Operation Analytics and Investigating Metric Spike

Project Description:
Case Study 1 (Job Data):

This project focuses on analyzing a job_data table to answer specific questions using SQL fundamentals. The table contains information about jobs, including job_id, actor_id, event, language, time_spent, org, and ds columns. The questions include calculating the number of jobs reviewed per hour per day, analyzing throughput, determining the percentage share of each language, and identifying duplicate rows.

Case Study 2 (Investigating Metric Spike):

This project involves analyzing three tables: users, events, and email_events. The goal is to calculate metrics related to user engagement, user growth, weekly retention, weekly engagement per device, and email engagement. SQL fundamentals will be used to extract insights from the dataset and provide valuable information for marketing campaigns and decision-making.

Approach:
As an individual working on this project, I followed a structured approach to analyze the job_data table. I began by understanding the table structure and column definitions. Using SQL queries and functions, I calculated the number of jobs reviewed per hour per day for November 2020, analyzed throughput by calculating events per second and 7-day rolling average, determined the percentage share of each language in the last 30 days, and identified duplicate rows. I prioritized data accuracy, optimized queries for efficiency, and maintained documentation of my workflow. The project aimed to provide valuable insights for marketing and investor assessments, achieved through the successful application of SQL fundamentals.

Tech-Stack Used:
For this project, I utilized MySQL Workbench 8.0 as the primary software tool. MySQL Workbench is an integrated development environment (IDE) for MySQL databases, providing a graphical interface for designing, querying, and managing databases.

Insights:

Case Study 1(Job Data):

Number of jobs reviewed: The number of jobs reviewed per day per hour indicates the level of activity in job review processes.

To calculate the number of jobs reviewed per hour per day for November 2020, I utilized SQL queries to filter the data for the specified time period and grouped the results by hour and day.

Query:

```
SELECT DATE(ds) AS date,
      HOUR(ds) AS hour,
      COUNT(*) AS jobs_reviewed
FROM sheet1
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY DATE(ds), HOUR(ds)
ORDER BY DATE(ds), HOUR(ds);
```

Output:

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

date	hour	jobs_reviewed
2020-11-01	0	3
2020-11-01	1	3
2020-11-01	2	6
2020-11-01	3	3
2020-11-01	4	2
2020-11-01	5	3
2020-11-01	6	6
2020- 2020-11-01		3
2020-11-01	8	7
2020-11-01	9	5
2020-11-01	10	1
2020-11-01	11	3
2020-11-01	12	5
2020-11-01	13	1
2020-11-01	14	3
2020-11-01	15	2
2020-11-01	16	5
2020-11-01	17	6
2020-11-01	18	2
2020-11-01	19	5
2020-11-01	20	3
2020-11-01	21	3
2020-11-01	22	1
2020-11-01	23	4
2020-11-02	0	3
2020-11-02	1	3
2020-11-02	2	1
2020-11-02	3	2
2020-11-02	4	2

Result 47

Output

Action Output

#	Time	Action	Message
1	21:38:07	SELECT DATE(ds) AS date, HOUR(ds) AS hour, COUNT(*) AS jobs_reviewed F...	680 row(s) returned

Throughput: Throughput refers to the number of events happening per second, representing the system's processing capacity.
To calculate the 7-day rolling average of throughput, I used SQL queries to aggregate the events per second and then calculated the average over a rolling window of 7 days. This helps identify any trends or variations in the throughput metric. I prefer the 7-day rolling average because it provides a smoother representation of the metric, reducing the impact of daily fluctuations and offering a more comprehensive view of the system's performance.
Query:

```
SELECT date,
      total_events,
      total_events / (24 * 60 * 60) AS throughput,
      AVG(total_events / (24 * 60 * 60)) OVER (ORDER BY date ROWS BETWEEN
6 PRECEDING AND CURRENT ROW) AS rolling_average
FROM (
   SELECT DATE(ds) AS date, COUNT(*) AS total_events
   FROM sheet1
   WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
   GROUP BY DATE(ds)
) AS subquery
GROUP BY date, total_events
ORDER BY date;
```

Output:

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

date	total_events	throughput	rolling_average
2020-11-01	85	0.0010	0.00100000
2020-11-02	85	0.0010	0.00100000
2020-11-03	80	0.0009	0.00096667
2020-11-04	82	0.0009	0.00095000
2020-11-05	84	0.0010	0.00096000
2020-11-06	80	0.0009	0.00095000
2020-11-07	76	0.0009	0.00094286
2020-11-08	85	0.0010	0.00094286
2020-11-09	66	0.0008	0.00091429
2020-11-10	73	0.0008	0.00090000
2020-11-11	81	0.0009	0.00090000
2020-11-12	94	0.0011	0.00091429
2020-11-13	79	0.0009	0.00091429
2020-11-14	74	0.0009	0.00091429
2020-11-15	69	0.0008	0.00088571

Result 48

Output

Action Output

#	Time	Action	Message
1	21:39:18	SELECT date, total_events, total_events / (24 * 60 * 60) AS throughput, AV...	30 row(s) returned

Percentage share of each language: The percentage share of each language in different contents provides insights into language preferences and content distribution. To calculate the percentage share of each language in the last 30 days, I used SQL queries to filter the data for the specified time period and performed calculations to determine the language distribution. By dividing the count of each language by the total count of contents, I obtained the percentage share for each language.

Query:

```
SELECT language,
    COUNT(*) AS job_count,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM sheet1 WHERE ds
BETWEEN '2020-11-01' AND '2020-11-30'), 2) AS percentage_share
FROM sheet1
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY language
ORDER BY percentage_share DESC;
```

Output:

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

language	job_count	percentage_share
Hindi	505	21.43
English	471	19.99
French	461	19.57
Italian	459	19.48
Spanish	456	19.35
Persian	3	0.13
Arabic	1	0.04

Result 50

Output

Action Output

#	Time	Action	Message
1	21:41:16	SELECT language, COUNT(*) AS job_count, ROUND(COUNT(*) * 100.0 / (SELE...	7 row(s) returned

Duplicate rows: Duplicate rows refer to rows in the dataset that have the same values. If actor_id and job_id are the same for an entry, then that entry is considered to be duplicate. Comparing the values across columns, I can identify rows with identical values and retrieve them from the table.

Query:

```
SELECT *
FROM sheet1
WHERE (actor_id, job_id) IN (
   SELECT actor_id, job_id
   FROM sheet1
   GROUP BY actor_id, job_id
   HAVING COUNT(*) > 1
)
ORDER BY actor_id, job_id;
```

Output:

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

ds	job_id	actor_id	event	language	time_spent	org
2020-11-16 09:05:49	1	1001	transfer	Hindi	99	C
2020-10-31 21:48:40	1	1001	transfer	Italian	38	B
2020-11-24 08:26:30	4	1001	transfer	French	9	A
2020-11-07 22:37:33	4	1001	transfer	Spanish	70	C
2020-11-22 08:22:05	4	1001	transfer	French	78	A
2020-11-07 19:11:48	4	1001	transfer	Hindi	24	A
2020-11-24 08:33:43	4	1001	transfer	Spanish	56	D
2020-11-23 03:38:57	4	1001	transfer	Hindi	32	B
2020-11-10 13:45:34	5	1001	decision	Italian	92	D
2020-10-25 05:51:09	5	1001	transfer	Spanish	61	A
2020-11-09 23:14:20	5	1001	decision	Italian	10	C
2020-11-28 13:30:15	6	1001	decision	French	21	B
2020-11-29 08:34:53	6	1001	decision	English	43	D
2020-10-31 16:27:08	7	1001	decision	English	5	A
2020-11-16 08:20:36	7	1001	decision	English	79	D
2020-10-30 06:10:30	7	1001	decision	English	29	A
2020-10-31 05:40:32	7	1001	decision	English	62	C
2020-10-17 11:52:58	8	1001	decision	Hindi	78	C
2020-10-19 01:59:57	8	1001	skip	French	46	D
2020-11-02 07:24:45	8	1001	skip	French	91	D
2020-11-04 04:11:43	8	1001	skip	Spanish	58	A
2020-11-20 02:05:42	9	1001	skip	Spanish	73	C
2020-11-19 18:40:48	9	1001	skip	English	76	C
2020-11-04 11:44:47	9	1001	skip	English	11	A
2020-11-05 09:25:32	9	1001	skip	Italian	90	D
2020-11-23 03:37:42	10	1001	skip	French	36	A
2020-11-09 05:20:07	10	1001	skip	Spanish	90	D

sheet1 51

Output

Action Output

#	Time	Action	Message
1	21:46:06	SELECT * FROM sheet1 WHERE (actor_id, job_id) IN (SELECT actor_id, job_id FRO...	3404 row(s) returned

Case Study 2 (Investigating Metric Spike):

User Engagement: User engagement is a measure of how active users are and indicates their satisfaction with a product or service.

To calculate the weekly user engagement, I utilized SQL queries to analyze unique user engagement events within the specified week.

Query:

```
SELECT WEEK(occurred_at) AS week, COUNT(DISTINCT user_id) AS
user_engagement
FROM events
WHERE event_type = 'engagement'
GROUP BY week;
```

Output:

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

week	user_engagement
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

1121

Result 12

Output

Action Output

#	Time	Action	Message
1	21:51:51	SELECT WEEK(occurred_at) AS week, COUNT(DISTINCT user_id) AS user_engagement ...	19 row(s) returned

User Growth: User growth measures the increase in the number of users over a specific period, reflecting the product's adoption and popularity.
To calculate user growth for a product, I used SQL queries to track the number of new users added over time. By comparing the count of users added in different weeks, we can identify growth of product.
Query:

```
SELECT WEEK(created_at) AS week, COUNT(DISTINCT user_id) AS user_growth
FROM user
GROUP BY week;
```

Output:

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

week	user_growth
0	197
1	300
2	299
3	325
4	322
5	341
6	344
7	353
8	350
9	353
10	377
11	382
12	391
13	396
14	411
15	395

Result 14

Output

Action Output

#	Time	Action	Message
1	21:57:53	SELECT WEEK(created_at) AS week, COUNT(DISTINCT user_id) AS user_growth FROM...	53 row(s) returned

Weekly Retention: Weekly retention evaluates the percentage of users who continue to use a product or service after signing up, indicating its ability to retain users.
To calculate the weekly retention of users, I employed SQL queries to track the state of users and identify users who remained active in consecutive weeks. By comparing the count of retained users to the initial sign-up cohort, I determined the retention rate for each week.
Query:

```
SELECT WEEK(created_at) AS week, COUNT(DISTINCT user_id) AS
weekly_retention
FROM user
WHERE state = 'active'
GROUP BY week;
```

Output:

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

week	weekly_retention
0	106
1	156
2	157
3	149
4	160
5	181
6	173
7	167
8	163
9	176
10	186
11	161
12	181
13	206
14	197
15	207

Result 16

Output

Action Output

#	Time	Action	Message
1	22:00:09	SELECT WEEK(created_at) AS week, COUNT(DISTINCT user_id) AS weekly_retention F...	53 row(s) returned

Weekly Engagement: Weekly engagement measures the level of user activity and satisfaction with a product or service on a weekly basis.
To calculate the weekly engagement per device, I utilized SQL queries to analyze user interactions and activities categorized by device type. By selecting distinct users and their devices and grouping by week and device, I assessed the level of engagement for each device category.
Query:

```
SELECT WEEK(occurred_at) AS week, device, COUNT(DISTINCT user_id) AS weekly_engagement
FROM events
GROUP BY week, device;
```

Output:

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

week	device	weekly_engagement
17	acer aspire desktop	12
17	acer aspire notebook	23
17	amazon fire phone	4
17	asus chromebook	23
17	dell inspiron desktop	20
17	dell inspiron notebook	48
17	hp pavilion desktop	17
17	htc one	19
17	ipad air	29
17	ipad mini	19
17	iphone 4s	27
17	iphone 5	69
17	iphone 5s	47
17	kindle fire	6
17	lenovo thinkpad	94
17	mac mini	7

Result 17

Output

Action Output

#	Time	Action	Message
1	22:01:12	SELECT WEEK(occurred_at) AS week, device, COUNT(DISTINCT user_id) AS weekly_e...	493 row(s) returned

Email Engagement: Email engagement reflects user involvement and interaction with the email service.

To calculate email engagement metrics, I used SQL queries to analyze user email activities such as opens, clicks, and responses.

Query:

```
SELECT action, COUNT(DISTINCT user_id) AS email_engagement
FROM email_events
GROUP BY action;
```

Output:

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

action	email_engagement
email_clickthrough	5277
email_open	5927
sent_reengagement_email	3653
sent_weekly_digest	4111

Result 20

Output

Action Output

#	Time	Action	Message
1	22:08:35	SELECT action, COUNT(DISTINCT user_id) AS email_engagement FROM email_events G...	4 row(s) returned

Results:

While working on this project, I have gained a better understanding of user analytics and SQL fundamentals. By analyzing Operation Analytics and Investigating Metric Spike, I was able to provide insights on various aspects such as number of jobs analyzed per hour per day, 7 day rolling average of throughput, percentage share of each language, user engagement, user growth, weekly retention, weekly engagement per device, and email engagement.

This project has helped me enhance my SQL skills, particularly in querying and manipulating data to derive meaningful insights. It has also improved my ability to interpret data and provide actionable recommendations based on the analysis. Overall, this project has deepened my understanding of user behavior analysis and its application in making informed decisions for product development and marketing strategies.