Operation Analytics and Investigating Metric Spike

Project Description

In this data analytics project, we will embark on a two-part journey. The first part involves delving into operational analytics to optimize company processes. You will use advanced SQL skills to analyse job data, detect duplicates, and uncover insights to enhance decision-making.

In the second part, you will investigate metric spikes by examining user engagement, growth, retention, device-specific engagement, and email engagement. Your SQL queries and insights will empower the company to make data-driven decisions and refine its operations.

Tech-Stack Used

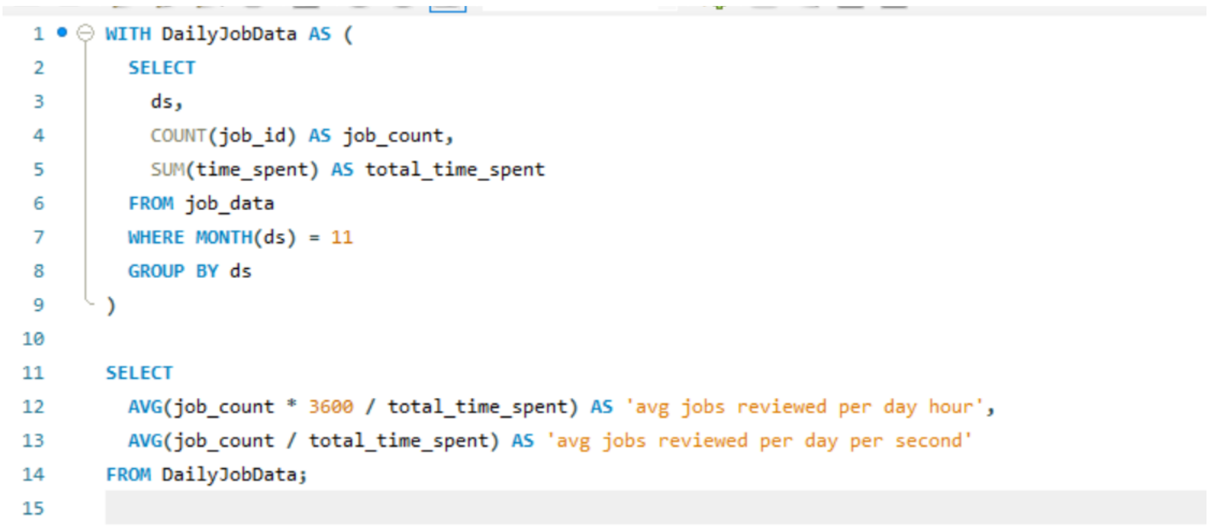
MySQL Workbench 8.0

TASK 1

Jobs Reviewed Over Time:

Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

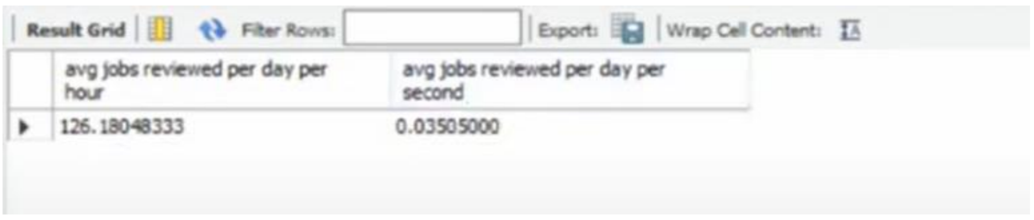
calculate the number of jobs reviewed per hour for each day in November 2020.



```
WITH DailyJobData AS (
  SELECT
    ds,
    COUNT(job_id) AS job_count,
    SUM(time_spent) AS total_time_spent
  FROM job_data
  WHERE MONTH(ds) = 11
  GROUP BY ds
)

SELECT
  AVG(job_count * 3600 / total_time_spent) AS 'avg jobs reviewed per day hour',
  AVG(job_count / total_time_spent) AS 'avg jobs reviewed per day per second'
FROM DailyJobData;
```

OUTPUT

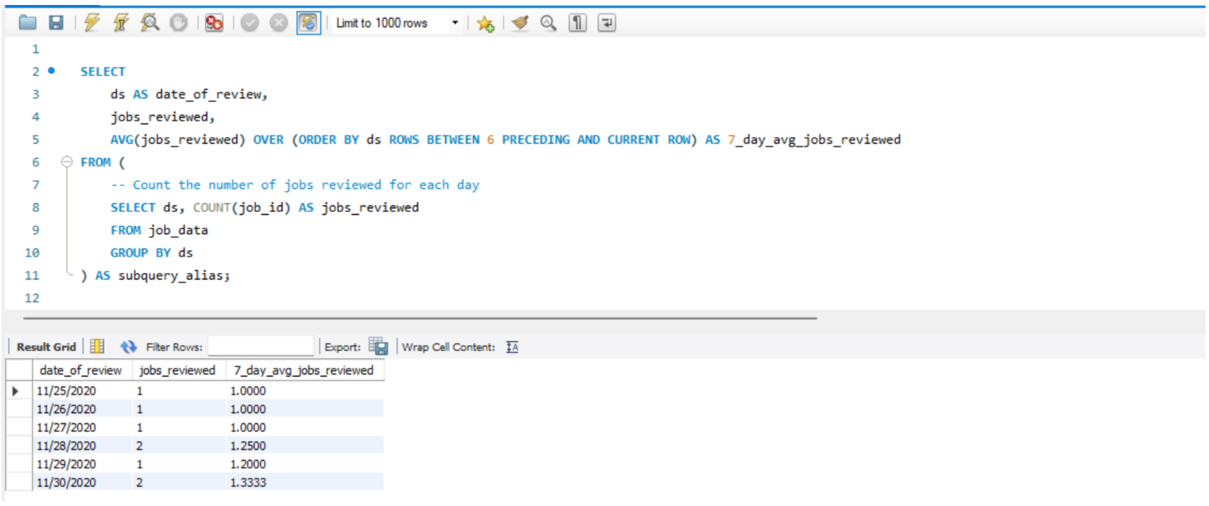


avg jobs reviewed per day per hour	avg jobs reviewed per day per second
126.18048333	0.03505000

Throughput Analysis

Objective: Calculate the 7-day rolling average of throughput (number of events per second).

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.



Limit to 1000 rows

```

SELECT
    ds AS date_of_review,
    jobs_reviewed,
    AVG(jobs_reviewed) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS 7_day_avg_jobs_reviewed
FROM (
    -- Count the number of jobs reviewed for each day
    SELECT ds, COUNT(job_id) AS jobs_reviewed
    FROM job_data
    GROUP BY ds
) AS subquery_alias;

```

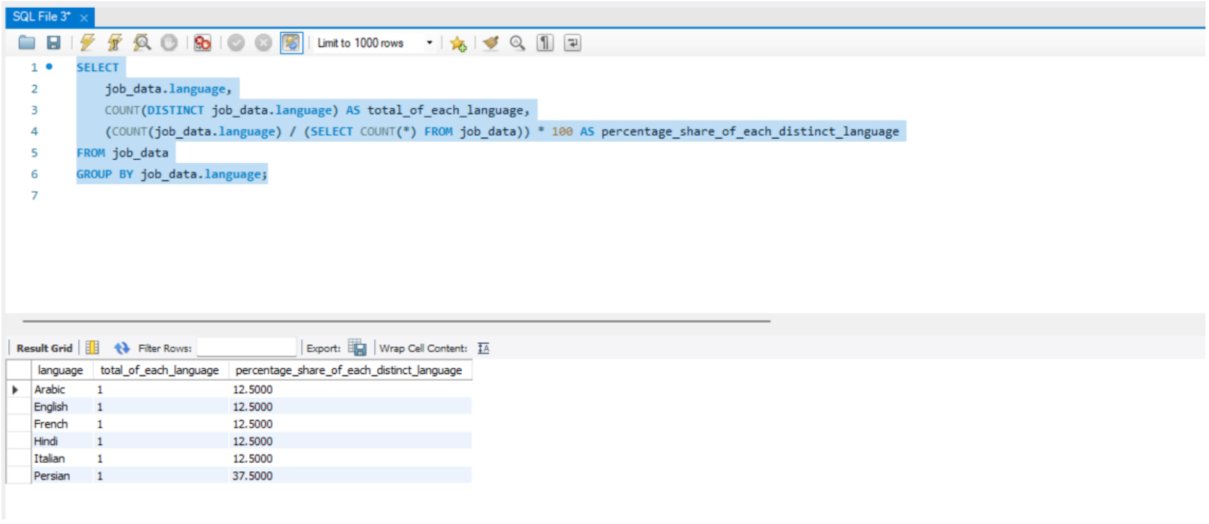
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

date_of_review	jobs_reviewed	7_day_avg_jobs_reviewed
11/25/2020	1	1.0000
11/26/2020	1	1.0000
11/27/2020	1	1.0000
11/28/2020	2	1.2500
11/29/2020	1	1.2000
11/30/2020	2	1.3333

Language share Analysis:

Objective: Calculate the percentage share of each language in the last 30 days.

calculate the percentage share of each language over the last 30 days.



SQL File 3*

Limit to 1000 rows

```
SELECT
    job_data.language,
    COUNT(DISTINCT job_data.language) AS total_of_each_language,
    (COUNT(job_data.language) / (SELECT COUNT(*) FROM job_data)) * 100 AS percentage_share_of_each_distinct_language
FROM job_data
GROUP BY job_data.language;

```

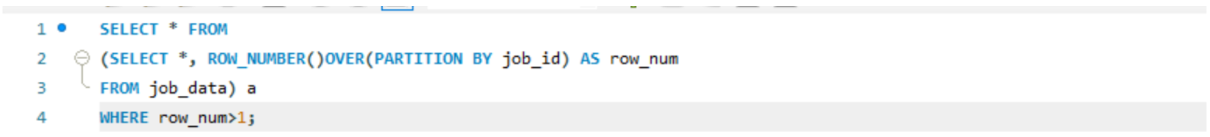
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

language	total_of_each_language	percentage_share_of_each_distinct_language
Arabic	1	12.5000
English	1	12.5000
French	1	12.5000
Hindi	1	12.5000
Italian	1	12.5000
Persian	1	37.5000

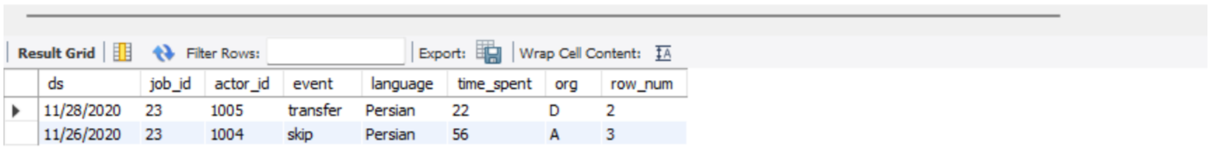
Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

display duplicate rows from the job_data table.



```sql
SELECT * FROM
(SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num
FROM job_data) a
WHERE row_num>1;
```



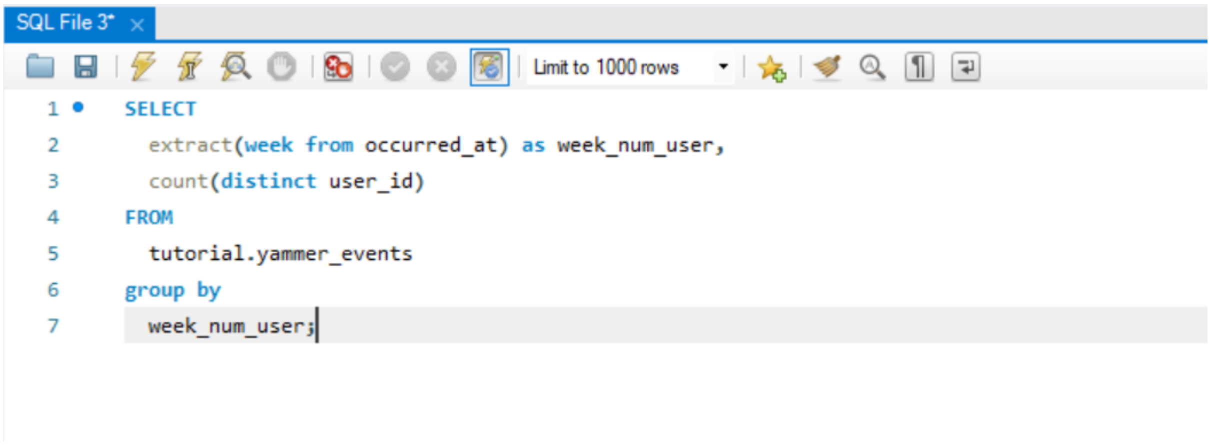
ds	job_id	actor_id	event	language	time_spent	org	row_num
11/28/2020	23	1005	transfer	Persian	22	D	2
11/26/2020	23	1004	skip	Persian	56	A	3

TASK 2

Weekly User Engagement:

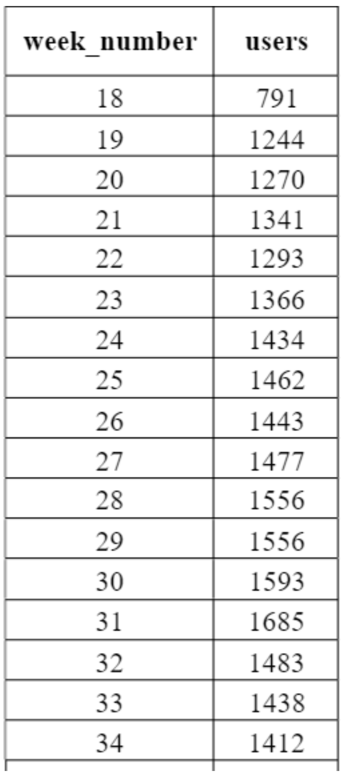
Objective: Measure the activeness of users on a weekly basis.

calculate the weekly user engagement.



```sql
SELECT
  extract(week from occurred_at) as week_num_user,
  count(distinct user_id)
FROM
  tutorial.yammer_events
group by
  week_num_user;
```

OUTPUT

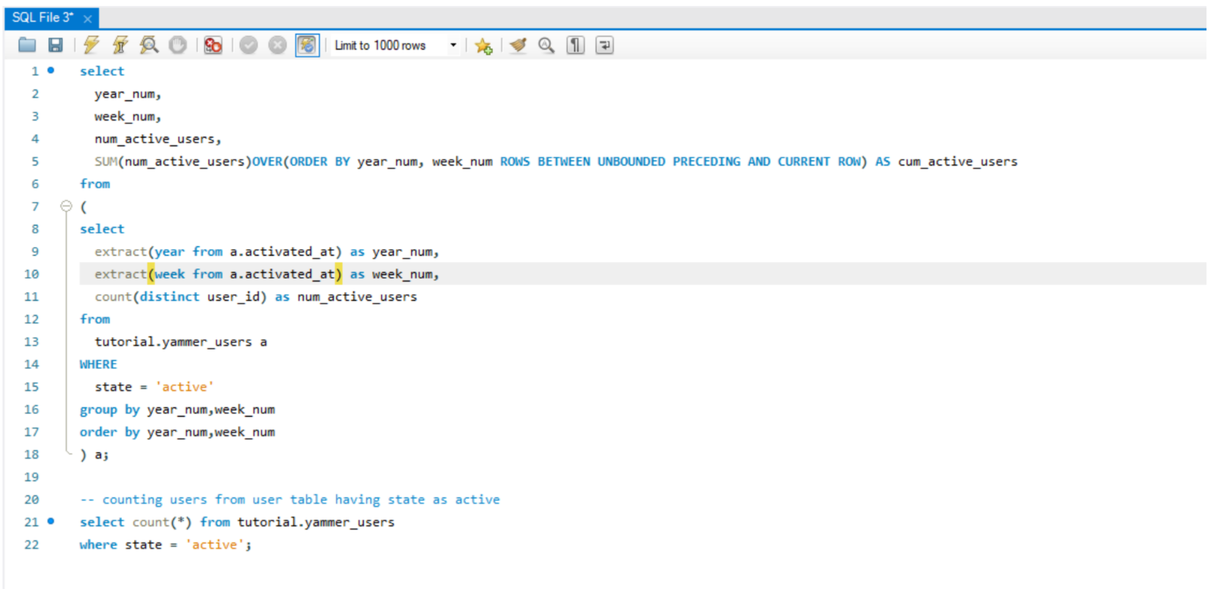


week_number	users
18	791
19	1244
20	1270
21	1341
22	1293
23	1366
24	1434
25	1462
26	1443
27	1477
28	1556
29	1556
30	1593
31	1685
32	1483
33	1438
34	1412

User Growth Analysis:

Objective: Analyze the growth of users over time for a product.

calculate the user growth for the product.



```sql
select
  year_num,
  week_num,
  num_active_users,
  SUM(num_active_users)OVER(ORDER BY year_num, week_num ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_users
from
(
select
  extract(year from a.activated_at) as year_num,
  extract(week from a.activated_at) as week_num,
  count(distinct user_id) as num_active_users
from
  tutorial.yammer_users a
WHERE
  state = 'active'
group by year_num,week_num
order by year_num,week_num
) a;

-- counting users from user table having state as active
select count(*) from tutorial.yammer_users
where state = 'active';
```

OUTPUT

https://drive.google.com/file/d/17sQgwD_vJCK2VauJnhKPdEk6rjph_123/view?usp=drive_link

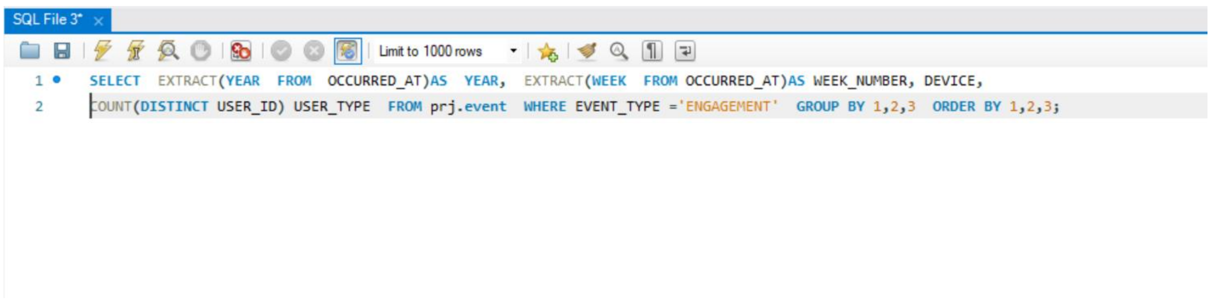
Weekly Retention Analysis:

Objective: Analyze the retention of users on a weekly basis after signing up for a product.

calculate the weekly retention of users based on their sign-up cohort.

OUTPUT

https://drive.google.com/file/d/15STeSVVmdkgoLIpC-hQdw34FV2_WaOJQ/view?usp=drive_link

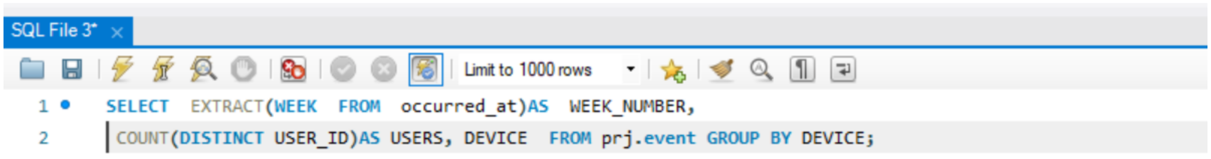


```sql
SELECT  EXTRACT(YEAR  FROM  OCCURRED_AT)AS  YEAR,  EXTRACT(WEEK  FROM OCCURRED_AT)AS WEEK_NUMBER, DEVICE,
COUNT(DISTINCT USER_ID) USER_TYPE  FROM prj.event  WHERE EVENT_TYPE ='ENGAGEMENT'  GROUP BY 1,2,3  ORDER BY 1,2,3;
```

Weekly Engagement Per Device:

Objective: Measure the activeness of users on a weekly basis per device.

calculate the weekly engagement per device.



```sql
SELECT  EXTRACT(WEEK  FROM  occurred_at)AS  WEEK_NUMBER,
COUNT(DISTINCT USER_ID)AS USERS, DEVICE  FROM prj.event GROUP BY DEVICE;
```

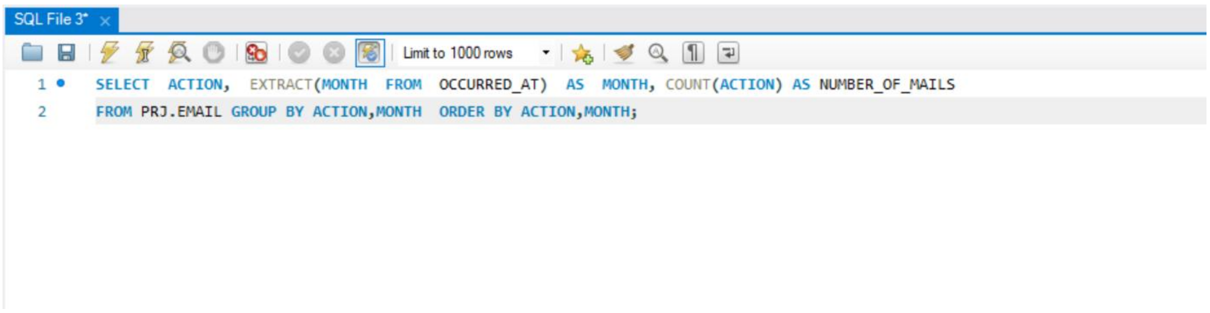
OUTPUT

https://drive.google.com/file/d/14VrXArk73h9jJPF3vmkXwtwoFEll4urK/view?usp=drive_link

Email Engagement Analysis:

Objective: Analyze how users are engaging with the email service.

calculate the email engagement metrics.



```sql
SELECT  ACTION,  EXTRACT(MONTH  FROM  OCCURRED_AT)  AS  MONTH, COUNT(ACTION) AS NUMBER_OF_MAILS
FROM PRJ.EMAIL GROUP BY ACTION,MONTH  ORDER BY ACTION,MONTH;
```

OUTPUT

email_opening_rate	email_clicking_rate
33.58338805	14.78988838