

# Operation Analytics and Investigating Metric Spike

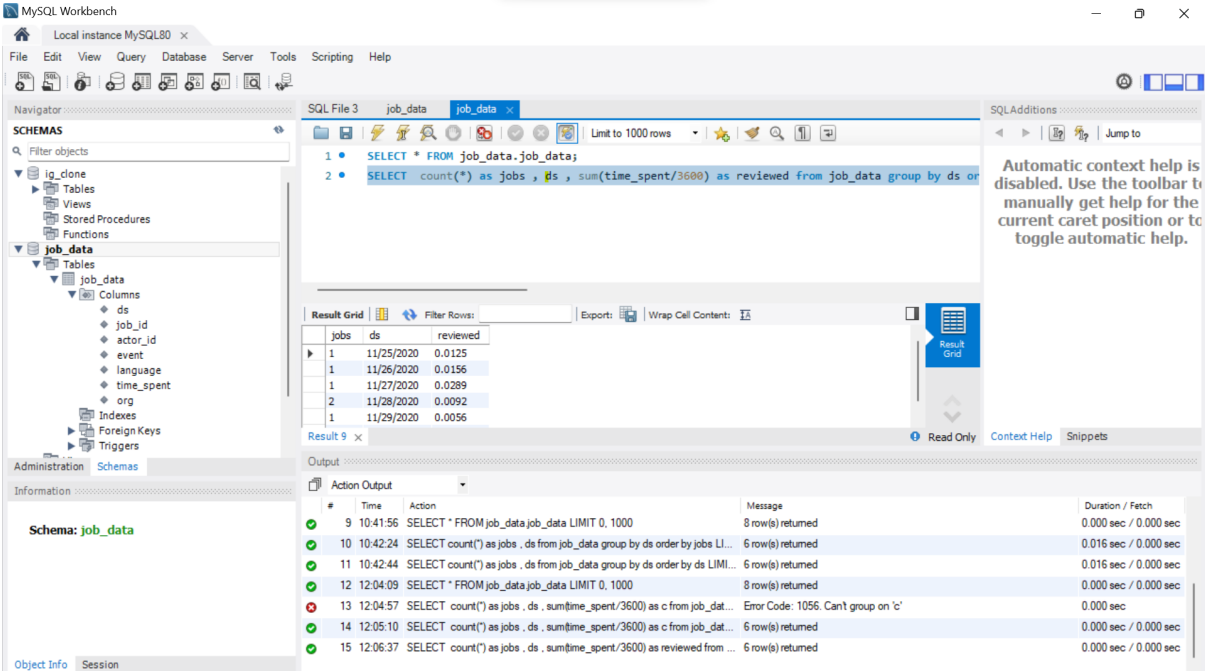
This is the third project as a part of training. This project requires an intermediate knowledge of SQL so doing this project is very much beneficial as it takes our knowledge to the next level. It has two case studies of which case study 1 is a job data analysis with four queries and case study 2 is investigating metric spike with five queries. Answering these queries will gain us more experience and confidence in SQL. There will be some queries in which we get stuck but it is all a part of self learning because to clear these doubts we research on the internet leading to create a strong foundation on basic concepts which is far better than spoon feeding.

## Approach

First we need to download all the datasets provided. There are two case studies. Case study1 contains a small dataset so I imported it into mysql using import wizard, but the dataset of case study 2 is very large which will consume a lot of time if we import it using import wizard so import it by following the instructions provided in the given video.

### Case study 1:

1. Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.



The screenshot shows the MySQL Workbench interface. The SQL Editor contains two queries:

```
1. SELECT * FROM job_data.job_data;
2. SELECT count(*) as jobs , ds , sum(time_spent/3600) as reviewed from job_data group by ds order by ds;
```

The Results window displays the output of the second query, showing a table with columns: jobs, ds, and reviewed. The data is grouped by date (ds) and ordered by date.

	jobs	ds	reviewed
1	1	11/25/2020	0.0125
1	1	11/26/2020	0.0156
1	1	11/27/2020	0.0289
2	1	11/28/2020	0.0092
1	1	11/29/2020	0.0056

The Action Output window shows the execution of the queries, including the error message for the second query: "Error Code: 1056. Can't group on 'c'".

Here, I need to calculate the number of jobs that are reviewed per hour for each day. Since the dataset is in November, I did not specify the month in the query. I counted the number of jobs and I used sum() on the time\_spent and divided it by 3600 to get the review per hour since the time\_spent is given in seconds and group and order it by the date.

2. 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.

The screenshot shows the MySQL Workbench interface with a SQL query in the 'SQL File 3' editor. The query is as follows:

```

3  SELECT
4  ds,
5  COUNT(ds) OVER (ORDER BY ds) AS noj,
6  SUM(time_spent) OVER (ORDER BY ds) AS sot,
7  AVG(time_spent) OVER (ORDER BY ds) AS rolling_avg
8  FROM
9  job_data;

```

The 'Result Grid' shows the following data:

ds	noj	sot	rolling_avg
11/25/2020	1	45	45.0000
11/26/2020	2	101	50.5000
11/27/2020	3	205	68.3333
11/28/2020	5	238	47.6000
11/28/2020	5	238	47.6000

The 'Action Output' pane shows the execution of the query, including the message: 'Error Code: 1140. In aggregated query without GROUP BY, expression #1...'.

Here, I am required to calculate the 7-day rolling average throughput but in the dataset only some days are given. I counted the number of jobs by counting the dates, I performed sum() on the time\_spent and also calculated the average. I used the window function in all these rather than group by.

I prefer using the daily metric for the given date but if the dates are consistent and if the dataset is long, I would prefer a 7-day average throughput.

3. Write an SQL query to calculate the percentage share of each language over the last 30 days.

The screenshot shows the MySQL Workbench interface with a SQL query in the 'SQL File 3' editor. The query is as follows:

```

12 language,
13 COUNT(*) AS total_count,
14 (COUNT(*) * 100.0 / SUM(COUNT(*) OVER ())) AS percentage_share
15 FROM
16 job_data
17 GROUP BY
18 language;

```

The 'Result Grid' shows the following data:

language	total_count	percentage_share
English	1	12.50000
Arabic	1	12.50000
Persian	3	37.50000
Hindi	1	12.50000
French	1	12.50000

The 'Action Output' pane shows the execution of the query, including the message: 'Error Code: 1146. Table 'job\_data.your\_table' doesn't exist'.

Here, I have to calculate the percentage share of each language so I selected the language and counted each of them and then i calculated the percentage share by calculating the  $\text{count} \times 100 / \text{sum of total count}$ . I did not specify the calculation for the last 30 days as the dataset does not contain much data.

4. Write an SQL query to display duplicate rows from the job\_data table.

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

```

20 SELECT *
21 FROM job_data
22 WHERE (ds,job_id,actor_id,event,language,time_spent,org) IN (
23     SELECT ds,job_id,actor_id,event,language,time_spent,org
24     FROM job_data
25     GROUP BY ds,job_id,actor_id,event,language,time_spent,org
26     HAVING COUNT(*) > 1

```

The Result Grid shows the columns: ds, job\_id, actor\_id, event, language, time\_spent, org. The Output panel shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
7	10:13:52	SELECT language, COUNT(*) AS total_count, (COUNT(*) * 100...	Error Code: 1146. Table 'job_data.your_table' doesn't exist	0.000 sec
8	10:14:16	select* from job_data.job_data LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
9	10:15:14	SELECT ds, COUNT(ds) OVER (ORDER BY ds) AS noj, SUM(t...	8 row(s) returned	0.000 sec / 0.000 sec
10	10:15:14	SELECT language, COUNT(*) AS total_count, (COUNT(*) * 100...	0 row(s) returned	0.000 sec / 0.000 sec
11	10:15:39	SELECT language, COUNT(*) AS total_count, (COUNT(*) * 100...	6 row(s) returned	0.000 sec / 0.000 sec
12	11:28:14	select* from job_data.job_data LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
13	11:30:08	SELECT * FROM job_data WHERE (ds,job_id,actor_id,event,language...	0 row(s) returned	0.000 sec / 0.000 sec

Here, I need to find out the duplicate rows. One way to find out is to check for the rows of which all the columns are repeating with any previous rows. To do that I selected all the rows from the job\_data which are repeating by giving a condition to select those which have the count of having the same column greater than one.

## Case study 2:

1. Write an SQL query to calculate the weekly user engagement.

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

```
1 select *from users;
2 select *from events;
3
4 select extract(week from occurred_at) as num ,count(distinct user_id) from events group by num;
```

The Result Grid shows the output of the query:

num	count(distinct user_id)
17	663
18	1068
19	1113
20	1154

The Output tab shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
49	19:45:40	select *from users LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
50	19:45:50	select *from users LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
51	19:52:04	select *from events LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
52	19:57:44	select extract(week from occurred_at) as num ,count(user_id) group by num	Error Code: 1054. Unknown column 'occured_at' in field list	0.000 sec
53	19:58:10	select extract(week from occurred_at) as num ,count(distinct user_id) group by num	Error Code: 1054. Unknown column 'occured_at' in field list	0.000 sec
54	19:58:40	select extract(week from occurred_at) as num ,count(user_id) from events group by num	19 row(s) returned	0.328 sec / 0.000 sec
55	19:59:40	select extract(week from occurred_at) as num ,count(distinct user_id) from events group by num	19 row(s) returned	0.375 sec / 0.000 sec

Here, I am required to calculate the weekly user engagement , so I used extract() to get the weeks and count the number of user\_id of that particular week . This way I got the number of users present in the event for each week.

2. Write an SQL query to calculate the user growth for the product.

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

```
14
15 select language, count(case when state='active' then 1 else null end) as user_growth from users group by language;
16
17 select language , count(state) from users group by language;
18
```

The Result Grid shows the output of the query:

language	count(state)
english	4773
german	515
indian	280
french	727
japanese	658
italian	198

The Output tab shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
8	20:12:24	select language, count(case when state='active' then 1 end) as user_growth from users group by language	12 row(s) returned	0.031 sec / 0.000 sec
9	20:15:14	select language, count(case when state='active' then 1 end) as user_growth from users group by language	12 row(s) returned	0.031 sec / 0.000 sec
10	20:15:36	select *from users LIMIT 0, 1000	1000 row(s) returned	0.015 sec / 0.000 sec
11	20:16:11	select language, count(case when state='active' then 0 end) as user_growth from users group by language	12 row(s) returned	0.015 sec / 0.000 sec
12	20:18:00	select language, count(case when state='active' then 1 else null end) as user_growth from users group by language	12 row(s) returned	0.016 sec / 0.000 sec
13	20:19:13	select language , count(state) group by language	Error Code: 1054. Unknown column 'language' in field list	0.015 sec
14	20:19:27	select language , count(state) from users group by language LIMIT 0, 1000	12 row(s) returned	0.032 sec / 0.000 sec

Here, I am required to calculate the user growth of the product. So for that I counted all the active states per language and presented it in the result grid. However, I am not sure if this is the required result.

3. Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

The screenshot shows MySQL Workbench with a query window containing the following SQL code:

```

1 WITH cohort AS (
2     SELECT
3         user_id,
4         EXTRACT(WEEK FROM created_at) AS created_week
5     FROM
6         users
7 ),
8 activation AS (
9     SELECT
10        users.user_id,
11        MIN(EXTRACT(WEEK FROM users.activated_at)) AS min_activated_week
12    FROM
13        users
14    WHERE

```

The output window shows the execution results with columns: #, Time, Action, Message, and Duration / Fetch. The output indicates that the query was executed successfully, returning 4 rows.

Here, I am required to calculate the weekly retention of users. I tried different ways but I was not getting the desired result.

4. Write an SQL query to calculate the weekly engagement per device.

The screenshot shows MySQL Workbench with a query window containing the following SQL code:

```

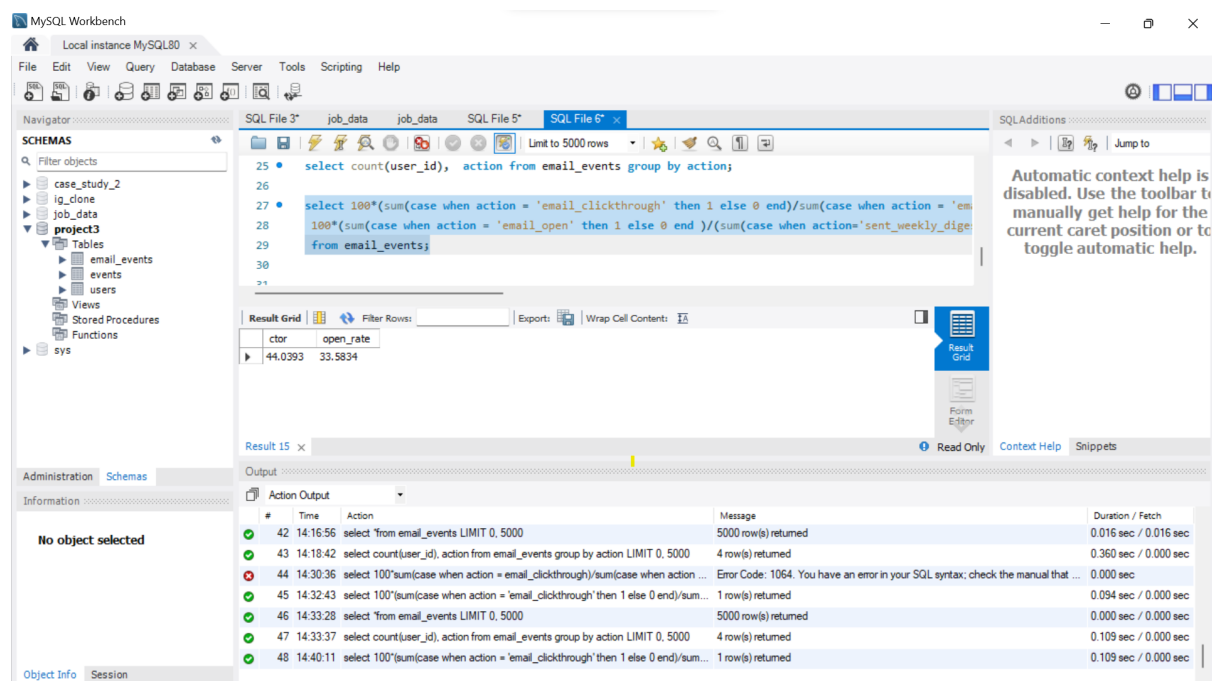
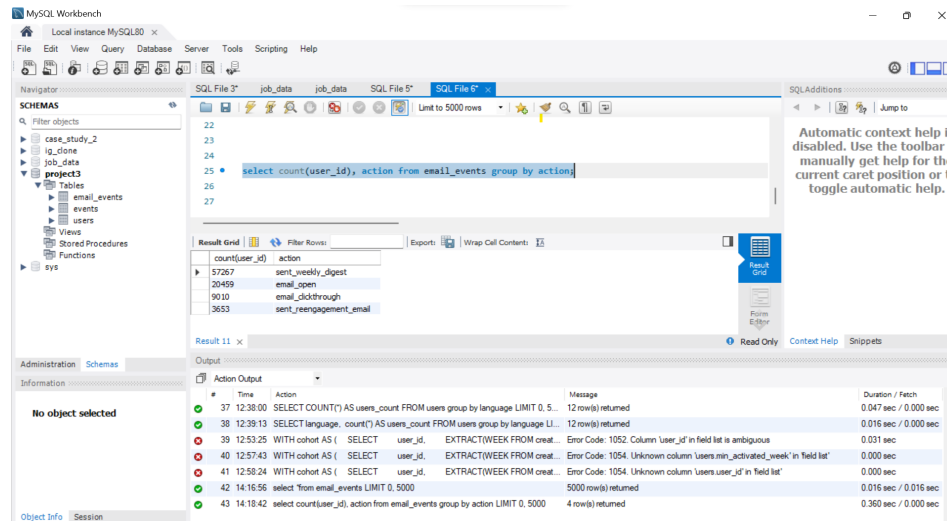
7 from(select extract(week from activated_at) as week_num , extract(year from activated_at) as year_num
8 from users where state='active' group by week_num,year_num
9 order by week_num,year_num)temp;
10
11 select extract(week from occurred_at) as weekdate, count(user_id) as no_of_users , device from events
12
13

```

The output window shows the execution results with columns: #, Time, Action, Message, and Duration / Fetch. The output indicates that the query was executed successfully, returning 491 rows.

Here, I have to calculate the weekly engagement per device. So to get the week, I first extracted the week using `extract()` from `occured_at` and I counted the `user_id` and group it by week to get the desired result.

5. Write an SQL query to calculate the email engagement metrics.



Here, I am required to calculate the email engagement metrics. So, first I check the different actions present in the database. Then I calculated the open rate and CTOR by applying the formula. CTOR is the click-to-open-rate which is the rate of number of clicks in the email out of the total emails opened. To calculate CTOR we use,  $\text{Email clicks} / \text{email open} * 100$ . Open rate is the rate of number of number of email open with respect to the number of emails sent. To calculate open rate we use,  $\text{Email open} / \text{email sent} * 100$ .

## Tech-Stack Used:

To do this project I used the mysql workbench 8.0 CE as per requirements which was already downloaded for the previous projects .. It is a visual tool for databases , developers etc. using mysql workbench adds a lot of advantages due its features. It provides user friendly graphical interfaces, it allows designing databases , it helps to plan and organise your database structures including the tables, columns, rows etc, it also allows to directly write the query and executes it simultaneously displaying the indication of errors whenever there is one and many more.another main advantage is that it is an open source software so the licensing cost is not required.

## Insights:

I gained several insights from doing this project. This project includes 2 case studies which are different from each other. Case study 1 consists of job\_data from which I learned to calculate the rolling average and percentage share. Case study 2 consists of data to investigate the metric spike. Here, I was introduced to many new terms like retention rate, cohort, engagement metrics etc. From doing email engagement metrics , I learned open rate and CTOR.I learned a lot from this project. Most importantly I learned to deal with large datasets.

## Result:

Doing this project made me realise that I have a lot more to learn even though I have grasped the basics. Each project will help me to learn more and more.I learned to implement mathematical calculations in the query. I also learned to import large datasets with ease. I learned different concepts . Learning by doing is really beneficial to understand the basic concepts to the core. This project really helps me to increase my familiarity with mysql .