Operation Analytics and Investigating Metric Spike

created by: Aravind D R

Case Study 1 (Job Data)

Project Description:

- This project is about solving some of the tasks given by the trainity team from the given dataset.
- I have created a database using the dataset. Then I have performed some SQL calculations based on the given tasks.

Things to be found out:

- Number of jobs reviewed
- 7days rolling average of throughput
- Percentage share of each language
- Duplicate rows

Approach & Insights

Job Data

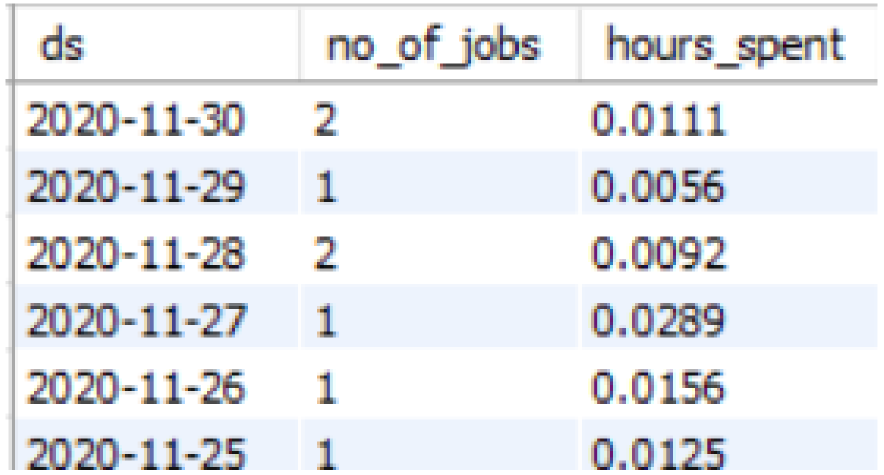
A. Number of jobs reviewed

- To find out the number of jobs reviewed over time run the below code.

Code:

select ds,count(job_id) as no_of_jobs, sum(time_spent)/3600 as hours_spent from job_datawhere ds between '2020-11-01' and '2020-11-30'group by ds;

Result:



ds	no_of_jobs	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

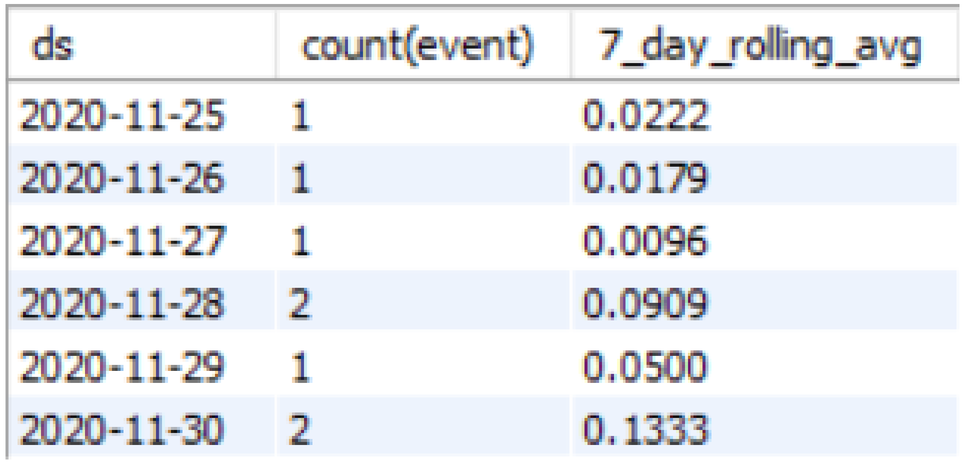
B.Throughput

- To find the throughput I must calculate the number of events happening per second.
- Then with the data I must find the 7days rolling average of throughput.

Code:

```
select ds,count(event),count(event)/sum(time_spent)
                over(partition by ds order by ds, event Rows between 6 preceding and current row ) as 7_day_rolling_avg
                from job_data group by ds;
```

Result:



ds	count(event)	7_day_rolling_avg
2020-11-25	1	0.0222
2020-11-26	1	0.0179
2020-11-27	1	0.0096
2020-11-28	2	0.0909
2020-11-29	1	0.0500
2020-11-30	2	0.1333

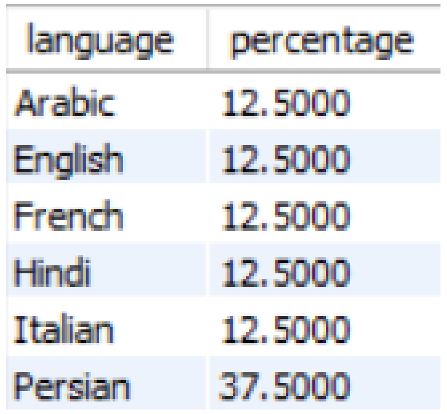
C.Percentage share of each language

- My task is to find the percentage share of each language in last 30 days.
- So here I have used cte method to find the result.

Code:

```
with tempdata as (
select language, row_number() over(partition by language order by language) as rownum
from job_data )
select language, max(rownum)*100/sum(count(language)) over () as percentage from tempdata
                                              group by language;
```

Result:



language	percentage
Arabic	12.5000
English	12.5000
French	12.5000
Hindi	12.5000
Italian	12.5000
Persian	37.5000

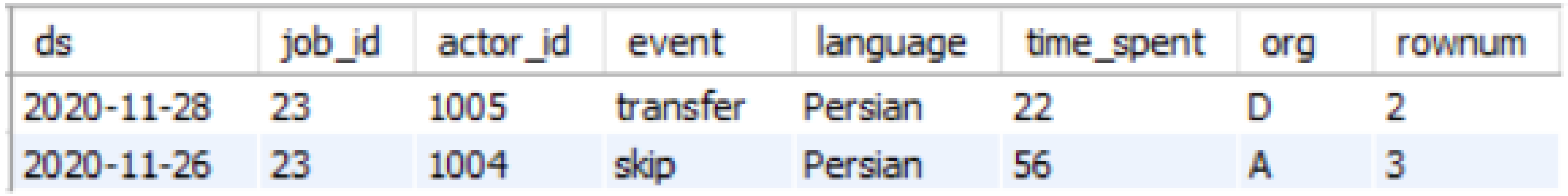
D. Duplicate Rows

- My task is to display the duplicate rows from the table.

Code:

with tempdata as(select *, row_number() over (partition by job_id order by job_id) as rownumfrom job_data)select * from tempdatawhere rownum > 1;

Result:



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

Approach & Insights

Investigating Metric Spike

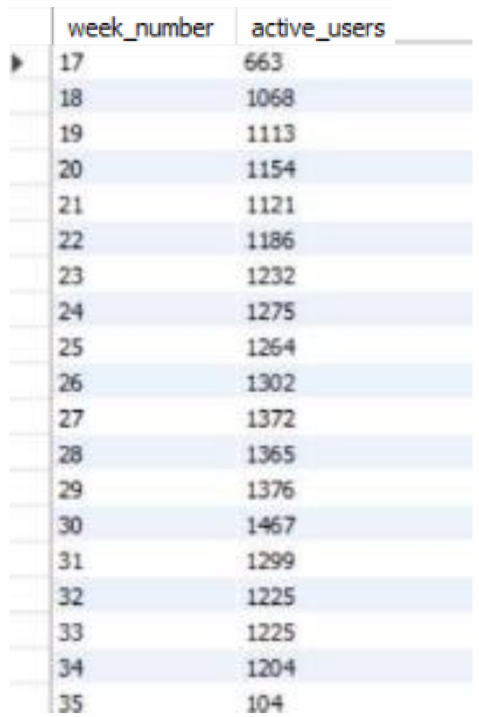
A. User Engagement

- According to the task I must calculate the weekly user engagement.
- For that I must find the engagement list in the events table.

Code:

```
  select extract(week FROM occurred_at) as week_number, count(distinct user_id) as active_users from events where event_type =
'engagement' group by 1;
```

Result:



week_number	active_users
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

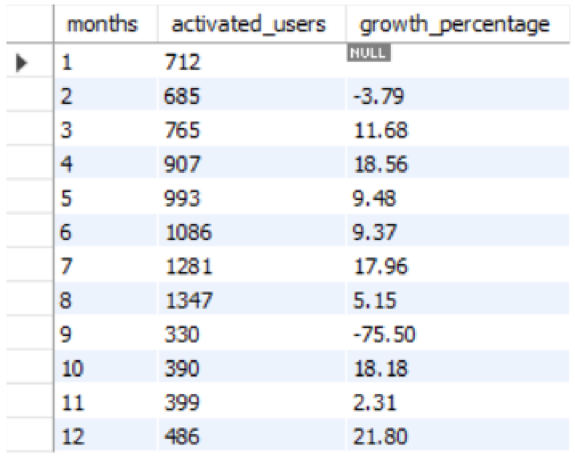
B. User Growth

- To calculate the amount of users growing overtime for a product.

Code:

```
with temp as (
select extract(month from created_at) as months, count(activated_at) as activated_users
from users
where activated_at not in("")
group by months
order by months )
select months, activated_users, round(((activated_users/lag(activated_users,1) over(order by months) - 1)*100),2) as growth_percentage from temp;
```

Result:



months	activated_users	growth_percentage
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

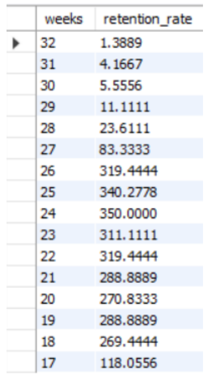
C. Weekly Retention

- To calculate the weekly retention of users sign-up cohort.

Code:

```
with m as(
 select *, count(*) as cohort, extract(week from occurred_at) as week1
from events where event_type = 'signup_flow' group by week1 ), n as (
select count(distinct user_id) as active_users, extract(week from occurred_at) as weeks
from events where event_type = "engagement" group by weeks )
 select weeks, active_users/cohort * 100 as retention_rate  from n,m group by 1;
```

Result:



weeks	retention_rate
32	1.3889
31	4.1667
30	5.5556
29	11.1111
28	23.6111
27	83.3333
26	319.4444
25	340.2778
24	350.0000
23	311.1111
22	319.4444
21	288.8889
20	270.8333
19	288.8889
18	269.4444
17	118.0556

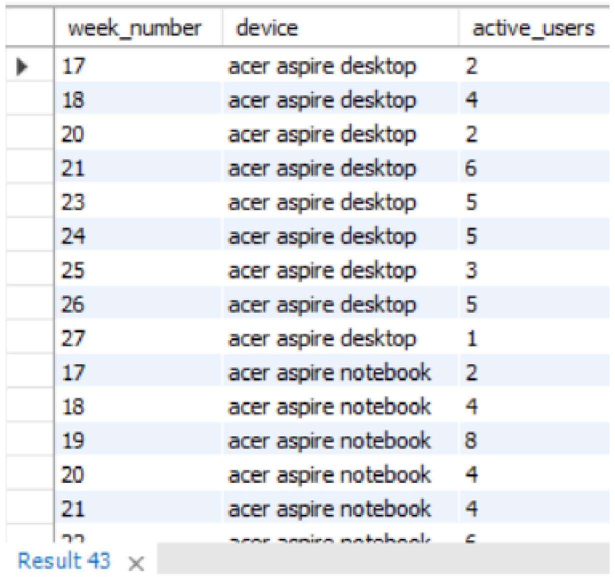
D. Weekly Engagement

- To measure the activeness of a user per device.

Code:

```
select extract(week FROM occurred_at) as week_number, device, count(distinct user_id) as active_users
from events where event_type = 'engagement' group by device,week_number;
```

Result:



week_number	device	active_users
17	acer aspire desktop	2
18	acer aspire desktop	4
20	acer aspire desktop	2
21	acer aspire desktop	6
23	acer aspire desktop	5
24	acer aspire desktop	5
25	acer aspire desktop	3
26	acer aspire desktop	5
27	acer aspire desktop	1
17	acer aspire notebook	2
18	acer aspire notebook	4
19	acer aspire notebook	8
20	acer aspire notebook	4
21	acer aspire notebook	4
22	acer aspire notebook	6

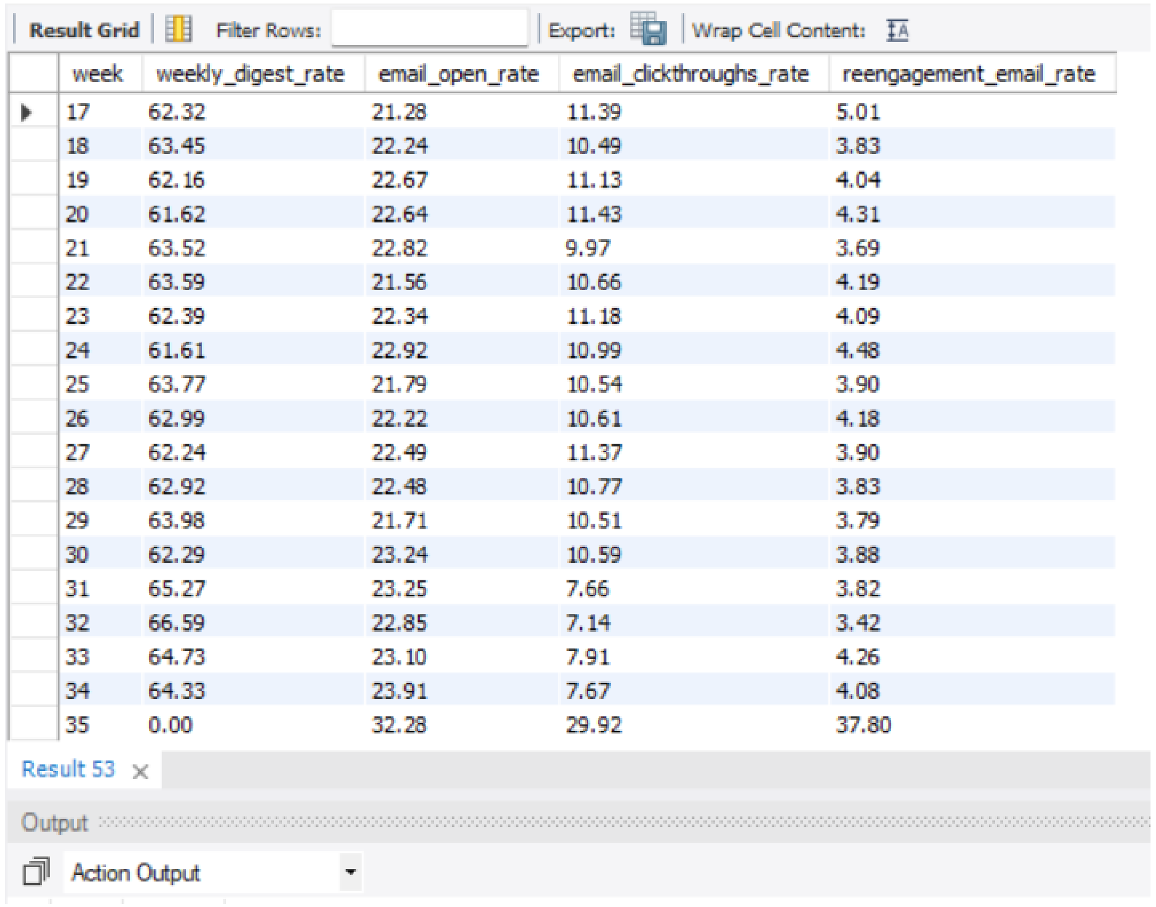
E. E-mail Engagement

- To calculate the email engagement metrics.

Code:

```
with temp as(  select extract(week from occurred_at) as week,
count(case when action='sent_weekly_digest' then user_id else null end) as weekly_digest,
count(case when action='email_open' then user_id else null end) as email_opens,
count(case when action='email_clickthrough' then user_id else null end) as email_clickthroughs,
count(case when action='sent_reengagement_email' then user_id else null end) as reengagement_emails,
count(user_id) as total from email_events group by week )
select week,
round((weekly_digest/total * 100),2) as weekly_digest_rate,
round((email_opens/total * 100),2) as email_open_rate,
round((email_clickthroughs/total * 100),2) as email_clickthroughs_rate,
round((reengagement_emails/total * 100),2) as reengagement_email_rate
from temp
group by week
 order by week;
```

Result



week	weekly_digest_rate	email_open_rate	email_clickthroughs_rate	reengagement_email_rate
17	62.32	21.28	11.39	5.01
18	63.45	22.24	10.49	3.83
19	62.16	22.67	11.13	4.04
20	61.62	22.64	11.43	4.31
21	63.52	22.82	9.97	3.69
22	63.59	21.56	10.66	4.19
23	62.39	22.34	11.18	4.09
24	61.61	22.92	10.99	4.48
25	63.77	21.79	10.54	3.90
26	62.99	22.22	10.61	4.18
27	62.24	22.49	11.37	3.90
28	62.92	22.48	10.77	3.83
29	63.98	21.71	10.51	3.79
30	62.29	23.24	10.59	3.88
31	65.27	23.25	7.66	3.82
32	66.59	22.85	7.14	3.42
33	64.73	23.10	7.91	4.26
34	64.33	23.91	7.67	4.08
35	0.00	32.28	29.92	37.80

Tech Stack Used

- MySQL Workbench 8.0 server
- Powerpoint for presentation

Result

- In this project I have learnt more about common table expression.
- I could find most of the contents in trainity course, so I learnt by myself through Googling, YouTube, etc
- Most importantly I have learnt about when, then conditions with case expressions, It can be widely used for better searching and filtering of queries.