Operation Analytics and Investigation Metric Spike

ABDUL HALIK H

Project Introduction:

- I was assigned as a Lead Data Analyst at Microsoft for this project.
- I'll be working on Operational Analytics which is a crucial process that involves analyzing a company's end-to-end operations.
- This This analysis helps identify areas for improvement within the company.
- Then I'll be working on Investigating Metric Spikes that involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales.

Tasks to be performed:

A. Job Data Analysis:

1. Job Reviewed Overtime
2. Throughput Analysis
3. Language Share Analysis
4. Duplicate Rows Detection

B. Investigation Metric Spike:

1. Weekly User Engagement
2. User Growth Analysis
3. Weekly Retention Analysis
4. Weekly Engagement / Device
5. Email Engagement Analysis

Tools Used :

- Microsoft PowerPoint
- MySQL Workbench
- Mode.com(Online tool)

Approach:

- To perform our analysis process we need to create a database to work on running SQL commands
- We can use the MySQL Workbench 8.0.32 Community Software and Mode website to create our database.
- The created SQL database will have multiple tables which contains multiple columns in which the data is stored.
- For this process we'll use advanced commands like AGGREGATE, CTE, Nested Queries, CROSS JOIN, WINDOWS Functions, etc .
- With all those done neatly, I'll help you with assistance in understanding how the attained results will be relevant to the queries by which you can find the required insights.

Case Study 1 : Job Data

- Using the table of attributes(**job_id,actor_id,event,language, time_spent,org,ds**) I need to work about the job data on the following queries:
- Tables (Attributes) used:
  1. **job_id**
  2. **actor_id**
  3. **event**
  4. **language**
  5. **time_spent**
  6. **org,ds**

Tasks:

- **No of jobs reviewed**
- **Throughput**
- **Percentage share of each language**
- **Duplicate rows**

No of jobs reviewed :

- I have used the **Time and Date** Function to sort out the **no of jobs** being **reviewed/hour/day for November 2020.**
- I find an average of **126 jobs** being reviewed per hour per day from **Nov 25th 2020 to Nov 30th 2020.**
- You can check the queries and output of them for any doubts.

QUERY :

```
7   #NO OF JOBS REVIEWED/HOUR/DAY FOR NOV 2020
8
9   SELECT
10  ds,
11  SUM(time_spent) AS total_time,
12  ROUND(1.0*COUNT(job_id)*3600/SUM(time_spent),2) AS total_jobs_reviewed
13  FROM
14  job
15  WHERE
16  ds BETWEEN '2020-11-01' AND '2020-11-30'
17  GROUP BY
18  ds
19  ORDER BY
20  ds;
```

OUTPUT :

Result Grid | Filter Rows:

ds	total_time	total_jobs_reviewed
2020-11-25	45	80.00
2020-11-26	56	64.29
2020-11-27	104	34.62
2020-11-28	33	218.18
2020-11-29	20	180.00
2020-11-30	40	180.00

Throughput :

- I have used a virtual table **CTE** to execute some **Nested Queries** to find the **throughput**(No. of events happening per second).
- Here I have calculated a rolling average of throughput for **7 days** as it is easier to monitor for a week than for each day in a month.
- Check the provided query and output for clarifications.

QUERY :

```
#7 DAY ROLLING AVG THROUGHPUT

WITH CTE AS (
SELECT
ds,
COUNT(job_id) AS total_jobs,
SUM(time_spent) AS total_time
FROM
job
WHERE
ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY
ds
)
SELECT
ds,
total_jobs,
total_time,
ROUND(SUM(total_jobs) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW ) /
SUM(total_time) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW),4) AS '7day_throughput'
FROM
CTE;
```

OUTPUT :

Result Grid | Filter Rows: | Export:

ds	total_jobs	total_time	7day_throughput
2020-11-25	1	45	0.0222
2020-11-26	1	56	0.0198
2020-11-27	1	104	0.0146
2020-11-28	2	33	0.0210
2020-11-29	1	20	0.0233
2020-11-30	2	40	0.0268

Percentage share of each language :

- The same **CTE** virtual table is used here also to find the percentage share of each language for **30 days** .
- I also applied **CROSS JOIN** function to work on the nested and outer table queries.
- I find that **Persian** has the **largest share** of **37.50** of all the other languages in the provided data set.
- I also included the queries and output to verify my insight.

QUERY :

```
#PERCENTAGE SHARE OF EACH LANGUAGE IN LAST 30 DAYS

WITH CTE AS (
SELECT
language,COUNT(job_id) AS total_jobs
FROM job
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY language
ORDER BY language
),
total AS (
SELECT
COUNT(job_id) AS no_of_jobs
FROM job
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
)
SELECT *,ROUND((total_jobs/no_of_jobs)*100,2) AS percent_share_of_jobs
FROM CTE CROSS JOIN total
ORDER BY percent_share_of_jobs;
```

OUTPUT :

Result Grid | Filter Rows: | Export:

language	total_jobs	no_of_jobs	percent_share_of_jobs
Arabic	1	8	12.50
English	1	8	12.50
French	1	8	12.50
Hindi	1	8	12.50
Italian	1	8	12.50
Persian	3	8	37.50

Duplicate Rows :

- I here used the **ROW_NUMBER()**of the **WINDOW Function** to find the no of occurrence of every distinct rows.
- I find that there is **no duplicate** rows present in the given dataset.
- Check on the queries and output provided here .

QUERY :

```
#DUPLICATE ROWS IN DATASET

SELECT
*
FROM
(
SELECT
*,
ROW_NUMBER() OVER(PARTITION BY ds,job_id,actor_id) AS no_of_occurences
FROM
job
) alias
WHERE no_of_occurences!=1;
```

OUTPUT :

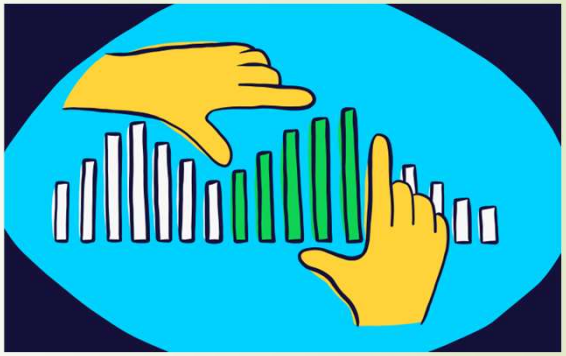
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ds	job_id	actor_id	event	language	time_spent	org	no_of_occurences

Case Study 2 : Investigating Metric Spike

- In 2nd Case Study I was asked to **investigate** about the **users, their actions, action on emails** using the following 3 tables :
  - **users**
  - **events**
  - **email_events**

- I need to answer the following questions based on the following :
  1. **User Engagement**
  2. **User Growth**
  3. **Weekly Retention**
  4. **Weekly Engagement**
  5. **Email Engagement**



User Engagement :

- I must find whether the users find **quality** in **product/service**.
- I also must know about the **activeness of a user**.
- I used **Date and Time WINDOW Function** to find the **user's engagement** and **login** information.
- You can look at the queries and its output to verify.

QUERY :

```
/* WEEKLY USER ENGAGEMENT */

SELECT
DATE_TRUNC('week', e.occurred_at) AS week_day,
COUNT(DISTINCT e.user_id) AS weekly_active_users
FROM tutorial.yammer_events e
WHERE e.event_type = 'engagement' AND e.event_name = 'login'
GROUP BY week_day
ORDER BY week_day
```

OUTPUT :

	week_day	weekly_active_users
1	2014-04-28 00:00:00	701
2	2014-05-05 00:00:00	1054
3	2014-05-12 00:00:00	1094
4	2014-05-19 00:00:00	1147
5	2014-05-26 00:00:00	1113
6	2014-06-02 00:00:00	1173
7	2014-06-09 00:00:00	1219
8	2014-06-16 00:00:00	1262
9	2014-06-23 00:00:00	1249
10	2014-06-30 00:00:00	1271
11	2014-07-07 00:00:00	1355
12	2014-07-14 00:00:00	1345
13	2014-07-21 00:00:00	1363
14	2014-07-28 00:00:00	1442
15	2014-08-04 00:00:00	1266
16	2014-08-11 00:00:00	1215
17	2014-08-18 00:00:00	1203
18	2014-08-25 00:00:00	1194

User Growth :

- I again used the **Date and Time** of the **WINDOW Function** to calculate the no of active users growing over time for a product
- Here I have calculated and made report for the **month January of 2013**
- Check on the queries and output to verify

QUERY :

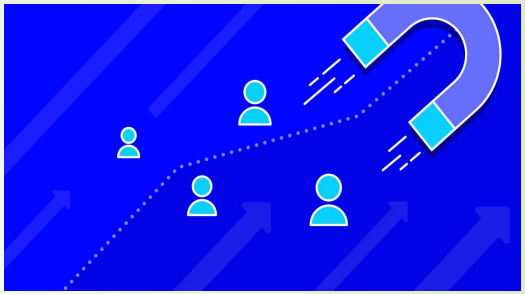
```
/* USER GROWTH FRO PRODUCT OVER TIME*/

SELECT
DATE_TRUNC('day',u.created_at) AS days_of_the_month,
COUNT(u.user_id) AS total_users,
COUNT(CASE
    WHEN u.state IN ('active') THEN u.user_id
    ELSE NULL
    END) AS active_users
FROM
tutorial.yammer_users u
WHERE
created_at BETWEEN '2013-01-01' AND '2013-01-31'
GROUP BY
1
ORDER BY
1;
```

OUTPUT :

	days_of_the_month	total_users	active_users
1	2013-01-01 00:00:00	13	7
2	2013-01-02 00:00:00	11	7
3	2013-01-03 00:00:00	14	6
4	2013-01-04 00:00:00	11	1
5	2013-01-05 00:00:00	3	2
6	2013-01-06 00:00:00	4	3
7	2013-01-07 00:00:00	13	4
8	2013-01-08 00:00:00	13	2
9	2013-01-09 00:00:00	11	6
10	2013-01-10 00:00:00	12	6
11	2013-01-11 00:00:00	11	6
12	2013-01-12 00:00:00	4	3
13	2013-01-13 00:00:00	3	2
14	2013-01-14 00:00:00	13	8
15	2013-01-15 00:00:00	15	11
16	2013-01-16 00:00:00	14	7
17	2013-01-17 00:00:00	13	9
18	2013-01-18 00:00:00	14	10
19	2013-01-19 00:00:00	4	1
20	2013-01-20 00:00:00	4	1
21	2013-01-21 00:00:00	13	7
22	2013-01-22 00:00:00	13	5
23	2013-01-23 00:00:00	14	7
24	2013-01-24 00:00:00	14	5
25	2013-01-25 00:00:00	16	8
26	2013-01-26 00:00:00	3	3
27	2013-01-27 00:00:00	4	1
28	2013-01-28 00:00:00	14	7
29	2013-01-29 00:00:00	14	3
30	2013-01-30 00:00:00	14	6

Weekly Retention :

- I have used the same **Date and Time Function** to find the **no of weeks** that the **users** getting **retained** after signing up for a product.
- Check on the queries and output for clarification.



QUERY :

```
/* WEEKLY RETENTION OF SIGN UP*/

SELECT DATE_TRUNC('week', z.occurred_at) AS week,
AVG(z.age_at_event) AS average_age_during_week,
COUNT(DISTINCT CASE WHEN z.user_age > 70 THEN z.user_id
                    ELSE NULL END) AS Ten_plus_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 70 AND z.user_age >=63 THEN z.user_id
                    ELSE NULL END) AS Nine_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 63 AND z.user_age >=56 THEN z.user_id
                    ELSE NULL END) AS Eight_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 56 AND z.user_age >=49 THEN z.user_id
                    ELSE NULL END) AS Seven_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 49 AND z.user_age >=42 THEN z.user_id
                    ELSE NULL END) AS Six_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 42 AND z.user_age >=35 THEN z.user_id
                    ELSE NULL END) AS Five_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 35 AND z.user_age >=28 THEN z.user_id
                    ELSE NULL END) AS Four_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 28 AND z.user_age >=21 THEN z.user_id
                    ELSE NULL END) AS Three_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 21 AND z.user_age >=14 THEN z.user_id
                    ELSE NULL END) AS Two_weeks,
COUNT(DISTINCT CASE WHEN z.user_age < 14 AND z.user_age >=7 THEN z.user_id
                    ELSE NULL END) AS One_week,
COUNT(DISTINCT CASE WHEN z.user_age < 7 THEN z.user_id
                    ELSE NULL END) AS Less_than_a_week
FROM(
SELECT e.occurred_at, u.user_id,
DATE_TRUNC('week',u.activated_at) AS activation_week,
EXTRACT('day' FROM e.occurred_at - u.activated_at) AS age_at_event,
EXTRACT('day' FROM '201-05-01' :: TIMESTAMP - u.activated_at) AS user_age
FROM tutorial.yammer_users u JOIN tutorial.yammer_events e ON e.user_id = u.user_id
WHERE e.event_type = 'engagement' AND e.event_name = 'login'
AND e.occurred_at >= '2014-05-01' AND e.occurred_at < '2014-09-01'
AND u.activated_at IS NOT NULL
) z
GROUP BY 1
ORDER BY 1
LIMIT 100;
```

OUTPUT :

week	average age during week	ten plus weeks	nine weeks	eight weeks	seven weeks	six weeks	five weeks	four weeks	three weeks	two weeks	one week	less than a week
2014-04-28 00:00:00	124.0072	0	0	0	0	0	0	0	0	0	0	701
2014-05-05 00:00:00	124.3817	0	0	0	0	0	0	0	0	0	0	1054
2014-05-12 00:00:00	131.9386	0	0	0	0	0	0	0	0	0	0	1094
2014-05-19 00:00:00	132.3266	0	0	0	0	0	0	0	0	0	0	1147
2014-05-26 00:00:00	132.3454	0	0	0	0	0	0	0	0	0	0	1113
2014-06-02 00:00:00	131.8311	0	0	0	0	0	0	0	0	0	0	1173
2014-06-09 00:00:00	131.0426	0	0	0	0	0	0	0	0	0	0	1219
2014-06-16 00:00:00	136.4806	0	0	0	0	0	0	0	0	0	0	1262
2014-06-23 00:00:00	136.2789	0	0	0	0	0	0	0	0	0	0	1249
2014-06-30 00:00:00	136.4193	0	0	0	0	0	0	0	0	0	0	1271
2014-07-07 00:00:00	135.8888	0	0	0	0	0	0	0	0	0	0	1355
2014-07-14 00:00:00	143.4488	0	0	0	0	0	0	0	0	0	0	1345
2014-07-21 00:00:00	141.7028	0	0	0	0	0	0	0	0	0	0	1363
2014-07-28 00:00:00	144.0787	0	0	0	0	0	0	0	0	0	0	1442
2014-08-04 00:00:00	140.7322	0	0	0	0	0	0	0	0	0	0	1266
2014-08-11 00:00:00	125.9943	0	0	0	0	0	0	0	0	0	0	1215
2014-08-18 00:00:00	128.0217	0	0	0	0	0	0	0	0	0	0	1203
2014-08-25 00:00:00	128.2698	0	0	0	0	0	0	0	0	0	0	1194

Weekly Engagement :

- I was insisted to report about **activeness of a user** in a product or service weekly .
- I used the same **Date and Time WINDOW Function** with the names of different devices .
- Check the queries and output I have provided with for doubts.

QUERY :

```
/* USER ENGAGEMENT PER DEVICE */
SELECT
DATE_TRUNC('week',e.occurred_at) AS week_day,
COUNT(DISTINCT e.user_id) AS weekly_active_users,
COUNT(DISTINCT CASE
              WHEN e.device IN('macbookpro','lenovo','thinkpad','macbook air','dell inspiron notebook',
              'asus chromebook','dell inspiron desktop','acer notebook','hp pavilion desktop',
              'acer aspire desktop','macb mini') THEN e.user_id
              ELSE NULL
              END) AS computer_users,
COUNT(DISTINCT CASE
              WHEN e.device IN('iphone 5','samsung galaxy s4','nexus 5','iphone 5s','iphone 4s',
              'nokia lumia 635','htc one','samsung galaxy note','amazon fire phone') THEN e.user_id
              ELSE NULL END) AS phone_users,
COUNT(DISTINCT CASE
               WHEN e.device IN('ipad air','nexus 7','ipad mini','nexus 10','kindle fire',
               'windows surface','samsung galaxy tablet') THEN e.user_id
               ELSE NULL
               END) AS tablet_users
FROM tutorial.yammer_events e
WHERE e.event_type = 'engagement' AND e.event_name = 'login'
GROUP BY 1
```

OUTPUT :

	week_day	weekly_active_users	computer_users	phone_users	tablet_users
1	2014-04-28 00:00:00	701	174	281	103
2	2014-05-05 00:00:00	1054	338	461	176
3	2014-05-12 00:00:00	1094	304	481	191
4	2014-05-19 00:00:00	1147	331	526	181
5	2014-05-26 00:00:00	1113	320	500	176
6	2014-06-02 00:00:00	1173	388	505	197
7	2014-06-09 00:00:00	1219	379	545	195
8	2014-06-16 00:00:00	1262	409	541	227
9	2014-06-23 00:00:00	1249	382	526	210
10	2014-06-30 00:00:00	1271	393	578	218
11	2014-07-07 00:00:00	1355	413	591	227
12	2014-07-14 00:00:00	1345	418	578	218
13	2014-07-21 00:00:00	1363	429	601	218
14	2014-07-28 00:00:00	1442	440	588	241
15	2014-08-04 00:00:00	1266	413	491	166
16	2014-08-11 00:00:00	1215	403	438	153
17	2014-08-18 00:00:00	1203	383	428	145
18	2014-08-25 00:00:00	1194	380	441	150

Email Engagement :

- I have used the **Date and Time WINDOW Function** to know user's **engagement metrices with** the **emails.**
- You find the queries and output about my report here.

QUERY :

```
/* EMAIL ENGAGEMENT */

SELECT DATE_TRUNC('week',e.occurred_at) AS week_day,
COUNT(CASE
      WHEN e.action = 'sent_weekly_digest' THEN e.user_id
      ELSE NULL
      END) AS weekly_emails,
COUNT(CASE
      WHEN e.action = 'sent_reengagement_email' THEN e.user_id
      ELSE NULL
      END) AS reengagement_emails,
COUNT(CASE
      WHEN e.action = 'email_open' THEN e.user_id
      ELSE NULL
      END) AS opens_emails,
COUNT(CASE
      WHEN e.action = 'email_clickthrough' THEN e.user_id
      ELSE NULL
      END) AS clickthrough_emails
FROM tutorial.yammer_emails e
GROUP BY 1
ORDER BY 1;
```

OUTPUT :

	week_day	weekly_emails	reengagement_emails	opens_emails	clickthrough_emails
1	2014-04-28 00:00:00	908	98	332	187
2	2014-05-05 00:00:00	2602	164	919	434
3	2014-05-12 00:00:00	2665	175	971	479
4	2014-05-19 00:00:00	2733	179	995	498
5	2014-05-26 00:00:00	2822	179	1026	453
6	2014-06-02 00:00:00	2911	199	993	492
7	2014-06-09 00:00:00	3003	190	1070	533
8	2014-06-16 00:00:00	3105	234	1161	563
9	2014-06-23 00:00:00	3207	187	1090	524
10	2014-06-30 00:00:00	3302	222	1168	559
11	2014-07-07 00:00:00	3399	214	1230	622
12	2014-07-14 00:00:00	3499	226	1260	607
13	2014-07-21 00:00:00	3592	206	1211	584
14	2014-07-28 00:00:00	3706	230	1386	633
15	2014-08-04 00:00:00	3793	206	1336	432
16	2014-08-11 00:00:00	3897	224	1357	430
17	2014-08-18 00:00:00	4012	257	1421	487
18	2014-08-25 00:00:00	4111	263	1533	493


Thank
you