Case Study: 3 (Operation Analytics and Investigating Spikes)

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.

- Investigating metric spike is also an important part of operation analytics where we have to 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 it's very important to investigate metric spike.

Plan: The project has been divided into two separate sections as there are two separate needs

- **Job Data:** The company wants to understand and predict the overall growth or decline of the company's fortune. This will help various team within the company to find the areas on which it must improve upon. To carry out the changes they require the following data
  - ➢ The number of jobs reviewed over time.
  - ➢ The number of events happening per second
  - ➢ Share of each language for different contents
  - ➢ Duplicate data

- **Investigating metric spikes:** The company wants to have an insight about the engagement of the users under the influence of the products periodically. For that they want to check the details of the following data
  - ➢ Measuring if the user is active or not
  - ➢ Checking the number of users growing overtime
  - ➢ Users getting retained weekly after signing up
  - ➢ To measure the activeness of a user
  - ➢ Users engaging with the email service

Prepare/Approach: Now we'll be figuring out the objectives on successfully fulfilling the requirements. I'll be using MySQL workbench 8.0 CE as the UI is very clean and pretty easy to navigate. Also, it has separate sections to write queries, check errors and collect output in a single window.

Hereby, I'll be creating a roadmap on how to fulfil the expectations of the company using the process of operation analytics and metric spike.

- **Job Data:**
  - ➢ **<u>Number of Jobs reviewed:</u>** We have to calculate the number of jobs reviewed per hour per day of November 2020
  - ➢ **<u>Throughput:</u>** We have to calculate the 7 day rolling average of throughput

- **Percentage share of each language:** We have to calculate the percentage share of each language in the last 30 days
- **Duplicate Rows:** We have to display the duplicates from the table

- **Investigating Metric Spike:**
  - **User Engagement:** We have to calculate user engagement in the company's app/website weekly.
  - **User Growth:** We have to calculate the increment in users for a product.
  - Weekly Retention: We have to calculate the weekly retention of users sign up cohort.
  - **Weekly Engagement:** We have to calculate the weekly engagement per device.
  - **Email Engagement:** We have to check the email engagement metrics.

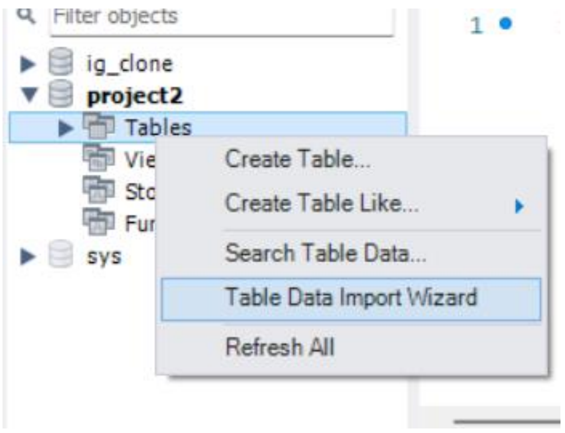
Process: At first we will be preparing a schema for Job Data and then we will create a new schema for Investigating Metric Spike.

- Now that we know the objectives and target, for continuing our process we'll be requiring to create a database using which we can carry out various SQL queries. Also, we have been provided a small sample data in csv format. Therefore, we'll be creating a new schema and then load the table under that schema.

Lets, create a schema under the name project2

Now lets load the table into the table section of the schema.

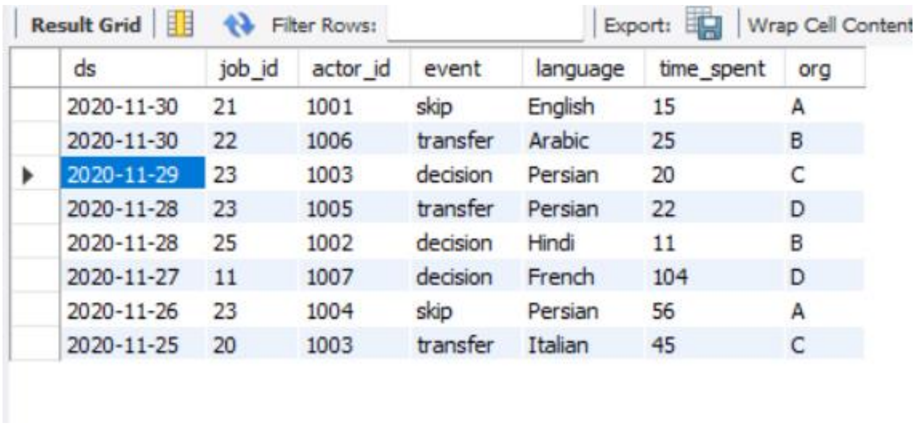
- Right click on Tables and select 'Table Data Import Wizard'.



- Click Browse and choose the file path where the downloaded csv file is loaded.

- Click next and follow the instructions. Once imported, to check if the table is imported or not type the following query
  SELECT * FROM project_2.`sql_project_1`;

The following table has been imported with total 8 entries.



Result Grid | Filter Rows: | Export: | Wrap Cell Content

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

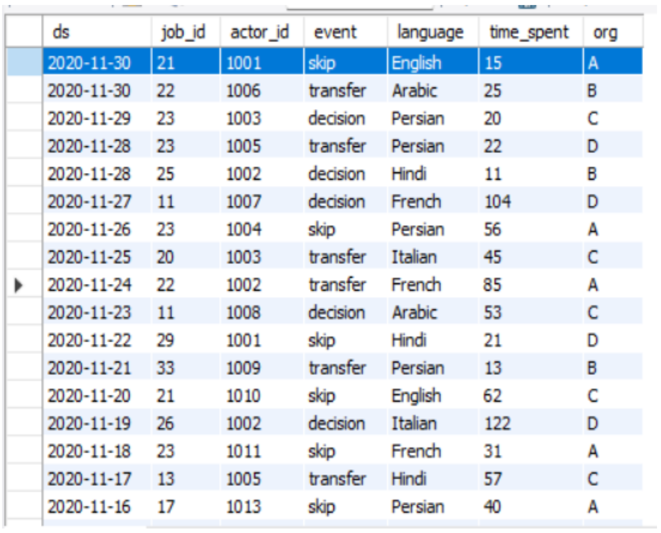
- Now to carry out the operation we have to create at least 30 entries as it won't be possible to complete our tasks with just 8 entries.
- We've added some entries from our side by using the following SQL query.

INSERT INTO sql_project_1 (ds, job_id, actor_id, event, language, time_spent, org)

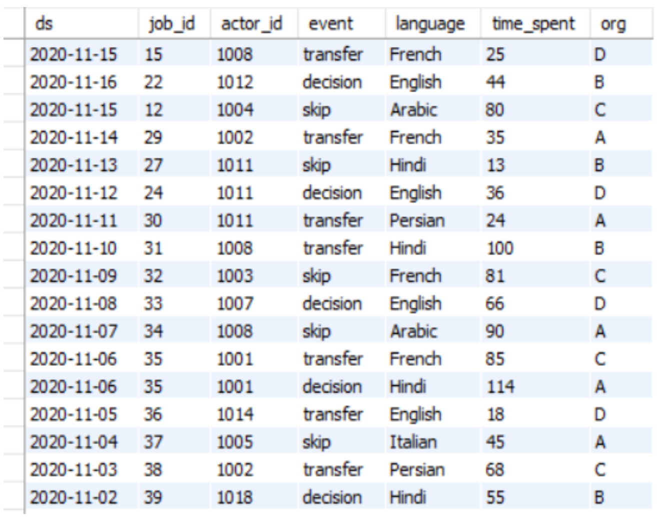
VALUES

(2020-11-24, 22, 1002, "transfer", "French", 85, "A"), (2020-11-23, 11, 1008, "decision", "Arabic", 53, "C"), (2020-11-22, 29, 1001, "skip", "Hindi", 21, "D"), (2020-11-21, 33, 1009, "transfer", "Persian", 13, "B"), (2020-11-20, 21, 1010, "skip", "English", 62, "C"), (2020-11-19, 26, 1002, "decision", "Italian", 122, "D"), (2020-11-18, 23, 1011, "skip", "French", 31, "A"), (2020-11-17, 13, 1005, "transfer", "Hindi", 57, "C"), (2020-11-16, 17, 1013, "skip", "Persian", 40, "A"), (2020-11-15, 15, 1008, "transfer", "French", 25, "D"), (2020-11-16, 22, 1012, "decision", "English", 44, "B"), (2020-11-15, 12, 1004, "skip", "Arabic", 80, "C"), (2020-11-14, 29, 1002, "transfer", "French", 35, "A"), (2020-11-13, 27, 1011, "skip", "Hindi", 13, "B"), (2020-11-12, 24, 1011, "decision", "English", 36, "D"), (2020-11-11, 30, 1011, "transfer", "Persian", 24, "A"), (2020-11-10, 31, 1008, "transfer", "Hindi", 100, "B"), (2020-11-09, 32, 1003, "skip", "French", 81, "C"), (2020-11-08, 33, 1007, "decision", "English", 66, "D"), (2020-11-07, 34, 1008, "skip", "Arabic", 90, "A"), (2020-11-06, 35, 1001, "transfer", "French", 85, "C"), (2020-11-06, 35, 1001, "decision", "Hindi", 114, "A"), (2020-11-05, 36, 1014, "transfer", "English", 18, "D"), (2020-11-04, 37, 1005, "skip", "Italian", 45, "A"), (2020-11-03, 38, 1002, "transfer", "Persian", 68, "C"), (2020-11-02, 39, 1018, "decision", "Hindi", 55, "B"), (2020-11-01, 40, 1004, "transfer", "Italian", 28, "D");

- The following output has been generated

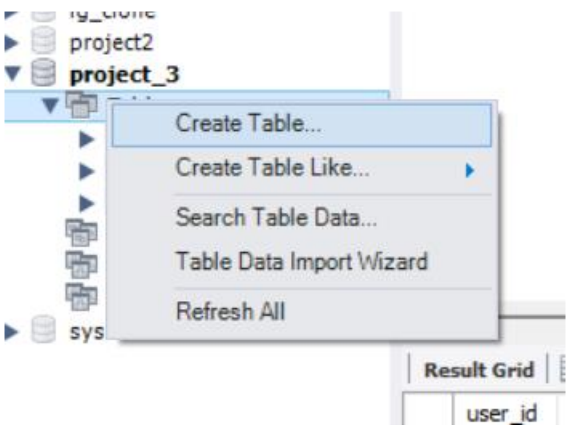


ds	job_id	actor_id	event	language	time_spent	org
2020-11-30	21	1001	skip	English	15	A
2020-11-30	22	1006	transfer	Arabic	25	B
2020-11-29	23	1003	decision	Persian	20	C
2020-11-28	23	1005	transfer	Persian	22	D
2020-11-28	25	1002	decision	Hindi	11	B
2020-11-27	11	1007	decision	French	104	D
2020-11-26	23	1004	skip	Persian	56	A
2020-11-25	20	1003	transfer	Italian	45	C
2020-11-24	22	1002	transfer	French	85	A
2020-11-23	11	1008	decision	Arabic	53	C
2020-11-22	29	1001	skip	Hindi	21	D
2020-11-21	33	1009	transfer	Persian	13	B
2020-11-20	21	1010	skip	English	62	C
2020-11-19	26	1002	decision	Italian	122	D
2020-11-18	23	1011	skip	French	31	A
2020-11-17	13	1005	transfer	Hindi	57	C
2020-11-16	17	1013	skip	Persian	40	A

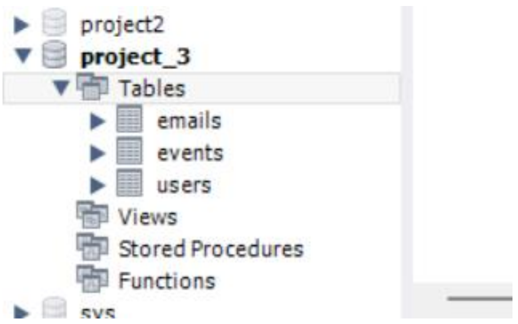


ds	job_id	actor_id	event	language	time_spent	org
2020-11-15	15	1008	transfer	French	25	D
2020-11-16	22	1012	decision	English	44	B
2020-11-15	12	1004	skip	Arabic	80	C
2020-11-14	29	1002	transfer	French	35	A
2020-11-13	27	1011	skip	Hindi	13	B
2020-11-12	24	1011	decision	English	36	D
2020-11-11	30	1011	transfer	Persian	24	A
2020-11-10	31	1008	transfer	Hindi	100	B
2020-11-09	32	1003	skip	French	81	C
2020-11-08	33	1007	decision	English	66	D
2020-11-07	34	1008	skip	Arabic	90	A
2020-11-06	35	1001	transfer	French	85	C
2020-11-06	35	1001	decision	Hindi	114	A
2020-11-05	36	1014	transfer	English	18	D
2020-11-04	37	1005	skip	Italian	45	A
2020-11-03	38	1002	transfer	Persian	68	C
2020-11-02	39	1018	decision	Hindi	55	B

❖ Now we'll be creating a new schema for Investigating Metric Spike.
  - Let us name the schema as project_3
  - Now right click on tables and click Table Data Import Wizard



  - Now lets import the csv files that we have been provided and name them as email, events and users



Analyze and Share: As the database and tables has been created. Now we can start analyzing the tasks given, step by step

- **For Job Data:**

1. **Number of jobs reviewed:** We will be calculating the number of jobs per hour per day for Nov,2020.

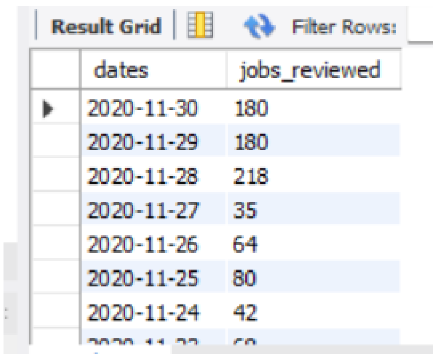
SELECT ds AS dates, round((count(job_id)/sum(time_spent)*3600)) AS "jobs_reviewed"

FROM sql_project_1

WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'

GROUP BY ds;

Upon execution, we can see that on date 2020-11-28, a total of 218 number of jobs have been reviewed.



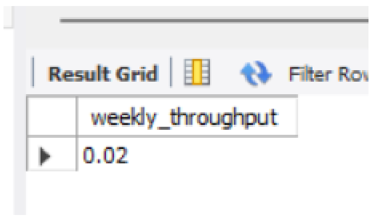
dates	jobs_reviewed
2020-11-30	180
2020-11-29	180
2020-11-28	218
2020-11-27	35
2020-11-26	64
2020-11-25	80
2020-11-24	42

2. **Throughput:** We will be calculating weekly throughput and daily throughput separately for better understanding. At first, we'll be calculating weekly throughput

SELECT ROUND(COUNT(event)/SUM(time_spent), 2) AS "weekly_throughput"

FROM sql_project_1

Hence, the weekly throughput is 0.02

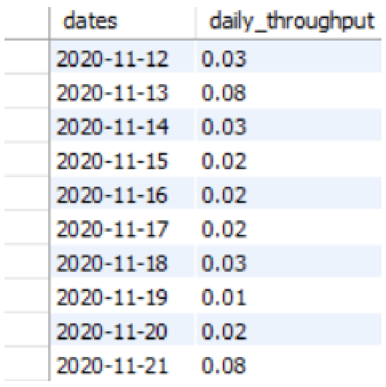


weekly_throughput
0.02

Now, moving on to calculating daily throughput

```
SELECT ds as dates, ROUND(COUNT(event)/SUM(time_spent), 2) AS "daily_throughput"

FROM sql_project_1

GROUP BY ds

ORDER by ds
```

Hence, we can see that on 2020-11-13 and 2020-11-21 the daily throughput is maximum i.e, 0.8



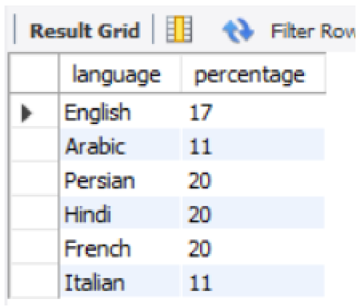
dates	daily_throughput
2020-11-12	0.03
2020-11-13	0.08
2020-11-14	0.03
2020-11-15	0.02
2020-11-16	0.02
2020-11-17	0.02
2020-11-18	0.03
2020-11-19	0.01
2020-11-20	0.02
2020-11-21	0.08

Now let's understand the fact that metrics are bound to rise up and fall down on daily and weekly basis. Therefore, we cannot deny the probability of achieving the results faster if the numbers are calculated on daily basis rather than weekly basis. Hence, rolling metrics will be great if calculated on daily purpose.

3. **Percentage share of language:** We'll be calculating the share of each language from last 30 days

```
SELECT language, ROUND(100*(COUNT(*)/total)) AS percentage

FROM sql_project_1

CROSS JOIN (SELECT COUNT(*) AS total FROM sql_project_1) as total_1

GROUP BY language
```

Hence, the following shows the percentage share of each language

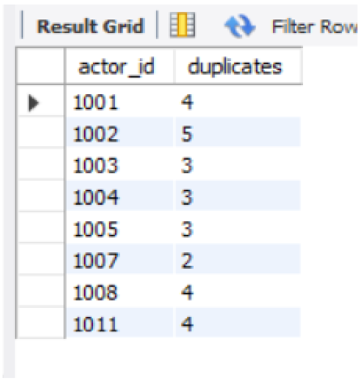


language	percentage
English	17
Arabic	11
Persian	20
Hindi	20
French	20
Italian	11

4. **Duplicate Rows:** Here we've been provided with a task to display the duplicates from the table. Since it has been stated that job_id and actor_id are unique identifier. Therefore, we will be checking duplicates for

For actor id

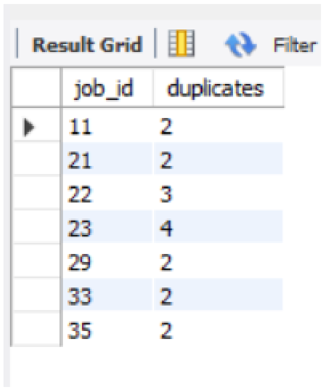
```
SELECT actor_id, COUNT(*) AS duplicates

FROM sql_project_1

GROUP BY actor_id

HAVING COUNT(*) > 1

ORDER BY actor_id;
```


Result Grid | Filter Row

actor_id	duplicates
1001	4
1002	5
1003	3
1004	3
1005	3
1007	2
1008	4
1011	4

For job_id

```
SELECT job_id, COUNT(*) AS duplicates

FROM sql_project_1

GROUP BY job_id

HAVING COUNT(*) > 1

ORDER BY job_id;
```


Result Grid | Filter

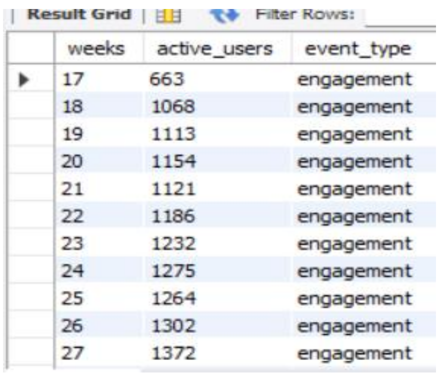
job_id	duplicates
11	2
21	2
22	3
23	4
29	2
33	2
35	2

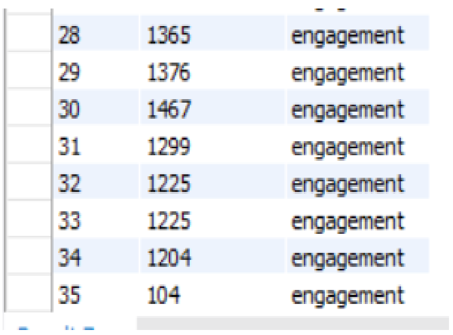
- **Investigating metric spike:**

1. **User Engagement:** We have to calculate the weekly user engagement so that we can determine the activeness of the user.

```
SELECT EXTRACT(WEEK FROM occurred_at) as "weeks", COUNT(DISTINCT user_id) AS "active_users", event_type

FROM events

WHERE event_type= 'engagement'

GROUP BY weeks
```

Upon execution we get the following results


Result Grid | Filter Rows:



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

2. **User Growth:** Here we have to calculate the amount of users growing over time.

SELECT months, monthly_users, ROUND(((monthly_users/LAG(monthly_users, 1) OVER (ORDER BY months) -1) *100)) AS "growth_in_%"

FROM

(

SELECT EXTRACT(month FROM created_at) AS months, COUNT(activated_at) AS monthly_users

FROM users

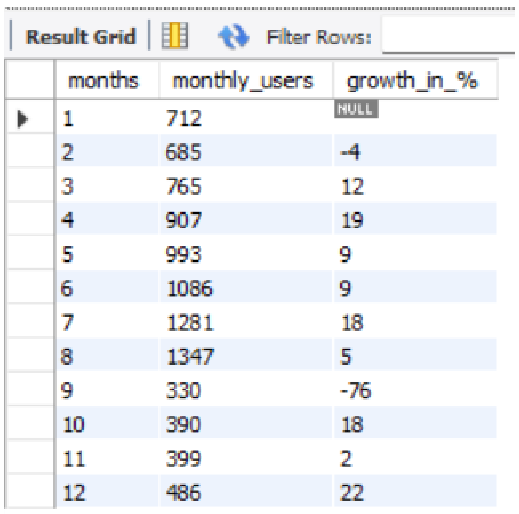
WHERE activated_at NOT IN ("")

GROUP BY months

ORDER BY months

) sub

Here, the LAG function is used to retrieve the user count from the previous month, which is then used to calculate the growth rate for the current month. The outer query outputs the month, monthly user count, and monthly growth rate for each month in the time period. Upon execution it gave the following results



months	monthly_users	growth_in_%
1	712	NULL
2	685	-4
3	765	12
4	907	19
5	993	9
6	1086	9
7	1281	18
8	1347	5
9	330	-76
10	390	18
11	399	2
12	486	22

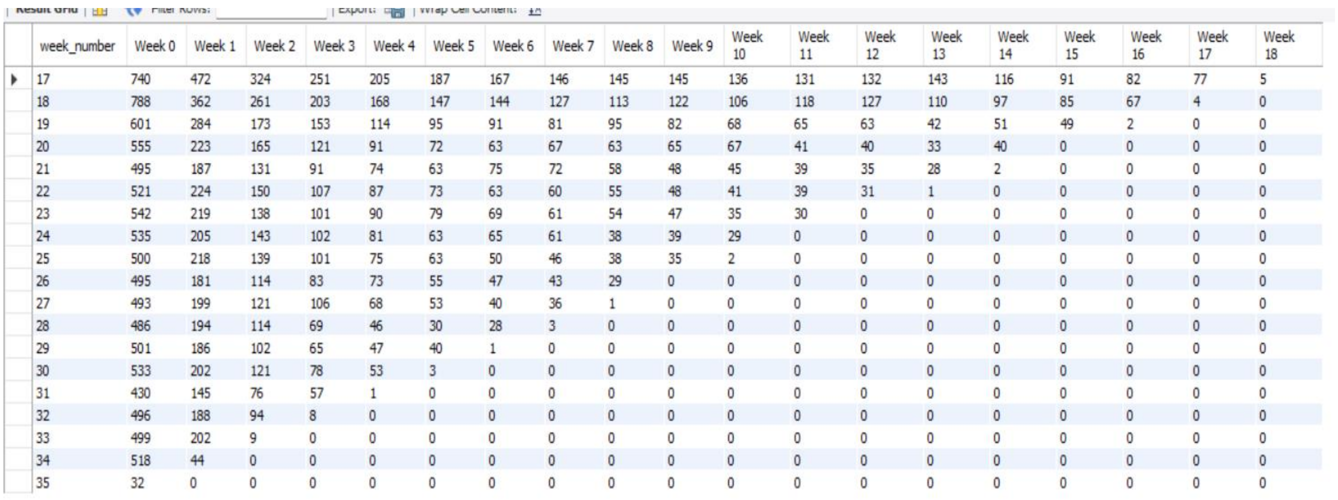
3. **Weekly Retention:** We have to calculate the number of users retained after signing up on weekly basis.

SELECT first AS "week_number",

SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS "Week 0",

```
SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS "Week 1",

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",

SUM(CASE WHEN week_number = 18 THEN 1 ELSE 0 END) AS "Week 18"

FROM

(

SELECT m.user_id, m.login_week, n.first, m.login_week - first AS week_number

FROM

(SELECT user_id, EXTRACT(WEEK FROM occurred_at) AS login_week FROM events

GROUP BY 1,2)m,

(SELECT user_id, MIN(EXTRACT(WEEK FROM occurred_at)) AS first FROM events

GROUP BY 1)n

WHERE m.user_id = n.user_id

) sub

GROUP BY first

ORDER BY first;
```

the query groups logins by users and calculates how many logins a user has had in each week since their first login. The output shows the number of users who logged in for each week since their first login. Upon execution we get the following results



week_number	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
17	740	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77	5
18	788	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4	0
19	601	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0	0
20	555	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0	0
21	495	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0	0
22	521	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0
23	542	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0
24	535	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0
25	500	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0
26	495	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0
27	493	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0
28	486	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0
29	501	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0
30	533	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0
31	430	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	496	188	94	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	499	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	518	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4. **Weekly Engagement:** Here we have to calculate weekly engagement of user per device.

SELECT EXTRACT(WEEK FROM occurred_at) AS "Week Numbers",

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("iphone 5") THEN user_id ELSE NULL END) AS "iPhone 5",

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 "Windows Surface",

COUNT(DISTINCT CASE WHEN device IN("macbook air") THEN user_id ELSE NULL END) AS "Macbook Air",

COUNT(DISTINCT CASE WHEN device IN("iphone 5s") THEN user_id ELSE NULL END) AS "iPhone 5S",

COUNT(DISTINCT CASE WHEN device IN("macbook pro") THEN user_id ELSE NULL END) AS "Macbook Pro",

COUNT(DISTINCT CASE WHEN device IN("kindle fire") THEN user_id ELSE NULL END) AS "Kindle Fire",

COUNT(DISTINCT CASE WHEN device IN("ipad mini") THEN user_id ELSE NULL END) AS "iPad Mini",

COUNT(DISTINCT CASE WHEN device IN("nexus 7") THEN user_id ELSE NULL END) AS "Nexus 7",

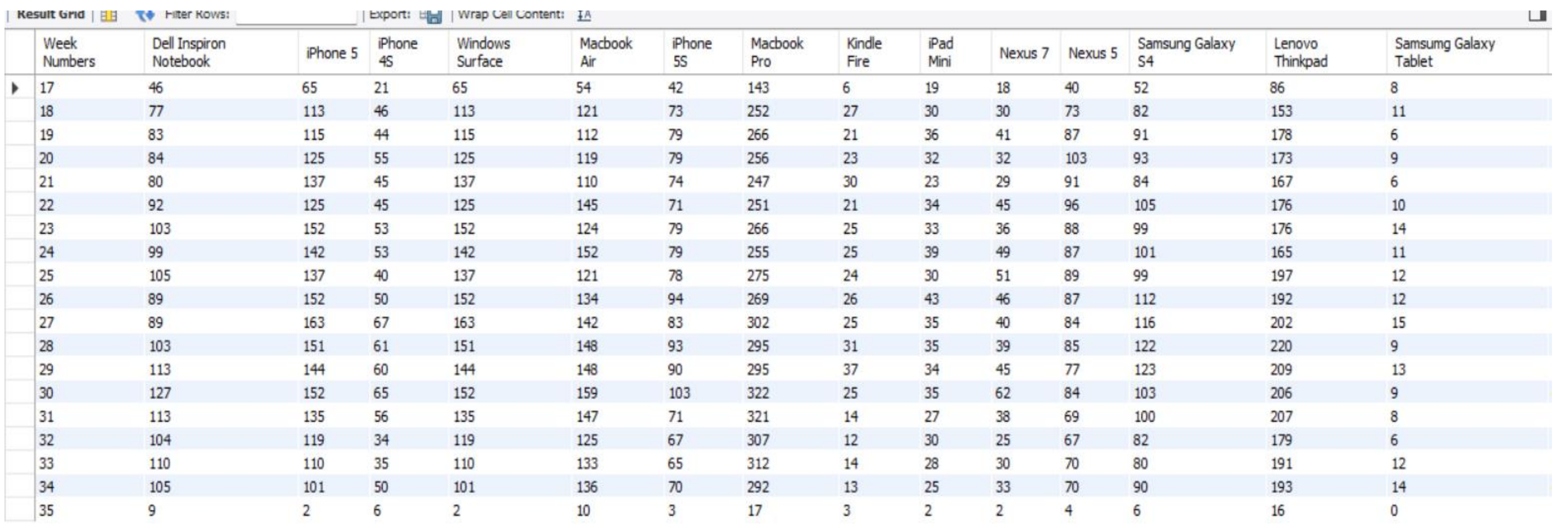
COUNT(DISTINCT CASE WHEN device IN("nexus 5") THEN user_id ELSE NULL END) AS "Nexus 5",

COUNT(DISTINCT CASE WHEN device IN("samsung galaxy s4") THEN user_id ELSE NULL END) AS "Samsung Galaxy S4",

COUNT(DISTINCT CASE WHEN device IN("Lenovo thinkpad") THEN user_id ELSE NULL END) AS "Lenovo Thinkpad",

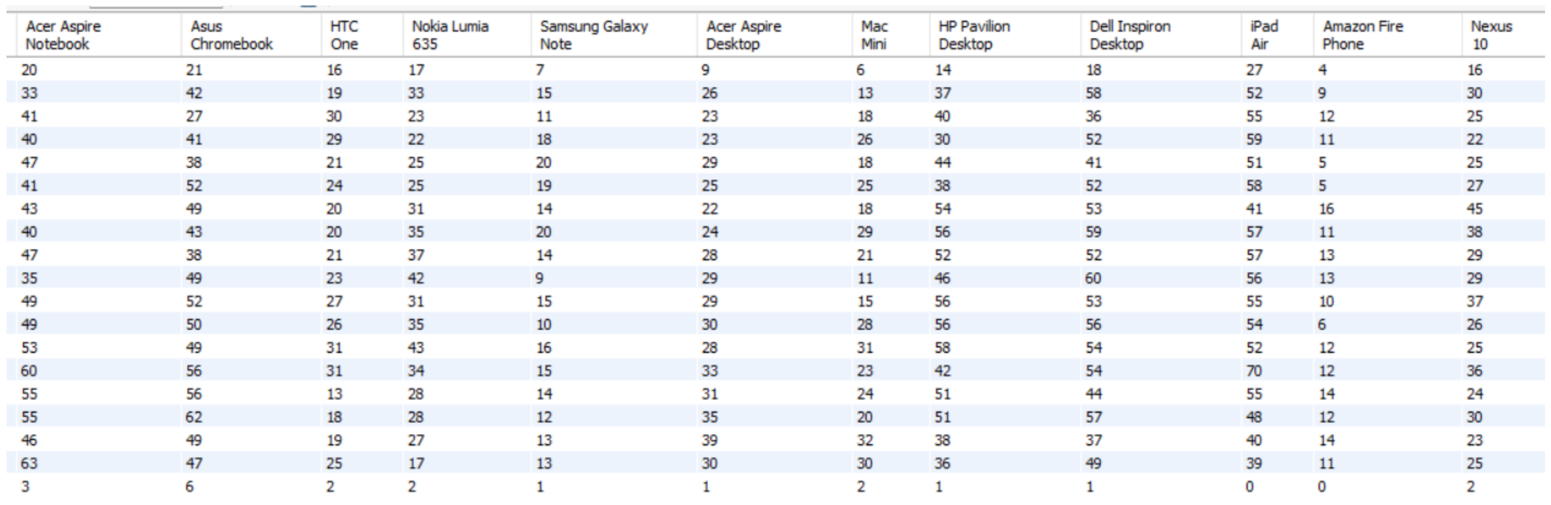
```
COUNT(DISTINCT CASE WHEN device IN("samsumg galaxy tablet") THEN user_id ELSE NULL END) AS
"Samsumg Galaxy Tablet",

COUNT(DISTINCT CASE WHEN device IN("acer aspire notebook") THEN user_id ELSE NULL END) AS
"Acer Aspire Notebook",

COUNT(DISTINCT CASE WHEN device IN("asus chromebook") THEN user_id ELSE NULL END) AS "Asus
Chromebook",

COUNT(DISTINCT CASE WHEN device IN("htc one") THEN user_id ELSE NULL END) AS "HTC One",

COUNT(DISTINCT CASE WHEN device IN("nokia lumia 635") THEN user_id ELSE NULL END) AS "Nokia
Lumia 635",

COUNT(DISTINCT CASE WHEN device IN("samsung galaxy note") THEN user_id ELSE NULL END) AS
"Samsung Galaxy Note",

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("mac mini") THEN user_id ELSE NULL END) AS "Mac Mini",

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("dell inspiron desktop") THEN user_id ELSE NULL END) AS
"Dell Inspiron Desktop",

COUNT(DISTINCT CASE WHEN device IN("ipad air") THEN user_id ELSE NULL END) AS "iPad Air",

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("nexus 10") THEN user_id ELSE NULL END) AS "Nexus 10"

FROM events

WHERE event_type = 'engagement'

GROUP BY 1
```

This SQL query selects data from the "events" table and extracts the week numbers from the "occurred_at" column. It then counts the number of unique user IDs associated with specific device types for each week. The results are grouped by week number. The query only includes events with the "engagement" event_type. Upon execution it gave the following results



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Week Numbers	Dell Inspiron Notebook	iPhone 5	iPhone 4S	Windows Surface	Macbook Air	iPhone 5S	Macbook Pro	Kindle Fire	iPad Mini	Nexus 7	Nexus 5	Samsung Galaxy S4	Lenovo Thinkpad	Samsumg Galaxy Tablet
17	46	65	21	65	54	42	143	6	19	18	40	52	86	8
18	77	113	46	113	121	73	252	27	30	30	73	82	153	11
19	83	115	44	115	112	79	266	21	36	41	87	91	178	6
20	84	125	55	125	119	79	256	23	32	32	103	93	173	9
21	80	137	45	137	110	74	247	30	23	29	91	84	167	6
22	92	125	45	125	145	71	251	21	34	45	96	105	176	10
23	103	152	53	152	124	79	266	25	33	36	88	99	176	14
24	99	142	53	142	152	79	255	25	39	49	87	101	165	11
25	105	137	40	137	121	78	275	24	30	51	89	99	197	12
26	89	152	50	152	134	94	269	26	43	46	87	112	192	12
27	89	163	67	163	142	83	302	25	35	40	84	116	202	15
28	103	151	61	151	148	93	295	31	35	39	85	122	220	9
29	113	144	60	144	148	90	295	37	34	45	77	123	209	13
30	127	152	65	152	159	103	322	25	35	62	84	103	206	9
31	113	135	56	135	147	71	321	14	27	38	69	100	207	8
32	104	119	34	119	125	67	307	12	30	25	67	82	179	6
33	110	110	35	110	133	65	312	14	28	30	70	80	191	12
34	105	101	50	101	136	70	292	13	25	33	70	90	193	14
35	9	2	6	2	10	3	17	3	2	2	4	6	16	0

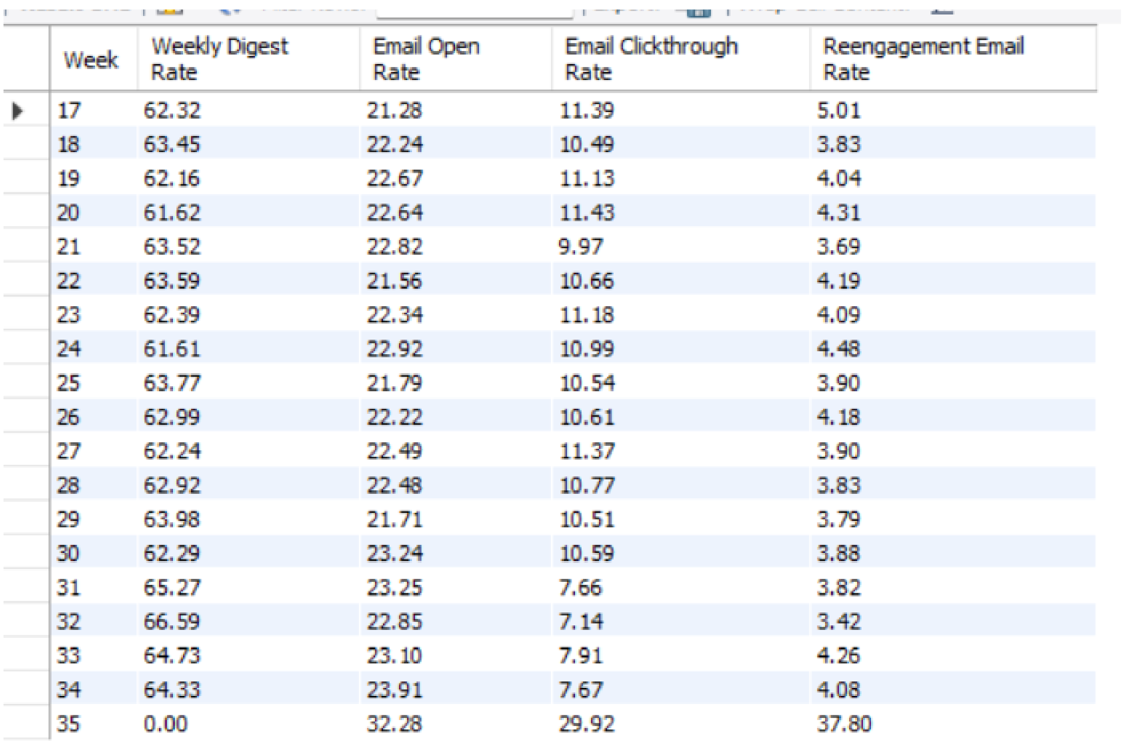


Acer Aspire Notebook	Asus Chromebook	HTC One	Nokia Lumia 635	Samsung Galaxy Note	Acer Aspire Desktop	Mac Mini	HP Pavilion Desktop	Dell Inspiron Desktop	iPad Air	Amazon Fire Phone	Nexus 10
20	21	16	17	7	9	6	14	18	27	4	16
33	42	19	33	15	26	13	37	58	52	9	30
41	27	30	23	11	23	18	40	36	55	12	25
40	41	29	22	18	23	26	30	52	59	11	22
47	38	21	25	20	29	18	44	41	51	5	25
41	52	24	25	19	25	25	38	52	58	5	27
43	49	20	31	14	22	18	54	53	41	16	45
40	43	20	35	20	24	29	56	59	57	11	38
47	38	21	37	14	28	21	52	52	57	13	29
35	49	23	42	9	29	11	46	60	56	13	29
49	52	27	31	15	29	15	56	53	55	10	37
49	50	26	35	10	30	28	56	56	54	6	26
53	49	31	43	16	28	31	58	54	52	12	25
60	56	31	34	15	33	23	42	54	70	12	36
55	56	13	28	14	31	24	51	44	55	14	24
55	62	18	28	12	35	20	51	57	48	12	30
46	49	19	27	13	39	32	38	37	40	14	23
63	47	25	17	13	30	30	36	49	39	11	25
3	6	2	2	1	1	2	1	1	0	0	2

5. **Email Engagement:** We have to calculate the email engagement metrics and check whether the users are engaging with the email services or not.

```
SELECT Week,

ROUND((weekly_digest/total*100),2) AS "Weekly Digest Rate",

ROUND((email_opens/total*100),2) AS "Email Open Rate",

ROUND((email_clickthroughs/total*100),2) AS "Email Clickthrough Rate",

ROUND((reengagement_emails/total*100),2) AS "Reengagement Email Rate"

FROM

(

SELECT EXTRACT(WEEK FROM occurred_at) AS Week,

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) AS
email_clickthroughs,

COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id ELSE NULL END) AS
reengagement_emails,

COUNT(user_id) AS total

FROM emails

GROUP BY 1

) sub

GROUP BY 1

ORDER BY 1;
```

This SQL query is used to calculate the rates of various email-related actions for each week. It calculates the rates of each action by dividing the count of that action by the total number of users and multiplying it by 100 to get the percentage. Upon execution we get the following results



Week	Weekly Digest Rate	Email Open Rate	Email Clickthrough Rate	Reengagement Email Rate
17	62.32	21.28	11.39	5.01
18	63.45	22.24	10.49	3.83
19	62.16	22.67	11.13	4.04
20	61.62	22.64	11.43	4.31
21	63.52	22.82	9.97	3.69
22	63.59	21.56	10.66	4.19
23	62.39	22.34	11.18	4.09
24	61.61	22.92	10.99	4.48
25	63.77	21.79	10.54	3.90
26	62.99	22.22	10.61	4.18
27	62.24	22.49	11.37	3.90
28	62.92	22.48	10.77	3.83
29	63.98	21.71	10.51	3.79
30	62.29	23.24	10.59	3.88
31	65.27	23.25	7.66	3.82
32	66.59	22.85	7.14	3.42
33	64.73	23.10	7.91	4.26
34	64.33	23.91	7.67	4.08
35	0.00	32.28	29.92	37.80

Insights and Results: This project provided enough knowledge to me about the industry needs on operation analytics. Slowly and steadily I came to understand the importance of operational analytics into business world. It helps in creating a bridge of connection between the customers and the products. Also, helps the authority to take neccasary steps so that they could serve the customers in every possible way.

- Learning experience: This project has been significant milestone for me as it has provided a distinct step by step process of doing carrying out queries using various new functions. It was way more challenging than the previous one as it involved a bit of critical thinking as well. Investigating metric spike is the trickiest one. While importing the CSV files to the schema, I encountered a problem where the events file won't load to mysql workbench as it was quite huge and a column has been showing an error constantly. Doing a bit of google search I cam to know that two major things needed to be done. At first, I disabled the strictness of mysql workbench and secondly converted the user_type column to VARCHAR as there were some values that doesn't fit the given datatype. This project evolved my analyzing attributes to a whole new level.
- Time Management: The time given for this project was enough for me as I had already completed the previous project before the deadline. This gave me some extra time to work on my project. The most time consuming part was on when I encountered the error while transferring the file to the schema.
- Attention to detail: This project requires high level of attention for a newbie like me. There were many challenges that needed to precision and attention at the same time. There were some places where I got stuck and it took too much time.
- Collaborations: It would be really unfair if I don't mention the websites who helped me completing this project by providing the essential concepts. I find these websites to be very helpful for anyone with basic language understanding.
  Stack overflow: www.stackoverflow.com
  SQL Tutorial: https://www.sqltutorial.org/
  Trainity Learning: https://trainity.link/data/learning
  Learn SQL: www.learnsql.com
- Practical application: Completing this project gave me an idea of operation analytics. It enhanced my capabilities of understanding the problem and also helped me by providing a detailed analysis of the operational problem on industrial level.

Tech-Stack Used: The following softwares has been used during the process

- My SQL Workbench 8.0 CE
- Microsoft Word 2019

Drive Link:

https://drive.google.com/drive/folders/16o-EQhXx3339ZOF6zLno4vRRmOz4vCWM
under the name "project_3"

Prepared by: Sushant Karmakar