

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKES

PROJECT DESCRIPTION:

This project will help us to analyse different the operational function in a company. This project deals with the topic Operation analytics under which the investigating metric spikes belongs. **Operation Analytics** provides real-time insights and in-depth analysis to optimize your business operations. This will help to make data-driven decisions effortlessly. In a company, the operation analytics will help them to analyse the day-to-day operations within the company so that they can make better decisions through the insights they get from the analysis. The main key aspect of the operation analytics is **Investigating metric spikes**, which will help to understand and explain the sudden changes in key metrics such as a high drop in the sale, or spike in the number of users etc.

In this project we have two case studies, the first one is **Job data analysis** and the second one is **Investigating metric spike**. We will use SQL queries to find out different insights from the data according to the needs specified in the questions.

APPROACH:

We first imported the data which is in csv file to the SQL workbench and then used the SQL workbench and SQL Queries to gain valuable insights from the different data that are given. The following are the steps that I took to analyse the data and for finding answers to the questions:

CASE STUDY 1: JOB DATA ANALYSIS

- A) **Jobs Reviewed Over Time:** Here I write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020 from the given dataset job_data.

MySQL Workbench interface showing a SQL query for Case Study 1. The query calculates the number of jobs reviewed per hour based on the time spent on each job.

```

3  # CASE STUDY 1
4
5  #Jobs reviewed over time:
6
7  * select ds, ((count(job_id))/(sum(time_spent)/3600)) as JobreviewPerHour
8    from job_data
9   group by ds;

```

The result grid shows the following data:

ds	JobreviewPerHour
11/25/2020	80.0000
11/26/2020	64.2857
11/27/2020	34.6154
11/28/2020	218.1818
11/29/2020	180.0000
11/30/2020	180.0000

The output pane shows the execution of the query, indicating that 6 rows were returned.

B) **Throughput Analysis:** Here I write an SQL query to calculate the 7-day rolling average of throughput (number of events per second).

MySQL Workbench interface showing a SQL query for Throughput analysis. The query calculates the weekly rolling average of throughput based on the number of events and time spent.

```

10
11
12 # Throughput analysis
13
14 * select ((count('event'))/(sum(time_spent))) as weekly_rollingaverage
15    from job_data;
16
17

```

The result grid shows the following data:

weekly_rollingaverage
0.0268

The output pane shows the execution of the query, indicating that 6 rows were returned. An error message is also visible: "Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'weekly_rollingaverage' at line 14".

C) **Language Share Analysis:** Here I write an SQL query to calculate the percentage share of each language over the last 30 days.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS

- amitdb
- iq_done
- new_db
 - email_events
 - events
 - job_data
 - users
 - Views
 - Stored Procedures
 - Functions
- sys

SQLProjectcodes*

events email_events

Limit to 1000 rows

```

19
20 #Language share analysis
21
22 • select language, count(*)*100/sum(count(*) over() as languagePercent
23 from job_data
24 where ds<= "11/30/2020 " and ds>= "11/01/2020"
25 group by language;

```

Result Grid

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

Administration Schemas

Information

No object selected

Result 6 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
6	10:08:29	select (count(event))/(sumtime_spent()) as weekly_rollingaverage from job_data #Language share analysis ...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
7	10:08:35	select (count(event))/(sumtime_spent()) as weekly_rollingaverage from job_data LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
8	10:17:01	select language, count(*)*100/sum(count(*) over() as languagePercent from job_data where ds<= "11/30/20...	6 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

29°C Sunny

Search

ENG IN

10:17

20-03-2024

D) **Duplicate Rows Detection:** Here I write an SQL query to display duplicate rows from the job_data table.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS

- amitdb
- iq_done
- new_db
 - email_events
 - events
 - job_data
 - users
 - Views
 - Stored Procedures
 - Functions
- sys

SQLProjectcodes*

events email_events job_data

Limit to 1000 rows

```

27 # Duplicate rows detection
28
29 • select ds,job_id,actor_id, count(*) as duplicate
30 from job_data
31 group by ds,job_id,actor_id
32 having count(*) > 1;
33

```

Result Grid

ds	job_id	actor_id	duplicate
----	--------	----------	-----------

Administration Schemas

Information

Table: job_data

Columns:

Column	Type
ds	text
job_id	int
actor_id	int
event	text
language	text
time_spent	int
org	text

Object Info Session

Result 14 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
17	10:33:06	select actor_id, count(*) as duplicates from job_data group by actor_id having duplicates > 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
18	10:33:06	select job_id, count(*) as duplicates from job_data group by job_id having duplicates > 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
19	10:35:04	select ds,job_id,actor_id, count(*) as duplicate from job_data group by ds,job_id,actor_id having count(*) > 1 ...	0 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

29°C Sunny

Search

ENG IN

10:35

20-03-2024

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

amitdb

ig_dione

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedures

Functions

sys

SQL Editor

33

34 • select actor_id, count(*) as duplicates

35 from job_data

36 group by actor_id

37 having duplicates >1;

38

39

Result Grid

actor_id duplicates

1003 2

Table: job_data

Columns:

ds text

job_id int

actor_id int

event text

language text

time_spent int

org text

Object Info Session

Output

Action Output

Time Action Message Duration / Fetch

10 10:33:06 select job_id, count(*) as duplicates from job_data group by job_id having duplicates >1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

19 10:35:04 select ds.job_id,actor_id, count(*) as duplicate from job_data group by ds.job_id,actor_id having count(*) > 1 ... 0 row(s) returned 0.000 sec / 0.000 sec

20 10:35:43 select actor_id, count(*) as duplicates from job_data group by actor_id having duplicates >1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

29°C Sunny

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

amitdb

ig_dione

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedures

Functions

sys

SQL Editor

40

41 • select job_id, count(*) as duplicates

42 from job_data

43 group by job_id

44 having duplicates >1;

45

46

Result Grid

job_id duplicates

23 3

Table: job_data

Columns:

ds text

job_id int

actor_id int

event text

language text

time_spent int

org text

Object Info Session

Output

Action Output

Time Action Message Duration / Fetch

19 10:35:04 select ds.job_id,actor_id, count(*) as duplicate from job_data group by ds.job_id,actor_id having count(*) > 1 ... 0 row(s) returned 0.000 sec / 0.000 sec

20 10:35:43 select actor_id, count(*) as duplicates from job_data group by actor_id having duplicates >1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

21 10:36:00 select job_id, count(*) as duplicates from job_data group by job_id having duplicates >1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec

29°C Sunny

CASE STUDY 2: INVESTIGATING METRIC SPIKES

Here I am working with three Data sets that are related to each other to get insights from the data.

A) **Weekly User Engagement:** Here I write an SQL query to calculate the weekly user engagement.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
58 # Weekly user engagement
59 select week(str_to_date(occurred_at,"%d-%m-%Y %H:%i")) as week,
60        count(distinct user_id) as num_of_users
61 from events
62 where event_type = 'engagement'
63 group by week;
```

The Result Grid displays the following data:

week	num_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

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
58 # Weekly user engagement
59 select week(str_to_date(occurred_at,"%d-%m-%Y %H:%i")) as week,
60        count(distinct user_id) as num_of_users
61 from events
62 where event_type = 'engagement'
63 group by week;
```

The Result Grid displays the following data:

week	num_of_users
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1121
34	1008
35	49

B) **User Growth Analysis:** Here I write an SQL query to calculate the user growth for the product over every month.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

amitdb

ig_done

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedures

Functions

sys

Administration Schemas Information

No object selected

Object Info Session Output

```
64
65
66 # User growth analysis
67
68
69 • select activated_year, activated_month, num_of_activeUsers,
70       sum(num_of_activeUsers) over
71       ( order by activated_year, activated_month ) as cumsum_activeUsers
72 from
73 (select year(str_to_date(activated_at,"%d-%m-%Y %H:%i")) as activated_year,
74     month(str_to_date(activated_at,"%d-%m-%Y %H:%i")) as activated_month,
75     count(distinct user_id) as num_of_activeUsers
76 from users
77 where state= "active"
78 group by activated_year, activated_month
79 order by activated_year, activated_month) as growth_table;
80
81
82
83
84
```

SQLAdditions

My Snippets

Context Help Snippets

Nifty bank 1.08%

Search

ENG IN

11:07

20-03-2024

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

amitdb

ig_done

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedures

Functions

sys

Administration Schemas Information

No object selected

Object Info Session Output

```
65
66 # User growth analysis
67
68
```

Result Grid

activated_year	activated_month	num_of_activeUsers	cumsum_activeUsers
2013	1	160	160
2013	2	160	320
2013	3	150	470
2013	4	181	651
2013	5	214	865
2013	6	213	1078
2013	7	284	1362
2013	8	316	1678
2013	9	330	2008
2013	10	390	2398
2013	11	399	2797
2013	12	486	3283
2014	1	552	3835
2014	2	525	4360
2014	3	615	4975
2014	4	776	5701

Result 2 x

SQLAdditions

My Snippets

Context Help Snippets

Read Only

31°C Sunny

Search

ENG IN

11:06

20-03-2024

The screenshot shows the MySQL Workbench interface with a query window titled 'sql2projectcodes'. The query is a simple SELECT statement: `SELECT * FROM users;`. The result grid displays 16 rows of data. The columns are 'activated_year', 'activated_month', 'num_of_activeUsers', and 'cumsum_activeUsers'. The data shows a steady increase in the number of active users over time, with a cumulative sum that grows exponentially.

activated_year	activated_month	num_of_activeUsers	cumsum_activeUsers
2013	6	213	1078
2013	7	284	1362
2013	8	316	1678
2013	9	330	2008
2013	10	390	2398
2013	11	399	2797
2013	12	486	3283
2014	1	552	3835
2014	2	525	4360
2014	3	615	4975
2014	4	726	5701
2014	5	779	6480
2014	6	873	7353
2014	7	997	8350
2014	8	1031	9381

c) **Weekly Retention Analysis:** Here I write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

The screenshot shows the MySQL Workbench interface with a query window titled 'sql2projectcodes'. The query is a complex SELECT statement that calculates weekly retention. It uses a series of CASE WHEN statements to calculate the retention for each week from week 0 to week 16. The query is as follows:

```

SELECT
  signup_week AS week_num,
  SUM(CASE WHEN week_nums = 0 THEN 1 ELSE 0 END) AS 'week 0',
  SUM(CASE WHEN week_nums = 1 THEN 1 ELSE 0 END) AS 'week 1',
  SUM(CASE WHEN week_nums = 2 THEN 1 ELSE 0 END) AS 'week 2',
  SUM(CASE WHEN week_nums = 3 THEN 1 ELSE 0 END) AS 'week 3',
  SUM(CASE WHEN week_nums = 4 THEN 1 ELSE 0 END) AS 'week 4',
  SUM(CASE WHEN week_nums = 5 THEN 1 ELSE 0 END) AS 'week 5',
  SUM(CASE WHEN week_nums = 6 THEN 1 ELSE 0 END) AS 'week 6',
  SUM(CASE WHEN week_nums = 7 THEN 1 ELSE 0 END) AS 'week 7',
  SUM(CASE WHEN week_nums = 8 THEN 1 ELSE 0 END) AS 'week 8',
  SUM(CASE WHEN week_nums = 9 THEN 1 ELSE 0 END) AS 'week 9',
  SUM(CASE WHEN week_nums = 10 THEN 1 ELSE 0 END) AS 'week 10',
  SUM(CASE WHEN week_nums = 11 THEN 1 ELSE 0 END) AS 'week 11',
  SUM(CASE WHEN week_nums = 12 THEN 1 ELSE 0 END) AS 'week 12',
  SUM(CASE WHEN week_nums = 13 THEN 1 ELSE 0 END) AS 'week 13',
  SUM(CASE WHEN week_nums = 14 THEN 1 ELSE 0 END) AS 'week 14',
  SUM(CASE WHEN week_nums = 15 THEN 1 ELSE 0 END) AS 'week 15',
  SUM(CASE WHEN week_nums = 16 THEN 1 ELSE 0 END) AS 'week 16'

```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

amitdb

ig_clone

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedure

Functions

sys

Administration Schemas

Information

No object selected

Object Info Session

SQL Snippets

```
106 SUM(CASE WHEN week_nums = 17 THEN 1 ELSE 0 END) AS 'week 17'
107 FROM
108 (SELECT
109     a.user_id, b.engagement_week, a.signup_week,
110     b.engagement_week - a.signup_week AS week_nums
111 FROM
112     (SELECT
113         user_id, WEEK(STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i')) AS signup_week
114     FROM events
115     WHERE event_type = 'signup_flow' AND event_name = 'complete_signup'
116     GROUP BY 1, 2) AS a
117 LEFT JOIN
118     (SELECT
119         user_id, WEEK(STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i')) AS engagement_week
120     FROM events
121     WHERE event_type = 'engagement'
122     GROUP BY 1, 2) AS b
123 ON b.user_id = a.user_id) AS c
124 GROUP BY signup_week
125 ORDER BY signup_week;
126
```

Output

Search

ENG IN

09:58

21-03-2024

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

amitdb

ig_clone

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedure

Functions

sys

Administration Schemas

Information

No object selected

Object Info Session

SQL Snippets

```
113 user_id, WEEK(STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i')) AS signup_week
114 FROM events
115 WHERE event_type = 'signup_flow' AND event_name = 'complete_signup'
116 GROUP BY 1, 2) AS a
```

Result Grid

week_num	week 0	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14	week 15	week 16	week 17
17	72	59	24	16	11	16	11	9	6	8	8	8	7	9	6	5	1	2
18	163	114	73	49	37	26	19	25	13	18	13	13	15	11	9	11	5	1
19	185	142	73	59	40	25	22	19	23	18	15	15	13	11	8	9	0	0
20	176	128	86	52	39	29	22	32	22	21	23	16	17	9	10	0	0	0
21	183	121	74	51	34	23	31	30	20	20	14	16	17	10	0	0	0	0
22	196	142	82	57	47	38	29	23	26	18	18	12	6	0	0	0	0	0
23	196	146	85	57	43	35	27	22	20	14	11	0	0	0	0	0	0	0
24	229	151	89	58	41	32	30	25	15	19	11	0	0	0	0	0	0	0
25	207	165	97	61	40	29	22	19	15	16	0	0	0	0	0	0	0	0
26	201	138	84	60	45	35	32	25	16	0	0	0	0	0	0	0	0	0
27	222	161	95	80	51	38	27	23	0	0	0	0	0	0	0	0	0	0
28	215	161	92	56	35	18	19	0	0	0	0	0	0	0	0	0	0	0
29	221	160	81	53	39	33	1	0	0	0	0	0	0	0	0	0	0	0
30	238	171	94	65	43	3	0	0	0	0	0	0	0	0	0	0	0	0

Result 17 x

Read Only

Context Help

Snippets

Output

Search

ENG IN

10:02

21-03-2024

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHMAS

amitdb

ig_clone

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedure

Functions

sys

Administration Schemas

Information

No object selected

SQL Snippets

My Snippets

113 user_id, WEEK(STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i')) AS signup_week

114 FROM events

115 WHERE event_type = 'signup_flow' AND event_name = 'complete_signup'

116 GROUP BY 1, 2) AS a

Result Grid

week_num	week 0	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14	week 15	week 16	week 17
23	196	146	85	57	51	43	35	27	22	20	14	11	0	0	0	0	0	0
24	229	151	89	58	41	32	30	25	15	19	11	0	0	0	0	0	0	0
25	207	165	97	61	40	29	22	19	15	16	0	0	0	0	0	0	0	0
26	201	138	84	60	45	35	32	25	16	0	0	0	0	0	0	0	0	0
27	222	161	95	80	51	38	27	23	0	0	0	0	0	0	0	0	0	0
28	215	161	92	56	35	18	19	0	0	0	0	0	0	0	0	0	0	0
29	221	160	81	53	39	33	1	0	0	0	0	0	0	0	0	0	0	0
30	238	171	94	65	43	3	0	0	0	0	0	0	0	0	0	0	0	0
31	193	136	69	52	1	0	0	0	0	0	0	0	0	0	0	0	0	0
32	245	70	37	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	261	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	259	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Result 17

Output

Read Only Context Help Snippets

Object Info Session

34°C Sunny

Search

ENG IN

10:02

21-03-2024

D) **Weekly Engagement Per Device:** Here I write an SQL query to calculate the weekly engagement per device.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHMAS

amitdb

ig_clone

new_db

Tables

email_events

events

job_data

users

Views

Stored Procedure

Functions

sys

Administration Schemas

Information

No object selected

SQL Snippets

My Snippets

110 # Weekly Engagement Per Device:

111

112 • select year(str_to_date(occurred_at, '%d-%m-%Y %H:%i')) as year,

113 week(str_to_date(occurred_at, '%d-%m-%Y %H:%i')) as week,

114 device, count(distinct user_id) as user_count

115 from events

116 where event_type= 'engagement'

117 group by 2,3,1

118 order by 2,3,1;

Result Grid

year	week	device	user_count
2014	17	acer aspire desktop	9
2014	17	acer aspire notebook	20
2014	17	amazon fire phone	4
2014	17	asus chromebook	21
2014	17	dell inspiron desktop	18
2014	17	dell inspiron notebook	46
2014	17	hp pavilion desktop	14
2014	17	htc one	16
2014	17	ipad air	27
2014	17	ipad mini	19

Result 11

Output

Read Only Context Help Snippets

Object Info Session

34°C Sunny

Search

ENG IN

12:43

20-03-2024

MySQL Workbench interface showing a SQL query and its results. The query is titled "# Weekly Engagement Per Device:" and is executed against the 'new_db' database. The results are displayed in a table with columns: year, week, device, and user_count.

```

110 # Weekly Engagement Per Device:
111
112 • select year(str_to_date(occurred_at,"%d-%m-%Y %H:%i")) as year,
113        week(str_to_date(occurred_at,"%d-%m-%Y %H:%i")) as week.

```

year	week	device	user_count
2014	17	ipad mini	19
2014	17	iphone 4s	21
2014	17	iphone 5	65
2014	17	iphone 5s	42
2014	17	kindle fire	6
2014	17	lenovo thinkpad	86
2014	17	mac mini	6
2014	17	macbook air	54
2014	17	macbook pro	143
2014	17	nexus 10	16
2014	17	nexus 5	40
2014	17	nexus 7	18
2014	17	nokia lumia 635	17
2014	17	samsung galaxy tablet	8
2014	17	samsung galaxy note	7
2014	17	samsung galaxy s4	52

And continues...

E) **Email Engagement Analysis:** Here I write an SQL query to calculate the email engagement metrics.

MySQL Workbench interface showing a SQL query titled "# Email Engagement Analysis". The query is executed against the 'new_db' database. The results are displayed in a table with columns: user_type, action, total_events, num_of_users, and averageEvents_per_user.

```

120 # Email Engagement Analysis
121
122 • SELECT
123     user_type,
124     action,
125     COUNT(*) AS total_events,
126     COUNT(DISTINCT user_id) AS num_of_users,
127     COUNT(*) / COUNT(DISTINCT user_id) AS averageEvents_per_user
128 FROM
129     email_events
130 GROUP BY 1 , 2;

```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

Schemas

- amitdb
- iq_done
- new_db
 - email_events
 - events
 - job_data
 - users
- Views
- Stored Procedures
- Functions
- sys

SQL Editor

```

124 action,
125 COUNT(*) AS total_events,
126 COUNT(DISTINCT user_id) AS num_of_users,
127 COUNT(*) / COUNT(DISTINCT user_id) AS averageEvents_per_user
128 FROM
129 email_events

```

Result Grid

user_type	action	total_events	num_of_users	averageEvents_per_user
1	email_clickthrough	2758	1529	1.8038
1	email_open	6511	1717	3.7921
1	sent_reengagement_email	892	892	1.0000
1	sent_weekly_digest	18412	1217	15.1290
2	email_clickthrough	2521	1529	1.6488
2	email_open	5562	1701	3.2698
2	sent_reengagement_email	1071	1071	1.0000
2	sent_weekly_digest	15232	1098	13.8725
3	email_clickthrough	3731	2219	1.6814
3	email_open	8386	2509	3.3424
3	sent_reengagement_email	1690	1690	1.0000
3	sent_weekly_digest	23623	1796	13.1531

Result 19 x

Object Info Session

34°C Sunny

Search

ENG IN

13:16

20-03-2024

TECH-STACK USED:

For the analysis of data, I used **the MySQL Workbench 8.0 CE**. And in this software, I imported the data sets given and use SQL queries to find different insights for different needs. The reason why we use the SQL is that since the database is relational and we want to join different data to get the information we want the SQL is the best choice. Also, for the dataset that is very large SQL is useful. For the reporting of the project, I used the **Microsoft Word**.

INSIGHTS AND RESULTS:

CASE STUDY 1: JOB DATA ANALYSIS

- Jobs reviewed overtime:** After the SQL query we can see that on an average of **218 jobs** were reviewed per hour on **28th NOV 2020**, which is the highest in all dates.
- Throughput Analysis:** The 7-day rolling average of the jobs reviewed per second is **0.0268**. Comparing the both averages that we have found about the jobs reviewed I think the average job reviewed per hour in each day will

make more sense to our analysis because this will help to tell that in which days of the week the mostly the jobs were reviewed, and what can we do to improve the work in the other days.

- C) **Language Share Analysis:** In this analysis we were able to find that which language was there in the most of the content that was reviewed overtime and found out that **37.5%** of content was in **Persian language** amongst all other languages.
- D) **Duplicate Rows Detection:** If we consider every column in each row, we cannot find any duplicate rows, that is by considering the entire row in a data we cannot find any duplicate rows. But when we consider about the unique actor id, we can find that there are **2 duplicates for the actor id 1003**. Similarly, when we consider the job id, we find that there are **3 duplicates for the job id 23**.

CASE STUDY 2: INVESTIGATING METRIC SPIKES

- A) **Weekly User Engagement:** The user engagement is **highest in the week 30 with 1467 users** and the user engagement is **least in the week 35 with 49 users**.
- B) **User Growth Analysis:** The year and month at which the **most users activated their account is 8th(august) month of 2014**. The Year and month in which the **least users activated their account is 3rd(march) month of 2013**. But in an overall view there was a significant increase in the number of users during this time.
- C) **Weekly Retention Analysis:** Here we can see that the user engagement is gradually decreasing as the number of weeks increases after the sign up of the user. Even if there are many people who activated the account, there is not enough engagement after the signup of the account.
- D) **Weekly Engagement Per Device:** This analysis helps us to understand that on a weekly basis how many users have used a particular device to login to their accounts. This will help the team to understand that which device is mostly used by users on a weekly basis.
- E) **Email Engagement Analysis:** For the user_type 1, **on an average of 4 events of email_open was occurred for a particular user**. Similarly, we can interpret the rest of the results.

As a result, this project helps me to learn about a variety field such as operation analytics and the aspect of investigating metric spikes. Also, this project developed my SQL understanding a lot more. I have learned a bit about the advanced SQL and the queries I have written made me understand that SQL can interpret and analyse the data in a wide variety of ways.