

PROJECT DESCRIPTION:-

I am provided with the Datasets and Tables based on dataset data I need to perform various tasks given, My goal is to use your advanced SQL skills to analyze the data and provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.

APPROACH:-

I tried to firstly understand the problem that is to be solved and to perform my task I made use of Mysql WorkBench, which gave me a great experience of running sql queries , and saving it for further analysis , while performing my task , I tried to the minimum optimal query to get the result , even if I got stuck getting desired result, I tried to debug and get the result .

Tech-Stack Used :-

MySQL Workbench 8.0.36 and MySQL Community Server 8.4.0 LTS is being used , I chose workbench to efficiently run all the queries on the single page , and easily debug ,undo and save that.

Insights:-

Case study 1:-

When I get to work on job_data I got to learn a lot of new concepts :-

- a) How to extract day from date
- b) How to apply cases within sql like we does in programming with if else
- c) How to calculate rolling-average-throughput
- d) Overall, the insights derived from this analysis can be valuable for workload management, resource allocation, and identifying opportunities for process improvement in job review operations.

Case Study 2:-

After evaluating all the tasks , I have come to gather the following insights:-

- a) We can measure the level of user activity on a weekly basis, providing insights into user engagement trends over time.
- b) Analyzing weekly user engagement metrics can help identify peak activity periods, user behavior patterns, and potential areas for improvement in product features or user experience.
- c) Evaluating email engagement metrics allows us to assess the effectiveness of email marketing campaigns and user communication strategies.

- d) By tracking metrics such as email open rates, click-through rates, and conversion rates, we can gauge user interest in email content and identify opportunities to improve email engagement.

Results:-

Case Study 1:-

A. 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.

```
• SELECT
    ds,
    day(ds) as day_of_month,
    count(*) AS total_jobs_reviewed,
    COUNT(*) / COUNT(DISTINCT SUBSTR(ds, 9, 2)) AS jobs_per_day,
    COUNT(*) / (COUNT(DISTINCT SUBSTR(ds, 9, 2)) * 24.0) AS jobs_per_hour
FROM
    job_data
WHERE
    ds LIKE '2020/11/%'
GROUP BY
    ds, day_of_month
ORDER BY
    ds, day_of_month;
```

B . 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.

```

34
35 • SELECT
36     date,
37     AVG(events_per_second) OVER (ORDER BY date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_avg_th
38 FROM (
39     SELECT
40         DATE(ds) AS date,
41         COUNT(*) / 86400.0 AS events_per_second -- Assuming 86400 seconds in a day
42     FROM
43         job_data
44     GROUP BY
45         date
46 ) AS daily_throughput
47 ORDER BY
48     date;

```

Result Grid

date	rolling_avg_throughput
NULL	0.00010000

Form Editor

C . 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.

```

49
50 • SELECT language, COUNT(language) / 100.0 AS percentage_share
51 FROM job_data
52 WHERE ds >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
53 GROUP BY language;
54

```

D. 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.

```

54
55 • SELECT *
56 FROM job_data
57 GROUP BY job_id, actor_id, event, language, time_spent, org, ds
58 HAVING COUNT(*) > 1;
59

```

Case Study 2:-

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.

```

74 • SELECT
75     YEARWEEK(activated_at) AS week,
76     COUNT(DISTINCT user_id) AS active_users
77 FROM
78     users
79 WHERE
80     activated_at IS NOT NULL
81 GROUP BY
82     YEARWEEK(activated_at)
83 ORDER BY
84     week;

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
week	active_users			
9381	9381			

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.

```

189
190 • SELECT
191     device,
192     COUNT(user_id) as number_of_users
193 FROM
194     events
195 GROUP BY
196     device
197 ORDER BY
198     number_of_users DESC;

```

device	number_of_users
macbook pro	114590
lenovo thinkpad	73956
macbook air	53572
iphone 5	51766
dell inspiron notebook	39338
samsung galaxy s4	37306
nexus 5	33004
iphone 5s	31858
dell inspiron desktop	20282
iphone 4s	19230
asus chromebook	19084
ipad air	18938
acer aspire notebook	17860
hp pavilion desktop	17762
nexus 7	13080
nokia lumia 635	11224
ipad mini	11182
acer aspire desktop	10346

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

```

157 • SELECT
158     cohort_week,
159     week_offset,
160     COUNT(DISTINCT user_id) AS active_users
161 FROM
162     (
163         SELECT
164             cohorts.cohort_week,
165             WEEK(u.occured_at) - WEEK(cohorts.cohort_occured_at) AS week_offset,
166             u.user_id
167         FROM
168             (
169                 SELECT
170                     user_id,
171                     WEEK(occured_at) AS cohort_week,
172                     MIN(occured_at) AS cohort_occured_at
173                 FROM
174                     events
175                 WHERE
176                     event_name = 'login' -- Assuming 'signup' event marks user sign-up
177                 GROUP BY
178                     user_id
179             ) AS cohorts
180         LEFT JOIN events u ON cohorts.user_id = u.user_id
181     ) AS subquery
182 GROUP BY
183     cohort_week, week_offset
184 ORDER BY
185     cohort_week, week_offset
186 LIMIT 0, 1000;
187

```

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.

```

121 • SELECT
122     e.device,
123     WEEK(e.occured_at) AS week,
124     COUNT(DISTINCT u.user_id) AS active_users
125 FROM
126     events e
127 INNER JOIN
128     users u ON e.user_id = u.user_id
129 GROUP BY
130     e.device, week
131 ORDER BY
132     e.device, week;

```

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

device	week	active_users
acer aspire desktop	198	
acer aspire notebook	338	
amazon fire phone	89	
asus chromebook	355	
dell inspiron desktop	360	
dell inspiron notebook	676	
hp pavilion desktop	339	
htc one	196	
ipad air	478	
ipad mini	292	
iphone 4s	409	
iphone 5	1025	
iphone 5s	626	
kindle fire	205	
lenovo thinkpad	1309	

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.

- ```
SELECT
 COUNT(*) AS total_emails_sent
FROM
 email_events
WHERE
 action = 'sent_weekly_digest';
```
- ```
SELECT
    COUNT(*) AS total_emails_clicked
FROM
    email_events
WHERE
    action = 'email_clickthrough';
```
- ```
SELECT
 COUNT(*) AS total_emails_clicked
FROM
 email_events
WHERE
 action = 'email_open';
```