

OPERATIONAL ANALYTICS AND INVESTIGATING METRIC SPIKE

1) PROJECT DESCRIPTION: This project involves deriving insights and using those insights to work towards more efficient operations and understand sudden changes in key metrics.

2) APPROACH: The project was executed using Advanced SQL and MySQL workbench. Insights were derived from the data using queries.

3) TECH-STACK USED: My SQL Workbench 8.0 CE was used in analyzing this data. It's portable, lightweight and easy to use. It was used to execute queries, analyze data and derive insights.

4) INSIGHTS:

CASE STUDY 1: JOB DATA ANALYSIS

(A) JOBS REVIEWED OVER TIME

- Calculate the number of jobs reviewed per hour for each day in November 2020.
- Your task: Write a SQL Query to calculate the number of jobs reviewed per hour for each day in November 2020.

```
22 #A. JOBS REVIEWED OVER TIME
23 • SELECT ds, count(job_id) AS jobs_per_day, SUM(time_spent)/3600 AS hours_spent
24 FROM job_data
25 WHERE ds >='2020-11-01' AND ds <='2020-11-30'
26 GROUP BY ds;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
ds	jobs_per_day	hours_spent	
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	2	0.0250	

(B) THROUGHPUT ANALYSIS

- Objective: Calculate the 7-day rolling average of Throughput (number of events per second).
- Task: Write a SQL query to calculate the 7-day rolling average of Throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

WEEKLY THROUGHPUT:

```
31
32 #B THROUGHPUT ANALYSIS
33 • SELECT COUNT(event)/SUM(time_spent) AS Weekly_Throughput
34 FROM job_data;
35
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Weekly_Throughput			
▶	0.0262			

DAILY THROUGHPUT:

```
36 #B THROUGHPUT ANALYSIS
37 • SELECT ds AS DATES, COUNT(event)/SUM(time_spent) AS Daily_Throughput
38 FROM job_data
39 GROUP BY ds
40 ORDER BY ds;
41
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	DATES	Daily_Throughput		
▶	2020-11-25	0.0222		
	2020-11-26	0.0179		
	2020-11-27	0.0096		
	2020-11-28	0.0606		
	2020-11-29	0.0500		
	2020-11-30	0.0500		

The 7-day rolling average throughput is 0.262. I prefer the 7 day rolling average metric over the daily one because it is less overwhelming to sort through and still gives us a useful insights that are just right enough to not be too macro. It is:

- Time-efficient, as there is less data to sort through than compared to the daily metric.
- Effective, is still detailed captures the small details and sorts off the unnecery details irrelevant to understand the bigger picture of things.

Which makes it the most efficient method for analyzing to gain valuable insights for improvement.

(C) LANGUAGE SHARE ANALYSIS

- Objective: Calculate the percentage share of each language in the last 30 days.
- Task: Write a SQL query to calculate the percentage share of each language over the last 30 days.

```
44 #C LANGUAGE SHARE ANALYSIS
45 • SELECT (COUNT(language)*100/(SELECT COUNT(*)language from job_data)) AS share, language
46 FROM job_data
47 WHERE ds >='2020-11-01' AND ds <='2020-11-30'
48 GROUP BY language;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	share	language			
▶	11.1111	English			
	11.1111	Arabic			
	33.3333	Persian			
	11.1111	Hindi			
	11.1111	French			
	22.2222	Italian			

Persian has the **highest share** amongst all the other languages at 33.3%. The second highest share of language after Persian is Italian at 22.2%.

(D) DUPLICATE ROWS DETECTION

- Objective: Identify duplicate rows in the data.
- Task: Write a SQL Query to display duplicate rows from the job_data table.

```
62 #D DUPLICATE ROWS
63 • SELECT * FROM (SELECT *,
64 row_number()over(partition by actor_id) as num_row
65 FROM job_data) a
66 WHERE num_row>1;
```

Result Grid									Filter Rows:	Export:	Wrap Cell Content:
	ds	job_id	actor_id	event	language	time_spent	org	num_row			
▶	2020-11-25	20	1003	transfer	Italian	45	C	2			
	2020-11-25	20	1003	transfer	Italian	45	C	3			

There are 2 duplicate rows sharing the actor_id, 1003.

CASE STUDY 2: INVESTIGATING METRIC SPIKE

(A) WEEKLY USER ENGAGEMENT

- Objective: Measure the activeness of users on a weekly basis.
- Task: Write a SQL query to calculate the weekly user engagement.

```
1  #A
2  • SELECT extract(week from occurred_at) AS week_number,
3     COUNT(distinct user_id) AS active_user
4  FROM events
5  WHERE event_type='engagement'
6  GROUP BY week_number
7  ORDER BY week_number;
```

	week_number	active_user
▶	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

```
9  • SELECT AVG(active_user) AS average_active_user
10 FROM (
11     SELECT extract(week from occurred_at) AS week_number,
12     COUNT(distinct user_id) AS active_user
13     FROM events
14     WHERE event_type='engagement'
15     GROUP BY week_number
16 ) AS weekly_active_users;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
	average_active_user		
	1158.6842		

There is a **steady increase** in the weekly user engagement up until **week 30** during which the highest number of users engaged with our platform at **1467 users**. Indicating that **unique users** engaged with our platform.

There is a drop in users after that however it **does not go below our average of 1158 users**.

There is a **significant dip** in users during **week 35** which saw the lowest user engagement at 104 users.

(B) USER GROWTH ANALYSIS

- Objective: Analyse the growth of users over time for a product.
- Task: Write a SQL Query to calculate the user growth for a product over time.

```
19      #B
20 •    SELECT year, week_num, num_users, sum(num_users) over(order by year, week_num) AS cum_users
21      FROM( SELECT extract(year from created_at) AS year,
22              extract(week from created_at) AS week_num,
23              COUNT(distinct user_id) AS num_users
24            FROM users
25           WHERE state='active\r'
26           GROUP BY year, week_num
27           ORDER BY year, week_num) a;
```

year	week_num	num_users	cum_users	year	week_num	num_users	cum_users	year	week_num	num_users	cum_users	year	week_num	num_users	cum_users
2013	0	23	23	2013	23	50	973	2013	45	91	2607	2014	15	164	5424
2013	1	30	53	2013	24	45	1018	2013	46	88	2695	2014	16	179	5603
2013	2	48	101	2013	25	57	1075	2013	47	102	2797	2014	17	170	5773
2013	3	36	137	2013	26	56	1131	2013	48	97	2894	2014	18	163	5936
2013	4	30	167	2013	27	52	1183	2013	49	116	3010	2014	19	185	6121
2013	5	48	215	2013	28	72	1255	2013	50	124	3134	2014	20	176	6297
2013	6	38	253	2013	29	67	1322	2013	51	102	3236	2014	21	183	6480
2013	7	42	295	2013	30	67	1389	2013	52	47	3283	2014	22	196	6676
2013	8	34	329	2013	31	67	1456	2014	0	83	3366	2014	23	196	6872
2013	9	43	372	2013	32	71	1527	2014	1	126	3492	2014	24	229	7101
2013	10	32	404	2013	33	73	1600	2014	2	109	3601	2014	25	207	7308
2013	11	31	435	2013	34	78	1678	2014	3	113	3714	2014	26	201	7509
2013	12	33	468	2013	35	63	1741	2014	4	130	3844	2014	27	222	7731
2013	13	39	507	2013	36	72	1813	2014	5	133	3977	2014	28	215	7946
2013	14	35	542	2013	37	85	1898	2014	6	135	4112	2014	29	221	8167
2013	15	43	585	2013	38	90	1988	2014	7	125	4237	2014	30	238	8405
2013	16	46	631	2013	39	84	2072	2014	8	129	4366	2014	31	193	8598
2013	17	49	680	2013	40	87	2159	2014	9	133	4499	2014	32	245	8843
2013	18	44	724	2013	41	73	2232	2014	10	154	4653	2014	33	261	9104
2013	19	57	781	2013	42	99	2331	2014	11	130	4783	2014	34	259	9363
2013	20	39	820	2013	43	89	2420	2014	12	148	4931				
2013	21	49	869	2013	44	96	2516	2014	13	167	5098				
2013	22	54	923	2013	45	91	2607	2014	14	162	5260				

The number of users is experiencing a steady growth despite a few fluctuations here and there.

The cumulative users have reached **9363** in over **1.5 years**.

(C) WEEKLY RETENTION ANALYSIS

- Objective: Analyse the retention of users on a weekly basis after signing up.
- Task: Write a SQL Query to calculate the weekly retention of users based on their sign-up cohort.

```
WITH cte1 AS (  
    SELECT DISTINCT user_id,  
        EXTRACT(week from occurred_at) AS signup_week  
    FROM events  
    WHERE event_type='signup_flow'  
    AND event_name='complete_signup'),  
cte2 AS (  
    SELECT DISTINCT user_id,  
        EXTRACT(week from occurred_at) as engagement_week  
    FROM events  
    WHERE event_type='engagement' )  
  
SELECT signup_week, COUNT(user_id) total_engaged_users,  
    SUM(case when retention_week > 1 then 1 else 0 end) as retained_users  
FROM(  
    SELECT a.user_id, a.signup_week,  
        b.engagement_week, b.engagement_week - a.signup_week AS retention_week  
    FROM cte1 a  
    LEFT JOIN cte2 b  
    ON a.user_id = b.user_id  
    ORDER BY a.user_id ) sub  
GROUP BY signup_week  
ORDER BY signup_week;
```

signup_week	total_engaged_users	retained_users
17	278	147
18	615	338
19	677	350
20	682	378
21	644	340
22	694	356
23	707	365
24	700	320
25	671	299
26	636	297
27	697	314
28	596	220
29	588	207
30	614	205
31	451	122
32	508	89
33	456	8
34	302	0
35	18	0

Weekly retention after signing up was at a **steady rise up until week 20**, which was its **highest at 378 retained users**. After that there were slight declines as well increase, which gradually became a **significant decline** and came down to **0 retained users for week 34 and 35**.

(D) WEEKLY ENGAGEMENT PER DEVICE

- Objective: Measure the activeness of users on a weekly basis per device.
- Task: Write s SQL query to calculate the weekly engagement per device.

```
SELECT
COUNT(event_type) AS event_num,
EXTRACT(week from occurred_at) AS weeknum,
device
from events
where event_type='engagement'
GROUP BY weeknum,device
ORDER BY weeknum, event_num;
```

Link to the table obtained:

<https://drive.google.com/file/d/1QJLR8POU45tblcyQc9aLuJy0eHqspifD/view?usp=sharing>

The **highest** engagement was on **week 31** with **3608** engagements on **MacBook Pro Max**.

The **lowest** engagement was on **week 35** with **4** engagements on **Dell Inspiron Desktop**.

(E) EMAIL ENGAGEMENT ANALYSIS

- Objective: Analyse how users are engaging with the email service.
- Task: Write a SQL query to calculate the email engagement metrics.

```
#E
SELECT
100*SUM(CASE WHEN email_at='email_open' then 1 else 0 end) /
SUM(CASE WHEN email_at='email_sent' then 1 else 0 end) AS email_openrate,
100*SUM(CASE WHEN email_at='email_clicked' then 1 else 0 end) /
SUM(CASE WHEN email_at='email_sent' then 1 else 0 end) AS email_clickrate
FROM(
SELECT *,
CASE
WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email_sent'
WHEN action = 'email_open' THEN 'email_open'
WHEN action = 'email_clickthrough' THEN 'email_clicked'
END AS email_at
FROM email_events) sub;
```

	email_openrate	email_clickrate
▶	33.5834	14.7899

The email open rate is **33.58** and the click rate is **14.78**. Which means that a little less than half the opened emails had a clickthrough to their link.

(5) RESULT: I have improved my SQL skills and learned to write more Advanced queries using functions like subqueries, case statements, over() function etc. It has taught me how to fetch data needed for analysis in the most efficient way possible. Alongside this, I got valuable hands-on experience of the kind of work that is done by a Data Analyst on a daily basis.