Project 3 : By Ramana Bansal

Operation Analytics

Case Study : Job Data

MySql File Link:

https://drive.google.com/file/d/1odSeV6OcupEstdC7pSt29lLOcwXeQer7/view?usp=share_link

Project Description:

The project deals with operation analytics. Here, we use end to end data from various company processes to derive insights regarding daily as well as long-term company operations. The data seems to be related to content from various languages. Some of the metrics studied in this project are related to the amount of work done, rate of work done, kinds of events processed etc.

We find the number of jobs reviewed, throughputs, rolling average of daily throughputs, percentage share of content language and finding the duplicated in data. The day-to-day operations as well as weekly percentages give insight into various operations.

Approach: MySql Workbench was used to create a database and required tables. Based on the sample given, data was inserted into the tables using sql queries. The sample was also merged into database using import wizard. The data was then analysed using MySql queries. MySql Workbench was used to run various queries and observe the outputs.

Tech-Stack Used: MySQL Workbench 8.0.33 : To Execute the Sql queries.

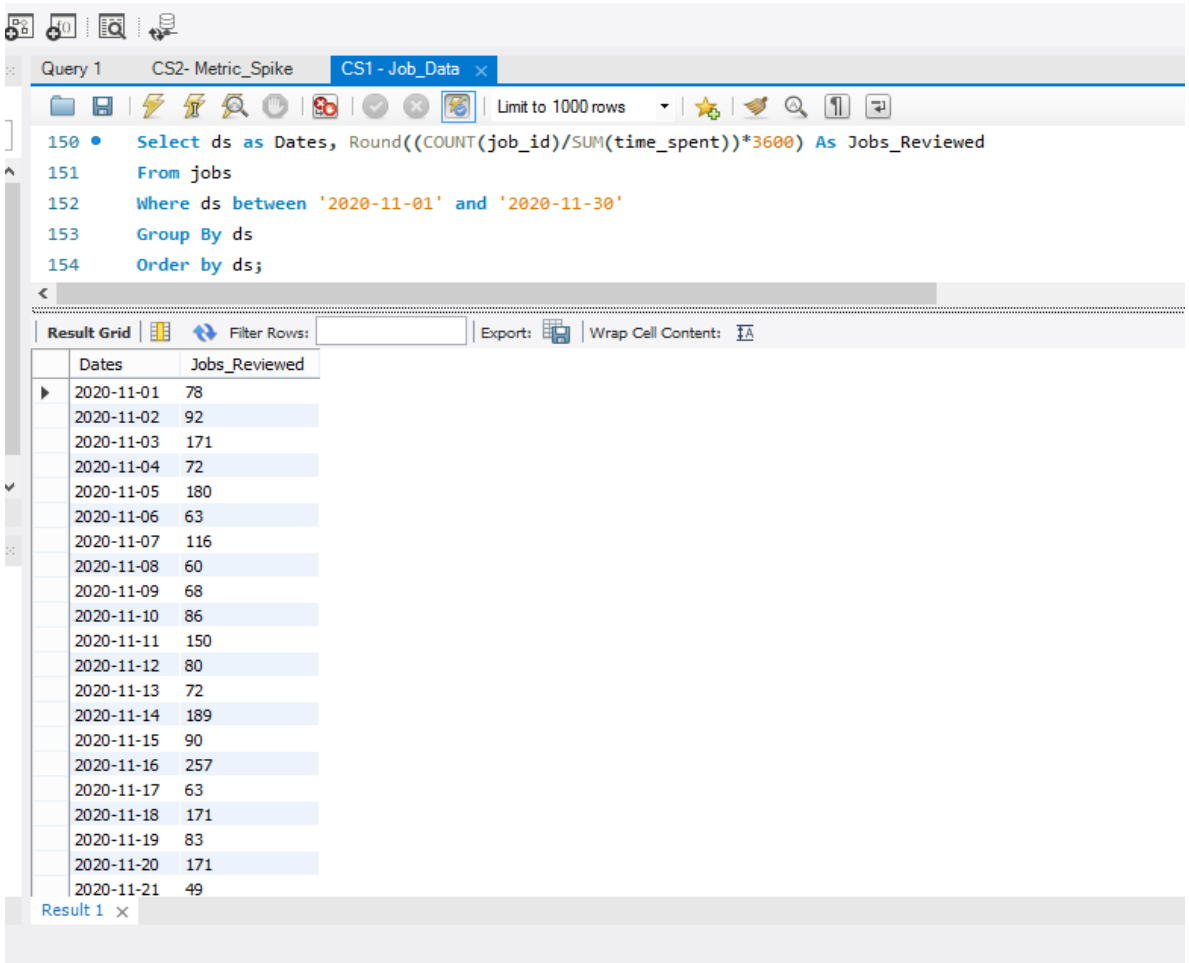
Insights: The project was helpful in getting a general idea about the real life application of data analysis, especially in company operations. It also helped in understanding some of the data-driven processes behind how companies use the data to analyze their performance and their key areas of profit, as well as the areas that might need further improvement.

Result: The project was a little confusing for me initially; however with the aid of doubt forum I was able to get my doubts cleared. It also gave insights regarding how data might be used to make decisions and derive insights in companies.

A. **Number of jobs reviewed:** Amount of jobs reviewed over time. Calculating the number of jobs reviewed per hour per day for November 2020.

```
Select ds as Dates, Round((COUNT(job_id)/SUM(time_spent))*3600) As Jobs_Reviewed
From jobs
Where ds between '2020-11-01' and '2020-11-30'
Group By ds
Order by ds;
```

Data required for November is filtered using between. Total number of jobs reviewed in November is divided by total time spent (in seconds) to get Review rate. The result is multiplied by 3600 to get per hour review rate.



Dates	Jobs_Reviewed
2020-11-01	78
2020-11-02	92
2020-11-03	171
2020-11-04	72
2020-11-05	180
2020-11-06	63
2020-11-07	116
2020-11-08	60
2020-11-09	68
2020-11-10	86
2020-11-11	150
2020-11-12	80
2020-11-13	72
2020-11-14	189
2020-11-15	90
2020-11-16	257
2020-11-17	63
2020-11-18	171
2020-11-19	83
2020-11-20	171
2020-11-21	49

B. **Throughput:** It is the no. of events happening per second.
Calculating 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

- daily-throughput : Total events per second for a day
- rolling_avg_throughput : Avg of last 7 daily_throughput (current row + previous 6 rows)
- 7_day_throughput : Total events per second for a week

```
select ds, Count, time_spent, daily_throughput,
Round(Avg(daily_throughput) Over (ORDER BY ds ROWS BETWEEN
6 PRECEDING AND CURRENT ROW),3 ) as rolling_avg_throughput,
7_day_throughput
from(
Select ds, count(event) as Count,
time_spent, Round((count(event)/sum(time_spent)),3) as
daily_throughput,
Round(count(event) OVER (ORDER BY ds ROWS BETWEEN 6
PRECEDING AND CURRENT ROW)/
sum(time_spent) OVER (ORDER BY ds ROWS BETWEEN 6
PRECEDING AND CURRENT ROW),3)
as 7_day_throughput
from jobs
group by ds) as A;
```

Daily throughput is calculated for each day. Windows function is used to get the average of current + previous 6 rows to get rolling average of throughput.

Rolling metrics might be more suitable since they are not volatile or based on short-term data as compared to daily metrics which, while still useful, do not help in understanding the trends underlying our operations.

Query 1 | CS2- Metric_Spike | CS1 - Job_Data | groupby_clause_err

Limit to 1000 rows

```
select ds, Count, time_spent, daily_throughput,
Round(Avg(daily_throughput) Over (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW),3 ) as rolling_avg_throughput,
7_day_throughput
from(
Select ds, count(event) as Count,
time_spent, Round((count(event)/sum(time_spent)),3) as daily_throughput,
Round(count(event) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)/
sum(time_spent) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW),3)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ds	Count	time_spent	daily_throughput	rolling_avg_throughput	7_day_throughput
2020-08-01	1	109	0.009	0.009	0.009
2020-08-02	1	72	0.014	0.012	0.011
2020-08-03	1	31	0.032	0.018	0.014
2020-08-04	1	61	0.016	0.018	0.015
2020-08-05	1	67	0.015	0.017	0.015
2020-08-06	1	82	0.012	0.016	0.014
2020-08-07	1	39	0.026	0.018	0.015
2020-08-08	1	75	0.013	0.018	0.016
2020-08-09	1	10	0.100	0.031	0.019
2020-08-10	1	58	0.017	0.028	0.018
2020-08-11	1	37	0.027	0.030	0.019
2020-08-12	1	88	0.011	0.029	0.018
2020-08-13	1	52	0.019	0.030	0.019
2020-08-14	1	45	0.022	0.030	0.019
2020-08-15	1	10	0.100	0.042	0.023

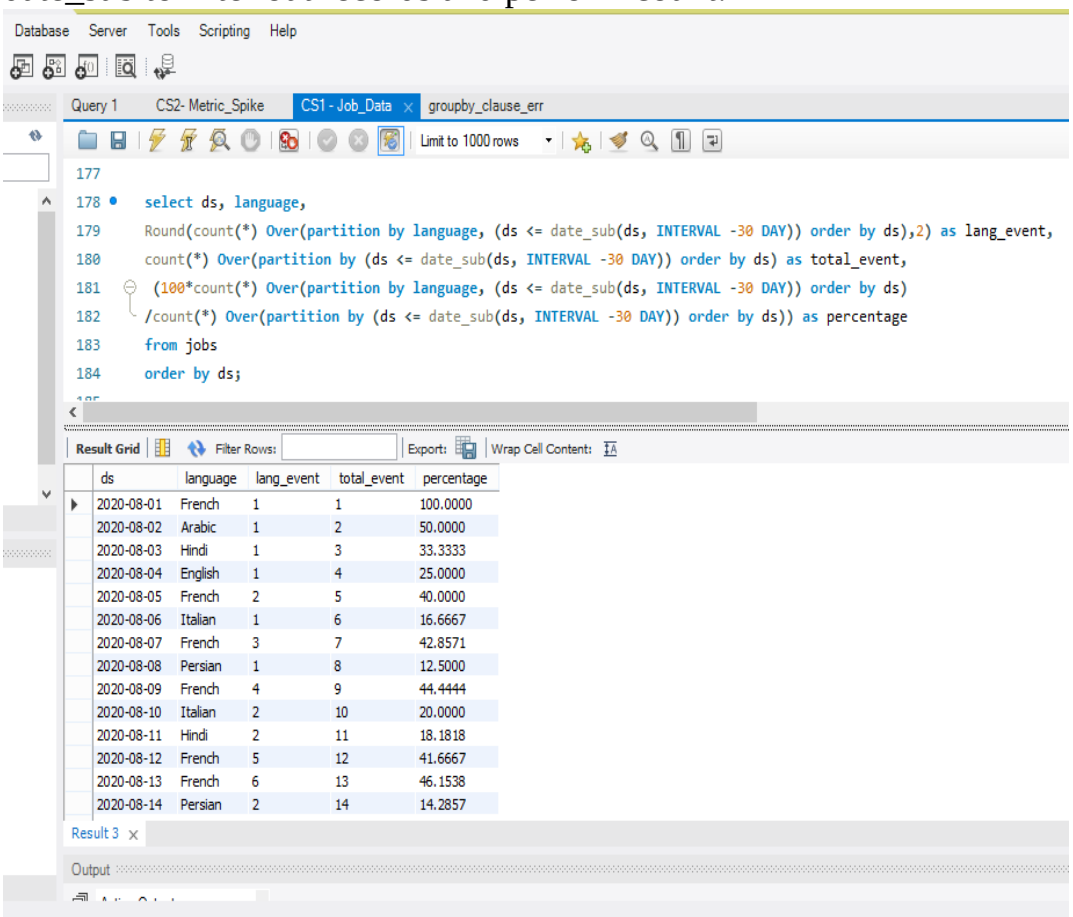
Result 2

Output

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

```
select ds, language,
Round(count(*) Over(partition by language, (ds <= date_sub(ds, INTERVAL -30 DAY)) order by ds),2) as lang_event,
count(*) Over(partition by (ds <= date_sub(ds, INTERVAL -30 DAY)) order by ds) as total_event,
 (100*count(*) Over(partition by language, (ds <= date_sub(ds, INTERVAL -30 DAY)) order by ds)
/count(*) Over(partition by (ds <= date_sub(ds, INTERVAL -30 DAY)) order by ds)) as percentage
from jobs
order by ds;
```

Date_sub is used to get the last 30 day of records. Window function uses date_sub to filter out records and perform count.



ds	language	lang_event	total_event	percentage
2020-08-01	French	1	1	100.0000
2020-08-02	Arabic	1	2	50.0000
2020-08-03	Hindi	1	3	33.3333
2020-08-04	English	1	4	25.0000
2020-08-05	French	2	5	40.0000
2020-08-06	Italian	1	6	16.6667
2020-08-07	French	3	7	42.8571
2020-08-08	Persian	1	8	12.5000
2020-08-09	French	4	9	44.4444
2020-08-10	Italian	2	10	20.0000
2020-08-11	Hindi	2	11	18.1818
2020-08-12	French	5	12	41.6667
2020-08-13	French	6	13	46.1538
2020-08-14	Persian	2	14	14.2857

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

SELECT ds, job_id, actor_id, event, language, time_spent, org, count(*) as duplicates
FROM jobs
GROUP BY ds, job_id, actor_id, event, language, time_spent, org
HAVING COUNT(*) > 1;

Rows that have all the values same in all columns.