

# Operation Analytics and Investigating metric spike

## Project Description:

The primary objective of the project is to leverage advanced SQL skills to analyse various datasets and tables, providing valuable insights to different departments within the company. The focus is on understanding and explaining sudden changes in key metrics, such as user engagement and sales, to identify areas for improvement in the company's operations. This project aims to facilitate data-driven decision-making processes and ultimately enhance overall performance.

## Approach:

**Data exploration and Understanding:** thoroughly exploring the provided datasets and tables to understand the structure, relationships, and relevance of the data. Identify key metrics and performance indicators crucial for operational analysis, such as daily user engagement etc.

**Metric Spike Investigation:** Defining criteria for identifying metric spikes based on the nature of the data and business objectives. Develop SQL queries to detect sudden changes or anomalies in key metrics over specific time periods.

**Root cause Analysis:** Utilize sql queries to drill down to granular data and identify patterns, correlations, or anomalies associated with the spikes.

## Tech Stack use:

**MYSQL Workbench:** It is a open source tool which can support multiple database systems that can perform complex calculations and data manipulation with ease.

## Insights

### Casestudy\_1

The dataset provides details about review process of applicants for various job profiles  
The column details of the dataset are:

- **job\_id:** Unique identifier of jobs
- **actor\_id:** Unique identifier of actor
- **event:** The type of event (decision/skip/transfer).
- **language:** The Language of the content
- **time\_spent:** Time spent to review the job in seconds.
- **org:** The Organization of the actor
- **ds:** The date in the format yyyy/mm/dd (stored as text).

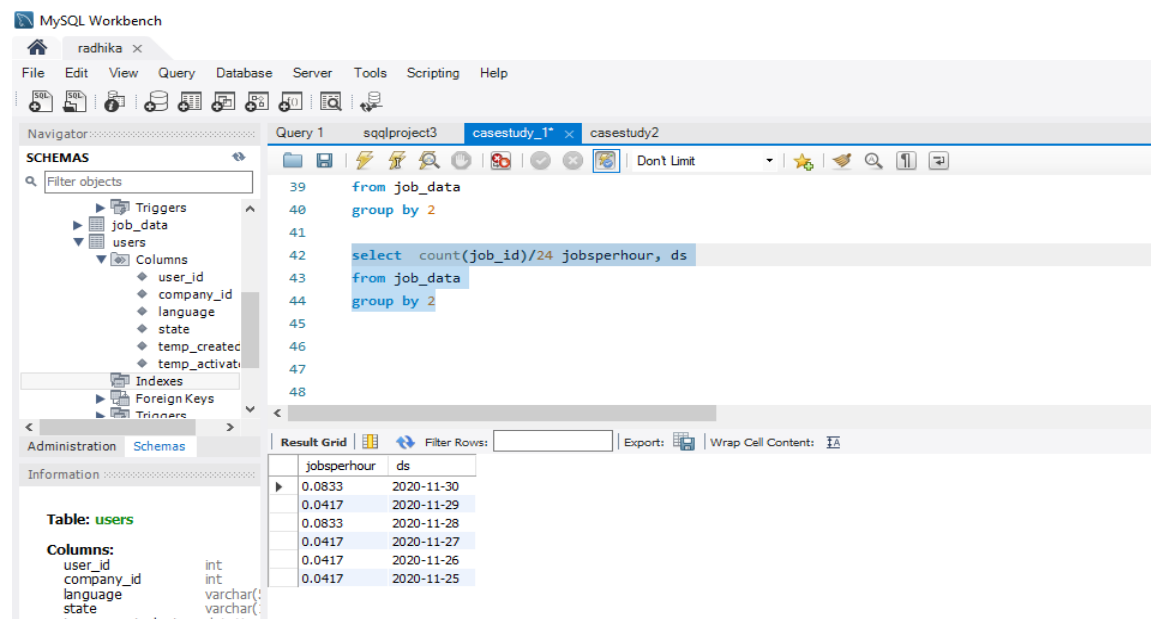
### Task 1: Jobs reviewed over time

No of jobs reviewed per hour each day in november 2020

SQL Code:

```
select count(job_id)/24 jobsperhour, ds  
from job_data  
group by 2
```

Analysing fluctuations in jobs across different days can help identify patterns in workload distribution and potentially correlate them with external factors such as weekdays, weekends, holidays or specific events. This can assist in resource planning, workload management, optimizing operational efficiency within the organization.



## Task 2 (Throughput Analysis)

SQL Code:

```
select ds, eventspersecond,  
avg(eventspersecond) over(order by ds rows between 6 preceding and current row)  
7dayrolling_average from  
(select count(event)/sum(time_spent) eventspersecond, ds  
from job_data  
group by 2  
order by 2) a
```

the daily metric can give us a sudden rise or dip in the throughput which indicates the effects of seasonal factors like end of the month, holidays and weekend activity.

Using 7 day rolling average offers a smoothed-out view of throughput, reducing the impact of daily fluctuations and providing a more stable measure of overall performance trends.

This approach is preferable for identifying long-term patterns and trends in throughput and is good for strategic decision making and trend analysis.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'job\_data' and 'users' tables. The main query window contains the following SQL code:

```

26 group by 2) a
27 Execute the selected portion of the script or everything, if there is no selection
28 select ds, eventspersecond,
29 avg(eventspersecond) over(order by ds rows between 6 preceding and current row) 7dayrolling_average from
30 (select count(event)/sum(time_spent) eventspersecond, ds
31 from job_data
32 group by 2
33 order by 2) a
34
35 select count(events) eventspersecond, ds

```

The 'Result Grid' shows the following data:

ds	eventspersecond	7dayrolling_average
2020-11-25	0.0222	0.02220000
2020-11-26	0.0179	0.02005000
2020-11-27	0.0096	0.01656667
2020-11-28	0.0606	0.02757500
2020-11-29	0.0500	0.03206000
2020-11-30	0.0500	0.03505000

### Task 3( language share)

SQL Code:

**select language, count(language)/(select count(\*) a from job\_data)\*100 p from job\_data group by language**

the query retrieves the percentage share of languages preferred or mostly in use. Very clearly from the query result, Persian language is most used.

Analysing the percentage share of languages can help identify the primary languages used in the events recorded in the last 30 days. This can indicate language specific marketing or understanding user demographics and preferences.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'job\_data' and 'users' tables. The main query window contains the following SQL code:

```

60
61
62 select language, count(language)/(select count(*) a from job_data)*100 p from job_data group by language
63
64
65
66
67
68
69

```

The 'Result Grid' shows the following data:

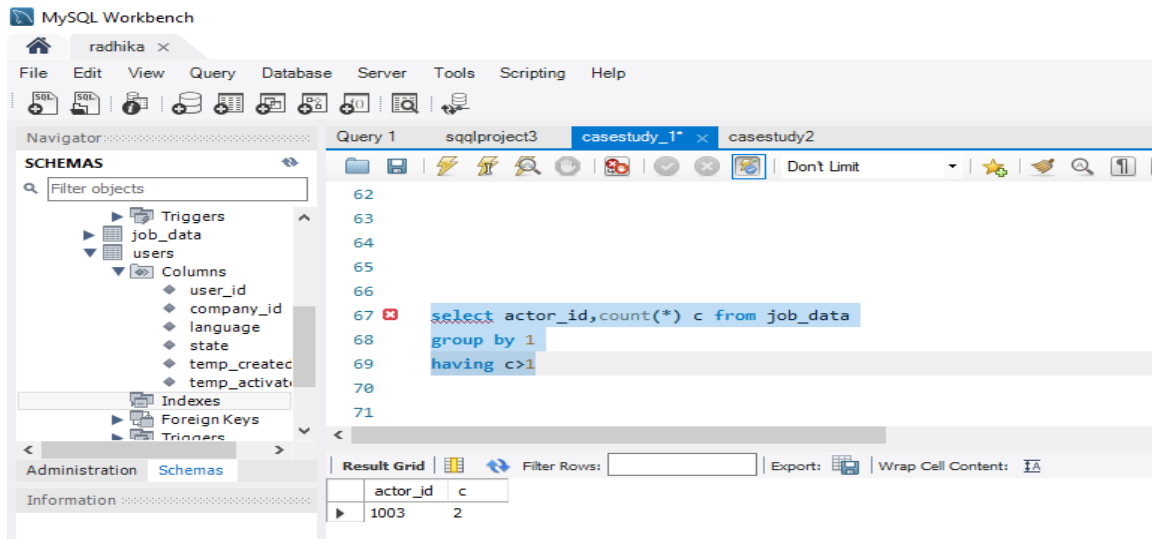
language	p
English	12.5000
Arabic	12.5000
Persian	37.5000
Hindi	12.5000
French	12.5000
Italian	12.5000

#### Task 4(duplicate rows)

To find duplicate rows we need to decide the column we are going to traverse for finding duplicates

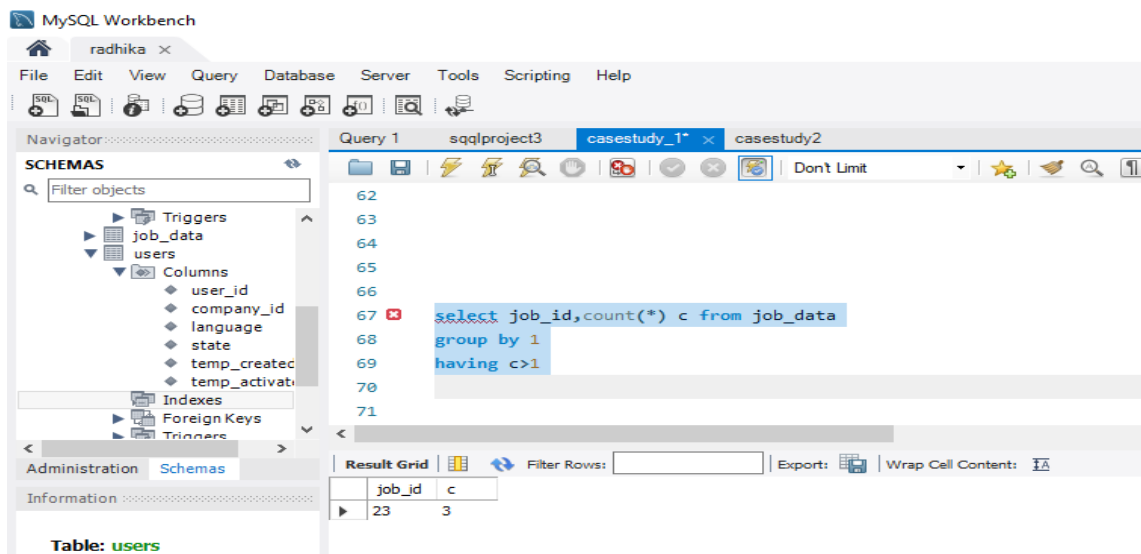
SQL Code:

```
select actor_id,count(*) c from job_data
group by 1
having c>1
```



If we consider actor\_id there are 2 duplicate entries whereas if we consider job\_id there are 3 duplicate entries

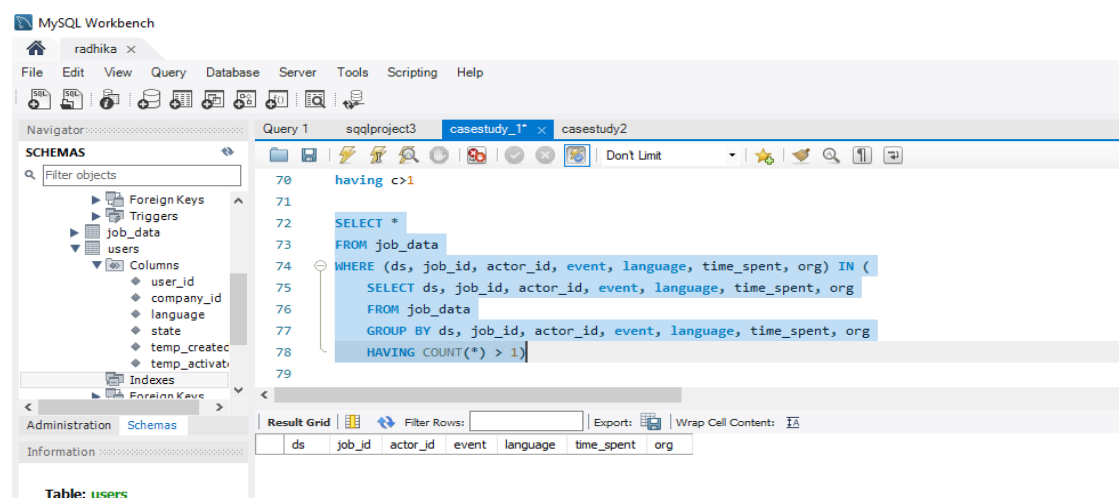
```
select job_id,count(*) c from job_data
group by 1
having c>1
```



But considering all the columns there are no duplicate entries in the table. duplicate job\_id can indicate same job was being reviewed by different actors or duplicate actors could mean single actor was reviewing different jobs hence considering all the columns there are no duplicate rows in the table.

SQL Code:

```
SELECT *
FROM job_data
WHERE (ds, job_id, actor_id, event, language, time_spent, org) IN (
  SELECT ds, job_id, actor_id, event, language, time_spent, org
FROM job_data
GROUP BY ds, job_id, actor_id, event, language, time_spent, org
HAVING COUNT(*) > 1)
```



## Casestudy\_2

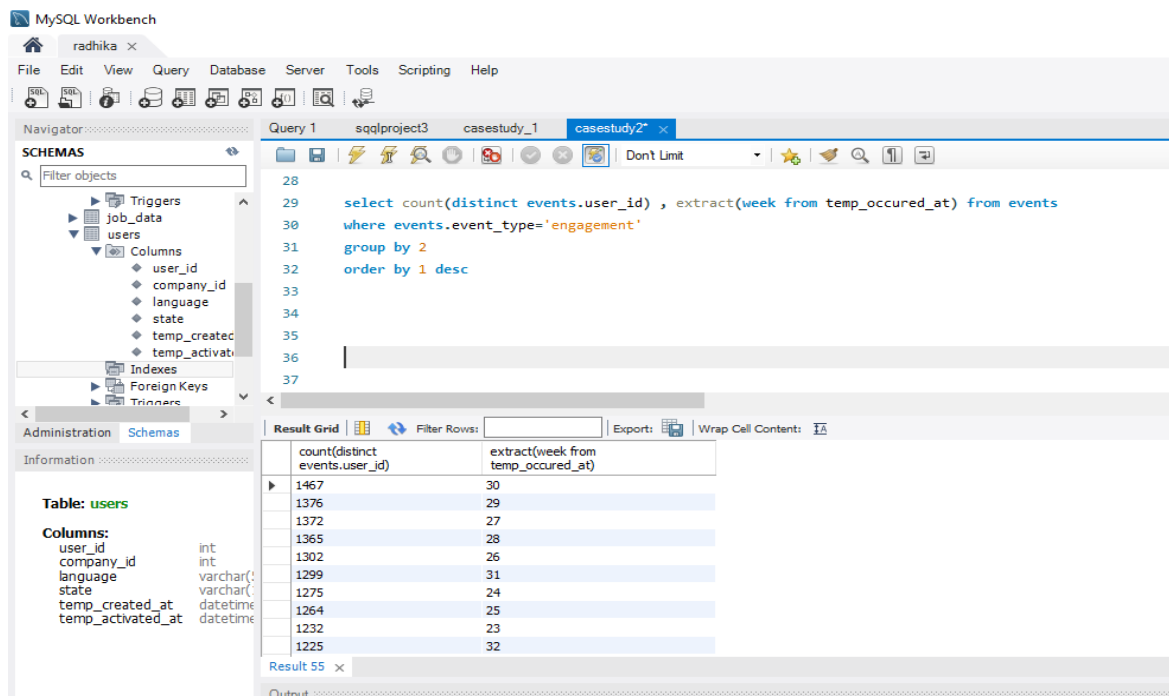
### Task1(weekly User engagement)

SQL Code:

```
select count( distinct events.user_id), week(events.temp_occured_at) w from events
where events.event_type='engagement'
group by 2
order by 1 desc
```

the count is calculated on weekly basis and the lowest activity can be seen on week 35 with 104 users and highest activity on week 30 with 1467 users

Analysing weekly user engagement can inform decisions related to marketing campaigns, product updates and user experience enhancements to improve overall engagement and lower count can indicate potential issues with retention and activity.



## Task 2(User growth over time)

SQL Code:

```

select num_active_users,y,w,sum(num_active_users) over (order by y,w) cc from
(select count(distinct user_id) num_active_users, year(temp_activated_at) y,
week(temp_activated_at) w from users
group by 2,3
order by 2,3) sub
  
```

we can see the user count started at 23 at starting of the year 2013 and has wen till 9381 to the end of year 2014. if we compare for both years separately the user growth has increased over the weeks.

Analyzing growth trajectory of the products user base over time can inform strategic decisions related to marketing, product development and user engagement to sustain and accelerate growth.

But we look into daily metric of num of active users created everyday the numbers may go low sometimes, considering the daily results and other related factors like seasonal factors like engagement on weekdays and weekends there can be adaption of improvements in acquisition strategies, products features, or market conditions

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'users'. The main editor shows a SQL query (Query 1) with the following code:

```

60
61 select num_active_users,y,w,sum(num_active_users) over (order by y,w) cc from
62 (select count(distinct user_id) num_active_users, year(temp_activated_at) y, week(temp_activated_at) w from users
63 group by 2,3
64 order by 2,3) sub
65 select* from users
66 where language='german'
67
68 select c.sw, sum(case when c.rw=1 then 1 else 0 end) week1retention,
69 sum(case when c.rw=2 then 1 else 0 end) week2retention,

```

Below the query editor, the 'Result Grid' shows the output of the query. The columns are 'num\_active\_users', 'y', 'w', and 'cc'. The data is as follows:

num_active_users	y	w	cc
23	2013	0	23
30	2013	1	53
48	2013	2	101
36	2013	3	137
30	2013	4	167
48	2013	5	215
38	2013	6	253
42	2013	7	295
34	2013	8	329
43	2013	9	372
32	2013	10	404

### Task 3(weekly customer Retention)

SQL Code:

```

select c.sw, sum(case when c.rw=1 then 1 else 0 end) week1retention,
sum(case when c.rw=2 then 1 else 0 end) week2retention,
sum(case when c.rw=3 then 1 else 0 end) week3retention,
sum(case when c.rw=4 then 1 else 0 end) week4retention,
sum(case when c.rw=5 then 1 else 0 end) week5retention,
sum(case when c.rw=6 then 1 else 0 end) week6retention,
sum(case when c.rw=7 then 1 else 0 end) week7retention,
sum(case when c.rw=8 then 1 else 0 end) week8retention
from
(select a.user_id, a.sw, b.ew, b.ew-a.sw rw from
((select user_id, min(week(temp_occured_at)) sw from events
where event_type='signup_flow' and event_name='complete_signup' group by 1
order by 2) a
join (select user_id, week(temp_occured_at) ew from events
where event_type='engagement' group by 1,2
order by 2) b
on a.user_id=b.user_id)) c
group by c.sw
order by c.sw

```

SQL Query:

```

81 join (select user_id, week(temp_occured_at) ew from events
82 where event_type='engagement' group by 1,2

```

Result Grid:

sw	week1retention	week2retention	week3retention	week4retention	week5retention	week6retention	week7retention	week8retention
17	59	24	16	11	16	11	9	6
18	114	73	49	37	26	19	25	13
19	142	73	59	40	25	22	19	23
20	128	86	52	39	29	22	32	22
21	121	74	51	34	23	31	30	20
22	142	82	57	47	38	29	23	26
23	146	85	57	51	43	35	27	22
24	151	89	58	41	32	30	25	15
25	165	97	61	40	29	22	19	15
26	138	84	60	45	35	32	25	16
27	161	95	80	51	38	27	23	0
28	161	92	56	35	18	19	0	0
29	160	81	53	39	33	1	0	0
30	171	94	65	43	3	0	0	0
31	136	69	52	1	0	0	0	0
32	174	81	8	0	0	0	0	0
33	187	8	0	0	0	0	0	0

Output:

Action Output

# Time Action Message

251 17:38:49 select num\_active\_users.y,w.sum(num\_active\_users) over (order by y,w) cc from (select co... 89 row(s) returned

The code written is for weekly retention for successive 8 weeks starting from the initial signup of the users for each week. we can add many weeks as we want to check retention. clearly from the results the retention has decreased over the weeks.

By analysing the retention rates, we can assess the effectiveness of onboarding process and engagement strategies higher retention rates indicate larger portion of users are continuing to engage with the platform and overall user experience is good but lower retention rates may indicate areas for improvement is user experience, feature adoption, or communication strategies to retain users.

#### Task 4(weekly engagement per device)

SQL Code:

```

select count(distinct events.user_id) countofusers,
extract(week from temp_occured_at) week, device from events
where events.event_type='engagement'
group by 3,2
order by 1 desc

```

the results can help identify trends in weekly user engagement and assess the impact of different devices on user interaction.

the top device used by the users are macbook pro followed by Lenovo thinkpad which can indicate on planning marketing campaigns targeting users of that device or improvements in the user experience specific to that device.

Overall this analysis provides valuable insights for optimizing user engagement strategies , device compatibility.



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'Foreign Keys', 'Triggers', 'job\_data', 'users', and 'Columns'. The 'users' table is selected, showing its columns: 'user\_id' (int), 'company\_id' (int), 'language' (varchar), 'state' (varchar), 'temp\_created\_at' (datetime), and 'temp\_activated\_at' (datetime). The main query editor shows the following SQL code:

```

42
43 se Execute the selected portion of the script or everything, if there is no selection
44 extract(week from temp_occured_at) week, device from events
45 where events.event_type='engagement'
46 group by 3,2
47 order by 1 desc
48

```

The 'Result Grid' at the bottom displays the results of the query:

countofusers	week	device
322	30	macbook pro
321	31	macbook pro
312	33	macbook pro
307	32	macbook pro
302	27	macbook pro
295	28	macbook pro
295	29	macbook pro
292	34	macbook pro
275	25	macbook pro
269	26	macbook pro
266	19	macbook pro
266	23	macbook pro
256	20	macbook pro
255	24	macbook pro
252	18	macbook pro

## Task 4(Email Engagement Analysis)

There are variety of different email engagement rates to determine the effectiveness for email marketing strategy and content, based on the dataset here are few email engagement rates

1. **Open rate**- which is emails opened divided by total sent emails.  
Based on the recent industry standards for professional services the expected rate is 28.31% and this can vary based on the particular industry or companies goals and In this case the rate is 35.72% which exceeded the target and based on companies goals and objectives, marketing strategy should be improvised accordingly.

SQL Code:

```
select ((select count(*) from email_events where action='email_open')/(select count(*) from email_events where action='sent_weekly_digest'))*100 as r
```

The screenshot shows the MySQL Workbench interface. The main query editor displays the following SQL code:

```

54
55 select ((select count(*) from email_events where action='email_open')/
56 (select count(*) from email_events where action='sent_weekly_digest'))*100 as emailopenrate
57
58
59
60

```

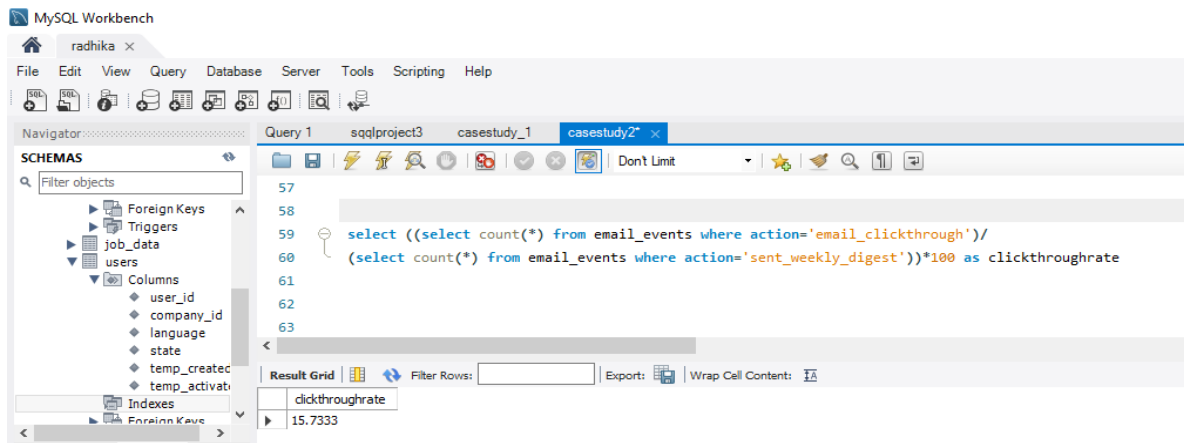
The 'Result Grid' at the bottom shows the result of the query:

emailopenrate
35.7256

2. **Click-through rate(CTR)** - which is links clicked in email divided by total sent emails.  
Based on the industry standards for professional services the expected rate is 3.01% and In this case the rate is 15.73% which is good and can indicate the user might have opened links multiple times.

SQL Code:

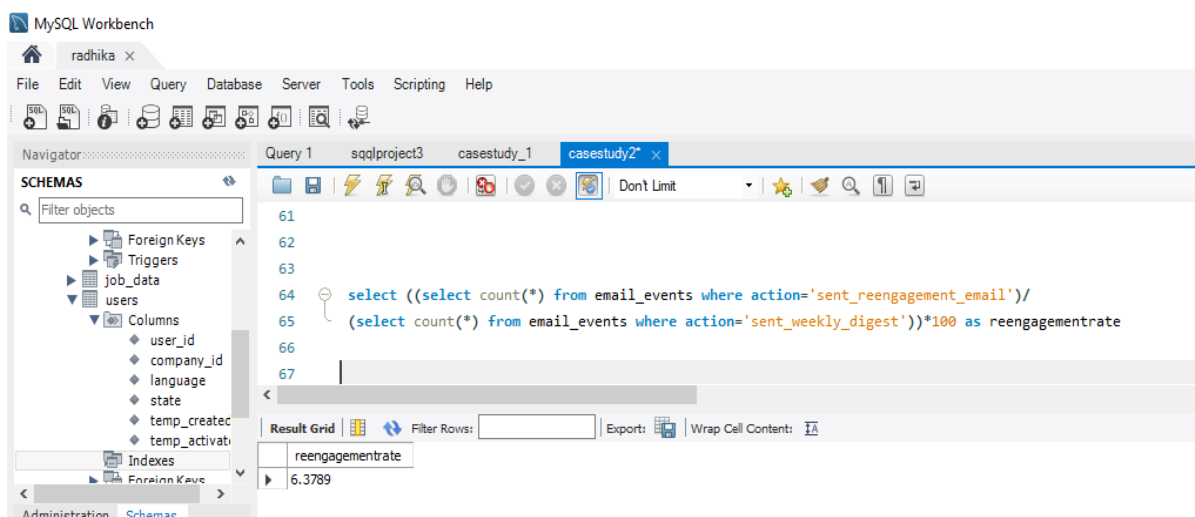
```
select ((select count(*) from email_events where action='email_clickthrough')/(select count(*) from email_events where action='sent_weekly_digest'))*100 as r
```



3. **Re-engagement rate**- which is emails sent to follow-up divided by total sent emails.  
The rate falls below 10% which is good as the follow-up mails to be sent could be less, but this clearly doesn't indicate the successful conversion has happened. So further analysis can be conducted on decision mailing.

SQL Code:

```
select ((select count(*) from email_events where action='sent_reengagement_email')/(select count(*) from email_events where action='sent_weekly_digest'))*100 as r
```



## **Result:**

**Understanding of operational processes:** the project provide me with a deeper understanding of operational process, including data collection, analysis and interpretation to improve overall business operations.

**Advanced SQL skills:** by working with various datasets and tables and performing complex SQL queries to derive insights. I have enhanced my proficiency in SQL, particularly in data manipulation, aggregation and window functions.

**Strategic Insights for performance improvement:** Analyzing trends and patterns in metrics like user engagement and throughput has provided strategic insights for performance improvement, enabling data-driven decision-making to optimize operations and enhance overall business performance.

**Preference for metric selection:** The project has also reinforced the importance of selecting appropriate metrics for analysis, considering factors such as granularity, stability and relevance for business objectives. For example choosing between daily metrics and rolling averages for throughput analysis depends on the desired level of granularity and the need for short-term versus long-term, insights.