Operation and Metric Analysis

Description:

Operational Analytics plays a vital role in improving business efficiency by analysing data from various processes to identify patterns, trends, and areas for optimization. This project focuses on leveraging SQL-based analysis to extract insights from company data and support data-driven decision-making. The project consists of two case studies that address different analytical challenges:

1. **Job Data Analysis**:- Examining job review trends, throughput efficiency, language distribution, and duplicate data to enhance operational workflow.
2. **Investigating Metric Spikes**:- Analysing fluctuations in user engagement, growth, retention, device activity, and email interactions to understand the factors driving these changes.

Tools Used: MySQL Workbench, SQL.

CASE STUDY 1 : Job data analysis

CREATE DATABASE :

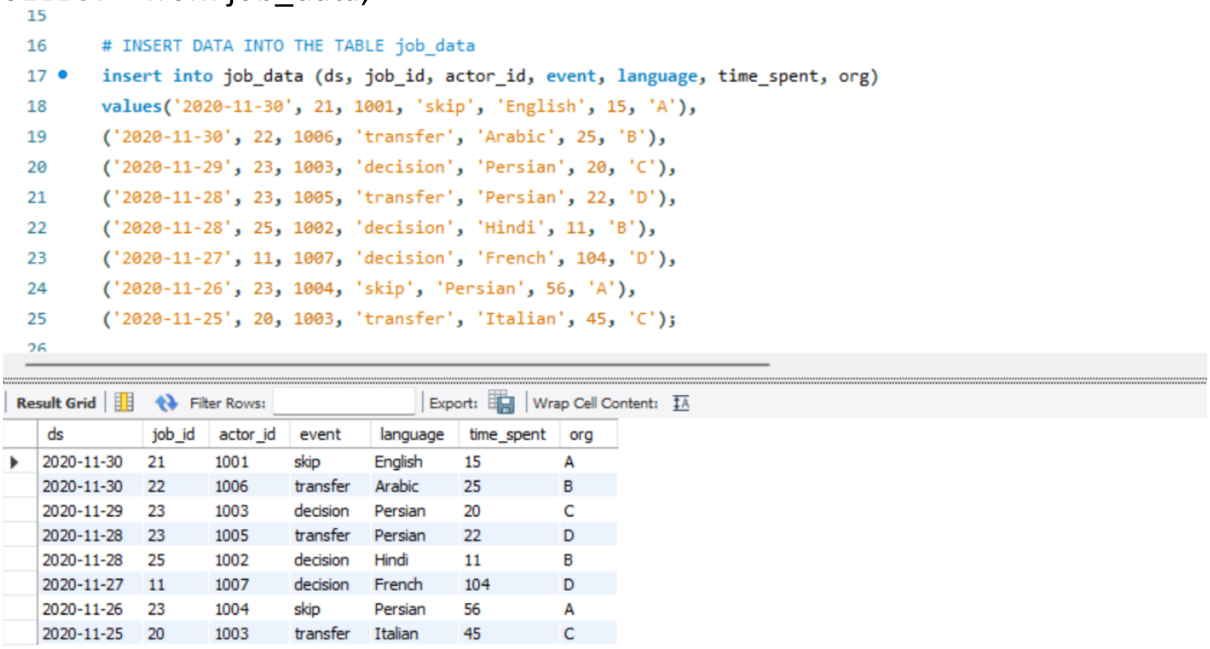
```
create database operation_analytics;
use operation_analytics;
```

CREATE TABLE job_data :

```
CREATE TABLE job_data (
 ds DATE,
 job_id INT NOT NULL,
 actor_id INT NOT NULL,
 event VARCHAR(10) NOT NULL,
 language VARCHAR(10) NOT NULL,
 time_spent INT NOT NULL,
 org CHAR(2)
);
```

INSERT DATA INTO THE TABLE job_data :

```
insert into job_data (ds, job_id, actor_id, event, language, time_spent, org)
values('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'),
('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),
('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');

SELECT * from job_data;
```



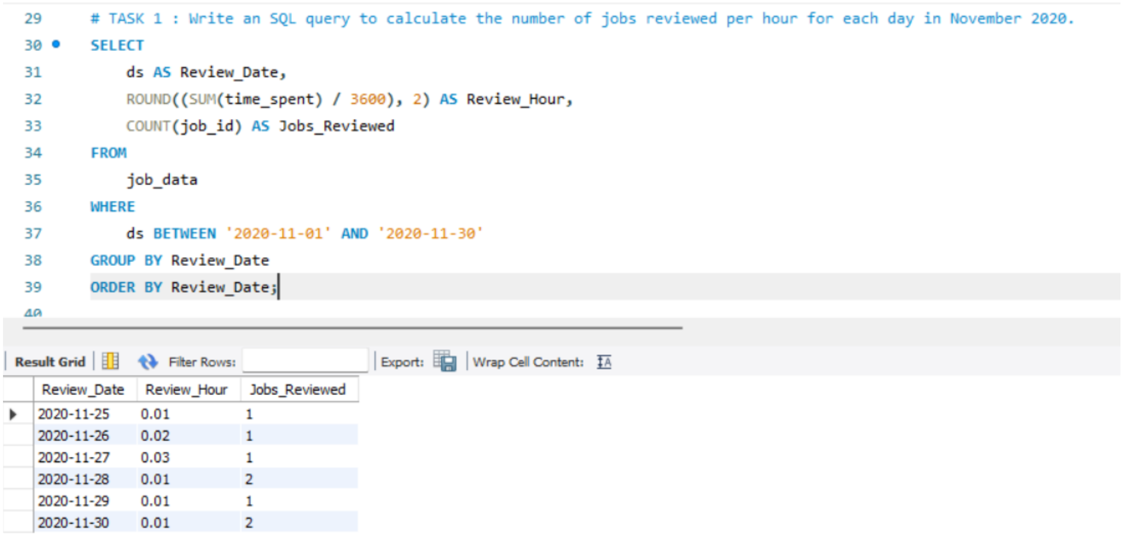
```
# INSERT DATA INTO THE TABLE job_data
insert into job_data (ds, job_id, actor_id, event, language, time_spent, org)
values('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),
('2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),
('2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),
('2020-11-28', 23, 1005, 'transfer', 'Persian', 22, 'D'),
('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),
('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),
('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),
('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');
```

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
2020-11-26	23	1004	skip	Persian	56	A
2020-11-25	20	1003	transfer	Italian	45	C

TASK 1 : Jobs Reviewed Over Time (Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020)

```
SELECT
 ds AS Review_Date,
 ROUND((SUM(time_spent) / 3600), 2) AS Review_Hour,
 COUNT(job_id) AS Jobs_Reviewed
FROM
 job_data
WHERE
```

```
 ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY Review_Date
ORDER BY Review_Date;
```



```
# TASK 1 : Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.
SELECT
    ds AS Review_Date,
    ROUND((SUM(time_spent) / 3600), 2) AS Review_Hour,
    COUNT(job_id) AS Jobs_Reviewed
FROM
    job_data
WHERE
    ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY Review_Date
ORDER BY Review_Date;
```

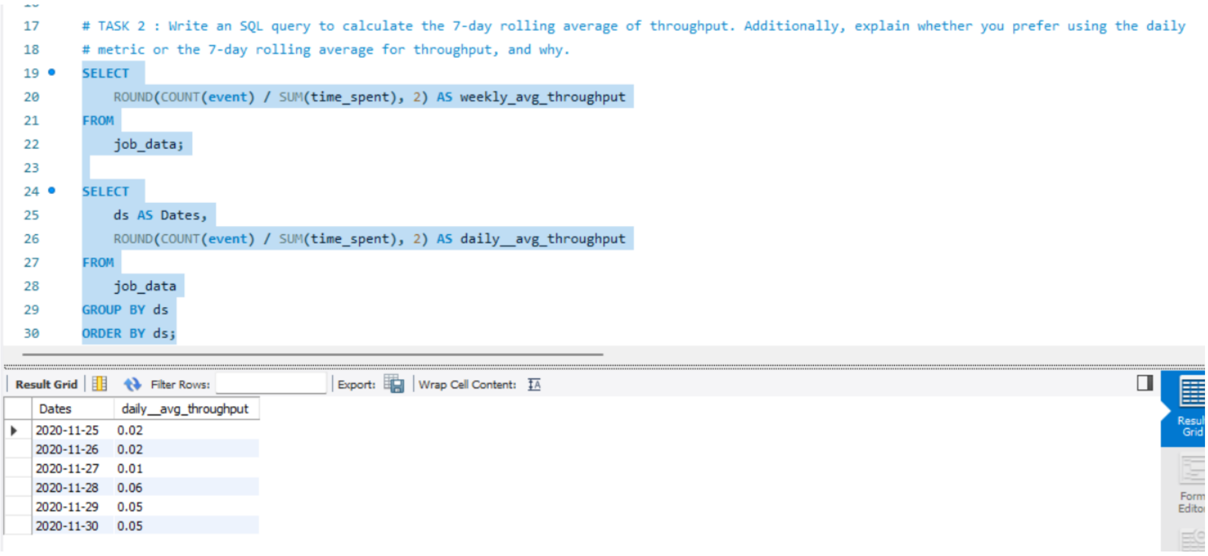
Review_Date	Review_Hour	Jobs_Reviewed
2020-11-25	0.01	1
2020-11-26	0.02	1
2020-11-27	0.03	1
2020-11-28	0.01	2
2020-11-29	0.01	1
2020-11-30	0.01	2

Insights:

- This query helps in understanding the workload distribution across different hours of the day.
- Analysing these trends can help in resource planning and workload balancing.
- If certain hours show a spike in job reviews, this might indicate peak work times for employees.

TASK 2 : Throughput Analysis (Write an SQL query to calculate the 7-day rolling average of throughput)

```
SELECT
 ROUND(COUNT(event) / SUM(time_spent), 2) AS weekly_avg_throughput
FROM
 job_data;

SELECT
 ds AS Dates,
 ROUND(COUNT(event) / SUM(time_spent), 2) AS daily_avg_throughput
FROM
 job_data
GROUP BY ds
ORDER BY ds;
```



```
# TASK 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
    ROUND(COUNT(event) / SUM(time_spent), 2) AS weekly_avg_throughput
FROM
    job_data;

SELECT
    ds AS Dates,
    ROUND(COUNT(event) / SUM(time_spent), 2) AS daily__avg_throughput
FROM
    job_data
GROUP BY ds
ORDER BY ds;
```

Dates	daily__avg_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:

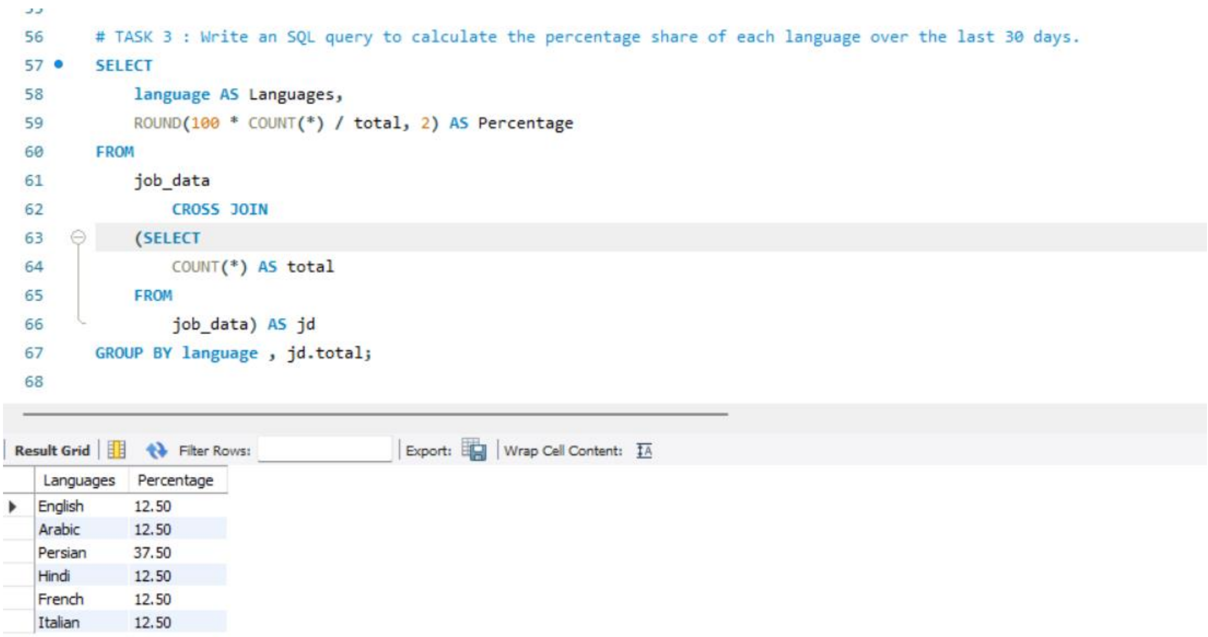
- The 7-day rolling average of throughput is between 0.01 to 0.06.
- The weekly average throughput is 0.03 events per second.
- The 7-day rolling average smooths out daily fluctuations, providing a clearer picture of long-term trends.
- A declining rolling average might indicate reduced engagement or operational slowdowns.

Explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why ?

- ➢ The 7-day rolling average is preferable for analysing throughput.
- ➢ Prefer the 7-day rolling average for tracking throughput because it provides a more stable and reliable view of job review trends. However, daily metrics when investigating short-term anomalies or immediate operational issues.

TASK 3 : Language Share Analysis (Write an SQL query to calculate the percentage share of each language over the last 30 days)

```
SELECT
 language AS Languages,
 ROUND(100 * COUNT(*) / total, 2) AS Percentage
FROM
 job_data
 CROSS JOIN
 (SELECT
 COUNT(*) AS total
 FROM
 job_data) AS jd
GROUP BY language , jd.total;
```



```
# TASK 3 : Write an SQL query to calculate the percentage share of each language over the last 30 days.
SELECT
    language AS Languages,
    ROUND(100 * COUNT(*) / total, 2) AS Percentage
FROM
    job_data
        CROSS JOIN
    (SELECT
        COUNT(*) AS total
    FROM
        job_data) AS jd
GROUP BY language , jd.total;
```

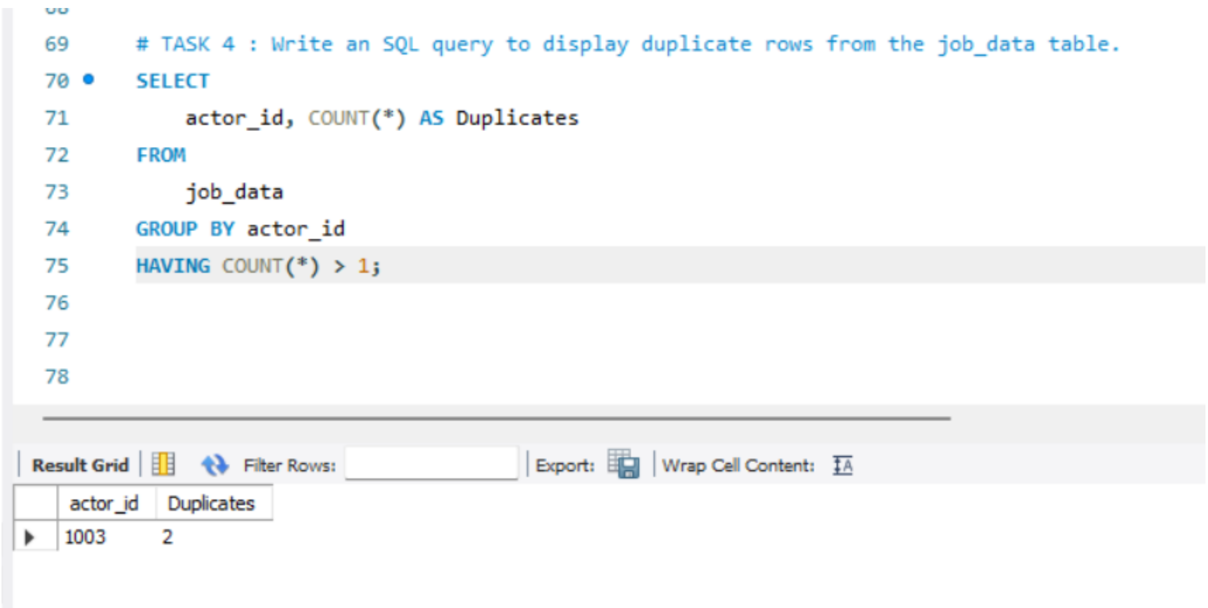
Languages	Percentage
English	12.50
Arabic	12.50
Persian	37.50
Hindi	12.50
French	12.50
Italian	12.50

Insights:

- From the table, we can observe that Persian language is the most used language with the percentage share of 37.50 followed by other languages having a equal share of 12.50 percentage.
- Identifies the dominant languages used in job reviews.
- A sudden rise in a particular language might indicate market expansion or new business needs.

TASK 4 : Duplicate Rows Detection (Write an SQL query to display duplicate rows from the job_data table)

```
SELECT
 actor_id, COUNT(*) AS Duplicates
FROM
 job_data
GROUP BY actor_id
HAVING COUNT(*) > 1;
```



```
# TASK 4 : Write an SQL query to display duplicate rows from the job_data table.
SELECT
    actor_id, COUNT(*) AS Duplicates
FROM
    job_data
GROUP BY actor_id
HAVING COUNT(*) > 1;
```

actor_id	Duplicates
1003	2

Insights:

- Out of total rows, we have 2 duplicate rows.
- The actor-id 1003 is having duplicate.
- Duplicate data can lead to inflated metrics and incorrect insights.
- If duplicates exist, checking data ingestion processes might be necessary to avoid redundancy.

CASE STUDY 2 : Investigating Metric Spike

Create table users :

```
CREATE TABLE users (
 user_id INT,
 created_at VARCHAR(100),
 company_id INT,
 language VARCHAR(50),
 activated_at VARCHAR(100),
 state VARCHAR(50)
);
```

To show path for the data files :

```
show variables like 'SECURE_FILE_PRIV';
```

Upload users data file :

```
load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv"
into table users
```

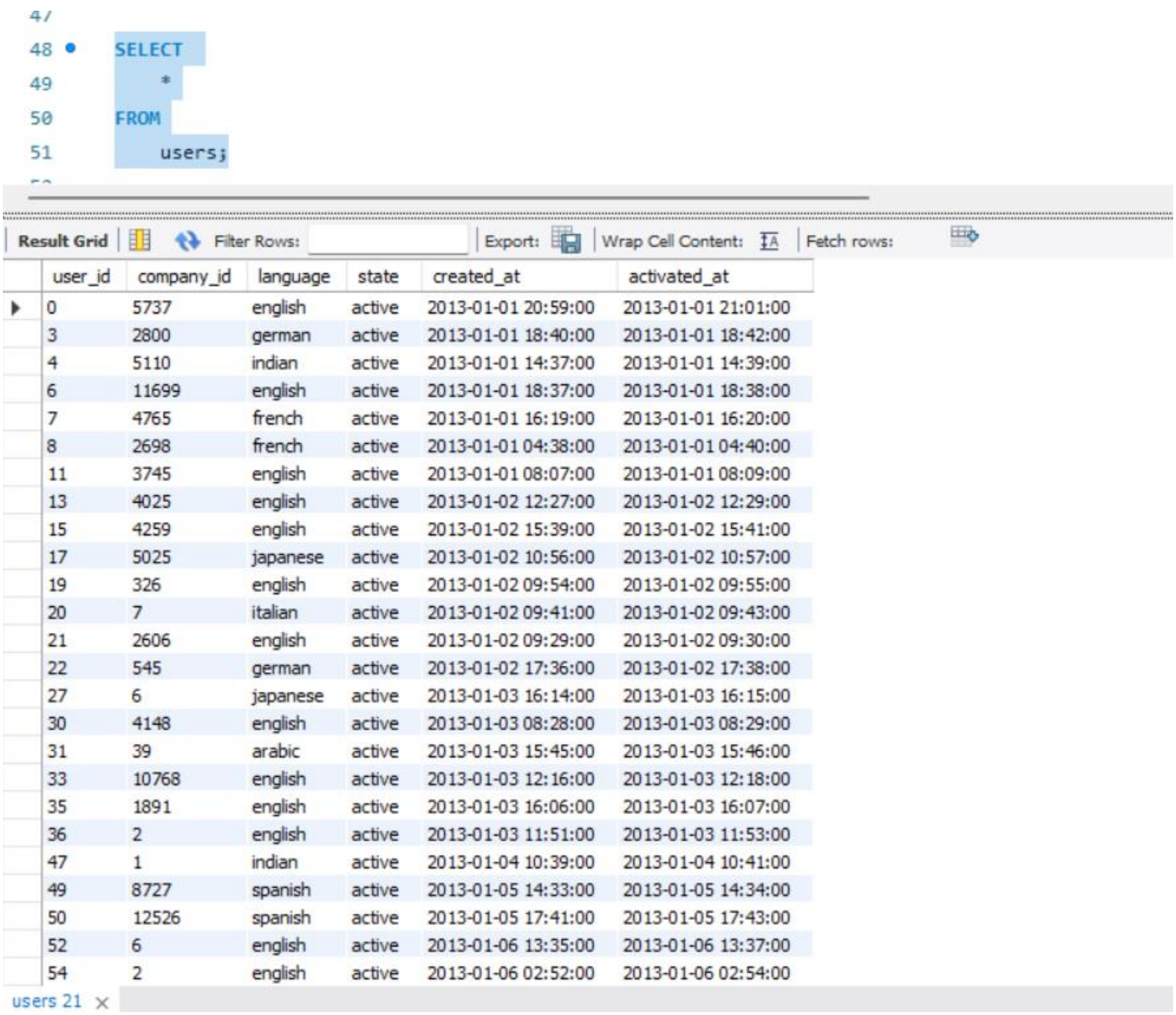
```
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```

Update column created_at(Varchar) into created_at(datetime):

```
alter table users add column temp_created_at datetime;

set SQL_SAFE_UPDATES = 0;

UPDATE users
SET
 temp_created_at = STR_TO_DATE(created_at, '%d-%m-%Y %H:%i');

set SQL_SAFE_UPDATES = 1;

alter table users drop column created_at;

alter table users change column temp_created_at created_at datetime;
```

Update column activated_at(Varchar) into activated_at(datetime):

```
alter table users add column temp_activated_at datetime;

set SQL_SAFE_UPDATES = 0;

UPDATE users
SET
 temp_activated_at = STR_TO_DATE(activated_at, '%d-%m-%Y %H:%i');

set SQL_SAFE_UPDATES = 1;

alter table users drop column activated_at;

alter table users change column temp_activated_at activated_at datetime;

SELECT * FROM users;
```



```
SELECT
    *
FROM
    users;
```

user_id	company_id	language	state	created_at	activated_at
0	5737	english	active	2013-01-01 20:59:00	2013-01-01 21:01:00
3	2800	german	active	2013-01-01 18:40:00	2013-01-01 18:42:00
4	5110	indian	active	2013-01-01 14:37:00	2013-01-01 14:39:00
6	11699	english	active	2013-01-01 18:37:00	2013-01-01 18:38:00
7	4765	french	active	2013-01-01 16:19:00	2013-01-01 16:20:00
8	2698	french	active	2013-01-01 04:38:00	2013-01-01 04:40:00
11	3745	english	active	2013-01-01 08:07:00	2013-01-01 08:09:00
13	4025	english	active	2013-01-02 12:27:00	2013-01-02 12:29:00
15	4259	english	active	2013-01-02 15:39:00	2013-01-02 15:41:00
17	5025	japanese	active	2013-01-02 10:56:00	2013-01-02 10:57:00
19	326	english	active	2013-01-02 09:54:00	2013-01-02 09:55:00
20	7	italian	active	2013-01-02 09:41:00	2013-01-02 09:43:00
21	2606	english	active	2013-01-02 09:29:00	2013-01-02 09:30:00
22	545	german	active	2013-01-02 17:36:00	2013-01-02 17:38:00
27	6	japanese	active	2013-01-03 16:14:00	2013-01-03 16:15:00
30	4148	english	active	2013-01-03 08:28:00	2013-01-03 08:29:00
31	39	arabic	active	2013-01-03 15:45:00	2013-01-03 15:46:00
33	10768	english	active	2013-01-03 12:16:00	2013-01-03 12:18:00
35	1891	english	active	2013-01-03 16:06:00	2013-01-03 16:07:00
36	2	english	active	2013-01-03 11:51:00	2013-01-03 11:53:00
47	1	indian	active	2013-01-04 10:39:00	2013-01-04 10:41:00
49	8727	spanish	active	2013-01-05 14:33:00	2013-01-05 14:34:00
50	12526	spanish	active	2013-01-05 17:41:00	2013-01-05 17:43:00
52	6	english	active	2013-01-06 13:35:00	2013-01-06 13:37:00
54	2	english	active	2013-01-06 02:52:00	2013-01-06 02:54:00

Create table events :

```
CREATE TABLE events (
 user_id INT,
 occurred_at VARCHAR(100),
 event_type VARCHAR(50),
 event_name VARCHAR(100),
 location VARCHAR(50),
 device VARCHAR(50),
 user_type INT
);
```

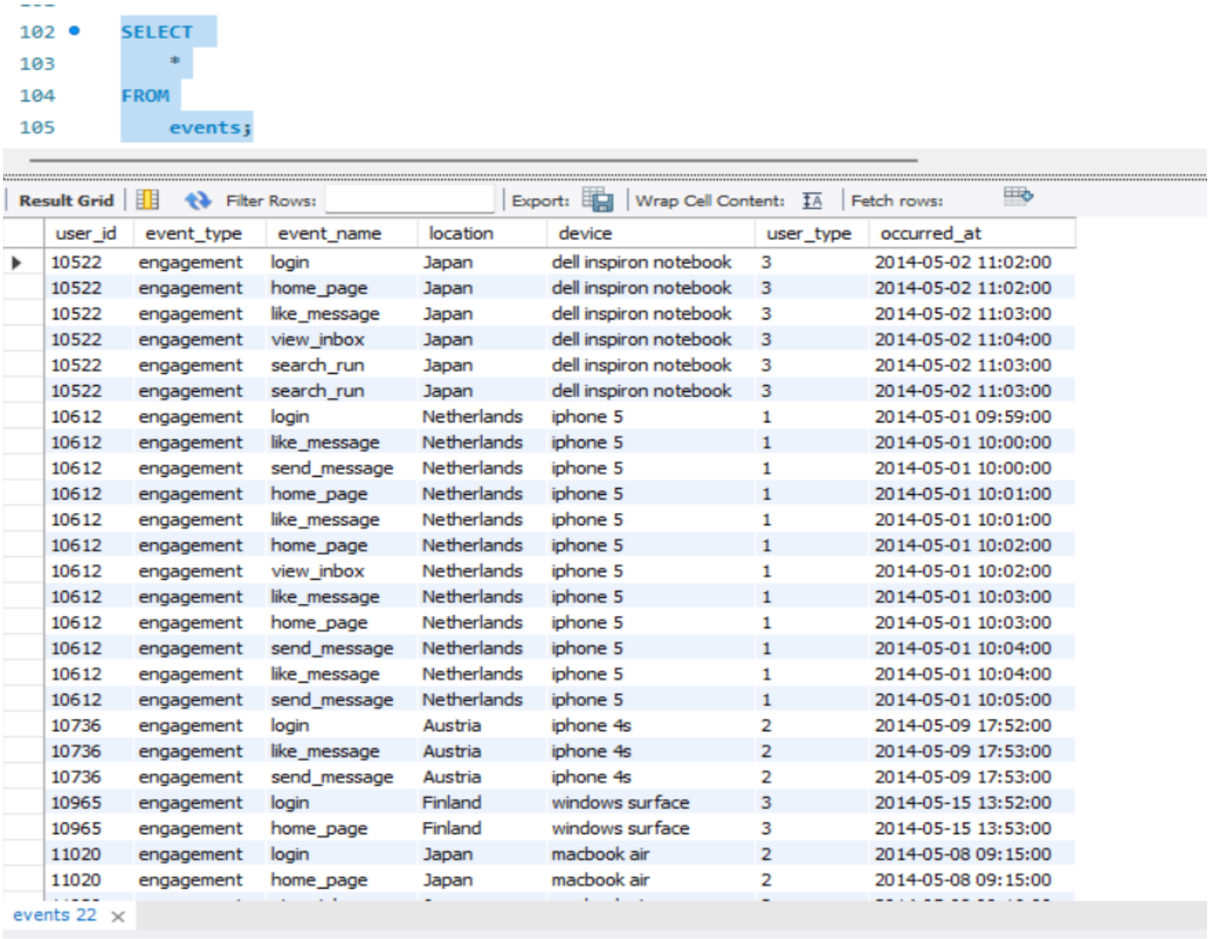
Upload events data file :

```
load data infile "C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/events.csv"
into table events
fields terminated by ','
```

```
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```

Update column occurred_at(Varchar) into occurred_at(datetime):

```
alter table events add column temp_created_at datetime;

set SQL_SAFE_UPDATES = 0;

UPDATE events
SET
 temp_created_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');

set SQL_SAFE_UPDATES = 1;

alter table events drop column occurred_at;

alter table events change column temp_created_at occurred_at datetime;

SELECT * FROM events;
```



```
SELECT
    *
FROM
    events;
```

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
10612	engagement	login	Netherlands	iphone 5	1	2014-05-01 09:59:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
10612	engagement	send_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:01:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:01:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:02:00
10612	engagement	view_inbox	Netherlands	iphone 5	1	2014-05-01 10:02:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:03:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:03:00
10612	engagement	send_message	Netherlands	iphone 5	1	2014-05-01 10:04:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:04:00
10612	engagement	send_message	Netherlands	iphone 5	1	2014-05-01 10:05:00
10736	engagement	login	Austria	iphone 4s	2	2014-05-09 17:52:00
10736	engagement	like_message	Austria	iphone 4s	2	2014-05-09 17:53:00
10736	engagement	send_message	Austria	iphone 4s	2	2014-05-09 17:53:00
10965	engagement	login	Finland	windows surface	3	2014-05-15 13:52:00
10965	engagement	home_page	Finland	windows surface	3	2014-05-15 13:53:00
11020	engagement	login	Japan	macbook air	2	2014-05-08 09:15:00
11020	engagement	home_page	Japan	macbook air	2	2014-05-08 09:15:00

Create table email_events

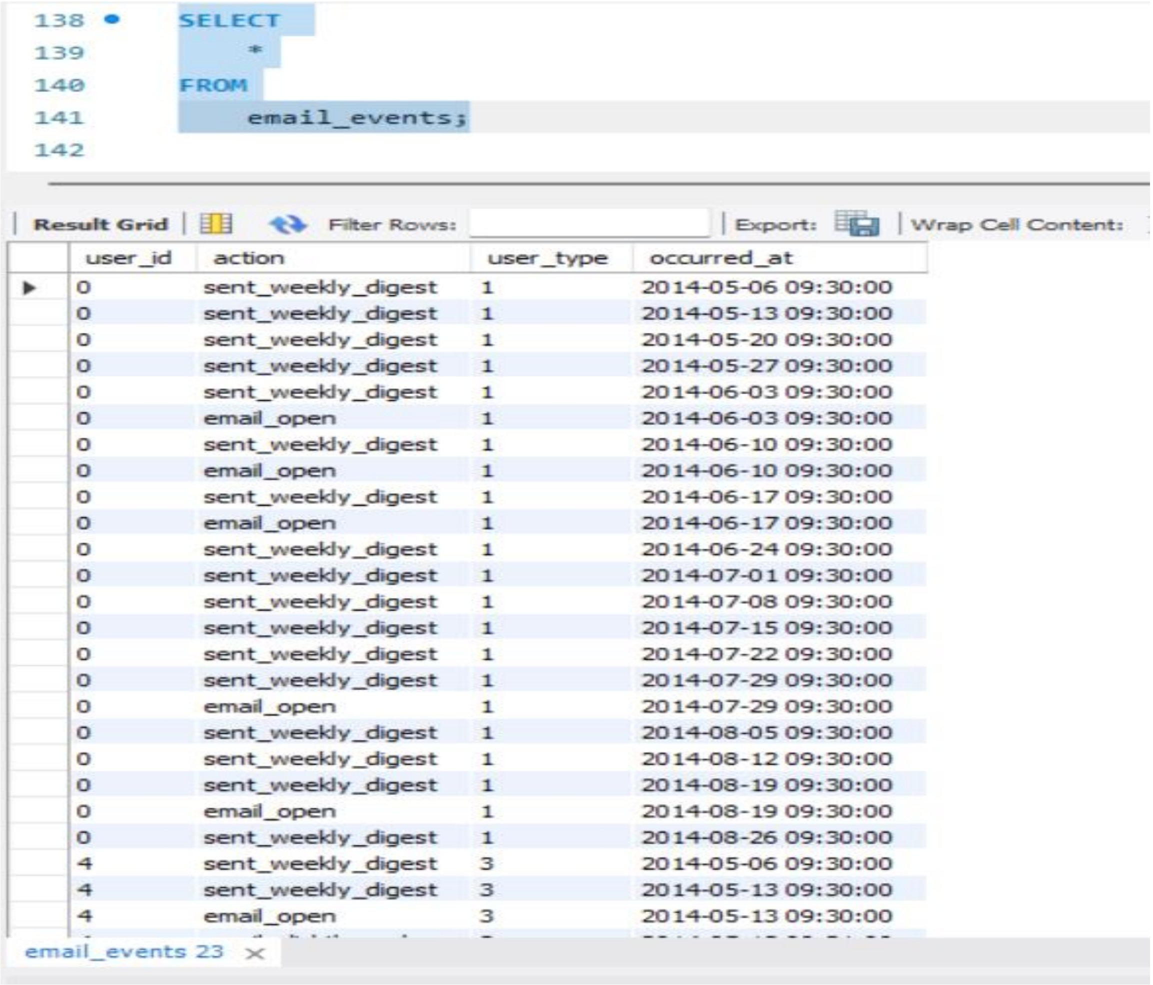
```
CREATE TABLE email_events (
 user_id INT,
 occurred_at VARCHAR(100),
 action VARCHAR(100),
 user_type INT
);
```

Upload events data file

```
load data infile "C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/email_events.csv"
into table email_events
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```

Update column occurred_at(Varchar) into occurred_at(datetime):

```
alter table email_events add column temp_created_at datetime;

set SQL_SAFE_UPDATES = 0;

UPDATE email_events
SET
 temp_created_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');

set SQL_SAFE_UPDATES = 1;

alter table email_events drop column occurred_at;

alter table email_events change column temp_created_at occurred_at
datetime;

SELECT * FROM email_events;
```

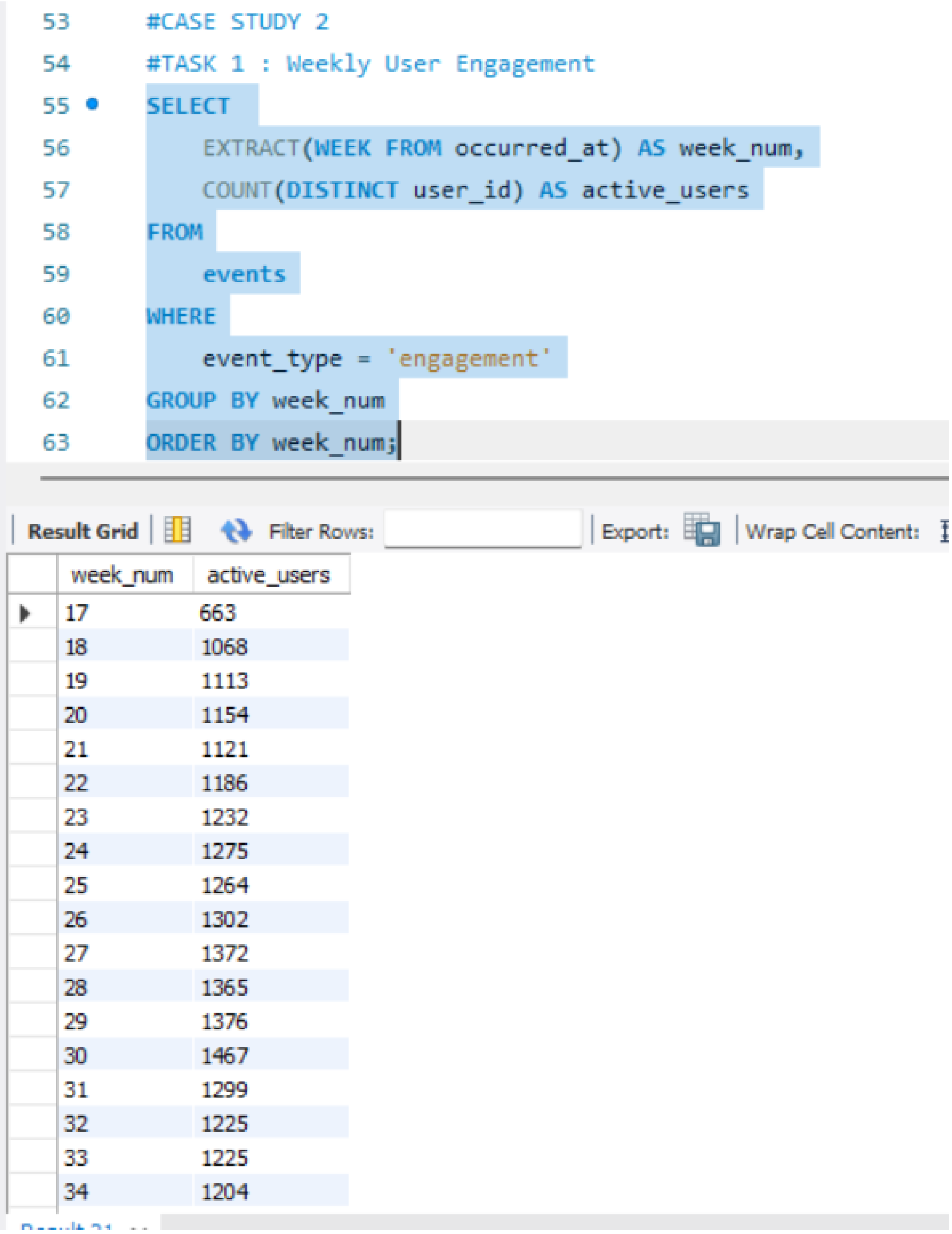


```
SELECT
    *
FROM
    email_events;
```

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
0	sent_weekly_digest	1	2014-06-10 09:30:00
0	email_open	1	2014-06-10 09:30:00
0	sent_weekly_digest	1	2014-06-17 09:30:00
0	email_open	1	2014-06-17 09:30:00
0	sent_weekly_digest	1	2014-06-24 09:30:00
0	sent_weekly_digest	1	2014-07-01 09:30:00
0	sent_weekly_digest	1	2014-07-08 09:30:00
0	sent_weekly_digest	1	2014-07-15 09:30:00
0	sent_weekly_digest	1	2014-07-22 09:30:00
0	sent_weekly_digest	1	2014-07-29 09:30:00
0	email_open	1	2014-07-29 09:30:00
0	sent_weekly_digest	1	2014-08-05 09:30:00
0	sent_weekly_digest	1	2014-08-12 09:30:00
0	sent_weekly_digest	1	2014-08-19 09:30:00
0	email_open	1	2014-08-19 09:30:00
0	sent_weekly_digest	1	2014-08-26 09:30:00
4	sent_weekly_digest	3	2014-05-06 09:30:00
4	sent_weekly_digest	3	2014-05-13 09:30:00
4	email_open	3	2014-05-13 09:30:00

TASK 1 : Weekly User Engagement (Write an SQL query to calculate the 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;
```



```
#CASE STUDY 2
#TASK 1 : 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;
```

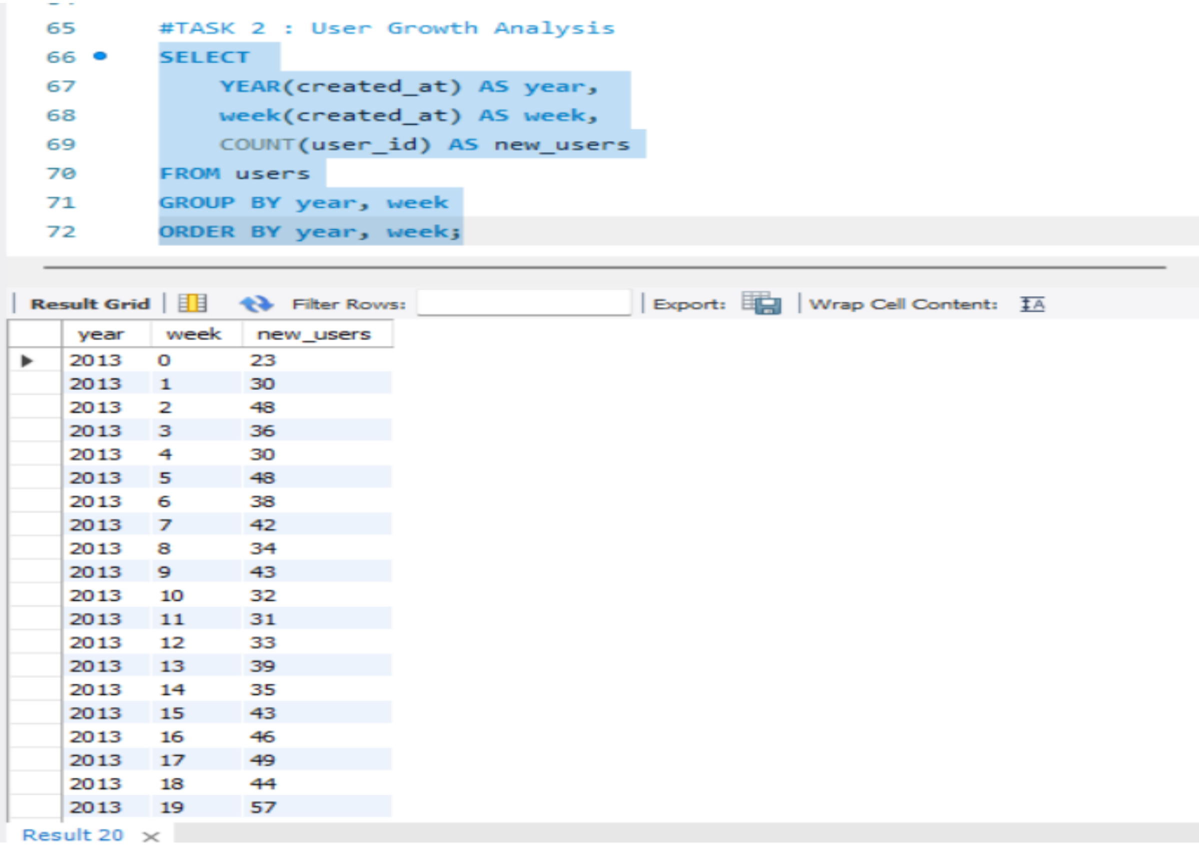
week_num	active_users
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204

Insights:

- Identifies the most engaged users and the weeks with high activity i.e. week = 30th, users = 1467.
- Helps in determining seasonality and behavioural patterns.
- A sharp decline in engagement might indicate an issue with the platform or user experience i.e. minimum engaged users = 104 in week 35th.

TASK 2 : User Growth Analysis (Write an SQL query to calculate the user growth for the product)

```
SELECT
 YEAR(created_at) AS year,
 week(created_at) AS week,
 COUNT(user_id) AS new_users
FROM users
GROUP BY year, week
ORDER BY year, week;
```



```
#TASK 2 : User Growth Analysis
SELECT
    YEAR(created_at) AS year,
    week(created_at) AS week,
    COUNT(user_id) AS new_users
FROM users
GROUP BY year, week
ORDER BY year, week;
```

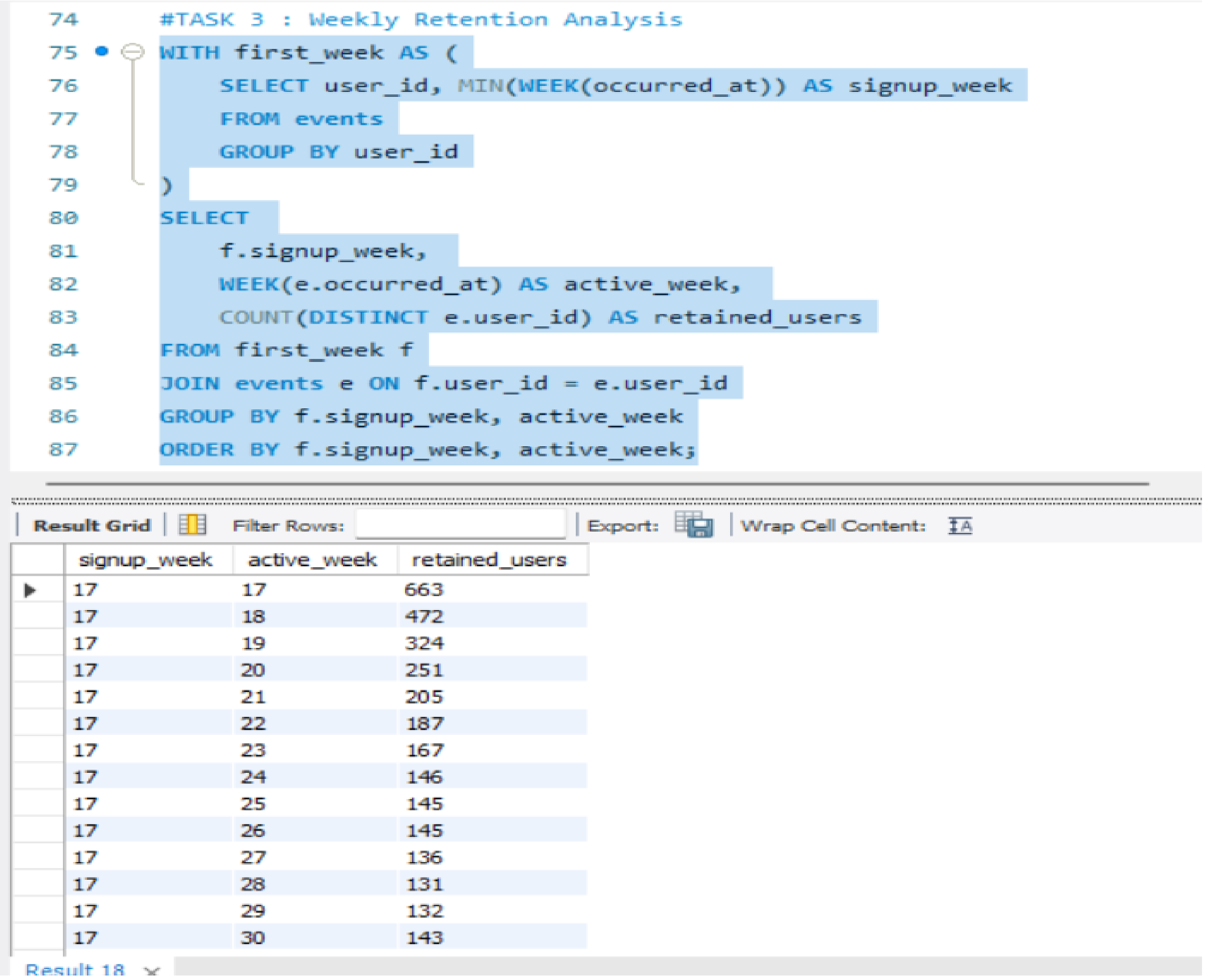
year	week	new_users
2013	0	23
2013	1	30
2013	2	48
2013	3	36
2013	4	30
2013	5	48
2013	6	38
2013	7	42
2013	8	34
2013	9	43
2013	10	32
2013	11	31
2013	12	33
2013	13	39
2013	14	35
2013	15	43
2013	16	46
2013	17	49
2013	18	44
2013	19	57

Insights:

- Helps track how fast the user base is expanding.
- Can correlate with marketing efforts or product updates to see what drives new user acquisition.
- A sudden drop in new users might indicate reduced marketing effectiveness.
- The 33rd week of 2014 shows the greatest number of users and the lowest was on 35th week of 2014.

TASK 3 : Weekly Retention Analysis (Write an SQL query to calculate the weekly retention of users based on their sign-up cohort)

```
WITH first_week AS (
 SELECT user_id, MIN(WEEK(occurred_at)) AS signup_week
 FROM events
 GROUP BY user_id
)
SELECT
 f.signup_week,
 WEEK(e.occurred_at) AS active_week,
 COUNT(DISTINCT e.user_id) AS retained_users
FROM first_week f
JOIN events e ON f.user_id = e.user_id
GROUP BY f.signup_week, active_week
ORDER BY f.signup_week, active_week;
```



```
#TASK 3 : Weekly Retention Analysis
WITH first_week AS (
    SELECT user_id, MIN(WEEK(occurred_at)) AS signup_week
    FROM events
    GROUP BY user_id
)
SELECT
    f.signup_week,
    WEEK(e.occurred_at) AS active_week,
    COUNT(DISTINCT e.user_id) AS retained_users
FROM first_week f
JOIN events e ON f.user_id = e.user_id
GROUP BY f.signup_week, active_week
ORDER BY f.signup_week, active_week;
```

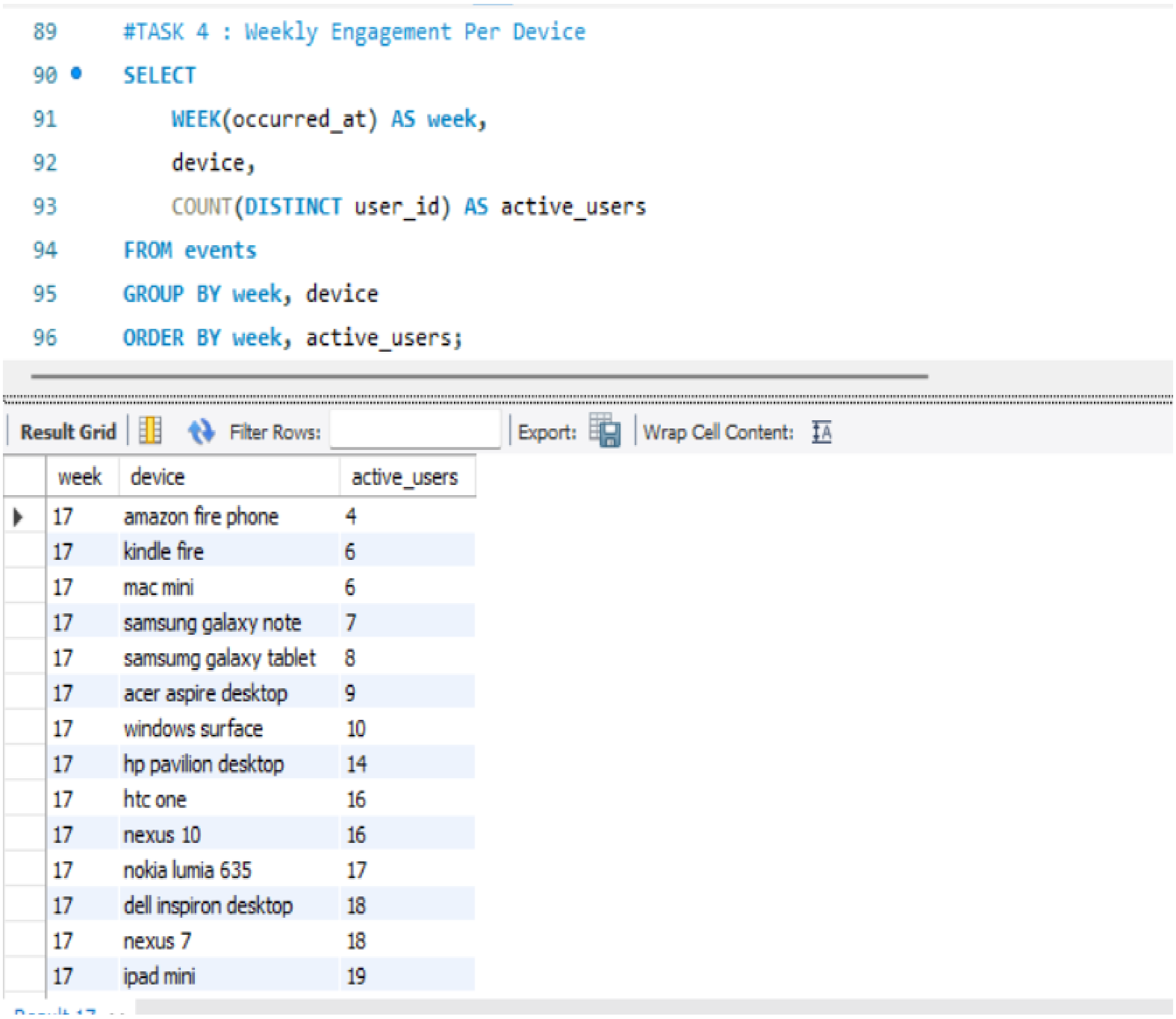
signup_week	active_week	retained_users
17	17	663
17	18	472
17	19	324
17	20	251
17	21	205
17	22	187
17	23	167
17	24	146
17	25	145
17	26	145
17	27	136
17	28	131
17	29	132
17	30	143

Insights:

- Measures how many users stay engaged after signing up.
- Helps in improving onboarding strategies to reduce drop-offs.
- If retention is low, adjustments like push notifications or personalized emails might be needed.

TASK 4 : Weekly Engagement Per Device(Write an SQL query to calculate the weekly engagement per device)

```
SELECT
 WEEK(occurred_at) AS week,
 device,
 COUNT(DISTINCT user_id) AS active_users
FROM events
GROUP BY week, device
ORDER BY week, active_users;
```



```
#TASK 4 : Weekly Engagement Per Device
SELECT
    WEEK(occurred_at) AS week,
    device,
    COUNT(DISTINCT user_id) AS active_users
FROM events
GROUP BY week, device
ORDER BY week, active_users;
```

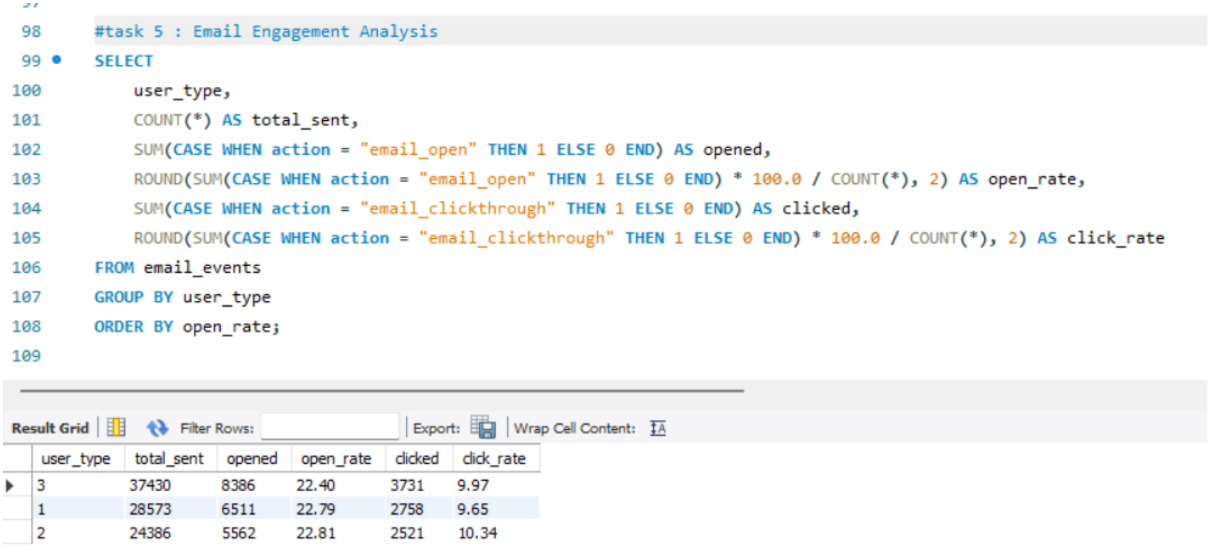
week	device	active_users
17	amazon fire phone	4
17	kindle fire	6
17	mac mini	6
17	samsung galaxy note	7
17	samsumg galaxy tablet	8
17	acer aspire desktop	9
17	windows surface	10
17	hp pavilion desktop	14
17	htc one	16
17	nexus 10	16
17	nokia lumia 635	17
17	dell inspiron desktop	18
17	nexus 7	18
17	ipad mini	19

Insights:

- Helps understand which device types contribute most to engagement.
- A sudden decline in a specific device type might indicate compatibility issues.
- Can inform optimization efforts for mobile or desktop platforms.

TASK 5 : Email Engagement Analysis (Write an SQL query to calculate the email engagement metrics)

```
SELECT
 user_type,
 COUNT(*) AS total_sent,
 SUM(CASE WHEN action = "email_open" THEN 1 ELSE 0 END) AS opened,
 ROUND(SUM(CASE WHEN action = "email_open" THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) AS open_rate,
 SUM(CASE WHEN action = "email_clickthrough" THEN 1 ELSE 0 END) AS clicked,
 ROUND(SUM(CASE WHEN action = "email_clickthrough" THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) AS click_rate
FROM email_events
GROUP BY user_type
ORDER BY open_rate;
```



```
#task 5 : Email Engagement Analysis
SELECT
    user_type,
    COUNT(*) AS total_sent,
    SUM(CASE WHEN action = "email_open" THEN 1 ELSE 0 END) AS opened,
    ROUND(SUM(CASE WHEN action = "email_open" THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) AS open_rate,
    SUM(CASE WHEN action = "email_clickthrough" THEN 1 ELSE 0 END) AS clicked,
    ROUND(SUM(CASE WHEN action = "email_clickthrough" THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) AS click_rate
FROM email_events
GROUP BY user_type
ORDER BY open_rate;
```

user_type	total_sent	opened	open_rate	clicked	click_rate
3	37430	8386	22.40	3731	9.97
1	28573	6511	22.79	2758	9.65
2	24386	5562	22.81	2521	10.34

Insights:

- High open rates indicate effective subject lines and sender reputation.
- High click rates suggest compelling email content.
- A drop in engagement might require email optimization or segmentation strategies.

Insights & Recommendations :

Case Study 1: Job Data Analysis

- **Jobs Reviewed Over Time:**
  - The number of jobs reviewed per hour fluctuates based on working hours and organizational workload.
  - Peak hours show higher activity, likely due to batch processing or scheduled reviews.
  - A sudden dip in review counts could indicate system downtime, reduced workforce availability, or lower job submissions.

- **Throughput Analysis (7-Day Rolling Average)**
  - The rolling average smooths out fluctuations and provides a more stable measure of performance.
  - It helps identify long-term trends instead of daily variations, making it easier to spot consistent growth or decline.
  - A sudden drop in throughput might indicate system inefficiencies, workforce reductions, or delays in job assignments.
- **Language Share Analysis**
  - The majority of jobs are reviewed in certain dominant languages (e.g., English, Spanish), while some languages have minimal representation.
  - This insight can guide the company in diversifying language support or allocating resources based on demand.
  - A shift in language trends over time may indicate emerging markets or changing user preferences.
- **Duplicate Rows Detection**
  - Identifying duplicates ensures data integrity and prevents inaccurate reporting.
  - Duplicate job entries could result from system errors or multiple actors reviewing the same job.
  - Cleaning up duplicates improves efficiency and ensures accurate analytics for decision-making.

Case Study 2: Investigating Metric Spikes

- **Weekly User Engagement**
  - User engagement varies weekly, with higher engagement on weekdays and possible dips on weekends.

- If engagement suddenly drops, it may indicate product issues, poor user experience, or external factors like holidays.
- A steady increase suggests user adoption and sustained interest in the platform.

User Growth Analysis

- Tracking new user sign-ups helps understand the effectiveness of marketing efforts.
- A decline in new users may indicate market saturation, competition, or ineffective promotions.
- Identifying the most successful acquisition channels can optimize marketing spend.

Weekly Retention Analysis

- Retention rates highlight whether users continue using the product after sign-up.
- A low retention rate suggests a poor onboarding experience, lack of engagement, or missing product features.
- Strategies such as personalized content, push notifications, or discounts can improve retention.

Weekly Engagement Per Device

- Mobile and desktop users may exhibit different engagement patterns.
- If mobile engagement is significantly lower, it could point to a poor mobile experience or app-related issues.
- Optimizing the platform for the most commonly used devices can enhance user satisfaction.

Email Engagement Analysis

- Email open and click rates indicate how well users respond to communication.
- Low open rates suggest poor subject lines or irrelevant content, while low click-through rates may indicate ineffective calls-to-action.
- A/B testing different email strategies can improve engagement and conversion rates.

Conclusion :

- The project demonstrated the power of SQL-based analytics in identifying trends, investigating metric spikes, and optimizing business decisions.

- By analysing job review data, we uncovered patterns in review frequency, language distribution, and duplicate records that could help streamline operations.
- The user engagement analysis highlighted key areas for improvement in retention, device experience, and email strategies.
- Using insights from data, companies can improve operational efficiency, enhance user engagement.