Operation Analytics and Investigating Metric Spike

PROJECT DESCRIPTION

The project is about analyzing the operations which are performed in company where overall focus is on the operations performed to increase the growth of company . In this project helps us to get a company's tasks or operations work to be done smoothly and gives us an insight how to improve our weak areas.

Investigation Metric Spike is a part of operation analytics where it helps to answer the question related to teams and helps overall project team that where or in which direction the project is actually going on.

APPROACH

Operation Analytics :I thought about the company's work done that are marketing, technical , sales , etc.

Investigating Metric spike: Here as name suggest first idea I got was related to metrics that means overall growth of company , teams workflow.

Approach for this project is straightforward:

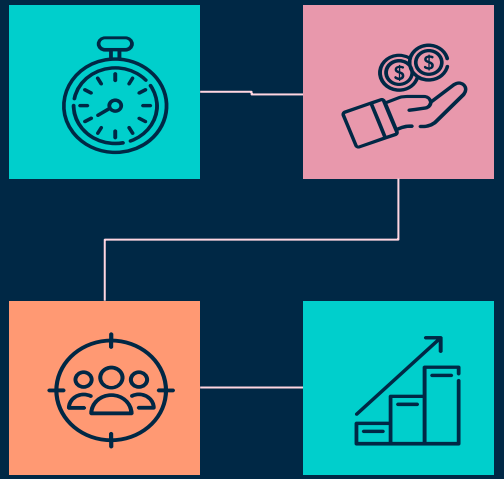
1. Created a database as per the provided resources in MySQL.
2. Read the question asked and started drafting on, which query can be used.
3. Input the query in MySQL and run that to get output for question asked.
4. The approach is quite simple as it contains the use of sql advanced functions that have been learnt during the session and used for completing this project.

TECH-STACK USED

For implementation of this project I have used MySQL Command Line Client , Version:8.0.27
Purpose of using MySQL is that it is already installed and I know how to use it.Also MySQL is mostly used to write and implement the sql queries.It is easy to use if installed once as compared to suggested installations
And mode.com.

INSIGHTS

While making this project I gained a knowledge about the queries to be used when any question comes related to data analytics. Also it gave me an insight that at what type of question which queries can be followed up.Also this project cover up the complex queries like nested queries and also gave me idea to use windows functions and data and time function in real time applications.



CASE STUDY 1:JOB DATA

Number of jobs reviewed:

```
mysql> SELECT
    ->         job_data.ds,
    ->         COUNT(*) / 24 AS no_of_job
    ->     FROM
    ->         job_data
    ->     WHERE
    ->         job_data.ds BETWEEN '2020-11-01' AND '2020-11-30'
    ->     GROUP BY
    ->         job_data.ds ;
+------------+-----------+
| ds         | no_of_job |
+------------+-----------+
| 2020-11-27 |    0.0417 |
| 2020-11-25 |    0.0417 |
| 2020-11-30 |    0.0833 |
| 2020-11-29 |    0.0417 |
| 2020-11-26 |    0.0417 |
| 2020-11-28 |    0.0833 |
+------------+-----------+
6 rows in set (0.17 sec)
```

```
SELECT
job_data.ds,
COUNT(*) / 24 AS no_of_job
FROM
job_data
WHERE
job_data.ds BETWEEN '2020-11-01' AND
'2020-11-30'
GROUP BY
  job_data.ds ;
```

Throughput :

```
mysql> WITH A AS(SELECT ds, COUNT(job_id) AS num_jobs, SUM(time_spent) AS total_time FROM job_data WHERE event IN('transfer','decision') AND ds BETWEEN '2020-11-01' AND
 '2020-11-30' GROUP BY ds  ) SELECT ds, 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 BET
WEEN 6 PRECEDING AND CURRENT ROW),2) AS throughput FROM A;
+------------+------------+
| ds         | throughput |
+------------+------------+
| 2020-11-25 |       0.02 |
| 2020-11-27 |       0.01 |
| 2020-11-28 |       0.02 |
| 2020-11-29 |       0.02 |
| 2020-11-30 |       0.03 |
+------------+------------+
5 rows in set (0.00 sec)
```

SELECT ds, COUNT(job_id) AS num_jobs, SUM(time_spent) AS total_time FROM job_data WHERE event IN('transfer','decision') AND ds BETWEEN '2020-11-01' AND '2020-11-30' GROUP BY ds) SELECT ds, 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 throughput ;

Percentage share of Each language :

language	percentage
Persian	200.00
Persian	200.00
Persian	200.00
Persian	200.00
Hindi	100.00
Arabic	100.00
Italian	100.00
French	100.00
Hindi	100.00
Arabic	100.00
Italian	100.00
French	100.00
Hindi	100.00
Arabic	100.00
Italian	100.00
French	100.00
Persian	100.00
Hindi	100.00
Arabic	100.00
Italian	100.00
French	100.00
Hindi	50.00
Arabic	50.00
Italian	50.00
French	50.00

25 rows in set (0.02 sec)

```
SELECT Language, COUNT(job_id) AS num_jobs
FROM job_data WHERE ds BETWEEN '2020-11-01'
AND '2020-11-30' GROUP BY language, total AS (
SELECT COUNT(job_id) AS total_jobs FROM
job_data WHERE ds BETWEEN '2020-11-01' AND
'2020-11-30' GROUP BY language  ) SELECT
language, ROUND(100.0*num_jobs/total_jobs,2)
AS percentage FROM A CROSS JOIN total  ORDER
BY  percentage DESC;
```

Duplicate Rows:

```
+--------+----------+----------+----------+------------+------+------------+
| job_id | actor_id | event    | language | time_spent | org  | ds         |
+--------+----------+----------+----------+------------+------+------------+
|     23 |     1005 | transfer | Persian  |         22 | D    | 2020-11-28 |
|     23 |     1004 | skip     | Persian  |         56 | A    | 2020-11-26 |
+--------+----------+----------+----------+------------+------+------------+
2 rows in set (0.00 sec)
```

```
SELECT
*,
ROW_NUMBER() OVER (PARTITION BY ds, job_id, actor_id) AS rownum
FROM
job_data
DELETE
FROM
CTE
WHERE
rownum > 1
```

Duplicate Rows:

```
+--------+----------+----------+----------+------------+------+------------+
| job_id | actor_id | event    | language | time_spent | org  | ds         |
+--------+----------+----------+----------+------------+------+------------+
|     23 |     1005 | transfer | Persian  |         22 | D    | 2020-11-28 |
|     23 |     1004 | skip     | Persian  |         56 | A    | 2020-11-26 |
+--------+----------+----------+----------+------------+------+------------+
2 rows in set (0.00 sec)
```

```
SELECT
*,
ROW_NUMBER() OVER (PARTITION BY ds, job_id, actor_id) AS rownum
FROM
job_data
DELETE
FROM
CTE
WHERE
rownum > 1
```

Duplicate Rows:

```
+--------+----------+----------+----------+------------+------+------------+
| job_id | actor_id | event    | language | time_spent | org  | ds         |
+--------+----------+----------+----------+------------+------+------------+
|     23 |     1005 | transfer | Persian  |         22 | D    | 2020-11-28 |
|     23 |     1004 | skip     | Persian  |         56 | A    | 2020-11-26 |
+--------+----------+----------+----------+------------+------+------------+
2 rows in set (0.00 sec)
```

```
SELECT
*,
ROW_NUMBER() OVER (PARTITION BY ds, job_id, actor_id) AS rownum
FROM
job_data
DELETE
FROM
CTE
WHERE
rownum > 1
```

CASE STUDY 2:Investigating Metrics Spike

User Engagement:

```
SELECT DATE_TRUNC('week', e.occurred_at),
COUNT(DISTINCT e.user_id) AS
weekly_active_users FROM
tutorial.yammer_events e WHERE e.event_type =
'engagement' AND e.event_name = 'login' GROUP
BY 1  ORDER BY 1
```

	date trunc	weekly active users
1	2014-04-28 00:00:00	701
2	2014-05-05 00:00:00	1054
3	2014-05-12 00:00:00	1094
4	2014-05-19 00:00:00	1147
5	2014-05-26 00:00:00	1113
6	2014-06-02 00:00:00	1173
7	2014-06-09 00:00:00	1219
8	2014-06-16 00:00:00	1262

User Growth:

```
SELECT DATE_TRUNC ('day',created_at) AS day,
COUNT(*) AS user,
COUNT (CASE WHEN activated_at IS NOT NULL THEN u.user_id
ELSE NULL END) AS activated_users
FROM users
 WHERE created_at >= '2021-04-01'AND created_at < '2021-04-30‘
GROUP BY 1
ORDER BY 1
```

Weekly Retention:

```
SELECT DATE_TRUNC('week', occurred_at) AS
week,
 COUNT(CASE WHEN e.event_type = 'engagement'
THEN e.user_id ELSE NULL END) AS engagement,
COUNT(CASE WHEN e.event_type = 'signup_flow'
THEN e.user_id ELSE NULL END) AS signup
FROM events
GROUP BY 1
 ORDER BY 1
```

	week	engagement	signup
1	2014-04-28 00:00:00	8709	440
2	2014-05-05 00:00:00	17532	884
3	2014-05-12 00:00:00	17047	960
4	2014-05-19 00:00:00	17890	955
5	2014-05-26 00:00:00	17193	978
6	2014-06-02 00:00:00	18608	1043
7	2014-06-09 00:00:00	18233	1073
8	2014-06-16 00:00:00	18976	1136

Weekly Engagement:

	week	weekly users	computer	phone	tablet
1	2014-04-28 00:00:00	701	423	231	147
2	2014-05-05 00:00:00	1054	720	373	248
3	2014-05-12 00:00:00	1094	724	386	269
4	2014-05-19 00:00:00	1147	772	409	280
5	2014-05-26 00:00:00	1113	727	386	262
6	2014-06-02 00:00:00	1173	800	391	285
7	2014-06-09 00:00:00	1219	808	446	283
8	2014-06-16 00:00:00	1262	825	437	307

Email Engagement:

```
SELECT DATE_TRUNC('week', occurred_at) AS week,
COUNT(CASE WHEN e.action = 'sent_weekly_digest' THEN e.user_id ELSE NULL
END) AS weekly_emails,
COUNT(CASE WHEN e.action = 'sent_reengagement_email' THEN e.user_id ELSE
NULL END) AS reengagement_emails,      COUNT(CASE WHEN e.action =
'email_open' THEN e.user_id ELSE NULL END) AS email_opens,      COUNT(CASE
WHEN e.action = 'email_clickthrough' THEN e.user_id ELSE NULL END) AS
email_clickthroughs
FROM emails_events
```

Data | Fields | Source — Export | Copy

	week	weekly emails	reengagement emails	email opens	email clickthroughs
1	2014-04-28 00:00:00	908	98	332	187
2	2014-05-05 00:00:00	2602	164	919	434
3	2014-05-12 00:00:00	2665	175	971	479
4	2014-05-19 00:00:00	2733	179	995	498
5	2014-05-26 00:00:00	2822	179	1026	453
6	2014-06-02 00:00:00	2911	199	993	492
7	2014-06-09 00:00:00	3003	190	1070	533
8	2014-06-16 00:00:00	3105	234	1161	563

RESULTS

While making this project I am able get hands on real time project like Operation Analytics and investigating metrics spike.Also I came to know how actually the company manage the data and how can I apply the sql to get particular output as per required. As this project was so brainstroming so that I get an actual idea of analytics in real world.I am also able to apply my learning like sql functions on any real time application and eagar to learn more thing and get an experience by working on real time applications.

THANK YOU