Project Description:

Operational Analytics plays a critical role in evaluating a company's overall operations so as to spot potential areas which needs enhancement or improvements.

It is vital for us to thoroughly examine spikes within metrics – as being the data analyst, we must be able to understand indicators such as customer involvement or sales receipt figures, noticeable dips, etc.

Just like those on board at Microsoft operating within analogous roles, our focus constitutes scrutinizing given datasets & deploying intricate SQL aptitude for purposes related to offering valuable analysis and guidance.

Approach:

This project is developed using SQL Workbench. Firstly I understood the project and its requirements. Then we created databases with the help of datasets provided to us. Then we imported files, after creating tables, from the provided data then performed different tasks executing queries to get the desired results, derived different insights provided by the team.

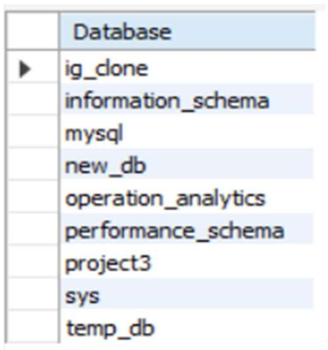
Execution:

Case-1: Job Data Analysis:

-- Task: A). Jobs Reviewed Over Time: --

```
create database operation_analytics;
show databases;
```

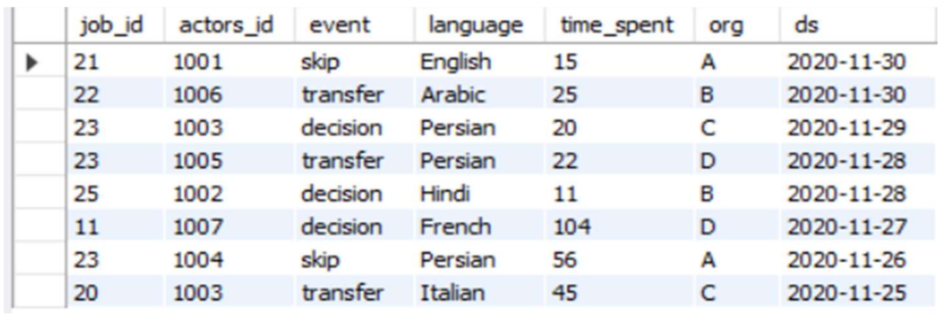
Output:-



```
Database
ig_clone
information_schema
mysql
new_db
operation_analytics
performance_schema
project3
sys
temp_db
```

```
use operation_analytics;
create table job_data(job_id int,
actors_id int,
event varchar(50),
language varchar(50),
time_spent int,
org varchar(1),
ds date
);
Insert into job_data (ds, job_id, actors_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');
         select * from job_data;
```

Output:-

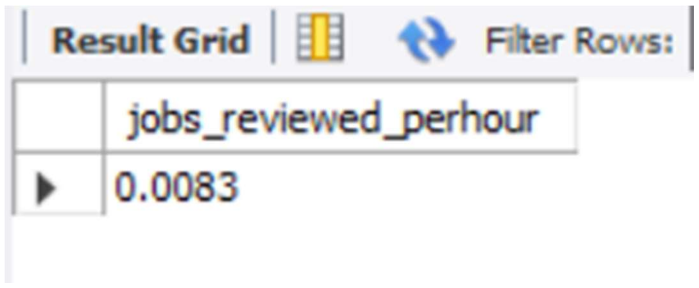


	job_id	actors_id	event	language	time_spent	org	ds
▶	21	1001	skip	English	15	A	2020-11-30
	22	1006	transfer	Arabic	25	B	2020-11-30
	23	1003	decision	Persian	20	C	2020-11-29
	23	1005	transfer	Persian	22	D	2020-11-28
	25	1002	decision	Hindi	11	B	2020-11-28
	11	1007	decision	French	104	D	2020-11-27
	23	1004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

-- Calculating the number of jobs reviewed per hour for each day in November 2020: --

```
select
count(distinct job_id)/(30*24) as jobs_reviewed_perhour
from job_data
where
ds like '2020-11%' ;
```

Output:-



	jobs_reviewed_perhour
▶	0.0083

-- Task: B) Throughput Analysis: --

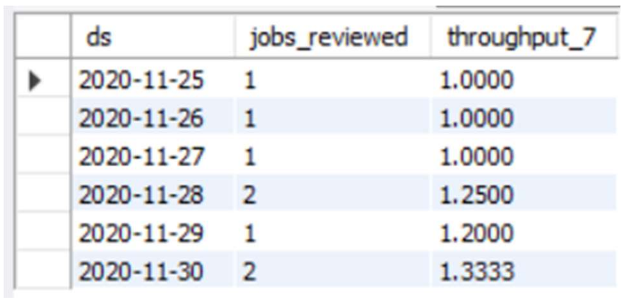
/*Throughput: It is the number of events happening per second.*/

-- Calculating calculate the 7-day rolling average of throughput: --

```
select ds,
count(distinct job_id) as jobs_reviewed,
avg(count(distinct job_id))over(order by ds rows between 6 preceding and
current row)
as throughput_7
from
job_data
```

```
where ds between '2020-11-01' and '2020-11-30'
group by ds
order by ds;
```

Output:-



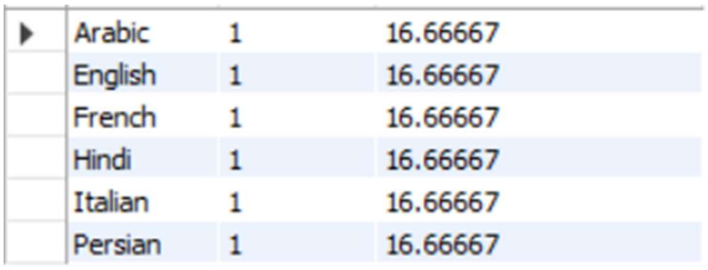
ds	jobs_reviewed	throughput_7
2020-11-25	1	1.0000
2020-11-26	1	1.0000
2020-11-27	1	1.0000
2020-11-28	2	1.2500
2020-11-29	1	1.2000
2020-11-30	2	1.3333

-- Task: C) Language Share Analysis: --

```
-- Calculating percentage share of each language over the last 30 days: --

select language,
count(distinct job_id) as num_jobs,
100.0* count(distinct job_id)/(select count(distinct job_id) from job_data)
as percentage_share_jobs
from job_data
group by language;
```

Output:-



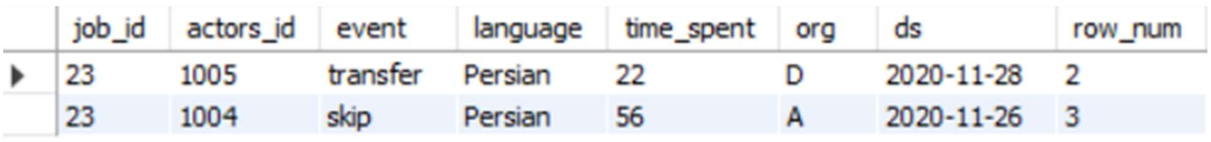
Arabic	1	16.66667
English	1	16.66667
French	1	16.66667
Hindi	1	16.66667
Italian	1	16.66667
Persian	1	16.66667

-- Task: D) Duplicate Rows Detection: --

```
-- Identifying duplicate rows from the job_data table: --


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

Output:-



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

Case-2: Investigating Metric Spike:

# Firstly we have to create tables to import the data from the provided datasets to perform tasks on them.

```
create database project3;
use project3;

# Table-1 users
create table users (
user_id int,
created_at varchar(100),
company_id int,
language varchar(50),
activated_at varchar(100),
state varchar(50)
);
#now as table is created, we can further import the data using following commands:
```

```
show variables like 'secure_file_priv';
load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/table-1
users.csv"
into table users
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;

alter table users add column temp_created_at datetime;
update users set temp_created_at = str_to_date(created_at, '%d-%m-%Y
%H:%i');
alter table users drop created_at;
alter table users change column temp_created_at created_at datetime;

alter table users add column temp_activated_at datetime;
update users set temp_activated_at = str_to_date(activated_at, '%d-%m-%Y
%H:%i');
alter table users drop activated_at;
alter table users change column temp_activated_at activated_at datetime;
```

# Table-2 events

```
 create table events(
 user_id int,
 occurred_at varchar(100),
 event_type varchar(50),
 event_name varchar(100),
 location varchar(50),
 device varchar(50),
 user_type int
 );

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/table-2
events.csv"
into table events
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```

```
select * from events;

alter table events add column temp_occurred_at datetime;
update events set temp_occurred_at = str_to_date(occurred_at, '%d-%m-%Y
%H:%i');
alter table events drop column occurred_at;
alter table events change column temp_occurred_at occurred_at datetime;
```

# Table-3 email_events

```
create table email_events (
user_id int,
occurred_at  varchar(100),
action varchar(100),
user_type int
);

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/table-3
email_events.csv"
into table email_events
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
select * from email_events;

alter table email_events add column temp_occurred_at datetime;
update email_events set temp_occurred_at = str_to_date(occurred_at, '%d-%m-
%Y %H:%i');
alter table email_events drop column occurred_at;
alter table email_events change column temp_occurred_at occurred_at datetime;
```

-- Task: A) Weekly User Engagement: --

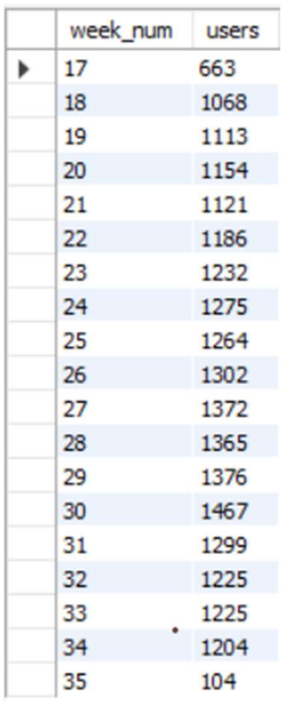
-- Calculating the weekly user engagement: --

```
select extract(week from occurred_at) as week_num,

count(distinct user_id) as users from events
```

```
group by week_num;
```

Output:-



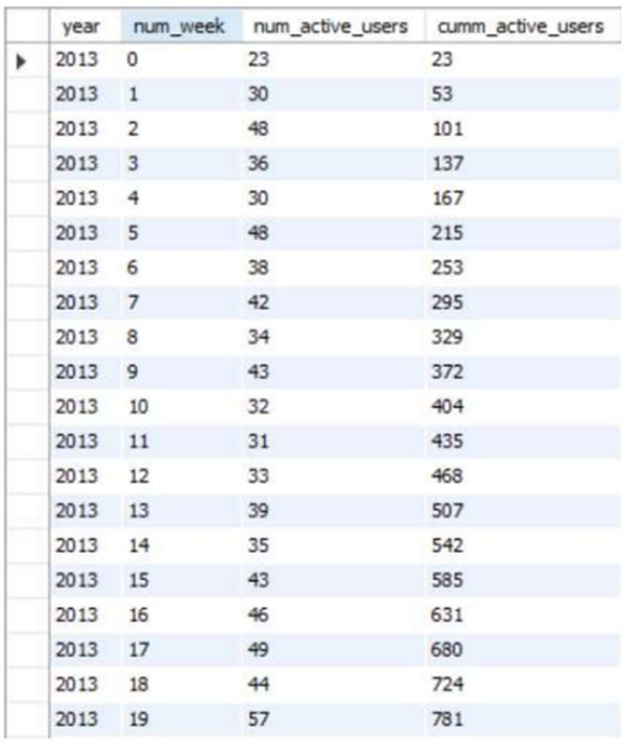
week_num	users
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

-- Task: B) User Growth Analysis: --

-- Calculating the user growth for the product: --

```
select year, num_week, num_active_users,
sum(num_active_users) over(order by year, num_week
rows between unbounded
preceding and current row) as cumm_active_users
from
(select
extract(year from activated_at) as year,
extract(week from activated_at) as num_week,
count(distinct user_id) as num_active_users
from users
where state = 'active'
group by year, num_week
order by year, num_week
)a
```

Output:-



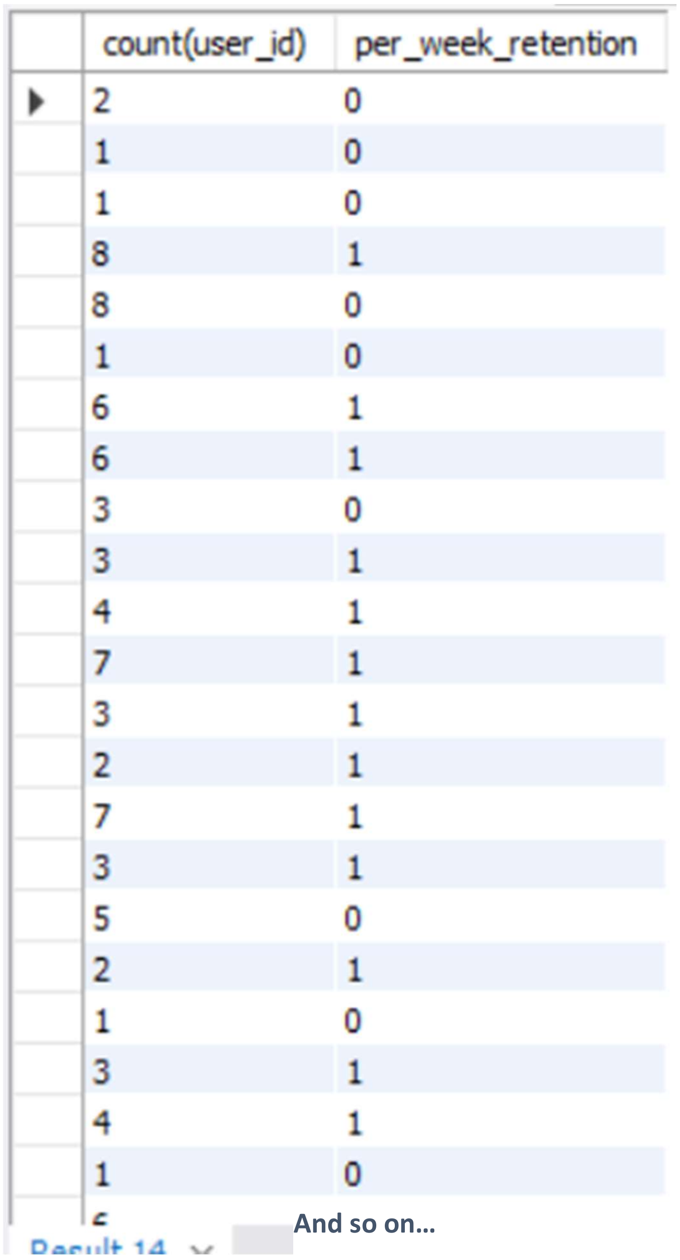
year	num_week	num_active_users	cumm_active_users
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724
2013	19	57	781

-- Task: C). Weekly Retention Analysis: --

-- Calculating the weekly retention of users based on their sign-up cohort: --

```
select count(user_id),
       sum(case when retention_week = 1 then 1 else 0 end) as
per_week_retention
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 events
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 events
where event_type = 'engagement')b
on a.user_id = b.user_id
)a
group by user_id
order by user_id;
```

Output:-



count(user_id)	per_week_retention
2	0
1	0
1	0
8	1
8	0
1	0
6	1
6	1
3	0
3	1
4	1
7	1
3	1
2	1
7	1
3	1
5	0
2	1
1	0
3	1
4	1
1	0

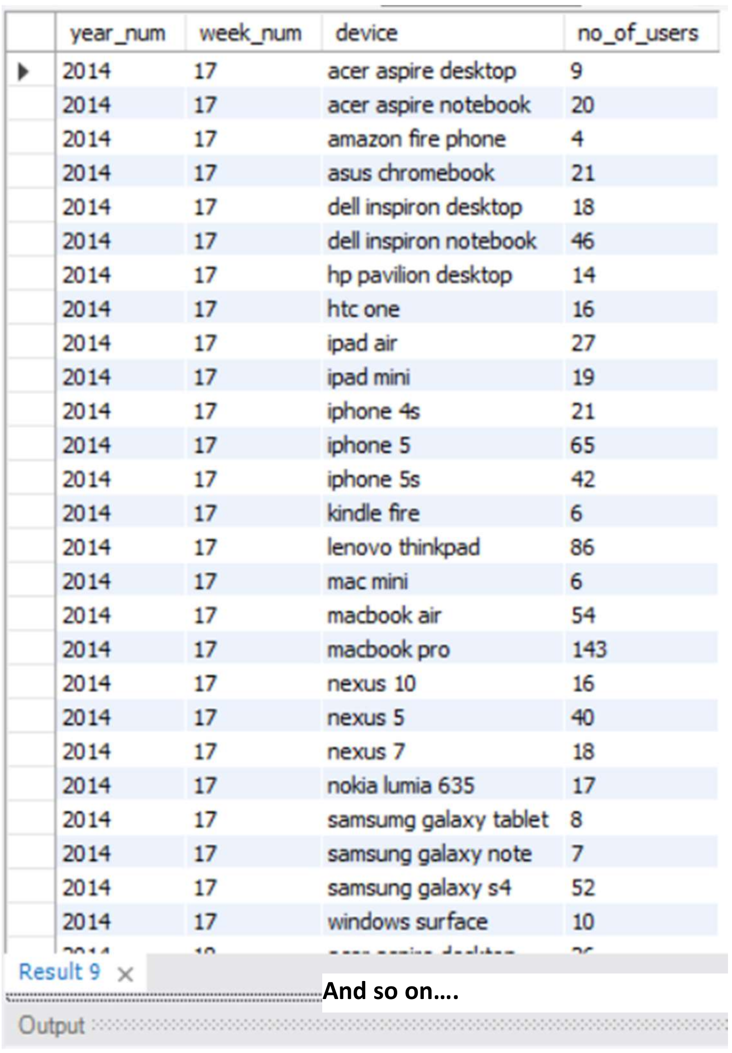
And so on...

-- Task: D). Weekly Engagement Per Device: --

-- Calculating the weekly engagement per device: --

```
select
extract(year from occurred_at) as year_num,
extract(week from occurred_at) as week_num,
device,
count(distinct user_id) as no_of_users
from `events`
where event_type = 'engagement'
group by 1,2,3
order by 1,2,3;
```

Output:-



year_num	week_num	device	no_of_users
2014	17	acer aspire desktop	9
2014	17	acer aspire notebook	20
2014	17	amazon fire phone	4
2014	17	asus chromebook	21
2014	17	dell inspiron desktop	18
2014	17	dell inspiron notebook	46
2014	17	hp pavilion desktop	14
2014	17	htc one	16
2014	17	ipad air	27
2014	17	ipad mini	19
2014	17	iphone 4s	21
2014	17	iphone 5	65
2014	17	iphone 5s	42
2014	17	kindle fire	6
2014	17	lenovo thinkpad	86
2014	17	mac mini	6
2014	17	macbook air	54
2014	17	macbook pro	143
2014	17	nexus 10	16
2014	17	nexus 5	40
2014	17	nexus 7	18
2014	17	nokia lumia 635	17
2014	17	samsumg galaxy tablet	8
2014	17	samsung galaxy note	7
2014	17	samsung galaxy s4	52
2014	17	windows surface	10

Result 9 ×

And so on....

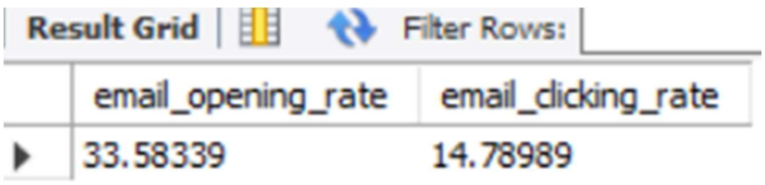
Output

-- Task: E). Email Engagement Analysis: --

-- Calculating the email engagement metrics: --

```
select
100.0 * sum(case when email_cat = 'email_opened' then 1 else 0 end)
      /sum(case when email_cat = 'email_sent' then 1 else 0 end)
as email_opening_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_clicking_rate
from
(
select *,
case when action in ('sent_weekly_digest', 'sent_reengagement_email')
    then 'email_sent'
    when action in ('email_open')
    then 'email_opened'
    when action in ('email_clickthrough')
    then 'email_clicked'
end as email_cat
from email_events
)a;
```

Output:-



```
email_opening_rate   email_clicking_rate
33.58339             14.78989
```

Tech-stack used:

MySQL Workbench,

Version: 8.0.34 (MySQL Community Server – GPL),

Compiled for: Win64 (x86_64)

The purpose of using MySQL was to extract and execute the necessary queries on tables, from the provided database, to get the accurate result and derive a precise insights that has been asked by the team.

Insights:

Case study-1 (Job data)

- The number of distinct jobs reviewed per hour for each day in November 2020 is 83.
- We used the 7-day rolling average of throughput as it gives the average for all the days right from day 1 to day 7 whereas, daily metric gives the average for only that particular day itself.
- The percentage share of all languages give came out to be same for past 30 days i.e. 16.66%
- There are two duplicate rows if we partition the data by job_id. But if we look the overall columns, all the rows are unique.

Case study-2 (Investigating metrics spike)

- The weekly user engagement increased from week 18th to week 31st on an average (there's a noticeable decrease in between thought) and then started declining from then onwards. This means that some of the users do not find much quality in the product/service in the last and middle of the weeks.
- There is a constant increase in cumulative users from 1st week of 2013 to the 35th week of 2014. But even after increase in numbers, the number of activate users kept fluctuating that reflects the new approaches should be taken to keep them engage.

- There’s a impressive retentions from some common users but not from the most which means there’s a need to find that loop hole lacking interests of users.
- The overall count of weekly engagement per device used is the most for MacBook users and iPhone users.
- The email opening rate is 33.58% and email clicking rate is 14.78%. The users are slowly engaging with the email which derives a positive responses from users which is good for a company.

Results:

In this project, we learned how to load lengthy datasets into database, how to manipulate .csv data using proper data type fields, how to apply advance MySQL functions. This helped me understand the real time industry work related curriculum. I learned about investigating metrics spikes, the dips, how to use data to get the desired insights to help company improving user’s needs.