5/28/2025

Operation Analytics and Investigating Metric Spike

Advanced SQL

Deepthy A
TRAINITY DATA ANALYTICS TRAINEE

Project Description

This project is broken up into two case studies that concentrate on the following topics:

- **Job Data Analysis**, which covers operational metrics like language distribution, job review throughput, and duplicate detection.
- **Examining Metric Spike**, which deals with abrupt shifts in user behaviour like declines in engagement or spikes in user registrations.

The goal is to use advanced SQL skills to extract actionable insights from massive amounts of data so that various departments, including marketing, operations, and support, can make data-driven decisions.

Approach

To tackle this project effectively, I divided my work into the following steps:

1. **Database Setup**: I created the necessary tables by importing the provided CSV files into MySQL Workbench.
2. **Exploratory Analysis**: I began by reviewing the structure of each table to understand the data types, column meanings, and relationships.
3. **Query Development**: I wrote optimized SQL queries for each task, ensuring that the logic aligns with the business goals of each department.
4. **Snapshot & Reporting**: After executing each query, I captured both the SQL code and result outputs to include them in this report.
5. **Insight Generation**: I analyzed each result to identify trends, anomalies, and opportunities for improvement.

Tech Stack Used

Tool	Version	Purpose
MySQL Workbench	8.0+	Writing and executing SQL queries, visualizing table schemas
MS Word	360	Preparing and exporting the report
Google Drive	Cloud	Hosting the final deliverable with public access

Case Study 1: Job Data Analysis

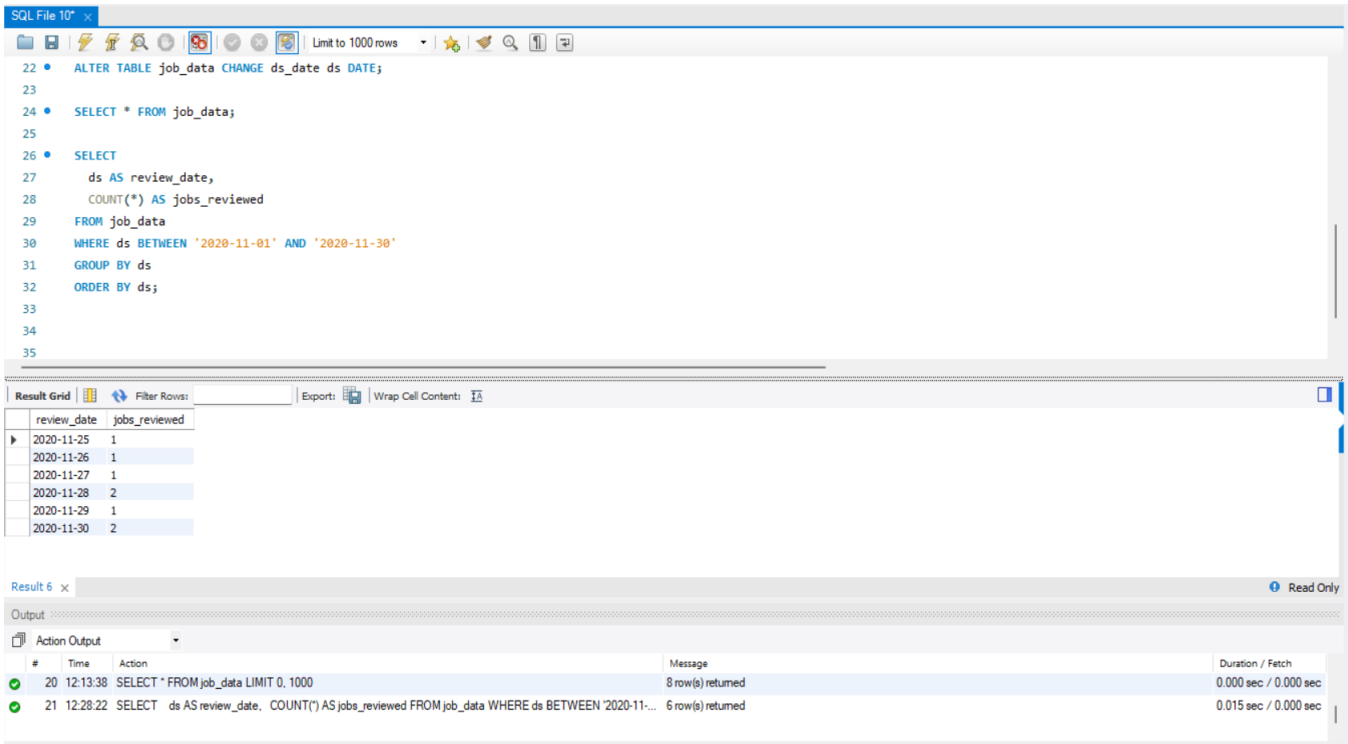
I have been analyzing the job_data table with these columns:

- job_id: Unique job identifier
- actor_id: ID of person handling the job
- event: Action taken (e.g., decision, skip, transfer)
- language: Language of the job
- time_spent: Time spent reviewing in seconds
- org: Organization name
- ds: Date (as a string in format yyyy/mm/dd)

Tasks:

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.*



Insight: The job review volume during November 2020 is quite low and sporadic. This could indicate low workload, underutilization of reviewers, or data not being fully populated. A consistent pattern is missing, which might prompt checks on data collection processes or system logging reliability.

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

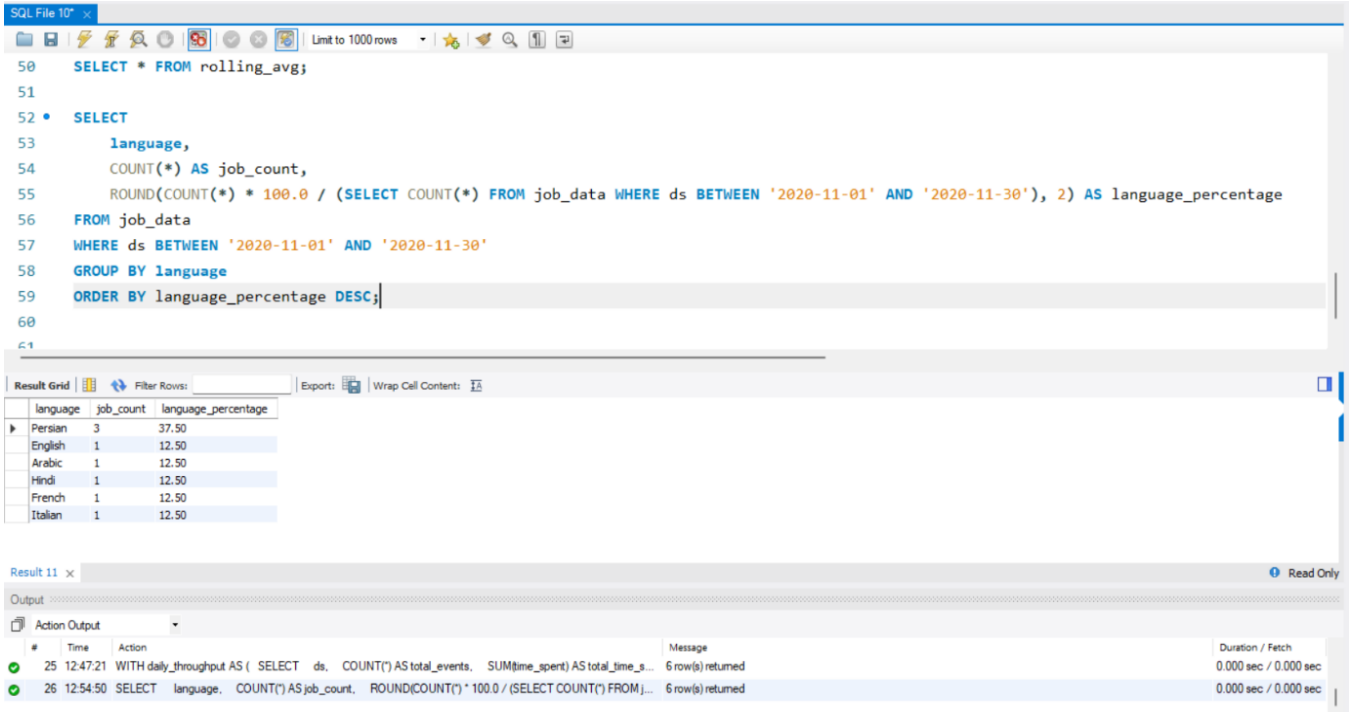
```sql
ORDER BY ds;

WITH daily_throughput AS (
  SELECT
    ds,
    COUNT(*) AS total_events,
    SUM(time_spent) AS total_time_spent,
    (COUNT(*) / SUM(time_spent)) AS throughput
  FROM job_data
  GROUP BY ds
),
rolling_avg AS (
  SELECT
    ds,
    throughput,
    ROUND(AVG(throughput) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW), 6) AS rolling_7_day_avg_throughput
  FROM daily_throughput
)
SELECT * FROM rolling_avg;

SELECT
```

```sql
    (COUNT(*) / SUM(time_spent)) AS throughput
  FROM job_data
  GROUP BY ds
),
rolling_avg AS (
  SELECT
    ds,
    throughput,
    ROUND(AVG(throughput) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW), 6) AS rolling_7_day_avg_throughput
  FROM daily_throughput
)
SELECT * FROM rolling_avg;
```

ds	throughput	rolling_7_day_avg_throughput
2020-11-25	0.0222	0.022200
2020-11-26	0.0179	0.020050
2020-11-27	0.0096	0.016567
2020-11-28	0.0606	0.027575
2020-11-29	0.0500	0.032060
2020-11-30	0.0500	0.035050

Insight: Daily throughput varies significantly, while the 7-day rolling average provides a much smoother and reliable view. The rolling average is more effective for spotting systemic trends and evaluating team performance over time rather than reacting to daily noise.

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



```sql
SELECT * FROM rolling_avg;

SELECT
    language,
    COUNT(*) AS job_count,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM job_data WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'), 2) AS language_percentage
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY language
ORDER BY language_percentage DESC;
```

language	job_count	language_percentage
Persian	3	37.50
English	1	12.50
Arabic	1	12.50
Hindi	1	12.50
French	1	12.50
Italian	1	12.50

Insight: Persian language dominates the job language share, suggesting this is the main focus area. This data can inform team composition, especially for multilingual reviewers or support agents, and influence language prioritization strategies.

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

```sql
ORDER BY language_percentage DESC;

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

ds	job_id	actor_id	event	language	time_spent	org	duplicate_count

#	Time	Action	Message	Duration / Fetch
2	11:03:43	SELECT * FROM job_analysis.job_data LIMIT 0, 1000	8 row(s) returned	0.016 sec / 0.000 sec
3	11:22:41	SELECT ds, job_id, actor_id, event, language, time_spent, org, COUNT(*) AS duplicate_count FROM j...	0 row(s) returned	0.000 sec / 0.000 sec

Insight: The absence of duplicate rows suggests good data hygiene and no immediate issues with repeated entries. This helps ensure accuracy in performance tracking and reporting. Routine duplicate checks should still be maintained to avoid future data pollution.

Case Study 2: Investigating Metric Spike

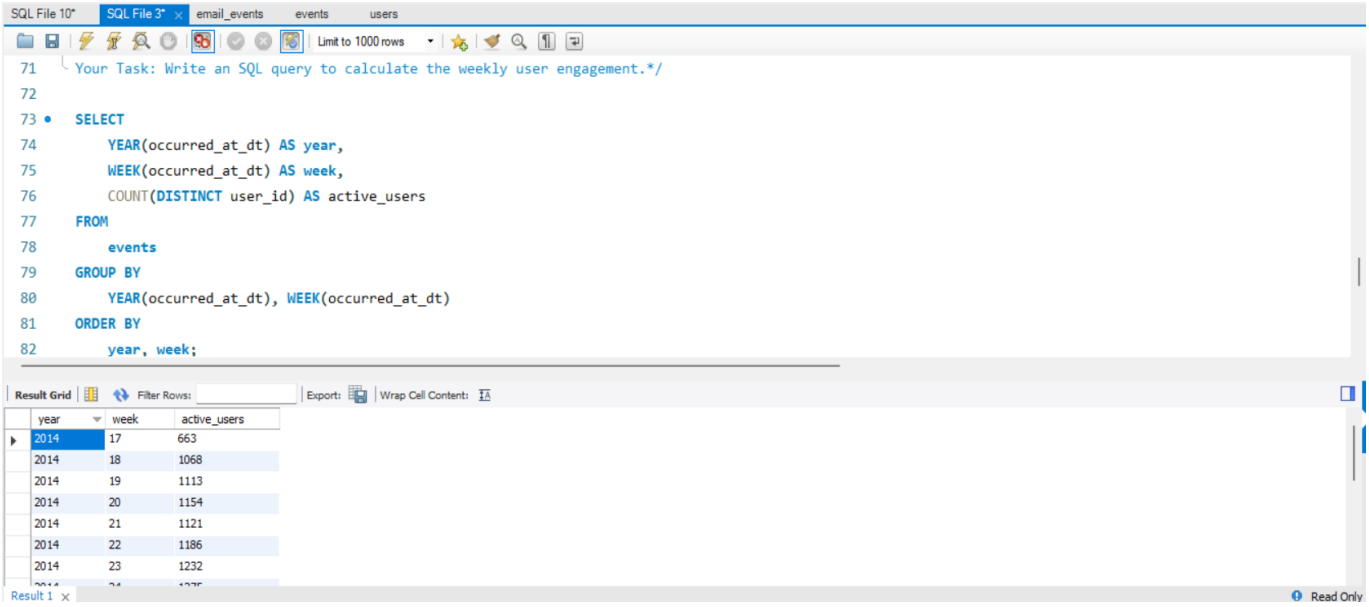
I worked with three tables:

- **users**: Contains one row per user, with descriptive information about that user's account.
- **events**: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).
- **email_events**: Contains events specific to the sending of emails.

Tasks:

A. **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.*



```sql
Your Task: Write an SQL query to calculate the weekly user engagement.*/

SELECT
    YEAR(occurred_at_dt) AS year,
    WEEK(occurred_at_dt) AS week,
    COUNT(DISTINCT user_id) AS active_users
FROM
    events
GROUP BY
    YEAR(occurred_at_dt), WEEK(occurred_at_dt)
ORDER BY
    year, week;
```

year	week	active_users
2014	17	663
2014	18	1068
2014	19	1113
2014	20	1154
2014	21	1121
2014	22	1186
2014	23	1232

Insights:

Week Range (2014)	Active Users	Observation
Week 17–20	663 → 1154	Strong early growth. Likely onboarding period or product gaining traction.
Week 21–24	1121 → 1275	Continued healthy increase. User engagement is climbing consistently.
Week 25–30	1264 → 1467	Peak engagement. Highest WAU in week 30, possibly due to campaigns, new features, or seasonal interest.
Week 31–34	1299 → 1204	Gradual decline. Still high but slightly tapering off - might indicate saturation or reduced novelty.
Week 35	**104**	Sudden drop. This is a **major red flag** - WAU crashed by over **90%** from the previous week.

B. **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*.

```
Your Task: Write an SQL query to calculate the user growth for the product.*/

SELECT
    YEAR(created_at_dt) AS year,
    WEEK(created_at_dt) AS week,
    COUNT(user_id) AS new_users
FROM users
GROUP BY YEAR(created_at_dt), WEEK(created_at_dt)
ORDER BY year, week;
```

year	week	new_users
2013	0	23
2013	1	30
2013	2	48
2013	3	36
2013	4	30
2013	5	48
2013	6	38
2013	7	42
2013	8	34
2013	9	43
2013	10	32
2013	11	31

Key Takeaways:

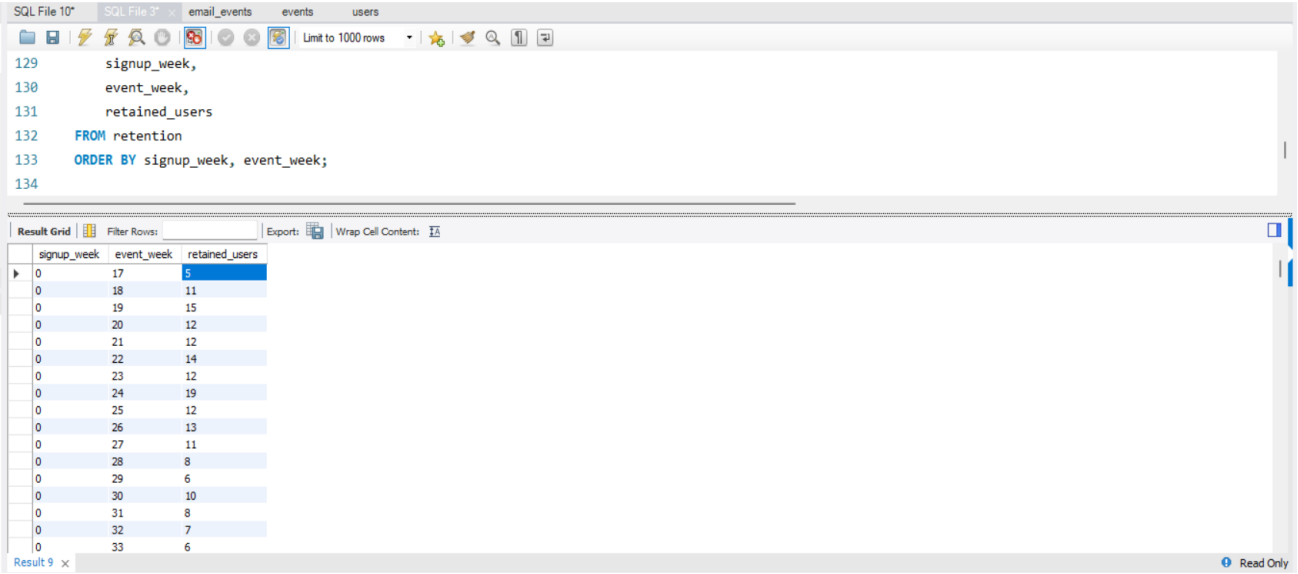
- The product demonstrated **healthy, accelerating growth** from 2013 to mid-2014.
- Growth efforts appear to have **paid off significantly** in 2014.
- Investigate anomalies (like Week 35 in 2014) further to confirm root causes.
- This growth trend would be promising for investors or stakeholders, especially if it continues beyond the available data.

C. **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.*

```sql
WITH user_signup_week AS (
    SELECT
        user_id,
        MIN(DATE(activated_at_dt)) AS signup_date,
        WEEK(MIN(DATE(activated_at_dt))) AS signup_week
    FROM users
    GROUP BY user_id
),
user_events AS (
    SELECT
        user_id,
        WEEK(DATE(occurred_at_dt)) AS event_week,
        DATE(occurred_at_dt) AS event_date
    FROM events
),
retention AS (
    SELECT
        s.signup_week,
        e.event_week,
        COUNT(DISTINCT e.user_id) AS retained_users
```

```sql
),
retention AS (
    SELECT
        s.signup_week,
        e.event_week,
        COUNT(DISTINCT e.user_id) AS retained_users
    FROM user_signup_week s
    JOIN user_events e ON s.user_id = e.user_id
    WHERE e.event_week >= s.signup_week
    GROUP BY s.signup_week, e.event_week
)
SELECT
    signup_week,
    event_week,
    retained_users
FROM retention
ORDER BY signup_week, event_week;

/*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.*/
```



```
        signup_week,
        event_week,
        retained_users
    FROM retention
    ORDER BY signup_week, event_week;
```

signup_week	event_week	retained_users
0	17	5
0	18	11
0	19	15
0	20	12
0	21	12
0	22	14
0	23	12
0	24	19
0	25	12
0	26	13
0	27	11
0	28	8
0	29	6
0	30	10
0	31	8
0	32	7
0	33	6

Insight: The retention pattern shows a typical decline over time, with fewer users staying active in later weeks. This confirms that initial user engagement is critical. By focusing efforts on retention campaigns, the drop-off curve can be flattened, improving long-term engagement.

D. **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.*

```sql
SELECT
    YEAR(occurred_at_dt) AS year,
    WEEK(occurred_at_dt) AS week,
    device,
    COUNT(DISTINCT user_id) AS weekly_active_users
FROM
    events
GROUP BY
    year, week, device
ORDER BY
    year, week, device;
```

year	week	device	weekly_active_users
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

Insight: Macbooks had higher engagement than some desktop models, signaling a stronger user preference for it. This emphasizes the importance of optimizing the Macbooks and Laptops user experience — from performance to layout - and possibly deprioritizing legacy devices with low engagement for support or testing.

E. **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*.

```sql
Your Task: Write an SQL query to calculate the email engagement metrics.
Please note that for each task, you should also provide insights and interpretations of the results obtained from your queries.*/
SELECT
    action,
    COUNT(DISTINCT user_id) AS unique_users,
    COUNT(*) AS total_events
FROM
    email_events
GROUP BY
    action
ORDER BY
    total_events DESC;
```

action	unique_users	total_events
sent_weekly_digest	4111	65205
email_open	5927	22849
email_clickthrough	5277	9862
sent_reengagement_email	3653	3653

Insight: While weekly digests are sent more frequently, open and click-through rates suggest a need for improvement in content relevance and delivery timing. The relatively low click-through rate compared to opens implies users are curious but not compelled to take action. A/B testing subject lines, optimizing CTA placement, or tailoring content based on user segments could boost these metrics.

Conclusion

The operational analytics initiative effectively found significant patterns and anomalies in the firm's job processing and user behavior data. In Case Study 1, job review analysis found intermittent activity and pointed to the prevalence of languages such as Persian in job content. Throughput analysis demonstrated the use of a 7-day rolling average over daily metrics as providing more level performance assessment, while duplicate detection validated data consistency.

In Case Study 2, user behavior and retention patterns revealed common drop-off habits, underscoring the significance of initial user interaction. Device usage habits demonstrated a stronger affinity for Macbook/Laptop devices, informing future design and development priorities. Email engagement analysis provided insights into campaign success and identified opportunities for enhancing user communication through customized approaches.

In total, this project illustrated how analytics using SQL can reveal actionable information, improve operational productivity, and enable data-driven decisions between business units.