

Operation Analytics and Investigating Metric Spike

DONE BY: G.REVAN

DESCRIPTION:

- Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect.
- Being one of the most important parts of a company, this kind of analysis is further used to predict the overall growth or decline of a company's fortune. It means better automation, better understanding between cross-functional teams, and more effective workflows.
- Investigating metric spike is also an important part of operation analytics as being a Data Analyst you must be able to understand or make other teams understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that its very important to investigate metric spike.
- You are working for a company like Microsoft designated as Data Analyst Lead and is provided with different data sets, tables from which you must derive certain insights out of it and answer the questions asked by different departments.
- We need to provide a case study on the the given datasets.

APPROACH:

We need to import our dataset i.e;tables into mysql and using different sql operations the following questions and analysis must be done.

TECHSTACK USED: MYSQL WORKBENCH

Case Study 1 (Job Data)

Below is the structure of the table with the definition of each column that you must work on:

- Table-1: job_data
- job_id: unique identifier of jobs
- actor_id: unique identifier of actor
- event: decision/skip/transfer
- language: language of the content
- time_spent: time spent to review the job in seconds
- org: organization of the actor
- ds: date in the yyyy/mm/dd format. It is stored in the form of text and we use presto to run. no need for date function

Case Study 2 (Investigating metric spike)

The structure of the table with the definition of each column that you must work on is present in the project image

- Table-1: users

- This table includes one row per user, with descriptive information about that user's account.

- Table-2: events

- This table includes one row per event, where an event is an action that a user has taken. These events include login events, messaging events, search events, events logged as users progress through a signup funnel, events around received emails.

- Table-3: email_events

- This table contains events specific to the sending of emails. It is similar in structure to the events table above.

TASKS FOR CASE STUDY-1

TASK -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?
- QUERY:
 - select
 - count(job_id) as jobs_reviewed,
 - date(ds) as date
 - from job_data
 - where ds between '1-11-2020' and '30-11-2020'
 - group by date;

TASKS FOR CASE STUDY-1

TASK -1

The screenshot displays the MySQL Workbench interface. The main window shows a SQL query in the 'Query 1' tab. The query is as follows:

```
1 select
2     count(job_id) as jobs_reviewed,
3     date(ds) as date
4 from job_data
5 where ds between '1-11-2020' and '30-11-2020'
6 group by date;
```

The 'Navigator' pane on the left shows the database structure, including schemas, tables, views, stored procedures, and functions. The 'Administration' pane at the bottom left shows the 'Schemas' tab, with the 'email' table selected. The 'Information' pane shows the columns of the 'email' table:

Column	Type
user_id	double
occurred_at	text
event_type	text
event_name	text
location	text
device	text
user_type	double

The 'Result Grid' pane at the bottom right shows the results of the query. The results are as follows:

jobs_reviewed	date
2	2020-11-30
1	2020-11-29
2	2020-11-28
1	2020-11-27
1	2020-11-26
1	2020-11-25

The 'SQL Additions' pane on the right shows a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

TASKS FOR CASE STUDY-1

TASK -2

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

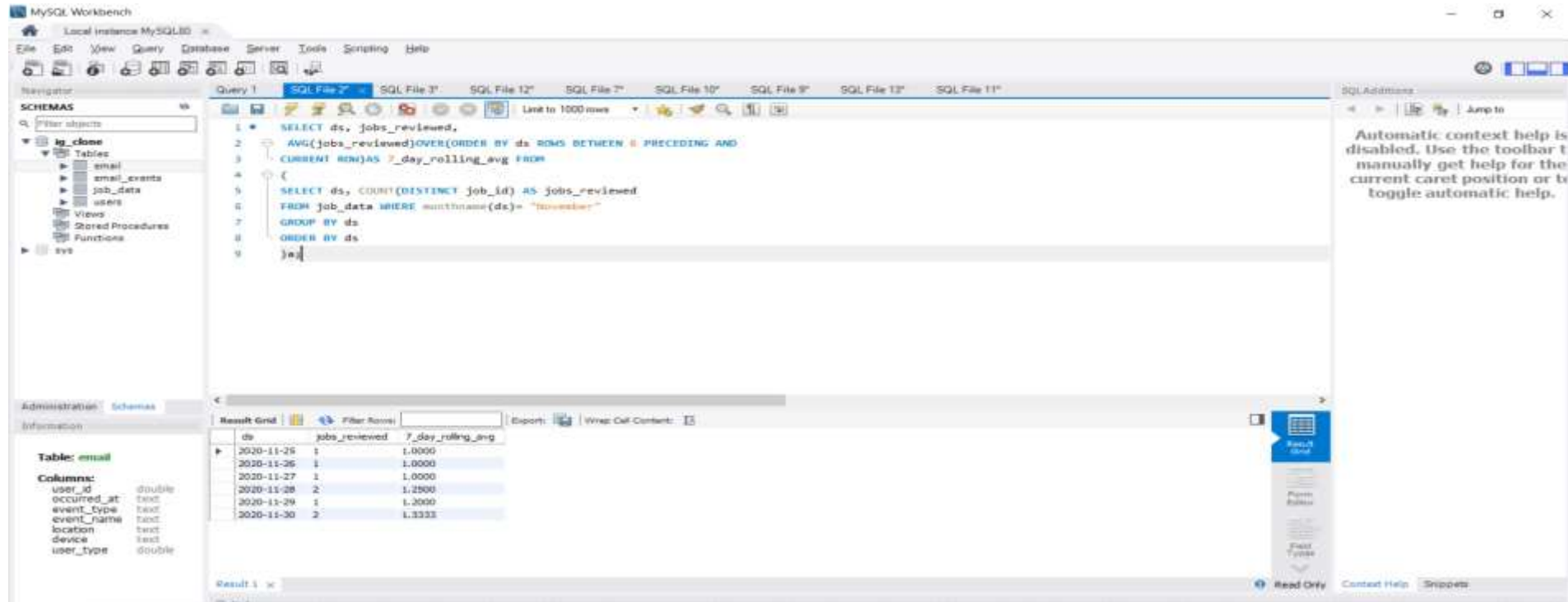
Your task: Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

QUERY:

```
SELECT ds, jobs_reviewed,  
       AVG(jobs_reviewed)OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND  
CURRENT ROW)AS 7_day_rolling_avg FROM  
(  
SELECT ds, COUNT(DISTINCT job_id) AS jobs_reviewed  
FROM job_data WHERE monthname(ds)= "November"  
GROUP BY ds  
ORDER BY ds  
)a;
```


TASKS FOR CASE STUDY-1

TASK -2



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the database structure, including tables like 'email', 'small_events', 'job_data', 'users', 'Views', 'Stored Procedures', and 'Functions'. The main editor window contains a SQL query for 'Query 1'.

```
1 SELECT ds, jobs_reviewed,  
2 AVG(jobs_reviewed)OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND  
3 CURRENT ROW)AS 7_day_rolling_avg FROM  
4 (  
5 SELECT ds, COUNT(DISTINCT job_id) AS jobs_reviewed  
6 FROM job_data WHERE monthname(ds)= "November"  
7 GROUP BY ds  
8 ORDER BY ds  
9 )as
```

The bottom panel shows the 'Result Grid' with the following data:

ds	jobs_reviewed	7_day_rolling_avg
2020-11-26	1	1.0000
2020-11-26	1	1.0000
2020-11-27	1	1.0000
2020-11-28	2	1.2500
2020-11-29	1	1.2000
2020-11-30	2	1.3333

The right sidebar shows the 'SQL Additions' panel with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

output: 7 day rolling is preferable for detail aspects as in each day most the events are not happening.

TASKS FOR CASE STUDY-1

TASK -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?

QUERY:

```
SELECT language,  
       count(language) as each_language_count,  
       (count(language)/(select count(*)from job_data)) * 100 as per_share_of_each_language  
FROM job_data  
group by language  
order by language;
```

TASKS FOR CASE STUDY-1

TASK -3

The screenshot displays the MySQL Workbench interface. The main window shows a SQL query in the editor, which is executed and the results are displayed in the Result Grid. The query calculates the count of each language and the percentage share of each language relative to the total number of records in the job_data table.

SQL Query:

```
1 SELECT language,
2     count(language) as each_language_count,
3     (count(language)/(select count(*) from job_data)) * 100 as per_share_of_each_language
4 FROM job_data
5 group by language
6 order by language;
```

Result Grid:

language	each_language_count	per_share_of_each_language
Arabic	1	12.5000
English	1	12.5000
French	1	12.5000
Hindi	1	12.5000
Italian	1	12.5000
Persian	3	37.5000

Table: email

Columns:

- user_id: double
- occurred_at: text
- event_type: text
- event_name: text
- location: text
- device: text
- user_type: double

SQL Additions:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

TASKS FOR CASE STUDY-1

TASK -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?

QUERY:

```
select *  
from (select *,row_number() over (partition by job_id) as row_num  
from jobdata)a  
where row_num>1;
```

TASKS FOR CASE STUDY-1

TASK -4

The screenshot displays the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar shows the 'SCHEMAS' tab with a tree view of databases, including 'my_clone' and 'sys'. The main editor window shows a SQL query in 'Query 1':

```
1 select *
2 from (select *,row_number() over (partition by job_id as row_num
3 from job_data)a
4 where row_num>1;
```

The bottom panel shows the 'Result Grid' with the following data:

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

The right sidebar contains the 'SQLAdditions' tab with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

TASKS FOR CASE STUDY-2

TASK -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?

QUERY:

```
select extract(week from occurred_at) as num_week,  
count(distinct user_id) as no_of_distinct_user  
from email  
group by num_week;
```

TASKS FOR CASE STUDY-2

TASK -1

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with 'lg_clone' selected, containing tables like 'email', 'email_events', 'job_data', 'users', 'Views', 'Stored Procedures', 'Functions', and 'sys'. The main editor window shows a SQL query in 'SQL File 7*':

```
1 select extract(week from occurred_at) as num_week,  
2 count(distinct user_id) as no_of_distinct_user  
3 from email  
4 group by num_week;
```

The 'Result Grid' at the bottom shows the query results:

num_week	no_of_distinct_user
17	85
18	194
19	208
20	45
21	12
22	7
23	3

The bottom right corner of the interface shows a 'SQLAdditions' panel with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

TASKS FOR CASE STUDY-2

TASK -2

User Growth: Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

QUERY:

```
select weeknum,year,num_active_user,  
SUM(num_active_user)over(order by year, weeknum rows between unbounded preceding and current row)as cum_active_users  
FROM  
(select  
extract(year from activated_at) as year,  
extract(week from activated_at)as weeknum,  
count(distinct user_id)as num_active_user  
FROM users  
WHERE state='active'  
group by weeknum,year  
order BY weeknum,year)a
```


TASKS FOR CASE STUDY-2

TASK -2

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view of the database structure, including tables like 'email', 'email_events', 'job_data', and 'users'. The main editor window shows a SQL query in 'SQL File 10'. The query is a window function that calculates cumulative active users over time.

```
1 select weeknum, year, num_active_user,  
2       SUM(num_active_user) over (order by year, weeknum rows between unbounded preceding and current row) as cum_active_users  
3 from  
4 (  
5   select  
6     extract(year from activated_at) as year,  
7     extract(week from activated_at) as weeknum,  
8     count(distinct user_id) as num_active_user  
9   from users  
10  where state = 'active'  
11  group by weeknum, year  
12  order by weeknum, year  
13 )
```

Below the query editor, the 'Result Grid' shows the output of the query. The grid has four columns: 'weeknum', 'year', 'num_active_user', and 'cum_active_users'. The data shows a steady increase in active users over the weeks of 2013.

weeknum	year	num_active_user	cum_active_users
4	2013	30	167
5	2013	48	215
6	2013	38	253
7	2013	42	295
8	2013	34	329
9	2013	43	372
10	2013	32	404
11	2013	31	435
12	2013	33	468
13	2013	39	507
14	2013	35	542
15	2013	43	585
16	2013	46	631
17	2013	49	680
18	2013	44	724

The right sidebar contains the 'SQL Additions' panel, which includes a 'Jump to' section and a note about automatic context help. The bottom status bar shows 'Read Only', 'Context Help', and 'Snippets' options.

TASKS FOR CASE STUDY-2

TASK -3

Weekly Retention: Users getting retained weekly after signing-up for a product.

Your task: Calculate the weekly retention of users-sign up cohort?

QUERY:

SELECT

COUNT(user_id)as users,

SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS week_1,

SUM(CASE WHEN retention_week = 2 THEN 1 ELSE 0 END) AS week_2,

SUM(CASE WHEN retention_week = 3 THEN 1 ELSE 0 END) AS week_3,

SUM(CASE WHEN retention_week = 4 THEN 1 ELSE 0 END) AS week_4,

SUM(CASE WHEN retention_week = 5 THEN 1 ELSE 0 END) AS week_5

FROM

(

TASKS FOR CASE STUDY-2

TASK -3

```
FROM (SELECT a.user_id,  
a.sign_up_week,  
b.engagement_week,  
b.engagement_week - a.sign_up_week as retention_week FROM (  
(select distinct user_id, extract(week from occurred_at) as sign_up_week  
from email  
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 email  
where event_type='engagement')b  
on a.user_id=b.user_id )  
order by  
a.user_id )a
```

TASKS FOR CASE STUDY-2

TASK -3

The screenshot displays the MySQL Workbench interface. The main window shows a SQL query in the 'Query Editor' tab. The query is a SELECT statement that counts users and calculates retention weeks based on sign-up and engagement dates. The 'Navigator' pane on the left shows the 'ig_clone' database with tables like 'email', 'email_events', 'job_data', and 'users'. The 'Result Grid' at the bottom shows the output of the query, which includes columns for 'users', 'week_1', 'week_2', 'week_3', 'week_4', and 'week_5'. The 'Table: email' information pane on the left lists columns and their data types. The 'SQL Additions' pane on the right contains a message about automatic context help.

```
1 SELECT
2 COUNT(user_id) as users,
3 SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END ) AS week_1,
4 SUM(CASE WHEN retention_week = 2 THEN 1 ELSE 0 END ) AS week_2,
5 SUM(CASE WHEN retention_week = 3 THEN 1 ELSE 0 END ) AS week_3,
6 SUM(CASE WHEN retention_week = 4 THEN 1 ELSE 0 END ) AS week_4,
7 SUM(CASE WHEN retention_week = 5 THEN 1 ELSE 0 END ) AS week_5
8 FROM
9 (
10 SELECT a.user_id,
11 a.sign_up_week,
12 b.engagement_week,
13 b.engagement_week - a.sign_up_week as retention_week
```

users	week_1	week_2	week_3	week_4	week_5
182	11	4	3	0	1

Table: email
Columns:
user_id double
occurred_at text
event_type text
event_name text
location text
device text
user_type double

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

TASKS FOR CASE STUDY-2

TASK -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?

QUERY:

```
select
extract(year from occurred_at)as year,
extract(week from occurred_at)as week,
device,
COUNT(distinct user_id) as num_users
FROM
email
WHERE
event_type='engagement'
group by 1,2,3
order by 1,2,3
```

TASKS FOR CASE STUDY-2

TASK -4

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with 'ig_clone' expanded, listing tables like 'email', 'email_events', 'job_data', and 'users'. The central editor window shows a SQL query for 'SQL File 13':

```
1 select
2   extract(year from occurred_at) as year,
3   extract(week from occurred_at) as week,
4   device,
5   COUNT(distinct user_id) as num_users
6 FROM
7   email
8 WHERE
9   event_type='engagement'
10 group by 1,2,3
11 order by 1,2,3
12
```

The bottom panel shows the 'Result Grid' with the following data:

year	week	device	num_users
2014	17	acer aspire desktop	2
2014	17	acer aspire notebook	2
2014	17	amazon fire phone	1
2014	17	asus chromebook	3
2014	17	dell inspiron desktop	1
2014	17	dell inspiron notebook	4
2014	17	hp pavilion desktop	2
2014	17	htc one	2
2014	17	ipad air	1
2014	17	ipad mini	3

The right sidebar contains a 'Context Help' window with the text: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

TASKS FOR CASE STUDY-2

TASK -5

Email Engagement: Users engaging with the email service.

Your task: Calculate the email engagement metrics?

QUERY:

select

100.0 *SUM(case when email_cat ='email_open' then 1 else 0 end)/SUM(case when email_cat='email_sent' then 1 else 0 end)as
email_opened_rate,

100.0* SUM(case when email_cat ='email_clicked' then 1 else 0 end)/SUM(case when email_cat='email_sent' then 1 else 0 end)as
email_clicked_rate

FROM (SELECT *,CASE

WHEN action in('sent_weekly_digest','sent_reenagagement_email')
then 'email_sent'

WHEN action in('email_open')
then 'email_open'

WHEN action in('email_clickthrough')
then 'email_clicked'

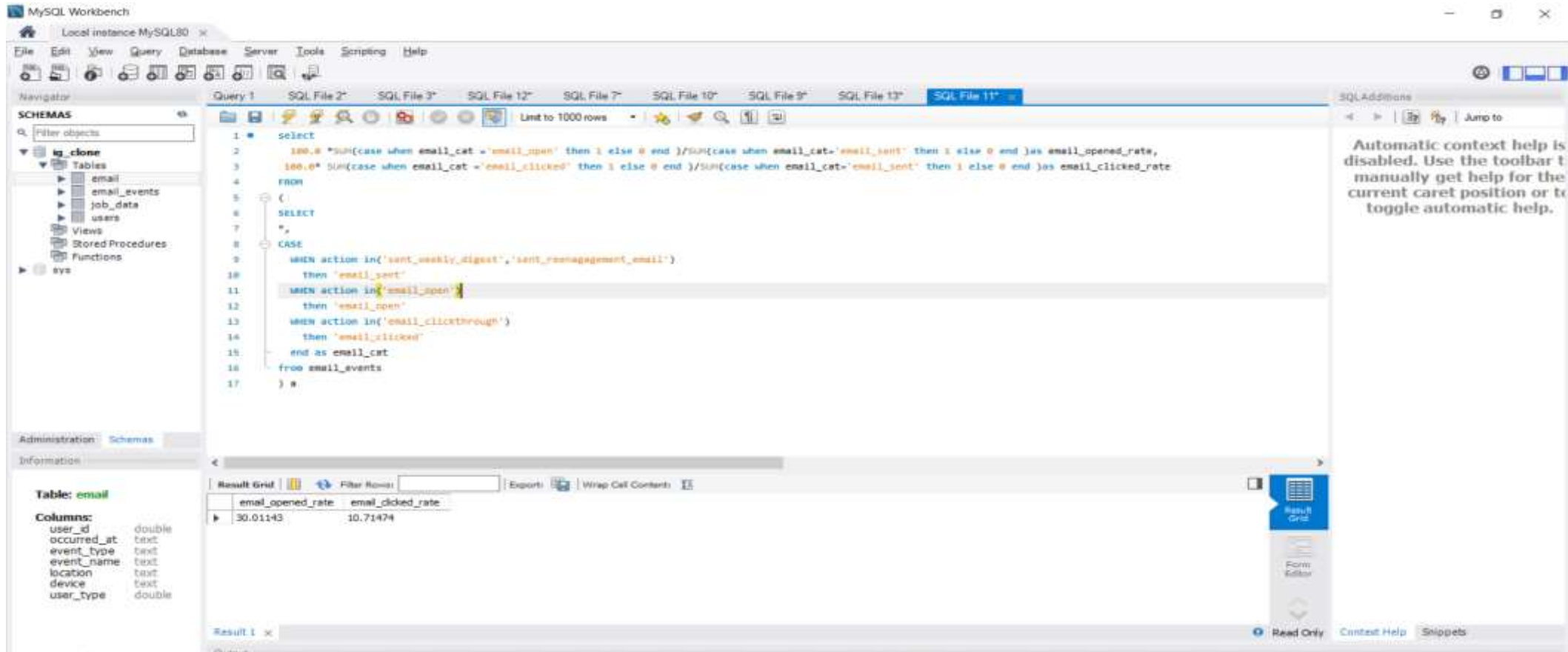
end as email_cat

from email_events

) a

TASKS FOR CASE STUDY-2

TASK -5



The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view of the database structure, including tables like 'email', 'email_events', 'job_data', and 'users'. The main editor window shows a SQL query (Query 1) that calculates email engagement rates. The query uses a CASE statement to categorize email actions into 'email_sent', 'email_open', and 'email_clicked'. The results are displayed in a table with two columns: 'email_opened_rate' and 'email_clicked_rate'. The values shown are 30.01143 and 10.71474 respectively. The bottom status bar indicates the query is in 'Read Only' mode.

```
1 select
2 100.0 * SUM(case when email_cat = 'email_open' then 1 else 0 end ) / SUM(case when email_cat = 'email_sent' then 1 else 0 end ) as email_opened_rate,
3 100.0 * SUM(case when email_cat = 'email_clicked' then 1 else 0 end ) / SUM(case when email_cat = 'email_sent' then 1 else 0 end ) as email_clicked_rate
4 from
5 (
6 select
7 *,
8 CASE
9   WHEN action in('sent_weekly_digest','sent_remanagement_email')
10    then 'email_sent'
11   WHEN action in('email_open')
12    then 'email_open'
13   WHEN action in('email_clickthrough')
14    then 'email_clicked'
15   end as email_cat
16 from email_events
17 ) #
```

email_opened_rate	email_clicked_rate
30.01143	10.71474

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

RESULT

We have learned different sql functions like :

- ❑ extract
- ❑ date
- ❑ avg
- ❑ over
- ❑ distinction
- ❑ and many more.