Operation Analytics and Investigating Metric Spike

<u>Table of content</u>

1. Introduction

Case Study 1: Job Data

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. For this project, perform different tasks and get the output that helps to provide insights to the team and helps the business to grow also the operational analysis further used to predict overall growth or decline of a companies fortune. In this projects we have to do some SQL queries for several tasks like find number of job reviewed, find throughput, percentage share of each language also find the duplicate rows and provide insights to teams.

Case study 2: Investigating metric spike

Second case study is about investigating metric spike. It is very important part of operation analysis as being a data analyst we must able to understand and answerable to the questions regarding daily engagement of users, dip of sales and many other questions. These questions must be answered daily for that need of investigation metrics is important. Understanding these areas give you a proper visualisation of your business requirements and its growth. In this project we have provided with three tables, users table, events table and email events table. Those table provide users engagements, we have to use these information to answer those questions like Calculate the weekly user engagement?, Calculate the user growth for product?, Calculate the weekly retention of users-sign up cohort? Etc. answering those questions will helps to the business improve their growth and maintain a better workflow.

2. Tech-stack used

In this project I have used:

- My SQL work bench 8.0 CE
- SQL

The purpose for I used My SQL woke bench software in this project to create and store database and also used for running SQL queries.

I used SQL for retrieve data from the database for a given tasks.

3. Approach

I used MySQL for implementing this project. Firstly I have to create tables, so I used MySQL work bench for creating those tables. Analyzed each table and its attribute also checked the connection with other tables. From each table found the primary key and foreign key for better understanding to use join in SQL. From the above information created entity diagram for the better visibility of the data base

4. SQL queries

4.1 Case study 1

4.1.1 Database creation

```
CREATE DATABASE  case_study2;
```

4.1.2 Table creation

Users table:

```
create table users(
user_id int primary key,
created_at TIMESTAMP DEFAULT NOW(),
company_id int,
language varchar(255),
activated_at varchar(255),
state varchar(255)
);
```

4.1.3 Steps to import data

- Select database case_study from schema
- Select users table from the database
- Right click on the user table then select ‘table data import wizard’
- Then select the path of the csv folder then click next
- Then select ‘use existing table’, then click next
- Check column names, then click next
- After that click next to import data

A. **Number of jobs reviewed:**

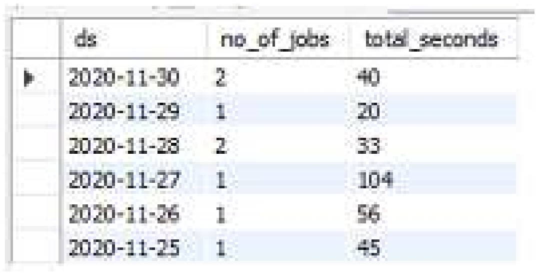
Your task: Calculate the number of jobs reviewed per hour per day for November 2020?

SQL queries 1:

```
/*Calculate the number of jobs reviewed per hour per day for November 2020?*/;

select ds, count(job_id) as no_of_jobs,sum(time_spend) as total_seconds
 from job_data
 group by ds
 having sum(time_spend) <= 3600;
```

output:



ds	no_of_jobs	total_seconds
2020-11-30	2	40
2020-11-29	1	20
2020-11-28	2	33
2020-11-27	1	104
2020-11-26	1	56
2020-11-25	1	45

Insights:

- Above output shows the number of jobs reviewed per day
- On November 30 2 jobs are reviewed total time taken is 40 sec. the next day 1 job reviewed then the next 2 jobs reviewed
- 104 is the longest time taken for review a job that is on November 27th

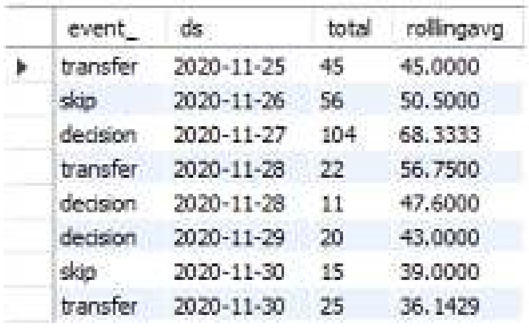
B. **Throughput:**

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?

SQL queries 2:

```
select a.* ,
 avg(total) over( order by ds rows between 6 preceding and current row) as rolllingavg
 from(
         select event_, ds, sum(time_spend) as total from job_data group by event_,ds) as a
```

output:



	event_	ds	total	rollingavg
▶	transfer	2020-11-25	45	45.0000
	skip	2020-11-26	56	50.5000
	decision	2020-11-27	104	68.3333
	transfer	2020-11-28	22	56.7500
	decision	2020-11-28	11	47.6000
	decision	2020-11-29	20	43.0000
	skip	2020-11-30	15	39.0000
	transfer	2020-11-30	25	36.1429

Insights:

- From the above output that shows the 7day rolling average of throughput.
- Here I prefer 7 day rolling average, because 7 day rolling average is a short term trend indicator. It tells you the average value moves over a week.

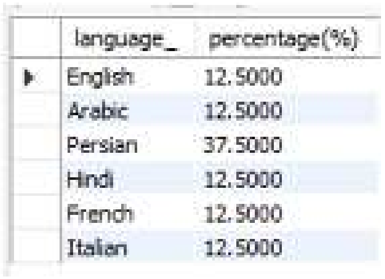
C. Percentage share of each language:

Your task: Calculate the percentage share of each language in the last 30 days?

SQL queries 3:

```
select b.language_, b.times * 100/(select sum(b.times) from (select language_, count(ds) as times
from job_data
group by language_) as b)'percentage(%)'
from (select language_, count(ds) as times
from job_data
group by language_) as b
```

output:



	language_	percentage(%)
▶	English	12.5000
	Arabic	12.5000
	Persian	37.5000
	Hindi	12.5000
	French	12.5000
	Italian	12.5000

Insights:

- From the above output we can see that percentage share of each language every language has 12.5% shares except Persian (37.5)

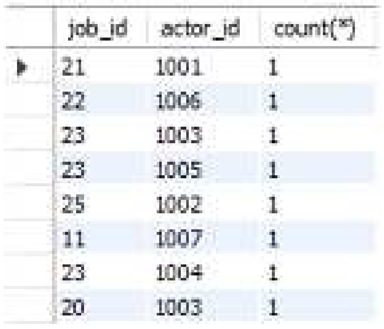
D. **Duplicate rows:**

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

SQL queries 4:

```
select job_id, actor_id, count(*) from job_data
group by job_id,actor_id
```

output:



job_id	actor_id	count(*)
21	1001	1
22	1006	1
23	1003	1
23	1005	1
25	1002	1
11	1007	1
23	1004	1
20	1003	1

Insights:

- ➢ Here from the above output shows that there are no duplicate rows in the table.
- ➢ Here job id and actor id columns has repeated values but its not a duplicated rows

4.2 Case study 2

4.2.1 Database creation:

```
Create database case_study2;
```

4.2.2 Table creation:

Users:

```
create table users(
user_id int primary key,
created_at TIMESTAMP DEFAULT NOW(),
company_id int,
language varchar(255),
activated_at varchar(255),
state varchar(255)
);
```

Events:

```
create table events_( user_id int,
occurred_at timestamp default now(),
event_type varchar(255),
event_name varchar(255),
location varchar(255),
device varchar(255),
user_type varchar(255),
foreign key (user_id) references users(user_id)
);
```

Email events:

```
create table email_events(
user_id int,
occurred_at timestamp default now(),
action_ varchar(255),
user_type int,
foreign key (user_id) references users(user_id)
);
```

Queries for inserting data:

```
load data infile 'file path' into table ‘table name’
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows;
```

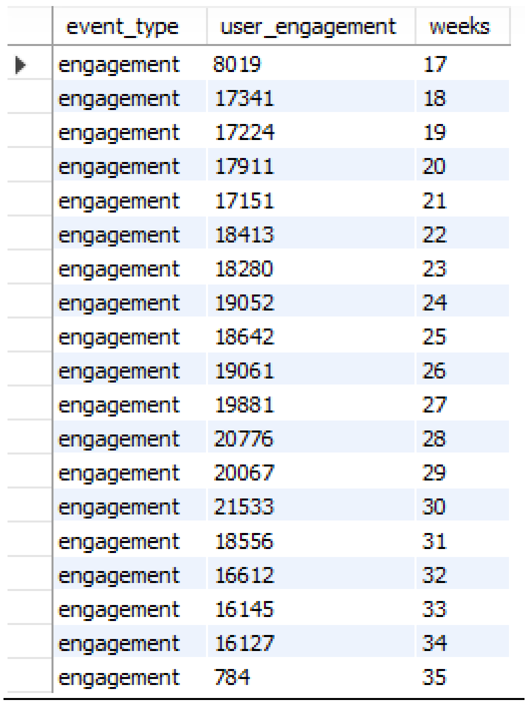
A. User Engagement:

Your task: Calculate the weekly user engagement?

SQL queries 1:

```
select  event_type, count(user_id) as user_engagement,extract(week from occurred_at) as weeks
from events_ where event_type = 'engagement'
group by event_type, weeks
order by weeks
```

output:



```
  event_type   user_engagement   weeks
▶ engagement   8019              17
  engagement   17341             18
  engagement   17224             19
  engagement   17911             20
  engagement   17151             21
  engagement   18413             22
  engagement   18280             23
  engagement   19052             24
  engagement   18642             25
  engagement   19061             26
  engagement   19881             27
  engagement   20776             28
  engagement   20067             29
  engagement   21533             30
  engagement   18556             31
  engagement   16612             32
  engagement   16145             33
  engagement   16127             34
  engagement   784               35
```

Insights:

- The output shows the weekly engagement of users. Here from week 17 to 30 user engagement are getting increased and from 31 to 35 users engagement are getting decreased

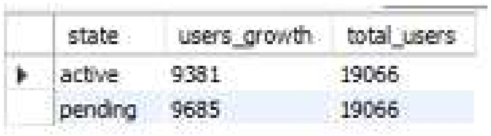
B. **User Growth:**

Your task: Calculate the user growth for product?

SQL queries 2:

```
select state, users_growth, sum(users_growth) over (rows between unbounded preceding and unbounded
following) as total_users
from (select state ,
count(user_id) as users_growth
from users
group by state) as a;
```

output:



	state	users_growth	total_users
▶	active	9381	19066
	pending	9685	19066

Insights:

➢ From the above output there are total of 19066 users from this 9381 users are active and 9685 users are pending. This output shows the users growth for the product

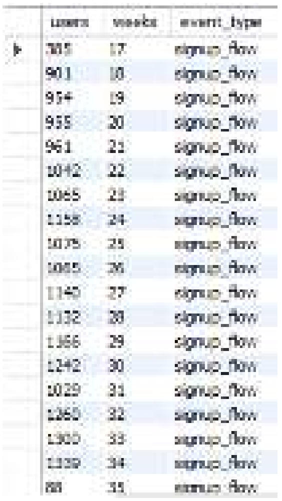
C. **Weekly Retention:**

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

SQL queries 3:

```
select  count(user_id)as users, extract(week from occurred_at) as weeks,event_type
from events_
where event_type = 'signup_flow'
group by event_type,weeks;
```

output:



	users	weeks	event_type
▶	305	17	signup_flow
	901	18	signup_flow
	954	19	signup_flow
	955	20	signup_flow
	961	21	signup_flow
	1042	22	signup_flow
	1065	23	signup_flow
	1158	24	signup_flow
	1075	25	signup_flow
	1065	26	signup_flow
	1140	27	signup_flow
	1132	28	signup_flow
	1366	29	signup_flow
	1242	30	signup_flow
	1029	31	signup_flow
	1260	32	signup_flow
	1300	33	signup_flow
	1339	34	signup_flow
	88	35	signup_flow

Insights:

- ➢ We can see from the above output users sign up from week 17 to 34 are getting increased, but from week 35 there is sudden dip in the sign up flow

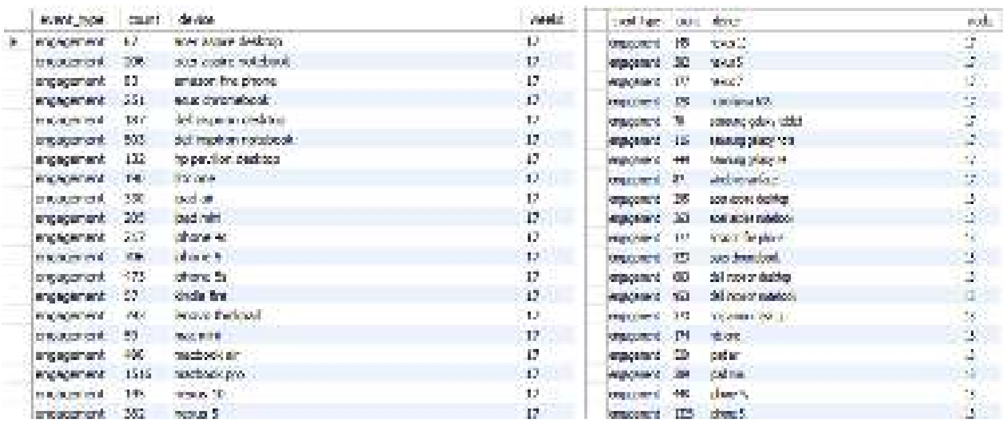
D. **Weekly Engagement:**

Your task: Calculate the weekly engagement per device?

SQL queries 4:

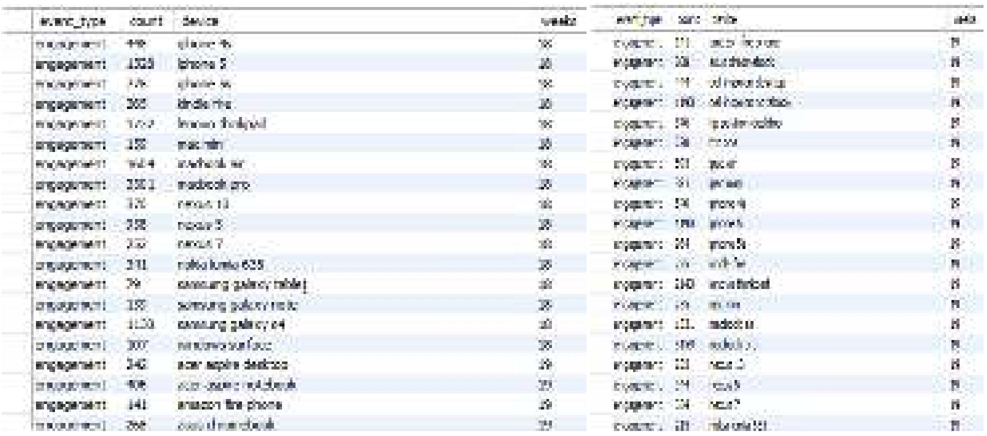
```
select event_type,count(event_type) as count , device, extract(week from occurred_at) as weeks
from events_
where event_type = 'engagement'
group by device , weeks,event_type
order by weeks
```

output



event_type	count	device	weeks
engagement	67	acer aspire desktop	17
engagement	208	acer aspire notebook	17
engagement	83	amazon fire phone	17
engagement	251	asus chromebook	17
engagement	187	dell inspiron desktop	17
engagement	503	dell inspiron notebook	17
engagement	132	hp pavilion desktop	17
engagement	140	htc one	17
engagement	330	ipad air	17
engagement	205	ipad mini	17
engagement	257	iphone 4s	17
engagement	[illegible]	iphone 5	17
engagement	473	iphone 5s	17
engagement	57	kindle fire	17
engagement	[illegible]	lenovo thinkpad	17
engagement	53	mac mini	17
engagement	496	macbook air	17
engagement	1516	macbook pro	17
engagement	145	nexus 10	17
engagement	302	nexus 5	17

event_type	count	device	weeks
engagement	178	nexus 10	[illegible]
engagement	302	nexus 5	[illegible]
engagement	177	nexus 7	[illegible]
engagement	179	nokia lumia 635	[illegible]
engagement	78	samsung galaxy tablet	[illegible]
engagement	115	samsung galaxy note	[illegible]
engagement	444	samsung galaxy s4	[illegible]
engagement	87	windows surface	[illegible]
engagement	236	acer aspire desktop	[illegible]
engagement	303	acer aspire notebook	[illegible]
engagement	117	amazon fire phone	[illegible]
engagement	323	asus chromebook	[illegible]
engagement	600	dell inspiron desktop	[illegible]
engagement	903	dell inspiron notebook	[illegible]
engagement	193	hp pavilion desktop	[illegible]
engagement	174	htc one	[illegible]
engagement	220	ipad air	[illegible]
engagement	284	ipad mini	[illegible]
engagement	446	iphone 4s	[illegible]
engagement	1125	iphone 5	[illegible]



event_type	count	device	weeks
engagement	449	iphone 4s	28
engagement	1323	iphone 5	28
engagement	776	iphone 5s	28
engagement	285	kindle fire	28
engagement	1747	lenovo thinkpad	28
engagement	158	mac mini	28
engagement	964	macbook air	28
engagement	2381	macbook pro	28
engagement	376	nexus 10	28
engagement	738	nexus 5	28
engagement	332	nexus 7	28
engagement	231	nokia lumia 635	28
engagement	79	samsung galaxy tablet	28
engagement	157	samsung galaxy note	28
engagement	1120	samsung galaxy s4	28
engagement	307	windows surface	28
engagement	242	acer aspire desktop	29
engagement	408	acer aspire notebook	29
engagement	141	amazon fire phone	29
engagement	268	asus chromebook	29

event_type	count	device	weeks
engagement	111	amazon fire phone	35
engagement	238	asus chromebook	35
engagement	119	dell inspiron desktop	35
engagement	190	dell inspiron notebook	35
engagement	156	hp pavilion desktop	35
engagement	134	htc one	35
engagement	151	ipad air	35
engagement	91	ipad mini	35
engagement	174	iphone 4s	35
engagement	198	iphone 5	35
engagement	201	iphone 5s	35
engagement	24	kindle fire	35
engagement	340	lenovo thinkpad	35
engagement	29	mac mini	35
engagement	111	macbook air	35
engagement	399	macbook pro	35
engagement	111	nexus 10	35
engagement	174	nexus 5	35
engagement	124	nexus 7	35
engagement	101	nokia lumia 635	35

Insights:

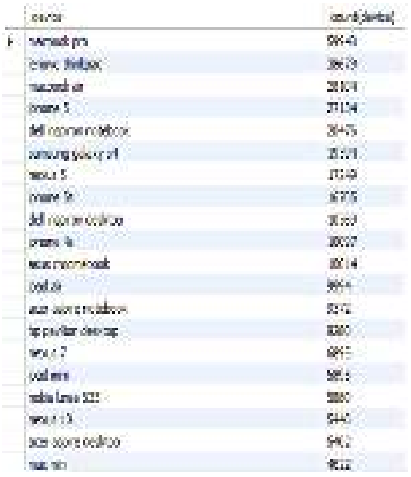
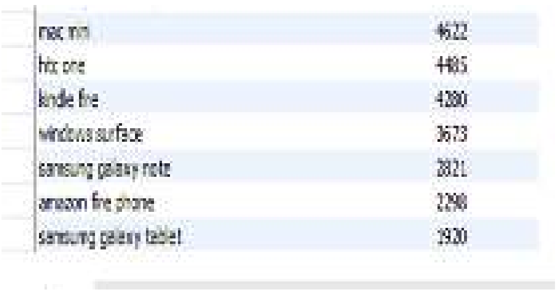
- ➢ Above output that shows the device that are used by users for weekly engagement, they several devices for this purpose.

- From the output we can see that their using phones, laptops, taps, pro books and even desktop.

SQL queries 5:

```
select device , count(device) as used from events_
group by device
order by devices;
```

output:

device	count(device)
macbook pro	50640
lenovo thinkpad	38679
macbook air	28103
iphone 5	27134
dell inspiron notebook	26475
samsung galaxy s4	25579
nexus 5	17349
iphone 5s	16775
dell inspiron desktop	10353
iphone 4s	10097
asus chromebook	10014
ipad air	9964
acer aspire notebook	8574
hp pavilion desktop	8080
nexus 7	6955
ipad mini	5905
nokia lumia 635	5880
nexus 10	5440
acer aspire desktop	5462
mac mini	4622
htc one	4485
kindle fire	4280
windows surface	3673
samsung galaxy note	2821
amazon fire phone	2298
samsung galaxy tablet	1920

Insights:

- This output shows devices used by users, here macbook pro which is a device used the by most of the users
- Samsung galaxy tablet which is least used device here

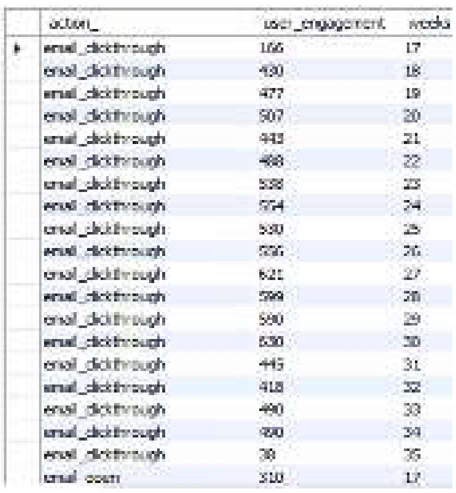
E. **Email Engagement:**

Your task: Calculate the email engagement metrics?

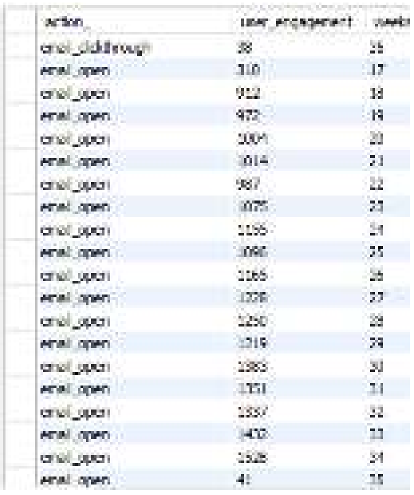
SQL queries 6:

```
select action_, count(user_id) as user_engagement, extract(week from occurred_at) as weeks from email_events
group by action_, weeks
order by action_
```

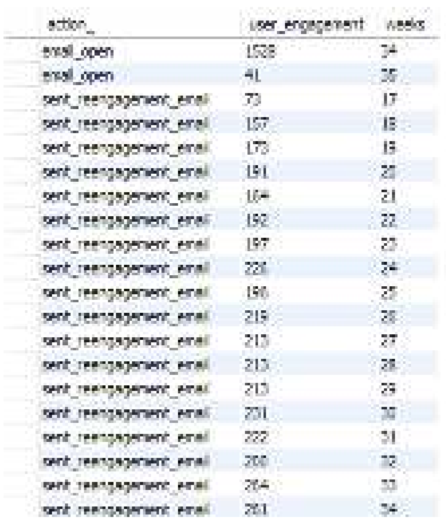
output:



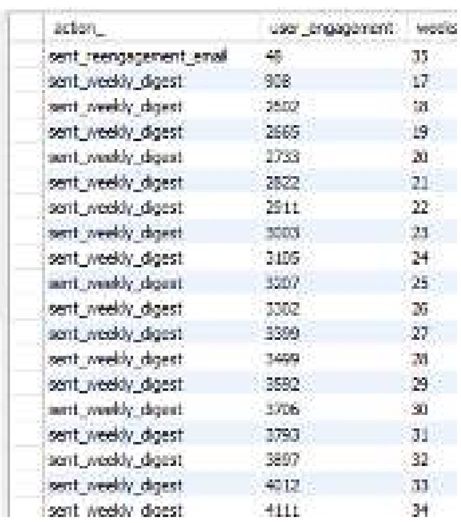
action_	user_engagement	weeks
email_clickthrough	166	17
email_clickthrough	430	18
email_clickthrough	477	19
email_clickthrough	507	20
email_clickthrough	443	21
email_clickthrough	488	22
email_clickthrough	538	23
email_clickthrough	554	24
email_clickthrough	530	25
email_clickthrough	536	26
email_clickthrough	621	27
email_clickthrough	599	28
email_clickthrough	590	29
email_clickthrough	630	30
email_clickthrough	445	31
email_clickthrough	418	32
email_clickthrough	490	33
email_clickthrough	490	34
email_clickthrough	38	35
email_open	310	17



action_	user_engagement	week
email_clickthrough	38	35
email_open	310	17
email_open	912	18
email_open	972	19
email_open	1004	20
email_open	1014	21
email_open	987	22
email_open	1075	23
email_open	1156	24
email_open	1096	25
email_open	1165	26
email_open	1228	27
email_open	1250	28
email_open	1219	29
email_open	1283	30
email_open	1351	31
email_open	1337	32
email_open	1400	33
email_open	1528	34
email_open	41	35



action_	user_engagement	weeks
email_open	1528	34
email_open	41	35
sent_reengagement_email	73	17
sent_reengagement_email	157	18
sent_reengagement_email	173	19
sent_reengagement_email	191	20
sent_reengagement_email	164	21
sent_reengagement_email	192	22
sent_reengagement_email	197	23
sent_reengagement_email	226	24
sent_reengagement_email	196	25
sent_reengagement_email	219	26
sent_reengagement_email	213	27
sent_reengagement_email	213	28
sent_reengagement_email	213	29
sent_reengagement_email	231	30
sent_reengagement_email	222	31
sent_reengagement_email	208	32
sent_reengagement_email	264	33
sent_reengagement_email	261	34



action_	user_engagement	weeks
sent_reengagement_email	48	35
sent_weekly_digest	908	17
sent_weekly_digest	2502	18
sent_weekly_digest	2665	19
sent_weekly_digest	2733	20
sent_weekly_digest	2822	21
sent_weekly_digest	2911	22
sent_weekly_digest	3003	23
sent_weekly_digest	3105	24
sent_weekly_digest	3207	25
sent_weekly_digest	3302	26
sent_weekly_digest	3399	27
sent_weekly_digest	3499	28
sent_weekly_digest	3592	29
sent_weekly_digest	3706	30
sent_weekly_digest	3793	31
sent_weekly_digest	3897	32
sent_weekly_digest	4012	33
sent_weekly_digest	4111	34

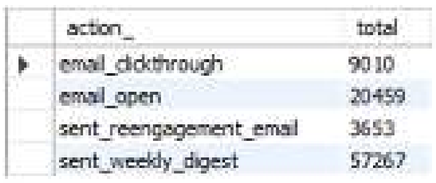
Insights:

➢ These output shows the users weekly email engagement, here there are 4 actions taken by the users like email_clickthrough, email_open, sent_reengagement_email, sent_weekly_digest.

SQL queries 7:

```
select action_, sum(user_engagement) as total from (select action_, count(user_id) as user_engagement,
extract(week from occurred_at) as weeks from email_events
group by action_, weeks
order by action_) as a
group by action_
```

output:



action_	total
email_clickthrough	9010
email_open	20459
sent_reengagement_email	3653
sent_weekly_digest	57267

Insights

- ➢ The output shows the email events, we can see that sent_weekly_digest it is the action taken by the most of the users

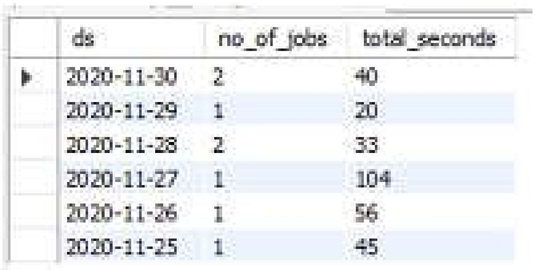
5. RESULT

5.1 Case study 1

- (SQL queries 1) From case study 1 my first task was to find the amount job reviewed over time. I have to use aggregate functions like count() and sum (). Then I group by the date column

```
select ds, count(job_id) as no_of_jobs,sum(time_spend) as total_seconds
 from job_data
 group by ds
 having sum(time_spend) <= 3600;
```

output:

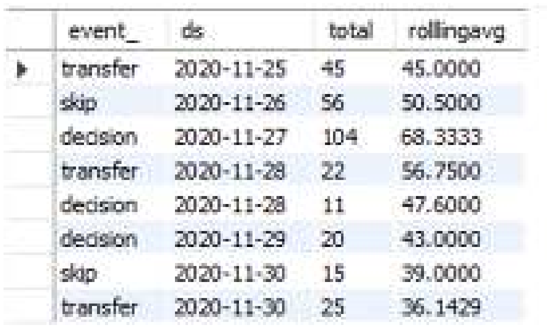


ds	no_of_jobs	total_seconds
2020-11-30	2	40
2020-11-29	1	20
2020-11-28	2	33
2020-11-27	1	104
2020-11-26	1	56
2020-11-25	1	45

- (SQL queries 2) second task was to find the 7 day rolling average of throughput. To find the 7 day rolling average I have to use the window function over(). This function help me to get the 7 day rolling average

```
select a.* ,
 avg(total) over( order by ds rows between 6 preceding and current row) as rolllingavg
 from(
select event_, ds, sum(time_spend) as total from job_data group by event_,ds) as a
```

output:

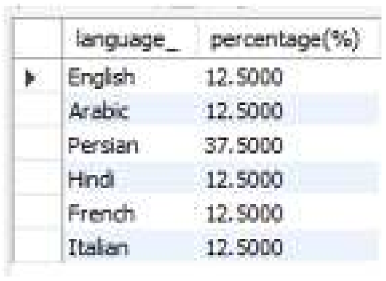


event_	ds	total	rollingavg
transfer	2020-11-25	45	45.0000
skip	2020-11-26	56	50.5000
decision	2020-11-27	104	68.3333
transfer	2020-11-28	22	56.7500
decision	2020-11-28	11	47.6000
decision	2020-11-29	20	43.0000
skip	2020-11-30	15	39.0000
transfer	2020-11-30	25	36.1429

- (SQL queries 3) the next task was to find the Share of each language for different contents. I used sub queries here to count the languages the it help me to find the percentage.

```
select b.language_, b.times * 100/(select sum(b.times) from (select language_, count(ds) as
times
from job_data
group by language_) as b)'percentage(%)'
from (select language_, count(ds) as times
from job_data
group by language_) as b
```

output:

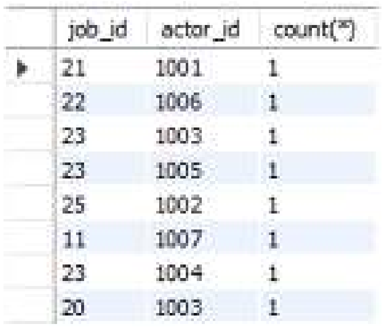


language_	percentage(%)
English	12.5000
Arabic	12.5000
Persian	37.5000
Hindi	12.5000
French	12.5000
Italian	12.5000

- (SQL queries 4) The last task from the case study 1 is fin duplicate rows. I used count() function for find the duplicate rows but there is non.

```
select job_id, actor_id, count(*) from job_data
group by job_id,actor_id
```

output:



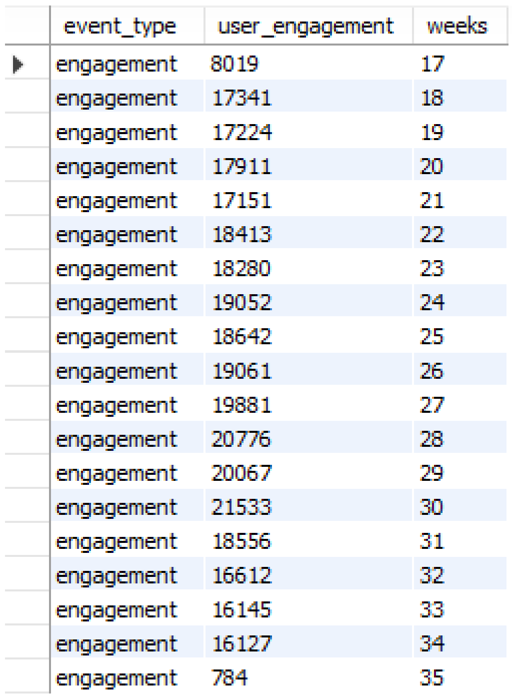
job_id	actor_id	count(*)
21	1001	1
22	1006	1
23	1003	1
23	1005	1
25	1002	1
11	1007	1
23	1004	1
20	1003	1

5.2 Case study 2

- (SQL queries 1) From case study 2 my first task was to Calculate the weekly user engagement. Used table 1 for find the user engagements and also I used extract() function to find the weeks from the date, also I used count() and group by

```
select event_type, count(user_id) as user_engagement,extract(week from occurred_at) as weeks
from events_ where event_type = 'engagement'
group by event_type, weeks
order by weeks
```

output:

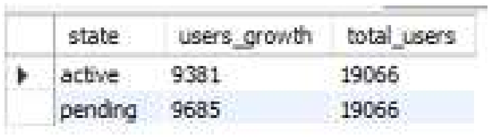


event_type	user_engagement	weeks
engagement	8019	17
engagement	17341	18
engagement	17224	19
engagement	17911	20
engagement	17151	21
engagement	18413	22
engagement	18280	23
engagement	19052	24
engagement	18642	25
engagement	19061	26
engagement	19881	27
engagement	20776	28
engagement	20067	29
engagement	21533	30
engagement	18556	31
engagement	16612	32
engagement	16145	33
engagement	16127	34
engagement	784	35

- (SQL queries 2) next tasks was Calculate the user growth for product. For finding this task I used table 1.here I used sub queries also I used functions like sum(), over() etc.

```
select state, users_growth, sum(users_growth) over (rows between unbounded preceding and unbounded following) as total_users
from (select state ,
count(user_id) as users_growth
from users
group by state) as a;
```

output:

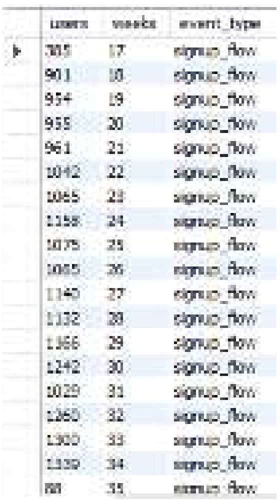


state	users_growth	total_users
active	9381	19066
pending	9685	19066

- (SQL queries 3) 3rd task from the case study 2 was Calculate the weekly retention of users-sign up cohort. I find weekly retention of users by using count(), extract() where condition etc.

```
select  count(user_id)as users, extract(week from occurred_at) as weeks,event_type
from events_
where event_type = 'signup_flow'
group by event_type,weeks;
```

output:

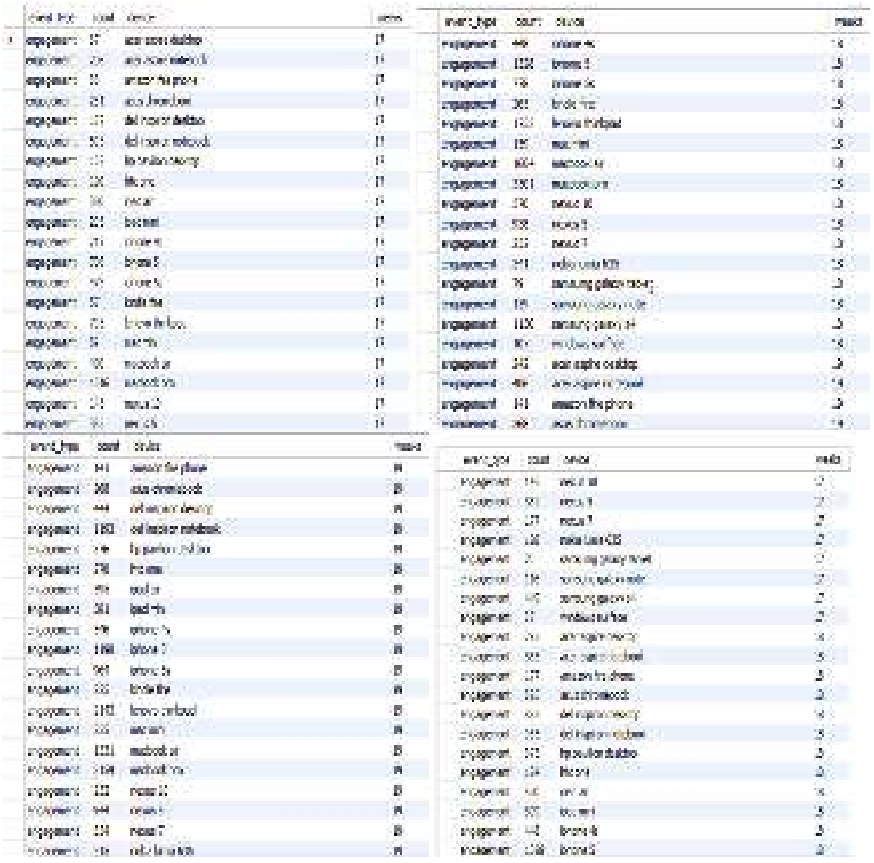


users	weeks	event_type
785	17	signup_flow
901	18	signup_flow
954	19	signup_flow
955	20	signup_flow
961	21	signup_flow
1042	22	signup_flow
1065	23	signup_flow
1158	24	signup_flow
1075	25	signup_flow
1065	26	signup_flow
1140	27	signup_flow
1132	28	signup_flow
1166	29	signup_flow
1242	30	signup_flow
1029	31	signup_flow
1260	32	signup_flow
1300	33	signup_flow
1339	34	signup_flow
88	35	signup_flow

- (SQL queries 4) the next task was to find the weekly engagement per device. For this purpose I have used count(), extract(), where condition, group by etc.

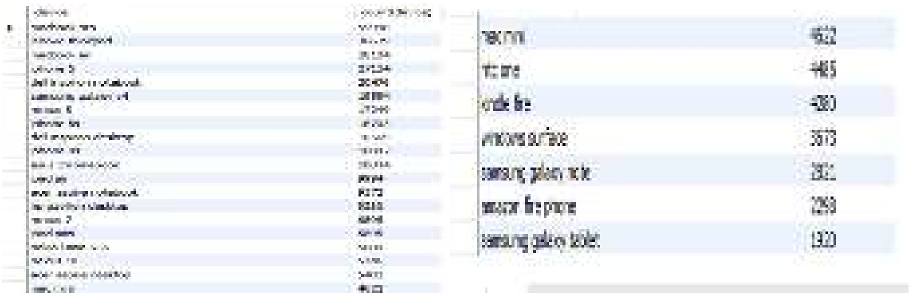
```
select event_type,count(event_type) as count , device, extract(week from occurred_at) as
weeks
from events_
where event_type = 'engagement'
group by device , weeks,event_type
order by weeks
```

output



select device , count(device) as used from events_
group by device
order by devices;

output:

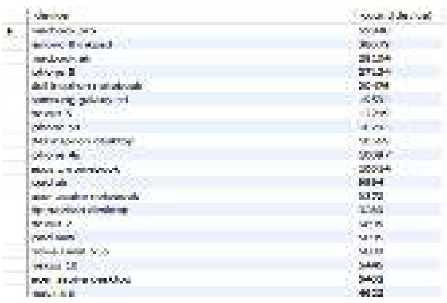


```
ipad mini                 4622
htc one                   4465
kindle fire               4280
windows surface           3573
samsung galaxy note       2821
amazon fire phone         2293
samsung galaxy tablet     1320
```

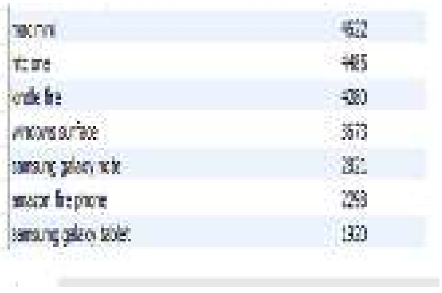
- (SQL queries 5) this queries was used to find the number of device used and how many times.

select device , count(device) as used from events_
group by device
order by devices;

output:



	device	[illegible]
▸	macbook pro	[illegible]
	lenovo thinkpad	[illegible]
	macbook air	[illegible]
	iphone 5	[illegible]
	dell inspiron notebook	[illegible]
	samsung galaxy s4	[illegible]
	nexus 5	[illegible]
	iphone 5s	[illegible]
	[illegible]	[illegible]
	iphone 4s	[illegible]
	[illegible]	[illegible]
	ipad air	[illegible]
	[illegible]	[illegible]
	hp pavilion desktop	[illegible]
	nexus 7	[illegible]
	[illegible]	[illegible]
	[illegible]	[illegible]
	nexus 10	[illegible]
	[illegible]	[illegible]
	mac mini	[illegible]

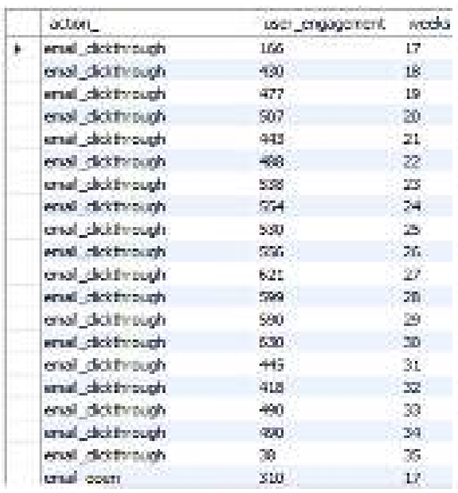


mac mini	4622
htc one	4485
kindle fire	4380
windows surface	3673
samsung galaxy note	2821
amazon fire phone	2293
samsung galaxy tablet	1320

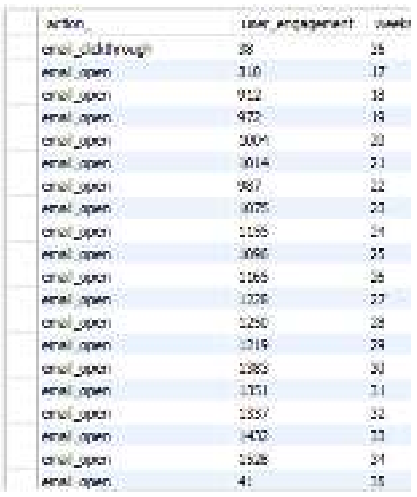
- (SQL queries 6) the last task from the case study 2 was Calculate the email engagement metrics. For this I have used count(), extract() function, group by etc.

```
select action_, count(user_id) as user_engagement, extract(week from occurred_at) as weeks
from email_events
group by action_, weeks
order by action_
```

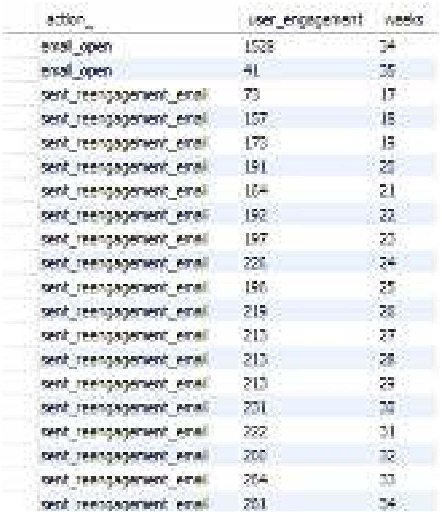
output:



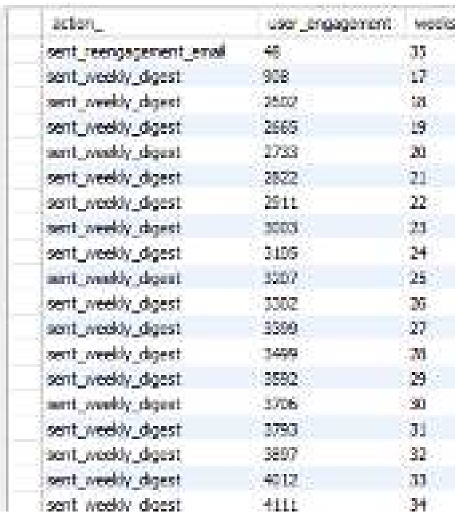
	action_	user_engagement	weeks
▸	email_clickthrough	166	17
	email_clickthrough	430	18
	email_clickthrough	477	19
	email_clickthrough	507	20
	email_clickthrough	443	21
	email_clickthrough	488	22
	email_clickthrough	538	23
	email_clickthrough	554	24
	email_clickthrough	530	25
	email_clickthrough	536	26
	email_clickthrough	621	27
	email_clickthrough	599	28
	email_clickthrough	590	29
	email_clickthrough	630	30
	email_clickthrough	445	31
	email_clickthrough	418	32
	email_clickthrough	490	33
	email_clickthrough	490	34
	email_clickthrough	38	35
	email_open	310	17



action_	user_engagement	weeks
email_clickthrough	38	35
email_open	310	17
email_open	912	18
email_open	972	19
email_open	1004	20
email_open	1014	21
email_open	987	22
email_open	1075	23
email_open	1156	24
email_open	1096	25
email_open	1165	26
email_open	1228	27
email_open	1250	28
email_open	1219	29
email_open	1383	30
email_open	1351	31
email_open	1397	32
email_open	1402	33
email_open	1528	34
email_open	41	35



action_	user_engagement	weeks
email_open	1528	34
email_open	41	35
sent_reengagement_email	73	17
sent_reengagement_email	157	18
sent_reengagement_email	173	19
sent_reengagement_email	191	20
sent_reengagement_email	164	21
sent_reengagement_email	192	22
sent_reengagement_email	197	23
sent_reengagement_email	226	24
sent_reengagement_email	196	25
sent_reengagement_email	219	26
sent_reengagement_email	213	27
sent_reengagement_email	213	28
sent_reengagement_email	213	29
sent_reengagement_email	231	30
sent_reengagement_email	222	31
sent_reengagement_email	200	32
sent_reengagement_email	264	33
sent_reengagement_email	201	34

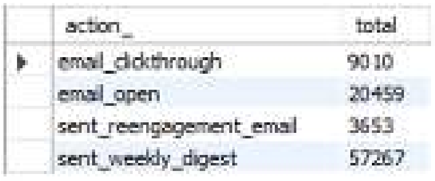


action_	user_engagement	weeks
sent_reengagement_email	46	35
sent_weekly_digest	908	17
sent_weekly_digest	2502	18
sent_weekly_digest	2665	19
sent_weekly_digest	2733	20
sent_weekly_digest	2822	21
sent_weekly_digest	2911	22
sent_weekly_digest	3003	23
sent_weekly_digest	3106	24
sent_weekly_digest	3207	25
sent_weekly_digest	3302	26
sent_weekly_digest	3399	27
sent_weekly_digest	3499	28
sent_weekly_digest	3592	29
sent_weekly_digest	3706	30
sent_weekly_digest	3793	31
sent_weekly_digest	3897	32
sent_weekly_digest	4012	33
sent_weekly_digest	4111	34

- (SQL queries 7) this queries was used to find the number of action taken by the user and how many times.

```
select action_, sum(user_engagement) as total from (select action_, count(user_id) as
user_engagement, extract(week from occurred_at) as weeks from email_events
group by action_, weeks
order by action_) as a
group by action_
```

output:



action_	total
email_clickthrough	9010
email_open	20459
sent_reengagement_email	3653
sent_weekly_digest	57267

6. Conclusion

This project mainly focused on operation analytics. Operation analytics helps companies to find which area they have to improve. Here in this project there are two different kinds of data sets were provided. One was job data and another was to find investigation metric spike. There are certain number of task was gives in each case studies. That required SQL queries, this project helped me to improve my existing skills and knowledge. The data sets given were bit complex and that required more understanding. I used some more function and queries to find answer, so it help me to familiar with some function like over(), count(), etc.

I think my answers will help the team to get some insights out of it and it will help the business to find where the dip happens, and increase the growth of the business.