Operation Analytics and Investigating metrics

Project Description:

The objective of this project is to analyse the areas of improvement in end to end operations of a company and provide some insights on investigating metric spikes such as dip in daily user engagement or drop sales based on data collected from the various teams, such as operation, support, and marketing.

Approach:

We perform analysis in Two steps / cases:

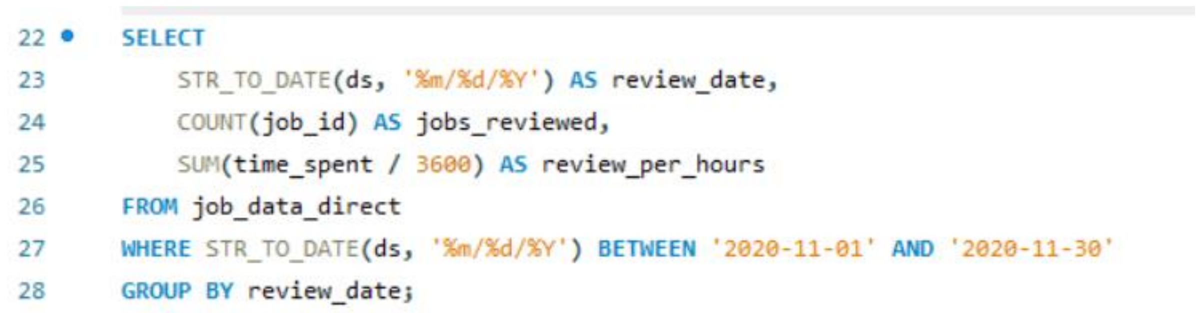
1. Job Data Analysis
2. Investigating Metric Spike

Case Study 1:

In **Job data analysis**:

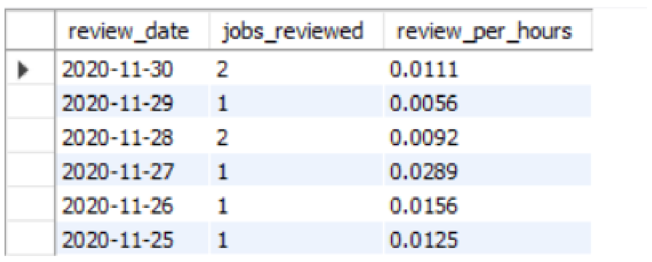
1. **Job reviewed over time**: Calculate the number of jobs reviewed per hour for each day in November 2020.

Query:



```
SELECT
    STR_TO_DATE(ds, '%m/%d/%Y') AS review_date,
    COUNT(job_id) AS jobs_reviewed,
    SUM(time_spent / 3600) AS review_per_hours
FROM job_data_direct
WHERE STR_TO_DATE(ds, '%m/%d/%Y') BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY review_date;
```

Output:

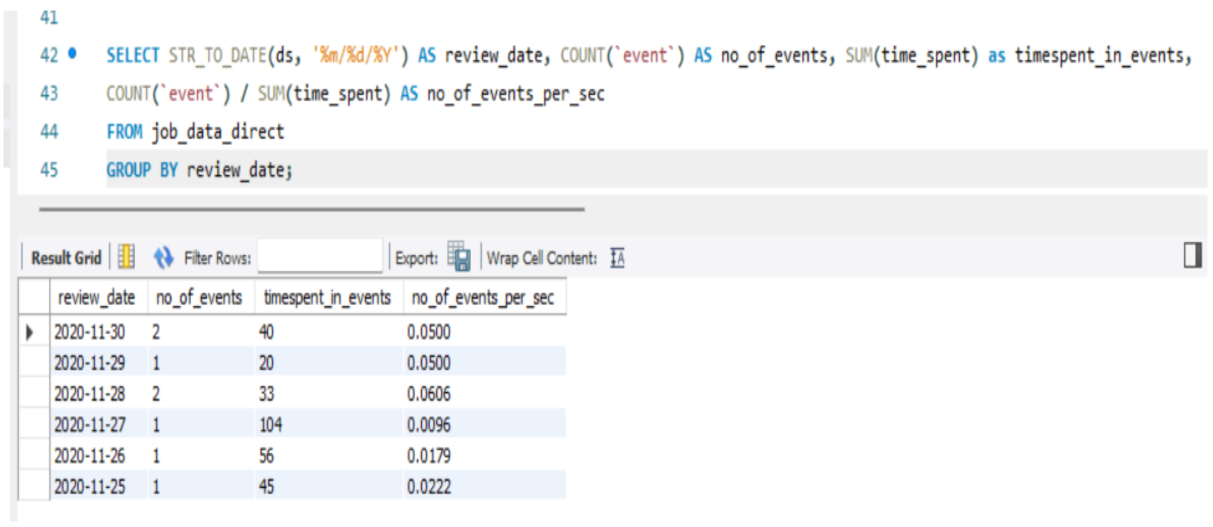


review_date	jobs_reviewed	review_per_hours
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

Interpretation: maximum of 2 jobs gets reviewed in month of November.

2. **Throughput Analysis**: Calculate the average number of events that occur per second over a period of seven days.

Query:



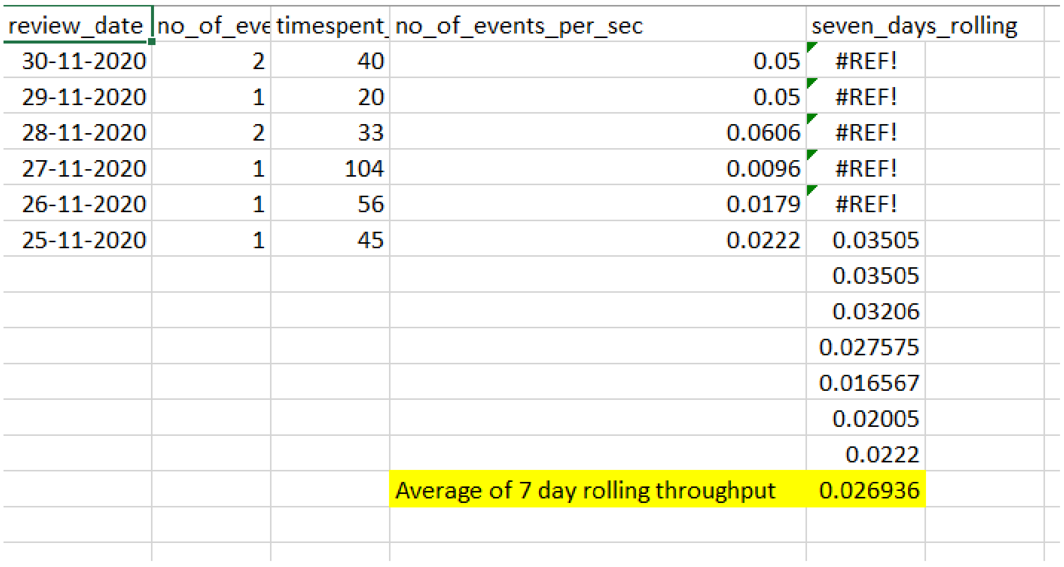
```
SELECT STR_TO_DATE(ds, '%m/%d/%Y') AS review_date, COUNT(`event`) AS no_of_events, SUM(time_spent) as timespent_in_events,
COUNT(`event`) / SUM(time_spent) AS no_of_events_per_sec
FROM job_data_direct
GROUP BY review_date;
```

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

review_date	no_of_events	timespent_in_events	no_of_events_per_sec
2020-11-30	2	40	0.0500
2020-11-29	1	20	0.0500
2020-11-28	2	33	0.0606
2020-11-27	1	104	0.0096
2020-11-26	1	56	0.0179
2020-11-25	1	45	0.0222

Interpretation: maximum of 2 events are occurred in a single day with a average of 0.05 events per seconds.

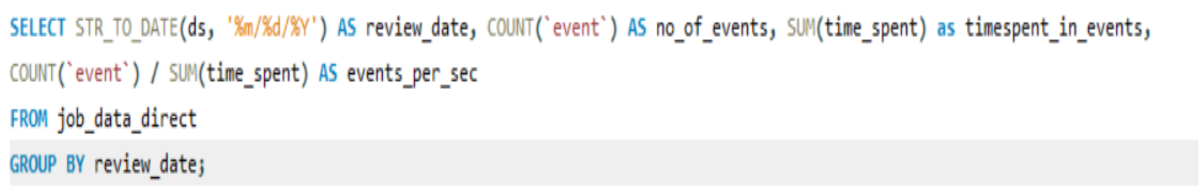
To calculate the 7-day rolling average of throughput, use the following functions in Microsoft Excel: AVG and OFFSET



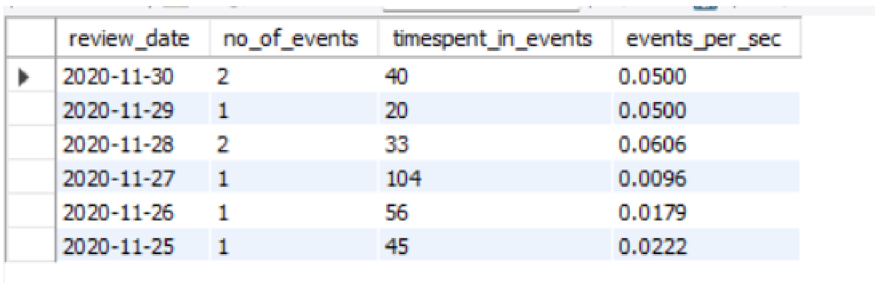
review_date	no_of_eve	timespent_	no_of_events_per_sec	seven_days_rolling
30-11-2020	2	40	0.05	#REF!
29-11-2020	1	20	0.05	#REF!
28-11-2020	2	33	0.0606	#REF!
27-11-2020	1	104	0.0096	#REF!
26-11-2020	1	56	0.0179	#REF!
25-11-2020	1	45	0.0222	0.03505
				0.03505
				0.03206
				0.027575
				0.016567
				0.02005
				0.0222
			Average of 7 day rolling throughput	0.026936

OR

I tried to solve the same problem from SQL subquery also

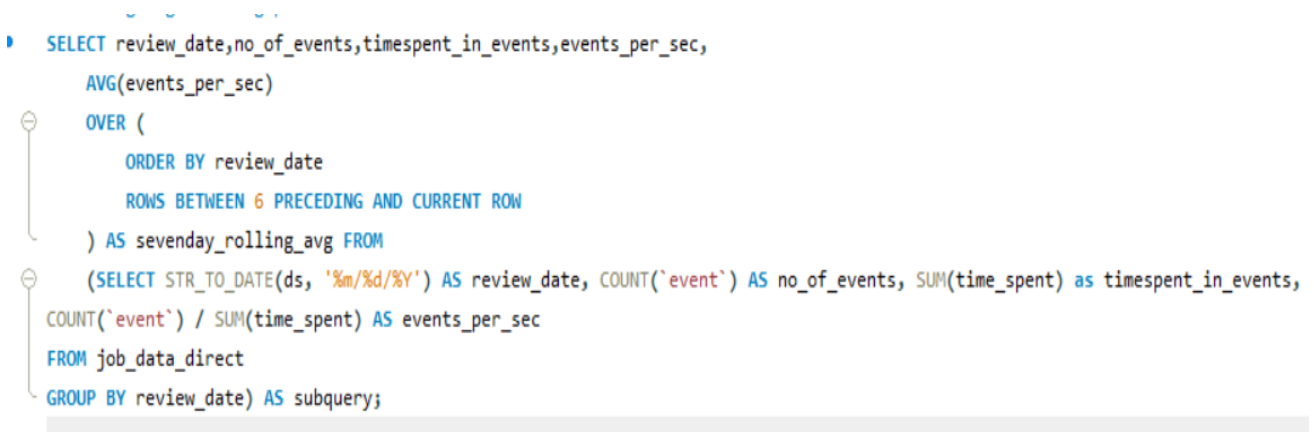


```
SELECT STR_TO_DATE(ds, '%m/%d/%Y') AS review_date, COUNT(`event`) AS no_of_events, SUM(time_spent) as timespent_in_events,
COUNT(`event`) / SUM(time_spent) AS events_per_sec
FROM job_data_direct
GROUP BY review_date;
```



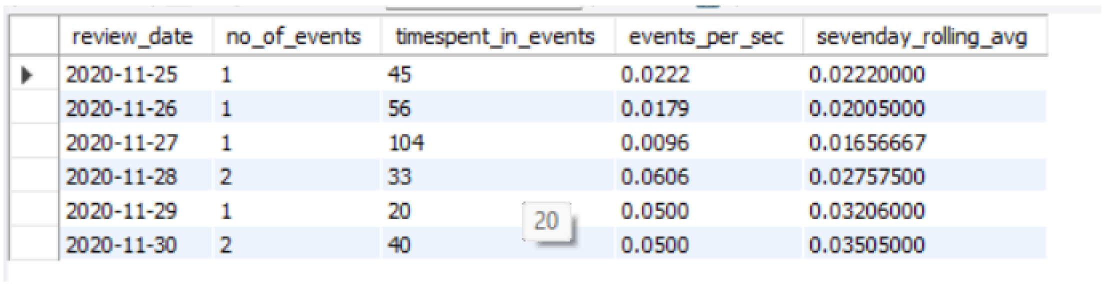
	review_date	no_of_events	timespent_in_events	events_per_sec
▶	2020-11-30	2	40	0.0500
	2020-11-29	1	20	0.0500
	2020-11-28	2	33	0.0606
	2020-11-27	1	104	0.0096
	2020-11-26	1	56	0.0179
	2020-11-25	1	45	0.0222

1 st create this table, and calculate 7 day rolling average throughput on this.



```sql
SELECT review_date,no_of_events,timespent_in_events,events_per_sec,
    AVG(events_per_sec)
    OVER (
        ORDER BY review_date
        ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
    ) AS sevenday_rolling_avg FROM
    (SELECT STR_TO_DATE(ds, '%m/%d/%Y') AS review_date, COUNT(`event`) AS no_of_events, SUM(time_spent) as timespent_in_events,
COUNT(`event`) / SUM(time_spent) AS events_per_sec
FROM job_data_direct
GROUP BY review_date) AS subquery;
```

Output:



	review_date	no_of_events	timespent_in_events	events_per_sec	sevenday_rolling_avg
▶	2020-11-25	1	45	0.0222	0.02220000
	2020-11-26	1	56	0.0179	0.02005000
	2020-11-27	1	104	0.0096	0.01656667
	2020-11-28	2	33	0.0606	0.02757500
	2020-11-29	1	20	0.0500	0.03206000
	2020-11-30	2	40	0.0500	0.03505000

Daily metric or the 7-day rolling average for throughput, both are the effective way analyze data and patterns, I would consider 7-day rolling average for long term decisions, but in this case dataset is small so I would prefer daily metric:

to get most up to date information,

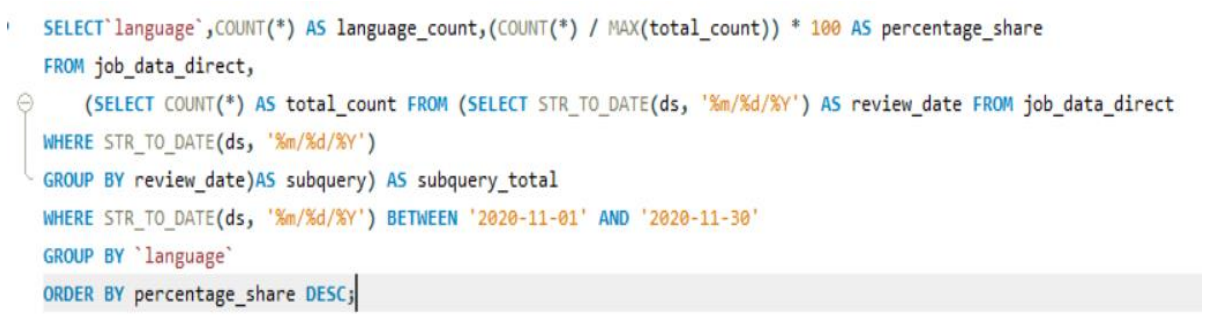
to understand the short-term fluctuations,

to get sudden spikes or dips.

Interpretations: Seven day rolling average of throughput is 0.027, means This means that, on average, there were 0.027 events or throughput occurrences per sec over the past seven days.

3. **Language Share Analysis:** Calculate the percentage share of each language in the last 30 days.

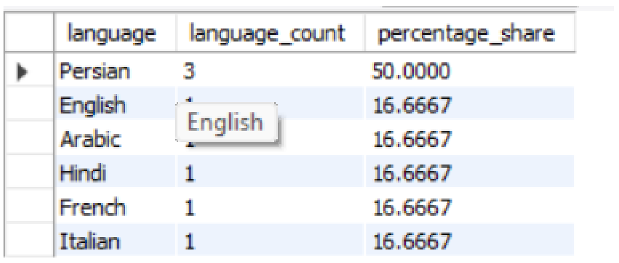
Query:



```
SELECT`language`,COUNT(*) AS language_count,(COUNT(*) / MAX(total_count)) * 100 AS percentage_share
FROM job_data_direct,
    (SELECT COUNT(*) AS total_count FROM (SELECT STR_TO_DATE(ds, '%m/%d/%Y') AS review_date FROM job_data_direct
WHERE STR_TO_DATE(ds, '%m/%d/%Y')
GROUP BY review_date)AS subquery) AS subquery_total
WHERE STR_TO_DATE(ds, '%m/%d/%Y') BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY `language`
ORDER BY percentage_share DESC;
```

Calculate the language count from job_data table, and for total count create another table in subquery based on review date for 30 days.

Output:

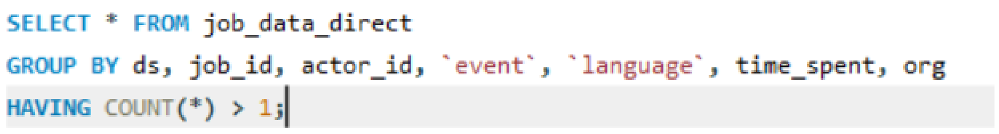


language	language_count	percentage_share
Persian	3	50.0000
English	1	16.6667
Arabic	1	16.6667
Hindi	1	16.6667
French	1	16.6667
Italian	1	16.6667

Interpretations: Persian language has captured the 50 % share.

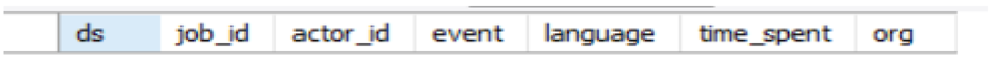
4. **Duplicate Rows Detection:** Identify duplicate rows in the data.

Query:



```
SELECT * FROM job_data_direct
GROUP BY ds, job_id, actor_id, `event`, `language`, time_spent, org
HAVING COUNT(*) > 1;
```

Output:



ds	job_id	actor_id	event	language	time_spent	org

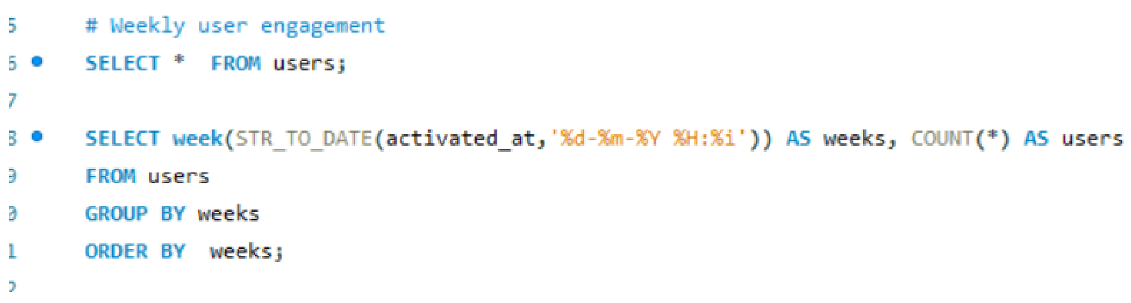
Interpretation: No Duplicate rows has identified.

Case Study 2:

In **Investigating Metric Spike:**

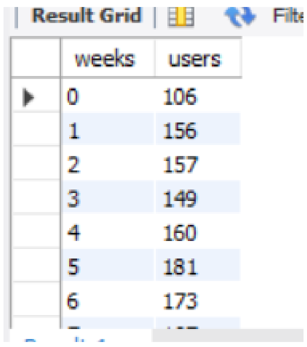
1. **Weekly User Engagement:** Measure the activeness of users on a weekly basis. i.e no. of users getting active on a weekly basis.

Query:



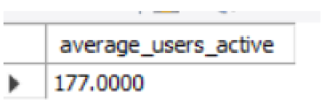
```
# Weekly user engagement
SELECT *  FROM users;

SELECT week(STR_TO_DATE(activated_at,'%d-%m-%Y %H:%i')) AS weeks, COUNT(*) AS users
FROM users
GROUP BY weeks
ORDER BY  weeks;
```

Output:

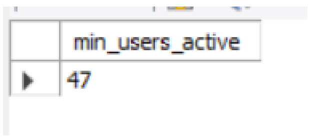

Result Grid

weeks	users
0	106
1	156
2	157
3	149
4	160
5	181
6	173

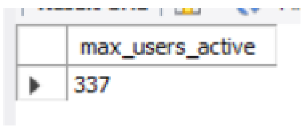
Interpretations:



average_users_active
177.0000



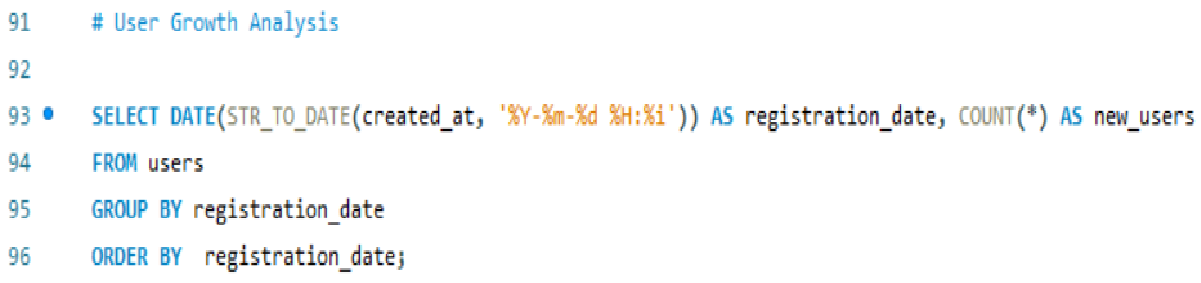
min_users_active
47



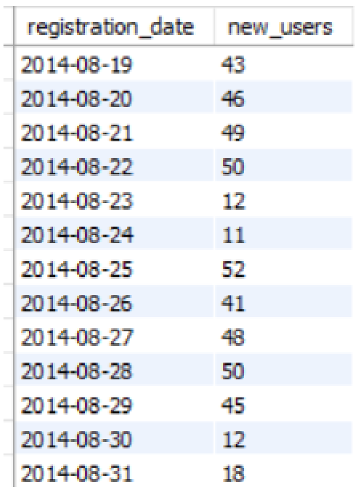
max_users_active
337

On an average 177 users gets activated per week, with max and min of 337 users and 47 users respectively .

2. **User Growth Analysis:** Analyze the growth of users over time for a product. i.e every day how many users are getting registered.



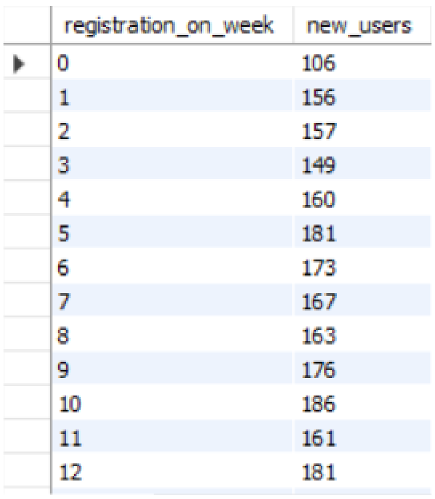
```
# User Growth Analysis

SELECT DATE(STR_TO_DATE(created_at, '%Y-%m-%d %H:%i')) AS registration_date, COUNT(*) AS new_users
FROM users
GROUP BY registration_date
ORDER BY  registration_date;
```



registration_date	new_users
2014-08-19	43
2014-08-20	46
2014-08-21	49
2014-08-22	50
2014-08-23	12
2014-08-24	11
2014-08-25	52
2014-08-26	41
2014-08-27	48
2014-08-28	50
2014-08-29	45
2014-08-30	12
2014-08-31	18

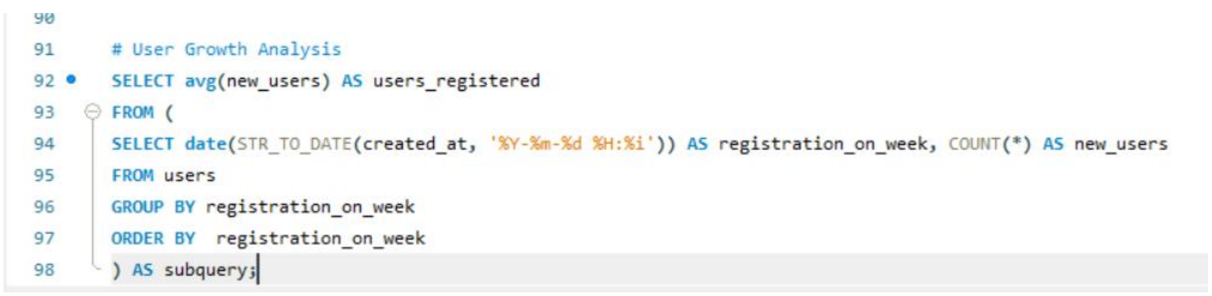
Users Growth on weekly basis:

```
SELECT week(STR_TO_DATE(created_at, '%Y-%m-%d %H:%i')) AS registration_on_week, COUNT(*) AS new_users
FROM users
GROUP BY registration_on_week
ORDER BY  registration_on_week;
```

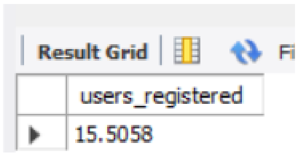


registration_on_week	new_users
0	106
1	156
2	157
3	149
4	160
5	181
6	173
7	167
8	163
9	176
10	186
11	161
12	181

Calculating average:



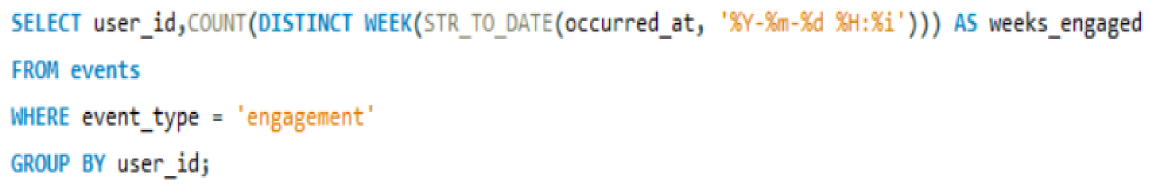
```
# User Growth Analysis
SELECT avg(new_users) AS users_registered
FROM (
SELECT date(STR_TO_DATE(created_at, '%Y-%m-%d %H:%i')) AS registration_on_week, COUNT(*) AS new_users
FROM users
GROUP BY registration_on_week
ORDER BY  registration_on_week
) AS subquery;
```



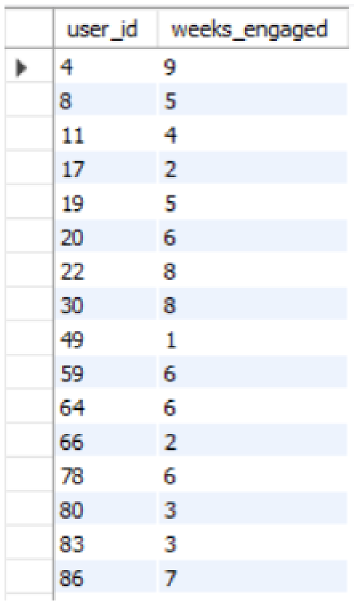
users_registered
15.5058

Interpretations: On an average of 15 users gets registered every day and 177 users are getting registered every week.

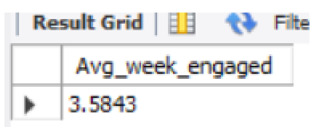
3. **Weekly Retention Analysis:** Analyze the retention of users on a weekly basis after signing up for a product.



```
SELECT user_id,COUNT(DISTINCT WEEK(STR_TO_DATE(occurred_at, '%Y-%m-%d %H:%i'))) AS weeks_engaged
FROM events
WHERE event_type = 'engagement'
GROUP BY user_id;
```

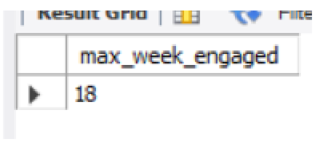


user_id	weeks_engaged
4	9
8	5
11	4
17	2
19	5
20	6
22	8
30	8
49	1
59	6
64	6
66	2
78	6
80	3
83	3
86	7



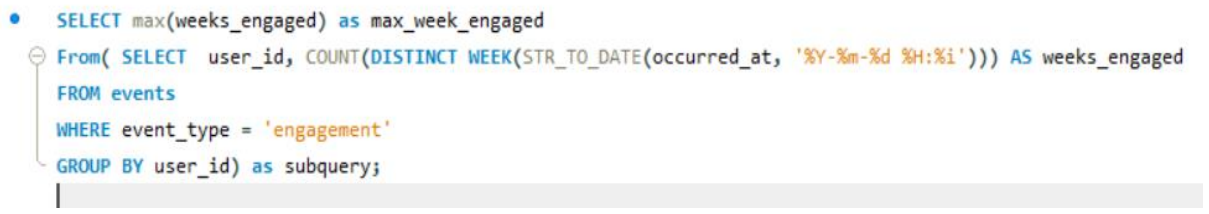
Result Grid

Avg_week_engaged
3.5843



Result Grid

max_week_engaged
18

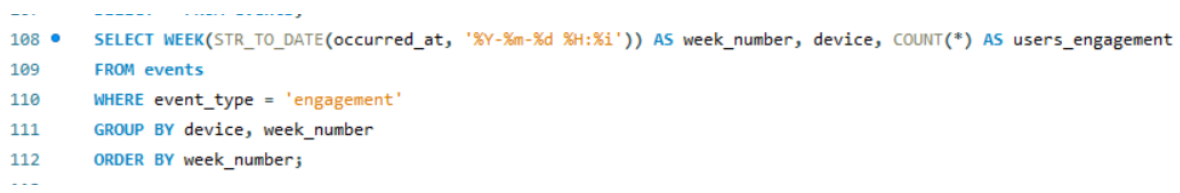


```
SELECT max(weeks_engaged) as max_week_engaged
From( SELECT  user_id, COUNT(DISTINCT WEEK(STR_TO_DATE(occurred_at, '%Y-%m-%d %H:%i'))) AS weeks_engaged
FROM events
WHERE event_type = 'engagement'
GROUP BY user_id) as subquery;
```

Interpretations: A user are getting engaged with a product with an average of 3.6 weeks and max of 18 weeks

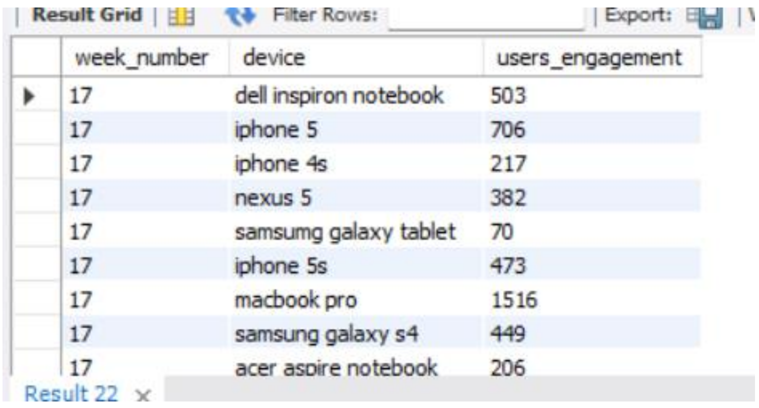
4. **Weekly Engagement Per Device:** Measure the activeness of users on a weekly basis per device.

Query:



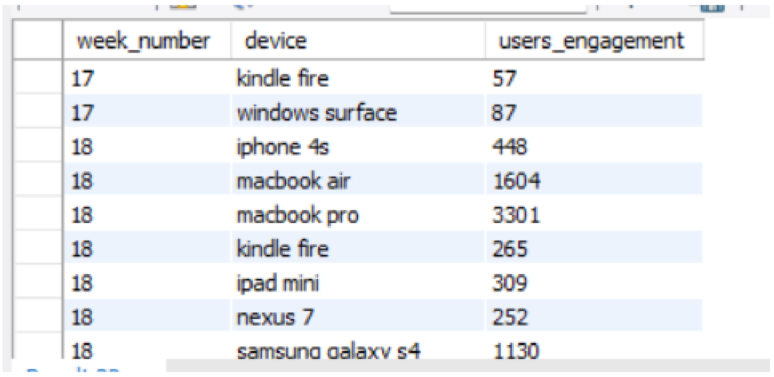
```
SELECT WEEK(STR_TO_DATE(occurred_at, '%Y-%m-%d %H:%i')) AS week_number, device, COUNT(*) AS users_engagement
FROM events
WHERE event_type = 'engagement'
GROUP BY device, week_number
ORDER BY week_number;
```

Output:


Result Grid | Filter Rows: | Export:

week_number	device	users_engagement
17	dell inspiron notebook	503
17	iphone 5	706
17	iphone 4s	217
17	nexus 5	382
17	samsumg galaxy tablet	70
17	iphone 5s	473
17	macbook pro	1516
17	samsung galaxy s4	449
17	acer aspire notebook	206

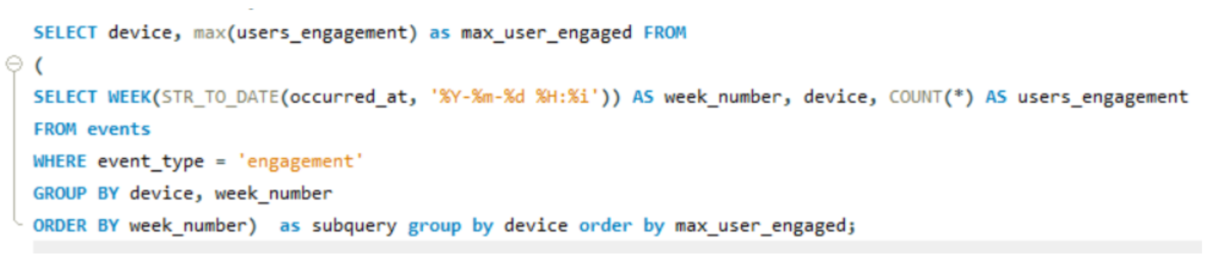
Result 22 ×

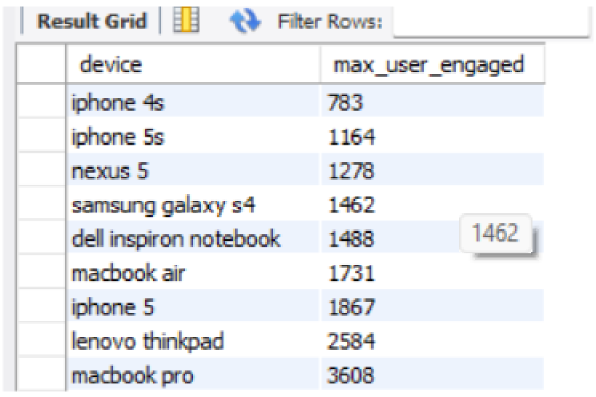


week_number	device	users_engagement
17	kindle fire	57
17	windows surface	87
18	iphone 4s	448
18	macbook air	1604
18	macbook pro	3301
18	kindle fire	265
18	ipad mini	309
18	nexus 7	252
18	samsung galaxy s4	1130

Interpretations: We can see number of users using different devices to use product in a week. Like in 1st week (17th week of year) 503 users use dell inspiron notebook to get engaged.

To know which device is mostly engaged:


```
SELECT device, max(users_engagement) as max_user_engaged FROM
(
SELECT WEEK(STR_TO_DATE(occurred_at, '%Y-%m-%d %H:%i')) AS week_number, device, COUNT(*) AS users_engagement
FROM events
WHERE event_type = 'engagement'
GROUP BY device, week_number
ORDER BY week_number)  as subquery group by device order by max_user_engaged;
```

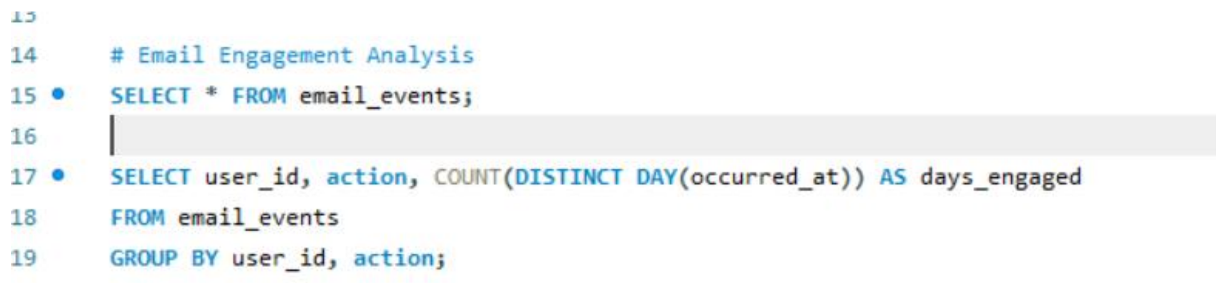

Result Grid | Filter Rows:

device	max_user_engaged
iphone 4s	783
iphone 5s	1164
nexus 5	1278
samsung galaxy s4	1462
dell inspiron notebook	1488
macbook air	1731
iphone 5	1867
lenovo thinkpad	2584
macbook pro	3608

Interpretation: Maximum of 3608 users are getting engaged by macbook pro users in a week.

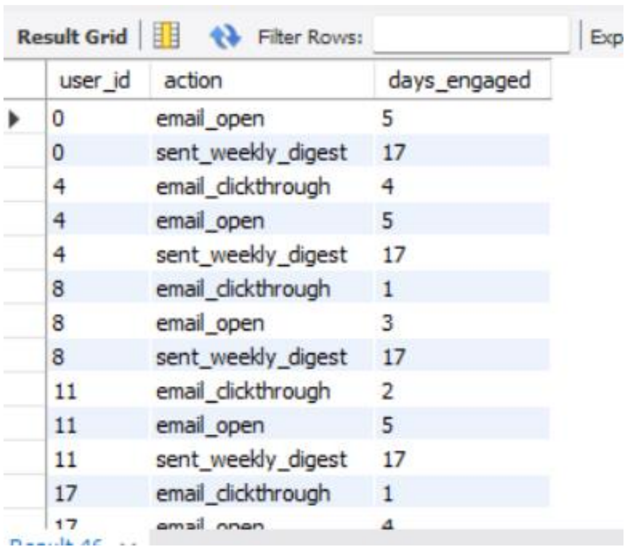
5. **Email Engagement Analysis:** Analyze how users are engaging with the email service. **i**.e.

Query:



```
# Email Engagement Analysis
SELECT * FROM email_events;

SELECT user_id, action, COUNT(DISTINCT DAY(occurred_at)) AS days_engaged
FROM email_events
GROUP BY user_id, action;
```

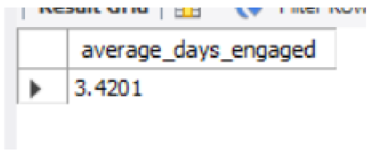
Output:



user_id	action	days_engaged
0	email_open	5
0	sent_weekly_digest	17
4	email_clickthrough	4
4	email_open	5
4	sent_weekly_digest	17
8	email_clickthrough	1
8	email_open	3
8	sent_weekly_digest	17
11	email_clickthrough	2
11	email_open	5
11	sent_weekly_digest	17
17	email_clickthrough	1
17	email_open	4

Interpretations: There are different action performed by users to get engaged with email services like user 0 opens 5 times his email and 17 times he was delivered a digest email.

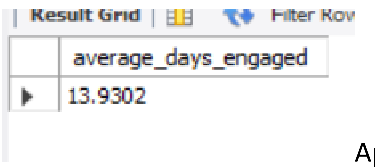
For action: email open



average_days_engaged
3.4201

Approx 3 times user opening a email.

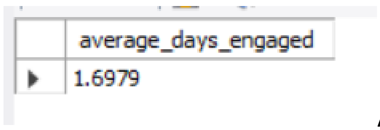
For action: sent_weekly_digest



average_days_engaged
13.9302

Approx 14 times users was delivered a digest mail.

For action: email_clickthrough



	average_days_engaged
▶	1.6979

Approx 2 time users getting a email with url link.

Tech_stack Used:

For this project we used **SQL** language for data analysis.
We use **MYSQL** as our RDBMS which used to store and manipulate data.

Insights: Interpretations are already mentioned after query.

Result:

We successfully perform the job data analysis by getting number of jobs reviewed, language share in job_data, and throughput analysis.

We successfully calculated investigating metric spike like weekly user engagement, user growth analysis, email engagement services.

Drive Link:

https://drive.google.com/drive/folders/1Mv03_MEOstUL5vkHBkocYcJr18ht8HxE?usp=drive_link