Operation Analytics and Investigating Metric Spike

trainity

Project #3

By:

ANURAG CHANGMAI
Data Analytics Trainee
27/06/2023

Project Description

Operation analytics is crucial for any organization that is aiming for success in its field. It not only helps the organization analyze its end-to-end operations but also aids in identifying the areas where it could further improve upon.

It also has the added benefit of bringing together different teams or units of the organization when trying to understand the operations. This creates a synergy in their operations, and consequently that of the organization itself.

In this project, I perform operation analytics for a hypothetical organization that intends to understand its user base. I also analyze the operations its users perform and understand engagement metrics over time.

Approach

This project starts by first obtaining a sample dataset of internal jobs performed for the benefit of the users, a dataset of the users themselves, along with their engagements on the organization's platform.

This data is then cleaned and stored in a database called *op_analytics*, using multiple tables such as *job_data*, *users*, *events*, and *email_events*.

Once this is done, SQL queries are executed on these tables to obtain specific data as per requirements.

Tech-stack used

In the execution of this project, the following software was used:

1. Oracle MySQL Workbench 8.0 v8.0.33

→ It was used to run SQL queries to create the database, perform operations on the tables within the database, and obtain desired outputs.

2. Microsoft Excel 2019 v2305

→ It was used to understand the data which was in CSV format, clean it, and prepare it to be loaded into SQL database.

Insights

In the following slides, insights on these points are explained:

1. From a user-benefit perspective, insights from internal data on translation jobs are obtained. This will be regarded as Case 1 in the upcoming slides.
2. From a user perspective, insights on user engagement, user growth, weekly user retention, weekly device-wise user engagement, and user email engagement are obtained. This will be regarded as Case 2 in the upcoming slides.

Case 1

This case makes use of data from the table shown below, which appears to deal with translation features provided on the organization's website.

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

Case 1: Number of jobs reviewed over time

SQL QUERY

```
USE op_analytics;
SELECT COUNT(job_id)/(30*24) AS jobs_reviewed_per_day,
       COUNT(DISTINCT job_id)/(30*24) AS distinct_jobs_reviewed_per_day
FROM job_data;
```

RESULTS

jobs_reviewed_per_day	distinct_jobs_reviewed_per_day
0.0111	0.0083

This shows that on average, **0.0111** jobs are reviewed per day. This average reduces to **0.0083** when we consider only the distinct jobs of the day.

Case 1: Throughput as a 7-day rolling average of all jobs performed

SQL QUERY

```
USE op_analytics;
SELECT ds AS date,
        jobs_reviewed,
        total_time,
        SUM(jobs_reviewed)
            OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)/SUM(total_time)
            AS throughput_7day_rolling_avg
FROM (SELECT ds,
        COUNT(job_id) AS jobs_reviewed,
        SUM(time_spent) AS total_time
    FROM job_data
    WHERE ds >= '2020-11-01' AND ds <= '2020-11-30'
    GROUP BY ds) win_fn
GROUP BY ds
```

RESULTS

date	jobs_reviewed	total_time	throughput_7day_rolling_avg
2020-11-25	1	45	0.0222
2020-11-26	1	56	0.0357
2020-11-27	1	104	0.0288
2020-11-28	2	33	0.1515
2020-11-29	1	20	0.3000
2020-11-30	2	40	0.2000

This shows that the throughput varies every day with the maximum being **0.3** and the minimum being **0.0222**. To come to this conclusion, we chose a 7-day rolling average instead of a daily metric as it provides finer details for the 6 days in question.

Case 1: Percentage share of each language

SQL QUERY

```
USE op_analytics;
SELECT language,
        COUNT(language) AS language_use_count,
        ROUND(((COUNT(language)/(SELECT count(*) FROM job_data))*100),2)
        AS language_use_percentage
FROM job_data
GROUP BY language
ORDER BY language_use_count DESC, language ASC;
```

RESULTS

language	language_use_count	language_use_percentage
Persian	3	37.50
Arabic	1	12.50
English	1	12.50
French	1	12.50
Hindi	1	12.50
Italian	1	12.50

This shows that **Persian** occupies the major share of activities @ **37.5%.** The remaining languages (**Arabic**, **English**, **French**, **Hindi**, and **Italian**) share the remainder at **12.5%** each.

Case 1: Identifying duplicate rows, if any

SQL QUERY

```
USE op_analytics;
SELECT *
FROM (SELECT *,
        ROW_NUMBER()
        OVER (PARTITION BY job_id) AS duplicate_count
        FROM job_data) win_fn
WHERE duplicate_count > 1;
```

RESULTS

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

This shows that there are **2 rows** which are duplicated on the basis of their **job_id**s. The **duplicate_count** column is used to enumerate the duplicate instances as rows.

Case 2: ER Diagram

Case 2: Weekly User Engagement

SQL QUERY

```
USE op_analytics;
SELECT EXTRACT(YEAR FROM occurred_at) AS Year,
       EXTRACT(WEEK FROM occurred_at) AS Week,
       COUNT(DISTINCT user_id) AS User_Count
FROM events
GROUP BY Year, Week
ORDER BY Year, Week;
```

Case 2: Weekly User Engagement (Cont.)

RESULTS

This shows the user count from weeks **17** to **35** in the year **2014**.

Weeks **17-18** saw a major increase of **520** users while weeks **34-35** saw a major decrease of **1325** users.

Year	Week	User_Count
2014	17	740
2014	18	1260
2014	19	1287
2014	20	1351
2014	21	1299
2014	22	1381
2014	23	1446
2014	24	1471
2014	25	1459
2014	26	1509
2014	27	1573
2014	28	1577
2014	29	1607
2014	30	1706
2014	31	1514
2014	32	1454
2014	33	1438
2014	34	1443
2014	35	118

Case 2: Weekly User Growth

SQL QUERY

```sql
USE op_analytics;
SELECT year,
        week,
        new_active_user,
        SUM(new_active_user)
        OVER (ORDER BY year, week ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_active_user
FROM (SELECT EXTRACT(YEAR FROM activated_at) AS Year,
             EXTRACT(WEEK FROM activated_at) AS Week,
             COUNT(DISTINCT user_id) AS new_active_user
        FROM users
        WHERE state = 'active'
        GROUP BY Year, Week) win_fn
```

Case 2: Weekly User Growth (Cont.)

RESULTS

This shows the weekly user growth from the **1st week** of **2013** to the **35th week** of **2014**. The **33rd week** of **2014** saw the highest number of new active users @ **261**, and the **35th week** of 2014 saw the lowest number of new active users @ **18.**

year	week	new_active_user	cumulative_active_user
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
2013	20	39	820
2013	21	49	869
2013	22	54	923
2013	23	50	973
2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183

year	week	new_active_user	cumulative_active_user
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813
2013	37	85	1898
2013	38	90	1988
2013	39	84	2072
2013	40	87	2159
2013	41	73	2232
2013	42	99	2331
2013	43	89	2420
2013	44	96	2516
2013	45	91	2607
2013	46	88	2695
2013	47	102	2797
2013	48	97	2894
2013	49	116	3010
2013	50	124	3134
2013	51	102	3236
2013	52	47	3283
2014	0	83	3366
2014	1	126	3492
2014	2	109	3601

year	week	new_active_user	cumulative_active_user
2014	3	113	3714
2014	4	130	3844
2014	5	133	3977
2014	6	135	4112
2014	7	125	4237
2014	8	129	4366
2014	9	133	4499
2014	10	154	4653
2014	11	130	4783
2014	12	148	4931
2014	13	167	5098
2014	14	162	5260
2014	15	164	5424
2014	16	179	5603
2014	17	170	5773
2014	18	163	5936
2014	19	185	6121
2014	20	176	6297
2014	21	183	6480
2014	22	196	6676
2014	23	196	6872
2014	24	229	7101
2014	25	207	7308
2014	26	201	7509
2014	27	222	7731
2014	28	215	7946
2014	29	221	8167
2014	30	238	8405

year	week	new_active_user	cumulative_active_user
2014	31	193	8598
2014	32	245	8843
2014	33	261	9104
2014	34	259	9363
2014	35	18	9381

Case 2: Weekly User Retention

SQL QUERY

```
USE op_analytics;
SELECT DISTINCT user_id,
        COUNT(user_id) AS user_id_count,
        SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS retention_per_week
FROM (SELECT win_fn1.user_id,
            win_fn1.signup_week,
            win_fn2.engagement_week,
            win_fn2.engagement_week - win_fn1.signup_week AS retention_week
        FROM ((SELECT DISTINCT user_id,
                    EXTRACT(WEEK FROM occurred_at) AS signup_week FROM events
                WHERE event_type = 'signup_flow' AND event_name = 'complete_signup') win_fn1
        LEFT JOIN (SELECT DISTINCT user_id,
                            EXTRACT(WEEK FROM occurred_at) AS engagement_week FROM events
                    WHERE event_type = 'engagement') win_fn2
        ON win_fn1.user_id = win_fn2.user_id)) win_fn3
GROUP BY user_id
ORDER BY user_id;
```

Case 2: Weekly User Retention (Cont.)

RESULTS

This shows the count of retained users and the number of weeks for which they were retained. Since this data pertains to all the users in the database, only a small section of the data is reproduced here. The entire data can be found in:

user_id	user_id_count	retention_per_week
11768	1	0
11770	1	0
11775	2	1
11778	3	0
11779	5	1
11780	2	1
11785	1	0
11787	3	1
11791	2	1
11793	6	1
11795	2	1
11798	6	1
11799	10	1
11801	2	1
11804	2	1
11806	1	0
11809	1	0
11811	2	1
11813	6	0
11816	3	0
11818	2	1
11820	4	1
11823	3	1
11824	7	1
11825	3	1
11826	2	1
11828	3	1

Case 2: Weekly Device-wise User Engagement

SQL QUERY

```sql
USE op_analytics;
SELECT EXTRACT(YEAR FROM occurred_at) AS Year,
       EXTRACT(WEEK FROM occurred_at) AS Week,
       device AS Device_Used,
       COUNT(DISTINCT user_id) AS User_Count
FROM events
WHERE event_type = 'engagement'
GROUP BY Year, Week, Device_Used
ORDER BY Year, Week, Device_Used
```

Case 2: Weekly Device-wise User Engagement (cont.)

RESULTS

This shows the number of users who have accessed the organization's platform from various devices over the weeks in **2014**. Once again, only a small section of the data is reproduced here, with the entire dataset available in:

Year	Week	Device_Used	User_Count
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

Case 2: User Email Engagement

SQL QUERY

```sql
USE op_analytics;
SELECT
    ROUND(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),2) AS Email_Open_Rate,
    ROUND(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),2) AS Email_Click_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) win_fn;
```

Case 2: User Email Engagement (Cont.)

RESULTS

This shows that **33.58%** of all emails sent are opened by the users. However, only **14.79%** of all emails sent receive an action (click) from the users.

Email_Open_Rate	Email_Click_Rate
33.58	14.79

Result

Based on the insights provided here, internal work practices can be ramped up should they be required to provide better services to users.

Moreover, the various teams of the organization can work together to implement steps for variable user interaction and retention. They can also focus on user experience on different devices to ensure uniform service is provided, irrespective of device format.

Concurrently, email marketing teams could implement various improvements to ensure higher email open and click rates. Campaigns could be run in these mailers that could aid in increased user retention and interaction, as well as new user creation.

This project has been very helpful in understanding advanced SQL concepts and provided insights into how an operations team performs in an organization and helps in its growth.