# PROJECT 3

## Operation Analytics and Investigating Metric Spike

**Project Description:** Project involves analysing user interactions and engagement with the Instagram app to provide valuable insights that can help the business grow. will be using MY SQL Workbench to provide insights which will help in making decisions in future requirements.

**Tech Stack Used:** MySQL Workbench 8.0 CE, MS EXCEL.

**Pre-requisite's:**

- Creating Database and tables. Importing data from CSV Files to SQL.
- Alter the data type of necessary date fields which are in Text to Timestamp.
- Understand the Table Columns before proceeding with the analysis.

**Analysis:**

## Case Study 1: Job Data Analysis

1. **Jobs Reviewed Over Time:**
    - Use Job_Data_Analysis Database.
    - From table job_data, filter the records for Nov 2020.
    - Calculated no.of jobs ran per sum of time spent for each job as jobs reviewed per hr in a day .
    - group by day and sort by jobs_reviewed_perhr_perdy in descending order.

**Insights:** Through this analysis, observed the hourly time review the jobs in a day. With 27/11/2020 being the lowest jobs reviewed with count of 34.61 and 28/11/2020 with highest jobs_reviewed with a count of 218.18.

```sql
CREATE DATABASE Job_Data_Analysis;
USE Job_Data_Analysis;
select * from job_data;
#Jobs reviewed per hour
SELECT
    ds AS day_of_week, (COUNT(job_id) /sum(time_spent)*3600) AS jobs_reviewed_perhr_perdy
FROM
    job_data
    WHERE ds <= '11/30/2020' AND ds >= '11/1/2020'
GROUP BY ds
ORDER BY jobs_reviewed_perhr_perdy DESC;
```

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

| day_of_week | jobs_reviewed_perhr_perdy |
|---|---|
| 11/28/2020 | 218.1818 |
| 11/30/2020 | 180.0000 |
| 11/29/2020 | 180.0000 |
| 11/25/2020 | 80.0000 |
| 11/26/2020 | 64.2857 |
| 11/27/2020 | 34.6154 |

## 2. Throughput Analysis:

- Using Common Table Expressions Query the data no of jobs, sum of time spent as total_time from job_data table.
- Group the results by ds.
- In the main query. calculated the 7-day rolling average for throughput (no of jobs per second).
- Using window functions fetch the sum of jobs/sum of total time for 6 preceding and current row of ds column and rounded to 2 digits from CTE table.

**Insights:** Rolling Average gives better idea, as it considers the previous data and is capable of projecting future insights. I prefer rolling average to daily metrics as rolling average in time series data to analyze trends, especially when short-term fluctuations can hide a longer-term trend or cycle.

```sql
#throughput rolling average.
WITH CTE AS (
 SELECT ds, COUNT(job_id) AS num_jobs, SUM(time_spent) AS total_time_spent
 FROM job_data
 GROUP BY ds)
 SELECT ds,
 SUM(num_jobs) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
 / SUM(total_time_spent) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_avg_7day
 FROM CTE;
```

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

| ds | rolling_avg_7day |
|---|---|
| 11/25/2020 | 0.0222 |
| 11/26/2020 | 0.0198 |
| 11/27/2020 | 0.0146 |
| 11/28/2020 | 0.0210 |
| 11/29/2020 | 0.0233 |
| 11/30/2020 | 0.0268 |

### 3. Language Share Analysis:

- Calculated the percentage share of each language for a span of 30 days.

```sql
SELECT
    language,
    COUNT(*) AS lang_count,
    ROUND((COUNT(*) * 100 / (SELECT
                COUNT(*)
            FROM
                job_data)),
        2) AS Percentage_language
FROM
    job_data
WHERE
    ds <= '11/30/2020' AND ds >= '11/1/2020'
GROUP BY language;
```

| language | lang_count | Percentage_language |
|---|---|---|
| English | 1 | 12.50 |
| Arabic | 1 | 12.50 |
| Persian | 3 | 37.50 |
| Hindi | 1 | 12.50 |
| French | 1 | 12.50 |
| Italian | 1 | 12.50 |

### 4. Duplicate Rows Detection:

- With this query detected the duplicate rows, as there are no duplicate rows in the data, results are empty. But the same approach can be used for the duplicate row's detection.

```sql
# Duplicate detection in a table.
SELECT
    ds, job_id, actor_id, COUNT(*) AS dup_count
FROM
    job_data
GROUP BY ds , job_id , actor_id
HAVING COUNT(*) > 1;
```

| ds | job_id | actor_id | dup_count |
|---|---|---|---|

# Case Study 2: Investigating Metric Spike

1. **Weekly User Engagement:**
    - Use database user_engagement.
    - From the events table, consider the data where event type is not sign-up flow.
    - Query the engagement count and calculate week of occurred_at as week_num using week function.
    - Group and order the results by week_num.

    **Insights**: Analyzing the weekly user engagement. With highest engagement in week 19 and lowest in week 35. This insight will further help in the analysis of weeks with lesser engagement.

```
use  user_engagement;
select * from events;
#weekly user engagement
SELECT
    COUNT(event_name) AS engagement_cnt,
    WEEK(occurred_at) AS week_num
FROM
    events
WHERE
    event_type != 'signup_flow'
    group by week_num order by engagement_cnt desc;
```

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

| engagement_cnt | week_num |
|---|---|
| 17224 | 19 |
| 16344 | 21 |
| 16075 | 30 |
| 15999 | 22 |
| 15415 | 28 |
| 15075 | 29 |
| 14884 | 27 |
| 14839 | 26 |
| 14807 | 24 |
| 14570 | 23 |
| 14429 | 25 |
| 13584 | 31 |
| 11921 | 32 |
| 11342 | 33 |
| 10902 | 34 |
| 8019 | 17 |
| 365 | 35 |

2. **User Growth Analysis:**
   - Created CTE for users table and filtered the data only for active users using where condition.
   - to query the count of user_id as user_growth, monthname and quarter from the created_at column using the respective functions.
   - From CTE, query the User growth, user_growth_metrics i.e., difference in growth compared to previous month.

   **Insights:**

   - From the previous insights there is a fluctuation in the consecutive weeks and a major drop in user engagement from week 34-35. Which raised the question, is it because of the drop in newly created users?
   - With this insight we will calculate the user growth and decrease monthly as weekly data is more, analyzing each week will be difficult.

```
#user growth for product monthly
with cte as(
SELECT
    COUNT(user_id) AS users_created,
    quarter(created_at) as quarter_of_the_yr,
    monthname(created_at) AS month_of_the_yr
FROM
    users
    where state="active"
    group by quarter_of_the_yr,month_of_the_yr
)
select quarter_of_the_yr,month_of_the_yr,users_created,users_created-LAG(users_created) over() as usr_growth_metrics from cte;
```

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

| quarter_of_the_yr | month_of_the_yr | users_created | usr_growth_metrics |
|---|---|---|---|
| 1 | January | 712 | NULL |
| 1 | February | 685 | -27 |
| 1 | March | 765 | 80 |
| 2 | April | 907 | 142 |
| 2 | May | 993 | 86 |
| 2 | June | 1086 | 93 |
| 3 | July | 1281 | 195 |
| 3 | August | 1347 | 66 |
| 3 | September | 330 | -1017 |
| 4 | October | 390 | 60 |
| 4 | November | 399 | 9 |
| 4 | December | 486 | 87 |

3. **Weekly Retention Analysis:**
   - In the sub query m and n, calculated the login week and first login from events table

- In the sub query sub, calculated the week number i.e., difference of login week and first login.
- In the main query calculated the weekly retention based on the weekly cohort.

**Insights:** From this analysis, users who have remained is not a major number.

Which might result in user engagement drop.

```sql
SELECT first AS "Week Numbers",
SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS "Week 17",
SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS "Week 18",
SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS "Week 19",
SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS "Week 20",
SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS "Week 21",
SUM(CASE WHEN week_number = 5 THEN 1 ELSE 0 END) AS "Week 22",
SUM(CASE WHEN week_number = 6 THEN 1 ELSE 0 END) AS "Week 23",
SUM(CASE WHEN week_number = 7 THEN 1 ELSE 0 END) AS "Week 24",
SUM(CASE WHEN week_number = 8 THEN 1 ELSE 0 END) AS "Week 25",
SUM(CASE WHEN week_number = 9 THEN 1 ELSE 0 END) AS "Week 26",
SUM(CASE WHEN week_number = 10 THEN 1 ELSE 0 END) AS "Week 27",
SUM(CASE WHEN week_number = 11 THEN 1 ELSE 0 END) AS "Week 28",
SUM(CASE WHEN week_number = 12 THEN 1 ELSE 0 END) AS "Week 29",
SUM(CASE WHEN week_number = 13 THEN 1 ELSE 0 END) AS "Week 30",
SUM(CASE WHEN week_number = 14 THEN 1 ELSE 0 END) AS "Week 31",
SUM(CASE WHEN week_number = 15 THEN 1 ELSE 0 END) AS "Week 32",
SUM(CASE WHEN week_number = 16 THEN 1 ELSE 0 END) AS "Week 33",
SUM(CASE WHEN week_number = 17 THEN 1 ELSE 0 END) AS "Week 34",
SUM(CASE WHEN week_number = 18 THEN 1 ELSE 0 END) AS "Week 35"
```

```sql
from (
SELECT m.user_id, m.login_week, n.first, m.login_week - first as week_number FROM
(SELECT  user_id, week(occurred_at) as login_week FROM events GROUP BY 1, 2) m,
(SELECT  user_id,  MIN(week(occurred_at)) AS first from events group BY 1) n
WHERE m.user_id = n.user_id ) sub
GROUP BY first
ORDER BY first;
```

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

| Week Numbers | Week 17 | Week 18 | Week 19 | Week 20 | Week 21 | Week 22 | Week 23 | Week 24 | Week 25 | Week 26 | Week 27 | Week 28 | Week 29 | Week 30 | Week 31 | Wee 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | 663 | 472 | 324 | 251 | 205 | 187 | 167 | 146 | 145 | 145 | 136 | 131 | 132 | 143 | 116 | 91 |
| 18 | 596 | 362 | 261 | 203 | 168 | 147 | 144 | 127 | 113 | 122 | 106 | 118 | 127 | 110 | 97 | 85 |
| 19 | 427 | 284 | 173 | 153 | 114 | 95 | 91 | 81 | 95 | 82 | 68 | 65 | 63 | 42 | 51 | 49 |
| 20 | 358 | 192 | 147 | 111 | 85 | 67 | 60 | 65 | 62 | 65 | 67 | 41 | 40 | 33 | 40 | 0 |
| 21 | 317 | 94 | 80 | 60 | 62 | 57 | 71 | 69 | 58 | 48 | 45 | 39 | 35 | 28 | 2 | 0 |
| 22 | 326 | 107 | 93 | 71 | 66 | 59 | 56 | 57 | 53 | 47 | 40 | 39 | 31 | 1 | 0 | 0 |
| 23 | 328 | 101 | 78 | 66 | 64 | 62 | 60 | 56 | 52 | 45 | 34 | 30 | 0 | 0 | 0 | 0 |
| 24 | 339 | 88 | 79 | 65 | 64 | 55 | 59 | 58 | 38 | 38 | 29 | 0 | 0 | 0 | 0 | 0 |
| 25 | 305 | 80 | 66 | 58 | 54 | 52 | 42 | 44 | 38 | 35 | 2 | 0 | 0 | 0 | 0 | 0 |
| 26 | 288 | 73 | 50 | 39 | 50 | 37 | 35 | 34 | 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 292 | 61 | 54 | 54 | 41 | 34 | 29 | 29 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | 274 | 66 | 42 | 32 | 24 | 20 | 24 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | 270 | 60 | 45 | 28 | 23 | 28 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 294 | 52 | 45 | 28 | 28 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | 215 | 40 | 19 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | 267 | 31 | 28 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | 286 | 50 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 34 | 279 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 35 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Week 33 | Week 34 | Week 35 |
|---|---|---|
| 82 | 77 | 5 |
| 67 | 4 | 0 |
| 2 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

**4.Weekly Engagement Per Device**

- From events table, filter the results where event_type is not sign-up flow and event, and name is login.
- Using CTE, query count(device) as device user per week, distinct count of users, week num.
- Group the results weekly and by device.
- Order the results by engagement per user in desc.
- From CTE calculate the avg engagement per user, avg weekly device usage.

**Insights:** From this analysis, I've found that mac book pro users have more engagement , while Samsung tablet has the least.

```sql
#weekly engagement per device
with cte as(
select device,count(device) as device_used_per_week, count(distinct user_id) as engagement_per_user ,
week(occurred_at) as Week_of_year from events
where event_type !="signup_flow" and event_name= "login"
group by Week_of_year,device
order by engagement_per_user desc
)
select device, avg(device_used_per_week) as avg_device_usage_weekly ,
avg(engagement_per_user) as avg_engagement_user_weekly from cte
group by device;
```

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

| device | avg_device_usage_weekly | avg_engagement_user_weekly |
|---|---|---|
| macbook pro | 293.2632 | 219.2632 |
| lenovo thinkpad | 188.9474 | 144.0000 |
| macbook air | 131.4737 | 101.7895 |
| iphone 5 | 136.8333 | 108.7778 |
| nexus 5 | 82.4737 | 65.0526 |
| dell inspiron notebook | 100.3684 | 76.4737 |
| samsung galaxy s4 | 91.0000 | 73.6842 |
| iphone 5s | 80.9444 | 63.6111 |
| ipad air | 48.9444 | 42.7778 |
| dell inspiron desktop | 50.9474 | 39.1579 |
| iphone 4s | 47.7368 | 37.8421 |
| acer aspire notebook | 45.6316 | 36.3684 |
| hp pavilion desktop | 47.2778 | 36.6111 |
| asus chromebook | 46.4737 | 35.8421 |
| nexus 7 | 35.7222 | 31.3333 |
| nexus 10 | 27.6667 | 24.3333 |
| nokia lumia 635 | 30.0526 | 23.8421 |
| ipad mini | 27.2632 | 24.1579 |
| kindle fire | 20.8947 | 17.6316 |
| acer aspire desktop | 27.3158 | 21.8421 |
| htc one | 22.0000 | 17.7368 |
| windows surface | 16.0000 | 14.2632 |
| mac mini | 23.1579 | 17.8947 |
| samsung galaxy note | 14.1667 | 11.3889 |
| amazon fire phone | 11.0000 | 8.8889 |
| samsumg galaxy tablet | 8.2778 | 7.5556 |

## 5.Email Engagement Analysis:

- From sub query calculated the distinct count of users.
- Using if function calculated sum of users based on the action from events table.
- In the main query, calculated the weekly growth for each action using the window function.

**Insights:** Overall picture of email engagement metrics is observed., And metrics seems to be in the positive side as the no of weekly digest emails sent increased weekly so as the email open and email clickthrough.

```sql
#email engagement metrics:
Select
week,
num_users,
time_weekly_digest_sent,
time_weekly_digest_sent-lag(time_weekly_digest_sent) over(order by week) as time_weekly_digest_sent_growth,
time_email_open,time_email_open-lag(time_email_open) over(order by week) as time_email_open_growth,
time_email_clickthrough,time_email_clickthrough-lag(time_email_clickthrough) over(order by week) as time_email_clickthrough_growth
From
(select week(occurred_at)as week,
count(distinct user_id) as num_users,
 sum(if(action='sent_weekly_digest',1,0)) as time_weekly_digest_sent,
sum(if(action='email_open',1,0)) as time_email_open,
sum(if(action='email_clickthrough',1,0)) as time_email_clickthrough
from email_events
group by 1
```

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

| week | num_users | time_weekly_digest_sent | time_weekly_digest_sent_growth | time_email_open | time_email_open_growth | time_email_clickthrough | time_email_clickthrough_growth |
|---|---|---|---|---|---|---|---|
| 17 | 981 | 908 | NULL | 310 | NULL | 166 | NULL |
| 18 | 2714 | 2602 | 1694 | 912 | 602 | 430 | 264 |
| 19 | 2787 | 2665 | 63 | 972 | 60 | 477 | 47 |
| 20 | 2874 | 2733 | 68 | 1004 | 32 | 507 | 30 |
| 21 | 2926 | 2822 | 89 | 1014 | 10 | 443 | -64 |
| 22 | 3029 | 2911 | 89 | 987 | -27 | 488 | 45 |
| 23 | 3134 | 3003 | 92 | 1075 | 88 | 538 | 50 |
| 24 | 3254 | 3105 | 102 | 1155 | 80 | 554 | 16 |
| 25 | 3343 | 3207 | 102 | 1096 | -59 | 530 | -24 |
| 26 | 3439 | 3302 | 95 | 1165 | 69 | 556 | 26 |

| week | num_users | time_weekly_digest_sent | time_weekly_digest_sent_growth | time_email_open | time_email_open_growth | time_email_clickthrough | time_email_clickthrough_growth |
|---|---|---|---|---|---|---|---|
| 26 | 3439 | 3302 | 95 | 1165 | 69 | 556 | 26 |
| 27 | 3543 | 3399 | 97 | 1228 | 63 | 621 | 65 |
| 28 | 3641 | 3499 | 100 | 1250 | 22 | 599 | -22 |
| 29 | 3734 | 3592 | 93 | 1219 | -31 | 590 | -9 |
| 30 | 3866 | 3706 | 114 | 1383 | 164 | 630 | 40 |
| 31 | 3950 | 3793 | 87 | 1351 | -32 | 445 | -185 |
| 32 | 4023 | 3897 | 104 | 1337 | -14 | 418 | -27 |
| 33 | 4200 | 4012 | 115 | 1432 | 95 | 490 | 72 |
| 34 | 4294 | 4111 | 99 | 1528 | 96 | 490 | 0 |
| 35 | 48 | 0 | -4111 | 41 | -1487 | 38 | -452 |

**Result:**

- **Gained the confidence in writing window functions, CTE.**
- **Wherever possible tried to reduce the query execution time for each question by including where clause to filter the data as this data set in huge, grouping the results to reduce the row size.**
- **Avoided writing Subqueries as much as possible.**