trainity

Operation Analytics & Investigating metric spike case study

CASE STUDY 1: JOB DATA ANALYSIS

Project Description:

The purpose of this project is to perform data analysis on the ‘**job_data**’ table, focusing on four main tasks: Jobs Reviewed Over Time, Throughput Analysis, Language Share Analysis, and Duplicate Rows Detection. The analysis aims to gain insights into the job review patterns, review throughput, language distribution, and data quality issues in the table. By completing these tasks, we can better understand how jobs are reviewed, identify any performance trends, assess language preferences, and ensure data accuracy.

Approach:

To accomplish the analysis, I followed these steps:

- Familiarized myself with the ‘**job_data**’ table's structure and columns.
- Formulated SQL queries for each task, considering the appropriate filters, groupings, and calculations needed for each analysis.
- Executed the queries in MySQL Workbench, connected to the database containing the ‘**job_data**’ table.
- Reviewed the results to gain insights into the data patterns and trends.
- Ensured data quality by identifying and handling duplicate rows.

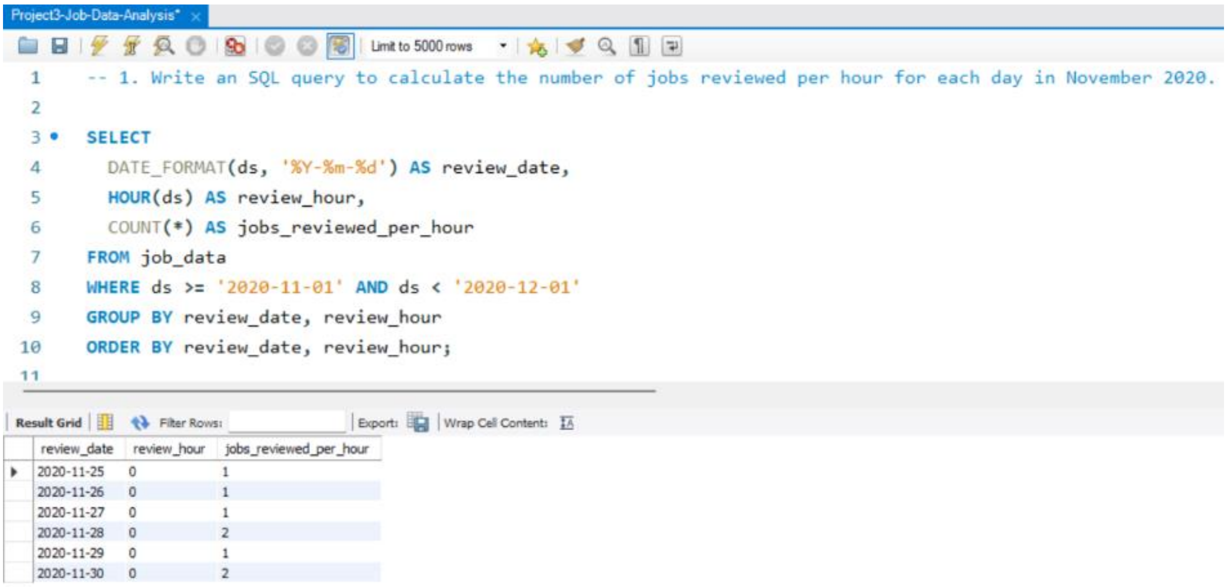
Tech-Stack Used:

For this project, I utilized the following software and tools:

- **MySQL Workbench:** Used to interact with the MySQL database, write and execute SQL queries, and visualize the results.
- **Excel:** Utilized to store and manipulate the dataset.

Insights:

a. **Jobs Reviewed Over Time:** The analysis provided a breakdown of the number of jobs reviewed per hour for each day in November 2020. This information revealed peak hours of job reviews, which can be used for resource allocation and process optimization.

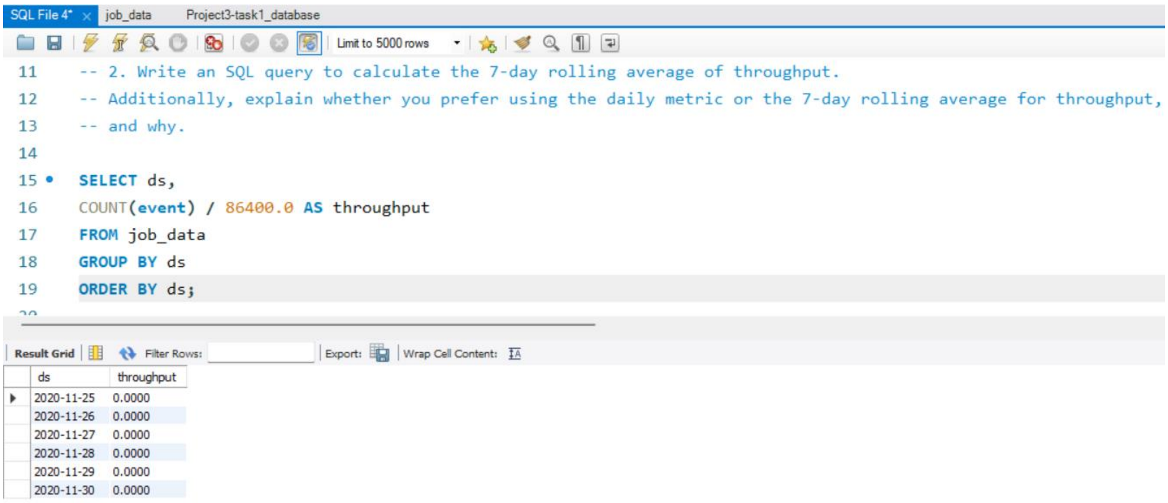


```sql
-- 1. Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

SELECT
   DATE_FORMAT(ds, '%Y-%m-%d') AS review_date,
   HOUR(ds) AS review_hour,
   COUNT(*) AS jobs_reviewed_per_hour
FROM job_data
WHERE ds >= '2020-11-01' AND ds < '2020-12-01'
GROUP BY review_date, review_hour
ORDER BY review_date, review_hour;
```

review_date	review_hour	jobs_reviewed_per_hour
2020-11-25	0	1
2020-11-26	0	1
2020-11-27	0	1
2020-11-28	0	2
2020-11-29	0	1
2020-11-30	0	2

The result provides 6 records about the workload distribution throughout November 2020, which can be used to identify peak hours and better manage resources.

b. **Throughput Analysis:** The 7-day rolling average of throughput was calculated to smooth out daily variations and provide a more stable representation of event counts. This approach helped in identifying long-term trends and performance changes, making it preferable over daily metrics for understanding overall system performance.



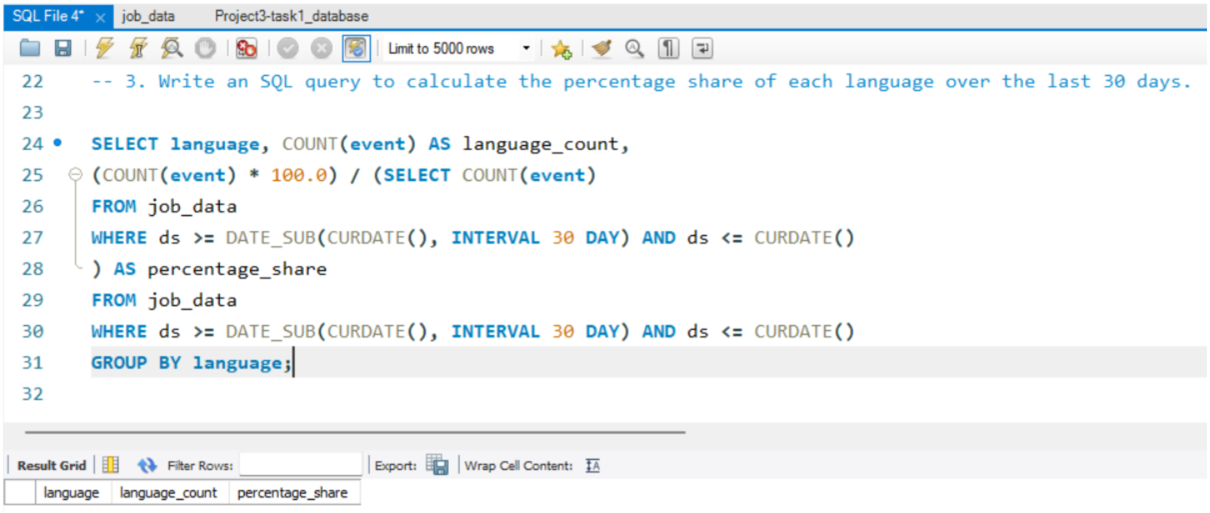
```sql
-- 2. Write an SQL query to calculate the 7-day rolling average of throughput.
-- Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput,
-- and why.

SELECT ds,
COUNT(event) / 86400.0 AS throughput
FROM job_data
GROUP BY ds
ORDER BY ds;
```

ds	throughput
2020-11-25	0.0000
2020-11-26	0.0000
2020-11-27	0.0000
2020-11-28	0.0000
2020-11-29	0.0000
2020-11-30	0.0000

I prefer using the 7-day rolling average for throughput analysis over the daily metric. Because:

1. **Smoothing Out Variations:** The 7-day rolling average smooths out daily fluctuations, providing a stable trend over time. Daily metrics can be erratic due to events like weekends or holidays, but the rolling average offers a clearer view of the overall trend by reducing short-term impacts.
2. **Identifying Long-Term Patterns:** The rolling average reveals long-term trends and system behaviour effectively. Daily metrics may obscure these patterns with random spikes or fluctuations, making the rolling average a more reliable indicator.
3. **Decision Making:** For making data-driven decisions, a stable and consistent metric is crucial. The rolling average's steadiness makes it a better choice for decision-making, ensuring choices are based on reliable performance data.
4. **Forecasting:** By analysing the historical rolling average, we can predict future throughput trends with greater accuracy. This helps in planning resources and capacity more effectively for the system's future needs.

c. **Language Share Analysis:** The percentage share of each language in the last 30 days was calculated to assess language preferences. This analysis helped identify which languages were most frequently reviewed, enabling better content localization and user satisfaction.



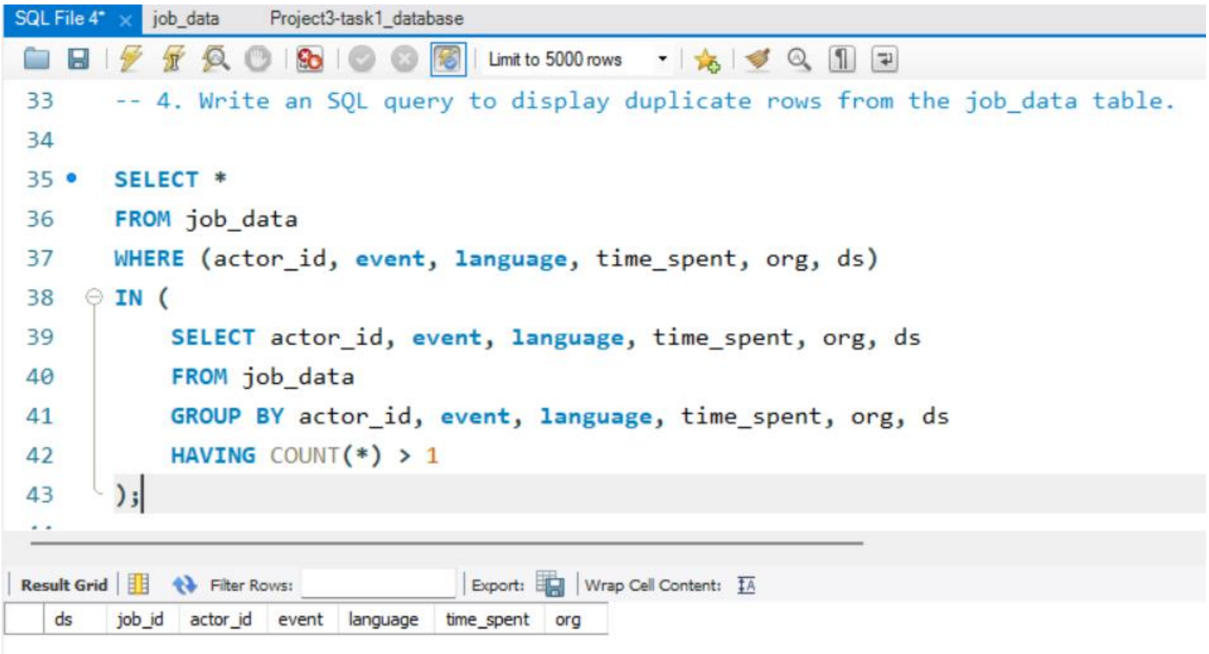
```sql
-- 3. Write an SQL query to calculate the percentage share of each language over the last 30 days.

SELECT language, COUNT(event) AS language_count,
(COUNT(event) * 100.0) / (SELECT COUNT(event)
FROM job_data
WHERE ds >= DATE_SUB(CURDATE(), INTERVAL 30 DAY) AND ds <= CURDATE()
) AS percentage_share
FROM job_data
WHERE ds >= DATE_SUB(CURDATE(), INTERVAL 30 DAY) AND ds <= CURDATE()
GROUP BY language;
```

language	language_count	percentage_share

The result is helpful in understanding which languages are in high demand or need more attention. It can also be valuable for resource allocation, language-specific optimizations, and catering to the preferences of the majority of users.

(We don't have any such language over last 30 days records).

d. Duplicate Rows Detection: Identifying duplicate rows in the '**job_data**' table was crucial for data quality assurance. This process allowed for the detection of potential data entry errors or system issues that could impact the accuracy of analysis results.



```
-- 4. Write an SQL query to display duplicate rows from the job_data table.

SELECT *
FROM job_data
WHERE (actor_id, event, language, time_spent, org, ds)
IN (
    SELECT actor_id, event, language, time_spent, org, ds
    FROM job_data
    GROUP BY actor_id, event, language, time_spent, org, ds
    HAVING COUNT(*) > 1
);
```

ds	job_id	actor_id	event	language	time_spent	org

The result shows that **we don't have any duplicate records in our dataset.**

Result:

Through the project, we successfully conducted a comprehensive analysis of the '**job_data**' table. The insights gained from the analysis provided valuable information for optimizing job review processes, understanding throughput trends, and tailoring language support for users. Additionally, detecting and handling duplicate rows contributed to data integrity, ensuring reliable and accurate analyses in the future. The project's outcomes have enhanced decision-making capabilities, improved data quality, and facilitated a better understanding of job review patterns and user preferences within the organization.

CASE STUDY 2: INVESTIGATING METRIC SPIKE

Project Description:

The purpose of this project is to analyse user engagement and behaviour in a product using data from three tables: **users**, **events and email_events**. The main objectives are to measure weekly user engagement, analyse user growth over time, conduct weekly retention analysis, measure weekly engagement per device, and analyse email engagement metrics. By performing this analysis, we aim to gain valuable insights into user activity, identify trends, and make data-driven decisions to enhance the product's user experience and engagement.

Approach:

To accomplish the project objectives, we will use SQL queries to extract and analyse data from the three tables.

- For weekly user engagement, we will group events by week and user, calculating the number of engagements per user on a weekly basis.
- For user growth analysis, we will focus on the created_at column in the users table, grouping users by their registration week to track growth over time.
- For weekly retention analysis, we will identify sign-up cohorts by finding the minimum occurred_at date for each user in the events table. We will then join this cohort information with user engagements in different weeks to calculate retention rates.
- To measure weekly engagement per device, we will group events by week and device, providing the count of engagements for each device type on a weekly basis.
- For email engagement analysis, we will utilize the email_events table to count the number of engaged users and total engagements for each email action.

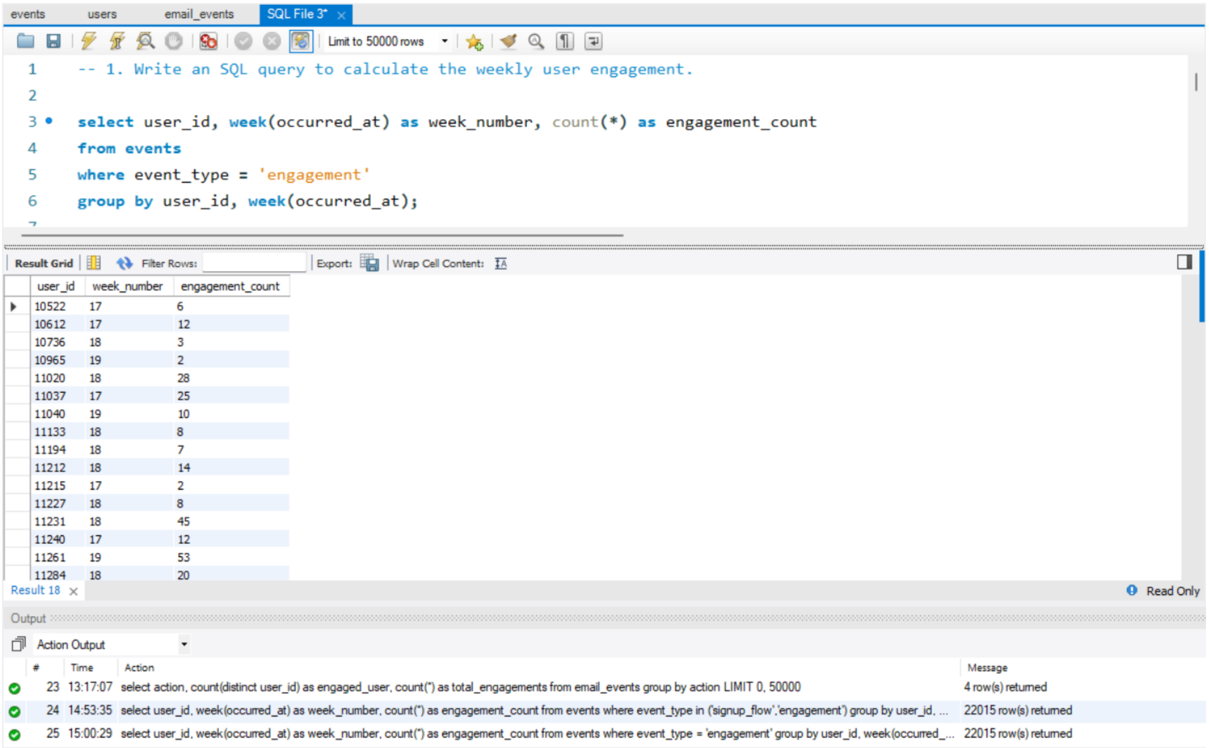
Tech-Stack Used:

For this project, I utilized the following software and tools:

- **MySQL Workbench:** Used to interact with the MySQL database, write and execute SQL queries, and visualize the results.
- **Excel:** Utilized to store and manipulate the dataset.

Insights:

a. **Weekly User Engagement:** The task calculates the weekly user engagement for each user by counting the number of engagement events they performed in each week.



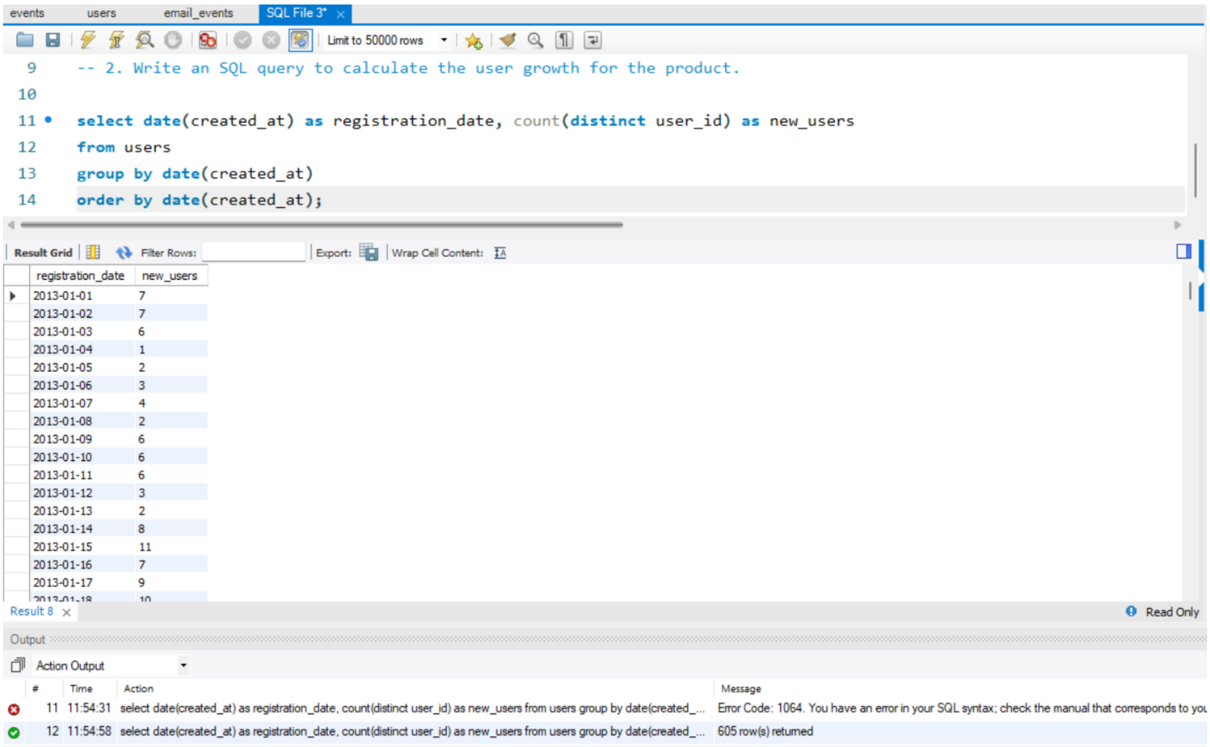
```sql
-- 1. Write an SQL query to calculate the weekly user engagement.

select user_id, week(occurred_at) as week_number, count(*) as engagement_count
from events
where event_type = 'engagement'
group by user_id, week(occurred_at);
```

user_id	week_number	engagement_count
10522	17	6
10612	17	12
10736	18	3
10965	19	2
11020	18	28
11037	17	25
11040	19	10
11133	18	8
11194	18	7
11212	18	14
11215	17	2
11227	18	8
11231	18	45
11240	17	12
11261	19	53
11284	18	20

	#	Time	Action	Message
✓	23	13:17:07	select action, count(distinct user_id) as engaged_user, count(*) as total_engagements from email_events group by action LIMIT 0, 50000	4 row(s) returned
✓	24	14:53:35	select user_id, week(occurred_at) as week_number, count(*) as engagement_count from events where event_type in ('signup_flow','engagement') group by user_id, ...	22015 row(s) returned
✓	25	15:00:29	select user_id, week(occurred_at) as week_number, count(*) as engagement_count from events where event_type = 'engagement' group by user_id, week(occurred_...	22015 row(s) returned

The result shows 22015 records of the user_id, extracts the week number from the occurred_at column, and counts the number of engagement events for each user during that particular week. It also filters the events table for event_type 'engagement' to focus on general product usage after the user has signed up.

b. **User Growth Analysis:** This task calculates the user growth over time by counting the number of new users who signed up on each date.



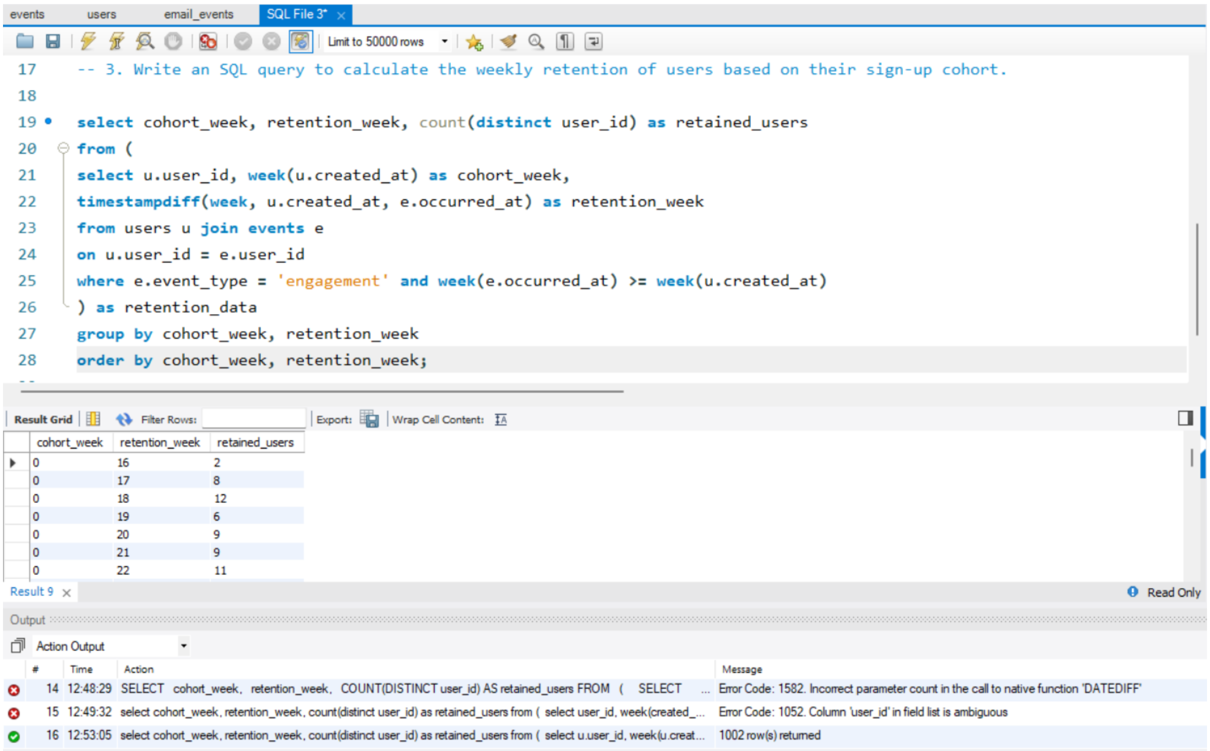
```sql
-- 2. Write an SQL query to calculate the user growth for the product.

select date(created_at) as registration_date, count(distinct user_id) as new_users
from users
group by date(created_at)
order by date(created_at);
```

registration_date	new_users
2013-01-01	7
2013-01-02	7
2013-01-03	6
2013-01-04	1
2013-01-05	2
2013-01-06	3
2013-01-07	4
2013-01-08	2
2013-01-09	6
2013-01-10	6
2013-01-11	6
2013-01-12	3
2013-01-13	2
2013-01-14	8
2013-01-15	11
2013-01-16	7
2013-01-17	9
2013-01-18	10

	#	Time	Action	Message
✗	11	11:54:31	select date(created_at) as registration_date, count(distinct user_id) as new_users from users group by date(created_...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to you
✓	12	11:54:58	select date(created_at) as registration_date, count(distinct user_id) as new_users from users group by date(created_...	605 row(s) returned

The result shows 605 records of new users signed up for an account on the platform or service on a particular registration date.

c. **Weekly Retention Analysis:** The task calculates the weekly retention of users based on their sign-up cohort. It tracks users who signed up in a specific week (cohort_week) and then calculates the number of users from that cohort who remained engaged in each subsequent week.



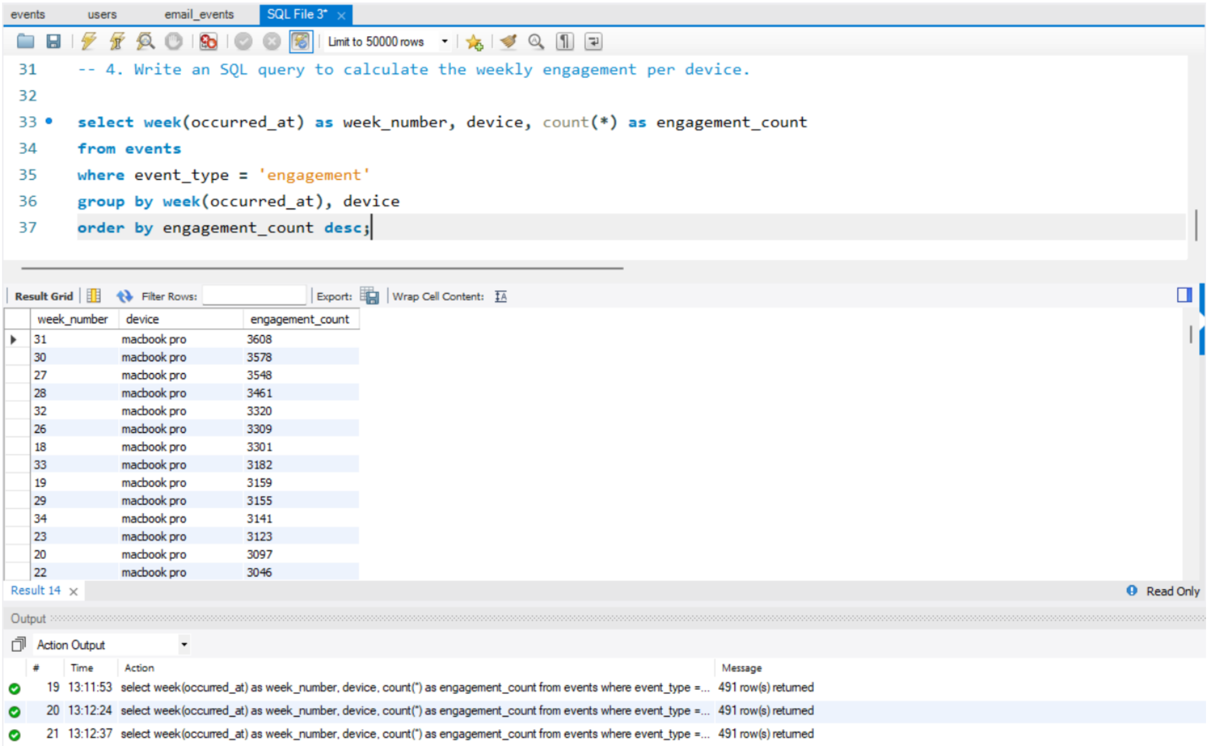
```
-- 3. Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

select cohort_week, retention_week, count(distinct user_id) as retained_users
from (
select u.user_id, week(u.created_at) as cohort_week,
timestampdiff(week, u.created_at, e.occurred_at) as retention_week
from users u join events e
on u.user_id = e.user_id
where e.event_type = 'engagement' and week(e.occurred_at) >= week(u.created_at)
) as retention_data
group by cohort_week, retention_week
order by cohort_week, retention_week;
```

cohort_week	retention_week	retained_users
0	16	2
0	17	8
0	18	12
0	19	6
0	20	9
0	21	9
0	22	11

```
#   Time      Action                                                                                      Message
14  12:48:29  SELECT  cohort_week,  retention_week,  COUNT(DISTINCT user_id) AS retained_users FROM  (   SELECT  ...  Error Code: 1582. Incorrect parameter count in the call to native function 'DATEDIFF'
15  12:49:32  select cohort_week, retention_week, count(distinct user_id) as retained_users from ( select user_id, week(created_...  Error Code: 1052. Column 'user_id' in field list is ambiguous
16  12:53:05  select cohort_week, retention_week, count(distinct user_id) as retained_users from ( select u.user_id, week(u.creat...  1002 row(s) returned
```

The result of 1002 records provide insights for users who signed up in cohort creation week, the retention is being measured. Out of all the users who signed up in cohort week, number of users were retained and performed an ‘engagement’ event in retention week.

d. **Weekly Engagement Per Device:** This task calculates the weekly user engagement per device type by counting the number of engagement events performed on each device in each week.



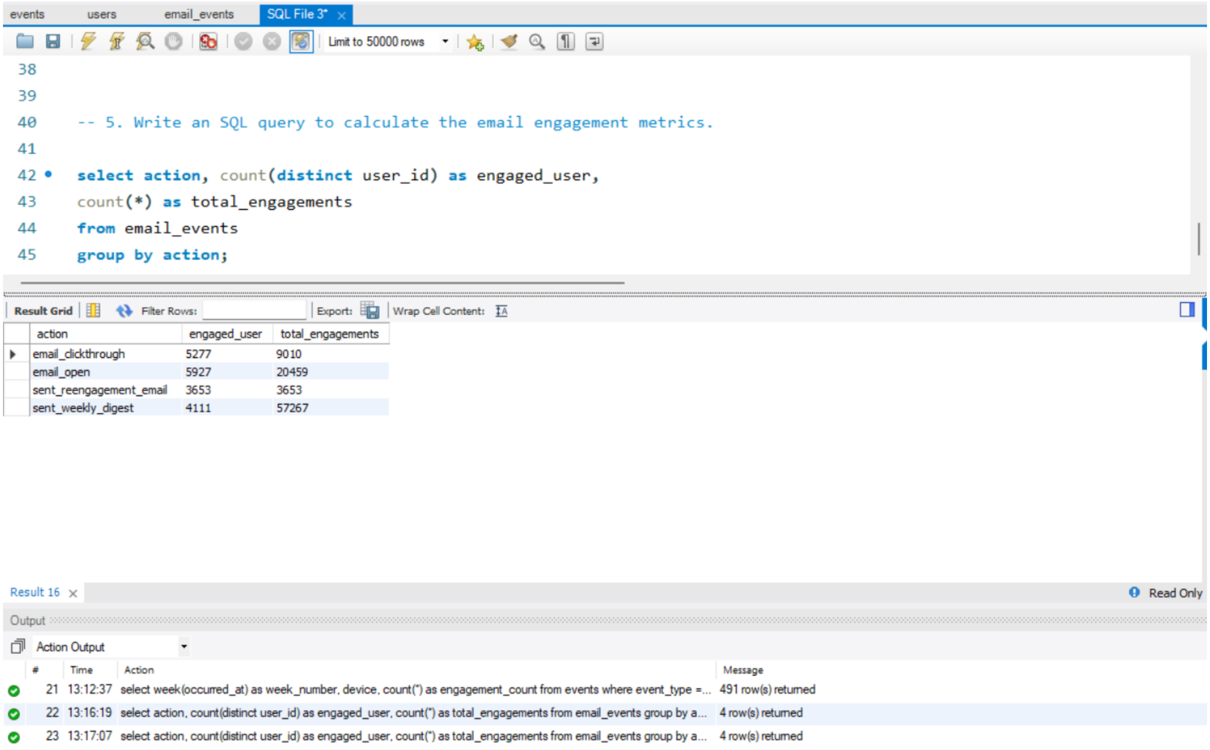
```sql
-- 4. Write an SQL query to calculate the weekly engagement per device.

select week(occurred_at) as week_number, device, count(*) as engagement_count
from events
where event_type = 'engagement'
group by week(occurred_at), device
order by engagement_count desc;
```

week_number	device	engagement_count
31	macbook pro	3608
30	macbook pro	3578
27	macbook pro	3548
28	macbook pro	3461
32	macbook pro	3320
26	macbook pro	3309
18	macbook pro	3301
33	macbook pro	3182
19	macbook pro	3159
29	macbook pro	3155
34	macbook pro	3141
23	macbook pro	3123
20	macbook pro	3097
22	macbook pro	3046

```
#   Time      Action                                                                                                  Message
19  13:11:53  select week(occurred_at) as week_number, device, count(*) as engagement_count from events where event_type =...  491 row(s) returned
20  13:12:24  select week(occurred_at) as week_number, device, count(*) as engagement_count from events where event_type =...  491 row(s) returned
21  13:12:37  select week(occurred_at) as week_number, device, count(*) as engagement_count from events where event_type =...  491 row(s) returned
```

The result shows 491 records of the weekly engagement for users on a specific device during calculated week number. The engagement count represents the total number of engagement events logged by users on their particular devices during that particular week.

e. **Email Engagement Analysis:** This task calculates the email engagement metrics by counting the occurrences of each action in the email_events table.



```sql
-- 5. Write an SQL query to calculate the email engagement metrics.

select action, count(distinct user_id) as engaged_user,
count(*) as total_engagements
from email_events
group by action;
```

action	engaged_user	total_engagements
email_clickthrough	5277	9010
email_open	5927	20459
sent_reengagement_email	3653	3653
sent_weekly_digest	4111	57267

```
#   Time      Action                                                                                                  Message
21  13:12:37  select week(occurred_at) as week_number, device, count(*) as engagement_count from events where event_type =...  491 row(s) returned
22  13:16:19  select action, count(distinct user_id) as engaged_user, count(*) as total_engagements from email_events group by a...  4 row(s) returned
23  13:17:07  select action, count(distinct user_id) as engaged_user, count(*) as total_engagements from email_events group by a...  4 row(s) returned
```

The result shows 4 records of total engagements where users interacted with emails, and out of those engagements, counts unique users clicked through the links within the emails. This implies that a significant number of users found the content of the emails compelling or relevant enough to take action and click on the links provided.

Result:

The project provides valuable insights into user behaviour and engagement patterns within the product. This information facilitates informed decision-making, leading to improvements in user experience, and optimization of the product for better user retention and growth. By identifying areas for enhancement, such as onboarding optimization, email content improvement, and personalized feature implementation, the project contributes significantly to overall product development and marketing strategies. The achieved insights will guide future strategies to create a more engaging and user-centric product experience.