OPERATION ANALYTICS AND INVESTIGATION METRIC SPIKE

3-07-2024

SAKSHI RAHIWAL

PROJECT DESCRIPTION

The project revolves around Operational Analytics, where the primary goal is to analysis a company's end-to-end operations, identify areas for improvement, and understand sudden changes in key metrics. As a Lead Data Analyst, I will work with various datasets and tables to derive valuable insights to aid different departments within the company. The project consists of two case studies, each with specific tasks and objectives.

APPROACH

To successfully complete this project in Operational Analytics, the following approach will be followed:

1. Database Setup:
    - Create a dedicated database for the project to store and manage the datasets.
    - Import the provided datasets into the database, ensuring data integrity and proper table structures.
    - Establish relationships between tables where necessary.
2. SQL Analysis:
    - Utilize SQL queries to perform the requested analyses for both case studies.
    - Ensure that SQL queries are well-optimized for efficient data retrieval.
    - For each task, carefully examine the data, apply relevant SQL operations, and generate meaningful insights.
3. Report Preparation:
    - Create a comprehensive report in either PDF or PPT format to present the findings and insights to the leadership team.
    - The report should be structured with clear sections, including an executive summary, methodology, results, and recommendations.

Database Setup:

Create a Database:

- Open MySQL Workbench and connect to your MySQL server.
- In the SQL editor, execute the following SQL command to create a new database for your project. Replace your_database_name with your desired database name:

```
CREATE DATABASE project3;
```

Use the Created Database:

- After creating the database, make sure to select and use it for all subsequent operations:

```
USE project3;
```

Import Datasets:

- Assuming you have CSV files containing your datasets, you can import them into MySQL Workbench as follows:
  - In MySQL Workbench, right-click on your database name in the Navigator panel.
  - Select "Table Data Import Wizard."
  - Follow the wizard's instructions to import each CSV file into a new table in your database.

Table Structures and Relationships:

- Once your datasets are imported, you can explore the table structures and relationships to understand how the data is organized.
- Use the following SQL commands to describe the table structures and view their relationships:
    - To see the structure of a specific table: DESC job_data;

      DESC users;

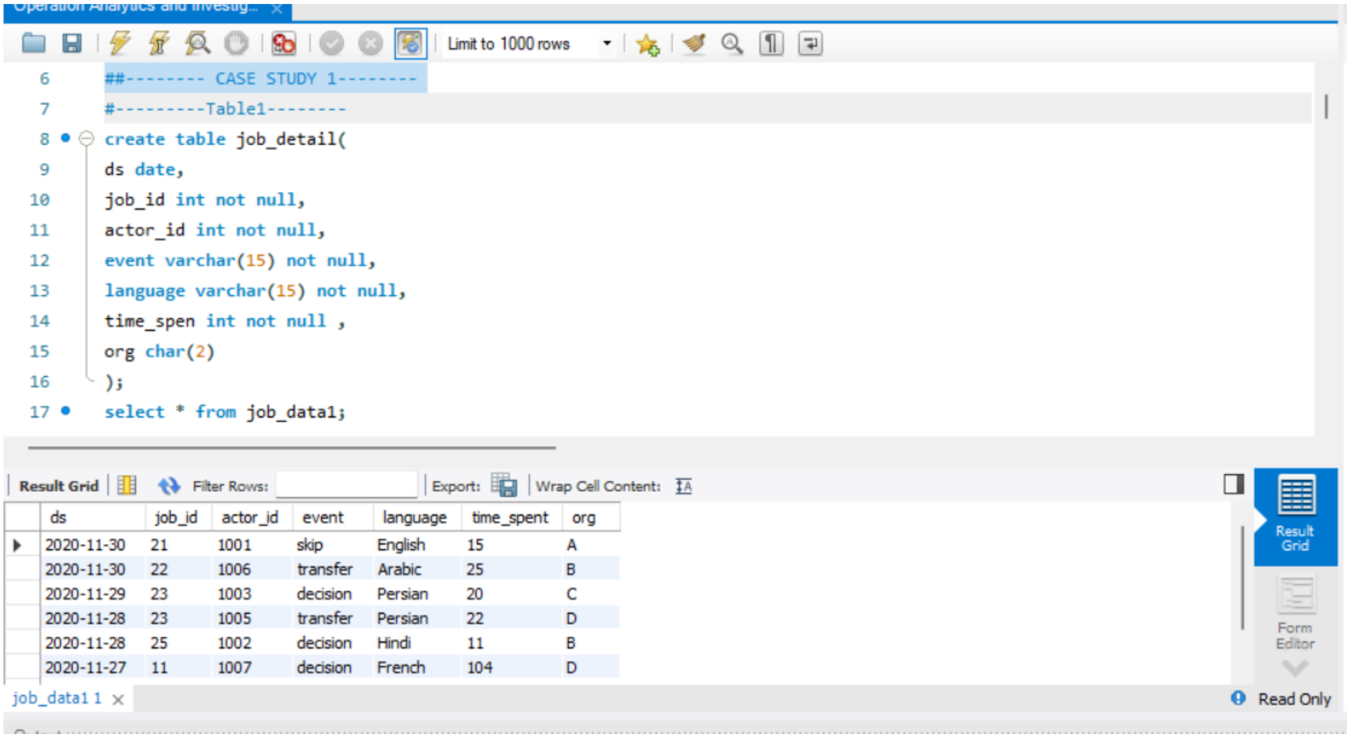
      DESC events;

      DESC email_events;

    - To list all tables in the database:

      SHOW TABLES;

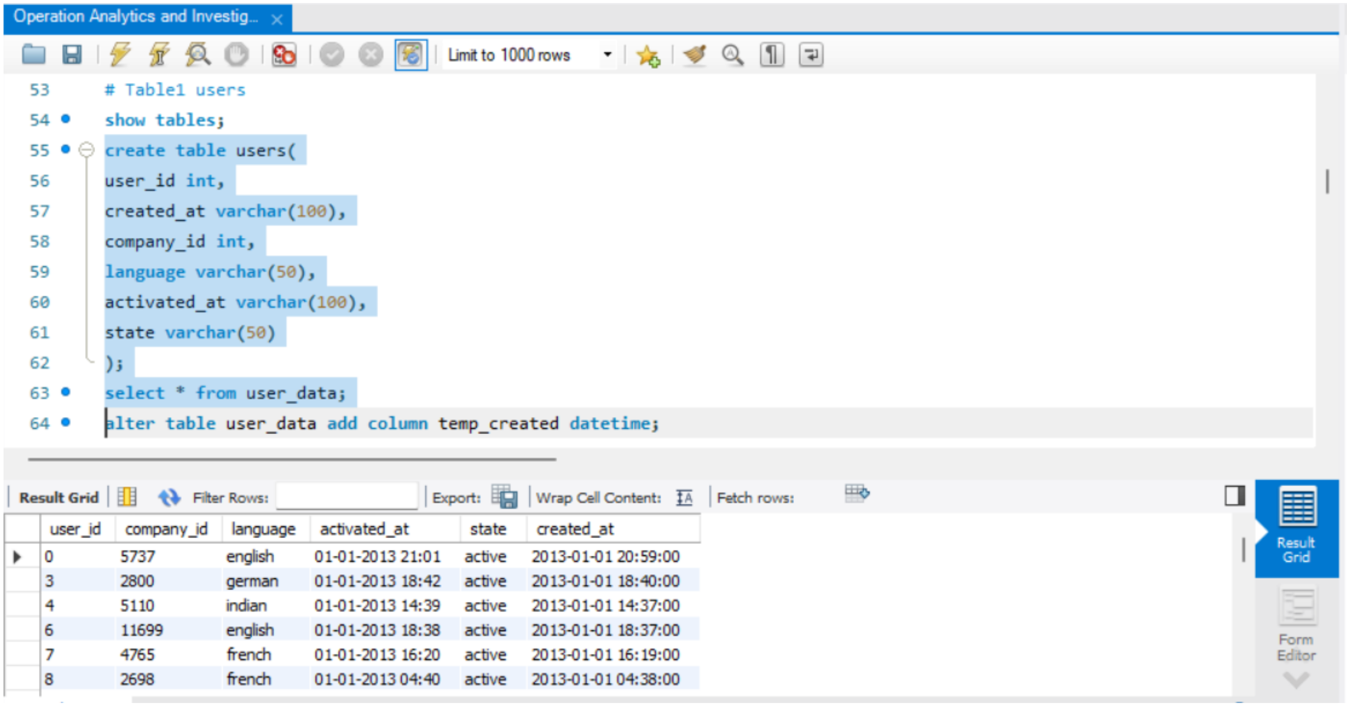
Job_Data Table:



```
##-------- CASE STUDY 1--------
#---------Table1--------
create table job_detail(
ds date,
job_id int not null,
actor_id int not null,
event varchar(15) not null,
language varchar(15) not null,
time_spen int not null ,
org char(2)
);
select * from job_data1;
```

ds	job_id	actor_id	event	language	time_spent	org
2020-11-30	21	1001	skip	English	15	A
2020-11-30	22	1006	transfer	Arabic	25	B
2020-11-29	23	1003	decision	Persian	20	C
2020-11-28	23	1005	transfer	Persian	22	D
2020-11-28	25	1002	decision	Hindi	11	B
2020-11-27	11	1007	decision	French	104	D

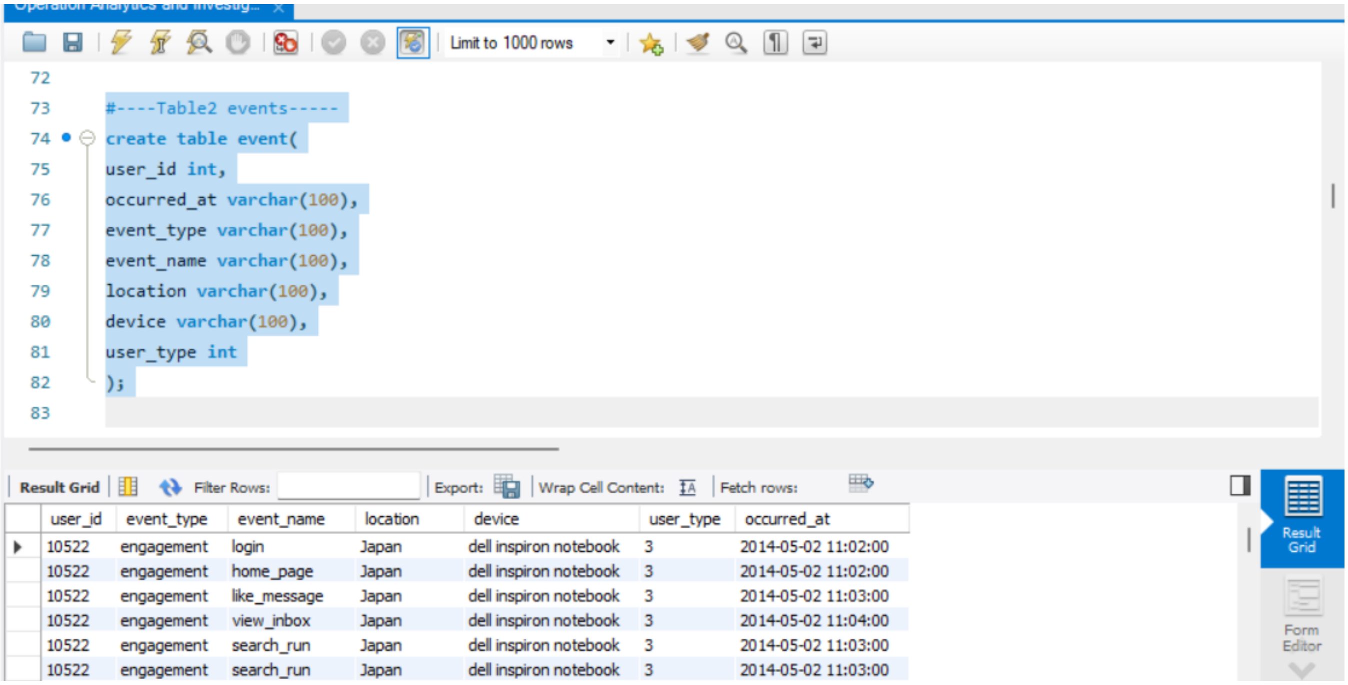
Users Table:



```
# Table1 users
show tables;
create table users(
user_id int,
created_at varchar(100),
company_id int,
language varchar(50),
activated_at varchar(100),
state varchar(50)
);
select * from user_data;
alter table user_data add column temp_created datetime;
```

user_id	company_id	language	activated_at	state	created_at
0	5737	english	01-01-2013 21:01	active	2013-01-01 20:59:00
3	2800	german	01-01-2013 18:42	active	2013-01-01 18:40:00
4	5110	indian	01-01-2013 14:39	active	2013-01-01 14:37:00
6	11699	english	01-01-2013 18:38	active	2013-01-01 18:37:00
7	4765	french	01-01-2013 16:20	active	2013-01-01 16:19:00
8	2698	french	01-01-2013 04:40	active	2013-01-01 04:38:00

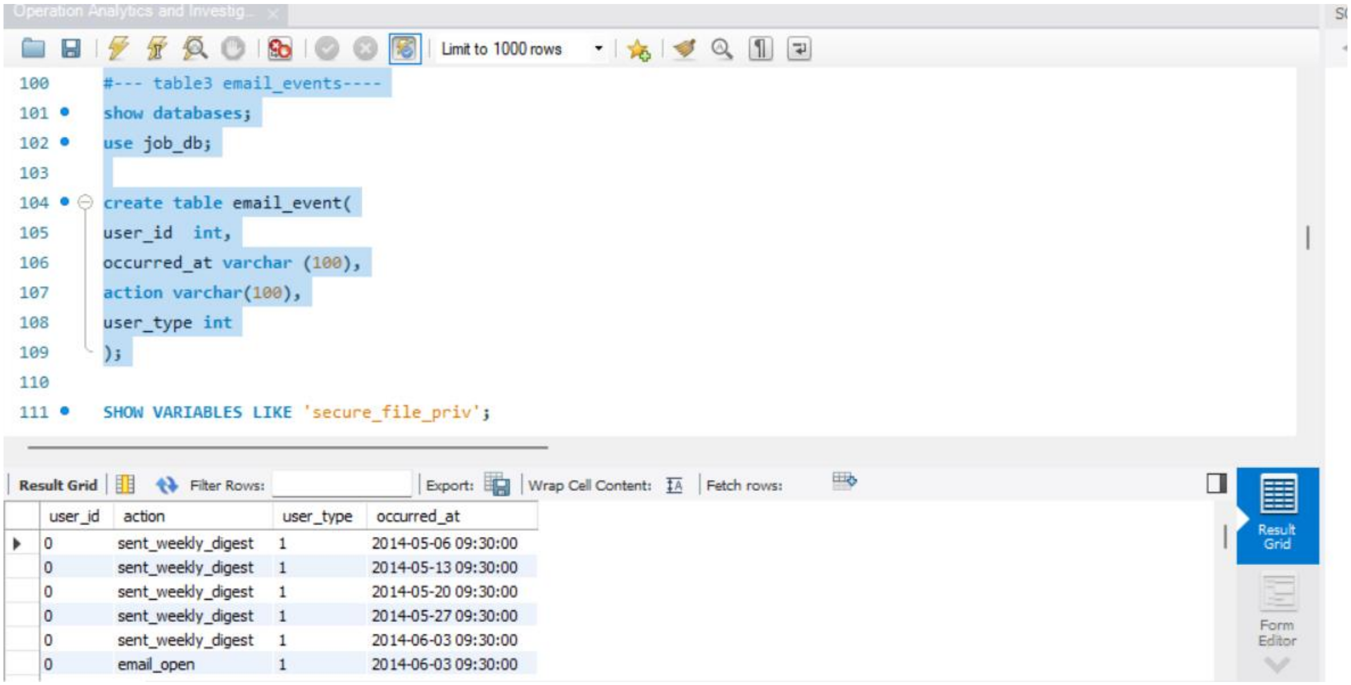
Event Table:



```
#----Table2 events-----
create table event(
user_id int,
occurred_at varchar(100),
event_type varchar(100),
event_name varchar(100),
location varchar(100),
device varchar(100),
user_type int
);
```

user_id	event_type	event_name	location	device	user_type	occurred_at
10522	engagement	login	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	home_page	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	like_message	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10522	engagement	view_inbox	Japan	dell inspiron notebook	3	2014-05-02 11:04:00
10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00

Event_Email Table:



```
#--- table3 email_events----
show databases;
use job_db;

create table email_event(
user_id  int,
occurred_at varchar (100),
action varchar(100),
user_type int
);

SHOW VARIABLES LIKE 'secure_file_priv';
```

user_id	action	user_type	occurred_at
0	sent_weekly_digest	1	2014-05-06 09:30:00
0	sent_weekly_digest	1	2014-05-13 09:30:00
0	sent_weekly_digest	1	2014-05-20 09:30:00
0	sent_weekly_digest	1	2014-05-27 09:30:00
0	sent_weekly_digest	1	2014-06-03 09:30:00
0	email_open	1	2014-06-03 09:30:00

ANALYSIS

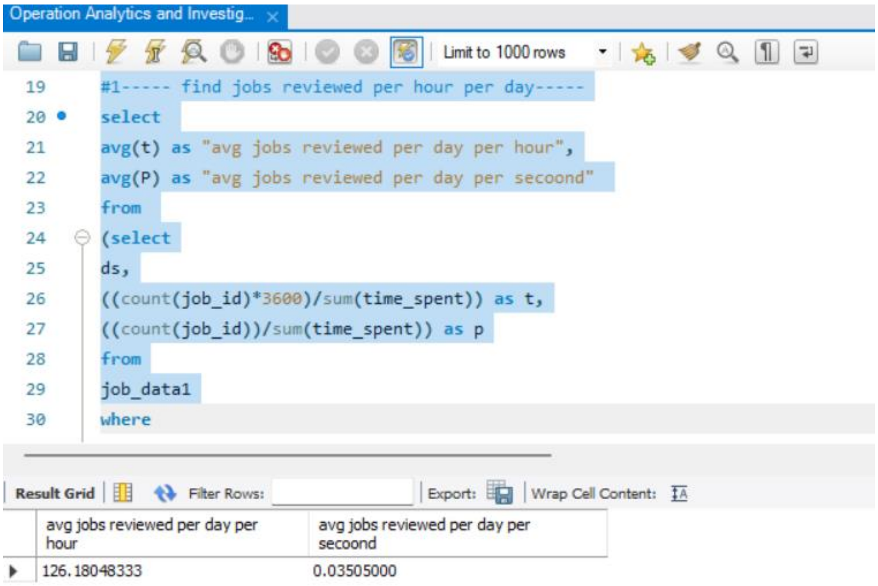
column, filters data fo

CASE STUDY 1: Job Data Analysis

TASK 1 - Jobs Reviewed Over Time:

- Objective: Calculate the number of jobs reviewed per hour foreach day in November 2020.

Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020



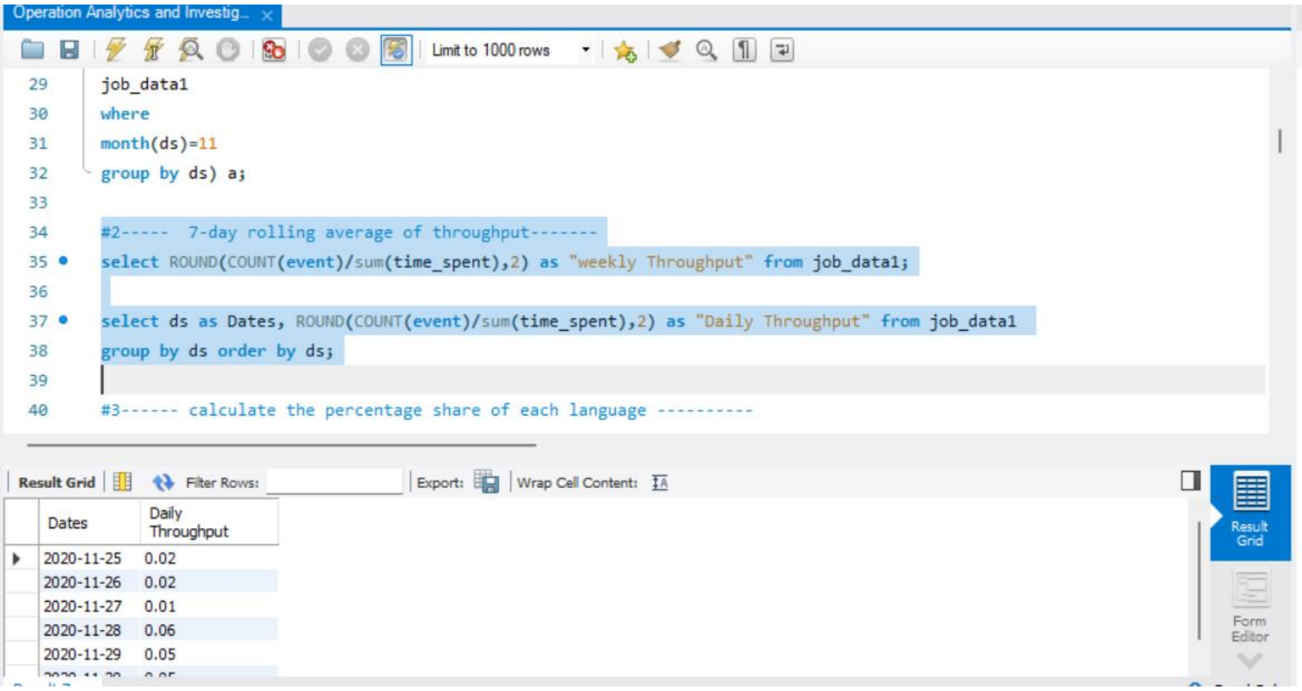
```
#1----- find jobs reviewed per hour per day-----
select
avg(t) as "avg jobs reviewed per day per hour",
avg(P) as "avg jobs reviewed per day per secoond"
from
(select
ds,
((count(job_id)*3600)/sum(time_spent)) as t,
((count(job_id))/sum(time_spent)) as p
from
job_data1
where
```

avg jobs reviewed per day per hour	avg jobs reviewed per day per secoond
126.18048333	0.03505000

Explanation: This query extract the date and hour from ds column filter data from November 2022, and 126.180 job reviewed pr hour

TASK 2- Throughput Analysis:

- **Objective: Calculate the 7-day rolling average of throughput (number of events per second**
- **Your Task: Write an SQL query to calculate the 7-day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.**

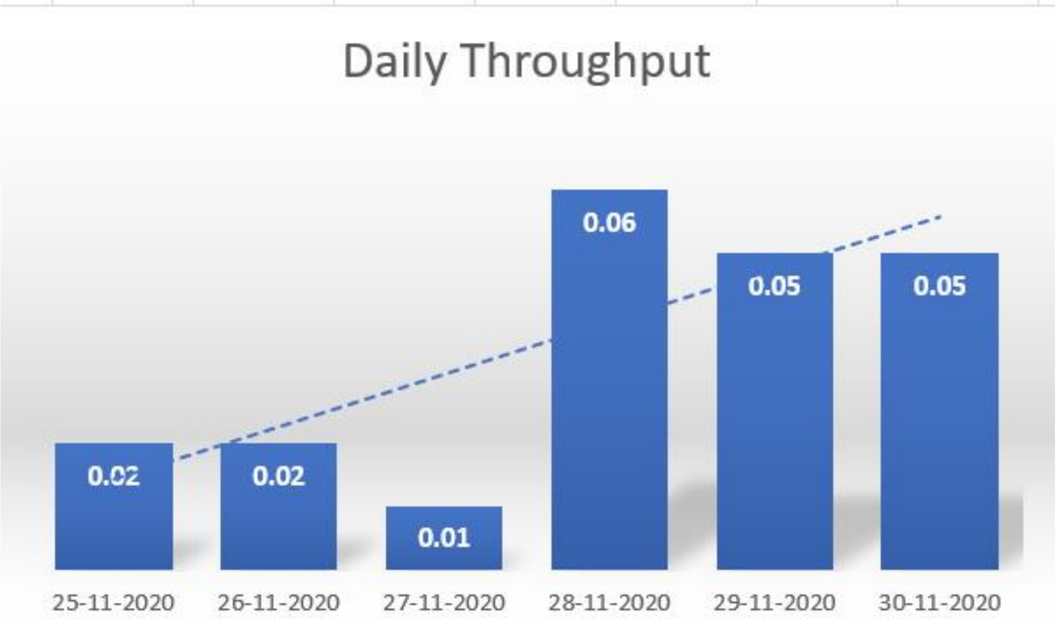


```
job_data1
where
month(ds)=11
group by ds) a;

#2-----  7-day rolling average of throughput-------
select ROUND(COUNT(event)/sum(time_spent),2) as "weekly Throughput" from job_data1;

select ds as Dates, ROUND(COUNT(event)/sum(time_spent),2) as "Daily Throughput" from job_data1
group by ds order by ds;

#3------ calculate the percentage share of each language ----------
```

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

Explanation:

This query will provide a result set with the date(ds) and the corresponding 7-day rolling average of throughput (rolling_ avg_throughput)



Daily Metric vs. 7-Day Rolling Average for Throughput:

Daily Metric:

- The daily metric provides insights into the throughput on a specific day.
- It can capture daily variations and spikes in throughput, which may be useful for short-term analysis.
- However, daily metrics can be highly sensitive to outliers and fluctuations, making it challenging to identify underlying trends.

7-Day Rolling Average:

- The 7-day rolling average smoothest out daily fluctuations and provides a more stable and trend-focused view of throughput.
- It is particularly useful for identifying longer-term trends and patterns in throughput.
- By averaging data over a 7-day period, it can help filter out noise and make it easier to identify significant changes or shifts in throughput trends.

Preference and Use Case:

The choice between the daily metric and the 7-day rolling average depends on the specific use case and the insights you want to gain:

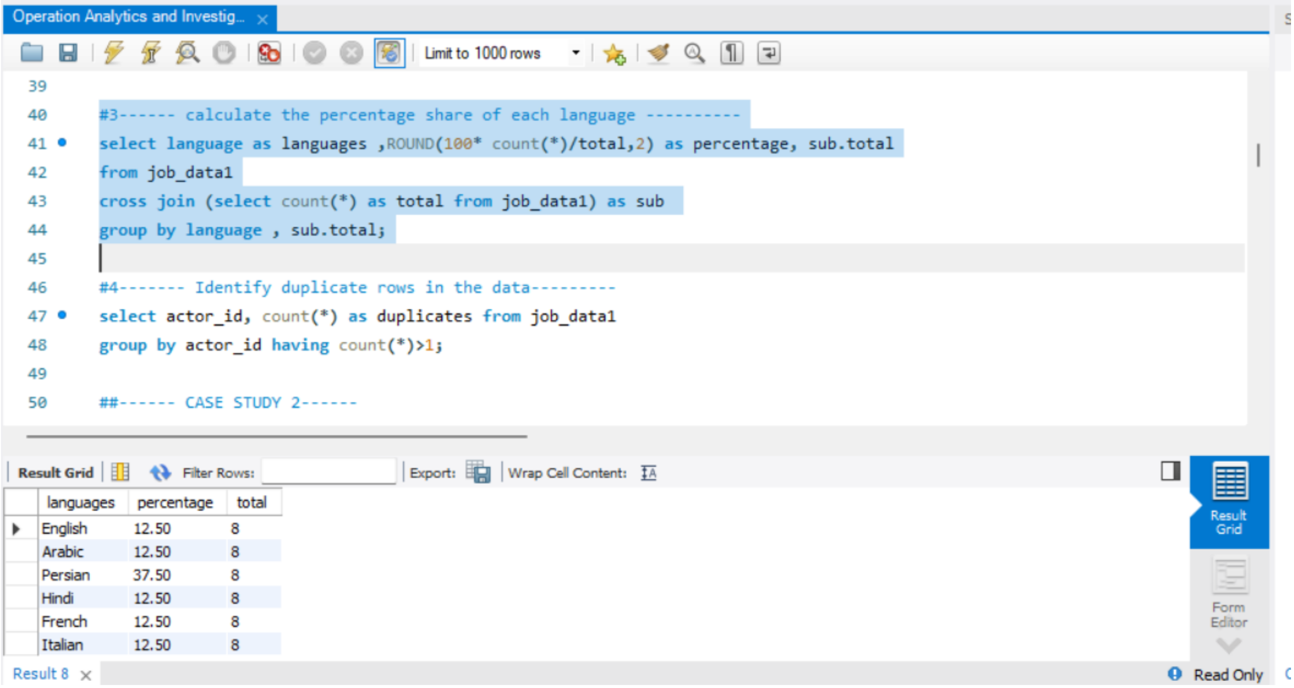
- If you need to understand short-term fluctuations or are interested in daily variations in throughput, the daily metric is more appropriate.

- If your goal is to identify broader trends, patterns, or anomalies in throughput, the 7-day rolling average is preferred. It provides a more stable and smoothed representation of throughput data, making it easier to spot trends over time.

TASK 3- Language Share Analysis:

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

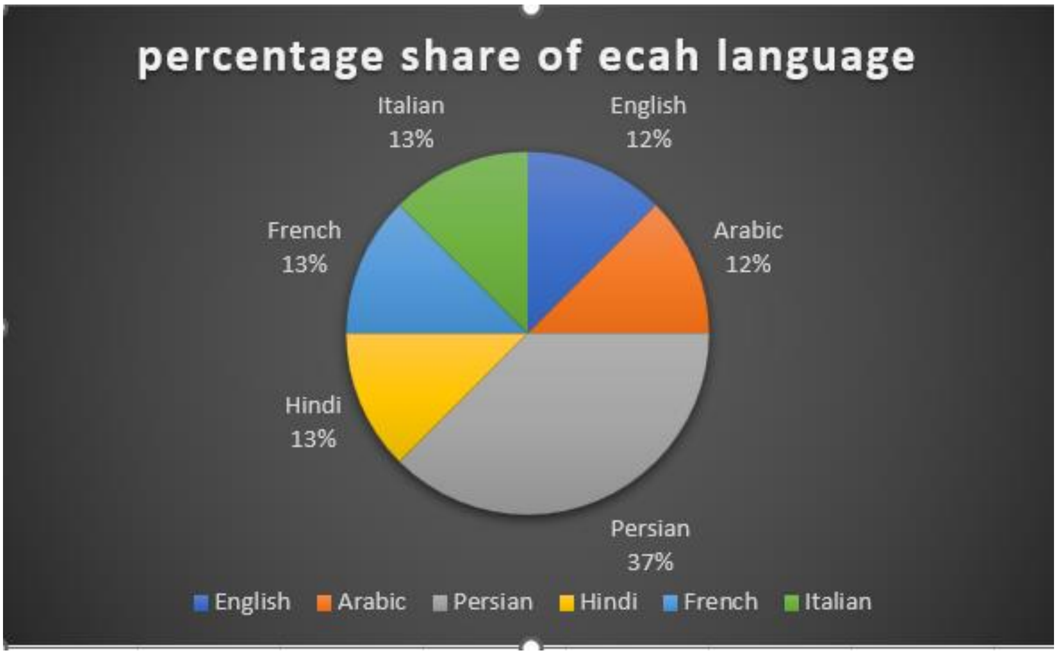
- **Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.**



```
#3------ calculate the percentage share of each language ----------
select language as languages ,ROUND(100* count(*)/total,2) as percentage, sub.total
from job_data1
cross join (select count(*) as total from job_data1) as sub
group by language , sub.total;

#4------- Identify duplicate rows in the data---------
select actor_id, count(*) as duplicates from job_data1
group by actor_id having count(*)>1;

##------ CASE STUDY 2------
```

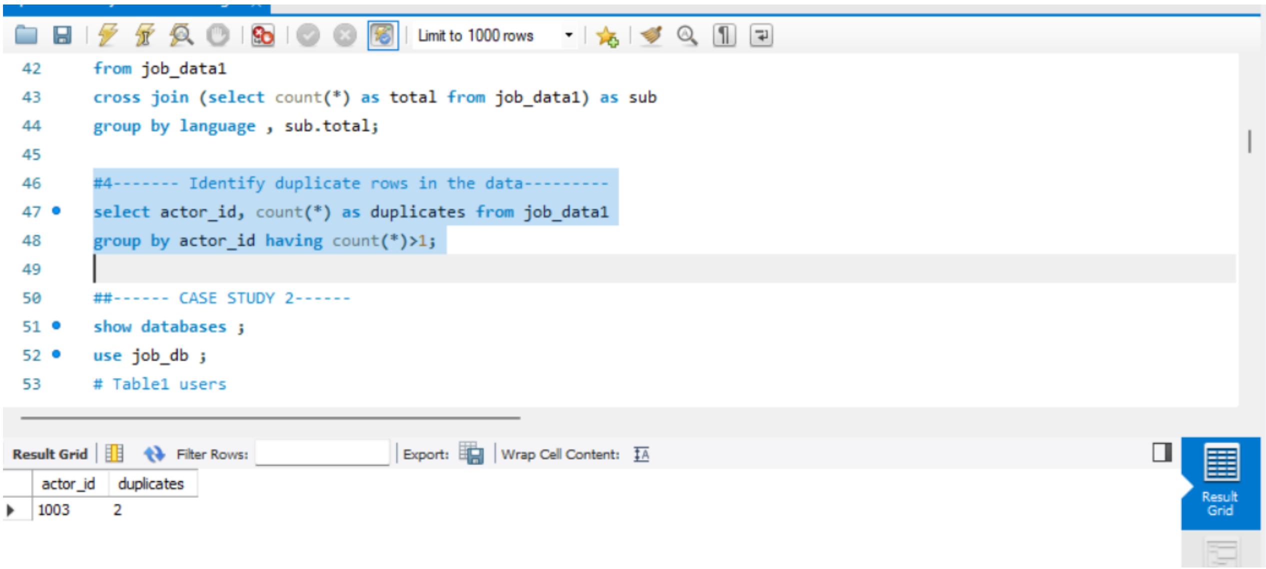
languages	percentage	total
English	12.50	8
Arabic	12.50	8
Persian	37.50	8
Hindi	12.50	8
French	12.50	8
Italian	12.50	8

Explanation: This query calculates the percentage share of each language in the last 30 days by first counting the total jobs for each language.



TASK 4- Duplicate Rows Detection:

- **Objective: Identify duplicate rows in the data.**
- **Your Task: Write an SQL query to display duplicate rows from the job_data table**.



```
from job_data1
cross join (select count(*) as total from job_data1) as sub
group by language , sub.total;

#4------- Identify duplicate rows in the data---------
select actor_id, count(*) as duplicates from job_data1
group by actor_id having count(*)>1;

##------ CASE STUDY 2------
show databases ;
use job_db ;
# Table1 users
```

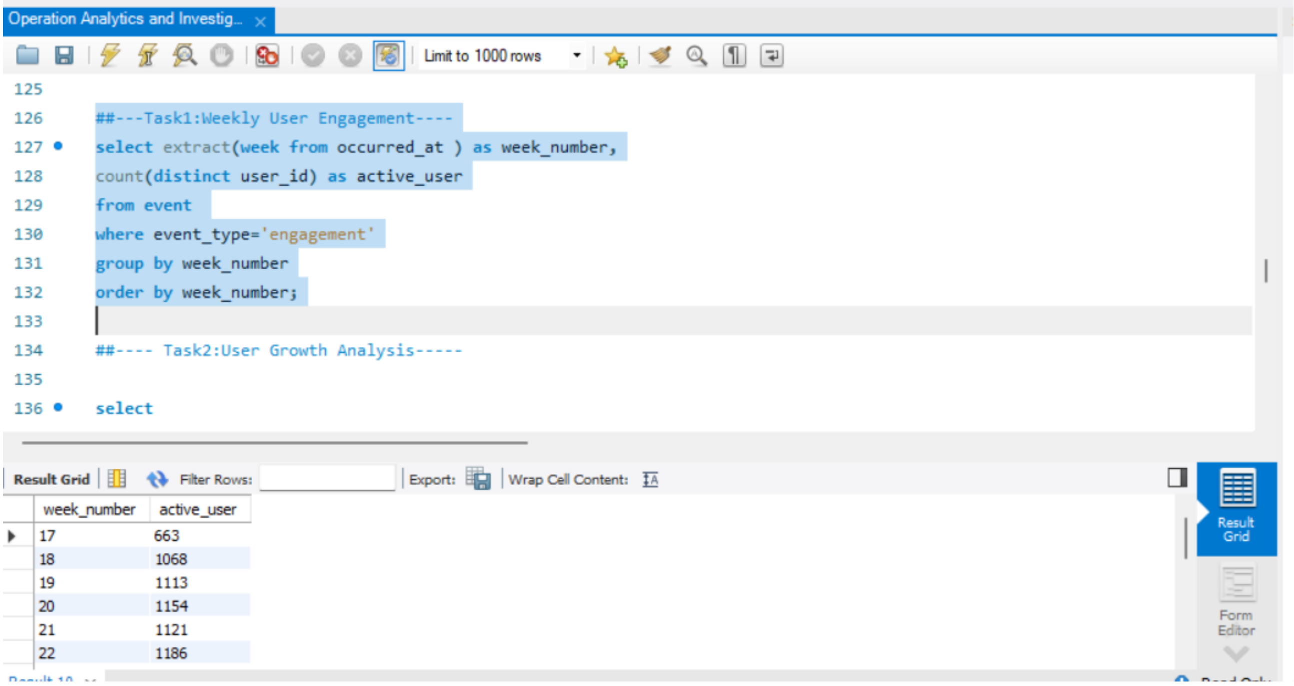
```
actor_id   duplicates
1003       2
```

Explanation: This query identifies 2 duplicate rows in the job_data table by grouping rows based on all columns and selecting those groups with a count greater than 1.

CASE STUDY 2: INVESTIGATING METRIC SPIKE

TASK 1- Weekly User Engagement:

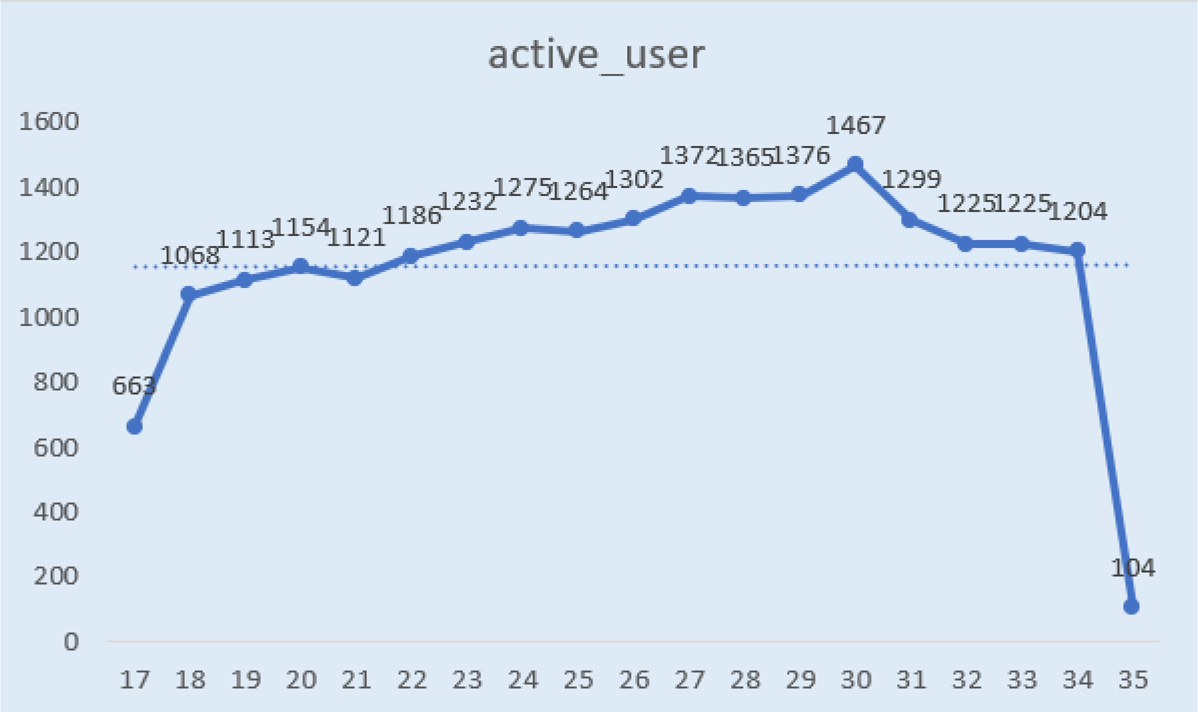
- **Objective: Measure the activeness of users on a weekly basis.**
- **Your Task: Write an SQL query to calculate the weekly user engagement.**



```
##---Task1:Weekly User Engagement----
select extract(week from occurred_at ) as week_number,
count(distinct user_id) as active_user
from event
where event_type='engagement'
group by week_number
order by week_number;

##---- Task2:User Growth Analysis-----

select
```

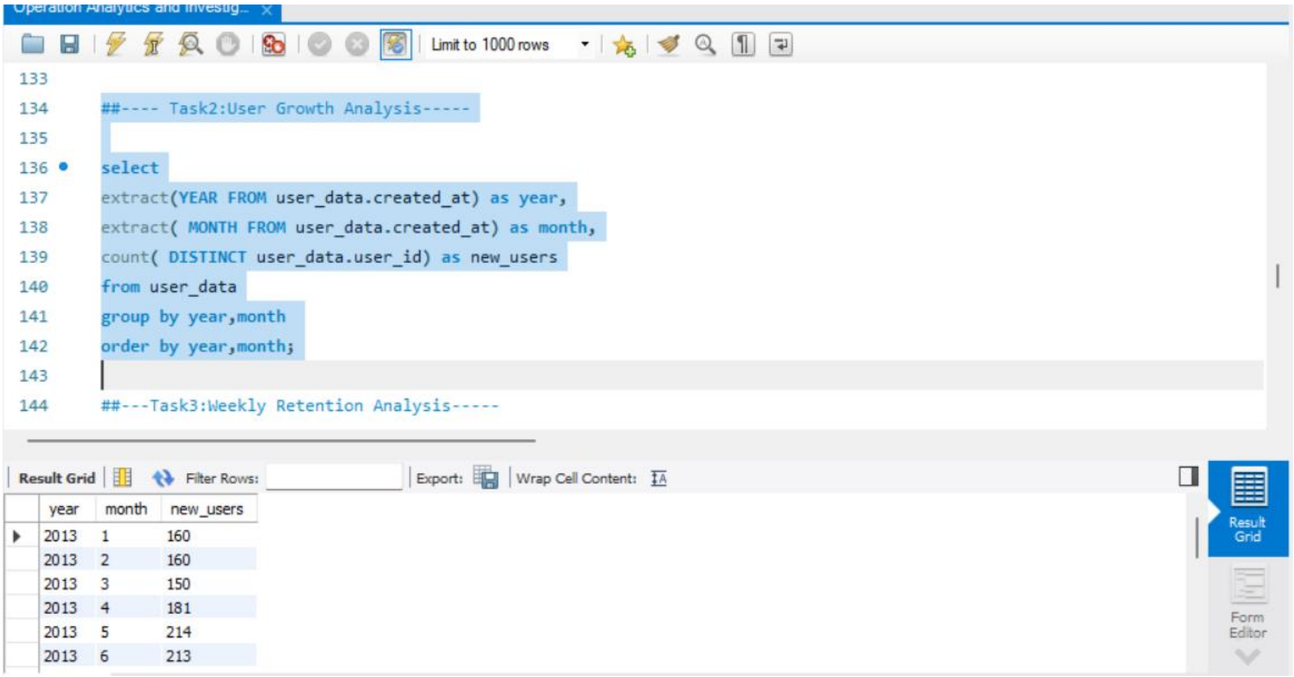
week_number	active_user
17	663
18	1068
19	1113
20	1154
21	1121
22	1186

Explanation: This query calculates the count of distinct user IDs for each week to measure user engagement on a weekly basis.



TASK 2 - User Growth Analysis:

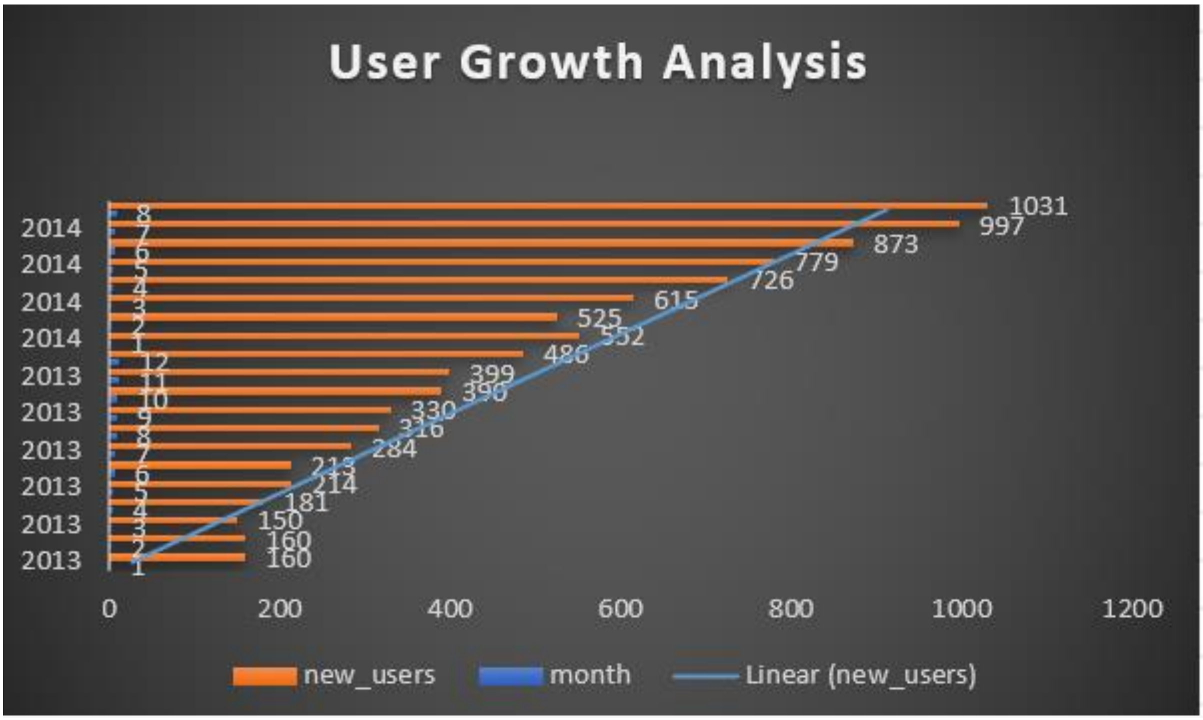
- **Objective: Analyse the growth of users over time for a product.**
- **Your Task: Write an SQL query to calculate the user growth for the product.**
-



```sql
##---- Task2:User Growth Analysis-----

select
extract(YEAR FROM user_data.created_at) as year,
extract( MONTH FROM user_data.created_at) as month,
count( DISTINCT user_data.user_id) as new_users
from user_data
group by year,month
order by year,month;

##---Task3:Weekly Retention Analysis-----
```

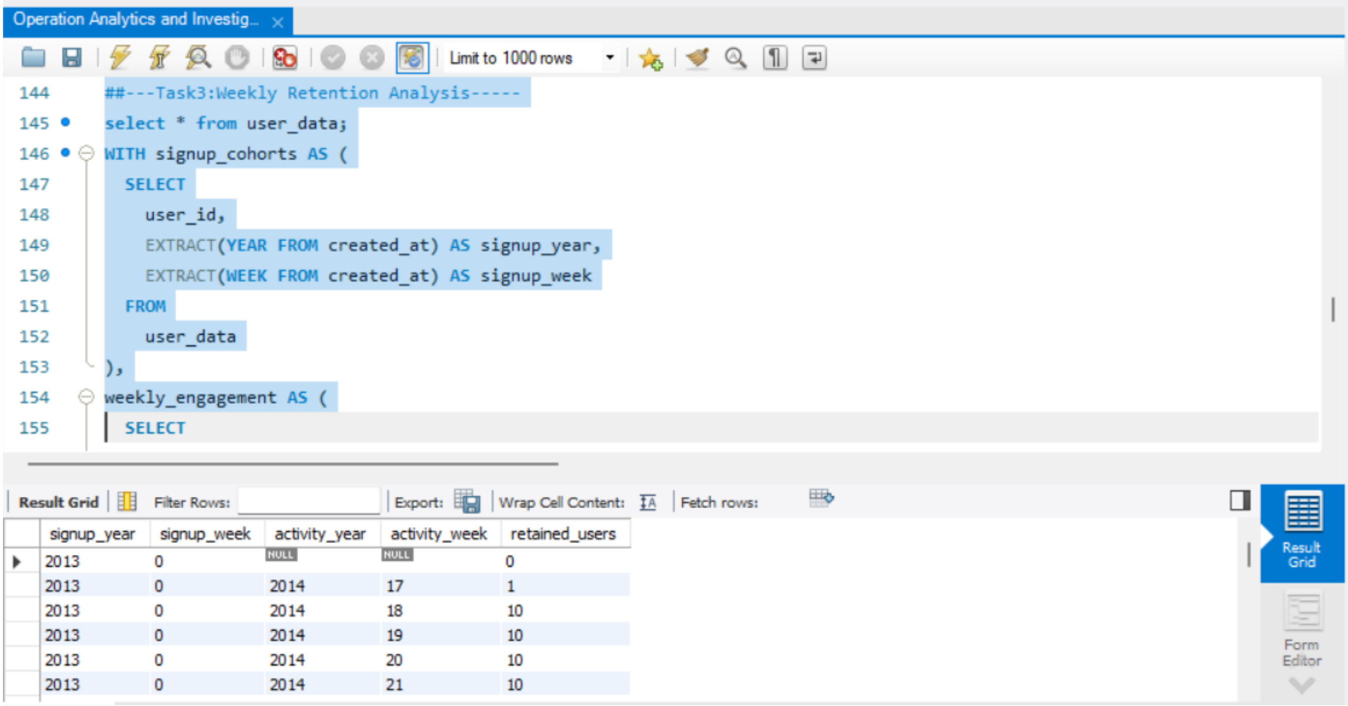
year	month	new_users
2013	1	160
2013	2	160
2013	3	150
2013	4	181
2013	5	214
2013	6	213

Explanation: This query tracks the growth of users over time by counting distinct user IDs for each month.



TASK 3 - Weekly Retention Analysis:

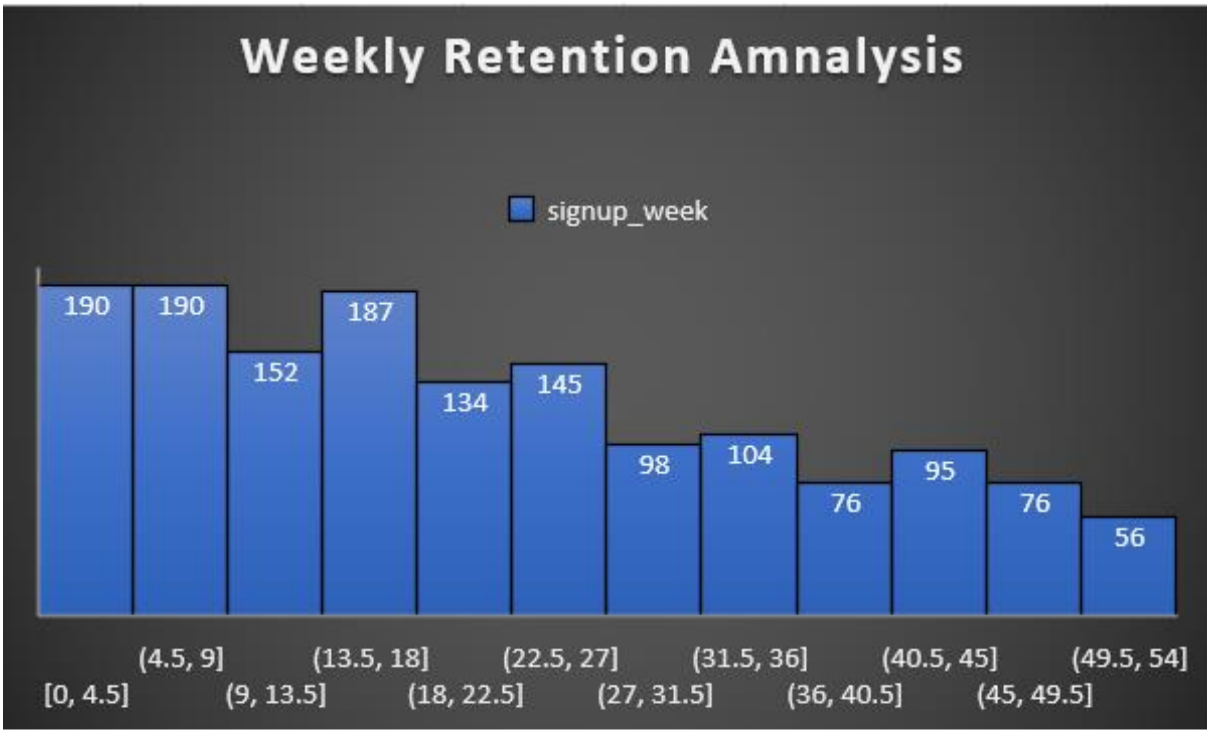
- ➢ **Objective: Analyse the retention of users on a weekly basis after signing up for a product.**
- ➢ **Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.**



```
##---Task3:Weekly Retention Analysis-----
select * from user_data;
WITH signup_cohorts AS (
  SELECT
    user_id,
    EXTRACT(YEAR FROM created_at) AS signup_year,
    EXTRACT(WEEK FROM created_at) AS signup_week
  FROM
    user_data
),
weekly_engagement AS (
  SELECT
```

signup_year	signup_week	activity_year	activity_week	retained_users
2013	0	NULL	NULL	0
2013	0	2014	17	1
2013	0	2014	18	10
2013	0	2014	19	10
2013	0	2014	20	10
2013	0	2014	21	10

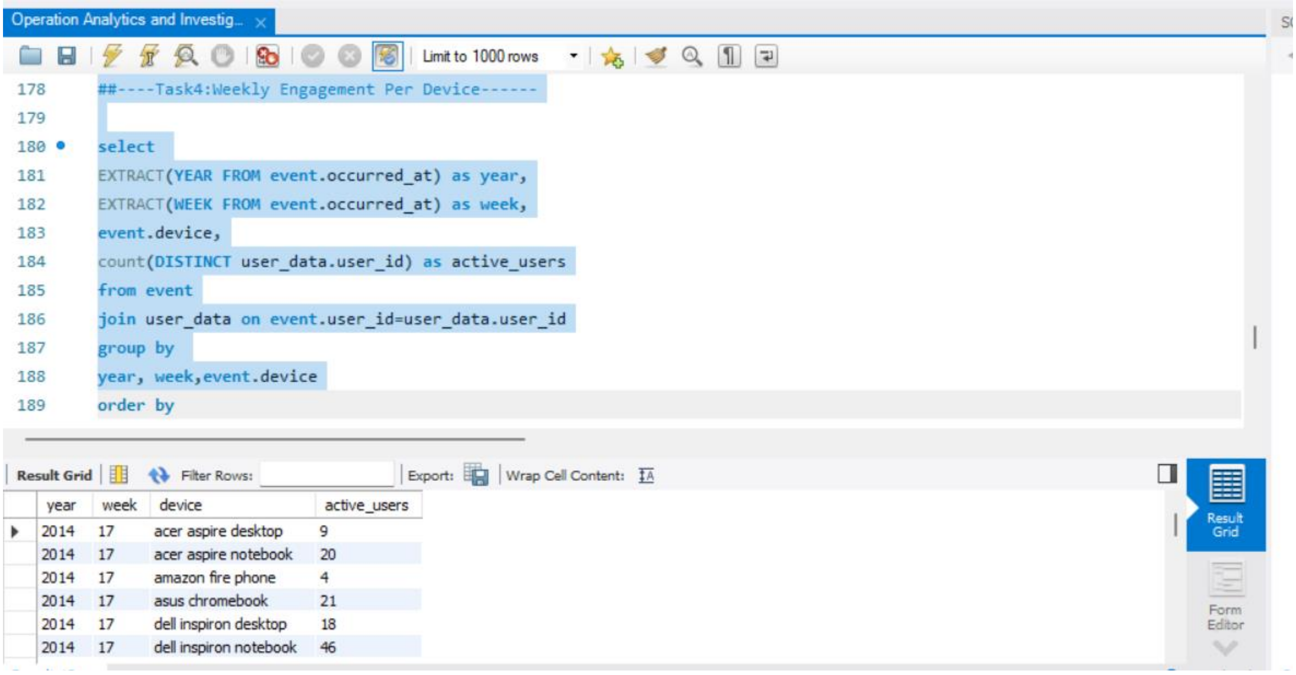
Explanation: This query calculates the weekly retention of users by comparing the week they signed up with subsequent weekly activity.



The query provides with a table showing the number of users who signed up in a particular week and how many of them were retained in subsequent weeks. This data helps you analyse user retention over time and understand the effectiveness of user engagement and retention strategies.

TASK 4 - Weekly Engagement Per Device:

- **Objective: Measure the activeness of users on a weekly basis per device.**
- **Your Task: Write an SQL query to calculate the weekly engagement per device**.
-

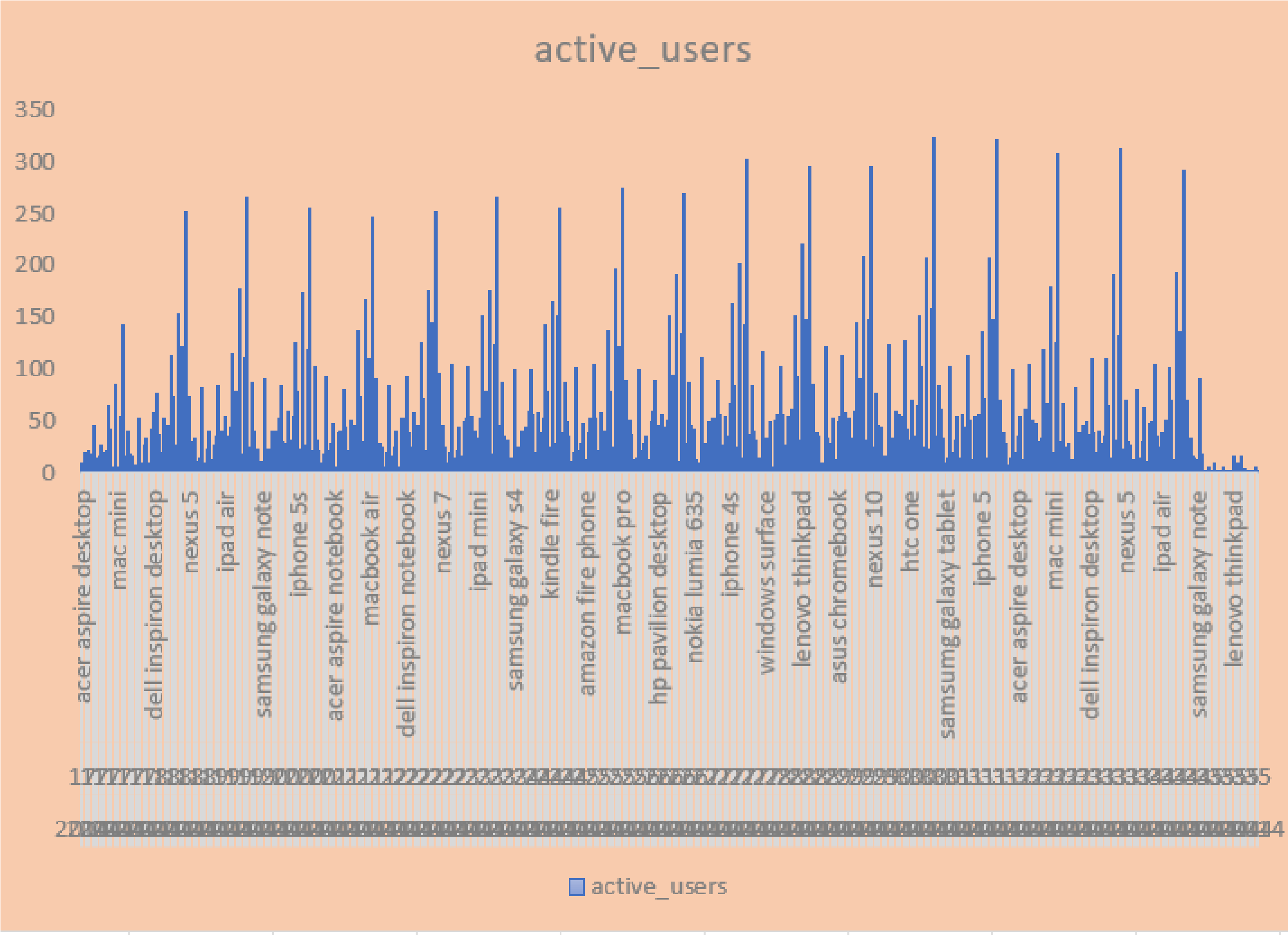


```
##----Task4:Weekly Engagement Per Device------

select
EXTRACT(YEAR FROM event.occurred_at) as year,
EXTRACT(WEEK FROM event.occurred_at) as week,
event.device,
count(DISTINCT user_data.user_id) as active_users
from event
join user_data on event.user_id=user_data.user_id
group by
year, week,event.device
order by
```

year	week	device	active_users
2014	17	acer aspire desktop	9
2014	17	acer aspire notebook	20
2014	17	amazon fire phone	4
2014	17	asus chromebook	21
2014	17	dell inspiron desktop	18
2014	17	dell inspiron notebook	46

Explanation: The query output provides you with a table that shows the weekly user engagement per device. For each week and device type (device), you can see the count of distinct users who engaged with the product using that specific device during that week.

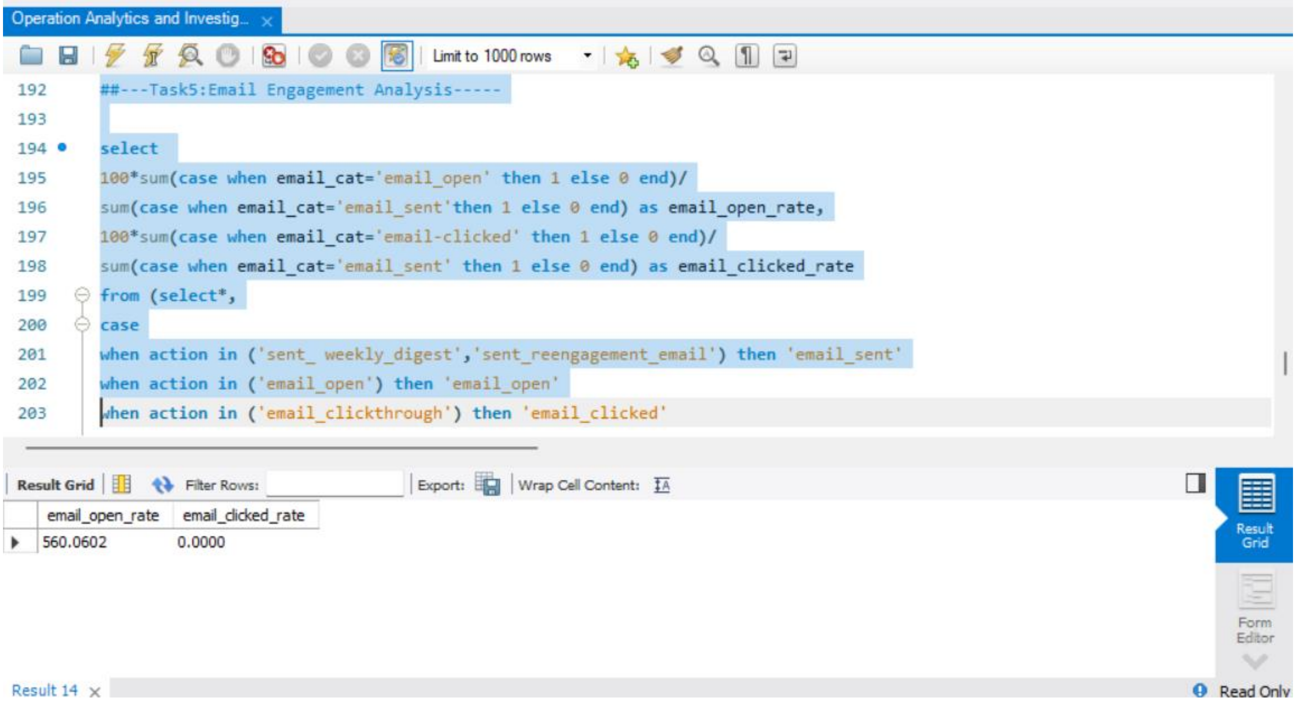
This data allows you to track and analyse user engagement patterns across different devices over time. It can help you understand which devices are most popular among your users and how user engagement on various devices evolves from week to week. Such insights can be valuable for making decisions related to device-specific optimizations and user experience enhancements.



TASK 5- Email Engagement Analysis:

- **Objective: Analyze how users are engaging with the email service.**
- **Your Task: Write an SQL query to calculate the email engagement metrics.**

○



```sql
##---Task5:Email Engagement Analysis-----

select
100*sum(case when email_cat='email_open' then 1 else 0 end)/
sum(case when email_cat='email_sent'then 1 else 0 end) as email_open_rate,
100*sum(case when email_cat='email-clicked' then 1 else 0 end)/
sum(case when email_cat='email_sent' then 1 else 0 end) as email_clicked_rate
from (select*,
case
when action in ('sent_ weekly_digest','sent_reengagement_email') then 'email_sent'
when action in ('email_open') then 'email_open'
when action in ('email_clickthrough') then 'email_clicked'
```

```
  email_open_rate   email_clicked_rate
▶ 560.0602          0.0000
```

Explanation: Output is valuable for understanding how users interact with emails sent by the company. It helps in assessing the effectiveness of email campaigns, identifying which types of email engagement events are most common, and gaining insights into user behaviour related to email communications.

For instance, we can use this information to evaluate the success of email marketing efforts, track user engagement trends over time, and make data-driven decisions to optimize email content and strategies.

Users are engaging with the email service: email_open_rate is 33.5834

RESULT

The project's outcomes will include:

- Data-driven insights and recommendations to improve various aspects of the company's operations.
- A clear understanding of the impact of different operational factors on key metrics.
- Improved decision-making processes within the organization.
- Enhanced operational efficiency and potentially increased profitability for the company.
- A valuable contribution to the company's overall success by leveraging data analysis and operational insights