Operation Analytics and Investigating Metric Spike

By: Abhijit Mandape

Project Description

- The project is a collection two used cases for Operation Analytics and Investigating Metric Spike

Approach

- For Case 1:
  - The given dataset is created using the SQL syntax using the MYSQL workbench.
  - The data contained within the tables was explored.
- For Case 2:
  - A separate schema was created and the data in CSV files were uploaded using the option “Table Data Import Wizard” in the MYSQL workbench.
  - The data contained within the tables and the connection between them was explored.

Tech-Stack Used

- MYSQL Workbench 8.0.33 for writing SQL queries.
- XAMPP Control Panel v 3.3.0 for creating local server.

Insights

- The clauses “COUNT”, “SUM”, “DATE”, “CASE”, were explored.
- Advanced JOINS and subquery implementation gave a good understanding of how more refined output can be obtained.

Result

- The project helped to learn and practice advanced topics like JOINS, subqueries, and complex calculations using the arithmetic operators.
- Exploring the use of “Table Data Import Wizard” gave insights on how easily to import large data sets without typing the contents manually with sql query.

Case Study 1 (Job Data)

Query to create table:

```
CREATE DATABASE holadb;

USE holadb;

CREATE TABLE job_data(
        job_id INT,
        actor_id INT,
        even_t VARCHAR(20),
        lan_guage VARCHAR(10),
        time_spent VARCHAR(5),
        org VARCHAR(20),
        ds DATETIME);
```

```
INSERT INTO job_data VALUES (21, 1001, "skip", "English", 15, "A", "2020-11-30"),
        (22,1006, "transfer", "Arabic", 25, "A", "2020-11-30"),
        (23,1003, "decision", "Persian", 20, "C", "2020-11-29"),
        (23,1005, "transfer", "Persian", 22, "D", "2020-11-28"),
        (25,1002, "decision", "Hindi", 11, "B", "2020-11-28"),
        (11,1007, "decision", "French", 104, "D", "2020-11-27"),
        (23,1004, "skip", "Persian", 56, "A", "2020-11-26" ),
        (20,1003, "transfer", "Italian", 45, "C", "2020-11-25");

SELECT * FROM job_data;
```

job_id	actor_id	even_t	lan_guage	time_spent	org	ds
21	1001	skip	English	15	A	2020-11-30 00:00:00
22	1006	transfer	Arabic	25	A	2020-11-30 00:00:00
23	1003	decision	Persian	20	C	2020-11-29 00:00:00
23	1005	transfer	Persian	22	D	2020-11-28 00:00:00
25	1002	decision	Hindi	11	B	2020-11-28 00:00:00
11	1007	decision	French	104	D	2020-11-27 00:00:00
23	1004	skip	Persian	56	A	2020-11-26 00:00:00
20	1003	transfer	Italian	45	C	2020-11-25 00:00:00

A. **Number of jobs reviewed:** Amount of jobs reviewed over time.
Your task: Calculate the number of jobs reviewed per hour per day for November 2020?

SQL query:

```
SELECT DATE(ds) AS 'date', ROUND((COUNT(job_id)/(SUM(time_spent/60)))) AS
        job_viewed_per_hour_per_day
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-31'
GROUP BY ds;
```

Output:

date	job_viewed_per_hour_per_day
2020-11-25	1
2020-11-26	1
2020-11-27	1
2020-11-28	4
2020-11-29	3
2020-11-30	3

B. **Throughput:** It is the no. of events happening per second.
Your task: Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

Sql Query:

```
SELECT date, job_viewed_hours_per_day,
        AVG(job_viewed_hours_per_day) OVER (PARTITION BY date) AS
        rolling_average_throughput
FROM (SELECT DATE(ds) AS 'date', ROUND(COUNT(job_id)/SUM(time_spent/60))
      AS job_viewed_hours_per_day
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-31'
GROUP BY ds) AS throughput;
```

Output:

date	job_viewed_hours_per_day	rolling_average_throughput
2020-11-25	1	1.0000
2020-11-26	1	1.0000
2020-11-27	1	1.0000
2020-11-28	4	4.0000
2020-11-29	3	3.0000
2020-11-30	3	3.0000

C. **Percentage share of each language:** Share of each language for different contents.
Your task: Calculate the percentage share of each language in the last 30 days?

SQL query:

```
SELECT DATE(ds) AS date, lan_guage, SUM(time_spent) AS total_time_mins,
        COUNT(job_id) AS job_count, count(job_id)*100 / sum(count(*)) OVER() AS
        percentage_share
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-31'
GROUP BY lan_guage
ORDER BY ds;
```

Output:

date	lan_guage	total_time_mins	job_count	percentage_share
2020-11-25	Italian	45	1	12.5000
2020-11-27	French	104	1	12.5000
2020-11-28	Hindi	11	1	12.5000
2020-11-29	Persian	98	3	37.5000
2020-11-30	English	15	1	12.5000
2020-11-30	Arabic	25	1	12.5000

D. **Duplicate rows:** Rows that have the same value present in them.
Your task: Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

SQL query:

```
SELECT ds, job_id, actor_id, even_t, lan_guage, time_spent, org,
        CASE WHEN COUNT(*) > 1 THEN 'Duplicate' ELSE 'No Duplicate' END AS
        Duplicate
FROM job_data
GROUP BY ds, job_id, actor_id, even_t, lan_guage, time_spent, org;
```

Output:

ds	job_id	actor_id	even_t	lan_guage	time_spent	org	Duplicate
2020-11-25 00:00:00	20	1003	transfer	Italian	45	C	No Duplicate
2020-11-26 00:00:00	23	1004	skip	Persian	56	A	No Duplicate
2020-11-27 00:00:00	11	1007	decision	French	104	D	No Duplicate
2020-11-28 00:00:00	23	1005	transfer	Persian	22	D	No Duplicate
2020-11-28 00:00:00	25	1002	decision	Hindi	11	B	No Duplicate
2020-11-29 00:00:00	23	1003	decision	Persian	20	C	No Duplicate
2020-11-30 00:00:00	21	1001	skip	English	15	A	No Duplicate
2020-11-30 00:00:00	22	1006	transfer	Arabic	25	A	No Duplicate

Case Study 2 (Investigating metric spike)

A. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.
Your task: Calculate the weekly user engagement?

SQL query:

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

Output:

week	user_engagement
17	539
18	1035
19	1051
20	1099
21	1066
22	1139
23	1196
24	1228
25	1207

B. **User Growth:** Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

SQL query:

```
SELECT DATE(created_at) AS date, COUNT(*) AS user_count,
        COUNT(*) - (SELECT COUNT(*)
                    FROM users AS prev
                    WHERE DATE(prev.created_at) < DATE(curr.created_at)) AS
                     user_growth
FROM users AS curr
GROUP BY DATE(created_at)
ORDER BY DATE(created_at);
```

Output:

date	user_count	user_growth
2013-01-01	13	13
2013-01-02	11	-2
2013-01-03	14	-10
2013-01-04	11	-27
2013-01-05	3	-46
2013-01-06	4	-48
2013-01-07	13	-43
2013-01-08	13	-56
2013-01-09	11	-71

C. **Weekly Retention:** Users getting retained weekly after signing-up for a product.
Your task: Calculate the weekly retention of users-sign up cohort?

SQL query:

Step 1: Identify the sign-up cohort for a specific week

```
SELECT WEEK(created_at) AS cohort_week, COUNT(DISTINCT user_id) AS cohort_size
FROM users
GROUP BY cohort_week
ORDER BY cohort_week;
```

Output:

cohort_week	cohort_size
0	197
1	300
2	299
3	325
4	322
5	341
6	344
7	353
8	350

Step 2: Calculate the weekly retention for each subsequent week

```
SELECT cohort_week, retention_week, COUNT(DISTINCT user_id) AS retained_users,
        COUNT(DISTINCT user_id) / (SELECT COUNT(DISTINCT user_id)
        FROM users WHERE WEEK(created_at) = cohort_week) * 100 AS retention_rate
FROM (SELECT WEEK(u.created_at) AS cohort_week, WEEK(e.occurred_at) AS
         retention_week, u.user_id
        FROM users u
        INNER JOIN events e ON u.user_id = e.user_id
        WHERE e.occurred_at > u.created_at
        GROUP BY cohort_week, retention_week, u.user_id) AS retention_data
GROUP BY cohort_week, retention_week
ORDER BY cohort_week, retention_week;
```

D. **Weekly Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.
Your task: Calculate the weekly engagement per device?

SQL query:

```
SELECT YEARWEEK(occurred_at) AS week, device, COUNT(*) AS engagement_count
FROM events
GROUP BY YEARWEEK(occurred_at), device
ORDER BY week, device;
```

Output:

week	device	engagement_count
201417	acer aspire desktop	69
201417	acer aspire notebook	184
201417	amazon fire phone	79
201417	asus chromebook	207
201417	dell inspiron desktop	139
201417	dell inspiron notebook	323
201417	hp pavilion desktop	109
201417	htc one	107
201417	ipad air	248

E. **Email Engagement:** Users engaging with the email service.
Your task: Calculate the email engagement metrics?

SQL Query:

```
SELECT WEEK(occurred_at) AS week,
        COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) AS
            unique_users_email_opened,
        COUNT(DISTINCT CASE WHEN action = 'email_clickthrough' THEN user_id END) AS
            unique_users_email_clicked,
        COUNT(DISTINCT CASE WHEN action = 'sent_weekly_digest' THEN user_id END) AS
            unique_sent_weekly_digest,
        COUNT(DISTINCT CASE WHEN action = 'sent_reengagement_email' THEN user_id END)
            AS unique_sent_reengagement_email
        FROM email_events
GROUP BY week(occurred_at)
ORDER BY week(occurred_at);
```

Output:

week	unique_users_email_opened	unique_users_email_clicked	unique_sent_weekly_digest	unique_sent_reengagement_email
17	310	166	908	73
18	900	425	2602	157
19	961	476	2665	173
20	989	501	2733	191
21	996	436	2822	164
22	965	478	2911	192
23	1057	529	3003	197
24	1136	549	3105	226
25	1084	524	3207	196