Operation Analytics
and
Investigating Metric Spike

Project Description

Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc., and help them derive insights out of the data they collect.

Being one of the most important parts of a company, this kind of analysis is further used to predict the overall growth or decline of a company's fortune. It means better automation, better understanding between cross-functional teams, and more effective workflows.

Investigating metric spike is also an important part of operation analytics as being a Data Analyst you must be able to understand or make other teams understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that its very important to investigate metric spike.

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

Tech-Stack Used

- For solving the tasks I have used MySQL Workbench 8.0 CE

MySQL Workbench is the official graphical user interface □GUI□ tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Case Study 1 (Job Data)

Job Data

1. **Number of jobs reviewed:** Amount of jobs reviewed over time.

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

Approach: I used the data from job_id column of the job_data table, in order to find jobs reviewed per hour per day of November I used the formula as given below

$$\frac{\textbf{\textit{Total Count of job\_id}}}{\mathbf{30}(\textbf{\textit{Days}}) * \mathbf{24}(\textbf{\textit{hours}})}$$

Using the logic as November has 30 Days and each day has 24 hours that will give the per day per hour.

Now, we have two situations here we have to find for DISTINCT (#1) and NON-DISTINCT(#2) jobs reviewed. For which I have used COUNT(DISTINCT job_id) nested function

```
#1 Distinct Jobs reviewed
SELECT
   COUNT(DISTINCT job_id)/(30*24) AS `Distinct Jobs Reviewed Per Day Per Hour`
FROM
   job_data;
```

Output

	Distinct Jobs Reviewed Per Day Per Hour
▶	0.0083

```
#2 Total Jobs reviewed (non-distinct)
 SELECT
    COUNT( job_id)/(30*24) AS `Jobs Reviewed Per Day Per Hour`
FROM
   job_data;
```

Output

	Jobs Reviewed Per Day Per Hour
▶	0.0111

Job Data

2. **Throughput:** It is the no. of events happening per second.

   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?

Approach:For calculating the throughput we will be using the 7-day rolling because 7-day rolling gives us the average for all the days right from day 1 to day 7 Whereas daily metric gives us average for only that particular day itself.

For calculating the 7-day rolling daily metric average of throughput:-

1. We will be first taking the count of job_id(distinct and non-distinct) and ordering them w.r.t ds (date of interview)
2. Then by using the ROW function we will be considering the rows between 6 preceding rows and the current row
3. Then we will be taking the average of the jobs_reviewed

Query 1

```
-- Throughput
# DISCTINCT JOBS
SELECT
    ds AS date_of_review,
    jobs_reviewed,
    AVG(jobs_reviewed)
OVER
    (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS `Throughput 7 day Rolling Average`
FROM
    (SELECT ds, count(distinct job_id) AS jobs_reviewed FROM job_data GROUP BY ds ORDER BY ds)a;
```

Output

	date_of_review	jobs_reviewed	Throughput 7 day Rolling Average
▶	2020-11-25	1	1.0000
	2020-11-26	1	1.0000
	2020-11-27	1	1.0000
	2020-11-28	2	1.2500
	2020-11-29	1	1.2000
	2020-11-30	2	1.3333

Query2

```
# NON_DISTINCT JOBS
SELECT
    ds AS date_of_review,
    jobs_reviewed,
    AVG(jobs_reviewed)
OVER
    (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS `Throughput 7 day Rolling Average(Non-Distinct Jobs)`
FROM
    (SELECT ds, count( job_id) AS jobs_reviewed FROM job_data GROUP BY ds ORDER BY ds)a;
```

Output

	date_of_review	jobs_reviewed	Throughput 7 day Rolling Average(Non-Distinct Jobs)
▶	2020-11-25	1	1.0000
	2020-11-26	1	1.0000
	2020-11-27	1	1.0000
	2020-11-28	2	1.2500
	2020-11-29	1	1.2000
	2020-11-30	2	1.3333

Job Data

3) **Percentage share of each language:** Share of each language for different contents.

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

Approach: To calculate the percentage share of each language, firstly I have divided the total number of languages by the total number of rows present in table. Afterwards, I have grouped the result by Language

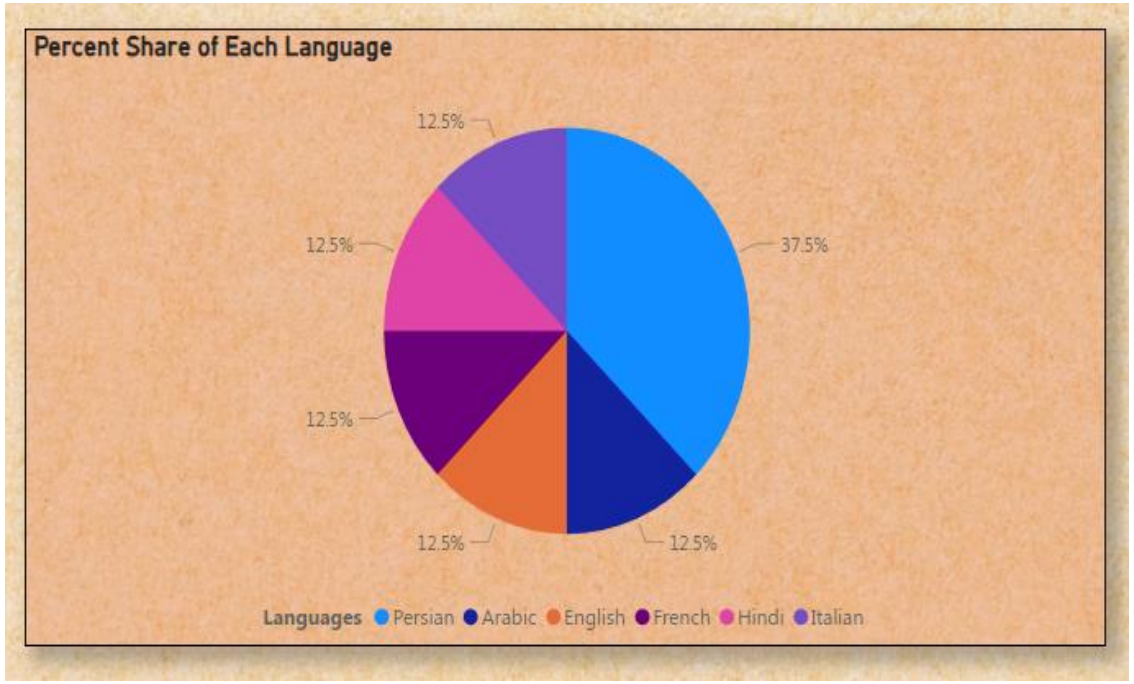
To represent the data, I have used Microsoft PowerBI to show the output in graphical form, for instance I have used Pie-Chart.

Query

```sql
-- Percentage share of each language
SELECT
    jd.job_id,
    jd.job_language,
    COUNT(jd.job_language) AS `Total Languages`,
    ROUND(((COUNT(jd.job_language)
    /
    (SELECT COUNT(*) FROM job_data))*100),2)
    AS `Percentage Share of each Language`
FROM
    job_data jd
GROUP BY
    jd.job_language;
```

Output

	job_id	job_language	Total Languages	Percentage Share of each Language
▶	21	English	1	12.50
	22	Arabic	1	12.50
	23	Persian	3	37.50
	25	Hindi	1	12.50
	11	French	1	12.50
	20	Italian	1	12.50



Job Data

4) **Duplicate rows:** Rows that have the same value present in them.

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

Approach: To view duplicate rows I have taken following steps:

i. We will use ROW_NUMBER function to find the rows having the same values.

ii. Then using PARTITION BY function OVER the column(parameter) i.e., job_id and naming the column as row_num.

iii. Then using the condition on row_num using WHERE clause, having row_num>1.\

Hence, I have obtained the desired output.

Query

```
-- Duplicate Rows
SELECT
    *
FROM
(   SELECT
    *,
    ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num
    FROM job_data
    ) a
    WHERE row_num>1;
```

Output

ds	job_id	actor_id	job_event	job_language	time_spent	org	row_num
2020-11-28	23	1005	transfer	Persian	22	D	2
2020-11-26	23	1004	skip	Persian	56	A	3

Case Study 2 (Investigating Metric Spike)

Investigating Metric Spike

1) **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.

   Task: Calculate the weekly user engagement?

Approach: Using EXTRACT and WEEK function, I have extracted the week number from occurred_at column of the events table. Then using COUNT function I have calculated the distinct users(user_id) , grouping the results by week number.

Following are the Query and the Output obtained.

```
-- User Engagement
    SELECT
    EXTRACT(WEEK FROM occurred_at) AS `Week Number`,
    COUNT(DISTINCT user_id) AS `Number of Users Engaged`
FROM
    `events`
GROUP BY  `Week Number`;
```

Week Number	Number of Users Engaged
17	740
18	1260
19	1287
20	1351
21	1299
22	1381
23	1446
24	1471
25	1459
26	1509
27	1573
28	1577
29	1607
30	1706
31	1514
32	1454
33	1438
34	1443
35	118

Investigating Metric Spike

2) **User Growth:** Amount of users growing over time for a product.

Task: Calculate the user growth for product?

Approach: Using the logic **"USER GROWTH = Number of ACTIVE USERS",** and following the below steps I obtained the desired output:

i. First, I have extracted the year and week from occurred_at column of the user table using EXTRACT, WEEK and YEAR functions.

ii. Then grouping the extracted week and year on the basis of year and week number.

iii. Then I have used the ORDER BY function to order by result on the basis of year and week number.

iv. At last, finding the Cum-Number of Active Users using the SUM, OVER, ROWS function BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

Query 1-

```sql
-- User Growth
SELECT
`Year`,
`Week`,
`Number of Active Users`,
SUM(`Number of Active Users`)OVER(ORDER BY `Year`, `Week` ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS `Cum-Number of Active Users`
FROM
(
SELECT
extract(YEAR FROM a.activated_at) AS`Year`,
extract(WEEK FROM a.activated_at) AS `Week`,
count(DISTINCT user_id) as`Number of Active Users`
FROM users a
WHERE state = 'active'
GROUP BY `Year`, `Week`
ORDER BY `Year` , `Week`
)a;
```

Query 2-

```sql
SELECT
    COUNT(*)
FROM
    users
WHERE
    state = 'active';
```

Output 2-

	COUNT(*)
▶	9381

Output 1 File Google-Drive Link:

https://drive.google.com/drive/folders/18yKoVPSGnqmgyylcMXEACWnc--zTv7_H?usp=drive_link

Investigating Metric Spike

3) **Weekly Retention:** Users getting retained weekly after signing-up for a product.

Task: Calculate the weekly retention of users-sign up cohort?

Approach: The weekly retention of users-sign up cohort can be calculated by two means i.e. either by specifying the week number (18 to 35) or for the entire column of occurred_at of the events table.

1. Firstly we will extract the week from occurred_at column using the EXTRACT, WEEK functions.
2. Then, we will select out those rows in which event_type = 'signup_flow' and event_name = 'complete_signup'.
3. Then using the LEFT JOIN we will join the two tables on the basis of user_id where event_type = 'engagement
4. Then we will use the GROUP BY function to group the output table on the basis of user_id.
5. Then we will use the ORDER BY function to order the result table on the basis of user_id.

In the next slide, is the Query written , and below it is the link to Google Drive containing the output file.

```sql
--  Weekly Retention
SELECT DISTINCT
    user_id,
    COUNT(user_id),
    SUM(CASE
        WHEN retention_week = 1 THEN 1
        ELSE 0
    END) as `Per Week Retention`
FROM
(
SELECT
    a.user_id,
    a.signup_week,
    b.engagement_week,
    b.engagement_week - a.signup_week AS retention_week
    FROM
    (
    (
    SELECT DISTINCT
    user_id,
    EXTRACT(WEEK FROM occurred_at) AS signup_week
    FROM
        `events`
    WHERE
        event_type = 'signup_flow' AND event_name = 'complete_signup'
        )a
    LEFT JOIN
    (SELECT
        DISTINCT user_id,
        EXTRACT(WEEK FROM occurred_at) as engagement_week
    FROM
        `events`
        )b
    on a.user_id = b.user_id
    )
    )d
    GROUP BY user_id
    ORDER BY user_id;
```

https://drive.google.com/drive/folders/18yKoVPSGnqmgyylcMXEACWnc--zTv7_H?usp=drive_link

Investigating Metric Spike

4) **Weekly Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.

   Task: Calculate the weekly engagement per device?

Approach: To find the weekly user engagement per device:-

1. Firstly we will extract the year_num and week_num from the occurred_at column of the events table using the extract, year and week function .
2. Then we will select those rows where event_type = 'engagement' using the WHERE clause.
3. Then by using the GROUP BY and ORDER BY function we will group and order the result on the basis of year_num, week_num and device.

Following are the Query written and the Output Google Drive Link

```sql
-- Weekly Engagement
SELECT
  extract(year from occurred_at) as year_num,
  extract(week from occurred_at) as week_num,
  device,
  COUNT(distinct user_id) as no_of_users
FROM
  `events`
where event_type = 'engagement'
GROUP by 1,2,3
order by 1,2,3;
```

https://drive.google.com/drive/folders/18yKoVPSGnqmgyylcMXEACWnc--zTv7_H?usp=drive_link

Investigating Metric Spike

5) **Email Engagement:** Users engaging with the email service.

Task: Calculate the email engagement metrics?

Approach: To find the email engagement metrics(rate) of users:-

1. At first I have categorized the action on the basis of email_sent, email_opened and email_clicked using the CASE, WHEN, THEN functions ,
2. Then I have selected the sum of category of email_opened divide by the sum of the category of email_sent and multiply the result by 100.0 and name is as email_opening_rate .
3. After that, I have selected the sum of category of email_clicked divide by the sum of the category of email_sent and multiply the result by 100.0 and name is as email_clicking_rate .
4. email_sent = ('sent_weekly_digest','sent_reengagement_email')
5. email_opened = 'email_open' ..
6. email_clicked = 'email_clickthrough'.

Following are the Query written and the Output obtained

```sql
-- Email Engagement

SELECT
  100*SUM(CASE when email_cat = 'email_opened' then 1 else 0 end)/SUM(CASE when email_cat = 'email_sent' then 1 else 0 end) as email_opening_rate,
  100*SUM(CASE when email_cat = 'email_clicked' then 1 else 0 end)/SUM(CASE when email_cat = 'email_sent' then 1 else 0 end) as email_clicking_rate
FROM
(
SELECT
  *,
  CASE
    WHEN action in ('sent_weekly_digest','sent_reengagement_email')
      then 'email_sent'
    WHEN action in ('email_open')
      then 'email_opened'
    WHEN action in ('email_clickthrough')
      then 'email_clicked'
  end as email_cat
from email_events
) a;
```

	email_opening_rate	email_clicking_rate
▶	33.5834	14.7899

Result

Hence, all the questions given as part of **Trainity Data Analytics Trainee Task 3 : Operation Analytics and Investigating Metric Spike** have been provided with answers.

In this task all the basic as well as advanced concepts related to SQL in Data Analytics have been implemented.