CASE STUDY1: JOB DATA ANALYSIS

Project Description:

The project stresses on analysis of job data for insights into the job review patterns, throughput, and language share analysis.

Approach:

- 1. Created the table job_data with columns job_id, actor_id, event, language, time_spent, org, and ds using the database case_study1.
- 2. Entries are added randomly utilizing the SOL functions.
- 3. Executed SQL queries for each task.
- 4. Reviewed obtained outputs to identify patterns.
- 5. Drew insights and interpretations from the results.

Tech-Stack Used:

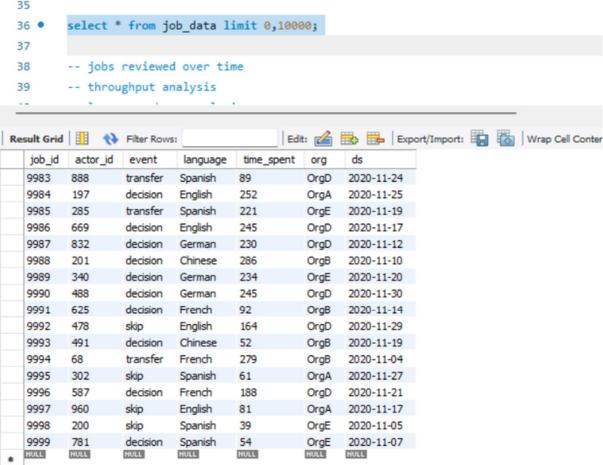
- 1. MySQL Workbench: For executing the SQL queries.
- 2. SQL: For data analysis.

Queries:

```
-- job data analysis
show databases;
create database case_study1;
use case_study1;

CREATE TABLE job_data (
    job_id INT PRIMARY KEY,
    actor_id INT,
    event VARCHAR(20),
    language VARCHAR(20),
    time_spent INT,
    org VARCHAR(50),
    ds DATE
-);
```

```
-- Insert 10,000 random rows into job_data
  INSERT INTO job_data (job_id, actor_id, event, language, time_spent, org, ds)
  SELECT
      FLOOR(1 + (RAND() * 1000)),
  ELT(FLOOR(1 + (RAND() * 3)), 'decision', 'skip', 'transfer'),
      ELT(FLOOR(1 + (RAND() * 5)), 'English', 'Spanish', 'French', 'German', 'Chinese'),
      FLOOR(1 + (RAND() * 300)),
      ELT(FLOOR(1 + (RAND() * 5)), 'OrgA', 'OrgB', 'OrgC', 'OrgD', 'OrgE'),
      DATE_ADD('2020-11-01', INTERVAL FLOOR(RAND() * 30) DAY)
⊖ FROM (
      SELECT a.N + b.N * 10 + c.N * 100 + d.N * 1000 AS n
          (SELECT 0 N UNION SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNI
          (SELECT 0 N UNION SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNI
          (SELECT 0 N UNION SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNI
          (SELECT 0 N UNION SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5 UNION SELECT 6 UNION SELECT 7 UNION SELECT 8 UNI
      LIMIT 100000
  ) t;
    35
```



```
39
       -- throughput analysis
       -- language share analysis
       -- duplicate rows detection
41
42
       -- jobs reviewed/day for nov 2020
43
       SELECT
44 •
45
           ds,
           COUNT(job_id) AS jobs_reviewed
46
47
       FROM job_data
       WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
48
       GROUP BY ds
49
       ORDER BY ds;
50
51
52
       -- 7-day rolling avg of throughput or daily metric
                                      Export: Wrap Cell Content: IA
jobs_reviewed
 2020-11-01
 2020-11-02 347
 2020-11-03
           358
 2020-11-04 339
 2020-11-05 343
 2020-11-06 332
 2020-11-07 354
 2020-11-08 336
 2020-11-09
           304
 2022 ** *0
```

```
-- 7-day rolling avg of throughput or daily metric
 53 • ⊖ WITH daily_events AS (
 54
             SELECT
 55
                  ds,
                  COUNT(*) / 86400 AS events_per_second
 56
             FROM job_data
 57
             GROUP BY ds
 58
 59
         SELECT
 60
             ds,
 61
             AVG(events_per_second) OVER (
 62
                  ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
 63
              ) AS rolling_avg_throughput
 64
         FROM daily_events;
 65
                                       Export: Wrap Cell Content: IA
Result Grid | Filter Rows:
               rolling_avg_throughput
   2020-11-01
             0.00380000
   2020-11-02 0.00390000
   2020-11-03 0.00396667
   2020-11-04 0.00395000
   2020-11-05 0.00396000
   2020-11-06 0.00393333
   2020-11-07 0.00395714
   2020-11-08 0.00397143
   2020-11-09 0.00390000
   2020-11-10 0.00392857
   2020-11-11 0.00391429
Result 26 ×
```

```
-- percentage share/language over last 30 days
8 • ○ WITH last_30_days AS (
           SELECT language, COUNT(*) AS language_count
           FROM job data
0
           WHERE ds >= DATE_SUB('2020-11-30', INTERVAL 30 DAY)
1
           GROUP BY language
2
3
       SELECT
4
           language,
5
           language_count,
6
           (language_count * 100.0) / SUM(language_count) OVER () AS 1
7
       FROM last_30 days
8
       ORDER BY language_share_percentage DESC;
9
                                    Export: Wrap Cell Content: TA
sult Grid Filter Rows:
                        language_share_percentage
          language_count
 language
French
          2036
                        20.33966
Chinese
         2020
                        20.17982
German
          1993
                        19.91009
English
         1992
                        19.90010
Spanish
         1969
                        19.67033
```

```
-- display duplicate rows
        SELECT
 82 •
            job_id, actor_id, event, language, time_spent, org, ds,
 83
           COUNT(*) AS duplicate_count
 84
        FROM job_data
 85
 86
        GROUP BY job_id, actor_id, event, language, time_spent, org, ds
 87
        HAVING COUNT(*) > 1;
        -- no duplicates in the dataset
        -- adding the duplicates
 89
        -- Insert 10 duplicate rows into job_data
 90
        -- Insert EXACT duplicates 10 times
 91
        INSERT INTO job_data (job_id, actor_id, event, language, time_spent, org, ds)
 92 •
 93
        (11001, 201, 'decision', 'English', 120, 'OrgA', '2020-11-10'),
        (11002, 202, 'skip', 'Spanish', 80, 'OrgB', '2020-11-11'),
 95
Export: Wrap Cell Content: IA
  job_id actor_id event language time_spent org ds duplicate_count
```

```
(11001, ZOI, DECISION, ENGIISH, 1ZO, OFGA, ZOZO-11-10 ),
        (11002, 202, 'skip', 'Spanish', 80, 'OrgB', '2020-11-11'),
100
        (11003, 203, 'transfer', 'French', 150, 'OrgC', '2020-11-12'),
101
        (11004, 204, 'decision', 'German', 200, 'OrgD', '2020-11-13'),
102
        (11005, 205, 'skip', 'Chinese', 90, 'OrgE', '2020-11-14');
103
104
        -- checking for duplicates again
105
106 •
        SELECT
            actor id, event, language, org, ds,
           COUNT(*) AS duplicate count
108
        FROM job_data
109
        GROUP BY actor_id, event, language, org, ds
110
        HAVING duplicate count > 1;
111
Export: Wrap Cell Content: IA
   actor_id event
                  language org ds
                                          duplicate_count
  61
          decision Spanish OrgE 2020-11-08
  958
        transfer German OrgE 2020-11-12 2
          transfer Spanish OrgE 2020-11-09 2
  902
        decision French OrgC 2020-11-01 2
  987
          transfer French OrgE 2020-11-23 2
  761
        skip English OrgC 2020-11-11 2
  451
          decision Chinese OrgC 2020-11-14 2
  98
        transfer Spanish OrgC 2020-11-23 2
          transfer French OrgE 2020-11-06 2
  686
       skip Chinese OrgA 2020-11-08 2
  286
          transfer Spanish OrgE 2020-11-20 2
  498
        decision English OrgC 2020-11-09 2
  758
  532
          skip Spanish OrgC 2020-11-20
          skip Chinese OrgA 2020-11-06 2
  701
          decision French OrgA 2020-11-18 2
  314
Result 29 ×
```

Insights:

- 1. Jobs reviewed/day: the daily review count shows job activity patterns. Spikes indicate high-priority days/ system anomalies while dips highlight holidays/outages.
- 2. Throughput analysis: the rolling average overcome daily fluctuations, hence providing long-term patterns. It is preferred over daily metrics for identifying consistent patterns whereas in case of spotting sudden spikes or dips, daily metric should be preferred.
- 3. Language share analysis: highest for French, least for Spanish.

4. Duplicate rows detection: for data inconsistency identification. There were no duplicate rows initially, so after inserting exact duplicates, checking whether the query works correctly. It did.

Results:

- 1. Developed and executed SQL queries for all tasks.
- 2. Provided insights into job review patterns, throughput trends, and language shares.
- 3. Validated duplicate detection with sample data.

CASE STUDY2: INVESTIGATING METRIC SPIKE

Project description:

This project focuses on analysing user engagement, growth, retention, and email activity to identify and investigate metric spikes. The goal is to uncover meaningful insights that can guide business decisions and product improvements. SQL queries were used to extract and analyse data from three key datasets: users, events, and email_events.

Approach:

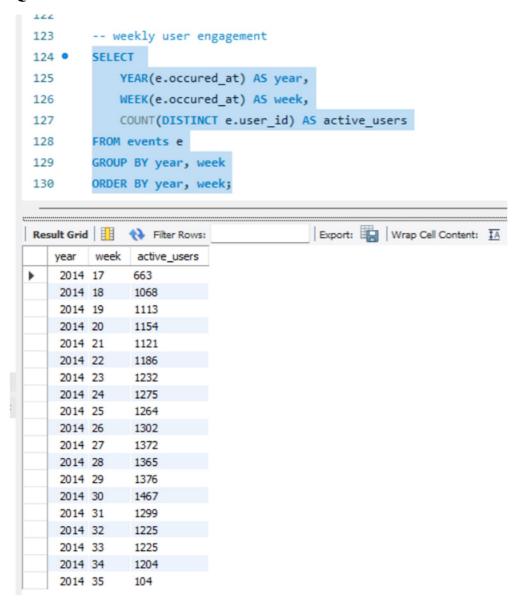
- 1. Data Exploration: Reviewed the structure of the three datasets:
 - o users: Contains user information, including signup dates.
 - events: Logs user actions, including timestamps and device information.
 - email_events: Tracks email interactions with columns for user actions and timestamps.
- 2. SQL Query Development: Constructed queries to address key metrics such as user engagement, growth, and email interaction.
- 3. Results Analysis: Evaluated outputs for trends, spikes, and anomalies.

4. Report Compilation: Documented queries, outputs, and insights.

Tech-stack used:

- 1. MySQL Workbench: For writing and executing SQL queries.
- 2. SQL: Structured Query Language for data analysis

Queries:



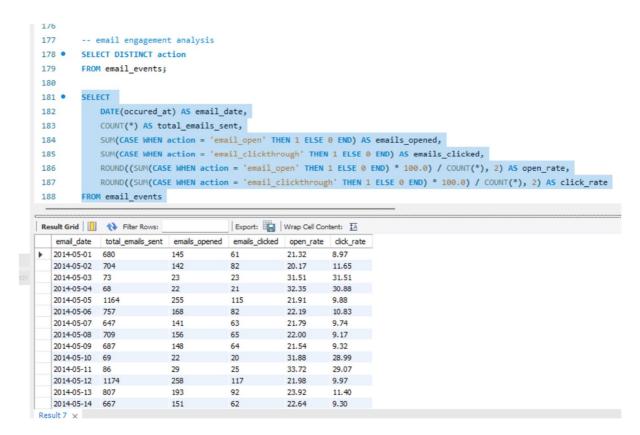
```
132
        -- user growth analysis
133 •
        SELECT
134
           YEAR(created_at) AS year,
           MONTH(created_at) AS month,
135
           COUNT(user_id) AS new_users
136
        FROM users
137
        GROUP BY year, month
138
139
        ORDER BY year, month;
                                     Export: Wr
year
       month new_users
  2013
       1
              160
  2013 2
              160
  2013 3
              150
  2013 4
             181
  2013 5
              214
  2013 6
              213
  2013 7
              284
  2013 8
              316
   2013 9
              330
  2013 10
              390
              399
  2013 11
  2013 12
              486
   2014 1
              552
  2014 2
              525
  2014 3
              615
  2014 4
              726
  2014 5
              779
  2014 6
              873
  2014 7
              997
  2014 8
              1031
Result 2 ×
```

```
-- weekly retention analysis
SELECT
         user id,
         DATE(created_at) AS signup_date
     FROM users
 ),
SELECT
         u.user_id,
         s.signup_date,
         YEAR(e.occured_at) AS year,
         WEEK(e.occured_at) AS week
     FROM events e
     JOIN users u ON e.user_id = u.user_id
      JOIN signup_cohort s ON u.user_id = s.user_id
  SELECT
     signup_date,
     year,
     week,
     COUNT(DISTINCT user_id) AS retained_users
  FROM weekly_engagement
  GROUP BY signup_date, year, week
  ORDER BY signup_date, year, week;
```

signup_date	year	week	retained_users
2013-01-01	2014	17	1
2013-01-01	2014	18	1
2013-01-01	2014	19	2
2013-01-01	2014	20	2
2013-01-01	2014	21	1
2013-01-01	2014	22	1
2013-01-01	2014	23	1
2013-01-01	2014	24	2
2013-01-01	2014	25	2
2013-01-01	2014	26	1
2013-01-01	2014	27	1
2013-01-01	2014	30	2
2013-01-01	2014	31	1
2013-01-02	2014	17	1
2013-01-02	2014	18	2
2013-01-02	2014	19	1
2013-01-02	2014	20	1
2013-01-02	2014	21	1
2013-01-02	2014	22	2
2013-01-02	2014	23	2

```
-- weekly engagement per device
167
168 •
        SELECT
           YEAR(e.occured_at) AS year,
169
           WEEK(e.occured_at) AS week,
170
           e.device,
171
            COUNT(DISTINCT e.user_id) AS active_users
172
173
        FROM events e
        GROUP BY year, week, e.device
174
        ORDER BY year, week, active_users DESC;
175
```

K	esult Gri	a 1111	Filter Rows:		export:	Wrap Cell Content:	10
	year	week	device	active_users			
•	2014	17	macbook pro	143			
	2014	17	lenovo thinkpad	86			
	2014	17	iphone 5	65			
	2014	17	macbook air	54			
	2014	17	samsung galaxy s4	52			
	2014	17	dell inspiron notebook	46			
	2014	17	iphone 5s	42			
	2014	17	nexus 5	40			
	2014	17	ipad air	27			
	2014	17	asus chromebook	21			
	2014	17	iphone 4s	21			
	2014	17	acer aspire notebook	20			
	2014	17	ipad mini	19			
	2014	17	dell inspiron desktop	18			
	2014	17	nexus 7	18			
	2014	17	nokia lumia 635	17			
	2014	17	htc one	16			
	2014	17	nexus 10	16			



Insights:

- 1. Weekly user engagement trends highlight active user patterns. Spikes may indicate successful feature launches or marketing campaigns.
- 2. Growth trends show the pace at which users join the platform. Sharp increases may point to effective onboarding strategies or viral events.
- 3. Retention metrics help identify how well the platform retains users after sign-up. Strong weekly retention suggests user satisfaction and engagement.
- 4. Device-level engagement provides insights into user preferences, helping prioritize mobile or desktop experiences.
- 5. Tracking email action rates evaluates the effectiveness of email campaigns. Low engagement may signal the need for improved content or targeting.

Results:

1. Successfully executed SQL queries to investigate user engagement, growth, retention, and email activity.

- 2. Identified potential metric spikes linked to user actions and email interactions.
- 3. Delivered actionable insights for product and marketing teams.