Task 3 : Operation Analytics and Investigating Metric Spike

Analysis done on the following points:-

Case Study 1 : Job Data

A. **Number of jobs reviewed:** Calculate the number of jobs reviewed per hour per day for November 2020?
B. **Throughput:** Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?
C. **Percentage share of each language:** Calculate the percentage share of each language in the last 30 days?
D. **Duplicate rows:** duplicate rows in the data. How will you display duplicates from the table?

Software used : MySQL Workbench 8.0 CE

Case Study 2: Investigating metric spike

A. **User Engagement:** Calculate the weekly user engagement?

B. **User Growth:** Calculate the user growth for product?

C. **Weekly Retention:** Calculate the weekly retention of users-sign up cohort?

D. **Weekly Engagement:** Calculate the weekly engagement per device?

E. **Email Engagement:** Calculate the email engagement metrics?

Job Data

- **Number of jobs reviewed:**Calculate the number of jobs reviewed per hour per day for November 2020?

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

1. use the data from **job_id** columns of the job_data table.
2. 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.

```
#  calculate the number of jobs reviewed per hour for each day in November 2020.

select count(job_id)/(30*24) as  number_of_jobs_reviewed_per_hour_per_day_non_distinct

from job_data ;
```

Output

	number_of_jobs_reviewed_per_hour_per_day_no
▶	0.0111

Job Data

- **<u>Throughput</u>:Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?**

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.

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

1. first taking the count of job_id(distinct and non-distinct) and ordering them w.r.t ds (date of interview).
2. using the ROW function we will be considering the rows between 6 preceding rows and the current row
3. be taking the average of the jobs_reviewed.

Job Data

```
# calculate the 7-day rolling average of throughput.
#Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.
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_non_distinct_job_id
FROM
(
SELECT ds, COUNT(job_id) AS jobs_reviewed
FROM job_data
GROUP BY ds ORDER BY ds
) a;
```

Output

date_of_review	jobs_reviewed	throughput_7_rolling_average_non_distinct_job_ic
11/25/2020	1	1.0000
11/26/2020	1	1.0000
11/27/2020	1	1.0000
11/28/2020	2	1.2500
11/29/2020	1	1.2000
11/30/2020	2	1.3333

Job Data

- **<u>Percentage share of each language:</u>** Calculate the percentage share of each language?

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

1. first divide the total number of languages (distinct/non-distinct) by the total number of rows presents in the table
2. do the grouping based on the languages.

```
# calculate the percentage share of each language over the last 30 days.
select job_data.job_id, job_data.language,count(distinct job_data.language) as total_of_each_language,
((count(job_data.language)/(select count(*) from job_data))*100) as percentage_share_of_each_distinct_language
from job_data
group by job_data.language;
```

<u>Output</u>

date_of_review	jobs_reviewed	throughput_7_rolling_average_non_distinct_job_j
11/25/2020	1	1.0000
11/26/2020	1	1.0000
11/27/2020	1	1.0000
11/28/2020	2	1.2500
11/29/2020	1	1.2000
11/30/2020	2	1.3333

Job Data

- **Duplicate rows :** duplicate rows in the data. How will you display duplicates from the table?

duplicate rows having the same value we will:-

1. need to find the duplicate row values
2. use the ROW_NUMBER function to find the row numbers having the same value
3. portioning the ROW_NUMBER function over the column (parameter) that we decided i.e. job_id
4. 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

```
# display duplicate rows from the job_data table.
SELECT * FROM
(SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num
FROM job_data) a
WHERE row_num>1;
```

Job Data

Output

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

Investigating Metric Spike

- **User Engagement :** Calculate the weekly user engagement?

find the weekly user engagement:-

1. extract the week from the occurred_at column of the events table using the **EXTRACT** function and **WEEK** function.
2. counting the number of distinct user_id from the events table
3. use the **GROUP BY** function to group the output w.r.t **week from occurred_at.**

```
SELECT
extract (week from occurred_at) as week_number, count(distinct user_id) as number_of_users
FROM
tutorial.yammer_events
group by
week_number;
```

Investigating Metric Spike

Output

week_number	number_of_users
18	791
19	1244
20	1270
21	1341
22	1293
23	1366
24	1434
25	1462
26	1443
27	1477
28	1556
29	1556
30	1593
31	1685
32	1483
33	1438
34	1412
35	1442

Investigating Metric Spike

- **User Growth:** Calculate the user growth for product?
- **User Growth = Number of active users per week**

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

1. extract the year and week for the **occurred_at** column of the **users** table using the **extract, year and week** functions.
2. group the extracted week and year on the basis of year and week Number
3. ordered the result on the basis of year and week number
4. find the cum_active_users using the **SUM, OVER** and **ROW** function **between unbounded preceding and current row**

Investigating Metric Spike

```
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 tutorial.yammer_users a WHERE
state = 'active'
group by year_num,week_num
order by year_num,week_num
) a;
```

Output

year_num	week_num	num_active_users	cum_active_users	year_num	week_num	num_active_users	cum_active_users
2013	1	67	67	2013	45	97	2564
2013	2	29	96	2013	46	94	2658
2013	3	47	143	2013	47	82	2740
2013	4	36	179	2013	48	103	2843
2013	5	30	209	2013	49	96	2939
2013	6	48	257	2013	50	117	3056
2013	7	41	298	2013	51	123	3179
2013	8	39	337	2013	52	104	3283
2013	9	33	370	2014	1	91	3374
2013	10	43	413	2014	2	122	3496
2013	11	33	446	2014	3	112	3608
2013	12	32	478	2014	4	113	3721
2013	13	33	511	2014	5	130	3851
2013	14	40	551	2014	6	132	3983
2013	15	35	586	2014	7	135	4118
2013	16	42	628	2014	8	127	4245
2013	17	48	676	2014	9	127	4372
2013	18	48	724	2014	10	135	4507
2013	19	45	769	2014	11	152	4659
2013	20	55	824	2014	12	132	4791
2013	21	41	865	2014	13	151	4942
2013	22	49	914	2014	14	161	5103
2013	23	51	965	2014	15	166	5269
2013	24	51	1016	2014	16	165	5434
2013	25	46	1062	2014	17	176	5610
2013	26	57	1119	2014	18	172	5782
2013	27	57	1176	2014	19	160	5942
2013	28	52	1228	2014	20	186	6128
2013	29	71	1299	2014	21	177	6305
2013	30	66	1365	2014	22	186	6491
2013	31	69	1434	2014	23	197	6688
2013	32	66	1500	2014	24	198	6886
2013	33	73	1573	2014	25	222	7108
2013	34	70	1643	2014	26	210	7318
2013	35	80	1723	2014	27	199	7517
2013	36	65	1788	2014	28	223	7740
2013	37	71	1859	2014	29	215	7955
2013	38	84	1943	2014	30	228	8183
2013	39	92	2035	2014	31	234	8417
2013	40	81	2116	2014	32	189	8606
2013	41	88	2204	2014	33	250	8856
2013	42	74	2278	2014	34	259	9115
2013	43	97	2375	2014	35	266	9381
2013	44	92	2467				

Investigating Metric Spike

```
select count(*) from tutorial.yammer_users where state = 'active';
```

Output

count
9381

Hence, there are in total 9381 active users from 1st week of 2013 to the 35th week of 2014

Investigating Metric Spike

- **<u>Weekly Retention</u>: Calculate the weekly retention of users-sign up cohort?**

Program/Query(Without Specifying the week number) :

weekly retention of users-sign up cohort can be calculated by two means i.e. either by specifying the week number (18 to 35) or for the entire column of occurred_at of the events table.

1. extract the week from occurred_at column using the **extract,week** functions.
2. select out those rows in which **event_type ='signup_flow' and event_name = 'complete_signup'.**
3. spectify the week number using the **extract** function.
4. using the **left join** we will join the two tables on the basis of user_id where **event_type ='engagement'**
5. use the **Group By** function to group the output table on the basis of user_id
6. use the **Order By** function to order the result table on the basis of user_id

Investigating Metric Spike

```
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 tutorial.yammer_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 tutorial.yammer_events
where event_type = 'engagement'
)b
on a.user_id = b.user_id
)
)d
group by user_id order by user_id;
```

Investigating Metric Spike

Output (Without Specifying week number)

Trainity_task_3_case_stuy_2_question_c.csv - Google_Drive

Investigating Metric Spike

Program/Query(Specifying the week number as 18) :

```
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 tutorial.yammer_events
WHERE event_type = 'signup_flow' and event_name = 'complete_signup' and extract(week from occurred_at) = 18
)a
LEFT JOIN
(SELECT distinct user_id, extract (week from occurred_at) as engagement_week FROM tutorial.yammer_events where event_type = 'engagement'
)b
on a.user_id = b.user_id
)
)d
group by user_id order by user_id;
```

Investigating Metric Spike

Output (Specifying week number as 18)

Trainity_task_3_case_stuy_2_question_c_18_week.csv – google drive

Investigating Metric Spike

- **Weekly Engagement:** Calculate the weekly engagement per device?

find the weekly user engagement per device:-

1. extract the year_num and week_num from the occurred_at column of the events table using the **extract, year** and **week** function
2. select those rows where **event_type ='engagement'** using the **WHERE** Clause.
3. 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

Investigating Metric Spike

```
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
tutorial.yammer_events
where event_type = 'engagement'
GROUP by 1,2,3
order by 1,2,3;
```

Output

question D weekly user engagement per device.csv - Google Drive

Investigating Metric Spike

- **<u>Email Engagement:</u> Calculate the email engagement metrics**

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

1. categorize the action on the basis of **email_sent, email_opened** and **email_clicked** using the **CASE, WHEN, THEN** functions
2. 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. 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'**

Investigating Metric Spike

```
SELECT
100.0*SUM(CASE when email_cat = 'email_opened' then 1 else 0 end)/SUM(CASE when email_cat = 'email_sent' then 1 else 0 end)
 as email_opening_rate,
100.0*SUM(CASE when email_cat = 'email_clicked' then 1 else 0 end)/SUM(CASE when email_cat = 'email_sent' then 1 else 0 end)
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 tutorial.yammer_emails
) a;
```

Output

Question E email_engagement_metrics.csv - Google Drive