

Operation Analytics and Investigating Metric Spike

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.

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.

Approach:

For the first case study, the number of jobs reviewed can be calculated by counting the number of rows in the job_data table. The number of jobs reviewed per hour per day for November 2020 can be calculated by filtering the job_data table for November 2020 and then aggregating the number of jobs by hour and day. The throughput can be calculated by counting the number of events per second and then computing the 7-day rolling average of the throughput. For the throughput metric, the 7-day rolling average is preferred because it helps to smooth out fluctuations in the data and provides a clearer picture of the underlying trend. The percentage share of each language can be calculated by aggregating the number of jobs by language and then dividing each count by the total number of jobs. To display duplicates from the table, one can group the data by all columns and then filter for groups with more than one row.

For the second case study, the weekly user engagement can be calculated by aggregating the number of events in the events table by week and user. The user growth for the product can be calculated by counting the number of unique users in the users table over time. The weekly

retention of users-sign up cohort can be calculated by dividing the number of users who return after signing up by the number of users who signed up in a given week. The weekly engagement per device can be calculated by aggregating the number of events in the events table by week, device, and user. The email engagement metrics can be calculated by aggregating the number of email events in the email_events table by week and user.

Tech-Stack Used :

My SQL Workbench 8.0 CE

Insights :

The approach for this SQL project would involve filtering, aggregating, and joining the data as necessary to answer the questions posed. The insights from the project would depend on the results obtained from the analysis, but could potentially provide information on the performance of different operations and the engagement of different users and devices.

Result :

Case Study 1 (Job Data)

Table-1: job_data

job_id: unique identifier of jobs

actor_id: unique identifier of actor

event: decision/skip/transfer

language: language of the content

time_spent: time spent to review the job in seconds

org: organization of the actor

ds: date in the yyyy/mm/dd format. It is stored in the form of text and we use presto to run. no need for date function

Task 1 :

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?

SELECT

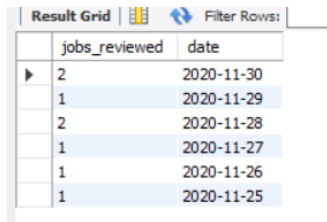
COUNT(job_id) AS jobs_reviewed,

DATE(ds) as date

FROM job_data

WHERE DATE(ds) BETWEEN '2020-11-01' AND '2020-11-30'

GROUP BY date ;



	jobs_reviewed	date
▶	2	2020-11-30
	1	2020-11-29
	2	2020-11-28
	1	2020-11-27
	1	2020-11-26
	1	2020-11-25

Task 2 :

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?

SELECT

event ,

```
weekofyear(ds) AS weeks ,
```

```
count(*) AS frequency
```

```
from job_data
```

```
group by 1,2;
```

Result Grid				Filter Rows:
	event	weeks	frequency	
▶	skip	49	1	
	transfer	49	1	
	decision	48	3	
	transfer	48	2	
	skip	48	1	

From this data, we can understand that there were a total of three events over the course of 49 weeks: skip, transfer, and decision. The frequency of each event is also given. Skip and transfer occurred once each week, while decision occurred three times a week.

Task 3 :

Percentage share of each language: Share of each language for different contents.

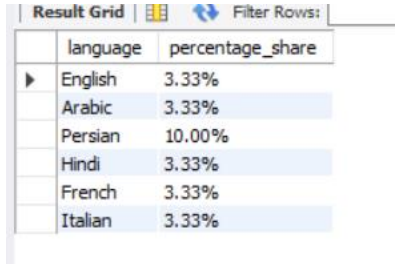
Your task: Calculate the percentage share of each language in the last 30 days?

```
select
```

```
language ,
```

```
CONCAT(ROUND((count(*)/30)*100,2),"%") as percentage_share
```

```
from job_data
group by 1 ;
```



	language	percentage_share
▶	English	3.33%
	Arabic	3.33%
	Persian	10.00%
	Hindi	3.33%
	French	3.33%
	Italian	3.33%

Task 4 :

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?

with result as

(

select

*,

row_number() over (partition by ds,job_id,actor_id,event,language,time_spent,org)
as num




from

job_data

)

select ds,job_id,actor_id,event,language,time_spent,org from result

where num>1 ;

Result Grid		Filter Rows:	<input type="text"/>	Export:		Wrap Cell Content:	
	ds	job_id	actor_id	event	language	time_spent	org

There is no duplicate rows

Case Study 2 (Investigating metric spike)

Table-1: users

This table includes one row per user, with descriptive information about that user’s account.

Table-2: events

This table includes one row per event, where an event is an action that a user has taken. These events include login events, messaging events, search events, events logged as users progress through a signup funnel, events around received emails.

Table-3: email_events

This table contains events specific to the sending of emails. It is similar in structure to the events table above.

Task 1 :

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?

```
select weekofyear(created_at) as weekly ,  
       count(user_id) as num_users from users where state = 'active'  
group by 1 ;
```

Result Grid		Filter F
	weekly	num_users
1	1	158
2	2	151
3	3	159
4	4	149
5	5	160
6	6	180
7	7	176
8	8	166
9	9	160
10	10	178
11	11	185
12	12	164
13	13	184
14	14	201
15	15	201
16	16	207
17	17	224
18	18	220
19	19	205
20	20	241
21	21	218
22	22	235
23	23	248
24	24	249

There ae 52 rows

Task 2 :

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

Your task: Calculate the user growth for product?

SELECT

DATE(u.created_at) AS date, e.event_name ,

COUNT(DISTINCT(u.user_id)) AS user_count

FROM

users u

JOIN events e ON u.user_id = e.user_id

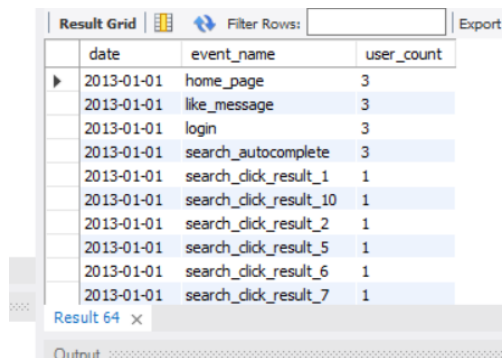
WHERE

u.state = 'active'

GROUP BY 1,2

ORDER BY

date ASC;



The screenshot shows a database query result grid with the following data:

	date	event_name	user_count
▶	2013-01-01	home_page	3
	2013-01-01	like_message	3
	2013-01-01	login	3
	2013-01-01	search_autocomplete	3
	2013-01-01	search_click_result_1	1
	2013-01-01	search_click_result_10	1
	2013-01-01	search_click_result_2	1
	2013-01-01	search_click_result_5	1
	2013-01-01	search_click_result_6	1
	2013-01-01	search_click_result_7	1

Below the table, there is a tab labeled "Result 64" and an "Output" section.

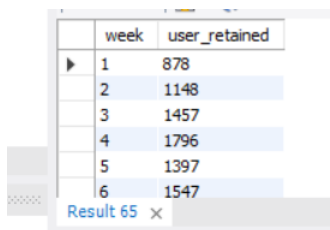
There are 1851 rows .

Task 3 :

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

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

```
select weekofyear(u.created_at) as week , count(e.user_id) as user_retained
from users u
left join
events e
on
u.user_id = e.user_id
where u.state = 'active'
group by 1
order by 1;
```



The screenshot shows a database query result with two columns: 'week' and 'user_retained'. The data is as follows:

week	user_retained
1	878
2	1148
3	1457
4	1796
5	1397
6	1547

Below the table, there is a tab labeled 'Result 65' with a close button 'x'.

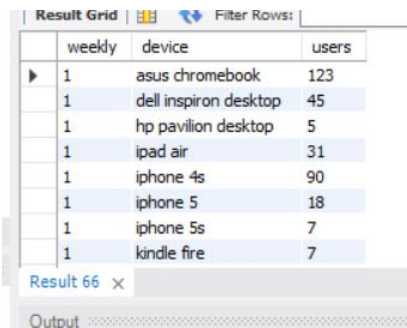
The number of rows are 52 .

Task 4 :

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?

```
select weekofyear(u.created_at) as weekly , e.device , count(u.user_id) as users
from events e
right join
users u
on
e.user_id = u.user_id
where e.event_type = 'engagement'
group by 1,2
order by 1;
```



The screenshot shows a 'Result Grid' window with a table containing 9 rows of data. The columns are 'weekly', 'device', and 'users'. The data is as follows:

weekly	device	users
1	asus chromebook	123
1	dell inspiron desktop	45
1	hp pavilion desktop	5
1	ipad air	31
1	iphone 4s	90
1	iphone 5	18
1	iphone 5s	7
1	kindle fire	7

Below the table, there is a tab labeled 'Result 66' and an 'Output' section.

The number of rows are 572

Task 5 :

Email Engagement: Users engaging with the email service.

Your task: Calculate the email engagement metrics?

```
SELECT u.user_id ,  
count(case when ee.action = 'sent_weekly_digest' then 1 else null end) as  
num_sent_weekly_digest ,  
  
count(case when ee.action = 'email_open' then 1 else null end) as num_email_open ,  
  
count(case when ee.action = 'email_clickthrough' then 1 else null end) as  
num_email_clickthrough ,  
  
count(case when ee.action = 'sent_reengagement_email' then 1 else null end) as  
num_sent_reengagement_email  
  
FROM users u  
  
left join  
  
email_events ee  
  
on  
  
u.user_id = ee.user_id  
  
group by 1 ;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	user_id	num_sent_weekly_digest	num_email_open	num_email_clickthrough	num_sent_reengagement_email
▶	0	17	5	0	0
	4	17	5	4	0
	8	17	3	1	0
	11	17	5	2	0
	17	17	4	1	0
	19	17	5	1	0
	20	17	8	3	0
	22	17	7	3	0

Result 67 x 0

The total number of rows 19066

```
select action , count(user_id) as num_times_used
from email_events
group by 1 ;
```

Result Grid   Filter Rows: Exp		
	action	num_times_used
▶	sent_weekly_digest	57267
	email_open	20459
	email_clickthrough	9010
	sent_reengagement_email	3653