OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

AGENDA

Create database → Perform Queries → Tech-Stack Used → Insights → Conclusion

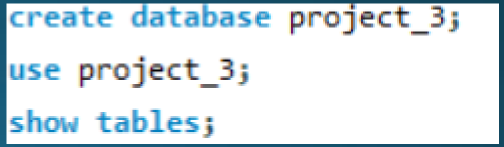
CREATING DATABASE

- We can create the database by using the following command.
- **SYNTAX : CREATE DATABASE DATABASE_NAME;**

- In order to use that particular database we need to use the command
- **SYNTAX : USE DATABASE_NAME;**

- In order to import the tables
- **Schemas – database_name – table – Table data import wizard – file path – next**.

- In order to see databases we have to use the command
- **SYNTAX : SHOW DATABASES;**



```
create database project_3;
use project_3;
show tables;
```

APPROACH

- There are 2 case studies for the given project.
- In case study 1 there is one table that is job_data.
- In case study 2 there are 3 tables they are events, users,email_events.
- These files are provided in csv files we have import them properly.
- Next check the data format given are imported properly .

PERFORM ANALYSIS

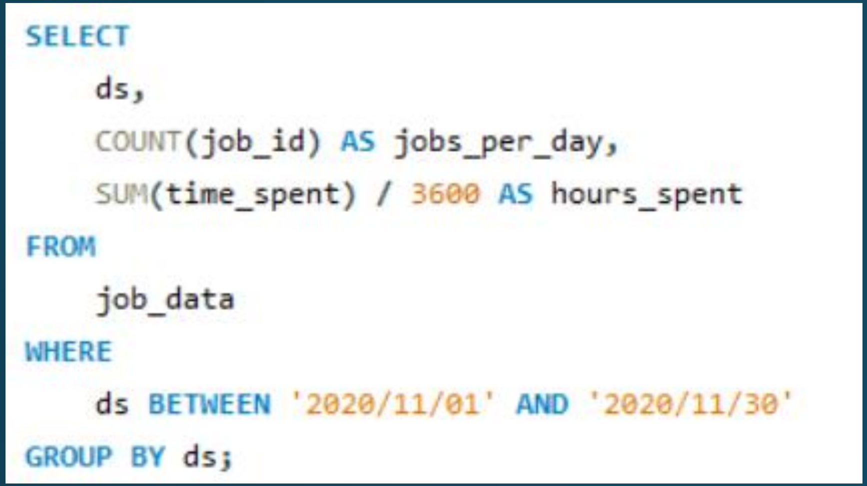
Case Study 1: Job Data Analysis

QUERY 1 : To write an SQL query to calculate the no. of jobs reviewed per hour for each day in November 2020.

PROJECT DESCRIPTION : JOBS REVIEWD OVER TIME

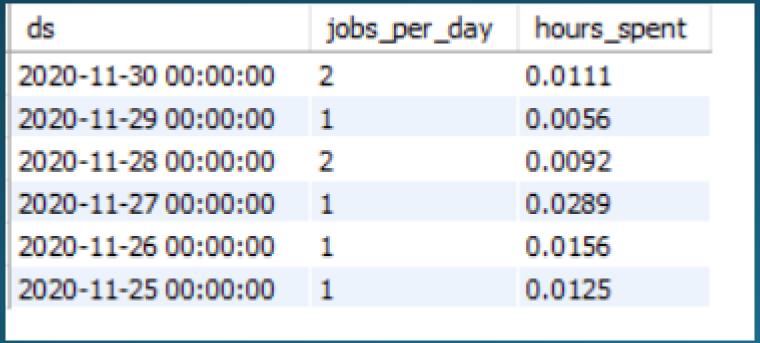
To calculate the number of jobs reviewed per hour for each day in Nov 2020.

•**QUERY 1:**



```
SELECT
    ds,
    COUNT(job_id) AS jobs_per_day,
    SUM(time_spent) / 3600 AS hours_spent
FROM
    job_data
WHERE
    ds BETWEEN '2020/11/01' AND '2020/11/30'
GROUP BY ds;
```

RESULT :



ds	jobs_per_day	hours_spent
2020-11-30 00:00:00	2	0.0111
2020-11-29 00:00:00	1	0.0056
2020-11-28 00:00:00	2	0.0092
2020-11-27 00:00:00	1	0.0289
2020-11-26 00:00:00	1	0.0156
2020-11-25 00:00:00	1	0.0125

PERFORM ANALYSIS

Case Study 1: Job Data Analysis

QUERY 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.

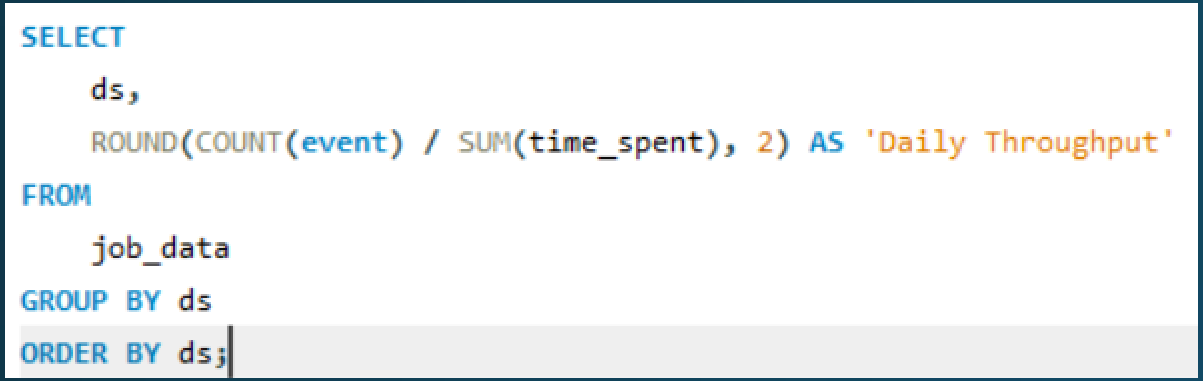
PROJECT DESCRIPTION : THROUGHPUT ANALYSIS

To Calculate the 7-day rolling average of throughput (no of events per second)

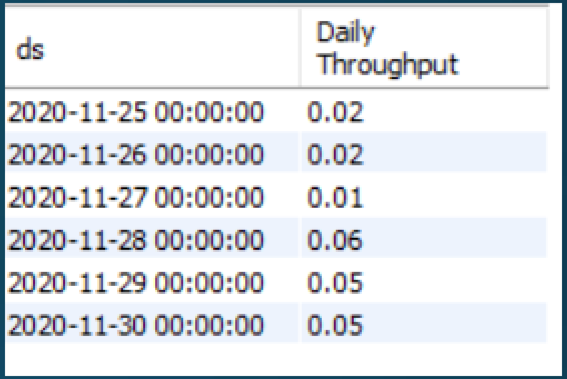
Prefer week throughput than daily throughput because daily throughput can fluctuate regularly.

•QUERY 2:

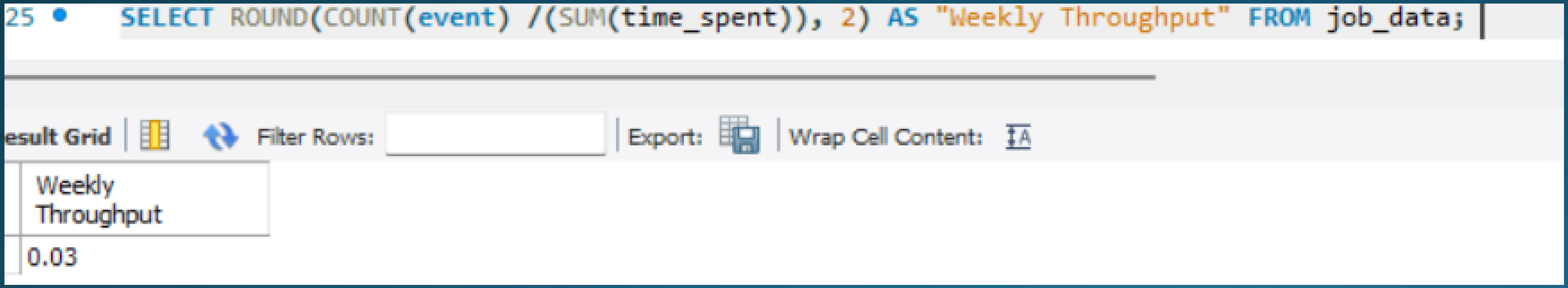
RESULT :



```sql
SELECT
    ds,
    ROUND(COUNT(event) / SUM(time_spent), 2) AS 'Daily Throughput'
FROM
    job_data
GROUP BY ds
ORDER BY ds;
```



ds	Daily Throughput
2020-11-25 00:00:00	0.02
2020-11-26 00:00:00	0.02
2020-11-27 00:00:00	0.01
2020-11-28 00:00:00	0.06
2020-11-29 00:00:00	0.05
2020-11-30 00:00:00	0.05



```sql
SELECT ROUND(COUNT(event) /(SUM(time_spent)), 2) AS "Weekly Throughput" FROM job_data;
```

Weekly Throughput
0.03

PERFORM ANALYSIS

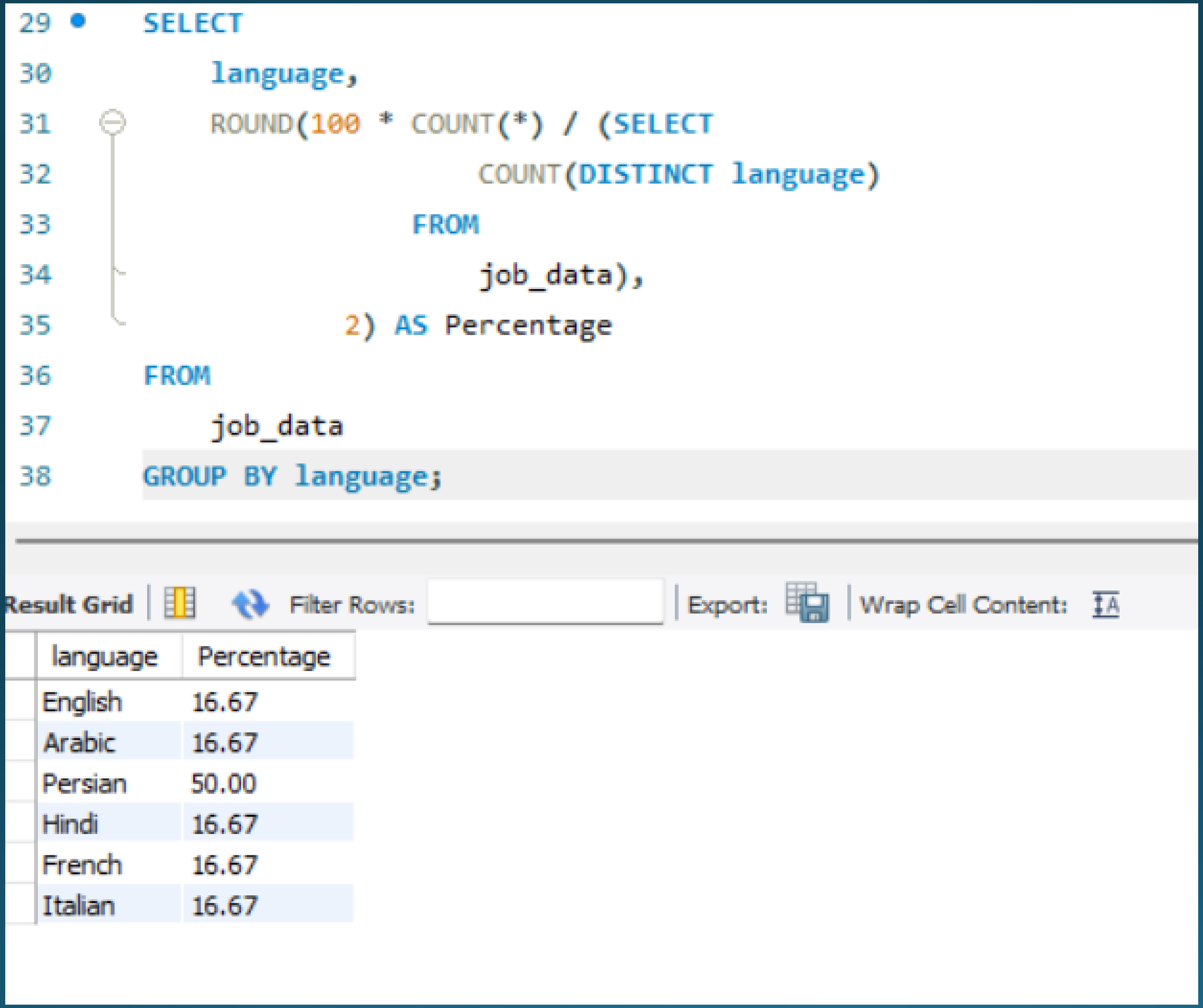
Case Study 1: Job Data Analysis

QUERY 3 : To write an SQL query to calculate the percentage share of each language over the last 30 days.

PROJECT DESCRIPTION : LANGUAGE SHARE ANALYSIS

To Calculate the percentage share of each language in the last 30 days

•QUERY 3:



```
SELECT
    language,
    ROUND(100 * COUNT(*) / (SELECT
                COUNT(DISTINCT language)
            FROM
                job_data),
          2) AS Percentage
FROM
    job_data
GROUP BY language;
```

RESULT :

language	Percentage
English	16.67
Arabic	16.67
Persian	50.00
Hindi	16.67
French	16.67
Italian	16.67

PERFORM ANALYSIS

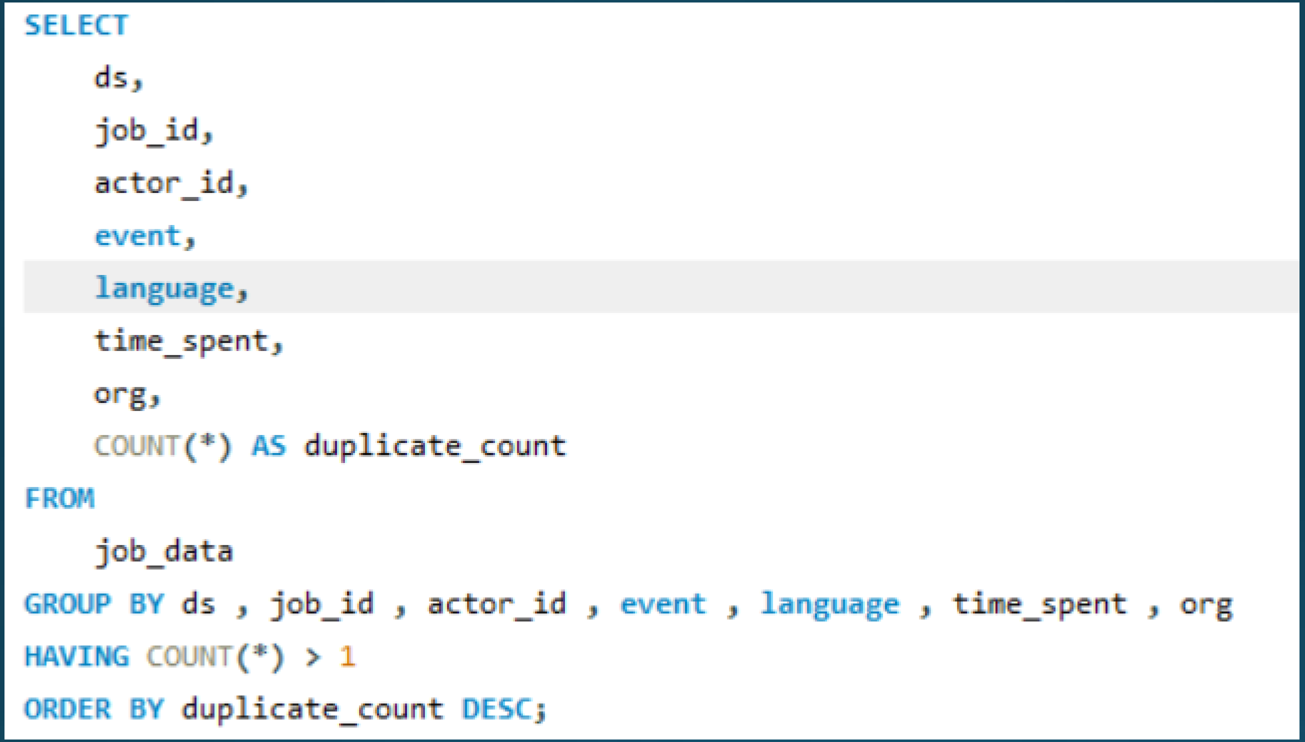
Case Study 1: Job Data Analysis

QUERY 4 : To write an SQL query to display duplicate rows from the job_data table.

PROJECT DESCRIPTION : DUPLICATE ROWS DETECTION

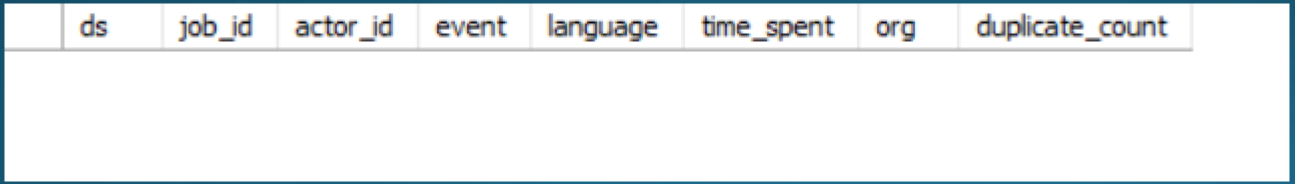
To identify the duplicate rows in the given data.

•**QUERY 4:**



```
SELECT
    ds,
    job_id,
    actor_id,
    event,
    language,
    time_spent,
    org,
    COUNT(*) AS duplicate_count
FROM
    job_data
GROUP BY ds , job_id , actor_id , event , language , time_spent , org
HAVING COUNT(*) > 1
ORDER BY duplicate_count DESC;
```

RESULT :



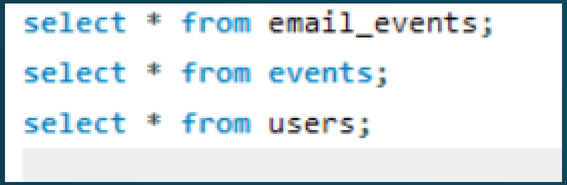
ds	job_id	actor_id	event	language	time_spent	org	duplicate_count

There is no duplicate data in the given table

PERFORM ANALYSIS

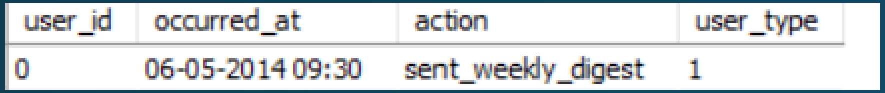
Case Study 2: Investigating Metric Spike

There are 3 tables



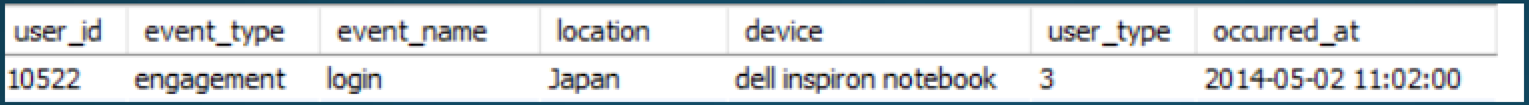
```
select * from email_events;
select * from events;
select * from users;
```

email_events



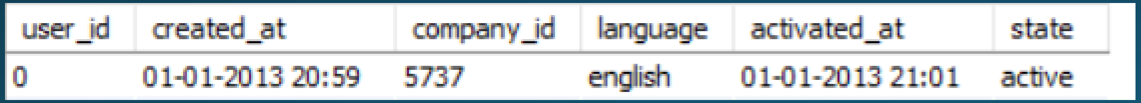
user_id	occurred_at	action	user_type
0	06-05-2014 09:30	sent_weekly_digest	1

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

users



user_id	created_at	company_id	language	activated_at	state
0	01-01-2013 20:59	5737	english	01-01-2013 21:01	active

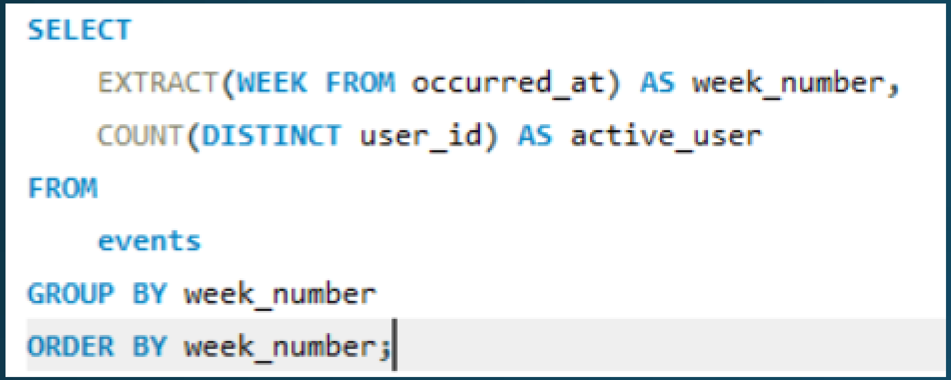
QUERY 1 : To write an SQL query to calculate the weekly user engagement.

PROJECT DESCRIPTION :WEEKLY USER ENGAGEMENT

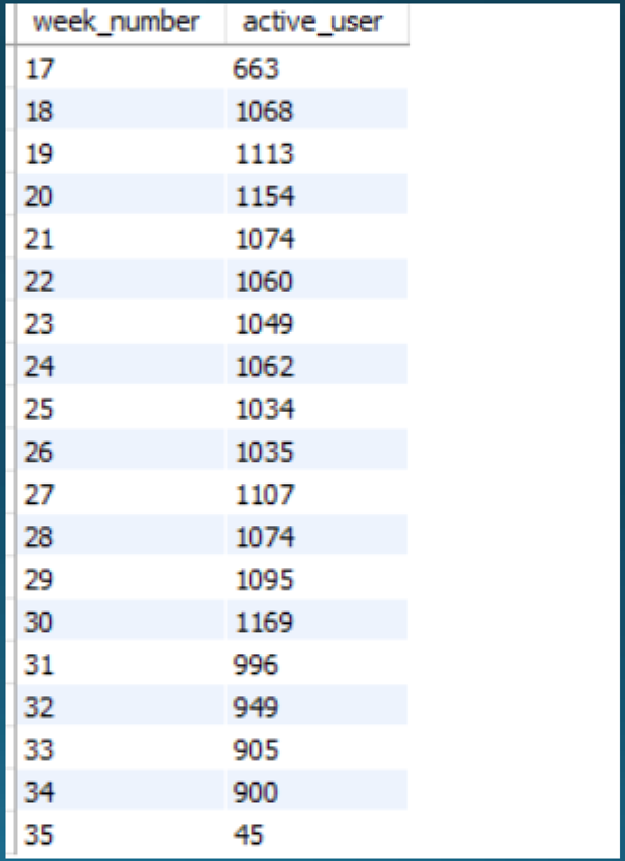
To measure the activeness of users on a weekly basis.

•QUERY 1:

RESULT :



```
SELECT
    EXTRACT(WEEK FROM occurred_at) AS week_number,
    COUNT(DISTINCT user_id) AS active_user
FROM
    events
GROUP BY week_number
ORDER BY week_number;
```



week_number	active_user
17	663
18	1068
19	1113
20	1154
21	1074
22	1060
23	1049
24	1062
25	1034
26	1035
27	1107
28	1074
29	1095
30	1169
31	996
32	949
33	905
34	900
35	45

PERFORM ANALYSIS

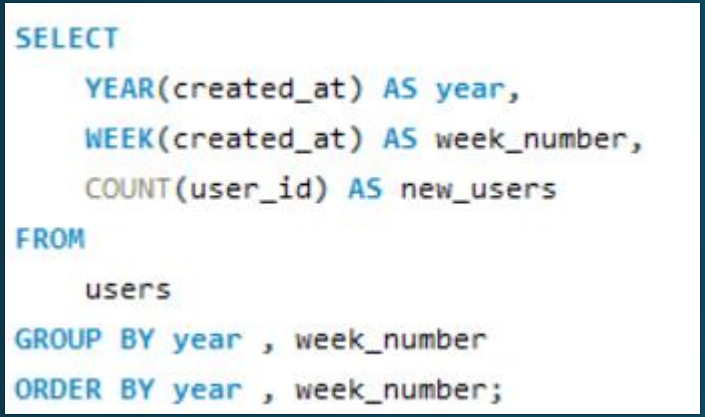
Case Study 2: Investigating Metric Spike

QUERY 2 : To write an SQL query to calculate the user growth for the product.

PROJECT DESCRIPTION :USER GROWTH ANALYSIS

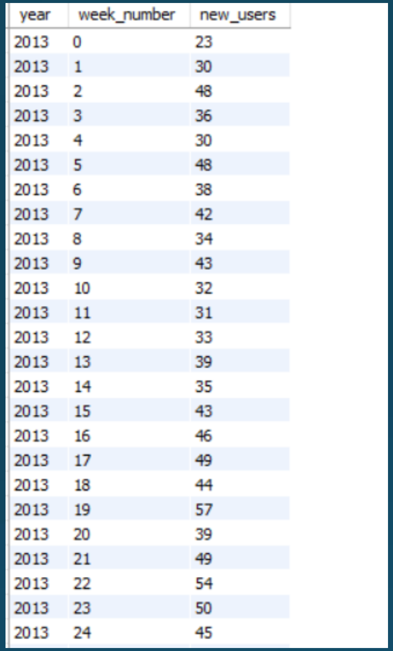
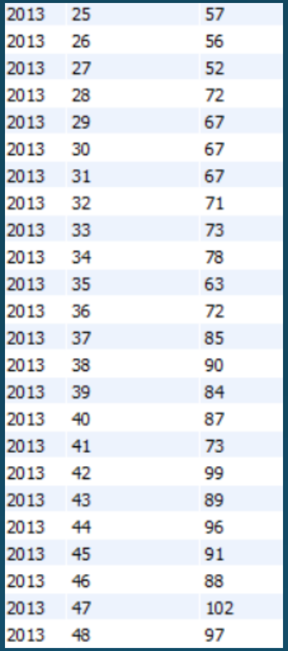
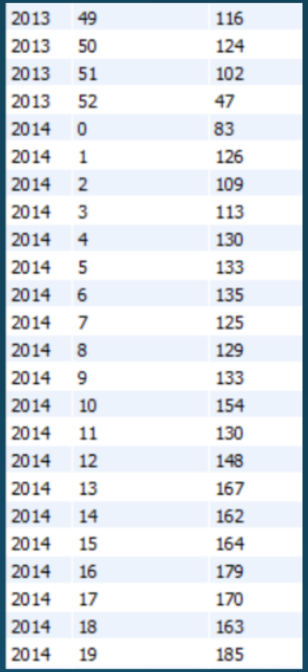
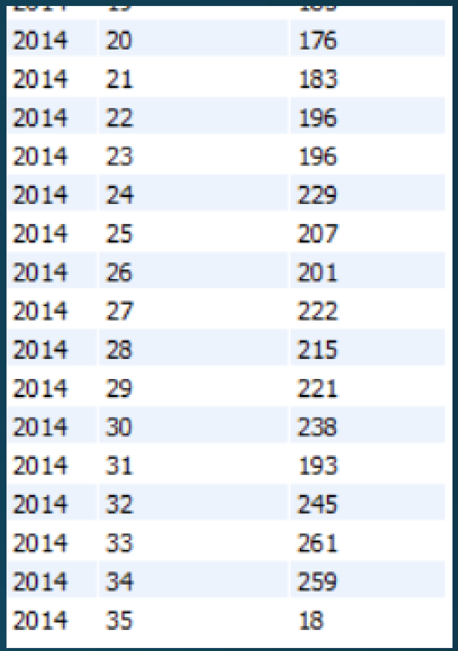
To analyze the growth of users over time for a product.

•QUERY 2:



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

RESULT :

year	week_number	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
2013	20	39
2013	21	49
2013	22	54
2013	23	50
2013	24	45
2013	25	57
2013	26	56
2013	27	52
2013	28	72
2013	29	67
2013	30	67
2013	31	67
2013	32	71
2013	33	73
2013	34	78
2013	35	63
2013	36	72
2013	37	85
2013	38	90
2013	39	84
2013	40	87
2013	41	73
2013	42	99
2013	43	89
2013	44	96
2013	45	91
2013	46	88
2013	47	102
2013	48	97
2013	49	116
2013	50	124
2013	51	102
2013	52	47
2014	0	83
2014	1	126
2014	2	109
2014	3	113
2014	4	130
2014	5	133
2014	6	135
2014	7	125
2014	8	129
2014	9	133
2014	10	154
2014	11	130
2014	12	148
2014	13	167
2014	14	162
2014	15	164
2014	16	179
2014	17	170
2014	18	163
2014	19	185
2014	20	176
2014	21	183
2014	22	196
2014	23	196
2014	24	229
2014	25	207
2014	26	201
2014	27	222
2014	28	215
2014	29	221
2014	30	238
2014	31	193
2014	32	245
2014	33	261
2014	34	259
2014	35	18

PERFORM ANALYSIS

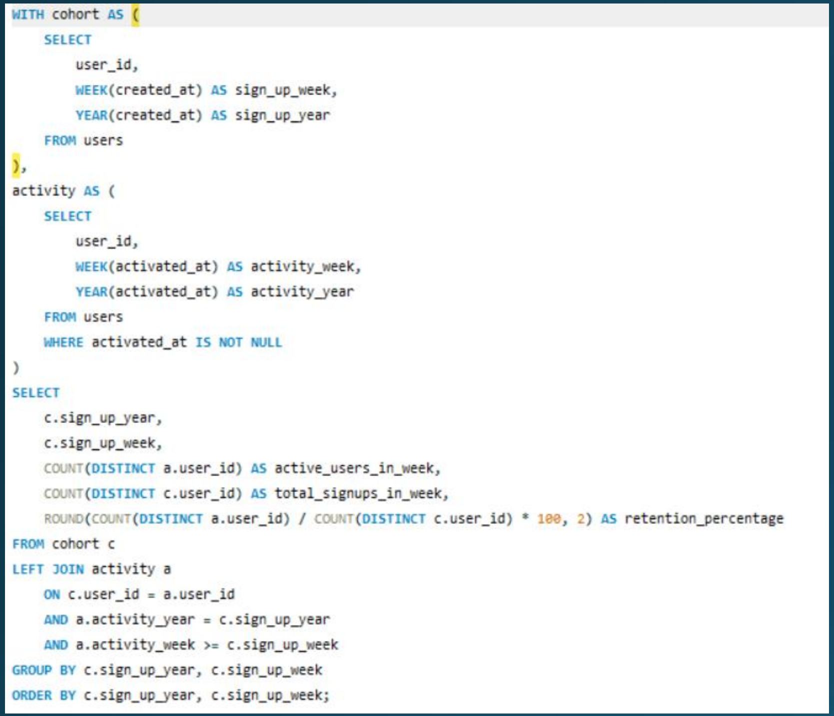
Case Study 2: Investigating Metric Spike

QUERY 3 : To write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

PROJECT DESCRIPTION :WEEKLY RETENTION ANALYSIS

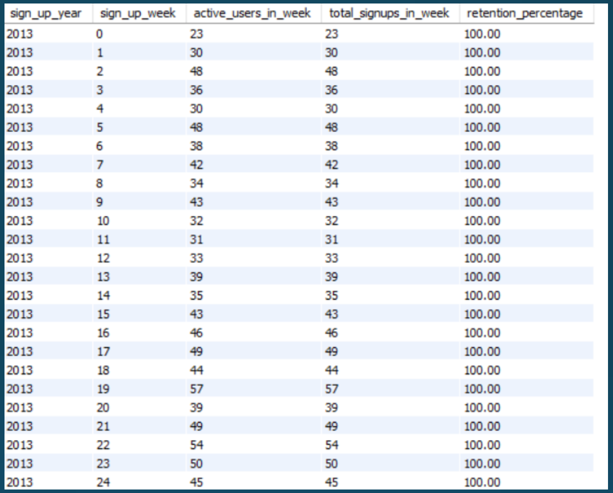
To analyze the retention of users on a weekly basis after signing up for a product.

•QUERY 3:

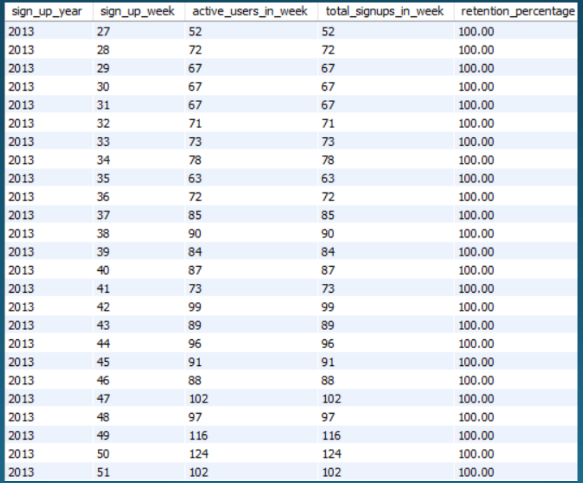


```sql
WITH cohort AS (
    SELECT
        user_id,
        WEEK(created_at) AS sign_up_week,
        YEAR(created_at) AS sign_up_year
    FROM users
),
activity AS (
    SELECT
        user_id,
        WEEK(activated_at) AS activity_week,
        YEAR(activated_at) AS activity_year
    FROM users
    WHERE activated_at IS NOT NULL
)
SELECT
    c.sign_up_year,
    c.sign_up_week,
    COUNT(DISTINCT a.user_id) AS active_users_in_week,
    COUNT(DISTINCT c.user_id) AS total_signups_in_week,
    ROUND(COUNT(DISTINCT a.user_id) / COUNT(DISTINCT c.user_id) * 100, 2) AS retention_percentage
FROM cohort c
LEFT JOIN activity a
    ON c.user_id = a.user_id
    AND a.activity_year = c.sign_up_year
    AND a.activity_week >= c.sign_up_week
GROUP BY c.sign_up_year, c.sign_up_week
ORDER BY c.sign_up_year, c.sign_up_week;
```

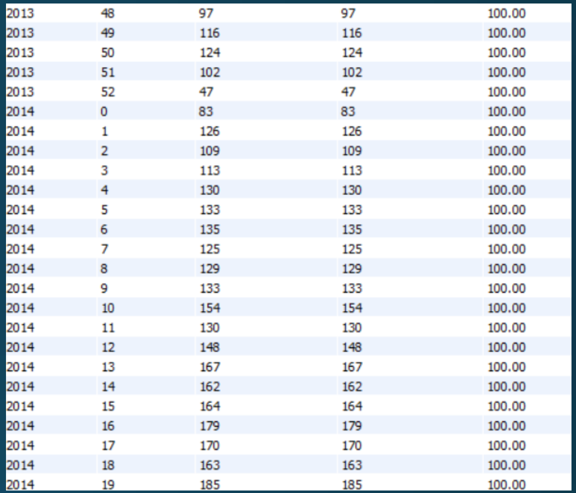
RESULT :



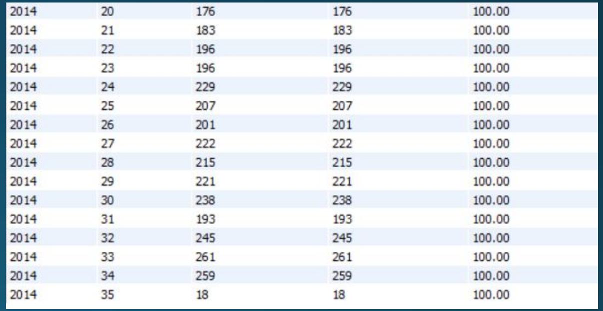
sign_up_year	sign_up_week	active_users_in_week	total_signups_in_week	retention_percentage
2013	0	23	23	100.00
2013	1	30	30	100.00
2013	2	48	48	100.00
2013	3	36	36	100.00
2013	4	30	30	100.00
2013	5	48	48	100.00
2013	6	38	38	100.00
2013	7	42	42	100.00
2013	8	34	34	100.00
2013	9	43	43	100.00
2013	10	32	32	100.00
2013	11	31	31	100.00
2013	12	33	33	100.00
2013	13	39	39	100.00
2013	14	35	35	100.00
2013	15	43	43	100.00
2013	16	46	46	100.00
2013	17	49	49	100.00
2013	18	44	44	100.00
2013	19	57	57	100.00
2013	20	39	39	100.00
2013	21	49	49	100.00
2013	22	54	54	100.00
2013	23	50	50	100.00
2013	24	45	45	100.00



sign_up_year	sign_up_week	active_users_in_week	total_signups_in_week	retention_percentage
2013	27	52	52	100.00
2013	28	72	72	100.00
2013	29	67	67	100.00
2013	30	67	67	100.00
2013	31	67	67	100.00
2013	32	71	71	100.00
2013	33	73	73	100.00
2013	34	78	78	100.00
2013	35	63	63	100.00
2013	36	72	72	100.00
2013	37	85	85	100.00
2013	38	90	90	100.00
2013	39	84	84	100.00
2013	40	87	87	100.00
2013	41	73	73	100.00
2013	42	99	99	100.00
2013	43	89	89	100.00
2013	44	96	96	100.00
2013	45	91	91	100.00
2013	46	88	88	100.00
2013	47	102	102	100.00
2013	48	97	97	100.00
2013	49	116	116	100.00
2013	50	124	124	100.00
2013	51	102	102	100.00



2013	48	97	97	100.00
2013	49	116	116	100.00
2013	50	124	124	100.00
2013	51	102	102	100.00
2013	52	47	47	100.00
2014	0	83	83	100.00
2014	1	126	126	100.00
2014	2	109	109	100.00
2014	3	113	113	100.00
2014	4	130	130	100.00
2014	5	133	133	100.00
2014	6	135	135	100.00
2014	7	125	125	100.00
2014	8	129	129	100.00
2014	9	133	133	100.00
2014	10	154	154	100.00
2014	11	130	130	100.00
2014	12	148	148	100.00
2014	13	167	167	100.00
2014	14	162	162	100.00
2014	15	164	164	100.00
2014	16	179	179	100.00
2014	17	170	170	100.00
2014	18	163	163	100.00
2014	19	185	185	100.00



2014	20	176	176	100.00
2014	21	183	183	100.00
2014	22	196	196	100.00
2014	23	196	196	100.00
2014	24	229	229	100.00
2014	25	207	207	100.00
2014	26	201	201	100.00
2014	27	222	222	100.00
2014	28	215	215	100.00
2014	29	221	221	100.00
2014	30	238	238	100.00
2014	31	193	193	100.00
2014	32	245	245	100.00
2014	33	261	261	100.00
2014	34	259	259	100.00
2014	35	18	18	100.00

PERFORM ANALYSIS

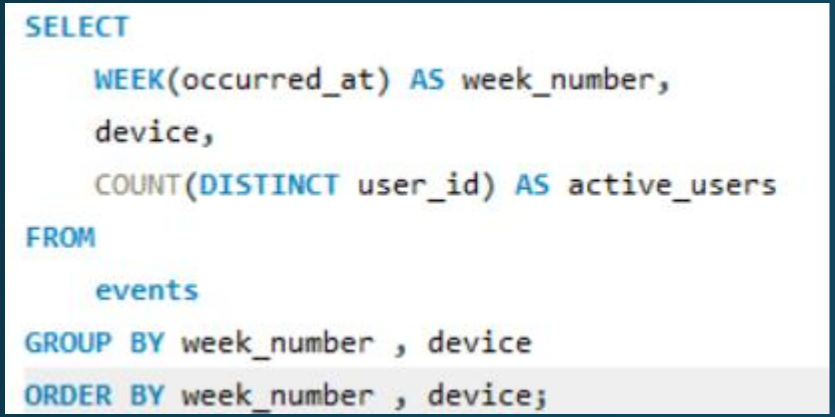
Case Study 2: Investigating Metric Spike

QUERY 4 : To write an SQL query to calculate the weekly engagement per device.

PROJECT DESCRIPTION :WEEKLY ENGAGEMENT PER DEVICE

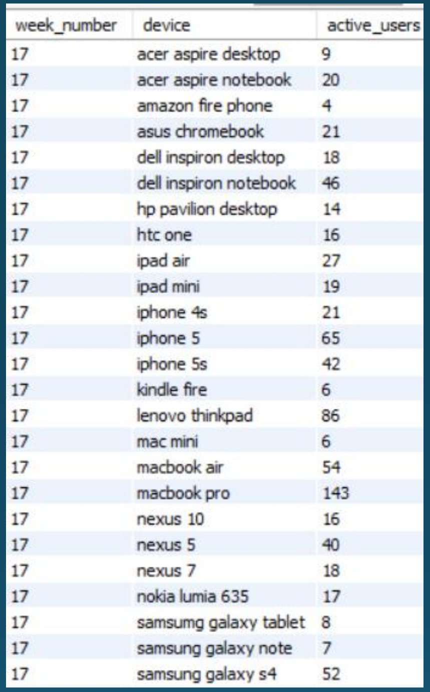
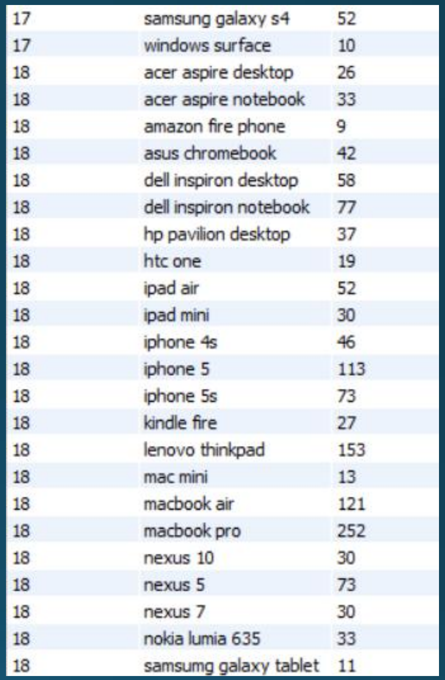
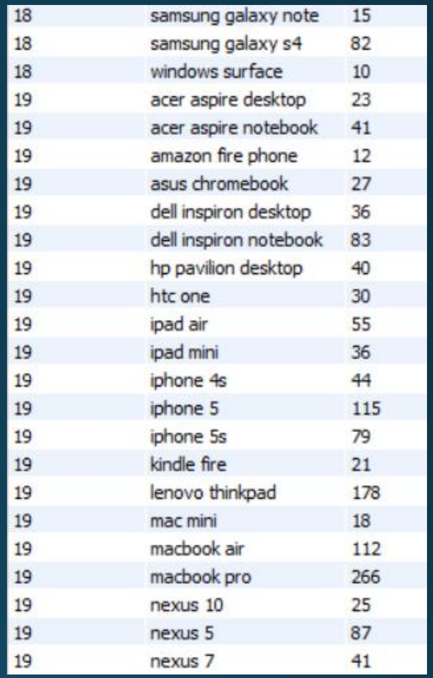
To measure the activeness of users on a weekly basis per device.

•QUERY 4:



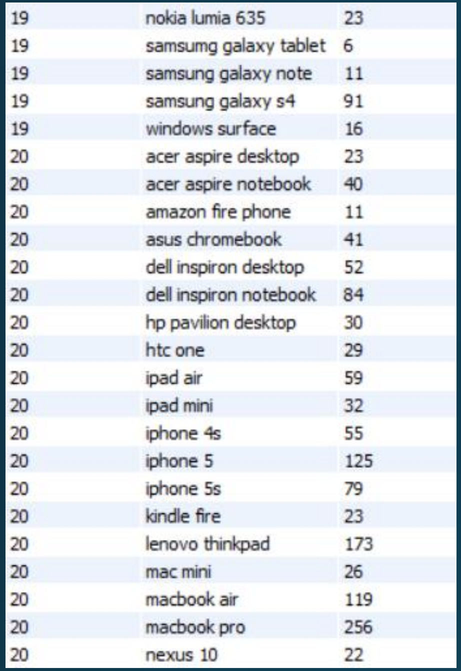
```
SELECT
    WEEK(occurred_at) AS week_number,
    device,
    COUNT(DISTINCT user_id) AS active_users
FROM
    events
GROUP BY week_number , device
ORDER BY week_number , device;
```

RESULT :

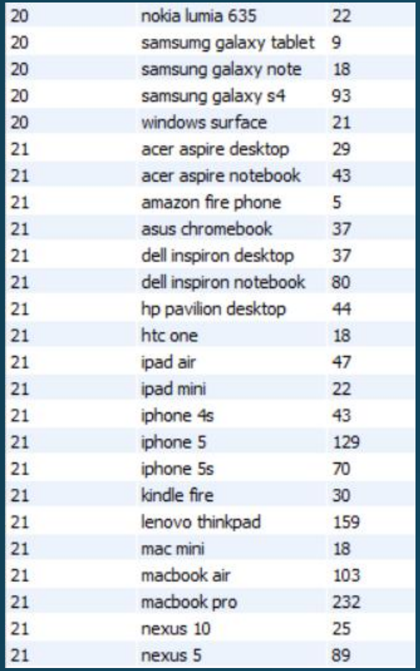




week_number	device	active_users
17	acer aspire desktop	9
17	acer aspire notebook	20
17	amazon fire phone	4
17	asus chromebook	21
17	dell inspiron desktop	18
17	dell inspiron notebook	46
17	hp pavilion desktop	14
17	htc one	16
17	ipad air	27
17	ipad mini	19
17	iphone 4s	21
17	iphone 5	65
17	iphone 5s	42
17	kindle fire	6
17	lenovo thinkpad	86
17	mac mini	6
17	macbook air	54
17	macbook pro	143
17	nexus 10	16
17	nexus 5	40
17	nexus 7	18
17	nokia lumia 635	17
17	samsumg galaxy tablet	8
17	samsung galaxy note	7
17	samsung galaxy s4	52
17	samsung galaxy s4	52
17	windows surface	10
18	acer aspire desktop	26
18	acer aspire notebook	33
18	amazon fire phone	9
18	asus chromebook	42
18	dell inspiron desktop	58
18	dell inspiron notebook	77
18	hp pavilion desktop	37
18	htc one	19
18	ipad air	52
18	ipad mini	30
18	iphone 4s	46
18	iphone 5	113
18	iphone 5s	73
18	kindle fire	27
18	lenovo thinkpad	153
18	mac mini	13
18	macbook air	121
18	macbook pro	252
18	nexus 10	30
18	nexus 5	73
18	nexus 7	30
18	nokia lumia 635	33
18	samsumg galaxy tablet	11
18	samsung galaxy note	15
18	samsung galaxy s4	82
18	windows surface	10
19	acer aspire desktop	23
19	acer aspire notebook	41
19	amazon fire phone	12
19	asus chromebook	27
19	dell inspiron desktop	36
19	dell inspiron notebook	83
19	hp pavilion desktop	40
19	htc one	30
19	ipad air	55
19	ipad mini	36
19	iphone 4s	44
19	iphone 5	115
19	iphone 5s	79
19	kindle fire	21
19	lenovo thinkpad	178
19	mac mini	18
19	macbook air	112
19	macbook pro	266
19	nexus 10	25
19	nexus 5	87
19	nexus 7	41

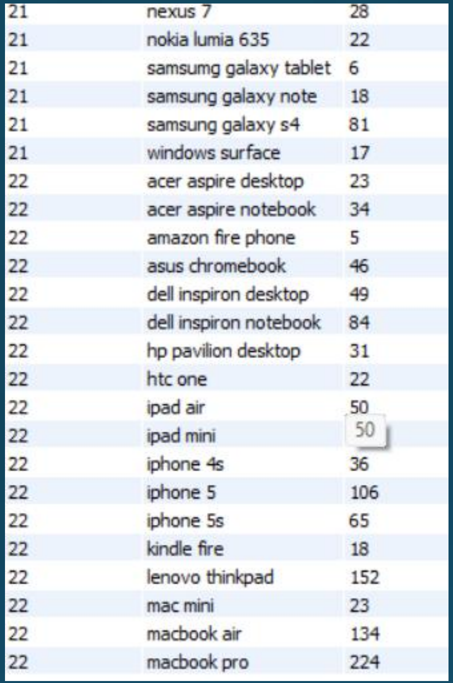
RESULT :



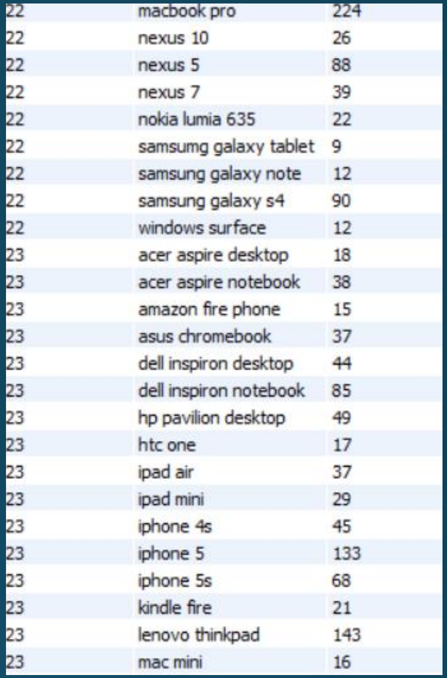
19	nokia lumia 635	23
19	samsumg galaxy tablet	6
19	samsung galaxy note	11
19	samsung galaxy s4	91
19	windows surface	16
20	acer aspire desktop	23
20	acer aspire notebook	40
20	amazon fire phone	11
20	asus chromebook	41
20	dell inspiron desktop	52
20	dell inspiron notebook	84
20	hp pavilion desktop	30
20	htc one	29
20	ipad air	59
20	ipad mini	32
20	iphone 4s	55
20	iphone 5	125
20	iphone 5s	79
20	kindle fire	23
20	lenovo thinkpad	173
20	mac mini	26
20	macbook air	119
20	macbook pro	256
20	nexus 10	22



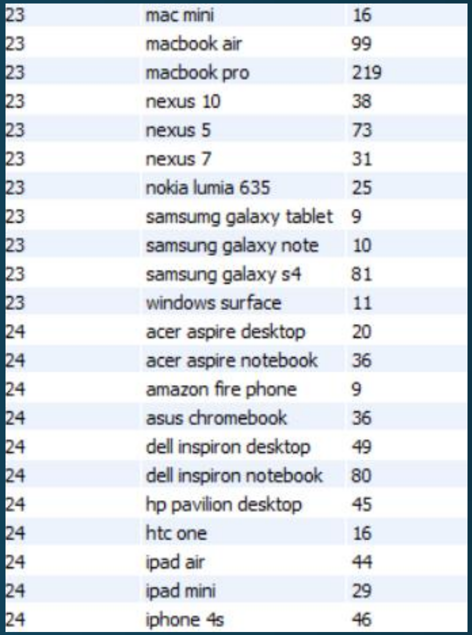
20	nokia lumia 635	22
20	samsumg galaxy tablet	9
20	samsung galaxy note	18
20	samsung galaxy s4	93
20	windows surface	21
21	acer aspire desktop	29
21	acer aspire notebook	43
21	amazon fire phone	5
21	asus chromebook	37
21	dell inspiron desktop	37
21	dell inspiron notebook	80
21	hp pavilion desktop	44
21	htc one	18
21	ipad air	47
21	ipad mini	22
21	iphone 4s	43
21	iphone 5	129
21	iphone 5s	70
21	kindle fire	30
21	lenovo thinkpad	159
21	mac mini	18
21	macbook air	103
21	macbook pro	232
21	nexus 10	25
21	nexus 5	89



21	nexus 7	28
21	nokia lumia 635	22
21	samsumg galaxy tablet	6
21	samsung galaxy note	18
21	samsung galaxy s4	81
21	windows surface	17
22	acer aspire desktop	23
22	acer aspire notebook	34
22	amazon fire phone	5
22	asus chromebook	46
22	dell inspiron desktop	49
22	dell inspiron notebook	84
22	hp pavilion desktop	31
22	htc one	22
22	ipad air	50
22	ipad mini	50
22	iphone 4s	36
22	iphone 5	106
22	iphone 5s	65
22	kindle fire	18
22	lenovo thinkpad	152
22	mac mini	23
22	macbook air	134
22	macbook pro	224



22	macbook pro	224
22	nexus 10	26
22	nexus 5	88
22	nexus 7	39
22	nokia lumia 635	22
22	samsumg galaxy tablet	9
22	samsung galaxy note	12
22	samsung galaxy s4	90
22	windows surface	12
23	acer aspire desktop	18
23	acer aspire notebook	38
23	amazon fire phone	15
23	asus chromebook	37
23	dell inspiron desktop	44
23	dell inspiron notebook	85
23	hp pavilion desktop	49
23	htc one	17
23	ipad air	37
23	ipad mini	29
23	iphone 4s	45
23	iphone 5	133
23	iphone 5s	68
23	kindle fire	21
23	lenovo thinkpad	143
23	mac mini	16



23	mac mini	16
23	macbook air	99
23	macbook pro	219
23	nexus 10	38
23	nexus 5	73
23	nexus 7	31
23	nokia lumia 635	25
23	samsumg galaxy tablet	9
23	samsung galaxy note	10
23	samsung galaxy s4	81
23	windows surface	11
24	acer aspire desktop	20
24	acer aspire notebook	36
24	amazon fire phone	9
24	asus chromebook	36
24	dell inspiron desktop	49
24	dell inspiron notebook	80
24	hp pavilion desktop	45
24	htc one	16
24	ipad air	44
24	ipad mini	29
24	iphone 4s	46

Table continues

PERFORM ANALYSIS

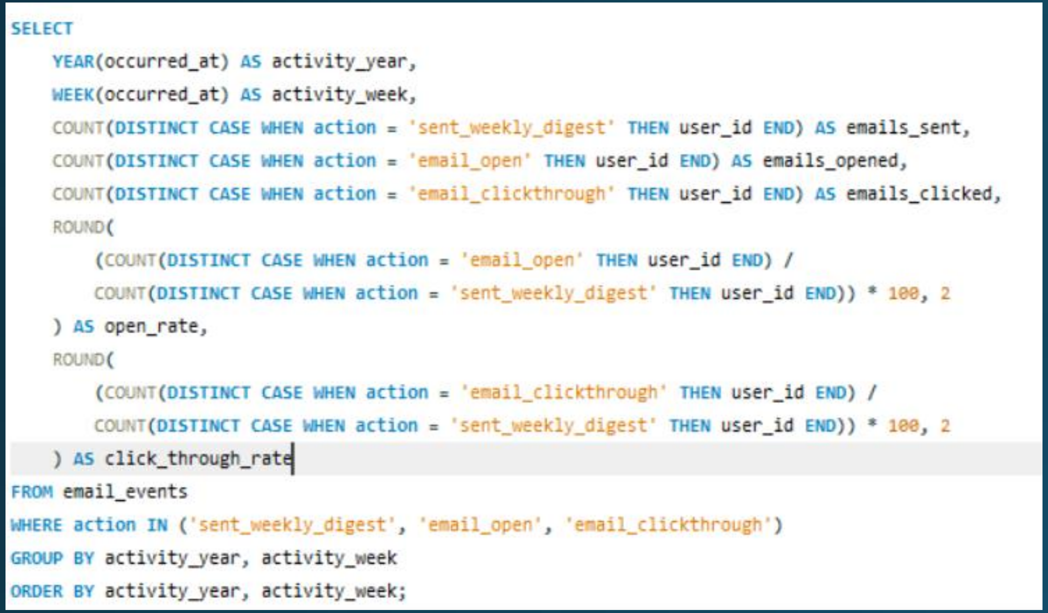
Case Study 2: Investigating Metric Spike

QUERY 5 : To write an SQL query to calculate the email engagement metrics.

PROJECT DESCRIPTION :EMAIL ENGAGEMENT ANALYSIS

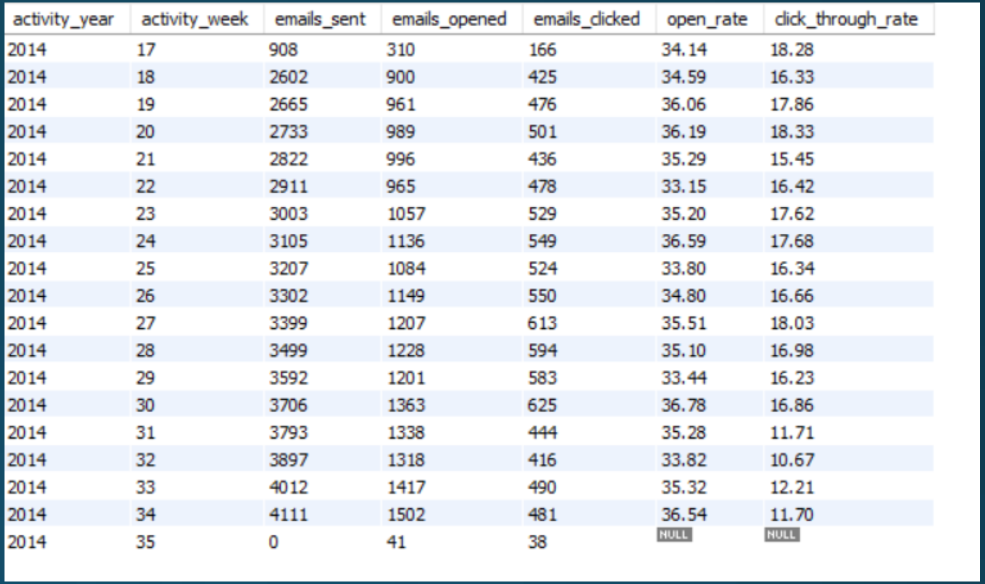
To analyze how users are engaging with the email service.

•QUERY 5:



```
SELECT
    YEAR(occurred_at) AS activity_year,
    WEEK(occurred_at) AS activity_week,
    COUNT(DISTINCT CASE WHEN action = 'sent_weekly_digest' THEN user_id END) AS emails_sent,
    COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) AS emails_opened,
    COUNT(DISTINCT CASE WHEN action = 'email_clickthrough' THEN user_id END) AS emails_clicked,
    ROUND(
        (COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) /
        COUNT(DISTINCT CASE WHEN action = 'sent_weekly_digest' THEN user_id END)) * 100, 2
    ) AS open_rate,
    ROUND(
        (COUNT(DISTINCT CASE WHEN action = 'email_clickthrough' THEN user_id END) /
        COUNT(DISTINCT CASE WHEN action = 'sent_weekly_digest' THEN user_id END)) * 100, 2
    ) AS click_through_rate
FROM email_events
WHERE action IN ('sent_weekly_digest', 'email_open', 'email_clickthrough')
GROUP BY activity_year, activity_week
ORDER BY activity_year, activity_week;
```

RESULT :



activity_year	activity_week	emails_sent	emails_opened	emails_clicked	open_rate	click_through_rate
2014	17	908	310	166	34.14	18.28
2014	18	2602	900	425	34.59	16.33
2014	19	2665	961	476	36.06	17.86
2014	20	2733	989	501	36.19	18.33
2014	21	2822	996	436	35.29	15.45
2014	22	2911	965	478	33.15	16.42
2014	23	3003	1057	529	35.20	17.62
2014	24	3105	1136	549	36.59	17.68
2014	25	3207	1084	524	33.80	16.34
2014	26	3302	1149	550	34.80	16.66
2014	27	3399	1207	613	35.51	18.03
2014	28	3499	1228	594	35.10	16.98
2014	29	3592	1201	583	33.44	16.23
2014	30	3706	1363	625	36.78	16.86
2014	31	3793	1338	444	35.28	11.71
2014	32	3897	1318	416	33.82	10.67
2014	33	4012	1417	490	35.32	12.21
2014	34	4111	1502	481	36.54	11.70
2014	35	0	41	38	NULL	NULL

TECH-STACK USED

SOFTWARE : MYSQL WORKBENCH 8.O CE

IMPORTANCE OF MYSQL :

1. Security: MySQL provides security features like user authentication and data encryption to protect data that database hold.
2. Speed: It executes the query very fast and high performance, even with large datasets.
3. Simplicity: It is easy to learn and use.
4. Accuracy: It ensures data integrity with features like constraints and transaction management.
5. Accessibility: We can accessible across platforms and integrates with various applications and programming languages.

INSIGHTS

- **Operational Analytics Importance** – Helps to identify inefficiency, optimize , and support data-driven decision-making.
- **Investigating Metric Spikes** –Will detect sudden changes (e.g., drops in engagement or sales) using SQL to find root causes.
- **Job Data Analysis** – Examines workload , throughput , language prefered, and data integrity.
- **User & Email Engagement Analysis** – Tracks user activity, retention, device-based interactions, and email effectiveness.
- **Advanced SQL Skills** – Enhances expertize in time-based aggregations, averages, cohort analysis, and window functions.

CONCLUSION

- Operational Analytics helps identify trends,optimize process,& investigate sudden metric change.
- Using Advanced SQL, analysts can know the user engagement ,and usage and take the appropriate decision which helps to get the things in a better way.
- Enabling the techniques of these helps to take the decision making easy.

THANK YOU