

Operation Analytics & Investigating Metric Spike

Project Description:

Operation Analytics is the analysis done for the complete end to end operations of a company. With this, the company finds the areas on which it must improve upon.

My job is to work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect.

This kind of analysis is further used to predict the overall growth or decline of a company.

Investigating metric spike is also an important part of operation analytics to understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that its very important to investigate metric spike.

I am working for a company like Microsoft designated as Data Analyst Lead and is provided with different data sets, tables from which I must derive certain insights out of it and answer the questions asked by different departments.

Approach:

Step1: To run the given commands for creating the database to work on.

(the database that has been provided in the attachments)

Step2: Perform the analysis by answering the questions asked using the SQL queries.

Step3: Analysing the dataset and results

Step4: Providing my insights based on the results.

Step5: Preparation of the Document

Tech Stack Used:

I used

- The online editor - <https://www.db-fiddle.com/> with **MySQL** database of **Version 8.0** for the Case study 1 (Job Data).
- **MySQL Workbench Version 8.0.33** for the Case study 2(Investigating Metric Spike).
- **Ms Word** for the preparation of the document to be presented.

Insights:

Based on the achieved results I can conclude the following things:

1. **The user growth percentage is fluctuating** but when seen as overall_growth it is rising.
2. I can see **redundant data (duplicates) from the job_data** which can be an issue.
3. **The user is actively using the product/service** which shows the good quality of the product/service provided.
4. **The users are engaging with email service quite nicely** which can be seen from the email engagement metric results.
5. **The product/service is also quite often used on different devices (platforms)** which is a good sign.

Results:

The detailed answers to the questions are below:

Case Study 1 (Job Data):

I have created the database using the data provided in the Excel spreadsheet

Query: CREATE TABLE jobs

```
(  
    ds      date,  
    job_id  int,  
    actor_id      int,  
    event  VARCHAR(50),  
    language    VARCHAR(50),  
    time_spent  int,  
    org    VARCHAR(50)  
);
```

INSERT INTO jobs

(ds, job_id, actor_id, event, language, time_spent, org)

VALUES

```
('2020-11-30', '21', '1001', 'skip', 'English', '15', 'A'),  
( '2020-11-30', '22', '1006', 'transfer', 'Arabic', '25', 'B'),  
( '2020-11-29', '23', '1003', 'decision', 'Persian', '20', 'C'),  
( '2020-11-28', '23', '1005', 'transfer', 'Persian', '22', 'D'),  
( '2020-11-28', '25', '1002', 'decision', 'Hindi', '11', 'B'),  
( '2020-11-27', '11', '1007', 'decision', 'French', '104', 'D'),  
( '2020-11-26', '23', '1004', 'skip', 'Persian', '56', 'A'),  
( '2020-11-25', '20', '1003', 'transfer', 'Italian', '45', 'C');
```

Result:

Query #1 Execution time: 1ms

ds	job_id	actor_id	event	language	time_spent	org
2020-11-30	21	1001	skip	English	15	A
2020-11-30	22	1006	transfer	Arabic	25	B
2020-11-29	23	1003	decision	Persian	20	C
2020-11-28	23	1005	transfer	Persian	22	D
2020-11-28	25	1002	decision	Hindi	11	B
2020-11-27	11	1007	decision	French	104	D
2020-11-26	23	1004	skip	Persian	56	A
2020-11-25	20	1003	transfer	Italian	45	C

A. Number of jobs reviewed: Calculate the number of jobs reviewed per hour per day for November 2020?

Query: **Select**
 ds,
 (count (distinct job_id) / (30*24)) as jobs_reviewed
from jobs

 where ds between '2020-11-01' and '2020-11-30'
 group by ds
 order by ds DESC

Result:

Query #1 Execution time: 0ms

ds	jobs_reviewed
2020-11-30	1.6000
2020-11-29	0.8000
2020-11-28	1.6000
2020-11-27	0.8000
2020-11-26	0.8000
2020-11-25	0.8000

- B. Throughput:** Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

Query: **select**

```
    ds,

    avg (Throughput) over (order by ds rows between 6 preceding and current row)
    as 7day_rolling_avg

from

    (select

        ds,

        (count (distinct event)/ sum(time_spent)) as Throughput

    from

        jobs

    where ds between '2020-11-01' and '2020-11-30'

    group by ds) as sub
```

Result:

I prefer the 7_day rolling average metric because it can give you the overview of the past 7 days data which can be of much more use than a day to day metric

Query #1 **Execution time: 1ms**

ds	7day_rolling_avg
2020-11-25	0.02220000
2020-11-26	0.02005000
2020-11-27	0.01656667
2020-11-28	0.02757500
2020-11-29	0.03206000
2020-11-30	0.03505000

C. Percentage share of each language: Calculate the percentage share of each language in the last 30 days?

Query: **select**
language,
round((count(language) / (select count(distinct language) from
jobs))*100,2) as Percentage_share_of_each_language

from
jobs
group by language

Result:

Query #1 Execution time: 1ms

language	Percentage_share_of_each_language
English	16.67
Arabic	16.67
Persian	50.00
Hindi	16.67
French	16.67
Italian	16.67

D. Duplicate rows: How will you display duplicates from the table?

Query: **select * from**
(select *, row_number() over (partition by job_id) as rownumber from jobs) as a
where rownumber>1

Result:

Query #1 Execution time: 0ms

ds	job_id	actor_id	event	language	time_spent	org	rownumber
2020-11-28	23	1005	transfer	Persian	22	D	2
2020-11-26	23	1004	skip	Persian	56	A	3

Case Study2 (Investigating Metric Spike):

A. User Engagement: Calculate the weekly user engagement?

Query: **SELECT**
EXTRACT(WEEK FROM created_at) AS week_number,
COUNT(DISTINCT user_id) AS num_of_users
FROM
users
GROUP BY week_number

Result:

week_number	num_of_users
0	197
1	300
2	299
3	325
4	322
5	341
6	344
7	353
8	350
9	353
10	377
11	382
12	391
13	396
14	411
15	395
16	465
17	450
18	460
19	467
20	477
21	474

week_number	num_of_users
22	503
23	526
24	543
25	529
26	535
27	555
28	568
29	582
30	618
31	539
32	622
33	625
34	647
35	196
36	164
37	164
38	166
39	180
40	174
41	172
42	191
43	195
44	194
45	191
46	179

week_number	num_of_users
28	568
29	582
30	618
31	539
32	622
33	625
34	647
35	196
36	164
37	164
38	166
39	180
40	174
41	172
42	191
43	195
44	194
45	191
46	179
47	207
48	213
49	216
50	221
51	235
52	87

B. User Growth: Calculate the user growth for product?

Query: **SELECT**

month,

active_users,

round(((active_users - lag(active_users,1)over(order by month)))/lag(active_users,1)over(order by month))*100,2) as growth

FROM

(**SELECT**

EXTRACT(month FROM created_at) AS month,

count(activated_at) as active_users

FROM

users

WHERE activated_at NOT IN ('')

GROUP BY month

ORDER BY month) as sub;

Result:

month	active_users	growth
1	712	NULL
2	685	-3.79
3	765	11.68
4	907	18.56
5	993	9.48
6	1086	9.37
7	1281	17.96
8	1347	5.15
9	330	-75.50
10	390	18.18
11	399	2.31
12	486	21.80

C. Weekly Retention: Calculate the weekly retention of users-sign up cohort?

Query: with signup as (

```
SELECT
    user_id ,
    event_type ,
    event_name,
    extract(week From occurred_at) as signup_week
FROM events
WHERE event_type = 'SIGNUP_FLOW' ),
```

engagement as (

```
SELECT
    user_id ,
    event_type ,
    event_name,
    extract(week from occurred_at) as
    engaging_week
FROM events
WHERE event_type = 'engagement')
```

select

```
distinct e.user_id ,
s.signup_week,
e.engaging_week ,
```

(e.engaging_week- s.signup_week) as retention_week

from signup as s

join engagement as e

on e.user_id = s.user_id ;

Result:

user_id	signup_week	engaging_week	retention_week	user_id	signup_week	engaging_week	retention_week
12882	20	20	0	12741	20	20	0
12882	20	22	2	12741	20	22	2
12882	20	25	5	12741	20	24	4
12883	20	20	0	12742	20	20	0
12887	20	20	0	12743	20	20	0
12888	20	20	0	12744	20	20	0
12888	20	21	1	12745	20	20	0
12889	20	20	0	12747	20	20	0
12889	20	21	1	12748	20	20	0
12890	20	20	0	12749	20	20	0
12891	20	20	0	12751	20	20	0
12893	20	20	0	12752	20	20	0
12894	20	20	0	12753	20	20	0
12897	20	20	0	12753	20	22	2
12897	20	21	1	12754	20	20	0
12899	20	20	0	12757	20	20	0
12900	20	20	0	12758	20	20	0
12902	20	20	0	12759	20	20	0
12902	20	21	1	12761	20	20	0
12903	20	20	0	12764	20	20	0
12903	20	21	1	12770	20	20	0
12904	20	20	0	12771	20	20	0
12907	20	20	0	12772	20	20	0
12910	20	20	0	12772	20	23	3

user_id	signup_week	engaging_week	retention_week	user_id	signup_week	engaging_week	retention_week
12292	19	19	0	11768	17	17	0
12292	19	21	2	11770	17	17	0
12292	19	22	3	11775	17	17	0
12293	19	19	0	11775	17	18	1
12293	19	21	2	11778	17	17	0
12294	19	19	0	11778	17	21	4
12296	19	19	0	11778	17	23	6
12297	19	19	0	11779	17	17	0
12297	19	20	1	11780	17	17	0
12301	19	19	0	11785	17	17	0
12302	19	19	0	11787	17	17	0
12303	19	19	0	11787	17	18	1
12310	19	19	0	11787	17	19	2
12311	19	19	0	11791	17	17	0
12311	19	20	1	11793	17	17	0
12312	19	19	0	11795	17	17	0
12313	19	19	0	11795	17	18	1
12315	19	19	0	11798	17	17	0
12318	19	19	0	11799	17	17	0
12319	19	19	0	11799	17	20	3
12323	19	19	0	11801	17	17	0
12324	19	19	0	11804	17	17	0
12325	19	19	0	11806	17	17	0
12326	19	19	0	11809	17	17	0

D. Weekly Engagement: Calculate the weekly engagement per device?

Query: **select**

extract(week from occurred_at) as week_num,

device,

count(distinct user_id) as num_of_users

from

events

where event_type = 'engagement'

group by 1,2

order by 1,3

Result:

	week_num	device	num_of_users		week_num	device	num_of_users
►	17	amazon fire phone	4		19	mac mini	18
	17	kindle fire	6		19	kindle fire	21
	17	mac mini	6		19	acer aspire desktop	23
	17	samsung galaxy note	7		19	nokia lumia 635	23
	17	samsung galaxy tablet	8		19	nexus 10	25
	17	acer aspire desktop	9		19	asus chromebook	27
	17	windows surface	10		19	htc one	30
	17	hp pavilion desktop	14		19	dell inspiron desktop	36
	17	htc one	16		19	ipad mini	36
	17	nexus 10	16		19	hp pavilion desktop	40
	17	nokia lumia 635	17		19	acer aspire notebook	41
	17	dell inspiron desktop	18		19	nexus 7	41
	17	nexus 7	18		19	iphone 4s	44
	17	ipad mini	19		19	ipad air	55
	17	acer aspire notebook	20		19	iphone 5s	79
	17	asus chromebook	21		19	dell inspiron notebook	83
	17	iphone 4s	21		19	nexus 5	87
	17	ipad air	27		19	samsung galaxy s4	91
	17	nexus 5	40		19	macbook air	112
	17	iphone 5s	42		19	iphone 5	115
	17	dell inspiron notebook	46		19	lenovo thinkpad	178
	17	samsung galaxy s4	52		19	macbook pro	266
	17	macbook air	54		20	samsung galaxy tablet	9
	17	iphone 5	65		20	amazon fire phone	11
	17	lenovo thinkpad	86		20	samsung galaxy note	18
	17	macbook pro	143		20	windows surface	21
	18	amazon fire phone	9		20	nexus 10	22
	18	windows surface	10		20	nokia lumia 635	22

week_num	device	num_of_users	week_num	device	num_of_users
18	samsung galaxy tablet	11	20	acer aspire desktop	23
18	mac mini	13	20	kindle fire	23
18	samsung galaxy note	15	20	mac mini	26
18	htc one	19	20	htc one	29
18	acer aspire desktop	26	20	hp pavilion desktop	30
18	kindle fire	27	20	ipad mini	32
18	ipad mini	30	20	nexus 7	32
18	nexus 10	30	20	acer aspire notebook	40
18	nexus 7	30	20	asus chromebook	41
18	acer aspire notebook	33	20	dell inspiron desktop	52
18	nokia lumia 635	33	20	iphone 4s	55
18	hp pavilion desktop	37	20	ipad air	59
18	asus chromebook	42	20	iphone 5s	79
18	iphone 4s	46	20	dell inspiron notebook	84
18	ipad air	52	20	samsung galaxy s4	93
18	dell inspiron desktop	58	20	nexus 5	103
18	iphone 5s	73	20	macbook air	119
18	nexus 5	73	20	iphone 5	125
18	dell inspiron notebook	77	20	lenovo thinkpad	173
18	samsung galaxy s4	82	20	macbook pro	256
18	iphone 5	113	21	amazon fire phone	5
18	macbook air	121	21	samsung galaxy tablet	6
18	lenovo thinkpad	153	21	windows surface	17
18	macbook pro	252	21	mac mini	18
19	samsung galaxy tablet	6	21	samsung galaxy note	20
19	samsung galaxy note	11	21	htc one	21
19	amazon fire phone	12	21	ipad mini	23
19	windows surface	16	21	nexus 10	25
week_num	device	num_of_users	week_num	device	num_of_users
21	acer aspire desktop	29	23	asus chromebook	49
21	nexus 7	29	23	dell inspiron desktop	53
21	kindle fire	30	23	iphone 4s	53
21	asus chromebook	38	23	hp pavilion desktop	54
21	dell inspiron desktop	41	23	iphone 5s	79
21	hp pavilion desktop	44	23	nexus 5	88
21	iphone 4s	45	23	samsung galaxy s4	99
21	acer aspire notebook	47	23	dell inspiron notebook	103
21	ipad air	51	23	macbook air	124
21	iphone 5s	74	23	iphone 5	152
21	dell inspiron notebook	80	23	lenovo thinkpad	176
21	samsung galaxy s4	84	23	macbook pro	266
21	nexus 5	91	24	amazon fire phone	11
21	macbook air	110	24	samsung galaxy tablet	11
21	iphone 5	137	24	htc one	20
21	lenovo thinkpad	167	24	samsung galaxy note	20
21	macbook pro	247	24	windows surface	22
22	amazon fire phone	5	24	acer aspire desktop	24
22	samsung galaxy tablet	10	24	kindle fire	25
22	windows surface	15	24	mac mini	29
22	samsung galaxy note	19	24	nokia lumia 635	35
22	kindle fire	21	24	nexus 10	38
22	htc one	24	24	ipad mini	39
22	acer aspire desktop	25	24	acer aspire notebook	40
22	mac mini	25	24	asus chromebook	43
22	nokia lumia 635	25	24	nexus 7	49
22	nexus 10	27	24	iphone 4s	53
22	ipad mini	34	24	hp pavilion desktop	56

E. **Email Engagement:** Calculate the email engagement metrics?

Query: **select**

```
100*sum(case when email_category = 'email_open' then 1 else 0
end)/sum(case when email_category = 'email_sent' then 1 else 0 end)
as email_open_rate_metric,
100*sum(case when email_category = 'email_click' then 1 else 0
end)/sum(case when email_category = 'email_sent' then 1 else 0 end)
as email_click_rate_metric
from (
    select
    case
    when action ='sent_weekly_digest'or'sent_reengagement_email'
    then 'email_sent'
    when action ='email_open' then 'email_open'
    when action ='email_clickthrough' then 'email_click'
    End as Email_category
    from email_events) as a ;
```

Result:

Result Grid			Filter Rows:
	email_open_rate_metric	email_click_rate_metric	
▶	35.7256	15.7333	