OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

MIRRA. G

PROJECT DESCRIPTION:

Operation Analytics and Investigating Metric Spike is a project that revolves around analysing user engagement and growth metrics for a digital product, focusing on understanding user behaviour, measuring engagement, tracking growth, evaluating retention, and analysing email engagement. By collecting and preprocessing data from various sources including user databases, event logs, and email service logs, SQL queries are developed to calculate key metrics such as weekly user engagement, user growth, retention, and email engagement. Insights derived from the analysis are visualized using charts, graphs, and dashboards to provide actionable recommendations for optimizing user engagement and growth strategies, ultimately enhancing product performance and user experience.

APPROACH:

In the project, SQL is utilized extensively to query and manipulate data from various databases containing user information, event logs, and email engagement records. SQL queries are crafted to extract relevant data and calculate key metrics such as weekly user engagement, user growth, retention rates, and email engagement metrics. By leveraging SQL effectively, the project ensures efficient data management, accurate metric calculation, and insightful analysis, contributing to informed decision-making processes aimed at optimizing user engagement and enhancing product growth.

TECH STACK USED:

MySQL Workbench 8.0 CE

CASE STUDY 1: JOB DATA ANALYSIS

1. **Jobs Reviewed Over Time:**
   Objective: **Calculate the number of jobs reviewed per hour for each day in November 2020.**
   Your Task: **Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.**

   QUERY:

   ```
   SELECT COUNT(job_id) / (30 * 24)
   AS number_of_jobs_reviewed_per_day_non_distinct
   FROM job_data;
   ```

   OUTPUT:

   Result Grid | Filter Rows:

	number_of_jobs_reviewed_per_day_non_distinct
▶	0.0111

   ANALYSIS:

   The output of the SQL query provides the average number of jobs reviewed per day over a 30-day period. It offers insights into the daily workload of reviewers, aiding in resource planning and performance evaluation. Comparing this metric over time or across teams can highlight trends and areas for improvement.

2. **Throughput Analysis:**
   Objective: **Calculate the 7-day rolling average of throughput (number of events per second).**
   Your Task: **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:

   ```
   SELECT ds as date_of_review, jobs_reviewed, AVG (jobs_reviewed)
   OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
   AS throughput_7_rolling_average
   FROM
   (
   ```

```
SELECT ds, COUNT(job_id) AS jobs_reviewed
FROM job_data
GROUP BY ds ORDER BY ds
) a;
```

OUTPUT:

Result Grid | Filter Rows: | Export: | Wr

date_of_review	jobs_reviewed	throughput_7_rolling_average
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

ANALYSIS:

The output presents the date of job reviews, the count of jobs reviewed each day, and the 7-day rolling average of job reviews. By comparing daily counts with the rolling average, it offers insights into trends and deviations in reviewer activity over time. This allows for performance monitoring, trend analysis, and informed decision-making to optimize productivity and efficiency in the review process.
Considering the nature of the data and the purpose of analysis, if short-term fluctuations are critical to understand or respond to (e.g., for operational decision-making), then the daily metric might be preferred. On the other hand, if the focus is on identifying longer-term trends while minimizing the impact of short-term noise, the 7-day rolling average could be more suitable.

3. **Language Share Analysis:**
   Objective: **Calculate the percentage share of each language in the last 30 days.**
   Your Task: **Write an SQL query to calculate the percentage share of each language over the last 30 days.**

QUERY:

```
SELECT job_id, language,
COUNT(language) AS total_of_each_language,
((COUNT(language) / (SELECT COUNT(*) FROM job_data)) * 100) AS
percentage_share_of_each_language
FROM job_data
GROUP BY language, job_id;
```

OUTPUT:

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

job_id	language	total_of_each_language	percentage_share_of_each_language
21	English	1	12.5000
22	Arabic	1	12.5000
23	Persian	3	37.5000
25	Hindi	1	12.5000
11	French	1	12.5000
20	Italian	1	12.5000

ANALYSIS:

The output provides insights into the distribution of job reviews across languages, presenting the total count and percentage share of each language in the dataset. It helps in understanding the prevalence of different languages in the review process, guiding resource allocation and content management strategies to align with user needs and preferences.

4. **Duplicate Rows Detection:**
   Objective: **Identify duplicate rows in the data.**
   Your Task: **Write an SQL query to display duplicate rows from the job_data table.**

QUERY:

```
SELECT *
FROM
(
SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num
FROM job_data
) a
WHERE row_num>1;
```

OUTPUT:

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

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

ANALYSIS:

The output of the SQL query highlights duplicate rows in the `job_data` table based on the `job_id` column. It achieves this by assigning row numbers to each row within partitions defined by `job_id`, then filtering out rows with row numbers greater than 1. This concise analysis provides valuable insights for data quality assessment and corrective actions to ensure dataset accuracy.

CASE STUDY 2: INVESTIGATING METRIC SPIKE

1. **Weekly User Engagement:**
   Objective: **Measure the activeness of users on a weekly basis.**
   Your Task: **Write an SQL query to calculate the weekly user engagement.**

   QUERY:

```
SELECT
   EXTRACT(WEEK FROM STR_TO_DATE(occurred_at, '%Y-%m-%d')) AS
week_number,
   COUNT(DISTINCT user_id) AS number_of_users
FROM
   events
GROUP BY
   EXTRACT(WEEK FROM STR_TO_DATE(occurred_at, '%Y-%m-%d'))
ORDER BY
   week_number;
```

   OUTPUT:

week_number	number_of_users
20	2346
21	259
24	2116
25	1970
28	1953
29	2712
33	2674
34	1056

ANALYSIS:

The output of the query provides a weekly count of distinct active users, revealing trends in user engagement over time. By examining the number of active users each week, businesses can identify patterns such as increases due to successful marketing campaigns or decreases that may indicate retention issues. Seasonal variations and the impact of specific events on user activity can also be observed. This data is crucial for understanding user behavior, planning resources, and making informed decisions to enhance user satisfaction and engagement. Overall, it offers a clear picture of how user activity fluctuates on a weekly basis, helping to drive strategic initiatives.

2. **User Growth Analysis:**
   Objective: **Analyze the growth of users over time for a product.**
   Your Task: **Write an SQL query to calculate the user growth for the product.**

QUERY:

```
SELECT
   year_num, week_num, num_active_users,
   SUM(num_active_users) OVER (ORDER BY year_num, week_num ROWS
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS
cum_active_users
FROM (
   SELECT
      EXTRACT(YEAR FROM STR_TO_DATE(a.activated_at, '%Y-%m-%d')) AS
year_num,
      EXTRACT(WEEK FROM STR_TO_DATE(a.activated_at, '%Y-%m-%d')) AS
week_num,
      COUNT(DISTINCT user_id) AS num_active_users
   FROM
      users a
   WHERE
      state = 'active'
   GROUP BY
      year_num, week_num
   ORDER BY
      year_num, week_num
) a;
```

OUTPUT:

Full output:

https://drive.google.com/file/d/1C7sDxRS8B-YR-mOM_F5-BvX5z4fCqX1o/view?usp=sharing

year_num	week_num	num_active_users	cum_active_users
2001	2	23	23
2001	7	13	36
2001	11	14	50
2001	15	29	79
2001	20	44	123
2001	24	15	138
2001	28	46	184
2001	33	54	238
2001	37	4	242
2001	41	16	258
2001	46	17	275
2001	50	5	280
2002	3	42	322

year_num	week_num	num_active_users	cum_active_users
2030	20	50	9009
2030	24	38	9047
2030	28	56	9103
2030	33	29	9132
2030	37	14	9146
2030	42	15	9161
2030	46	6	9167
2030	50	19	9186
2031	3	29	9215
2031	11	35	9250
2031	20	18	9268
2031	29	55	9323
2031	33	20	9343
2031	42	16	9359
2031	50	22	9381

ANALYSIS:

The output of the query provides a weekly count of active users and a cumulative total of active users over time, segmented by year and week. This data is crucial for tracking user engagement trends and understanding growth patterns. The `num_active_users` column shows how many users were activated and marked as active each week, while the `cum_active_users` column accumulates this count over the weeks. Analyzing this output can reveal periods of significant user activation, highlight successful onboarding or marketing efforts, and help identify any seasonal or cyclical trends in user engagement. This comprehensive view aids in strategic planning and improving user retention initiatives.

3. **Weekly Retention Analysis:**
   Objective: **Analyze the retention of users on a weekly basis after signing up for a product.**
   Your Task: **Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.**

QUERY:

```
SELECT
   DISTINCT user_id,
   COUNT(user_id),
   SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS
per_week_retention
FROM
(
```

```
    SELECT
        a.user_id,
        a.signup_week,
        b.engagement_week,
        b.engagement_week - a.signup_week AS retention_week
    FROM
    (
        SELECT DISTINCT
            user_id,
            EXTRACT(WEEK FROM STR_TO_DATE(occurred_at, '%Y-%m-%d')) AS
signup_week
        FROM events
        WHERE event_type = 'signup_flow'
          AND event_name = 'complete_signup'
    ) a
    LEFT JOIN
    (
        SELECT DISTINCT
            user_id,
            EXTRACT(WEEK FROM STR_TO_DATE(occurred_at, '%Y-%m-%d')) AS
engagement_week
        FROM events
        WHERE event_type = 'engagement'
    ) b
    ON a.user_id = b.user_id
) d
GROUP BY user_id
ORDER BY user_id;
```

OUTPUT:

Full output:
https://drive.google.com/file/d/1D3jj2CwxgctBbsyY9c5LANzLuHYNXYBY/view?usp=sharing

user_id	COUNT(user_id)	per_week_retention
11768	1	0
11770	1	0
11775	1	0
11778	2	0
11779	1	0
11780	1	0
11785	1	0
11787	1	0
11791	1	0
11793	2	0
11795	1	0
11798	2	0
11799	6	0
11801	1	0

user_id	COUNT(user_id)	per_week_retention
11901	2	0
11906	6	1
11908	2	1
11909	4	1
11914	4	1
11919	2	0
11920	1	0
11924	1	0
11926	3	0
11928	4	0
11929	1	0
11931	4	0
11933	3	0
11936	2	0

ANALYSIS:

The output of this query provides insights into user retention by identifying the number of users who engage with the platform within the first week after signing up. Specifically, the `COUNT(user_id)` column represents the total number of engagement events per user, while the `per_week_retention` column counts how many users engage within their first week of signing up. This data is crucial for understanding early user engagement and retention patterns, as it highlights the effectiveness of initial onboarding processes and the platform's ability to retain users shortly after their signup. High retention in the first week can indicate a strong onboarding experience, while low retention might suggest areas for improvement in engaging new users.

4. **Weekly Engagement Per Device:**
   Objective: **Measure the activeness of users on a weekly basis per device.**
   Your Task: **Write an SQL query to calculate the weekly engagement per device.**

QUERY:

```
SELECT
   EXTRACT(YEAR FROM STR_TO_DATE(occurred_at, '%Y-%m-%d')) AS
year_num,
   EXTRACT(WEEK FROM STR_TO_DATE(occurred_at, '%Y-%m-%d')) AS
week_num,
   device,
   COUNT(DISTINCT user_id) AS no_of_users
FROM
   events
WHERE
   event_type = 'engagement'
GROUP BY
   year_num, week_num, device
ORDER BY
   year_num, week_num, device;
```

OUTPUT:

Full output:
https://drive.google.com/file/d/1-WdLg76Z0AMcYgjIUqxoFD-bc3hEcV5j/view?usp=sharing

year_num	week_num	device	no_of_users
2001	20	acer aspire desktop	5
2001	20	acer aspire notebook	10
2001	20	amazon fire phone	2
2001	20	asus chromebook	9
2001	20	dell inspiron desktop	8
2001	20	dell inspiron notebook	22
2001	20	hp pavilion desktop	4
2001	20	htc one	5
2001	20	ipad air	8
2001	20	ipad mini	6
2001	20	iphone 4s	10
2001	20	iphone 5	22
2001	20	iphone 5s	18
2001	20	kindle fire	2
2001	20	lenovo thinkpad	40

year_num	week_num	device	no_of_users
2031	33	iphone 4s	6
2031	33	iphone 5	2
2031	33	iphone 5s	3
2031	33	kindle fire	3
2031	33	lenovo thinkpad	16
2031	33	mac mini	2
2031	33	macbook air	10
2031	33	macbook pro	17
2031	33	nexus 10	2
2031	33	nexus 5	4
2031	33	nexus 7	2
2031	33	nokia lumia 635	2
2031	33	samsung galaxy note	1
2031	33	samsung galaxy s4	6
2031	33	windows surface	3

ANALYSIS:

The output of this query provides a detailed view of user engagement across different devices on a weekly basis. By extracting the year and week from the `occurred_at` timestamp and grouping by these time periods along with the device type, the query counts the number of distinct users engaged each week per device. This analysis reveals user preferences and engagement patterns across various devices over time. For instance, an increase in the number of mobile device users compared to desktop users over several weeks could indicate a shift in user behavior towards mobile usage. Such insights are valuable for making informed decisions on where to focus development resources and marketing efforts, ensuring that the platform optimally supports the devices that users prefer. Additionally, identifying weeks with significant changes in engagement can help pinpoint the impact of specific events or updates on user activity.

5. **Email Engagement Analysis:**
   Objective: **Analyze how users are engaging with the email service.**
   Your Task: **Write an SQL query to calculate the email engagement metrics.**

QUERY:

```
SELECT
   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) AS
email_opening_rate,
   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) AS
email_clicking_rate
FROM
(
   SELECT
      *,
      CASE
```

```
            WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN
'email_sent'
            WHEN action = 'email_open' THEN 'email_opened'
            WHEN action = 'email_clickthrough' THEN 'email_clicked'
        END AS email_cat
    FROM
        email_events
) a;
```

OUTPUT:

	email_opening_rate	email_clicking_rate
▶	33.58339	14.78989

ANALYSIS:

The output of this query calculates the email opening and clicking rates, providing key metrics for evaluating the effectiveness of email campaigns. The `email_opening_rate` represents the percentage of sent emails that were opened by recipients, while the `email_clicking_rate` indicates the percentage of sent emails that resulted in a click-through. These rates are derived by dividing the number of `email_opened` and `email_clicked` actions by the total number of `email_sent` actions, respectively. Analyzing these rates helps in understanding how engaging and compelling the email content is to the audience. High opening and clicking rates suggest effective subject lines and content that resonate with recipients, whereas low rates may indicate a need for improved email strategies, such as more enticing subject lines, better targeting, or enhanced call-to-actions within the emails.

RESULT :

Hence, we have implemented all the tasks given as a part of the Operation Analytics and Investigating Metric Spike project and provided the queries and outputs along with the analysis.