

BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Proses pelaksanaan praktik kerja magang yang dilakukan oleh peserta magang berperan dan memiliki kedudukan internship dalam functional unit Corporate Human Resources Kompas Gramedia Group. Selama menjalani praktik kerja magang peserta dibimbing secara langsung oleh tim People Data Analyst yaitu bersama Kak Irsa Salsabilah yang memiliki jabatan sebagai Freelance People Data Analyst, Kak Dahayu Bethari Widyandri yang memiliki jabatan sebagai Freelance People Data Analyst, kemudian Arthur sebagai internship people data analyst serta Supervisi dalam team ini ialah Mas Bramanto Ranggamukti sebagai seorang Lead People Data Analyst yang tentunya memiliki peran penting dalam struktur Kompas Gramedia Group. Pada people data analyst tidak hanya mencakup freelance, internship dan lead saja melainkan dipimpin langsung oleh CEO yaitu mas Arki Sudito yang dimana memiliki peran dalam mengarahkan dan mengelola seluruh kegiatan di dalam Growth Center. Untuk lebih detail terkait dengan struktur organisasi people data analyst terlihat pada Gambar 3.1 Struktur organisasi people data analyst.



Gambar 3.1 Struktur Organisasi People Data Analyst

Dalam struktur organisasi diatas juga memperlihatkan peran peserta magang sebagai internship people data analyst yang berkaitan erat dengan pekerjaan yang berfokus utama pada teknologi sistem informasi. Functional unit Corporate Human Resources, khususnya pada departement People Data analyst di Growth Center Kompas Gramedia Group memiliki tanggung jawab dalam bekerja sama dengan berbagai unit seperti tim developmen, tim engagement, tim discovery dan juga tim activation. People Data Analyst berperan dalam mengimplementasikan seluruh ilmu data yang nantinya dapat diolah kembali dengan memperoleh saran atau informasi kedepannya untuk dapat meningkatkan kembali kinerja produktivitas pada keseluruhan bisnis.

People Data Analyst juga bertugas dalam melakukan dan menjalankan berbagai project yang berfokus utama pada data sehingga mampu menghasilkan wawasan ataupun informasi untuk pengambilan Keputusan yang lebih baik kedepannya. Salah satu tugas utama sebagai seorang People Data Analyst ialah mampu merancang, membangun, serta mengimplementasikan visualisasi data pada sebuah dashboard pada platform Growth Center contohnya seperti membentuk sebuah diagram atau grafik yang nantinya mampu memberikan sebuah gambaran atau pemahaman yang lebih baik terkait dengan pola atau informasi yang terjadi dalam data tersebut. Jika mengalami kendala dalam report ataupun request data maka dapat dikoordinasikan kembali secara langsung dengan unit People Data Analyst untuk dilakukan perbaikan dan pemeliharaan data kembali.

Proses pratik kerja magang yang dilakukan oleh peserta magang bertanggung jawab dalam beberapa project dan request data dari unit lainnya. Seluruh proses project dan data request memiliki alur kerja yang dimulai dari disampaikan oleh Lead People Data Analyst yaitu Mas Bramanto kepada tim freelance People Data Analyst untuk menangani setiap permasalahan project, pembuatan project hingga data request. Selanjutnya setelah tim freelance menerima tugas tersebut nantinya akan ditugaskan kepada masing – masing dari tim People

Data Analyst dengan melalui meeting Weekly Team Data untuk menentukan PIC ataupun siapa yang akan menghandle tugas atau pekerjaan tersebut.

Koordinasi yang dilakukan dalam tim People Data Analyst yaitu dengan menggunakan Google Meet sebagai platform utama untuk mengadakan meeting mingguan dan melakukan discuss terkait pekerjaan yang ingin disampaikan serta dengan menggunakan Google Meet ini mampu mendukung kelancaran dalam proses bisnis. Sedangkan untuk berkoordinasi dengan tim development, tim engagement, tim discovery dan juga tim activation dapat melalui platform WhatsApp. Berikut ini jadwal koordinasi dan komunikasi yang dilakukan dalam functional unit Corporate Human Resources pada perusahaan Kompas Gramedia Group terlihat pada tabel 3.1 Koordinasi dan Komunikasi Kompas Gramedia Group.

Tabel 3.1 Koordinasi dan Komunikasi Kompas Gramedia Group

Koordinasi	Jadwal	Deskripsi
Weekly Meeting All HR Expertise, SBP and Intern	Setiap Senin, Pukul 10:00 s/d 11:00	a. Check in, b. Discussion on New Weekly Handle.
Weekly Meeting People Data Analyst	Setiap Senin, Pukul 14:00 s/d 16:00	a. Update pekerjaan yang telah dilakukan, b. Pembagian tugas untuk project atau request data yang masuk, c. Update seluruh kegiatan melalui platform Asana.

Meeting Team People Data Analyst	Sesuai kebutuhan dari team People Data Analyst	Melakukan meeting terkait project yang akan dating seperti dashboard satisfaction streamlit.
-------------------------------------	--	---

3.1.1 Weekly Meeting All HR Expertise, SBP and Intern

Pada functional unit Corporate Human Resources melakukan kegiatan meeting mingguan bersama dengan seluruh tim dari naungan Corporate Human Resources yang dimana Weekly meeting All HR Expertise dilakukan setiap hari Senin pada pukul 10:00 WIB sampai dengan 11:00 WIB dengan melalui platform Zoom Meeting. Meeting weekly All HR Expertise melakukan pembahasan mengenai Discussion on New Weekly handle yang dimana dalam diskusi meeting ini melakukan pembahasan mengenai kegiatan yang telah dikerjakan oleh setiap unit dalam Growth Center serta melakukan updatean mengenai kegiatan atau aktivitas apa yang akan dilakukan di minggu tersebut. Pada Gambar 3.2 memperlihatkan jadwal Weekly Meeting All HR.



Gambar 3.2 Jadwal Weekly Meeting All HR Expertise x SBP X All

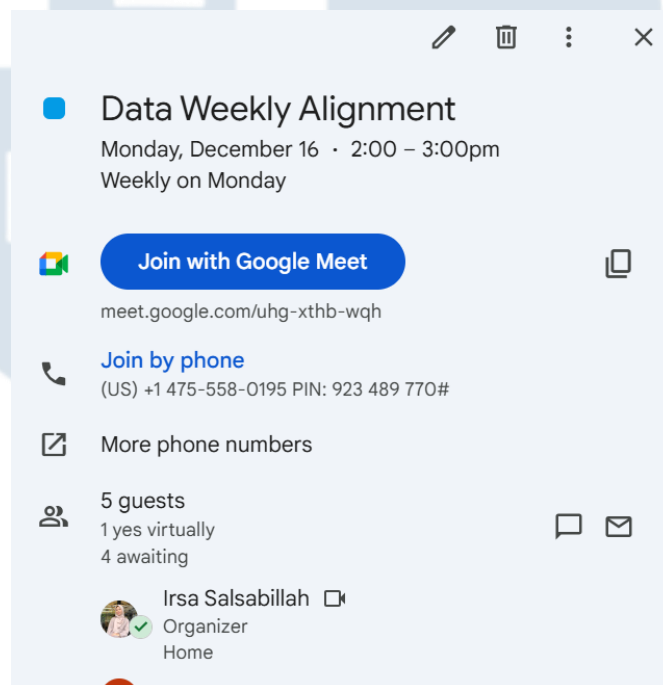
Weekly update juga menjelaskan mengenai project yang akan dilaksanakan kedepannya seperti Growth Conversation yang dimana project ini di handle oleh Kak Andini Nabila Sari yang dimana Growth Conversation ini akan dilakukan selama 3 kuartal dan digunakan sebagai bentuk fasilitas pertumbuhan bagi seluruh karyawan Growth Center baik freelance maupun internship yang dimana dapat diharapkan mampu membantu seluruh karyawan dapat menemukan the best version of ourselves melalui sesi 1 on 1 dan juga dapat menciptakan 360 derajat feedback loop antara karyawan dengan streamlead, kemudian streamlead dengan manajemen.

Sebelumnya kegiatan ini telah dilakukan sehingga hanya akan melanjutkan kuartal ke 3 untuk dapat memaksimalkan tracking, maintaining dan improving performance karyawan sehingga berguna untuk menyelaraskan dengan pertumbuhan organisasi, memberikan support dan arahan untuk karyawan mencapai personal dan professional goals dan skills serta mampu meningkatkan hubungan interpersonal baik itu antara mentor dan karyawan untuk dapat memberikan pengalaman kerja yang lebih baik. Pada weekly meeting all HR ini juga memberikan penjelasan mengenai pelaksanaan Growth Conversation, timeline kegiatan hingga detail dari agenda aktivitas tersebut tetapi tidak hanya project ini saja setiap minggunya akan ada aktivitas ataupun project baru yang nantinya akan dijelaskan dalam Weekly meeting All HR Expertise.

3.1.2 Weekly Meeting People Data Analyst

Pada weekly meeting People Data Analyst yang dimana dilaksanakan pada setiap minggunya yaitu di hari Senin dan dapat digantikan menjadi hari berikutnya jika terdapat hari libur dan seringkali dimulai pada pukul 14:00 sampai dengan pukul 16:00. Meeting weekly

team data ini dilakukan dengan melalui platform Google Gmeet yang sebelumnya telah di set setiap minggunya melalui Google Calendar yang dilakukan oleh Kak Irsa Salsabillah sebagai organizer. Weekly meeting team data ini juga diikuti oleh seluruh karyawan team data mencakup mbak dahayu, kak irsa, Arthur dan peserta magang sebagai Internship Kompas Gramedia Group. Pada Gambar 3.3 memperlihatkan jadwal weekly meeting team People Data Analyst.



Gambar 3.3 Jadwal Weekly Meeting Team Data

Weekly meeting ini tentunya dilakukan setiap minggunya guna untuk mendiskusikan bersama terkait dengan apa saja yang telah dikerjakan pada minggu sebelumnya melakukan pembagian tugas terkait project yang akan dijalankan selanjutnya dan request data dari tim lainnya. Untuk dapat mengetahui progress pekerjaan tersebut tentunya dalam weekly meeting ini dilakukan update dengan melalui platform Asana sebagai salah satu platform penunjang koordinasi pekerjaan dalam team

data yang dimana dalam pembagian pekerjaan ini nantinya akan ditentukan siapa yang akan handle atau menjadi PIC dalam project tersebut. Serta dalam weekly meeting ini akan melakukan tugas rutin yaitu melakukan pembuatan report monthly baik itu assessment maupun all platform yang nantinya dari seluruh report tersebut akan dilakukan presentasi kepada seluruh karyawan Growth Center.

3.2 Tugas dan Uraian Kerja Magang

Pratik kerja magang yang dilaksanakan oleh peserta magang pada perusahaan Kompas Gramedia Group berlangsung selama 6 bulan yang dimulai pada periode 1 Juli 2024 sampai dengan 31 Januari 2025 dengan melibatkan peserta magang sebagaimana yang telah dicantumkan dalam Letter of Acceptance yaitu sebagai seorang Internship People Data Analyst. Peserta magang juga terlibat secara langsung dalam berbagai pekerjaan sebagai seorang people data analyst yang dimana mencakup sesi onboarding, weekly meeting HR Expertise X SBP X Intern, Weekly meeting people data analyst yang dimana telah menjalankan tugas sesuai dengan job description yang telah diberikan oleh perusahaan Kompas Gramedia Group yang dimana tugas seorang People Data Analyst mencakup:

- a. Melakukan compile dan cleaning data
- b. Merekomendasikan dan menerapkan tata Kelola data untuk data yang disimpan di platform Growth Center
- c. Mencatat kebutuhan analisis pengguna dan buat dashboard yang sesuai dengan berbagai platform (Looker Studio/Metabase/Tableau)
- d. Memastikan user memahami cara menggunakan dashboard dan menyiapkan dashboard untuk dibagikan kepada stakeholder di bawah supervisi PIC Data (contoh dengan tableau online); dan
- e. Merekomendasikan cara untuk meningkatkan keandalan, efisiensi dan kualitas data.

Sehingga, seluruh job description sebagai seorang Internship People Data Analyst peserta magang memiliki peran dalam mengelola seluruh data pada seluruh platform perusahaan Kompas Gramedia Group seperti Kognisi.id, Kognisi.mykg, Discovery dan Capture sehingga dari keseluruhan data tersebut dapat diolah sehingga mampu menjadi sebuah informasi yang akurat dan tepat serta nantinya dapat membantu stakeholder dalam membuat Keputusan dan rekomendasi yang tepat untuk perkembangan perusahaan. Pada perusahaan Kompas Gramedia tools yang digunakan sebagai penunjang pekerjaan mencakup Tableau Online, Dbeaver, Ms. Excel, Ms. Power Point, Asana dan Streamlit. Berikut penjelasan terkait uraian pekerjaan yang dilakukan peserta magang dalam merancang sebuah dashboard streamlit discovery terlihat pada tabel 3.2 Uraian Aktivitas Pratik Kerja Magang pada perusahaan Kompas Gramedia Group sebagai seorang Internship People Data Analyst yaitu:

Tabel 3.2 Uraian Aktivitas Pratik Kerja Magang

No	Job	Deskripsi Pekerjaan
1.	Connect database, data Capture dan SAP	Melakukan connect database pada streamlit terhadap data – data yang diperlukan untuk membuat dashboard Discovery.
2.	Page user growth	Membuat halaman user growth yang berisikan jumlah user yang mengakses Discovery dan Capture secara lifetime, his year dan this month.
3.	Page demography	Membuat halaman demography yang berisikan demography user yang mengakses platform Discovery dan Capture.
4.	Page result traits	Membuat halaman result traits yang berisikan distribusi jumlah seluruh user Kompas

		Gramedia group yang mengakses setiap platform
5.	Page internal KG	Membuat halaman internal KG yang berisikan test taker per unit serta distribusinya dalam unit, gender, generation dan layer.
6.	Page layer traits	Membuat halaman layer traits yang berisikan karyawan KG layer traits summary pada platform Discovery.

Timeline pekerjaan yang dilaksanakan selama praktik kerja magang sebagai seorang people data analyst dalam membuat project dashboard streamlit Discovery selengkapnya dapat dilihat pada tabel 3.3 Uraian Aktivitas dan Output Project.

Tabel 3.3 Uraian Aktivitas dan Output Project

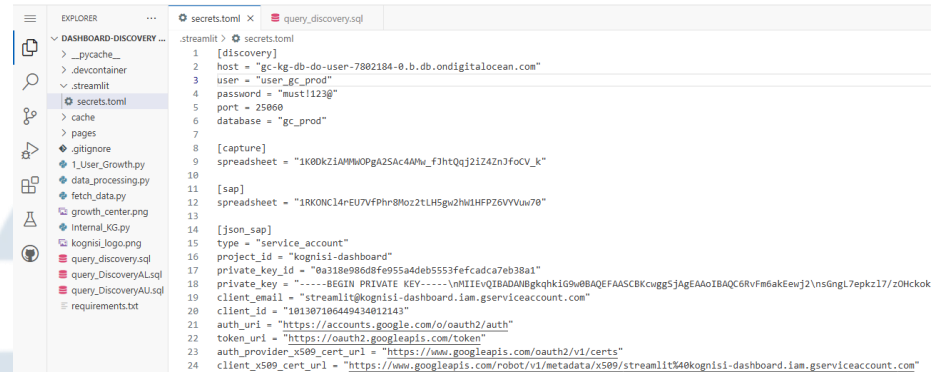
No	Job	Minggu Ke-	Tanggal Mulai	Tanggal Selesai
1.	Connect data	1 - 2	01 Juli 2024	12 Juli 2024
2.	Page user growth	3 - 4	15 Juli 2024	27 Juli 2024
3.	Page demography	5 – 7	29 Juli 2024	09 Agustus 2024
4.	Page result traits	8 - 10	12 Agustus 2024	30 Agustus 2024
5.	Page internal KG	11 – 14	02 September 2024	30 September 2024
6.	Page layer traits	15 – 19	01 Oktober 2024	01 November 2024

Tugas yang dilakukan selama proses praktik kerja magang yaitu merancang sebuah dashboard streamlit platform Discovery dan Capture agar nantinya dapat diakses oleh stakeholder secara real time. Berikut ini adalah penjelasan untuk setiap bagian yang dilakukan dalam perancangan dashboard tsreamlit discovery.

3.2.1 Connect Data

Peserta magang yang telah bergabung pada perusahaan Kompas Gramedia Group sebagai salah satu team people data analyst pada pertengahan tahun yaitu di bulan 1 Juli 2024 maka peserta magang membuat sebuah perancangan dashboard dengan menggunakan streamlit yang dimana hal ini dilakukan karena setelah peserta magang mengamati seluruh platform pembelajaran yang ada pada perusahaan Kompas Gramedia Group ternyata pada platform Discovery dan Capture sering kali dilakukan analisis secara manual dengan membuat report perbulan dan stakeholder kadangkala melakukan request terkait kebutuhan yang diinginkan. Sehingga dari kendala itu peserta magang memberikan sebuah ide dengan melakukan perancangan dashboard streamlit hingga seluruh kebutuhan stakeholder dapat dijangkau dan diakses secara real time pada dashboard streamlit discovery.

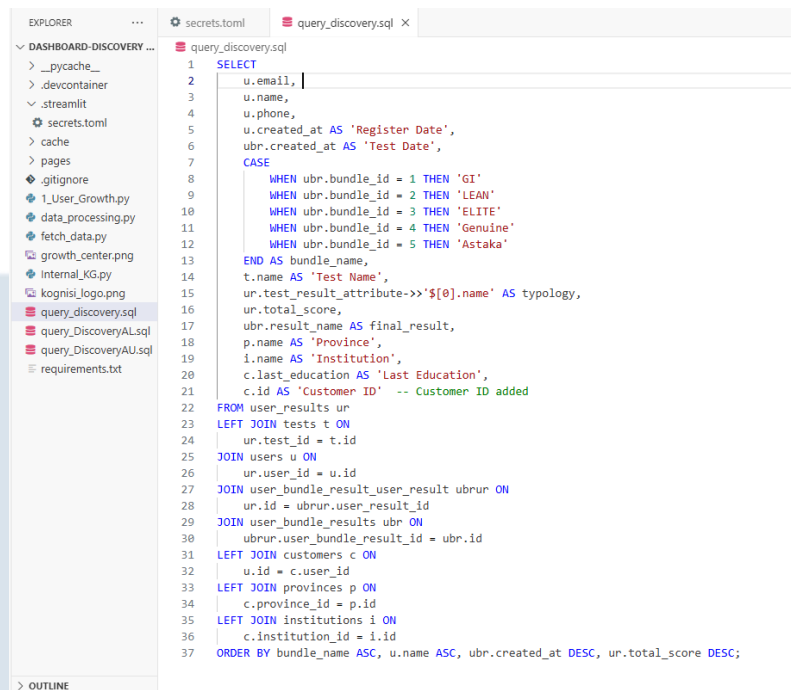
Pada minggu ke – 1 dan minggu ke – 2 peserta magang tentunya melakukan analisis terlebih dahulu terkait kebutuhan – kebutuhan yang diperlukan stakeholder kemudian dilakukan perancangan kasar terkait dengan tampilan dashboard setelah hal tersebut dilakukan maka selanjutnya peserta magang akan melakukan connect data terlebih dahulu pada dashboard streamlit untuk connect data sendiri disini peserta magang menggunakan streamlit dan melakukan penulisan log codingan pada codespace yang difasilitasi oleh streamlit. Connect data ini tentunya harus dilakukan secara hati – hati karena hal ini mencakup 3 sumber data yaitu database discovery, capture dan sap.



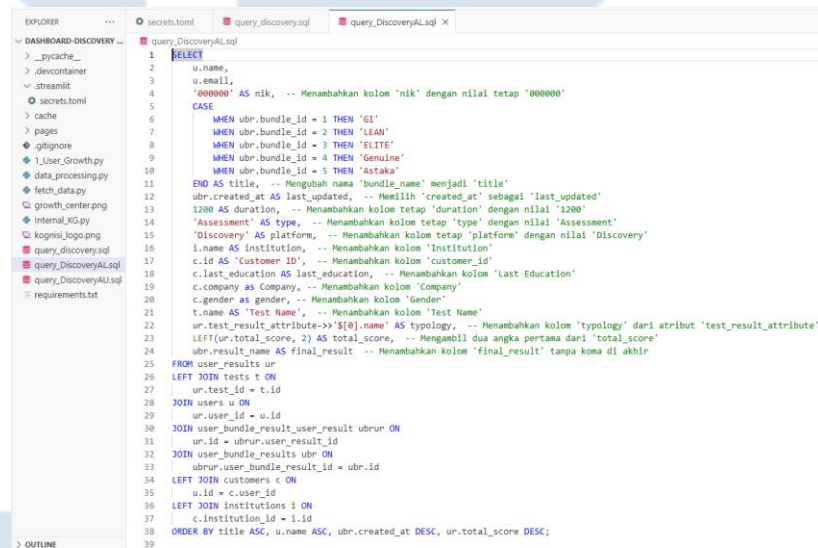
Gambar 3.4 Syntax Connect Data

Maka dari itu untuk connect data sendiri dibuat pada secrets.toml yang terlihat pada Gambar 3.4 Syntax Connect Data. Penggunaan secrets.toml ini yang dimana merupakan file konfigurasi yang digunakan pada aplikasi streamlit untuk dapat menyimpan data sensitive seperti API keys, password, atau variabel lainnya. File ini mampu menjaga keamanan data dengan memastikan bahwa informasi sensitive tidak dicantumkan langsung di kode sumber terutama jika repositori kode diunggah ke platform public seperti GitHub.

Setelah melakukan connect data tentunya akan dilakukan pemanggilan data yang diperlukan untuk membuat dashboard streamlit untuk pemanggilan data sendiri menggunakan syntax query yang dimana menggunakan 3 syntax query yaitu Query Discovery, Query Discovery AL, dan Query Discovery AU untuk syntaxnya terlihat pada Gambar berikut ini.



Gambar 3.5 Syntax Query Discovery



Gambar 3.6 Syntax Query Discovery AL

```

1 SELECT
2   u.name,
3   u.email,
4   u.created_at AS 'created_at',
5   'Discovery' AS platform, -- Menambahkan kolom platform dengan nilai 'Discovery'
6   CASE
7     WHEN ubr.created_at IS NOT NULL THEN 'Active' -- Jika terdapat Test Date maka status Active
8     ELSE 'Passive' -- Jika tidak ada Test Date maka status Passive
9   END AS learner_status
10 FROM user_results ur
11 JOIN users u ON
12   ur.user_id = u.id
13 LEFT JOIN user_bundle_result_user_result ubr ON
14   ur.id = ubrur.user_result_id
15 LEFT JOIN user_bundle_results ubr ON
16   ubrur.user_bundle_result_id = ubr.id
17 ORDER BY u.name ASC, u.created_at DESC;
18

```

Gambar 3.7 Syntax Query Discovery AU

Setelah seluruh proses pemanggilan query dilakukan maka selanjutnya yaitu melakukan fetch data yang dimana merupakan proses pengambilan data dari suatu sumber seperti database, API ataupun file query yang telah dibuat sebelumnya. Hal ini biasanya melibatkan pengambilan data dari lokasi penyimpanan eksternal ke aplikasi agar data tersebut dapat diolah atau ditampilkan. Berikut syntax fetch data yang dilakukan pada pembuatan dashboard streamlit discovery terlihat pada Gambar 3.8 Syntax Fetch Data.

```

1 import streamlit as st
2 import pandas as pd
3 import pymysql
4 from sshunnel import SSHunnelForwarder
5 import paramiko
6 from io import StringIO
7 import gspread
8 from oauth2client.service_account import ServiceAccountCredentials
9 import tomli
10
11
12 # Function to connect to Discovery and fetch data for the main query
13 @st.cache_resource
14 def fetch_data_discovery():
15     return fetch_data_from_query('query_discovery.sql')
16
17
18 # Function to connect to Discovery and fetch data for Active Learners
19 @st.cache_resource
20 def fetch_data_discovery_al():
21     return fetch_data_from_query('query_DiscoveryAL.sql')
22
23
24 # Function to connect to Discovery and fetch data for Active Users
25 @st.cache_resource
26 def fetch_data_discovery_au():
27     return fetch_data_from_query('query_DiscoveryAU.sql')
28
29
30 # Helper function to fetch data based on SQL file
31 def fetch_data_from_query(query_file):
32     try:
33         connection_kwargs = {
34             'host': st.secrets["discovery"]["host"],
35             'port': st.secrets["discovery"]["port"],
36             'user': st.secrets["discovery"]["user"],
37             'password': st.secrets["discovery"]["password"],
38             'database': st.secrets["discovery"]["database"],
39             'cursorclass': pymysql.cursors.DictCursor,
40         }
41
42

```

Gambar 3.8 Syntax Fetch Data

Setelah proses fetch data dilakukan maka selanjutnya yaitu data processing sebagai tahapan akhir dalam melakukan connect data terlihat pada Gambar 3.9 Syntax Data Processing.

```

1 import streamlit as st
2 import pandas as pd
3 from fetch_data import fetch_data_discovery, fetch_data_discovery_al, fetch_data_discovery_au, fetch_data_sap, fetch_data_capture
4
5 @st.cache_data
6 def fill_empty_with_na(df):
7     """Fill empty or None values in the DataFrame with 'N/A'."""
8     return df.replace(['', None, '#VALUE!'], 'N/A').fillna('N/A')
9
10 @st.cache_data
11 def fetch_discovery_data():
12     df_discovery = fetch_data_discovery()
13     df_discovery['email'] = df_discovery['email'].str.strip().str.lower()
14     return fill_empty_with_na(df_discovery)
15
16 @st.cache_data
17 def fetch_discovery_al_data():
18     df_discovery_al = fetch_data_discovery_al()
19     df_discovery_al['email'] = df_discovery_al['email'].str.strip().str.lower()
20     return fill_empty_with_na(df_discovery_al)
21
22 @st.cache_data
23 def fetch_discovery_au_data():
24     df_discovery_au = fetch_data_discovery_au()
25     df_discovery_au['email'] = df_discovery_au['email'].str.strip().str.lower()
26     return fill_empty_with_na(df_discovery_au)
27
28 @st.cache_data
29 def fetch_sap_data():
30     selected_columns = ['name_sap', 'email', 'nik', 'unit', 'subunit', 'admin_hr', 'layer', 'generation', 'gender', 'division', 'department']
31     df_sap = fetch_data_sap(selected_columns)
32     df_sap['email'] = df_sap['email'].str.strip().str.lower()
33     df_sap['nik'] = df_sap['nik'].astype(str).str.zfill(6)
34     return fill_empty_with_na(df_sap)
35
36 @st.cache_data
37 def fetch_capture_data():
38     df_capture_sheet1, df_capture_sheet2, df_capture_sheet3 = fetch_data_capture()
39     df_capture_sheet1['email'] = df_capture_sheet1['email'].str.strip().str.lower()
40     df_capture_sheet2['email'] = df_capture_sheet2['email'].str.strip().str.lower()
41     df_capture_sheet3['email'] = df_capture_sheet3['email'].str.strip().str.lower()
42     df_capture_sheet1 = fill_empty_with_na(df_capture_sheet1)
43     df_capture_sheet2 = fill_empty_with_na(df_capture_sheet2)
44     df_capture_sheet3 = fill_empty_with_na(df_capture_sheet3)
45     return df_capture_sheet1, df_capture_sheet2, df_capture_sheet3
46
47 @st.cache_data
48 def finalize_data():
49     df_discovery = fetch_discovery_data()
50     df_discovery_al = fetch_discovery_al_data()
51     df_discovery_au = fetch_discovery_au_data()
52     df_sap = fetch_sap_data()
53     df_capture_sheet1, df_capture_sheet2, df_capture_sheet3 = fetch_capture_data()
54
55     # Pastikan 'gender' tetap ada saat menggabungkan df_discovery_al dan df_capture_sheet1
56     df_capture_sheet1['gender'] = None # Menambahkan kolom gender ke df_capture_sheet1 dengan nilai default None
57     df_combined_al_capture = pd.concat([df_discovery_al, df_capture_sheet1], ignore_index=True)
58
59     # Drop 'nik' column if it exists
60     if 'nik' in df_combined_al_capture.columns:
61         df_combined_al_capture = df_combined_al_capture.drop(columns=['nik'])
62
63     # Merge with SAP to determine status_learner
64     df_merged = pd.merge(df_combined_al_capture, df_sap, on='email', how='left', indicator=True)
65     df_merged['status_learner'] = df_merged['_merge'].apply(lambda x: 'Internal' if x == 'both' else 'External')
66     df_merged.drop(columns=['_merge'], inplace=True)
67
68     # Convert 'last_updated' to date
69     if 'last_updated' in df_merged.columns:
70         df_merged['last_updated'] = pd.to_datetime(df_merged['last_updated'], errors='coerce').dt.date
71
72     # Pastikan tidak ada duplikasi kolom gender
73     if 'gender_x' in df_merged.columns and 'gender_y' in df_merged.columns:
74         df_merged['gender'] = df_merged['gender_x'].combine_first(df_merged['gender_y'])
75         df_merged = df_merged.drop(columns=['gender_x', 'gender_y'])
76
77     # Normalisasi nilai gender menjadi 'female', 'male', atau 'n/a'
78     def normalize_gender(value):
79         value = str(value).strip().lower() # Pastikan nilai tidak mengandung spasi
80         if value in ['male', 'laki-laki', 'laki - laki', 'pria', 'm']:
81             return 'male'
82         elif value in ['female', 'perempuan', 'wanita', 'f']:
83             return 'female'
84         else:
85             return 'n/a'
86
87     df_merged['gender'] = df_merged['gender'].apply(normalize_gender)
88
89     # Konversi tipe data 'Customer ID' ke string
90     df_merged['Customer ID'] = df_merged['Customer ID'].astype(str)
91     df_merged = fill_empty_with_na(df_merged)
92
93     # Concatenate Discovery AU and Capture Sheet2 vertically
94     df_combined_au_capture = pd.concat([df_discovery_au, df_capture_sheet2], ignore_index=True)
95
96     # Convert 'created_at' to date
97     if 'created_at' in df_combined_au_capture.columns:
98         df_combined_au_capture['created_at'] = pd.to_datetime(df_combined_au_capture['created_at'], errors='coerce').dt.date
99
100     df_combined_au_capture = fill_empty_with_na(df_combined_au_capture)
101
102     # Convert dates in Capture Sheet3
103     if 'scheduled_at' in df_capture_sheet3.columns:
104         df_capture_sheet3['scheduled_at'] = pd.to_datetime(df_capture_sheet3['scheduled_at'], errors='coerce').dt.date
105     if 'done_at' in df_capture_sheet3.columns:
106         df_capture_sheet3['done_at'] = pd.to_datetime(df_capture_sheet3['done_at'], errors='coerce').dt.date
107
108     df_capture_sheet3 = fill_empty_with_na(df_capture_sheet3)
109
110     return df_discovery, df_sap, df_merged, df_combined_au_capture, df_capture_sheet3
111

```

Gambar 3.9 Syntax Data Processing

Untuk perancangan dashboard streamlit discovery proses data processing dilakukan dengan tujuan utamanya yaitu:

- a. Konsistensi data dengan memastikan format data seragam untuk mempermudah analisis dan penggabungan.
- b. Penanganan data tidak valid dengan membersihkan data dari nilai kosong,error, atau tidak valid
- c. Integrasi data dengan menggabungkan data dari berbagai sumber dengan cara yang logis dan terstruktur.
- d. Kesiapan data untuk analisis yang dimana dilakukan konversi tipe data misalnya tanggal dan menstandarkan nilai kolom misalnya gender agar siap digunakan analisis atau visualisasi.
- e. Pencegahan error hal ini dilakukan untuk menjamin data sudah bersih dan bebas dari duplikasi atau inkonsistensi yang dapat menyebabkan masalah saat digunakan pada aplikasi streamlit.

3.2.2 Page User Growth

Page user growth merupakan halaman pertama pada dashboard streamlit discovery yang dimana pada halaman ini dilakukan pada minggu ke – 3 dan minggu ke – 4 dengan membuat perancangan terlebih dahulu setelah perancangan selesai dilakukan maka akan dilanjutkan dengan implementasi yang dimana pada halaman user growth terdapat akumulasi seluruh active user secara real time baik itu active user maupun passive user dan pada halaman tersebut dapat di filter sesuai dengan yang diinginkan seperti life time, this year, this month, status maupun platform.

Pada dashboard streamlit discovery halaman user growth jua menampilkan visualisasi seperti line chart register user distribution dan bar chart platform distribution. Pada Gambar 3.10 Syntax Page User Growth memperlihatkan codingan pembuatan halaman tersebut.

EXPLORER

DASHBOARD-Discovery ...

__pycache__

.devcontainer

.streamlit

secrets.toml

cache

pages

2_Demography.py

3_Result_Traits.py

4_Internal_KG.py

5_Layer_Traits.py

.gitignore

1_User_Growth.py

data_processing.py

fetch_data.py

growth_center.png

Internal_KG.py

kognisi_logo.png

query_discovery.sql

query_DiscoveryAL.sql

query_DiscoveryAU.sql

requirements.txt

OUTLINE

TIMELINE

1_User_Growth.py X 2_Demography.py

1_User_Growth.py > ...

1 import pandas as pd

2 import streamlit as st

3 import altair as alt

4 from data_processing import finalize_data

5 from datetime import datetime

6

7 # Set the title and favicon that appear in the Browser's tab bar.

8 st.set_page_config(

9 page_title='User Growth',

10 page_icon='📊',

11)

12

13 # Return data from data_processing

14 df_discovery, df_sap, df_merged, df_combined_au_capture, df_capture_sheet3 = finalize_data()

15

16 # Display logo at the top of the sidebar

17 st.logo('kognisi_logo.png')

18

19 # Add filters to the sidebar

20 st.sidebar.header('Filters')

21

22 # Filter for Platform

23 platform_options = ['All'] + df_combined_au_capture['platform'].unique().tolist()

24 selected_platform = st.sidebar.selectbox('Select Platform', platform_options)

25

26 # Filter for Learner Status

27 status_options = ['All'] + df_combined_au_capture['learner_status'].unique().tolist()

28 selected_status = st.sidebar.selectbox('Select Status', status_options)

29

30 # Set the main title and description

31 st.markdown('')

32 # 📊 User Growth

33

34 In the User Growth Discovery - Capture, the key user metrics include:

35 1. **Overall**: Users who have registered at least one content on one platform.

36 2. **Active User**: Users who have accessed at least one content on one platform.

37 3. **Passive User**: Users who have not accessed any content on any platform.

38 '''

39

40 # Add some spacing

41 st.markdown('---')

42

43 # Create date filter for created_at

44 min_value = df_combined_au_capture['created_at'].min()

45 max_value = df_combined_au_capture['created_at'].max()

46

47 # Initialize session state for date filters if not already present

48 if 'from_date' not in st.session_state:

49 st.session_state.from_date = min_value

50 if 'to_date' not in st.session_state:

51 st.session_state.to_date = max_value

52

53 # Default date range

54 from_date = st.session_state.from_date

55 to_date = st.session_state.to_date

56

57 # Create buttons for shortcut filters

58 st.write('***Choose the data period:***')

59 col1, col2, col3 = st.columns(3)

UNIVERSITAS

MULTIMEDIA

NUSANTARA

41

Perancangan Dashboard Discovery..., Sherry Andiyani, Universitas Multimedia Nusantara

```

61 with col1:
62     if st.button('Lifetime'):
63         st.session_state.from_date = min_value
64         st.session_state.to_date = max_value
65
66 with col2:
67     if st.button('This Year'):
68         current_year = datetime.now().year
69         st.session_state.from_date = datetime(current_year, 1, 1).date()
70         st.session_state.to_date = datetime.now().date()
71
72 with col3:
73     if st.button('This Month'):
74         current_year = datetime.now().year
75         current_month = datetime.now().month
76         st.session_state.from_date = datetime(current_year, current_month, 1).date()
77         st.session_state.to_date = datetime.now().date()
78
79 # Allow manual date input
80 from_date, to_date = st.date_input(
81     '**Or pick the date manually:**',
82     values=[st.session_state.from_date, st.session_state.to_date],
83     min_value=min_value,
84     max_value=max_value
85 )
86 st.session_state.from_date = from_date
87 st.session_state.to_date = to_date
88
89 # Filter the data based on the selected date range
90 filtered_df = df_combined_aud_capture[
91     (df_combined_aud_capture['created_at'] <= to_date) &
92     (df_combined_aud_capture['created_at'] >= from_date)
93 ]
94
95 # Further filter based on selected platform
96 if selected_platform != 'All':
97     filtered_df = filtered_df[filtered_df['platform'] == selected_platform]
98
99 # Further filter based on selected learner status
100 if selected_status != 'All':
101     filtered_df = filtered_df[filtered_df['learner_status'] == selected_status]
102
103 # Active Users section
104 st.header('Active User', divider='gray')
105
106 # Calculate the distinct counts of users based on email
107 total_count = filtered_df['email'].nunique() # Total registered users (unique emails)
108 Active_count = filtered_df[filtered_df['learner_status'] == 'Active']['email'].nunique() # Unique Active users
109 Passive_count = filtered_df[filtered_df['learner_status'] == 'Passive']['email'].nunique() # Unique Passive users
110
111 # Display metrics column
112 col1, col2, col3 = st.columns(3)
113 with col1:
114     st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>Overall: <span style='color: red;'>{total_count}</span></strong></p>",
115                 unsafe_allow_html=True)
116 with col2:
117     st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>Active User: <span style='color: red;'>{Active_count}</span></strong></p>",
118                 unsafe_allow_html=True)
119 with col3:
120     st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>Passive User: <span style='color: red;'>{Passive_count}</span></strong></p>",
121                 unsafe_allow_html=True)
122
123 # Register User Distribution
124 st.subheader('Register User Distribution', divider='gray')
125
126 # Create a line chart for active users based on created_at
127 active_user_counts = (filtered_df
128     .groupby('created_at')

```

UNIVERSITAS
MULTIMEDIA
NUSANTARA

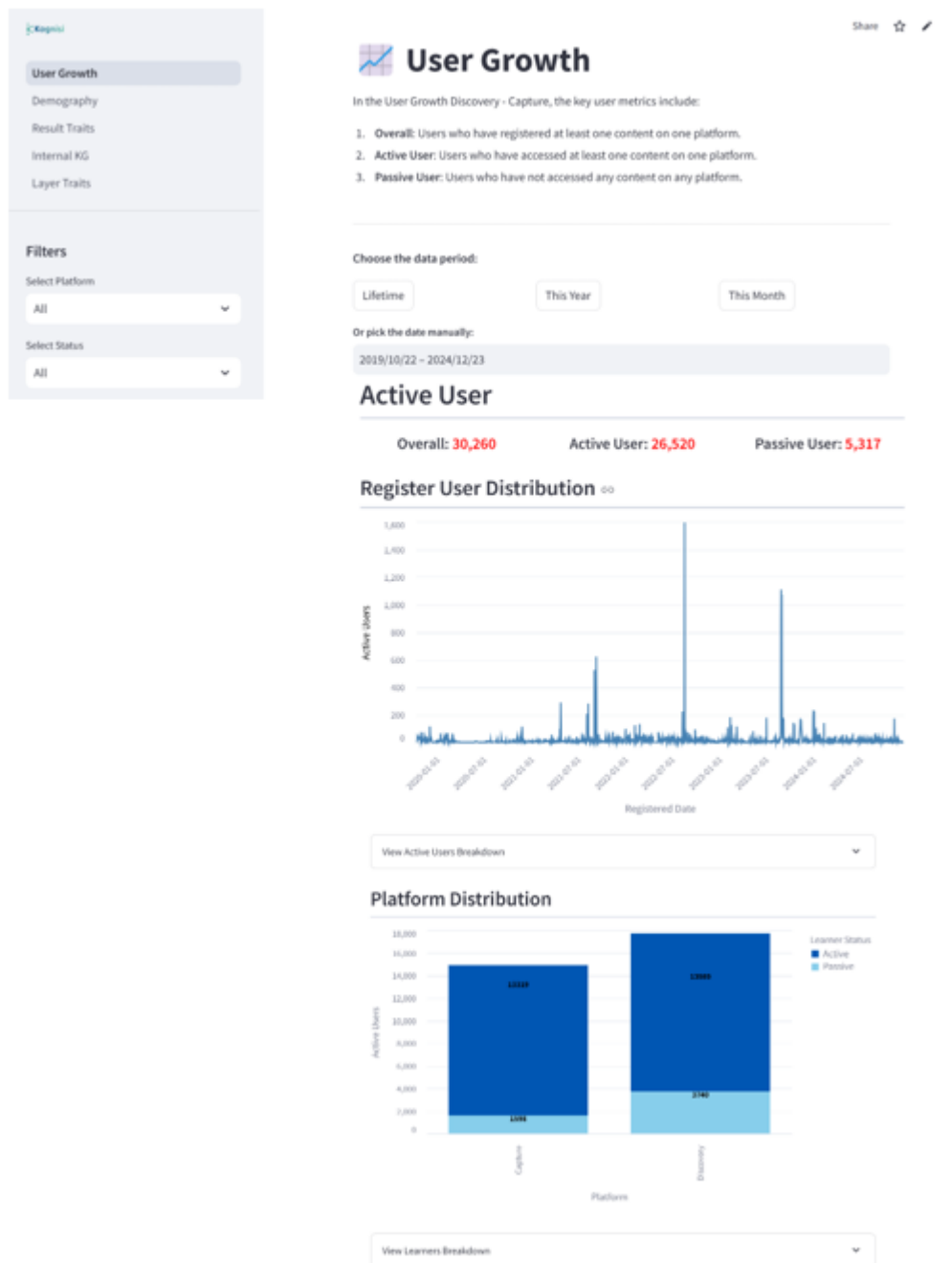
```

130         .reset_index()
131
132     # Create a line chart for active learners
133     line_chart = alt.Chart(active_user_counts).mark_line(
134         stroke='steelblue', # Change the line color
135         strokeWidth=2 # Increase line width for better visibility
136     ).encode(
137         x=alt.X('created_at:T', title='Registered Date',
138             axis=alt.Axis(format='%Y-%m-%d', labelAngle=-45)), # Tilt x-axis labels
139         y=alt.Y('active_users:Q', title='Active Users', axis=alt.Axis(titleColor='black')),
140         tooltips=[
141             alt.Tooltip('created_at:T', title='Registered Date', format='%Y-%m-%d'),
142             alt.Tooltip('active_users:Q', title='Active Users')
143         ]
144     ).properties(
145         width=600,
146         height=400
147     )
148
149     # Display the chart
150     st.altair_chart(line_chart, use_container_width=True)
151
152
153     # Breakdown data table for active users
154     with st.expander("View Active Users Breakdown"):
155         st.dataframe(active_user_counts)
156
157     # Platform Distribution
158     st.subheader('Platform Distribution', divider='gray')
159
160     # Create a bar chart breakdown by platform and learner_status
161     platform_breakdown = (filtered_df
162         .groupby(['platform', 'learner_status'])
163         .agg(active_users=('enroll', 'nunique'))
164         .reset_index())
165
166     # Generate the stacked bar chart using Altair
167     bar_chart = alt.Chart(platform_breakdown).mark_bar().encode(
168         x=alt.X('platform:O', title='Platform'),
169         y=alt.Y('active_users:Q', title='Active Users'),
170         color=alt.Color('learner_status:N', title='Learner Status', scale=alt.Scale(domain=['Active', 'Passive'], range=['#0056b3', '#87ceeb'])),
171         tooltip=[
172             alt.Tooltip('platform:O', title='Platform'),
173             alt.Tooltip('learner_status:N', title='Learner Status'),
174             alt.Tooltip('active_users:Q', title='Active Users')
175         ]
176     ).properties(
177         width=600,
178         height=400
179     )
180
181     # Add a text layer to display active user counts within each color section
182     text_active = bar_chart.mark_text(
183         align='center',
184         baseline='middle',
185         dy=2, # Positioning text slightly inside the active (blue) bar
186         fontWeight='bold'
187     ).transform_filter(
188         alt.datum.learner_status == 'Active'
189     ).encode(
190         text=alt.Text('active_users:Q'),
191         color=alt.value('black') # White text for better contrast on the dark blue bar
192     )
193
194     text_passive = bar_chart.mark_text(
195         align='center',
196         baseline='middle',
197         dy=5, # Positioning text slightly inside the passive (light blue) bar
198         fontWeight='bold'
199     ).transform_filter(
200         alt.datum.learner_status == 'Passive'
201     ).encode(
202         text=alt.Text('active_users:Q'),
203         color=alt.value('black') # Black text for better contrast on the light blue bar
204     )
205
206     # Layer the bar chart with the text labels for both active and passive users
207     layered_chart = bar_chart + text_active + text_passive
208
209     # Display the bar chart in Streamlit
210     st.altair_chart(layered_chart, use_container_width=True)
211
212     # Breakdown data table for active users
213     with st.expander("View Learners Breakdown"):
214         st.dataframe(platform_breakdown)
215

```

Gambar 3.10 Syntax Page User Growth

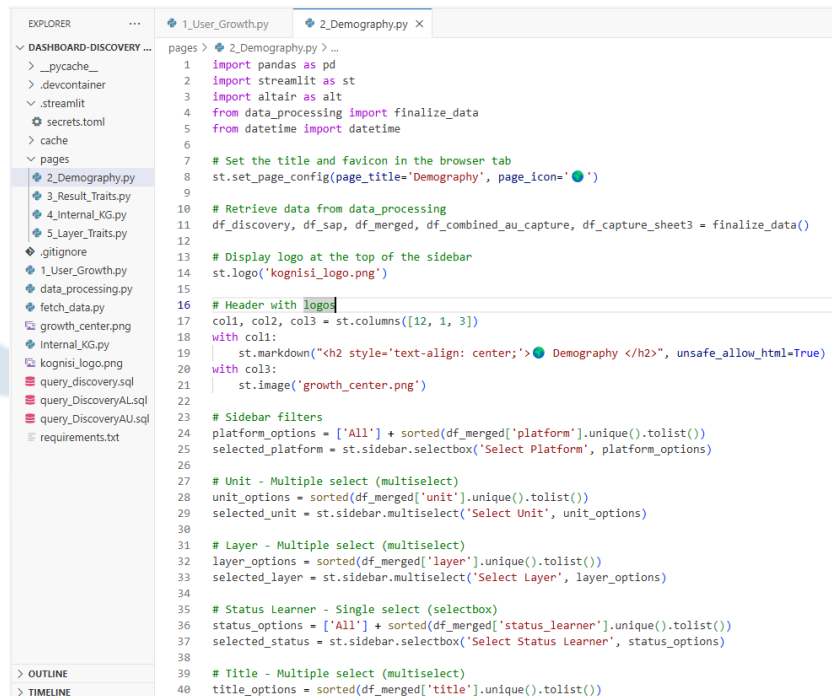
Setelah syntax tersebut dijalankan maka halaman user growth pada dashboard streamlit terlihat seperti tampilan pada Gambar 3.11 Output Page User Growth.



Gambar 3.11 Output Page User Growth

3.2.3 Page Demography

Page demography merupakan halaman kedua pada dashboard streamlit discovery yang dimana pada halaman ini dilakukan pada minggu ke – 5 sampai dengan minggu ke – 7 dengan membuat perancangan terlebih dahulu setelah perancangan selesai dilakukan maka akan dilanjutkan dengan implementasi yang dimana pada halaman demography terdapat akumulasi seluruh active user secara real time baik itu active user maupun passive user dan pada halaman tersebut dapat di filter sesuai dengan yang diinginkan seperti life time, this year, this month, platform, unit, layer, status, title dan company. Pada dashboard streamlit discovery halaman user growth juga menampilkan visualisasi seperti line chart test date distribution dan bar chart demography distribution yang nantinya dapat difilter sesuai keinginan stakeholder. Breakdown dalam demography mencakup unit, layer, gender, company, generation, institution, dan last education. Pada Gambar 3.12 Syntax Page Demography memperlihatkan codingan pembuatan halaman tersebut.



```
1 import pandas as pd
2 import streamlit as st
3 import altair as alt
4 from data_processing import finalize_data
5 from datetime import datetime
6
7 # Set the title and favicon in the browser tab
8 st.set_page_config(page_title="Demography", page_icon="📊")
9
10 # Retrieve data from data_processing
11 df_discovery, df_sap, df_merged, df_combined_au_capture, df_capture_sheet3 = finalize_data()
12
13 # Display logo at the top of the sidebar
14 st.logo("kognisi_logo.png")
15
16 # Header with logos
17 col1, col2, col3 = st.columns([12, 1, 3])
18 with col1:
19     st.markdown("<h2 style='text-align: center;'>📊 Demography </h2>", unsafe_allow_html=True)
20 with col3:
21     st.image("growth_center.png")
22
23 # Sidebar filters
24 platform_options = ["All"] + sorted(df_merged["platform"].unique().tolist())
25 selected_platform = st.sidebar.selectbox("Select Platform", platform_options)
26
27 # Unit - Multiple select (multiselect)
28 unit_options = sorted(df_merged["unit"].unique().tolist())
29 selected_unit = st.sidebar.multiselect("Select Unit", unit_options)
30
31 # Layer - Multiple select (multiselect)
32 layer_options = sorted(df_merged["layer"].unique().tolist())
33 selected_layer = st.sidebar.multiselect("Select Layer", layer_options)
34
35 # Status Learner - Single select (selectbox)
36 status_options = ["All"] + sorted(df_merged["status_learner"].unique().tolist())
37 selected_status = st.sidebar.selectbox("Select Status Learner", status_options)
38
39 # Title - Multiple select (multiselect)
40 title_options = sorted(df_merged["title"].unique().tolist())
```

```

41 selected_title = st.sidebar.multiselect('Select Title', title_options)
42
43 # Company - Multiple select (multiselect)
44 company_options = sorted(df_merged['Company'].unique().tolist())
45 selected_company = st.sidebar.multiselect('Select Company', company_options)
46
47 # Main title and description
48 st.markdown('')
49 In the Active Learners Growth Discovery - Capture, the key user metrics include:
50 1. **Overall**: Users test on the Discovery and Capture platforms.
51 2. **Internal User**: Active learners who are Kompas Gramedia Group employees.
52 3. **External User**: Active learners who are not Kompas Gramedia Group employees.
53 ''')
54 st.markdown('---')
55
56 # Date filter setup
57 min_value, max_value = df_merged['last_updated'].min(), df_merged['last_updated'].max()
58
59 # Initialize session state for date filters
60 if 'from_date' not in st.session_state:
61     st.session_state.from_date, st.session_state.to_date = min_value, max_value
62
63 # Shortcut buttons for date selection
64 st.write('Choose the date period:')
65 col1, col2, col3 = st.columns(3)
66 with col1:
67     if st.button('Lifetime'):
68         st.session_state.from_date, st.session_state.to_date = min_value, max_value
69 with col2:
70     if st.button('This Year'):
71         current_year = datetime.now().year
72         st.session_state.from_date, st.session_state.to_date = datetime(current_year, 1, 1).date(), datetime.now().date()
73 with col3:
74     if st.button('This Month'):
75         current_year, current_month = datetime.now().year, datetime.now().month
76         st.session_state.from_date, st.session_state.to_date = datetime(current_year, current_month, 1).date(), datetime.now().date()
77
78 # Ensure date selections are within valid range
79 st.session_state.from_date = max(st.session_state.from_date, min_value)
80 st.session_state.to_date = min(st.session_state.to_date, max_value)
81
82 # Manual date input
83 from_date, to_date = st.date_input(
84     'Or pick the date manually:',
85     value=(st.session_state.from_date, st.session_state.to_date),
86     min_value=min_value,
87     max_value=max_value
88 )
89
90 # Update session state with manual input
91 st.session_state.from_date, st.session_state.to_date = from_date, to_date
92
93 # Data filtering based on selected dates and platform
94 filtered_df = df_merged[
95     (df_merged['last_updated'] >= from_date) &
96     (df_merged['last_updated'] <= to_date)
97 ]
98
99 if selected_platform != 'All':
100     filtered_df = filtered_df[filtered_df['platform'] == selected_platform]
101
102 if selected_status != 'All':
103     filtered_df = filtered_df[filtered_df['status_learner'] == selected_status]
104
105 if selected_unit:
106     filtered_df = filtered_df[filtered_df['unit'].isin(selected_unit)]
107
108

```

UNIVERSITAS
MULTIMEDIA
NUSANTARA


```

176 bar_chart = alt.Chart(data).mark_bar().encode(
177     x=alt.X(f'{x_col}:Q', title='Active Learners'),
178     y=alt.Y(f'{y_col}:O', title=title, sort='-x'),
179     tooltip=[
180         alt.Tooltip(f'{y_col}:O', title=title),
181         alt.Tooltip(f'{x_col}:Q', title='Active Learners')
182     ]
183 ).properties(width=600, height=400)
184
185 # Text layer for counts
186 text = bar_chart.mark_text(
187     align='left',
188     fontWeight='bold',
189     baseline='middle',
190     dx=3 # Adjusts position of the text to fit nicely
191 ).encode(
192     text=alt.Text(f'{x_col}:Q', format=',') # Display counts
193 )
194
195 return bar_chart + text
196
197 # Breakdown visualization based on selected demographic option
198 breakdown_mapping = {
199     'Unit': 'unit',
200     'Layer': 'layer',
201     'Generation': 'generation',
202     'Institution': 'institution',
203     'Last Education': 'last_education',
204     'Company': 'Company'
205 }
206
207 if selected_breakdown in breakdown_mapping:
208     # Menghitung jumlah active learners berdasarkan breakdown yang dipilih
209     counts = (
210         filtered_df
211         .groupby(['status_learner', breakdown_mapping[selected_breakdown]])
212         .agg(active_learners=('email', 'nunique'))
213         .reset_index()
214     )
215
216 # Penanganan untuk Institution, Last Education, dan Company
217 if selected_breakdown in ['Institution', 'Last Education', 'Company']:
218     # Hitung total active learners untuk setiap kategori
219     total_counts = (
220         counts.groupby(breakdown_mapping[selected_breakdown])
221         .agg(total_active_learners=('active_learners', 'sum'))
222         .reset_index()
223     )
224     # Gabungkan total dengan data utama
225     counts = counts.merge(total_counts, on=breakdown_mapping[selected_breakdown])
226     # Hitung persentase untuk 100% stacked bar chart
227     counts['percentage'] = (counts['active_learners'] / counts['total_active_learners']) * 100
228
229 # Batasi hanya 10 data teratas berdasarkan total_active_learners untuk Institution
230 if selected_breakdown == 'Institution':
231     counts = counts.sort_values(by='total_active_learners', ascending=False).head(10)
232     st.subheader("Top 10 Institution") # Subheader khusus untuk Institution

```

```

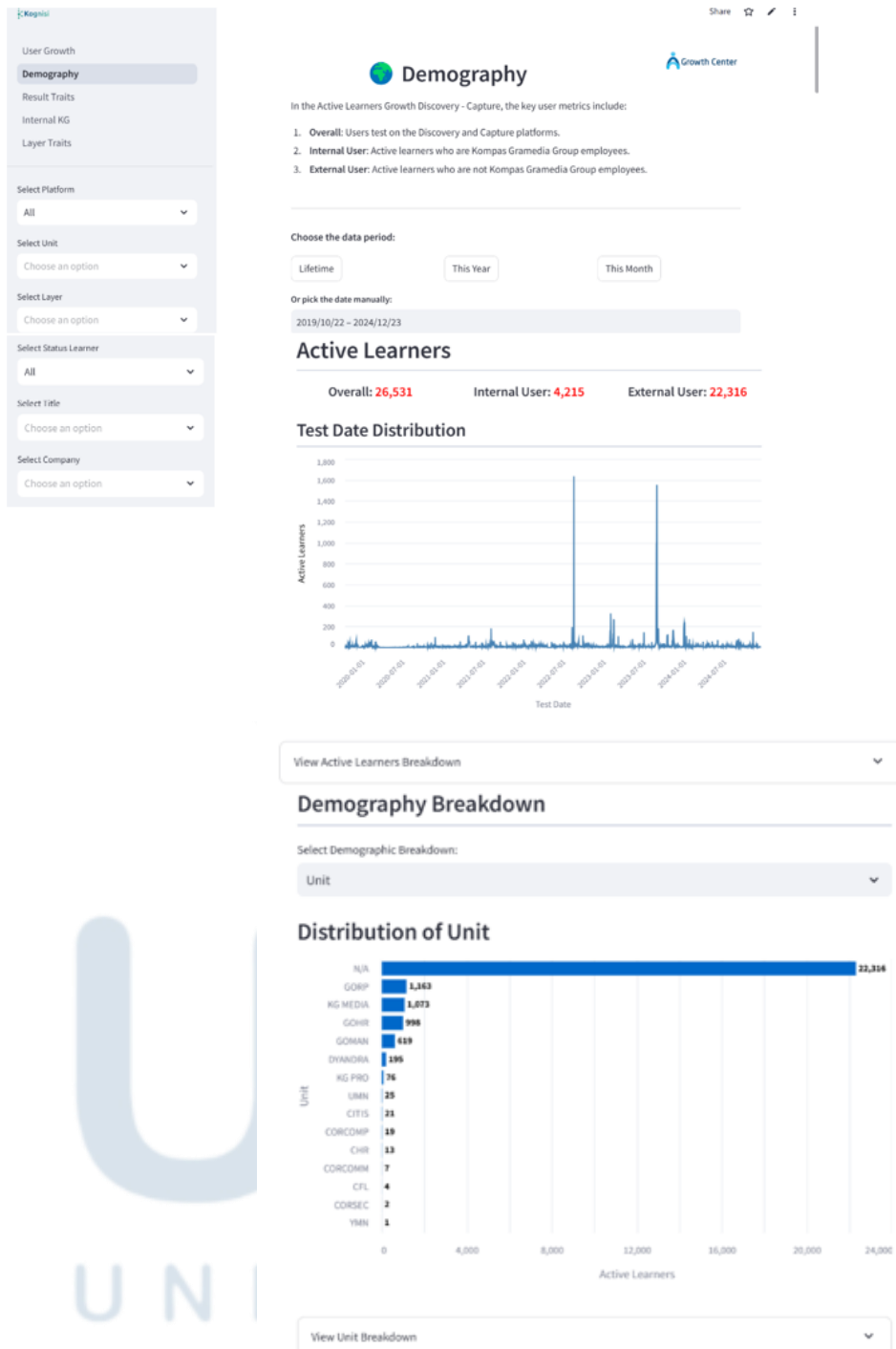
233     else:
234         st.subheader(f"Distribution of {selected_breakdown}") # Subheader umum untuk Last Education dan Company
235
236     # Visualisasi stacked bar chart
237     stacked_bar_chart = alt.Chart(counts).mark_bar().encode(
238         x=alt.X('percentage:Q', title='Active learners (X)'),
239         y=alt.Y(f'{breakdown_mapping[selected_breakdown]}:N', title=selected_breakdown, sorts='-x'),
240         color=alt.Color(
241             'status_learner:N',
242             title='User Type',
243             scale=alt.Scale(domain=['Internal', 'External'], range=['blue', 'orange'])
244         ),
245         tooltip=[
246             alt.Tooltip(f'{breakdown_mapping[selected_breakdown]}:N', title=selected_breakdown),
247             alt.Tooltip('active_learners:Q', title='Active Learners'),
248             alt.Tooltip('status_learner:N', title='Status learner')
249         ]
250     ).properties(width=600, height=400)
251
252     st.altair_chart(stacked_bar_chart, use_container_width=True)
253
254     # Menampilkan data dalam expander
255     with st.expander(f"View {selected_breakdown} Breakdown"):
256         st.dataframe(counts)
257
258     # Penanganan untuk breakdown lain (selain Institution, Last Education, dan Company)
259     else:
260         # Tambahkan subheader untuk Unit, Layer, Generation
261         st.subheader(f"Distribution of {selected_breakdown}")
262
263         # Mengurutkan berdasarkan jumlah active learners dalam urutan menurun
264         counts = counts.sort_values(by='active_learners', ascending=False)
265
266         # Visualisasi bar chart
267         st.altair_chart(
268             create_bar_chart_with_text(counts, 'active_learners', breakdown_mapping[selected_breakdown], selected_breakdown),
269             use_container_width=True
270         )
271
272         # Menampilkan data dalam expander
273         with st.expander(f"View {selected_breakdown} Breakdown"):
274             st.dataframe(counts)
275
276     # Penanganan khusus untuk Gender
277     elif selected_breakdown == 'Gender':
278         # Menghitung jumlah active learners berdasarkan gender
279         gender_counts = {
280             'filtered_df':
281                 counts
282                 .groupby(['status_learner', 'gender'])
283                 .agg(active_learners=('email', 'nunique'))
284                 .reset_index()
285                 .sort_values(by='active_learners', ascending=False)
286         }
287
288         # Subheader untuk Gender Distribution
289         st.subheader("Distribution of Gender")
290
291         # Visualisasi pie chart untuk Gender
292         pie_chart_gender = alt.Chart(gender_counts).mark_bar().encode(
293             theta=alt.theta('active_learners:Q', title='Active learners'),
294             color=alt.color('gender:N', title='Gender'),
295             tooltip=[
296                 alt.Tooltip('gender:N', title='Gender'),
297                 alt.Tooltip('active_learners:Q', title='Active learners')
298             ]
299         ).properties(width=600, height=400)
300
301         st.altair_chart(pie_chart_gender, use_container_width=True)
302
303         # Menampilkan data dalam expander
304         with st.expander("View Gender Breakdown"):
305             st.dataframe(gender_counts)

```

Gambar 3.12 Syntax Page Demography

Setelah syntax tersebut dijalankan maka halaman demography pada dashboard streamlit terlihat seperti tampilan pada Gambar 3.13

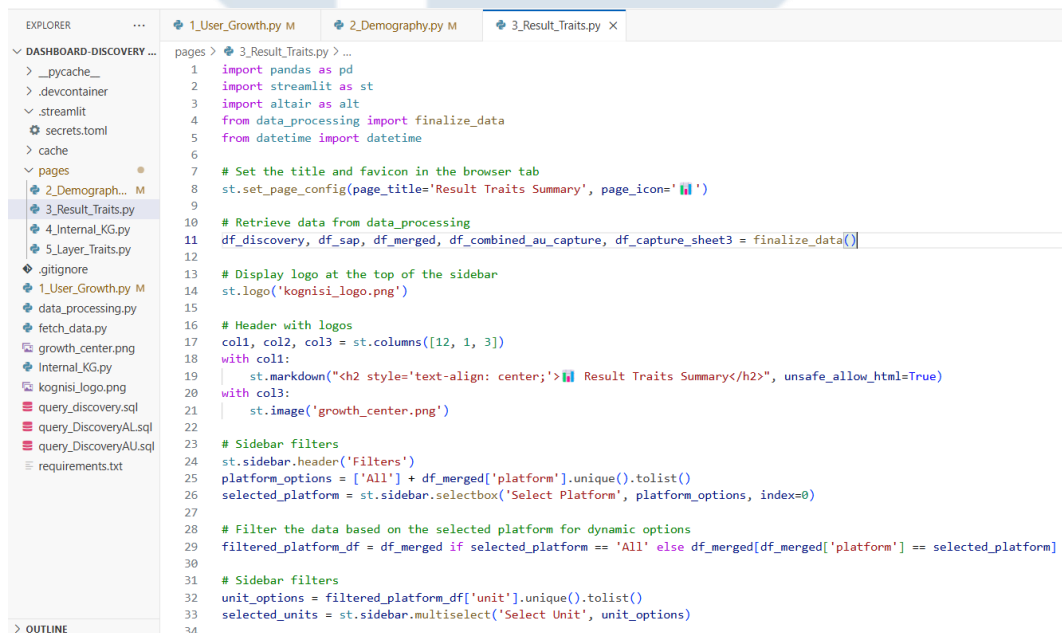
Output Page Demography



Gambar 3.13 Output Page Demography

3.2.4 Page Result Traits

Page result traits merupakan halaman ketiga pada dashboard streamlit discovery yang dimana pada halaman ini dilakukan pada minggu ke – 8 sampai dengan minggu ke – 10 dengan membuat perancangan terlebih dahulu setelah perancangan selesai dilakukan maka akan dilanjutkan dengan implementasi yang dimana pada halaman result traits terdapat filter sesuai dengan yang diinginkan seperti life time, this year, this month, platform, unit, layer, status, title dan company. Pada dashboard streamlit discovery halaman result traits juga menampilkan visualisasi seperti bar chart platform distribution pada setiap unit yang ada pada perusahaan Kompas Gramedia Group serta yang nantinya dapat difilter sesuai keinginan stakeholder. Pada Gambar 3.14 Syntax Result Traits memperlihatkan codingan pembuatan halaman tersebut.



```
1 import pandas as pd
2 import streamlit as st
3 import altair as alt
4 from data_processing import finalize_data
5 from datetime import datetime
6
7 # Set the title and favicon in the browser tab
8 st.set_page_config(page_title='Result Traits Summary', page_icon='📊')
9
10 # Retrieve data from data_processing
11 df_discovery, df_sap, df_merged, df_combined_capture, df_capture_sheet3 = finalize_data()
12
13 # Display logo at the top of the sidebar
14 st.logo('kognisi_logo.png')
15
16 # Header with logos
17 col1, col2, col3 = st.columns([12, 1, 3])
18 with col1:
19     st.markdown("<h2 style='text-align: center;'>📊 Result Traits Summary</h2>", unsafe_allow_html=True)
20 with col3:
21     st.image('growth_center.png')
22
23 # Sidebar filters
24 st.sidebar.header('Filters')
25 platform_options = ['All'] + df_merged['platform'].unique().tolist()
26 selected_platform = st.sidebar.selectbox('Select Platform', platform_options, index=0)
27
28 # Filter the data based on the selected platform for dynamic options
29 filtered_platform_df = df_merged if selected_platform == 'All' else df_merged[df_merged['platform'] == selected_platform]
30
31 # Sidebar filters
32 unit_options = filtered_platform_df['unit'].unique().tolist()
33 selected_units = st.sidebar.multiselect('Select Unit', unit_options)
34
```

```

35 layer_options = filtered_platform_df['layer'].unique().tolist()
36 selected_layers = st.sidebar.multiselect('Select Layer', layer_options)
37
38 status_options = ['All'] + filtered_platform_df['status_learner'].unique().tolist()
39 selected_status = st.sidebar.selectbox('Select Status Learner', status_options)
40
41 title_options = filtered_platform_df['title'].unique().tolist()
42 selected_titles = st.sidebar.multiselect('Select Title', title_options)
43
44 company_options = filtered_platform_df['Company'].unique().tolist()
45 selected_company = st.sidebar.multiselect('Select Company', company_options)
46
47 # Date filter setup
48 min_value, max_value = df_merged['last_updated'].min(), df_merged['last_updated'].max()
49
50 # Initialize session state for date filters
51 if 'from_date' not in st.session_state:
52     st.session_state.from_date, st.session_state.to_date = min_value, max_value
53
54 # Shortcut buttons for date selection
55 st.write("""Choose the data period: """)
56 col1, col2, col3 = st.columns(3)
57 with col1:
58     if st.button('Lifetime'):
59         st.session_state.from_date, st.session_state.to_date = min_value, max_value
60 with col2:
61     if st.button('This Year'):
62         current_year = datetime.now().year
63         st.session_state.from_date, st.session_state.to_date = datetime(current_year, 1, 1).date(), datetime.now().date()
64 with col3:
65     if st.button('This Month'):
66         current_year, current_month = datetime.now().year, datetime.now().month
67         st.session_state.from_date, st.session_state.to_date = datetime(current_year, current_month, 1).date(), datetime.now().date()
68
69 # Ensure date selections are within valid range
70 st.session_state.from_date = max(st.session_state.from_date, min_value)
71 st.session_state.to_date = min(st.session_state.to_date, max_value)
72
73 # Manual date input
74 from_date, to_date = st.date_input(
75     """Or pick the date manually: """,
76     value=(st.session_state.from_date, st.session_state.to_date),
77     min_value=min_value,
78     max_value=max_value
79 )
80
81 # Update session state with manual input
82 st.session_state.from_date, st.session_state.to_date = from_date, to_date
83
84 # Data filtering based on selected filters
85 filtered_df = df_merged[
86     (df_merged['last_updated'] <= to_date) &
87     (df_merged['last_updated'] >= from_date)
88 ]
89
90 if selected_platform != 'All':
91     filtered_df = filtered_df[filtered_df['platform'] == selected_platform]
92 if selected_status != 'All':
93     filtered_df = filtered_df[filtered_df['status_learner'] == selected_status]
94 if selected_units:
95     filtered_df = filtered_df[filtered_df['unit'].isin(selected_units)]
96 if selected_layers:
97     filtered_df = filtered_df[filtered_df['layer'].isin(selected_layers)]
98 if selected_titles:
99     filtered_df = filtered_df[filtered_df['title'].isin(selected_titles)]
100 if selected_company:
101     filtered_df = filtered_df[filtered_df['Company'].isin(selected_company)]
102
103 # Display horizontal stacked bar chart of platform per unit if 'All' is selected for platform
104 if selected_platform == 'All':
105     st.write("""### Platform Distribution""")
106
107     # Group data to calculate counts and unique emails
108     platform_unit_df = (
109         filtered_df.groupby(['unit', 'platform'])
110         .agg(count=('email', 'size'), Active_Learners=('email', 'nunique'))
111         .reset_index()
112     )
113
114     # Calculate total counts per unit
115     platform_unit_df['total'] = platform_unit_df.groupby('unit')['count'].transform('sum')
116     platform_unit_df['percent'] = platform_unit_df['count'] / platform_unit_df['total'] * 100
117

```

```

118 # Sort units by total count of all platforms
119 sorted_units = (
120     platform_unit_df.groupby('unit')['count']
121     .sum()
122     .sort_values(ascending=False)
123     .index.tolist()
124 )
125
126 # Create the 100% stacked horizontal bar chart with count labels
127 platform_unit_chart = (
128     alt.Chart(platform_unit_df)
129     .mark_bar()
130     .encode(
131         y=alt.Y('unit:N', sort=sorted_units, title='Unit'),
132         x=alt.X('percent:Q', stack='normalize', title='Percentage (%)'),
133         color=alt.Color('platform:N', title='Platform'),
134         tooltip=['platform', 'unit', 'Active_Learners']
135     )
136     .properties(width=600, height=400)
137 )
138
139 # Combine the bar chart and the text labels
140 st.altair_chart(platform_unit_chart, use_container_width=True)
141
142 # Data download for Growth Inventory
143 with st.expander("Data Distribution Platform"):
144     st.write(platform_unit_df)
145     csv = platform_unit_df.to_csv(index=False).encode('utf-8')
146     st.download_button("Download Data", data=csv, file_name="DistributionPlatform.csv", mime="text/csv",
147                        help="Click here to download the data as a CSV file")
148
149
150 # Active Users section for Discovery
151 if selected_platform == 'Discovery':
152     st.header('Active Learners - Discovery', divider='gray')
153     total_count = filtered_df['email'].nunique()
154     internal_count = filtered_df[filtered_df['status_learner'] == 'Internal']['Customer ID'].nunique()
155     external_count = filtered_df[filtered_df['status_learner'] == 'External']['Customer ID'].nunique()
156
157 # Display metrics columns
158 col1, col2, col3 = st.columns(3)
159 with col1:
160     st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>Overall: <span style='color: red;'>{total_count}</span></strong></p>",
161                unsafe_allow_html=True)
162 with col2:
163     st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>Internal User: <span style='color: red;'>{internal_count}</span></strong></p>",
164                unsafe_allow_html=True)
165 with col3:
166     st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>External User: <span style='color: red;'>{external_count}</span></strong></p>",
167                unsafe_allow_html=True)
168
169 # Active learners by bundle_name
170 bundle_names = ['GI', 'LEAN', 'ELITE', 'Genuine', 'Astaka']
171 if 'title' in filtered_df.columns:
172     # Group by bundle_name and count unique Customer IDs
173     df_active_learners = filtered_df.groupby(['Customer ID', 'title']).size().reset_index(name='test_count')
174     bundle_counts = (bundle: df_active_learners[df_active_learners['title'] == bundle]['Customer ID'].nunique() for bundle in bundle_names)
175
176 # Display active learners counts
177 st.markdown("<h3>ACTIVE LEARNERS by Bundle</h3>", unsafe_allow_html=True)
178 col1, col2, col3, col4, col5 = st.columns(5)
179 for i, bundle in enumerate(bundle_names):
180     with eval(f'col{i + 1}'):
181         st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>{bundle}: <span style='color: red;'>{bundle_counts.get(bundle, 0)}</span></strong></p>", unsafe_allow_html=True)
182
183 # Create bar chart for top bundles
184 bundle_chart_data = pd.DataFrame.from_dict(bundle_counts, orient='index', columns=['count']).reset_index()
185 bundle_chart_data.columns = ['Bundle', 'Active Learners']
186
187 # Create the bar chart using Altair
188 # Stacked bar chart for Growth Inventory
189 st.subheader("Top Test Discovery")
190 bundle_bar_chart = (
191     alt.Chart(bundle_chart_data)
192     .mark_bar()
193     .encode(
194         x=alt.X('Active Learners:Q', title='Active Learners Count'),
195         y=alt.Y('Bundle:N', title='Bundle', sort='-x'),
196         color=alt.Color('Bundle:N', title='Bundle'),
197         tooltip=['Bundle:N', 'Active Learners:Q']
198     )
199     .properties(width=600, height=400)
200 )
201
202 # Display the chart
203 st.altair_chart(bundle_bar_chart, use_container_width=True)
204
205 # Data download for Top Test Discovery
206 with st.expander("Top Test Discovery"):
207     st.write(bundle_chart_data)
208     csv = bundle_chart_data.to_csv(index=False).encode('utf-8')
209     st.download_button("Download Data", data=csv, file_name="Top_Test_Discovery.csv", mime="text/csv", help="Click here to download the data as a CSV file")
210

```

```

213 # Active learners data for Bundle GI
214 gi_active_learners = filtered_df[filtered_df['title'] == 'GI']
215
216 # Get highest scores
217 highest_scores = gi_active_learners.loc[gi_active_learners.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
218 gi_active_learners_data = highest_scores[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
219
220 # Stacked bar chart for Growth Inventory
221 st.subheader("Growth Inventory")
222 gi_filtered = filtered_df[filtered_df['title'] == 'GI']
223 gi_distribution = gi_filtered.groupby(['Test Name', 'typology']).agg({'Customer ID': 'nunique'}).reset_index()
224 gi_distribution.columns = ['Test Name', 'typology', 'Active Users']
225
226 # Calculate percentages
227 total_active_users_per_test = gi_distribution.groupby('Test Name')['Active Users'].transform('sum')
228 gi_distribution['Percentage'] = (gi_distribution['Active Users'] / total_active_users_per_test * 100).round(2)
229
230 # Plot chart
231 chart = alt.Chart(gi_distribution).mark_bar().encode(
232     x='Test Name',
233     y='Active Users',
234     color='typology',
235     tooltip=[
236         alt.Tooltip('Test Name:N', title='Test Name'),
237         alt.Tooltip('typology:N', title='Typology'),
238         alt.Tooltip('Active Users:Q', title='Active Learners'),
239         alt.Tooltip('Percentage:Q', title='Percentage', format='.1f') # Format percentage with one decimal place
240     ]
241 ).properties(
242     width=600,
243     height=400
244 ).configure_mark(
245     opacity=0.8
246 ).configure_axis(
247     labelFontSize=12,
248     titleFontSize=14
249 ).configure_title(
250     fontSize=16
251 )
252
253 st.altair_chart(chart, use_container_width=True)
254
255 # Data download for Growth Inventory
256 with st.expander("Data Growth Inventory"):
257     st.write(gi_distribution)
258     csv = gi_distribution.to_csv(index=False).encode('utf-8')
259     st.download_button("Download Data", data=csv, file_name="Growth_Inventory.csv", mime="text/csv", help="Click here to download the data as a CSV file")
260
261
262 # Repeat for LEAN
263 lean_active_learners = filtered_df[filtered_df['title'] == 'LEAN']
264
265 # Get highest scores for LEAN
266 highest_scores_lean = lean_active_learners.loc[lean_active_learners.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
267 lean_active_learners_data = highest_scores_lean[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
268
269 # Stacked bar chart for LEAN
270 st.subheader("LEAN")
271 lean_filtered = filtered_df[filtered_df['title'] == 'LEAN']
272 lean_distribution = lean_filtered.groupby(['Test Name', 'typology']).agg({'Customer ID': 'nunique'}).reset_index()
273 lean_distribution.columns = ['Test Name', 'typology', 'Active Users']
274
275 # Calculate percentages for LEAN
276 total_active_users_per_test_lean = lean_distribution.groupby('Test Name')['Active Users'].transform('sum')
277 lean_distribution['Percentage'] = (lean_distribution['Active Users'] / total_active_users_per_test_lean * 100).round(2)
278
279 # Plot chart for LEAN
280 chart_lean = alt.Chart(lean_distribution).mark_bar().encode(
281     x='Test Name',
282     y='Active Users',
283     color='typology',
284     tooltip=[
285         alt.Tooltip('Test Name:N', title='Test Name'),
286         alt.Tooltip('typology:N', title='Typology'),
287         alt.Tooltip('Active Users:Q', title='Active Learners'),
288         alt.Tooltip('Percentage:Q', title='Percentage', format='.1f') # Format percentage with one decimal place
289     ]
290 ).properties(
291     width=600,
292     height=400
293 ).configure_mark(
294     opacity=0.8
295 ).configure_axis(
296     labelFontSize=12,
297     titleFontSize=14
298 ).configure_title(
299     fontSize=16
300 )
301
302 st.altair_chart(chart_lean, use_container_width=True)
303
304 # Data download for LEAN
305 with st.expander("Data LEAN"):
306     st.write(lean_distribution)
307     csv_lean = lean_distribution.to_csv(index=False).encode('utf-8')
308     st.download_button("Download Data", data=csv_lean, file_name="LEAN.csv", mime="text/csv", help="Click here to download the data as a CSV file")

```

```

309
310 # Display final result distribution
311 st.subheader("FINAL RESULT LEAN")
312 lean_active_learners = filtered_df[filtered_df['title'] == 'LEAN']
313
314 # Get highest scores for LEAN
315 highest_scores_lean = lean_active_learners.loc[lean_active_learners.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
316 lean_active_learners_data = highest_scores_lean[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
317
318 # Get unique combinations of Customer ID and Test Date
319 unique_combinations = lean_active_learners_data[['Customer ID', 'last_updated']].drop_duplicates()
320
321 # Merge back to get final_result
322 unique_results = unique_combinations.merge(lean_active_learners_data[['Customer ID', 'last_updated', 'final_result']],
323                                           on=['Customer ID', 'last_updated'],
324                                           how='left').drop_duplicates()
325
326 # Aggregate final_result for pie chart
327 final_result_counts = unique_results['final_result'].value_counts().reset_index()
328 final_result_counts.columns = ['Final Result', 'Count']
329
330 # Calculate percentage
331 final_result_counts['Percentage'] = (final_result_counts['Count'] / final_result_counts['Count'].sum()) * 100
332
333 # Plot pie chart for final results
334 pie_chart = alt.Chart(final_result_counts).mark_arc().encode(
335     theta='Count:Q',
336     color='Final Result:N',
337     tooltip=[
338         alt.Tooltip('Final Result:N', title='Final Result'),
339         alt.Tooltip('Count:Q', title='Active Learners'),
340         alt.Tooltip('Percentage:Q', title='Percentage', format='.1f') # Format percentage with one decimal place
341     ]
342 ).properties(
343     width=400,
344     height=400
345 )
346
347 st.altair_chart(pie_chart, use_container_width=True)
348
349 # Expander for final result data
350 with st.expander("View Final Result Data"):
351     st.write(final_result_counts)
352     csv_final = final_result_counts.to_csv(index=False).encode('utf-8') # Convert to CSV
353     st.download_button(
354         label="Download Final Result Data",
355         data=csv_final,
356         file_name="Final_Results_LEAN.csv",
357         mime="text/csv",
358         help="Click here to download the final result data as a CSV file"
359     )
360
361 # Repeat for ELITE
362 elite_active_learners = filtered_df[filtered_df['title'] == 'ELITE']
363
364 # Get highest scores for ELITE
365 highest_scores_elite = elite_active_learners.loc[elite_active_learners.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
366 elite_active_learners_data = highest_scores_elite[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
367
368 # Stacked bar chart for ELITE
369 st.subheader("ELITE")
370 elite_filtered = filtered_df[filtered_df['title'] == 'ELITE']
371 elite_distribution = elite_filtered.groupby(['Test Name', 'typology']).agg({'Customer ID': 'nunique'}).reset_index()
372 elite_distribution.columns = ['Test Name', 'typology', 'Active Users']
373
374 # Calculate percentages for ELITE
375 total_active_users_per_test_elite = elite_distribution.groupby('Test Name')['Active Users'].transform('sum')
376 elite_distribution['Percentage'] = (elite_distribution['Active Users'] / total_active_users_per_test_elite * 100).round(2)
377
378 # Plot chart for ELITE
379 chart_elite = alt.Chart(elite_distribution).mark_bar().encode(
380     x='Test Name',
381     y='Active Users',
382     color='typology',
383     tooltip=[
384         alt.Tooltip('Test Name:N', title='Test Name'),
385         alt.Tooltip('typology:N', title='Typology'),
386         alt.Tooltip('Active Users:Q', title='Active Learners'),
387         alt.Tooltip('Percentage:Q', title='Percentage', format='.1f') # Format percentage with one decimal place
388     ]
389 ).properties(
390     width=600,
391     height=400
392 ).configure_mark(
393     opacity=0.8
394 ).configure_axis(
395     labelFontSize=12,
396     titleFontSize=14
397 ).configure_title(

```

```

398         fontSize=16
399     )
400
401     st.altair_chart(chart_elite, use_container_width=True)
402
403     # Data download for ELITE
404     with st.expander("Data ELITE"):
405         st.write(elite_distribution)
406         csv_elite = elite_distribution.to_csv(index=False).encode('utf-8')
407         st.download_button("Download Data", data=csv_elite, file_name="ELITE.csv", mime="text/csv", help="Click here to download the data as a CSV file")
408
409     # Display final result distribution
410     st.subheader("FINAL RESULT ELITE")
411     elite_active_learners = filtered_df[filtered_df['title'] == 'ELITE']
412
413     # Get highest scores for ELITE
414     highest_scores_elite = elite_active_learners.loc[elite_active_learners.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
415     elite_active_learners_data = highest_scores_elite[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
416
417     # Get unique combinations of Customer ID and Test Date
418     unique_combinations = elite_active_learners_data[['Customer ID', 'last_updated']].drop_duplicates()
419
420     # Merge back to get final_result
421     unique_results = unique_combinations.merge(elite_active_learners_data[['Customer ID', 'last_updated', 'final_result']],
422         on=['Customer ID', 'last_updated'],
423         how='left').drop_duplicates()
424
425     # Aggregate final_result for pie chart
426     final_result_counts = unique_results['final_result'].value_counts().reset_index()
427     final_result_counts.columns = ['Final Result', 'Count']
428
429     # Calculate percentage
430     final_result_counts['Percentage'] = (final_result_counts['Count'] / final_result_counts['Count'].sum()) * 100
431
432     # Plot pie chart for final results
433     pie_chart = alt.Chart(final_result_counts).mark_arc().encode(
434         theta='Count:Q',
435         color='Final Result:N',
436         tooltip=[
437             alt.Tooltip('Final Result:N', title='Final Result'),
438             alt.Tooltip('Count:Q', title='Active Learners'),
439             alt.Tooltip('Percentage:Q', title='Percentage', format='.1f') # Format percentage with one decimal place
440         ]
441     ).properties(
442         width=400,
443         height=400
444     )
445
446     st.altair_chart(pie_chart, use_container_width=True)
447
448     # Expander for final result data
449     with st.expander("View Final Result Data"):
450         st.write(final_result_counts)
451         csv_final = final_result_counts.to_csv(index=False).encode('utf-8') # Convert to CSV
452         st.download_button(
453             label="Download Final Result Data",
454             data=csv_final,
455             file_name="Final Results_ELITE.csv",
456             mime="text/csv",
457             help="Click here to download the final result data as a CSV file"
458         )
459
460     st.subheader("Genuine")
461     # Filter DataFrame untuk bundle_name yang spesifik
462     genuine_filtered = filtered_df[filtered_df['title'] == 'Genuine']
463
464     # Get highest scores
465     highest_scores = genuine_filtered.loc[genuine_filtered.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
466     genuine_active_learners_data = highest_scores[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
467
468     # Add a rank column from 1 to 9 based on total_score for each Customer ID and Test Date
469     genuine_active_learners_data['rank'] = genuine_active_learners_data.groupby(['Customer ID', 'last_updated'])['total_score'].rank(ascending=False, method='first')
470
471     # Filter ranks from 1 to 9
472     genuine_active_learners_data = genuine_active_learners_data[genuine_active_learners_data['rank'] <= 9]
473
474     # Create a select box for rank selection with a unique key
475     genuine_rank = st.selectbox("Select Top for Genuine", options=list(range(1, 10)), key='genuine_rank_select')
476
477     # Filter data based on the selected rank
478     filtered_rank_data = genuine_active_learners_data[genuine_active_learners_data['rank'] == genuine_rank]
479
480     # Count the number of unique users for each test name at the selected rank
481     user_count_by_test = filtered_rank_data.groupby('Test Name')['Customer ID'].nunique().reset_index()
482     user_count_by_test.columns = ['Test Name', 'Total Active Users']

```

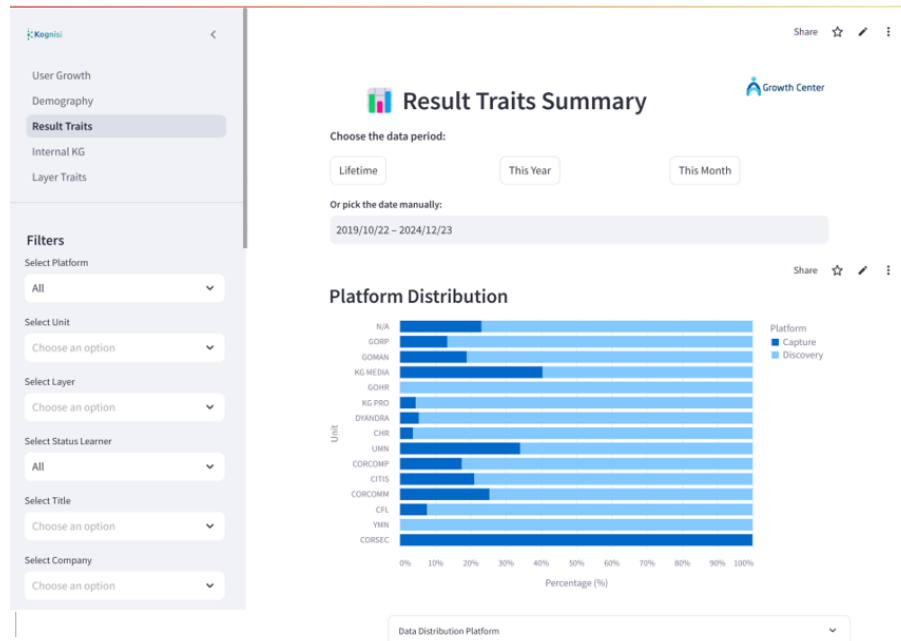
```

484 # Display the results
485 st.write(f"Genuine Top (genuine_rank)")
486 st.dataframe(user_count_by_test)
487
488 st.subheader("Astaka")
489 # Filter DataFrame untuk bundle_name yang spesifik
490 astaka_filtered = filtered_df[filtered_df['title'] == 'Astaka']
491
492 # Get highest scores
493 highest_scores = astaka_filtered.loc[astaka_filtered.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
494 astaka_active_learners_data = highest_scores[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
495
496 # Add a rank column from 1 to 6 based on total_score for each Customer ID and Test Date
497 astaka_active_learners_data['rank'] = astaka_active_learners_data.groupby(['Customer ID', 'last_updated'])['total_score'].rank(ascending=False, method='first').astype(int)
498
499 # Filter ranks from 1 to 6
500 astaka_active_learners_data = astaka_active_learners_data[astaka_active_learners_data['rank'] <= 6]
501
502 # Create a select box for rank selection with a unique key
503 astaka_rank = st.selectbox("Select Top for Astaka", options=list(range(1, 7)), key='astaka_rank_select')
504
505 # Filter data based on the selected rank
506 filtered_rank_data = astaka_active_learners_data[astaka_active_learners_data['rank'] == astaka_rank]
507
508 # Count the number of unique users for each test name at the selected rank
509 user_count_by_test = filtered_rank_data.groupby('Test Name')['Customer ID'].nunique().reset_index()
510 user_count_by_test.columns = ['Test Name', 'Total Active Users']
511
512 # Display the results
513 st.write(f"Astaka Top (astaka_rank)")
514 st.dataframe(user_count_by_test)
515
516 if selected_platform == 'Capture':
517 st.header('Active Learners - Capture', divider='gray')
518 total_count = filtered_df['email'].nunique()
519 internal_count = filtered_df[filtered_df['status_learner'] == 'Internal']['email'].nunique()
520 external_count = filtered_df[filtered_df['status_learner'] == 'External']['email'].nunique()
521
522 # Display metrics columns
523 col1, col2, col3 = st.columns(3)
524 with col1:
525 st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>Overall: <span style='color: red;'>{total_count:,}</span></strong></p>",
526             unsafe_allow_html=True)
527 with col2:
528 st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>Internal User: <span style='color: red;'>{internal_count:,}</span></strong></p>",
529             unsafe_allow_html=True)
530 with col3:
531 st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>External User: <span style='color: red;'>{external_count:,}</span></strong></p>",
532             unsafe_allow_html=True)
533
534 # Count unique emails per title
535 title_counts = filtered_df.groupby('title')['email'].nunique().reset_index()
536 title_counts.columns = ['title', 'active_learners']
537
538 # Sort the data from highest to lowest
539 title_counts = title_counts.sort_values(by='active_learners', ascending=False)
540
541 # Create the bar chart
542 chart = alt.Chart(title_counts).mark_bar().encode(
543     x=alt.X('active_learners:Q', title='Active Learners'),
544     y=alt.Y('title:O', title='Title', sort=alt.EncodingSortField(field='active_learners', order='descending')),
545     color=alt.Color('title:O', scales=alt.Scale(scheme='category10')),
546     tooltips=['title', 'active_learners']
547 ).properties(
548     title='Active Learners by Title'
549 )
550
551 # Add text layer to display active learners count on top of bars
552 text = chart.mark_text(
553     align='left',
554     baseline='middle',
555     color='black',
556     fontSize=12,
557     fontWeight='bold',
558     dy=-5 # Adjusts the vertical position of the text
559 ).encode(
560     text='active_learners:Q'
561 )
562
563 # Combine the bar chart and text layer
564 final_chart = chart + text
565
566 # Display the chart
567 st.altair_chart(final_chart, use_container_width=True)
568
569 # Data download for Top Test Capture
570 with st.expander("Top Test Capture"):
571 st.write(title_counts) # Now includes unit information
572 csv = title_counts.to_csv(index=False).encode('utf-8')
573 st.download_button(
574     "Download Data",
575     data=csv,
576     file_name="Top_Test_Capture.csv",
577     mime="text/csv",
578     help="Click here to download the data as a CSV file"
579 )
580
581

```

Gambar 3.14 Syntax Page Result Traits

Setelah syntax tersebut dijalankan maka halaman result traits pada dashboard streamlit terlihat seperti tampilan pada Gambar 3.15 Output Page Result Traits.



Gambar 3.15 Output Page Result Traits

3.2.5 Page Internal KG

Page internal KG merupakan halaman keempat pada dashboard streamlit discovery yang dimana pada halaman ini dilakukan pada minggu ke – 11 sampai dengan minggu ke – 14 dengan membuat perancangan terlebih dahulu setelah perancangan selesai dilakukan maka akan dilanjutkan dengan implementasi yang dimana pada halaman internal KG terdapat filter sesuai dengan yang diinginkan seperti bundle name dan test name. Pada dashboard streamlit discovery halaman internal KG juga menampilkan visualisasi seperti heatmap test taker per unit, pie chart overall result, bar chart result per unit, result per gender result per generation dan result per layer pada perusahaan Kompas Gramedia Group serta yang nantinya dapat difilter sesuai keinginan stakeholder. Pada

Gambar 3.16 Syntax Internal KG memperlihatkan codingan pembuatan halaman tersebut.

```

1 import pandas as pd
2 import streamlit as st
3 import plotly.express as px
4 from data_processing import finalize_data
5 from datetime import datetime
6
7 # Set the title and favicon in the browser tab
8 st.set_page_config(page_title='Internal KG', page_icon='🧠')
9
10 # Retrieve data from data_processing
11 df_discovery, df_sap, df_merged, df_combined_au_capture, df_capture_sheet3 = finalize_data()
12
13 # Display logo at the top of the sidebar
14 st.logo('kognisi_logo.png')
15
16 # Header with logos
17 col1, col2, col3 = st.columns([12, 1, 3])
18 with col1:
19     st.markdown("<h2 style='text-align: center;'>🧠 Internal KG</h2>", unsafe_allow_html=True)
20 with col3:
21     st.image('growth_center.png')
22
23 # Fungsi untuk memuat dan memproses data dengan caching
24 @st.cache_data
25 def load_and_process_data():
26     # Mengambil dan memproses data
27     df_discovery, df_sap, df_merged, df_combined_au_capture, df_capture_sheet3 = finalize_data()
28
29     # Memuat data dengan caching
30     merged_df = load_and_process_data()
31
32     # Memfilter data untuk pengguna internal
33     internal_df = df_merged[df_merged['status_learner'] == 'Internal']
34
35     # Dropdown untuk memilih Bundle Name tanpa opsi "All"
36     bundle_names = internal_df['title'].dropna().unique()
37     selected_bundle = st.sidebar.selectbox("Select Bundle Name", bundle_names)
38
39     # Memfilter nama tes berdasarkan Bundle Name yang terkait dengan platform Discovery
40     test_name_options = internal_df[
41         (internal_df['title'] == selected_bundle) &
42         (internal_df['platform'] == 'Discovery')
43     ][['Test Name']].dropna().unique()
44
45     if test_name_options.size == 0:
46         st.sidebar.warning("No Test Names available for the selected Bundle Name in the Discovery platform.")
47     else:
48         selected_test = st.sidebar.selectbox("Select Test Name", test_name_options)
49
50     # Menerapkan filter
51     internal_df = internal_df[internal_df['title'] == selected_bundle]
52     internal_df = internal_df[internal_df['Test Name'] == selected_test]
53
54     # Treemap untuk jumlah peserta tes per unit
55     treemap_data = internal_df.groupby(['Test Name', 'unit']).size().reset_index(name='Unit Count')
56     fig3 = px.treemap(
57         treemap_data,
58         title='Test Taker Per Unit',
59         path=[["Test Name", "unit"]],
60         values="Unit Count",
61         hover_data=["Unit Count"],
62         color="unit"
63     )
64     fig3.update_traces(textinfo='label+value', hovertemplate='<b>%[label]</b><br>User : %(value)')
65     fig3.update_layout(width=400, height=650)
66     st.plotly_chart(fig3, use_container_width=True)
67
68     # Diagram lingkaran untuk distribusi typology
69     typology_counts = internal_df['typology'].value_counts().reset_index()
70     typology_counts.columns = ['typology', 'User Count']
71     typology_counts = typology_counts.sort_values(by='User Count', ascending=False)
72     fig_typology = px.pie(
73         typology_counts,
74         names='typology',
75         values='User Count',
76         title='Overall Results'

```

```

77 )
78 fig_typology.update_traces(
79     hovertemplate='<b>%(label)</b><br>%(value)',
80     textinfo='percent'
81 )
82 fig_typology.update_layout(
83     legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1)
84 )
85 st.plotly_chart(fig_typology, use_container_width=True)
86
87 # Diagram batang bertumpuk untuk hasil per unit
88 # Grouping by unit and typology, counting users
89 typology_counts = internal_df.groupby(['unit', 'typology']).size().reset_index(name='User Count')
90
91 # Sorting units by total user count
92 unit_total_counts = typology_counts.groupby('unit')['User Count'].sum().reset_index()
93 unit_total_counts = unit_total_counts.sort_values(by='User Count', ascending=False)
94 sorted_units = unit_total_counts['unit']
95
96 # Creating the stacked bar chart
97 fig_stacked = px.bar(
98     typology_counts,
99     x='unit',
100     y='User Count', # No percentage, using actual user count
101     color='typology',
102     title='Result Per Unit',
103     labels={'User Count': 'User Count'}, # Update label to show actual counts
104     text='User Count',
105     height=400
106 )
107
108 # Updating layout to stack bars
109 fig_stacked.update_layout(
110     barmode='stack', # Stacked bars
111     xaxis_title='Unit',
112     yaxis_title='User Count', # Updated y-axis label to reflect actual counts
113     xaxis_categoryorder='array',
114     xaxis_categoryarray=sorted_units, # Sorting units by total count
115     yaxis=dict(tickformat=',', title='User Count') # Ensure y-axis shows raw counts
116 )
117
118 # Updating traces to show raw user count in the bars
119 fig_stacked.update_traces(texttemplate='%{text:.0f}') # Display raw count in the bars
120 st.plotly_chart(fig_stacked, use_container_width=True)
121
122 # Diagram batang bertumpuk untuk gender berdasarkan count
123 gender_counts = internal_df.groupby(['gender', 'typology']).size().reset_index(name='User Count')
124 gender_total_counts = gender_counts.groupby('gender')['User Count'].sum().reset_index()
125 gender_total_counts = gender_total_counts.sort_values(by='User Count', ascending=False)
126 sorted_genders = gender_total_counts['gender']
127
128 fig_gender = px.bar(
129     gender_counts,
130     x='gender',
131     y='User Count',
132     color='typology',
133     title='Result Per Gender',
134     labels={'User Count': 'User Count'},
135     text='User Count',
136     height=400
137 )
138 fig_gender.update_layout(
139     barmode='stack',
140     xaxis_title='Gender',
141     yaxis_title='User Count',
142     xaxis_categoryorder='array',
143     xaxis_categoryarray=sorted_genders,
144     yaxis=dict(tickformat=',', title='User Count')

```

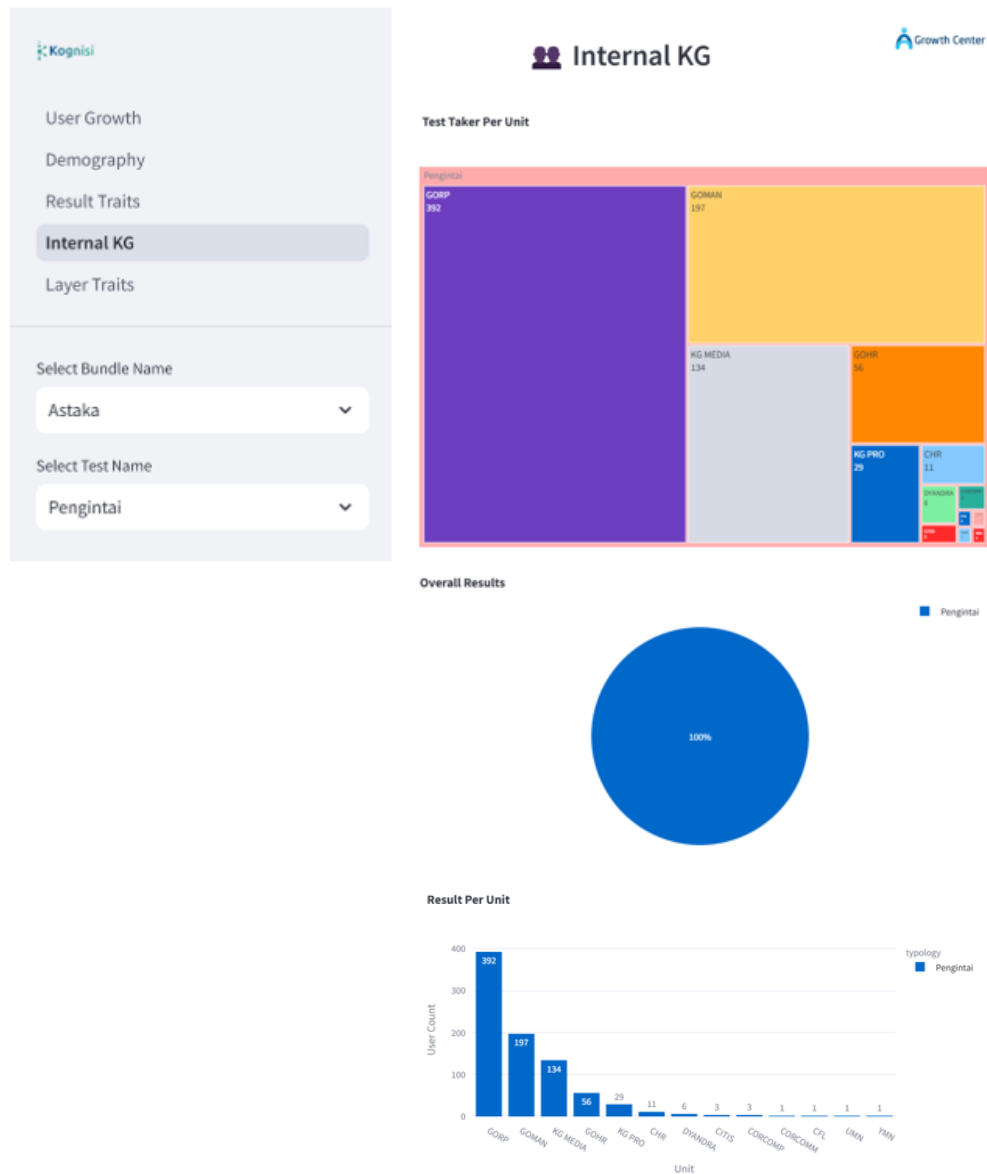
```

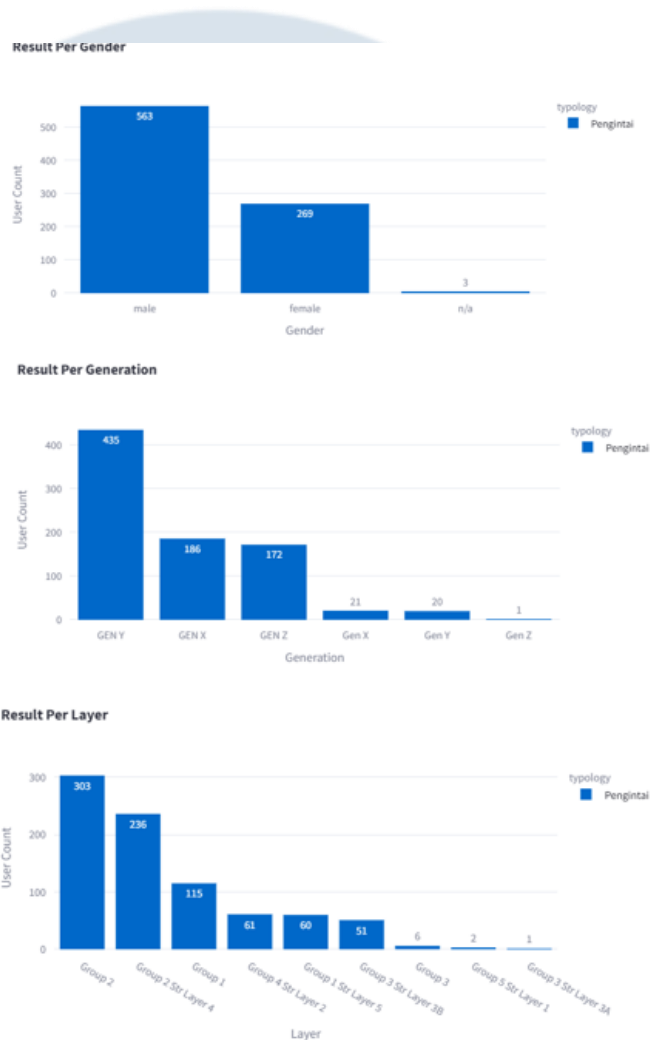
145 )
146 fig_gender.update_traces(texttemplate='%{text:.0f}')
147 st.plotly_chart(fig_gender, use_container_width=True)
148
149 # Diagram batang bertumpuk untuk generasi berdasarkan count
150 generation_counts = internal_df.groupby(['generation', 'typology']).size().reset_index(name='User Count')
151 generation_total_counts = generation_counts.groupby('generation')['User Count'].sum().reset_index()
152 generation_total_counts = generation_total_counts.sort_values(by='User Count', ascending=False)
153 sorted_generations = generation_total_counts['generation']
154
155 fig_generation = px.bar(
156     generation_counts,
157     x='generation',
158     y='User Count',
159     color='typology',
160     title='Result Per Generation',
161     labels={'User Count': 'User Count'},
162     text='User Count',
163     height=400
164 )
165 fig_generation.update_layout(
166     barmode='stack',
167     xaxis_title='Generation',
168     yaxis_title='User Count',
169     xaxis_categoryorder='array',
170     xaxis_categoryarray=sorted_generations,
171     yaxis=dict(tickformat=',', title='User Count')
172 )
173 fig_generation.update_traces(texttemplate='%{text:.0f}')
174 st.plotly_chart(fig_generation, use_container_width=True)
175
176 # Diagram batang bertumpuk untuk layer berdasarkan count
177 layer_counts = internal_df.groupby(['layer', 'typology']).size().reset_index(name='User Count')
178 layer_total_counts = layer_counts.groupby('layer')['User Count'].sum().reset_index()
179 layer_total_counts = layer_total_counts.sort_values(by='User Count', ascending=False)
180 sorted_layers = layer_total_counts['layer']
181
182 fig_layer = px.bar(
183     layer_counts,
184     x='layer',
185     y='User Count',
186     color='typology',
187     title='Result Per Layer',
188     labels={'User Count': 'User Count'},
189     text='User Count',
190     height=400
191 )
192 fig_layer.update_layout(
193     barmode='stack',
194     xaxis_title='Layer',
195     yaxis_title='User Count',
196     xaxis_categoryorder='array',
197     xaxis_categoryarray=sorted_layers,
198     yaxis=dict(tickformat=',', title='User Count')
199 )
200 fig_layer.update_traces(texttemplate='%{text:.0f}')
201 st.plotly_chart(fig_layer, use_container_width=True)
202

```

Gambar 3.16 Syntax Page Internal KG

Setelah syntax tersebut dijalankan maka halaman Internal KG pada dashboard streamlit terlihat seperti tampilan pada Gambar 3.17 Output Page Internal KG.





Gambar 3.17 Output Page Internal KG

3.2.6 Page layer Traits

Page layer traits merupakan halaman kelima pada dashboard streamlt discovery yang dimana pada halaman ini dilakukan pada minggu ke – 15 sampai dengan minggu ke – 19 dengan membuat perancangan terlebih dahulu setelah perancangan selesai dilakukan maka akan dilanjutkan dengan implementasi yang dimana pada halaman layer traits terdapat filter sesuai dengan yang diinginkan seperti layer, unit, dan bundle

name. Pada dashboard streamlit discovery halaman layer traits juga menampilkan visualisasi seperti tabel summary setiap traits, bar chart Genuine dan bar chart Astaka pada perusahaan Kompas Gramedia Group serta yang nantinya dapat difilter sesuai keinginan stakeholder. Pada Gambar 3.18 Syntax Layer Traits memperlihatkan codingan pembuatan halaman tersebut.



```

1 import pandas as pd
2 import streamlit as st
3 import altair as alt
4 from data_processing import finalize_data
5
6 # Setting page title and favicon
7 st.set_page_config(page_title='Layer Traits Summary')
8
9 # Adding logo and header to the sidebar and main page
10 st.logo('kognisi_logo.png')
11 col1, col2, col3 = st.columns([1, 12, 3])
12 with col2:
13     st.markdown("<h2 style='text-align: center;'>KG EMPLOYEE LAYER TRAITS SUMMARY</h2>", unsafe_allow_html=True)
14     st.image('growth_center.png')
15
16 # Dashboard guidelines
17 st.markdown("""
18     ## Panduan Layer Traits Summary
19     - **Layer 1** adalah Group 5 Str Layer 1.
20     - **Layer 2** adalah Group 4 Str Layer 2.
21     - **Layer 3** adalah Group 3 Str Layer 3A dan Group 3 Str Layer 3B.
22     - **Layer 4** adalah Group 2 Str Layer 4.
23     - **Layer 5** adalah Group 1 Str Layer 5.
24     - **Non Struktural** adalah Group 1, Group 2, Group 3, Group 4 dan Group 5
25 """)
26
27 # Caching function for loading and processing data
28 @st.cache_data
29 def load_and_process_data():
30     # Load and process data
31     df_discovery, df_sup, df_merged, df_combined_au_capture, df_capture_sheet3 = finalize_data()
32     return df_merged
33
34 # Load data with caching
35 df_merged = load_and_process_data()
36
37 # Filter data for internal users
38 internal_df = df_merged[df_merged['status_learner'] == 'Internal']
39
40 # Mapping for grouping layers
41 layer_mapping = {
42     'Group 5 Str Layer 1': 'Layer 1',
43     'Group 4 Str Layer 2': 'Layer 2',
44     'Group 3 Str Layer 3B': 'Layer 3',
45     'Group 3 Str Layer 3A': 'Layer 3',
46     'Group 2 Str Layer 4': 'Layer 4',
47     'Group 1 Str Layer 5': 'Layer 5',
48     'Group 1': 'Non Struktural',
49     'Group 2': 'Non Struktural',
50     'Group 3': 'Non Struktural',
51     'Group 4': 'Non Struktural',
52     'Group 5': 'Non Struktural'
53 }
54
55 # Apply the mapping to create a new column for grouped layers
56 internal_df['layer_group'] = internal_df['layer'].map(layer_mapping)
57
58 # Sidebar filters with multiselect
59 st.sidebar.header("Filter Options")
60 selected_layers = st.sidebar.multiselect(
61     "Select Layers",
62     options=['Layer 1', 'Layer 2', 'Layer 3', 'Layer 4', 'Layer 5', 'Non Struktural'],
63     default=[]
64 )
65
66 # Filter based on selected layers
67 df_filtered = internal_df[internal_df['layer_group'].isin(selected_layers)] if selected_layers else internal_df
68
69 # Add unit filter to the sidebar
70 selected_units = st.sidebar.multiselect(
71     "Select Unit",
72     options=df_filtered['unit'].unique(),
73     default=[]
74 )
75
76 # Apply unit filter if units are selected
77 if selected_units:
78     df_filtered = df_filtered[df_filtered['unit'].isin(selected_units)]

```



```

82 # Active learners by bundle
83 bundle_names = ['GI', 'LEAN', 'ELITE', 'Genuine', 'Astaka']
84
85 # Create filtered dataframe for active learners
86 if 'title' in df_filtered.columns:
87     df_active_learners = df_filtered.groupby(['Customer ID', 'last_updated', 'title']).size().reset_index(name='test_count')
88
89     # Count active learners per bundle
90     bundle_counts = {bundle: df_active_learners[df_active_learners['title'] == bundle]['Customer ID'].nunique() for bundle in bundle_names}
91
92     # Display active learners counts
93     st.markdown("<h3>ACTIVE LEARNERS</h3>", unsafe_allow_html=True)
94     col1, col2, col3, col4, col5 = st.columns(5)
95
96     for i, bundle in enumerate(bundle_names):
97         with eval(f'col{i + 1}'):
98             st.markdown(f"<p style='font-size: 20px; text-align: center;'><strong>{bundle}</strong>: <span style='color: red;'>
99                 {bundle_counts[bundle]:,}</span></strong></p>", unsafe_allow_html=True)
100
101
102 # 1. Get the latest test results for each email and Test Name
103 latest_test_results = df_filtered.loc[df_filtered.groupby(['email', 'Test Name'])['last_updated'].idxmax()]
104
105 # 2. Count participants based on bundle_name
106 participant_counts = df_filtered.groupby('title')['email'].nunique().reset_index()
107 participant_counts.columns = ['title', 'jumlah_partisipan']
108
109 # 3. Count users based on typology from the latest results
110 typology_user_counts = latest_test_results.groupby('typology')['email'].nunique().reset_index()
111 typology_user_counts.columns = ['typology', 'jumlah']
112
113 # 4. Get unique Test Name for each bundle_name
114 test_names = df_filtered[['title', 'Test Name']].drop_duplicates()
115
116 # 5. Get all unique typology results for each Test Name
117 typology_results = latest_test_results.groupby(['Test Name', 'typology'])['email'].nunique().reset_index()
118 typology_results.columns = ['Test Name', 'typology', 'jumlah']
119
120 # 6. Calculate percentages for each typology per Test Name
121 total_users_per_test = typology_results.groupby('Test Name')['jumlah'].transform('sum')
122 typology_results['persentase'] = (typology_results['jumlah'] / total_users_per_test * 100).round(2)
123
124 # 7. Combine results into one DataFrame
125 result_df = pd.merge(participant_counts, test_names, on='title', how='left')
126 result_df = pd.merge(result_df, typology_results, on='Test Name', how='left')
127
128 # 8. Add rows for Overall ELITE based on filtered results
129 final_results_elite = latest_test_results[latest_test_results['title'] == 'ELITE'].groupby('final_result')['email'].nunique().reset_index()
130 overall_elite_rows = [
131     'title': 'ELITE',
132     'jumlah_partisipan': participant_counts.loc[participant_counts['title'] == 'ELITE', 'jumlah_partisipan'].values[0],
133     'Test Name': 'Overall ELITE',
134     'typology': row['final_result'],
135     'jumlah': row['email'],
136     'persentase': (row['email'] / participant_counts.loc[participant_counts['title'] == 'ELITE', 'jumlah_partisipan'].values[0] * 100).round(2)
137 ] for _, row in final_results_elite.iterrows()
138
139 overall_elite_df = pd.DataFrame(overall_elite_rows)
140
141 # 9. Add rows for Overall LEAN based on filtered results
142 final_results_lean = latest_test_results[latest_test_results['title'] == 'LEAN'].groupby('final_result')['email'].nunique().reset_index()
143 overall_lean_rows = [
144     'title': 'LEAN',
145     'jumlah_partisipan': participant_counts.loc[participant_counts['title'] == 'LEAN', 'jumlah_partisipan'].values[0],
146     'Test Name': 'Overall LEAN',
147     'typology': row['final_result'],
148     'jumlah': row['email'],
149     'persentase': (row['email'] / participant_counts.loc[participant_counts['title'] == 'LEAN', 'jumlah_partisipan'].values[0] * 100).round(2)
150 ] for _, row in final_results_lean.iterrows()
151
152 overall_lean_df = pd.DataFrame(overall_lean_rows)
153
154 # 10. Combine result_df with overall_elite_df and overall_lean_df
155 result_df = pd.concat([result_df, overall_elite_df, overall_lean_df], ignore_index=True)
156
157 # 11. Filter for desired bundles: GI, ELITE, LEAN
158 filtered_bundles = ['GI', 'ELITE', 'LEAN']
159 result_df = result_df[result_df['title'].isin(filtered_bundles)]
160

```

```

161 # 12. Set categorical order and sort
162 result_df['title'] = pd.Categorical(result_df['title'], categories=filtered_bundles, ordered=True)
163 result_df = result_df.sort_values(['title', 'Test Name']).reset_index(drop=True)
164
165 # 13. Combine number of users with result_df without additional calculations
166 combined_result_df = pd.merge(result_df, typology_user_counts[['typology']], on='typology', how='left')
167
168 # 14. Create a custom sort order for 'Test Name'
169 def custom_sort_order(test_name):
170     if test_name in ["Mindset", "Overall ELITE", "Overall LEAN"]:
171         return 0
172     return 1
173
174 combined_result_df['sort_order'] = combined_result_df['Test Name'].apply(custom_sort_order)
175
176 # Sort by bundle_name and custom sort order
177 combined_result_df = combined_result_df.sort_values(['title', 'sort_order', 'Test Name']).reset_index(drop=True)
178
179 # 15. Bundle filter
180 st.sidebar.header("Bundle Name Filter")
181 selected_bundle = st.sidebar.selectbox("Select Bundle Name", options=filtered_bundles, index=0)
182
183 # Filter combined_result_df based on selected bundle
184 if selected_bundle == 'LEAN':
185     # Hapus baris dengan typology 'The Olympian' dan 'The Spectator' untuk bundle LEAN
186     combined_result_df = combined_result_df[
187         (combined_result_df['title'] == selected_bundle) &
188         (~combined_result_df['typology'].isin(['The Olympian', 'The Spectator']))
189     ]
190 elif selected_bundle == 'ELITE':
191     # Hapus baris dengan typology 'Citizen' dan 'Governor' untuk bundle ELITE
192     combined_result_df = combined_result_df[
193         (combined_result_df['title'] == selected_bundle) &
194         (~combined_result_df['typology'].isin(['Citizen', 'Governor']))
195     ]
196 else:
197     # Filter untuk bundle lain
198     combined_result_df = combined_result_df[combined_result_df['title'] == selected_bundle]
199
200 # 16. Display results in Streamlit, excluding 'sort_order'
201 selected_layers_title = ', '.join(selected_layers) if selected_layers else 'All Layers'
202 selected_units_title = ', '.join(selected_units) if selected_units else 'All Units'
203 st.subheader(f"{selected_bundle} Traits Summary for {selected_layers_title} ({selected_units_title})")
204 st.dataframe(combined_result_df.drop(columns=['sort_order']))
205 # Menghitung total pengguna untuk setiap Test Name
206 total_users_per_test = combined_result_df.groupby('Test Name')['jumlah'].transform('sum')
207
208 # Menghitung persentase untuk setiap typology pada masing-masing Test Name
209 combined_result_df['persentase'] = (combined_result_df['jumlah'] / total_users_per_test * 100).round(2)
210
211 # Membuat stacked bar chart dengan warna otomatis untuk setiap nilai typology
212 stacked_bar_chart = alt.Chart(combined_result_df).mark_bar().encode(
213     x=alt.X('persentase:Q', title='Persentase Active Learners'),
214     y=alt.Y('Test Name:N', title='Test Name', sort='-x'), # Menggunakan Test Name untuk sumbu Y
215     color=alt.Color(
216         'typology:N',
217         title='Typology',
218         scale=alt.Scale(scheme='category10') # Palet warna otomatis untuk berbagai nilai typology
219     ),
220     tooltip=[
221         alt.Tooltip('Test Name:N', title='Test Name'),
222         alt.Tooltip('jumlah:Q', title='Jumlah'),
223         alt.Tooltip('typology:N', title='Typology'),
224         alt.Tooltip('persentase:Q', title='Persentase (%)')
225     ]
226 )
227 .properties(
228     title=f'Distribusi {selected_layers_title} Berdasarkan traits {selected_bundle} di {selected_units_title}',
229     width=600,
230     height=400
231 )
232 .configure_axis(
233     labelFontSize=12,
234     titleFontSize=14
235 )
236
237 # Menampilkan stacked bar chart di Streamlit
238 st.altair_chart(stacked_bar_chart, use_container_width=True)
239
240 # Filter for 'Genuine' bundle

```

```

240 genuine_filtered = df_filtered[df_filtered['title'] == 'Genuine']
241
242 # Get the highest scores
243 highest_scores = genuine_filtered.loc[genuine_filtered.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
244
245 # Select relevant columns for the genuine active learners data
246 genuine_active_learners_data = highest_scores[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
247
248 # Add a rank column from 1 to 9 based on total_score for each Customer ID and Test Date
249 genuine_active_learners_data['rank'] = genuine_active_learners_data.groupby(['Customer ID', 'last_updated'])['total_score'].
250 rank(ascending=False, method='first').astype(int)
251
252 # Filter ranks from 1 to 9
253 genuine_active_learners_data = genuine_active_learners_data[genuine_active_learners_data['rank'] <= 9]
254
255 # Add a select box for filtering by rank
256 selected_rank = st.selectbox("Select Rank", options=range(1, 10), index=0)
257
258 # Filter the data based on the selected rank
259 filtered_data_by_rank = genuine_active_learners_data[genuine_active_learners_data['rank'] == selected_rank]
260
261 # Get the latest Test Date results for each email
262 latest_results = filtered_data_by_rank.loc[filtered_data_by_rank.groupby('email')['last_updated'].idxmax()]
263
264 # Calculate jumlah_partisipan (total unique email for the bundle)
265 total_participants = latest_results['email'].nunique()
266
267 st.subheader(f"Genuine TOP {selected_rank} Traits Summary for {selected_layers_title} ({selected_units_title})")
268
269 # Prepare the data for display
270 summary_data = (
271     latest_results
272     .groupby(['title', 'Test Name']) # Group by bundle_name and Test Name
273     .agg(
274         jumlah_partisipan=('email', 'nunique'), # Count of unique email based only on Test Name
275         jumlah=('email', 'count') # Count of total entries based on email per Test Name
276     )
277     .reset_index()
278 )
279
280 # Add the total_participants to the summary_data
281 summary_data['jumlah_partisipan'] = total_participants
282
283 # Calculate the percentage
284 summary_data['persentase'] = (summary_data['jumlah'] / summary_data['jumlah_partisipan'] * 100).round(2)
285
286 # Reorder the columns as specified
287 summary_data = summary_data[['title', 'jumlah_partisipan', 'Test Name', 'jumlah', 'persentase']]
288 st.dataframe(summary_data)
289
290 # Visualization: Simple bar chart for jumlah
291 bar_chart = alt.Chart(summary_data).mark_bar().encode(
292     x=alt.X('Test Name:O', title='Test Name'),
293     y=alt.Y('jumlah:Q', title='Jumlah'),
294     tooltips=['Test Name', 'jumlah', 'persentase']
295 ).properties(
296     title=f'Distribusi Genuine TOP {selected_rank}'
297 )
298
299 # Display the bar chart
300 st.altair_chart(bar_chart, use_container_width=True)
301
302 # Filter for 'Astaka' bundle
303 astaka_filtered = df_filtered[df_filtered['title'] == 'Astaka']
304
305 # Get the highest scores
306 highest_scores = astaka_filtered.loc[astaka_filtered.groupby(['email', 'last_updated', 'Test Name'])['total_score'].idxmax()]
307
308 # Select relevant columns for the genuine active learners data
309 astaka_active_learners_data = highest_scores[['name', 'email', 'Customer ID', 'title', 'last_updated', 'Test Name', 'total_score', 'final_result']]
310
311 # Add a rank column from 1 to 6 based on total_score for each Customer ID and Test Date
312 astaka_active_learners_data['rank'] = astaka_active_learners_data.groupby(['Customer ID', 'last_updated'])['total_score'].
313 rank(ascending=False, method='first').astype(int)
314
315 # Filter ranks from 1 to 6
316 astaka_active_learners_data = astaka_active_learners_data[astaka_active_learners_data['rank'] <= 6]
317
318 # Add a select box for filtering by rank
319 selected_rank = st.selectbox("Select Rank", options=range(1, 7), index=0)
320
321 # Filter the data based on the selected rank
322 filtered_data_by_rank = astaka_active_learners_data[astaka_active_learners_data['rank'] == selected_rank]
323
324 # Get the latest Test Date results for each email
325 latest_results = filtered_data_by_rank.loc[filtered_data_by_rank.groupby('email')['last_updated'].idxmax()]

```

```

325 latest_results = filtered_data_by_rank.loc[filtered_data_by_rank.groupby('email')['last_updated'].idxmax()]
326
327 # Calculate jumlah_partisipan (total unique email for the bundle)
328 total_participants = latest_results['email'].nunique()
329
330 # Prepare the data for display
331 summary_data = (
332     latest_results
333     .groupby(['title', 'Test Name']) # Group by bundle_name and Test Name
334     .agg(
335         jumlah_partisipan=('email', 'nunique'), # Count of unique email based only on Test Name
336         jumlah=('email', 'count') # Count of total entries based on email per Test Name
337     )
338     .reset_index()
339 )
340
341 # Add the total_participants to the summary_data
342 summary_data['jumlah_partisipan'] = total_participants
343
344 # Calculate the percentage
345 summary_data['persentase'] = (summary_data['jumlah'] / summary_data['jumlah_partisipan'] * 100).round(2)
346
347 # Reorder the columns as specified
348 summary_data = summary_data[['title', 'jumlah_partisipan', 'Test Name', 'jumlah', 'persentase']]
349
350 # Display the filtered data in Streamlit
351 st.subheader(f"Astaka TOP {selected_rank} Traits Summary for {selected_layers_title} ({selected_units_title})")
352 st.dataframe(summary_data)
353
354 # Visualization: Simple bar chart for jumlah
355 bar_chart = alt.Chart(summary_data).mark_bar().encode(
356     x=alt.X('Test Name:O', title='Test Name'),
357     y=alt.Y('jumlah:Q', title='Jumlah'),
358     tooltip=['Test Name', 'jumlah', 'persentase']
359 ).properties(
360 )
361
362 # Display the bar chart
363 st.altair_chart(bar_chart, use_container_width=True)
364

```

Gambar 3.18 Syntax Page Layer Traits

Setelah syntax tersebut dijalankan maka halaman layer traits pada dashboard streamlit terlihat seperti tampilan pada Gambar 3.19 Output Page Layer Traits.



Bundle Name Filter

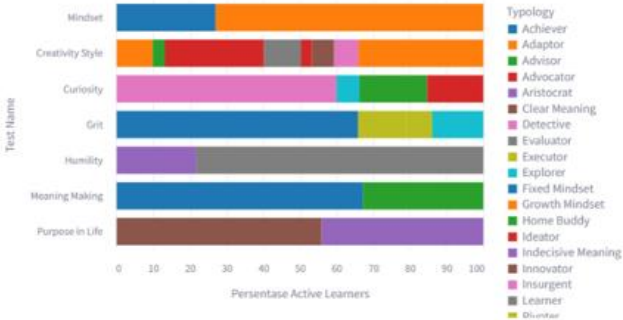
Select Bundle Name

GI

GI Traits Summary for All Layers (All Units)

	title	jumlah_partisipan	Test Name	typology	jumlah	persentase
14	GI	954	Grit	Achiever	628	65.83
15	GI	954	Grit	Executor	124	13
16	GI	954	Grit	Pivoter	69	7.23
17	GI	954	Grit	Planner	133	13.94
18	GI	954	Humility	Aristocrat	207	21.7
19	GI	954	Humility	Learner	747	78.3
20	GI	954	Meaning Making	Purpose Driven	641	67.19
21	GI	954	Meaning Making	Task Driven	313	32.81
22	GI	954	Purpose in Life	Clear Meaning	533	55.87
23	GI	954	Purpose in Life	Indecisive Meaning	421	44.13

Distribusi All Layers Berdasarkan traits GI di All Units

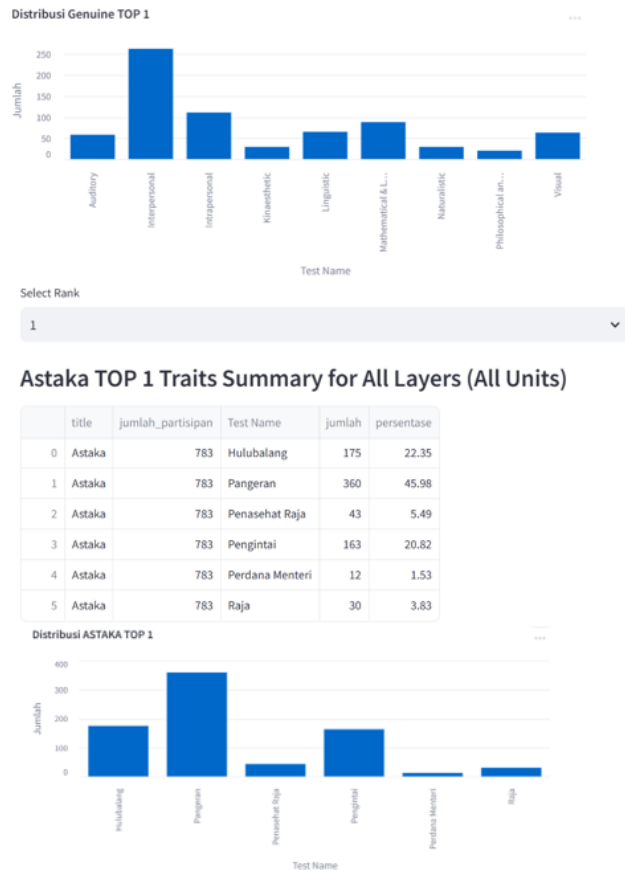


Select Rank

1

Genuine TOP 1 Traits Summary for All Layers (All Units)

	title	jumlah_partisipan	Test Name	jumlah	persentase
0	Genuine	726	Auditory	58	7.99
1	Genuine	726	Interpersonal	263	36.23
2	Genuine	726	Intrapersonal	111	15.29
3	Genuine	726	Kinaesthetic	29	3.99
4	Genuine	726	Linguistic	65	8.95
5	Genuine	726	Mathematical & Logical	88	12.12
6	Genuine	726	Naturalistic	29	3.99
7	Genuine	726	Philosophical and Ethical	20	2.75
8	Genuine	726	Visual	63	8.68



Gambar 3.19 Output Page Layer Traits

3.3 Kendala yang Ditemukan

Selama proses melaksanakan program pratik kerja magang, peserta magang tentunya menghadapi beberapa hambatan ataupun kesulitan baik dalam menjalankan tugas harian hingga project dashboard streamlit discovery yang dikerjakan. Berikut ini beberapa kendala atau hambatan yang dihadapi peserta magang sebagai seorang internship People Data Analyst di perusahaan Kompas Gramedia Group yaitu sebagai berikut:

- Waktu diskusi menjadi salah satu hambatan selama proses pratik kerja magang hal ini dikarena untuk dapat berdiskusi dengan mentor diperlukan penjadwalan melalui google kalender karena sistem kerja

pada perusahaan Kompas Gramedia Group yang hybrid dengan jadwal masuk kantor hanya 2 hari saja dalam 1 minggu sehingga peserta magang kesulitan untuk berdiskusi lebih detail dengan mentor baik itu untuk mendapatkan arahan maupun feedback.

- b. Istilah – istilah pada data yang akan diolah sulit untuk dimengerti seperti group description ternyata adalah unit dari Kompas Gramedia Group masih banyak lagi istilah – istilah lainnya sehingga nantinya akan terjadi kesalahpahaman dalam penjelasan istilah pada data yang akan digunakan.
- c. Internet perusahaan Kompas Gramedia Group yang sedikit kurang stabil sehingga kesulitan bagi seorang peserta magang untuk dapat menyelesaikan pekerjaan yang diberikan pada saat itu.

3.4 Solusi atas kendala yang ditemukan

Berdasarkan dari hambatan dan kesulitan yang dihadapi selama program pratik kerja magang pada perusahaan Kompas Gramedia Group terdapat solusi yang dimana mampu mengatasi seluruh hambatan yang dihadapi peserta magang sebagai seorang Internship People Data Analyst yaitu sebagai berikut :

- a. Solusi terkait dengan waktu untuk dapat berdiskusi dengan mentor biasanya peserta magang akan melakukan chat terlebih dahulu melalui platform whatsapp untuk dapat menanyakan waktu kosong yang dimiliki mentor agar nantinya dapat di set pada google calendar.
- b. Solusi terkait dengan istilah pada data peserta magang tentunya akan mengexplore terlebih dahulu seluruh informasi yang berkaitan dengan istilah tersebut dan kemudian disesuaikan kembali jika sudah biasanya akan dilakukan cross check kembali ke mentor dari setiap istilah tersebut agar tidak terjadi miss komunikasi.

- c. Solusi terkait dengan internet perusahaan yang kurang stabil peserta magang akan menggunakan internet pribadi agar dapat menyelesaikan ataupun melanjutkan pekerjaan – pekerjaan.

