Operation Analytics and Investigating Metric Spike

- **Samruddhi Pawar**

Project Description

In today's data-driven world, businesses must analyse their operations and metrics to stay competitive. This project aims to provide insights into two different scenarios of operation and metric analytics. One of the primary goals of the project is to provide a dashboard that displays key performance indicators (KPIs). The dashboard will provide a real-time view of KPIs to help managers make data-driven decisions. Both case studies will provide valuable insights into two different scenarios and will enable businesses to make data-driven decisions, improve operational efficiency, and enhance customer satisfaction.

Operational Analytics is a crucial process that involves analysing a company's end-to-end operations. This analysis helps identify areas for improvement within the company. A Data Analyst needs to work closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect.

One of the key aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales. These questions need to be answered daily, making it crucial to understand how to investigate these metric spikes.

Various datasets and tables are been provided, and the task is to derive insights from this data to answer questions posed by different departments within the company. The goal is to use advanced SQL skills to analyse the data and provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.

Case Study 1: Operation Analytics of Job Data

The first case study involves analysing job data to improve operational efficiency. It focuses on job data analysis, where I will calculate the number of jobs reviewed per hour. Additionally, I will calculate the 7-day rolling average of throughput, which represents the number of events happening per second, and discuss the preference between daily metrics. Furthermore, I will determine the percentage share of each language in the last 30 days and provide a method to display duplicate rows if identified in the data.

Case Study 2: Investigating Metric Spike

The second case study involves analysing product to draw insights and display key performance indicators (KPIs). It revolves around investigating metric spikes. I will analyse user engagement, user growth, weekly retention of user sign-up cohort, and email engagement metrics. These metrics will help evaluate the activeness of users, measure product growth, understand user retention, and assess email service engagement. Through this analysis, I will identify areas of improvement and provide valuable insights to the respective teams. Throughout the project, I will thoroughly examine the data tables and consider factors such as event types, user actions, and time spent to ensure accurate analysis and meaningful results.

Approach:

My approach to tackling the problem involved several key steps. First, I thoroughly examined and understood the provided dataset in the project. This involved analysing the tables and their respective columns, as well as identifying the specific tasks I needed to perform.
To work with the dataset, I utilized MySQL as my database management system. This allowed me to effectively query and manipulate the data to extract the necessary insights. Once I had a clear understanding of the dataset and the tasks at hand, I proceeded to analyse the data step by step. This involved applying various SQL queries, aggregations, filters and joins to obtain the required information and perform the requested calculations.

Throughout the analysis process, I closely monitored the results and ensured they aligned with the expected outcomes. I paid attention to any patterns, trends, or anomalies observed in the data, which helped me derive meaningful insights and draw relevant conclusions. The process involved careful examination, analysis, and interpretation of the data to fulfil the project requirements and provide accurate and meaningful results.

1) Prepare & Analyse
   First, the data was prepared and analysed using SQL. Various SQL functions like SELECT, WHERE, GROUP BY, JOIN, etc. are used to extract meaningful information from the dataset.
2) Identify KPIs
   Once the data is cleaned and transformed, the next step is to identify Key Performance Indicators (KPIs) that are relevant to the business. These KPIs include throughput, user engagement, user retention, etc.
3) Visualize Data
   The final step is to create interactive charts and dashboards using Tableau. Tableau helps to create interactive visualizations that help in identifying trends and patterns.

Approach for Case Study 1:

In this project, I worked with the "job_data" table to analyse job-related information. I calculated the number of jobs reviewed per hour per day for November 2020 to understand the workload patterns. For throughput, I calculated the 7-day rolling average to smoothen the event frequency trend. Analysing the percentage share of each language in the last 30 days helped identify the dominant languages used in job content. To address duplicate rows, I used queries to display rows with identical values. Overall, this analysis provided insights into job review activity, event frequency, language distribution, and data quality.

Approach for Case Study 2:

In this project, I analysed data from the "users," "events," and "email_events" tables. Calculating weekly user engagement involved grouping events and counting distinct users. User growth was determined by analysing user creation dates. Weekly retention was assessed by comparing sign-up dates with subsequent interactions. Email engagement metrics were analysed using email-specific events. These analyses provided insights into user activity, product growth, user retention, and email service effectiveness. By investigating metric spikes, I helped identify reasons for changes in engagement, sales, and

other metrics. These insights supported operational improvements and decision-making across different teams within the company.

Tech-Stack Used:

1) ***MS Excel:***
   It is used as a preparation base for data. The software enables users to seamlessly import and organize data from various sources, facilitating a structured foundation for analysis. Data cleaning becomes an intuitive process with Excel's capabilities, allowing users to identify and rectify issues like missing values and duplicates.
2) ***MySQL:***
   For Case Study 1, I leveraged SQL as the primary tool for data analysis. By using SQL queries, I interacted with the "job_data" table to extract relevant information. SQL allowed me to calculate the number of jobs reviewed per hour per day for November 2020, as well as the 7-day rolling average of throughput. Additionally, I used SQL to determine the percentage share of each language in the last 30 days and identify any duplicate rows in the data. With SQL's querying capabilities, I could efficiently analyse the job data and derive meaningful insights.
   In Case Study 2, I relied solely on SQL for data analysis. I utilized SQL queries to interact with the database tables, including "users," "events," and "email_events." SQL allowed me to retrieve relevant data, perform calculations, and generate insights. With SQL's powerful querying capabilities, I was able to analyse user engagement, user growth, weekly retention, and email engagement metrics. By utilizing SQL as the primary tool, I could effectively handle the data and derive valuable insights from the available dataset.
3) ***Tableau:***
   It was primarily used to create interactive charts and dashboards. Using the built-in data connections, I was able to work more efficiently, even when collating data from multiple sources and file types. Additionally, files with the same column names can also be combined into one data source, saving time on copying and pasting. Tableau's drag-and-drop interface is intuitive and dynamic, allowing for more flexibility and experimentation.

Data Overview:

Case Study 1

The dataset provides details about the review process of applicants for various job profiles.

The Column Details of the Dataset are:

1. **job_id:** Unique identifier of jobs
2. **actor_id:** Unique identifier of actor
3. **event:** The type of event (decision/skip/transfer)

4. **language:** The Language of the content
5. **time_spent:** Time spent to review the job in seconds
6. **org:** The Organization of the actor
7. **ds:** The date in the format yyyy/mm/dd (stored as text).

Case Study 2:
There were three tables that I worked with:
1. users: Contains one row per user, with descriptive information about that user's account.
2. events: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).
3. email_events: Contains events specific to the sending of emails.

Source of data:
https://docs.google.com/spreadsheets/d/1yYB0LLkteyXzptJ8iHFy8Z_etns3o4RGoVG4Wc-i6nU/edit?usp=sharing

Database Generation:

1) ***Case Study 1:***

A. Create Database:

```
create database job_db;
```

B. Create Table:

1. *Job-data table:*
   This SQL code creates a table named **job_data** to store information about users. Here's a breakdown of each part of the code:

```
create table job_data
(
job_id int,
actor_id int,
event varchar(50),
language varchar(50),
time_spent time,
org varchar(100),
ds date
);
insert into job_data (job_id, actor_id, event, language, time_spent, org, ds)
values
('21', '1001', 'skip', 'English', '15', 'A', '2020-11-30'),
('22', '1006', 'transfer', 'Arabic', '25', 'B', '2020-11-30'),
('23','1003', 'decision', 'Persian', '20', 'C', '2020-11-29'),
('23', '1005', 'transfer', 'Persian', '22', 'D', '2020-11-28'),
('25', '1002', 'decision', 'Hindi', '11', 'B', '2020-11-28'),
('11', '1007', 'decision', 'French', '104', 'D', '2020-11-27'),
('23', '1004', 'skip', 'Persian', '56', 'A', '2020-11-26'),
('20', '1003', 'transfer', 'Italian', '45', 'C', '2020-11-25')
;
```

Table Name: job_data
This is the name of the table I have created to store user-related information.

- job_id int: This column stores the job's id in an integer (INT) data type.
- actor_id int: This column stores the actor's id in an integer (INT) data type.
- event varchar(50): This column stores the event type of the job data. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.
- language varchar(50): This column stores the language of the job data. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.
- time_spent time: This column stores time spent on reviewing a job in the time data type with the default format type of "hh:mm:ss"
- org varchar(100): This column stores the organization type of the job data. It's a variable-length character (VARCHAR) data type with a maximum length of 100 characters.
- ds date: This column stores the date of the job in the date data type with the default format type of "YYYY-MM-DD"

2) *Case Study 2:*

A. Create Database:

```
create database investigating_metrics;
```

B. Create Table:

1. This SQL code creates a table named **users** to store information about the user's account. Here's a breakdown of each part of the code:

```
create table users
(
user_id int,
created_at datetime,
company_id int,
language varchar(50),
activated_at datetime,
state varchar(50)
);
```

Table Name: users
This is the name of the table I have created to store user's account-related information.

- user_id int: This column stores the user's id of their accounts in an integer (INT) data type.

- created_at datetime: This column stores the date and time at which the user created his/her account in the datetime data type with the default format type of “YYYY-MM-DD HH-MI-SS”
- company_id int: This column stores the company’s id in an integer (INT) data type.
- language varchar(50): This column stores the language in which the user created their account. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.
- activated_at datetime: This column stores the date and time at which the user activated his/her account in the datetime data type with the default format type of “YYYY-MM-DD HH-MI-SS”
- state varchar(50): This column stores the state of the account i.e., whether it is active or inactive. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.

2. This SQL code creates a table named **events** that occurred in the user’s account. Here's a breakdown of each part of the code:

```
create table events
(
user_id int,
occurred_at datetime,
event_type varchar(50),
event_name varchar(50),
location varchar(50),
device varchar(50),
user_type int
);
```

Table Name: events
This is the name of the table I have created to store the events that occurred in a user’s account.

- user_id int: This column stores the user’s id of events of their accounts in an integer (INT) data type.
- occurred_at datetime: This column stores the date and time at which the event has occurred in the user’s account in the datetime data type with the default format type of “YYYY-MM-DD HH-MI-SS”
- event_type varchar(50): This column stores the type of event like engagement, or signup flow that occurred in the user’s account. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.

- event_name varchar(50): This column stores the name of the event that occurred in the user's account. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.
- location varchar(50): This column stores the location where the event has occurred in the user's account. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.
- device varchar(50): This column stores the name of the device with which the user is operating his/her account. It's a variable-length character (VARCHAR) data type with a maximum length of 50 characters.
- user_type int: This column stores the user's type of their accounts in an integer (INT) data type.

3. This SQL code creates a table named **events** that occurred in the user's account. Here's a breakdown of each part of the code:

```
create table email_events
(
user_id int,
occurred_at datetime,
action varchar(100),
user_type int
);
```

Table Name: email_events

This is the name of the table I have created to store the events specific to the sending of emails that occurred in a user's account.

- user_id int: This column stores the user's id of events specific to emails that were sent to the user's accounts in an integer (INT) data type.
- occurred_at datetime: This column stores the date and time at which the email has been sent to the user's account in the datetime data type with the default format type of "YYYY-MM-DD HH-MI-SS"
- action varchar(100): This column stores the type of action that was carried out while sending the mail. It's a variable-length character (VARCHAR) data type with a maximum length of 100 characters
- user_type int: This column stores the user's type of their accounts specific to sending emails in an integer (INT) data type.

Results:

Case Study 1: Job Data Analysis

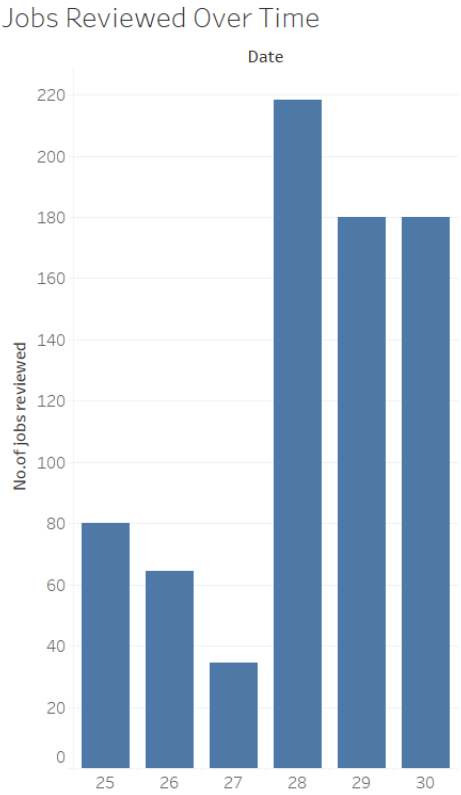
1) Jobs Reviewed Over Time:

Task: Calculate the number of jobs reviewed per hour for each day in November 2020.

```sql
select distinct ds as Days,
Count(job_id) / (sum(time_spent) / 3600) as no_of_jobs_reviewed
from job_data
group by Days;
```

Output:

	Days	no_of_jobs_reviewed
▶	2020-11-30	180.0000
	2020-11-29	180.0000
	2020-11-28	218.1818
	2020-11-27	34.6154
	2020-11-26	64.2857
	2020-11-25	80.0000



Insights:

- On average, 126 jobs were reviewed per hour in November.
- The highest number of jobs was reviewed on 28 November 2020 with 218 jobs per hour.
- The lowest number of jobs was reviewed on 27 November with 34 jobs per hour.

2) *Throughput Analysis:*

Task: Calculate the 7-day rolling average of throughput (number of events per second).

```sql
# Throughput Analysis:
# Weekly Througput
select round(count(event)/sum(time_spent),2) as "Weekly Throughput"
from job_data;
```

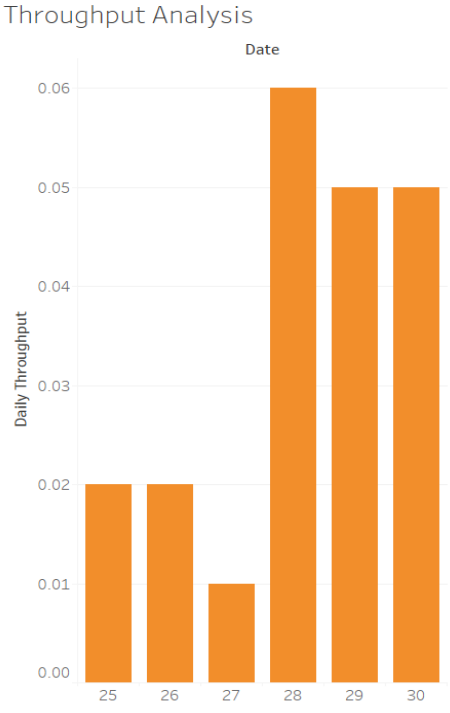
Output:

	Weekly Throughput
▶	0.03

```sql
# Daily Throughput
select ds as Dates, round(count(event)/sum(time_spent),2) as "Daily Throughput"
from job_data
group by ds
order by ds;
```

Output:

	Dates	Daily Throughput
▶	2020-11-25	0.02
	2020-11-26	0.02
	2020-11-27	0.01
	2020-11-28	0.06
	2020-11-29	0.05
	2020-11-30	0.05



Insights:

- The weekly throughput is 0.03.
- On the date 2020-11-28, the throughput is highest at 0.06
  On the date 2020-11-27, the throughput is lowest at 0.01
- Metrics will always go up and down on a weekly and daily basis. You will get numbers

faster every day or minute if you want. As a result, rolling metrics are superb at showing if your metrics are trending up or down on a daily level.

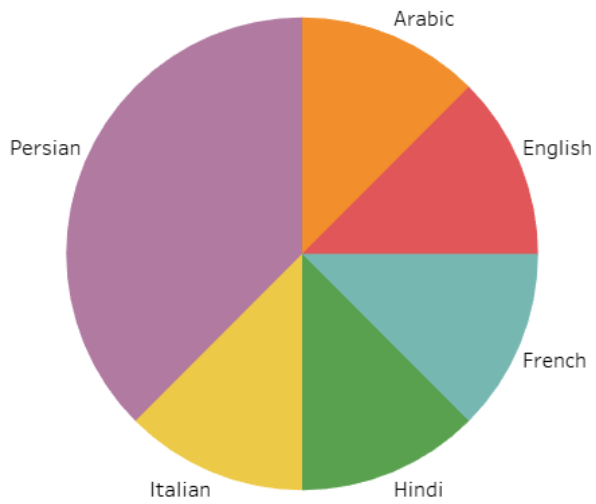
3) *Language Share Analysis:*

-*Task*: Calculate the percentage share of each language in the last 30 days.

```
# Language Share Analysis:
select language, count(language) as total_language,
count(language)*100/sum(count(language))
over() as percentage
from job_data
group by language
order by language;
```

Output:

language	total_language	percentage
Arabic	1	12.5000
English	1	12.5000
French	1	12.5000
Hindi	1	12.5000
Italian	1	12.5000
Persian	3	37.5000



Insights:

- Persian is the most used language with a percentage share of 37.5 %.
- The rest of the languages, Arabic, Hindi, Italian, French, and English have the same percentage share of 12.5%

4) ***Duplicate Rows Detection:***

-*Task*: Identify duplicate rows in the data.

```
# Duplicate rows detection:
select *, count(*) AS Duplicates
from job_data
group by ds, job_id, actor_id, event, language, time_spent, org
having count(*) > 1;
```

Output:

	job_id	actor_id	event	language	time_spent	org	ds	Duplicates

Insights:

- No duplicate rows were present in the data

Case Study 2: Investigating Metric Spike

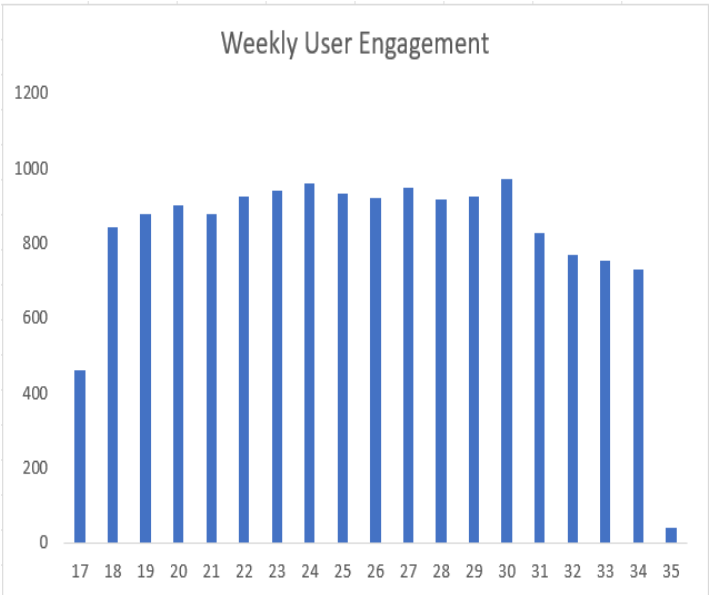
1) ***Weekly User Engagement:***

- Task: Measure the activeness of users on a weekly basis.

```
# Weekly User Engagement:
select extract(week from occurred_at) as week_num,
count(distinct user_id) as active_users
from events
where event_type = 'engagement'
group by week_num
order by week_num;
```

Output:

week_num	active_users
17	462
18	843
19	877
20	902
21	878
22	925
23	941
24	961
25	931
26	920
27	949
28	917
29	923
30	970
31	827
32	770
33	754
34	731
35	42



Insights:

- The highest engagement was seen on week 30, with 970 users.
- The lowest engagement was seen on week 17, with 462 users.

2) *User Growth Analysis:*

- *Task*: Analyse the growth of users over time for a product.

```
# User Growth Analysis:
select year, week_num, num_users, sum(num_users)
over(order by year, week_num) as cum_users
from (
select extract(year from created_at) as year, extract(week from created_at) as week_num, count(distinct user_id) as num_users
from users
where state = 'active'
group by year, week_num
order by year, week_num)sub;
```

Output:

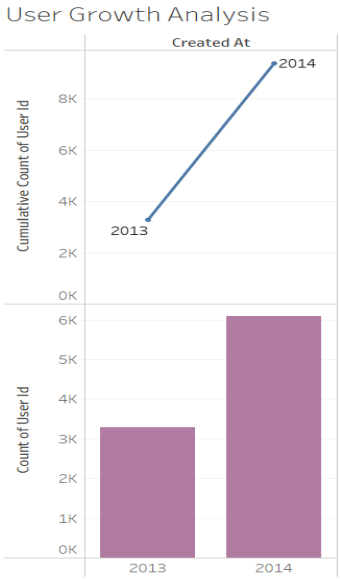
year	week_num	num_users	cum_users
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724

year	week_num	num_users	cum_users
2013	19	57	781
2013	20	39	820
2013	21	49	869
2013	22	54	923
2013	23	50	973
2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813
2013	37	85	1898

year	week_num	num_users	cum_users
2013	38	90	1988
2013	39	84	2072
2013	40	87	2159
2013	41	73	2232
2013	42	99	2331
2013	43	89	2420
2013	44	96	2516
2013	45	91	2607
2013	46	88	2695
2013	47	102	2797
2013	48	97	2894
2013	49	116	3010
2013	50	124	3134
2013	51	102	3236
2013	52	47	3283
2014	0	83	3366
2014	1	126	3492
2014	2	109	3601
2014	3	113	3714

year	week_num	num_users	cum_users
2014	4	130	3844
2014	5	133	3977
2014	6	135	4112
2014	7	125	4237
2014	8	129	4366
2014	9	133	4499
2014	10	154	4653
2014	11	130	4783
2014	12	148	4931
2014	13	167	5098
2014	14	162	5260
2014	15	164	5424
2014	16	179	5603
2014	17	170	5773
2014	18	163	5936
2014	19	185	6121
2014	20	176	6297
2014	21	183	6480
2014	22	196	6676

2014	23	196	6872
2014	24	229	7101
2014	25	207	7308
2014	26	201	7509
2014	27	222	7731
2014	28	215	7946
2014	29	221	8167
2014	30	238	8405
2014	31	193	8598
2014	32	245	8843
2014	33	261	9104
2014	34	259	9363
2014	35	18	9381



Insights:

- The 33rd week of 2014 saw the greatest number of users actively engaging with the product or service with 261 users.
- The 35th week of 2014 had the lowest number of active users i.e., 18.

3) *Weekly Retention Analysis:*

- *Task*: Analyse the retention of users on a weekly basis after signing up for a product.

```sql
SELECT first AS "Week Numbers",
SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS "Week 0",
SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS "Week 1",
SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS "Week 2",
SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS "Week 3",
SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS "Week 4",
SUM(CASE WHEN week_number = 5 THEN 1 ELSE 0 END) AS "Week 5",
SUM(CASE WHEN week_number = 6 THEN 1 ELSE 0 END) AS "Week 6",
SUM(CASE WHEN week_number = 7 THEN 1 ELSE 0 END) AS "Week 7",
SUM(CASE WHEN week_number = 8 THEN 1 ELSE 0 END) AS "Week 8",
SUM(CASE WHEN week_number = 9 THEN 1 ELSE 0 END) AS "Week 9",
SUM(CASE WHEN week_number = 10 THEN 1 ELSE 0 END) AS "Week 10",
SUM(CASE WHEN week_number = 11 THEN 1 ELSE 0 END) AS "Week 11",
SUM(CASE WHEN week_number = 12 THEN 1 ELSE 0 END) AS "Week 12",
SUM(CASE WHEN week_number = 13 THEN 1 ELSE 0 END) AS "Week 13",
SUM(CASE WHEN week_number = 14 THEN 1 ELSE 0 END) AS "Week 14",
SUM(CASE WHEN week_number = 15 THEN 1 ELSE 0 END) AS "Week 15",
SUM(CASE WHEN week_number = 16 THEN 1 ELSE 0 END) AS "Week 16",
SUM(CASE WHEN week_number = 17 THEN 1 ELSE 0 END) AS "Week 17",
SUM(CASE WHEN week_number = 18 THEN 1 ELSE 0 END) AS "Week 18"
From
(
SELECT m.user_id, m.login_week, n.first, m.login_week - first as week_number
FROM
(SELECT user_id, EXTRACT(WEEK FROM occurred_at) AS login_week from events
GROUP BY 1, 2) m,
(SELECT user_id, MIN(EXTRACT(WEEK FROM occurred_at)) AS first from events
GROUP BY 1) n
WHERE m.user_id = n.user_id
) sub
GROUP BY first
ORDER BY first;
```

Output:

Week Numbers	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
17	462	284	206	179	152	147	135	126	121	119	116	104	104	115	88
18	559	243	197	168	144	133	140	124	117	126	110	120	129	115	99
19	428	158	116	117	95	87	85	79	92	82	75	69	64	43	51
20	368	117	95	75	59	49	46	38	44	50	47	28	28	28	33
21	325	87	67	42	45	42	47	44	40	28	31	25	21	20	2
22	335	103	76	56	44	37	36	38	33	34	24	30	26	2	0
23	333	89	60	48	40	39	35	35	34	28	22	20	0	0	0
24	342	81	64	48	42	34	36	38	24	21	18	0	0	0	0
25	310	71	50	44	38	36	30	28	23	19	2	0	0	0	0
26	290	60	39	26	31	21	16	18	13	0	0	0	0	0	0
27	295	51	32	29	22	16	15	14	1	0	0	0	0	0	0
28	282	57	33	19	13	15	12	3	0	0	0	0	0	0	0
29	274	50	35	18	12	8	0	0	0	0	0	0	0	0	0
30	299	51	38	20	15	0	0	0	0	0	0	0	0	0	0
31	217	33	15	14	0	0	0	0	0	0	0	0	0	0	0
32	267	27	21	0	0	0	0	0	0	0	0	0	0	0	0
33	286	40	1	0	0	0	0	0	0	0	0	0	0	0	0
34	279	3	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Week 15	Week 16	Week 17	Week 18
70	64	63	4
86	65	4	0
51	2	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Week Numbers	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18
17	462	284	206	179	152	147	135	126	121	119	116	104	104	115	88	70	64	63	4
18	559	243	197	168	144	133	140	124	117	126	110	120	129	115	99	86	65	4	0
19	428	158	116	117	95	87	85	79	92	82	75	69	64	43	51	51	2	0	0
20	368	117	95	75	59	49	46	38	44	50	47	28	28	28	33	0	0	0	0
21	325	87	67	42	45	42	47	44	40	28	31	25	21	20	2	0	0	0	0
22	335	103	76	56	44	37	36	38	33	34	24	30	26	2	0	0	0	0	0
23	333	89	60	48	40	39	35	35	34	28	22	20	0	0	0	0	0	0	0
24	342	81	64	48	42	34	36	38	24	21	18	0	0	0	0	0	0	0	0
25	310	71	50	44	38	36	30	28	23	19	2	0	0	0	0	0	0	0	0
26	290	60	39	26	31	21	16	18	13	0	0	0	0	0	0	0	0	0	0
27	295	51	32	29	22	16	15	14	1	0	0	0	0	0	0	0	0	0	0
28	282	57	33	19	13	15	12	3	0	0	0	0	0	0	0	0	0	0	0
29	274	50	35	18	12	8	0	0	0	0	0	0	0	0	0	0	0	0	0
30	299	51	38	20	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	217	33	15	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	267	27	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	286	40	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	279	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Insights:

- In the initial weeks, the retention rate is quite high. As weeks go by, the rate deteriorates
- The highest weekly retention rate is in week 18, with 559 users.

4) ***Weekly Engagement Per Device***:

- Task: Measure the activeness of users on a weekly basis per device.

```sql
# Weekly Engagement Per Device:
SELECT EXTRACT(WEEK FROM occurred_at) AS "Week Numbers",
COUNT(DISTINCT CASE WHEN device IN('dell inspiron notebook') THEN user_id ELSE
NULL END) AS "Dell Inspiron Notebook",
COUNT(DISTINCT CASE WHEN device IN('iphone 5') THEN user_id ELSE
NULL END) AS "iphone 5",
COUNT(DISTINCT CASE WHEN device IN('iphone 4s') THEN user_id ELSE
NULL END) AS "iphone 4s",
COUNT(DISTINCT CASE WHEN device IN('windows surface') THEN user_id ELSE
NULL END) AS "windows surface",
COUNT(DISTINCT CASE WHEN device IN('macbook air') THEN user_id ELSE
NULL END) AS "macbook air",
COUNT(DISTINCT CASE WHEN device IN('iphone 5s') THEN user_id ELSE
NULL END) AS "iphone 5s",
COUNT(DISTINCT CASE WHEN device IN('macbook pro') THEN user_id ELSE
NULL END) AS "macbook pro",
COUNT(DISTINCT CASE WHEN device IN('kindle fire') THEN user_id ELSE
NULL END) AS "kindle fire",
COUNT(DISTINCT CASE WHEN device IN('ipad mini') THEN user_id ELSE
NULL END) AS "ipad mini",
COUNT(DISTINCT CASE WHEN device IN('nexus 7') THEN user_id ELSE
NULL END) AS "nexus 7",
COUNT(DISTINCT CASE WHEN device IN('nexus 5') THEN user_id ELSE
NULL END) AS "nexus 5",
COUNT(DISTINCT CASE WHEN device IN('samsung galaxy s4') THEN user_id ELSE
NULL END) AS "samsung galaxy s4",
COUNT(DISTINCT CASE WHEN device IN('lenovo thinkpad') THEN user_id ELSE
NULL END) AS "lenovo thinkpad",
COUNT(DISTINCT CASE WHEN device IN('samsumg galaxy tablet') THEN user_id ELSE
NULL END) AS "samsumg galaxy tablet",
COUNT(DISTINCT CASE WHEN device IN('acer aspire notebook') THEN user_id ELSE
NULL END) AS "acer aspire notebook",
COUNT(DISTINCT CASE WHEN device IN('asus chromebook') THEN user_id ELSE
NULL END) AS "asus chromebook",
COUNT(DISTINCT CASE WHEN device IN('htc one') THEN user_id ELSE
NULL END) AS "htc one",
COUNT(DISTINCT CASE WHEN device IN('nokia lumia 635') THEN user_id ELSE
NULL END) AS "nokia lumia 635",
COUNT(DISTINCT CASE WHEN device IN('samsung galaxy note') THEN user_id ELSE
NULL END) AS "samsung galaxy note",
COUNT(DISTINCT CASE WHEN device IN('acer aspire desktop') THEN user_id ELSE
NULL END) AS "acer aspire desktop",
COUNT(DISTINCT CASE WHEN device IN('mac mini') THEN user_id ELSE
NULL END) AS "mac mini",
COUNT(DISTINCT CASE WHEN device IN('hp pavilion desktop') THEN user_id ELSE
NULL END) AS "hp pavilion desktop",
COUNT(DISTINCT CASE WHEN device IN('dell inspiron desktop') THEN user_id ELSE
NULL END) AS "dell inspiron desktop",
COUNT(DISTINCT CASE WHEN device IN('ipad air') THEN user_id ELSE
NULL END) AS "ipad air",
COUNT(DISTINCT CASE WHEN device IN('amazon fire phone') THEN user_id ELSE
NULL END) AS "amazon fire phone",
COUNT(DISTINCT CASE WHEN device IN('nexus 10') THEN user_id ELSE
NULL END) AS "nexus 10"
FROM events
WHERE event_type = 'engagement'
GROUP BY 1
ORDER BY 1;
```

Output:

	Week Numbers	Dell Inspiron Notebook	iphone 5	iphone 4s	windows surface	macbook air	iphone 5s	macbook pro	kindle fire	ipad mini	nexus 7	nexus 5
▶	17	29	49	14	3	35	31	104	4	11	12	28
	18	54	78	40	5	93	54	201	23	23	21	63
	19	61	98	33	11	88	61	208	14	26	32	66
	20	68	89	42	14	82	64	207	17	27	23	75
	21	64	102	36	16	79	52	184	21	21	24	70
	22	77	91	33	12	117	57	187	14	23	35	73
	23	80	114	42	8	88	59	192	19	26	25	65
	24	72	109	43	15	114	61	192	19	26	36	61
	25	83	92	29	15	81	48	195	16	21	33	71
	26	61	105	37	12	94	64	192	15	28	31	62
	27	57	105	49	22	103	51	212	21	23	26	61
	28	76	90	41	24	101	55	187	22	25	22	57
	29	76	86	42	15	99	58	192	28	18	24	56
	30	84	95	37	9	88	65	219	15	22	35	61
	31	67	81	30	11	89	43	216	7	15	26	46
	32	57	75	18	6	71	42	189	8	12	16	39
	33	60	61	15	8	82	38	193	7	16	18	45
	34	57	55	26	13	79	37	169	6	10	17	44
	35	4	0	3	1	6	0	5	1	1	0	3

samsung galaxy s4	lenovo thinkpad	samsumg galaxy tablet	acer aspire notebook	asus chromebook	htc one	nokia lumia 635	samsung galaxy note	acer aspire desktop
40	59	6	15	16	9	10	6	7
57	128	8	27	28	15	26	14	17
62	137	6	34	21	26	18	9	14
67	127	8	33	33	24	15	15	16
68	131	5	35	31	17	14	16	25
80	136	7	32	37	21	18	10	20
75	132	8	36	32	15	20	8	18
69	114	6	33	32	15	23	13	20
72	147	8	36	29	13	25	9	19
77	135	10	28	32	16	29	7	24
79	139	7	37	36	17	21	10	15
85	140	2	32	29	15	29	8	16
82	139	7	31	32	22	35	10	16
64	143	6	39	37	15	24	10	23
66	125	5	35	29	7	19	9	18
47	106	3	32	41	14	16	10	21
42	115	8	25	36	12	17	9	24
48	123	8	41	35	19	10	3	15
2	6	0	2	1	2	2	0	1

mac mini	hp pavilion desktop	dell inspiron desktop	ipad air	amazon fire phone	nexus 10
4	12	14	18	4	11
7	28	43	39	6	21
14	32	30	45	8	20
22	23	47	48	6	18
15	32	31	38	2	17
18	25	40	41	5	24
16	46	35	32	15	34
23	43	43	39	9	26
17	31	36	45	11	22
9	32	38	39	11	21
11	37	27	34	7	28
20	39	33	34	5	20
22	32	33	33	6	18
16	26	32	45	6	26
14	31	31	33	6	12
12	32	34	29	8	19
20	21	23	18	6	13
21	22	28	23	6	13
1	0	1	0	0	0

Week Numbers	dell inspiron noteboo	iphone 5	iphone 4s	windows surface	macbook air	iphone 5s	macbook pro	kindle fire	ipad mini	nexus 7	nexus 5	samsung galaxy s4	lenovo thinkpad	samsumg galaxy tablet	acer aspire noteboo	asus chromeb ook	samsung galaxy note	mac mini	hp pavilion desktop	ipad air	htc one	dell inspiron desktop	amazon fire phone	acer aspire desktop	nokia lumia 635	nexus 10
17	29	49	14	3	35	31	104	4	11	12	28	40	59	6	15	16	9	10	6	7	4	12	14	18	4	11
18	54	78	40	5	93	54	201	23	23	21	63	57	128	8	27	28	15	26	14	17	7	28	43	39	6	21
19	61	98	33	11	88	61	208	14	26	32	66	62	137	6	34	21	26	18	9	14	14	32	30	45	8	20
20	68	89	42	14	82	64	207	17	27	23	75	67	127	8	33	33	24	15	15	16	22	23	47	48	6	13
21	64	102	36	16	79	52	184	21	21	24	70	68	131	5	35	31	17	14	16	25	15	32	31	38	2	17
22	77	91	33	12	117	57	187	14	23	35	73	80	136	7	32	37	21	18	10	20	18	25	40	41	5	24
23	80	114	42	8	88	59	192	19	26	25	65	75	132	8	36	32	15	20	8	18	16	46	35	32	15	34
24	72	109	43	15	114	61	192	19	26	36	61	69	114	6	33	32	15	23	13	20	23	43	43	39	9	25
25	83	92	29	15	81	48	195	16	21	33	71	72	147	8	36	29	13	25	9	19	17	31	36	45	11	22
26	61	105	37	12	94	64	192	15	28	31	62	77	135	10	28	32	16	29	7	24	9	32	38	39	11	21
27	57	105	49	22	103	51	212	21	23	26	61	79	139	7	37	36	17	21	10	15	11	37	27	34	7	23
28	76	90	41	24	101	55	187	22	25	22	57	85	140	2	32	29	15	29	8	16	20	39	33	34	5	20
29	76	86	42	15	99	58	192	28	18	24	56	82	139	7	31	32	22	35	10	16	22	32	33	33	6	13
30	84	95	37	9	88	65	219	15	22	35	61	64	143	6	39	37	15	24	10	23	16	26	32	45	6	25
31	67	81	30	11	89	43	216	7	15	26	46	66	125	5	35	29	7	19	9	18	14	31	31	33	6	12
32	57	75	18	6	71	42	189	8	12	16	39	47	106	3	32	41	14	16	10	21	12	32	34	29	8	19
33	60	61	15	8	82	38	193	7	16	18	45	42	115	8	25	36	12	17	9	24	20	21	23	18	6	13
34	57	55	26	13	79	37	169	6	10	17	44	48	123	8	41	35	19	10	3	15	21	22	28	23	6	13
35	4	0	3	1	6	0	5	1	1	0	3	2	6	0	2	1	2	2	0	1	1	0	1	0	0	0

Insights:

- The most used device is the MacBook Pro. Its highest usage has been recorded in the 30th week with 219 users.
- The least used devices are the iPhone 5, iPhone 5s, nexus 7, Samsung Galaxy tablet, Samsung Galaxy Note, HP Pavilion desktop, iPad Air, Amazon Fire phone, and Nexus 10 with 0 users in the 35th week.

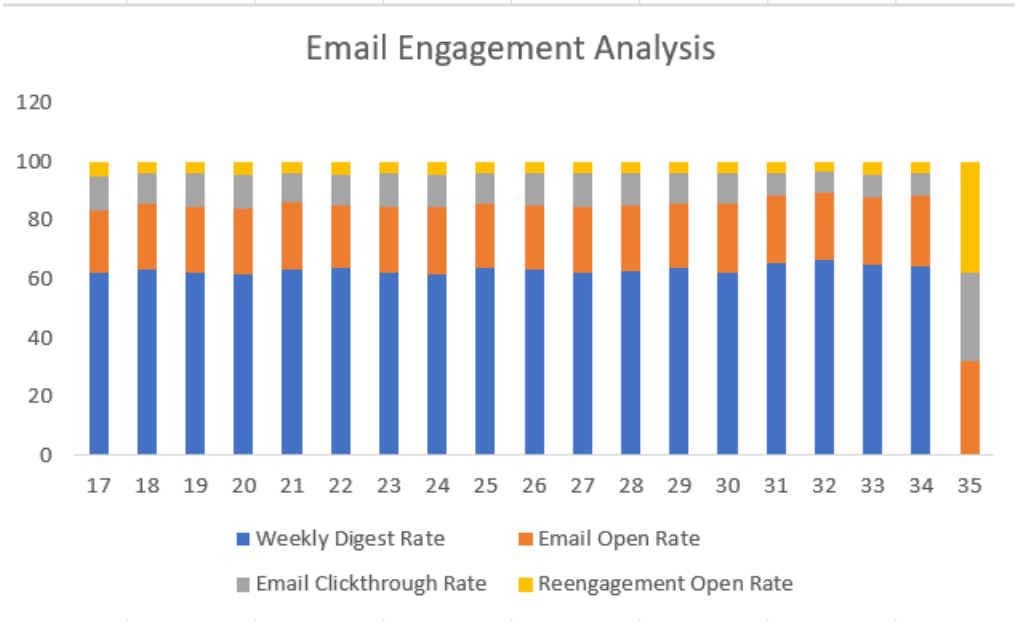
5) *Email Engagement Analysis:*

- *Task*: Analyse how users are engaging with the email service.

```
# Email Engagement Analysis:
SELECT Week,
ROUND((weekly_digest/total*100),2) as "Weekly Digest Rate",
ROUND((email_opens/total*100),2) as "Email Open Rate",
ROUND((email_clickthroughs/total*100),2) as "Email Clickthrough Rate",
ROUND((reengagement_emails/total*100),2) as "Reengagement Email Rate"
FROM
(
SELECT EXTRACT(WEEK FROM occurred_at) as Week,
COUNT(CASE WHEN action = 'sent_weekly_digest' THEN user_id ELSE NULL END) as
weekly_digest,
COUNT(CASE WHEN action = 'email_open' THEN user_id ELSE NULL END) as
email_opens,
COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id ELSE NULL END) as
email_clickthroughs,
COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id ELSE NULL END) as
reengagement_emails,
COUNT(user_id) AS total
FROM email_events
GROUP BY 1
) sub
GROUP BY 1
ORDER BY 1;
```

Output:

Week	Weekly Digest Rate	Email Open Rate	Email Clickthrough Rate	Reengagement Email Rate
17	62.32	21.28	11.39	5.01
18	63.45	22.24	10.49	3.83
19	62.16	22.67	11.13	4.04
20	61.62	22.64	11.43	4.31
21	63.52	22.82	9.97	3.69
22	63.59	21.56	10.66	4.19
23	62.39	22.34	11.18	4.09
24	61.61	22.92	10.99	4.48
25	63.77	21.79	10.54	3.90
26	62.99	22.22	10.61	4.18
27	62.24	22.49	11.37	3.90
28	62.92	22.48	10.77	3.83
29	63.98	21.71	10.51	3.79
30	62.29	23.24	10.59	3.88
31	65.27	23.25	7.66	3.82
32	66.59	22.85	7.14	3.42
33	64.73	23.10	7.91	4.26
34	64.33	23.91	7.67	4.08
35	0.00	32.28	29.92	37.80



Insights:

- The highest weekly digest rate is 66.59 in week 32
- The highest email open rate is 32.28 in week 35
- The highest email clickthrough rate is 29.92 in week 35
- The highest re-engagement email rate is 37.80 in week 35

Summary:

- In Case Study 1, my analysis focused on job data. By calculating the number of jobs reviewed per hour per day for November 2020, I was able to understand the reviewing patterns and workload distribution during that period.
- Additionally, computing the 7-day rolling average of throughput provided a smoothed metric to monitor event rates, allowing for better trend analysis.
- Determining the percentage share of each language in the last 30 days helped identify language preferences in content, aiding in decision-making related to localization efforts.
- Lastly, addressing duplicate rows in the data ensured data accuracy and eliminated redundancy, contributing to cleaner and more reliable insights.
- Tracking weekly user engagement to provide visibility into user activity and their level of satisfaction with the product or service.
- Measuring user growth over time helped assess the product's adoption and evaluate the effectiveness of marketing strategies.
- Calculating weekly retention of sign-up cohorts highlighted the rate at which users continued to use the product after signing up.
- Assessing weekly engagement per device revealed device-specific user behaviour, aiding in improving the user experience.

Conclusion:

Through this project, I was able to understand how important Operational Analytics is for organizations as it helps in identifying and understanding areas where improvement is required.
In this project, I was able to get insights about various questions like the rate of job reviews, sharing of languages, patterns of user engagement weekly, growth of users, etc. which can be communicated to the management team as per the requirements using which they can make proper data-driven decisions.

This project has been highly beneficial as it allowed me to apply my SQL skills and gain hands-on experience in data analysis. By deriving these insights, I have contributed to optimizing operations and decision-making within the organization. The project has provided me with a deeper understanding of data-driven analysis and its potential to drive business growth and improve operational efficiency.