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"),

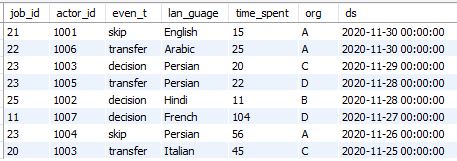
(25,1002, "decision", "Hindi", 11, "B", "2020-11-28"),

(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;



1. **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:



1. **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))

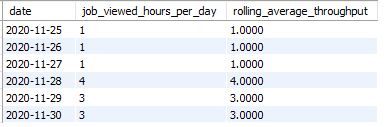
AS job\_viewed\_hours\_per\_day

FROM job\_data

WHERE ds BETWEEN '2020-11-01' AND '2020-11-31'

GROUP BY ds) AS throughput;

Output:



1. **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

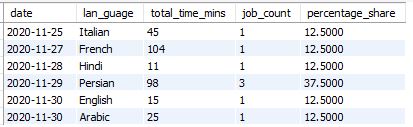
FROM job\_data

WHERE ds BETWEEN '2020-11-01' AND '2020-11-31'

GROUP BY lan\_guage

ORDER BY ds;

Output:



1. **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:

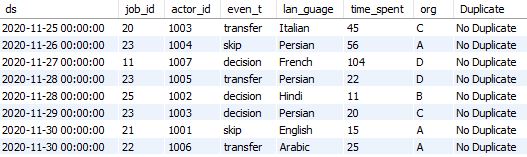
SELECT ds, job\_id, actor\_id, even\_t, lan\_guage, time\_spent, org,

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:



###### 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.   
   **Your task:** Calculate the weekly user engagement?

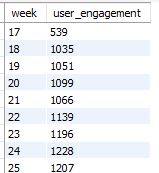
SQL query:

SELECT WEEK(occurred\_at) AS week, COUNT(DISTINCT user\_id) AS user\_engagement

FROM events

GROUP BY week;

Output:



1. **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

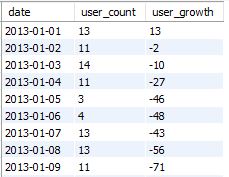
user\_growth

FROM users AS curr

GROUP BY DATE(created\_at)

ORDER BY DATE(created\_at);

Output:



1. **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

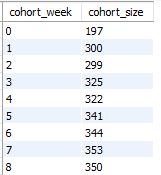
SELECT WEEK(created\_at) AS cohort\_week, COUNT(DISTINCT user\_id) AS cohort\_size

FROM users

GROUP BY cohort\_week

ORDER BY cohort\_week;

Output:



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;

1. **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:

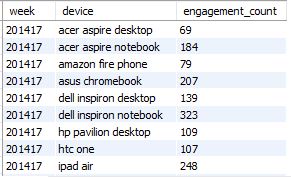
SELECT YEARWEEK(occurred\_at) AS week, device, COUNT(\*) AS engagement\_count

FROM events

GROUP BY YEARWEEK(occurred\_at), device

ORDER BY week, device;

Output:



1. **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:

