**Operational Analytics - project-3**

---

## 1. Project Description

This project focuses on **Operational Analytics**, aiming to investigate company-wide operations using SQL. The analysis involves two case studies:

1. **Job Data Analysis** – Understanding performance and quality across job reviews.
2. **Metric Spike Investigation** – Analyzing behavioral trends and engagement metrics to detect changes in user activity.

## 2. Project Approach

- Data was provided in `.csv` files and imported into Pandas (Python) for exploratory analysis and SQL logic simulation.
- Each business question was broken into tasks, followed by SQL query simulation using DataFrame operations.
- Resulting insights were interpreted and summarized to guide strategic recommendations.

## 3. Tech Stack

- **Python (Pandas)**: For simulating SQL queries and data manipulation.
- **MySQL Workbench**: Intended SQL engine for final deployment.
- **Excel/Google Sheets**: For exporting output snapshots.
- **Google Drive**: For storing and sharing final report.

## 4. Case Study 1: Job Data Analysis

### Task 1: Jobs Reviewed Per Hour (Nov 2020)

**SQL Query:**

```
SELECT
  DATE(ds) AS review_date,
  HOUR(TIMESTAMP(ds)) AS review_hour,
  COUNT(*) AS jobs_reviewed
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY review_date, review_hour
ORDER BY review_date, review_hour;
```

**Output :**

| ds | hour | jobs_reviewed |
|---|---|---|
| 2020-11-25 | 18 | 1 |
| 2020-11-26 | 6 | 1 |
| 2020-11-27 | 20 | 1 |

**Conclusion:**
Hourly review activity was generally balanced with low concurrency. Useful for detecting peak workflow hours.

**Insights:**

- No visible clustering of job reviews; possibly auto-assigned or evenly distributed workflows.
- Recommending more structured time-window allocation for reviewer efficiency.

**Task 2: 7-Day Rolling Throughput**

## SQL Query:

```
WITH daily_stats AS (
  SELECT
    ds,
    COUNT(*) AS total_jobs,
    SUM(time_spent) AS total_time
  FROM job_data
  GROUP BY ds
)
SELECT
  ds,
  total_jobs / total_time AS throughput,
   AVG(total_jobs / total_time) OVER (ORDER BY ds ROWS BETWEEN 6
PRECEDING AND CURRENT ROW) AS rolling_avg_7d
FROM daily_stats;
```

## Output :

| ds | throughput | rolling_avg_7d |
|---|---|---|
| 2020-11-28 | 0.0606 | NULL |
| 2020-11-29 | 0.0500 | NULL |

## Conclusion:

Rolling averages smooth out anomalies, offering better trend analysis over time.

## Insights:

- Sudden throughput spikes could indicate system errors or batch task releases.

- Recommend monitoring both raw and smoothed metrics for balance.

**Task 3: Language Share (Last 30 Days)**

**SQL Query:**

```
SELECT
  language,
    ROUND(100.0 * COUNT(*) / (SELECT COUNT(*) FROM job_data
WHERE ds >= DATE_SUB(MAX(ds), INTERVAL 30 DAY)), 2) AS share
FROM job_data
WHERE ds >= DATE_SUB(MAX(ds), INTERVAL 30 DAY)
GROUP BY language;
```

**Output:**

| Language | Share (%) |
|---|---|
| Persian | 37.5 |
| English | 12.5 |
| Arabic | 12.5 |
| Hindi | 12.5 |
| French | 12.5 |

**Insights:**

- Persian dominates the review language pool. May indicate geographical focus.
- Recommending localization improvements and language-based staffing allocation.

**Task 4: Duplicate Detection**

**SQL Query:**

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

**Output:**

- **No duplicate rows detected**.

**Insights:**

- Data integrity confirmed. No remedial action needed for duplicates.

## Case Study 2: Investigating Metric Spike

### Task 1: Weekly User Engagement

**SQL Query:**

```
SELECT
  DATE_TRUNC('week', occurred_at) AS week,
  COUNT(DISTINCT user_id) AS active_users
FROM events
GROUP BY week;
```

**Output :**

| Week | Active Users |
|---|---|
| 2014-04-28 | 701 |
| 2014-05-05 | 1054 |

**Insights:**

- Consistent weekly increase in engagement; useful for retention analysis and campaign planning.

**Task 2: User Growth Over Time**

**SQL Query:**

```
SELECT
  DATE_TRUNC('week', created_at) AS week,
   COUNT(user_id) OVER (ORDER BY DATE_TRUNC('week', created_at))
AS cumulative_users
FROM users;
```

**Output :**

| **Week** | **Cumulative Users** |
|---|---|
| 2013-01-07 | 55 |
| 2013-01-14 | 102 |

**Insights:**

- Growth trajectory is healthy and incremental, signaling good acquisition strategy.

**Task 3: Weekly Retention by Cohort**

**SQL Query:**

```
SELECT
  signup_week,
  event_week,
  COUNT(DISTINCT user_id) AS retained_users
FROM (
  SELECT
    u.user_id,
    DATE_TRUNC('week', u.created_at) AS signup_week,
    DATE_TRUNC('week', e.occurred_at) AS event_week
  FROM users u
  JOIN events e ON u.user_id = e.user_id
) t
GROUP BY signup_week, event_week;
```

**Output :**

| Signup Week | Event Week | Retained Users |
|---|---|---|
| 2012-12-31 | 2014-04-28 | 2 |

**Insights:**

- Small but consistent retention across long periods. Indicates long-term value from early users.

**Task 4: Weekly Engagement Per Device**

**SQL Query:**

```
SELECT
  DATE_TRUNC('week', occurred_at) AS week,
  device,
  COUNT(DISTINCT user_id) AS active_users
FROM events
GROUP BY week, device;
```

**Output :**

| Week | Device | Active Users |
|---|---|---|
| 2014-04-28 | ASUS Chromebook | 23 |

**Insights:**

- Chromebook and mobile engagement are higher than desktop in early weeks. Recommend responsive UI focus.

**Task 5: Email Engagement**

**SQL Query:**

```
SELECT
  DATE_TRUNC('week', occurred_at) AS week,
  action,
  COUNT(*) AS count
FROM email_events
GROUP BY week, action;
```

**Output :**

| Week | Action | Count |
|---|---|---|
| 2014-04-28 | email_clickthrough | 187 |
| 2014-04-28 | sent_weekly_digest | 908 |

**Insights:**

- Digest emails are the most frequent and should be optimized for higher click-through rates.
- Re-engagement campaigns need follow-up tracking.