Operation Analytics and Investigating Metric Spike

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

Analysis done with these points:

Case Study 1 (Job Data):

Below is the structure of the table with the definition of each column that we must work on:

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

Then using this table answering the questions that follows:

- Number of jobs reviewed: Amount of jobs reviewed over time.

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

- Throughput: It is the no. of events happening per second.

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?

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

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

- Duplicate rows: Rows that have the same value present in them.

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

<u>Case Study 2 (Investigating metric spike):</u>

The structure of the table with the definition of each column that we must work on:

<u>Table-1: users</u>

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

<u>Table-2: events</u>

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.

<u>Table-3: email_events</u>

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

Using this dataset answering the questions that follows:

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

Task: Calculate the weekly user engagement?

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

Task: Calculate the user growth for product?

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

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

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

Task: Calculate the weekly engagement per device?

- Email Engagement: Users engaging with the email service.

Task: Calculate the email engagement metrics?

Software Used: Google cloud bigquery 1.42.0

Case Study 1 (Job Data):

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

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

To find the number of jobs reviewed per hour per day for November 2020:

1. We will use the data from job_id columns of the job_data table.

2. Then we will divide the total count of job_id (distinct and non-distinct) by (30 days * 24 hours) for finding the number of jobs reviewed per day.

Query (non-distinct job_id):

```
select

count(job_id)/(30*24) as no_of_job_reviewed_per_day_per_hour

from `practice-324303.trainity.job_data`
```

Result:

no_of_job_reviewed_per_day_per_hour

0.011111111111111112

Query (distinct job_id):

```
select
count(distinct job_id)/(30*24) as no_of_job_reviewed_per_day_per_hour
from `practice-324303.trainity.job_data`
```

Result :

no_of_job_reviewed_per_day_per_hour

0.0083333333333333332

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

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?

For calculating the throughput we will be using the 7-day rolling because 7-day rolling gives us the average for all the days right from day 1 to day 7. Whereas daily metric gives us average for only that particular day itself.

For calculating the 7-day rolling daily metric average of throughput:-

1. We will be first taking the count of job_id(distinct and non-distinct) and ordering them w.r.t ds (date of interview)

2. Then by using the ROW function we will be considering the rows between preceding rows and the current row

3. Then we will be taking the average of the jobs_reviewed

Query (distinct job_id):

```
SELECT ds as date_of_review, jobs_reviewed, AVG(jobs_reviewed)
OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS
throughput_7_rolling_average
FROM
(
SELECT ds, COUNT( DISTINCT job_id) AS jobs_reviewed
FROM `practice-324303.trainity.job_data`
GROUP BY ds ORDER BY ds
) a;
```

Result:

date_of_review	jobs_reviewed	throughput_7_rolling_average
2020-11-26	1	0.66666666666666663
2020-11-25	1	0.5
2020-11-27	1	0.75
2020-11-28	2	1.0
2020-11-29	1	1.0
2020-11-30	2	1.142857142857142 8

Query (non-distinct job_id):

```
SELECT ds as date_of_review, jobs_reviewed, AVG(jobs_reviewed)
OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS
throughput_7_rolling_average
FROM
(
SELECT ds, COUNT(job_id) AS jobs_reviewed
FROM `practice-324303.trainity.job_data`
GROUP BY ds ORDER BY ds
) a;
```

Result:

date_of_review	jobs_reviewed	throughput_7_rolling_average
2020-11-25	1	1.0
2020-11-26	1	1.0
2020-11-27	1	1.0
2020-11-28	2	1.25
2020-11-29	1	1.2
2020-11-30	2	1.3333333333333333

Percentage share of each language: Share of each language for different contents. Calculate the percentage share of each language in the last 30 days?

To calculate the percentage share of each language (distinct and non_distinct):-

1. We will first divide the total number of languages (distinct/non-distinct) by the total number of rows presents in the table.

2. Then we will do the grouping based on the languages.

Query:

```
select language,
count(language) as total_of_each_language,
(count(language)/(select count(job_id) from `practice-
324303.trainity.Job_data`)*100) as
percentage_share_of_each_language
from `practice-324303.trainity.Job_data`
group by language;
```

Result:

language	total_of_each_language	percentage_share_of_each_language
English	1	12.5
Persian	3	37.5
Hindi	1	12.5
French	1	12.5
Arabic	1	12.5
Italian	1	12.5

Duplicate rows: Rows that have the same value present in them.

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

To view the duplicate rows having the same value:-

1. First decide primary key column of the table to find the duplicate row values

2. After deciding the column we will use the ROW_NUMBER function to find the row numbers having the same value

3. Then we will partioning the ROW_NUMBER function over the column that we decided i.e. job_id

4. Then using the WHERE function we will find the row_num having value greater than 1 i.e. row_num > 1 based on the occurrence of the job_id in the table

Query:

```
select *
From (
  SELECT *,
  row_number()over(partition by job_id ) as row_num
  FROM `practice-324303.trainity.Job_data`) a
where row_num > 1;
```

Result:

ds	job_id	actor_id	event	language	time_spent	org	row_num
2020-11-29	23	1003	decision	Persian	20	C	2
2020-11-28	23	1005	transfer	Persian	22	D	3

Case Study 2 (Investigating metric spike):

User Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service. Calculate the weekly user engagement?

To find the weekly user engagement:-

1. First, We will extract the week from the occurred_at column of the events table using the EXTRACT function.

2. Then we will be counting the number of distinct user_id from the events table

3. Then we will use the GROUP BY function to group the output w.r.t week from occurred_at and lastly doing ordering of the table w.r.t week_num.

Query:

```
select
extract(week from occurred_at) as week_num,
count(distinct user_id) as num_of_users
from `practice-324303.trainity.events`
group by week_num
order by week_num;
```

Result:

week_num	num_of_users
17	740
18	1260
19	1287
20	1351
21	1299
22	1381
23	1446

24	1471
25	1459
26	1509
27	1573
28	1577
29	1607
30	1706
31	1514
32	1454
33	1438
34	1443
35	118

User Growth: Amount of users growing over time for a product. Calculate the user growth for product?

To find the user growth (number of active users per week):-

1. First we will the extract the year and week for the occurred_at column of the users table using the extract, year and week functions

2. Then we will group the extracted week and year on the basis of year and week number

3. Then we ordered the result on the basis of year and week number

4. Then we will find the cumm_active_users using the SUM, OVER and ROW function between unbounded preceding and current row

Query:

```
select
year_num,
week_num,
num_active_users,
SUM(num_active_users)OVER(ORDER BY year_num, week_num ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_users
from
```

```
(
select
extract (year from a.activated_at) as year_num,
extract (week from a.activated_at) as week_num,
count(distinct user_id) as num_active_users
from
`practice-324303.trainity.users` a
WHERE
state = 'active'
group by year_num,week_num

) a
order by year_num,week_num
```

Result:

year_num	week_num	num_active_users	cum_active_users
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724
2013	19	57	781
2013	20	39	820
2013	21	49	869
2013	22	54	923
2013	23	50	973

2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813
2013	37	85	1898
2013	38	90	1988
2013	39	84	2072
2013	40	87	2159
2013	41	73	2232
2013	42	99	2331
2013	43	89	2420
2013	44	96	2516
2013	45	91	2607
2013	46	88	2695
2013	47	102	2797
2013	48	97	2894
2013	49	116	3010
2013	50	124	3134
2013	51	102	3236
2013	52	47	3283
2014	0	83	3366
2014	1	126	3492
2014	2	109	3601
2014	3	113	3714
2014	4	130	3844
2014	5	133	3977

2014	6	135	4112
2014	7	125	4237
2014	8	129	4366
2014	9	133	4499
2014	10	154	4653
2014	11	130	4783
2014	12	148	4931
2014	13	167	5098
2014	14	162	5260
2014	15	164	5424
2014	16	179	5603
2014	17	170	5773
2014	18	163	5936
2014	19	185	6121
2014	20	176	6297
2014	21	183	6480
2014	22	196	6676
2014	23	196	6872
2014	24	229	7101
2014	25	207	7308
2014	26	201	7509
2014	27	222	7731
2014	28	215	7946
2014	29	221	8167
2014	30	238	8405
2014	31	193	8598
2014	32	245	8843
2014	33	261	9104
2014	34	259	9363
2014	35	18	9381

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

Calculate the weekly retention of users-sign up cohort?

The weekly retention of users-sign up cohort can be calculated by

1. Firstly we will extract the week from occurred_at column using the extract,week functions

2. Then, we will select out those rows in which event_type = 'signup_flow’ and

event_name = 'complete_signup’

3. Then using the left join we will join the two tables on the basis of user_id where event_type = 'engagement’

5. Then we will use the Group By function to group the output table on the basis of user_id

6. Then we will use the Order By function to order the result table on the basis of user_id

Query:

```
SELECT
distinct user_id,
COUNT(user_id),
SUM(CASE WHEN retention_week = 1 Then 1 Else 0 END) as per_week_retention
FROM
(
SELECT
a.user_id,
a.signup_week,
b.engagement_week,
b.engagement_week - a.signup_week as retention_week
FROM
(
(SELECT distinct user_id, extract(week from occurred_at) as signup_week from
 `practice-324303.trainity.events`
WHERE event_type = 'signup_flow'
and event_name = 'complete_signup'
)a
LEFT JOIN
(SELECT distinct user_id, extract (week from occurred_at) as engagement_week
 FROM trainity
)b
on a.user_id = b.user_id
)
)d
group by user_id
order by user_id;
```

Result :

Link for the result uploaded in drive:
https://drive.google.com/file/d/1NwM2oEDoZ-4IXeVT5f9xwA1BFvgClfVB/view?usp=sharing

Weekly Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly. Calculate the weekly engagement per device?

To find the weekly user engagement per device:-

1. Firstly we will extract the year_num and week_num from the occurred_at column of the events table using the extract, year and week function

2. Then we will select those rows where event_type = 'engagement' using the WHERE clause

3. Then by using the Group By and Order By function we will group and order the result on the basis of year_num, week_num and 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 no_of_users
FROM
`practice-324303.trainity.events`
where event_type = 'engagement'
GROUP by 1,2,3
order by 1,2,3;
```

Result:

Link for the result uploaded in drive:

https://drive.google.com/file/d/1m9S8lVE_U_DyhDKyKeV1eRT3h-wg-FYl/view?usp=sharing

Email Engagement: Users engaging with the email service. Calculate the email engagement metrics?

To find the email engagement metrics(rate) of users:-

1. We will first categorize the action on the basis of email_sent, email_opened and email_clicked using the CASE, WHEN, THEN functions

2. Then we select the sum of category of email_opened divide by the sum of the category of email_sent and multiply the result by 100.0 and name is as email_opening_rate

3. Then we select the sum of category of email_clicked divide by the sum of the category of email_sent and multiply the result by 100.0 and name is as email_clicking_rate

4. email_sent = ('sent_weekly_digest','sent_reengagement_email’)

5. email_opened = 'email_open’

6. email_clicked = 'email_clickthrough'

Query:

```
SELECT
SUM(CASE when email_cat = 'email_opened' then 1 else 0 end)/SUM(CASE when
email_cat = 'email_sent' then 1 else 0 end)*100 as email_opening_rate,
SUM(CASE when email_cat = 'email_clicked' then 1 else 0 end)/SUM(CASE when
email_cat = 'email_sent' then 1 else 0 end)*100 as email_clicking_rate
FROM
(
SELECT
*,
CASE
WHEN action in ('sent_weekly_digest','sent_reengagement_email')
then 'email_sent'
WHEN action in ('email_open')
then 'email_opened'
WHEN action in ('email_clickthrough')
then 'email_clicked'
end as email_cat
from `practice-324303.trainity.email_events`
) a
```

Result:

email_opening_rate	email_clicking_rate
33.583388049901508	14.789888378200919