



Operation Analytics and Investigating Metric Spike



CONTENTS

01 Description

02 Approach

03 Tech Stack

04 Insights

Description

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.

Approach

The data set is observed and studied to understand the different categories and parameters of the data provided.

This data set is executed to draw useful insights on the business proceedings

Tech-Stack Used

- MySQL workbench is used to run all the queries in this project.
- This software is very handy to manage the dataset and draw useful insights.

Insights

- Learnt how to write window function
- How to join two query like with the assistance of window function.
- Finally, I got to know that there are various function that help to generate data according to group and order with window function.

Case Study 1

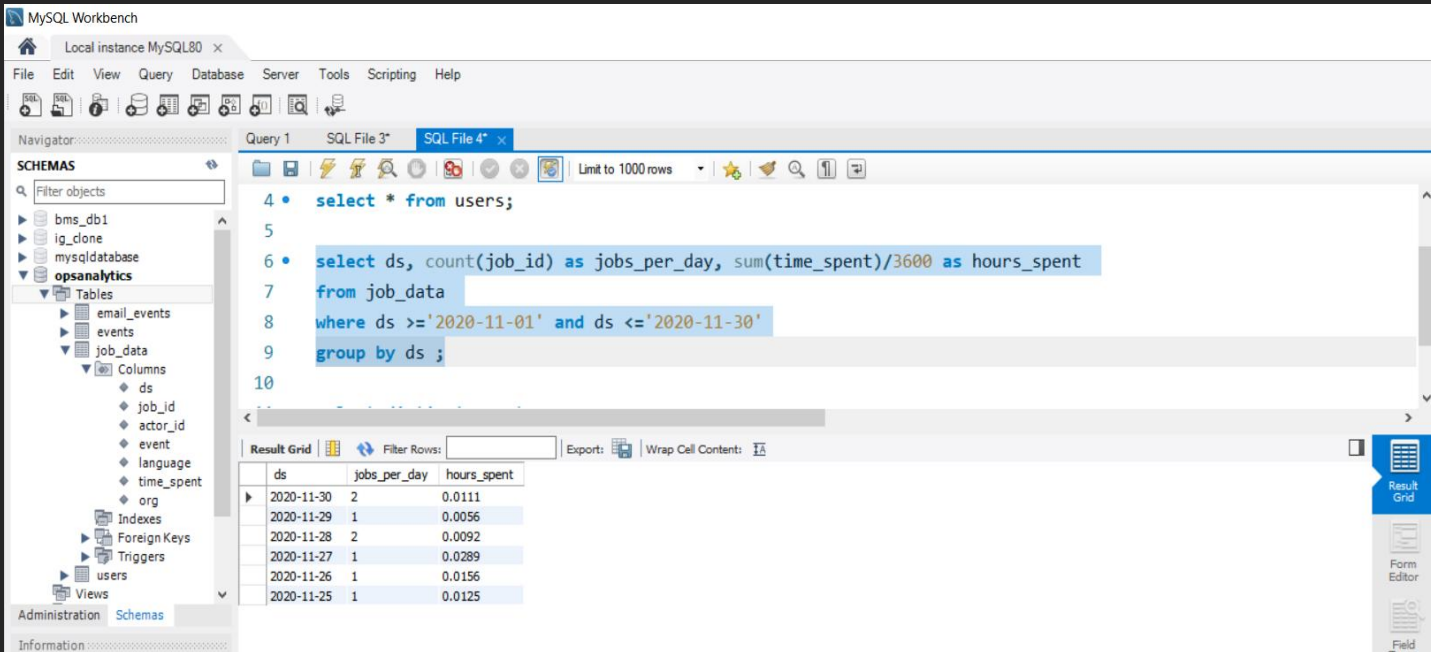
Job Data

Number of jobs reviewed

Your Task: Calculate the number of jobs reviewed per hour per day for November 2020?

SQL :

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



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' panel displays a tree view of databases, including 'bms_db1', 'ig_clone', 'mysqldatabase', and 'opsanalytics'. The 'opsanalytics' database is selected, and its 'Tables' are listed, including 'email_events', 'events', 'job_data', and 'users'. The 'job_data' table is highlighted. The main editor window shows a SQL query in 'Query 1':

```
4 • select * from users;  
5  
6 • select ds, count(job_id) as jobs_per_day, sum(time_spent)/3600 as hours_spent  
7   from job_data  
8   where ds >='2020-11-01' and ds <='2020-11-30'  
9   group by ds ;  
10
```

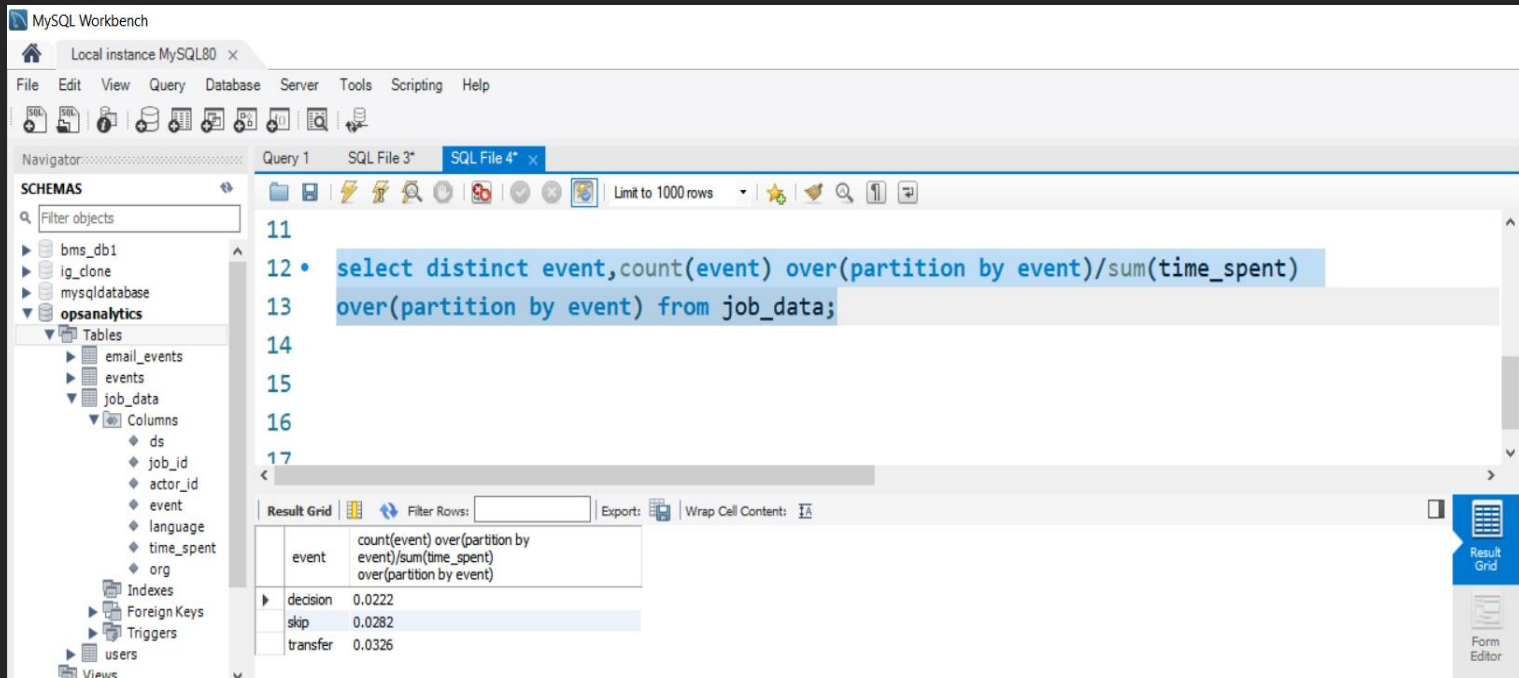
The 'Result Grid' at the bottom displays the results of the query, showing a table with three columns: 'ds', 'jobs_per_day', and 'hours_spent'. The data is as follows:

ds	jobs_per_day	hours_spent
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

Throughput

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?

```
select distinct event, count(event) over(partition by event)/sum(time_spent)
over(partition by event) from job_data;
```



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'opsanalytics' database selected, showing tables like 'email_events', 'events', 'job_data', and 'users'. The main query editor displays the following SQL query:

```
11
12 • select distinct event, count(event) over(partition by event)/sum(time_spent)
13    over(partition by event) from job_data;
14
15
16
17
```

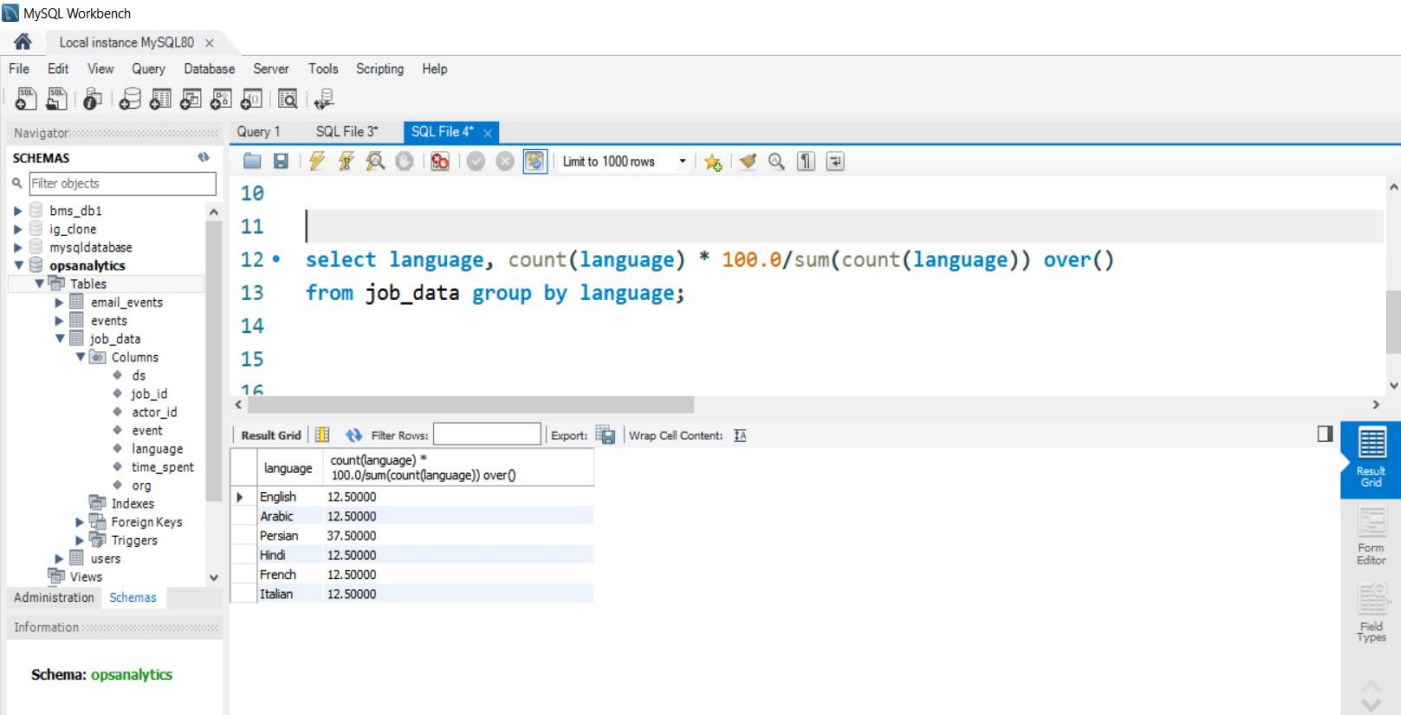
The query results are displayed in the 'Result Grid' at the bottom, showing the following data:

event	count(event) over(partition by event)/sum(time_spent) over(partition by event)
decision	0.0222
skip	0.0282
transfer	0.0326

Percentage share of each language

Your Task: Calculate the percentage share of each language in the last 30 days?

SQL : `select language, count(language) * 100.0/sum(count(language)) over() • from job_data group by language;`



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'opsanalytics' database selected. The main editor window contains the following SQL query:

```
10  
11  
12 • select language, count(language) * 100.0/sum(count(language)) over()  
13 from job_data group by language;  
14  
15  
16
```

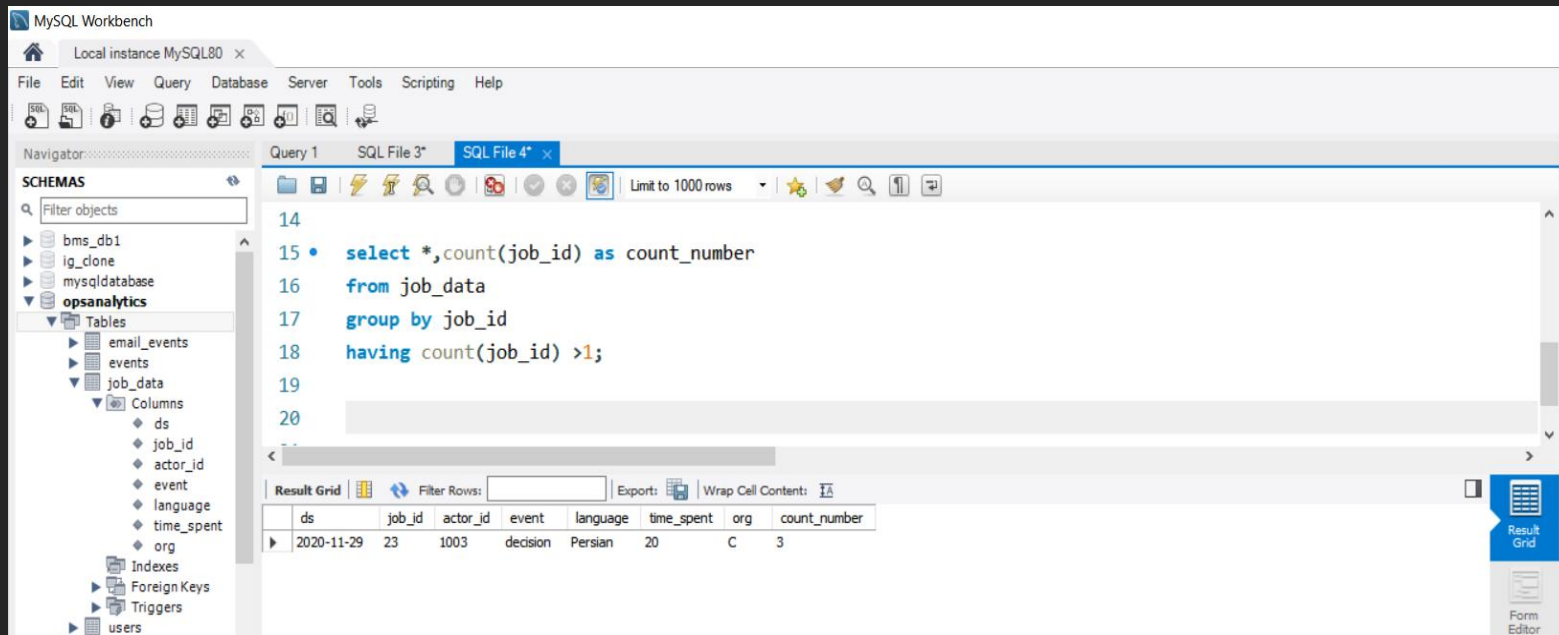
Below the query editor, the 'Result Grid' is visible, showing the results of the query. The table has two columns: 'language' and 'count(language) * 100.0/sum(count(language)) over()'. The results are as follows:

language	count(language) * 100.0/sum(count(language)) over()
English	12.50000
Arabic	12.50000
Persian	37.50000
Hindi	12.50000
French	12.50000
Italian	12.50000

Duplicate rows

Your Task: Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

SQL : `select *,count(job_id) as count_number from job_data group by job_id having count(job_id) >1;`



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases and tables. The 'job_data' table is selected under the 'opsanalytics' database. The main editor window shows a SQL query:

```
14
15 • select *,count(job_id) as count_number
16 from job_data
17 group by job_id
18 having count(job_id) >1;
19
20
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has columns: ds, job_id, actor_id, event, language, time_spent, org, and count_number. The first row of data is:

ds	job_id	actor_id	event	language	time_spent	org	count_number
2020-11-29	23	1003	decision	Persian	20	C	3

Case Study 2

Investigating metric spike

User Engagement

Your Task: Calculate the weekly user engagement?

SQL : `select week(occurred_at) as week , count(user_id) as user_engage from events group by week(occurred_at);`

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, including 'bms_db1', 'ig_clone', 'mysqldb', and 'opsanalytics'. The 'opsanalytics' database is selected, showing its tables: 'email_events', 'events', and 'job_data'. The 'job_data' table is expanded, showing columns like 'ds', 'job_id', 'actor_id', 'event', 'language', 'time_spent', 'org', 'Indexes', 'Foreign Keys', 'Triggers', 'users', and 'Views'. The 'Administration' tab is active, showing 'Schema: opsanalytics'.

The main query editor shows the following SQL query:

```
select week(occurred_at) as week , count(user_id) as user_engage
from events group by week(occurred_at);
```

The 'Result Grid' pane displays the results of the query, showing a table with two columns: 'week' and 'user_engage'. The results are as follows:

week	user_engage
17	689
18	1661
19	1737
20	1756
21	1795
23	1991
22	2066
24	2243
25	2005
29	1973
26	2164
30	2340
28	2125
27	2210
31	2041
32	2210
33	1481
34	305

The status bar at the bottom indicates 'Result 20' and 'Read Only'.

User Growth

Your Task: Calculate the user growth for product? (couldn't solved by myself)

SQL : select distinct user_id,device,week(occurred_at) as a,count(device)over(partition by device) from events order by a,device;

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' panel displays a tree view of databases, with 'opsanalytics' selected. The 'Tables' list under 'opsanalytics' includes 'email_events', 'events', and 'job_data'. The 'Columns' list for 'job_data' includes 'ds', 'job_id', 'actor_id', 'event', 'language', 'time_spent', 'org', 'Indexes', 'Foreign Keys', 'Triggers', and 'users'. The 'Administration' tab is active, showing 'Schema: opsanalytics'.

The main query editor displays the following SQL query:

```
22
23 • select distinct user_id,device,week(occurred_at) as a,count(device) over(partition by device)
24   from events order by a,device;
25
```

The 'Result Grid' tab is active, showing the results of the query. The table has four columns: 'user_id', 'device', 'a', and 'count(device) over(partition by device)'. The results are as follows:

user_id	device	a	count(device) over(partition by device)
11900	acer aspire desktop	17	442
11864	acer aspire desktop	17	442
11894	acer aspire notebook	17	830
11497	acer aspire notebook	17	830
11859	amazon fire phone	17	266
11818	asus chromebook	17	1045
11834	asus chromebook	17	1045
11865	asus chromebook	17	1045
11841	dell inspiron desktop	17	841
10522	dell inspiron notebook	17	1688
11850	dell inspiron notebook	17	1688
11809	dell inspiron notebook	17	1688
11838	dell inspiron notebook	17	1688
11811	hp pavilion desktop	17	850
11896	hp pavilion desktop	17	850
11888	htc one	17	384
11840	htc one	17	384
11823	ipad air	17	1307

Weekly Retention

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

SQL : select distinct user_id, count(a) over(partition by user_id) as weekly_retention
from(select distinct user_id, week(occurred_at) as a from events order by user_id) as c;

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'opsanalytics' selected. The main editor shows a SQL query for calculating weekly retention. The 'Result Grid' at the bottom displays the query results.

SQL Query:

```
25  
26 • select distinct user_id, count(a) over(partition by user_id) as weekly_retention  
27 from(select distinct user_id, week(occurred_at) as a  
28 from events order by user_id) as c;
```

Result Grid:

user_id	weekly_retention
10522	1
10612	1
10736	1
10965	1
11020	1
11037	1
11040	1
11133	1
11194	1
11212	1
11215	1
11227	1
11231	1
11240	1
11261	1
11284	1
11301	1
11308	2
11364	1

Weekly Engagement

Your Task: Calculate the weekly engagement per device?

SQL : select week(occurred_at) as week , count(device) over(partition by device) as device_engage
from events group by week(occurred_at) order by week;

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'opsanalytics' schema selected. The main editor window shows a SQL query in 'Query 1' (SQL File 4*). The query is:

```
25  
26 • select week(occurred_at) as week , count(device) over(partition by device) as device_engage  
27 from events group by week(occurred_at) order by week;  
28
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has two columns: 'week' and 'device_engage'. The results are as follows:

week	device_engage
17	1
18	2
19	1
20	5
21	5
22	1
23	5
24	1
25	5
26	1
27	1
28	2
29	1
30	5
31	1
32	2
33	1
34	2

The status bar at the bottom indicates 'Result 24' and 'Read Only'.

Email Engagement

Your Task: Calculate the email engagement metrics?

SQL : select distinct user_id, action, count(action) over(partition by action) as total_email_engaging_service from email_events order by user_id;

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases and tables. The 'opsanalytics' database is selected, showing tables like 'email_events', 'events', and 'job_data'. The 'Columns' pane for 'email_events' is visible, listing columns such as 'user_id', 'action', 'ds', 'job_id', 'actor_id', 'event', 'language', 'time_spent', 'org', 'Indexes', 'Foreign Keys', 'Triggers', 'users', and 'Views'. The main query editor displays the following SQL query:

```
28
29 • select distinct user_id, action, count(action)
30   over(partition by action) as total_email_engaging_service
31   from email_events order by user_id;
```

Below the query editor, the 'Result Grid' shows the results of the query. The grid has three columns: 'user_id', 'action', and 'total_email_engaging_service'. The results are as follows:

user_id	action	total_email_engaging_service
0	email_open	317
0	sent_weekly_digest	1022
4	email_clickthrough	97
4	email_open	317
4	sent_weekly_digest	1022
8	email_clickthrough	97
8	email_open	317
8	sent_weekly_digest	1022
11	email_clickthrough	97
11	email_open	317
11	sent_weekly_digest	1022
17	email_clickthrough	97
17	email_open	317
17	sent_weekly_digest	1022
19	email_clickthrough	97
19	email_open	317
19	sent_weekly_digest	1022
20	email_clickthrough	97
20	email_open	317

The status bar at the bottom indicates 'Result 25' and 'Read Only'.



THANK YOU