OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKES

Presented by
Ashiq Paul

PROJECT DESCRIPTION

Operation Analytics and Investigating Metric Spikes

This project analyzes the operational data and investigate the metric spikes across different departments using SQL Queries in MySQL Workbench. It involves job review trends and user engagement analysis for business insights.

APPROACH

The Approach taken for this project

- Imported the CSVs into the MySQL Workbench
- Explored the table structures
- Wrote and tested the SQL Queries
- Analyzed and captured the outputs
- Summarized the insights
- Prepared the report

TECH STACK

The Tech-Stacks Used

MySQL Workbench

I used SQL for this project for querying and analyzing the large datasets that was provided.

Microsoft Excel

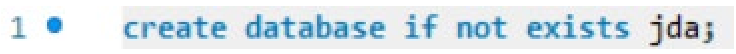
To view the csv files and remove any blanks present.

Microsoft Word / Adobe Acrobat / Canva

Microsoft Word Used for preparing and structuring the report and converting to PDF. Canva used for the title page.

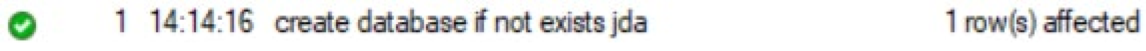
Job Data Analytics

Created a Database named ‘jda’

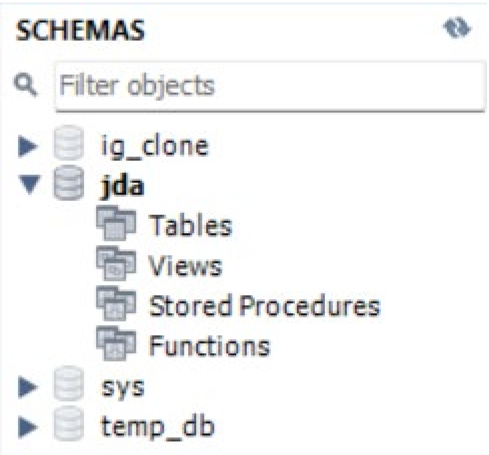


```
create database if not exists jda;
```

Output:



```
1   14:14:16   create database if not exists jda            1 row(s) affected
```



Then I imported the csv file into the database with the ‘Table Data Import Wizard’

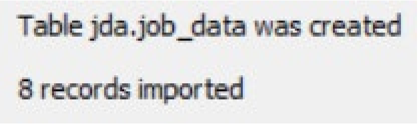
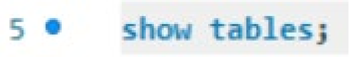


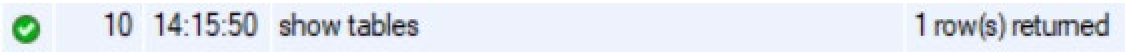
Table jda.job_data was created

8 records imported

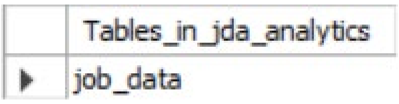


```
show tables;
```

Output:

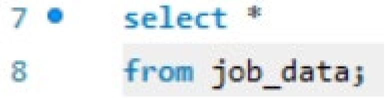


```
10   14:15:50   show tables            1 row(s) returned
```



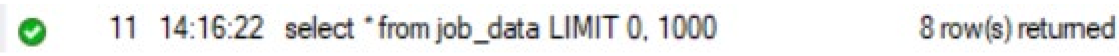
	Tables_in_jda_analytics
▶	job_data

Checking the table and its data

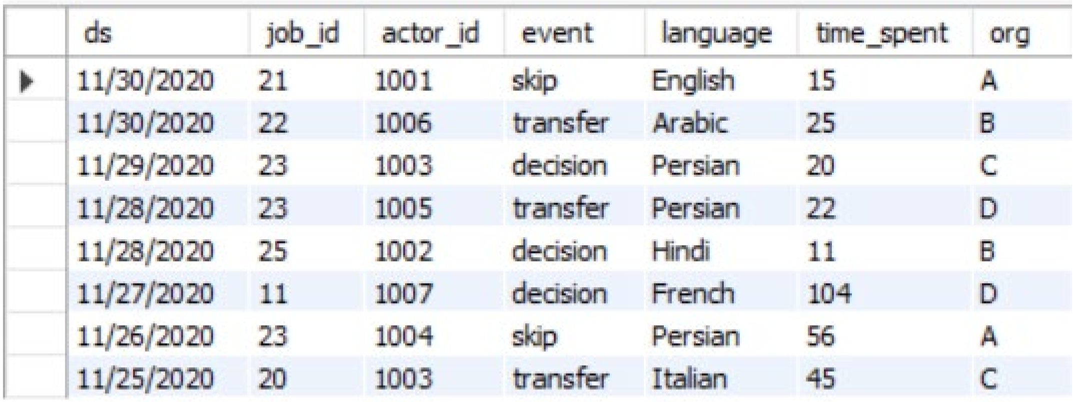


```
select *
from job_data;
```

Output:



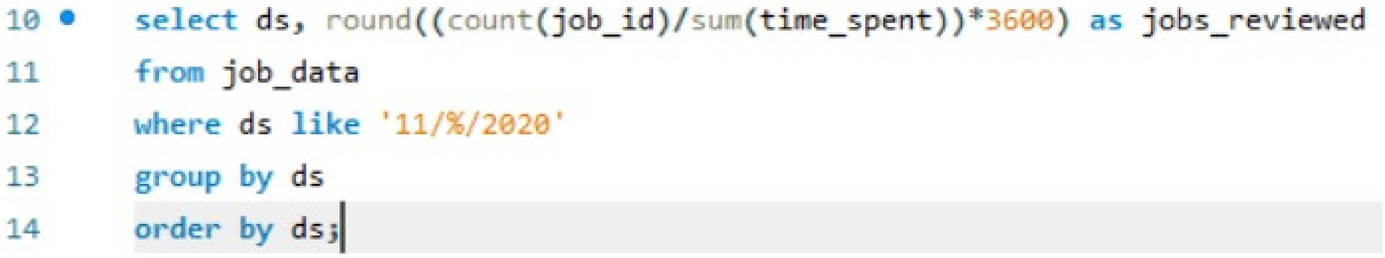
```
11   14:16:22   select * from job_data LIMIT 0, 1000            8 row(s) returned
```



	ds	job_id	actor_id	event	language	time_spent	org
▶	11/30/2020	21	1001	skip	English	15	A
	11/30/2020	22	1006	transfer	Arabic	25	B
	11/29/2020	23	1003	decision	Persian	20	C
	11/28/2020	23	1005	transfer	Persian	22	D
	11/28/2020	25	1002	decision	Hindi	11	B
	11/27/2020	11	1007	decision	French	104	D
	11/26/2020	23	1004	skip	Persian	56	A
	11/25/2020	20	1003	transfer	Italian	45	C

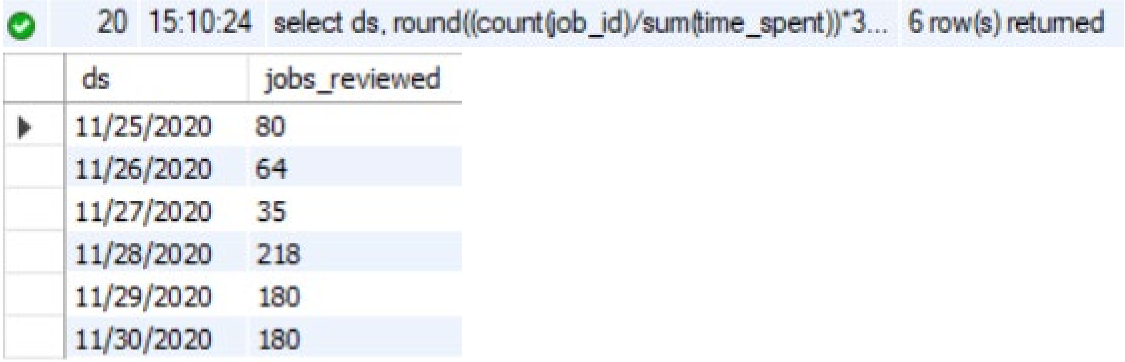
A. Jobs Reviewed Over Time

Hourly jobs reviewed per day for November 2020
Since the given time is in seconds we multiply by 3600 to convert it into hours



```
select ds, round((count(job_id)/sum(time_spent))*3600) as jobs_reviewed
from job_data
where ds like '11/%/2020'
group by ds
order by ds;
```

Output:



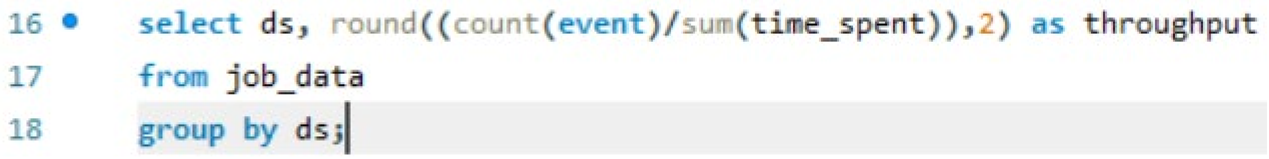
```
20  15:10:24  select ds, round((count(job_id)/sum(time_spent))*3...  6 row(s) returned
```

	ds	jobs_reviewed
▶	11/25/2020	80
	11/26/2020	64
	11/27/2020	35
	11/28/2020	218
	11/29/2020	180
	11/30/2020	180

The highest jobs reviewed were on 28 November 2020 with the lowest being on 27 November 2020, it varies between days. From this we can investigate why the jobs reviewed were lower.

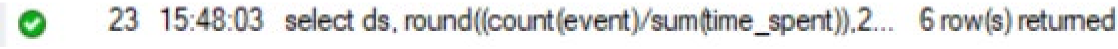
B. Throughput Analysis

7-day rolling average of throughput

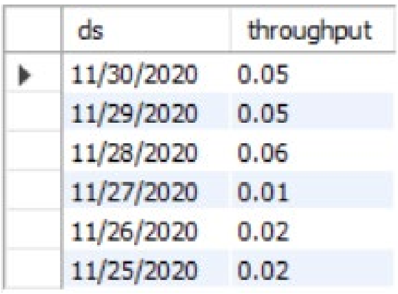


```
select ds, round((count(event)/sum(time_spent)),2) as throughput
from job_data
group by ds;
```

Output:



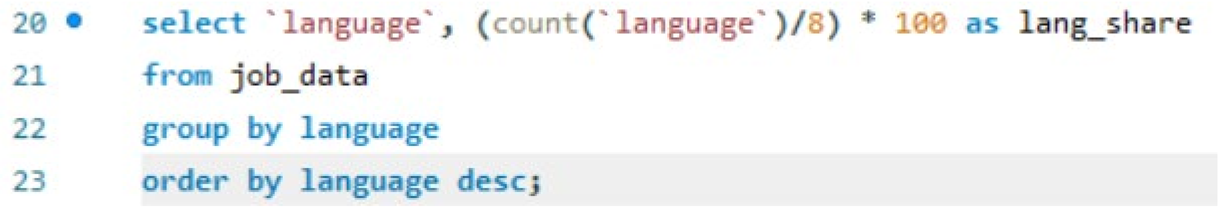
```
23  15:48:03  select ds, round((count(event)/sum(time_spent)),2...  6 row(s) returned
```



ds	throughput
11/30/2020	0.05
11/29/2020	0.05
11/28/2020	0.06
11/27/2020	0.01
11/26/2020	0.02
11/25/2020	0.02

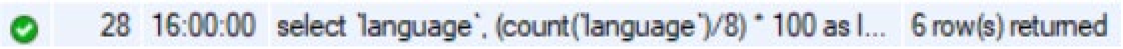
The 7-day rolling average of throughput shows a clean and reliable view of the data and we can observe the trends over time without any daily fluctuations.

C. Language Share Analysis

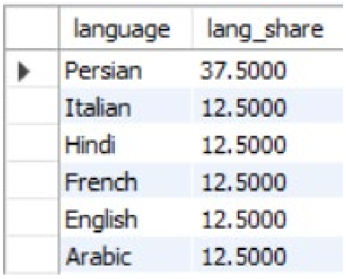


```sql
select `language`, (count(`language`)/8) * 100 as lang_share
from job_data
group by language
order by language desc;
```

Output:



```
28   16:00:00   select `language`, (count(`language`)/8) * 100 as l...   6 row(s) returned
```

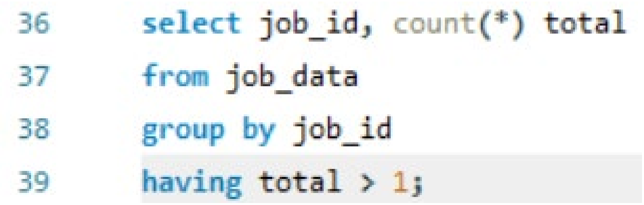


language	lang_share
Persian	37.5000
Italian	12.5000
Hindi	12.5000
French	12.5000
English	12.5000
Arabic	12.5000

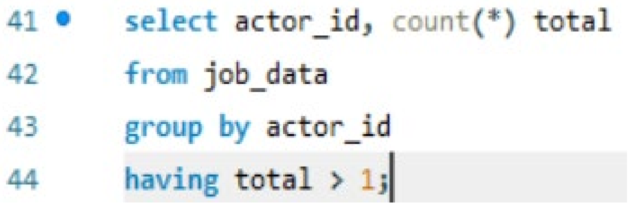
Persian language has the highest percentage of language share, rest of the languages are evenly shared.

D. Duplicate Row Detection

Identify the duplicate rows in the data

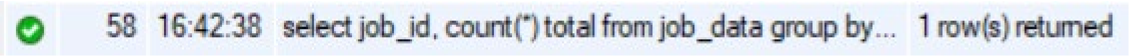


```sql
select job_id, count(*) total
from job_data
group by job_id
having total > 1;
```

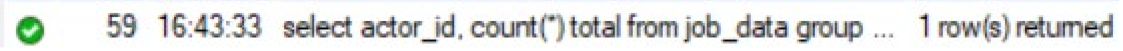


```sql
select actor_id, count(*) total
from job_data
group by actor_id
having total > 1;
```

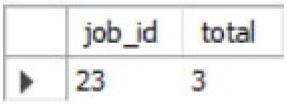
Output:



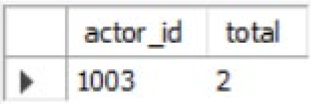
```
58   16:42:38   select job_id, count(*) total from job_data group by...   1 row(s) returned
```



```
59   16:43:33   select actor_id, count(*) total from job_data group ...   1 row(s) returned
```



job_id	total
23	3

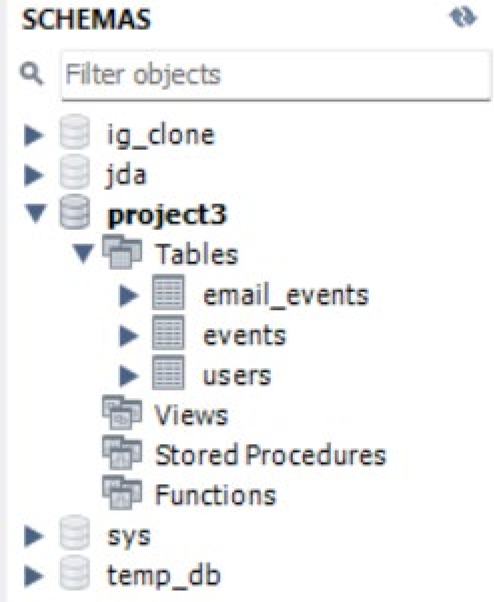


actor_id	total
1003	2

‘job_id’ and ‘actor_id’ are unique identifiers and they contain duplicate rows
There are 2 duplicate rows for ‘job_id’ and 1 duplicate row for ‘actor_id’.

Investigating Metric Spike

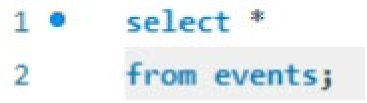
Created a Database named ‘Project3’ and imported ‘users’, ‘events’ and ‘email_events’ csv files into the database as per the guide video provided.



```
SCHEMAS
Filter objects
▶ ig_clone
▶ jda
▼ project3
    ▼ Tables
        ▶ email_events
        ▶ events
        ▶ users
      Views
      Stored Procedures
      Functions
▶ sys
▶ temp_db
```

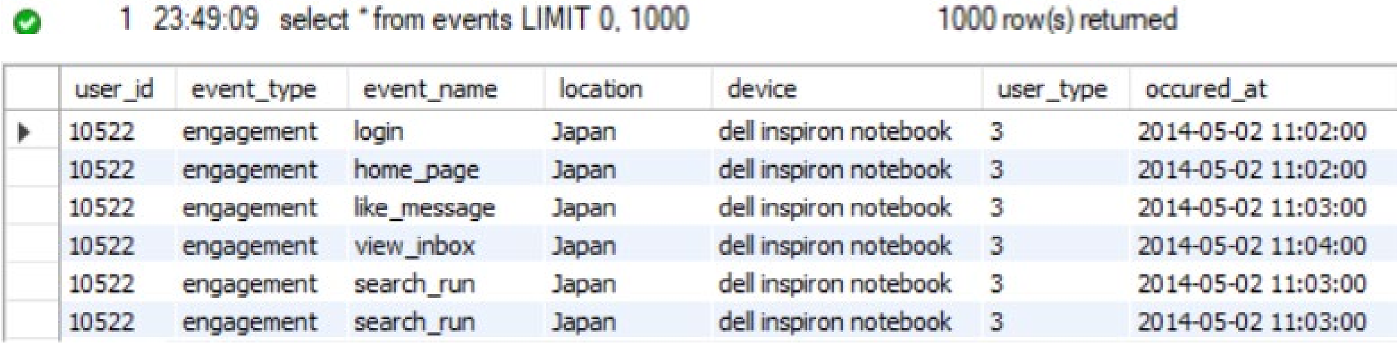
A. Weekly User Engagement

First we check the events table



```sql
select *
from events;
```

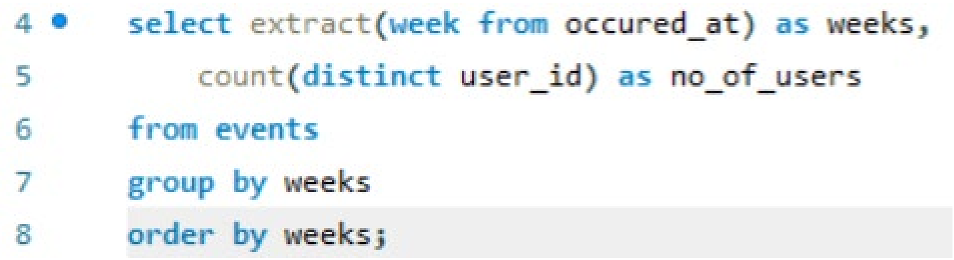
Ouput:



```
1  23:49:09   select * from events LIMIT 0, 1000                    1000 row(s) returned
```

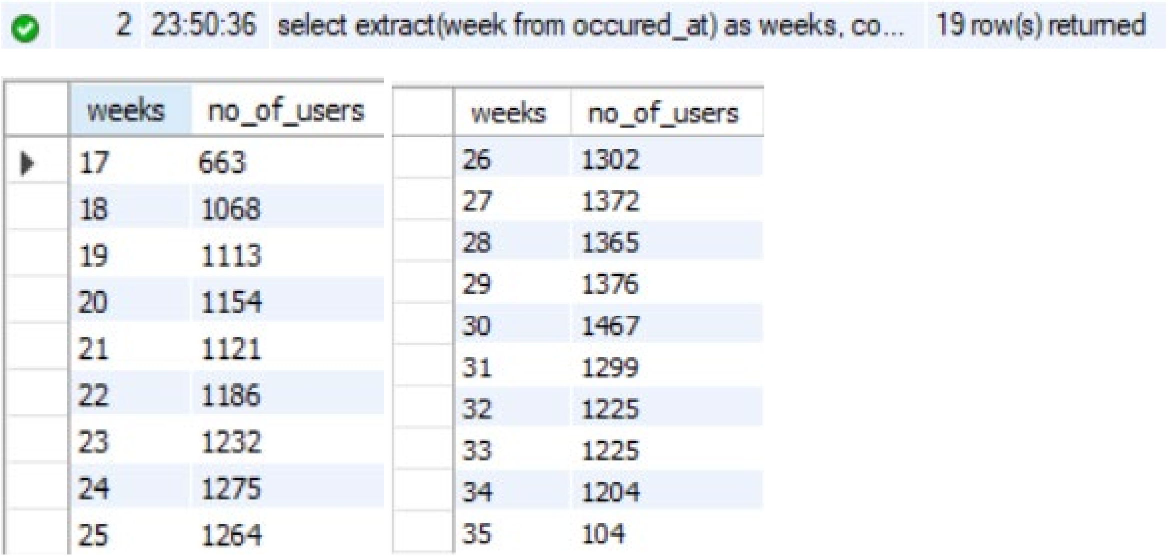
user_id	event_type	event_name	location	device	user_type	occured_at
10522	engagement	login	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	home_page	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	like_message	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10522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

To measure the activeness of users on a weekly basis



```sql
select extract(week from occured_at) as weeks,
    count(distinct user_id) as no_of_users
from events
group by weeks
order by weeks;
```

Output:



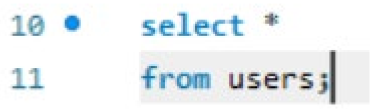
```
2  23:50:36  select extract(week from occured_at) as weeks, co...  19 row(s) returned
```

weeks	no_of_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
35	104

The highest user engagement is on week 30 and the lowest user engagement is on week 35. We can see the fluctuations in the observed time period.

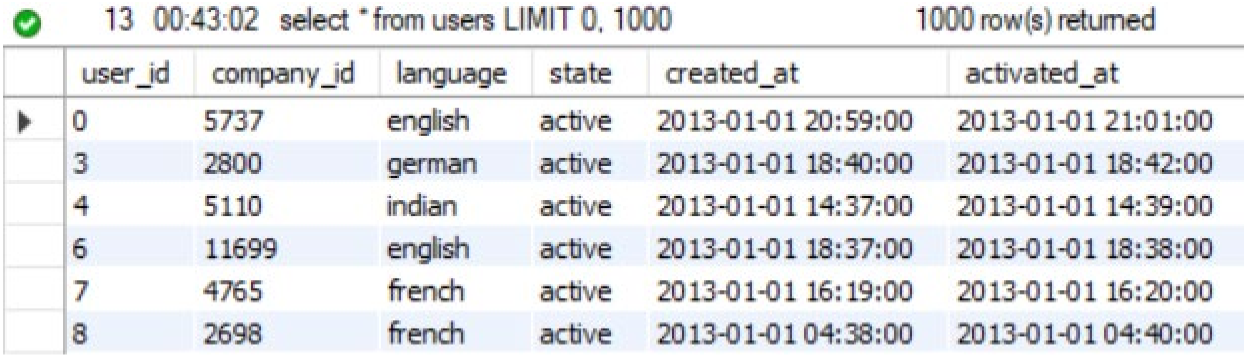
This helps to track the weekly user engagement and identify any spikes or dips.

B. User Growth Analysis



```
select *
from users;
```

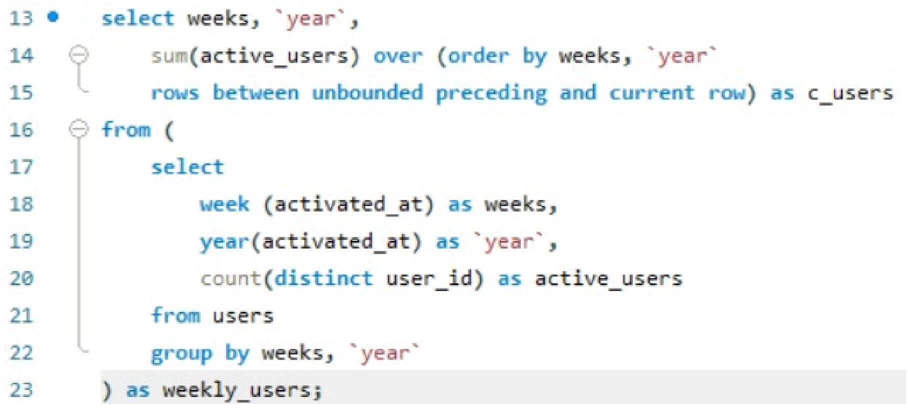
Output:



```
13  00:43:02  select * from users LIMIT 0, 1000      1000 row(s) returned
```

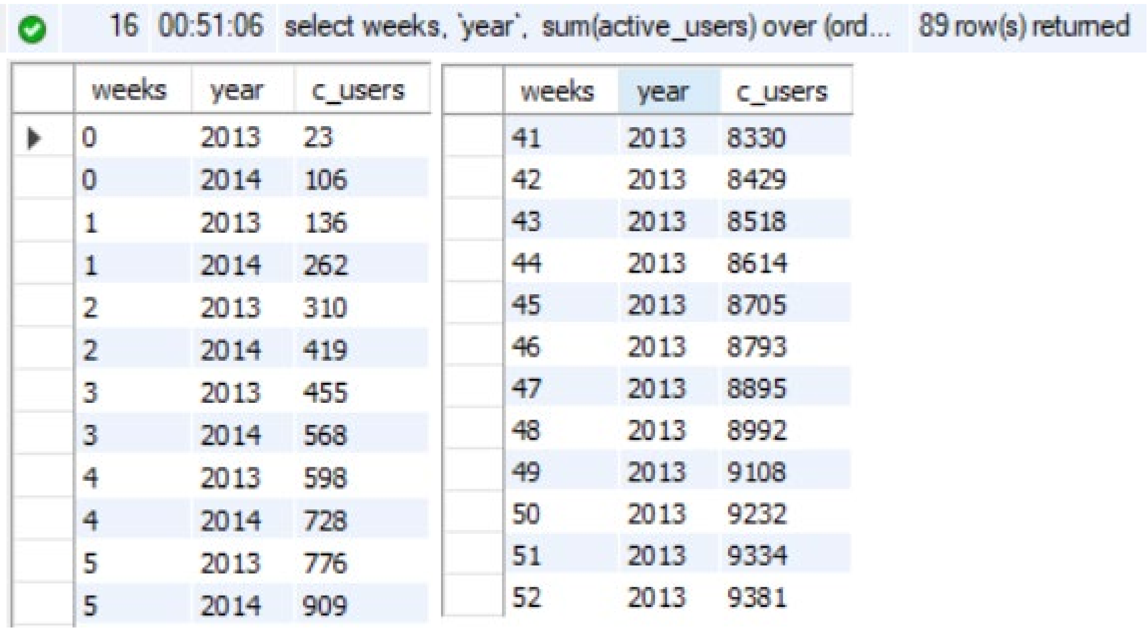
user_id	company_id	language	state	created_at	activated_at
0	5737	english	active	2013-01-01 20:59:00	2013-01-01 21:01:00
3	2800	german	active	2013-01-01 18:40:00	2013-01-01 18:42:00
4	5110	indian	active	2013-01-01 14:37:00	2013-01-01 14:39:00
6	11699	english	active	2013-01-01 18:37:00	2013-01-01 18:38:00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

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



```
select weeks, `year`,
    sum(active_users) over (order by weeks, `year`
    rows between unbounded preceding and current row) as c_users
from (
    select
        week (activated_at) as weeks,
        year(activated_at) as `year`,
        count(distinct user_id) as active_users
    from users
    group by weeks, `year`
) as weekly_users;
```

Output:



16 00:51:06 select weeks, `year`, sum(active_users) over (ord... 89 row(s) returned

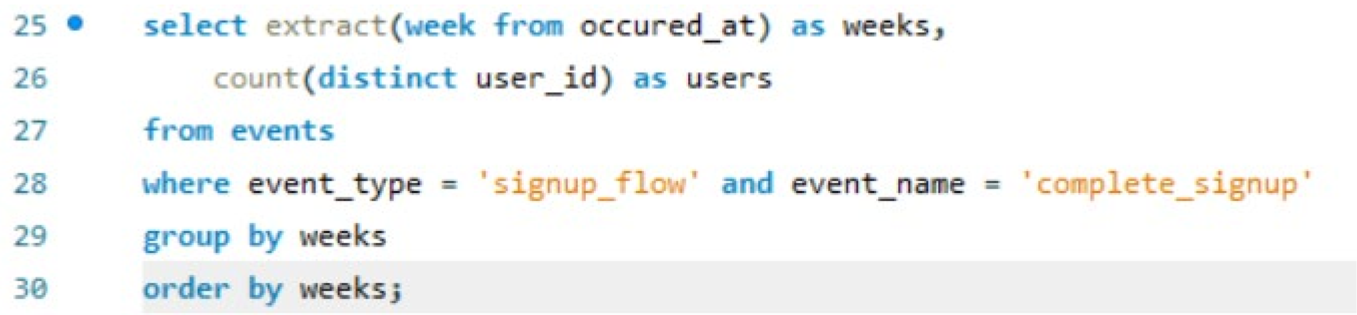
weeks	year	c_users
0	2013	23
0	2014	106
1	2013	136
1	2014	262
2	2013	310
2	2014	419
3	2013	455
3	2014	568
4	2013	598
4	2014	728
5	2013	776
5	2014	909

weeks	year	c_users
41	2013	8330
42	2013	8429
43	2013	8518
44	2013	8614
45	2013	8705
46	2013	8793
47	2013	8895
48	2013	8992
49	2013	9108
50	2013	9232
51	2013	9334
52	2013	9381

The above query calculated the weekly user growth and provided a cumulative total over time. From this we can see that the user growth has been positive over time but there are some fluctuations.

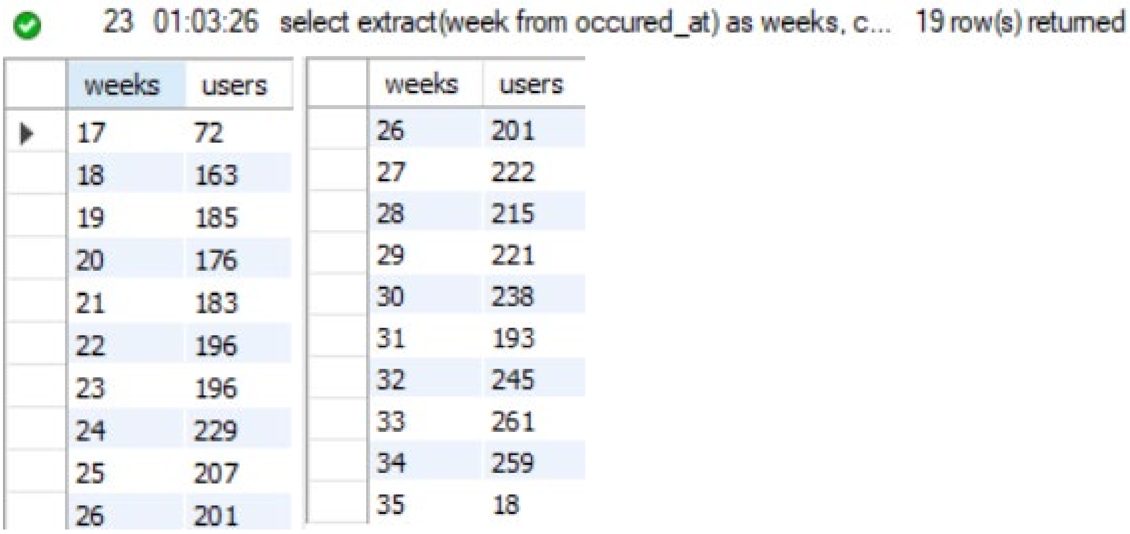
C. Weekly Retention Analysis

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



```sql
select extract(week from occured_at) as weeks,
    count(distinct user_id) as users
from events
where event_type = 'signup_flow' and event_name = 'complete_signup'
group by weeks
order by weeks;
```

Output:



23 01:03:26 select extract(week from occured_at) as weeks, c... 19 row(s) returned

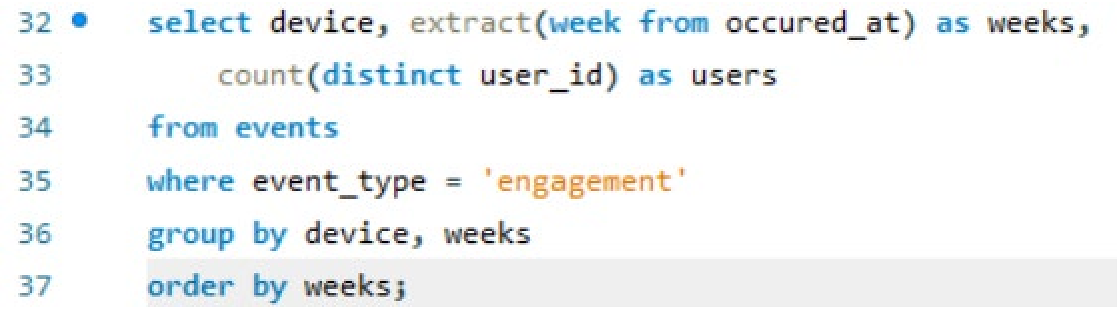
weeks	users
17	72
18	163
19	185
20	176
21	183
22	196
23	196
24	229
25	207
26	201

weeks	users
26	201
27	222
28	215
29	221
30	238
31	193
32	245
33	261
34	259
35	18

The weekly user retention has seen fluctuations and a decline at week 35.
Few weeks shows better retention which could be due to product updated or promotions.
But this shows the importance of improving user experience and engagement.

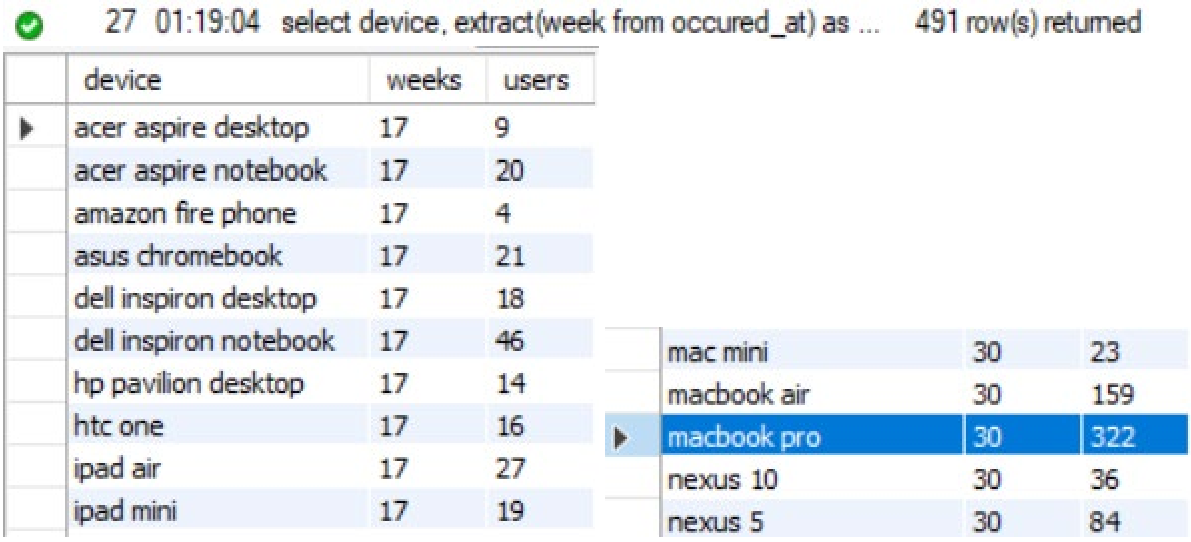
D. Weekly Engagement Per Device

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



```sql
select device, extract(week from occured_at) as weeks,
    count(distinct user_id) as users
from events
where event_type = 'engagement'
group by device, weeks
order by weeks;
```

Output:



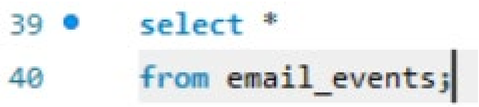
```
27  01:19:04  select device, extract(week from occured_at) as ...   491 row(s) returned
```

device	weeks	users
acer aspire desktop	17	9
acer aspire notebook	17	20
amazon fire phone	17	4
asus chromebook	17	21
dell inspiron desktop	17	18
dell inspiron notebook	17	46
hp pavilion desktop	17	14
htc one	17	16
ipad air	17	27
ipad mini	17	19
mac mini	30	23
macbook air	30	159
macbook pro	30	322
nexus 10	30	36
nexus 5	30	84

The user engagement varies across different devices. Most users were found to be using the device “Macbook Pro” with 322 users on week 30. Also some devices shows higher engagement than other devices.

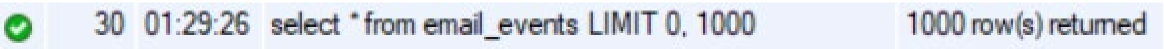
E. Email Engagement Analysis

First we check the email_events table

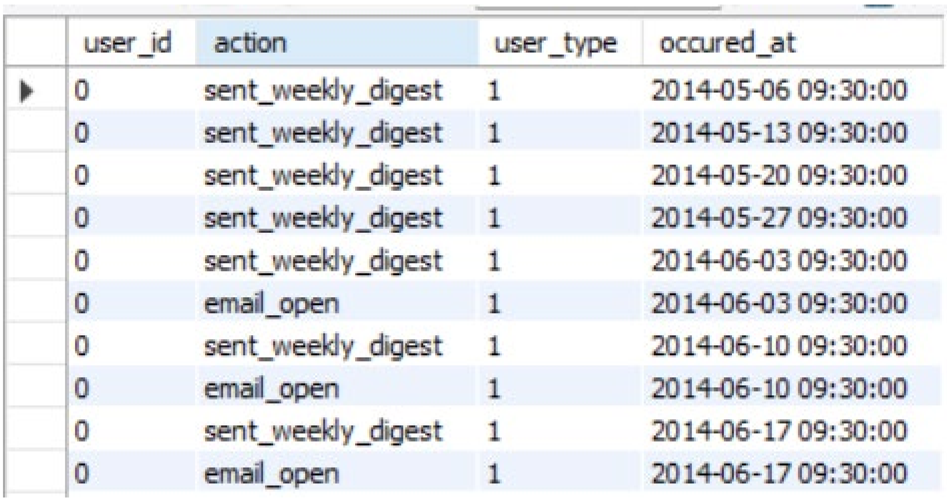


```sql
select *
from email_events;
```

Output:

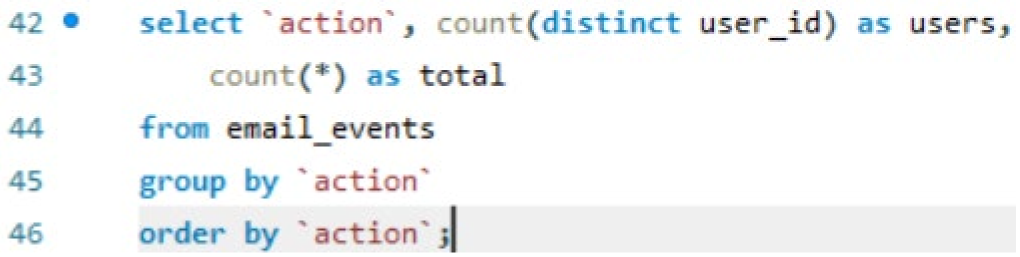


```
30  01:29:26  select * from email_events LIMIT 0, 1000   1000 row(s) returned
```



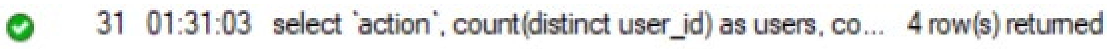
	user_id	action	user_type	occured_at
▶	0	sent_weekly_digest	1	2014-05-06 09:30:00
	0	sent_weekly_digest	1	2014-05-13 09:30:00
	0	sent_weekly_digest	1	2014-05-20 09:30:00
	0	sent_weekly_digest	1	2014-05-27 09:30:00
	0	sent_weekly_digest	1	2014-06-03 09:30:00
	0	email_open	1	2014-06-03 09:30:00
	0	sent_weekly_digest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

To analyze how users are engaging with the email service

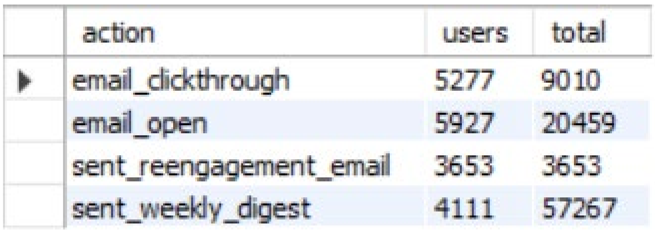


```
select `action`, count(distinct user_id) as users,
     count(*) as total
from email_events
group by `action`
order by `action`;
```

Output:

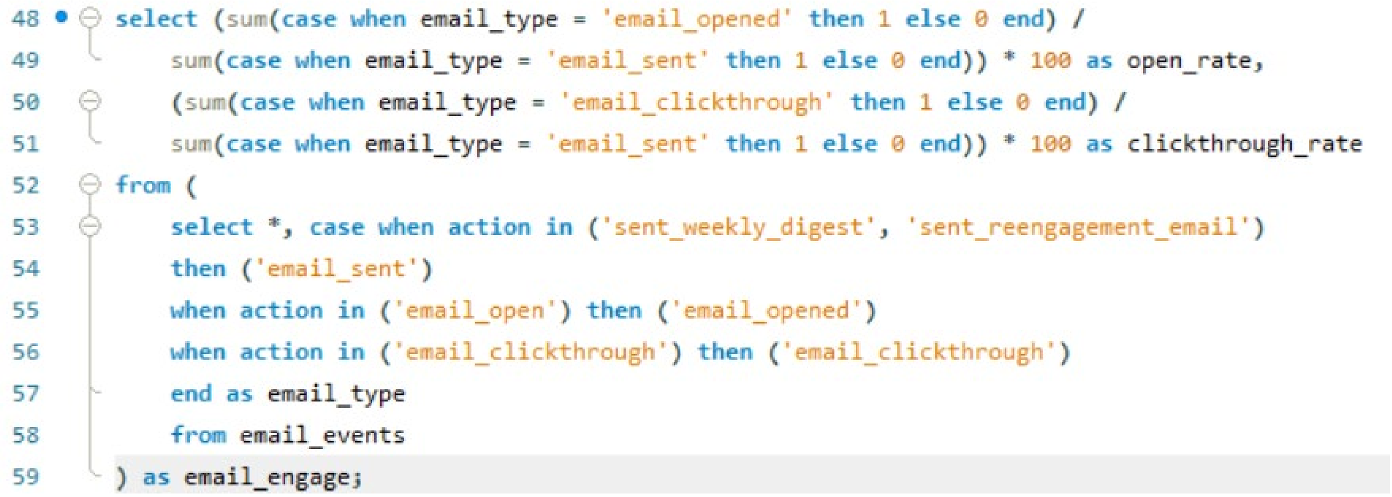


```
31  01:31:03  select `action`, count(distinct user_id) as users, co...  4 row(s) returned
```



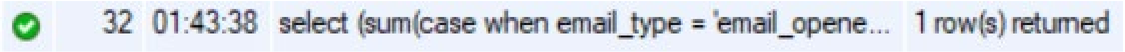
	action	users	total
▶	email_clickthrough	5277	9010
	email_open	5927	20459
	sent_reengagement_email	3653	3653
	sent_weekly_digest	4111	57267

To find the rate at which the users are clicking and opening the email

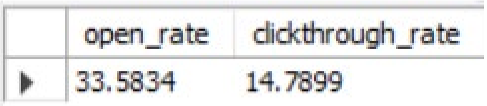


```
select (sum(case when email_type = 'email_opened' then 1 else 0 end) /
    sum(case when email_type = 'email_sent' then 1 else 0 end)) * 100 as open_rate,
    (sum(case when email_type = 'email_clickthrough' then 1 else 0 end) /
    sum(case when email_type = 'email_sent' then 1 else 0 end)) * 100 as clickthrough_rate
from (
    select *, case when action in ('sent_weekly_digest', 'sent_reengagement_email')
    then ('email_sent')
    when action in ('email_open') then ('email_opened')
    when action in ('email_clickthrough') then ('email_clickthrough')
    end as email_type
    from email_events
) as email_engage;
```

Output:



```
32  01:43:38  select (sum(case when email_type = 'email_opene...  1 row(s) returned
```



	open_rate	clickthrough_rate
▶	33.5834	14.7899

We can see that the email opening rate is around 33.6% and email clickthrough rate is around 14.8%. From this we can understand that 33.6% of users are interested in the emails being sent and 14.8% indicates that users are opening the emails but only a small percentage are engaging further. This suggests an opportunity to optimize the email content and increase the users interaction.

RESULT

This project has helped me apply advanced SQL Queries to comprehensive operational and engagement analytics for a digital platform. I was able to successfully investigate the user behaviour metrics such as the weekly activity, growth trends, device usage and email engagement.

THANK YOU