Operation Analytics and Investigating Metric Spike

(Advanced SQL)

Project Description:

We are working for a company like Microsoft designated as Data Analyst Lead and is provided with different data sets & tables from which we must derive certain insights out of it and answer the questions asked by different departments.

A) Case Study 1 (Job Data):

1. Number of jobs reviewed: Number of jobs reviewed over time. We have to calculate the number of jobs reviewed per hour per day for November 2020.

2. Throughput: It is the no. of events happening per second. We have to calculate 7 day rolling average of throughput. For throughput, do we prefer daily metric or 7-day rolling and why.

3. Percentage share of each language: Share of each language for different contents. We have to calculate the percentage share of each language in the last 30 days.

4. Duplicate rows: Rows that have the same value present in them. We have to display duplicates from the table.

B) Case Study 2 (Investigating metric spike):

1. User Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service. We have to calculate the weekly user engagement.

2. User Growth: Number of users growing over time for a product. We have to calculate the user growth for product.

3. Weekly Retention: Users getting retained weekly after signing-up for a product. We have to calculate the weekly retention of users-sign up cohort.

4. Weekly Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly. We have to calculate the weekly engagement per device.

5. Email Engagement: Users engaging with the email service. We have to calculate the email engagement metrics.

Approach:

We plan to execute SQL queries on given database to create insights for the teams to make data driven decision.

The SQL queries will give the following insights:

A) Case Study 1 (Job Data):

1. We calculated the number of jobs reviewed per hour per day for November 2020.

2. We calculated 7 day rolling average of throughput. For throughput, we prefer 7-day rolling average. Because, rolling averages are useful for finding long-term trends otherwise disguised by occasional fluctuations.

3. We calculated the percentage share of each language in the last 30 days.

4. We displayed duplicates from the table.

B) Case Study 2 (Investigating metric spike):

1. We calculated the weekly user engagement.

2. We calculated the user growth for product.

3. We calculated the weekly retention of users-sign up cohort.

4. We calculated the weekly engagement per device.

5. We calculated the email engagement metrics.

Tech-Stack Used:

MySQL by Oracle Corporation – Importing Database, Running SQL Queries to get insights

Excel by Microsoft Corporation – For extracting & manipulating data

Word by Microsoft Corporation – For creating a project report

Insights:

A) Case Study 1 (Job Data):

1. Number of jobs reviewed:

```
SELECT
     ds,COUNT(job_id) AS jobs_per_day, SUM(time_spent)/3600
AS hours_spent
FROM
     opsanalytics.job_data
WHERE
     ds >='2020-11-01'  AND ds <='2020-11-30'
group by
     ds;
```

2. Throughput:

```
WITH CTE AS (
SELECT
     ds,
COUNT(job_id) AS num_jobs,
SUM(time_spent) AS total_time
FROM
     opsanalytics.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_7d
FROM
     CTE
```

We prefer 7-day rolling average. Because, rolling averages are useful for finding long-term trends otherwise disguised by occasional fluctuations.

3. Percentage share of each language:

```
WITH CTE AS (
SELECT
Language,
COUNT(job_id) AS num_jobs
FROM
     opsanalytics.job_data
WHERE
     event IN(‘transfer’,’decision’)
AND
     ds BETWEEN ‘2020-11-01’ AND ‘2020-11-30’
GROUP BY
     language
),
total AS (
SELECT
     COUNT(job_id) AS total_jobs
FROM
     opsanalytics.job_data
WHERE
event IN(‘transfer’,’decision’)
AND
     ds BETWEEN ‘2020-11-01’ AND ‘2020-11-30’
GROUP BY
     language
)
SELECT
     language, ROUND(100.0*num_jobs/total_jobs,2) AS
perc_jobs
FROM
     CTE
ORDER BY
     perc_job DESC
```

4. Duplicate rows:

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

B) Case Study 2 (Investigating metric spike):

1. User Engagement:

```
SELECT
      DATE_TRUNC(‘week’, e.occurred_at), COUNT(DISTINCT e.user_id)
AS
      weekly_active_users
FROM
      metric_spike.events e
WHERE
      e.event_type = ‘engagement’ AND e.event_name = ‘login’
GROUP BY 1
ORDER BY 1
```

2. User Growth:

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

3. Weekly Retention:

```
SELECT DATE_TRUNC(‘week’, z.occurred_at) AS “week”,
AVG(z.age_at_event) AS "Average age during week",
COUNT(DISTINCT CASE WHEN z.user_age > 70
THEN z.user_id ELSE NULL END) AS "10+ weeks",
COUNT(DISTINCT CASE WHEN z.user_age < 70 AND z.user_age >=63
THEN z.user_id ELSE NULL END) AS "9 weeks",
COUNT(DISTINCT CASE WHEN z.user_age < 63 AND z.user_age >=56
THEN z.user_id ELSE NULL END) AS "8 weeks",
COUNT(DISTINCT CASE WHEN z.user_age < 56 AND z.user_age >=49
THEN z.user_id ELSE NULL END) AS "7 weeks”,
COUNT(DISTINCT CASE WHEN z.user_age < 49 AND z.user_age >=42
THEN z.user_id ELSE NULL END) AS "6 weeks”,
COUNT(DISTINCT CASE WHEN z.user_age < 42 AND z.user_age >=35
THEN z.user_id ELSE NULL END) AS "5 weeks”,
COUNT(DISTINCT CASE WHEN z.user_age < 35 AND z.user_age >=28
THEN z.user_id ELSE NULL END) AS "4 weeks”,
COUNT(DISTINCT CASE WHEN z.user_age < 28 AND z.user_age >=21
THEN z.user_id ELSE NULL END) AS "3 weeks”,
COUNT(DISTINCT CASE WHEN z.user_age < 21 AND z.user_age >=14
THEN z.user_id ELSE NULL END) AS "2 weeks”,
COUNT(DISTINCT CASE WHEN z.user_age < 14 AND z.user_age >=7
THEN z.user_id ELSE NULL END) AS "1 weeks”,
COUNT(DISTINCT CASE WHEN z.user_age < 7 AND z.user_age >=63
THEN z.user_id ELSE NULL END) AS "Less than a week”,
FROM( SELECT e.occurred_at, u.user_id, DATE_TRUNC(“week”,u.activated_at) AS
activation_week,
EXTRACT(‘day’ FROM e.occurred_at - u.activated_at)
AS age_at_event, EXTRACT(‘day’ FROM ‘201-09-01’::TIMESTAMP - u.activated_at)
AS user_age FROM tutorial.yammer_users u
JOIN tutorial.yammer_events e ON e.user_id = u.user_id
AND e.event_type = ‘engagement’
AND e.evnetn_name= ‘login’
AND e.occurred_at >= ‘2014-05-01’
AND e.occurred_at < ‘2014-09-01’
WHERE u.activated_at IS NOT NULL ) z
GROUP BY 1 ORDER BY 1 LIMIT 100
```

4. Weekly Engagement:

```
SELECT
     DATE_TRUNC(‘week’, occurred_at) AS week,
COUNT(DISTINCT e.user id) AS weekly active users,
COUNT(DISTINCT CASE WHEN e.device IN(‘macbook pro’,’lenovo
thinkpad’,’macbook air’,’’dell inspiron notebook’,’asus
chromebook’,’dell inspiron desktop’,’acer aspire
notebook’,’hp pavilion desktop’,’acer aspire desktop’,’mac
mini’)
THEN e.user id ELSE NULL END)
AS
     computer,
COUNT(DISTINCT CASE WHEN e.device IN(‘iphone 5’,’samsung
galaxy s4’,’nexus 5’,’iphone 5s’,’iphone 4s’,’nokia lumia
635’,’htc one’,’samsung galaxy note’,’amazon fire phone’)
THEN e.user id ELSE NULL END) AS phone,
COUNT(DISTINCT CASE WHEN e.device IN(‘ipad air’,’nexus
7’,’ipad mini’,’nexus 10’,’kindle fire’,’windows
surface’,’samsung galaxy tablet’)
THEN e.user id ELSE NULL END) AS tablet
FROM
     events e
WHERE
     e.event type = ‘engagement’ AND e.event name = ‘login’
GROUP BY 1 ORDER BY 1 LIMIT 100
```

5. 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 email events e
GROUP BY 1
```

Result:

We have run all the above-mentioned SQL queries and got answers of the questions which will help the team to take data driven decisions.

Drive Link:

https://drive.google.com/drive/folders/15A1EFbT-bxsl3SeEPFdiXPmcNSp8Uz7S?usp=sharing