PROJECT 3

Operation Analytics and Investigating Metric Spike

Advanced SQL

Project description:

This project focuses on Operational Analytics to enhance the company's efficiency. As a Lead Data Analyst at a major tech company like Microsoft, my responsibility will be to analyze various datasets and tables to derive insights that can answer concerns raised by different departments like operations, support, and marketing.

A key aspect of my role will be investigating sudden changes in key metric like drops in sales or decreases in daily user engagement. By advanced SQL skills, I will analyze causes of sudden changes , enabling the company to make informed decisions.

Approach:

- The database was created using the structured data provided.
- The questions were carefully reviewed to understand the requirements of the provided questions.
- The tables and columns that will be used to answer each question were identified.
- Required information was extracted using MySQL queries using MySQL software.
- The insights derived from the analysis were carefully reviewed to further take informed decisions.

Tech-Stack Used: I am using MySQL Workbench 8.0.38-winx64 CE and it is ideal for this project as it provides a user-friendly interface for running SQL queries, managing databases, visualising trends, making it easier to analyze and derive insights efficiently.

Case Study 1: Job Data Analysis

Tasks:

A. Jobs Reviewed Over Time: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

Given that the 'ds' column only stores the date without any hour information, it is impossible to calculate the number of jobs reviewed per hour. The hourly breakdown is only possible if the dataset includes both the date and time for when each job was reviewed.

Query 1: approach 1 – we can only calculate totals per day, not per hour.

```
use project_3;

select ds as review_date,count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent
from job_data
where ds between '2020-11-01'  and  '2020-11-30'
group by ds
order by ds;
```

Result:

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

Query 2: approach 2 – we can calculate the average number of jobs reviewed per hour for each day assuming there is an even distribution throughout the day over month.

```
use project_3;

select ds, count(job_id)/ 30*24 as job_per_hour
from job_data
group by ds;
```

Result:

	ds	job_per_hour
▶	2020-11-30	1.6000
	2020-11-29	0.8000
	2020-11-28	1.6000
	2020-11-27	0.8000
	2020-11-26	0.8000
	2020-11-25	0.8000

B. Throughput Analysis: Write an 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.

Query:

```
use project_3;

SELECT ds AS review_date, daily_throughput, AVG(daily_throughput)
    OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_7_day_avg
FROM
( select ds, count(job_id) as daily_throughput
from job_data
group by ds
order by ds) as count_table ;
```

Result:

	review_date	daily_throughput	rolling_7_day_avg
▶	2020-11-25	1	1.0000
	2020-11-26	1	1.0000
	2020-11-27	1	1.0000
	2020-11-28	2	1.2500
	2020-11-29	1	1.2000
	2020-11-30	2	1.3333

Insights:

Daily Metric vs. 7-Day Rolling Average

1. Daily Metric: The daily metric gives throughput for each specific day. It is useful for identifying day to day fluctuations but daily metric can be volatile, especially if there are irregular patterns in the data.

2. 7-Day Rolling Average: The 7-day rolling average smooths out daily fluctuations by averaging the throughput over a week making it easier to spot long term trends and patterns.

I would prefer the 7-day rolling average for throughput analysis to understand trends over. The rolling average provides a more stable measure by smoothing out fluctuations that helps in making informed decisions based on consistent patterns. But keeping an eye on the daily metric might be necessary as well if immediate responsiveness to changes is important.

C. Language Share Analysis: Write an SQL query to calculate the percentage share of each language over the last 30 days.

Query:

```
use project_3;

Select language, count(language) as total_language,
       ROUND((COUNT(*) * 100.0) / SUM(COUNT(*)) OVER (),2) AS percentage_share
from job_data
group by language
order by percentage_share DESC;
```

Result:

language	total_language	percentage_share
Persian	3	37.50
English	1	12.50
Arabic	1	12.50
Hindi	1	12.50
French	1	12.50
Italian	1	12.50

Insights:

The above table shows that Persian language accounts for 37.5% of job reviews so it is the most prevalent language in the dataset.

Other languages each have a 12.5% share, showing a balanced but lesser representation.

D. Duplicate Rows Detection: Write an SQL query to display duplicate rows from the job_data table.

Query:

```
use project_3;

 select *
 from (
      select*, row_number() over(partition by job_id) as row_num
      from job_data) as rowcount
where row_num > 1;
```

Result:

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

Insights:

The above table shows that rows 2 and 3 are duplicates when partitioning the data by job_id.

Case Study 2: Investigating Metric Spike

Tasks:

A. Weekly User Engagement: Write an SQL query to calculate the weekly user engagement.

Query:

```
use project_3;

select extract(week from occurred_at) as weeks , count(distinct user_id) as active_users
from events
where event_type = 'engagement'
group by weeks;
```

Result:

weeks	active_users
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

Insights:

The table shows weekly user engagement. The user engagement has been increasing from 17th week to 30th week but has been declining after that showing users are not finding quality in product or service.

B. User Growth Analysis: Write an SQL query to calculate the user growth for the product.

Query:

```
use project_3;

SELECT
    DATE_FORMAT(activated_at, '%Y-%m') AS month,
    COUNT(*) AS new_users,
    SUM(COUNT(*)) OVER (ORDER BY DATE_FORMAT(activated_at, '%Y-%m')) AS cumulative_users
FROM
    users
GROUP BY
    month
ORDER BY
    month;
```

Result:

month	new_users	cumulative_users
2013-01	160	160
2013-02	160	320
2013-03	150	470
2013-04	181	651
2013-05	214	865
2013-06	213	1078
2013-07	284	1362
2013-08	316	1678
2013-09	330	2008
2013-10	390	2398
2013-11	399	2797
2013-12	486	3283
2014-01	552	3835
2014-02	525	4360
2014-03	615	4975
2014-04	726	5701
2014-05	779	6480
2014-06	873	7353
2014-07	997	8350
2014-08	1031	9381

Insights:

The table shows the growth of users over time for a product. From Jan,2013 to Aug,2014 there are over 9300 active users providing a view of overall user growth.

C. Weekly Retention Analysis: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Query:

```sql
use project_3;

WITH cohorts AS (
    SELECT
        DATE(activated_at) AS cohort_start_date,
        COUNT(*) AS total_users
    FROM users
    GROUP BY 1
),
weekly_stats AS (
    SELECT
        DATE(u.activated_at) AS cohort_start_date,
        YEARWEEK(e.occurred_at, 0) AS year_week,
        COUNT(DISTINCT e.user_id) AS active_users
    FROM users u
    JOIN events e ON u.user_id = e.user_id
    WHERE e.event_type = 'engagement'
    GROUP BY cohort_start_date, year_week
)
    SELECT
        cohorts.cohort_start_date,
        weekly_stats.year_week,
        weekly_stats.active_users,
        cohorts.total_users AS total_users,
        weekly_stats.active_users / cohorts.total_users * 100 AS retention_rate
    FROM cohorts
    JOIN weekly_stats
        ON cohorts.cohort_start_date = weekly_stats.cohort_start_date
    ORDER BY cohort_start_date, year_week;
```

Insights:

The result shows how many users from a specific signup cohort remain active over subsequent weeks.

D. Weekly Engagement Per Device: Write an SQL query to calculate the weekly engagement per device.

Query:

```
use project_3;

SELECT
    DATE_FORMAT(occurred_at, '%Y-%u') AS week,
    device,
    COUNT(DISTINCT user_id) AS active_users
FROM events
GROUP BY week, device;
```

Insights:

The activeness of users on a weekly basis per device shows that MacBook pro has the most active users.

E. Email Engagement Analysis: Write an SQL query to calculate the email engagement metrics.

Query:

```
SELECT
    DATE_FORMAT(occurred_at, '%Y-%u') AS week,
    COUNT(CASE WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 1 ELSE NULL END) AS em
    COUNT(CASE WHEN action = 'email_open' THEN 1 ELSE NULL END) AS emails_opened,
    COUNT(CASE WHEN action = 'email_clickthrough' THEN 1 ELSE NULL END) AS emails_clicked,
    ROUND(
        100.0 * COUNT(CASE WHEN action = 'email_open' THEN 1 ELSE NULL END) /
        NULLIF(COUNT(CASE WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 1 ELSE NULL
        2
    ) AS email_open_rate,
    ROUND(
        100.0 * COUNT(CASE WHEN action = 'email_clickthrough' THEN 1 ELSE NULL END) /
        NULLIF(COUNT(CASE WHEN action = 'email_open' THEN 1 ELSE NULL END), 0),
        2
    ) AS email_click_rate
FROM email_events
WHERE action IN ('sent_weekly_digest', 'sent_reengagement_email', 'email_open', 'email_clickthrough')
GROUP BY week
ORDER BY week;
```

Result:

week	emails_sent	emails_opened	emails_clicked	email_open_rate	email_click_rate
2014-18	1006	332	187	33.00	56.33
2014-19	2766	919	434	33.22	47.23
2014-20	2840	971	479	34.19	49.33
2014-21	2912	995	498	34.17	50.05
2014-22	3001	1026	453	34.19	44.15
2014-23	3110	993	492	31.93	49.55
2014-24	3193	1070	533	33.51	49.81
2014-25	3339	1161	563	34.77	48.49
2014-26	3394	1090	524	32.12	48.07
2014-27	3524	1168	559	33.14	47.86
2014-28	3613	1230	622	34.04	50.57
2014-29	3725	1260	607	33.83	48.17
2014-30	3798	1211	584	31.89	48.22
2014-31	3936	1386	633	35.21	45.67
2014-32	3999	1336	432	33.41	32.34
2014-33	4121	1357	430	32.93	31.69
2014-34	4269	1421	487	33.29	34.27
2014-35	4374	1533	493	35.05	32.16

Insights:

The above table shows how users are engaging with the email service. The data is on the weekly basis showing metrics like email open rate and email click rate. This query helps in understanding how users are engaging with email campaigns over time which provides insights into the effectiveness of email content.

Result:

Through this project I was able to manage a structured database for the project. The project helped in gaining understanding of managing tables, applying joins, and performing complex queries to extract relevant information through hands-on experience in performing data analysis using SQL queries, filtering data that provided valuable insights.

From this project, I learned how to identify and analyze unexpected changes in metrics to understand trends, helping improve operational performance and make informed decisions.

Also, applied MySQL in a real-world case, translating theoretical knowledge into practical use which is essential for any data-driven role.