan, serta departemen mana yang paling sering menggunakan *Teams* pada gambar 3.28. *Dashboard* ini dimulai dengan visualisasi metrik yang menunjukkan total pembuatan *meeting* di *Microsoft Teams* setiap tahunnya. Selanjutnya, terdapat visualisasi yang menggambarkan rata-rata pembuatan *meeting* di *Teams*, memberikan gambaran umum tentang frekuensi pembuatan *meeting* secara keseluruhan. Kemudian, terdapat visualisasi rata-rata durasi penggunaan *Teams*, yang memberikan informasi tentang durasi rata-rata setiap sesi *meeting*. *Dashboard* juga menampilkan informasi mengenai departemen-departemen yang paling sering menggunakan *Teams*, disertai dengan nama departemen dan total pembuatan *meeting* yang mereka lakukan. Selain itu, terdapat visualisasi *barplot* yang menggambarkan total pembuatan *meeting* oleh masing-masing departemen. Visualisasi berikutnya menunjukkan *top 15* departemen dengan rata-rata durasi penggunaan *Teams*, diikuti dengan visualisasi total pembuatan *meeting Microsoft Teams* berdasarkan bulan. Kemudian, terdapat visualisasi *barplot* yang menampilkan *top 15* penggunaan *Teams* berdasarkan masing-masing *organizer meeting*. Terakhir, *dashboard* ini menyediakan detail dari setiap pembuatan *meeting* di *Microsoft Teams*.

3.2.2.8 Data *Kaspersky* dan *Data IT Asset*

Gambar 3.29 Penarikan Data *Helpdesk* pada data aset dan *data kav*.

Pada minggu 12 hingga 13, penarikan data pada *Helpdesk* dilakukan untuk memeriksa apakah terdapat perbedaan antara data pada *IT Asset* dan data dari *Kaspersky (KAV)* pada gambar 3.29. Pemeriksaan ini bertujuan untuk melihat kesesuaian jumlah aset perangkat dengan jumlah aplikasi yang terdeteksi. Jika ditemukan perbedaan, misalnya jumlah perangkat di *IT Asset* lebih sedikit dibandingkan jumlah perangkat yang terdeteksi di *KAV*, hal ini menandakan adanya aplikasi yang tidak tercatat di inventaris. Dalam kasus tersebut, perlu dilakukan penelusuran lebih lanjut untuk memastikan penggunaan aplikasi tersebut, dan memastikan bahwa seluruh aplikasi yang digunakan telah sesuai dengan jumlah lisensi *software* yang dimiliki oleh perusahaan. Pada mulanya, data tersebut ditarik melalui *database* perusahaan untuk data *KAV* serta data *IT* dengan *Python* agar dapat dilakukan *komparasi* pada kedua data tersebut.

```python
import re

def Application_Name_Standart(name):
    patterns = [
        (r'[redacted]', '[redacted]'),
    ]

    for pattern, category in patterns:
        match = re.search(pattern, name, re.IGNORECASE)
        if match:
            return category.format(match.group(1) if match.groups() else '')
    return None

kav_application['ApplicationNameStandart'] = kav_application['wstrDisplayName'].apply(Application_Name_Standart)
```

Gambar 3.30 Melakukan Standarisasi Nama *Software*

Pada Gambar 3.30, selanjutnya membuat kode untuk melakukan *standarisasi* nama pada *software*, karena ketika pengguna melakukan *download* aplikasi tersebut, ketika masuk pada *database KAV* akan memunculkan nama aplikasi yang

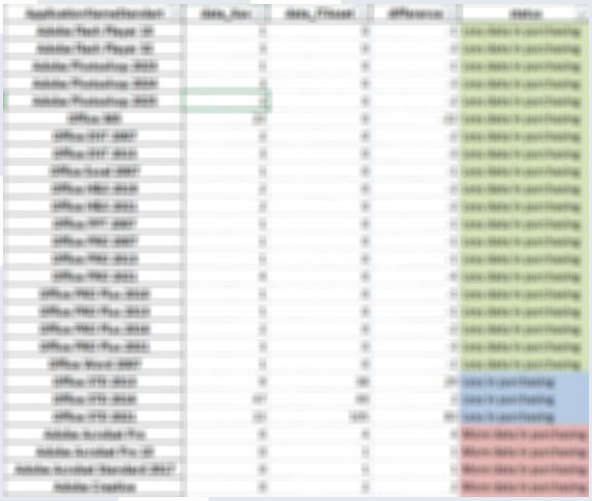
terlalu panjang dan lengkap hingga versinya. Hal tersebut membuat tidak efisien ketika melakukan *komparasi* pada data aset *IT*, jadinya memerlukan *standarisasi* nama aplikasi agar memudahkan untuk menyamakan dengan nama pada di *IT Asset*. *Code* ini berisikan pembersihan nama, diawali dengan pembuatan parameter untuk diubah dan *pattern* tersebut yang memiliki nilai yang mirip maka akan diubah dengan nama yang sudah di-*standarisasi*. Nama aplikasi tersebut lebih singkat dan memudahkan untuk melakukan *komparasi* dengan data *Asset*.

```
# Filter dan hitung jumlah kemunculan di masing-masing dataset
count_kav = kav_application[
    kav_application['wstrName'].str.contains(r'CORP', case=False, na=False)
]['ApplicationNameStandart'].value_counts().rename('data_Kav')

count_testing = testing_data[
    testing_data['LOBBusinessUnit'].str.contains(r'CORPORATE', case=False, na=False)
]['ApplicationNameStandart'].value_counts().rename('data_ITAsset')

# Gabungkan berdasarkan ApplicationNameStandart
comparison_df = pd.concat([count_kav, count_testing], axis=1).fillna(0)

# Hitung selisih
comparison_df['difference'] = comparison_df['data_ITAsset'] - comparison_df['data_Kav']

# Tambahkan status di mana data lebih banyak atau tidak ada
def check_status(row):
    if row['data_Kav'] == 0:
        return "More data in purchasing"
    elif row['data_ITAsset'] == 0:
        return "Less data in purchasing"
    elif row['difference'] > 0:
        return "Less in purchasing"
    elif row['difference'] < 0:
        return "More in puchasing"
    else:
        return "Same count"

comparison_df['status'] = comparison_df.apply(check_status, axis=1)

# Urutkan berdasarkan ApplicationNameStandart (index) secara ascending
comparison_df = comparison_df.sort_index()
```

Gambar 3.31 *Filter* data perusahaan, komparasi, dan konversi ke *Excel*.

Setelah melakukan *standarisasi* pada nama *software*, pada Gambar 3.31 melanjutkan proses dengan melakukan *filter data* berdasarkan perusahaan tertentu untuk membandingkan data *KAV* dengan data *IT Asset*. Setelah proses *filter* dilakukan, lalu kembali melakukan *standarisasi* nama aplikasi pada kedua data tersebut agar dapat disesuaikan dan dibandingkan dengan lebih akurat. Berikutnya, menambahkan kolom *status* untuk menentukan kondisi perbandingan data.

Misalnya, jika jumlah data pada *KAV* sama dengan 0, maka data tersebut dikategorikan sebagai "lebih dari *purchasing*", begitu pula sebaliknya. Sedangkan jika jumlah perbedaan pada kolom tertentu lebih besar dari 0, maka akan dikategorikan sebagai "lebih kurang dari *purchasing*", dan sebaliknya.



Gambar 3.32 *Ouput* pada format Excel.

Setelah proses penyesuaian dan pemberian *status* selesai (seperti ditunjukkan pada Gambar 3.38), data akan diurutkan berdasarkan nama aplikasi yang telah distandarisasi. Jika seluruh proses perbandingan telah dilakukan dan hasilnya telah sesuai, maka data akan diekspor ke dalam format *Excel*. File hasil *ekspor* ini kemudian akan diserahkan kepada tim terkait untuk keperluan evaluasi dan penanganan masalah *lisensi* serta penggunaan aplikasi. Dari hasil analisis ini, tim pengelola aplikasi dapat mengidentifikasi potensi efisiensi biaya, terutama jika ditemukan banyak aplikasi yang *ambigu* atau tercatat ganda dalam data pembelian *lisensi* aplikasi.

3.2.2.9 *Engineer Resolution Analysis* dan *Service Catalog SLA*

Gambar 3.33 *Filter* manual *task resolution detail* dari *raw data Excel.*

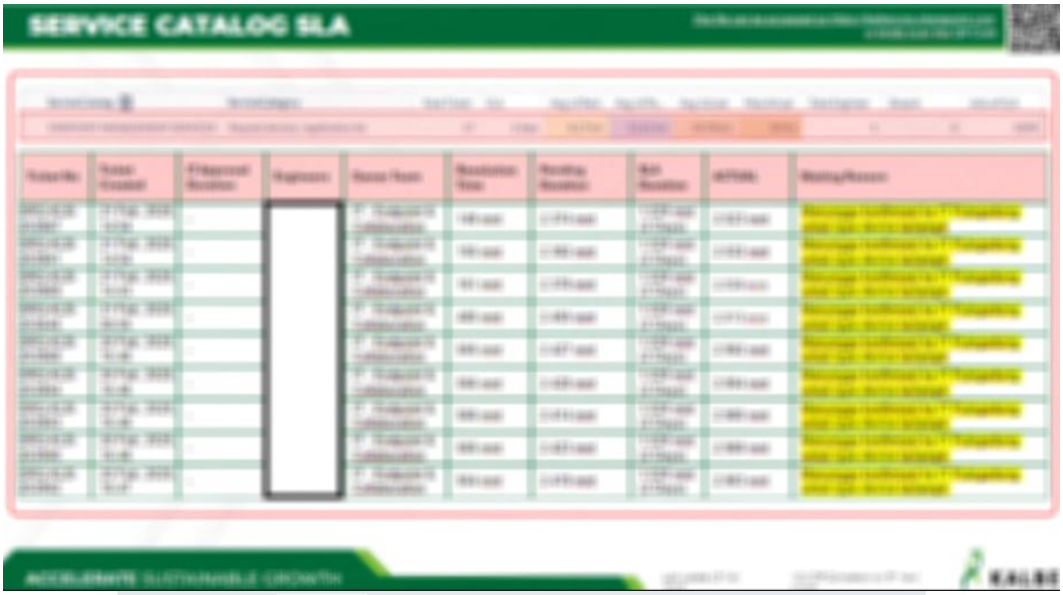
Pada minggu ke 13, *Engineer Resolution Analysis* dan *Service Catalog SLA* merupakan bagian dari laporan bulanan yang berfokus pada analisis terkait penyelesaian tiket. *Engineer Resolution Analysis* adalah analisis kinerja para *engineer* berdasarkan tiket yang mereka tangani, yang mengacu pada data dari *Task Resolution Analysis*. Tujuan dari analisis ini adalah untuk memastikan bahwa setiap *engineer* memberikan penjelasan yang jelas dan lengkap pada bagian *task resolution*, sehingga *customer* yang menunggu penyelesaian tiket dapat memahami *progres* dari tiket tersebut. Pada Gambar 3.33, melakukan proses pengumpulan data dimulai dengan melakukan *ekspor* data dari *Metabase* ke format *Excel*, kemudian dilakukan *filter* berdasarkan bulan yang sedang dianalisis. Selanjutnya, melakukan pencarian secara manual terhadap *engineer* yang memberikan alasan yang terlalu singkat atau tidak mengisi penjelasan sama sekali pada *task resolution*. Setelah itu, akan ditunjukkan tiket-tiket yang sebelumnya dipegang oleh *engineer* yang memiliki pengisian *task resolution* yang kurang lengkap atau kosong, untuk kemudian dilakukan evaluasi lebih lanjut.

Gambar 3.34 *Filter* tiket per *engineer* yang tidak memberikan *detail* penyelesaian.

Setelah *engineer* yang mengisi *task resolution* dengan penjelasan singkat atau kurang informatif diidentifikasi (seperti yang ditunjukkan pada Gambar 3.34), langkah berikutnya adalah melakukan *highlight* terhadap *engineer* tersebut. Tujuannya adalah untuk memberikan perhatian khusus dan memberikan masukan bahwa pengisian *task resolution* perlu dilakukan dengan lebih rinci dan berbobot. Dengan demikian, diharapkan ke depannya setiap *engineer* dapat memberikan penjelasan yang lebih jelas dan informatif, sehingga *user* yang menunggu penyelesaian tiket dapat memahami alasan serta *progress* tiket yang sedang ditangani secara lebih baik.

Gambar 3.35 *Service Catalog* keseluruhan

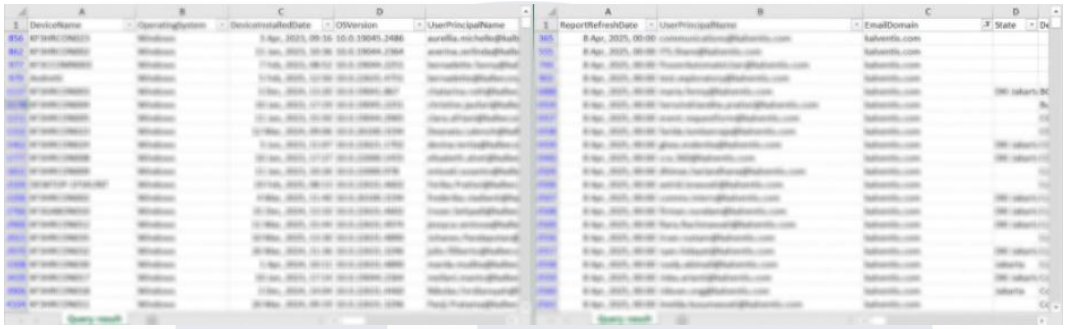
Pada Gambar 3.35 ditampilkan *Service Catalog SLA* secara keseluruhan, yang berdasarkan data dari tabel *Service Catalog* dan *Service Category*. Analisis ini bertujuan untuk mengidentifikasi *engineer* yang memiliki waktu *pending* terlama dan berpotensi mengalami *ticket breach* (pelanggaran *SLA*). Evaluasi ini penting dilakukan untuk memastikan setiap *engineer* memanfaatkan waktu *pending* sesuai dengan ketentuan yang berlaku untuk masing-masing tiket. Analisis dimulai dari *dashboard Service Catalog* yang menampilkan metrik-metrik penting seperti rata-rata waktu pengerjaan, waktu *pending*, jumlah tiket yang ditangani, serta jenis layanan (*service*) yang dikerjakan. Pada visualisasi tersebut, tiket yang ditandai dengan warna kuning menunjukkan tiket dengan waktu *pending* yang cukup lama, meskipun masih dalam batas *SLA*. Sedangkan tiket yang diberi tanda warna merah menunjukkan waktu *pending* yang lama dan hampir melewati batas *SLA*. Evaluasi ini bertujuan untuk meningkatkan kesadaran *engineer* dalam mengelola waktu *pending*, serta memastikan penyelesaian tiket tetap sesuai dengan standar layanan yang telah ditetapkan. Ketika sudah mendapatkan *service catalog* yang sudah dikuningkan dan dimerahkan akan dilakukan *highlight* untuk dimasukan dalam *report*.



Gambar 3.36 *Service Catalog* dengan pending lama dan risiko *breach*.

Gambar 3.36 menunjukkan contoh hasil dari satu *Service Catalog* yang ditandai dengan warna merah, yang mengindikasikan bahwa tiket tersebut hampir mengalami *breach* atau pelanggaran *SLA* akibat waktu *pending* yang terlalu lama. Pada contoh ini, *Service Catalog* yang dianalisis adalah kategori *Endpoint*, dengan rata-rata durasi *pending* yang melebihi 4–5 hari. Durasi ini jelas tidak sesuai dengan kesepakatan *SLA*, di mana waktu penyelesaian tiket seharusnya maksimal dalam 2 hari. Kondisi ini menunjukkan perlunya evaluasi dan tindak lanjut untuk mempercepat proses penyelesaian tiket pada kategori tersebut.

3.2.2.10 Komparasi *Data License 365* dan *Bitlocker 365*



Gambar 3.37 Data *License 365* dan *Bitlocker 365*.

Pada Minggu ke 14, data ini merupakan data *license* pada *Microsoft* yang ditarik melalui *Metabase O365* dan *Bitlocker 365* (Tim *Microsoft Admin*). Terdapat dua data: yang pertama adalah data *license* dari penggunaan 365, dan yang kedua penggunaan *Bitlocker 365*. Penggunaan 365 berisikan data *user* dengan *license* non-*pro* yang hanya bisa mengakses pada *Office 365*, sedangkan untuk penggunaan *Bitlocker 365* bisa mengakses *Office 365 Pro* seperti menggunakan basis *cloud*, *enterprise*, dan *Windows Pro*. Tujuan pada kali ini adalah untuk mengetahui perbandingan nilai data pada *license 365* dan juga *Bitlocker 365* dari masing-masing *domain*, sekaligus karena *license Bitlocker* seharusnya lebih sedikit karena hanya orang tertentu yang mempunyai *license* ini. Penanganan data ini dilakukan melalui *software Python* untuk dibersihkan, dipisahkan, lalu menghitung *license*, dan melakukan perbandingan berdasarkan masing-masing *domain*.

Gambar 3.38 Kode Menangani Data.

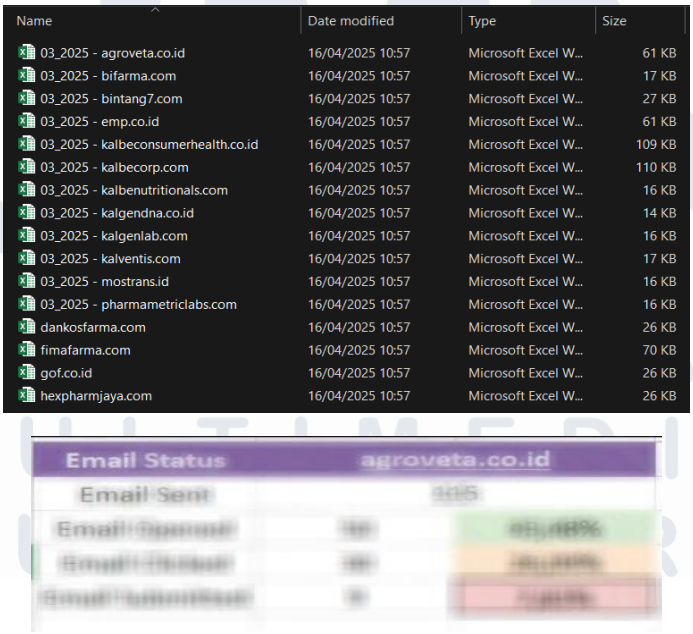
Sebelum melakukan perbandingan data, terlebih dahulu dilakukan pembersihan dan penanganan data karena ditemukan ketidakteraturan, seperti penulisan *domain* dalam huruf kapital seluruhnya atau penggunaan huruf kapital pada bagian awal *domain*. Untuk memastikan konsistensi, semua data pada kolom terkait diubah menjadi huruf kecil menggunakan fungsi *string* .lower(). Selanjutnya, dilakukan proses pemisahan (*splitting*) untuk mengambil *domain* dari alamat *email*, karena pada data *Bitlocker* tidak tersedia kolom khusus untuk *domain*. Hasil pemisahan ini disimpan dalam kolom baru yang berisi nama *domain*. Setelah itu, dilakukan penghitungan jumlah *user* berdasarkan *domain* pada masing-masing *dataframe* (*License 365* dan *Bitlocker 365*). Agar proses perbandingan dapat dilakukan, nama kolom dan format *domain* disamakan di kedua *dataframe*. Kemudian, kedua *dataframe* dilakukan penggabungan (*merge*), dan dilakukan perbandingan jumlah *user* berdasarkan *domain*. Setelah penggabungan, lalu menghitung selisih jumlah *user* antara *License 365* dan *Bitlocker*, serta menampilkan hasilnya dalam bentuk persentase. Dengan langkah-langkah tersebut, data berhasil disesuaikan dengan kebutuhan dan permintaan dari tim *Microsoft Admin*.

Domain	Tenan	Total User	Bitlocker user	Percentage	Differer
[redacted]	Corp	4	4	100%	0%
[redacted]	Corp	374	330	88%	12%
[redacted]	[redacted]	297	148	50%	50%
[redacted]	[redacted]	31	26	84%	16%
[redacted]	[redacted]	38	31	82%	18%
[redacted]	[redacted]	2.427	1064	44%	56%
[redacted]	[redacted]	120	41	34%	66%
[redacted]	[redacted]	36	15	42%	58%
[redacted]	[redacted]	290	69	24%	76%
[redacted]	[redacted]	102	56	55%	45%
[redacted]	[redacted]	17	22	129%	-29%
[redacted]	[redacted]	188	140	74%	26%
[redacted]	[redacted]	38	8	21%	79%
[redacted]	[redacted]	84	22	26%	74%
[redacted]	[redacted]	33	25	76%	24%
[redacted]	[redacted]	62	26	42%	58%
[redacted]	[redacted]	2	0	0%	100%
[redacted]	[redacted]	595	263	44%	56%
[redacted]	[redacted]	89	53	60%	40%
[redacted]	[redacted]	153	102	67%	33%
[redacted]	[redacted]	24	17	71%	29%

Gambar 3.39 Data *License 365* dan *Bitlocker 365*.

Setelah melalui berbagai proses pengolahan dan pembersihan data, hasil akhirnya telah siap diserahkan kepada tim *Microsoft Admin*, yang juga bertanggung jawab terhadap pengelolaan lisensi *Microsoft*. Data ini menghasilkan *output* berupa perbandingan antara jumlah lisensi *Microsoft 365* dan *Bitlocker 365* per *domain*. Dari hasil analisis persentase, ditemukan bahwa sebagian besar *domain* memiliki jumlah pengguna yang konsisten antara *Microsoft 365* dan *Bitlocker*, meskipun ada beberapa kasus di mana jumlah pengguna *Bitlocker* lebih sedikit. Hal ini masih dapat diterima karena *Bitlocker* hanya digunakan oleh *user* tertentu. Namun, ditemukan juga anomali di mana jumlah pengguna *Bitlocker* justru lebih banyak dibandingkan lisensi *Microsoft 365*. Kejanggalan ini perlu ditelusuri lebih lanjut untuk memastikan validitas data. Jika setelah pemeriksaan data tersebut terbukti benar, maka tim *Microsoft Admin* perlu melakukan penyesuaian, karena secara ideal jumlah lisensi *Microsoft 365* seharusnya lebih besar dari *Bitlocker*. Terlebih lagi, biaya lisensi *Bitlocker* lebih tinggi dibandingkan lisensi *Microsoft 365* reguler. Dengan adanya temuan ini, tim terkait dapat melakukan evaluasi dan efisiensi penggunaan lisensi secara lebih akurat serta mencegah pemborosan anggaran akibat kelebihan lisensi *Bitlocker*.

3.2.2.11 Data *Mail Cleansing, merge,* dan *compare* persentase



Name	Date modified	Type	Size
03_2025 - agroveta.co.id	16/04/2025 10:57	Microsoft Excel W...	61 KB
03_2025 - bifarma.com	16/04/2025 10:57	Microsoft Excel W...	17 KB
03_2025 - bintang7.com	16/04/2025 10:57	Microsoft Excel W...	27 KB
03_2025 - emp.co.id	16/04/2025 10:57	Microsoft Excel W...	61 KB
03_2025 - kalbeconsumerhealth.co.id	16/04/2025 10:57	Microsoft Excel W...	109 KB
03_2025 - kalbecorp.com	16/04/2025 10:57	Microsoft Excel W...	110 KB
03_2025 - kalbenutritionals.com	16/04/2025 10:57	Microsoft Excel W...	16 KB
03_2025 - kalgendna.co.id	16/04/2025 10:57	Microsoft Excel W...	14 KB
03_2025 - kalgenlab.com	16/04/2025 10:57	Microsoft Excel W...	16 KB
03_2025 - kalventis.com	16/04/2025 10:57	Microsoft Excel W...	17 KB
03_2025 - mostrans.id	16/04/2025 10:57	Microsoft Excel W...	16 KB
03_2025 - pharmametriclabs.com	16/04/2025 10:57	Microsoft Excel W...	16 KB
dankosfarma.com	16/04/2025 10:57	Microsoft Excel W...	26 KB
fimafarma.com	16/04/2025 10:57	Microsoft Excel W...	70 KB
gof.co.id	16/04/2025 10:57	Microsoft Excel W...	26 KB
hexpharmjaya.com	16/04/2025 10:57	Microsoft Excel W...	26 KB

Email Status	agroveta.co.id
Email Sent	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]

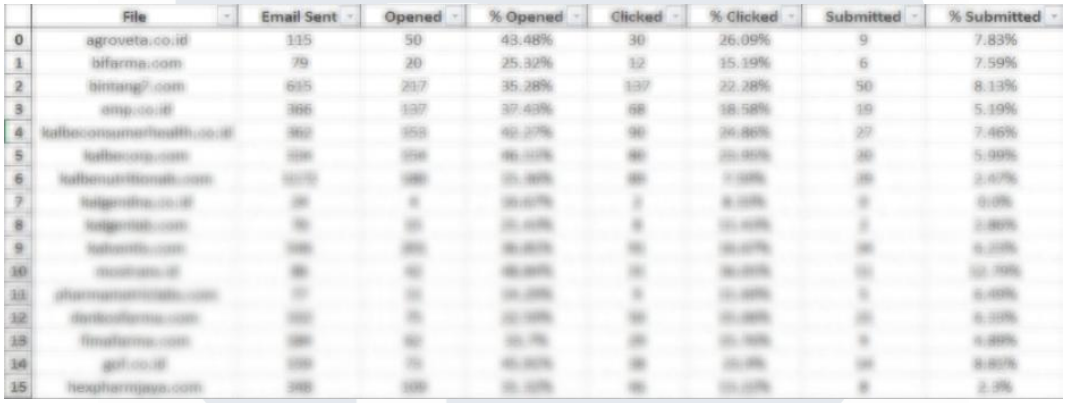
Gambar 3.40 *Data Mail* dari semua kalbe group.

Pada minggu ke 14. pekerjaan ini, bertanggung jawab untuk melakukan proses *data cleansing* dan *preprocessing* terhadap data *summary*. Awalnya, tim *Microsoft Admin* mengumpulkan data dari masing-masing perusahaan dalam bentuk file *Excel* yang dikirim melalui *email*. Namun, karena jumlah file yang sangat banyak, proses ini menjadi tidak efisien—terutama saat tim harus membuat ringkasan (*summary*) dari seluruh data, yang salah satu tujuannya adalah untuk mengidentifikasi *email* yang terindikasi *phishing*. Untuk mengatasi hal tersebut, terdapat inisiatif untuk menggabungkan seluruh file *Excel* menjadi satu file terintegrasi yang merangkum informasi dari tiap perusahaan. File gabungan ini menyajikan data secara ringkas, mencakup jumlah *email* yang dibuka (*opened*), dikirim (*submitted*), dan diklik (*clicked*), sehingga proses analisis menjadi jauh lebih efisien dan efektif.

Gambar 3.41 Data *Cleansing-Merge*.

Kode ini digunakan untuk melakukan menggabungkan dan membersihkan data dari file *email* yang akan diberikan tim terkait. Pertama, menggabungkan seluruh file data *email* yang akan dikomparasi. Setelah penggabungan, dilakukan proses pembersihan data (*data cleaning*) untuk menghilangkan *string* atau teks yang tidak diperlukan dan mengganggu analisis terutama dengan nama *domain* yang seperti 03_25 omg.com menjadi omg.com. Selanjutnya, membuat kode untuk

menghitung persentase interaksi pada setiap file, seperti berapa banyak pengguna yang telah membuka (*opened*), mengirim (*submitted*), atau mengeklik (*clicked*) *email*. Hasil perhitungan ini disajikan dalam bentuk persentase per perusahaan, sehingga memudahkan pembaca untuk memahami tingkat interaksi *email* di masing-masing *domain*. Setelah seluruh proses selesai, data akhir diekspor ke file *Excel* baru yang siap digunakan.



	File	Email Sent	Opened	% Opened	Clicked	% Clicked	Submitted	% Submitted
0	agroveta.co.id	115	50	43.48%	30	26.09%	9	7.83%
1	bifarma.com	79	20	25.32%	12	15.19%	6	7.59%
2	bintang7.com	615	217	35.28%	137	22.28%	50	8.13%
3	emp.co.id	366	137	37.43%	68	18.58%	19	5.19%
4	kalbeconsumerhealth.co.id	362	153	42.27%	90	24.86%	27	7.46%
5	kalbecorp.com	334	154	46.11%	80	23.95%	20	5.99%
6	kalbenutritionals.com	1172	580	21.48%	85	7.59%	28	2.47%
7	kalgendra.co.id	24	4	16.67%	2	8.33%	0	0.0%
8	kalgenlab.com	76	15	21.44%	8	11.43%	2	2.86%
9	kalsentis.com	346	201	36.81%	91	16.67%	34	6.23%
10	mostrans.id	86	42	48.84%	11	14.47%	11	12.79%
11	pharmanetindo.com	77	31	39.29%	9	11.48%	5	6.49%
12	sankofarma.com	332	75	22.59%	50	15.06%	20	6.33%
13	tmafarma.com	189	62	32.7%	29	15.76%	9	4.89%
14	gof.co.id	159	75	45.91%	38	23.9%	14	8.81%
15	hexpharmjaya.com	348	109	31.32%	46	13.15%	8	2.3%

Gambar 3.42 *Output* Data *Mail*.

Setelah kode tersebut dijalankan pada 18 file *Excel* yang berbeda, seluruh data berhasil digabungkan menjadi satu file terintegrasi yang menyajikan informasi terkait presentase *email* secara menyeluruh. File ini berisi data dari 18 *domain* perusahaan, mencakup jumlah *email* yang dikirim, dibuka, dan diklik—masing-masing disajikan dalam bentuk kuantitas dan persentase. Dengan data yang telah dibersihkan dan disusun sesuai kebutuhan tim *Microsoft Admin*, file ini kemudian diserahkan kepada tim terkait untuk digunakan dalam presentasi kepada tim *Microsoft Admin*. Tujuan utamanya adalah untuk membantu proses identifikasi dan pencegahan *email phishing*, yang belakangan ini semakin sering terjadi di lingkungan perusahaan.

3.2.2.12 Operational Pharma *Report* Maret 2025

Minggu ke-14–15. Pada periode Maret hingga Mei 2025, sebagai bagian dari tim *Data Analyst*, saya terlibat dalam pembuatan *Operational Pharma Report* yang ditujukan untuk *Pharma Group* di Kalbe. Laporan ini disusun

dengan tujuan untuk memantau performa operasional *IT Service Management (ITSM)*, khususnya terkait *incident* dan *service request* pada setiap layanan yang tersedia.

Melalui laporan ini, performa masing-masing layanan dapat dianalisis berdasarkan jumlah dan penyelesaian tiket yang masuk. Dengan demikian, manajemen dapat memperoleh gambaran menyeluruh mengenai efektivitas dan efisiensi operasional TI di lingkungan *Pharma Group*. Pembuatan laporan ini dilakukan secara rutin setiap bulan oleh tim *SOSM* sebagai bentuk evaluasi dan pengawasan terhadap kinerja operasional, sehingga potensi masalah dapat diidentifikasi lebih awal dan ditangani secara proaktif.

Gambar 3.43 Operational Pharma *Report* Bulan Maret 2025

Berdasarkan hasil laporan operasional pada bulan Maret 2025 (Gambar 3.43), jumlah tiket kategori *Incident* menunjukkan penurunan sebanyak 13 tiket dibandingkan bulan sebelumnya. Penurunan ini merupakan sinyal yang positif karena semakin sedikit tiket *Incident*, maka dapat diasumsikan semakin berkurang pula gangguan atau masalah yang terjadi pada layanan TI. Namun, di sisi lain, jumlah *Service Request* tetap sama seperti bulan sebelumnya. Berbeda dengan *Incident*, penurunan jumlah *Service Request* justru mengindikasikan hal yang kurang baik. Kondisi ini dapat mencerminkan rendahnya tingkat penggunaan layanan TI oleh pengguna atau adanya hambatan dalam proses permintaan layanan. Penurunan ini bisa disebabkan oleh beragam faktor, seperti kurangnya pemahaman *user* terhadap layanan yang tersedia, akses yang terbatas, atau potensi masalah dalam komunikasi dan dokumentasi layanan TI. Rangkuman eksekutif untuk bulan ini juga mencatat beberapa temuan penting terkait pelaksanaan layanan. Pada periode bulan Mei, tercatat

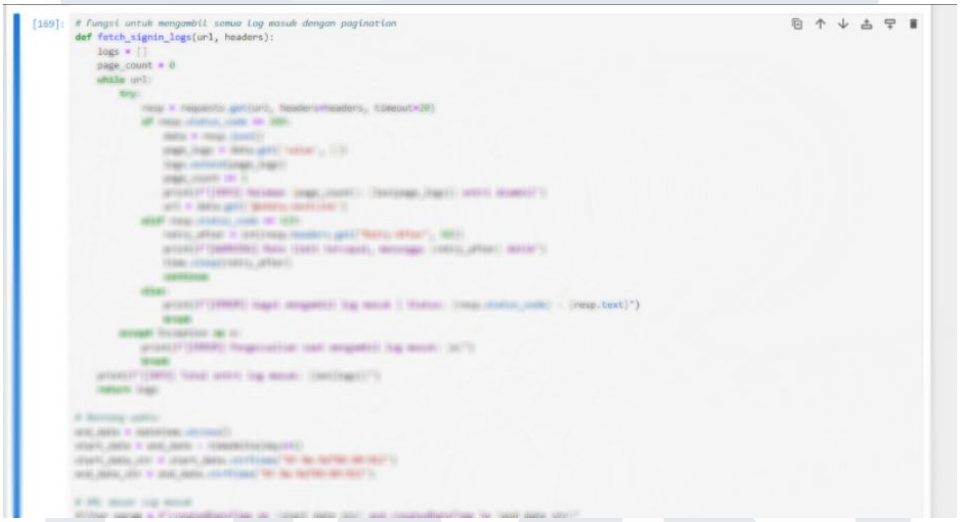
sejumlah *engineer* yang melanggar *Service Level Agreement (SLA)*, khususnya pada waktu respon. Salah satu penyebab utama pelanggaran ini adalah tiket yang sudah di-*assign* namun tidak segera diambil atau diproses oleh *engineer*. Hal ini menyebabkan waktu respon terlewati dan tiket mengalami *breach*. Selain itu, ditemukan penyalahgunaan fitur *Waiting for Customer* oleh beberapa *engineer*.

Alasan yang diberikan dalam fitur ini sering kali tidak valid atau tidak relevan dengan kondisi sebenarnya, sehingga memperlambat proses penyelesaian tiket. Di sisi lain, terdapat juga beberapa tiket yang masih melanggar batas waktu respon yang telah ditentukan, serta adanya kekurangan dalam pengisian bagian *task resolution detail*, di mana informasi yang dicantumkan masih minim atau kurang informatif. Dari sisi pengalaman pengguna, juga ditemukan beberapa kasus di mana *user* memberikan *rating* yang kurang baik terhadap layanan TI. Salah satu penyebab utamanya adalah tiket yang ditutup meskipun penyelesaian masalah belum benar-benar tuntas, yang menyebabkan ketidakpuasan dari sisi pengguna. Secara keseluruhan, temuan-temuan tersebut menjadi masukan penting untuk peningkatan proses layanan ke depannya. Diperlukan perhatian lebih dalam hal *monitoring* tiket, ketepatan waktu respon, dan akurasi dokumentasi oleh *engineer* agar kualitas layanan TI terus membaik dan memenuhi ekspektasi pengguna di lingkungan operasional *Pharma*.

3.2.2.13 Penarikan data *SignIn Log* dengan *Microsoft API*

Pada minggu ke 15 hingga minggu 17, Dalam proses penarikan data kali ini, melakukan pengambilan data *Sign-In Log* menggunakan *Microsoft Graph API* melalui bahasa pemrograman *Python*. Penarikan data ini dilakukan atas permintaan dari tim *Microsoft Admin*, yang bertujuan untuk memantau aktivitas *login* setiap karyawan. *Monitoring* ini mencakup deteksi *login* yang mencurigakan, permasalahan akses aplikasi, serta memastikan bahwa setiap akses dilakukan oleh pihak yang sah dan sesuai. Sebelum proses penarikan data dilakukan, terlebih

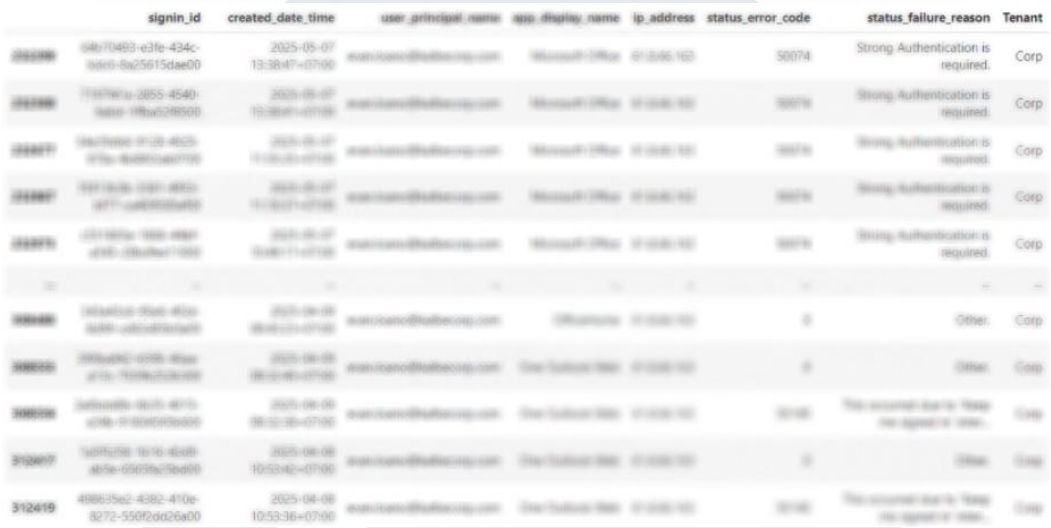
dahulu melakukan koordinasi dengan tim *Microsoft Admin* untuk mendapatkan akses yang diperlukan. Beberapa *permission* diberikan melalui *Microsoft Entra (Azure)*, di antaranya adalah *AuditLog.Read.All*, yang berfungsi untuk memberikan akses penuh terhadap seluruh data *audit log* termasuk *Sign-In Log*; *Directory.Read.All*, yang memungkinkan akses terhadap informasi direktori pengguna dan grup untuk melengkapi data seperti nama pengguna dan struktur organisasi; serta *User.Read.All* sebagai akses tambahan opsional untuk menampilkan informasi pengguna secara lebih rinci bila diperlukan dalam analisis. Dengan seluruh akses tersebut, proses penarikan data *Sign-In Log* dapat dilakukan dengan menyeluruh dan sesuai kebutuhan *monitoring* keamanan akses pengguna.



Gambar 3.44 Kode *Python* untuk menarik data.

Setelah *permission* berhasil diberikan (*granted*), langkah selanjutnya adalah melakukan implementasi kode sebagaimana ditunjukkan pada Gambar 3.44. Kode ini digunakan untuk mengakses *tenant* dan menarik seluruh data dari *Sign-In Log*. Kode ini telah disesuaikan dengan *permission* yang diberikan dan dirancang untuk menangani data dalam jumlah besar melalui mekanisme *pagination*, guna mencegah beban berlebih baik di sisi *server* maupun *client*. Selanjutnya, dilakukan *filtering* berdasarkan waktu pada data *Sign-In Log*. Hal ini penting karena *Microsoft* hanya menyimpan data *log* hingga 180 hari (6 bulan). Tanpa *filter* waktu, proses penarikan data dapat menimbulkan *error*, dan proses ini dilengkapi dengan *looping*

untuk masing-masing *tenant*, sehingga data dapat ditarik dari kedua *tenant* sekaligus, yaitu *tenant Corp* dan *Pharma*.

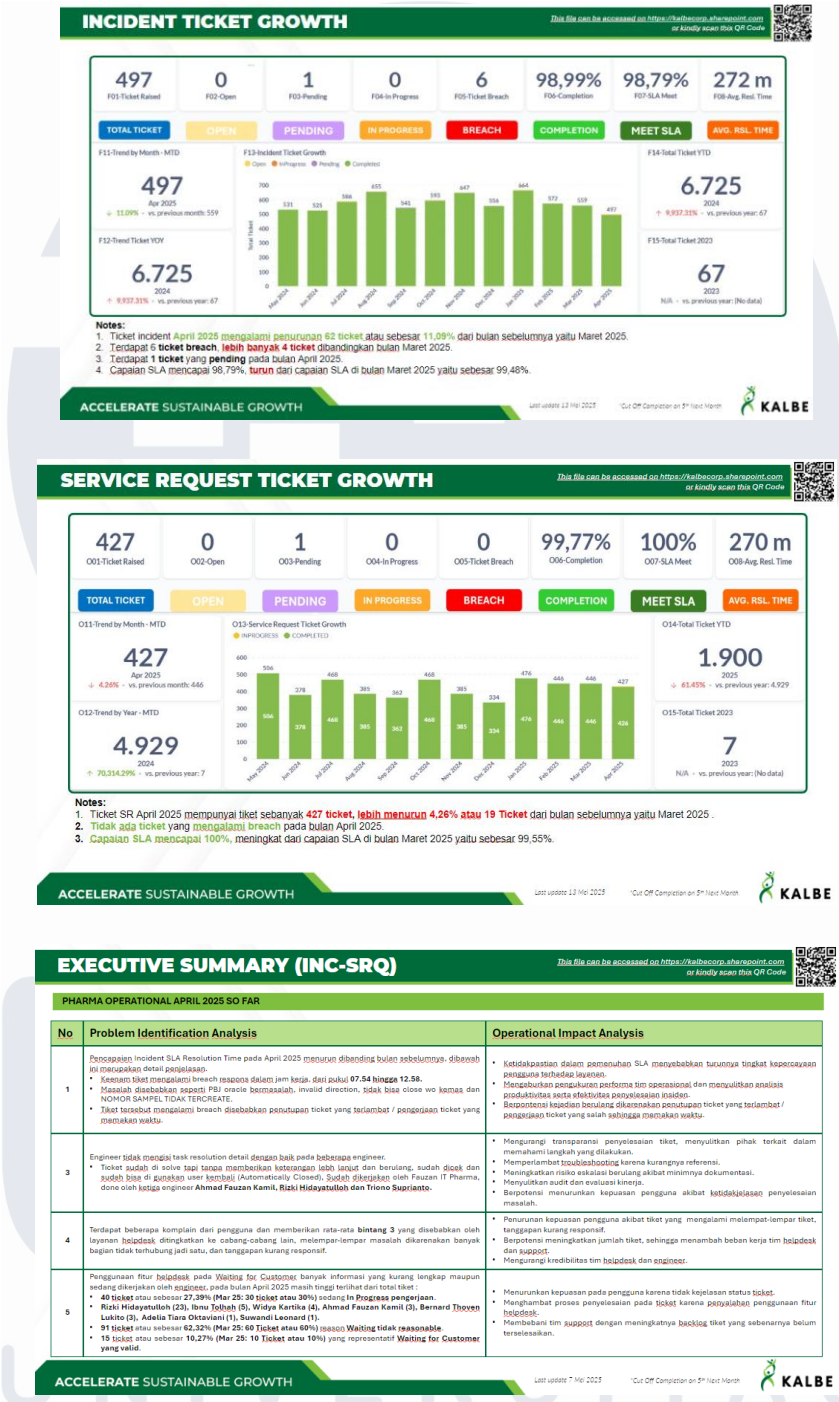


	signin_id	created_date_time	user_principal_name	app_display_name	ip_address	status_error_code	status_failure_reason	Tenant
[redacted]	[redacted]	2025-05-07 13:38:47+07:00	[redacted]	[redacted]	[redacted]	50074	Strong Authentication is required.	Corp
[redacted]	[redacted]	2025-05-07 [redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Strong Authentication is required.	Corp
[redacted]	[redacted]	2025-05-07 [redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Strong Authentication is required.	Corp
[redacted]	[redacted]	2025-05-07 [redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Strong Authentication is required.	Corp
[redacted]	[redacted]	2025-05-07 [redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Strong Authentication is required.	Corp
...	...	...	...	...	...	...	...	...
[redacted]	[redacted]	2025-04-08 [redacted]	[redacted]	[redacted]	[redacted]	0	Other.	Corp
[redacted]	[redacted]	2025-04-08 [redacted]	[redacted]	[redacted]	[redacted]	0	Other.	Corp
[redacted]	[redacted]	2025-04-08 [redacted]	[redacted]	[redacted]	[redacted]	[redacted]	This account does not exist in tenant '[redacted]' and cannot access the application '[redacted]' in that tenant. The account needs to be added as an external user in the tenant first. Sign out and sign in again with a different Azure Active Directory user account.	Corp
[redacted]	[redacted]	2025-04-08 [redacted]	[redacted]	[redacted]	[redacted]	0	Other.	Corp
312419	49863a2-4382-410e-8272-550f2dd26a00	2025-04-08 10:53:36+07:00	[redacted]	[redacted]	[redacted]	[redacted]	This account does not exist in tenant '[redacted]' and cannot access the application '[redacted]' in that tenant. The account needs to be added as an external user in the tenant first. Sign out and sign in again with a different Azure Active Directory user account.	Corp

Gambar 3.45 Hasil penarikan data *Sign in Log*.

Proses eksekusi kode tersebut memerlukan waktu yang cukup panjang, yaitu sekitar 1 hingga 2 hari kerja, hingga seluruh data berhasil ditarik. Data yang diperoleh mencakup informasi penting seperti waktu *login* terakhir, nama pengguna, alamat *IP*, status *error* (jika ada), serta identitas *tenant* dari perusahaan terkait. Total data yang berhasil ditarik mencapai kurang lebih 600.000 baris, berasal dari *tenant Corp* dan *tenant Pharma*. Setelah data berhasil dikumpulkan, informasi tersebut akan diserahkan kepada tim *Microsoft Admin* untuk dilakukan analisis dan penyelidikan lebih lanjut. Hal ini bertujuan untuk mengidentifikasi potensi permasalahan dalam proses *login*, serta mendeteksi aktivitas yang mencurigakan yang mungkin terjadi pada akun pengguna.

3.2.2.14 Operational Pharma *Report* April 2025



Gambar 3.46 Operational Pharma *Report* Bulan April 2025

Pada minggu 17 - 18, berdasarkan hasil laporan operasional bulan April 2025 yang ditampilkan pada Gambar 3.46, jumlah tiket pada kategori *Incident* mengalami penurunan yang cukup signifikan, yaitu sebanyak 62 tiket

dibandingkan bulan sebelumnya. Penurunan ini menjadi indikator positif karena menandakan semakin sedikitnya gangguan atau permasalahan pada layanan TI yang dilaporkan oleh pengguna. Namun, berbeda halnya dengan kategori *Service Request*, yang juga mengalami penurunan sebanyak 19 tiket. Penurunan pada jenis tiket ini justru mengindikasikan kondisi yang kurang baik, karena menurunnya jumlah permintaan layanan dapat mencerminkan rendahnya pemanfaatan layanan TI oleh *user*.

Hal ini bisa disebabkan oleh berbagai faktor seperti kurangnya *awareness*, hambatan dalam proses permintaan, atau akses layanan yang tidak optimal. Dalam rangkuman temuan bulan ini, terdapat sejumlah hal yang perlu menjadi perhatian khusus. Salah satunya adalah masih ditemukannya tiket yang mengalami *breach* pada *Resolution Time*. Penyebab utama dari pelanggaran ini adalah keterlambatan dalam proses penyelesaian tiket atau proses penutupan yang tidak dilakukan tepat waktu. Beberapa *engineer* juga masih menunjukkan konsistensi yang kurang dalam mengisi kolom *Task Resolution Detail* secara informatif. Masalah ini terjadi secara berulang dan menunjukkan perlunya peningkatan dalam dokumentasi hasil pekerjaan. Selain itu, fitur *Waiting for Customer* juga masih sering disalahgunakan. Ditemukan bahwa alasan yang dicantumkan oleh *engineer* dalam status tersebut tidak sesuai dengan kondisi sebenarnya—beberapa tiket bahkan masih dikerjakan namun tetap diberi status menunggu pelanggan. Hal ini tentu mengganggu akurasi data operasional dan memperlambat proses penyelesaian secara keseluruhan. Secara umum, meskipun terdapat perbaikan dari sisi penurunan jumlah tiket *Incident*, masih terdapat beberapa aspek dalam kualitas layanan dan kedisiplinan operasional yang perlu ditingkatkan, khususnya dalam hal dokumentasi penyelesaian, pemanfaatan fitur secara tepat, dan ketepatan waktu dalam penyelesaian tiket. Semua temuan ini diharapkan dapat menjadi landasan perbaikan berkelanjutan untuk mendukung peningkatan kualitas layanan TI di lingkungan operasional *Pharma*.

3.2.2.15 Model *Machine Learning Waiting Reasons (Text Classification)*

Pada minggu ke 15 hingga minggu 20, dalam sistem *IT Service Management (ITSM)*, terdapat berbagai jenis tiket seperti *change request*, *incident*, dan *service request*. Selama proses penanganan, *engineer* terkadang perlu melakukan penundaan (*waiting*) terhadap tiket tertentu, yang disebabkan oleh berbagai faktor, seperti kendala dari pihak pengguna, proses penarikan data yang memerlukan waktu lama, atau kondisi lain yang membuat penyelesaian tiket tidak dapat segera dilakukan. Dalam situasi tersebut, *engineer* diharuskan mengisi alasan penundaan (*waiting reason*) pada tiket yang bersangkutan. Untuk mempermudah proses *monitoring* dan evaluasi, tim *SOSM* mengelompokkan alasan-alasan *waiting* tersebut ke dalam tiga kategori utama. Pertama adalah kategori *In Progress*, yang berarti alasan penundaan berkaitan langsung dengan pekerjaan yang sedang berjalan. Kedua adalah *Unclear Reason*, yaitu alasan yang diberikan tidak cukup jelas atau tidak informatif, misalnya deskripsi yang terlalu singkat atau panjang namun tidak menyampaikan informasi yang berarti. Ketiga adalah *Reasonable*, di mana alasan yang diberikan dapat dipahami dengan jelas dan dianggap masuk akal oleh tim terkait.

Selama ini, proses pengelompokan tersebut dilakukan secara manual oleh tim *SOSM*, yang tentunya sangat memakan waktu dan kurang efisien, terutama karena proses ini dilakukan secara rutin untuk kebutuhan pelaporan bulanan. Dalam pelaporan tersebut, Departemen *IT* perlu meninjau kembali semua tiket yang berada dalam status *waiting* beserta alasan yang diberikan. Jika ditemukan alasan yang tidak masuk akal, *engineer* terkait akan dimasukkan ke dalam proses *audit* sebagai bagian dari evaluasi performa. Untuk mengatasi permasalahan ini, memerlukan pengembangan model *text classification* yang mampu memprediksi kategori *waiting reason* secara otomatis berdasarkan teks yang dituliskan oleh *engineer*. Dengan adanya model ini, tim *SOSM* tidak perlu lagi melakukan pelabelan secara manual, sehingga proses pelaporan bulanan menjadi lebih cepat dan efisien, khususnya dalam hal *audit* terhadap tiket-tiket yang masuk dalam status *waiting*. Data yang digunakan untuk pelatihan model berasal dari periode tahun 2024 hingga

April 2025, dengan total 10.092 baris data dan 9 kolom, termasuk salah satunya kolom *"Waiting Reason"* yang menjadi fokus klasifikasi, sebagaimana ditampilkan dalam *Table 3.2*.

Tabel 3.2 Kolom Dataset

No	Kolom	Tipe data	Informasi
1	TicketNo	object	Nomor pada pembuatan layanan service (ticket)
2	TicketCreated	datetime64[ns]	Tanggal pembuatan ticket
3	TicketTeam	object	Tim yang melaporkan ticket
4	TaskSequence	object	Tugas yang dikerjakan oleh engineer
5	TaskNo	object	Nomor pada task
6	Engineer	object	Nama engineer yang menangani ticket
7	OwnerTeam	object	Tim yang menangani tiket
8	Unclear Reason	object	Kategori pada alasan (In progress, Reasonable, Unclear Reasons)
9	WaitingReason	object	Alasan engineer melakukan penungguan ticket

Sebelum membangun model *text classification*, terdapat serangkaian tahapan yang perlu dilakukan untuk memastikan pemahaman yang menyeluruh terhadap proses analisis data yang akan dijalankan. Tahapan ini dimulai dengan penyusunan *flowchart* berdasarkan kerangka kerja *CRISP-DM* (*Cross Industry Standard Process for Data Mining*) [13]. *CRISP-DM* merupakan metodologi standar yang banyak digunakan dalam pengembangan proyek *data mining* dan *machine learning* karena mampu memberikan alur kerja yang sistematis dan terstruktur dari awal hingga akhir proyek. *CRISP-DM* terdiri dari enam tahapan utama yang saling terhubung. Proses diawali dengan pemahaman terhadap kebutuhan dan tujuan

bisnis atau dikenal sebagai *business understanding* [13]. Dalam konteks ini, contohnya adalah kebutuhan untuk mengklasifikasikan alasan *waiting reason* agar proses operasional *IT* lebih efisien. Setelah memahami tujuan bisnis, dilanjutkan dengan memahami karakteristik data yang dimiliki. Tahap ini dikenal sebagai *data understanding*, di mana dilakukan eksplorasi awal terhadap data teks, visualisasi menggunakan *word cloud*, dan pengecekan kualitas data seperti keberadaan nilai kosong atau inkonsistensi.

Langkah berikutnya adalah *data preparation*, di mana dilakukan pembersihan dan transformasi data agar siap digunakan untuk pemodelan. Ini termasuk proses *natural language processing* seperti penghapusan *stopwords*, *stemming*, dan *tokenisasi* agar data teks menjadi lebih terstruktur. Setelah data siap, tahap *modeling* dilakukan, yaitu pembangunan model klasifikasi teks menggunakan beberapa algoritma *machine learning*. Dalam praktiknya, dilakukan komparasi terhadap 10 algoritma berbeda untuk menemukan model dengan performa terbaik, di mana *Support Vector Machine* (*SVM*) yang dioptimalkan dengan *fine-tuning SBERT* terbukti memberikan hasil akurasi tertinggi, yaitu di atas 98%. Setelah model dibangun, tahap selanjutnya adalah *evaluation*, di mana model diuji menggunakan berbagai metrik seperti *confusion matrix*, *accuracy score*, dan *classification report* untuk melihat seberapa baik model mampu mengenali pola dalam data. Terakhir adalah tahap *deployment*, yaitu mengimplementasikan model ke dalam proses bisnis agar dapat digunakan untuk memprediksi data baru secara otomatis dan mendukung pengambilan keputusan yang lebih cepat dan efisien. Berikut ini merupakan gambar dari *CRISP-DM* dalam pembuatan model *Text Classification* pada *Gambar 3.47*.

Gambar 3.47 *CRISP-DM* [14]

1. *Business Understanding*

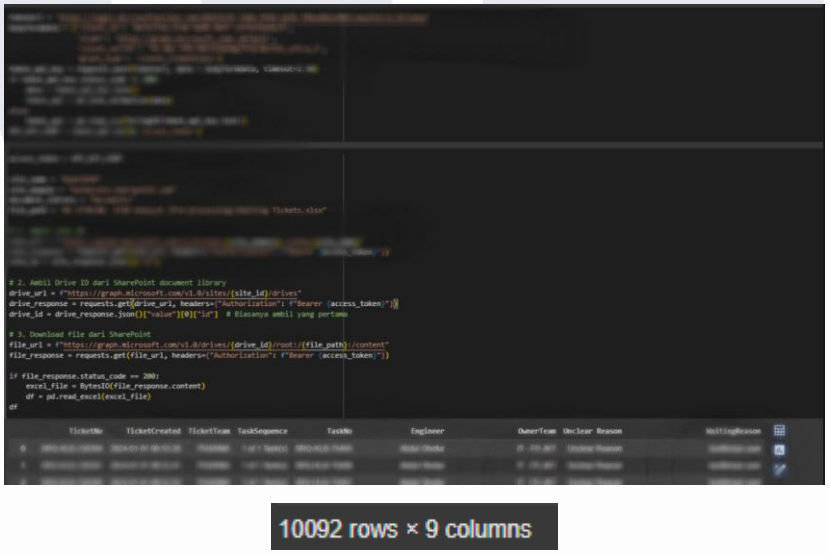
Pada tahap *business understanding* dalam proyek model ini, fokus utama adalah memahami pola dan alasan di balik status *"waiting reason"* yang diberikan oleh *engineer* ketika menangani tiket. Terdapat indikasi bahwa dalam beberapa kasus, *engineer* memanfaatkan status menunggu ini secara tidak optimal—misalnya dengan memperpanjang durasi pengerjaan tanpa penjelasan yang memadai. Situasi ini menimbulkan kebutuhan untuk melakukan analisis lebih mendalam terhadap bagaimana *engineer* menyampaikan alasan mereka.

Oleh karena itu, diperlukan pengembangan sebuah model yang dapat membantu mengelompokkan alasan-alasan tersebut ke dalam kategori yang lebih terstruktur. Tujuannya adalah untuk mengetahui apakah alasan yang diberikan tergolong jelas dan logis (*reasonable*), menunjukkan bahwa tugas memang sedang dikerjakan (*in progress*), atau justru tidak memberikan kejelasan sama sekali (*unclear reasons*). Dengan memahami karakteristik dari setiap alasan, perusahaan dapat memperoleh gambaran yang lebih akurat mengenai perilaku *engineer* dalam menangani tiket. Hal ini juga penting dalam membantu proses evaluasi kinerja dan memastikan bahwa waktu yang dihabiskan dalam status menunggu benar-benar digunakan secara efektif.

2. *Data Understanding*

Pada tahap *data understanding*, dilakukan eksplorasi terhadap data yang berkaitan dengan aktivitas *engineer* dalam menangani tiket, khususnya pada

bagian alasan menunggu (*waiting reason*). Data ini berisi teks penjelasan yang dicantumkan oleh *engineer* ketika tiket berada dalam status *Waiting*. Tujuannya adalah untuk memahami struktur data, mengevaluasi kualitas data, serta mengidentifikasi kolom-kolom yang berpotensi digunakan dalam pengembangan model klasifikasi. Tahap awal proses ini memerlukan penarikan data dari Microsoft SharePoint melalui Microsoft Graph API, karena data disimpan dalam format Excel yang diunggah ke dalam SharePoint perusahaan. Proses ini digambarkan pada Gambar 3.48.



Gambar 3.48 Penarikan data *Waiting*

Tahap ini melakukan penarikan data *SharePoint* (*Excel*) *Microsoft API* melalui *Python*, lalu mengubah format data tersebut menjadi *dataframe* agar *Python* dapat membaca format tersebut dalam bentuk *dataframe*, hasilnya terdapat 10.092 baris dan 9 kolom pada data *waiting*.

```
RangeIndex: 10092 entries, 0 to 10091
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   TicketNo        10092 non-null  object
 1   TicketCreated   10092 non-null  datetime64[ns]
 2   TicketTeam      10092 non-null  object
 3   TaskSequence    10092 non-null  object
 4   TaskNo          9900 non-null   object
 5   Engineer        10092 non-null  object
 6   OwnerTeam       10092 non-null  object
 7   Unclear Reason  10092 non-null  object
 8   WaitingReason   10092 non-null  object
dtypes: datetime64[ns](1), object(8)
memory usage: 709.7+ KB
```

Gambar 3.49 Informasi dataset

Data *waiting* terdapat 9 kolom dan memiliki isi sebanyak 10.092 dengan hampir mayoritas tipe data berbentuk 8 *object* dan 1 *datetime*, walaupun terdapat *NaN* pada kolom *TaskNo* namun hal itu tidak menganggu pada melakukan pembuatan model klasifikasi.

```
df['Unclear Reason'].value_counts()
```

Unclear Reason	count
In Progress	5369
Unclear Reason	4047
Reasonabel	676

dtype: int64

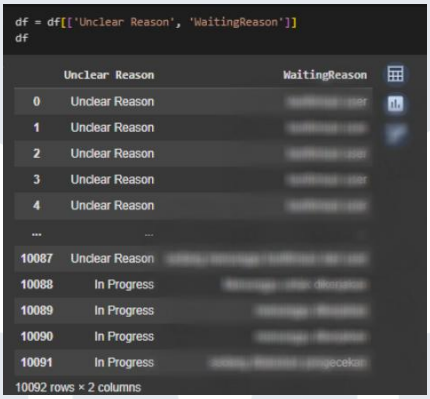
Gambar 3.50 Kelas pada *Unclear Reasons*

Selanjutnya dilakukan eksplorasi awal terhadap data dengan menggunakan fungsi *value_counts* untuk mengetahui distribusi jumlah data pada masing-masing kelas. Terdapat tiga kelas utama yang akan digunakan dalam pembuatan *model*. Kelas pertama adalah *In Progress*, yang menggambarkan situasi di mana *engineer* memang sedang mengerjakan *tiket* tersebut, dengan jumlah data sebanyak 5.369 baris. Kelas kedua adalah *Unclear Reason*, yaitu ketika *engineer* memberikan alasan yang tidak jelas atau ambigu saat *tiket* berada dalam status menunggu, dengan jumlah data sebanyak 4.047

baris. Terakhir, terdapat kelas *Reasonable*, yang menunjukkan bahwa *engineer* memberikan alasan menunggu yang logis dan dapat diterima, dengan jumlah data sebanyak 676 baris. Analisis ini penting untuk memberikan gambaran mengenai seberapa seimbang jumlah data pada masing-masing kelas, khususnya pada kolom alasan menunggu (*waiting reason*). Pemahaman terhadap distribusi ini akan sangat berguna dalam proses pembangunan model klasifikasi ke depannya.

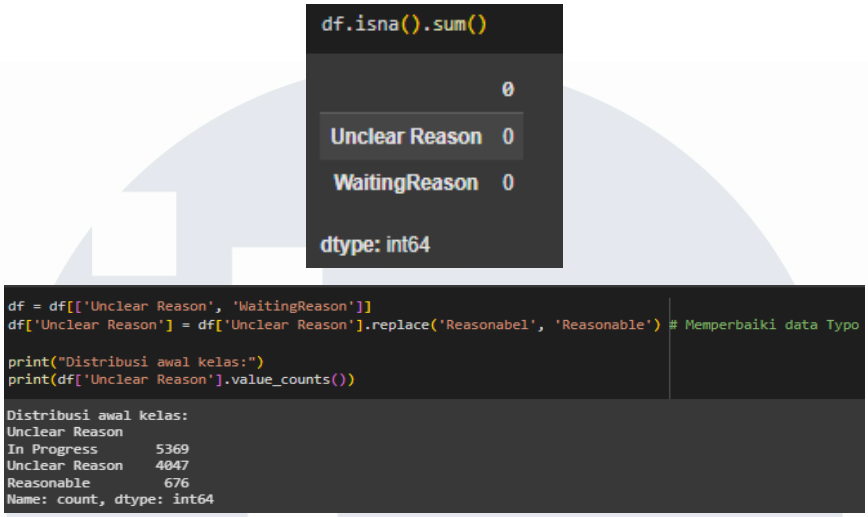
3. *Data Preparation*

Tahapan selanjutnya adalah *data preparation*, yaitu tahap yang memerlukan persiapan data sebelum melakukan pembuatan *model*, seperti melakukan *label encoding*, pemilihan kolom yang sesuai, maupun menghapus baris yang memiliki *value NaN*.



Gambar 3.51 Pemilihan kolom

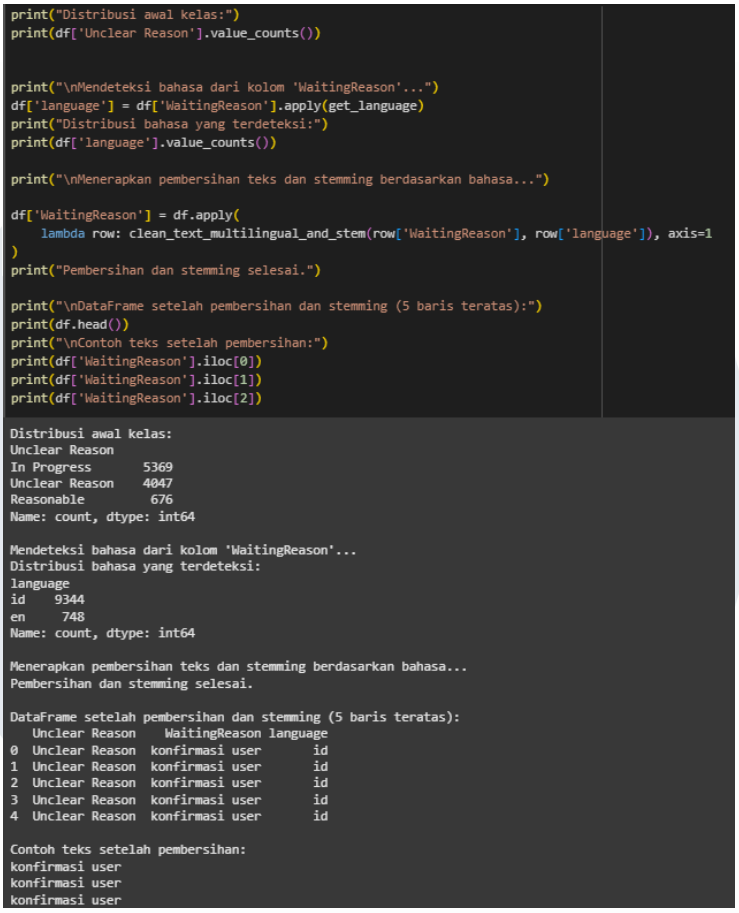
Gambar 3.51 menunjukkan proses pemilihan kolom yang dilakukan untuk keperluan analisis lanjutan. Pada tahap ini, dipilih dua kolom utama dari *dataframe*, yaitu *Unclear Reason* dan *WaitingReason*. Pemilihan ini bertujuan untuk memfokuskan data hanya pada bagian yang relevan, sehingga dapat mempermudah proses pembangunan *model classification* pada tahap berikutnya. Dengan menyederhanakan data hanya pada kolom yang diperlukan, proses *modeling* dapat berjalan lebih efisien dan terarah.



Gambar 3.52 Pengecekan *nan values* & merubah nama kelas *Reasonable*

Gambar 3.52 menunjukkan hasil dari penggunaan fungsi *isna()* yang bertujuan untuk memeriksa apakah terdapat baris dengan nilai *NaN* pada data. Dari hasil yang ditampilkan, dapat diketahui bahwa seluruh baris pada kolom yang dianalisis tidak memiliki nilai kosong, sehingga data dinyatakan bersih dan siap untuk diproses lebih lanjut dalam tahap analisis maupun pemodelan.

Selain itu, pada tahap ini juga dilakukan penyesuaian nama kelas untuk menjaga konsistensi penamaan. Salah satunya adalah perubahan nama kelas *"reasonabel"* menjadi *"reasonable"*, karena terdapat kesalahan penulisan pada data awal. Perubahan ini penting untuk memastikan bahwa semua label kelas ditulis dengan ejaan yang benar, agar tidak menimbulkan *error* atau ketidaksesuaian saat *model* membaca dan memproses data tersebut.



```
print("Distribusi awal kelas:")
print(df['Unclear Reason'].value_counts())


print("\nMendeteksi bahasa dari kolom 'WaitingReason'...")
df['language'] = df['WaitingReason'].apply(get_language)
print("Distribusi bahasa yang terdeteksi:")
print(df['language'].value_counts())

print("\nMenerapkan pembersihan teks dan stemming berdasarkan bahasa...")

df['WaitingReason'] = df.apply(
    lambda row: clean_text_multilingual_and_stem(row['WaitingReason'], row['language']), axis=1
)
print("Pembersihan dan stemming selesai.")

print("\nDataFrame setelah pembersihan dan stemming (5 baris teratas):")
print(df.head())
print("\nContoh teks setelah pembersihan:")
print(df['WaitingReason'].iloc[0])
print(df['WaitingReason'].iloc[1])
print(df['WaitingReason'].iloc[2])
```

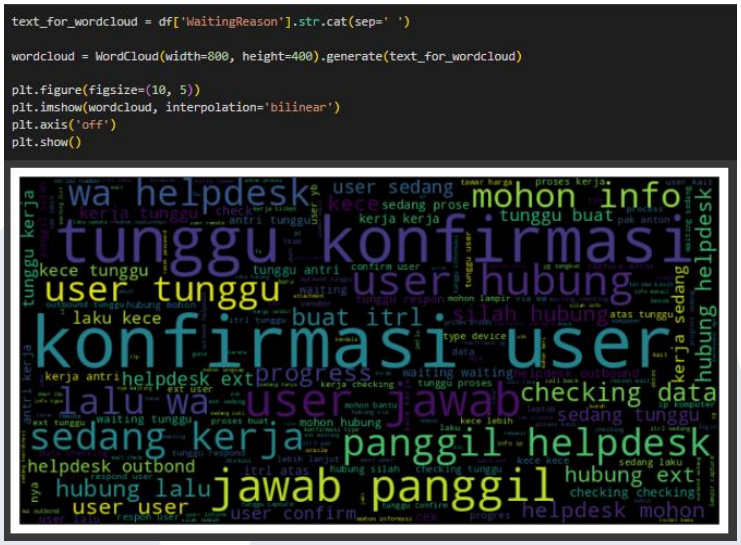
```
Distribusi awal kelas:
Unclear Reason
In Progress       5369
Unclear Reason    4047
Reasonable         676
Name: count, dtype: int64

Mendeteksi bahasa dari kolom 'WaitingReason'...
Distribusi bahasa yang terdeteksi:
language
id    9344
en     748
Name: count, dtype: int64

Menerapkan pembersihan teks dan stemming berdasarkan bahasa...
Pembersihan dan stemming selesai.

DataFrame setelah pembersihan dan stemming (5 baris teratas):
   Unclear Reason    WaitingReason language
0  Unclear Reason  konfirmasi user       id
1  Unclear Reason  konfirmasi user       id
2  Unclear Reason  konfirmasi user       id
3  Unclear Reason  konfirmasi user       id
4  Unclear Reason  konfirmasi user       id

Contoh teks setelah pembersihan:
konfirmasi user
konfirmasi user
konfirmasi user
```

Gambar 3.53 Pembersihan pada *text*

Selanjutnya, pada Gambar 3.53, dilakukan tahap pembersihan teks atau proses *Natural Language Preprocessing (NLP)* sebelum membangun *model*. Proses ini penting dilakukan agar teks dapat lebih mudah dipahami oleh *model*. Jika teks tidak dibersihkan terlebih dahulu, maka *model* akan kesulitan dalam mengenali pola dari kata-kata yang tidak konsisten atau tidak relevan. Dalam pembuatan *model* ini, pemilihan *library* untuk pembersihan teks menjadi hal yang penting dan perlu diperhatikan dengan saksama, karena hasil dari tahapan ini akan sangat berpengaruh terhadap kualitas *model* yang dihasilkan. Salah satu tahapan awal yang dilakukan adalah penghapusan *stop words*, yaitu kata-kata umum yang tidak memberikan makna penting terhadap konteks kalimat, baik dalam bahasa Indonesia maupun bahasa Inggris.

Hal ini karena teks pada kolom *Waiting Reason* mengandung campuran dua bahasa tersebut. Setelah menghapus *stop words*, dilakukan proses *stemming* untuk kedua bahasa—yaitu bahasa Indonesia dan bahasa Inggris—dengan tujuan mengubah kata menjadi bentuk dasarnya. Kemudian, dibuat fungsi khusus untuk mengidentifikasi apakah suatu kalimat ditulis dalam bahasa Indonesia atau bahasa Inggris, agar proses *stemming* dapat dilakukan sesuai bahasa masing-masing. Tahapan selanjutnya adalah proses *text cleaning*, yaitu menghapus simbol-simbol atau karakter khusus yang tidak memiliki kontribusi terhadap makna kalimat, seperti tanda baca, angka, atau karakter lain yang mengganggu. Setelah semua proses *NLP* diterapkan, teks pada kolom *Waiting Reason* menjadi lebih bersih, terstruktur, dan siap digunakan pada proses pemodelan di tahap berikutnya.

```
df['WaitingReason'] = df['WaitingReason'].astype(str)

text_df = df['WaitingReason'].str.cat(sep=' ')

wordcloud = WordCloud(width=800, height=400).generate(text_df)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



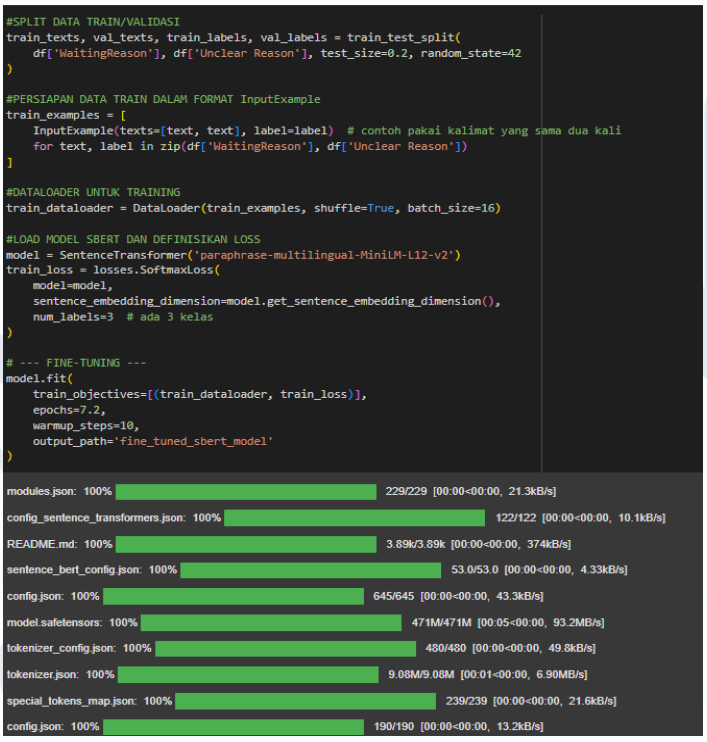
Gambar 3.54 Visualisasi *WordCloud*

Setelah proses *NLP* diterapkan pada data teks, Gambar 3.54 menampilkan visualisasi dalam bentuk *word cloud* untuk membandingkan kondisi teks sebelum dan sesudah dilakukan pembersihan. Pada visualisasi sebelum penerapan *NLP*, masih terlihat banyak kata-kata umum yang muncul mendominasi seperti “dari”, “tidak”, “dalam”, dan kata-kata lain yang secara makna tidak memberikan kontribusi signifikan terhadap pemahaman konteks. Setelah proses pembersihan dilakukan, hasil *word cloud* menunjukkan perbedaan yang cukup signifikan. Teks menjadi lebih padat makna karena kata-kata yang tidak relevan telah dihilangkan, dan kata-kata yang sebelumnya beragam bentuknya telah disederhanakan melalui proses *stemming*. Dengan hasil ini, teks menjadi lebih ringkas dan relevan, sehingga dapat membantu *model* dalam memahami konteks dan pola yang lebih akurat pada tahap pemodelan selanjutnya. Proses ini juga memberikan kontribusi penting dalam meningkatkan kualitas dan efisiensi *model* yang akan dibangun.

4. Modeling

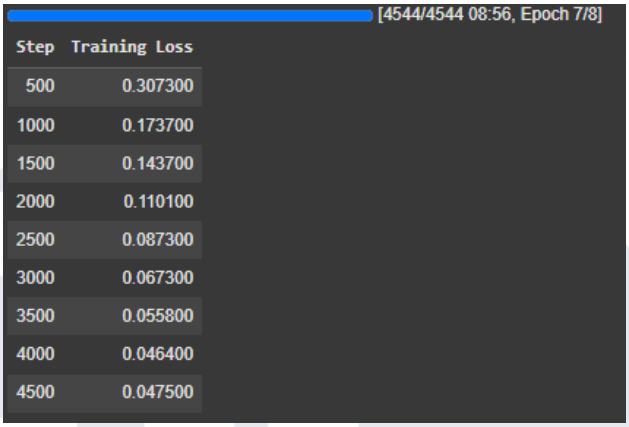
Setelah seluruh proses *data preparation* dilakukan—seperti pengecekan nilai kosong (*NaN*), penyesuaian nama kelas, penerapan proses *Natural*

Language Preprocessing (NLP), hingga visualisasi awal menggunakan *word cloud*—tahapan selanjutnya adalah melakukan *modeling*. Seluruh data telah dipastikan bersih dan siap digunakan untuk pelatihan *model*. Dalam proses pemodelan ini, digunakan pendekatan *sentence embedding* dengan *model Sentence-BERT (SBERT)*. Pemilihan *SBERT* dilakukan karena karakteristik teks pada kolom *Waiting Reason* umumnya terdiri dari dua kata atau lebih yang memiliki makna tertentu saat digabungkan. Pendekatan seperti *TF-IDF* atau *FastText* cenderung hanya memperhatikan frekuensi kata atau vektor per kata tanpa memahami konteks kalimat secara utuh, sehingga kurang optimal dalam menangani teks penjelasan yang bersifat semantik seperti ini. *SBERT* memiliki keunggulan dalam memahami makna dari keseluruhan kalimat karena dirancang untuk menghasilkan representasi vektor yang mencerminkan hubungan antar-kata dalam satu kesatuan makna. Dengan demikian, pemilihan *SBERT* dinilai tepat dan efektif untuk menangani jenis data teks yang digunakan dalam proyek ini, sehingga dapat meningkatkan akurasi *model* dalam mengklasifikasikan alasan penundaan tiket menjadi *Reasonable*, *In Progress*, atau *Unclear*.



```
#SPLIT DATA TRAIN/VALIDASI
train_texts, val_texts, train_labels, val_labels = train_test_split(
    df['WaitingReason'], df['Unclear Reason'], test_size=0.2, random_state=42
)

#PERSIAPAN DATA TRAIN DALAM FORMAT InputExample
train_examples = [
    InputExample(texts=[text, text], label=label)  # contoh pakai kalimat yang sama dua kali
    for text, label in zip(df['WaitingReason'], df['Unclear Reason'])
]

#DATALOADER UNTUK TRAINING
train_dataloader = DataLoader(train_examples, shuffle=True, batch_size=16)

#LOAD MODEL SBERT DAN DEFINISIKAN LOSS
model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')
train_loss = losses.SoftmaxLoss(
    model=model,
    sentence_embedding_dimension=model.get_sentence_embedding_dimension(),
    num_labels=3  # ada 3 kelas
)

# --- FINE-TUNING ---
model.fit(
    train_objectives=[(train_dataloader, train_loss)],
    epochs=7.2,
    warmup_steps=10,
    output_path='fine_tuned_sbert_model'
)
```

```
modules.json: 100% 229/229 [00:00<00:00, 21.3kB/s]
config_sentence_transformers.json: 100% 122/122 [00:00<00:00, 10.1kB/s]
README.md: 100% 3.89k/3.89k [00:00<00:00, 374kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 4.33kB/s]
config.json: 100% 645/645 [00:00<00:00, 43.3kB/s]
model.safetensors: 100% 471M/471M [00:05<00:00, 93.2MB/s]
tokenizer_config.json: 100% 480/480 [00:00<00:00, 49.8kB/s]
tokenizer.json: 100% 9.08M/9.08M [00:01<00:00, 6.90MB/s]
special_tokens_map.json: 100% 239/239 [00:00<00:00, 21.6kB/s]
config.json: 100% 190/190 [00:00<00:00, 13.2kB/s]
```



Gambar 3.55 Model *Sbert* dengan *fine tuning*

Pada Gambar 3.55 ditampilkan arsitektur *model SBERT* yang telah melalui proses *fine-tuning*. Proses *fine-tuning* ini bertujuan untuk mengoptimalkan performa *SBERT* dalam menghasilkan representasi vektor numerik dari teks, sambil tetap mempertahankan maknanya—khususnya untuk kalimat-kalimat pendek yang terdiri dari dua kata atau lebih seperti yang ditemukan pada data *Waiting Reason*. Hal ini menjadikan *SBERT* sangat cocok digunakan dalam konteks klasifikasi alasan menunggu yang diberikan oleh *engineer*. Langkah pertama dalam proses ini dimulai dengan membagi data ke dalam dua bagian, yaitu data pelatihan dan data validasi, menggunakan fungsi *train_test_split*. Kolom *WaitingReason* digunakan sebagai *input* teks, sedangkan label klasifikasi diambil dari kolom *Unclear Reason*, yang sudah dikategorikan menjadi tiga kelas utama: *Reasonable*, *In Progress*, dan *Unclear Reason*. Selanjutnya, data pelatihan diformat menggunakan *InputExample*, yaitu format khusus yang digunakan oleh *SBERT* saat melakukan *fine-tuning*. Karena tugas yang dijalankan adalah klasifikasi satu kalimat, setiap entri teks dipasangkan dengan dirinya sendiri, kemudian dihubungkan dengan label kelas yang sesuai.

Data yang telah diformat ini kemudian dimasukkan ke dalam *DataLoader*, dengan konfigurasi *shuffle* untuk mengacak urutan data selama pelatihan, serta menggunakan *batch size* sebesar 16 agar proses pelatihan dapat dilakukan dalam ukuran kelompok data yang seimbang. *Model SBERT* yang

digunakan adalah *"paraphrase-multilingual-MiniLM-L12-v2"*, yang mendukung banyak bahasa termasuk Bahasa Indonesia dan Inggris—dua bahasa utama yang muncul dalam data teks. *Model* ini kemudian di-*fine-tune* menggunakan fungsi *SoftmaxLoss*, karena tugas klasifikasi yang dilakukan bersifat *multi-class*. Proses pelatihan dijalankan selama 7.2 *epochs*, dengan tambahan 10 langkah *warm-up* untuk membantu stabilisasi proses pembelajaran pada awal pelatihan. Hasil dari *fine-tuning* ini adalah *model SBERT* yang telah terlatih secara khusus untuk memahami dan merepresentasikan teks *Waiting Reason*. *Model* tersebut kemudian disimpan dalam direktori *fine_tuned_sbert_model*, dan siap digunakan untuk proses *embedding* serta pemodelan tahap selanjutnya.

```
model_finetuned = SentenceTransformer('fine_tuned_sbert_model')

X = model_finetuned.encode(df['WaitingReason'].tolist(), convert_to_numpy=True)
y = df['Unclear Reason'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Gambar 3.56 *Train Test Split & Scaler*

Lalu dengan melalui *fine-tuning SBERT* dan sudah terbentuk dengan file *SBERT*, berikutnya pada Gambar 3.56 menunjukkan penerapan *model SBERT* hasil *fine-tuning* (*fine_tuned_sbert_model*) untuk mengubah data teks menjadi representasi numerik. Pada tahap ini, kolom *WaitingReason* digunakan sebagai data fitur (*x*), sedangkan kolom *Unclear Reason* digunakan sebagai label atau *target* (*y*). Data kemudian dibagi menggunakan fungsi *train_test_split*, dengan pembagian sebesar 70% untuk data pelatihan dan 30% untuk data pengujian. Parameter *random_state=42* digunakan agar proses pengacakan data dilakukan secara konsisten dan dapat direproduksi.

Sebelum melanjutkan ke tahap pelatihan *model machine learning*, dilakukan proses standarisasi menggunakan *StandardScaler*. Proses ini

bertujuan untuk menyamakan skala antar fitur dalam vektor hasil *embedding*, sehingga *model* dapat belajar secara lebih optimal dan terhindar dari *bias* yang mungkin muncul akibat perbedaan skala. Setelah semua tahapan ini selesai, data sudah siap digunakan untuk pelatihan model klasifikasi.

```python
# Daftar model
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "SVM": SVC(class_weight='balanced'),
    "Random Forest": RandomForestClassifier(n_estimators=100),
    "Gradient Boosting": GradientBoostingClassifier(),
    "Naive Bayes (Gaussian)": GaussianNB(),
    "Decision Tree": DecisionTreeClassifier(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "XGBoost": xgb.XGBClassifier(use_label_encoder=False, eval_metric='mlogloss'),
    "LightGBM": lgb.LGBMClassifier(),
    "Extra Trees": ExtraTreesClassifier()
}

labels = ['Unclear', 'Reasonabel', 'In Progress']

model_scores = {}

for name, model in models.items():
    print(f"\n🔹 Model: {name}")
    start_time = time.time()

    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"Training + Prediction time: {elapsed_time:.2f} seconds")
```

Gambar 3.57 Algoritma *Machine Learning*

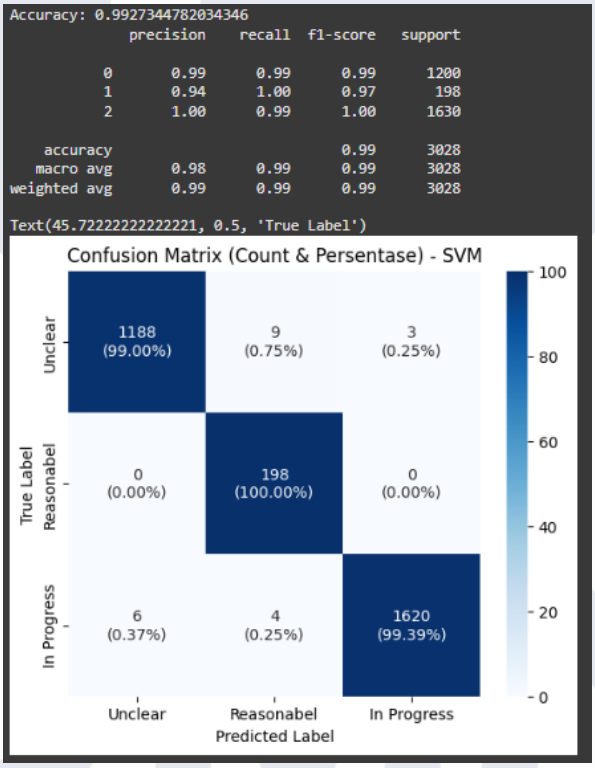
Selanjutnya pada Gambar 3.57 dilakukan pemilihan *model machine learning* yang digunakan untuk proses *klasifikasi teks*. Pada tahap ini, dilakukan *komparasi* terhadap 10 algoritma *machine learning* yang berbeda untuk mengetahui *model* mana yang memberikan performa terbaik dalam menangani data teks pada kolom *Waiting Reason*. Sepuluh *model* yang diuji adalah: *Support Vector Machine (SVM)*, *Naive Bayes (GaussianNB)*, *K-Nearest Neighbors (KNN)*, *XGBoost*, *Extra Trees*, *Random Forest*, *Logistic Regression*, *LightGBM*, *Gradient Boosting*, dan *Decision Tree* [15]–[17]. Pemilihan *model-model* ini didasarkan pada kebutuhan operasional yang menginginkan *model* yang tidak hanya akurat, tetapi juga ringan dan cepat dalam proses pelatihannya. Oleh karena itu, setiap *model* dievaluasi dari sisi akurasi, waktu pelatihan, serta kemampuannya dalam melakukan *prediksi* secara konsisten.

5. Evaluation

Tabel Evaluasi Model (Urut Berdasarkan Kinerja Terbaik):

	Model	Accuracy	Precision	Recall	F1 Score	Average Score	Elapsed Time (s)
0	SVM	0.992734	0.977172	0.994622	0.985567	0.987524	0.569483
1	Naive Bayes (Gaussian)	0.991744	0.974932	0.991050	0.982711	0.985109	0.030293
2	K-Nearest Neighbors	0.990423	0.979022	0.985502	0.982209	0.984289	0.661105
3	XGBoost	0.990753	0.976306	0.984447	0.980305	0.982953	7.080439
4	Extra Trees	0.990092	0.975755	0.984038	0.979825	0.982428	0.982182
5	Random Forest	0.989762	0.975616	0.983687	0.979580	0.982161	4.091433
6	Logistic Regression	0.990092	0.976852	0.981227	0.979020	0.981798	4.375345
7	LightGBM	0.990092	0.976850	0.981154	0.978983	0.981770	15.050167
8	Gradient Boosting	0.989762	0.974121	0.982355	0.978166	0.981101	180.555909
9	Decision Tree	0.985139	0.961226	0.973356	0.967133	0.971713	1.497553

```
Accuracy: 0.9927344782034346
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1200
           1       0.94      1.00      0.97       198
           2       1.00      0.99      1.00      1630

    accuracy                           0.99      3028
   macro avg       0.98      0.99      0.99      3028
weighted avg       0.99      0.99      0.99      3028

Text(45.7222222222221, 0.5, 'True Label')
```



Gambar 3.58 Komparasi Model & Penggunaan Model *SVM*

Tahap selanjutnya adalah *evaluation*, seperti yang ditunjukkan pada Gambar 3.58. Berdasarkan hasil *evaluasi* tersebut, algoritma yang memberikan performa terbaik adalah *Support Vector Machine (SVM)*, dengan rata-rata *akurasi* sebesar 98,75% dan waktu pelatihan hanya 0,5 detik. *SVM* menunjukkan kemampuan yang sangat baik dalam memahami pola teks, dan yang terpenting, tidak menunjukkan gejala *overfitting*. *Model* ini tetap konsisten

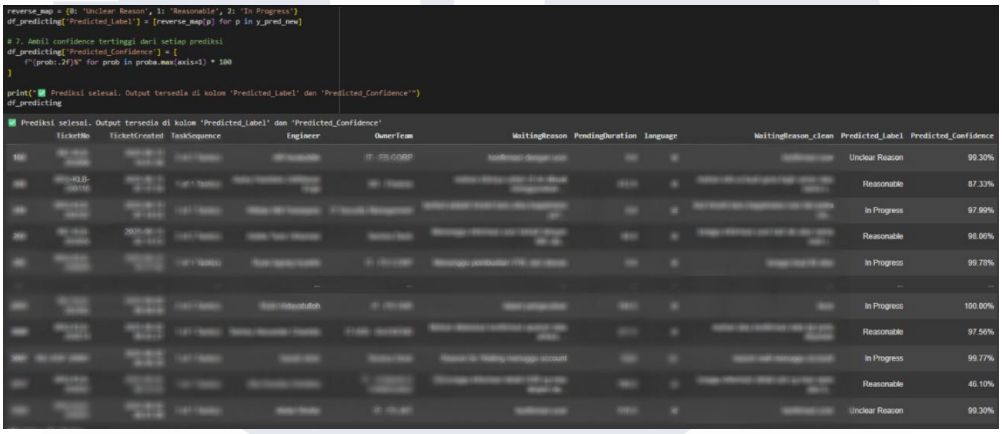
dan akurat, bahkan ketika digunakan untuk memprediksi data baru. Sebagai pembanding, algoritma lain seperti *Gaussian Naive Bayes* juga menghasilkan *akurasi* yang cukup tinggi, yaitu 98,51%, dan memiliki waktu pelatihan yang sangat cepat, hanya 0,03 detik. Namun, saat diuji dengan data tiket baru di bulan Juni, *model* ini mengalami kesalahan *klasifikasi*. Hal ini menunjukkan bahwa meskipun performa awalnya terlihat menjanjikan, *model* tersebut belum mampu menangkap konteks dan makna kalimat secara menyeluruh seperti yang dapat dilakukan oleh *SVM*.

Pada Gambar 3.56, *SVM* menunjukkan performa yang stabil dan akurat, termasuk saat diuji dengan data baru. *Evaluasi model* menggunakan beberapa *metrik*, yaitu *confusion matrix*, *accuracy score*, dan *classification report*. Dari *confusion matrix*, terlihat bahwa *model* berhasil mengklasifikasikan data dengan *akurasi prediksi* yang benar di atas 99%, menandakan kemampuan *generalisasi* yang baik. Keberhasilan ini sangat dipengaruhi oleh representasi teks dari *SBERT*, yang mampu menangkap makna semantik kalimat lebih baik dibandingkan metode tradisional. Hal ini membuat *model* lebih memahami konteks kalimat *waiting reason*, yang umumnya terdiri dari dua kata atau lebih. Hasil *classification report* menunjukkan rata-rata nilai *precision*, *recall*, dan *f1-score* sebesar 98%, mencerminkan performa *klasifikasi* yang sangat baik. Meski demikian, terdapat penurunan sedikit pada kelas *Reasonable*, kemungkinan akibat jumlah data yang lebih sedikit pada kelas tersebut. Secara keseluruhan, *SVM* terbukti unggul dan layak digunakan dalam operasional *IT Service Management (ITSM)* di PT Kalbe Farma Tbk sebagai model klasifikasi alasan menunggu tiket.

*6. **Deployment***

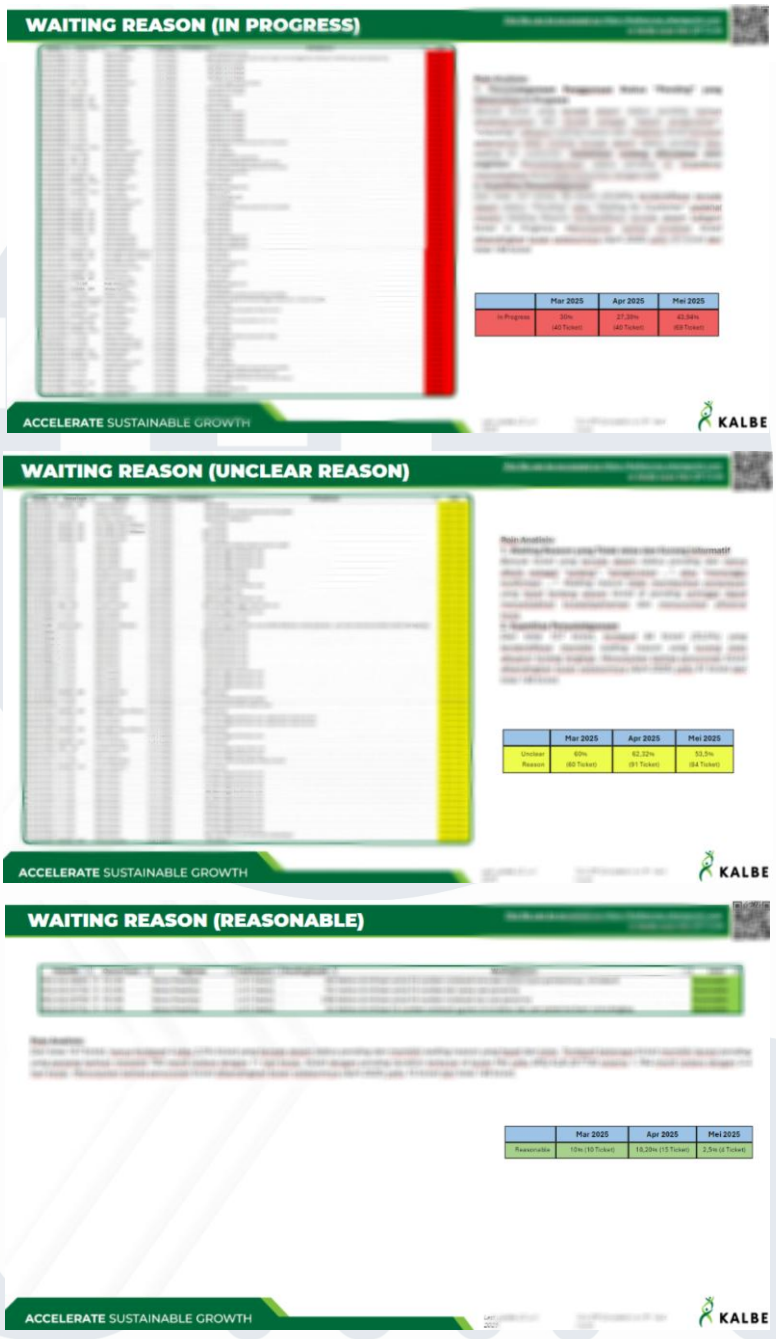
Setelah *model* dievaluasi menggunakan *metrik* seperti *confusion matrix*, *accuracy score*, dan *classification report*, serta dilakukan pemilihan *model* terbaik berdasarkan performa keseluruhan, tahap selanjutnya adalah *deployment*. Tahap ini merupakan proses implementasi *model* agar dapat

digunakan secara langsung untuk memprediksi data baru ke depannya. Dengan *model* yang sudah siap digunakan, proses *klasifikasi Waiting Reason* tidak lagi perlu dilakukan secara manual atau dilabeli satu per satu, sehingga dapat menghemat waktu dan meningkatkan efisiensi dalam operasional *IT Service Management*. *Model* yang telah dilatih dan diuji kini dapat menjadi bagian dari sistem otomatis untuk membantu tim IT dalam memonitor dan memahami alasan penundaan tiket secara lebih cepat dan konsisten.



Gambar 3.59 Implementasi Model *SVM* dengan *SBert.*

Gambar 3.59 menunjukkan hasil dari implementasi *model* SVM yang telah dilatih menggunakan representasi teks dari *SBERT fine-tuning*. Data yang digunakan pada tahap ini berasal dari tiket bulan Juni, sebagai bagian dari uji coba penerapan *model* pada data baru di dunia nyata. Hasil *prediksi* yang ditampilkan mencakup nilai *confidence* atau tingkat keyakinan *model* terhadap *prediksi* yang diberikan. Secara keseluruhan, *model* menunjukkan performa yang sangat baik, dengan mayoritas *prediksi* memiliki tingkat kepercayaan di atas 90%. Hal ini mengindikasikan bahwa *model* mampu memahami dan mengklasifikasikan teks *Waiting Reason* secara akurat dan konsisten, serta siap digunakan dalam proses operasional untuk menggantikan pelabelan manual yang selama ini memakan waktu.

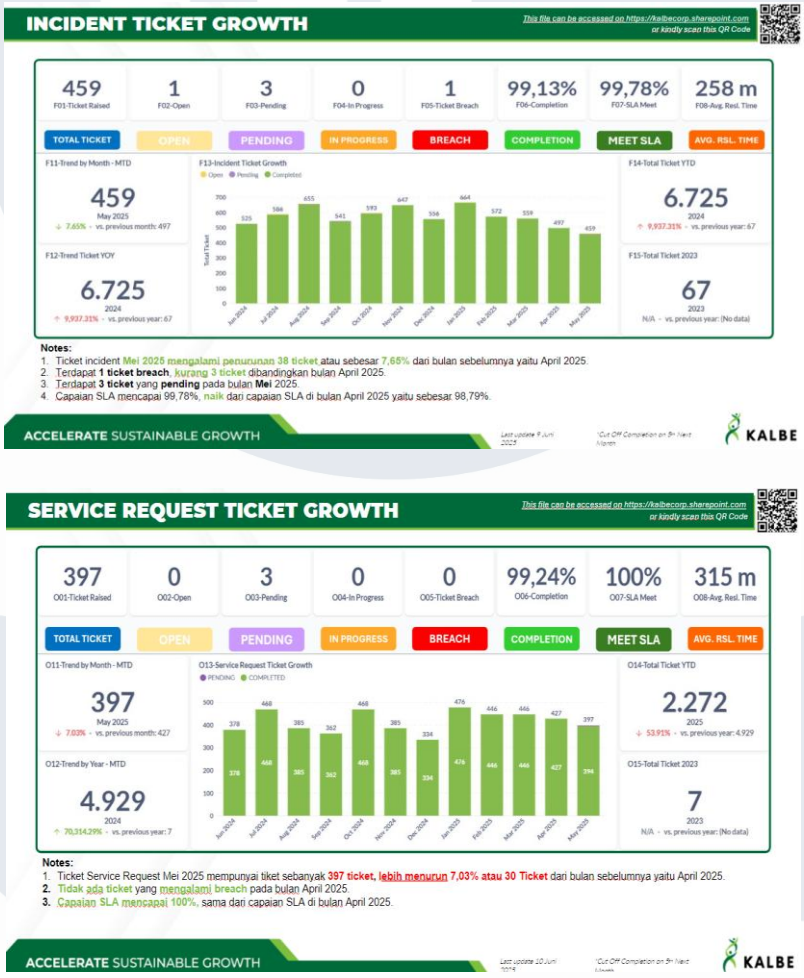


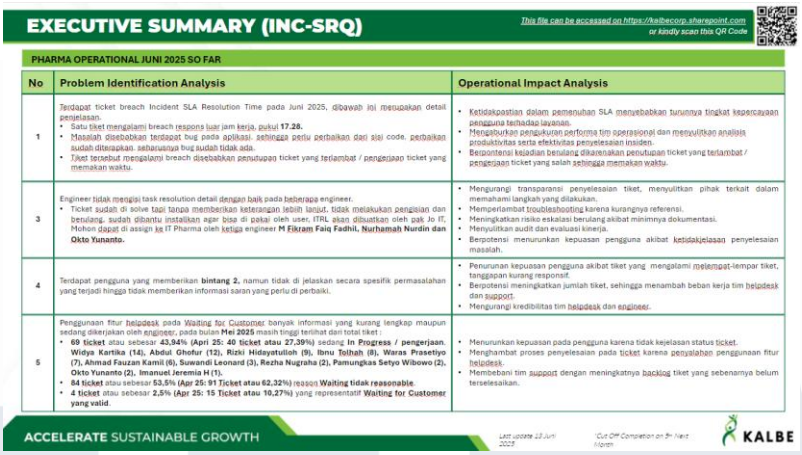
Gambar 3.60 Implementasi pada Operational *May* 2025

Dari hasil *model* yang telah dibangun, tahap selanjutnya adalah penerapan pada proses operasional, seperti yang ditunjukkan pada Gambar 3.60. Dengan adanya *model* ini, proses pelabelan *"Waiting Reason"* tidak lagi perlu dilakukan secara manual satu per satu. Hal ini membawa dampak signifikan terhadap efisiensi waktu kerja. Jika sebelumnya proses pelabelan

dapat memakan waktu antara 1 hingga 2 jam, kini hanya membutuhkan waktu sekitar 1 menit untuk mendapatkan hasil *klasifikasi* secara otomatis. Data hasil *prediksi* pun langsung dapat digunakan untuk keperluan presentasi dan analisis pada laporan operasional bulanan, sehingga mempercepat alur kerja dan mendukung pengambilan keputusan yang lebih cepat dan akurat.

3.2.2.16 Operational Pharma Report Mei 2025





EXECUTIVE SUMMARY (INC-SRQ)

This file can be accessed on https://kalbecorp.sharepoint.com or kindly scan this QR Code

PHARMA OPERATIONAL JUNI 2025 SO FAR

No	Problem Identification Analysis	Operational Impact Analysis
1	Terdapat ticket breach Incident SLA Resolution Time pada Juni 2025, dibawah ini merupakan detail penjelasan. • Satu tiket mengalami breach respons luar jam kerja, pukul 17.28. • Masalah disebabkan terdapat bug pada aplikasi, sehingga perlu perbaikan dari sisi code, perbaikan sudah diterapkan, sehausnya bug sudah tidak ada. • Tiket tersebut mengalami breach disebabkan penutupan ticket yang terlambat / pengerjaan ticket yang memakan waktu.	• Ketidakpastian dalam pemenuhan SLA menyebabkan turunnya tingkat kepercayaan pengguna terhadap layanan. • Mengaburkan pengukuran performa tim operasional dan menyulitkan analisis produktivitas serta efektivitas penyelesaian insiden. • Berpotensi kejadian berulang dikarenakan penutupan ticket yang terlambat / pengerjaan ticket yang salah sehingga memakan waktu.
3	Engineer tidak mengisi task resolution detail dengan baik pada beberapa engineer. • Ticket sudah di solve tapi tanpa memberikan keterangan lebih lanjut, tidak melakukan pengisian dan pending, sudah dibantu installkan agar bisa di pakai oleh user, ITRL akan dibuatkan oleh pak Jo IT. Mohon dapat di assign ke IT Pharma oleh ketiga engineer **M Fikram Faiq Fadhli, Nurhamah Nurdin dan Okto Yunanto.**	• Mengurangi transparansi penyelesaian tiket, menyulitkan pihak terkait dalam memahami langkah yang dilakukan. • Memperlambat troubleshooting karena kurangnya referensi. • Meningkatkan risiko eskalasi berulang akibat minimnya dokumentasi. • Menyulitkan audit dan evaluasi kinerja. • Berpotensi menurunkan kepuasan pengguna akibat ketidakjelasan penyelesaian masalah.
4	Terdapat pengguna yang memberikan **bintang 2,** namun tidak di jelaskan secara spesifik permasalahan yang terjadi hingga tidak memberikan informasi saran yang perlu di perbaiki.	• Penurunan kepuasan pengguna akibat tiket yang mengalami melempat-lempar tiket, tanggapan kurang responsif. • Berpotensi meningkatkan jumlah tiket, sehingga menambah beban kerja tim helpdesk dan support. • Mengurangi kredibilitas tim helpdesk dan engineer.
5	Penggunaan fitur helpdesk pada Waiting for Customer banyak informasi yang kurang lengkap maupun sedang dikerjakan oleh engineer, pada bulan **Mei 2025** masih tinggi terlihat dari total tiket : • **69 ticket** atau sebesar **43,94% (Apri 25: 40 ticket atau 27,20%)** sedang **In Progress / pengerjaan. Widya Kartika (14), Abdul Ghofur (12), Rizki Hidayatulloh (9), Ibnu Tolhah (8), Waras Prasetiyo (7), Ahmad Fauzan Kamil (6), Suwandi Leonard (3), Rezha Nugraha (2), Pamungkas Setyo Wibowo (2), Okto Yunanto (2), Imanuel Jeremia H (1).** • **64 ticket** atau sebesar **53,5% (Apr 25: 91 Ticket atau 62,32%)** reason **Waiting tidak reasonable.** • **4 ticket** atau sebesar **2,5% (Apr 25: 15 Ticket atau 10,27%)** yang representatif **Waiting for Customer yang valid.**	• Menurunkan kepuasan pada pengguna karena tidak kejelasan status ticket. • Menghambat proses penyelesaian pada ticket karena penyalahan penggunaan fitur helpdesk. • Membebani tim support dengan meningkatnya backlog tiket yang sebenarnya belum terselesaikan.

ACCELERATE SUSTAINABLE GROWTH

Last update 13 Juni 2025

Cut Off Completion on 5th Next Month

KALBE

Gambar 3.61 Operational Pharma *Report* Bulan Mei 2025

Berdasarkan hasil laporan operasional bulan Mei 2025, jumlah tiket pada kategori *Incident* kembali mengalami penurunan sebanyak 38 tiket dibandingkan bulan sebelumnya. Penurunan ini merupakan sinyal positif, karena semakin sedikit jumlah tiket *Incident* menandakan bahwa potensi gangguan atau permasalahan pada layanan TI cenderung menurun. Namun, kondisi berbeda terjadi pada tiket *Service Request*, yang juga mengalami penurunan sebesar 30 tiket. Tidak seperti *Incident*, penurunan jumlah *Service Request* justru mengindikasikan hal yang kurang baik. Hal ini dapat mencerminkan menurunnya tingkat pemanfaatan layanan TI oleh *user*, yang bisa disebabkan oleh berkurangnya kebutuhan layanan, kurangnya *awareness* terhadap fitur layanan yang tersedia, atau adanya kendala dalam proses pengajuan permintaan. Dalam ringkasan pelaksanaan layanan bulan ini, tercatat adanya pelanggaran terhadap *SLA Resolution Time*, di mana satu tiket melewati batas waktu penyelesaian karena ditutup terlalu lama dan berada di luar jam kerja. Kondisi ini dapat terjadi akibat proses pengerjaan yang memakan waktu terlalu lama atau kelalaian dalam melakukan penutupan tiket secara tepat waktu. Selain itu, masih ditemukan beberapa *engineer* yang belum mengisi bagian *Task Resolution Detail* secara lengkap, atau bahkan tidak mengisi sama sekali. Padahal, informasi dalam bagian ini sangat penting untuk mendokumentasikan solusi yang diberikan secara transparan dan akuntabel.

Pada bulan Mei ini juga mulai diimplementasikan model prediksi untuk mengevaluasi alasan yang dituliskan oleh *engineer* saat menggunakan status *Waiting for Customer*. Hasil dari implementasi awal menunjukkan bahwa masih banyak *engineer* yang memberikan alasan dengan pernyataan yang kurang jelas atau ambigu. Bahkan, beberapa tiket ternyata masih dalam proses pengerjaan, namun tetap diberi status menunggu pelanggan (*Waiting for Customer*). Hal ini menunjukkan bahwa masih diperlukan pemahaman yang lebih baik terkait penggunaan status tersebut, agar digunakan secara akurat dan tidak disalahartikan. Secara keseluruhan, meskipun terjadi peningkatan dari sisi jumlah tiket yang lebih rendah, laporan bulan Mei ini menegaskan perlunya peningkatan kedisiplinan dalam dokumentasi, serta penguatan dalam penggunaan fitur sistem secara tepat. Evaluasi terhadap implementasi model prediksi juga menjadi landasan penting dalam peningkatan operasional ke depannya, khususnya untuk memastikan validasi status yang diinput oleh *engineer* dapat dilakukan secara lebih objektif dan efisien.

3.3 Kendala yang Ditemukan

Proses pelaksanaan magang di PT Kalbe Farma, Tbk dengan peran sebagai Data Analyst berjalan dengan cukup baik secara keseluruhan. Meskipun demikian, selama pelaksanaannya masih ditemukan beberapa kendala yang menjadi tantangan dalam menjalankan tugas dan tanggung jawab sebagai seorang Data Analyst:

1. Mahasiswa menghadapi tantangan dalam proses penarikan data melalui *API*, khususnya dalam memahami mekanisme autentikasi token, struktur endpoint, serta cara penggunaannya menggunakan bahasa pemrograman Python. Hal ini memerlukan waktu dan pembelajaran tambahan di awal pelaksanaan magang.
2. Mahasiswa memerlukan adaptasi terhadap budaya kerja, sistem yang digunakan, dan prosedur pada perusahaan, hal ini dapat menurunkan produktivitas pada awal pelaksanaan magang.

3. Tugas yang dikerjakan oleh magang diluar konteks Data Analyst, seperti melakukan pengeditan video, dan melakukan otomisasi pada penggunnan power automate.
4. Kurangnya pemahaman awal mengenai konsep dan modul dalam *IT Service Management (ITSM)* juga menjadi hambatan. Sehingga membutuhkan waktu lebih lama untuk memahami struktur dan proses dalam *ITSM*, sebelum dapat mengerjakan analisis dan pengolahan data terkait modul tersebut secara optimal.

3.4 Solusi atas Kendala yang Ditemukan

Dari beberapa kendala yang ditemukan saat pelaksanaan proses kerja magang, berikut ini merupakan solusi dalam mengatasi beberapa kendala tersebut:

1. Pelatihan dari mentor, penjelasan dari tim, serta pencarian mandiri melalui dokumentasi *API* sangat membantu dalam mempercepat proses pemahaman terhadap konsep *API*, terutama dalam mengenal komponen penting seperti *client secret*, *token ID*, dan alur autentikasi. Melalui proses ini, mahasiswa juga dapat memahami berbagai jenis error yang umum terjadi saat penarikan data—seperti *error 400 (bad request)*—sehingga mampu melakukan *troubleshooting* secara lebih cepat dan efektif.
2. Proses adaptasi dengan lingkungan kerja dilakukan dengan membangun komunikasi yang baik dengan tim, mencari informasi secara terkait sistem yang digunakan, serta memahami prosedur dan peraturan yang berlaku di perusahaan agar setiap pekerjaan dapat dilakukan dengan sesuai.
3. Mahasiswa melakukan pencarian mandiri terkait *tools* baru yang digunakan di luar konteks utama *Data Analyst*, dengan memanfaatkan berbagai platform untuk memahami cara kerja dan penerapannya pada *tools* yang baru secara lebih efektif.
4. Aktif berinteraksi dengan tim dan mentor untuk memahami modul-modul yang ada dalam *IT Service Management (ITSM),* serta melakukan pencarian informasi secara mandiri melalui berbagai platform. Hal ini dilakukan untuk

memperdalam pemahaman terhadap model dan alur kerja dalam *ITSM* secara menyeluruh.