

# Project: Operation Analytics and Investigating Metric Spike

NAME: MUKESH CHANDRA KAMILA

**Description:**

In this project, I was designated as Data Analyst Lead and provided with different data sets. I was focusing on analysing the data which is provided by the company. So, the task is to derive insights and answer the questions asked by different departments. So that these insights are then used by operation team, support team and marketing team to predict the overall growth or decline of the company. It means better automation, better understanding between cross effective workflows.

In Case Study 1, there is job\_data table wherever in Case Study 2 there are users, events and email\_event tables.

**Approach:**

To execute the project, I used SQL. So, the first approach was to run SQL queries to create a database using the raw data provided. Once the database was created, I imported the raw data and run various sorting and data extracting queries to get the required insights.

**Tech-Stack:** PostgreSQL 6.19

**Used:**

PostgreSQL 6.19 was used in this project execution. The ease of access and set up with convenient user interface made it a good tool for the project.

**Insights:**

In this project, I learned about advanced SQL. And how to analyse the problem statement and the functions I can use in SQL to solve the problem statement and write the queries to get required output. Following are the questions that has been answered with the help of SQL.

In this project, I have achieved and gain knowledge how to deal with data with help of SQL. And how to interact and run queries to get desired output from database.

**Q1.** Task: Calculate the number of jobs reviewed per hour per day for November 2020?

```
select ds as Dates, count(job_id)::decimal/sum(time_spent)*3600
as num_jobs_reviewed
from job_data
where ds between '2020-11-01' and '2020-11-30'
group by ds;
```

Page | 3

**Q2. Task:** Calculate 7 day rolling average of throughput?

For throughput, do you prefer daily metric or 7-day rolling and why?

**Query:**

**--Weekly throughput**

```
select round(cast(count(event)as decimal)/sum(time_spent),2)
```

```
as weekly_throughput
```

```
from job_data;
```

--or

```
select round(count(event)::decimal /sum(time_spent),2) as weekly_throughput
```

```
from job_data;
```

**--Daily throughput**

```
select ds as Dates,round(count(event)::decimal/sum(time_spent),2)
```

```
as daily_throughput
```

```
from job_data
```

```
group by ds
```

```
order by ds;
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree is expanded to 'PostgreSQL 15' > 'Databases (6)' > 'Operation\_Analytics'. The 'Query' tab is active, displaying the following SQL code:

```
19
20 /* Throughput: It is the no. of events happening per second.
21 Task: Calculate 7 day rolling average of throughput?
22 For throughput, do you prefer daily metric or 7-day rolling and why? */
23
24 --Weekly throughput
25 select round(cast(count(event)as decimal)/sum(time_spent),2) as weekly_throughput
26 from job_data;
27 --or
28 select round(count(event)::decimal /sum(time_spent),2) as weekly_throughput
29 from job_data;
30
31 --Daily throughput
32 select ds as Dates,round(count(event)::decimal/sum(time_spent),2) as daily_throughput
33 from job_data
34 group by ds
35 order by ds;
```

Below the query editor, the 'Data Output' tab shows the results of the query. The first column is labeled 'weekly\_throughput' and is of type 'numeric'. The first row shows a value of 0.03.

weekly_throughput
0.03

At the bottom, the status bar indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.050'.

pgAdmin 4

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents Processes Operation\_Analytics.sql

Servers (1)

- PostgreSQL 15
  - Databases (6)
    - Operation\_Analytics**
      - Casts
      - Catalogs
      - Event Triggers
      - Extensions
      - Foreign Data Wrappers
      - Languages
      - Publications
      - Schemas
      - Subscriptions
      - dvdrental
      - exercise
      - instagram\_users
      - postgres
      - superstore
      - Login/Group Roles
      - Tablespaces

Operation\_Analytics/postgres@PostgreSQL 15

No limit

Query Query History

```

21 Task: Calculate 7 day rolling average of throughput?
22 For throughput, do you prefer daily metric or 7-day rolling and why? */
23
24 --Weekly throughput
25 select round(cast(count(event) as decimal)/sum(time_spent),2) as weekly_throughput
26 from job_data;
27 --or
28 select round(count(event)::decimal /sum(time_spent),2) as weekly_throughput
29 from job_data;
30
31 --Daily throughput
32 select ds as Dates,round(count(event)::decimal/sum(time_spent),2) as daily_throughput
33 from job_data
34 group by ds
35 order by ds;
  
```

Data Output Messages Notifications

	dates date	daily_throughput numeric
1	2020-11-25	0.02
2	2020-11-26	0.02
3	2020-11-27	0.01
4	2020-11-28	0.06
5	2020-11-29	0.05
6	2020-11-30	0.05

Total rows: 6 of 6 Query complete 00:00:00.073

**Q3. Task:** Calculate the percentage share of each language in the last 30 days?

**Query:**

```
select language as Languages,round(100*count(*)::decimal/total,2)
```

```
as percentage
```

```
from job_data
```

```
cross join (select count(*) as total from job_data)sub
```

```
group by language,total;
```

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Servers' tree is expanded to 'PostgreSQL 15' > 'Databases (6)' > 'Operation\_Analytics'. The 'Query' tab is active, displaying the following SQL query:

```
36
37
38 /* Percentage share of each language: Share of each language for different contents.
39 Task: Calculate the percentage share of each language in the last 30 days? */
40
41 select language as Languages,round(100*count(*)::decimal/total,2) as percentage
42 from job_data
43 cross join (select count(*) as total from job_data)sub
44 group by language,total;
45
46
```

Below the query editor, the 'Data Output' tab shows the results of the query. The results are displayed in a table with two columns: 'languages' (character varying (50)) and 'percentage' (numeric). The table contains 6 rows of data:

	languages	percentage
1	Italian	12.50
2	Hindi	12.50
3	French	12.50
4	English	12.50
5	Arabic	12.50
6	Persian	37.50

At the bottom of the interface, a status bar indicates 'Total rows: 6 of 6' and 'Query complete 00:00:00.073'.

**Q4. Task:** display duplicates rows in the data.

**Query:**

```
select actor_id,count(*) as duplicates
```

```
from job_data
```

```
group by actor_id
```

```
having count(*)> 1;
```

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Servers' tree is expanded to 'PostgreSQL 15' > 'Databases (6)' > 'Operation\_Analytics'. The main panel displays the 'Query' tab with the following SQL code:

```
46
47 /* Duplicate rows: Rows that have the same value present in them.
48 Task: display duplicates rows in the data. */
49
50 select actor_id,count(*) as duplicates
51 from job_data
52 group by actor_id
53 having count(*)> 1;
54
55
56
```

Below the query editor, the 'Data Output' tab is active, showing a table with the following data:

actor_id	duplicates
1	2

The status bar at the bottom indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.088'.

## Investigating Metric Spike

**Q1. Task:** Calculate the weekly user engagement?

**Query:**

```
select extract (week from occurred_at) as week_numbers,  
       count (distinct user_id) as weekly_active_users  
from events  
where event_type = 'engagement'  
group by 1;
```

The screenshot shows the pgAdmin 4 web interface. The left sidebar displays the database structure, including Servers (1), PostgreSQL 15, Databases (6), and the selected database Operation\_Analytics. The main panel shows the SQL query editor with the following query:

```
/* User Engagement: To measure the activeness of a user.  
Task: Calculate the weekly user engagement? */  
  
select extract(week from occurred_at) as week_numbers,  
       count(distinct user_id) as weekly_active_users  
from events  
where event_type = 'engagement'  
group by 1;
```

The query results are displayed in the Data Output tab, showing a table with two columns: week\_numbers (numeric) and weekly\_active\_users (bigint). The results are as follows:

	week_numbers	weekly_active_users
1	18	701
2	19	1054
3	20	1094
4	21	1147
5	22	1113
6	23	1173
7	24	1219
8	25	1263
9	26	1249
10	27	1271
11	28	1355

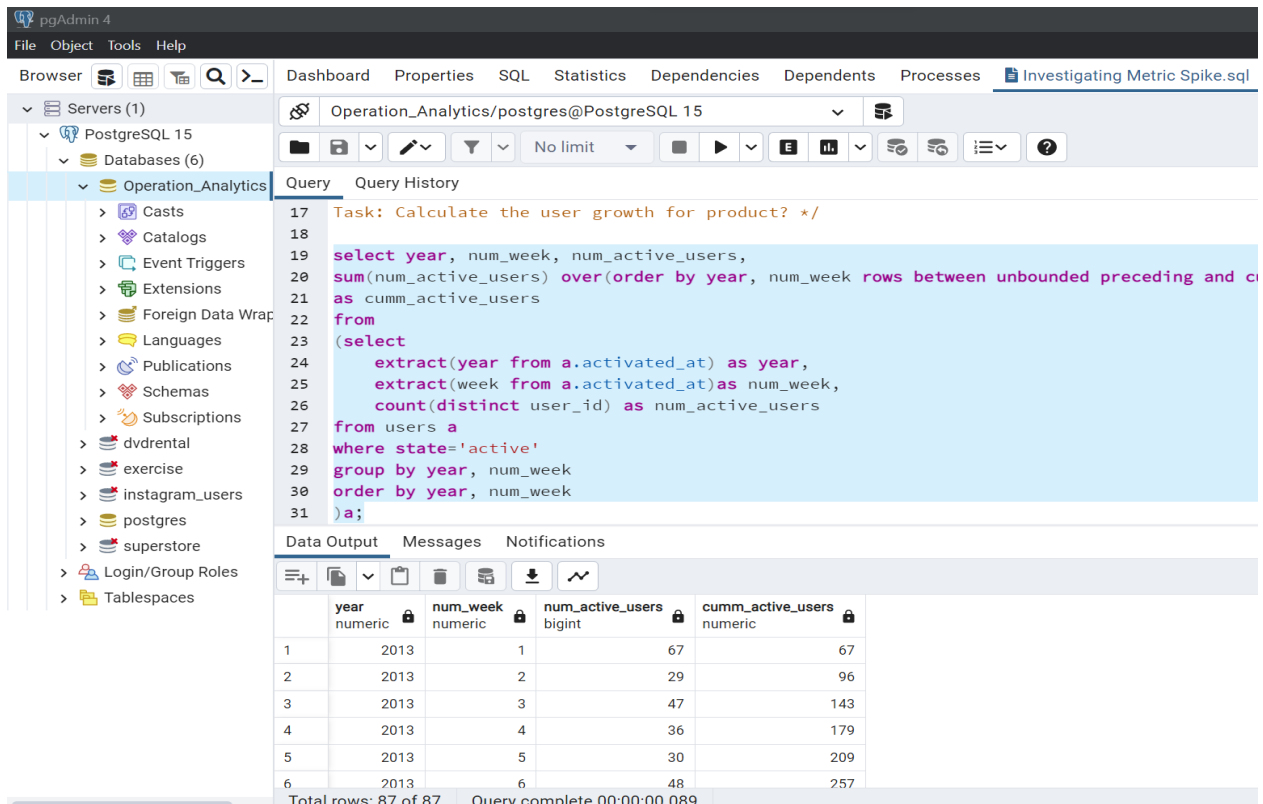
The status bar at the bottom indicates "Total rows: 18 of 18" and "Query complete 00:00:00.352".



## Q2. Task: Calculate the user growth for product?

### Query:

```
select year, num_week, num_active_users,  
sum(num_active_users) over(order by year, num_week rows between unbounded  
preceding and current row)  
as cumm_active_users  
from  
(select  
    extract(year from a.activated_at) as year,  
    extract(week from a.activated_at)as num_week,  
    count(distinct user_id) as num_active_users  
from users a  
where state='active'  
group by year, num_week  
order by year, num_week  
)a;
```



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including 'Servers (1)' with 'PostgreSQL 15' and 'Databases (6)'. The 'Operation\_Analytics' database is selected, showing various objects like 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wraps', 'Languages', 'Publications', 'Schemas', 'Subscriptions', 'dvdrental', 'exercise', 'instagram\_users', 'postgres', 'superstore', 'Login/Group Roles', and 'Tablespaces'.

The main pane shows the 'Query' editor with the following SQL query:

```
Task: Calculate the user growth for product? */  
select year, num_week, num_active_users,  
sum(num_active_users) over(order by year, num_week rows between unbounded preceding and  
as cumm_active_users  
from  
(select  
    extract(year from a.activated_at) as year,  
    extract(week from a.activated_at)as num_week,  
    count(distinct user_id) as num_active_users  
from users a  
where state='active'  
group by year, num_week  
order by year, num_week  
)a;
```

The 'Data Output' tab shows the results of the query. The table has 6 columns: 'year', 'num\_week', 'num\_active\_users', and 'cumm\_active\_users'. The data is as follows:

	year	num_week	num_active_users	cumm_active_users
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

The status bar at the bottom indicates 'Total rows: 87 of 87' and 'Query complete 00:00:00.089'.

**Q3.** Task: Calculate the weekly retention of users-sign up cohort?

**Query:**

```
select count(user_id),
       sum(case when retention_week = 1 then 1 else 0 end) as per_week_retention
from
(
select a.user_id,
       a.sign_up_week,
       b.engagement_week,
       b.engagement_week - a.sign_up_week as retention_week
from
(
(select distinct user_id, extract(week from occurred_at) as sign_up_week
from 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 events
where event_type = 'engagement')b
on a.user_id = b.user_id
))sub
group by user_id
order by user_id;
```

pgAdmin 4

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents Processes **Investigating Metric Spike.sql**

Servers (1)

- PostgreSQL 15
  - Databases (6)
    - Operation\_Analytics**
      - Casts
      - Catalogs
      - Event Triggers
      - Extensions
      - Foreign Data Wrappers
      - Languages
      - Publications
      - Schemas
      - Subscriptions
      - dvdrental
      - exercise
      - instagram\_users
      - postgres
      - superstore
      - Login/Group Roles
      - Tablespaces

Operation\_Analytics/postgres@PostgreSQL 15

Query Query History

```

37 select count(user_id),
38       sum(case when retention_week = 1 then 1 else 0 end) as per_week_retention
39 from
40 (
41   select a.user_id,
42          a.sign_up_week,
43          b.engagement_week,
44          b.engagement_week - a.sign_up_week as retention_week
45   from
46   (
47     (select distinct user_id, extract(week from occurred_at) as sign_up_week
48     from events
49     where event_type = 'signup_flow'
50     and event_name = 'complete_signup'
51     and extract(week from occurred_at)=18)a
52     left join
53     (select distinct user_id, extract(week from occurred_at) as engagement_week
54     from events
55     where event_type = 'engagement')b
56     on a.user_id = b.user_id
57   )sub
58   group by user_id
59   order by user_id;

```

Data Output Messages Notifications

	count bigint	per_week_retention bigint
1	1	0
2	1	0
3	2	1
4	3	0
5	5	1
6	2	1
7	1	0
8	3	1
9	2	1
10	6	1
11	2	1

Total rows: 81 of 81    Query complete 00:00:00.270

#### Q4. Task: Calculate the weekly engagement per device?

##### Query:

select

```
extract(year from occurred_at) as num_years,  
extract(week from occurred_at) as num_weeks,  
device,  
count(distinct user_id) as num_of_users
```

from events

where event\_type = 'engagement'

group by 1,2,3

order by 1,2,3;

The screenshot shows the pgAdmin 4 web interface. The left sidebar displays the database structure, including 'Servers (1)' with 'PostgreSQL 15' and 'Databases (6)'. The 'Operation\_Analytics' database is selected, showing various objects like 'Casts', 'Catalogs', 'Event Triggers', etc. The main pane is in 'Query' mode, showing a SQL query for 'Weekly Engagement'. The query is as follows:

```
/* Weekly Engagement: To measure the activeness of a user.  
Task: Calculate the weekly engagement per device? */  
  
select  
    extract(year from occurred_at) as num_years,  
    extract(week from occurred_at) as num_weeks,  
    device,  
    count(distinct user_id) as num_of_users  
from events  
where event_type = 'engagement'  
group by 1,2,3  
order by 1,2,3;
```

Below the query editor, the 'Data Output' tab is active, displaying a table with 7 rows of results. The table has columns: num\_years, num\_weeks, device, and num\_of\_users. The data is as follows:

	num_years	num_weeks	device	num_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

At the bottom of the interface, it shows 'Total rows: 468 of 468' and 'Query complete 00:00:00.947'.

**Q5.** Task: Calculate the email engagement metrics?

**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 email
)a;
```

pgAdmin 4

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents Processes Investigating Metric Spike.sql\*

Operation\_Analytics/postgres@PostgreSQL 15

No limit

Query Query History

```

76 Task: Calculate the email engagement metrics? */
77 select
78 100.0 * sum(case when email_cat = 'email_opened' then 1 else 0 end)
79      /sum(case when email_cat = 'email_sent' then 1 else 0 end)
80 as email_opening_rate,
81 100.0 * sum(case when email_cat = 'email_clicked' then 1 else 0 end)
82      /sum(case when email_cat = 'email_sent' then 1 else 0 end)
83 as email_clicking_rate
84 from
85 (
86 select *,
87 case when action in ('sent_weekly_digest', 'sent_reengagement_email')
88      then 'email_sent'
89      when action in ('email_open')
90      then 'email_opened'
91      when action in ('email_clickthrough')
92      then 'email_clicked'
93 end as email_cat
94 from email
95 )a;

```

Data Output Messages Notifications

	email_opening_rate numeric	email_clicking_rate numeric
1	33.5833880499015102	14.7898883782009192

Total rows: 1 of 1 Query complete 00:00:00.105