

Operation Analytics and Investigating Metric Spike

(Advanced SQL)

➤ **Project Description:**

This project is about operation analytics and investigating metric spike. Operation analytics is type of business analytics. It focuses on improving existing operations. There are two case studies, first one is job data case study and second is investigating metric spike. With the help of this, the company will find the areas on which it should improve.

I will do analysis using various advanced SQL queries. I am going to find various information like number of jobs reviewed, percentage share of each language, etc.

➤ **Approach:**

First, I will do schema queries. Schema queries are the queries which create database also insert data in database. After creating database, I will do one by one analysis of given question.

I will focus on to evaluating data and getting more accurate information from the raw data.

➤ **Tech-Stack Used:**

MySQL 8.0

➤ **Insights:**

During this project I worked on different commands of SQL. Also, project helps me to gain more knowledge about SQL doing the hands-on. I learned about the date functions also windows function of SQL. I used date function, OVER function, etc. to get results.

➤ **Result:**

1. **Case Study 1 (Job Data):**

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

SQL Command:

```
SELECT
    COUNT(DISTINCT job_id)/(30*24) AS n_job
FROM
    job_data
WHERE
    ds BETWEEN '2020-11-01' AND '2020-11-30';
```

Output:

n_job
0.0111

The screenshot shows the DB Fiddle web application interface. The browser address bar displays 'db-fiddle.com'. The left sidebar contains a 'Fiddle Title' field, a 'Fiddle Description' field, and a 'Private Fiddle' toggle. The main area is divided into three sections: 'Schema SQL' with a table definition for 'job_data', 'Query SQL' with the SQL command, and 'Results' showing the output of the query. The 'Results' section displays a table with one row and one column, 'n_job', with the value '0.0111'. The bottom of the screen shows a Windows taskbar with various application icons and system information.

DB Fiddle - SQL Database Playground

db-fiddle.com

Database: MySQL v8.0

Run Save Load Example Collaborate

Sign in Have any feedback?

Fiddle Title

50 characters remaining

Fiddle Description

300 characters remaining

Private Fiddle PRO

This setting cannot be modified after saving the fiddle.

Upgrade to PRO

50% OFF for Early Adopters

Show Keyboard Shortcuts

MADE IT WITH ADOBE STOCK. Get 10 free images.

Limited time offer: Get 10 free Adobe Stock images. ads via Carbon

DB Fiddle - Crafted with by Status200 in the United Kingdom

Terms of Use Privacy Policy Status200 Ltd © 2018

Type here to search

1: 0.35 KB/s
1: 0.18 KB/s

6:34 PM
22/10/2022

Schema SQL

```
13 insert into job_data (job_id,
14 actors_id, event, language, time_spent, org, ds)
15 values
16 (21, 1001, 'skip', 'English', 15, 'A', '2020-11-30'),
17 (22, 1006, 'transfer', 'Arabic', 25, 'B', '2020-11-30'),
18 (23, 1003, 'decision', 'Persian', 20, 'C', '2020-11-29'),
19 (24, 1005, 'transfer', 'Persian', 22, 'D', '2020-11-28'),
20 (25, 1002, 'decision', 'Hindi', 11, 'B', '2020-11-28'),
21 (26, 1007, 'decision', 'French', 104, 'D', '2020-11-27'),
22 (27, 1004, 'skip', 'Persian', 56, 'A', '2020-11-26'),
23 (28, 1008, 'transfer', 'Italian', 45, 'C', '2020-11-25'),
24 (28, 1008, 'transfer', 'Italian', 45, 'C', '2020-11-25'),
25 (28, 1008, 'transfer', 'Italian', 45, 'C', '2020-11-25'),
26
27
```

Query SQL

```
1 /**
2 Case1-A: Number of jobs reviewed: Amount of jobs reviewed over time.
3 Your task: Calculate the number of jobs reviewed per hour per day for November 2020?
4 **/
5
6 SELECT
7     COUNT(DISTINCT job_id)/(30*24) AS n_job
8 FROM
9     job_data
10 WHERE
11     ds BETWEEN '2020-11-01' AND '2020-11-30'
```

Results

Query #1 Execution time: 1ms

n_job
0.0111

Copy as Markdown

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

SQL Command:

```

SELECT
    ds,
    n_job,
    AVG(n_job) OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING
AND CURRENT ROW)
    AS throughput_7
FROM
    (
        SELECT
            ds,
            COUNT(DISTINCT job_id) AS n_job
        FROM
            job_data
        WHERE
            ds BETWEEN '2020-11-01' AND '2020-11-30'
        GROUP BY
            ds
        ORDER BY
            ds
    )a

```

Output:

ds	n_job	throughput_7
25/11/2020	1	1.0000
26/11/2020	1	1.0000
27/11/2020	1	1.0000
28/11/2020	2	1.2500
29/11/2020	1	1.2000
30/11/2020	2	1.3333

The screenshot shows the DB Fiddle - SQL Database Playground interface. The Schema SQL editor contains the following SQL code:

```

1  CREATE TABLE job_data (
2    org varchar(255),
3    ds date,
4    event varchar(255),
5    language varchar(255),
6    time_spent int,
7    org varchar(255)
8  );
9  INSERT INTO job_data (org, ds, event, language, time_spent, org) VALUES
10 (1, '2020-11-25', 'skip', 'English', 15, 'A', '2020-11-30'),
11 (2, '2020-11-26', 'transfer', 'Arabic', 25, 'B', '2020-11-30'),
12 (3, '2020-11-27', 'decision', 'Persian', 20, 'C', '2020-11-29'),
13 (4, '2020-11-28', 'transfer', 'Persian', 22, 'D', '2020-11-28'),
14 (5, '2020-11-28', 'decision', 'Hindi', 11, 'B', '2020-11-28'),
15 (6, '2020-11-28', 'decision', 'French', 104, 'D', '2020-11-27');

```

The Query SQL editor contains the following SQL code:

```

1  AS throughput_7
2  FROM
3  (
4    SELECT
5      ds,
6      COUNT(DISTINCT job_id) AS n_job
7    FROM
8      job_data
9    WHERE
10     ds BETWEEN '2020-11-01' AND '2020-11-30'
11   GROUP BY
12     ds
13   ORDER BY
14     ds
15 ) a

```

The Results section shows the following table:

ds	n_job	throughput_7
2020-11-25	1	1.0000
2020-11-26	1	1.0000
2020-11-27	1	1.0000
2020-11-28	2	1.2500

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

SQL Command:

```

SELECT
language,
num_jobs,
100.0 * num_jobs/ total_jobs AS per_jobs
FROM
(
SELECT
language,
COUNT(DISTINCT job_id) AS num_jobs
FROM
job_data
GROUP BY
language
) a
CROSS JOIN
(
SELECT
COUNT(DISTINCT job_id) AS total_jobs
FROM job_data
) b

```

Output:

language	num_jobs	per_jobs
Arabic	1	12.50000
English	1	12.50000
French	1	12.50000
Hindi	1	12.50000
Italian	1	12.50000
Persian	3	37.50000

The screenshot shows the DB Fiddle - SQL Database Playground interface. The top navigation bar includes links to various services like Gmail, YouTube, Maps, Translate, Google Keep, WhatsApp, PBL TalentNext Trac..., Online Training, Trainity Dashboard, Wiley, and Mountblue Hiring. The main interface is divided into three sections: Fiddle Title, Schema SQL, and Query SQL. The Schema SQL section contains the following SQL code:

```
1 create table job_data
2 (
3 job_id int,
4 actors_id int,
5 event varchar(255),
6 language varchar(255),
7 time_spent int,
8 org varchar(255),
9 ds date
10 );
11 select * from job_data;
12 truncate table job_data;
13
14 insert into job_data (job_id,
15 actors_id, event, language, time_spent, org, ds)
16 values (1, 1, 'Arabic', 'Arabic', 12.5, 'Arabic', '2022-10-22');
17
18 insert into job_data (job_id,
19 actors_id, event, language, time_spent, org, ds)
20 values (2, 1, 'English', 'English', 12.5, 'English', '2022-10-22');
21
22 insert into job_data (job_id,
23 actors_id, event, language, time_spent, org, ds)
24 values (3, 1, 'French', 'French', 12.5, 'French', '2022-10-22');
25
26 insert into job_data (job_id,
27 actors_id, event, language, time_spent, org, ds)
28 values (4, 1, 'Hindi', 'Hindi', 12.5, 'Hindi', '2022-10-22');
29
30 insert into job_data (job_id,
31 actors_id, event, language, time_spent, org, ds)
32 values (5, 1, 'Italian', 'Italian', 12.5, 'Italian', '2022-10-22');
33
34 insert into job_data (job_id,
35 actors_id, event, language, time_spent, org, ds)
36 values (6, 3, 'Persian', 'Persian', 37.5, 'Persian', '2022-10-22');
```

The Query SQL section contains the following SQL code:

```
1 /**
2 Case1-C:Percentage share of each language: Share of each language for different
3 contents.
4 Your task: Calculate the percentage share of each language in the last 30 days?
5 **/
6 SELECT
7 language,
8 num_jobs,
9 100.0 * num_jobs / total_jobs AS per_jobs
10 FROM
11 (
12 SELECT
13 language,
14 COUNT(DISTINCT job_id) AS num_jobs
15 FROM job_data
16 GROUP BY language)
```

The Results section displays the output of the query as a table:

language	num_jobs	per_jobs
Arabic	1	12.50000
English	1	12.50000
French	1	12.50000
Hindi	1	12.50000
Italian	1	12.50000
Persian	3	37.50000

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

SQL Command:

```
SELECT *
FROM
(
SELECT *,
ROW_NUMBER()OVER(PARTITION BY job_id) AS rownum
FROM
```

job_data) a
WHERE
rownum>1

Output:

job_id	actors_id	event	language	time_spent	org	ds	rownum
28	1008	transfer	Italian	45	C	25/11/2020	2

DB Fiddle - SQL Database Playground

Schema SQL

```

1 create table job_data
2 (
3   job_id int ,
4   actors_id int ,
5   event varchar(255) ,
6   language varchar(255) ,
7   time_spent int ,
8   org varchar(255) ,
9   ds date
10 );
11 select * from job_data ;
12 truncate table job_data;
13
14 insert into job_data (job_id ,
15 actors_id , event , language , time_spent , org , ds)

```

Query SQL

```

1 /**
2 Case1-CD: Duplicate rows: Rows that have the same value present in them.
3 Your task: Let's say you see some duplicate rows in the data. How will you display
4 duplicates from the table?
5 **/
6 SELECT *
7 FROM
8 (
9   SELECT * ,
10    ROW_NUMBER() OVER (PARTITION BY job_id AS rownum
11   FROM
12    job_data) a
13  WHERE
14    rownum > 1

```

Results

Query #1 Execution time: 0ms

job_id	actors_id	event	language	time_spent	org	ds	rownum
28	1008	transfer	Italian	45	C	2020-11-25	2

2. Case Study 2 (Investigating metric spike):

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

SQL Command:

```

SELECT
  EXTRACT(week from occurred_at) AS nweek,
  COUNT(DISTINCT a.user_id)
FROM
  tutorial.yammer_events a
GROUP BY
  nweek;

```

Output:

nweek	count
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

The screenshot displays the app.mode.com web interface. The browser address bar shows the URL: `app.mode.com/editor/basic_datascience/reports/14e39797cbcb/queries/5370b9ff1185`. The user is logged in as Kishori Sarjerao Patil. The main workspace is titled 'Query 1' and contains a SQL query for calculating weekly user engagement. The query is as follows:

```
Case 2-A: 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
  EXTRACT(week from occurred_at) AS nweek,
  COUNT(DISTINCT a.user_id)
FROM
  tutorial.yammer_events a
GROUP BY
  nweek
```

The query results are displayed in a table with two columns: 'nweek' and 'count'. The results show a steady increase in user engagement over time, with a slight dip at the end of the period shown.

nweek	count
18	791
19	1244
20	1270
21	1341
22	1293
23	1366
24	1434
25	1462
26	1443
27	1477

The interface also includes a sidebar with a list of tables and definitions, and a bottom status bar showing the query execution details: 'Showing rows 1-18 of 18 Columns: 2 Size: 2888 Run: a few seconds ago Returned in: 714ms'.

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

SQL Command:

```
SELECT
  year,
  nweek,
  num_active,
  SUM(num_active)OVER(ORDER BY year,nweek ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW) AS num_active
FROM
  (
  SELECT
    EXTRACT(year from a.activated_at) AS year,
    EXTRACT(week from a.activated_at) AS nweek,
    COUNT(DISTINCT user_id) AS num_active
  FROM
    tutorial.yammer_users a
  WHERE
    state='active'
  GROUP BY
    year,
    nweek
  ORDER BY
    year,
    nweek
  ) a
```

Output:

year	nweek	num_active_duplicate_column_name_1	num_active
2013	1	67	67
2013	2	29	96
2013	3	47	143
2013	4	36	179
2013	5	30	209
2013	6	48	257
2013	7	41	298
2013	8	39	337
2013	9	33	370
2013	10	43	413
2013	11	33	446
2013	12	32	478
2013	13	33	511
2013	14	40	551

2013	15	35	586
2013	16	42	628
2013	17	48	676
2013	18	48	724
2013	19	45	769
2013	20	55	824
2013	21	41	865
2013	22	49	914
2013	23	51	965
2013	24	51	1016
2013	25	46	1062
2013	26	57	1119
2013	27	57	1176
2013	28	52	1228
2013	29	71	1299
2013	30	66	1365
2013	31	69	1434
2013	32	66	1500
2013	33	73	1573
2013	34	70	1643
2013	35	80	1723
2013	36	65	1788
2013	37	71	1859
2013	38	84	1943
2013	39	92	2035
2013	40	81	2116
2013	41	88	2204
2013	42	74	2278
2013	43	97	2375
2013	44	92	2467
2013	45	97	2564
2013	46	94	2658
2013	47	82	2740
2013	48	103	2843
2013	49	96	2939
2013	50	117	3056
2013	51	123	3179
2013	52	104	3283
2014	1	91	3374
2014	2	122	3496

2014	3	112	3608
2014	4	113	3721
2014	5	130	3851
2014	6	132	3983
2014	7	135	4118
2014	8	127	4245
2014	9	127	4372
2014	10	135	4507
2014	11	152	4659
2014	12	132	4791
2014	13	151	4942
2014	14	161	5103
2014	15	166	5269
2014	16	165	5434
2014	17	176	5610
2014	18	172	5782
2014	19	160	5942
2014	20	186	6128
2014	21	177	6305
2014	22	186	6491
2014	23	197	6688
2014	24	198	6886
2014	25	222	7108
2014	26	210	7318
2014	27	199	7517
2014	28	223	7740
2014	29	215	7955
2014	30	228	8183
2014	31	234	8417
2014	32	189	8606
2014	33	250	8856
2014	34	259	9115
2014	35	266	9381

Untitled Report | Mode

app.mode.com/editor/basic_datascience/reports/14e39797cbbf/queries/7a9cd1c2773f

Query2

```

5 SELECT
6   year,
7   nweek,
8   num_active,
9   SUM(num_active)OVER(ORDER BY year,nweek ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS num_active
10  FROM
11  (
12    SELECT
13      EXTRACT(year from a.activated_at) AS year,
14      EXTRACT(week from a.activated_at) AS nweek,
15      COUNT(DISTINCT user_id) AS num_active
16    FROM
17      tutorial.yammer_users a
18    WHERE
19      state='active'
20    GROUP BY
  
```

Mode Public Warehouse (everyone)

Tables Definitions

Search public (everyone)

- tutorial.yammer_users
- tutorial.yammer_events
- tutorial
 - aapl_historical_stock_price
 - accounts
 - animal_crossing_achievements
 - animal_crossing_art
 - animal_crossing_bags
 - animal_crossing_bottoms
 - animal_crossing_construction
 - animal_crossing_dress_up
 - animal_crossing_fencing
 - animal_crossing_fish
 - animal_crossing_floors
 - animal_crossing_fossils
 - animal_crossing_headwear
 - animal_crossing_housewares
 - animal_crossing_insects
 - animal_crossing_miscellaneous
 - animal_crossing_music
 - animal_crossing_other
 - animal_crossing_photos

	year	nweek	num_active_duplicate_column_name_1	num_active
1	2013	1	67	67
2	2013	2	29	96
3	2013	3	47	143
4	2013	4	36	179
5	2013	5	30	209
6	2013	6	48	257
7	2013	7	41	298
8	2013	8	39	337
9	2013	9	33	370

Showing rows: 1-87 of 87 Columns: 4 Size: 3KB Run: a few seconds ago Returned in: 560ms

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

SQL Command:

```

SELECT
    COUNT(user_id),
    SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) as week_1
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
)
ORDER BY
    a.user_id
) a

```

Output:

count	week_1
317	64

The screenshot shows the Mode Public Warehouse interface. The top bar includes navigation links like 'Gmail', 'YouTube', 'Maps', 'Translate', 'Google Keep', 'Freshers My Nauki', 'WhatsApp', 'PBL TalentNext Trac...', 'Online Training', and 'Trainity Dashboard'. The main area displays a SQL query in a dark-themed editor with a 'Run' button and 'Limit 100' option. The query is a complex join between two subqueries. Below the editor, the 'Data' tab shows the results in a table with two columns: 'count' and 'week_1'. The first row shows a count of 317 for week_1. The right sidebar lists various tables in the 'tutorial' schema, including 'tutorial.yammer_users', 'tutorial.yammer_events', and 'tutorial.yammer_emails'. The bottom status bar indicates 'Showing rows: 1 of 1', 'Columns: 2', 'Size: 168', 'Run: a few seconds ago', and 'Returned in: 784ms'.

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

SQL Command:
 SELECT

```

EXTRACT(year FROM occurred_at) AS year,
EXTRACT(week from occurred_at) AS week,
device,
COUNT(distinct user_id)
FROM
tutorial.yammer_events
WHERE
event_type ='engagement'
GROUP BY
1,2,3
ORDER by
1,2,3

```

Output:

year	week	device	count
2014	18	acer aspire desktop	10
2014	18	acer aspire notebook	21
2014	18	amazon fire phone	4
2014	18	asus chromebook	23
2014	18	dell inspiron desktop	21
2014	18	dell inspiron notebook	49
2014	18	hp pavilion desktop	15
2014	18	htc one	16
2014	18	ipad air	30
2014	18	ipad mini	21
2014	18	iphone 4s	21
2014	18	iphone 5	70
2014	18	iphone 5s	45
2014	18	kindle fire	6
2014	18	lenovo thinkpad	90
2014	18	macbook air	57
2014	18	macbook pro	154
2014	18	mac mini	8
2014	18	nexus 10	16
2014	18	nexus 5	43
2014	18	nexus 7	20
2014	18	nokia lumia 635	19
2014	18	samsung galaxy tablet	8
2014	18	samsung galaxy note	7
2014	18	samsung galaxy s4	56

2014	18	windows surface	10
2014	19	acer aspire desktop	26
2014	19	acer aspire notebook	34
2014	19	amazon fire phone	9
2014	19	asus chromebook	42
2014	19	dell inspiron desktop	58
2014	19	dell inspiron notebook	78
2014	19	hp pavilion desktop	37
2014	19	htc one	19
2014	19	ipad air	52
2014	19	ipad mini	29
2014	19	iphone 4s	47
2014	19	iphone 5	114
2014	19	iphone 5s	70
2014	19	kindle fire	26
2014	19	lenovo thinkpad	155
2014	19	macbook air	119
2014	19	macbook pro	248
2014	19	mac mini	12
2014	19	nexus 10	30
2014	19	nexus 5	73
2014	19	nexus 7	29
2014	19	nokia lumia 635	34
2014	19	samsung galaxy tablet	11
2014	19	samsung galaxy note	15
2014	19	samsung galaxy s4	80
2014	19	windows surface	10
2014	20	acer aspire desktop	22
2014	20	acer aspire notebook	40
2014	20	amazon fire phone	12
2014	20	asus chromebook	26
2014	20	dell inspiron desktop	36
2014	20	dell inspiron notebook	82
2014	20	hp pavilion desktop	40
2014	20	htc one	32
2014	20	ipad air	53
2014	20	ipad mini	37
2014	20	iphone 4s	40

2014	20	iphone 5	113
2014	20	iphone 5s	77
2014	20	kindle fire	20
2014	20	lenovo thinkpad	176
2014	20	macbook air	110
2014	20	macbook pro	261
2014	20	mac mini	19
2014	20	nexus 10	25
2014	20	nexus 5	84
2014	20	nexus 7	41
2014	20	nokia lumia 635	22
2014	20	samsung galaxy tablet	6
2014	20	samsung galaxy note	11
2014	20	samsung galaxy s4	90
2014	20	windows surface	15
2014	21	acer aspire desktop	23
2014	21	acer aspire notebook	40
2014	21	amazon fire phone	10
2014	21	asus chromebook	39
2014	21	dell inspiron desktop	52
2014	21	dell inspiron notebook	84
2014	21	hp pavilion desktop	31
2014	21	htc one	27
2014	21	ipad air	54
2014	21	ipad mini	32
2014	21	iphone 4s	56
2014	21	iphone 5	128
2014	21	iphone 5s	75
2014	21	kindle fire	22
2014	21	lenovo thinkpad	177
2014	21	macbook air	119
2014	21	macbook pro	256
2014	21	mac mini	25
2014	21	nexus 10	23
2014	21	nexus 5	99
2014	21	nexus 7	31
2014	21	nokia lumia 635	21

The screenshot shows a web-based SQL editor interface. At the top, there's a browser window with the URL `app.mode.com/editor/basic_datascience/reports/14e39797cbb/bqueries/cd06280a72b`. Below the browser, there's a navigation bar with various icons and a search bar. The main area is divided into two panels. The left panel shows a SQL query editor with a query that filters for 'engagement' events and counts distinct user IDs by year, week, and device. The right panel shows a table of results with columns 'year', 'week', 'device', and 'count'. The table contains 9 rows of data. At the bottom, there's a status bar showing 'Showing rows: 1-100 of 100', 'Columns: 4', 'Size: 4KB', 'Run: a few seconds ago', and 'Returned in: 689ms'.

E. Email Engagement: Users engaging with the email service:

SQL Command:

```
SELECT
    100.0 *SUM(CASE WHEN email_cat = 'email_open' THEN 1 ELSE 0
END)/SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS
email_open_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_clicked_rate FROM
(
    SELECT
        *,
        CASE WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email')
THEN 'email_sent' WHEN action IN ('email_open') THEN 'email_open' WHEN action in
('email_clickthrough') THEN 'email_clicked' END AS email_cat
FROM
    tutorial.yammer_emails
) a
```

Output:

email_open_rate	email_clicked_rate
33.58338804990151	14.789888378200919

Untitled Report | Mode

app.mode.com/editor/basic_datascience/reports/14e39797dcbf/queries/a142b28ed2a5

Gmail YouTube Maps Translate Google Keep Freshers My Naukri WhatsApp PBL TalentNext Trac... Online Training Trainity Dashboard

Kishori Sarjarao Patil | Untitled Report
Report - Embed Share Schedule View

Query 4

Run Limit 100 Format SQL View History...

3 Your task: Calculate the email engagement metrics?
4 **/
5
6 SELECT
7 100.0 *SUM(CASE WHEN email_cat = 'email_open' THEN 1 ELSE 0 END) /SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_open_rate,
8 100.0 *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_clicked_rate FROM
9
10 - (
11 SELECT
12 *,
13 CASE WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email_sent' WHEN action IN ('email_open') THEN 'email_open' WHEN action in ('email_clickthrough') THEN 'email_clicked' FROM
14 FROM
15 tutorial.yammer_emails
16)

Ready

Data Fields Source

	email_open_rate	email_clicked_rate
1	33.5834	14.7899

Export Copy

Showing rows: 1 of 1 Columns: 2 Size: 328 Run: a few seconds ago Returned in: 614ms

Mode Public Warehouse (everyone)

Tables Definitions

Search public (everyone)

- tutorial.yammer_users
- tutorial.yammer_events
- tutorial.yammer_emails

tutorial

- apl_historical_stock_price
- accounts
- animal_crossing_achievements
- animal_crossing_art
- animal_crossing_bags
- animal_crossing_bottoms
- animal_crossing_construction
- animal_crossing_dress_up
- animal_crossing_fencing
- animal_crossing_fish
- animal_crossing_floors
- animal_crossing_fossils
- animal_crossing_headwear
- animal_crossing_housewares
- animal_crossing_insects
- animal_crossing_miscellaneous
- animal_crossing_music
- animal_crossing_other

Type here to search

1:224 KB/s
L:94.3 KB/s

407 PM
30/10/2022