

Operation Analytics and Investigating Metric Spike

Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect.

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 its very important to investigate metric spike.

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

You are required to provide a detailed report for the below two operations mentioning the answers for the related questions:

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?

For this, we need to find the count of jobs based on job_id , per hour per day . As we have 30 days in Nov, 24 hours in a day, we will divide the count with 30*24 and select the output. Here we can do this either with just count or distinct count of Job_id's .

i. Count of job id's (without distinct)

Here we are calculating the number of jobs reviewed per hour per day of Nov 2022

Query :

```
Select Count(job_id)/(30*24) as Total_Jobs_PerHour_PerDay_Nov2020  
from Sample.Job_data;
```

Output :

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Total_Jobs_PerHour_PerDay_Nov2020			
▶	0.0111			

ii. Distinct Count of Job_id's

Here we are calculating the number of distinct jobs reviewed per hour per day of Nov 2022

Query:

**Select Count(distinct job_id)/(30*24) as
Distinct_Jobs_PerHour_PerDay_Nov2020**

from Sample.Job_data;

Output :

Result Grid		Filter Rows:
	Distinct_Jobs_PerHour_PerDay_Nov2020	
▶	0.0083	

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?

Throughput is the measure of number of tasks done per a specific time spent. 7 day rolling average of throughput is the preferable over daily metrics as we can get the cumulative data per week, which might be useful and relevant over daily metrics.

To calculate this, we first need to calculate Total Job count and total time_spent for Job data.

Next we can use that data to find the rolling average of the Throughput by ordering them , with rows between 6 preceding and current row in order to get 7 day rolling average of Job count per time spent .

Query:

Select ds,Job_count, Total_time_spent,

Sum(Job_Count) over (order by ds rows between 6 preceding and current row)/Sum(Total_time_spent) over (order by ds rows between 6 preceding and current row)

as Throughput_7DayRollingAvg

from (

Select ds,count(job_id) as Job_count ,

Sum(time_spent) as Total_time_spent

from Sample.Job_data

group by ds

order by ds) Sub;

Output :

Result Grid				
		Filter Rows:	Export:	Wrap Cell Content:
	ds	Job_count	Total_time_spent	Throughput_7DayRollingAvg
▶	2020-11-25	1	45	0.0222
	2020-11-26	1	56	0.0198
	2020-11-27	1	104	0.0146
	2020-11-28	2	33	0.0210
	2020-11-29	1	20	0.0233
	2020-11-30	2	40	0.0268

Result 18 x

Distinct Job Count Throughput :

Here the only difference is that we are calculating distinct count of jobs.

Query:

Select ds, Distinct_Job_count, Total_time_spent,

Sum(Distinct_Job_Count) over (order by ds rows between 6 preceding and current row)/Sum(Total_time_spent) over (order by ds rows between 6 preceding and current row)

as Throughput_7DayRollingAvg

from (

Select ds,count(distinct job_id) as Distinct_Job_count ,

Sum(time_spent) as Total_time_spent

from Sample.Job_data

group by ds

order by ds) Sub;

Output :

Result Grid				
		Filter Rows:	Export:	Wrap Cell Content:
	ds	Distinct_Job_count	Total_time_spent	Throughput_7DayRollingAvg
▶	2020-11-25	1	45	0.0222
	2020-11-26	1	56	0.0198
	2020-11-27	1	104	0.0146
	2020-11-28	2	33	0.0210
	2020-11-29	1	20	0.0233
	2020-11-30	2	40	0.0268

Result 20 ×

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?

For this, we need to divide the count of each language by the total number of languages present , and calculate it's percentage by multiplying it with 100.

Query:

Select Language,

count(language) as Language_Count ,

**(count(language)/(select count(*) from Sample.Job_data))*100 as
Percentage_share_ofEach_language**

from Sample.Job_data

Group by language;

Output:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Language	Language_Count	Percentage_share_ofEach_language	
English	1	12.5000	
Arabic	1	12.5000	
Persian	3	37.5000	
Hindi	1	12.5000	
French	1	12.5000	
Italian	1	12.5000	

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?

We can calculate the duplicate rows by using Row_Number() function . we can add a new column named row_num , which is the partiion on basis of Job_id , which is unique column in Job_data table with row_number() function. With that we can get the number of each row. So If the row number >1 , that means it is a duplicate row in the table.

Query:

Select *




from (

Select * , row_number() over (partition by Job_ID) Row_Num

from Sample.job_data) R

where Row_num > 1;

Output:

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 								
	ds	job_id	actor_id	event	language	time_spent	org	Row_Num
▶	2020-11-28	23	1005	transfer	Persian	22	D	2
	2020-11-26	23	1004	skip	Persian	56	A	3

We have 2 duplicate rows in the given table , with same Job_ID = 23.

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?

Here we need to find the number of users are active , in each week . For this, first we need to extract the week number from the occurred_at column of the events table, and then count the number of distinct users are there with event_type as engagement grouping it by week number.

Query:

Select Extract (Week from occurred_at) as Week_number ,

Count(distinct user_id) as Num_of_users

from Sample.events

where event_type = 'engagement'

group by Week_number;

Output :

Result Grid			Filter Rows:
	Week_number	Num_of_users	
▶	17	85	
	18	80	
	19	14	
	20	5	
	21	1	
	22	1	
	23	2	

B. User Growth: Amount of users growing over time for a product.

Your task: Calculate the user growth for product?

Here we are extracting the year, month, week and calculate the count of distinct users who are active, based on the user_id field. And finally we can use that data to know the cumulative active users by ordering it by year, month, week, over rows from all its top rows to that current row.

Query:

Select activation_year, activation_month, activation_week,

Active_users,

Sum(Active_users) over (order by activation_year, activation_month, activation_week
rows between unbounded preceding and current row) as Cumulative_active_users

from (

Select Extract(Year from activated_at) as activation_year ,

Extract(Month from activated_at) as activation_month ,

Extract(Week from activated_at) as activation_week,





Count(distinct user_id) as Active_users

from Sample.users





where state='Active'





Group by activation_year , activation_month, activation_week) Sub;





Output :

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 					
	activation_year	activation_month	activation_week	Active_users	Cumulative_active_users
▶	2013	1	0	23	23
	2013	1	1	30	53
	2013	1	2	48	101
	2013	1	3	36	137
	2013	1	4	23	160
	2013	2	4	7	167
	2013	2	5	48	215
	2013	2	6	38	253
	2013	2	7	42	295
	2013	2	8	25	320
	2013	3	8	9	329
	2013	3	9	43	372
	2013	3	10	32	404
	2013	3	11	31	435
	2013	3	12	33	468
	2013	3	13	2	470
	2013	4	13	37	507
	2013	4	14	35	542
	2013	4	15	43	585
	2013	4	16	46	631

Result 28 x

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 					
	activation_year	activation_month	activation_week	Active_users	Cumulative_active_users
	2013	5	17	29	680
	2013	5	18	44	724
	2013	5	19	57	781
	2013	5	20	39	820
	2013	5	21	45	865
	2013	6	21	4	869
	2013	6	22	54	923
	2013	6	23	50	973
	2013	6	24	45	1018
	2013	6	25	57	1075
	2013	6	26	3	1078
	2013	7	26	53	1131
	2013	7	27	52	1183
	2013	7	28	72	1255
	2013	7	29	67	1322
	2013	7	30	40	1362
	2013	8	30	27	1389
	2013	8	31	67	1456
	2013	8	32	71	1527
	2013	8	33	73	1600

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 					
	activation_year	activation_month	activation_week	Active_users	Cummulative_active_users
	2013	8	32	71	1527
	2013	8	33	73	1600
	2013	8	34	78	1678
	2013	9	35	63	1741
	2013	9	36	72	1813
	2013	9	37	85	1898
	2013	9	38	90	1988
	2013	9	39	20	2008
	2013	10	39	64	2072
	2013	10	40	87	2159
	2013	10	41	73	2232
	2013	10	42	99	2331
	2013	10	43	67	2398
	2013	11	43	22	2420
	2013	11	44	96	2516
	2013	11	45	91	2607
	2013	11	46	88	2695
	2013	11	47	102	2797
	2013	12	48	97	2894
	2013	12	49	116	3010

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 					
	activation_year	activation_month	activation_week	Active_users	Cummulative_active_users
	2013	12	49	116	3010
	2013	12	50	124	3134
	2013	12	51	102	3236
	2013	12	52	47	3283
	2014	1	0	83	3366
	2014	1	1	126	3492
	2014	1	2	109	3601
	2014	1	3	113	3714
	2014	1	4	121	3835
	2014	2	4	9	3844
	2014	2	5	133	3977
	2014	2	6	135	4112
	2014	2	7	125	4237
	2014	2	8	123	4360
	2014	3	8	6	4366
	2014	3	9	133	4499
	2014	3	10	154	4653
	2014	3	11	130	4783
	2014	3	12	148	4931
	2014	3	13	44	4975

	activation_year	activation_month	activation_week	Active_users	Cumulative_active_users
	2014	4	14	162	5260
	2014	4	15	164	5424
	2014	4	16	179	5603
	2014	4	17	98	5701
	2014	5	17	72	5773
	2014	5	18	163	5936
	2014	5	19	185	6121
	2014	5	20	176	6297
	2014	5	21	183	6480
	2014	6	22	196	6676
	2014	6	23	196	6872
	2014	6	24	229	7101
	2014	6	25	207	7308
	2014	6	26	45	7353
	2014	7	26	156	7509
	2014	7	27	222	7731
	2014	7	28	215	7946
	2014	7	29	221	8167
	2014	7	30	183	8350
	2014	8	30	55	8405

2014	8	30	55	8405
2014	8	31	193	8598
2014	8	32	245	8843
2014	8	33	261	9104
2014	8	34	259	9363
2014	8	35	18	9381

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

Your task: Calculate the weekly retention of users-sign up cohort?

We can calculate this by first calculating the number of users engaging and number of users signed up , and join them using the left join to calculate the retention week and then we can sum the retention week number, which is the difference between engagement week number and signup week number, to get the per_week retention.

Query:

Select user_id , count(user_id) as users_count ,

Sum(case when retention_week = 1 then 1 else 0 end) as per_week_retention

from(

Select signUp.user_id, Engagement.Week_engagement , Signup.week_signup ,

Engagement.Week_engagement - Signup.week_signup as retention_week

```

from (

(select distinct User_id , extract( Week from occurred_at) as Week_Signup

from Sample.Events

where event_type= 'signup_flow' and event_name = 'complete_signup' ) SignUp

left join

(select distinct user_id , extract( Week from occurred_at) as Week_Engagement

from Sample.Events

where event_type = 'Engagement' ) Engagement

on signUp.user_id = Engagement.user_id

) ) sub

group by user_id

order by user_id;

```

Output :

Please find the output file in the below link

https://drive.google.com/file/d/1N0m4JA0ttVvsLD-VXqOxy6UrzfkbtSEa/view?usp=share_link

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?

Here we can extract the Year, week from the occurred_at field of the events table, and then count the distinct users , whose event type is engagement, and we can group them by year,week ,device.

Query :

```

Select Extract(Year from occurred_at) as Year_num,

Extract(week from occurred_at) as Week_num, device,

count(distinct user_id) as Distinct_User_count

from sample.events

```

```
where event_type = 'engagement'
```

```
group by year_num, week_num ,device
```

```
order by week_num, Distinct_User_count ;
```

Output :

Please find the output file in the below link

https://drive.google.com/file/d/13RF1JY3BzteGyYqfYbSV-k5SaJ_hgHeb/view?usp=share_link

E. Email Engagement: Users engaging with the email service.

Your task: Calculate the email engagement metrics?

For this we can first get the email status of each row in the email_events table with the case statements. And then based on that we can calculate the percentage of users opening the emails and percentage of users clicking on the emails.

Query :

Select

```
Sum(case when email_status = 'email_opened' then 1 else 0 end)/
```

```
Sum(case when email_status = 'email_sent' then 1 else 0 end) *100.0 as  
Email_opening_Percentage,
```

```
Sum(case when email_status = 'email_clicked' then 1 else 0 end)/
```

```
Sum(case when email_status = 'email_sent' then 1 else 0 end) *100.0 as  
Email_clicking_Percentage
```

```
from(
```

```
Select *, case
```

```
when action in ('sent_weekly_digest','sent_reengagement_email') then  
'email_sent'
```

```
when action = 'email_open' then 'email_opened'
```

```
when action = 'email_clickthrough' then 'email_clicked'
```

```
else 'None'
```

```
end as email_status
```

```
from sample.email_events) Sub;
```

Output :

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	Email_opening_Percentage	Email_clicking_Percentage			
▶	33.58339	14.78989			

Tech-Stack Used : SQL Workbench - mysql Ver 8.0.33

Thank you,

G A R Sireesha.