Project 3

Operation Analytics and Investigating Metric Spike

Nilesh Nayak
PUNE

Project Description:

Operation Analytics

Operations analytics is the procedure of gathering, examining, and deriving meaning from data in order to acquire insights into many operational facets of a company or organization. To enhance processes, increase efficiency, and make wise judgments, this method entails using data from several sources. It frequently entails the application of numerous analytical approaches, including statistical analysis, data mining, predictive modelling, and machine learning, to find patterns, trends, and opportunities for process improvement. Operational analytics is a subset of business analytics that focuses on immediate action.

Typically, data for an operations analytics project would be gathered from sources such as production systems, supply chains, customer interactions, and others. You can find bottlenecks, inefficiencies, and locations where resources can be better deployed by examining this data. This can result in better decision-making, lower costs, higher customer happiness, and overall corporate success.

Metric Spike Investigation

Metric spike investigation is the act of studying and comprehending unexpected and significant increases (or spikes) in specific metrics or key performance indicators (KPIs). These surges can occur in a variety of areas, including website traffic, sales, client queries, and any other observable statistic.

The investigation involves several steps:

- **Detection**: Identifying the metric that has experienced a sudden spike.
- **Isolation**: Determining the time and context in which the spike occurred, as well as the affected segments or areas.
- **Analysis**: Investigating the potential causes of the spike, which could include marketing campaigns, external events, technical issues, or changes in user behaviour.
- **Validation**: Confirming the accuracy of the data and the legitimacy of the spike. Sometimes, data anomalies or errors can lead to false spikes.
- **Action**: Taking appropriate actions based on the findings. This might involve optimizing resources to meet increased demand, fixing technical issues, or capitalizing on the opportunity presented by the spike.

The above project "Operations analytics and Investigation of Metric Spike" is having four datasets like job_data, users, events, email_events. Those data set is used to get answers of the questions for two case studies of job data analysis and metric spike.

First Case study is all about total job reviewed over time, language share analysis, duplicate row detection and throughput analysis.

Second Case study is about user growth analysis, weekly user engagement, user retention analysis, weekly engagement of device, email engagement.

Approach:

- Import Data: Data is given in CSV format and it is imported into SQL workbench using SQL queries;
- Understanding data: Database named '**project3**' is given in tabular format having total 7 tables with table names given below:

Table Name	Number of Rows	Number of Columns
job_data	8	7
users	9381	6
events	325255	7
email_events	90389	4

- Understanding of data-types and table schemas;
- Converting the date column to DATE type as it is read like string while importing the datasets;
- Recognition and understanding of Referential-Integrity-Constraint between tables;
- Formation of modular queries using SQL techniques;
- Merging of two or more queries to get actual answers for the given task;
- Capturing results/output.

Tech-Stack Used:

I have executed the query on MY SQL workbench installed on windows 10 operating system, more details are given below:

Software Details	
Name:	Local instance MySQL80
Host:	localhost
Port:	3306
Login User:	root
Current User:	root@localhost
SSL cipher:	SSL not used
Server	
Product:	MySQL Community Server - GPL
Version:	8.0.34
Connector	
Version:	C++ 8.1.0

I choose this MY SQL workbench because of the following reasons:

- It is available for my operating system;
- It takes less memory to install in the system;
- It allows access to data directly;
- Easy to understand and intuitive GUI;
- I can analyse multiple table at once;
- It provides cross-platform support;

Insights:

- There are 3 jobs (“skip”, “transfer”, “decision”) which is available in only 6 languages throughout 4 organisations.
- Users registered for the app is from 12 languages.
- Users activated within 180 seconds after the registration.

count(user_id)	TIMESTAMPDIFF(second, created_at, activated_at)
4447	120
4932	60
1	180
1	0

- All users are active. The state column of users table contains some Unicode character as it shows the length 7 instead of 6.
- There were total 18 unique events has been organised for engagement and signup_flow of the users, 3 types of users participated in these events from 47 different locations and they used 26 types of devices to interact in the events.
- There were total 4 types of email-events (“sent_weekly_digest”, “email_open”, “email_clickthrough”, “sent_reengagement_email”) to track engagement of the users for different events.

Result:

Case Study 1: Job Data Analysis

Jobs Reviewed Over Time: Calculate the number of jobs reviewed per hour for each day in November 2020.

Task:

Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

Query and Output:

Limit to 10000 rows

```sql
-- Case Study 1:
# A) Write an SQL query to calculate the number of jobs reviewed
-- per hour for each day in November 2020.
select ds,
    count(job_id) as no_of_jobs_reviewed,
    sum(time_spent)/3600 as time_spent_in_hours,
    count(job_id) * 3600 /sum(time_spent) as per_day_per_hour,
    avg(count(job_id) * 3600 /sum(time_spent)) over() as avg_per_hour_per_day
from job_data
where ds between '2020-11-01' and '2020-11-30'
group by ds;
```

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

ds	no_of_jobs_reviewed	time_spent_in_hours	per_day_per_hour	avg_per_hour_per_day
2020-11-30 00:00:00	2	0.0111	180.0000	126.18048618
2020-11-29 00:00:00	1	0.0056	180.0000	126.18048618
2020-11-28 00:00:00	2	0.0092	218.1818	126.18048618
2020-11-27 00:00:00	1	0.0289	34.6154	126.18048618
2020-11-26 00:00:00	1	0.0156	64.2857	126.18048618
2020-11-25 00:00:00	1	0.0125	80.0000	126.18048618

Insights and Interpretations:

- Minimum Job reviewed per hour on 27th November and maximum job reviewed per hour on 30th and 29th November.
- Average number of job reviewed per hour per day is around 126.

Throughput Analysis: Calculate the 7-day rolling average of throughput (number of events per second).

Task:

Write an SQL query to calculate the 7-day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

Query and Output:

```
# B) Write an SQL query to calculate the 7-day rolling average of throughput.
-- - Additionally, explain whether you prefer using the daily metric or
-- - the 7-day rolling average for throughput, and why?
-- SET GLOBAL sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));
with daily_rolling as (select ds,
    count(job_id) as drj,
    sum(time_spent) as drt
from job_data
where ds between '2020-11-01' and '2020-11-30'
group by ds),
seven_days_rolling as (select ds,
    sum(drj) over(order by ds rows between 6 preceding  and current row) as srj,
    sum(drt) over(order by ds rows between 6 preceding  and current row) as srt
from daily_rolling
where ds between '2020-11-01' and '2020-11-30'
order by ds
)
select daily_rolling.ds, drj/drt as daily, srj/srt as seven_day
from daily_rolling inner join seven_days_rolling
on daily_rolling.ds = seven_days_rolling.ds
order by daily_rolling.ds;
```

ds	daily	seven_day
2020-11-25 00:00:00	0.0222	0.0222
2020-11-26 00:00:00	0.0179	0.0198
2020-11-27 00:00:00	0.0096	0.0146
2020-11-28 00:00:00	0.0606	0.0210
2020-11-29 00:00:00	0.0500	0.0233
2020-11-30 00:00:00	0.0500	0.0268

Insights and Interpretations:

- It can be observed that the seven-day rolling average is better than daily rolling stats for initial days and daily rolling stats increases in later days.
- We need to focus on 7-day rolling average as it mitigates the offset time (more in daily stats).

Language Share Analysis: Calculate the percentage share of each language in the last 30 days.

Task:

Write an SQL query to calculate the percentage share of each language over the last 30 days.

Query and Output:

```sql
# C) Write an SQL query to calculate the percentage share of each language over the last 30 days.
with extra_query as
(select max(ds)  as recent_data from job_data)
select distinct language, 100*(count(job_id) over(
    partition by language rows between unbounded preceding and unbounded following) /count(*) over(
    order by ds rows between unbounded preceding and unbounded following)) as percentage_share
 from
(select * From job_data cross join extra_query
where datediff(recent_data, date(ds)) between 0 and 30) full_query;
```

language	percentage_share
Italian	12.5000
Persian	37.5000
French	12.5000
Hindi	12.5000
Arabic	12.5000
English	12.5000

Insights and Interpretations:

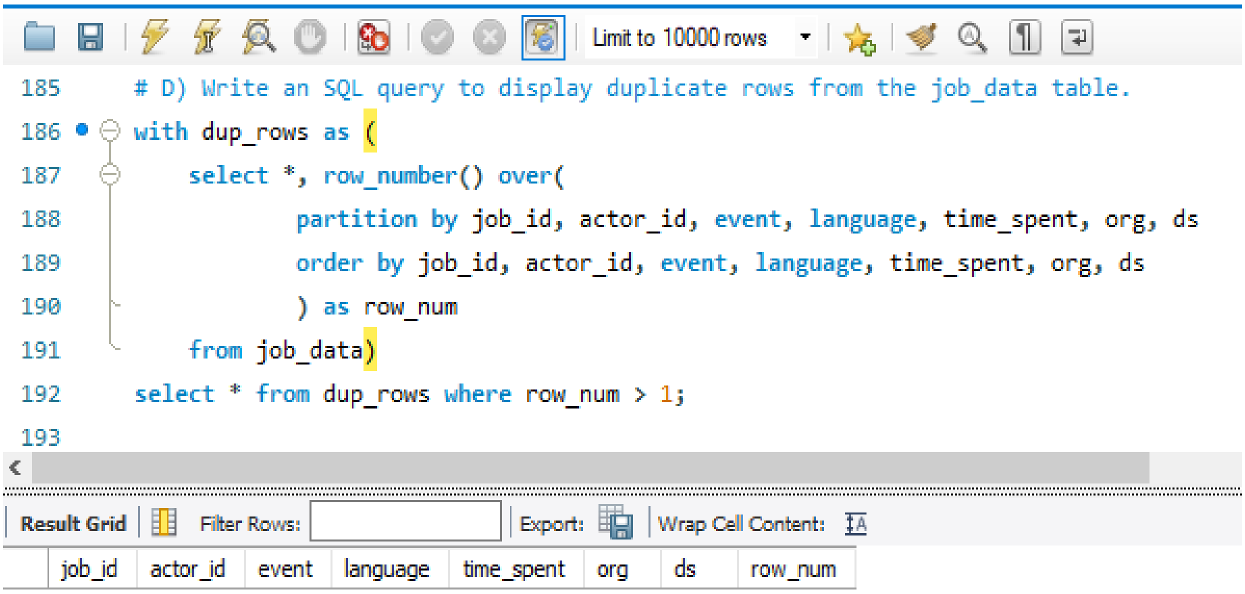
- It can be observed that the Persian language is having more share (37.5%) for job review and other five languages is having equal share.

Duplicate Rows Detection: Identify duplicate rows in the data.

Task:

Write an SQL query to display duplicate rows from the job_data table.

Query and Output:



Insights and Interpretations:

- There are no any duplicate rows in the job_data table.

Case Study 2: Investigating Metric Spike

Weekly User Engagement: Measure the activeness of users on a weekly basis.

Task:

Write an SQL query to calculate the weekly user engagement.

Query and Output:

```sql
# A) Write an SQL query to calculate the weekly user engagement
-- (Measure the activeness of users on a weekly basis).
select count(distinct(user_id)), count(distinct(event_name)) from events;

select *, user_engagement-lag(user_engagement, 1) over(order by week) as engagement_growth
from(
select year(occurred_at) as year, week(occurred_at) as week,
count(event_name) as engagement, count(user_id) as user_engagement
from events
where event_type = 'engagement'
group by week(occurred_at)) eng_table;
```

year	week	engagement	user_engagement	engagement_growth
2014	17	8019	8019	NULL
2014	18	17341	17341	9322
2014	19	17224	17224	-117
2014	20	17911	17911	687
2014	21	17151	17151	-760
2014	22	18413	18413	1262
2014	23	18280	18280	-133
2014	24	19052	19052	772
2014	25	18642	18642	-410
2014	26	19061	19061	419
2014	27	19881	19881	820
2014	28	20776	20776	895

Insights and Interpretations:

- The user engagement/activeness is fluctuating in initial weeks but later stabilizes.

User Growth Analysis: Analyze the growth of users over time for a product.

Task:

Write an SQL query to calculate the user growth for the product.

Query and Output:

Limit to 10000 rows

```sql
# B) Write an SQL query to calculate the user growth for the product
-- (Analyze the growth of users over time for a product).
select *,
quarterly_active_users-lag(quarterly_active_users, 1) over(order by year) as quarterly_user_growth
from(
select year(DATE(activated_at)) as year,
    quarter(DATE(activated_at)) as quarter,
    state,
    count(user_id) as quarterly_active_users
from users
where state like "%active%" and activated_at is not null
group by year(DATE(activated_at)), quarter(DATE(activated_at))) qrtr_usr_growth;
-- ---------------------------------------------------------------------------
```

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

year	quarter	state	quarterly_active_users	quarterly_user_growth
2013	1	active	470	NULL
2013	2	active	608	138
2013	3	active	930	322
2013	4	active	1275	345
2014	1	active	1692	417
2014	2	active	2378	686
2014	3	active	2028	-350

Insights and Interpretations:

- In the year 2013 the user base is growing for each quarter but it reduced in 3rd quarter of 2014.
- Quarter 2 of 2014 records maximum growth of users.

Weekly Retention Analysis: Analyze the retention of users on a weekly basis after signing up for a product.

Task:

Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Query and Output:

```sql
 # C) Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.
-- (Analyze the retention of users on a weekly basis after signing up for a product)
with sign_up_user as(
select distinct user_id, extract(week from occurred_at) as sign_up_week
from events
where event_type = 'signup_flow'
and event_name = 'complete_signup'),
engaged_user as(
select distinct user_id, extract(week from occurred_at) as engagement_week
from events
where event_type = 'engagement')
select sign_up_week, engagement_week, count(sign_up_user.user_id) as total_user,
count(sign_up_user.user_id)-lag(count(sign_up_user.user_id), 1)
over(partition by sign_up_week order by engagement_week) as retained_user
from sign_up_user
left join
engaged_user
on sign_up_user.user_id = engaged_user.user_id
group by sign_up_week, engagement_week
order by sign_up_week, engagement_week;
```

sign_up_week	engagement_week	total_user	retained_user
17	17	72	NULL
17	18	59	-13
17	19	24	-35
17	20	16	-8
17	21	11	-5
17	22	16	5
17	23	11	-5
17	24	9	-2
17	25	6	-3
17	26	8	2
17	27	8	0
17	28	8	0
17	29	7	-1
17	30	9	2
17	31	6	-3
17	32	5	-1
17	33	1	-4
17	34	2	1
18	18	163	NULL
18	19	114	-49

Insights and Interpretations:

- Total 179 rows have been reflected and it clearly shows that there are negative trends in user retention for weekly cohort. Though very less number of times user retention is positive.

Weekly Engagement Per Device: Measure the activeness of users on a weekly basis per device.

Task:

Write an SQL query to calculate the weekly engagement per device.

Query and Output:

```
# D) Write an SQL query to calculate the weekly engagement per device.
-- (Measure the activeness of users on a weekly basis per device)
select occurred_at, extract(year from occurred_at) as year,
extract(week from occurred_at) as week,
device, count(distinct(user_id)) as total_user
from events
where event_type = 'engagement'
group by device, extract(week from occurred_at)
order by extract(year from occurred_at), extract(week from occurred_at);
```

occurred_at	year	week	device	total_user
2014-05-01 12:13:00	2014	17	ipad mini	19
2014-05-01 04:44:00	2014	17	iphone 4s	21
2014-05-01 10:00:00	2014	17	iphone 5	65
2014-05-02 11:41:00	2014	17	iphone 5s	42
2014-05-03 11:23:00	2014	17	kindle fire	6
2014-05-02 09:28:00	2014	17	lenovo thinkpad	86
2014-05-01 15:21:00	2014	17	mac mini	6
2014-05-02 15:33:00	2014	17	macbook air	54
2014-05-02 07:20:00	2014	17	macbook pro	143
2014-05-02 18:47:00	2014	17	nexus 10	16
2014-05-02 13:09:00	2014	17	nexus 5	40
2014-05-02 10:12:00	2014	17	nexus 7	18
2014-05-01 08:39:00	2014	17	nokia lumia 635	17
2014-05-01 15:12:00	2014	17	samsumg galaxy tablet	8
2014-05-03 14:01:00	2014	17	samsung galaxy note	7

Insights and Interpretations:

- Total 491 rows have been reflected and it is observed that each week the macbook pro user was maximum to interact in the events. It seems macbook pro is very popular among users.

Email Engagement Analysis: Analyze how users are engaging with the email service.

Task:

Write an SQL query to calculate the email engagement metrics.

Query and Output:

Limit to 10000 rows

```sql
# E) Write an SQL query to calculate the email engagement metrics.
-- (Analyze how users are engaging with the email service)
set @email_click=(select count(user_id) from email_events where action = 'email_clickthrough');
set @email_opened=(select count(user_id) from email_events where action = 'email_open');
set @email_sent=(select count(user_id) from email_events where action = 'sent_reengagement_email')+
                (select count(user_id) from email_events where action = 'sent_weekly_digest');

select @email_click*100/@email_sent as percent_email_clicked,
       @email_opened*100/@email_sent as percent_email_opened;

```

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

percent_email_clicked	percent_email_opened
14.7899	33.5834

Insights and Interpretations:

- Total 60920 email was sent out of which 9010 (14.7899%) users clicked the sent email and 20459 (33.5834%) users opened the email.
- It is good strategy to send email to users for promotional activities as 33.6% users clicks and reads the email-content.

Conclusion:

The "Two case studies" of the project is carried out utilizing SQL queries on MYSQL workbench. This initiative has tracked users' insights and examined their engagements with the app.

By accomplishing this assignment, I learned about SQL advanced queries and query optimization. I thoroughly comprehended the data and learned how to link it using a join query. I investigated strategies for working with SQL's date type capability. I've learned about window functions and common table expressions(CTE). I understood about the cohort analysis to find the retention of the users thorough the time given or size based or segment based.

Drive Link:

https://drive.google.com/drive/folders/1vSQW5fLT2B8L5paGd2faZkN1NX9mRW_W?usp=sharing