Operation Analytics and Investigating Metric Spike

PROJECT DESCRIPTION

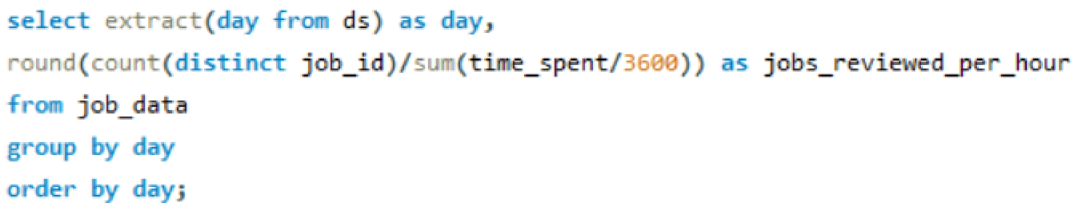
Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. This analysis helps identify areas for improvement within the company. One of the key aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales.

INSIGHTS

CASE STUDY 1: Job Data Analysis

A. Jobs Reviewed Over Time

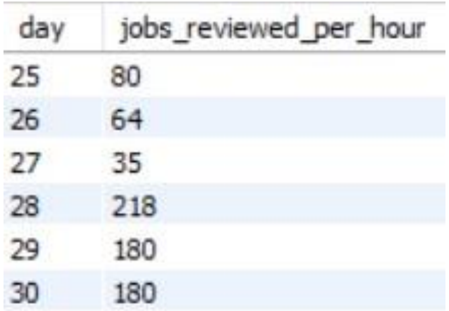
SQL Query:



```
select extract(day from ds) as day,
round(count(distinct job_id)/sum(time_spent/3600)) as jobs_reviewed_per_hour
from job_data
group by day
order by day;
```

In the table job_data, time_spent is given in seconds So converting it into hour and dividing it to number of users will give the jobs reviewed per hour.

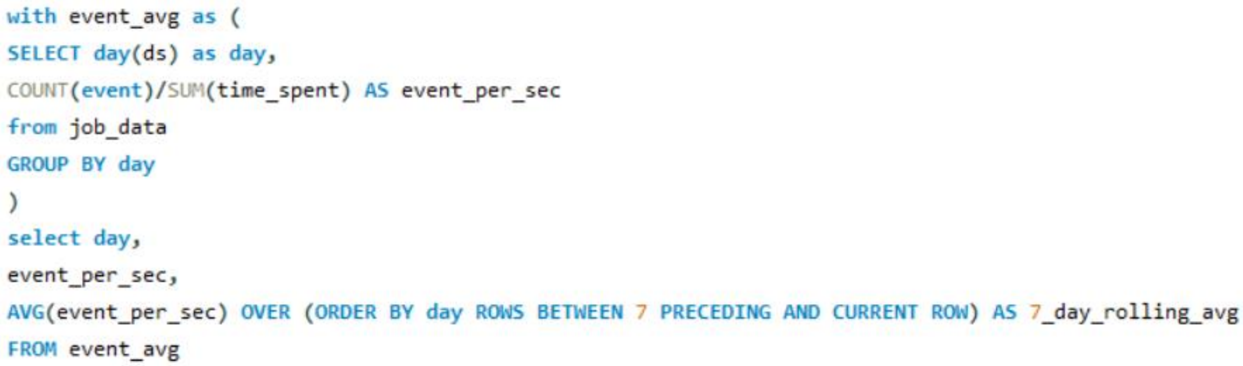
OUTPUT:



day	jobs_reviewed_per_hour
25	80
26	64
27	35
28	218
29	180
30	180

B. Throughput Analysis:

SQL Query:



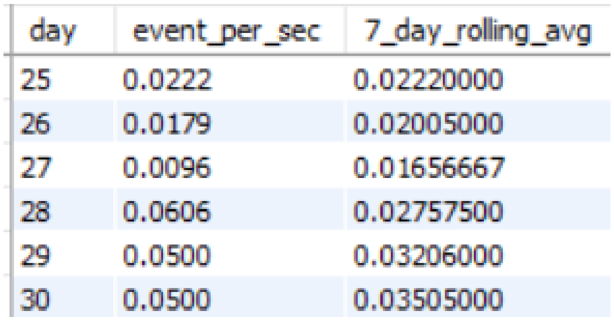
```
with event_avg as (
SELECT day(ds) as day,
COUNT(event)/SUM(time_spent) AS event_per_sec
from job_data
GROUP BY day
)
select day,
event_per_sec,
AVG(event_per_sec) OVER (ORDER BY day ROWS BETWEEN 7 PRECEDING AND CURRENT ROW) AS 7_day_rolling_avg
FROM event_avg
```

This shows the 7-day moving average vs events per second for the given dataset.

Rolling average helps us to identify a trend in conditions when it's difficult to identify a trend. It smoothens the curve and helps to analyze the trend easily.

Here the trend is in increasing direction

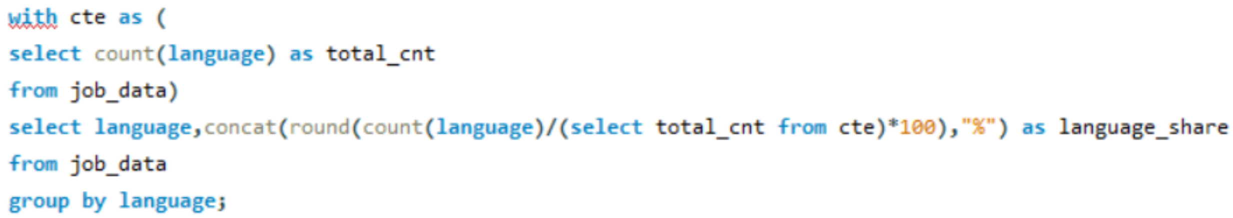
OUTPUT:



day	event_per_sec	7_day_rolling_avg
25	0.0222	0.02220000
26	0.0179	0.02005000
27	0.0096	0.01656667
28	0.0606	0.02757500
29	0.0500	0.03206000
30	0.0500	0.03505000

C. Language Share Analysis:

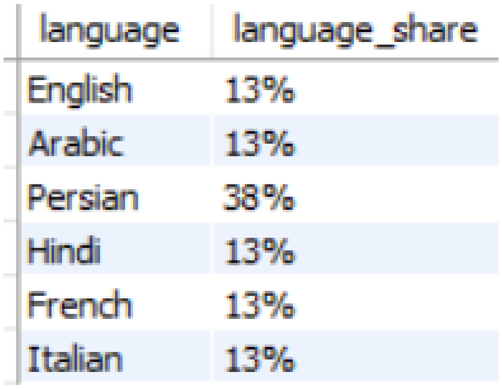
SQL Query:



```
with cte as (
select count(language) as total_cnt
from job_data)
select language,concat(round(count(language)/(select total_cnt from cte)*100),"%") as language_share
from job_data
group by language;
```

This query shows the percentage of language used in the job data.

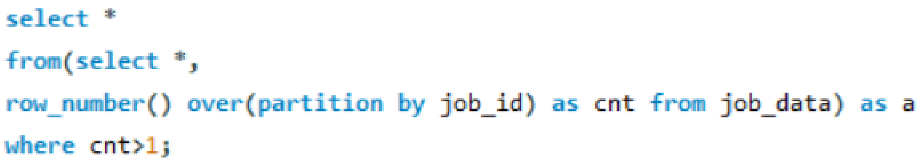
OUTPUT:



language	language_share
English	13%
Arabic	13%
Persian	38%
Hindi	13%
French	13%
Italian	13%

D. Duplicate Rows Detection

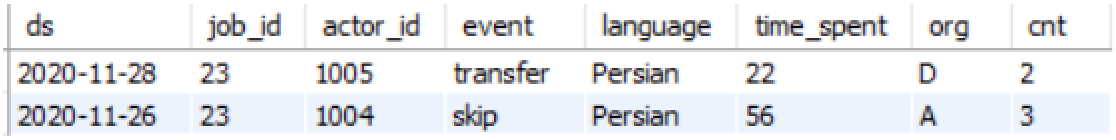
SQL Query:



```
select *
from(select *,
row_number() over(partition by job_id) as cnt from job_data) as a
where cnt>1;
```

This query returns the duplicate rows in job data.

OUTPUT:

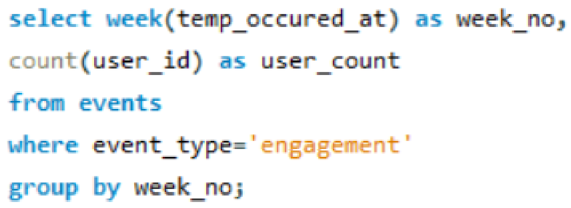


ds	job_id	actor_id	event	language	time_spent	org	cnt
2020-11-28	23	1005	transfer	Persian	22	D	2
2020-11-26	23	1004	skip	Persian	56	A	3

CASE STUDY 2: **Investigating Metric Spike**

A. **Weekly User Engagement**

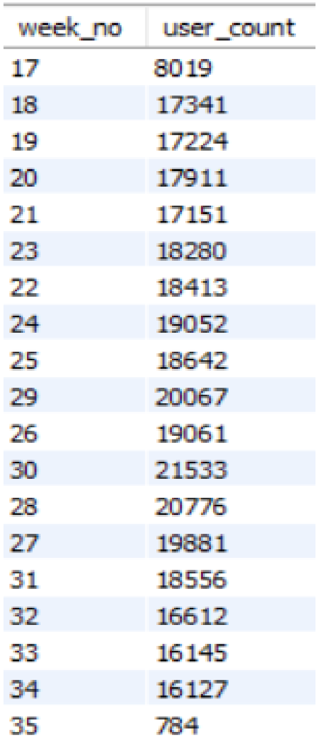
SQL Query:



```
select week(temp_occured_at) as week_no,
count(user_id) as user_count
from events
where event_type='engagement'
group by week_no;
```

This query returns the number of users who are engaged in some event each week

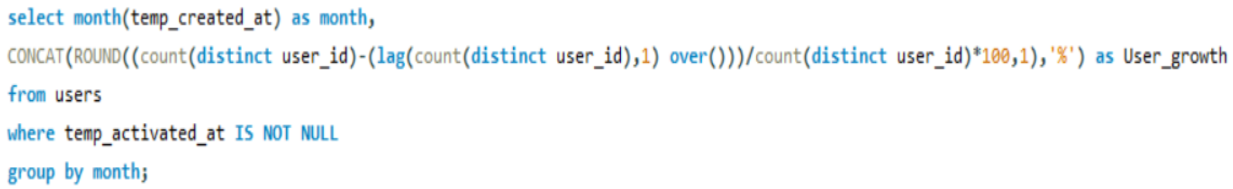
OUTPUT:



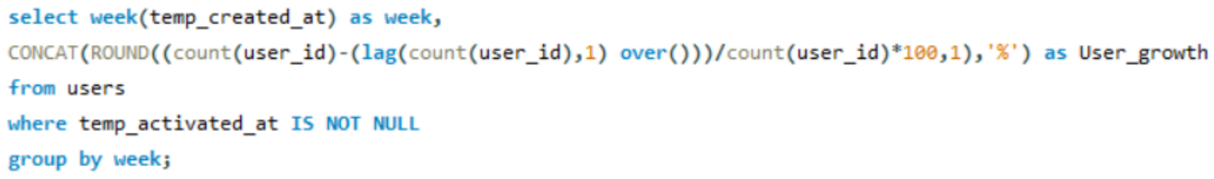
week_no	user_count
17	8019
18	17341
19	17224
20	17911
21	17151
23	18280
22	18413
24	19052
25	18642
29	20067
26	19061
30	21533
28	20776
27	19881
31	18556
32	16612
33	16145
34	16127
35	784

B. **User Growth Analysis:**

SQL Query:



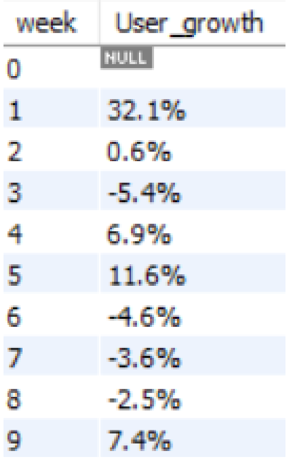
```
select month(temp_created_at) as month,
CONCAT(ROUND((count(distinct user_id)-(lag(count(distinct user_id),1) over()))/count(distinct user_id)*100,1),'%') as User_growth
from users
where temp_activated_at IS NOT NULL
group by month;
```



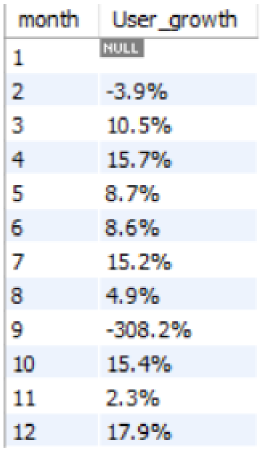
```
select week(temp_created_at) as week,
CONCAT(ROUND((count(user_id)-(lag(count(user_id),1) over()))/count(user_id)*100,1),'%') as User_growth
from users
where temp_activated_at IS NOT NULL
group by week;
```

This query shows the monthly and weekly change in number of users.

OUTPUT:



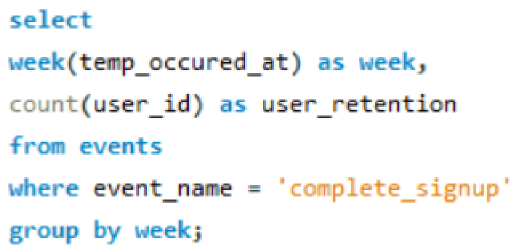
week	User_growth
0	NULL
1	32.1%
2	0.6%
3	-5.4%
4	6.9%
5	11.6%
6	-4.6%
7	-3.6%
8	-2.5%
9	7.4%



month	User_growth
1	NULL
2	-3.9%
3	10.5%
4	15.7%
5	8.7%
6	8.6%
7	15.2%
8	4.9%
9	-308.2%
10	15.4%
11	2.3%
12	17.9%

C. **Weekly Retention Analysis:**

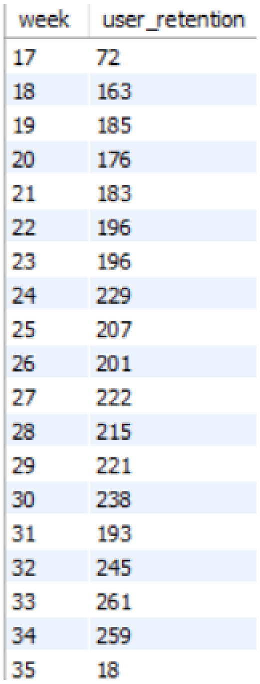
SQL Query:



```
select
week(temp_occured_at) as week,
count(user_id) as user_retention
from events
where event_name = 'complete_signup'
group by week;
```

This query return the number of users who are done complete signup

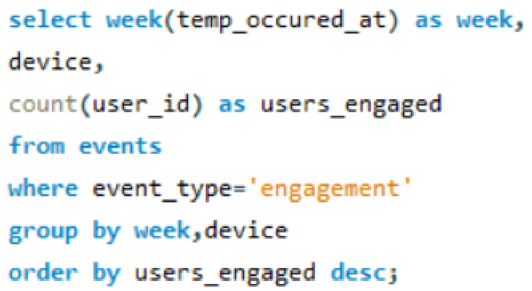
OUTPUT:



week	user_retention
17	72
18	163
19	185
20	176
21	183
22	196
23	196
24	229
25	207
26	201
27	222
28	215
29	221
30	238
31	193
32	245
33	261
34	259
35	18

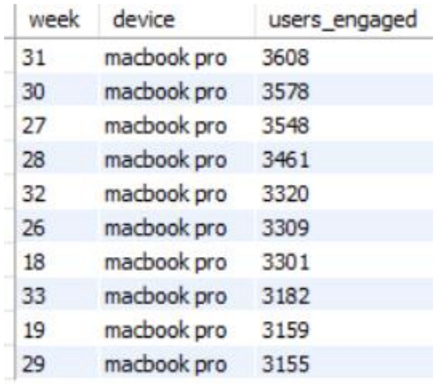
D. **Weekly Engagement Per Device:**

SQL Query:



```
select week(temp_occured_at) as week,
device,
count(user_id) as users_engaged
from events
where event_type='engagement'
group by week,device
order by users_engaged desc;
```

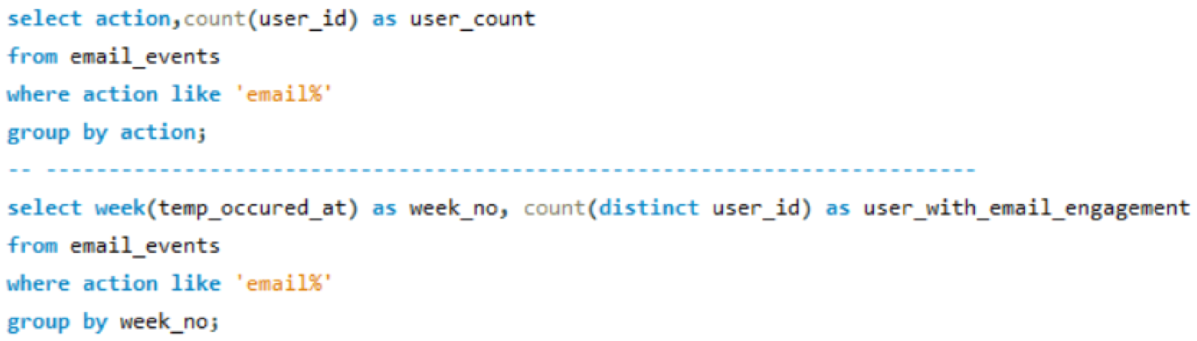
OUTPUT:



week	device	users_engaged
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

E. **Email Engagement Analysis:**

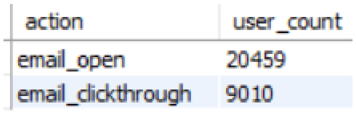
SQL Query:



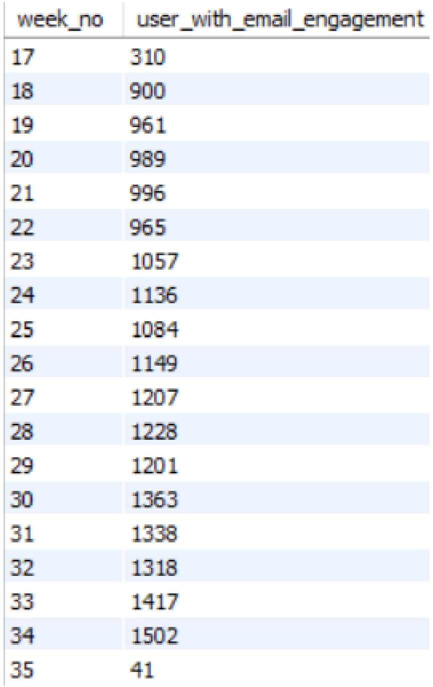
```
select action,count(user_id) as user_count
from email_events
where action like 'email%'
group by action;
-- ------------------------------------------------------------------------
select week(temp_occured_at) as week_no, count(distinct user_id) as user_with_email_engagement
from email_events
where action like 'email%'
group by week_no;
```

This query returns the number of users who are engaged in email related events.

OUTPUT:



action	user_count
email_open	20459
email_clickthrough	9010



week_no	user_with_email_engagement
17	310
18	900
19	961
20	989
21	996
22	965
23	1057
24	1136
25	1084
26	1149
27	1207
28	1228
29	1201
30	1363
31	1338
32	1318
33	1417
34	1502
35	41