PROJECT 3

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

A MADHAVA VARMA

DESCRIPTION

The project is to make Operation Analytics as a Lead Data Analyst of a company. The primary purpose is to obtain insights like daily user engagement, drop in sales etc. that can be used to make strategic decisions for scaling up the business.

TECH-STACK USED

MySQL Workbench 8.0 CE and SQL are used for the project. They are used as MySQL is known for its performance, especially in cases where we handle large volumes of data, which is required for analyzing user interactions on a large platform like Instagram.

ANALYSIS ON

Case Study 1: Job Data Analysis

a. Jobs Reviewed Over Time

b. Throughput Analysis

c. Language Share Analysis

d. Duplicate Rows Detection

Case Study 2: Investigating Metric Spike

a. Weekly User Engagement

b. User Growth Analysis

c. Weekly Retention Analysis

d. Weekly Engagement Per Device

e. Email Engagement Analysis

CASE STUDY 1: JOB DATA ANALYSIS

(a)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.

STEPS:

1. We use the data provided in the **job_id** column from the **job_data** table to get the number of jobs reviewed per hour for each day in November 2020.
2. We get the **count** of total jobs from **distinct job_id** and divide it by **(24*30)** from **job_data** table**.**

CASE STUDY 1: JOB DATA ANALYSIS

(a)Jobs Reviewed Over Time

QUERY:

```
select count (distinct job_id) / (30*24)
as no_of_jobs_reviewed_per_day
from job_data;
```

RESULT

no_of_jobs_reviewed_per_day
0.0083

INSIGHT:

The jobs reviewed per day in November,2020 is about 0.0083 which is very low.

CASE STUDY 1: JOB DATA ANALYSIS

(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 throughput, and why.

Description: I prefer using 7-day rolling for calculating the throughput 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.

STEPS:

1. We need to select **username** column from the **user**s table.
2. Next, we need to use **left join** function to left join **photos** and **users** table **on users.id = photos.user_id** as both of them have common data.
3. Now we have to get the data where **photos.id** is **NULL** from the **users** table.

CASE STUDY 1: JOB DATA ANALYSIS

(b) Throughput Analysis

QUERY:

```
SELECT ds as review_date, jobs_reviewed,
AVG(jobs_reviewed) OVER
(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW) as throughput
FROM( SELECT ds, COUNT(DISTINCT job_id) as
jobs_reviewed
FROM job_data
GROUP BY ds ORDER BY ds) a;
```

RESULT:

review_date	jobs_reviewed	throughput
2020-11-25 00:00:00	1	1.0000
2020-11-26 00:00:00	1	1.0000
2020-11-27 00:00:00	1	1.0000
2020-11-28 00:00:00	2	1.2500
2020-11-29 00:00:00	1	1.2000
2020-11-30 00:00:00	2	1.3333

INSIGHT:

The throughput obtained using 7 day rolling is about 1.

CASE STUDY 1: JOB DATA ANALYSIS

(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.

STEPS:

1. We first get the number of jobs per language using the language and job_id columns from the job_data table.
2. Now, we caculate the percentage share of each language.

CASE STUDY 1: JOB DATA ANALYSIS

(c) Language Share Analysis

QUERY:

```
with clang as
(SELECT language, count(job_id) AS
no_of_jobs
FROM job_data
GROUP BY language)
SELECT *, ROUND(no_of_jobs*100/(SELECT
sum(no_of_jobs) FROM clang),2) AS
Percentage
FROM clang;
```

RESULT:

language	no_of_jobs	Percentage
English	1	12.50
Arabic	1	12.50
Persian	3	37.50
Hindi	1	12.50
French	1	12.50
Italian	1	12.50

INSIGHT:

There are a total of six languages involved. All the languages have equal share except for Persian whose share is thrice the share of anyone of the others'.

CASE STUDY 1: JOB DATA ANALYSIS

(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.

STEPS:

1. We will use **ROW_NUM()** function to get the number of duplicate rows present.
2. We use the **job_id** column as a parameter for the **ROW_NUM() function.**
3. Next, using **WHERE** function we will conider the count of rows which are more than 1 to get the rows that are repeated i.e. duplicate rows.

CASE STUDY 1: JOB DATA ANALYSIS

(d) Duplicate Rows Detection

QUERY:

```
SELECT *FROM
(SELECT *, ROW_NUMBER()
OVER (PARTITION BY job_id) AS nrow
FROM job_data) a
WHERE nrow>1;
```

INSIGHT:

Hence, the following are the most used hashtags on the Instagram platform.

RESULT:

job_id	actor_id	evnt	language	time_spent	org	ds	row_num
23	1005	transfer	Persian	22	D	2020-11-28 00:00:00	2
23	1004	skip	Persian	56	A	2020-11-26 00:00:00	3

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(a) 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.

STEPS:

1. We will use **select** statement, **week** function in **occurred_at** column to **extract** number of weeks and **count** function in **distinct user_id** column to get number of users from **events** table.
2. Now we use, **group by** clause to get weekly user engagement.

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(a) Weekly User Engagement:

QUERY:

```
select week(occurred_at) as nweek,
count(distinct user_id) as users_count
from events
group by nweek;
```

INSIGHT:

The number of users engaging are over 1000 almost every week.

RESULT:

nweek	users_count
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
34	1204
35	104

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(b) User Growth Analysis

Objective: Analyze the growth of users over time for a product.
Your Task: Write an SQL query to calculate the user growth for the product.

STEPS:

1. We will use extract to extract year and week from activated_at column from users table.
2. Using group by clause we will group extracted year and week on the basis of year and week number.
3. Then we will use order by to sort the output based on extracted year and week
4. We will use sum, over and row function between unbounded preceding and current row to find cumm_active_users.

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(b) User Growth Analysis

QUERY:

```
select year, nweeks, active_users, sum(active_users) over(order by
year, nweeks ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as
total_ausers
from (select extract(year from activated_at) as year,
extract(week from activated_at) as nweeks,
count(distinct user_id) as active_users
from users
group by year, nweeks
order by year, nweeks) a;
```

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(b) User Growth Analysis

RESULT:

year	nweeks	active_users	total_ausers
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724
2013	19	57	781
2013	20	39	820
2013	21	49	869
2013	22	54	923
2013	23	50	973

year	nweeks	active_users	total_ausers
2013	48	97	2894
2013	49	116	3010
2013	50	124	3134
2013	51	102	3236
2013	52	47	3283
2014	0	83	3366
2014	1	126	3492
2014	2	109	3601
2014	3	113	3714
2014	4	130	3844
2014	5	133	3977
2014	6	135	4112
2014	7	125	4237
2014	8	129	4366
2014	9	133	4499
2014	10	154	4653
2014	11	130	4783
2014	12	148	4931
2014	13	167	5098
2014	14	162	5260
2014	15	164	5424
2014	16	179	5603
2014	17	170	5773
2014	18	163	5936

year	nweeks	active_users	total_ausers
2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813
2013	37	85	1898
2013	38	90	1988
2013	39	84	2072
2013	40	87	2159
2013	41	73	2232
2013	42	99	2331
2013	43	89	2420
2013	44	96	2516
2013	45	91	2607
2013	46	88	2695
2013	47	102	2797

2014	19	185	6121
2014	20	176	6297
2014	21	183	6480
2014	22	196	6676
2014	23	196	6872
2014	24	229	7101
2014	25	207	7308
2014	26	201	7509
2014	27	222	7731
2014	28	215	7946
2014	29	221	8167
2014	30	238	8405
2014	31	193	8598
2014	32	245	8843
2014	33	261	9104
2014	34	259	9363
2014	35	18	9381

INSIGHT: There are in total 9381 active users from 1st week of 2013 to the 35th week of 2014 and the users increased by over 400 times the initial user count.

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(c) Weekly Retention Analysis

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

STEPS:

1. First we will use **extract** function to extract week from **occurred_at** column from **events** table.
2. Then we will select the rows in which **event_type** = '**flow**' and **event_name** = '**signup**'.
3. After that we will use **left join** on **user_id** to join the tables in which **event_type** = '**engagement**'.
4. Using **group by** clause in **user_id** we will get weekly retention for each user.
5. Then we will use **order by** to get the output on the basis of **user_id**.

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(c) Weekly Retention Analysis

QUERY:

```
Select distinct user_id, count(user_id) as num_user,
sum(case when retention_week = 1 then 1 else 0 end) as weekly_rtntion
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
 where event_type = 'engagement') b
 on a.user_id = b.user_id)) d
group by user_id
order by user_id;
```

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(c) Weekly Retention Analysis

RESULT:

CLICK_HERE!!!
Weekly_Retention_Analysis_Result_drive_link

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(d) Weekly Engagement

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.

STEPS:

1. We will **extract year** and **week** from **occurred_at** column from **events** table.
2. Then we will select **device** column and use **count** function to get number of users.
3. Next, we use **where** clause we will select rows where **event_type**='**engagement**'.
4. We will use **group by** and **order by** function to group and order the output based on **year**, **nweeks** and **device**.

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(d) Weekly Engagement

QUERY:

```
select year(occurred_at) as year, week(occurred_at) as
nweeks, device,
count(distinct user_id) as user_count
from events
where event_type= ‘engagement’
group by 1,2,3
order by 1,2,3;
```

RESULT:

Click_HERE!!
Weekly_Engagement_Result_drive_link

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(e) 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.

STEPS:

1. First we will categorize the action into '**email_opened**', '**email_sent**', '**email_clicked**' using **when**, **case**, **then** functions.
2. We will **divide sum** of category '**email_opened**' and **sum** of category '**email_sent**' and **multiply by 100** as **opening_rate**.
3. Then we will **divide sum** of category '**email_clicked**' and **sum** of category '**email_sent**' and **multiply** by **100** as **clicking_rate**.
4. Then we categorize as:

   email_opened = ('email_open')
   email_sent = ('sent_weekly_digest','sent_reengagement_email')
   email_clicked = ('email_clickthrough')

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(e) Email Engagement Analysis:

QUERY:

```
Select 100*SUM(CASE when email_action = 'email_opened' then 1 else 0 end) / SUM(CASE when email_action =
'email_sent' then 1 else 0 end) as opening_rate,
100*SUM(CASE when email_action = 'email_clicked' then 1 else 0 end) / SUM(CASE when email_action = 'email_sent'
then 1 else 0 end) as clicking_rate
from
(select *,
CASE WHEN action in ('email_open')
then 'email_opened'
WHEN action in ('sent_weekly_digest','sent_reengagement_email')
then 'email_sent'
WHEN action in ('email_clickthrough')
then 'email_clicked'
end as email_action
from email_events) a;
```

CASE STUDY-2 INVESTIGATING METRIC SPIKE

(e) Email Engagement Analysis:

RESULT:

	opening_rate	dicking_rate
▶	33.5834	14.7899

INSIGHT:

The opening rate of emails sent is about 33% and clicking rate is almost 15%.

Working on this project helped me understand the advanced level concepts of SQL. It helped me gain experience on handling MySql effectively and how Data Analytics is implemented using it in the real world by a company and how I should approach the data provided as a Data Analyst to get valuable insights fot the company’s growth. It also helped me gain experience in handling large sets of data.

THANK YOU.