

# **PROJECT 2: Operation Analytics and Investigating Metric Spike**

Advanced SQL

NAME: K. Yaswanth

GMAIL: [kunayaswanth123@gmail.com](mailto:kunayaswanth123@gmail.com)

To find best reports for the below two case studies operation analytics and investigating metric spike.

SUBMITTING TO TRAINITY :)

SPECIAL THANKS TO TRAINITY TEAM

## Project Description:

### CASE STUDY 1:

The first case study involves analysing job\_data to improve operational efficiency. The project includes finding various metrics such as throughput, productivity and providing recommendations for users to improving overall efficiency.

### CASE STUDY 2:

The second case study involves analysing the data to identify patterns and trends such as User growth, user engagement, retention, email metrics

## APPROACH: -

First, the data prepared and analysed using SQL. Various SQL functions like SELECT, WHERE, GROUP BY, JOIN, etc. are used to extract meaningful information from the dataset. Using proper window functions like over() at the required part to get better result.

## TECH STACK USED:

MySQL, mode.com (used the datasets yammer.events, yammer.email, yammer.users)

*(Taken the help of power labs for case study 2 in retention problem and cohort analysis)*

## Case Study 1 :- Operation Analytics :-

**1. Number of jobs reviewed:** Amount of jobs reviewed over time.

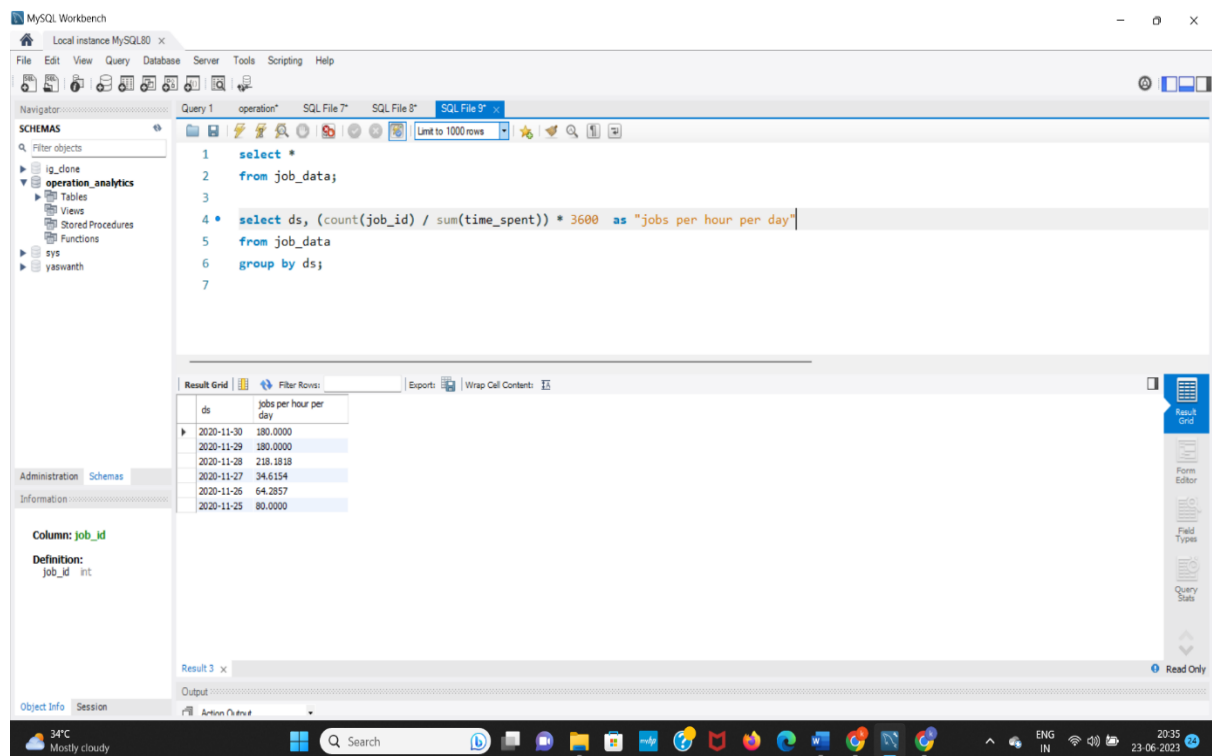
**Your task:** Calculate the number of jobs reviewed per hour per day for November 2020?

First We create the table job\_data

```
create table job_data(  
job_id int,  
actors_id int,  
event varchar(255),  
language varchar(255),  
time_spent int,  
org varchar(255),  
ds date);
```

```
INSERT INTO job_data  
(ds, job_id, actors_id, event, language, time_spent, org)  
VALUES  
( '2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),  
( '2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),  
( '2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),  
( '2020-11-28', 23, 1005, 'transfer', 'Persian', 22, 'D'),  
( '2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),  
( '2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),  
( '2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),  
( '2020-11-25', 20, 1004, 'transfer', 'Italian', 45, 'C');
```

Ans:-



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 select *  
2 from job_data;  
3  
4 select ds, (count(job_id) / sum(time_spent)) * 3600 as "jobs per hour per day"  
5 from job_data  
6 group by ds;  
7
```

The Results grid displays the output of the query:

ds	jobs per hour per day
2020-11-30	180.0000
2020-11-29	180.0000
2020-11-28	218.1818
2020-11-27	34.6154
2020-11-26	64.2857
2020-11-25	80.0000

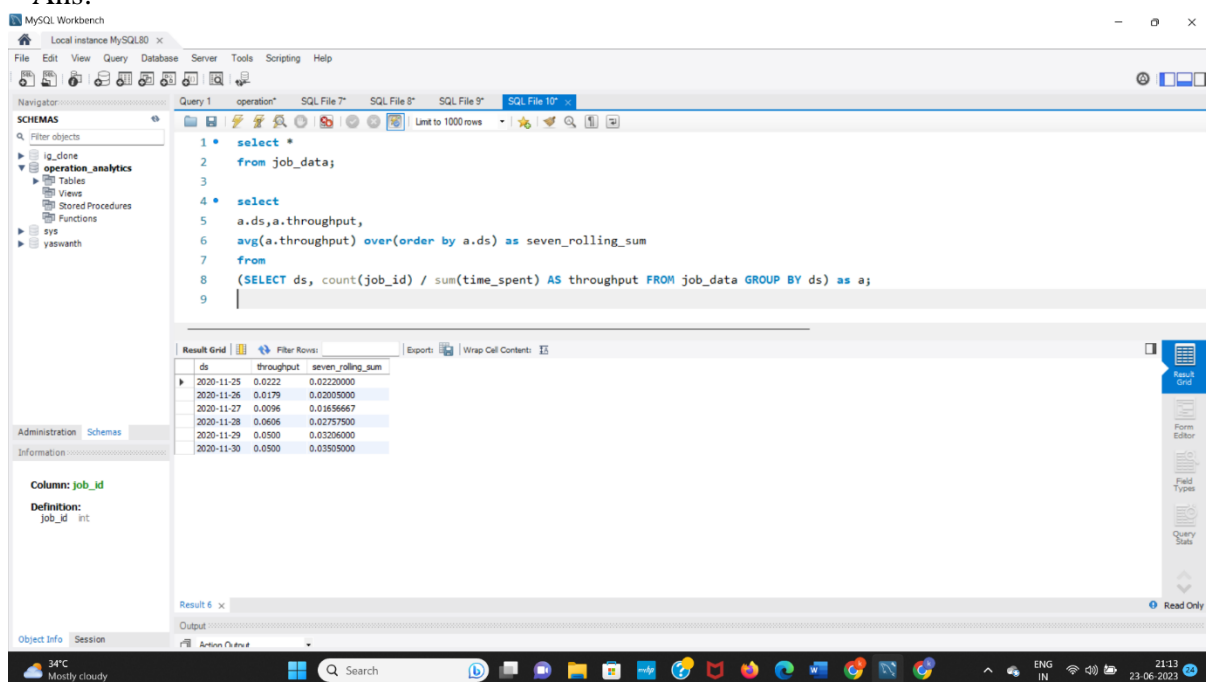
The left sidebar shows the Schemas pane with a tree view of the database structure. The bottom status bar shows the system clock as 20:35 on 23-06-2023.

Explanation :- To calculate no of jobs we use count(job\_id), for that we calculate need hour from seconds so we multiply by 3600 , we group it with date because they asked per day in question.

**2. Throughput:** It is the no. of events happening per second.

**Your task:** Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

Ans:-



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 * select *
2 from job_data;
3
4 * select
5 a.ds,a.throughput,
6 avg(a.throughput) over(order by a.ds) as seven_rolling_sum
7 from
8 (SELECT ds, count(job_id) / sum(time_spent) AS throughput FROM job_data GROUP BY ds) as a;
9
```

The Result Grid shows the following data:

ds	throughput	seven_rolling_sum
2020-11-25	0.0222	0.022200000
2020-11-26	0.0179	0.020950000
2020-11-27	0.0096	0.016966667
2020-11-28	0.0606	0.027575000
2020-11-29	0.0500	0.032060000
2020-11-30	0.0500	0.035050000

The Object Info panel shows the column job\_id with the definition job\_id int.

Explanation:

Daily throughput vs 7 day rolling throughput

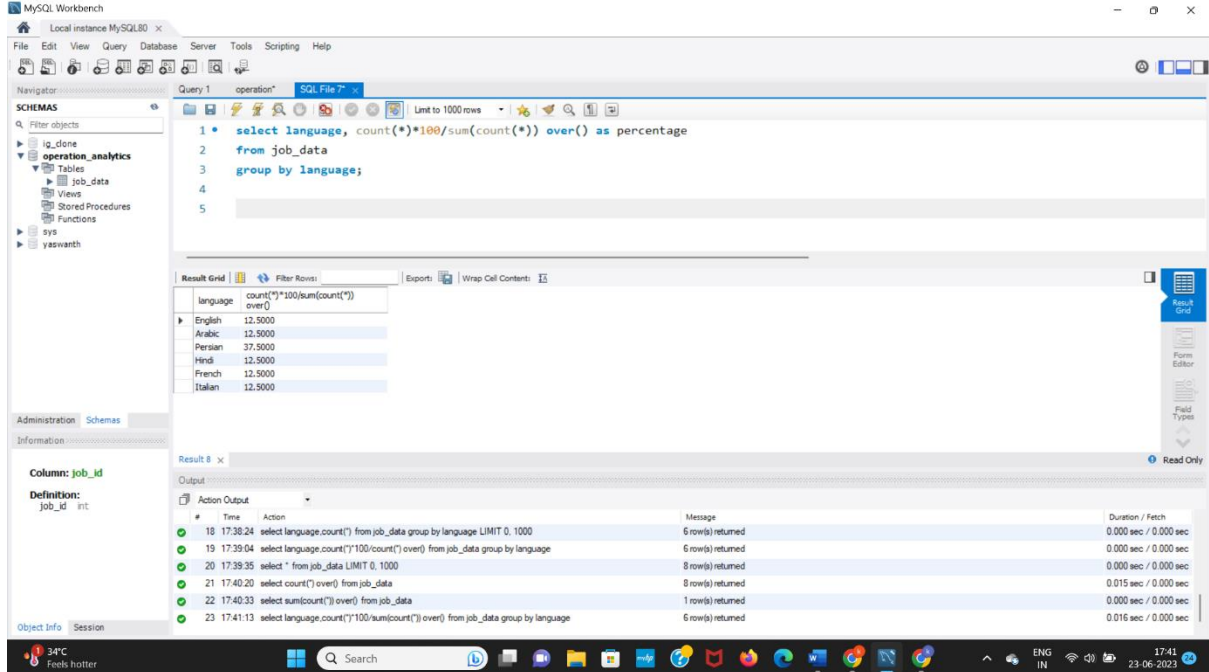
To find daily throughput: it is no of events per sec. So, we count(job\_id) to the total seconds group by date. Then, we get daily throughput.

Then seven day rolling throughput :-

We use over() window function in that order by then we find cumulative average in step which is rolling average

**3. Percentage share of each language:** Share of each language for different contents.  
**Your task:** Calculate the percentage share of each language in the last 30 days?

Ans:-



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 select language, count(*)*100/sum(count(*) over() as percentage
2 from job_data
3 group by language;
4
5
```

The Result Grid shows the following data:

language	count(*)*100/sum(count(*) over()
English	12.5000
Arabic	12.5000
Persian	37.5000
Hindi	12.5000
French	12.5000
Italian	12.5000

The bottom panel shows the execution log with the following output:

#	Time	Action	Message	Duration / Fetch
18	17:38:24	select language.count(*) from job_data group by language LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
19	17:39:04	select language.count(*)*100/sum(count(*) over() from job_data group by language	6 row(s) returned	0.000 sec / 0.000 sec
20	17:39:35	select * from job_data LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
21	17:40:20	select count(*) over() from job_data	8 row(s) returned	0.015 sec / 0.000 sec
22	17:40:33	select sum(count(*) over() from job_data	1 row(s) returned	0.000 sec / 0.000 sec
23	17:41:13	select language.count(*)*100/sum(count(*) over() from job_data group by language	6 row(s) returned	0.016 sec / 0.000 sec

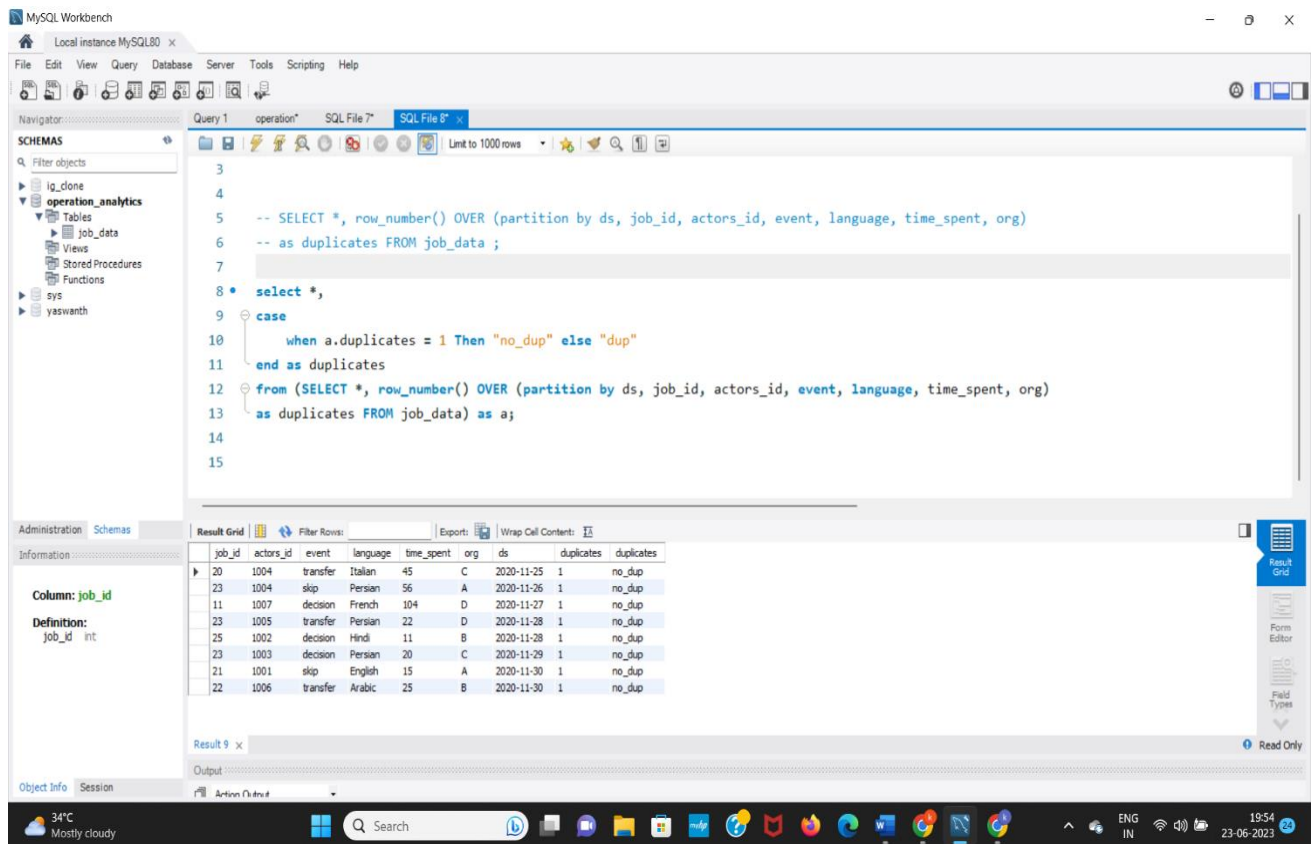
Explanation :-

Percentage share of each language is multiply by 100. Total contents is 8, we find it by using over() window function(total inputs = total outputs) then group by each language and we aggregate the language share.

**4. Duplicate rows:** Rows that have the same value present in them.

**Your task:** Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

Ans:



```
3
4
5  -- SELECT *, row_number() OVER (partition by ds, job_id, actors_id, event, language, time_spent, org)
6  -- as duplicates FROM job_data ;
7
8  * select *,
9
10     case
11     when a.duplicates = 1 Then "no_dup" else "dup"
12     end as duplicates
13   from (SELECT *, row_number() OVER (partition by ds, job_id, actors_id, event, language, time_spent, org)
14         as duplicates FROM job_data) as a;
15
```

job_id	actors_id	event	language	time_spent	org	ds	duplicates	duplicates
20	1004	transfer	Italian	45	C	2020-11-25	1	no_dup
23	1004	skip	Persian	56	A	2020-11-26	1	no_dup
11	1007	decision	French	104	D	2020-11-27	1	no_dup
23	1005	transfer	Persian	22	D	2020-11-28	1	no_dup
25	1002	decision	Hindi	11	B	2020-11-28	1	no_dup
23	1003	decision	Persian	20	C	2020-11-29	1	no_dup
21	1001	skip	English	15	A	2020-11-30	1	no_dup
22	1006	transfer	Arabic	25	B	2020-11-30	1	no_dup

Explanation:-

1. We use row\_number() with over() window function to assign same row numbers and in that part with date, job\_id, actors\_id, event, language, time\_spent, org because we are in need of finding duplicates.
2. When we get duplicates greater than 2 then it is dup
3. When we get duplicates = 1 then there is no dup (no duplicates)
4. Then we select duplicates finding > 1

## CASE STUDY 2: INVESTIGATING METRIC SPIKE

1. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.

**Your task:** Calculate the weekly user engagement?

Ans:

The screenshot shows a web application interface for a data warehouse. The top navigation bar includes the user's name 'kuna yaswanth' and the report title 'Untitled Report'. The main content area displays a SQL query in a dark-themed editor, which has been executed successfully. The query is as follows:

```
select
  extract(week from a.occurred_at) as wnum,
  count(distinct a.user_id)
from tutorial.yammer_events a
group by wnum;
```

Below the query editor, a table of results is displayed with two columns: 'wnum' and 'count'. The table contains 15 rows of data, showing the count of distinct users for each week. The status bar at the bottom indicates 'Showing rows 1-16 of 18', 'Columns 2', 'Size 288B', 'Run a few seconds ago', and 'Returned in 670ms'.

	wnum	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
10	27	1477
11	28	1556
12	29	1556
13	30	1593
14	31	1685
15	32	1483

Explanation: - To find weekly user engagement.

Weekly – so group by week, Engagement – use events table

Aggregating by counting different users.

## 2. User Growth: Amount of users growing over time for a product.

**Your task:** Calculate the user growth for product?

The screenshot displays the Mode Studio web interface. At the top, there's a navigation bar with tabs for 'Inbox (3,825)', 'kunayasanth123', 'Trinity Dashboard', 'Untitled Report | Mode', and another 'Trinity Dashboard'. The browser address bar shows 'app.mode.com/editor/yaswanthkuna/reports/27f3d44d0a00/queries/2be3a07ce1b8'. Below the navigation bar, the main workspace is titled 'Untitled Report' and 'Report • Share View'. The central area contains a SQL editor with a query to calculate user growth. The query uses a window function to calculate the cumulative count of distinct users over time. Below the editor, a table shows the results of the query, with columns for weeknum, num\_user, and sum. The results show a steady increase in the number of users over 9 weeks. On the right side, there's a sidebar with a search bar and a list of tables in the 'tutorial.yammer' database, including 'tutorial.yammer\_users', 'tutorial.yammer\_events', 'tutorial.yammer\_emails', and 'tutorial.yammer\_experiments'. The 'tutorial.yammer\_users' table is selected, and its schema is displayed, showing columns like 'user\_id', 'created\_at', 'company\_id', 'language', 'activated\_at', and 'state'. At the bottom, there's a status bar showing 'Showing rows 1-52 of 52', 'Columns 3', 'Size 2KB', 'Run a few seconds ago', and 'Returned in 492ms'.

```
10 -- count(distinct a.user_id)
11 -- from tutorial.yammer_events a
12 -- group by weeknum;
13 select weeknum, num_user,
14 sum(num_user) over (order by weeknum rows between unbounded preceding and current row )
15 FROM
16 (
17 select
18 extract(week from a.activated_at) as weeknum,
19 count(distinct user_id) as num_user
20 from tutorial.yammer_users a
21 where state = 'active'
22 group by weeknum
23 ) a
```

weeknum	num_user	sum
1	158	158
2	151	309
3	159	468
4	149	617
5	160	777
6	180	957
7	176	1133
8	166	1299
9	160	1459

Explanation :- amount of users growing over time

To find total users for every week we group by week and find count of different users whose state is active

Then to find growth we find the cumulative of users using window function.



**3. Weekly Retention:** Users getting retained weekly after signing-up for a product.  
**Your task:** Calculate the weekly retention of users-sign up cohort?

Ans:-

The screenshot shows the Mode Studio interface with a SQL query editor and a results table. The query calculates weekly retention by joining sign-up and engagement events for the same user, identifying those who engaged in the week following their sign-up.

```
98 with cte as
99 (
100 select user_id, sum(case when retention_rate = 1 then 1 else 0 end) as retained_users
101 from (
102 select a.user_id, a.sign_up_week, b.engagement_week, b.engagement_week - a.sign_up_week as retention_rate
103 from
104 (select distinct user_id, extract(week from occurred_at) as sign_up_week
105 from tutorial.yammer_events
106 where event_type = 'signup_flow' and event_name = 'complete_signup'
107 and extract(week from occurred_at) = 18)a
108 left join
109 (select distinct user_id, extract(week from occurred_at) as engagement_week
110 from tutorial.yammer_events
111 where event_type = 'engagement')b
112 on a.user_id = b.user_id) c
113 group by user_id)
114
115 select user_id as "retained users for a week"
116 from cte
117 where retained_users = 1;
```

The results table, titled "retained users for a week", displays the following data:

	retained users for a week
1	11775
2	11779
3	11780
4	11787
5	11791
6	11793

At the bottom of the interface, a status bar indicates: "Showing rows 1-64 of 64 Columns 1 Size 5128 Run a few seconds ago Returned in 819ms".

Explanation: retention = next\_day – today

1. Find people signup in first week left join
2. Find people engage in the next weeks with user\_id is common in both tables
3. From this we select a.user\_id, a.sign\_up\_week, b.engagement\_week, b.engagement\_week – a.sign\_up\_week = retention\_rate
4. From this we assign retention rate value 1 with 1 other with 0 using case statement we make this table as common table expression.
5. From this common table expression we select users having retained\_rate as 1
6. Finally, we get all the retained users.

**4. Weekly Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.

**Your task:** Calculate the weekly engagement per device?

Ans:-

The screenshot displays the Mode Studio web application interface. At the top, there's a navigation bar with the user's name 'kuna yaswanth' and a report titled 'Untitled Report'. Below this, a SQL query is entered in a text area:

```
25
26 select
27   extract(week from occurred_at) as weeks,
28   device,
29   count(distinct user_id)
30 from tutorial.yammer_events
31 where event_type = 'engagement'
32 group by weeks, device;
```

The query has been executed successfully, as indicated by the 'Succeeded in 886ms' message. Below the query editor, a table of results is shown with columns: weeks, device, and count. The table lists 13 rows of data, showing weekly engagement counts for various devices like 'acer aspire desktop', 'acer aspire notebook', 'amazon fire phone', etc.

	weeks	device	count
1	18	acer aspire desktop	10
2	18	acer aspire notebook	21
3	18	amazon fire phone	4
4	18	asus chromebook	23
5	18	dell inspiron desktop	21
6	18	dell inspiron notebo...	49
7	18	hp pavilion desktop	15
8	18	htc one	16
9	18	ipad air	30
10	18	ipad mini	21
11	18	iphone 4s	21
12	18	iphone 5	70
13	18	iphone 5s	45

On the right side of the interface, there's a sidebar showing a list of tables in the 'tutorial.yammer' database, including 'tutorial.yammer\_users', 'tutorial.yammer\_events', 'tutorial.yammer\_emails', and 'tutorial.yammer\_experiments'. Below this, the 'yammer\_users' table is selected, showing its schema with fields like 'user\_id', 'created\_at', 'company\_id', 'language', 'activated\_at', and 'state'.

**Explanation:-** From events table, weekly engagement per device. So, group by week, device. And event\_type = “engagement”, Counting the users.

## 5. Email Engagement: Users engaging with the email service.

**Your task:** Calculate the email engagement metrics?

The screenshot shows the Mode Studio interface with a SQL query editor and a results table. The query calculates email engagement metrics by grouping data by company ID and calculating the percentage of opened and clicked emails relative to the total sent emails.

```
select
100.0 * sum(case when e_cat = 'email_open' then 1 else 0 end) / sum(case when e_cat = 'email_sent' then 1 else 0 end) as "opened_percent",
100.0 * sum(case when e_cat = 'email_clicked' then 1 else 0 end) / sum(case when e_cat = 'email_sent' then 1 else 0 end) as "clicked_percent"
FROM
(select *,
case
when action in ('sent_reengagement_email', 'sent_weekly_digest')
then 'email_sent'
when action in ('email_open')
then 'email_open'
when action in ('email_clickthrough')
then 'email_clicked'
end as e_cat
from
tutorial.yammer_emails) a ;
```

The results table shows the following data:

Data	Fields	Source
1	opened_percent: 33.5834, clicked_percent: 14.7899	

The interface also shows a sidebar with a list of tables: tutorial.yammer\_users, tutorial.yammer\_events, tutorial.yammer\_emails, and tutorial.yammer\_experiments. The bottom status bar indicates the query was successful in 590ms.

Explanation :-

1. Here, we have distinct action such as email\_open, sent\_reengagement\_email, sent\_weekly\_digest, email\_sent
2. sent\_reengagement\_email as sent using case, sent\_weekly\_digest as sent using case, email\_open as open all these as e\_cat
3. finally, we select opened percent as (count(open)) / count(sent) as open
4. then we select clicked percent as (count(click)) / count(sent) as clicked
5. we get these two things as email engagement.

## Another way:-

The screenshot displays the Mode Studio web application. At the top, there's a browser tab bar with several open tabs including 'Inbox (3,825)', 'Trinity Dashboard', 'Untitled Report | Mode', and 'cohort meaning - Google Search'. The address bar shows the URL 'app.mode.com/editor/yaswanthkuna/reports/27f3d44d0a00/queries/2be3a07ce1b8'. Below the browser, the Mode Studio interface is visible. The top bar shows the user 'kuna yaswanth' and the report name 'Untitled Report'. The main area is divided into a query editor and a results table. The query editor contains a SQL query that calculates email engagement metrics grouped by week. The results table shows 12 rows of data with columns for weeks, sent\_weekly\_digest, email\_open, email\_clickthrough, and reengagement. A right-hand sidebar shows a list of tables in the 'tutorial.yammer' database, including 'tutorial.yammer\_users', 'tutorial.yammer\_events', 'tutorial.yammer\_emails', 'tutorial.yammer\_experiments', and 'tutorial.yammer\_emails'. The bottom of the screen shows a Windows taskbar with the date '24-06-2023' and time '13:52'.

```
61 -- from
62 -- tutorial.yammer_emails) a ;
63
64
65 select
66 extract(week from occurred_at) as weeks,
67 count(distinct (case when action = 'sent_weekly_digest' then user_id end)) as sent_weekly_digest,
68 count(distinct (case when action = 'email_open' then user_id end)) as email_open,
69 count(distinct (case when action = 'email_clickthrough' then user_id end)) as email_clickthrough,
70 count(distinct (case when action = 'sent_reengagement_email' then user_id end)) as reengagement
71 from tutorial.yammer_emails
72 group by weeks
```

	weeks	sent_weekly_digest	email_open	email_clickthrough	reengagement
1	18	908	332	187	98
2	19	2602	910	431	164
3	20	2665	961	478	175
4	21	2733	984	495	179
5	22	2822	1008	447	179
6	23	2911	972	483	199
7	24	3003	1058	527	190
8	25	3105	1144	558	234
9	26	3207	1080	517	187
10	27	3302	1157	554	222
11	28	3399	1215	619	214
12	29	3499	1244	604	226

Explanation: email engagement metrics.

users engaging in distinct action over 7 days means we group by week and then count all the distinct actions

(all distinct actions means count(sent\_weekly\_digest), count(sent\_reengagement), count(email\_clickthrough), count(email\_open))

We get the actions of the mail done by users in 7 days.

### Insights:-

1. With the help of this project, I gained hands on practical experience with window functions
2. With the help of this project, I gained at most knowledge on subqueries and usage of them in the practical way including hands on
3. Solved the complex problems like retention with better way
4. Through project labs learnt some good knowledge on retention and cohort analysis.

### Result: -

It was a great learning opportunity for me as I was able to get hands on experience, and also improve my skills in MySQL. This project also helped me understand and complex queries, retention, cohort analysis. The guidance and resources offered by the team made learning easier and the project a success. Project labs helped me to solve retention problems. I'm grateful and thankful for the wonderful opportunity.

Thank you trainity 😊