OPERATION AND METRIC ANALYTICS

By- Abdul Gafoor

PROJECT DESCRIPTION:

Operation Analytics and Investigating Metric Spike

- Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect.
- Being one of the most important parts of a company, this kind of analysis is further used to predict the overall growth or decline of a company's fortune. It means better automation, better understanding between cross-functional teams, and more effective workflows.
- Investigating metric spike is also an important part of operation analytics as being a Data Analyst you must be able to understand or make other teams understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that its very important to investigate metric spike.
- You are working for a company like Microsoft designated as Data Analyst Lead and is provided with different data sets, tables from which you must derive certain insights out of it and answer the questions asked by different departments.

APPROACH:

- Firstly, go through the dataset to know more about the data, tables, columns and the rows.
- Analyze the data given in the dataset to write the sql query .
- Use SQL queries to get the data asked by the product manager. Write the SQL query to retrieve the data from the database/dataset in the software you are using.

Tech-Stack Used:

- PostgreSQL server
- PostgreSQL documentation
- Pg Admin
- MySQL Workbench 8.0

These software's provide better data security, more In-built functions and On demand scalability and it also provides auto completing feature which makes it easy for the developer to write the queries.

Insights:

- While making this project I learnt about the SQL, how to implement the queries and about the various in built functions that can be used to get the data.

- I got a good exposure about SQL and ProgreSQL, how SQL and the functions of ProgreSQL can be used in analysing the data from the dateset, how the queries work and are executed and how I play a major role in retrieving the data instantly without searching for it in the dataset.

- You need to understand and alnalyze the dataset to write the correct query and get the results.

Results:

Case Study 1 : *Job Data*

A) 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?

Query:

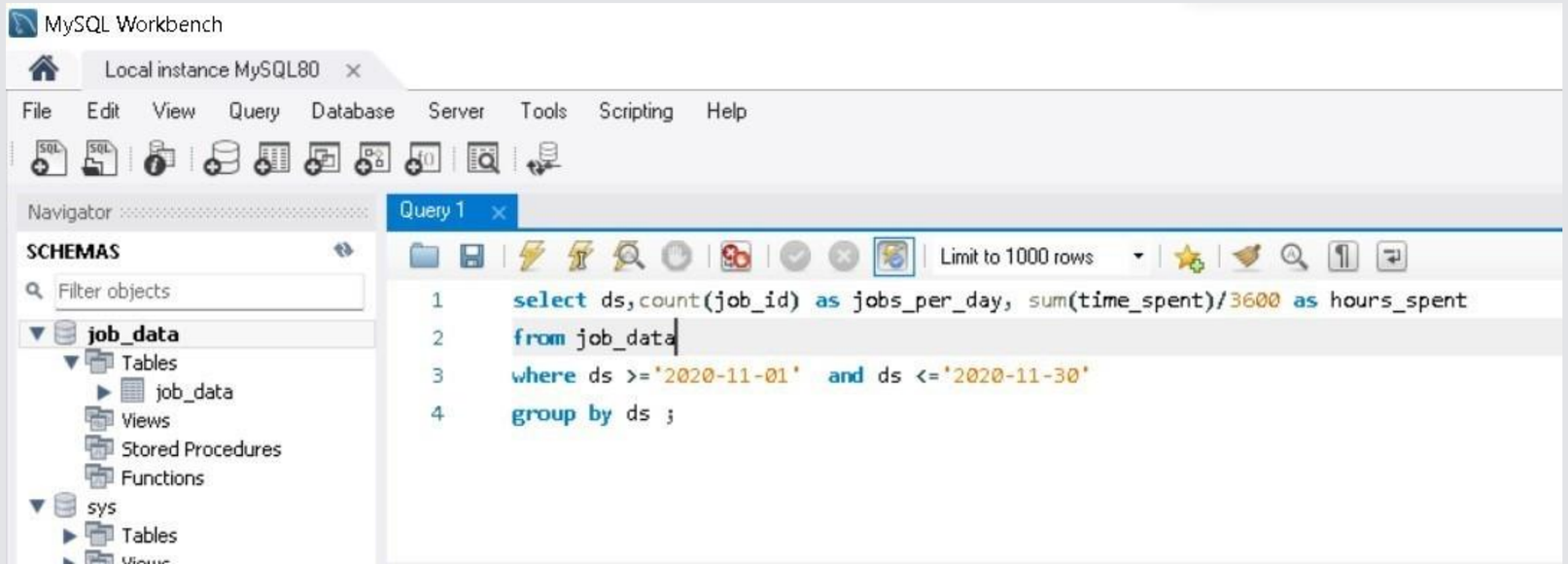


Table (output):

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

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

Result Grid | Form Editor | Field Types

Result 1 | Read Only

Output

In this I have used **sum** and comparison operators in the query to get the result.

B) 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?

Query:

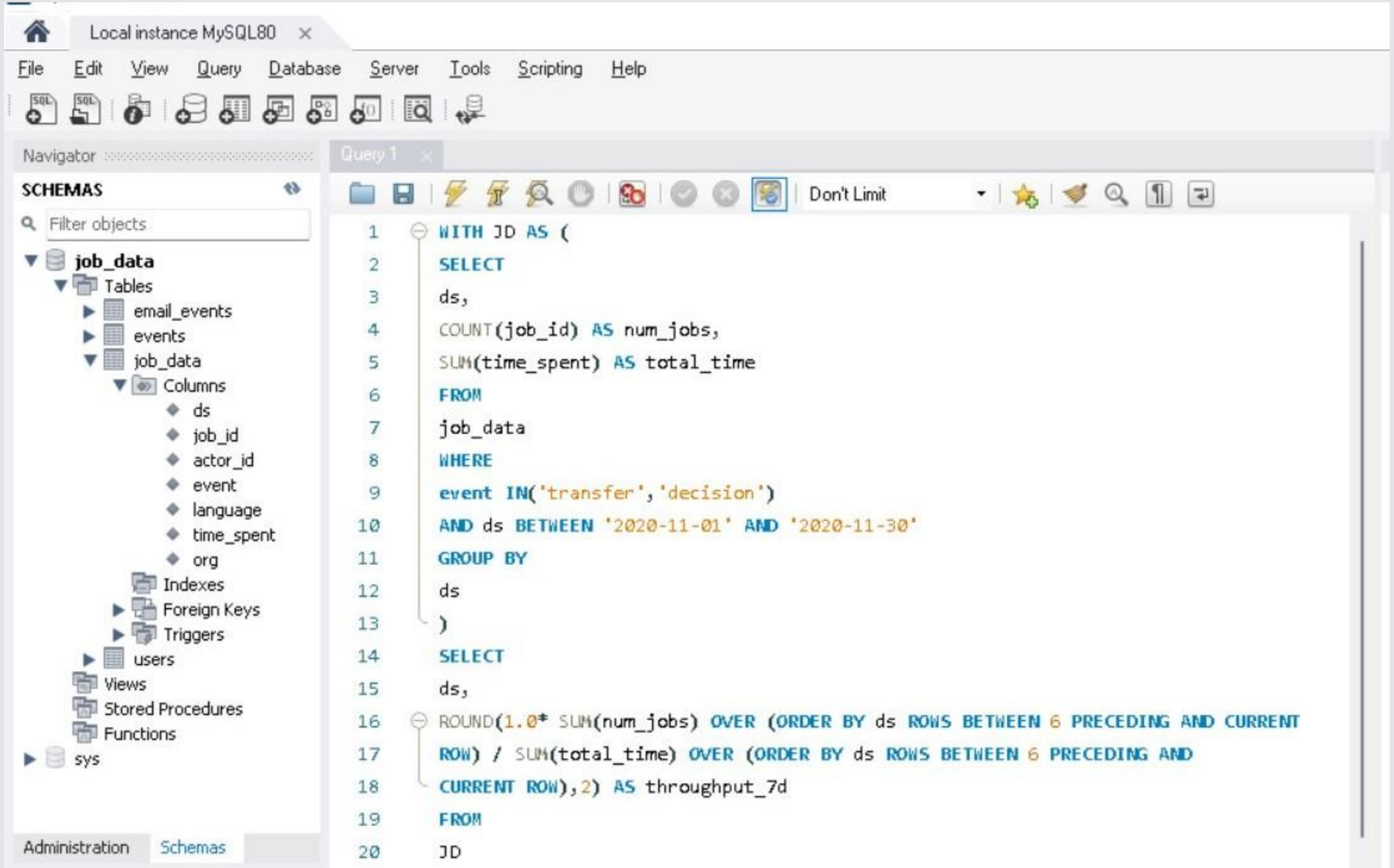
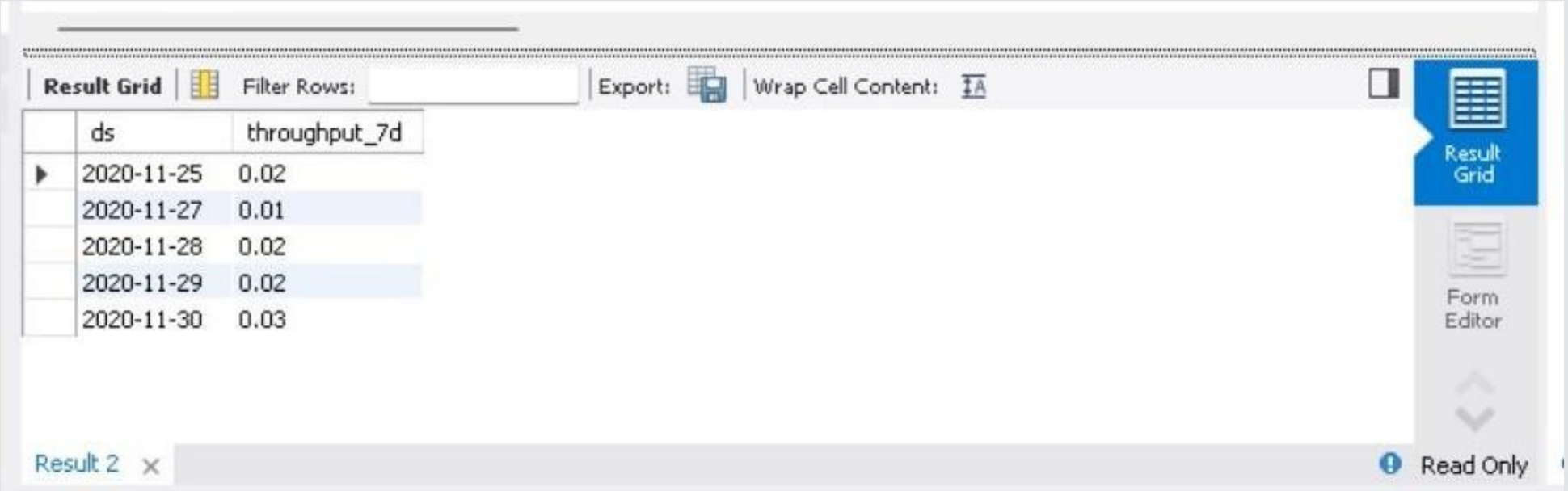


Table (output):



ds	throughput_7d
2020-11-25	0.02
2020-11-27	0.01
2020-11-28	0.02
2020-11-29	0.02
2020-11-30	0.03

- Here I have used ROund , SUM and With function to get the throughput.

C) 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?

Query:

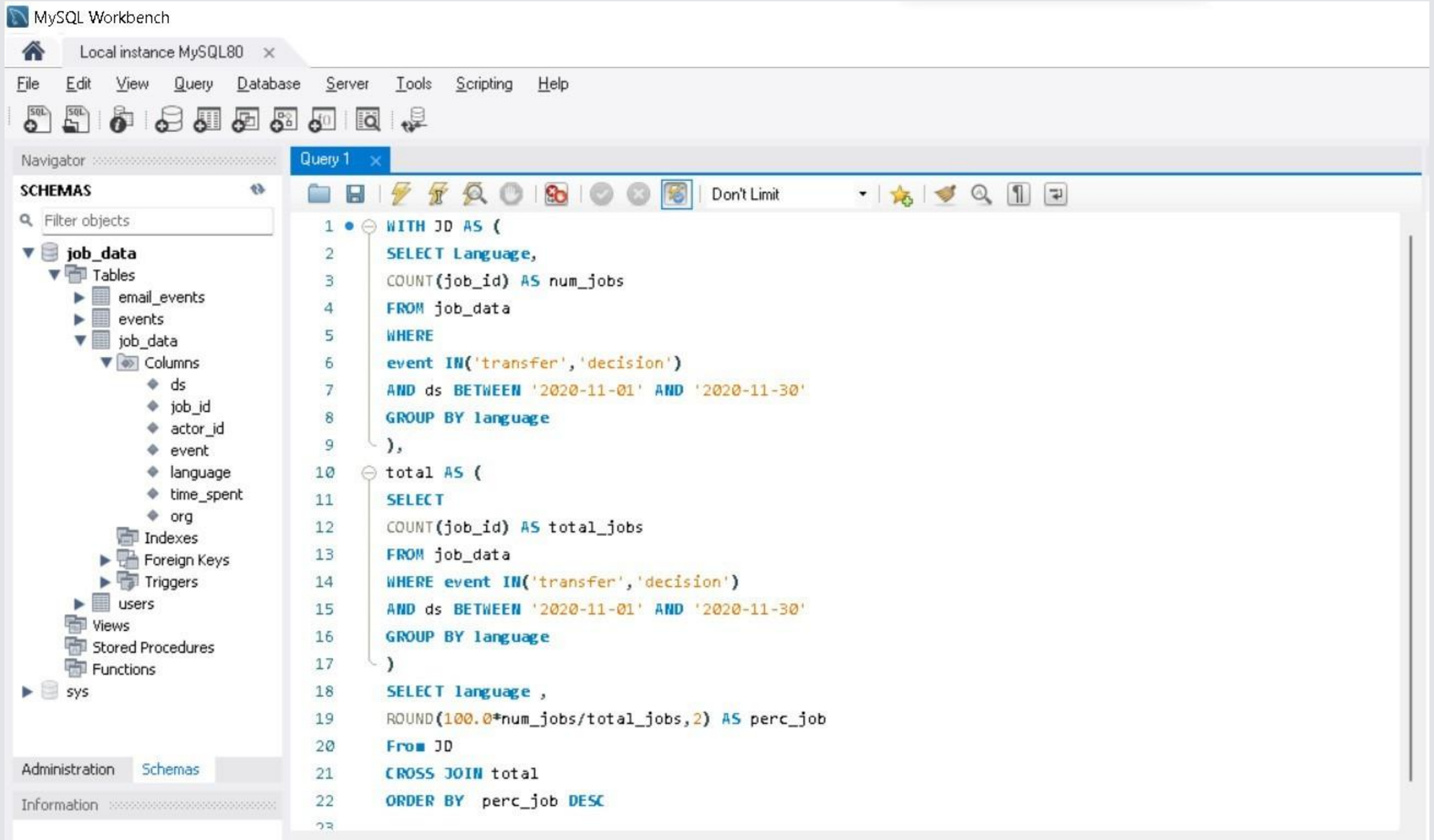


Table (output):

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

language	perc_job
Persian	200.00
Persian	200.00
Persian	200.00
Persian	200.00
Italian	100.00
French	100.00
Hindi	100.00
Arabic	100.00
Persian	100.00
Italian	100.00
French	100.00
Hindi	100.00
Arabic	100.00
Italian	100.00
French	100.00
Hindi	100.00
Arabic	100.00
Italian	100.00
French	100.00
Hindi	100.00
Arabic	100.00
Italian	50.00
French	50.00
Hindi	50.00
Arabic	50.00

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

In this query i used , count , Round function to calculate the percentage

D) 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?

Query:

Query 1

Don't Limit

```
select ds,job_id
from job_data
group by ds
having count(ds) > 1;
```

- Here we use Count , and group by to find the duplicate rows

Table (output):

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

ds	job_id
2020-11-30	21
2020-11-28	23

count(ds)	job_id
1	23
1	11
1	23
1	20
2	21
2	23

- From the above table we can conclude that Job_id 22 and 23 have duplicate values

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.

Your task: Calculate the weekly user engagement?

Query:

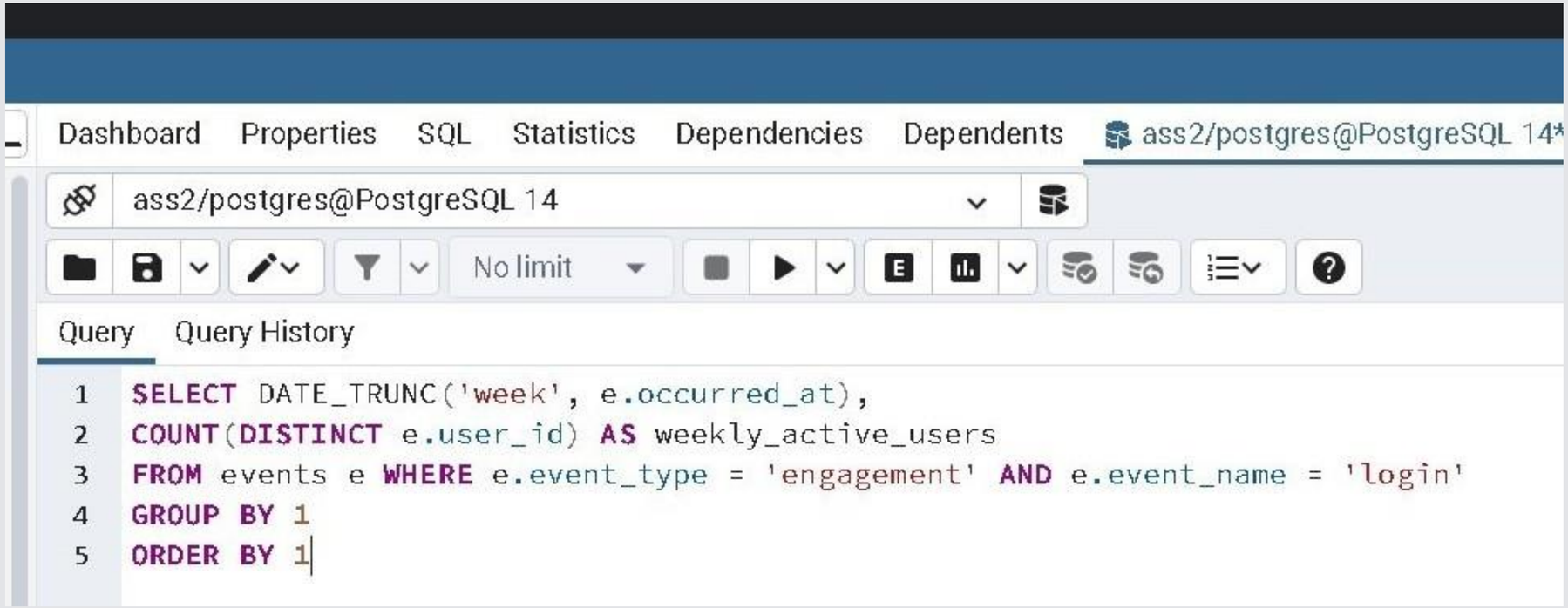
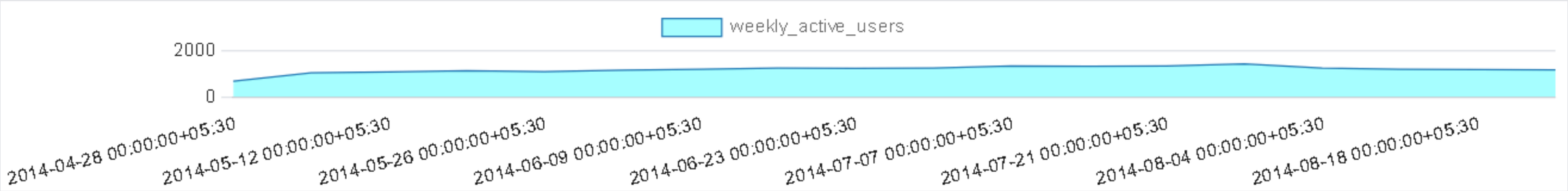


Table (output):

	A	B	C
1	week	weekly_active_users	
2	2014-04-28 00:00:00+05:30	701	
3	2014-05-05 00:00:00+05:30	1054	
4	2014-05-12 00:00:00+05:30	1094	
5	2014-05-19 00:00:00+05:30	1147	
6	2014-05-26 00:00:00+05:30	1113	
7	2014-06-02 00:00:00+05:30	1173	
8	2014-06-09 00:00:00+05:30	1219	
9	2014-06-16 00:00:00+05:30	1262	
10	2014-06-23 00:00:00+05:30	1249	
11	2014-06-30 00:00:00+05:30	1271	
12	2014-07-07 00:00:00+05:30	1355	
13	2014-07-14 00:00:00+05:30	1345	
14	2014-07-21 00:00:00+05:30	1363	
15	2014-07-28 00:00:00+05:30	1442	
16	2014-08-04 00:00:00+05:30	1266	
17	2014-08-11 00:00:00+05:30	1215	
18	2014-08-18 00:00:00+05:30	1203	
19	2014-08-25 00:00:00+05:30	1194	
20			

- Here the output is extracted and downloaded in the form of csv file to make it look simpler, this feature is provided by pg Admin.

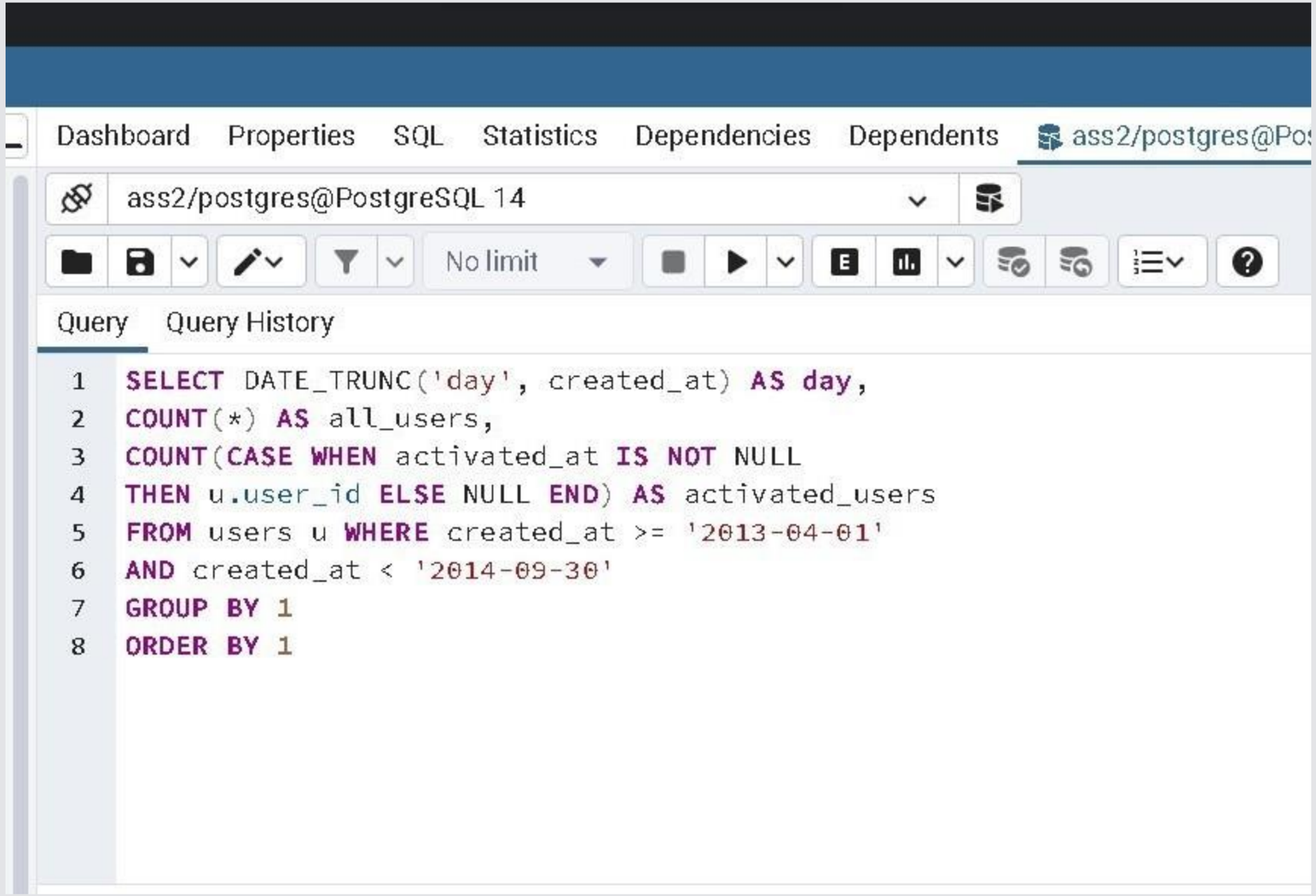
Graph:


weekly_active_users
2000
0
2014-04-28 00:00:00+05:30
2014-05-12 00:00:00+05:30
2014-05-26 00:00:00+05:30
2014-06-09 00:00:00+05:30
2014-06-23 00:00:00+05:30
2014-07-07 00:00:00+05:30
2014-07-21 00:00:00+05:30
2014-08-04 00:00:00+05:30
2014-08-18 00:00:00+05:30

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

Your task: Calculate the user growth for product?

Query:



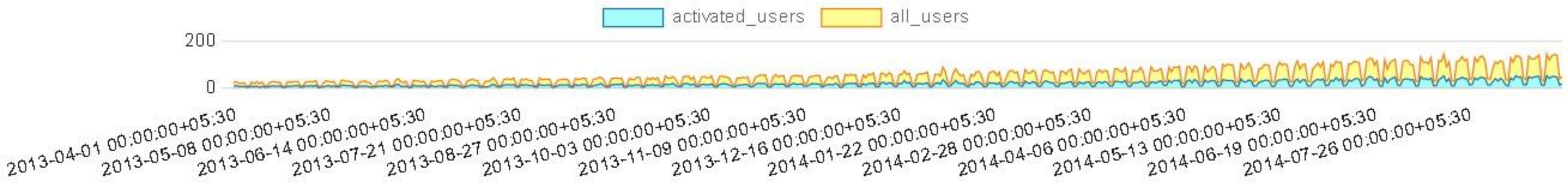

```sql
SELECT DATE_TRUNC('day', created_at) AS day,
COUNT(*) AS all_users,
COUNT(CASE WHEN activated_at IS NOT NULL
THEN u.user_id ELSE NULL END) AS activated_users
FROM users u WHERE created_at >= '2013-04-01'
AND created_at < '2014-09-30'
GROUP BY 1
ORDER BY 1
```

Table (output):

	A	B	C	D
1	day	all_users	activated_users	
2	2013-04-01 00:00:00+05:30	15	7	
3	2013-04-02 00:00:00+05:30	17	10	
4	2013-04-03 00:00:00+05:30	14	5	
5	2013-04-04 00:00:00+05:30	14	6	
6	2013-04-05 00:00:00+05:30	16	6	
7	2013-04-06 00:00:00+05:30	5	3	
8	2013-04-07 00:00:00+05:30	5	3	
9	2013-04-08 00:00:00+05:30	17	8	
10	2013-04-09 00:00:00+05:30	14	4	
11	2013-04-10 00:00:00+05:30	19	8	
12	2013-04-11 00:00:00+05:30	17	2	
13	2013-04-12 00:00:00+05:30	18	6	
14	2013-04-13 00:00:00+05:30	6	4	
15	2013-04-14 00:00:00+05:30	5	3	
16	2013-04-15 00:00:00+05:30	17	6	
17	2013-04-16 00:00:00+05:30	17	9	
18	2013-04-17 00:00:00+05:30	19	7	
19	2013-04-18 00:00:00+05:30	16	7	
20	2013-04-19 00:00:00+05:30	15	9	
21	2013-04-20 00:00:00+05:30	4	2	
22	2013-04-21 00:00:00+05:30	5	2	
23	2013-04-22 00:00:00+05:30	19	8	
24	2013-04-23 00:00:00+05:30	17	6	
25	2013-04-24 00:00:00+05:30	19	8	
26	2013-04-25 00:00:00+05:30	17	10	
27	2013-04-26 00:00:00+05:30	19	9	
28	2013-04-27 00:00:00+05:30	4	3	
29	2013-04-28 00:00:00+05:30	6	4	
30	2013-04-29 00:00:00+05:30	16	9	
31	2013-04-30 00:00:00+05:30	18	7	

66	2013-06-04 00:00:00+05:30	24	10
67	2013-06-05 00:00:00+05:30	20	7
68	2013-06-06 00:00:00+05:30	17	5
69	2013-06-07 00:00:00+05:30	21	8
70	2013-06-08 00:00:00+05:30	6	2
71	2013-06-09 00:00:00+05:30	6	1
72	2013-06-10 00:00:00+05:30	21	12
73	2013-06-11 00:00:00+05:30	22	7
74	2013-06-12 00:00:00+05:30	22	9
75	2013-06-13 00:00:00+05:30	19	6
76	2013-06-14 00:00:00+05:30	19	10
77	2013-06-15 00:00:00+05:30	7	5
78	2013-06-16 00:00:00+05:30	5	2
79	2013-06-17 00:00:00+05:30	18	11
80	2013-06-18 00:00:00+05:30	21	5
81	2013-06-19 00:00:00+05:30	23	6
82	2013-06-20 00:00:00+05:30	22	9
83	2013-06-21 00:00:00+05:30	23	9
84	2013-06-22 00:00:00+05:30	6	3
85	2013-06-23 00:00:00+05:30	7	3
86	2013-06-24 00:00:00+05:30	21	12
87	2013-06-25 00:00:00+05:30	24	13
88	2013-06-26 00:00:00+05:30	26	10
89	2013-06-27 00:00:00+05:30	22	11
90	2013-06-28 00:00:00+05:30	21	7
91	2013-06-29 00:00:00+05:30	6	1
92	2013-06-30 00:00:00+05:30	6	3
93	2013-07-01 00:00:00+05:30	23	8
94	2013-07-02 00:00:00+05:30	24	10
95	2013-07-03 00:00:00+05:30	24	13

Graph:


activated_users
all_users
200
0
2013-04-01 00:00:00+05:30
2013-05-08 00:00:00+05:30
2013-06-14 00:00:00+05:30
2013-07-21 00:00:00+05:30
2013-08-27 00:00:00+05:30
2013-10-03 00:00:00+05:30
2013-11-09 00:00:00+05:30
2013-12-16 00:00:00+05:30
2014-01-22 00:00:00+05:30
2014-02-28 00:00:00+05:30
2014-04-06 00:00:00+05:30
2014-05-13 00:00:00+05:30
2014-06-19 00:00:00+05:30
2014-07-26 00:00:00+05:30

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

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

Query:

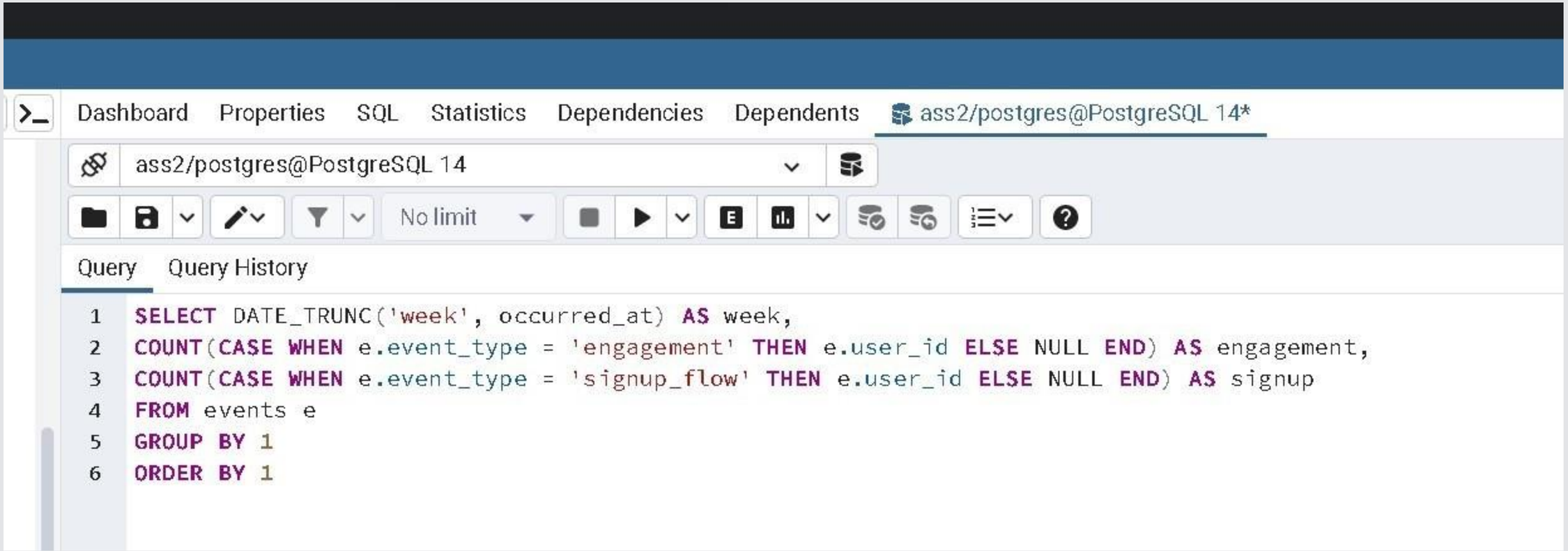
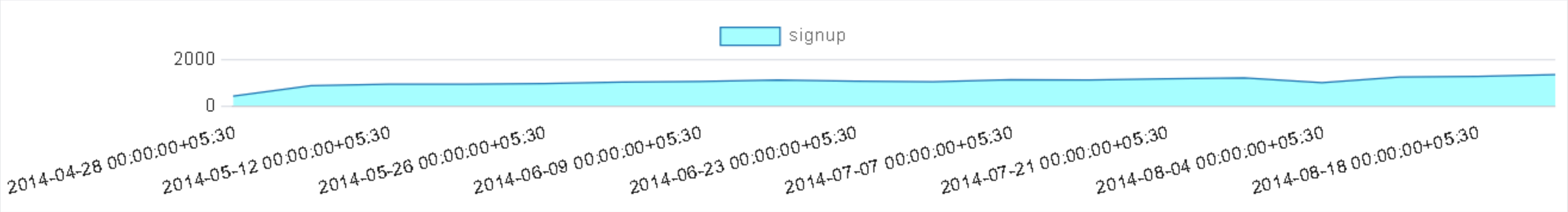


Table (output):

	A	B	C
1	week	engagement	signup
2	2014-04-28 00:00:00+05:30	8709	440
3	2014-05-05 00:00:00+05:30	17532	884
4	2014-05-12 00:00:00+05:30	17047	960
5	2014-05-19 00:00:00+05:30	17890	955
6	2014-05-26 00:00:00+05:30	17193	978
7	2014-06-02 00:00:00+05:30	18608	1043
8	2014-06-09 00:00:00+05:30	18233	1073
9	2014-06-16 00:00:00+05:30	18976	1136
10	2014-06-23 00:00:00+05:30	18859	1081
11	2014-06-30 00:00:00+05:30	18959	1057
12	2014-07-07 00:00:00+05:30	19965	1147
13	2014-07-14 00:00:00+05:30	20723	1130
14	2014-07-21 00:00:00+05:30	20132	1192
15	2014-07-28 00:00:00+05:30	21472	1228
16	2014-08-04 00:00:00+05:30	18341	1017
17	2014-08-11 00:00:00+05:30	16612	1270
18	2014-08-18 00:00:00+05:30	16158	1290
19	2014-08-25 00:00:00+05:30	16166	1376
20			

Graph:


signup
2000
0
2014-04-28 00:00:00+05:30
2014-05-12 00:00:00+05:30
2014-05-26 00:00:00+05:30
2014-06-09 00:00:00+05:30
2014-06-23 00:00:00+05:30
2014-07-07 00:00:00+05:30
2014-07-21 00:00:00+05:30
2014-08-04 00:00:00+05:30
2014-08-18 00:00:00+05:30

D) 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?

Query:

Dashboard Properties SQL Statistics Dependencies Dependents ass2/postgres@PostgreSQL 14*

ass2/postgres@PostgreSQL 14

No limit

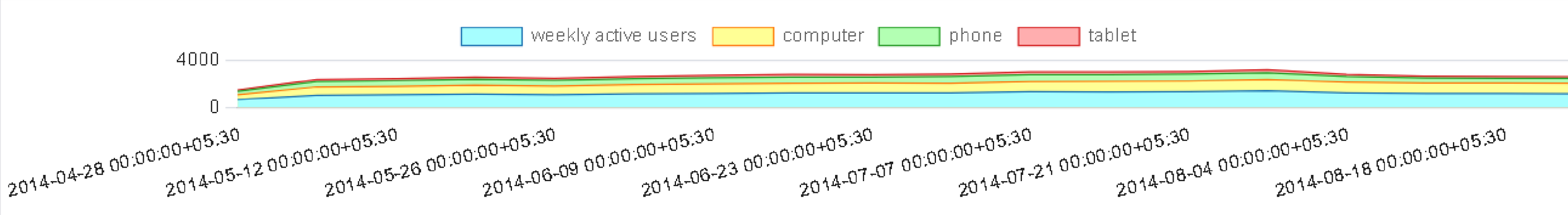
Query Query History

```sql
SELECT DATE_TRUNC('week', occurred_at) AS week,
COUNT(DISTINCT e.user_id) AS "weekly active users",
COUNT(DISTINCT CASE WHEN e.device IN('macbook pro','lenovo thinkpad','macbook air','dell inspiron notebook',
'asus chromebook','dell inspiron desktop',
'acer aspire notebook','hp pavilion desktop','acer aspire desktop','mac mini')
THEN e.user_id ELSE NULL END) AS computer,
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,
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
FROM events e
WHERE e.event_type = 'engagement'
AND e.event_name = 'login'
GROUP BY 1
ORDER BY 1
LIMIT 100
```

Table (output):

	A	B	C	D	E	F
1	week	weekly active users	computer	phone	tablet	
2	2014-04-28 00:00:00+05:30	701	415	281	103	
3	2014-05-05 00:00:00+05:30	1054	712	461	176	
4	2014-05-12 00:00:00+05:30	1094	715	481	191	
5	2014-05-19 00:00:00+05:30	1147	758	526	181	
6	2014-05-26 00:00:00+05:30	1113	716	500	176	
7	2014-06-02 00:00:00+05:30	1173	791	505	197	
8	2014-06-09 00:00:00+05:30	1219	798	545	195	
9	2014-06-16 00:00:00+05:30	1262	812	541	227	
10	2014-06-23 00:00:00+05:30	1249	834	526	210	
11	2014-06-30 00:00:00+05:30	1271	805	578	218	
12	2014-07-07 00:00:00+05:30	1355	877	591	227	
13	2014-07-14 00:00:00+05:30	1345	900	578	218	
14	2014-07-21 00:00:00+05:30	1363	903	601	218	
15	2014-07-28 00:00:00+05:30	1442	951	588	241	
16	2014-08-04 00:00:00+05:30	1266	913	491	166	
17	2014-08-11 00:00:00+05:30	1215	886	438	153	
18	2014-08-18 00:00:00+05:30	1203	875	428	145	
19	2014-08-25 00:00:00+05:30	1194	864	441	150	
20						
21						

Graph:


weekly active users
computer
phone
tablet
4000
0
2014-04-28 00:00:00+05:30
2014-05-12 00:00:00+05:30
2014-05-26 00:00:00+05:30
2014-06-09 00:00:00+05:30
2014-06-23 00:00:00+05:30
2014-07-07 00:00:00+05:30
2014-07-21 00:00:00+05:30
2014-08-04 00:00:00+05:30
2014-08-18 00:00:00+05:30

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

Your task: Calculate the email engagement metrics?

Query:

Dashboard Properties SQL Statistics Dependencies Dependents ass2/postgres@PostgreSQL 14*

ass2/postgres@PostgreSQL 14

No limit

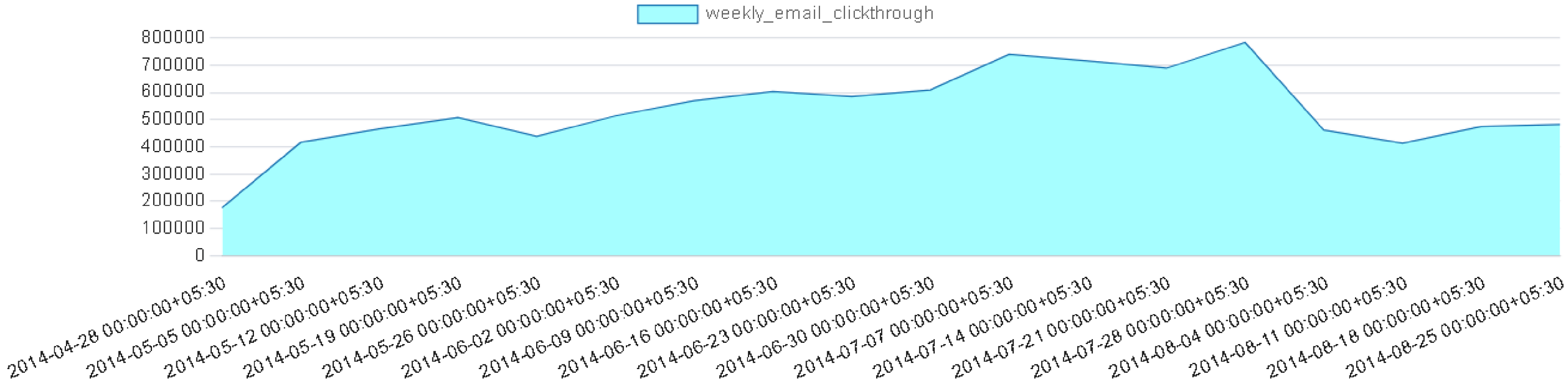
Query Query History

```sql
SELECT DATE_TRUNC('week', occurred_at) AS week,
COUNT(CASE WHEN e.action = 'sent weekly digest' THEN e.user_id
ELSE NULL END) AS "weekly email",
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 "email opens",
COUNT(CASE WHEN e.action = 'email clickthrough' THEN e.user_id
ELSE NULL END) AS "email clickthroughs"
FROM email_events e
GROUP BY 1
```

Table (output):

	A	B	C	D	E	F
1	week	weekly_active_user	weekly_email_open	weekly_email_clickthrough		
2	2014-04-28 00:00:00+05:30	529770	281485	176421		
3	2014-05-05 00:00:00+05:30	1383173	748239	416565		
4	2014-05-12 00:00:00+05:30	1459972	807252	465996		
5	2014-05-19 00:00:00+05:30	1562340	868222	508170		
6	2014-05-26 00:00:00+05:30	1528567	837872	438816		
7	2014-06-02 00:00:00+05:30	1675576	885705	514473		
8	2014-06-09 00:00:00+05:30	1773600	974845	570411		
9	2014-06-16 00:00:00+05:30	1912242	1066423	603765		
10	2014-06-23 00:00:00+05:30	1926260	1030526	585546		
11	2014-06-30 00:00:00+05:30	2001068	1080145	609933		
12	2014-07-07 00:00:00+05:30	2250837	1249579	740493		
13	2014-07-14 00:00:00+05:30	2312688	1266045	717183		
14	2014-07-21 00:00:00+05:30	2293806	1213196	689886		
15	2014-07-28 00:00:00+05:30	2590335	1453679	784635		
16	2014-08-04 00:00:00+05:30	2209513	1169269	461997		
17	2014-08-11 00:00:00+05:30	2052487	1063707	413547		
18	2014-08-18 00:00:00+05:30	2157707	1141452	475704		
19	2014-08-25 00:00:00+05:30	2193130	1203444	482742		
20						

Graph:


weekly_email_clickthrough
800000
700000
600000
500000
400000
300000
200000
100000
0
2014-04-28 00:00:00+05:30
2014-05-05 00:00:00+05:30
2014-05-12 00:00:00+05:30
2014-05-19 00:00:00+05:30
2014-05-26 00:00:00+05:30
2014-06-02 00:00:00+05:30
2014-06-09 00:00:00+05:30
2014-06-16 00:00:00+05:30
2014-06-23 00:00:00+05:30
2014-06-30 00:00:00+05:30
2014-07-07 00:00:00+05:30
2014-07-14 00:00:00+05:30
2014-07-21 00:00:00+05:30
2014-07-28 00:00:00+05:30
2014-08-04 00:00:00+05:30
2014-08-11 00:00:00+05:30
2014-08-18 00:00:00+05:30
2014-08-25 00:00:00+05:30

Monthly query:

Dashboard Properties SQL Statistics Dependencies Dependents ass2/postgres@PostgreSQL 14*

ass2/postgres@PostgreSQL 14

No limit

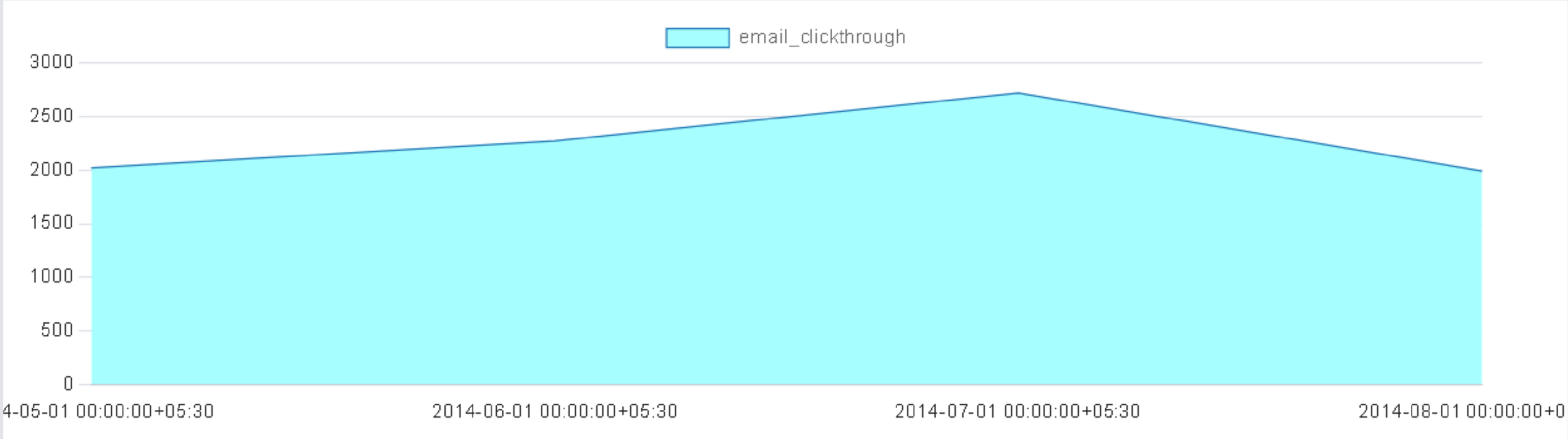
Query Query History

```sql
SELECT DATE_TRUNC('month',occurred_at) AS month ,
       COUNT(CASE WHEN action ='email_open' THEN user_id ELSE NULL END) email_open,
       COUNT(CASE WHEN action ='email_clickthrough' THEN user_id ELSE NULL END) email_clickthrough,
       COUNT(CASE WHEN action ='sent_weekly_digest' THEN user_id ELSE NULL END) sent_reengagement_email,
       COUNT(CASE WHEN action ='sent_reengagement_email' THEN user_id ELSE NULL END) sent_reengagement_email
FROM email_events
GROUP BY 1
ORDER BY 1
```

Table (output):

	A	B	C	D	E	F	G	H
1	month	email_open	email_clickthrough	sent_reengagement_email	sent_reengagement_email-2			
2	2014-05-0	4212	2023	11730	758			
3	2014-06-0	4658	2274	13155	889			
4	2014-07-0	5611	2721	15902	933			
5	2014-08-0	5978	1992	16480	1073			
6								
7								
8								

Graph:



- Here we have imported the outputs in csv format so that all the values can be shown in the table , and the graph is included by using pg Admin.


THE
END