Operation Analytics and Investigating Metric Spike

BY AAYUSHI SINGH

Project 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.

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.

Approach

To analyze the data first we have to transfer our data which is excel(.csv) format to sql language.

The first dataset consist only 7 column and 8 rows so it easily can be uploaded manually but the second database for investing metric spike consist of more than 1900 rows which is impossible to upload manually
After uploading the data we analyze all the tasks and understand all the parameters needed to get your desired qualities

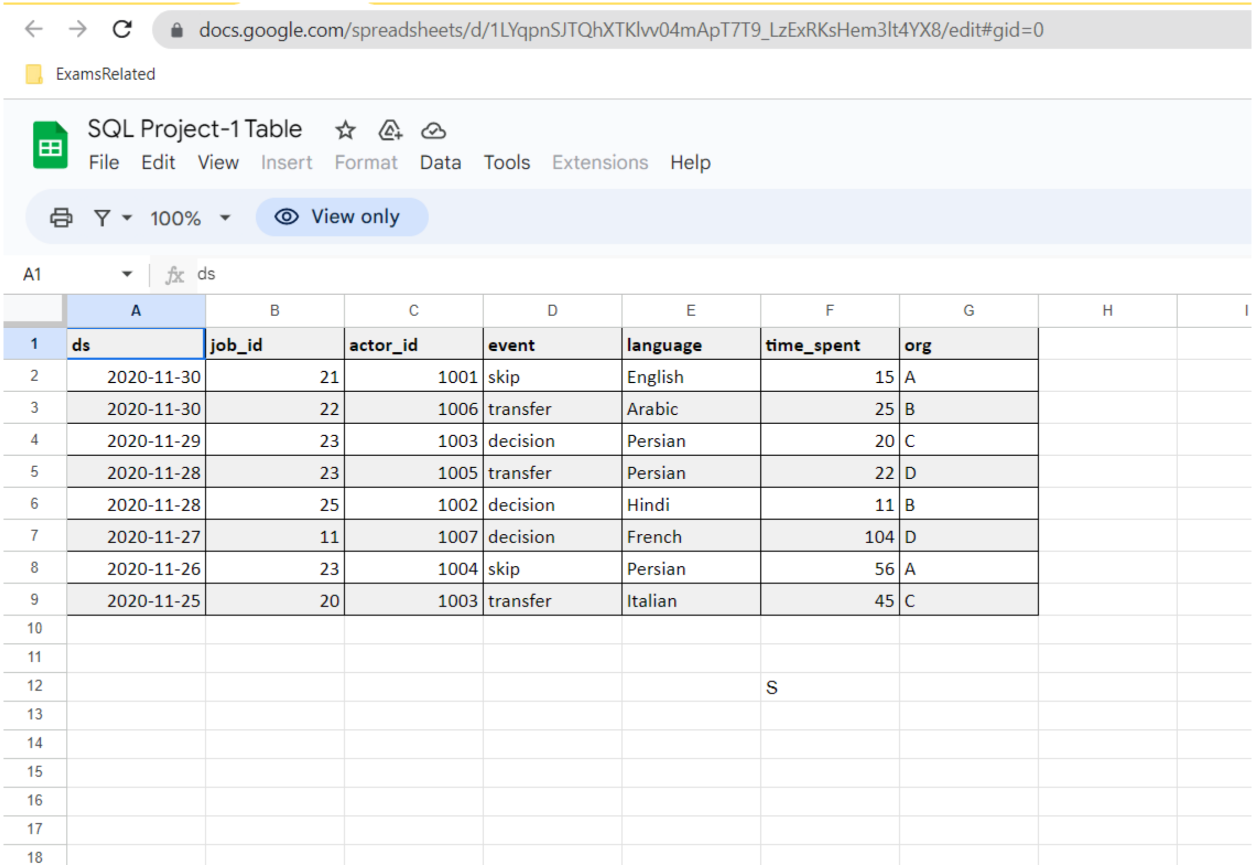
Tech Stack Used

My sql workbench
Excel

Results and Insights

Operational analytics

Dataset one

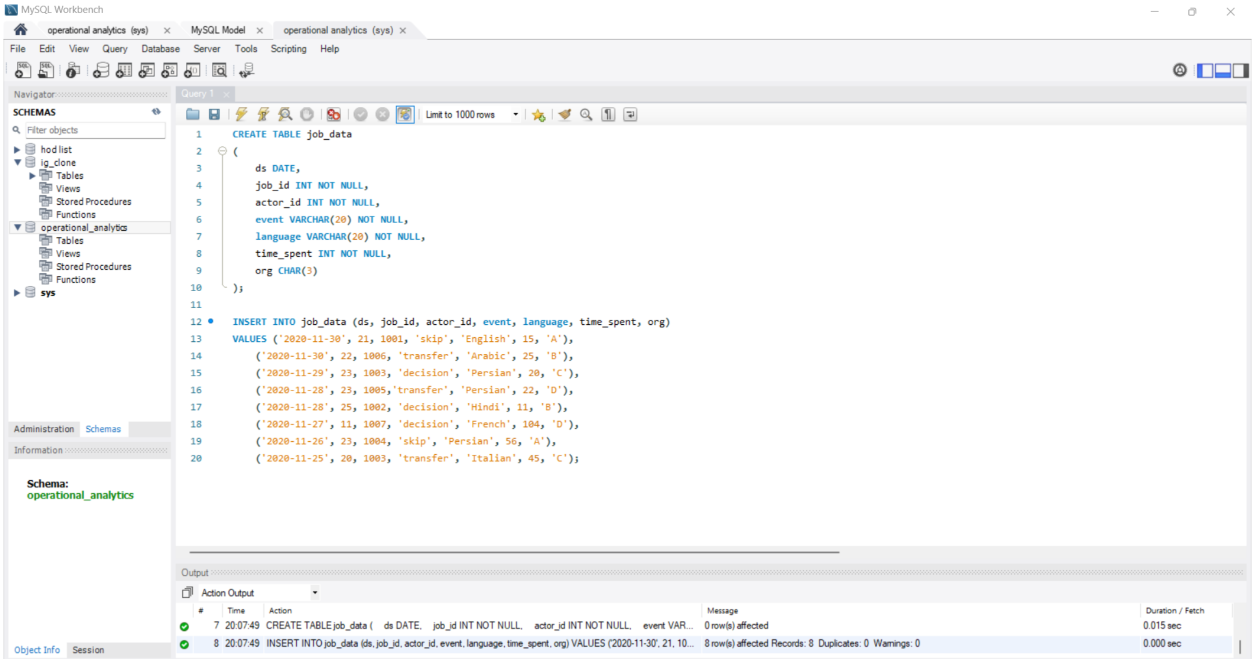


ds	job_id	actor_id	event	language	time_spent	org
2020-11-30	21	1001	skip	English	15	A
2020-11-30	22	1006	transfer	Arabic	25	B
2020-11-29	23	1003	decision	Persian	20	C
2020-11-28	23	1005	transfer	Persian	22	D
2020-11-28	25	1002	decision	Hindi	11	B
2020-11-27	11	1007	decision	French	104	D
2020-11-26	23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

Link to dataset
https://docs.google.com/spreadsheets/d/1LYqpnSJTQhXTKlvv04mApT7T9_LzExRKsHem3lt4YX8/edit#gid=0

- **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

We input our dataset into the Mysql workbench for the analysis of the data as for the purpose of creation of database



Schema:
operational_analytics

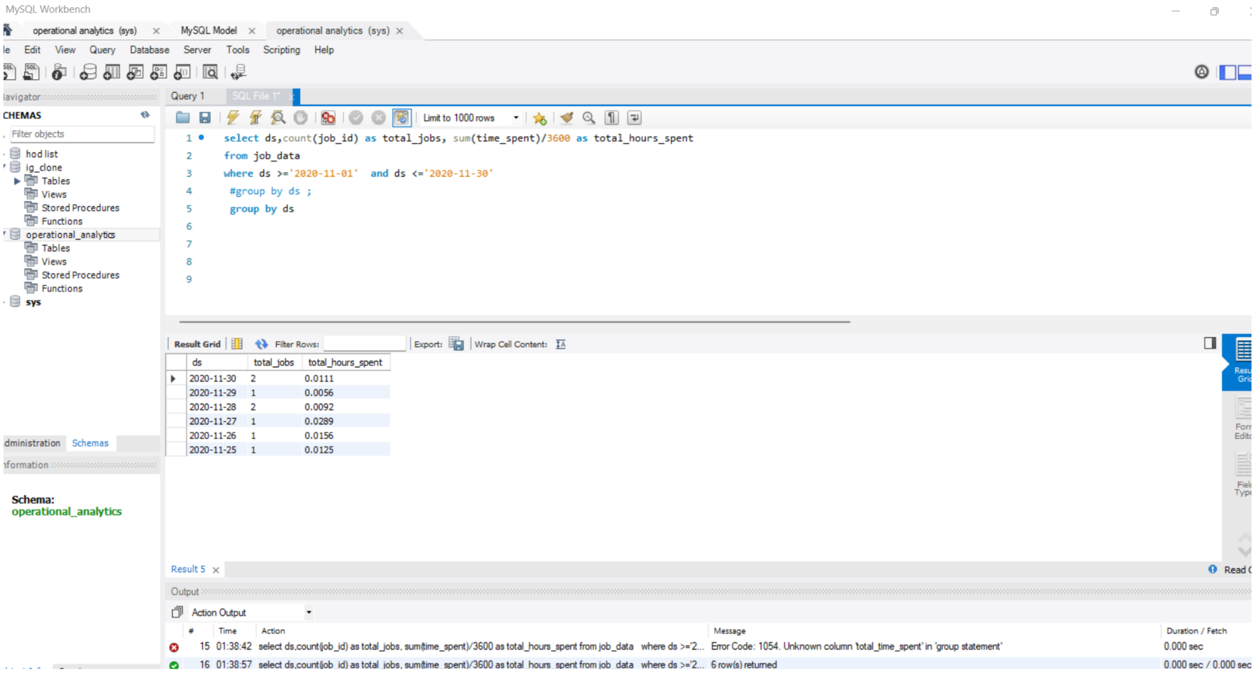
```sql
CREATE TABLE job_data
(
    ds DATE,
    job_id INT NOT NULL,
    actor_id INT NOT NULL,
    event VARCHAR(20) NOT NULL,
    language VARCHAR(20) NOT NULL,
    time_spent INT NOT NULL,
    org CHAR(3)
);

INSERT INTO job_data (ds, job_id, actor_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, 1003, 'transfer', 'Italian', 45, 'C');
```

```
  #  Time      Action                                                                                  Message                                                  Duration / Fetch
  7  20:07:49  CREATE TABLE job_data (   ds DATE,   job_id INT NOT NULL,   actor_id INT NOT NULL,   event VAR...  0 row(s) affected                                        0.015 sec
  8  20:07:49  INSERT INTO job_data (ds, job_id, actor_id, event, language, time_spent, org) VALUES ('2020-11-30', 21, 10...  8 row(s) affected Records: 8  Duplicates: 0  Warnings: 0  0.000 sec
```

TASKS

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



Schema:
operational_analytics

```sql
select ds,count(job_id) as total_jobs, sum(time_spent)/3600 as total_hours_spent
from job_data
where ds >='2020-11-01'  and ds <='2020-11-30'
 #group by ds ;
 group by ds
```

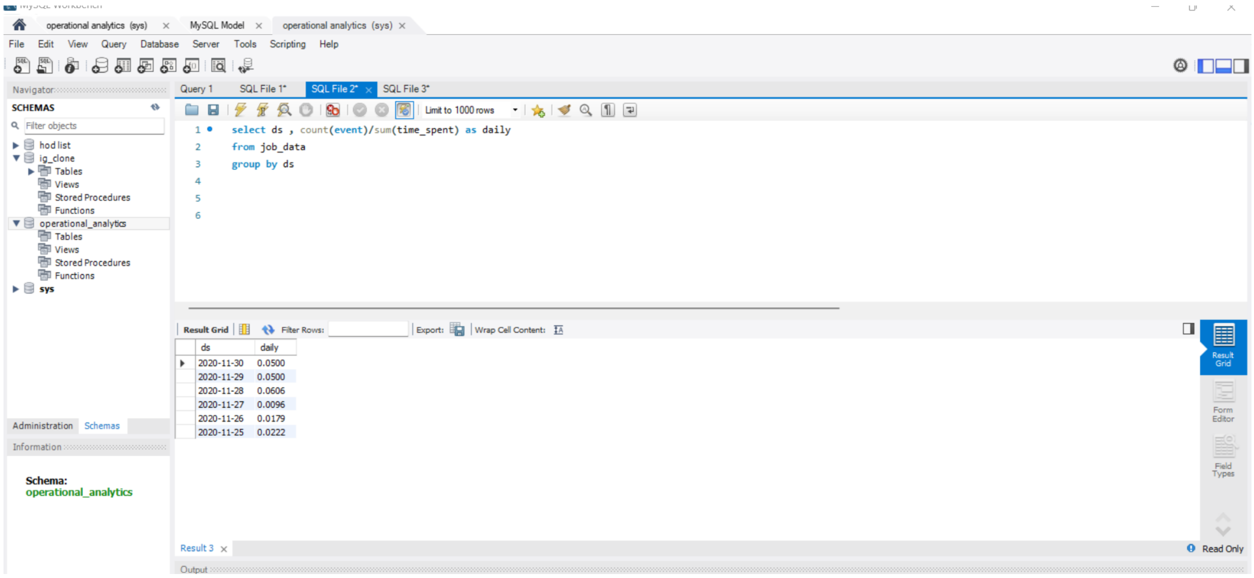
ds	total_jobs	total_hours_spent
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

```
#   Time      Action                                                                                          Message                                                                 Duration / Fetch
15  01:38:42  select ds,count(job_id) as total_jobs, sum(time_spent)/3600 as total_hours_spent from job_data  where ds >='2...  Error Code: 1054. Unknown column 'total_time_spent' in 'group statement'  0.000 sec
16  01:38:57  select ds,count(job_id) as total_jobs, sum(time_spent)/3600 as total_hours_spent from job_data  where ds >='2...  6 row(s) returned                                                       0.000 sec / 0.000 sec
```

Insight : To calculated the no of jobs per hour
we need to convert our data into hours(in data given in seconds) therefore divide it by 60*60=3600
(calculate total time spend in nov)

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?

We will first find the daily metric



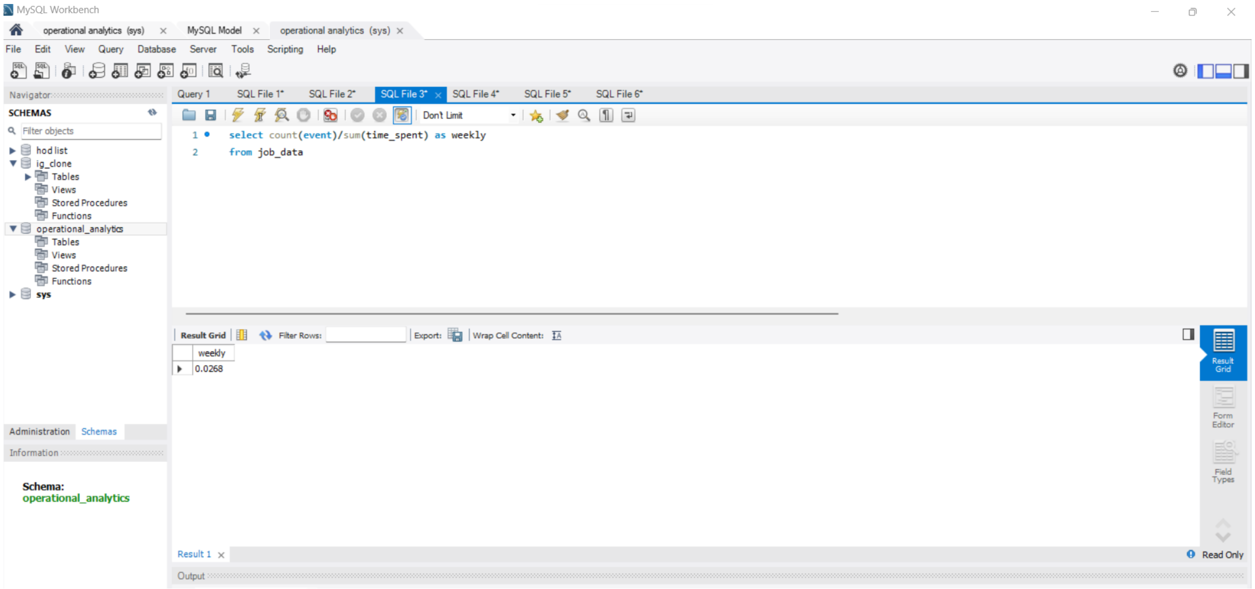
Schema: operational_analytics

```
select ds , count(event)/sum(time_spent) as daily
from job_data
group by ds
```

ds	daily
2020-11-30	0.0500
2020-11-29	0.0500
2020-11-28	0.0606
2020-11-27	0.0096
2020-11-26	0.0179
2020-11-25	0.0222

Insight: To take out the daily metric we have to individually take event of each day by its time spend therefore we use ds, to distribute and order the data as per the results

To find out average of weekly metric



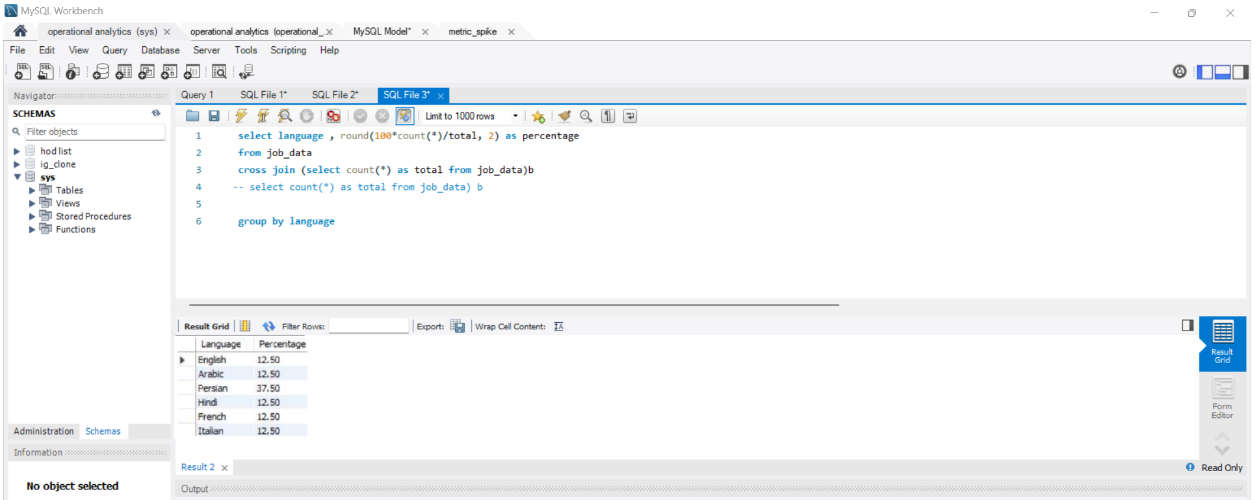
Schema: operational_analytics

```
select count(event)/sum(time_spent) as weekly
from job_data
```

weekly
0.0268

Insight: This we take the average on of the data as whole rather than individual day therefore simply by putting the average we can find the throughput of the week
If we want to deeply analyze the the development to understand trends or to make a monthly analysis then daily metric to understand the everyday metrics
If we want a vague week by week analysis for every month to understand weekly development or year end analysis then 7 day rolling to chart the charaterstics

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?



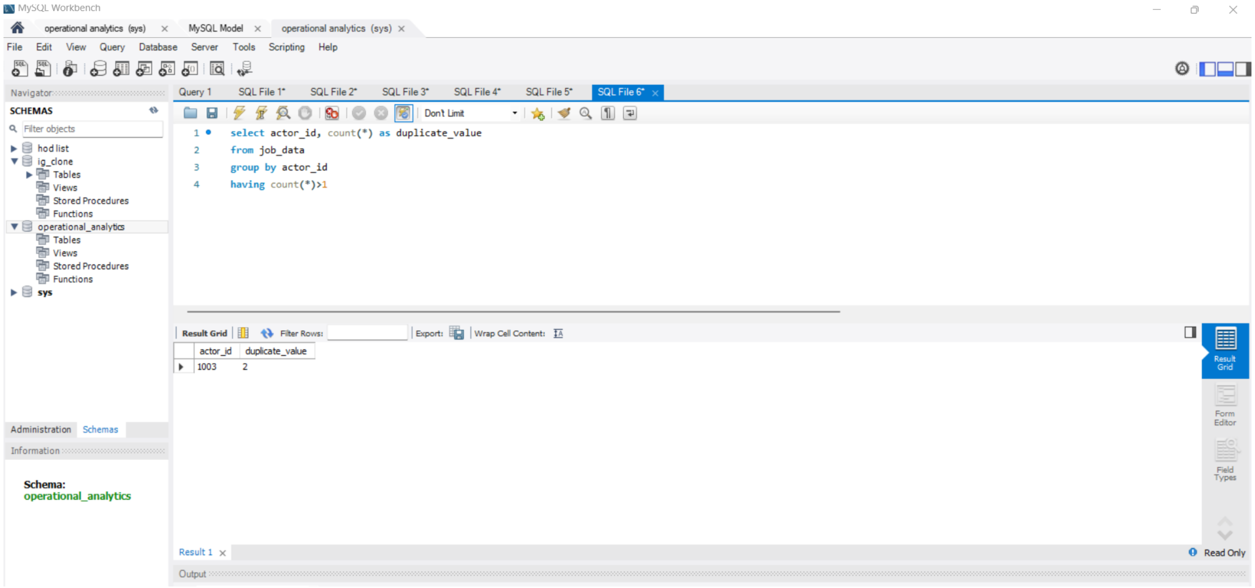
```sql
select language , round(100*count(*)/total, 2) as percentage
from job_data
cross join (select count(*) as total from job_data)b
-- select count(*) as total from job_data) b

group by language
```

Language	Percentage
English	12.50
Arabic	12.50
Persian	37.50
Hindi	12.50
French	12.50
Italian	12.50

Insight: To find the percentage of the table we will use self join and cross join to the table and as percentage are taken up to 2 decimal places we will round the resultantant value

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?



Schema:
operational_analytics

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

```
actor_id  duplicate_value
1003      2
```

Investigating metric spike

Dataset

Table-1: users
This table includes one row per user, with descriptive information about that user’s account

https://drive.google.com/file/d/104VAIgSUXtRtqxEKULCD2_5Vh5LCoNHG/view?usp=share_link

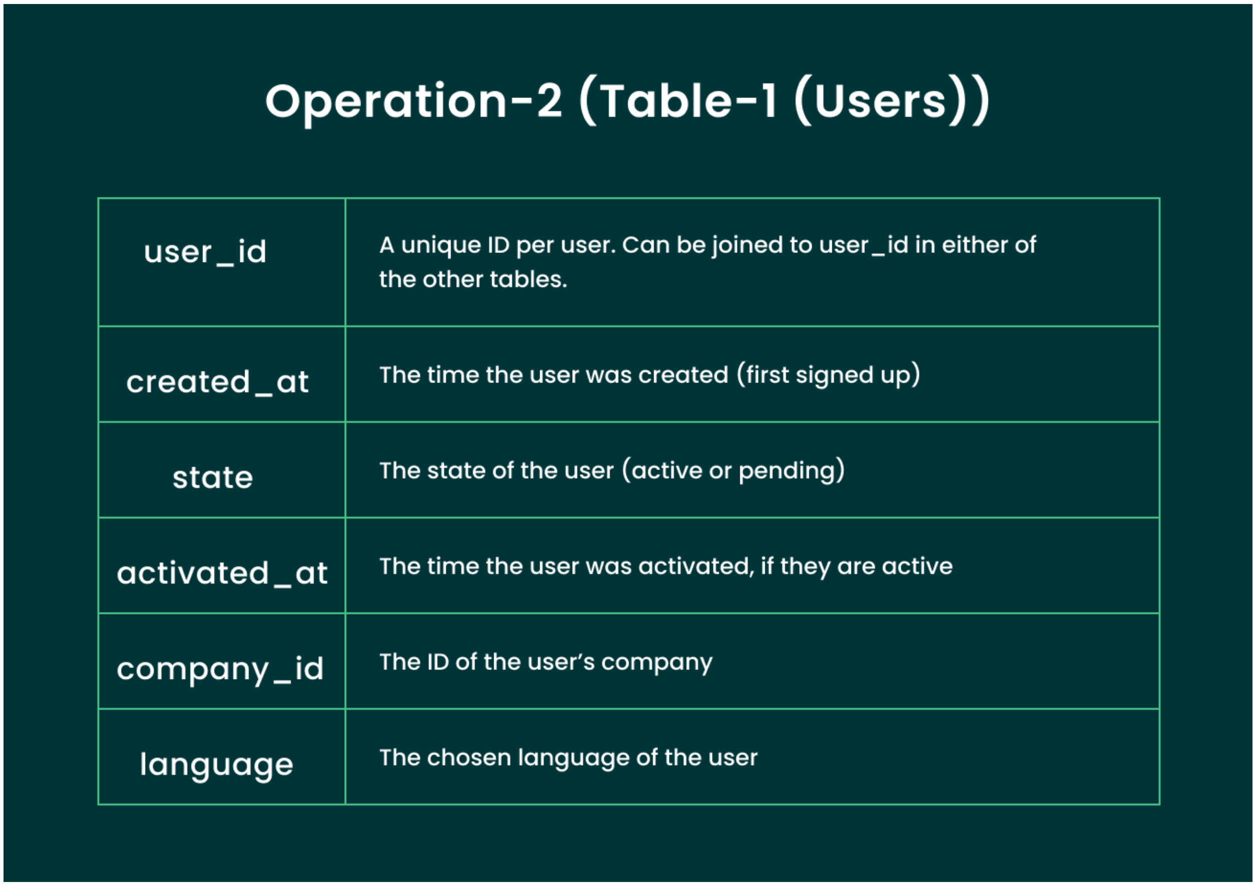
Table-2: events
This table includes one row per event, where an event is an action that a user has taken. These events include login events, messaging events, search events, events logged as users progress through a signup funnel, events around received emails

https://drive.google.com/file/d/1qMHK5ZQ0vVoMjGz31K163LiTF-Fk39Ab/view?usp=share_link

Table-3: email_events
This table contains events specific to the sending of emails. It is similar in structure to the events table above.

https://drive.google.com/file/d/1OBQ83KtPUtI5LfogubcwjgvTUgtQUO4l/view?usp=share_link

The tables consist of

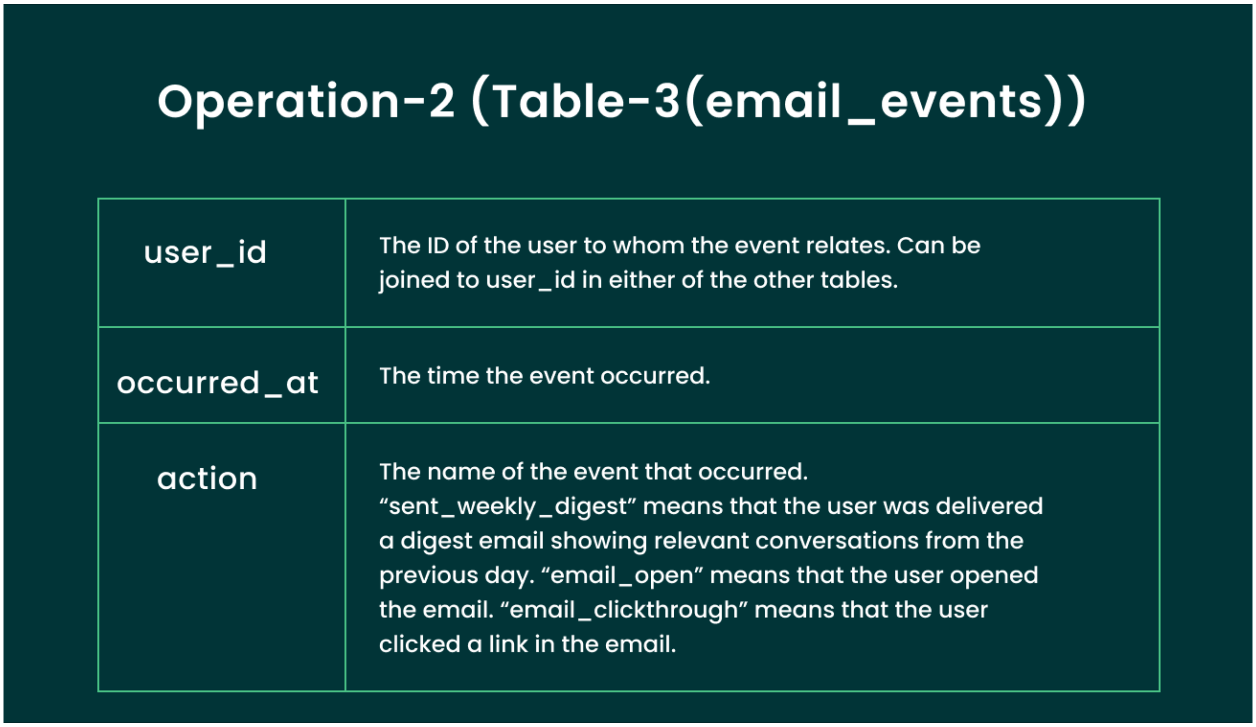


Operation-2 (Table-1 (Users))

user_id	A unique ID per user. Can be joined to user_id in either of the other tables.
created_at	The time the user was created (first signed up)
state	The state of the user (active or pending)
activated_at	The time the user was activated, if they are active
company_id	The ID of the user’s company
language	The chosen language of the user

Operation-2 (Table-2(events))

user_id	The ID of the user logging the event. Can be joined to user_id in either of the other tables.
occurred_at	The time the event occurred.
event_type	The general event type. There are two values in this dataset: “signup_flow”, which refers to anything occuring during the process of a user’s authentication, and “engagement”, which refers to general product usage after the user has signed up for the first time
event_name	The specific action the user took. Possible values include: create_user: User is added to Yammer’s database during signup process enter_email: User begins the signup process by entering her email address enter_info: User enters her name and personal information during signup process complete_signup: User completes the entire signup/ authentication process home_page: User loads the home page like_message: User likes another user’s message login: User logs into Yammer search_autocomplete: User selects a search result from the autocomplete list search_run: User runs a search query and is taken to the search results page search_click_result_X: User clicks search result X on the results page, where X is a number from 1 through 10. send_message: User posts a message view_inbox: User views messages in her inbox
location:	The country from which the event was logged (collected through IP address).
device:	The type of device used to log the event.



Operation-2 (Table-3(email_events))

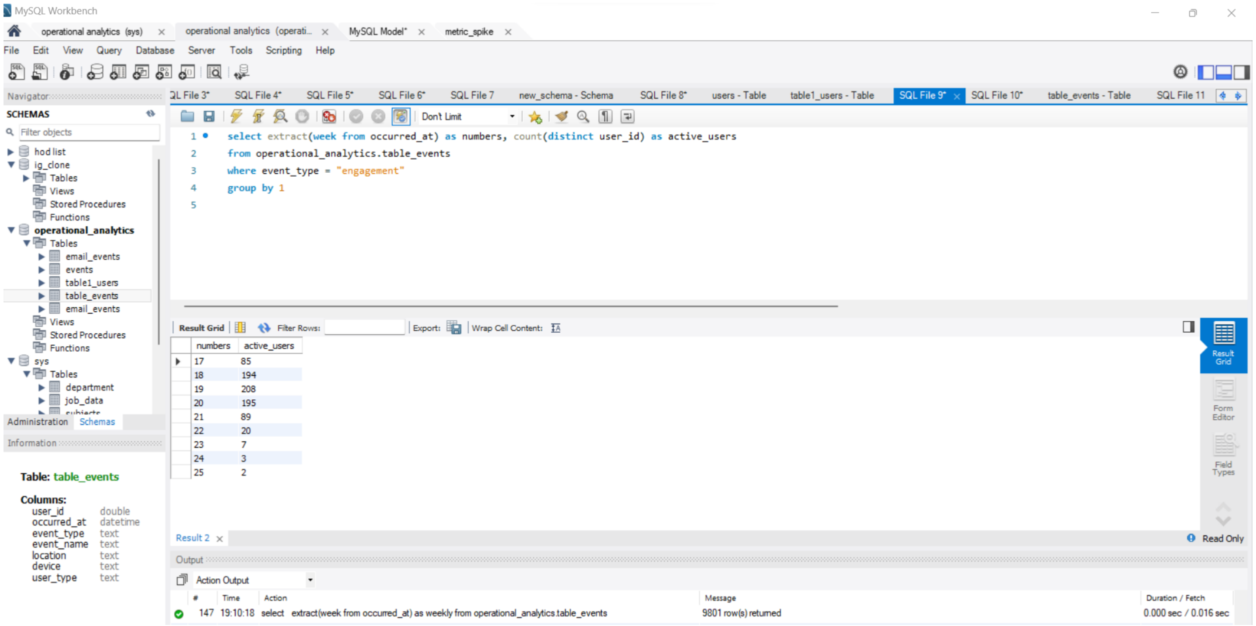
user_id	The ID of the user to whom the event relates. Can be joined to user_id in either of the other tables.
occurred_at	The time the event occurred.
action	The name of the event that occurred. “sent_weekly_digest” means that the user was delivered a digest email showing relevant conversations from the previous day. “email_open” means that the user opened the email. “email_clickthrough” means that the user clicked a link in the email.

Steps to upload data on sql

Go to your mysql workbench

- Click on the schema where you want to upload the tables on, right click on the table and click on “table data import wizard” and import your data
- In the event table while importing the data change the event_type from int to text to successfully import your data or else it will lead to data(as observed in import log)

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



```
select extract(week from occurred_at) as numbers, count(distinct user_id) as active_users
from operational_analytics.table_events
where event_type = "engagement"
group by 1
```

numbers	active_users
17	85
18	194
19	208
20	195
21	89
22	20
23	7
24	3
25	2

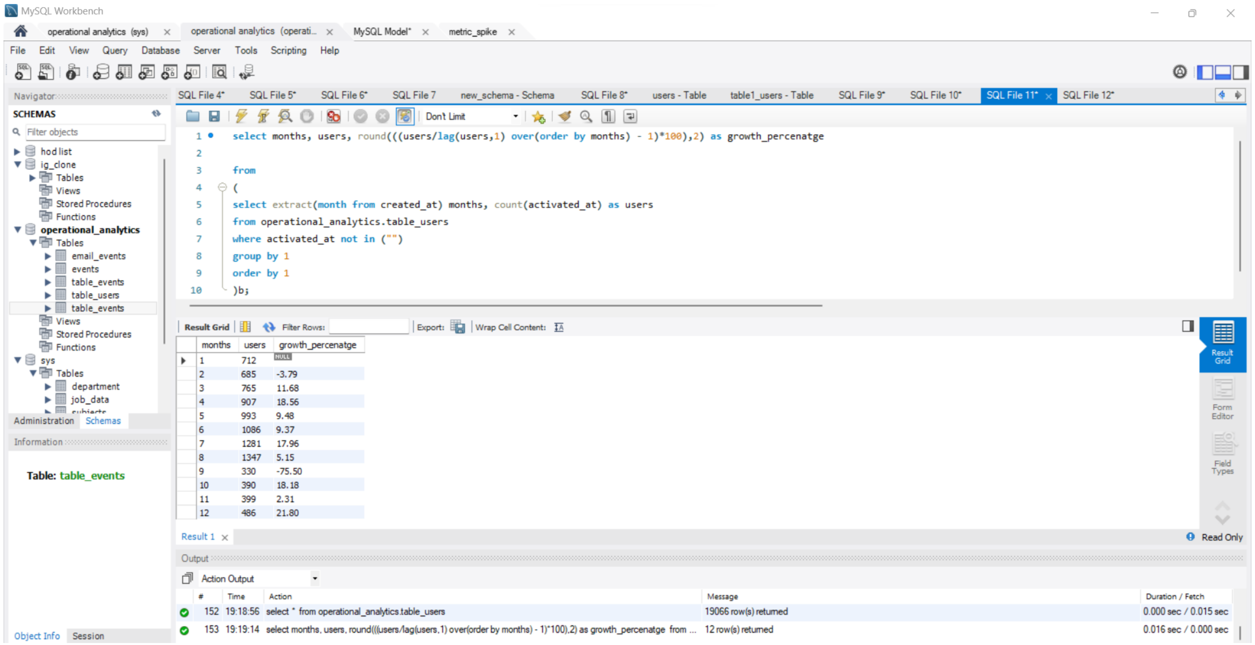
Table: table_events

Columns:

user_id	double
occurred_at	datetime
event_type	text
event_name	text
location	text
device	text
user_type	text

```
#    Time      Action                                                                             Message              Duration / Fetch
147  19:10:18  select  extract(week from occurred_at) as weekly from operational_analytics.table_events  9801 row(s) returned  0.000 sec / 0.016 sec
```

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



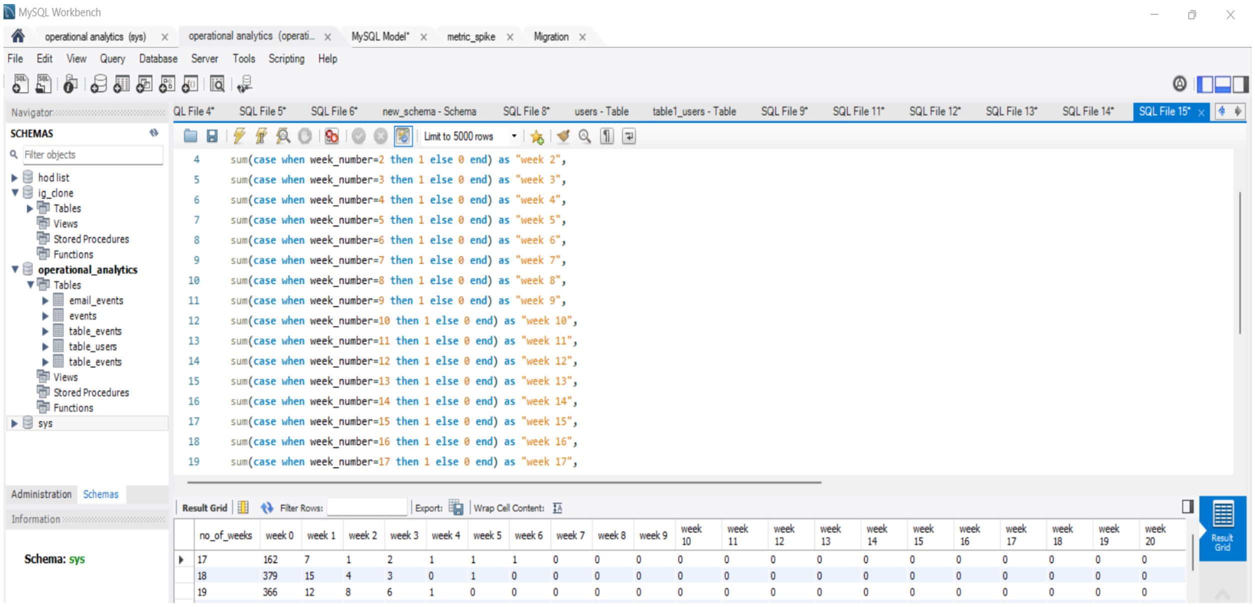
```
select months, users, round(((users/lag(users,1) over(order by months) - 1)*100),2) as growth_percenatge

from
(
select extract(month from created_at) months, count(activated_at) as users
from operational_analytics.table_users
where activated_at not in ("")
group by 1
order by 1
)b;
```

months	users	growth_percenatge
1	712	NULL
2	685	-3.79
3	765	11.68
4	907	18.56
5	993	9.48
6	1086	9.37
7	1281	17.96
8	1347	5.15
9	330	-75.50
10	390	18.18
11	399	2.31
12	486	21.80

```
#    Time      Action                                                                                                  Message               Duration / Fetch
152  19:18:56  select * from operational_analytics.table_users                                                          19066 row(s) returned  0.000 sec / 0.015 sec
153  19:19:14  select months, users, round(((users/lag(users,1) over(order by months) - 1)*100),2) as growth_percenatge  from ...  12 row(s) returned  0.016 sec / 0.000 sec
```

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

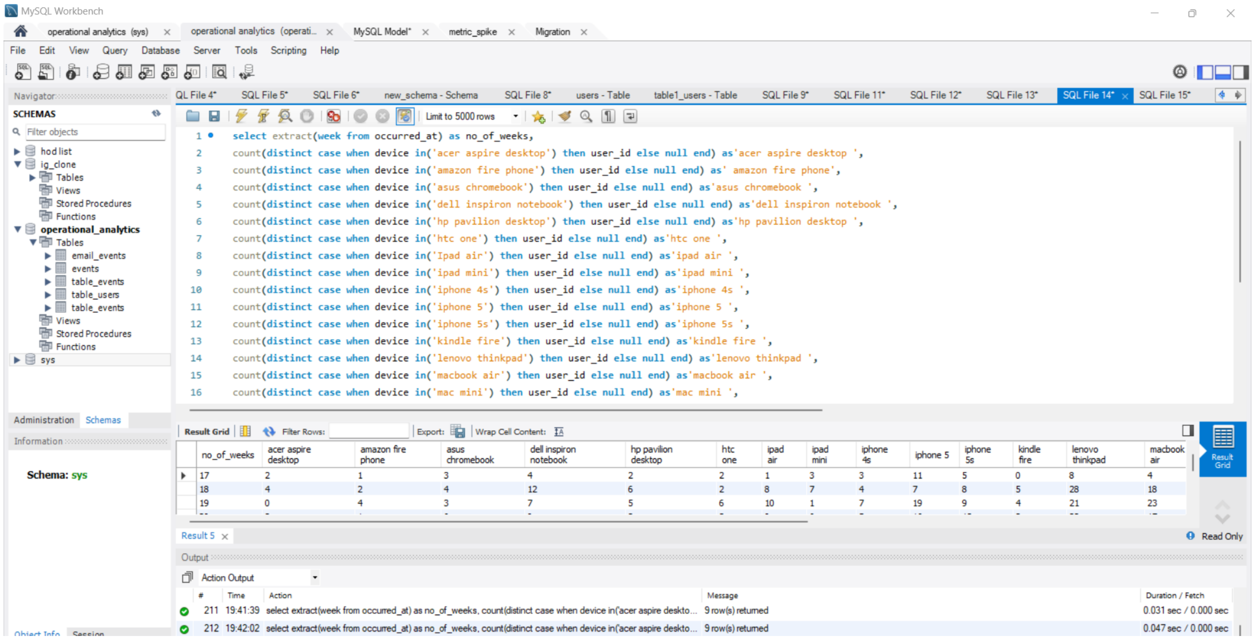


```
sum(case when week_number=2 then 1 else 0 end) as "week 2",
sum(case when week_number=3 then 1 else 0 end) as "week 3",
sum(case when week_number=4 then 1 else 0 end) as "week 4",
sum(case when week_number=5 then 1 else 0 end) as "week 5",
sum(case when week_number=6 then 1 else 0 end) as "week 6",
sum(case when week_number=7 then 1 else 0 end) as "week 7",
sum(case when week_number=8 then 1 else 0 end) as "week 8",
sum(case when week_number=9 then 1 else 0 end) as "week 9",
sum(case when week_number=10 then 1 else 0 end) as "week 10",
sum(case when week_number=11 then 1 else 0 end) as "week 11",
sum(case when week_number=12 then 1 else 0 end) as "week 12",
sum(case when week_number=13 then 1 else 0 end) as "week 13",
sum(case when week_number=14 then 1 else 0 end) as "week 14",
sum(case when week_number=15 then 1 else 0 end) as "week 15",
sum(case when week_number=16 then 1 else 0 end) as "week 16",
sum(case when week_number=17 then 1 else 0 end) as "week 17",
```

no_of_weeks	week 0	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14	week 15	week 16	week 17	week 18	week 19	week 20
17	162	7	1	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	379	15	4	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	366	12	8	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Insight: to analyse the user retention we have individually break in each week to see no user retained

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?



```sql
select extract(week from occurred_at) as no_of_weeks,
count(distinct case when device in('acer aspire desktop') then user_id else null end) as'acer aspire desktop ',
count(distinct case when device in('amazon fire phone') then user_id else null end) as' amazon fire phone',
count(distinct case when device in('asus chromebook') then user_id else null end) as'asus chromebook ',
count(distinct case when device in('dell inspiron notebook') then user_id else null end) as'dell inspiron notebook ',
count(distinct case when device in('hp pavilion desktop') then user_id else null end) as'hp pavilion desktop ',
count(distinct case when device in('htc one') then user_id else null end) as'htc one ',
count(distinct case when device in('Ipad air') then user_id else null end) as'ipad air ',
count(distinct case when device in('ipad mini') then user_id else null end) as'ipad mini ',
count(distinct case when device in('iphone 4s') then user_id else null end) as'iphone 4s ',
count(distinct case when device in('iphone 5') then user_id else null end) as'iphone 5 ',
count(distinct case when device in('iphone 5s') then user_id else null end) as'iphone 5s ',
count(distinct case when device in('kindle fire') then user_id else null end) as'kindle fire ',
count(distinct case when device in('lenovo thinkpad') then user_id else null end) as'lenovo thinkpad ',
count(distinct case when device in('macbook air') then user_id else null end) as'macbook air ',
count(distinct case when device in('mac mini') then user_id else null end) as'mac mini ',
```

no_of_weeks	acer aspire desktop	amazon fire phone	asus chromebook	dell inspiron notebook	hp pavilion desktop	htc one	ipad air	ipad mini	iphone 4s	iphone 5	iphone 5s	kindle fire	lenovo thinkpad	macbook air
17	2	1	3	4	2	2	1	3	3	11	5	0	8	4
18	4	2	4	12	6	2	8	7	4	7	8	5	28	18
19	0	4	3	7	5	6	10	1	7	19	9	4	21	23

```
#    Time      Action                                                                                          Message             Duration / Fetch
211  19:41:39  select extract(week from occurred_at) as no_of_weeks, count(distinct case when device in('acer aspire deskto...  9 row(s) returned  0.031 sec / 0.000 sec
212  19:42:02  select extract(week from occurred_at) as no_of_weeks, count(distinct case when device in('acer aspire deskto...  9 row(s) returned  0.047 sec / 0.000 sec
```

Insight: to analyze on what are the devices in the device column use excel filter and sort to understand the devices, based on excel analysis the devices present were

acer aspire desktop

amazon fire phone

asus chromebook

dell inspiron notebook

hp pavilion desktop

htc one

ipad air

Ipad mini

iphone 4s

iphone 5

Iphone 5s

kindle fire

lenovo thinkpad

macbook air

mac mini

Macbook pro

nexus 10

nexus 5

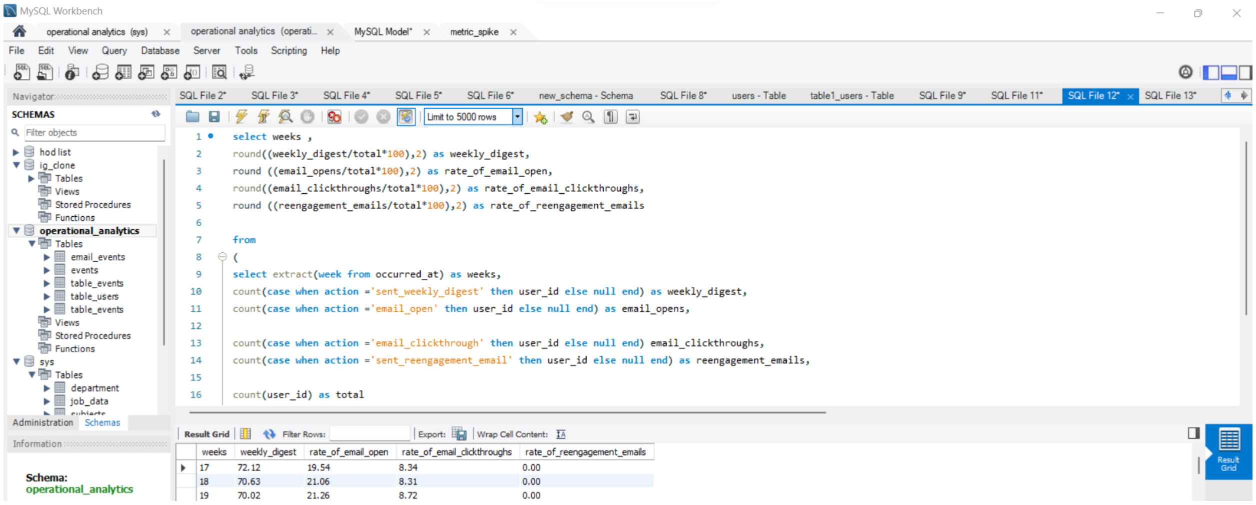
nokia lumia 635

samsung galaxy note

samsung galaxy s4

windows surface

5. Email Engagement: Users engaging with the email service.
Your task: Calculate the email engagement metrics?



```sql
select weeks ,
round((weekly_digest/total*100),2) as weekly_digest,
round ((email_opens/total*100),2) as rate_of_email_open,
round((email_clickthroughs/total*100),2) as rate_of_email_clickthroughs,
round ((reengagement_emails/total*100),2) as rate_of_reengagement_emails

from
(
select extract(week from occurred_at) as weeks,
count(case when action ='sent_weekly_digest' then user_id else null end) as weekly_digest,
count(case when action ='email_open' then user_id else null end) as email_opens,

count(case when action ='email_clickthrough' then user_id else null end) email_clickthroughs,
count(case when action ='sent_reengagement_email' then user_id else null end) as reengagement_emails,

count(user_id) as total
```

weeks	weekly_digest	rate_of_email_open	rate_of_email_clickthroughs	rate_of_reengagement_emails
17	72.12	19.54	8.34	0.00
18	70.63	21.06	8.31	0.00
19	70.02	21.26	8.72	0.00

Insight: For measuring engagement metrics we analyze the action table in the email_events table through excel we see that only four actions are possible

Sent_weekly_digest'

Email_open

Email_clickthrough

Sent_reengagement_email