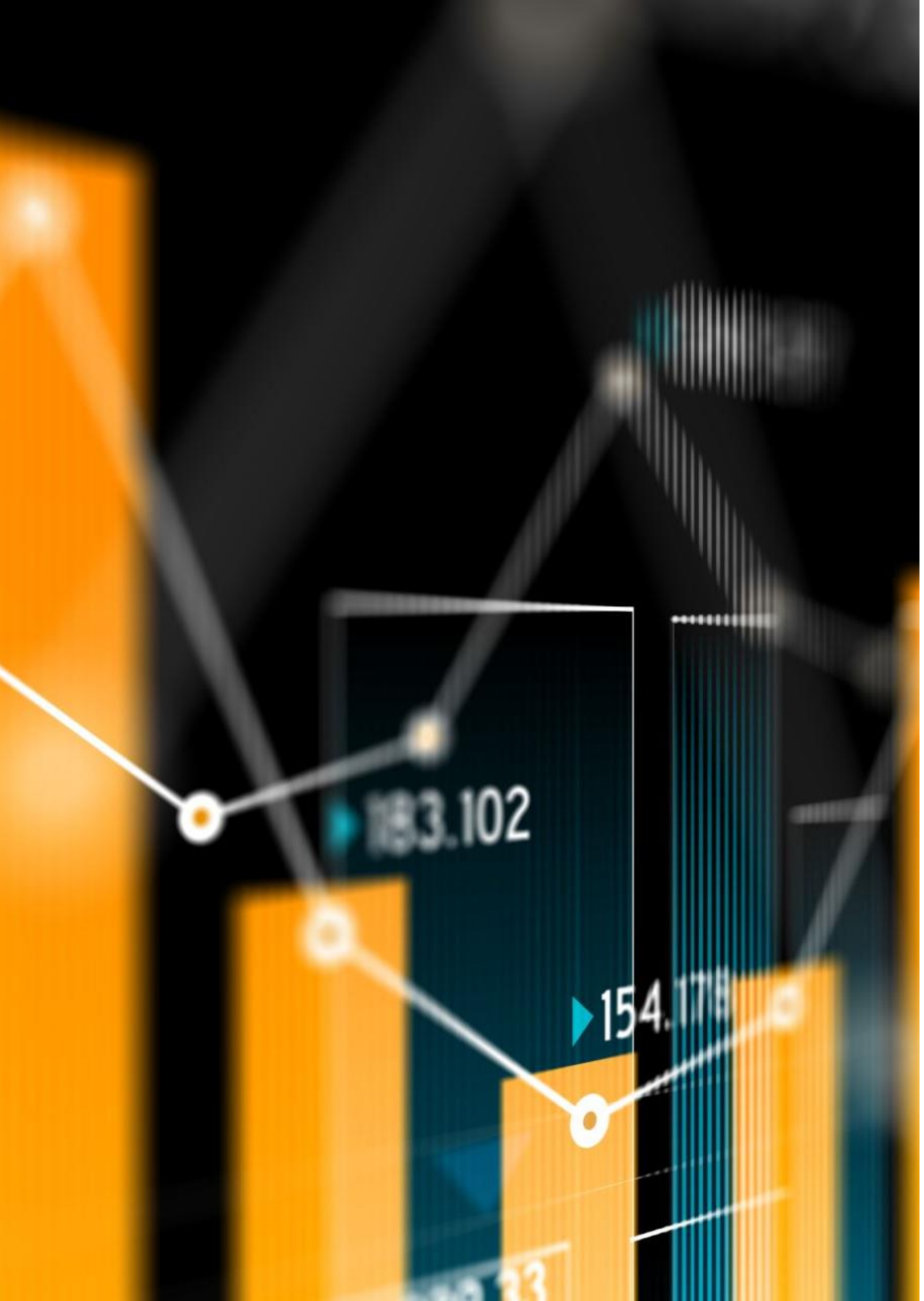
Operation Analytics and Investigating Metric Spike

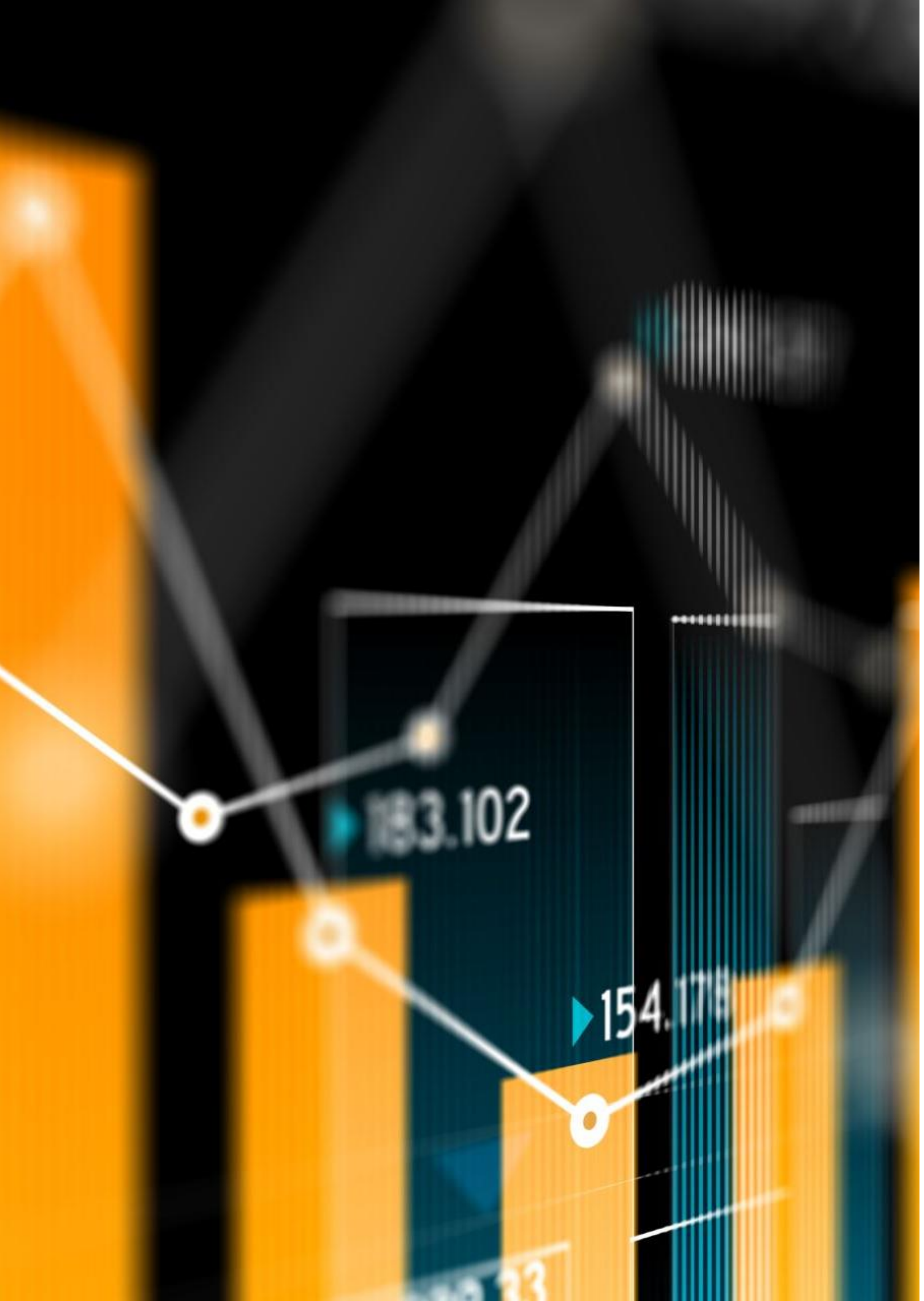


Operation Analytics and Investigating Metric Spike

Project Description:

The project is about finding the insights needed in the two case studies which is job data and the investing metric spike.

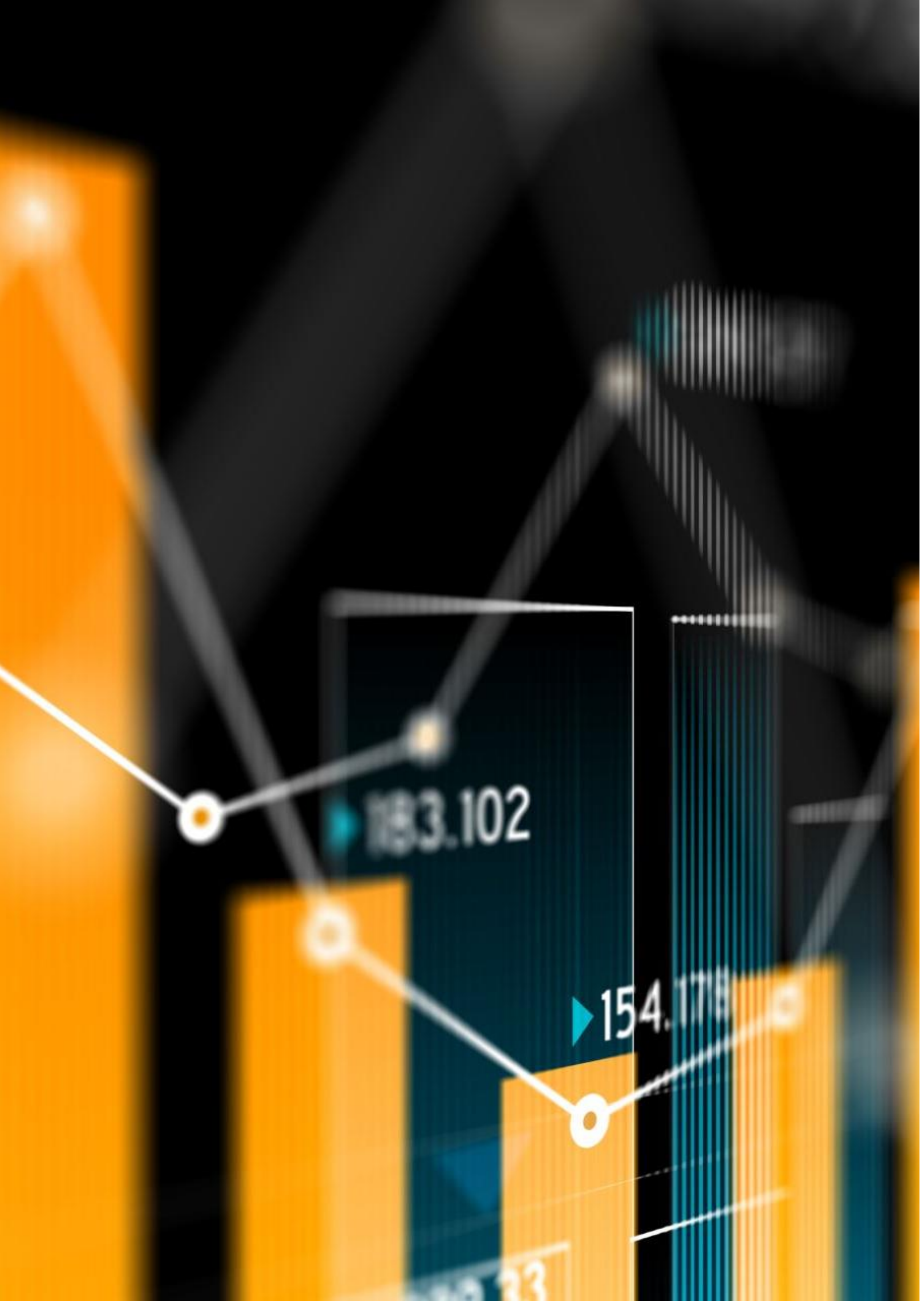
The tools used to answer the question asked by the stakeholders is MySQL.



Operation Analytics and Investigating Metric Spike

Approach :

First the dataset have been loaded in MySQL database then the database have been thoroughly studied to understand the attributes then used multiple queries to find out the answers for the questions asked by the stakeholders.



Operation Analytics and Investigating Metric Spike

Tech-Stack Used:

The software used to create the solution for this project is MySQL Workbench and the version of the software is 8.0.32. It is most popular and easy way to access the MySQL Database.

I think it will be more efficient for this project.

Operation Analytics and Investigating Metric Spike

Insights:

Case Study 1 (Job Data):

A. Number of jobs reviewed:

SQL query:

```
SELECT ds, round(count(*)/(sum(time_spent)/3600))
AS jobs_reviewed_per_hour_per_day_for_November_2020
FROM job_data GROUP BY ds ORDER BY ds;
```

Output:

ds	jobs_reviewed_per_hour_per_day_for_November
2020-11-01	218
2020-11-02	180
2020-11-03	180
2020-11-04	240
2020-11-05	111
2020-11-06	64
2020-11-07	30
2020-11-08	218
2020-11-09	180
2020-11-10	144
2020-11-11	216
2020-11-12	71
2020-11-13	103
2020-11-14	327
2020-11-15	164
2020-11-16	180
2020-11-17	180
2020-11-18	240
2020-11-19	89
2020-11-20	120
2020-11-21	327
2020-11-22	171
2020-11-23	180
2020-11-24	240
2020-11-25	111
2020-11-26	64
2020-11-27	144
2020-11-28	218
2020-11-29	180
2020-11-30	180

Operation Analytics and Investigating Metric Spike

Insights:

Case Study 1 (Job Data):

B. Throughput:

I prefer 7 day rolling average if there is more variance in the datapoints so that the graph generating it will be more smooth by reducing the noise or it could be useful in comparing weekly average else if we need to specifically compare daily then daily average should be used.

SQL query: (Daily)

```
SELECT ds, count(*)/sum(time_spent) AS
jobs_reviewed_per_hour_per_day_for_November_2020 FROM job_data
GROUP BY ds ORDER BY ds ;
```

Output:

ds	jobs_reviewed_per_hour_per_day_for_November
2020-11-01	0.0606
2020-[illegible]	[illegible]00
2020-11-03	0.0500
2020-11-04	0.0667
2020-11-05	0.0308
2020-11-06	0.0179
2020-11-07	0.0083
2020-11-08	0.0606
2020-11-09	0.0500
2020-11-10	0.0400
2020-11-11	0.0600
2020-11-12	0.0198
2020-11-13	0.0286
2020-11-14	0.0909
2020-11-15	0.0455
2020-11-16	0.0500
2020-11-17	0.0500
2020-11-18	0.0667
2020-11-19	0.0248
2020-11-20	0.0333
2020-11-21	0.0909
2020-11-22	0.0476
2020-11-23	0.0500
2020-11-24	0.0667
2020-11-25	0.0308
2020-11-26	0.0179
2020-11-27	0.0400
2020-11-28	0.0606
2020-11-29	0.0500
2020-11-30	0.0500

Operation Analytics and Investigating Metric Spike

Insights:

Case Study 1 (Job Data):

B. Throughput:

SQL query: (Weekly)

```
SELECT weekday(ds) AS week_day,
count(*)/sum(time_spent) AS
jobs_reviewed_per_hour_per_day_for_November_2020
FROM job_data GROUP BY week_day ORDER BY week_day;
```

Operation Analytics and Investigating Metric Spike

Output:

week_day	jobs_reviewed_per_hour_per_day_for_November
0	0.0500
1	0.0500
2	0.0483
3	0.0233
4	0.0274
5	0.0286
6	0.0533

0.0533

Operation Analytics and Investigating Metric Spike

Insights:

Case Study 1 (Job Data):

C. Percentage share of each language:

SQL query:

```
SELECT `language`, (count(*)/(SELECT count(*) FROM job_data)) * 100 as percent FROM job_data GROUP BY `language` ;
```

Operation Analytics and Investigating Metric Spike

Output:

language	percent
English	11.1111
Arabic	20.0000
Persian	40.0000
Hindi	11.1111
French	8.8889
Italian	8.8889

Operation Analytics and Investigating Metric Spike

Insights:

Case Study 1 (Job Data):

D. Duplicate rows:

SQL query:

```
SELECT job_id, count(*) as duplicate_counts FROM job_data GROUP BY job_id HAVING count(*) > 1;
```

Operation Analytics and Investigating Metric Spike

Output:

job_id	duplicate_counts
21	2
22	4
23	4
20	4
18	2
30	2
65	2
90	2
27	2

Operation Analytics and Investigating Metric Spike

Insights:

Case Study 2 (Investing metric spike):

A. User Engagement:

SQL query:

```
select extract(week from occured_at) as week_in_numbers, count(distinct user_id) as count_of_active_users from events  where event_type = 'engagement' group by week_in_numbers;
```

Operation Analytics and Investigating Metric Spike

Output:

week_in_numbers	count_of_active_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

Operation Analytics and Investigating Metric Spike

B. User Growth:

SQL query:

```
select month_in_numbers, count_of_users,
round(((count_of_users-lag(count_of_users, 1) over())/
lag(count_of_users, 1) over()) * 100, 2) as
user_growth_over_time from
```

```
(select extract(month from created_at) as month_in_numbers,
 count(activated_at) as count_of_users
 from users where extract(year from created_at) = 2013
 group by month_in_numbers
  union all

select extract(month from created_at) as month_in_numbers,
 count(activated_at) as count_of_users
 from users where extract(year from created_at) = 2014
 group by month_in_numbers
 ) as a;
```

Output:

month_in_numbers	count_of_users	user_growth_over_time
1	320	NULL
2	320	0.00
3	300	-6.25
4	362	20.67
5	428	18.23
6	426	-0.47
7	568	33.33
8	632	11.27
9	660	4.43
10	780	18.18
11	[illegible]	2.31
12	[illegible]	21.80

1	1104	13.58
2	1050	-4.89
3	1230	17.14
4	1452	18.05
5	1558	7.30
6	1746	12.07
7	1994	14.20
8	2062	3.41

Operation Analytics and Investigating Metric Spike

C. Weekly Retention:

SQL query:

```
select distinct week_number,first_week, count(week_number) over(partition by week_number, first_week order by first_week)  as total_users_retained__by_week FROM (select a.user_id, a.login_week, b.first_week, a.login_week - b.first_week as week_number from(select user_id, extract(week from occured_at) as login_week from events group by user_id, login_week ) as a,(select user_id, min(extract(week from occured_at)) as first_week from events group by user_id) as b where a.user_id = b.user_id) as c;
```

Output: (only week 0 output since it would consume lot of space)

week_number	first_week	total_users_retained__by_week
0	17	740
0	18	788
0	19	601
0	20	555
0	21	495
0	22	521
0	23	542
0	24	535
0	25	500
0	26	495
0	27	493
0	28	486
0	29	501
0	30	533
0	31	430
0	32	496
0	33	499
0	34	518
0	35	32

D. Weekly Engagement:

SQL query:

select a.week_number, a.event_name, total_engagement_by_week_over_event_name,total_engagement_by_week , over_all_total_engagement_by_event_name from

(select extract(week from occured_at) as week_number, event_name, count(*) as total_engagement_by_week_over_event_name

from events where event_type = 'engagement' group by week_number, event_name order by week_number) as a

```
inner join

(select event_name, count(*) as
over_all_total_engagement_by_event_name from events group by
event_name) as b

on a.event_name = b.event_name inner join

(select extract(week from occured_at) as week_number, count(*)
as total_engagement_by_week
from events group by week_number) as c

on a.week_number = c.week_number

order by week_number;
```

Output: (only week 17 output since it would consume lot of space)

week_number	event_name	total_engagement_by_week_over_event_name	total_engagement_by_week	over_all_total_engagement_by_event_name
17	login	887	8404	38610
17	home_page	2341	8404	94065
17	like_message	1520	8404	59248
17	view_inbox	1376	8404	55936
17	search_run	340	8404	13019
17	send_message	826	8404	33105
17	search_autocomplete	394	8404	17820
17	search_click_result_10	15	8404	506
17	search_click_result_7	36	8404	709
17	search_click_result_8	12	8404	690
17	search_click_result_1	43	8404	1413
17	search_click_result_3	40	8404	1134
17	search_click_result_2	55	8404	1499
17	search_click_result_5	25	8404	968
17	search_click_result_6	40	8404	805
17	search_click_result_9	25	8404	784
17	search_click_result_4	44	8404	1264

E. Email Engagement:

SQL query: (number of users engaging in email_events)

```
select action , count(*) from email_events group by action;
```

Output :

action	count(*)
sent_weekly_digest	57267
email_open	20459
email_clickthrough	9010
sent_reengagement_email	3653

SQL query: (Distinct number users engaging in email_events)

```
select action , count(distinct user_id) from email_events group by action;
```

Output:

action	count(distinct user_id)
email_clickthrough	5277
email_open	5927
sent_reengagement_email	3653
sent_weekly_digest	4111

SQL query: (number users engaging in email_events by week)

select distinct week_number, action , count(action) over(partition by action, week_number order by action) as countfrom(select distinct a.user_id, a.action, week_number from(select user_id, action, extract(week from occured_at) as week_number from email_events order by week_number) as a,email_events as bwhere a.user_id = b.user_id) as corder by week_number;

Output:

week_number	action	count
17	email_clickthrough	166
17	sent_weekly_digest	908
17	email_open	310
17	sent_reengagement_email	73
18	email_clickthrough	425
18	sent_weekly_digest	2602
18	sent_reengagement_email	157
18	email_open	900
19	email_clickthrough	476
19	sent_weekly_digest	2665
19	email_open	961
19	sent_reengagement_email	173
20	email_clickthrough	501
20	sent_weekly_digest	2733
20	email_open	989
20	sent_reengagement_email	191
21	email_clickthrough	436
21	sent_weekly_digest	2822
21	email_open	996
21	sent_reengagement_email	164
22	email_clickthrough	478
22	sent_weekly_digest	2911
22	email_open	965
22	sent_reengagement_email	192
23	email_clickthrough	529
23	sent_weekly_digest	3003
23	email_open	1057
23	sent_reengagement_email	197
24	email_clickthrough	549
24	sent_weekly_digest	3105
24	email_open	1136
24	sent_reengagement_email	226
25	email_clickthrough	524
25	email_open	1084
25	sent_reengagement_email	196
25	sent_weekly_digest	3207
26	email_clickthrough	550
26	email_open	1149
26	sent_reengagement_email	219
26	sent_weekly_digest	3302

27	email_open	1207
27	sent_reengagement_email	213
27	email_clickthrough	613
27	sent_weekly_digest	3399
28	email_open	1228
28	sent_reengagement_email	213
28	email_clickthrough	594
28	sent_weekly_digest	3499
29	email_open	1201
29	sent_reengagement_email	213
29	email_clickthrough	583
29	sent_weekly_digest	3592
30	email_open	1363
30	sent_reengagement_email	231
30	email_clickthrough	625
30	sent_weekly_digest	3706
31	email_open	1338
31	sent_reengagement_email	222
31	email_clickthrough	444
31	sent_weekly_digest	3793

32	email_open	1318
32	email_clickthrough	416
32	sent_reengagement_email	200
32	sent_weekly_digest	3897
33	email_clickthrough	490
33	sent_reengagement_email	264
33	email_open	1417
33	sent_weekly_digest	4012
34	email_clickthrough	481
34	email_open	1502
34	sent_reengagement_email	261
34	sent_weekly_digest	4111
35	email_clickthrough	38
35	email_open	41
35	sent_reengagement_email	48

Results:

In case study there was insufficient data so I had to generate some data for my dataset using the sample data given which was interesting and bit challenging.

In case study 2 the dataset was so huge it was taking tremendous amount of time to import then I tried to learn different approach like Load data but it got failed somehow then I found a tool which was mentioned in the forum section of our learning platform where I learned about csv to SQL convertor where I learned to convert the file and I still had issues because of the empty spaces in the datetime datatype so I just imported as text and then replaced the white spaces with null values and then converted into the datetime datatype which was really challenging.