PROJECT 3: OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

Project Description:

- In this project our role is of lead data analyst at a company like Microsoft we have to perform operation analytics on the given data , our goal is to derive valuable insights from the given data ,these analysis helps identify areas for improvement within the company.
- One of the key aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales

CASE STUDY 1: JOB DATA ANALYSIS

First of all we create a database named job_data then we create a table named jobs after the table is created we export the data from csv file into MYSQL.

In case study 1 we will be working with a table named jobs

TASK (A): JOBS REVIEWED OVER TIME

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

Code:

```
SELECT job_id,ds,
COUNT(*) AS jobs_reviewed,
SUM(time_spent) AS total_time_spent,
COUNT(*) / (SUM(time_spent) / 3600) AS jobs_review_per_hour
FROM jobs
GROUP BY job_id , ds
order by ds;
```

MYSQL QUERIES USED:

SELECT: Specifies the columns from where you want to retrieve from the table.

FROM: Specifies the table from which you want to retrieve data.

COUNT(*):This counts the no.of rows in the group

COUNT(*)/SUM(time_spent)/3600): this division calculates the jobs reviewed per hour.

GROUP BY: It is used to group rows that have the same values

ORDER BY: It specifies the columns by which the result set should be sorted

OUTPUT:

job_id	ds	jobs_reviewed	total_time_spent	jobs_review_per_hour
20	11/25/2020	1	45	80.0000
23	11/26/2020	1	56	64.2857
11	11/27/2020	1	104	34.6154
23	11/28/2020	1	22	163.6364
25	11/28/2020	1	11	327.2728
23	11/29/2020	1	20	180.0000
21	11/30/2020	1	15	240.0000
22	11/30/2020	1	25	144.0000

- INSIGHTS:
- Job_id 11 has highest total time spent and jobs reviewed per hour is least when compared to all others.
- Job_id 25 has least total time spent and jobs reviewed per hour is highest when compared to all others.
- Total time spent and jobs review per hour are inversely proportional to each other.
- On 28th November no.of jobs review per hour are highest and on 27th November lowest no.of jobs review per hour

TASK B:THROUGHPUT ANALYSIS

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.

CODE:

```
SELECT job_id,
AVG(SUM(time_spent))
OVER (ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
AS rolling_avg_throughput
FROM jobs
GROUP BY job_id
ORDER BY job_id;
```

- AVG() function: It is used to calculate average value of a column from a table.
- Over: It is used with window functions to perform calculations on a specific set of rows related to the current row.
- explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why?
- I will choose both based on the nature of the data, analytical goals considering the goals of our analysis we will choose if we want long term trends and overall performance then rolling average might be more appropriate, for short term decision-making or identifying daily patterns , a daily metric could be more useful.

OUTPUT:

job_id	rolling_avg_throughput
11	50.6000
20	49.6667
21	15.0000
22	20.0000
23	46.0000
25	37.2500

- INSIGHTS:
  job_id 11 has the highest rolling avg throughput
- Job_is 21 has the lowest rolling avg throughput

- TASK 3: LANGUAGE SHARE ANALYSIS
- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.
- CODE:

```
select language,
count(*) as lang_count,
total_count_per_lang,
((lang_count/total_count_per_lang)*100)*30 as percent_lang_share
from(
select language,
count(*) as lang_count,
sum(count(*)) over() as total_count_per_lang
from jobs
 group by language    ) as total_lang_count
group by language,total_count_per_lang
order by language;
```

- **COUNT(*) AS lang_count**: Counts the number of jobs for each language.
- SUM(COUNT(*)) OVER () AS total_count_per_lang: Uses the window function SUM over an empty window (denoted by OVER ()) to calculate the total count of jobs for all languages. This is done for each row to get the total count for all languages.
- ((COUNT(*) / total_count_per_lang) * 100) * 30 AS percent_lang_share:
- It is used to calculate percentage share of language and it is multiplied by 30 to find share of each language for last 30 days
- We have used subquery to solve this task.

language	lang_count	total_count_per_lang	percent_lang_share
Arabic	1	8	375.0000
English	1	8	375.0000
French	1	8	375.0000
Hindi	1	8	375.0000
Italian	1	8	375.0000
Persian	1	8	1125.0000

INSIGHTS:
Persian language has the highest percentage share.

- TASK 4: Duplicate Rows Detection
- Your Task: Write an SQL query to display duplicate rows from the jobs table.

CODE:

```
select job_id,language,
count(*) as duplicate_count
from jobs
group by job_id,language
having count(*)>1;
```

- Having is used to apply conditions to the result set after grouping has been performed

INSIGHTS:
Persian language and job_id 23 are the duplicates which are repeated 3 times

OUTPUT:

job_id	language	duplicate_count
23	Persian	3

CASE STUDY 2:
INVESTIGATING METRIC SPIKE

In this case we will be dealing with three tables, we will create a database named metric, then we will export data from csv files to the tables in mysql.

Task A: Weekly User Engagement

Your Task: Write an SQL query to calculate the weekly user engagement.

Code:

```
select extract(week from occured_at) as week_num,
count(DISTINCT user_id) as weekly_user_engagement
from events
where event_type='engagement'
group by week_num
order by week_num;
```

- Extract function: It is used to retrieve specific components such as year,month,day,etc.. From a date or timestamp.
- Count function: It is used to count the number of rows that meet a specified condition in a table.
- Distinct : It is used to retrieve unique values from a specified column.

Output:

week_num	weekly_user_engagement
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302

week_num	weekly_user_engagement
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

Insights:
Week number 30 has the highest weekly user engagement
Week number 35 has the lowest weekly user engagement

- Task B: User Growth Analysis
- Your Task: Write an SQL query to calculate the user growth for the product.
- Code:

```
select year,week_num,user_num,sum(user_num)
 over  (order by year,week_num) as user_data
from(
select extract(year from created_at) as year,
extract(week from created_at) as week_num,
count(distinct user_id) as user_num
from users
where not state='active’
group by year,week_num
order by year,week_num) as table_data;
```

year	week_num	user_num	user_data
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

year	week_num	user_num	user_data
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
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

year	week_num	user_num	user_data
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

year	week_num	user_num	user_data
2013	47	102	2797
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

year	week_num	user_num	user_data
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
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

year	week_num	user_num	user_data
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

- Insights:
- From 2013 to 2014 there is continuous growth in user number and user data
- In the last week of 2014 the user number is very low but the user data is high
- On week 33 of 2014 it has highest number of users
- On week 35 of 2014 it has lowest number of users

◈ Task C: Weekly Retention Analysis

Code:

```
WITH cte1 AS (    SELECT DISTINCT       user_id,
EXTRACT(WEEK FROM occured_at) AS sign_up_week
FROM  events
WHERE  event_type = 'signup_flow'
AND event_name = 'complete_signup'
AND EXTRACT(WEEK FROM occured_at) = 18),
cte2 AS (    SELECT DISTINCT       user_id,
EXTRACT(WEEK FROM occured_at) AS engagement_week
FROM   events
WHERE  event_type = 'engagement')
```

```
SELECT   COUNT(DISTINCT user_id) AS total_engaged_users,
SUM(CASE WHEN retention_week = 0 THEN 1 ELSE 0 END) AS retained_users
FROM (
SELECT
a.user_id,
a.sign_up_week,
b.engagement_week,
COALESCE(b.engagement_week - a.sign_up_week, 0) AS retention_week
FROM   cte1 a
LEFT JOIN    cte2 b ON a.user_id = b.user_id
ORDER BY  a.user_id) sub_quer;
```

- With clause : It is used to define Common Table Expression
- Cte1: Common Table Expression 1
- Extract function: It is used to retrieve specific components such as year,month,day,etc.. From a date or timestamp.
- COALESCE function: It is used to return the first non-null expression among its arguments.
- Cte2: common table expression 2.

total_engaged_users	retained_users
163	163

◈ Task D: Weekly Engagement Per Device

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

Code:

```
with cte as (select extract(year from occured_at)||'-'||
extract(week from occured_at) as weeknum,
device, count(distinct user_id) as user_count
from events
where event_type = 'engagement'
group by weeknum, device
order by weeknum)
select weeknum,device,user_count
from cte;
```

- With clause : It is used to define Common Table Expression
- Cte1: Common Table Expression 1
- Extract function: It is used to retrieve specific components such as year,month,day,etc.. From a date or timestamp.

weeknum	device	user_count
1	acer aspire desktop	198
1	acer aspire notebook	338
1	amazon fire phone	89
1	asus chromebook	355
1	dell inspiron desktop	360
1	dell inspiron notebook	677
1	hp pavilion desktop	339
1	htc one	196
1	ipad air	478
1	ipad mini	292
1	iphone 4s	409
1	iphone 5	1025
1	iphone 5s	626
1	kindle fire	205
1	lenovo thinkpad	1309
1	mac mini	150

weeknum	device	user_count
1	macbook air	950
1	macbook pro	1952
1	nexus 10	273
1	nexus 5	621
1	nexus 7	355
1	nokia lumia 635	211
1	samsumg galaxy tablet	107
1	samsung galaxy note	119
1	samsung galaxy s4	803
1	windows surface	182

◈ Task E: Email Engagement Analysis

Your Task: Write an SQL query to calculate the email engagement metrics.

Code:

```
 select
100*sum(case when email_cat = 'email_open' then 1 else 0 end)/
sum(case when email_cat = 'email_sent'     then 1 else 0 end) as email_open_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_click_rate
from(select*,
Case  when action in ('sent_weekly_digest','sent_reengagement_email’)
then 'email_sent'
when action in ('email_open') then 'email_open'
when action in ('email_clickthrough') then 'email_clicked'
end as email_cat
from `email events`)sub;
```

Output:

email_open_rate	email_click_rate
33.5834	14.7899