Operation Analytics and investigating Metric spike

Project description:

- To do analysis on the following case studies
- To get thoroughly understanding importance of operation and investigating metric queries

Software used:

- **MySQL Workbench 8.0 CE**
- **Mode.com**

Case study 1 : **Job data**

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

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

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

Your 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.

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

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

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

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**.**

  Your task: Calculate the weekly user engagement?

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

  Your task: Calculate the user growth for product?

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

  Your 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.

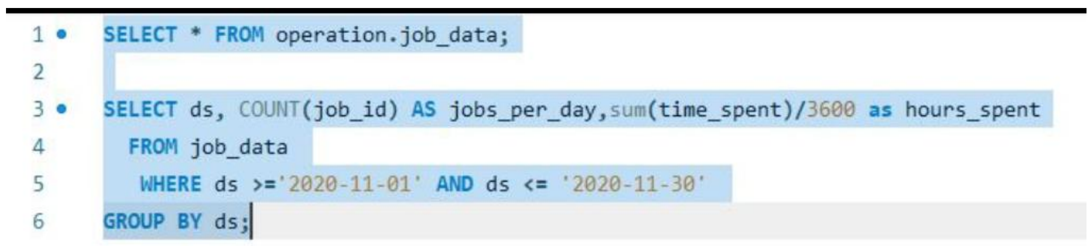
Your task: Calculate the weekly engagement per device?

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

Case1:job data

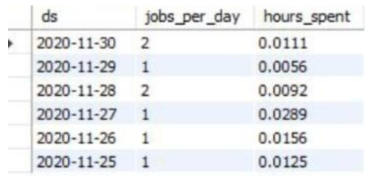
Task 1: Calculate the number of jobs reviewed per hour per day for November 2020?

SQL query:



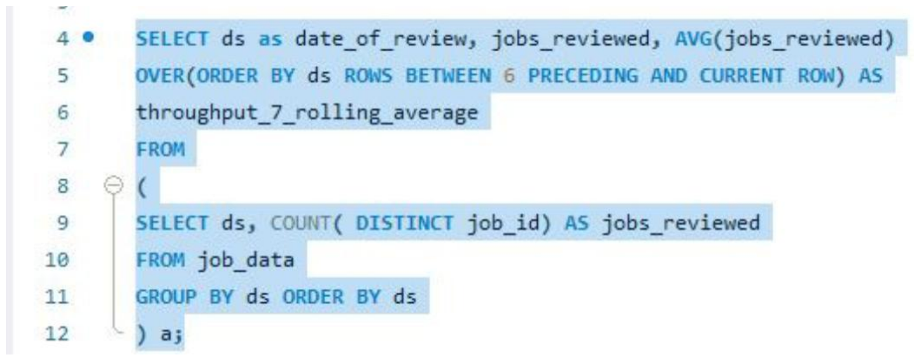
```
SELECT * FROM operation.job_data;

SELECT ds, COUNT(job_id) AS jobs_per_day,sum(time_spent)/3600 as hours_spent
  FROM job_data
   WHERE ds >='2020-11-01' AND ds <= '2020-11-30'
GROUP BY ds;
```

Output:



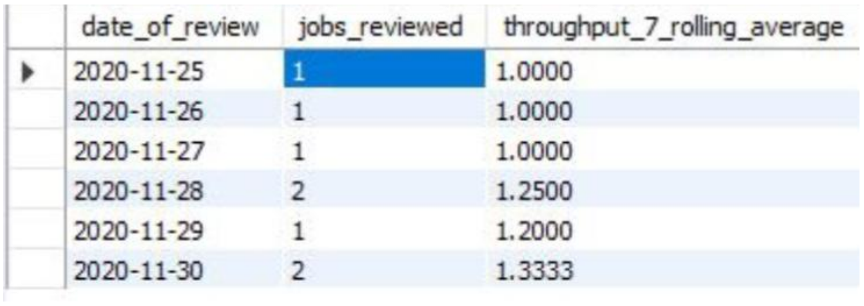
ds	jobs_per_day	hours_spent
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

Task 2: Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?
SQL QUERY:



```
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 job_data
GROUP BY ds ORDER BY ds
) a;
```

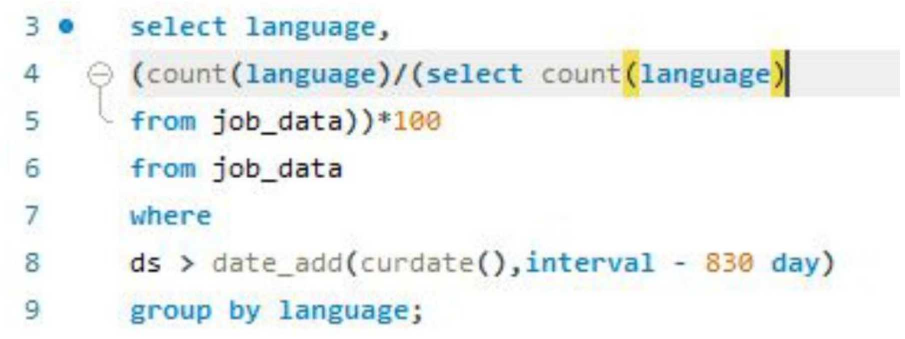
Output:



	date_of_review	jobs_reviewed	throughput_7_rolling_average
▶	2020-11-25	1	1.0000
	2020-11-26	1	1.0000
	2020-11-27	1	1.0000
	2020-11-28	2	1.2500
	2020-11-29	1	1.2000
	2020-11-30	2	1.3333

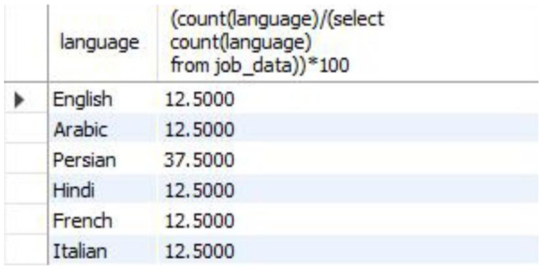
Task 3: Calculate the percentage share of each language in the last 30 days?

SQL QUERY:



```
select language,
(count(language)/(select count(language)
from job_data))*100
from job_data
where
ds > date_add(curdate(),interval - 830 day)
group by language;
```

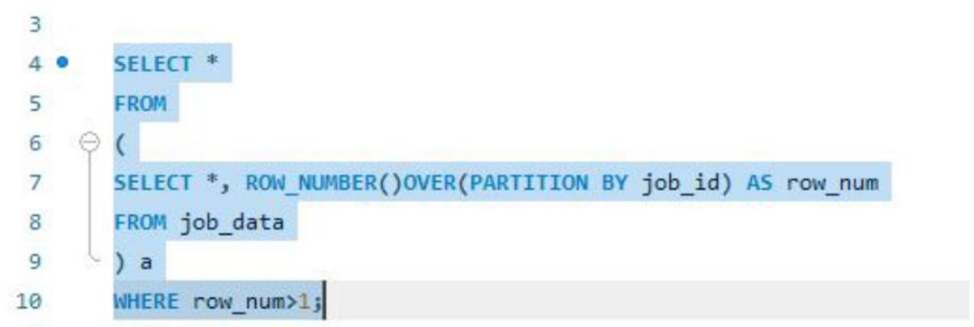
OUTPUT:



	language	(count(language)/(select count(language) from job_data))*100
▶	English	12.5000
	Arabic	12.5000
	Persian	37.5000
	Hindi	12.5000
	French	12.5000
	Italian	12.5000

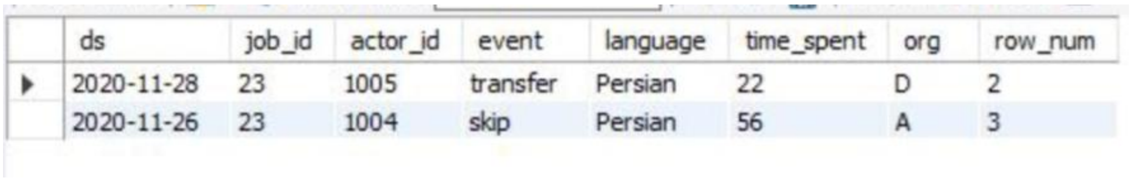
Task 4: Let’s say you see some duplicate rows in the data. How will you display duplicates from the table?

SQL QUERY:



```
SELECT *
FROM
(
SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num
FROM job_data
) a
WHERE row_num>1;
```

OUTPUT:

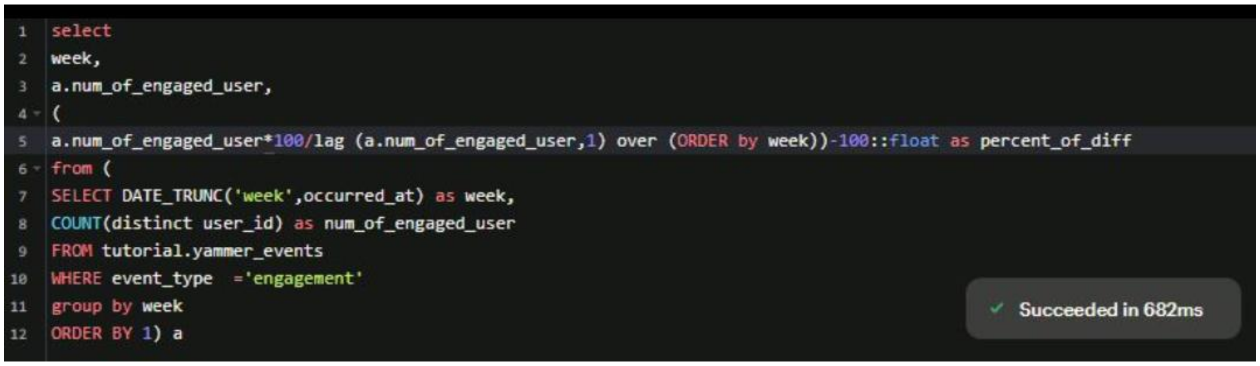


ds	job_id	actor_id	event	language	time_spent	org	row_num
2020-11-28	23	1005	transfer	Persian	22	D	2
2020-11-26	23	1004	skip	Persian	56	A	3

Case2:
investigating metrics spike

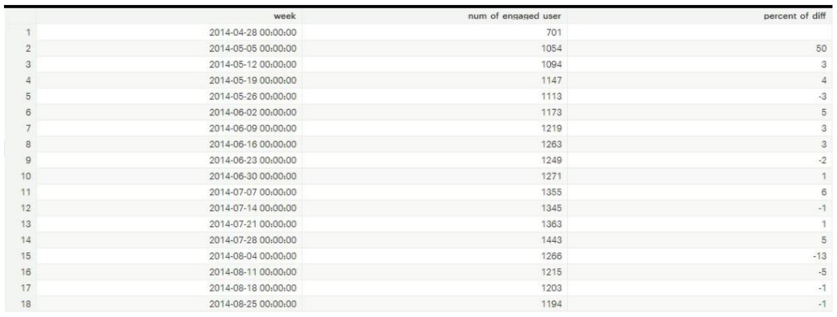
Task 1: Calculate the weekly user engagement?

SQL QUERY:



```
select
week,
a.num_of_engaged_user,
(
a.num_of_engaged_user*100/lag (a.num_of_engaged_user,1) over (ORDER by week))-100::float as percent_of_diff
from (
SELECT DATE_TRUNC('week',occurred_at) as week,
COUNT(distinct user_id) as num_of_engaged_user
FROM tutorial.yammer_events
WHERE event_type  ='engagement'
group by week
ORDER BY 1) a
```

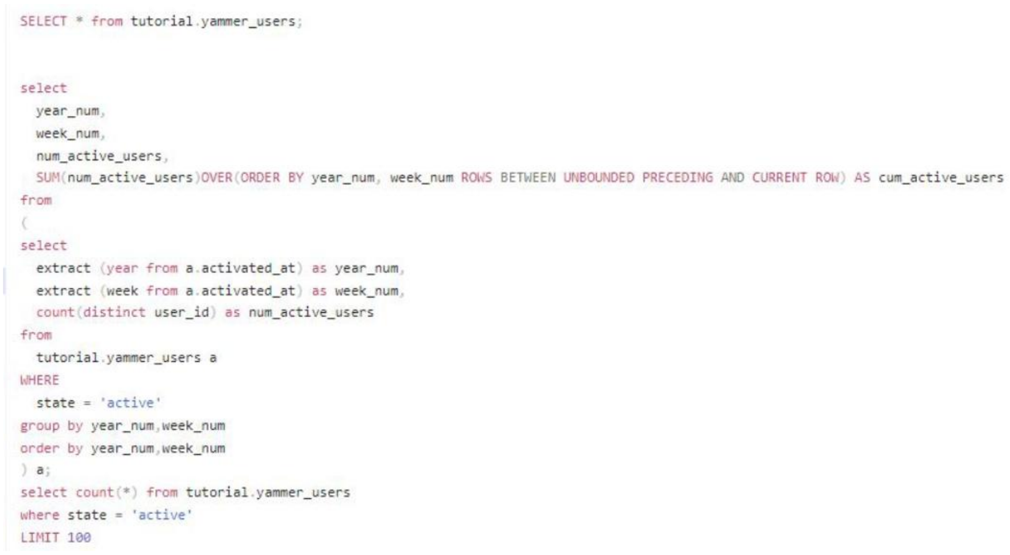
OUTPUT:



	week	num of engaged user	percent of diff
1	2014-04-28 00:00:00	701	
2	2014-05-05 00:00:00	1054	50
3	2014-05-12 00:00:00	1094	3
4	2014-05-19 00:00:00	1147	4
5	2014-05-26 00:00:00	1113	-3
6	2014-06-02 00:00:00	1173	5
7	2014-06-09 00:00:00	1219	3
8	2014-06-16 00:00:00	1263	3
9	2014-06-23 00:00:00	1249	-2
10	2014-06-30 00:00:00	1271	1
11	2014-07-07 00:00:00	1355	6
12	2014-07-14 00:00:00	1345	-1
13	2014-07-21 00:00:00	1363	1
14	2014-07-28 00:00:00	1443	5
15	2014-08-04 00:00:00	1266	-13
16	2014-08-11 00:00:00	1215	-5
17	2014-08-18 00:00:00	1203	-1
18	2014-08-25 00:00:00	1194	-1

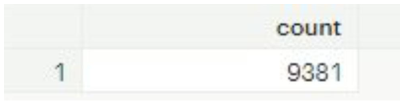
Task 2: Calculate the user growth for product?

SQL QUERY:



```
SELECT * from tutorial.yammer_users;


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;
select count(*) from tutorial.yammer_users
where state = 'active'
LIMIT 100
```

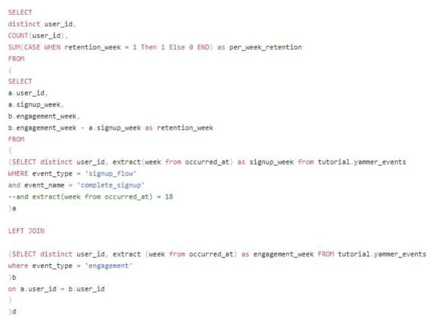
OUTPUT:



	count
1	9381

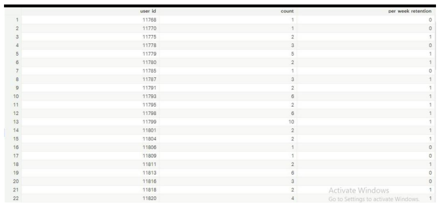
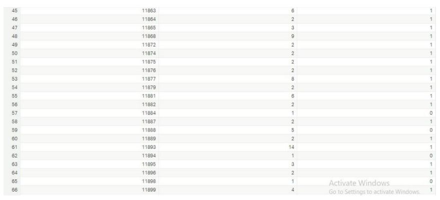
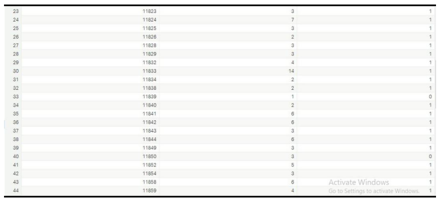
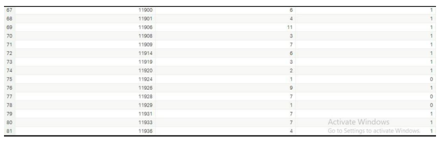
Task 3: Calculate the weekly retention of users-sign up cohort?

SQL 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 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
```

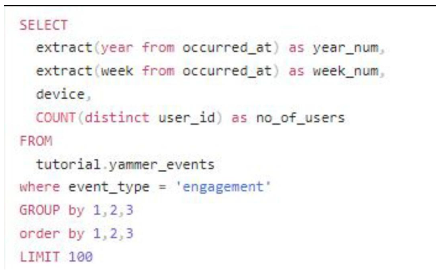
OUTPUT:

	user id	count	per week retention
1	11768	1	0
2	11770	1	0
3	11775	2	1
4	11778	3	0
5	11779	5	1
6	11780	2	1
7	11785	1	0
8	11787	3	1
9	11791	2	1
10	11793	6	1
11	11795	2	1
12	11798	6	1
13	11799	10	1
14	11801	2	1
15	11804	2	1
16	11806	1	0
17	11809	1	0
18	11811	2	1
19	11813	6	0
20	11816	3	0
21	11818	2	1
22	11820	4	1
23	11823	3	1
24	11824	7	1
25	11825	3	1
26	11826	2	1
27	11828	3	1
28	11829	3	1
29	11832	4	1
30	11833	14	1
31	11834	2	1
32	11838	2	1
33	11839	1	0
34	11840	2	1
35	11841	6	1
36	11842	6	1
37	11843	3	1
38	11844	6	1
39	11849	3	1
40	11850	3	0
41	11852	5	1
42	11854	3	1
43	11858	6	1
44	11859	4	1
45	11863	6	1
46	11864	2	1
47	11865	3	1
48	11868	9	1
49	11872	2	1
50	11874	2	1
51	11875	2	1
52	11876	2	1
53	11877	8	1
54	11879	2	1
55	11881	6	1
56	11882	2	1
57	11884	1	0
58	11887	2	1
59	11888	5	0
60	11889	2	1
61	11893	14	1
62	11894	1	0
63	11895	3	1
64	11896	2	1
65	11898	1	0
66	11899	4	1
67	11900	6	1
68	11901	4	1
69	11906	11	1
70	11908	3	1
71	11909	7	1
72	11914	6	1
73	11919	3	1
74	11920	2	1
75	11924	1	0
76	11926	9	1
77	11928	7	0
78	11929	1	0
79	11931	7	1
80	11933	7	1
81	11936	4	1

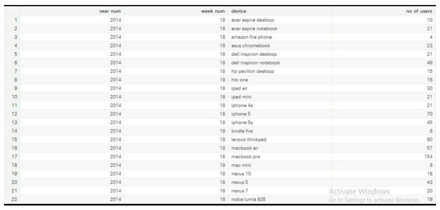
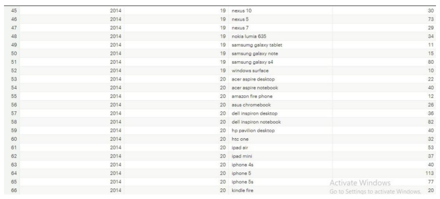
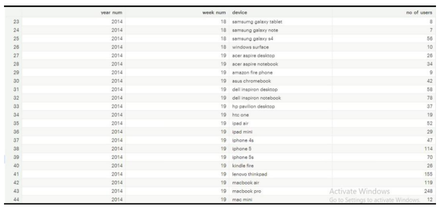
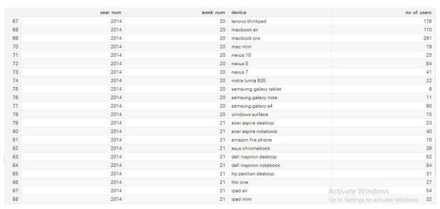
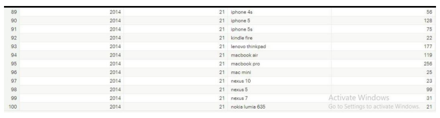
Task 4: Calculate the weekly engagement per device?

SQL 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
  tutorial.yammer_events
where event_type = 'engagement'
GROUP by 1,2,3
order by 1,2,3
LIMIT 100
```

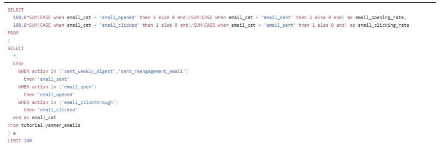
OUTPUT:

	year num	week num	device	no of users
1	2014	18	acer aspire desktop	10
2	2014	18	acer aspire notebook	21
3	2014	18	amazon fire phone	4
4	2014	18	asus chromebook	23
5	2014	18	dell inspiron desktop	21
6	2014	18	dell inspiron notebook	49
7	2014	18	hp pavilion desktop	15
8	2014	18	htc one	16
9	2014	18	ipad air	30
10	2014	18	ipad mini	21
11	2014	18	iphone 4s	21
12	2014	18	iphone 5	70
13	2014	18	iphone 5s	45
14	2014	18	kindle fire	6
15	2014	18	lenovo thinkpad	90
16	2014	18	macbook air	57
17	2014	18	macbook pro	154
18	2014	18	mac mini	8
19	2014	18	nexus 10	16
20	2014	18	nexus 5	43
21	2014	18	nexus 7	20
22	2014	18	nokia lumia 635	19
23	2014	18	samsumg galaxy tablet	8
24	2014	18	samsung galaxy note	7
25	2014	18	samsung galaxy s4	56
26	2014	18	windows surface	10
27	2014	19	acer aspire desktop	26
28	2014	19	acer aspire notebook	34
29	2014	19	amazon fire phone	9
30	2014	19	asus chromebook	42
31	2014	19	dell inspiron desktop	58
32	2014	19	dell inspiron notebook	78
33	2014	19	hp pavilion desktop	37
34	2014	19	htc one	19
35	2014	19	ipad air	52
36	2014	19	ipad mini	29
37	2014	19	iphone 4s	47
38	2014	19	iphone 5	114
39	2014	19	iphone 5s	70
40	2014	19	kindle fire	26
41	2014	19	lenovo thinkpad	155
42	2014	19	macbook air	119
43	2014	19	macbook pro	248
44	2014	19	mac mini	12
45	2014	19	nexus 10	30
46	2014	19	nexus 5	73
47	2014	19	nexus 7	29
48	2014	19	nokia lumia 635	34
49	2014	19	samsumg galaxy tablet	11
50	2014	19	samsung galaxy note	15
51	2014	19	samsung galaxy s4	80
52	2014	19	windows surface	10
53	2014	20	acer aspire desktop	22
54	2014	20	acer aspire notebook	40
55	2014	20	amazon fire phone	12
56	2014	20	asus chromebook	26
57	2014	20	dell inspiron desktop	36
58	2014	20	dell inspiron notebook	82
59	2014	20	hp pavilion desktop	40
60	2014	20	htc one	32
61	2014	20	ipad air	53
62	2014	20	ipad mini	37
63	2014	20	iphone 4s	40
64	2014	20	iphone 5	113
65	2014	20	iphone 5s	77
66	2014	20	kindle fire	20
67	2014	20	lenovo thinkpad	176
68	2014	20	macbook air	110
69	2014	20	macbook pro	261
70	2014	20	mac mini	19
71	2014	20	nexus 10	25
72	2014	20	nexus 5	84
73	2014	20	nexus 7	41
74	2014	20	nokia lumia 635	22
75	2014	20	samsumg galaxy tablet	6
76	2014	20	samsung galaxy note	11
77	2014	20	samsung galaxy s4	90
78	2014	20	windows surface	15
79	2014	21	acer aspire desktop	23
80	2014	21	acer aspire notebook	40
81	2014	21	amazon fire phone	10
82	2014	21	asus chromebook	39
83	2014	21	dell inspiron desktop	52
84	2014	21	dell inspiron notebook	84
85	2014	21	hp pavilion desktop	31
86	2014	21	htc one	27
87	2014	21	ipad air	54
88	2014	21	ipad mini	32
89	2014	21	iphone 4s	56
90	2014	21	iphone 5	128
91	2014	21	iphone 5s	75
92	2014	21	kindle fire	22
93	2014	21	lenovo thinkpad	177
94	2014	21	macbook air	119
95	2014	21	macbook pro	256
96	2014	21	mac mini	25
97	2014	21	nexus 10	23
98	2014	21	nexus 5	99
99	2014	21	nexus 7	31
100	2014	21	nokia lumia 635	21

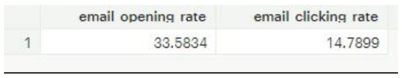
Task 5: Calculate the email engagement metrics?

SQL QUERY:



```
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
LIMIT 100
```

OUTPUT:



	email opening rate	email clicking rate
1	33.5834	14.7899

<u>RESULT</u>:

We got knowledge of advance SQL queries which could make our job easy