Operation Analytics & Investigating Metric Spike

Trainity Project Report

Rohit Kumar
rohitk.ug20.ce@nitp.ac.in

Description

The study carried out for the entire end-to-end operations of a corporation is covered in this report's discussion of Operation Analytics and Investigating Metric Spike. This helps the business identify the areas where it needs to make improvements. The VP, top officials, operational team, support staff, marketing team, sales team, etc. can all benefit from this report.

Being one of the most crucial components of a business, this form of analysis is also utilized to forecast the general upward or downward trend in a company's fortune. Better automation, improved communication among cross-functional teams, and more efficient workflows are the results.

This report is based on the "Job data" dataset, which was used in the previous case study's operation analytics. Case Study 2 will use the "Yammer dataset" to investigate metric spikes and provide answers to queries about user engagement, user growth, email engagement metrics, and other topics.

Approach

This report will be divided up primarily into two sections.

1) Operation Analytics

I. **Number of jobs reviewed -** To number of jobs reviewed per hour per day for November 2020?

II. **Throughput -** will calculate the 7-day rolling average of jobs which is done.

III. **Percentage share of each language -**To find percentage share of each language in the last 30 days?

IV. **Duplicate rows -** To highlight the duplicate rows

Approach

2) Investigating Metric Spike

I. **User Engagement -** we will check this on two metrics

a) First, you get number of users are on weekly basis.

b) Second, you see how much users doing activity on weekly basis.

II. **User Growth -** To check our product growth, I'll consider two approaches

a) Growth per product (user activity) on weekly basis

b) Growth of active user on weekly basis

III. **Weekly Retention -** here I'll simply apply cohort analysis to calculate the weekly retention of users-sign up.

IV. **Weekly Engagement -** here we will see calculation of weekly engagement per device

V. **Email Engagement -** here we'll see email engagement metrics

Tech-Stack Used

- For this Project I used **PostgreSQL PG Admin 4 version 6.8**

About pgAdmin 4

Version	6.8
Application Mode	Desktop
Current User	pgadmin4@pgadmin.org
NW.js Version	0.55.0
Browser	Chromium 92.0.4515.107
Operating System	Windows-10-10.0.19045-SP0

- This project is based on the provided “Case_Study” dataset.
  [**CASE_STUDY_Dataset**]

ANALYSIS
&
RESULT

pgAdmin 4

pgAdmin File Object Tools Help

Browser

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
create table job_data
(
    ds date ,
    job_id int,
    actor_id int,
    event varchar,
    language varchar,
    time_spent int,
    org varchar
);
```

Data Output Explain Messages Notifications

```
CREATE TABLE

Query returned successfully in 132 msec.
```

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

23°C Haze ENG 13:12 15-02-2023

pgAdmin 4

pgAdmin File Object Tools Help

Browser

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
select * from job_data;
```

Data Output Explain Messages Notifications

ds date	job_id integer	actor_id integer	event character varying	language character varying	time_spent integer	org character varying

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

23°C Haze ENG 13:13 15-02-2023

pgAdmin 4

pgAdmin File Object Tools Help

Browser

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
COPY job_data (ds, job_id,actor_id,event,language,time_spent,org)
FROM 'E:\Trainity\Own\Operation Analytics & Investigating Metric Spikes\DATASET\Table-1 Job_data.csv'
DELIMITER ','
CSV HEADER;
```

Data Output Explain Messages Notifications

```
COPY 11

Query returned successfully in 92 msec.
```

Query returned successfully in 92 msec.

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

23°C Haze ENG 13:14 15-02-2023

pgAdmin 4

pgAdmin File Object Tools Help

Browser

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
select * from job_data;
```

Data Output Explain Messages Notifications

	ds date	job_id integer	actor_id integer	event character varying	language character varying	time_spent integer	org character varying
1	2020-11-30	21	1001	skip	English	15	A
2	2020-11-30	22	1006	transfer	Arabic	25	B
3	2020-11-29	23	1003	decision	Persian	20	C
4	2020-11-28	23	1005	transfer	Persian	22	D
5	2020-11-28	25	1002	decision	Hindi	11	B
6	2020-11-27	11	1007	decision	French	104	D
7	2020-11-26	23	1004	skip	Persian	56	A
8	2020-11-25	20	1003	transfer	Italian	45	C
9	[null]	[null]	[null]	[null]	[null]	[null]	[null]
10	[null]	[null]	[null]	[null]	[null]	[null]	[null]

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

23°C Haze ENG 13:14 15-02-2023

1 (I) Numbers of Job Reviewed

- Number of jobs reviewed per hour per day for November 2020

- 0.0083 jobs reviewed per hour per day in November 2020

Query:-

```
SELECT
COUNT(DISTINCT job_id)/(30 * 24.0) AS jobs_reviewed
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30';
```

pgAdmin 4

pgAdmin File Object Tools Help

Browser

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
SELECT
COUNT(DISTINCT job_id)/(30 * 24.0) AS jobs_reviewed
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30';
```

Data Output Explain Messages Notifications

	jobs_reviewed numeric
1	0.00833333333333333333

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

23°C Haze ENG 13:20 15-02-2023

1 (II) Throughput

- Calculate the 7-day rolling average of jobs which is done

Note: here I prefer 7-day rolling throughput I find there is less job done on daily basis.

ds	jobs_reviewed	throughput_7day
2020-11-25	1	1.00
2020-11-26	1	1.00
2020-11-27	1	1.00
2020-11-28	2	1.25
2020-11-29	1	1.20
2020-11-30	2	1.33

1 (II) Throughput

Query:-

```
WITH BASE AS (
SELECT ds,
COUNT(DISTINCT job_id) AS jobs_reviewed
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY ds )
SELECT
ds, jobs_reviewed,
ROUND( AVG(jobs_reviewed) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW),2) AS throughput_7day
FROM BASE;
```

pgAdmin 4

pgAdmin File Object Tools Help

Browser

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
WITH BASE AS (
SELECT ds,
COUNT(DISTINCT job_id) AS jobs_reviewed
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY ds )
SELECT
ds, jobs_reviewed,
ROUND( AVG(jobs_reviewed) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW),2) AS throughput_7day
FROM BASE;
```

Data Output Explain Messages Notifications

	ds date	jobs_reviewed bigint	throughput_7day numeric
1	2020-11-25	1	1.00
2	2020-11-26	1	1.00
3	2020-11-27	1	1.00
4	2020-11-28	2	1.25
5	2020-11-29	1	1.20
6	2020-11-30	2	1.33

✓ Successfully run. Total query runtime: 134 msec. 6 rows affected.

✓ Successfully run. Total query runtime: 113 msec. 6 rows affected.

23°C Haze ENG 15:02 15-02-2023

1 (III) Percentage share of each language

- find percentage share of each language in the last 30 days?

Language	num_jobs	total_jobs	perc_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

1 (III) Percentage share of each language

Query:-

```
WITH Base AS
( SELECT Language,
                COUNT(job_id) AS num_jobs
FROM job_data
                WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
                GROUP BY language),
total AS ( SELECT COUNT( job_id) AS total_jobs
FROM job_data
)
SELECT language, num_jobs, total_jobs, ROUND(100.0*num_jobs/total_jobs,2) AS perc_jobs
FROM Base
CROSS JOIN
total ORDER BY language
```

pgAdmin 4

File Object Tools Help

Browser

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
WITH Base AS
( SELECT Language,
    COUNT(job_id) AS num_jobs
FROM job_data
    WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
    GROUP BY language),
total AS ( SELECT COUNT( job_id) AS total_jobs
FROM job_data
)
SELECT language, num_jobs, total_jobs, ROUND(100.0*num_jobs/total_jobs,2) AS perc_jobs
FROM Base
CROSS JOIN
total ORDER BY language
```

Data Output Explain Messages Notifications

	language character varying	num_jobs bigint	total_jobs bigint	perc_jobs numeric
1	Arabic	1	8	12.50
2	English	1	8	12.50
3	French	1	8	12.50
4	Hindi	1	8	12.50
5	Italian	1	8	12.50
6	Persian	3	8	37.50

23°C Haze ENG 16:24 15-02-2023

1 (IV) Duplicate rows

- Highlight the duplicate rows:-

- Here we can highlight duplicate rows basis on their row number ('dup_flag' column) which is based on their occurrence repetition.
- All the rows have dup_flag higher than 1 are duplicate rows.

Note : Here all key column are consider for duplication entries There is no duplicate row to show present in dataset

1 (IV) Duplicate rows

Query:-

```
SELECT *
FROM (
SELECT *, ROW_NUMBER() OVER (PARTITION BY job_id, actor_id, event, language)
AS dup_flag
FROM job_data ) AS a
WHERE dup_flag > 1;
```

pgAdmin 4

pgAdmin File Object Tools Help

Browser

Dashboard Properties SQL Statistics Dependencies Dependents ops & investigation/postgres@PostgreSQL 14 *

No limit

ops & investigation/postgres@PostgreSQL 14

Query Editor Query History

```
SELECT *
FROM (
SELECT *,
ROW_NUMBER() OVER (PARTITION BY job_id, actor_id, event, language) AS dup_flag
FROM job_data
) as a
WHERE
dup_flag > 1;
```

Data Output Explain Messages Notifications

	ds date	job_id integer	actor_id integer	event character varying	language character varying	time_spent integer	org character varying	dup_flag bigint
1	[null]	[null]	[null]	[null]	[null]	[null]	[null]	2
2	[null]	[null]	[null]	[null]	[null]	[null]	[null]	3

Successfully run. Total query runtime: 128 msec. 2 rows affected.

Successfully run. Total query runtime: 99 msec. 0 rows affected.

- ig_clone
- ops & investigation
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
      - Tables
      - Trigger Functions
      - Types
      - Views
  - Subscriptions
- postgres
- Login/Group Roles

23°C Haze ENG 16:53 15-02-2023

Insights

i. This is very less amount of data to take insights from however,
ii. We can conclude there is less quantity of work for each actor i.e getting 1 or 2 job per day
iii. Dataset was clean and there is no single duplicate entry
iv. We are getting equal quantity of work base on language and have stable work load for each day.

2 (I) User Engagement

A) User engagement basis on activeness of user on weekly basis

Query:-

```
SELECT
STRFTIME('%Y',occurred_at) AS year,
STRFTIME('%W',occurred_at) AS weeknum ,
COUNT(DISTINCT user_id) AS No_of_active_users
FROM events
GROUP BY weeknum;
```

2 (I) User Engagement

B) User engagement basis on activity of user on weekly basis

Query:-

```
SELECT
STRFTIME('%Y',occurred_at) AS year,
STRFTIME('%W', occurred_at) AS weekofyear, COUNT(event_type) AS num_of_engmnt
FROM events
WHERE event_type = 'engagement'
GROUP BY weekofyear
ORDER BY weekofyear ;
```

2 (II) User Growth

A) Growth per product (user activity) on weekly basis

Query:-

```
SELECT *,
ROUND(100.0 * (LEAD(usge_per_product,1)OVER(PARTITION BY event_name, year
ORDER BY usge_per_product) - usge_per_product )/usge_per_product,2) AS '% GROWTH'
FROM(  SELECT event_name, Year,week,  usge_per_product,
SUM(usge_per_product)OVER(PARTITION BY event_name, year ORDER BY event_name, year, week ROWS BETWEEN
unbounded PRECEDING AND CURRENT ROW) AS totalusge_per_product
FROM (  SELECT *,  strftime('%W', occurred_at) AS week,  STRFTIME('%Y', occurred_at) AS Year,
COUNT(event_name) AS usge_per_product
FROM events
WHERE event_type = 'engagement'
GROUP BY event_name, week ) AS BASe )
ORDER BY event_name, year, week;
```

2 (II) User Growth

B) Growth of number of unique active users on weekly basis

Query:-

```
SELECT *,
ROUND( 100.0 * (LEAD(num_active_user,1)OVER (PARTITION BY year ORDER BY weeknum, num_active_user) -
num_active_user )/num_active_user,2)AS '% GROWTH'
FROM( SELECT year, weeknum, num_active_user,
SUM(num_active_user)OVER(ORDER BY year,weeknum
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_users
FROM ( SELECT STRFTIME('%Y', activated_at) AS year,
STRFTIME('%W', activated_at) AS weeknum,
COUNT(DISTINCT user_id) AS num_active_user
FROM users
WHERE  state='active'
GROUP BY year, weeknum
ORDER BY year, weeknum ) )
ORDER BY year, weeknum ;
```

2 (III) Weekly Retention

It is application of cohort analysis to calculate the weekly retention of users-sign up.

Note – retention outcome of this question is very large to attached here

Query:-

```
SELECT DISTINCT Year, signup_week, COUNT(user_id) Total_Users,
SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS week_1,
SUM(CASE WHEN retention_week = 2 THEN 2 ELSE 0 END) AS week_2,
SUM(CASE WHEN retention_week = 3 THEN 3 ELSE 0 END) AS week_3,
SUM(CASE WHEN retention_week = 4 THEN 4 ELSE 0 END) AS week_4,
SUM(CASE WHEN retention_week = 5 THEN 5 ELSE 0 END) AS week_5,
SUM(CASE WHEN retention_week = 6 THEN 6 ELSE 0 END) AS week_6,
SUM(CASE WHEN retention_week = 7 THEN 7 ELSE 0 END) AS week_7,
SUM(CASE WHEN retention_week = 8 THEN 8 ELSE 0 END) AS week_8,
SUM(CASE WHEN retention_week = 9 THEN 9 ELSE 0 END) AS week_9,
SUM(CASE WHEN retention_week = 10 THEN 10 ELSE 0 END) AS week_10,
SUM(CASE WHEN retention_week = 11 THEN 11 ELSE 0 END) AS week_11,
SUM(CASE WHEN retention_week = 12 THEN 12 ELSE 0 END) AS week_12,
SUM(CASE WHEN retention_week = 13 THEN 13 ELSE 0 END) AS week_13,
SUM(CASE WHEN retention_week = 14 THEN 14 ELSE 0 END) AS week_14,
SUM(CASE WHEN retention_week = 15 THEN 15 ELSE 0 END) AS week_15,
SUM(CASE WHEN retention_week = 16 THEN 16 ELSE 0 END) AS week_16,
```

2 (III) Weekly Retention

```
SUM(CASE WHEN retention_week = 17 THEN 17 ELSE 0 END) AS week_17,
SUM(CASE WHEN retention_week = 18 THEN 18 ELSE 0 END) AS week_18
FROM (  SELECT a.user_id,
a.signup_week,
a.Year,
b.login_week,
b.login_week - a.signup_week AS retention_week
FROM (
( SELECT DISTINCT user_id,
STRFTIME('%W', occurred_at) AS signup_week,
STRFTIME('%Y', occurred_at) AS Year
FROM events
WHERE  event_type = 'signup_flow' AND event_name = 'complete_signup' ) a
LEFT JOIN
( SELECT DISTINCT user_id,
STRFTIME('%W', occurred_at) AS login_week
FROM events
WHERE event_name = 'login'
ORDER BY user_id ) b
ON a.user_id = b.user_id )
ORDER BY a.user_id )
GROUP BY signup_week;
```

2 (IV) Weekly Engagement

- calculation of weekly engagement per device

Query:-

```
SELECT
STRFTIME('%Y', occurred_at) AS year,
STRFTIME('%W' , occurred_at) AS week, device,
COUNT(DISTINCT user_id) AS 'No. of users'
FROM  events
WHERE  event_type ='engagement'
GROUP BY 1,2,3
ORDER BY 3,2,1
```

2 (V) Email Engagement

- You’ll see email engagement metrics
- You you’ll absolute as well as relative email engagement metrics

Total number of sent emails	60920
Percentage of emails get view	33.58 %
Total number of open emails	20459
Percentage of overall of clicks	14.79 %
Total no. Of clicks on emails	9010
Percentage of clicks out of open mails	44.04%

2 (V) Email Engagement

Query:-

```
SELECT
SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS Total_email_sent,
SUM(CASE WHEN email_cat = 'email_open' THEN 1 ELSE 0 END)AS Total_email_open,
SUM(CASE WHEN email_cat = 'email_clicked' THEN 1 ELSE 0 END) AS Total_email_clicked,
ROUND(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), 2) AS email_open_rate,
ROUND( 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), 2) AS overall_email_clicked_rate,
ROUND ( 100.0 *SUM(CASE WHEN email_cat = 'email_clicked' THEN 1 ELSE 0 END)/SUM(CASE WHEN email_cat =
'email_open' THEN 1 ELSE 0 END), 2) AS email_clicked_rate_per_open_eamil
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 email_events )
```

Insights

I. From week 17 to 30 there is up down trend of user engagement with overall upward trend and after 30 there is downward trend in both activeness of users and activity done by user

II. On the above metric we can say there is direct relation between activeness of users and activity done by user

III. There is tremendous growth on after week 17 and then have stable up & down trend while activeness of users has up and down trend

IV. Till the week 30 user signup have positive growth afterward there is downfall while there is retention rate is good

V. most of users using our product with macbook or laptops, and there is less no. Of users with cell phone device using our product.

VI. Out of 100 sent emails there is only 33 emails get open and out of 33 emails of 14-15 emails getting encourage people for clicks, we have to improve our email standard.

Results & Conclusions

a) This project was very helpful to understand and strengthen key advance concept of sql
b) Here i learn use of windows function, lead & lag function, diff date function, case function and application of it.
c) It was helpful toward practicing of basic to intermediate sql quarries, and it was fun to write them, getting stuck and resolve error.

THANK
YOU !