T TRAINITY

PROJECT: OPERATIONS & METRIC ANALYSIS

SUBMITTED BY: MUDIT VYAS

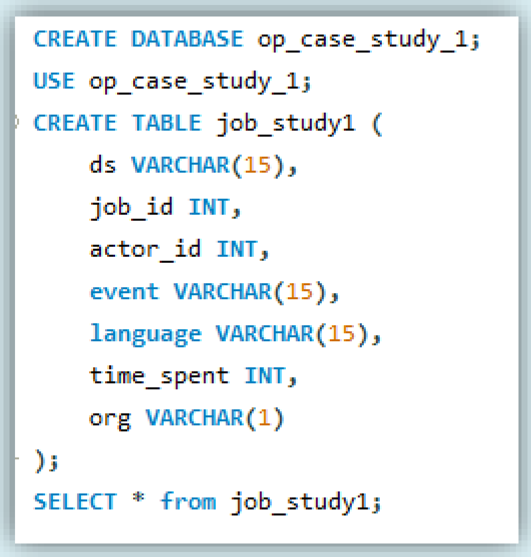
Aim: To use advanced SQL skills to analyse the data and provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.

Project Overview: Feedback and review is backbone for any organisation, task or project. For Many Companies, review of daily data is necessary to optimise their daily operations. Operational Analytics is a crucial process that involves analysing a company's end-to-end operations. This analysis helps identify areas for improvement within the company.
Investigating metric spike is one of the important aspect of Operational Analytics. The analysis of sudden changes in metric spikes may help company to review their methods and executions to gain maximum profitable outcomes. In this project we have used SQL queries to filter given data and answer questionnaires useful of different departments.

Tech-Stack Used: To write and execute SQL queries, MySQL Workbench 8.0 CE is used due to it's user friendly dashboard, user interference, connectivity, security, time analysis of queries and large data handling capabilities.

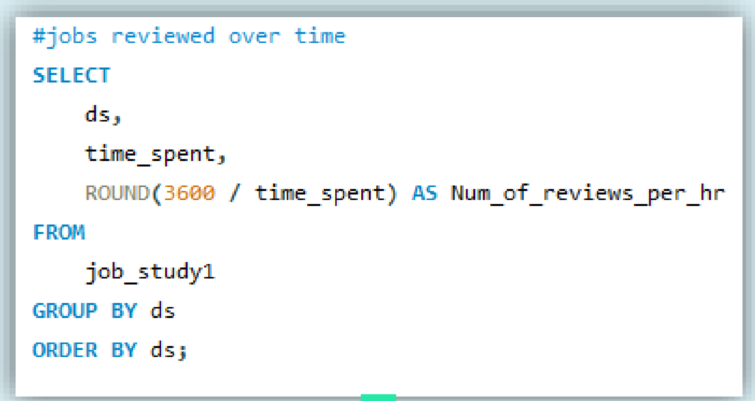
Case Study 1: Job Data Analysis

Data Preparation: Based on sample data table of 8 rows given, we formed a data table of about 250 rows. The Table was made in Excel only, as it was quick and can focus later on writing queries only, although using SQL queries also data could be entered using INSERT INTO clause and entering tuples from csv file.
Database is created and table is created in it. Using Import Table Wizard data was imported into table.

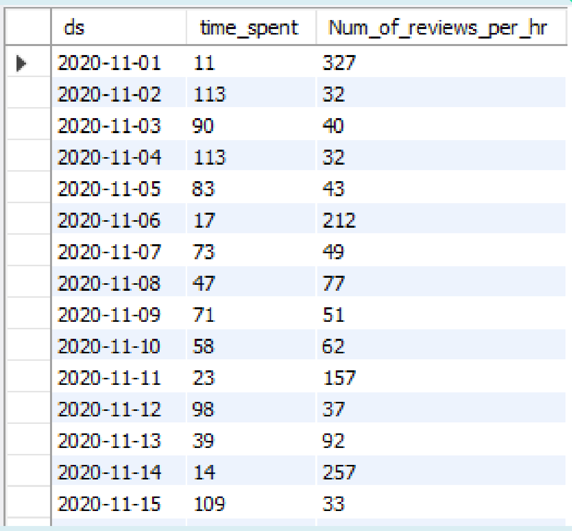
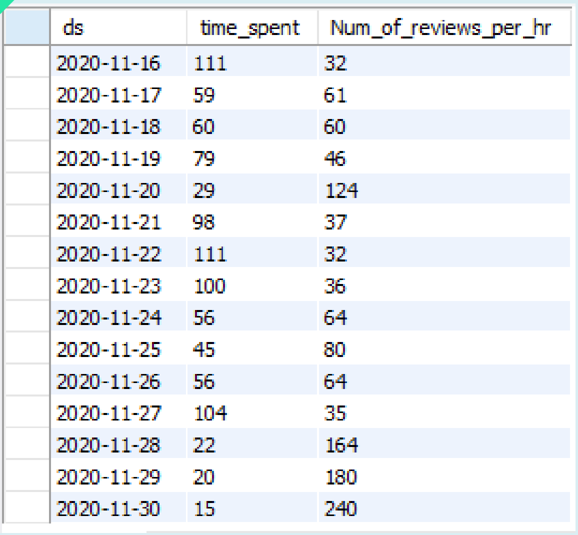


```sql
CREATE DATABASE op_case_study_1;
USE op_case_study_1;
CREATE TABLE job_study1 (
    ds VARCHAR(15),
    job_id INT,
    actor_id INT,
    event VARCHAR(15),
    language VARCHAR(15),
    time_spent INT,
    org VARCHAR(1)
);
SELECT * from job_study1;
```

A) Jobs Reviewed over Time: To check jobs reviewed per hour each day in November 2020. This data shall help to check efficiency of task performance.

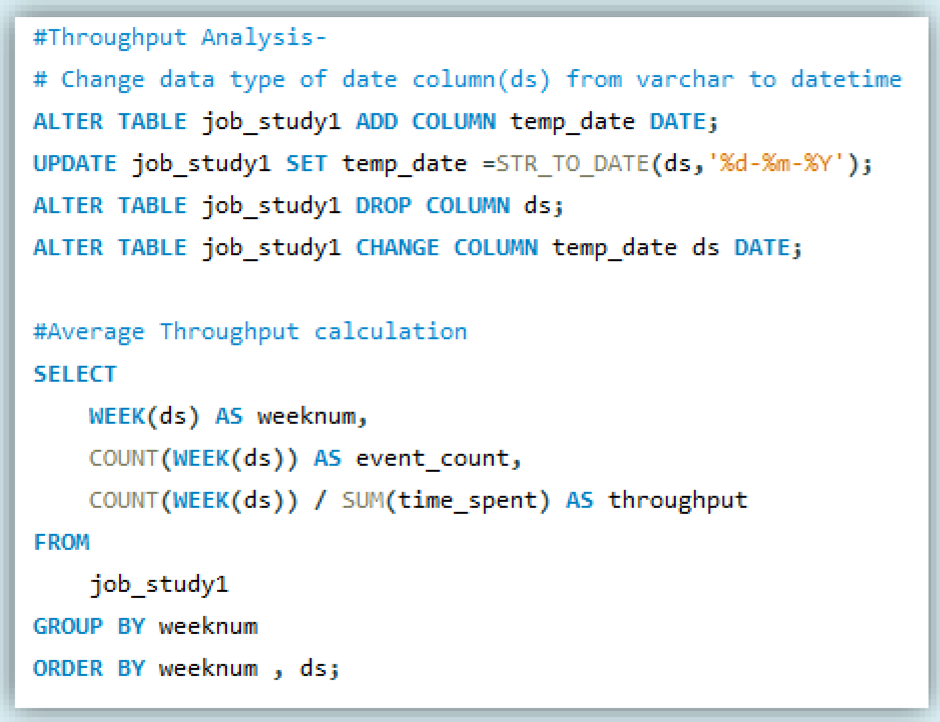


```sql
#jobs reviewed over time
SELECT
    ds,
    time_spent,
    ROUND(3600 / time_spent) AS Num_of_reviews_per_hr
FROM
    job_study1
GROUP BY ds
ORDER BY ds;
```

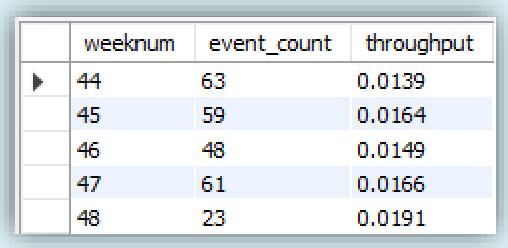



ds	time_spent	Num_of_reviews_per_hr
2020-11-01	11	327
2020-11-02	113	32
2020-11-03	90	40
2020-11-04	113	32
2020-11-05	83	43
2020-11-06	17	212
2020-11-07	73	49
2020-11-08	47	77
2020-11-09	71	51
2020-11-10	58	62
2020-11-11	23	157
2020-11-12	98	37
2020-11-13	39	92
2020-11-14	14	257
2020-11-15	109	33
2020-11-16	111	32
2020-11-17	59	61
2020-11-18	60	60
2020-11-19	79	46
2020-11-20	29	124
2020-11-21	98	37
2020-11-22	111	32
2020-11-23	100	36
2020-11-24	56	64
2020-11-25	45	80
2020-11-26	56	64
2020-11-27	104	35
2020-11-28	22	164
2020-11-29	20	180
2020-11-30	15	240

B) Throughput Analysis: It is useful to know the engagement of users with product, in order to check proper functionality of system. So, we check the 7-day rolling average of throughput (i.e. number of events per second)



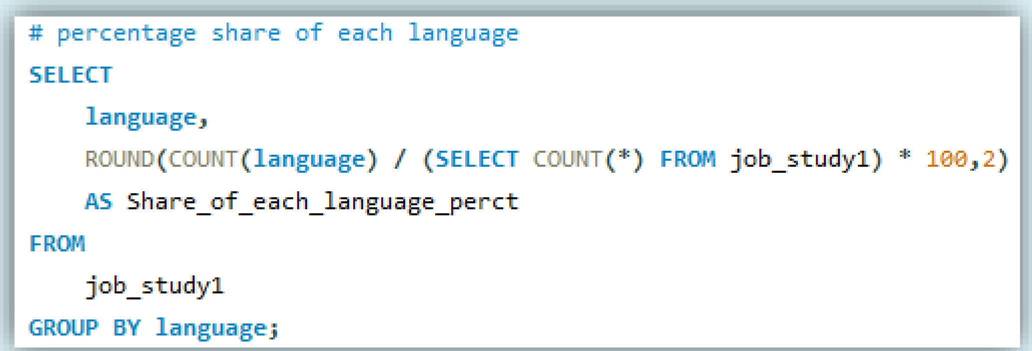
```sql
#Throughput Analysis-
# Change data type of date column(ds) from varchar to datetime
ALTER TABLE job_study1 ADD COLUMN temp_date DATE;
UPDATE job_study1 SET temp_date =STR_TO_DATE(ds,'%d-%m-%Y');
ALTER TABLE job_study1 DROP COLUMN ds;
ALTER TABLE job_study1 CHANGE COLUMN temp_date ds DATE;

#Average Throughput calculation
SELECT
    WEEK(ds) AS weeknum,
    COUNT(WEEK(ds)) AS event_count,
    COUNT(WEEK(ds)) / SUM(time_spent) AS throughput
FROM
    job_study1
GROUP BY weeknum
ORDER BY weeknum , ds;
```



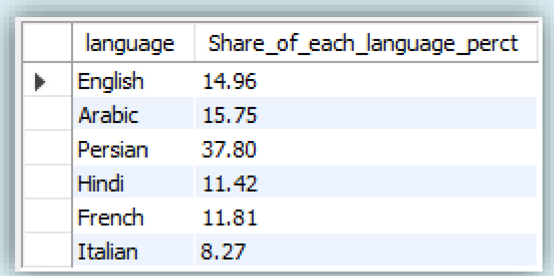
weeknum	event_count	throughput
44	63	0.0139
45	59	0.0164
46	48	0.0149
47	61	0.0166
48	23	0.0191

For Throughput Analysis, daily analysis would be preferred over 7 day average as results are almost similar hence it would be difficult to find any sensible conclusion.

<u>C) Language Share Analysis:</u> Percentage share of each language for the past 30 days.

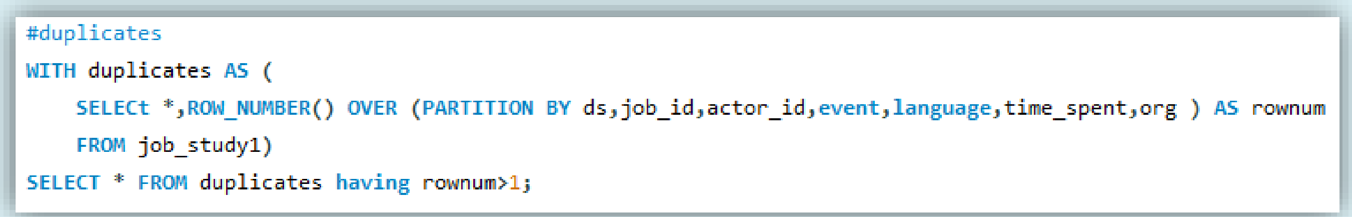


```
# percentage share of each language
SELECT
    language,
    ROUND(COUNT(language) / (SELECT COUNT(*) FROM job_study1) * 100,2)
    AS Share_of_each_language_perct
FROM
    job_study1
GROUP BY language;
```

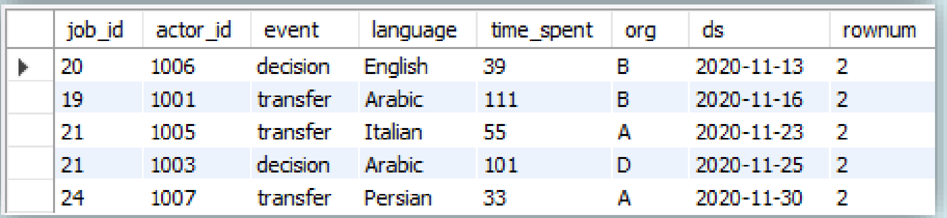


language	Share_of_each_language_perct
English	14.96
Arabic	15.75
Persian	37.80
Hindi	11.42
French	11.81
Italian	8.27

<u>D) Duplicate Row Detection:</u> To identify duplicate rows in the data.



```
#duplicates
WITH duplicates AS (
    SELECT *,ROW_NUMBER() OVER (PARTITION BY ds,job_id,actor_id,event,language,time_spent,org ) AS rownum
    FROM job_study1)
SELECT * FROM duplicates having rownum>1;
```

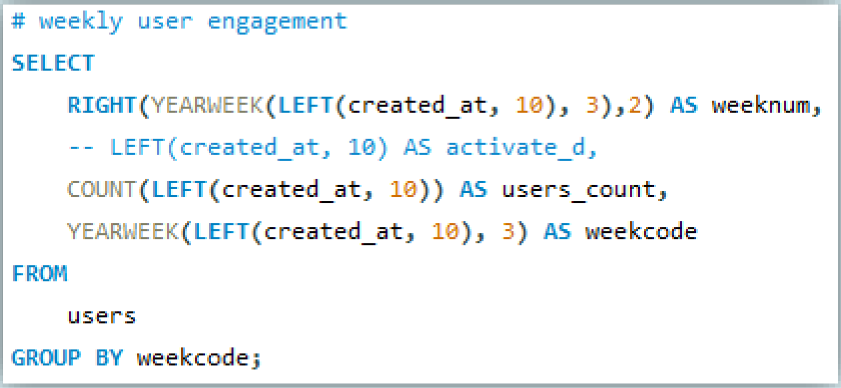


job_id	actor_id	event	language	time_spent	org	ds	rownum
20	1006	decision	English	39	B	2020-11-13	2
19	1001	transfer	Arabic	111	B	2020-11-16	2
21	1005	transfer	Italian	55	A	2020-11-23	2
21	1003	decision	Arabic	101	D	2020-11-25	2
24	1007	transfer	Persian	33	A	2020-11-30	2

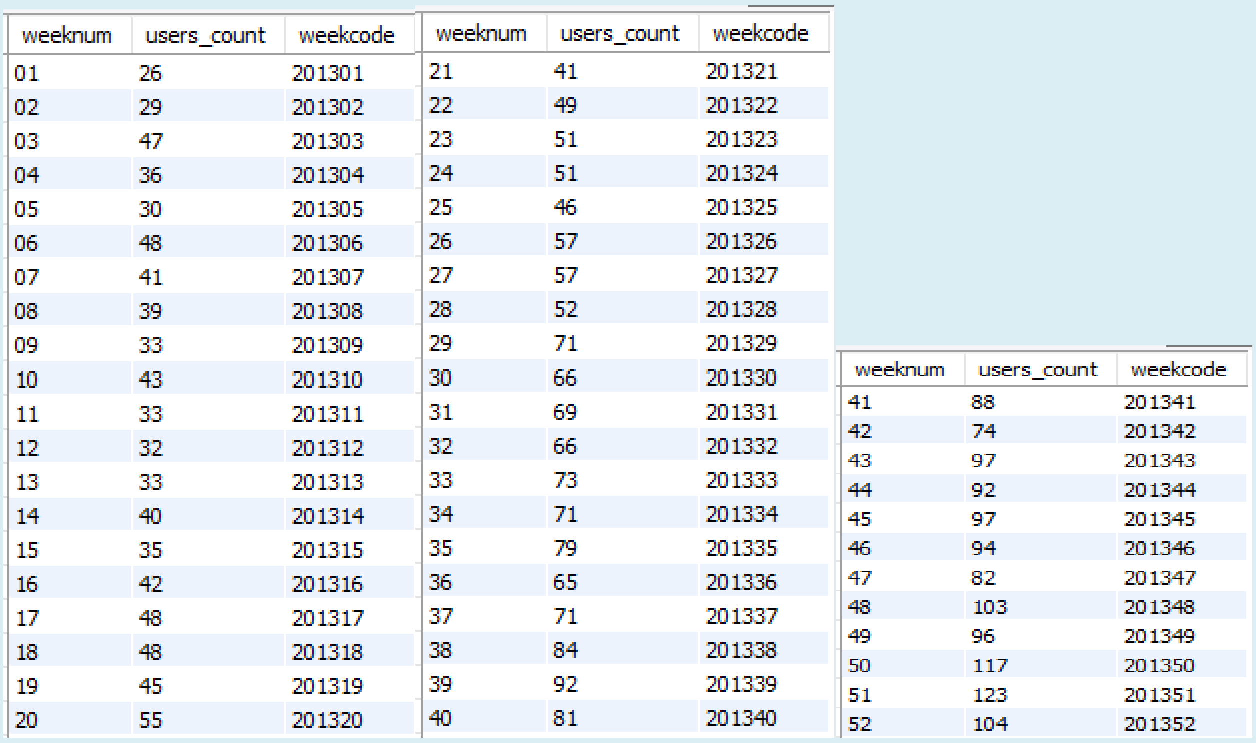
Case Study 2: Investigating Metric Spike

The 3 tables used in case study named ‘users’, ‘events’ and ‘email_events’ would take way longer time to upload using Import Wizard. An alternate method was used to import tables. Datatype of date-time column is also modified from VARCHAR to DATETIME.

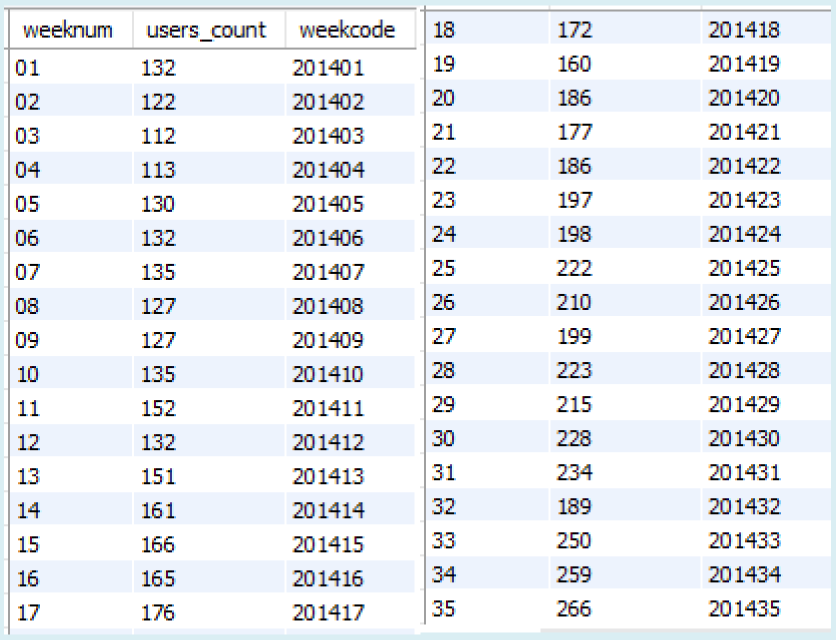
A) Weekly User Engagement: Measure the engagement of User on weekly basis



```
# weekly user engagement
SELECT
    RIGHT(YEARWEEK(LEFT(created_at, 10), 3),2) AS weeknum,
    -- LEFT(created_at, 10) AS activate_d,
    COUNT(LEFT(created_at, 10)) AS users_count,
    YEARWEEK(LEFT(created_at, 10), 3) AS weekcode
FROM
    users
GROUP BY weekcode;
```



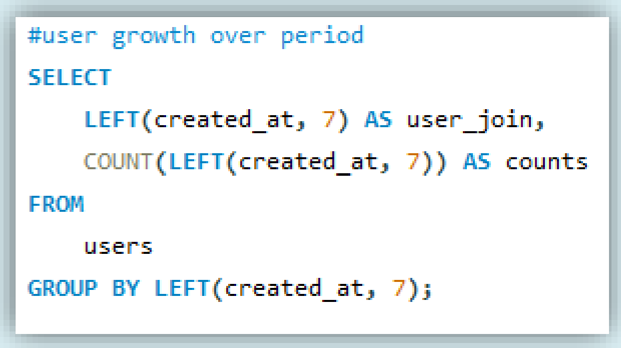
weeknum	users_count	weekcode
01	26	201301
02	29	201302
03	47	201303
04	36	201304
05	30	201305
06	48	201306
07	41	201307
08	39	201308
09	33	201309
10	43	201310
11	33	201311
12	32	201312
13	33	201313
14	40	201314
15	35	201315
16	42	201316
17	48	201317
18	48	201318
19	45	201319
20	55	201320
21	41	201321
22	49	201322
23	51	201323
24	51	201324
25	46	201325
26	57	201326
27	57	201327
28	52	201328
29	71	201329
30	66	201330
31	69	201331
32	66	201332
33	73	201333
34	71	201334
35	79	201335
36	65	201336
37	71	201337
38	84	201338
39	92	201339
40	81	201340
41	88	201341
42	74	201342
43	97	201343
44	92	201344
45	97	201345
46	94	201346
47	82	201347
48	103	201348
49	96	201349
50	117	201350
51	123	201351
52	104	201352



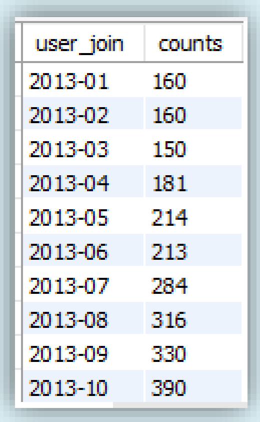
weeknum	users_count	weekcode
01	132	201401
02	122	201402
03	112	201403
04	113	201404
05	130	201405
06	132	201406
07	135	201407
08	127	201408
09	127	201409
10	135	201410
11	152	201411
12	132	201412
13	151	201413
14	161	201414
15	166	201415
16	165	201416
17	176	201417
18	172	201418
19	160	201419
20	186	201420
21	177	201421
22	186	201422
23	197	201423
24	198	201424
25	222	201425
26	210	201426
27	199	201427
28	223	201428
29	215	201429
30	228	201430
31	234	201431
32	189	201432
33	250	201433
34	259	201434
35	266	201435

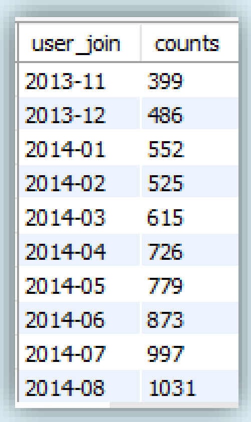
The table shows lower user engagement in the year 2013, however there was a gradual increase week-by-week. In the year 2014 with an overall increase as compared to previous year the user engagement peaked to the highest in the last week of August.

<u>B) User Growth Analysis:</u> To calculate user growth for the product
Here user growth is observed on monthly basis.



```
#user growth over period
SELECT
    LEFT(created_at, 7) AS user_join,
    COUNT(LEFT(created_at, 7)) AS counts
FROM
    users
GROUP BY LEFT(created_at, 7);
```



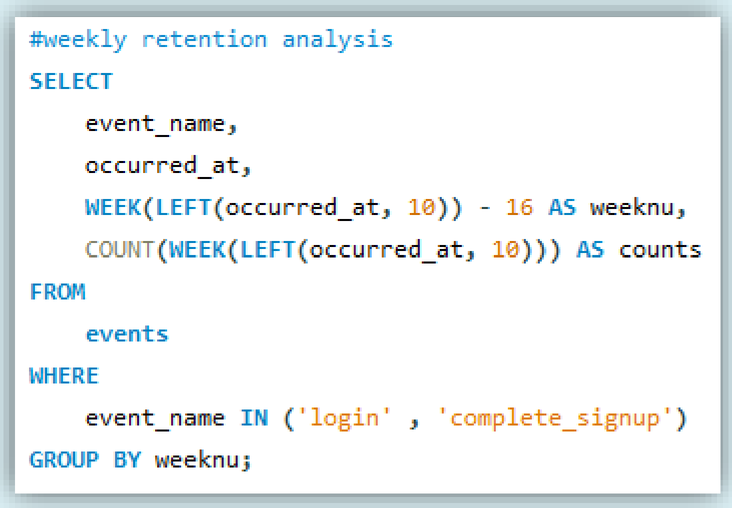


user_join	counts
2013-01	160
2013-02	160
2013-03	150
2013-04	181
2013-05	214
2013-06	213
2013-07	284
2013-08	316
2013-09	330
2013-10	390
2013-11	399
2013-12	486
2014-01	552
2014-02	525
2014-03	615
2014-04	726
2014-05	779
2014-06	873
2014-07	997
2014-08	1031

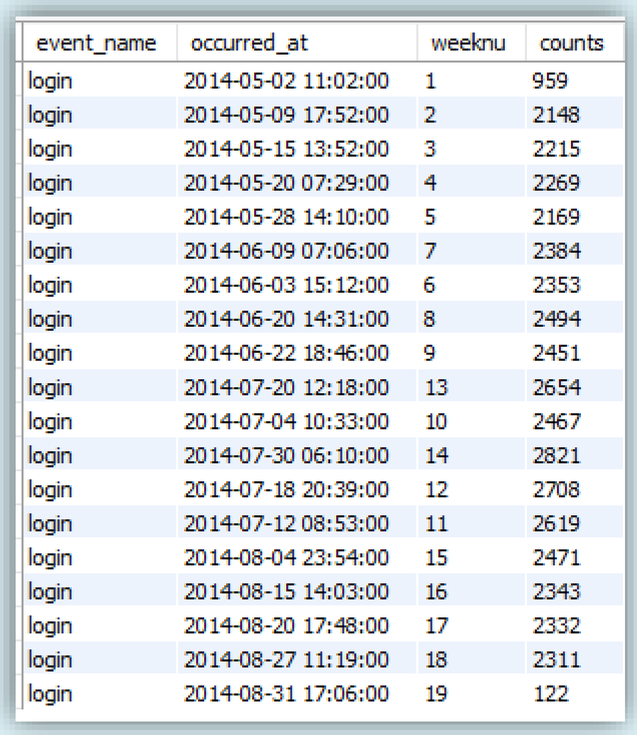
We can observe that users have significantly increased over the period of time.

C) Weekly Retention Analysis: Calculate the weekly retention of users based on their sign-up cohort.
This question ca be viewed in two different ways-

- To check number of total logins per week and observe the trend



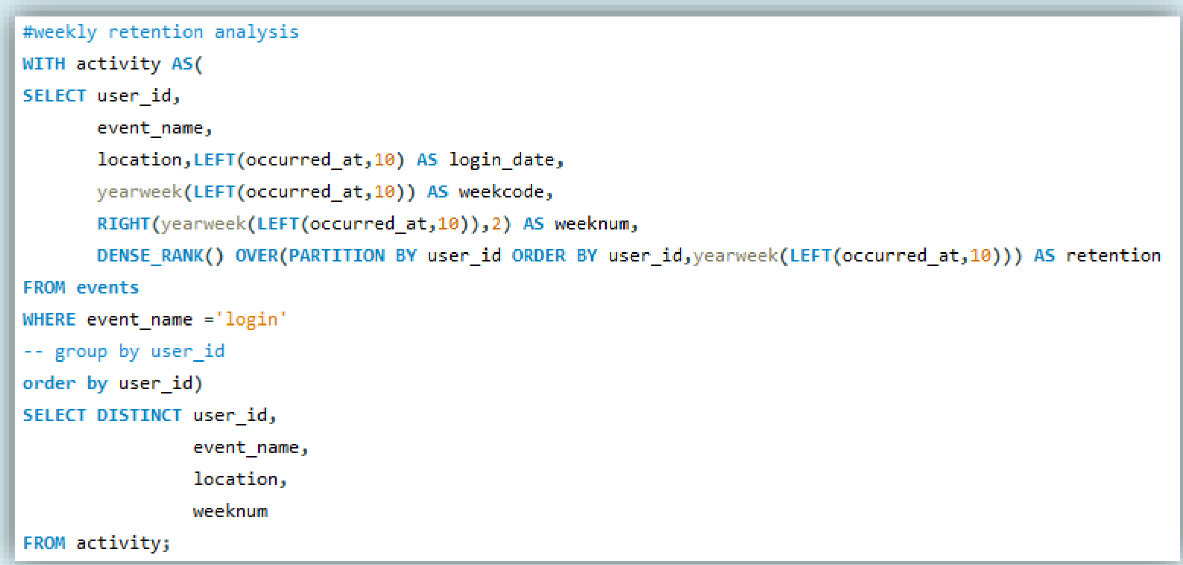
```
#weekly retention analysis
SELECT
    event_name,
    occurred_at,
    WEEK(LEFT(occurred_at, 10)) - 16 AS weeknu,
    COUNT(WEEK(LEFT(occurred_at, 10))) AS counts
FROM
    events
WHERE
    event_name IN ('login' , 'complete_signup')
GROUP BY weeknu;
```



event_name	occurred_at	weeknu	counts
login	2014-05-02 11:02:00	1	959
login	2014-05-09 17:52:00	2	2148
login	2014-05-15 13:52:00	3	2215
login	2014-05-20 07:29:00	4	2269
login	2014-05-28 14:10:00	5	2169
login	2014-06-09 07:06:00	7	2384
login	2014-06-03 15:12:00	6	2353
login	2014-06-20 14:31:00	8	2494
login	2014-06-22 18:46:00	9	2451
login	2014-07-20 12:18:00	13	2654
login	2014-07-04 10:33:00	10	2467
login	2014-07-30 06:10:00	14	2821
login	2014-07-18 20:39:00	12	2708
login	2014-07-12 08:53:00	11	2619
login	2014-08-04 23:54:00	15	2471
login	2014-08-15 14:03:00	16	2343
login	2014-08-20 17:48:00	17	2332
login	2014-08-27 11:19:00	18	2311
login	2014-08-31 17:06:00	19	122

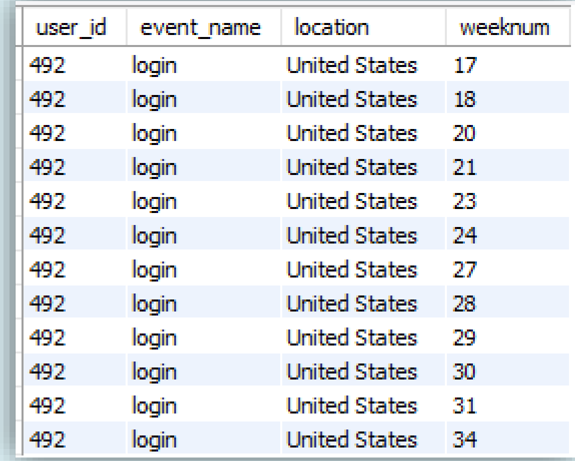
The sign-ups/logins increased significantly (about 125%) by 2nd week itself. The sign-ups showed peak during 14th week since launch. Later trend decreased with lowest in 19th week since launch.

- To check if each user logged-in/signed-up at least once in a week and observe consistency.



```
#weekly retention analysis
WITH activity AS(
SELECT user_id,
       event_name,
       location,LEFT(occurred_at,10) AS login_date,
       yearweek(LEFT(occurred_at,10)) AS weekcode,
       RIGHT(yearweek(LEFT(occurred_at,10)),2) AS weeknum,
       DENSE_RANK() OVER(PARTITION BY user_id ORDER BY user_id,yearweek(LEFT(occurred_at,10))) AS retention
FROM events
WHERE event_name ='login'
-- group by user_id
order by user_id)
SELECT DISTINCT user_id,
               event_name,
               location,
               weeknum
FROM activity;
```

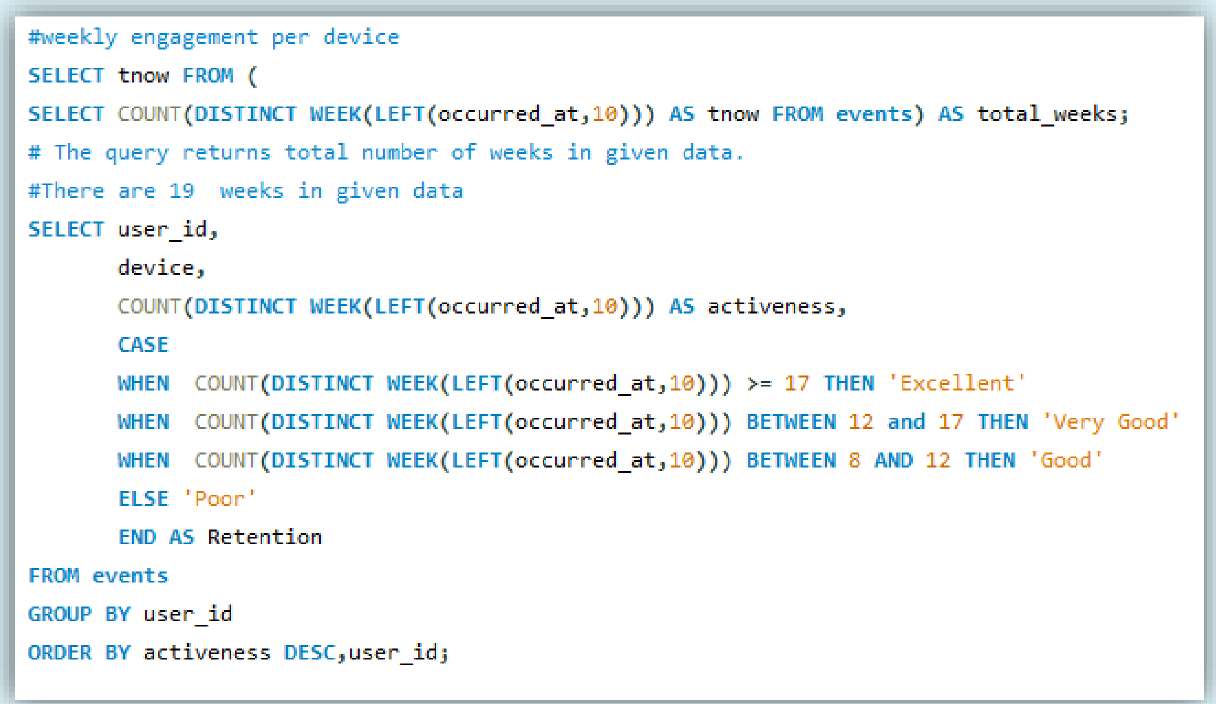
The query gives weekly engagement of every user if logged in atleast once a week. Suppose we need to check if (say) user_id number **492** was active every week or not:



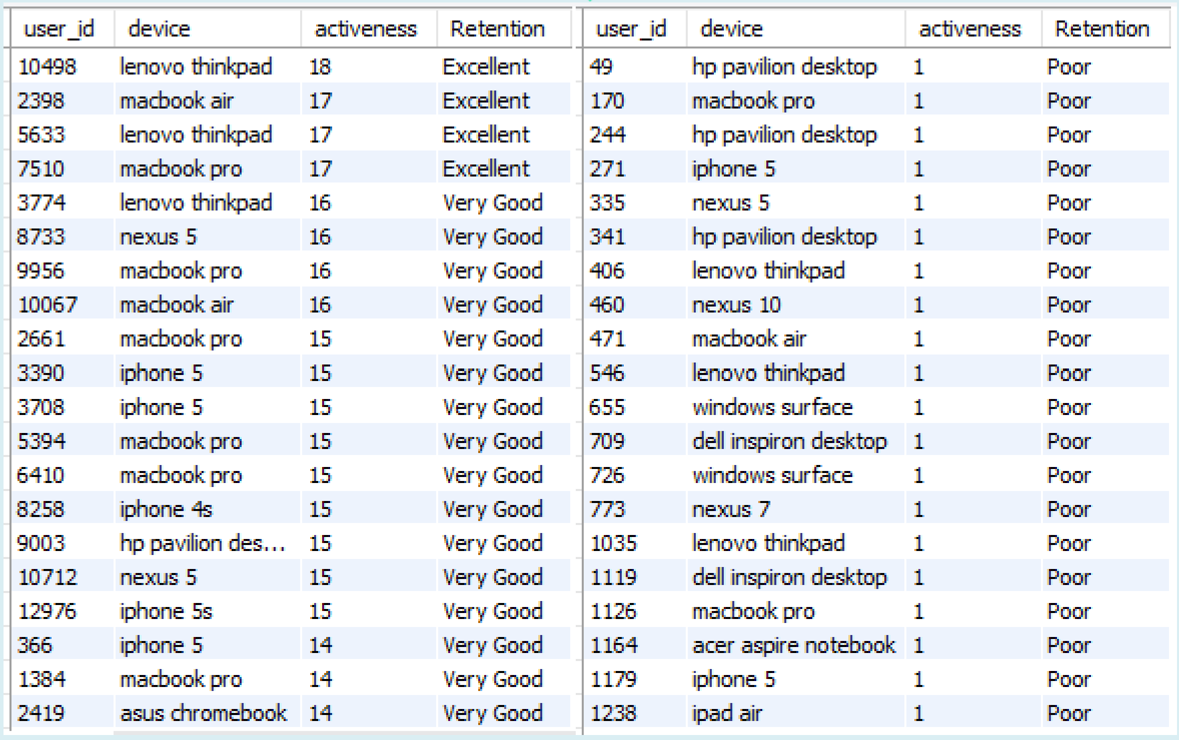
user_id	event_name	location	weeknum
492	login	United States	17
492	login	United States	18
492	login	United States	20
492	login	United States	21
492	login	United States	23
492	login	United States	24
492	login	United States	27
492	login	United States	28
492	login	United States	29
492	login	United States	30
492	login	United States	31
492	login	United States	34

Results shows that user_id 492 signed up on 17th week . Since then he was inactive on 19th,22nd,25th,26th,32nd and 33rd week.

D) Weekly Engagement per device: Measure the activeness of users on a weekly basis per device.
Just like the above problem statement of weekly retention per user, we can check every device’s engagement per week. But here we check total number of weeks every device is used out of total number of weeks given in dataset.



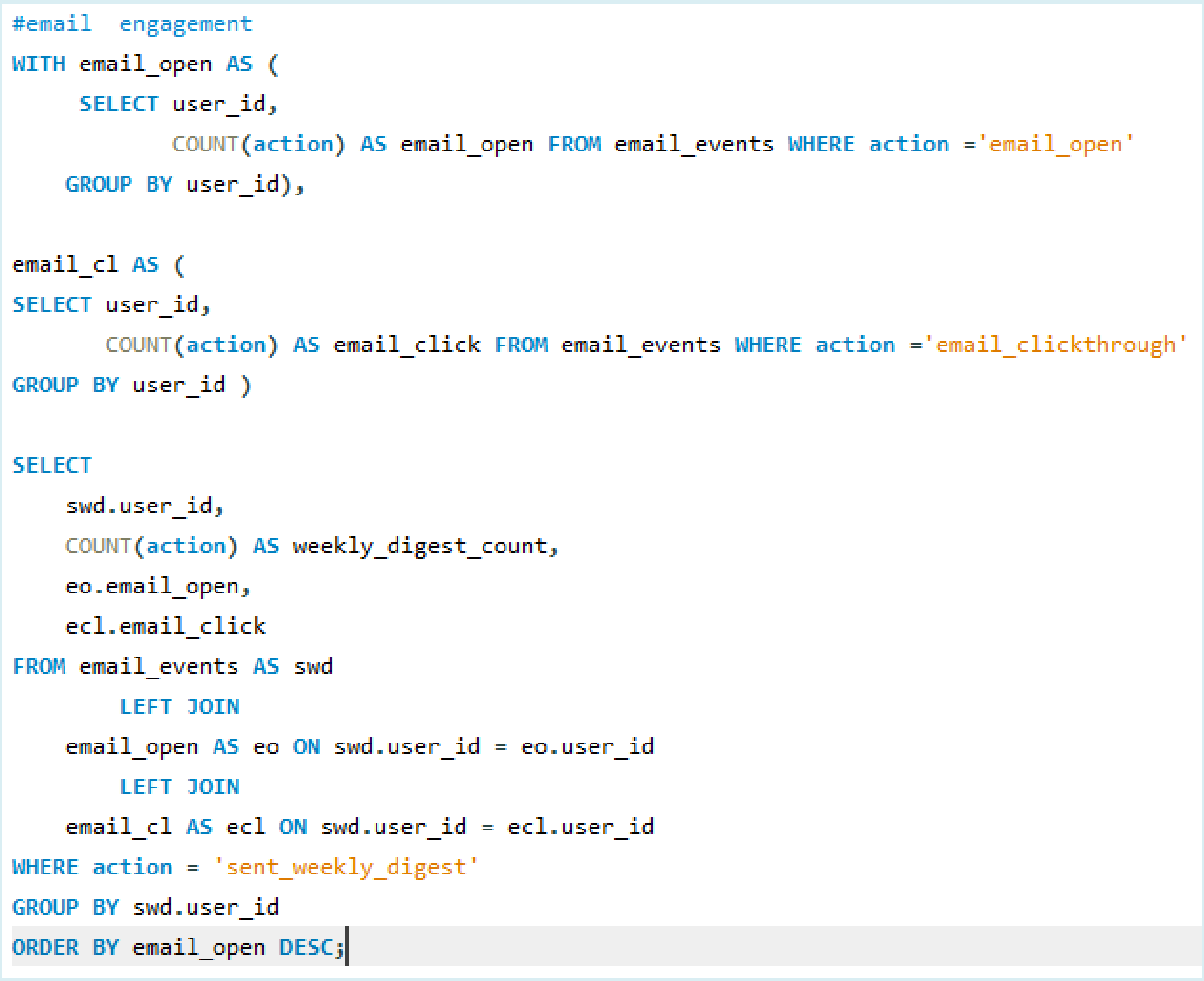
```
#weekly engagement per device
SELECT tnow FROM (
SELECT COUNT(DISTINCT WEEK(LEFT(occurred_at,10))) AS tnow FROM events) AS total_weeks;
# The query returns total number of weeks in given data.
#There are 19  weeks in given data
SELECT user_id,
       device,
       COUNT(DISTINCT WEEK(LEFT(occurred_at,10))) AS activeness,
       CASE
       WHEN  COUNT(DISTINCT WEEK(LEFT(occurred_at,10))) >= 17 THEN 'Excellent'
       WHEN  COUNT(DISTINCT WEEK(LEFT(occurred_at,10))) BETWEEN 12 and 17 THEN 'Very Good'
       WHEN  COUNT(DISTINCT WEEK(LEFT(occurred_at,10))) BETWEEN 8 AND 12 THEN 'Good'
       ELSE 'Poor'
       END AS Retention
FROM events
GROUP BY user_id
ORDER BY activeness DESC,user_id;
```



user_id	device	activeness	Retention
10498	lenovo thinkpad	18	Excellent
2398	macbook air	17	Excellent
5633	lenovo thinkpad	17	Excellent
7510	macbook pro	17	Excellent
3774	lenovo thinkpad	16	Very Good
8733	nexus 5	16	Very Good
9956	macbook pro	16	Very Good
10067	macbook air	16	Very Good
2661	macbook pro	15	Very Good
3390	iphone 5	15	Very Good
3708	iphone 5	15	Very Good
5394	macbook pro	15	Very Good
6410	macbook pro	15	Very Good
8258	iphone 4s	15	Very Good
9003	hp pavilion des...	15	Very Good
10712	nexus 5	15	Very Good
12976	iphone 5s	15	Very Good
366	iphone 5	14	Very Good
1384	macbook pro	14	Very Good
2419	asus chromebook	14	Very Good
49	hp pavilion desktop	1	Poor
170	macbook pro	1	Poor
244	hp pavilion desktop	1	Poor
271	iphone 5	1	Poor
335	nexus 5	1	Poor
341	hp pavilion desktop	1	Poor
406	lenovo thinkpad	1	Poor
460	nexus 10	1	Poor
471	macbook air	1	Poor
546	lenovo thinkpad	1	Poor
655	windows surface	1	Poor
709	dell inspiron desktop	1	Poor
726	windows surface	1	Poor
773	nexus 7	1	Poor
1035	lenovo thinkpad	1	Poor
1119	dell inspiron desktop	1	Poor
1126	macbook pro	1	Poor
1164	acer aspire notebook	1	Poor
1179	iphone 5	1	Poor
1238	ipad air	1	Poor

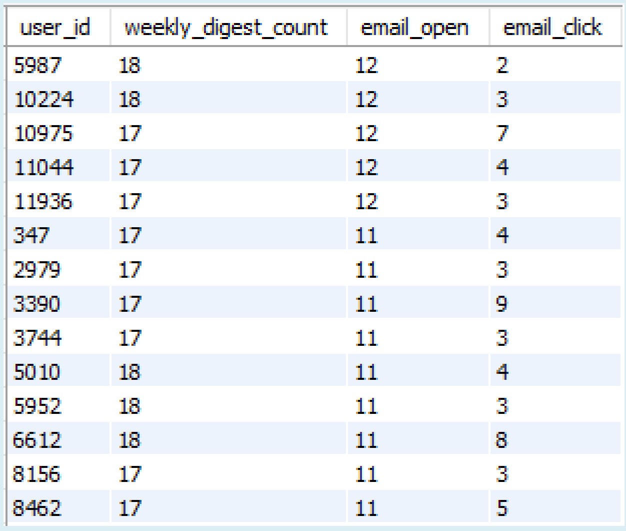
The results shows top 20 and bottom 20 devices engaged.

E) Email Engagement Analysis: Analyse how users are engaging with the email service.



```
#email  engagement
WITH email_open AS (
     SELECT user_id,
            COUNT(action) AS email_open FROM email_events WHERE action ='email_open'
    GROUP BY user_id),

email_cl AS (
SELECT user_id,
       COUNT(action) AS email_click FROM email_events WHERE action ='email_clickthrough'
GROUP BY user_id )

SELECT
    swd.user_id,
    COUNT(action) AS weekly_digest_count,
    eo.email_open,
    ecl.email_click
FROM email_events AS swd
        LEFT JOIN
    email_open AS eo ON swd.user_id = eo.user_id
        LEFT JOIN
    email_cl AS ecl ON swd.user_id = ecl.user_id
WHERE action = 'sent_weekly_digest'
GROUP BY swd.user_id
ORDER BY email_open DESC;
```

The query returns following table, which shows how users interacted with emails:

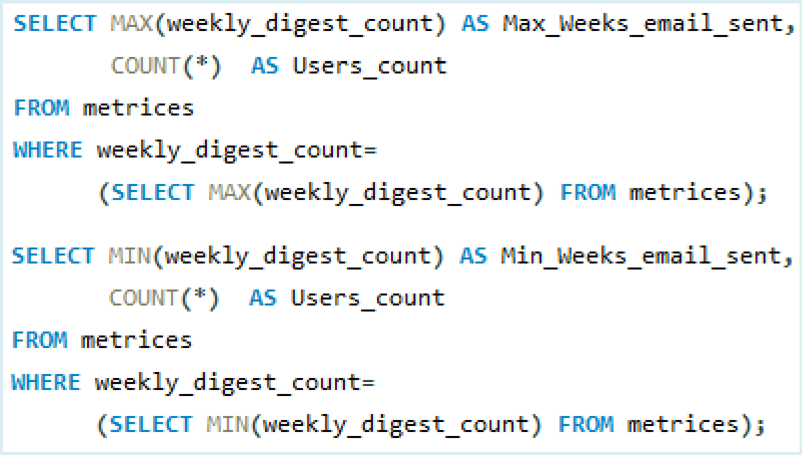


user_id	weekly_digest_count	email_open	email_click
5987	18	12	2
10224	18	12	3
10975	17	12	7
11044	17	12	4
11936	17	12	3
347	17	11	4
2979	17	11	3
3390	17	11	9
3744	17	11	3
5010	18	11	4
5952	18	11	3
6612	18	11	8
8156	17	11	3
8462	17	11	5

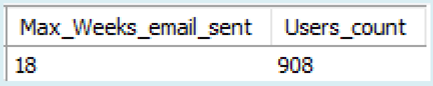
The data shows that mails are sent for 17 or 18 weeks every user. In response to those emails people either opened or ignored the mails followed by clicked the link in email.

Let's check Insights in detail:

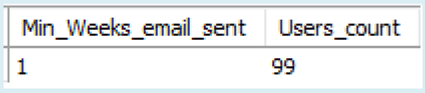
Max/Min. Emails sent to Users and Respective Users Count



```
SELECT MAX(weekly_digest_count) AS Max_Weeks_email_sent,
       COUNT(*)  AS Users_count
FROM metrices
WHERE weekly_digest_count=
      (SELECT MAX(weekly_digest_count) FROM metrices);

SELECT MIN(weekly_digest_count) AS Min_Weeks_email_sent,
       COUNT(*)  AS Users_count
FROM metrices
WHERE weekly_digest_count=
      (SELECT MIN(weekly_digest_count) FROM metrices);
```

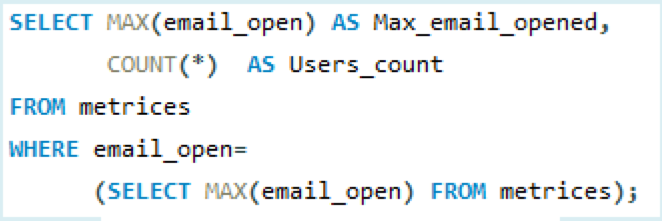


Max_Weeks_email_sent	Users_count
18	908

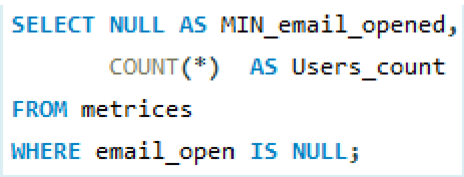


Min_Weeks_email_sent	Users_count
1	99

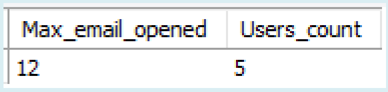
Max/Min Emails Opened and Respective Users Count



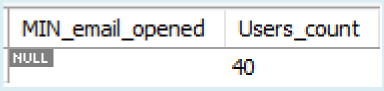
```
SELECT MAX(email_open) AS Max_email_opened,
       COUNT(*)  AS Users_count
FROM metrices
WHERE email_open=
      (SELECT MAX(email_open) FROM metrices);
```



```
SELECT NULL AS MIN_email_opened,
       COUNT(*)  AS Users_count
FROM metrices
WHERE email_open IS NULL;
```

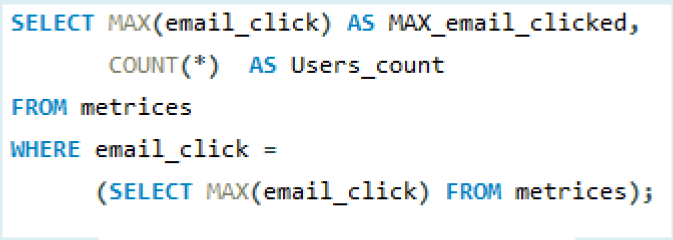


Max_email_opened	Users_count
12	5

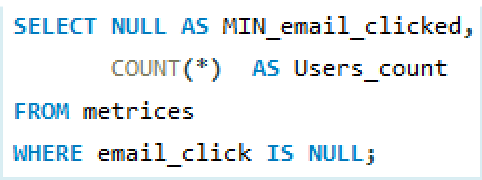


MIN_email_opened	Users_count
NULL	40

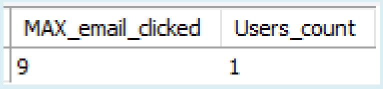
Max/Min Email Links Clicked and Respective Users Count



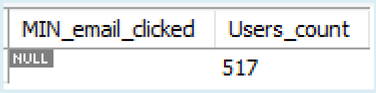
```
SELECT MAX(email_click) AS MAX_email_clicked,
       COUNT(*)  AS Users_count
FROM metrices
WHERE email_click =
      (SELECT MAX(email_click) FROM metrices);
```



```
SELECT NULL AS MIN_email_clicked,
       COUNT(*)  AS Users_count
FROM metrices
WHERE email_click IS NULL;
```



MAX_email_clicked	Users_count
9	1



MIN_email_clicked	Users_count
NULL	517

THANK YOU!!

(Looking Forward for Valuable Suggestions and Feedbacks)