SUBMISSION OF ASSIGNMENT Raaj Tilak Sarma

FILES (Attached)

- bright-money-python.html
    - File contains python code for all the 7 operations (queries) to be done over the tables
- db.sql

How the tables were created?

- In a PostGresql server hosted in PgAdmin4
- Manually some sample data was fed to check for the operations
- Few values have been left null or filled with dummy values

Input Queries:

- INSERT INTO public."Customer"(id, "Name", "Email", customer_id, address, created_time) VALUES (“john”,”john@gmail.com”,234455,”{“city”:”Dibrugarh”,”state”:”Assam”}, 2020-01-23 14:57:23.512484-05);
- INSERT INTO public."Account"(id, account_id, account_no, account_balance, created_time, user_id) VALUES (7, gtr567uhg9cvbnh7, 12345678900987, 1000.78, 1, 2020-01-22 19:57:23.512484+00)
- INSERT INTO public."Transaction"(amount, id, transaction_id, transaction_time, type, account_id) VALUES (78.28, 7, hy764k8hfh679ksn,2019-12-29 19:30:28.973662+00,’CREDIT’, 7);

Note : Whenever we input into the “transaction” table we need to UPDATE the account_balance in the “account” table as well.

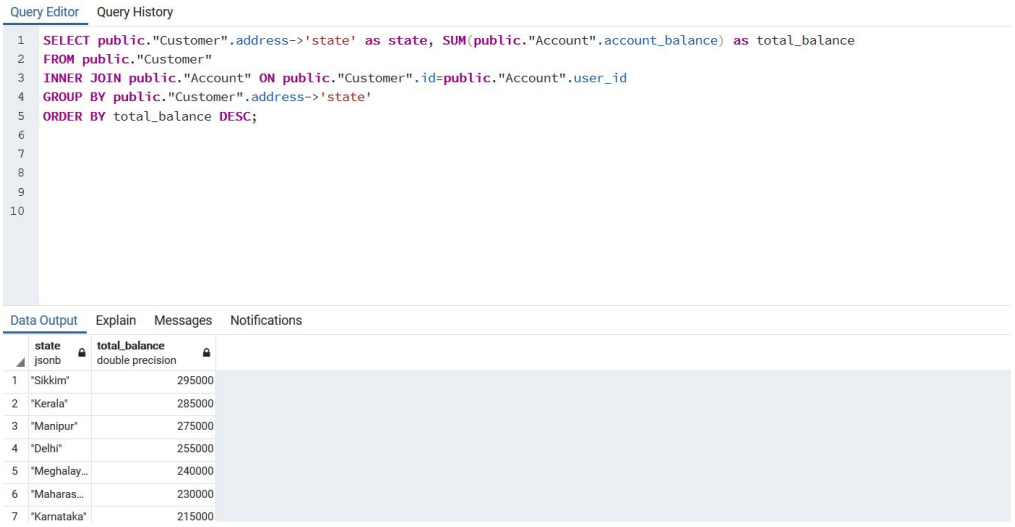
```
DECLARE @type varchar(8)
DECLARE @transaction_amount DOUBLE PRECISION
DECLATE @account_id INT
SET @type = ‘CREDIT’ or ‘DEBIT’
SET @transaction_amount = 1
SET @account_id = <account id>

IF @type = ‘CREDIT’
        UPDATE public."Account" set amount = amount + @transaction_amount where account_id==@account_id;
ELSE
    UPDATE public."Account" set amount = amount - @transaction_amount where account_id==@account_id;
```

Queries for Q1-Q3

I could not fully execute the SQL queries correctly. Started with the python code, could not find time.

Task1 : Fetch all state names and total of account balance for the users belonging to those States.

Query : “SELECT public."Customer".address->'state', SUM(public."Account".account_balance) as total_balance
FROM public."Customer"
INNER JOIN public."Account" ON public."Customer".id=public."Account".user_id
GROUP BY public."Customer".address->'state'
ORDER BY total_balance DESC;”



Task 2 :
A. For the state with the highest account balance total, get the daily total of account balances for the past 10 days based on transaction history from the ‘Transaction’ table.
Note : the optimised query is added. Please check point 4 from improvements before going into the query.
Query : SELECT * from public."DayWiseBalance" WHERE public.”DayWiseBalance”.state = (SELECT public."Customer".address->'state', SUM(public."Account".account_balance) as total_balance FROM public."Customer"
INNER JOIN public."Account" ON public."Customer".id=public."Account".user_id
GROUP BY public."Customer".address->'state'
ORDER BY total_balance DESC LIMIT 1)
ORDER BY public."DayWiseBalance".date DESC LIMIT 10;

Task 3 :
C. Repeat a. and b. for 10 richest users belonging to each of the 5 poorest states.
Query(this will provide the top 10 richest of 5 poorest states) : SELECT
public."Customer".customer_id, SUM(public."Account".account_balance) as total_balance FROM
public.”CUSTOMER”
INNER JOIN public.”Account”
ON public."Customer".id=public."Account".user_id
GROUP BY public."Customer".customer_id
WHERE public."Customer".address->'state' IN
(SELECT public."Customer".address->'state' as state FROM public."Customer"
INNER JOIN public."Account" ON public."Customer".id=public."Account".user_id
GROUP BY public."Customer".address->'state'
ORDER BY total_balance LIMIT 5)
ORDER BY total_balance DESC LIMIT 10;

Bonus Question Answer

Some possible improvements

1. “Email” may not be indexed in the table “Customer”. The idea behind this is since no query involves this column, there is no use of indexing it.
2. “account_balance” may not be indexed in table “Account”. The motive behind this is the same as above.
3. We can have “state” as a separate column in table “Customer” since it is so frequently used and seems like it does not contain null values. Another optimization that can save an inner join for task 1 is if we store the ‘State’ column in the Account table itself, denoting which state the account belongs to. The ‘State’ information mustn’t be stored in the Account table as it’s attributes belong to Customer only, but if the queries need to be optimised further, we can take this path.
4. Space and Time complement each other. So if we need to optimise the queries, we should focus on storing some relevant data. Something which will be computed every time.
   We can make another table “DayWiseBalance” with columns -

Date	timestamptz	Index
State	character varying(16)	Index
Balance	Double Precision	

   This table will be updated in the EOD of everyday and the values obtained for Task1 will be inserted. This will make the query for the balance of the last 10 days easier and we won't need to query the Transaction table at all.
   Also we can keep the data of the last 10 days in this table and delete the older ones, thereby keeping this table to hardly 350 entries.

Case Study

Q. What data would you collect to find out which states to focus on?

Answer : A pandemic like COVID-19 will encompass several sectors and therefore different kinds of data could have their own value.
The following data can be some which could be important to collect.

1. Existing hospital beds, ventilators, active quarantine centers, doctor-patient ratio
2. Food availability and sustainability (agriculture) state-wise
3. Population density
4. State-wise economy
5. State-wise facilities for people with other diseases (like cancer)
6. Test-kits per population available state-wise
7. Inter-state travel history data
8. Inter-state upcoming travel schedule data
9. Average percentage of people wearing masks and taking basic precautions
10. Age distribution of state population
11. State-wise respiratory illness data of last few years
12. Contact tracing efficiency state-wise
13. Flight/Train schedule for coming month

Q. How would you make the decision?

Answer : All the above data would have to be weighted to make a calculated decision.

1. Food availability and healthcare are priorities here and both depend on the economy. Currently, the state economy is something I would look into to classify which are struggling most financially. Also there will be states which have higher population density which adds more risk. A state which is financially the weakest and has higher population density should be prioritized to be supported economically.
2. States which are stronger economically can use the extra funds in quickly building new COVID-19 specific hospitals and if the Central government can provide help in this process.
3. An advisory to states to take special care of stranded people (not from the state). This will lead to less travel.
4. States which have more people suffering from other diseases (like Cancer) should be prioritized as death rates will spike if such people are not kept in isolation.
5. Average age district/taluka wise helps in prioritising resource distribution if there is shortage.
6. Concentrate on the agricultural chain. Helping poor farmers in states will help maintain food supply. Of course how to choose such states depends on current facilities of storage available etc. in the states.
7. Travel schedule data will help to visualize which states are going to be flooded with more people travelling.
8. Contact tracing efficiency data state-wise will help in understanding how proactive a state is at lower levels.

9. Consistently taking briefings from all states (even if a 5 minute briefing) every day so as to provide help as and when required without delay.
10. Pooling shortages and requirements of states together so they can mutually support each other if such chances arise.