TRAINITY PROJECT

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE USING ADVANCED SQL

PROJECT DESCRIPTION

- In this project, we will explore a real-world scenario where advanced SQL techniques are used to perform operation analytics and investigate metric spikes.
- We will work with a sample dataset to analyze operational data and identify the causes of sudden spikes in a specific metric.

APPROACH

1. Data Understanding and Exploration:

Begin by loading the dataset into your SQL environment and understanding its schema

2. Data Preprocessing and Cleaning:

Handle missing values, duplicates, and outliers in the dataset. Convert data types as needed, especially the timestamp column.

3. Metric Selection:

Choose a metric that you want to investigate for potential spikes. It could be a performance indicator, user activity, or any other relevant metric.

4. Metric Trend Analysis:

Write SQL queries to aggregate and summarize the chosen metric over different time intervals (e.g., daily, weekly).

INSIGHTS

- My experience working on the project gave me valuable insights into how advanced SQL techniques can be used to effectively extract insights from the database.
- Through the use of advanced SQL, I was able to conduct operational analytics and investigate metric spikes, enabling me to identify trends and patterns in the data.

TECH-STACK USED

Mysql workbench

Case Study 1 (Job Data)

1A)Calculate the number of jobs reviewed per hour for each day for November 2020?

SQL QUERY:

```
SELECT
    ds,
  (COUNT(job_id) /
    SUM(time_spent))*3600   as job_reviewed_per_hour
FROM job_data
group by ds;
```

QUERY OUTPUT:

ds	job_reviewed_per_hour
11/30/2020	180.0000
11/29/2020	180.0000
11/28/2020	218.1818
11/27/2020	34.6154
11/26/2020	64.2857
11/25/2020	80.0000

1B)Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why

SQL QUERY:

```
WITH grp AS (SELECT ds, COUNT(job_id) AS num_of_jobs, SUM(time_spent) AS total_time_spent
FROM job_data GROUP BY ds) SELECT ds AS DATE , ROUND(1.0*SUM(num_jobs)
OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) /
SUM(total_time) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW),2) AS Rolling_average_for_7days_throughput FROM grp;
```

QUERY OUTPUT:

DATE	Rolling_average_for_7days_throughput
11/25/2020	0.02
11/26/2020	0.02
11/27/2020	0.01
11/28/2020	0.02
11/29/2020	0.02
11/30/2020	0.03

INSIGHT OF THE QUERY:

- In MySQL, daily metrics and rolling average are both useful in analyzing system performance and making decisions about resource allocation and optimization.
- Daily metrics provide a snapshot of performance over a fixed time period, while rolling average offers a more flexible and up-to-date view of trends over a sliding time window.
- The preferred approach depends on the specific use case and the insights required from the data.

1C)Calculate the percentage share of each language in the last 30 days.

SQL QUERY:

```
SELECT
    language AS Languages,
    ROUND(100 * COUNT(*) / (SELECT
                    total
                FROM
                    (SELECT
                        COUNT(*) AS total
                    FROM
                        job_data) AS tb),
            2) AS Percentage
FROM
    job_data
```

QUERY OUTPUT:

	Languages	Percentage
▶	english	12.50
	arabic	12.50
	persian	37.50
	hindi	12.50
	french	12.50
	italian	12.50

1D)Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

SQL QUERY:

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

QUERY OUTPUT:

job_id	actor_id	event	language	time_spent	org	ds

INSIGHT OF THE QUERY:

There are no duplicate in this table

Case Study 2: Investigating Metric Spike

2A) Write an SQL query to calculate the weekly user engagement

```sql
SELECT WEEK(occurred_at) AS Week,COUNT(user_id) AS users FROM events
WHERE event_type='engagement' GROUP BY 1 ORDER BY 1;
```

QUERY OUTPUT:

Week	users
17	8019
18	17341
19	17224
20	17911
21	17151
22	18413
23	18280
24	19052
25	18642
26	19061
27	19881
28	20776
29	20067
30	21533
31	18556
32	16612
33	16145

2B)Write an SQL query to calculate the user growth for the product.?

SQL QUERY:

```
    WITH CTE AS (SELECT MONTH(created_at) as MONTH,COUNT(id) AS USERS
FROM users GROUP BY 1)SELECT MONTH,USERS,ROUND((USERS/LAG(USERS,1)
OVER (ORDER BY MONTH)-1)*100,2) AS  Growth_Percent FROM CTE;
```

QUERY OUTPUT:

MONTH	USERS	Growth_Percent
1	9	NULL
2	10	11.11
3	7	-30.00
4	8	14.29
5	9	12.50
6	8	-11.11
7	9	12.50
8	11	22.22
9	6	-45.45
10	12	100.00

2C)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.

```
SELECT   first_week,
SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS Week_0,
 SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS Week_1,
SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS Week_2,
SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS Week_3,
SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS Week_4,
SUM(CASE WHEN week_number = 5 THEN 1 ELSE 0 END) AS Week_5,
SUM(CASE WHEN week_number = 6 THEN 1 ELSE 0 END) AS Week_6,
SUM(CASE WHEN week_number = 7 THEN 1 ELSE 0 END) AS Week_7,
SUM(CASE WHEN week_number = 8 THEN 1 ELSE 0 END) AS Week_8,
SUM(CASE WHEN week_number = 9 THEN 1 ELSE 0 END) AS Week_9,
SUM(CASE WHEN week_number = 10 THEN 1 ELSE 0 END) AS Week_10,
 SUM(CASE WHEN week_number = 11 THEN 1 ELSE 0 END) AS Week_11,
SUM(CASE WHEN week_number = 12 THEN 1 ELSE 0 END) AS Week_12,
SUM(CASE WHEN week_number = 13 THEN 1 ELSE 0 END) AS Week_13,
SUM(CASE WHEN week_number = 14 THEN 1 ELSE 0 END) AS Week_14,
SUM(CASE WHEN week_number = 15 THEN 1 ELSE 0 END) AS Week_15,
SUM(CASE WHEN week_number = 16 THEN 1 ELSE 0 END) AS Week_16,
SUM(CASE WHEN week_number = 17 THEN 1 ELSE 0 END) AS Week_17,
SUM(CASE WHEN week_number = 18 THEN 1 ELSE 0 END) AS Week_18,
SUM(CASE WHEN week_number = 19 THEN 1 ELSE 0 END) AS Week_19,
```

```
FROM (SELECT a.user_id,a.login_week,b.first_week as
first_week,a.login_week-first_week as week_number FROM
(SELECT
user_id,week(occurred_at) AS login_week FROM events
GROUP BYuser_id,week(occurred_at)) a,(SELECT user_id,
min(week(occurred_at)) AS weeks
FROM events GROUP BY user_id) b where a.user_id=b.user_id)
as with_week_number
group by weeks order by week;
```

first_week	Week_0	Week_1	Week_2	Week_3	Week_4	Week_5	Week_6	Week_7	Week_8	Week_9	Week_10	Week_11	Week_12	Week_13	Week_14	Week_15	Week_16	Week_17	Week_18	Week_19
17	663	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77	5	0
18	596	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4	0	0
19	427	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0	0	0
20	358	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0	0	0
21	317	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0	0	0
22	326	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0	0
23	328	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0	0
24	339	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0	0
25	305	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0	0
26	288	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0	0
27	292	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0	0
28	274	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0	0
29	270	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0	0
30	294	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	215	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	267	188	94	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	286	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	279	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2D)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.

```
SELECT WEEK(occurred_at) AS WEEK_NUM,
COUNT(DISTINCT CASE WHEN device IN('dell inspiron notebook')THEN user_id
ELSE NULL END) AS "Dell inspiron notebook",
COUNT(DISTINCT CASE WHEN device IN('iphone 5')THEN user_id ELSE NULL
END) AS "Iphone 5",
COUNT(DISTINCT CASE WHEN device IN('iphone 4s')THEN user_id ELSE NULL
END) AS "Iphone 4s",
COUNT(DISTINCT CASE WHEN device IN('windows surface')THEN user_id
ELSE NULL END) AS "Windows surface",
COUNT(DISTINCT CASE WHEN device IN('macbook air')THEN user_id ELSE
NULL END) AS "Macbook air",
COUNT(DISTINCT CASE WHEN device IN('iphone 5s')THEN user_id ELSE
NULL END) AS "Iphone 5s",
COUNT(DISTINCT CASE WHEN device IN('macbook pro')THEN user_id ELSE
NULL END) AS "Macbook pro",
COUNT(DISTINCT CASE WHEN device IN('kindle fire')THEN user_id ELSE
NULL END) AS "Kindle fire",
COUNT(DISTINCT CASE WHEN device IN('ipad mini')THEN user_id ELSE
NULL END) AS "Ipad mini",
COUNT(DISTINCT CASE WHEN device IN('nexus 7')THEN user_id ELSE NULL
END) AS "Nexus 7",
COUNT(DISTINCT CASE WHEN device IN('nexus 5')THEN user_id ELSE NULL
END) AS "Nexus 5",
```

```
COUNT(DISTINCT CASE WHEN device IN('samsung galaxy s4')THEN user_id
ELSE NULL END) AS "Samsung galaxy s4",
COUNT(DISTINCT CASE WHEN device IN('lenovo thinkpad')THEN user_id
ELSE NULL END) AS "Lenovo thinkpad",
COUNT(DISTINCT CASE WHEN device IN('samsung galaxy tablet')THEN user_id
ELSE NULL END) AS "Samsung galaxy tablet",
COUNT(DISTINCT CASE WHEN device IN('acer aspire notebook')THEN user_id
ELSE NULL END) AS "Acer aspire notebook",
COUNT(DISTINCT CASE WHEN device IN('asus chromebook')THEN user_id
ELSE NULL END) AS "Asus chromebook",
COUNT(DISTINCT CASE WHEN device IN('htc one')THEN user_id ELSE NULL
END) AS "Htc one",

COUNT(DISTINCT CASE WHEN device IN('nokia lumia

635')THEN user_id ELSE NULL END) AS "Nokia lumia 635",
```

```
COUNT(DISTINCT CASE WHEN device IN('samsung galaxy
note')THEN user_id ELSE NULL END) AS "Samsung galaxy note",
COUNT(DISTINCT CASE WHEN device IN('acer aspire
desktop')THEN user_id ELSE NULL END) AS "Acer aspire desktop",
COUNT(DISTINCT CASE WHEN device IN('mac mini')THEN
user_id ELSE NULL END) AS "Mac mini",
COUNT(DISTINCT CASE WHEN device IN('hp pavilion
desktop')THEN user_id ELSE NULL END) AS "Hp pavilion desktop",
COUNT(DISTINCT CASE WHEN device IN('Dell inspiron
desktop')THEN user_id ELSE NULL END) AS "Dell inspiron desktop",
COUNT(DISTINCT CASE WHEN device IN('ipad air')THEN user_id
ELSE NULL END) AS "Ipad air",
COUNT(DISTINCT CASE WHEN device IN('amazon fire
phone')THEN user_id ELSE NULL END) AS "Amazon fire phone",
COUNT(DISTINCT CASE WHEN device IN('nexus 10')THEN
user_id ELSE NULL END) AS "Nexus 10"
FROM events WHERE event_type='engagement' GROUP BY 1 ORDER BY 1;
```

WEEK_NUM	Dell inspiron notebook	Iphone 5	Iphone 4s	Windows surface	Macbook air	Iphone 5s	Macbook pro	Kindle fire	Ipad mini	Nexus 7	Nexus 5	Samsung galaxy s4	Lenovo thinkpad	Samsung galaxy tablet	Acer aspire notebook
17	46	65	21	10	54	42	143	6	19	18	40	52	86	0	20
18	77	113	46	10	121	73	252	27	30	30	73	82	153	0	33
19	83	115	44	16	112	79	266	21	36	41	87	91	178	0	41
20	84	125	55	21	119	79	256	23	32	32	103	93	173	0	40
21	80	137	45	17	110	74	247	30	23	29	91	84	167	0	47
22	92	125	45	15	145	71	251	21	34	45	96	105	176	0	41
23	103	152	53	14	124	79	266	25	33	36	88	99	176	0	43
24	99	142	53	22	152	79	255	25	39	49	87	101	165	0	40
25	105	137	40	22	121	78	275	24	30	51	89	99	197	0	47
26	89	152	50	21	134	94	269	26	43	46	87	112	192	0	35
27	89	163	67	33	142	83	302	25	35	40	84	116	202	0	49
28	103	151	61	33	148	93	295	31	35	39	85	122	220	0	49
29	113	144	60	28	148	90	295	37	34	45	77	123	209	0	53
30	127	152	65	19	159	103	322	25	35	62	84	103	206	0	60
31	113	135	56	19	147	71	321	14	27	38	69	100	207	0	55
32	104	119	34	10	125	67	307	12	30	25	67	82	179	0	55
33	110	110	35	15	133	65	312	14	28	30	70	80	191	0	46
34	105	101	50	18	136	70	292	13	25	33	70	90	193	0	63
35	9	2	6	3	10	3	17	3	2	2	4	6	16	0	3

Asus chromebook	Htc one	Nokia lumia 635	Samsung galaxy note	Acer aspire desktop	Mac mini	Hp pavilion desktop	Dell inspiron desktop	Ipad air	Amazon fire phone	Nexus 10
21	16	0	0	0	6	0	0	27	0	16
42	19	0	0	0	13	0	0	52	0	30
27	30	0	0	0	18	0	0	55	0	25
41	29	0	0	0	26	0	0	59	0	22
38	21	0	0	0	18	0	0	51	0	25
52	24	0	0	0	25	0	0	58	0	27
49	20	0	0	0	18	0	0	41	0	45
43	20	0	0	0	29	0	0	57	0	38
38	21	0	0	0	21	0	0	57	0	29
49	23	0	0	0	11	0	0	56	0	29
52	27	0	0	0	15	0	0	55	0	37
50	26	0	0	0	28	0	0	54	0	26
49	31	0	0	0	31	0	0	52	0	25
56	31	0	0	0	23	0	0	70	0	36
56	13	0	0	0	24	0	0	55	0	24
62	18	0	0	0	20	0	0	48	0	30
49	19	0	0	0	32	0	0	40	0	23
47	25	0	0	0	30	0	0	39	0	25
6	2	0	0	0	2	0	0	0	0	2

2E) Calculate the email engagement metrics?

SQL QUERY:

```
SELECT WEEK(occurred_at) AS Week,
COUNT(DISTINCT CASE WHEN action IN('sent_weekly_digest')THEN user_id ELSE
NULL END) AS "Sent weekly digest",
COUNT(DISTINCT CASE WHEN action IN('email_open')THEN user_id ELSE NULL
END) AS "Email open",
COUNT(DISTINCT CASE WHEN action IN('email_clickthrough')THEN user_id ELSE
NULL END) AS "Email clickthrough",
COUNT(DISTINCT CASE WHEN action IN('sent_reengagement_email')THEN user_id
ELSE NULL END) AS "Sent reengagement email"
FROM email_events GROUP BY 1 ORDER BY 1;
```

QUERY OUTPUT:

Week	Sent weekly digest	Email open	Email clickthrough	Sent reengagement email
17	908	310	166	73
18	2602	900	425	157
19	2665	961	476	173
20	2733	989	501	191
21	2822	996	436	164
22	2911	965	478	192
23	3003	1057	529	197
24	3105	1136	549	226
25	3207	1084	524	196
26	3302	1149	550	219
27	3399	1207	613	213
28	3499	1228	594	213
29	3592	1201	583	213
30	3706	1363	625	231
31	3793	1338	444	222
32	3897	1318	416	200
33	4012	1417	490	264
34	4111	1502	481	261
35	0	41	38	48

RESULT:

Operation analytics and investigating metrics involve using advanced SQL queries to analyze data spikes, identify patterns, and uncover insights. This process helps businesses understand sudden changes in their key metrics, enabling informed decision-making and proactive responses to anomalies in their operations or user behaviors.