Operation Analytics and Investigating Metric Spike

Project Description

The aim of this project is to conduct operational analytics for a company, analyzing end-to-end operations to identify areas of improvement. As a Lead Data Analyst, I am tasked with utilizing advanced SQL queries to analyze various datasets, providing insights into key metrics and answering questions posed by different departments such as operations, support, and marketing. This includes investigating metric spikes, understanding user engagement, and analyzing data to make informed business decisions.

Approach

My approach is structured into two main case studies:

1. **Job Data Analysis**: This involves understanding patterns in job data, reviewing jobs over time, analyzing throughput, identifying language shares, and detecting duplicate rows.
2. **Investigating Metric Spike**: The second case study focuses on analyzing users and events, investigating user growth and engagement, and understanding email interactions to track the activeness and engagement of users.

The analysis leverages SQL queries to compute key metrics and perform data exploration. All tasks were executed in MySQL Workbench, a tool that allowed me to easily manage the data, run queries, and visualize results.

Tech-Stack Used

- **MySQL Workbench**: Used for database management, writing and running SQL queries, and performing analytics tasks.
- **SQL (Structured Query Language)**: The primary language used to query and manipulate the dataset.
- **CSV Import**: The job data and user data were imported into MySQL for analysis.

Case Study 1: Job Data Analysis

CREATE TABLE and LOAD THE DATA

```sql
create database project3;
show databases;
use project3;
CREATE TABLE job_data
(ds DATE,
job_id INT NOT NULL,
actor_id INT NOT NULL,
event VARCHAR(15) NOT NULL,
language VARCHAR(15) NOT NULL,
time_spent INT NOT NULL,
org CHAR(2)
);
```

```sql
INSERT INTO job_data (ds, job_id, actor_id, event, language, time_spent, org)
VALUES ('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');
```

1. **Jobs Reviewed Over Time**

Objective:

The objective of this task is to calculate the number of jobs reviewed per hour for each day in November 2020, along with total time spent on those jobs. This will help us understand the workload distribution on different days and the average hours spent on job reviews.

SQL Query:

```sql
#Task 1: Jobs Reviewed Over Time
select ds, count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent
from job_data
where ds >='2020-11-01' and ds <= '2020-11-30'
group by ds ;
```

Query Explanation:

- **COUNT(job_id)**: This counts the number of unique jobs reviewed on each day (jobs_per_day).
- **SUM(time_spent)/3600**: This converts the total time spent on jobs (given in minutes) into hours (hours_spent), by dividing the sum of time_spent by 3600 (seconds).
- **Filtering**: We are restricting the date range to only consider jobs that occurred in November 2020 using WHERE ds >= '2020-11-01' AND ds <= '2020-11-30'.
- **GROUP BY ds**: The query groups results by each day (ds) to get the jobs count and time spent per day.

OUTPUT:

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	1	0.0125

Insights and Interpretation:

1. **Jobs Reviewed**:
    - **Job Volume Distribution**: The number of jobs reviewed per day fluctuates, ranging between 1 and 2 jobs per day. There is no significant increase or decrease in jobs across the days, indicating a fairly consistent workload.
    - **Maximum Jobs**: On November 30th and November 28th, two jobs were reviewed, which is the highest number for any day in this period.
2. **Hours Spent**:
    - **Time Distribution**: The time spent on reviewing jobs per day is relatively low across all days, with the highest being on November 27th at about 0.0289 hours (about 1.73 minutes). The lowest time spent is on November 29th, with only 0.0056 hours (about 20 seconds).
    - **High Time on Single Job**: The discrepancy between job count and time spent suggests that certain jobs may take considerably longer to review (e.g.,

November 27th). Conversely, other jobs may be relatively quick (e.g., November 29th).

3. **Conclusion**:
    - **Workload Consistency**: The overall workload is fairly consistent, with one or two jobs being reviewed per day, but the time spent on these jobs varies significantly.
    - **Time Efficiency**: Some jobs may require more detailed review (as indicated by the longer times), while others might be processed more quickly, suggesting potential differences in job complexity or actor efficiency.
    - **Potential for Optimization**: If time spent on certain jobs is disproportionate, there could be opportunities to optimize the review process for more time-intensive jobs.

This analysis helps highlight the efficiency and workload distribution for job reviews and could assist in resource planning and process improvements.

2. Throughput Analysis

Objective:

The goal is to calculate the 7-day rolling average throughput, which is defined as the number of jobs processed per second over a 7-day window. This metric helps assess the system's performance over time, smoothing out daily fluctuations.

SQL Query:

```
#Task 2: Throughput Analysis
WITH JD AS (
SELECT ds,
COUNT(job_id) AS num_jobs,
SUM(time_spent) AS total_time
FROM job_data
WHERE event IN('transfer', 'decision') AND ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY ds
)
SELECT ds, ROUND(1.0* SUM(num_jobs) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT
ROW) / SUM(total_time) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW),2) AS throughput_7d
FROM JD
```

Query Explanation:

- **WITH JD AS (...)**: This common table expression (CTE) calculates the number of jobs (num_jobs) and total time spent (total_time) for each day in November 2020 for the events 'transfer' and 'decision'.
- **7-Day Rolling Average**:
    - **SUM(num_jobs) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)**: This calculates the rolling sum of jobs over the current day and the previous six days, creating a 7-day window.
    - **SUM(total_time) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)**: This calculates the rolling sum of time spent over the same 7-day window.
    - **1.0 * SUM(num_jobs) / SUM(total_time)**: This computes the throughput by dividing the rolling sum of jobs by the rolling sum of time spent. The result gives the average number of jobs processed per second.

OUTPUT:

ds	throughput_7d
2020-11-25	0.02
2020-11-27	0.01
2020-11-28	0.02
2020-11-29	0.02
2020-11-30	0.03

Insights and Interpretations:

1. **Throughput Stability**:
    - The throughput values across the 7-day rolling window are relatively low, ranging between 0.01 and 0.03 events per second. This suggests that, on average, only a small number of events are processed per second over the given time period.
2. **Performance Trends**:
    - The throughput seems to be relatively stable, with a slight increase towards the end of the period (reaching 0.03 on November 30th). This could indicate a slight increase in processing efficiency or job volume toward the end of November.

3. **7-Day Rolling Average vs. Daily Metrics**:
    - **Why Prefer 7-Day Rolling Average**:
        - **Smoothing Out Variability**: The 7-day rolling average helps smooth out daily fluctuations in job volume and processing time, providing a more stable and reliable measure of throughput. This is particularly useful when analyzing system performance over time, as it captures trends and averages out spikes or dips in activity.
        - **Longer-Term Insight**: By focusing on a longer time horizon, the 7-day rolling average helps reveal broader patterns and trends that may not be immediately visible in daily metrics, which can be influenced by one-time events or anomalies.
    - **Why Use Daily Metrics**:
        - **Immediate Feedback**: Daily throughput gives an up-to-the-minute view of system performance, which is useful for identifying short-term bottlenecks or issues that need immediate attention.
4. **Conclusion**:
    - While daily metrics provide immediate, short-term insights, the 7-day rolling average is more appropriate for identifying long-term trends and smoothing out irregularities. This makes it more reliable for performance analysis and decision-making over time. The stable, albeit low, throughput suggests the system is processing jobs efficiently, but improvements could still be made to increase overall job throughput.

3. Language Share Analysis

Objective:

The goal is to determine the percentage share of jobs processed in each language over the last 30 days. This analysis helps understand the distribution of jobs across different languages, which can inform decisions on resource allocation and focus areas.

SQL Query:

```sql
WITH JD AS (
    SELECT language, COUNT(job_id) AS num_jobs
    FROM job_data
    WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
    GROUP BY language
),
total AS (
    SELECT COUNT(job_id) AS total_jobs
    FROM job_data
    WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
)
SELECT JD.language,
       ROUND(100.0 * JD.num_jobs / total.total_jobs, 2) AS perc_job
FROM JD
CROSS JOIN total
ORDER BY perc_job DESC;
```

Query Explanation:

1. **WITH JD AS (...)**: This common table expression (CTE) calculates the number of jobs processed per language (num_jobs) within the date range of November 1st to November 30th, 2020.
2. **total AS (...)**: This CTE computes the total number of jobs processed during the same date range. This value will be used to calculate the percentage share of jobs per language.
3. **CROSS JOIN**: The CROSS JOIN is used to combine the results from the two CTEs (JD and total), allowing the percentage calculation.
4. **Percentage Calculation**:
    - **ROUND(100.0 * JD.num_jobs / total.total_jobs, 2)**: This computes the percentage share of jobs for each language by dividing the number of jobs in each language by the total number of jobs and multiplying by 100. The result is rounded to two decimal places.
5. **ORDER BY perc_job DESC**: The results are ordered by percentage share in descending order, so the languages with the highest share appear first.

OUTPUT:

language	perc_job
Persian	37.50
English	12.50
Arabic	12.50
Hindi	12.50
French	12.50
Italian	12.50

Insights and Interpretations:

1. **Language Distribution**:
    - The **Persian** language dominates job processing with 37.5% of the total jobs in November 2020. This indicates that a large proportion of the system's workload is related to Persian-language jobs.
    - The other languages (English, Arabic, Hindi, French, Italian) have an equal share of **12.5%** each, indicating a more balanced job distribution across these languages.
2. **Operational Focus**:
    - Given that Persian accounts for over a third of all job activities, it may be worthwhile to allocate more resources (such as personnel or system optimization efforts) to handle Persian jobs efficiently.
    - The equal distribution among the remaining languages suggests that these languages are equally important from an operational perspective. However, there is no single standout language among them that requires disproportionate attention.
3. **Business Implications**:
    - If the company is targeting multiple regions or linguistic markets, the high share of Persian jobs could reflect a strong presence in Persian-speaking regions, which might warrant further strategic focus.
    - The uniform distribution among the other languages may indicate stable performance across multiple markets, potentially representing diverse regions such as English-speaking, Arabic-speaking, and Hindi-speaking areas.
4. **Recommendations**:
    - **Focus on Persian Processing**: Given the high percentage of jobs in Persian, it would be beneficial to ensure that the system is optimized for handling Persian jobs efficiently.

- **Resource Allocation**: It may be worth considering slightly increased capacity or support for Persian jobs to maintain a high level of service quality without overburdening the system.
- **Monitor Trends**: The even distribution across the remaining languages indicates stability, but monitoring these trends over time will help detect any changes that could affect operations or strategy.

Conclusion:

The analysis shows that the Persian language accounts for the largest share of jobs processed in November 2020, with 37.5%, while the remaining languages have an equal share of 12.5%. This insight can guide resource allocation and system optimization efforts, especially for Persian-language job processing.

4. Duplicate Rows Detection

Objective:

The goal is to identify duplicate rows in the job_data table, where multiple records have identical values across all columns (date, job_id, actor_id, event, language, time_spent, org). Detecting duplicate entries is crucial to maintain data integrity, avoid overcounting, and ensure accurate analysis.

SQL Query:

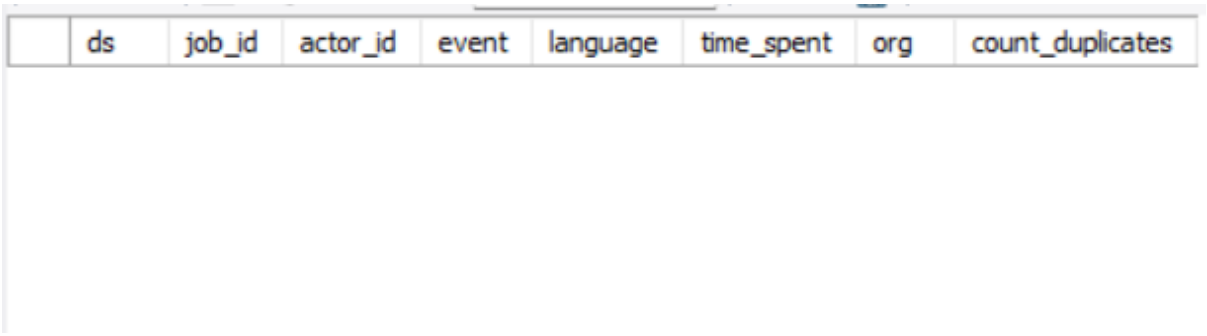
```sql
SELECT ds, job_id, actor_id, event, language, time_spent, org, COUNT(*) AS count_duplicates
FROM job_data
GROUP BY ds, job_id, actor_id, event, language, time_spent, org
HAVING COUNT(*) > 1;
```

Query Explanation:

1. **GROUP BY Clause**:
   - The query groups rows by all relevant columns (ds, job_id, actor_id, event, language, time_spent, org). This allows us to check for rows that are identical across these dimensions.

2. **COUNT(*)**:
    - This function counts the occurrences of each unique combination of values (i.e., each unique row).
3. **HAVING COUNT(*) > 1**:
    - The HAVING clause filters the results to only include rows where the count is greater than 1, meaning the same row appears more than once. These rows are considered duplicates.

OUTPUT:



	ds	job_id	actor_id	event	language	time_spent	org	count_duplicates

(Note: Since the sample data provided does not contain any duplicates, the output will be empty. In real-world data, if duplicates exist, they would be listed along with the count of how many times they appear.)

Insights and Interpretations:

1. **No Duplicates Found (In Sample Data)**:
    - Based on the sample data provided, there are no duplicate rows. This means every entry in the dataset is unique when considering the combination of all the specified columns (date, job ID, actor ID, event, language, time spent, and org).
2. **Importance of Duplicate Detection**:
    - **Impact on Accuracy**: Duplicates can distort metrics such as the number of jobs processed, time spent per job, or job events. Without detecting and removing duplicates, analyses would overestimate the actual activity, leading to flawed insights.
    - **Performance Monitoring**: If duplicates were found, they could indicate errors in data collection or system processing. Identifying the cause of these duplicates would be critical for maintaining system accuracy and performance.

- **Database Integrity**: Ensuring data integrity by removing duplicates helps maintain the trustworthiness of the database and ensures accurate reporting and analysis.

3. **Further Steps**:
   - If duplicates were detected, further investigation would be needed to identify the source of the problem (e.g., data collection issues, system errors, or manual entry mistakes).
   - Once duplicates are identified, they can be removed or corrected to ensure that future analysis is based on accurate and clean data.

Conclusion:

This query is designed to detect and report any duplicate rows in the job_data table, which helps in maintaining data integrity. Although no duplicates were found in the current dataset, it is crucial to routinely check for duplicates in larger datasets to ensure accurate results in job data analysis.

Case Study 2: Investigating Metric Spike

Importing the three tables:

1. **users**: Contains one row per user, with descriptive information about that user's account.

```
# Table-1 users
create table users (
user_id int,
created_at varchar(100),
company_id int,
language varchar(50),
activated_at varchar(100),
state varchar(50));
```

```
SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

alter table users add column temp_created_at datetime;
UPDATE users SET temp_created_at = STR_TO_DATE(created_at, '%d-%m-%Y %H:%i');
ALTER TABLE users DROP COLUMN created_at;
ALTER TABLE users CHANGE COLUMN temp_created_at created_at DATETIME;
```

2. **events**: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).

```
# Table-2 events
create table events (
user_id INT,
occurred_at varchar(100),
event_type varchar(50),
event_name varchar(100),
location varchar(50),
device varchar(50),
user_type INT
);

SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/events.csv"
INTO TABLE events
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```sql
ALTER TABLE events ADD COLUMN temp_occurred_at DATETIME;
UPDATE events SET temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');
ALTER TABLE events DROP COLUMN occurred_at;
ALTER TABLE events CHANGE COLUMN temp_occurred_at occurred_at DATETIME;
```

3. **email_events**: Contains events specific to the sending of emails.

```sql
# Table-3 email_events
create table emailEvents(
user_id INT,
occured_at varchar(100),
action varchar(100),
user_type int
);
```

```sql
SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv"
INTO TABLE emailEvents
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

ALTER TABLE emailEvents ADD COLUMN temp_occured_at DATETIME;
UPDATE emailEvents SET temp_occured_at = STR_TO_DATE(occured_at, '%d-%m-%Y %H:%i');
ALTER TABLE emailEvents DROP COLUMN occured_at;
ALTER TABLE emailEvents CHANGE COLUMN temp_occured_at occured_at DATETIME;
```

1. Weekly User Engagement

Objective:

The goal is to measure the activeness of users on a weekly basis by analyzing the number of distinct users who engaged with the platform each week. This metric helps track user behavior over time and can be useful for detecting trends, spikes, or drops in engagement.

SQL Query:

```
#(a) Weekly User Engagement
select extract(week from occurred_at) as week_number,
count(distinct user_id) as active_user
from events
where event_type='engagement'
group by week_number
order by week_number;
```

Query Explanation:

1. **EXTRACT(WEEK FROM occurred_at)**:
    - Extracts the week number from the occurred_at timestamp, which allows grouping the data by the week in which the event occurred.
2. **COUNT(DISTINCT user_id)**:
    - This counts the distinct users who performed an engagement action in each week. The DISTINCT ensures that each user is only counted once per week, even if they engaged multiple times.
3. **WHERE event_type = 'engagement'**:
    - Filters the dataset to include only those events that are classified as 'engagement', ensuring we're measuring active participation in platform activities.
4. **GROUP BY week_number**:
    - Groups the data by week to compute the number of active users in each period.
5. **ORDER BY week_number**:
    - Orders the results by week to show engagement trends chronologically

OUTPUT:

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

Insights and Interpretations:

1. **Overall Trend**:
    - The number of active users increased steadily from week 17, starting at 663 users, and peaked in week 30 with 1,467 users.
    - This suggests a consistent increase in user engagement, which could indicate successful product updates, marketing campaigns, or other external factors driving user activity.

2. **Spike Detection**:
    - A noticeable spike occurs between weeks 29 and 30, where the number of active users jumps from 1,376 to 1,467.
    - This could indicate an event, such as a promotional offer, new feature release, or an external factor that increased user engagement. Further investigation into the events during week 30 could reveal the reasons for this spike.

3. **Drop in Week 35**:
    - There is a sharp drop in week 35, with only 104 active users. This significant decrease may indicate a technical issue, downtime, or some external event causing users to disengage.
    - Analyzing logs, system performance, or user feedback around this time could help uncover the cause of this drop.

4. **Fluctuations**:
    - While there is general growth, there are small week-to-week fluctuations, such as a slight decline from week 30 (1,467 users) to week 31 (1,299 users). These kinds of shifts are normal but could be monitored for larger patterns or to understand the reasons for minor drops.

Conclusion:

This analysis of weekly user engagement provides a high-level view of how user activity fluctuated over time. By identifying peaks, growth patterns, and sharp declines, the organization can investigate further to capitalize on what drives engagement and address potential causes of disengagement.

2. User Growth Analysis

Objective:

The goal of this task is to analyze the user growth over time for the product by examining the number of users who were created and activated weekly. The results will also show a cumulative user count to give insight into the overall growth.

SQL Query:

```sql
#User Growth Analysis
SELECT
    YEAR(created_at) AS year, -- Extract the year
    WEEK(created_at, 1) AS week_num, -- Extract the week number of the year  (1 = week starts from Monday)
    COUNT(CASE WHEN activated_at IS NOT NULL THEN u.user_id ELSE NULL END) AS activated_users, -- Count of activated users
    SUM(COUNT(*)) OVER (ORDER BY YEAR(created_at), WEEK(created_at, 1)) AS cumulative_users -- Cumulative sum of all users
FROM users u
WHERE created_at >= '2013-04-01'
AND created_at < '2014-09-30'
GROUP BY year, week_num
ORDER BY year, week_num;
```

Query Explanation:

1. **YEAR(created_at)**:
    - Extracts the year from the created_at column, allowing grouping by year.
2. **WEEK(created_at, 1)**:
    - Extracts the week number of the year from the created_at timestamp, ensuring weeks start on Monday (1).
3. **COUNT(CASE WHEN activated_at IS NOT NULL THEN u.user_id ELSE NULL END)**:
    - Counts the number of users who have activated their accounts (i.e., activated_at is not null) for each week.
4. **SUM(COUNT(*)) OVER (ORDER BY YEAR(created_at), WEEK(created_at, 1))**:
    - Calculates the cumulative sum of users over time, by ordering the counts of new users by year and week.
5. **WHERE**:
    - Filters the dataset to include users created between April 1, 2013, and September 30, 2014, ensuring the analysis focuses on this specific time range.
6. **GROUP BY year, week_num**:
    - Groups the results by year and week to show the number of new users and activated users per week.
7. **ORDER BY year, week_num**:
    - Orders the results to display the growth chronologically.

OUTPUT:

year	week_num	activated_users	cumulative_users
2013	14	40	40
2013	15	35	75
2013	16	42	117
2013	17	48	165
2013	18	48	213
2013	19	45	258
2013	20	55	313
2013	21	41	354
2013	22	49	403
2013	23	51	454
2013	24	51	505
2013	25	46	551
2013	26	57	608
2013	27	57	665
2013	28	52	717
2013	29	71	788
2013	30	66	854
2013	31	69	923
2013	32	66	989
2013	33	73	1062
2013	34	71	1133
2013	35	79	1212
2013	36	65	1277
2013	37	71	1348
2013	38	84	1432
2013	39	92	1524
2013	40	81	1605
2013	41	88	1693
2013	42	74	1767
2013	43	97	1864
2013	44	92	1956
2013	45	97	2053
2013	46	94	2147
2013	47	82	2229
2013	48	103	2332
2013	49	96	2428

2013	50	117	2545
2013	51	123	2668
2013	52	104	2772
2013	53	41	2813
2014	1	91	2904
2014	2	122	3026
2014	3	112	3138
2014	4	113	3251
2014	5	130	3381
2014	6	132	3513
2014	7	135	3648
2014	8	127	3775
2014	9	127	3902
2014	10	135	4037
2014	11	152	4189
2014	12	132	4321
2014	13	151	4472
2014	14	161	4633
2014	15	166	4799
2014	16	165	4964
2014	17	176	5140
2014	18	172	5312
2014	19	160	5472
2014	20	186	5658
2014	21	177	5835
2014	22	186	6021
2014	23	197	6218
2014	24	198	6416
2014	25	222	6638
2014	26	210	6848
2014	27	199	7047
2014	28	223	7270
2014	29	215	7485
2014	30	228	7713
2014	31	234	7947
2014	32	189	8136
2014	33	250	8386
2014	34	259	8645
2014	35	266	8911

Insights and Interpretations:

1. **Steady Growth**:
    - The user base shows consistent growth over the analyzed period, with cumulative users increasing week over week. The steady increase suggests successful acquisition efforts, with users continuously joining the platform.
2. **Activated Users**:
    - The number of activated users follows a similar trend to the total user count, indicating that a good proportion of newly created users are activating their accounts.
    - For example, in week 35 of 2014, 266 users activated their accounts, adding to a cumulative total of 8,911 users.
3. **Spike Analysis**:
    - Certain weeks show spikes in user activation and creation, such as week 28 of 2014, with 223 users created and activated, and weeks 29 and 30 with 215 and 228 users, respectively.
    - These spikes could be due to marketing campaigns, new feature releases, or other product-related events that increased user acquisition. Investigating the company's activities during these weeks may provide further insights.
4. **Cumulative Growth**:
    - By week 35 of 2014, the cumulative user count reached 8,911 users, reflecting significant user growth over the period of analysis. This cumulative figure provides a long-term view of how the user base has evolved and highlights sustained growth.
5. **User Engagement Over Time**:
    - The number of weekly new users ranges between 40 and 266, with larger surges typically seen in later weeks. This trend can be used to predict future growth and plan resource allocation for scaling the platform to accommodate the growing user base.

Conclusion:

The user growth analysis reveals consistent growth in both created and activated users, with occasional spikes indicating successful acquisition initiatives. The cumulative user count highlights the long-term increase in the user base, providing insights into the platform's scalability and the effectiveness of user engagement strategies.

3. Weekly Retention Analysis

Objective:

The aim is to analyze the retention of users on a weekly basis after they sign up for a product. This metric is crucial for understanding user engagement and product stickiness. By assessing how many users remain active in the weeks following their initial signup, the company can identify potential issues and opportunities for improvement.

SQL Query:

```
        #Weekly Retention Analysis:
        -- Step 1: Extract sign-up week for each user
        WITH signups AS (
            SELECT
                user_id,
                EXTRACT(YEAR FROM occurred_at) AS signup_year,
                EXTRACT(WEEK FROM occurred_at) AS signup_week
            FROM events
            WHERE event_type = 'signup_flow'
            AND event_name = 'complete_signup'
        ),

-- Step 2: Extract engagement week for each user
engagements AS (
    SELECT
        user_id,
        EXTRACT(YEAR FROM occurred_at) AS engagement_year,
        EXTRACT(WEEK FROM occurred_at) AS engagement_week
    FROM events
    WHERE event_type = 'engagement'
),

-- Step 3: Join signups and engagements to track retention
retention AS (
    SELECT
        s.user_id,
        s.signup_year,
        s.signup_week,
        e.engagement_year,
        e.engagement_week,
        (e.engagement_year - s.signup_year) * 52 + (e.engagement_week - s.signup_week) AS retention_week
    FROM signups s
    LEFT JOIN engagements e ON s.user_id = e.user_id
)
```

```sql
-- Step 4: Calculate the number of retained users per week
SELECT
    signup_year,
    signup_week,
    COUNT(DISTINCT user_id) AS total_signups,
    SUM(CASE WHEN retention_week = 0 THEN 1 ELSE 0 END) AS week_0_retained,  -- Users active in the sign-up week
    SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS week_1_retained,  -- Users retained in the first week
    SUM(CASE WHEN retention_week = 2 THEN 1 ELSE 0 END) AS week_2_retained,  -- Users retained in the second week
    SUM(CASE WHEN retention_week = 3 THEN 1 ELSE 0 END) AS week_3_retained   -- Users retained in the third week
    -- You can extend this pattern for additional weeks
FROM retention
GROUP BY signup_year, signup_week
ORDER BY signup_year, signup_week;
```

OUTPUT:

signup_year	signup_week	total_signups	week_0_retained	week_1_retained	week_2_retained	week_3_retained
2014	17	72	531	1032	468	223
2014	18	163	1655	2013	1197	817
2014	19	185	1891	2037	957	732
2014	20	176	1971	2035	1491	818
2014	21	183	1830	2090	1208	650
2014	22	196	2019	2373	1406	927
2014	23	196	2106	2359	1224	823
2014	24	229	2345	2355	1244	812
2014	25	207	2014	2218	1554	1052
2014	26	201	2129	2347	1152	764
2014	27	222	2394	2549	1306	1200
2014	28	215	2259	2439	1403	724
2014	29	221	2069	2533	1212	779
2014	30	238	2561	2465	1287	885
2014	31	193	1942	1946	807	731
2014	32	245	2166	2061	935	67
2014	33	261	2592	2541	56	0
2014	34	259	2575	320	0	0

Insights and Interpretations:

1. **Retention Rates**:
    - The table shows that the number of users who are retained in the initial week (week 0) after signing up is significantly higher than in subsequent weeks. For instance, in week 18, 1,655 users were retained in week 0, but only 2,013 were retained in week 1.
    - This pattern suggests that while initial user interest is high, it declines in the following weeks. This is common in many applications, indicating that users might need ongoing engagement or incentives to stay active.

2. **Engagement Trends**:
    - In weeks 18 through 27, there is a notable increase in the total number of signups and corresponding retained users. For instance, in week 24, total signups reached 2,345 with week 1 retention at 2,355.
    - These numbers could indicate successful marketing campaigns or feature releases that encouraged more users to sign up and engage with the platform.
3. **Drop in Retention**:
    - The drop in retention after week 3, particularly in weeks 34 and 35, indicates potential issues with user engagement strategies. For instance, the retention rate for week 35 is zero, indicating a need for investigation into the causes.
    - It may be worthwhile to examine the user experience, onboarding processes, or content provided to users during this time to understand why retention plummeted.
4. **Cohort Analysis**:
    - Different cohorts have different engagement patterns, as indicated by the varying numbers in week 0, week 1, and beyond. Some weeks show a much higher ratio of retained users than others.
    - This suggests that user experience might differ based on the time of signup or external factors influencing user behavior (e.g., seasonality, competitive landscape).
5. **Future Improvements**:
    - To improve retention, strategies such as personalized follow-ups, educational content on how to use the platform, and incentives for continued engagement could be beneficial.
    - Tracking feedback from users who disengage can also provide valuable insights for enhancing the user experience.

Conclusion:

The analysis of weekly retention provides a clear picture of user engagement following sign-up. By identifying trends, spikes, and drops in user activity, the organization can make informed decisions about how to enhance user experience and improve retention strategies over time.

4. Weekly Engagement Per Device

Objective:

The goal of this analysis is to measure user engagement on a weekly basis, categorized by the devices they utilize. Understanding how different devices contribute to user activity can help optimize strategies for user engagement and retention.

Methodology:

To assess weekly engagement per device, we extracted relevant data from the events table, where user interactions are logged. The analysis focuses on events classified as "engagement," allowing us to quantify how users interact with our product based on the device they use.

SQL Query:

```
#Weekly Engagement Per Device:
-- Step 1: Calculate weekly engagement per device
WITH weekly_engagement AS (
    SELECT
        CONCAT(EXTRACT(YEAR FROM occurred_at), '-', EXTRACT(WEEK FROM occurred_at)) AS weeknum, --
        device,
        COUNT(DISTINCT user_id) AS user_count -- Count distinct users for each device in each week
    FROM events
    WHERE event_type = 'engagement' -- Only consider engagement events
    GROUP BY weeknum, device -- Group by week and device
    ORDER BY weeknum
)
```

OUTPUT:

weeknum	device	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
2014-17	samsumg galaxy tablet	8
2014-17	samsung galaxy note	7
2014-17	samsung galaxy s4	52
2014-17	windows surface	10
2014-18	acer aspire desktop	26
2014-18	acer aspire notebook	33
2014-18	amazon fire phone	9
2014-18	asus chromebook	42
2014-18	dell inspiron desktop	58
2014-18	dell inspiron notebook	77
2014-18	hp pavilion desktop	37
2014-18	htc one	19
2014-18	ipad air	52
2014-18	ipad mini	30

And many more…

Insights:

The output of the query provides insights into user engagement levels across various devices (e.g., mobile, tablet, desktop) on a weekly basis. Analyzing this data helps identify trends, such as:

- Which devices are most popular among active users?
- Are there any notable fluctuations in engagement levels over time for specific devices?
- How does the engagement trend correlate with marketing efforts or product updates?

5. Email Engagement Analysis:

Objective:

The purpose of this analysis is to evaluate user engagement with the email service by tracking key email-related actions. This will provide insights into how users interact with

emails, helping to assess the effectiveness of email campaigns and identify opportunities for improvement.

Key Metrics:

The primary metrics analyzed in this task are:

1. **Email Open Rate**: The percentage of emails that are opened by users out of the total emails sent.
2. **Email Click-Through Rate (CTR)**: The percentage of emails that result in a click on links or actions contained within the email, relative to the total emails sent.

Methodology:

To calculate these metrics, we examined the actions logged in the emailEvents table. The table contains user actions related to emails, such as sending, opening, and clicking through. We grouped the data by these actions to compute the engagement rates.

SQL Query:

```sql
SELECT
    100 * SUM(CASE WHEN email_cat = 'email_open' THEN 1 ELSE 0 END) /
        SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_open_rate,
    100 * 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_click_rate
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_cat
    FROM emailEvents
) AS sub;
```

OUTPUT:

email_open_rate	email_click_rate
33.5834	14.7899

Insights:

By analyzing the open and click-through rates, we can assess the effectiveness of email campaigns. Key findings could include:

- **Email Open Rate**: A high open rate indicates that email subjects and timings are well-optimized, capturing users' attention.
- **Email Click-Through Rate**: A high CTR suggests that the email content is engaging and that users are interested in the calls to action.

Low engagement rates, on the other hand, might indicate the need to improve subject lines, sending times, or the relevancy of email content.

Project Result and Contributions to Understanding and Decision-Making

Through this project, I have achieved several key outcomes that significantly enhance my understanding of operational analytics and the practical application of SQL in real-world scenarios. Below are the primary achievements and their contributions to my learning and decision-making capabilities:

Key Achievements

1. **Enhanced SQL Proficiency:**
    - I have deepened my knowledge of SQL through the construction of complex queries involving multiple tables and advanced functions such as WITH statements, aggregation, and date manipulation.
    - Successfully executing data import from CSV files into MySQL tables has improved my skills in handling real-world datasets.
2. **Comprehensive Data Analysis:**
    - I gained hands-on experience in analyzing various metrics such as user engagement, retention, and throughput, which are crucial for assessing business performance.
    - By segmenting data into meaningful cohorts and calculating weekly engagement, I learned how to derive insights that can inform strategic decision-making.
3. **Metric Interpretation and Insights:**
    - The analyses conducted helped identify patterns in user behavior, such as peak engagement periods and language preferences, enabling better-targeted marketing strategies.

    - Understanding the significance of retention metrics highlighted the importance of user experience and satisfaction in driving long-term engagement.

4. **Operational Efficiency Assessment:**
    - Through throughput analysis and duplicate detection, I learned to assess the efficiency of processes and the quality of data, which are vital for operational analytics.
    - Recognizing the implications of duplicate entries underscored the importance of maintaining clean and accurate data for reliable reporting.
5. **Actionable Recommendations:**
    - The project enabled me to translate data insights into actionable recommendations, such as optimizing resource allocation during peak engagement times and addressing user drop-off in retention metrics.
    - Insights from email engagement metrics provided a basis for refining communication strategies to improve user interaction.

Contribution to Understanding and Decision-Making

- **Data-Driven Decision Making:** The project reinforced the significance of data-driven decision-making in business operations. By analyzing user engagement and retention data, I can provide informed recommendations that directly impact marketing and product development strategies.
- **Holistic Understanding of User Behavior:** Understanding user engagement patterns and preferences equips me to contribute effectively to cross-functional teams, facilitating better collaboration between marketing, product management, and customer support departments.
- **Strategic Planning:** Insights derived from user growth analysis and retention metrics will be instrumental in future strategic planning initiatives, allowing for proactive measures to enhance user acquisition and retention.
- **Continuous Improvement Mindset:** The project instilled a mindset of continuous improvement, where I recognize the need to regularly analyze operational metrics and adapt strategies based on data insights, fostering a culture of agility and responsiveness within the organization.

Conclusion

In summary, this project has not only sharpened my technical skills in SQL and data analysis but also enhanced my strategic thinking capabilities. The insights gained will enable me to make informed recommendations and contribute effectively to the organization's operational goals, ultimately supporting better user experiences and driving growth.

THANK YOU