Project -3

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

Project description:

This project involves analyzing large set of data and we have derive insight to answer the questions posted by the company from various department.

Approach:

This project is done by using “MySQL Workbench” to analyse the data and provide valuable insight that can help improve the company’s operation and understand sudden changes in key metrics.

Tech-stark used:

In this project I used MySQL workbench because we can easily import data from various source and it is all-in-one tool for both database management and data analysis, allowing working efficiently and intuitively with MySQL database.

Insights:

I learned how to work on a real-time project and it enhanced my knowledge and insight into the code.

DESCRIPTION ABOUT THE PROJECT:

A) CASE-I : JOB DATA ANALYSIS

1. *JOBS REVIEWED OVER TIME:*

Task: write the SQL query to calculate the number of jobs reviewed per hour for each day in NOVEMBER 2020.

Code:

```
SELECT
 ds AS DATE,
 COUNT(job_id)AS joint_job_id,
 ROUND((SUM(time_spent)/3600),2)AS total_Time_sp_Hr,
 ROUND((COUNT(job_id)/(SUM(time_spent)/3600)),2)AS Job_Review_PHr_PDay
 FROM
  job_data
  WHERE
  ds BETWEEN '2020-11-01' AND '2020-11-30'
  GROUP BY ds
  ORDER BY ds;
```

Table:

DATE	job_review_PHr_PDay	joint_job_id	total_time_sp_hr
2020-11-30	2	0.01	205.71
2020-11-29	1	0.01	163.64
2020-11-28	1	0.00	327.27
2020-11-27	1	0.03	34.62
2020-11-26	1	0.02	64.29
2020-11-25	1	0.01	80.00

2. Throughput Analysis

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 .

Code:

```
SELECT
  ROUND(COUNT(event)/SUM(time_spent),2)AS weekly_avg_throughput
  FROM
  job_data;
  SELECT
  ds AS dates,
  ROUND(COUNT(event)/SUM(time_spent),2)AS Daily_avg_throughput
  FROM
  job_data
  GROUP BY ds
  ORDER BY ds;
```

Table:

date	job_reviwed_pHr_pday	joint_job_id	total_time_sp_hr
2020-11-28	327.27	1	0.00
2020-11-30	205.71	2	0.01
2020-11-29	163.64	1	0.01
2020-11-25	80.00	1	0.01
2020-11-26	64.29	1	0.02

2020-11-27	34.62	1	0.03

*3.*Language Share Analysis *:*

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

Code:

```
SELECT
  language,
  ROUND(100*COUNT(*)/total,2)AS percentage,
  jd.total
  FROM
  job_data
  CROSS JOIN
  (SELECT
  COUNT(*) AS total
  FROM
  job_data)AS jd
  GROUP BY language,jd.total;
```

Table:

language	percentage	total
English	14.29	7
Persian	42.86	7
Hindi	14.29	7
French	14.29	7
Italian	14.29	7

4.Duplicate Rows Detection:

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

Code:

```
SELECT
   actor_id,COUNT(*)As Duplicate
   FROM
   job_data
   GROUP BY actor_id
   HAVING COUNT(*)>1;
```

Table:

	actor_id	Duplicate
1	1003	2

B) CASE-II:INVESTIGATING METRIC SPIKE

1.Weekly User Engagement:

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 active_users
FROM
 events
 WHERE
     event_type ='engagement'
 GROUP BY week_num
 ORDER BY week_num;
```

Table:

week_num	active_user
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

2.User Growth Analysis:

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

Code:

```
WITH weekly_active_users AS(
SELECT
  EXTRACT(YEAR FROM activated_at) AS year,
  EXTRACT(WEEK FROM activated_at)AS week_number,
  COUNT(DISTINCT user_id) AS num_of_users
  FROM users
  GROUP BY year, week_number
  )
  SELECT
  year,
  week_number,
  num_of_users,
  SUM(num_of_users) OVER(ORDER BY year,week_number)AS cumulative_users
  FROM weekly_active_users
  ORDER BY YEAR , week_number;
```

Table:

year	week_number	num_of_users	cumulative_users
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

year	week_number	num_of_users	cumulative_users
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
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
2013	48	97	2894

year	week_number	num_of_users	cumulative_users
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
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

year	week_number	num_of_users	cumulative_users
2014	34	259	9363
2014	35	18	9381

3.Weekly Engagement Per Device:

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

Code:

```
select extract(week from occured_at) as week,extract(year from occured_at)as year,device,
count(distinct user_id) as count from events
where event_type='engagement'
group by 1,2,3
order by 1,2,3
```

Table:

year	week_num	num_of_user	cumulative_user
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
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
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
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

RESULT:

I learned , how to worked with large amount of data and it helps me to find the insight analysis of big data.I learned to code for big data in MySQL.