Operation Analytics and Investigating Metric Spike

Data Analysis Using SQL

Description:

Operation Analytics is the process of analyzing a company's end-to-end operations to identify areas that can be improved. It involves collecting and analyzing data from various sources to identify patterns, trends, and correlations. This analysis is used to optimize the company's operations, improve workflows, and increase efficiency.

Investigating metric spikes is an important aspect of Operation Analytics. It involves analyzing sudden changes or spikes such as sales, engagement, or customer satisfaction. This analysis helps to identify the root cause of the spike and take corrective action if necessary.

Name: Shakti Prasad Nayak	26/03/2023

Approach

Here, in this project, we are analyzing two case studies for Operation Analytics and Investing Metric Spike. We are analyzing csv files given below-

Files for case study 1
https://drive.google.com/file/d/1LxLroe-8bZdVk85gue5Qb3SMsUJrazBB/view?usp=sharing

Files for case study 2
https://drive.google.com/file/d/19I1HprWLn5U12S1_5GMBUBuwcwk7e3ZW/view?usp=sharing
https://drive.google.com/file/d/12D-vM6NrZaErvybZz9vYYYWxuLPazDgy/view?usp=sharing
https://drive.google.com/file/d/14eSiwrY_swtw41PFp-wvO5ZI7FJ4beOV/view?usp=sharing

For case study 1 create a table named job_data and the schema for table is as follows

```
CREATE TABLE `job_data` (
`ds` datetime DEFAULT NULL,
`job_id` int DEFAULT NULL,
`actor_id` int DEFAULT NULL,
`event` text,
`language` text,
`time_spent` int DEFAULT NULL,
`org` text
)
```

For case study 2 create 3 tables named users, events and email_events. The schemas for tables are as follows

```
CREATE TABLE `users` (
`user_id` bigint DEFAULT NULL,
`created_at` datetime DEFAULT NULL,
`company_id` bigint DEFAULT NULL,
`language` text,
`activated_at` text,
`state` text
)

CREATE TABLE `events` (
`user_id` bigint DEFAULT NULL,
`occurred_at` datetime DEFAULT NULL,
`event_type` text,
`event_name` text,
`location` text,
`device` text,
`user_type` text
)

CREATE TABLE `email_events` (
`user_id` bigint DEFAULT NULL,
`occurred_at` datetime DEFAULT NULL,
`action` text,
`user_type` text
)
```

Then import the csv files into the corresponding table using table data import wizard. Now, we are ready to analyze the tables we have just imported from csv datasets to fetch relevant information.

Tech-Stack Used

MySQL Workbench is a powerful visual database design tool that helps developers and database administrators to create, manage, and troubleshoot MySQL databases. Some of the features of MySQL Workbench include:

1. Visual Database Design: MySQL Workbench's visual database design tools allow users to create, modify and visualize database schema with ease. This can save a lot of time and effort compared to writing SQL scripts manually.
2. SQL Editor: The SQL editor in MySQL Workbench is a powerful tool that supports syntax highlighting, auto complete, and SQL code analysis. It makes writing and debugging SQL queries faster and more efficient.
3. Database Administration: MySQL Workbench provides a wide range of tools for database administration, including backup and recovery, performance tuning, and user management. These tools are essential for maintaining the health and security of a database system.
4. Query Optimization: MySQL Workbench includes a query optimization tool that helps users analyze and optimize complex SQL queries. This can improve query performance and reduce database load.
5. Cross-platform Support: MySQL Workbench is available on multiple platforms, including Windows, macOS, and Linux. This makes it a versatile tool that can be used by developers and DBAs on different operating systems.

Insights

1. For Case study 1 there are a total 34 jobs reviewed and the hours spent in the sum of 16 days is 0.5415.
2. The 7 day rolling average at its peak in the last few days.
3. Persian language is widely used among actors.
4. For case study 2, the average number of users engaged per day is 1158.
5. The user growth is negative in February and September.
6. The growth is higher in engagement activity than signup.
7. Most of the users access the service using computers.

Results

1. The company should make its system Persian language friendly.
2. The company should focus on making strategies to increase user growth in February and September.
3. The user interface of the application should be computer friendly as most users rely on it.

Case Study 1

Number of jobs reviewed:

The number of jobs reviewed per hour per day for November 2020 is found by executing the following query-

```
select ds,count(job_id) as jobs_reviewed, sum(time_spent)/3600
as hours_spent
from job_data
where ds >='2020-11-01' and ds <='2020-11-30'
group by ds ;
```

The resultset obtained after executing the above query is

ds	obs_reviewed	ours_spent
2020-11-30 0:00:00	2	0.0111
2020-11-29 0:00:00	1	0.0056
2020-11-28 0:00:00	3	0.0247
2020-11-27 0:00:00	3	0.0867
2020-11-26 0:00:00	3	0.0517
2020-11-25 0:00:00	2	0.0278
2020-11-24 0:00:00	2	0.0297
2020-11-23 0:00:00	2	0.0386
2020-11-22 0:00:00	2	0.0372
2020-11-20 0:00:00	1	0.0119
2020-11-17 0:00:00	3	0.0522
2020-11-16 0:00:00	2	0.0306
2020-11-18 0:00:00	3	0.0456
2020-11-19 0:00:00	1	0.0175
2020-11-21 0:00:00	2	0.025
2020-11-15 0:00:00	2	0.0456

Throughput:

To calculate the 7-day rolling average of throughput, we would start by summing the throughput values for the past 7 days, and then dividing by 7 to get the average value. Then, we would shift the window of 7 days by one day and repeat the calculation until we reach the end of the dataset.
Note that the first 6 days of the dataset don't have a full 7-day window, so we can't calculate the rolling average for those days.
The below sql query will show the daily metric and 7-day rolling average results-

```
SELECT ds,
daily_metric,
CASE
WHEN ROW_NUMBER() OVER (ORDER BY ds) >= 7
THEN avg(daily_metric) OVER (ORDER BY ds ROWS BETWEEN 6
PRECEDING AND CURRENT ROW) ELSE 0
END AS rolling_avg
FROM (
SELECT ds,
COUNT(DISTINCT job_id) AS daily_metric
FROM job_data
GROUP BY ds
ORDER BY ds
) AS mytable;
```

The resultset obtained after executing the above query is-

ds	laily_metric	olling_avg
2020-11-15 0:00:00	2	0
2020-11-16 0:00:00	2	0
2020-11-17 0:00:00	3	0
2020-11-18 0:00:00	2	0
2020-11-19 0:00:00	1	0

2020-11-20 0:00:00	1	0
2020-11-21 0:00:00	1	1.7143
2020-11-22 0:00:00	1	1.5714
2020-11-23 0:00:00	2	1.5714
2020-11-24 0:00:00	1	1.2857
2020-11-25 0:00:00	2	1.2857
2020-11-26 0:00:00	2	1.4286
2020-11-27 0:00:00	2	1.5714
2020-11-28 0:00:00	3	1.8571
2020-11-29 0:00:00	1	1.8571
2020-11-30 0:00:00	2	1.8571

A daily metric can provide more granular and immediate insights into the day-to-day performance of the system, while a 7-day rolling average can help identify longer-term trends and patterns that may not be apparent in daily metrics alone.
If the goal is to track the impact of a recent change to the system, using daily metrics may be more appropriate to observe the immediate effects. On the other hand, if the goal is to identify performance trends over a longer period of time, such as identifying seasonality or cyclical patterns, a 7-day rolling average may provide a better representation of the underlying trends.
In our case, we will choose daily metric as it indicates performance on daily basis.

Percentage share of each language:

The percentage share of each language is calculated by dividing total count of specific language by total count of all languages, multiplied by 100.
The sql query below will find the percentage share of each language-

```
select `language`, round(100*count(`language`)/(SELECT COUNT(*)
FROM job_data ), 2) AS `percentage share`
from job_data
group by `language`;
```

The resultset obtained after executing the above query is-

language	percentage share
English	17.65
Arabic	5.88
Persian	26.47
Hindi	11.76
French	20.59
Italian	17.65

Duplicate rows:

The below sql query will find the duplicate rows in our table-

```
SELECT *
FROM job_data
WHERE (job_id, actor_id) IN
(
SELECT job_id, actor_id
FROM job_data
GROUP BY job_id, actor_id
HAVING COUNT(*) > 1
) order by ds;
```

The resultset obtained after executing the above query is-

ds	job_id	actor_id	event	language	time_spent	org
2020-11-18 0:00:00	24	1005	transfer	Italian	543	
2020-11-18 0:00:00	24	1005	transfer	Italian	543	
2020-11-21 0:00:00	22	1007	decision	Persian	45	
2020-11-21 0:00:00	22	1007	decision	Persian	45	

2020-11-22 0:00:00	20	1008	kip	lindi	67(	
2020-11-22 0:00:00	20	1008	kip	lindi	67(	
2020-11-26 0:00:00	23	1004	kip	ersian	56\	
2020-11-26 0:00:00	23	1004	kip	ersian	56\	
2020-11-27 0:00:00	11	1007	lecision	rench	104)	
2020-11-27 0:00:00	11	1007	lecision	rench	104)	

Case Study 2

User Engagement:

The below sql query will find the user engagement based on each week-

```
select week(occurred_at) as `week number`,
count(distinct user_id) as `users engagement`
from `events` where event_type = "engagement"
group by `week number` order by `week number`;
```

The resultset obtained after executing above query is given below-

week number	sers engagement
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

User Growth:

The growth percentage is calculated as follows: (Users from current month / Users from previous month - 1) * 100.
The below sql query will find the user engagement based on each week-

```
select Months, Users,
ROUND(((Users/LAG(Users, 1) OVER () - 1)*100), 2) AS "Growth in
%"
from (
SELECT monthname(created_at) AS Months, COUNT(activated_at) AS
Users
FROM users
WHERE activated_at NOT IN ('') GROUP BY Months
) AS Mytable;
```

The resultset obtained after executing the above query is-

‡ Months	Users	Growth in %
January	712	
February	685	-3.79
March	765	11.68
April	907	18.56

May	993	9.48
June	1086	9.37
July	1281	17.96
August	1347	5.15
September	330	-75.5
October	390	18.18
November	399	2.31
December	486	21.8

Weekly Retention:

The "Weekly retention" refers to the percentage of users who continue to use a product or service on a weekly basis after signing up.
To calculate the weekly retention rate for a user sign-up cohort, you need to track the number of users who signed up in a particular week (the cohort) and how many of those users continue to use the product on a weekly basis.
The below sql query will find out the weekly retention based on signup and engagement events-

```
select week(occurred_at) as `week`,
count(distinct case when event_type = 'signup_flow' then user_id
else null end) as signup,
count(distinct case when event_type = 'engagement' then user_id
else null end) as engagement
from `events` group by `week` order by `week`;
```

The resultset obtained after executing the above query-

week	signup	engagement
17	149	663
18	355	1068
19	359	1113
20	373	1154
21	361	1121
22	391	1186

23	410	1232
24	425	1275
25	402	1264
26	408	1302
27	423	1372
28	427	1365
29	452	1376
30	477	1467
31	408	1299
32	474	1225
33	474	1225
34	498	1204
35	32	104

Weekly Engagement:

Weekly engagement refers to the level of user activity and interaction with a product or service on a weekly basis. It measures whether the user finds value and quality in the product or service on a regular basis.

To calculate weekly engagement per device, we will need to track the user activity data for each device used by the user.

To find all the device lists we need to execute the following sql query-

```
select distinct device FROM p2.`events`;
```

This will return a list of devices present in our table-

device
lell inspiron notebook
phone 5
phone 4s
vindows surface
nacbook air
phone 5s
nacbook pro
indle fire
ad mini
exus 7

exus 5
amsung galaxy s4
enovo thinkpad
amsumg galaxy tablet
cer aspire notebook
sus chromebook
tc one
okia lumia 635
amsung galaxy note
cer aspire desktop
ac mini
p pavilion desktop
ell inspiron desktop
ad air
mazon fire phone
exus 10

Now we can group the device as per their type eg. Computer, Tablet, Mobile. The following sql query will group the device as per their type and finds the engagement-

```
select week(occurred_at) as week_number,
count(distinct user_id) AS `Weekly Active users`,


COUNT(DISTINCT CASE
WHEN device IN ('dell inspiron notebook', 'windows surface',
'macbook air', 'macbook pro', 'lenovo thinkpad', 'acer aspire
notebook', 'asus chromebook', 'acer aspire desktop', 'mac mini',
'hp pavilion desktop', 'dell inspiron desktop')
THEN user_id
ELSE NULL
END) AS "Computer",
COUNT(DISTINCT CASE
WHEN device IN ('ipad mini', 'nexus 7', 'nexus 5', 'samsumg
galaxy tablet', 'ipad air', 'nexus 10')
```

```
THEN user_id
ELSE NULL
END) AS "Tablet",
COUNT(DISTINCT CASE
WHEN device IN ('iphone 5', 'iphone 4s', 'iphone 5s', 'kindle
fire', 'samsung galaxy s4', 'htc one', 'nokia lumia 635',
'samsung galaxy note', 'amazon fire phone')
THEN user_id
ELSE NULL
END) AS "Mobile"


from `events` WHERE event_type = "engagement" group by
week_number order by week_number;
```

The resultset obtained after executing the above query is-

week_number	Weekly Active users	Computer	Tablet	Mobile
17	663	399	124	228
18	1068	724	210	409
19	1113	736	243	416
20	1154	767	251	448
21	1121	734	214	431
22	1186	805	254	434
23	1232	819	246	478
24	1275	836	269	477
25	1264	858	258	453
26	1302	840	260	510
27	1372	912	256	529
28	1365	934	237	523
29	1376	933	240	540
30	1467	979	280	530
31	1299	938	215	439
32	1225	900	198	381
33	1225	899	195	373

34	1204	887	196	388
35	104	69	10	25

To find the engagement per device we need to execute the following sql query-

```
select week(occurred_at) as week_number,
count(distinct user_id) AS `Weekly Active users`,
COUNT(DISTINCT CASE WHEN device IN('dell inspiron notebook')
THEN user_id ELSE NULL END) AS "dell inspiron notebook",
COUNT(DISTINCT CASE WHEN device = 'iphone 5' THEN user_id ELSE
NULL END) AS "iphone 5",
COUNT(DISTINCT CASE WHEN device = 'iphone 4s' THEN user_id ELSE
NULL END) AS "iphone 4s",
COUNT(DISTINCT CASE WHEN device = 'windows surface' THEN user_id
ELSE NULL END) AS "windows surface",
COUNT(DISTINCT CASE WHEN device = 'macbook air' THEN user_id
ELSE NULL END) AS "macbook air",
COUNT(DISTINCT CASE WHEN device = 'iphone 5s' THEN user_id ELSE
NULL END) AS "iphone 5s",
COUNT(DISTINCT CASE WHEN device = 'macbook pro' THEN user_id
ELSE NULL END) AS "macbook pro",
COUNT(DISTINCT CASE WHEN device = 'kindle fire' THEN user_id
ELSE NULL END) AS "kindle fire",
COUNT(DISTINCT CASE WHEN device = 'ipad mini' THEN user_id ELSE
NULL END) AS "ipad mini",
COUNT(DISTINCT CASE WHEN device = 'nexus 7' THEN user_id ELSE
NULL END) AS "nexus 7",
COUNT(DISTINCT CASE WHEN device = 'nexus 5' THEN user_id ELSE
NULL END) AS "nexus 5",
COUNT(DISTINCT CASE WHEN device = 'samsung galaxy s4' THEN
user_id ELSE NULL END) AS "samsung galaxy s4",
COUNT(DISTINCT CASE WHEN device = 'lenovo thinkpad' THEN user_id
ELSE NULL END) AS "lenovo thinkpad",
```

```
COUNT(DISTINCT CASE WHEN device = 'samsumg galaxy tablet' THEN
user_id ELSE NULL END) AS "samsung galaxy tablet",
COUNT(DISTINCT CASE WHEN device = 'acer aspire notebook' THEN
user_id ELSE NULL END) AS "acer aspire notebook",
COUNT(DISTINCT CASE WHEN device = 'asus chromebook' THEN user_id
ELSE NULL END) AS "asus chromebook",
COUNT(DISTINCT CASE WHEN device = 'htc one' THEN user_id ELSE
NULL END) AS "htc one",
COUNT(DISTINCT CASE WHEN device = 'nokia lumia 635' THEN user_id
ELSE NULL END) AS "nokia lumia 635",
COUNT(DISTINCT CASE WHEN device = 'samsung galaxy note' THEN
user_id ELSE NULL END) AS "samsung galaxy note",
COUNT(DISTINCT CASE WHEN device = 'acer aspire desktop' THEN
user_id ELSE NULL END) AS "acer aspire desktop",
COUNT(DISTINCT CASE WHEN device = 'mac mini' THEN user_id ELSE
NULL END) AS "mac mini",
COUNT(DISTINCT CASE WHEN device = 'hp pavilion desktop' THEN
user_id ELSE NULL END) AS "hp pavilion desktop",
COUNT(DISTINCT CASE WHEN device = 'dell inspiron desktop' THEN
user_id ELSE NULL END) AS "dell inspiron desktop",
COUNT(DISTINCT CASE WHEN device = 'ipad air' THEN user_id ELSE
NULL END) AS "ipad air",
COUNT(DISTINCT CASE WHEN device = 'amazon fire phone' THEN
user_id ELSE NULL END) AS "amazon fire phone",
COUNT(DISTINCT CASE WHEN device = 'nexus 10' THEN user_id ELSE
NULL END) AS "nexus 10"


from `events` WHERE event_type = "engagement" group by
week_number order by week_number;
```

The resultset obtained after executing the above query is-

# vee k_n imb er	We ekly Acti ve iser s	lell nsp ron note oo k	phone ne 5	pho ie -s	win low surf ace	mac oo k air	pho ie 5s	mac oo k pro	kind e ire	pad nini	nex us 7	nex us 5	sam sun g gala xy s4	leno vo hin kpa d	sam sun g gala xy abl et	acer aspi e note oo k	asu s chro nebo ok	htc one	noki a umi a 635	sam sun g gala xy note	acer aspi e les top	mac mini	hp pavi on les top	dell nsp ron les top	pad air	ama zon ire pho ne	nex us 0
17	663	46	65	21	10	54	42	143	6	19	18	40	52	86	8	20	21	16	17	7	9	6	14	18	27	4	16
18	1068	77	113	46	10	121	73	252	27	30	30	73	82	153	11	33	42	19	33	15	26	13	37	58	52	9	30
19	1113	83	115	44	16	112	79	266	21	36	41	87	91	178	6	41	27	30	23	11	23	18	40	36	55	12	25
20	1154	84	125	55	21	119	79	256	23	32	32	103	93	173	9	40	41	29	22	18	23	26	30	52	59	11	22
21	1121	80	137	45	17	110	74	247	30	23	29	91	84	167	6	47	38	21	25	20	29	18	44	41	51	5	25
22	1186	92	125	45	15	145	71	251	21	34	45	96	105	176	10	41	52	24	25	19	25	25	38	52	58	5	27
23	1232	103	152	53	14	124	79	266	25	33	36	88	99	176	14	43	49	20	31	14	22	18	54	53	41	16	45
24	1275	99	142	53	22	152	79	255	25	39	49	87	101	165	11	40	43	20	35	20	24	29	56	59	57	11	38
25	1264	105	137	40	22	121	78	275	24	30	51	89	99	197	12	47	38	21	37	14	28	21	52	52	57	13	29
26	1302	89	152	50	21	134	94	269	26	43	46	87	112	192	12	35	49	23	42	9	29	11	46	60	56	13	29
27	1372	89	163	67	33	142	83	302	25	35	40	84	116	202	15	49	52	27	31	15	29	15	56	53	55	10	37
28	1365	103	151	61	33	148	93	295	31	35	39	85	122	220	9	49	50	26	35	10	30	28	56	56	54	6	26
29	1376	113	144	60	28	148	90	295	37	34	45	77	123	209	13	53	49	31	43	16	28	31	58	54	52	12	25
30	1467	127	152	65	19	159	103	322	25	35	62	84	103	206	9	60	56	31	34	15	33	23	42	54	70	12	36
31	1299	113	135	56	19	147	71	321	14	27	38	69	100	207	8	55	56	13	28	14	31	24	51	44	55	14	24
32	1225	104	119	34	10	125	67	307	12	30	25	67	82	179	6	55	62	18	28	12	35	20	51	57	48	12	30

33	1225	110	110	35	15	133	65	312	14	28	30	70	80	191	12	46	49	19	27	13	39	32	38	37	40	14	23
34	1204	105	101	50	18	136	70	292	13	25	33	70	90	193	14	63	47	25	17	13	30	30	36	49	39	11	25
35	104	9	2	6	3	10	3	17	3	2	2	4	6	16	0	3	6	2	2	1	1	2	1	1	0	0	2

Email Engagement:

The Email engagement refers to the metrics involved in the actions of email events eg. Email opened, Clicked link etc.

The following sql query will find the email engagement based on different events-

```
select `week number`,
Round(Weekly_mail/total * 100, 2) as `Weekly mail in %`,
Round(Opened_mail/total * 100, 2) as `Opened mail in %`,
Round(Clicked_mail/total * 100, 2) as `Clicked link in %`,
Round(Reengagement_mail/total * 100, 2) as `Re-engagement in %`
from
(select week(occurred_at) as `week number`,
count(case when `action` = 'sent_weekly_digest' then user_id
else null end) AS Weekly_mail,
count(case when `action` = 'email_open' then user_id else null
end) AS Opened_mail,
count(case when `action` = 'email_clickthrough' then user_id
else null end) AS Clicked_mail,
count(case when `action` = 'sent_reengagement_email' then
user_id else null end) AS Reengagement_mail,
count(user_id) as total
from `email_events` group by `week number`) as `weekly metrics`
group by `week number` order by `week number`;
```

The resultset obtained after executing the above query is-

week number	Weekly mail in %	Opened mail in %	Clicked link in %	Re-engagement in %
17	62.32	21.28	11.39	5.01

18	63.45	22.24	10.49	3.83
19	62.16	22.67	11.13	4.04
20	61.62	22.64	11.43	4.31
21	63.52	22.82	9.97	3.69
22	63.59	21.56	10.66	4.19
23	62.39	22.34	11.18	4.09
24	61.61	22.92	10.99	4.48
25	63.77	21.79	10.54	3.9
26	62.99	22.22	10.61	4.18
27	62.24	22.49	11.37	3.9
28	62.92	22.48	10.77	3.83
29	63.98	21.71	10.51	3.79
30	62.29	23.24	10.59	3.88
31	65.27	23.25	7.66	3.82
32	66.59	22.85	7.14	3.42
33	64.73	23.1	7.91	4.26
34	64.33	23.91	7.67	4.08
35	0	32.28	29.92	37.8