

Operation Analytics and Investigating Metric Spike

Name: Tanmay Dangat
Project 3

1. **Project Description:** As per the instructions and report, I have assigned a task to perform the data analysis of the **Operation Analytics and Investigating Metric Spike**. We have to work on the provided dataset and collect the useful insights that can help the marketing team for further campaigns. The primary focus is on optimizing workflows ,enhance automation, and predicting the company's overall growth or decline.
2. **Approach :**To perform all the queries and complete the given task. According to the instructions . In MYSQL Workbench, execute all the queries , analyze the uncover patterns, trends,anomalies it and collect the useful insights out of it
3. **Tech Stack used :** To perform the queries I have used MYSQL workbench because it is a powerful tool for database development, management, and administration specifically designed for MySQL databases , and Google docs to make the analysis of the project
4. **Insights:** As a beginner, it helped me to understand how the complex queries work and how to understand the business and that insights actually works.
5. **Results:** Following are all the executed queries with the Output

A) Case Study 1: Job Data Analysis

- **Jobs Reviewed Over Time:**
 - Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.
 - Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

●

Syntax:

```
select avg(t) as 'avg jobs reviewed per day per hour',  
avg (p) as 'avg jobs reviewed per day per second'  
from(select ds,  
((count(job_id)*3600)/sum(time_spend)) as t,  
((count(job_id))/sum(time_spend)) as p  
from job_data  
where month (ds)=11  
group by ds) a;
```

Insights of the below query : The number of jobs reviewed per hour per day in November 2020 varies, with higher activity on some days and lower activity on others.



Navigator

SCHEMAS

Filter objects

- ig_clone
 - Tables
 - comments
 - follows
 - likes
 - photo_tags
 - photos
 - tags
 - users
 - Views
 - Stored Procedures
 - Functions
- project_3
 - Tables
 - job_data
 - Columns
 - ds
 - job_id
 - actor_id
 - event
 - language
 - time_spend
 - org
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - sys
 - temp_db

Administration Schemas

Information

Query 1 x

Limit to 50000 rows

```
161
162 #1 Task reviewed over time
163
164 • select* from job_data;
165 • select avg(t) as 'avg jobs reviewd per day per hour',
166       avg (p) as 'avg jobs reviewed per day per second'
167   from(select ds,
168        ((count(job_id)*3600)/sum(time_spend)) as t,
169        ((count(job_id))/sum(time_spend)) as p
170       from job_data
171      where month (ds)=11
172     group by ds) a;
```

Result Grid Filter Rows: Export: Wrap Cell Content: [F6](#)

	avg jobs reviewd per day per hour	avg jobs reviewed per day per second
▶	126.18048333	0.03505000

Result 2 x

B.Throughput Analysis:

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

Syntax:

```
select ROUND(count(event)/sum(time_spend), 2) as "weekly throughput" From  
job_data;
```

```
select ds as dates, round(count(event)/sum(time_spend), 2) as "Daily  
throughput" From job_data
```

```
group by ds order by ds;
```

Insights of the below query: Insights: The 7-day rolling average of throughput provides a smoothed view of the data, allowing you to observe trends over time without being affected by daily fluctuations.

Continue using the 7-day rolling average for throughput analysis, as it provides a more stable representation of performance trends. This can help in identifying long-term patterns and making more informed decisions.



Query 1 x



```
171 where month (ds)=11
172 group by ds) a;
173
174 #2 Throughput analysis
175
176 • select ROUND(count(event)/sum(time_spend), 2) as "weekly throughput" From job_data;
177 • select ds as dates, round(count(event)/sum(time_spend), 2) as "Daily throughput" From job_data
178 group by ds order by ds;
179
180
181
182
183
184
185
186
187
188
189
190
```

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

	dates	Daily throughput
▶	2020-11-25	0.02
	2020-11-26	0.02
	2020-11-27	0.01
	2020-11-28	0.06
	2020-11-29	0.05
	2020-11-30	0.05

Result 4 x

C. Language Share Analysis:

- Objective: Calculate the percentage share of each language in the last 30 days.
- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

Syntax:

```
select language as languages, Round (100* count(*)/total, 2) AS Percentage,  
sub.total
```

```
from job_data
```

```
Cross Join(select count(*) as total from job_data) as sub
```

```
group by language, sub.total;
```

Insights of the below query: The language distribution in the last 30 days is relatively balanced, with Persian having the highest share.

Consider investing in language-specific content or features to enhance user engagement in languages with lower shares.



Query 1 x



181 #3 language share analysis

182

```
183 • select language as languages, Round(100* count(*)/total,2) AS Percentage, sub.total  
184 from job_data
```

```
185 Cross Join( Select count(*) as total from job_data) as sub  
186 group by language, sub.total;
```

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

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

	languages	Percentage	total
▶	English	12.50	56
	Arabic	12.50	56
	Persian	37.50	56
	Hindi	12.50	56
	French	12.50	56
	Italian	12.50	56

Result 5 x

D.Duplicate Rows Detection:

- Objective: Identify duplicate rows in the data.
- Your Task: Write an SQL query to display duplicate rows from the `job_data` table.

Syntax:

```
Select actor_id, Count(*) as Duplicate from job_data  
group by actor_id having count(*)>1;
```

Insights of the below query:

There are total of 7 duplicate rows in the table, and the actor_id 1003 has 14 duplicate rows



Query 1 x



```
189      #4 Duplicate rows detection
190
191 •      select actor_id, count(*) as Duplicate from job_data
192      group by actor_id having count(*)>1;
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	actor_id	Duplicate
▶	1001	7
	1006	7
	1003	14
	1005	7
	1002	7
	107	7
	1004	7

Result 6 x

Case Study 2: Investigating Metric Spike

Tasks:

1. Weekly User Engagement:

- Objective: Measure the activeness of users on a weekly basis.
- Your Task: Write an SQL query to calculate the weekly user engagement.

Syntax:

```
select extract (week from occurred_at) as weeks, count(distinct user_id) as  
no_of_users from events where event_type="engagement"
```

```
group by weeks order by weeks;
```

Insights of the below query:

User engagement seems to have peaked around week 30 and has shown some fluctuations over the observed period.

Look for patterns related to content updates, marketing campaigns, or any external events that might have influenced user behavior.

Use these insights to plan future engagement strategies



Navigator

SCHEMAS

Filter objects

- ig_clone
- project_3
 - Tables
 - events
 - emailevents
 - job_data
 - user
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- sys
- temp_db

```
84
85
86
87
88 #1. calc the weekly user engagement
89 • select * from events;
90 • select extract(week from occurred_at) as weeks,
91 count(distinct user_id) as no_of_users from events
92 where event_type="engagement"
93 group by weeks order by weeks;
94
95
96
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	weeks	no_of_users
▶	17	663
	18	1068
	19	1113
	20	1154
	21	1121
	22	1186
	23	1232
	24	1275
	25	1264
	26	1302
	27	1372
	28	1365
	29	1376
	30	1467
	31	1299
	32	1225
	33	1225

Result 6 x

Administration Schemas

Information

No object selected

2. Weekly Retention Analysis:

- a. Objective: Analyze the retention of users on a weekly basis after signing up for a product.
- b. Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Syntax:

```
select extract(week from occurred_at) as weeks,  
count(distinct user_id) as no_of_users from events  
where event_type="signup_flow" and event_name="complete_signup"  
group by weeks order by weeks;
```

Insight of the below query:

Weekly user retention shows a gradual decline over time.

Focus on improving user retention strategies. Identify key touchpoints in the user journey where users might be dropping off and work on enhancing user experience, engagement, and value during those stages.



SCHEMAS

Filter objects

- ig_clone
- project_3
 - Tables
 - events
 - emailevents
 - job_data
 - user
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- sys
- temp_db

```
141
142
143
144
145
146
147 #3. calc the weekly retention of users-signup cohort
148 • select * from events;
149 • select extract(week from occurred_at) as weeks,
150 count(distinct user_id) as no_of_users from events
151 where event_type="signup_flow" and event_name="complete_signup"
152 group by weeks order by weeks;
153
```

	weeks	no_of_users
▶	17	72
	18	163
	19	185
	20	176
	21	183
	22	196
	23	196
	24	229
	25	207
	26	201
	27	222
	28	215
	29	221
	30	238
	31	193
	32	245
	33	261

3.Weekly Engagement Per Device:

- Objective: Measure the activeness of users on a weekly basis per device.
- Your Task: Write an SQL query to calculate the weekly engagement per device.

Syntax:

```
select device, extract (week from occurred_at) as weeks, count(distinct  
user_id) as no_of_users from events
```

```
where event_type="engagement"
```

```
group by device, weeks order by weeks;
```

Insights of the below query:

Engagement varies across different devices and weeks. Some devices consistently show higher engagement than others.

Consider optimizing the user experience for devices that show lower engagement. Additionally, monitor device trends over time to adapt your strategies and prioritize user engagement on devices with the highest potential.

*This is sample output of 22 rows only. There 491 rows returned to the query, which could not be accommodated in single page.

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- ig_clone
- project_3**
 - Tables
 - events
 - emailevents
 - job_data
 - user
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- sys
- temp_db

153

154

155

156

157

158

159 #4. calc the weekly user engagement per device

160 • select * from events;

161 • select device, extract(week from occurred_at) as weeks,

162 count(distinct user_id) as no_of_users from events

163 where event_type="engagement"

164 group by device, weeks order by weeks;

165

Result Grid

Filter Rows:

Export: Wrap Cell Content:

	device	weeks	no_of_users
▶	acer aspire desktop	17	9
	acer aspire notebook	17	20
	amazon fire phone	17	4
	asus chromebook	17	21
	dell inspiron desktop	17	18
	dell inspiron notebook	17	46
	hp pavilion desktop	17	14
	htc one	17	16
	ipad air	17	27
	ipad mini	17	19
	iphone 4s	17	21
	iphone 5	17	65
	iphone 5s	17	42
	kindle fire	17	6
	lenovo thinkpad	17	86
	mac mini	17	6
	macbook air	17	54

Result 10 x

Output

Administration Schemas

Information

No object selected

4.Email Engagement Analysis:

- Objective: Analyze how users are engaging with the email service.
- Your Task: Write an SQL query to calculate the email engagement metrics.

Syntax:

```
select * from email_events_table;
```

```
select count(action) as action_count, action from email_events_table group  
by action;
```

```
select
```

```
(sum(case when
```

```
email_category="email_opened" then 1 else 0 end)/sum(case when  
email_category="email_sent" then 1 else 0 end))*100 as open_rate,
```

```
(sum(case when
```

```
email_category="email_clickthrough" then 1 else 0 end)/sum(case when  
email_category="email_sent" then 1 else 0 end))*100 as click_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_clickthrough")

end as email_category

from email_events_table) as alias;

Insights of the below query: The users are engaging as the open rate of email is 33.58% per user and the click of those is 14.78 % per user.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'project_3' with tables like 'events', 'emailevents', 'job_data', and 'user'. The main editor window contains a SQL query labeled '#5. calc the users email engagement metrics'. The query calculates the open rate and click rate for email events. Below the query, the 'Result Grid' shows the output of the query, which is a single row with two columns: 'open_rate' and 'click_rate'.

```
171 #5. calc the users email engagement metrics
172 select * from emailevents;
173 select count(action) as action_count, action from emailevents group by action;
174 select
175     (sum(case when
176         email_category="email_opened" then 1 else 0 end)/sum(case when email_category="email_sent" then 1 else 0 end))*100 as open_rate,
177     (sum(case when
178         email_category="email_clickthrough" then 1 else 0 end)/sum(case when email_category="email_sent" then 1 else 0 end))*100 as click_rate
179 from (
180     select *,
181     case
182         when action in ("sent_weekly_digest", "sent_reengagement_email") then ("email_sent")
183         when action in ("email_open") then ("email_opened")
184         when action in ("email_clickthrough") then ("email_clickthrough")
185     end as email_category
186 from emailevents) as alias;
```

open_rate	click_rate
33.5834	14.7899

Drive link:

Summary of the project:

During this project , I have learned a lot about . It helped me to understand the analysis , it provides useful and complex queries that made me understand more about the project. It also helped me to learn professionally based on what I have learned. Overall, the project was really beneficial. It improved my skills, gave me useful information, and helped me make better decisions