**SQL Case Study – 1**

**‘Operation Analytics’**

By: Shrey Shah

**PROJECT DESCRIPTION:**

The case study describes the ‘Operation Analytics’ where we have a table which stores the details about the jobs as follows:

* job\_id: unique identifier of jobs
* actor\_id: unique identifier of actor
* event: decision/skip/transfer
* language: language of the content
* time\_spent: time spent to review the job in seconds
* org: organization of the actor,
* ds: date in the yyyy/mm/dd format.

Some observations:

* The job\_id and actor\_id have been specified as unique keys for both but in any table we can have only 1 primary key.
* For convenience, we have not kept any key as the primary key because we have few records and we have to perform some queries and cannot afford to lose data.
* As we clearly do not have information or resources about the event column we will consider all 3 as the categories in the event.

In this project we are going to get answers of some questions for the analysis purpose by performing some querying on the given data.

QA: Calculate the number of jobs reviewed per hour per day for November 2020?

QB: 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?

QC: Calculate the percentage share of each language in the last 30 days?

QD: Let’s say you see some duplicate rows in the data. How will you display duplicates from the table?

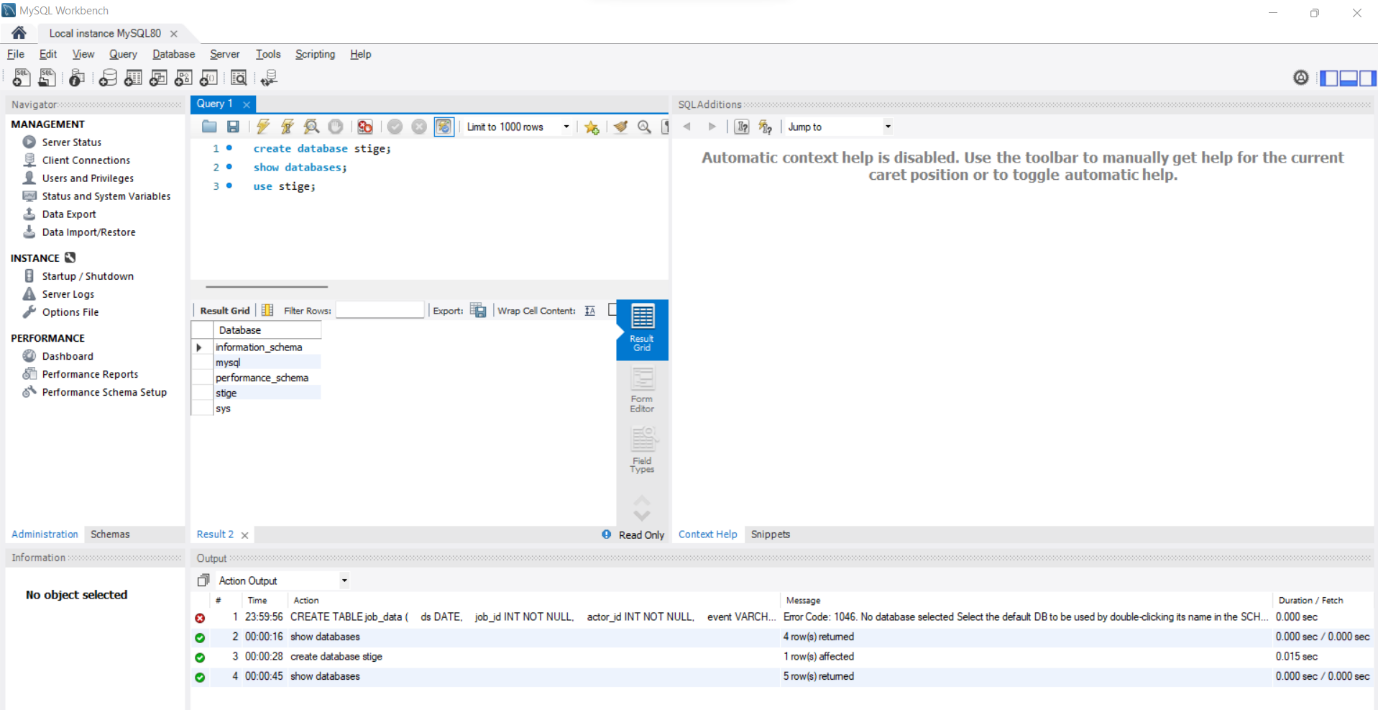
**APPROACH:**

For this project, I have used the small dataset of 8 records provided to develop queries to find answers of the questions posed above for the analytics purpose.

I have used MySQL Workbench for trying the SQL queries.

Initially, I established a local connection with the MySQL Server using the Workbench.

Next, I created a database named ‘stige’ for personal use as follows.



Once this was done, I created a table named ‘job\_data’ with all the columns as described above with the SQL query:

CREATE TABLE job\_data

(

ds DATE,

job\_id INT NOT NULL,

actor\_id INT NOT NULL,

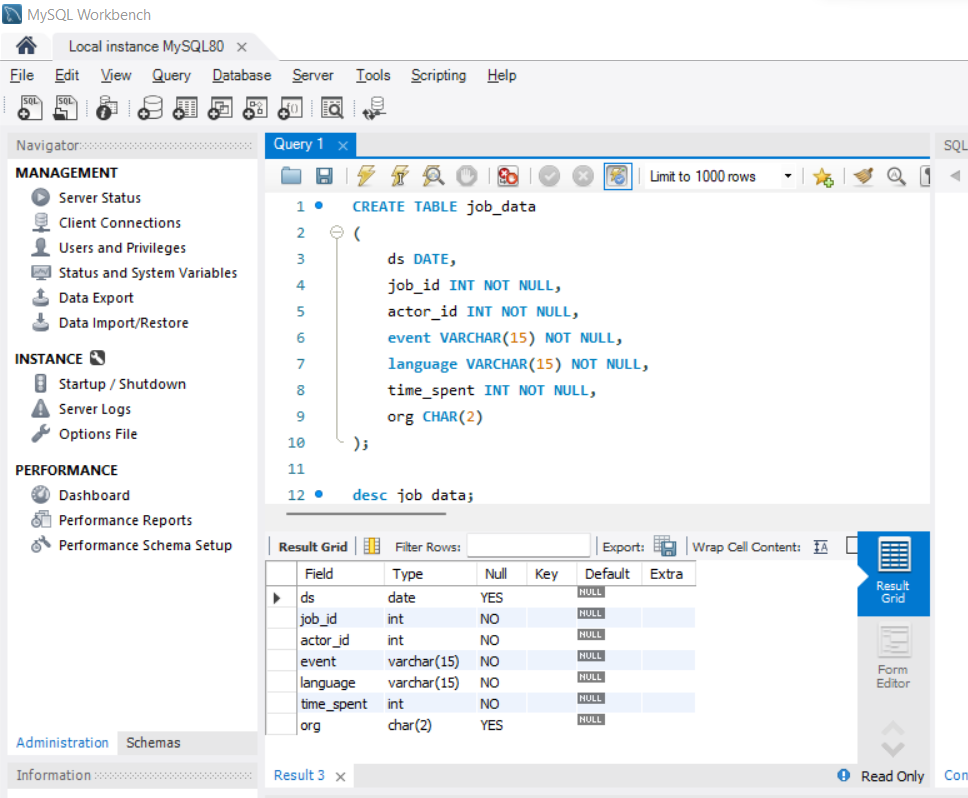
event VARCHAR(15) NOT NULL,

language VARCHAR(15) NOT NULL,

time\_spent INT NOT NULL,

org CHAR(2)

);



After this finally, I added the dataset rows with the following query and displayed the result:

INSERT INTO job\_data (ds, job\_id, actor\_id, event, language, time\_spent, org)

VALUES ('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),

('2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),

('2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),

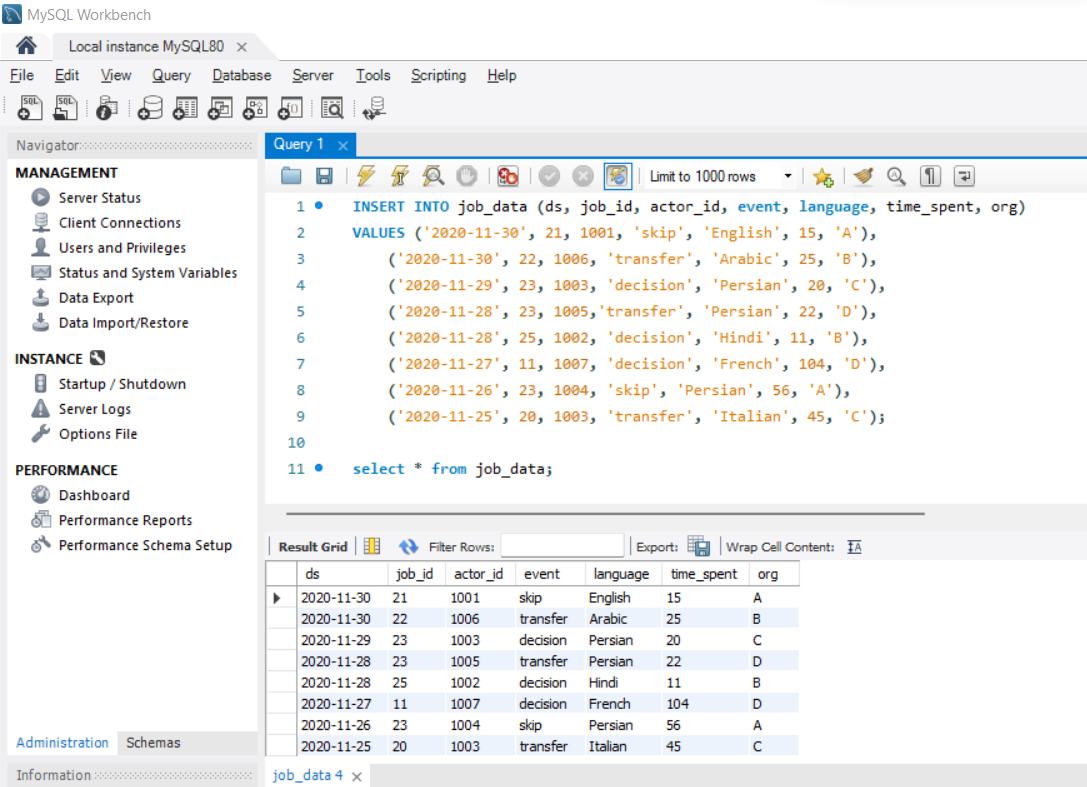
('2020-11-28', 23, 1005,'transfer', 'Persian', 22, 'D'),

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

('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),

('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),

('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');



**TECHSTACK USED:**

I installed the MySQL Installer for Windows.

Version: 8.0.28.0

It internally downloads the MySQL Server, MySQL Workbench and MySQL Shell all having the same version.

I have used MySQL Installer rather than MySQL Workbench (standalone) because the Installer is a compact utility which internally downloads the above mentioned applications and reduces the job of manually installing all of them.

Since we have the case study in SQL, I chose this technology and application.

**INSIGHTS:**

QA) Calculate the number of jobs reviewed per hour per day for November 2020?

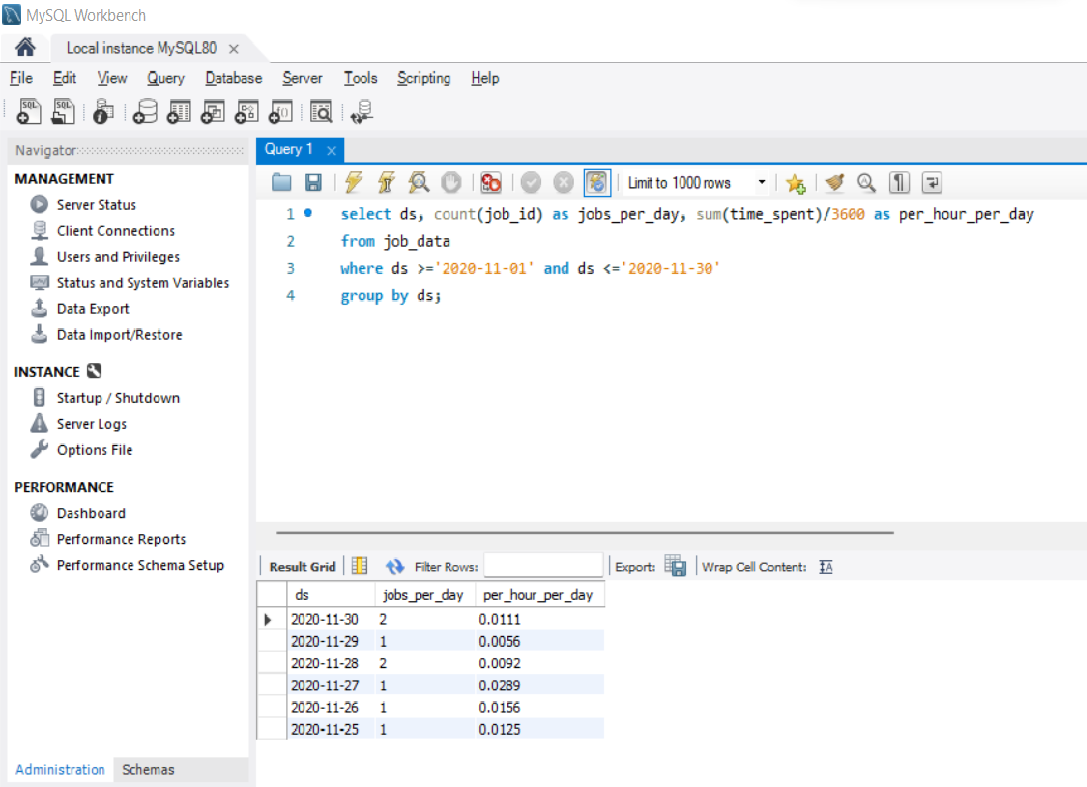
select ds, count(job\_id) as jobs\_per\_day, sum(time\_spent)/3600 as per\_hour\_per\_day

from job\_data

where ds >='2020-11-01' and ds <='2020-11-30'

group by ds;

In this query, we have calculated the number of jobs grouped by their dates and the time spent per hour per day based on the time in seconds / 3600.



QB) 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?

with 7day\_rolling\_avg as(select ds, COUNT(job\_id) as number\_of\_jobs, SUM(time\_spent) as total\_time

from job\_data

where ds >= '2020-11-01' AND ds <= '2020-11-30'

group by ds

)

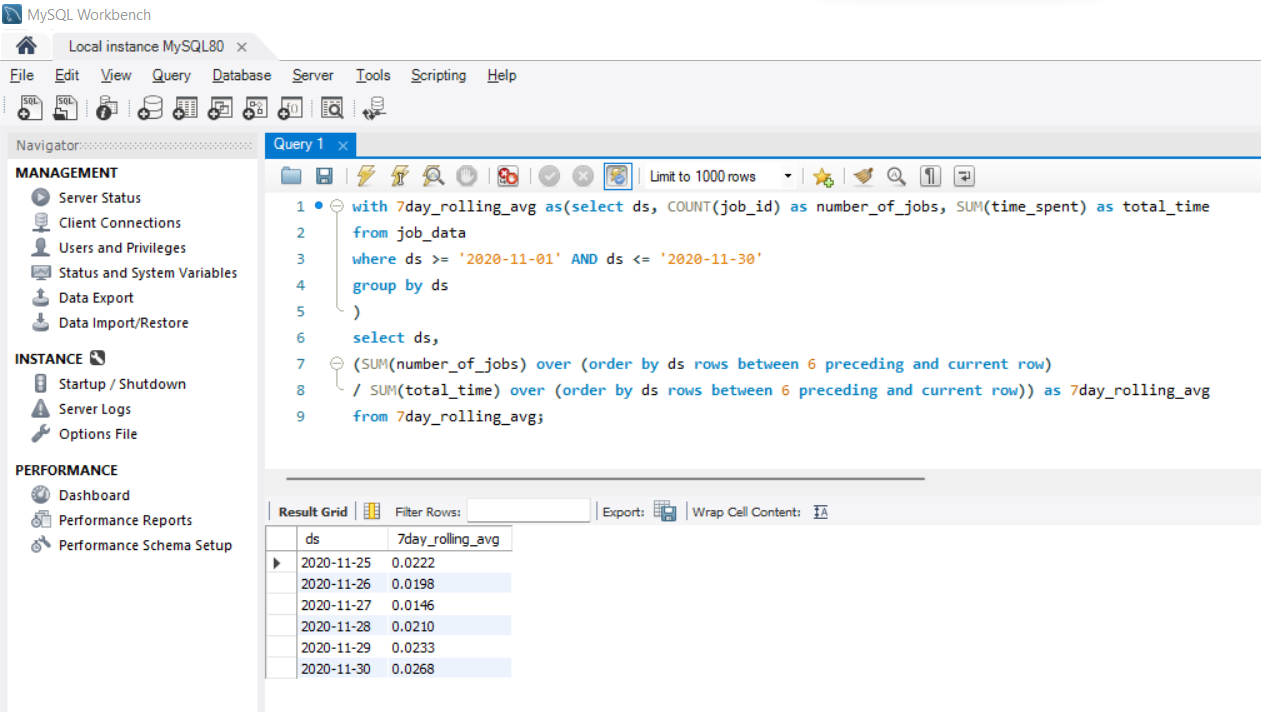
select ds,

(SUM(number\_of\_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)) as 7day\_rolling\_avg

from 7day\_rolling\_avg;

In this query, we have mixed 2 queries. Firstly we find out the number of jobs and the total time of the jobs grouped by date. Using this output as the input table for the second query, we have found out the cumulative average using the 6 rows above the current row and the current row to find the 7 day cumulative / running average.



QC) Calculate the percentage share of each language in the last 30 days?

with percent\_share\_language as (

select language, count(language) as count\_of\_language

from job\_data

where ds >= '2020-11-01' and ds <= '2020-11-30'

group by language),

total as (

select count(language) as total\_language

from job\_data

where ds >= '2020-11-01' and ds <= '2020-11-30'

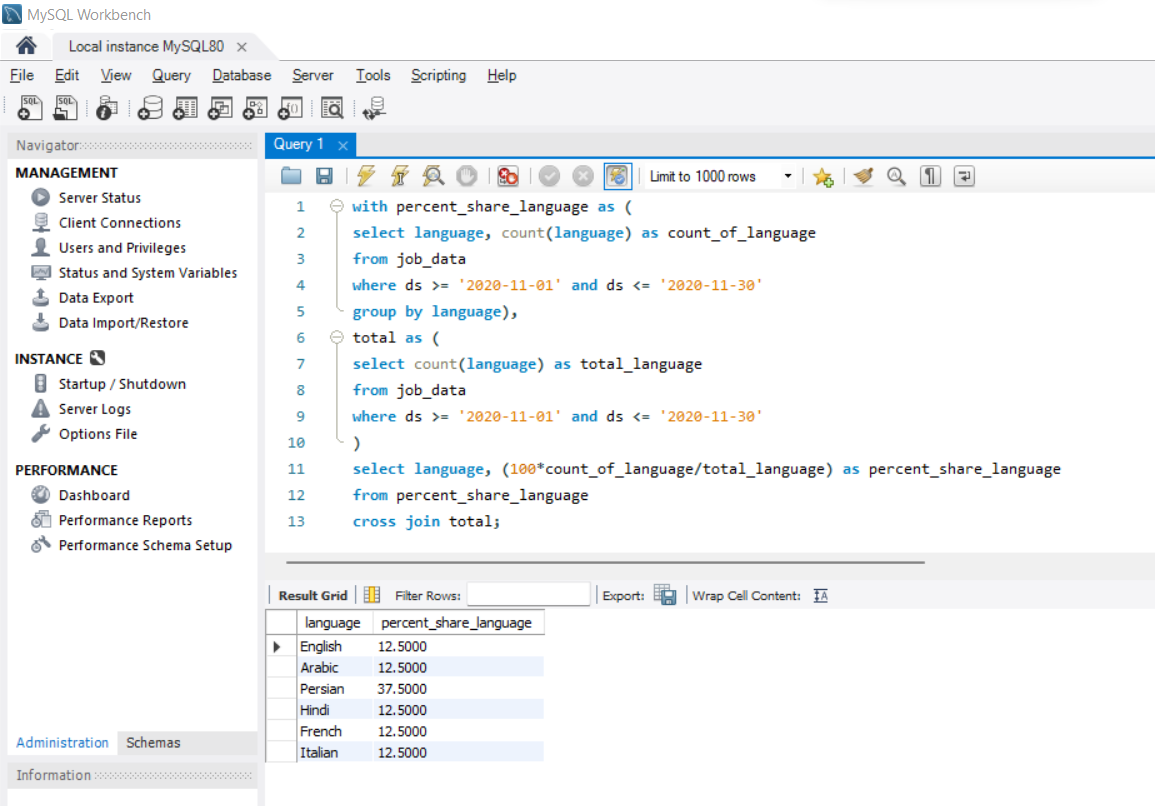
)

select language, (100\*count\_of\_language/total\_language) as percent\_share\_language

from percent\_share\_language

cross join total;

In this query, we are again using 3 queries. Firstly, we are counting the number of individual languages / jobs grouped by the languages. Secondly, we are calculating the sum of all languages / jobs as the total. Lastly, we are performing the ratio of the individual language counts by the total and multiply by 100 to get the percentage of language share.



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

with duplicate\_rows as (

select \*, row\_number() over (partition by ds, job\_id, actor\_id) as rownum

from job\_data)

delete from duplicate\_rows

where rownum > 1;

In this query, we are performing the display and deletion or the duplicate records. We are creating a new column ‘rownum’ which stores the count. If a row is repeated then the rownum will hold the value > 1 and that indicates that it is a duplicate record which can be deleted. In our dataset, we did not have any such duplicate records and hence there will be no result.

**RESULTS:**

* Working on this SQL project, has helped me understand the very new and complex, advanced functions and SQL writing techniques.
* I did trial and error many times with new functions like Windows functions, and writing multiple queries within a single query.
* This project has helped me improve my SQL querying skills to the next level from earlier when I just was able to work with basic SQL clauses.