

# **Operation Analytics and Investigating Metric Spike Project Report**

## **Description:**

Operation Analytics is a subset of business analytics that uses data analysis and business intelligence to improve efficiency and streamline everyday operations in real time. It involves examining the current and historical performance of operations and maintenance-related investments and measuring them across relevant metrics such as cost, schedules, budgets and other performance-related parameters. Operational analytics can be used for a number of different areas, such as marketing campaigns, product improvement, demand planning, supply chain management and asset utilization. It enables continuous monitoring of data and discovery of insights to help teams make better decisions on the fly.

Being one of the most important parts of a company, this kind of analysis is further used to predict the overall growth or decline of a company's fortune. It means better automation, better understanding between cross-functional teams, and more effective workflows. Investigating metric spike is also an important part of operation analytics as being a Data Analyst you must be able to understand or make other teams 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 it's very important to investigate metric spike.

## **Approach:**

I began by reviewing the raw data and understanding the task at hand. Next, I used MySQL Workbench to create a database and table. I then started working on my queries to achieve the desired results. After writing the queries, I executed them and checked for errors. If there were any errors, I modified the code to fix them. Once all queries were executed without errors, I saved the file and attached my code and prepared report using MS Word.

## **Tech-Stack Used:**

I used MySQL Workbench v8.0.32 during the project execution to query the database and perform SQL operations. By using MySQL Workbench, I was able to create a database and table that met the requirements of the task. This allowed me to write queries that were tailored to the specific needs of the task.

In addition to MySQL Workbench, I also used Microsoft Word 2019 to create documentation and prepare reports. This tool allowed me to effectively communicate the

results of my work. By using these tools together, I was able to successfully complete the project and achieve all of its objectives.

## Conclusions:

From the results, I can draw the following conclusions:

1. The user growth percentage shows some variation, but the overall\_growth trend is positive.
2. I notice some duplicate data in the job\_data, which could be a problem.
3. The user is using the product/service frequently, which indicates the high quality of the product/service.
4. The users are responding well to the email service, as shown by the email engagement metric results.
5. The product/service is compatible with different devices (platforms), which is a good sign."

## Results:

### Case Study 1 (Job Data):

A. **Number of jobs reviewed:** Amount of jobs reviewed over time.

**Your task:** 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

**Explanation:** The query selects the number of distinct job IDs reviewed in each day between November 1st and November 30th, 2020. The count is divided by 720 (30 days \* 24 hours) to get the average number of jobs reviewed per hour for each day. The results are ordered by date in descending order.

### O/P:

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:** It is the no. of events happening per second.

**Your task:** Let's say the above metric is called 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

**Explanation:** The query calculates the average throughput of jobs for each day between November 1st and November 30th, 2020. The average is calculated over a rolling window of 7 days. The results are ordered by date in descending order.

**O/P:** 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:** Share of each language for different contents.

**Your task:** 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

**Explanation:** The query selects the language column from the jobs table and calculates the percentage share of each language in the table. The percentage share is calculated by dividing the count of each language by the total number of distinct languages in the table and multiplying by 100. The results are grouped by language

**O/P:**

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:** Rows that have the same value present in them.

**Your task:** Let's say you see some duplicate rows in the data. 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

**Explanation:** The query selects all columns from the jobs table and assigns a row number to each row within each job\_id partition. The query then selects all rows where the row number is greater than 1. This effectively removes the first row for each job\_id.

**O/P:**

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 Study-2 (Investigating Metric Spike):

A. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.

**Your task:** 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

**Explanation:** The query selects the week number from the created\_at column of the users table and counts the number of distinct user IDs for each week. The results are grouped by week number.

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

O/P:

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:** Amount of users growing over time for a product.

**Your task:** 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;

**Explanantion:** The query selects the month and the number of active users for each month from the users table. The query then calculates the percentage growth in active users from the previous month using the lag function. The results are ordered by month

**O/P:**

Result Grid     Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 			
	month	active_users	growth
1	1	712	NULL
2	2	685	-3.79
3	3	765	11.68
4	4	907	18.56
5	5	993	9.48
6	6	1086	9.37
7	7	1281	17.96
8	8	1347	5.15
9	9	330	-75.50
10	10	390	18.18
11	11	399	2.31
12	12	486	21.80

C. **Weekly Retention:** Users getting retained weekly after signing-up for a product.  
**Your task:** 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

**Explanantion:** The query selects the user\_id, signup\_week, engaging\_week and retention\_week for each user who has signed up and engaged with the platform. The retention\_week is calculated as the difference between the signup\_week and the engaging\_week for each user.

**O/P:**

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:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.

**Your task:** 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

**Explanation:** The query selects the week number, device and the number of distinct users who engaged with the platform for each week. The results are grouped by week number and device and ordered by week number and the number of users.

**O/P:**

	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:** Users engaging with the email service.



**Your task:** 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

**Explanation:** The query calculates the email open rate and email click rate metrics for the email events. The email open rate metric is calculated as the percentage of email opens divided by the number of emails sent. The email click rate metric is calculated as the percentage of email clicks divided by the number of emails sent.

**O/P:**

Result Grid     Filter Rows: <input type="text"/>		
	email_open_rate_metric	email_click_rate_metric
▶	35.7256	15.7333

Result Grid     Filter Rows: <input type="text"/>		
	email_open_rate_metric	email_click_rate_metric
▶	35.7256	15.7333