



OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

ABHISHEK JAGTAP

PROJECT DESCRIPTION

The project is focused on Operation Analytics and Investigating Metric Spike using advanced SQL. The project aims to analyze the complete end-to-end operations of company, with the help of which the company can then find the areas that need improvement. The data collected is used to predict the overall growth or decline of a company's fortune, resulting in better automation, better understanding between cross-functional teams, and more effective workflows

APPROACH

To execute the project, I firstly collected and analyzed the relevant data sets and tables provided by the company. I then applied advanced SQL queries to derive insights and answer the questions asked by different departments.

TECH-STACK USED

- My SQL workbench
- Mode analytics (Mode is a collaborative data platform that combines SQL, R, Python, and visual analytics in one place. Connect, analyze, and share, faster)

INSIGHTS

In this project, I gained insights into the importance of operation analytics in predicting a company's growth or decline. I also learned how to use advanced SQL queries to extract insights from large data sets and how to use data visualization techniques to better understand the data.

RESULT

As a result of this project, I was able to provide a detailed report for the two operations mentioned in the project description. I was able to calculate the number of jobs reviewed per hour per day for November 2020, the 7-day rolling average of throughput, the percentage share of each language in the last 30 days, and how to display duplicate rows from the table. Additionally, I was able to calculate the weekly user engagement, user growth for a product, weekly retention of users-sign up cohort, and weekly engagement of users.

CASE STUDY 1

Job Data Analysis

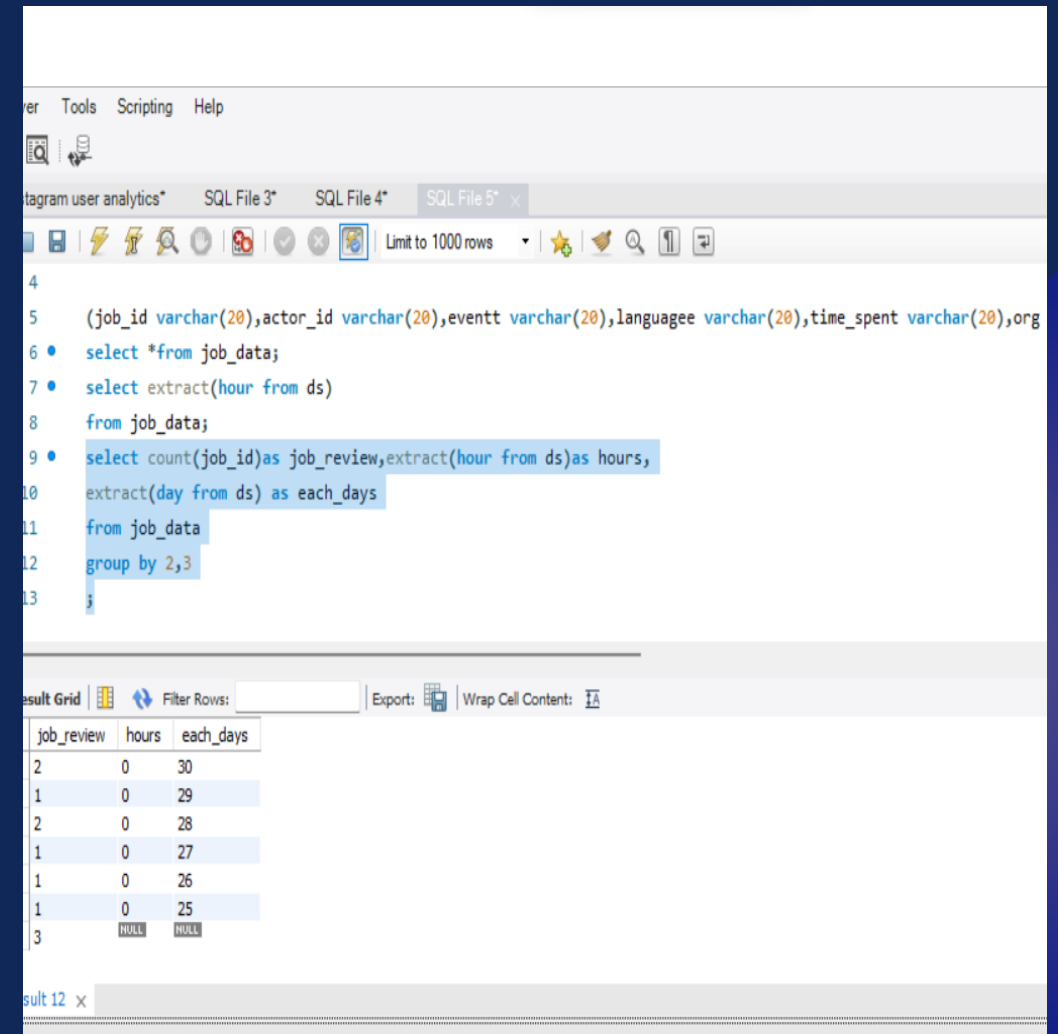
TASK 1

A. Jobs Reviewed Over Time:

- Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.
- Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

Query

```
select
count(job_id)as job_review,
extract(hour from ds)as hours
,extract(day from ds) as each_days
from job_data
group by 2,3;
```



The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

```
select
count(job_id)as job_review,
extract(hour from ds)as hours
,extract(day from ds) as each_days
from job_data
group by 2,3;
```

The results grid displays the output of the query, showing the number of jobs reviewed per hour for each day in November 2020. The columns are labeled 'job_review', 'hours', and 'each_days'. The results are as follows:

job_review	hours	each_days
2	0	30
1	0	29
2	0	28
1	0	27
1	0	26
1	0	25
3	NULL	NULL

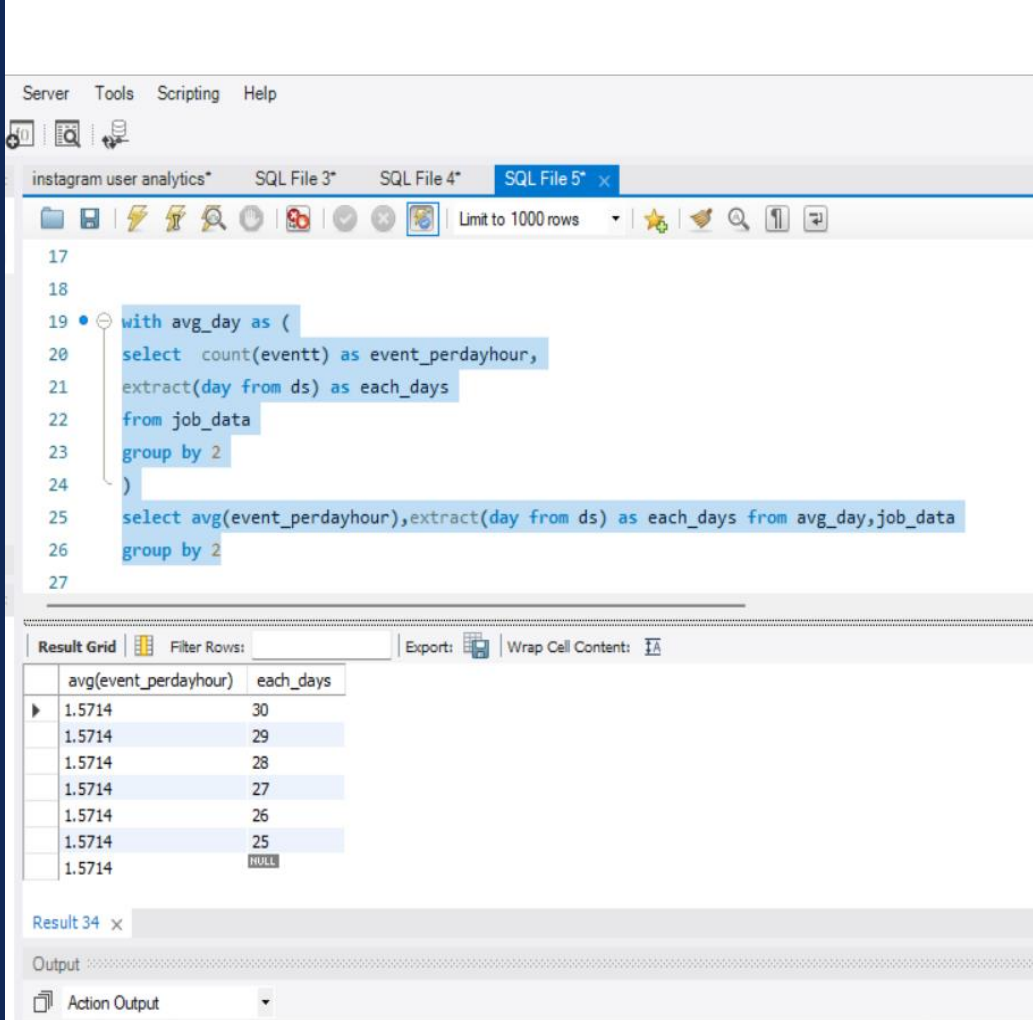
TASK 2

B. Throughput Analysis:

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

Query

```
with avg_day as
( select
count(eventtt) as event_perdayhour,
extract(day from ds) as each_days
from job_data
group by 2 )
select avg(event_perdayhour),
extract(day from ds) as each_days
from avg_day,job_data
group by 2;
```



The screenshot shows a SQL IDE interface with a menu bar (Server, Tools, Scripting, Help) and a toolbar. The main window displays a SQL query in a file named 'SQL File 5*'. The query is as follows:

```
17
18
19 with avg_day as (
20   select count(eventtt) as event_perdayhour,
21         extract(day from ds) as each_days
22   from job_data
23   group by 2
24 )
25 select avg(event_perdayhour),extract(day from ds) as each_days from avg_day,job_data
26 group by 2
27
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The grid has two columns: 'avg(event_perdayhour)' and 'each_days'. The results are as follows:

avg(event_perdayhour)	each_days
1.5714	30
1.5714	29
1.5714	28
1.5714	27
1.5714	26
1.5714	25
1.5714	NULL

The IDE also shows a 'Filter Rows' section, an 'Export' button, and a 'Wrap Cell Content' checkbox. The 'Result 34' tab is active, and the 'Output' and 'Action Output' sections are visible at the bottom.

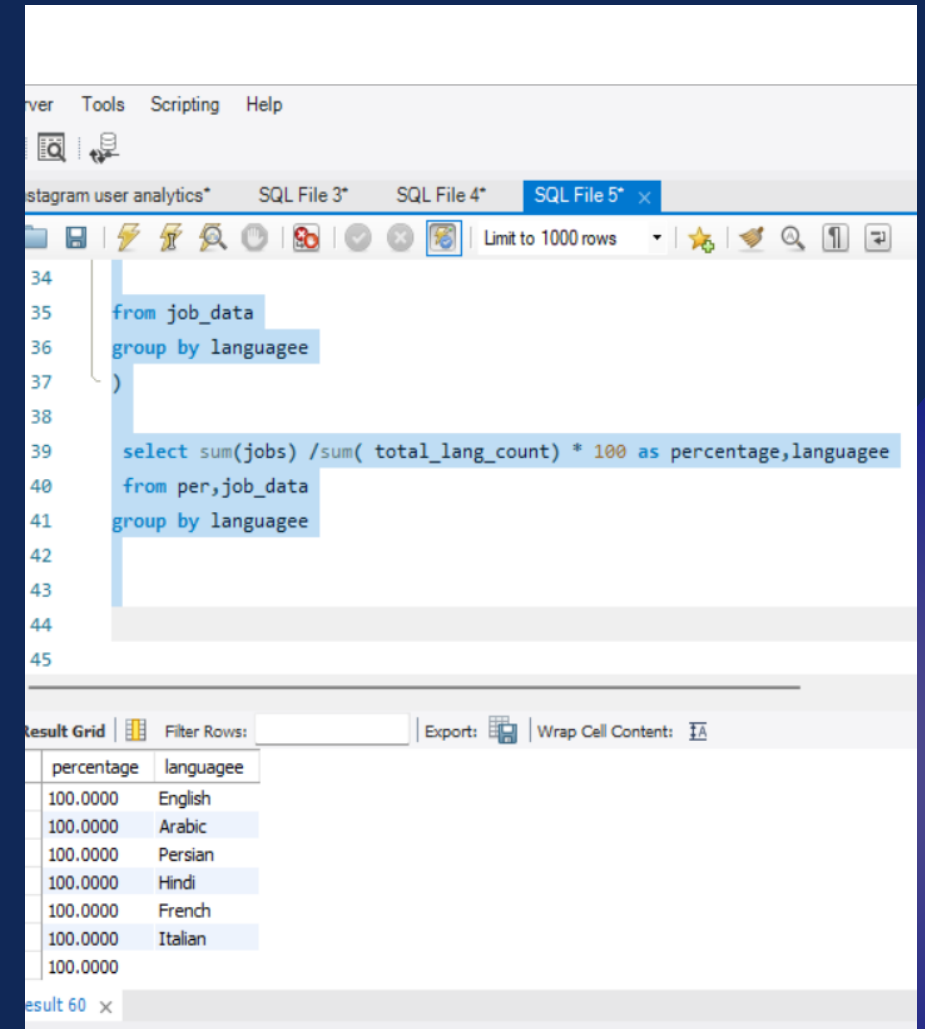
TASK 3

C. Language Share Analysis:

- Objective: Calculate the percentage share of each language in the last 30 days.
- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

Query

```
with per as
(select
count(languagee) as total_lang_count,
count( job_id)as jobs
from job_data
group by language )
select
sum(jobs) /sum( total_lang_count) * 100 as
percentage,language
from per,job_data
group by languagee;
```



TASK 4

D. Duplicate Rows Detection:

- Objective: Identify duplicate rows in the data.
- Your Task: Write an SQL query to display duplicate rows from the job_data table.

Query

```
select *,
count(*)
from job_data
group by
job_id,actor_id,eventt,languagee,tim_spent,
org,ds
having
count(*)>1;
```

The screenshot shows a SQL IDE interface with two queries and their results. The first query is a SELECT statement that filters for duplicate job_ids. The second query is a SELECT statement that displays all columns for rows with duplicate primary keys. The results are shown in a grid format.

```
57
58
59 • select job_id,count(job_id)
60 from job_data
61 group by job_id
62 having count(job_id)>1;
63
64 • select *,count(*)
65 from job_data
66 group by job_id,actor_id,eventt,languagee,time_spent,org,ds
67 having count(*)>1
68
```

job_id	actor_id	eventt	languagee	time_spent	org	ds	count(*)
							2

CASE STUDY 2

Investigating Metric Spike

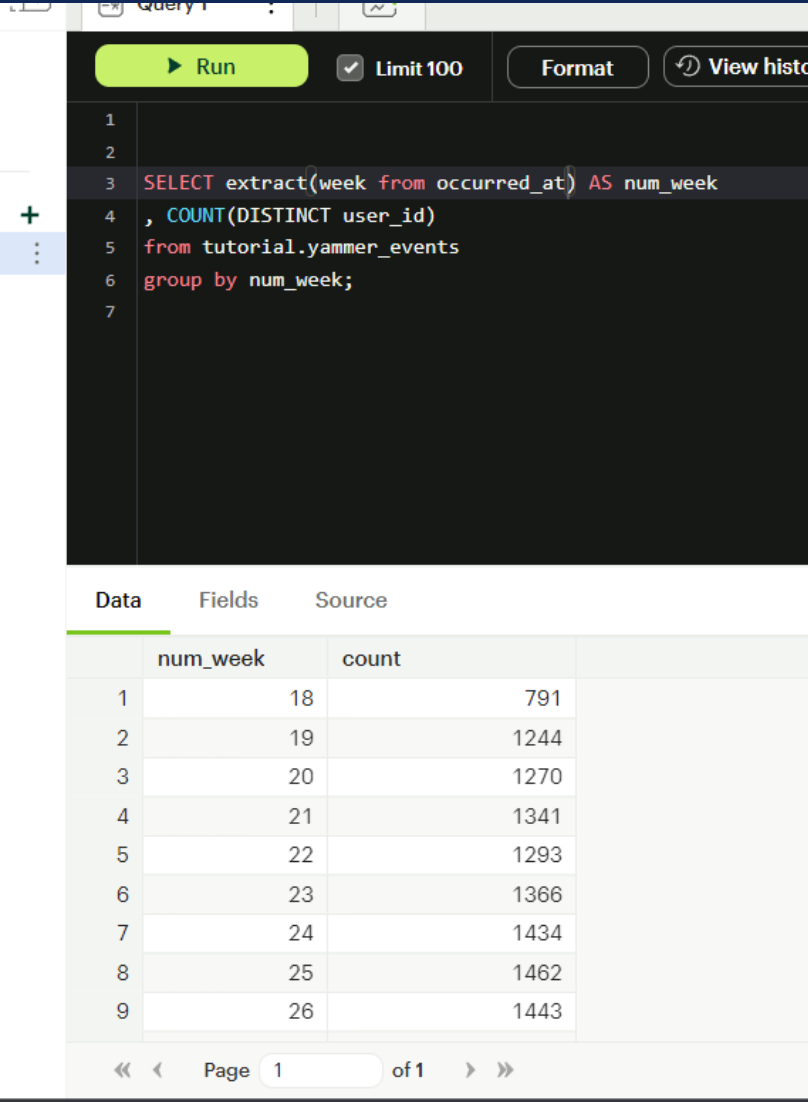
TASK 1

A. Weekly User Engagement:

- Objective: Measure the activeness of users on a weekly basis.
- Your Task: Write an SQL query to calculate the weekly user engagement.

Query

```
SELECT
    extract(week from occurred_at) AS num_week,
    COUNT(DISTINCT user_id)
from tutorial.yammer_events
group by num_week;
```



The screenshot shows a SQL query editor interface. At the top, there are buttons for 'Run' (a green button with a play icon), 'Limit 100' (with a checkmark), 'Format', and 'View history'. Below these is a text area containing the SQL query. The query is as follows:

```
1
2
3 SELECT extract(week from occurred_at) AS num_week
4 , COUNT(DISTINCT user_id)
5 from tutorial.yammer_events
6 group by num_week;
7
```

Below the query editor is a table with the results. The table has three columns: 'Data', 'Fields', and 'Source'. The 'Data' column contains a list of numbers from 1 to 9. The 'Fields' column contains the text 'num_week'. The 'Source' column contains the text 'count'. The table is as follows:

Data	Fields	Source
	num_week	count
1	18	791
2	19	1244
3	20	1270
4	21	1341
5	22	1293
6	23	1366
7	24	1434
8	25	1462
9	26	1443

At the bottom of the table, there is a pagination bar that says 'Page 1 of 1' with navigation arrows.

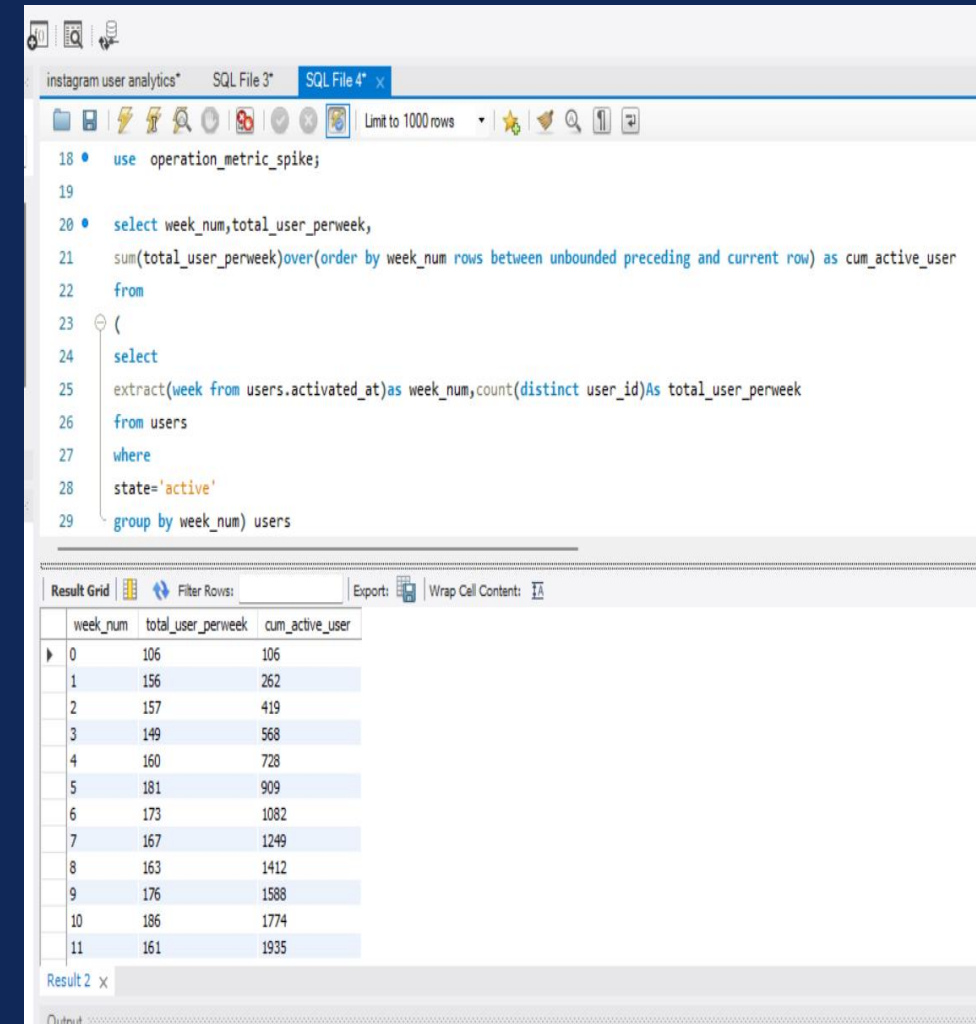
TASK 2

B. User Growth Analysis:

- Objective: Analyze the growth of users over time for a product.
- Your Task: Write an SQL query to calculate the user growth for the product.

Query

```
select
week_num,total_user_perweek,sum(total_user_perweek)
over(order by week_num rows between unbounded preceding and current row) as cum_active_user
from
(select
extract(week from users.activated_at)as week_num
,count(distinct user_id)As total_user_perweek
from users
where state='active'
group by week_num) users
```



The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

```
18 use operation_metric_spike;
19
20 select week_num,total_user_perweek,
21 sum(total_user_perweek)over(order by week_num rows between unbounded preceding and current row) as cum_active_user
22 from
23 (
24 select
25 extract(week from users.activated_at)as week_num,count(distinct user_id)As total_user_perweek
26 from users
27 where
28 state='active'
29 group by week_num) users
```

The results grid displays the following data:

week_num	total_user_perweek	cum_active_user
0	106	106
1	156	262
2	157	419
3	149	568
4	160	728
5	181	909
6	173	1082
7	167	1249
8	163	1412
9	176	1588
10	186	1774
11	161	1935

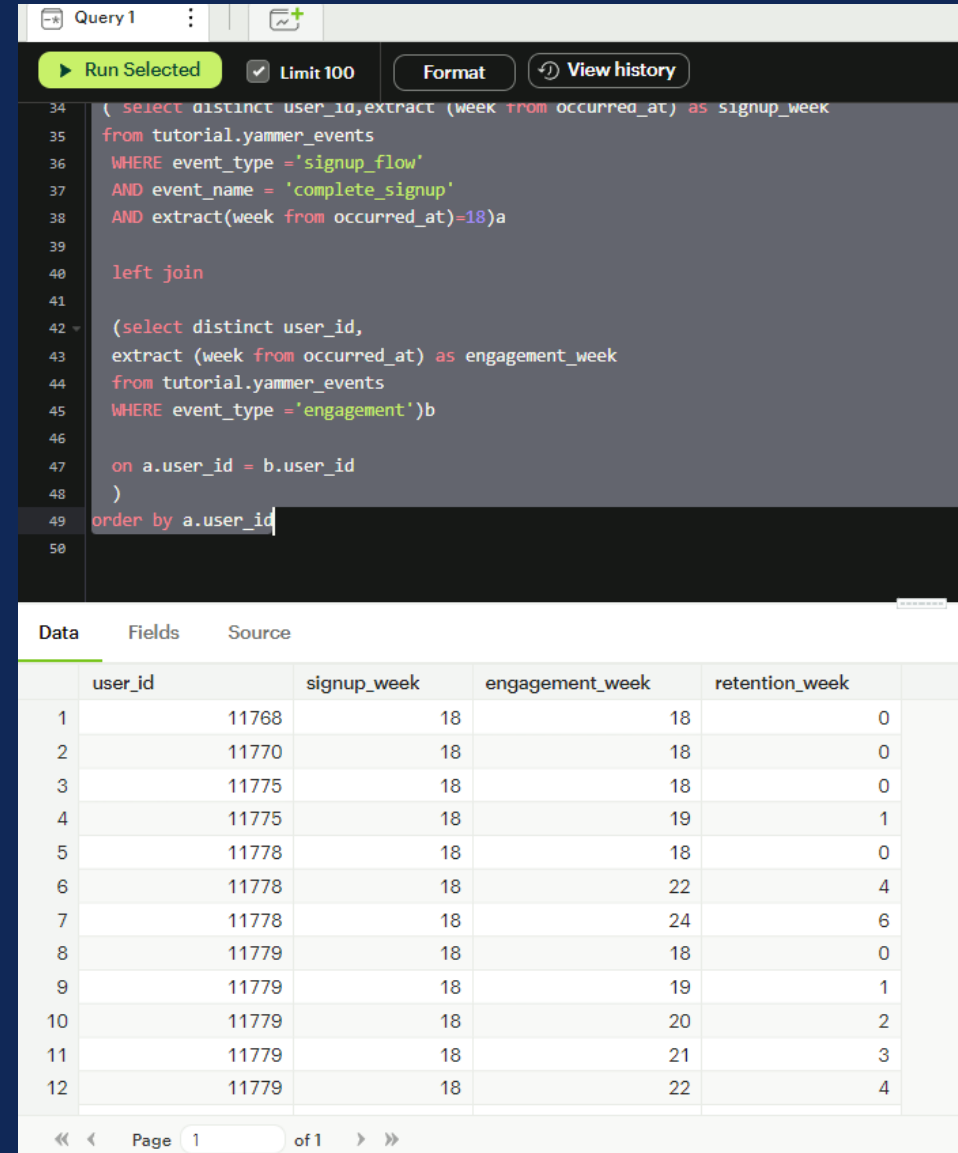
TASK 3

C. Weekly Retention Analysis:

- Objective: Analyze the retention of users on a weekly basis after signing up for a product.
- Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Query

```
SELECT a.user_id, a.signup_week, b.engagement_week,  
b.engagement_week-a.signup_week as retention_week  
  
FROM  
  
( ( select distinct user_id,extract (week from occurred_at) as  
signup_week  
  
from tutorial.yammer_events  
  
WHERE event_type ='signup_flow' AND event_name =  
'complete_signup' AND extract(week from  
occurred_at)=18)a  
  
left join  
  
(select distinct user_id,  
extract (week from occurred_at) as engagement_week  
from tutorial.yammer_events  
WHERE event_type ='engagement')b  
  
on a.user_id = b.user_id )  
  
order by a.user_id
```



Query 1

Run Selected Limit 100 Format View history

```
( select distinct user_id,extract (week from occurred_at) as signup_week  
from tutorial.yammer_events  
WHERE event_type ='signup_flow'  
AND event_name = 'complete_signup'  
AND extract(week from occurred_at)=18)a  
  
left join  
  
(select distinct user_id,  
extract (week from occurred_at) as engagement_week  
from tutorial.yammer_events  
WHERE event_type ='engagement')b  
  
on a.user_id = b.user_id  
)  
order by a.user_id
```

Data	Fields	Source
	user_id	signup_week engagement_week retention_week
1	11768	18 18 0
2	11770	18 18 0
3	11775	18 18 0
4	11775	18 19 1
5	11778	18 18 0
6	11778	18 22 4
7	11778	18 24 6
8	11779	18 18 0
9	11779	18 19 1
10	11779	18 20 2
11	11779	18 21 3
12	11779	18 22 4

Page 1 of 1

TASK 4

D. Weekly Engagement Per Device:

- Objective: Measure the activeness of users on a weekly basis per device.
- Your Task: Write an SQL query to calculate the weekly engagement per device.

Query

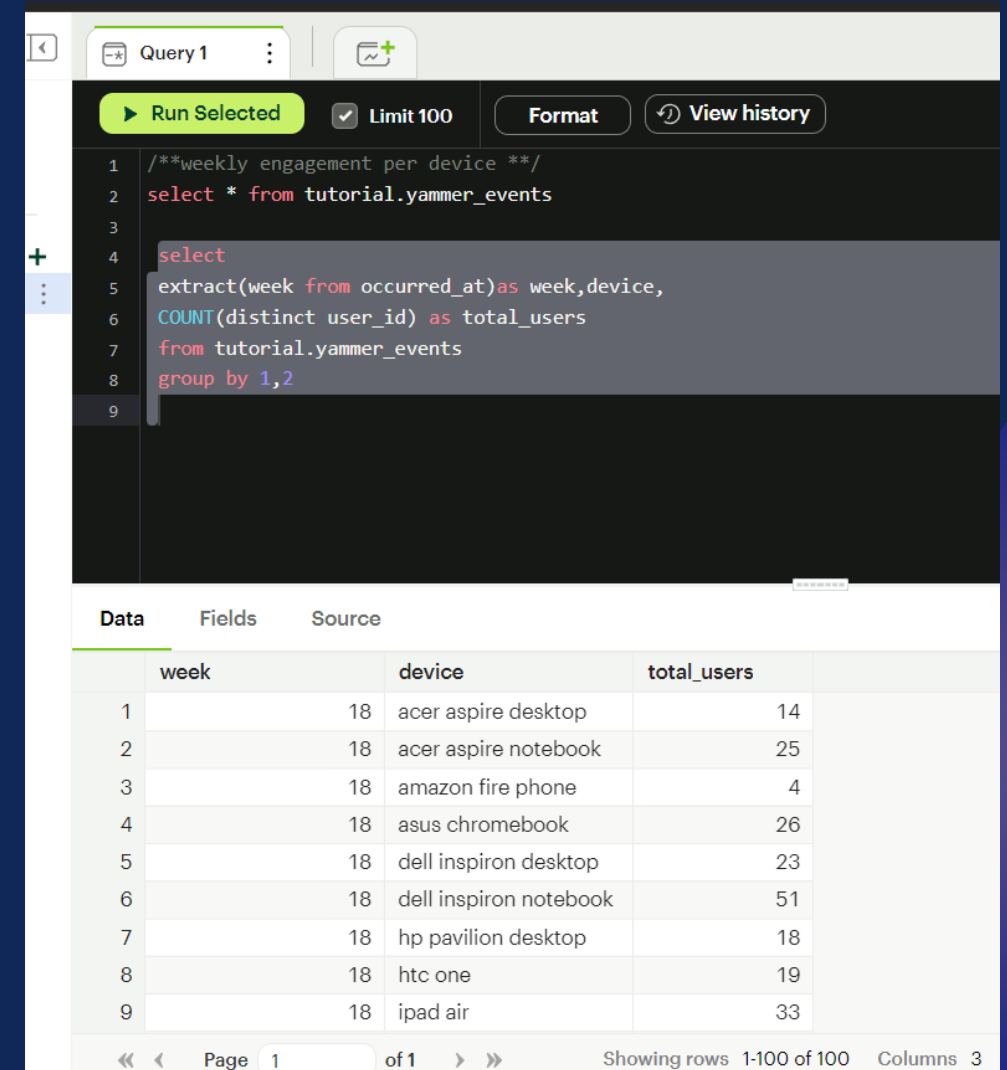
Select

Extract(week from occurred_at) as week,
device,

count (distinct user_id) as total_users

From tutorial.yammer.events

Group 1,2;



The screenshot shows a SQL query editor interface. At the top, there's a tab labeled 'Query 1'. Below the tab are buttons: 'Run Selected' (highlighted in green), 'Limit 100' (checked), 'Format', and 'View history'. The query editor contains the following SQL code:

```
1  /**weekly engagement per device **/  
2  select * from tutorial.yammer_events  
3  
4  select  
5  extract(week from occurred_at) as week, device,  
6  COUNT(distinct user_id) as total_users  
7  from tutorial.yammer_events  
8  group by 1,2  
9
```

Below the query editor is a table with the following data:

	Data	Fields	Source
	week	device	total_users
1	18	acer aspire desktop	14
2	18	acer aspire notebook	25
3	18	amazon fire phone	4
4	18	asus chromebook	26
5	18	dell inspiron desktop	23
6	18	dell inspiron notebook	51
7	18	hp pavilion desktop	18
8	18	htc one	19
9	18	ipad air	33

At the bottom of the table, there's a pagination bar showing 'Page 1 of 1' and 'Showing rows 1-100 of 100 Columns 3'.

TASK 5

E. Email Engagement Analysis:

- Objective: Analyze how users are engaging with the email service.
- Your Task: Write an SQL query to calculate the email engagement metrics.

Query

SELECT

100.0 *SUM (case when email_category = 'email_opened' then 1 else 0 END)/SUM(case when email_category = 'email_sent' then 1 else 0 END)as email_openedrate,

100.0 *SUM(case when email_category = 'email_clicked' then 1 else 0 END)/SUM(case when email_category = 'email_sent' then 1 else 0 END) as email_clickedrate

FROM

(SELECT *,

case WHEN action in ('sent_reengagement_email','sent_weekly_digest') then 'email_sent' when action in ('email_open') then 'email_opened'

when action in('email_clickthrough') then 'email_clicked' end as email_category

from tutorial.yammer_emails)a

itled Report

Query 1

Run Selected Limit 100 Format View history

```
2 /**select * from tutorial.yammer_emails
3
4 select distinct action from tutorial.yammer_emails**/
5
6 SELECT
7 100.0 *SUM (case when email_category = 'email_opened' then 1 else 0 END)/SUM(case when email_category = 'email_sent' then 1 else 0 END)as email_openedrate,
8 100.0 *SUM(case when email_category = 'email_clicked' then 1 else 0 END)/SUM(case when email_category = 'email_sent' then 1 else 0 END) as email_clickedrate
9 FROM
10 (
11 SELECT *,
12 case
13 WHEN action in ('sent_reengagement_email','sent_weekly_digest')
14 then 'email_sent'
15 when action in ('email_open')
16 then 'email_opened'
17 when action in('email_clickthrough')
18 then 'email_clicked'
19 end as email_category
20 from tutorial.yammer_emails
21 )a
22
```

Data Fields Source

	email_openedrate	email_clickedrate
1	33.5834	14.7899

Export



THANK YOU