SQL Case Study – 3

Problem Statement:

You are the database developer of an international bank. You are responsible for managing the bank's database. You want to use the data to answer a few questions about your customers regarding withdrawal, deposit and so on, especially about the transaction amount on a particular date across various regions of the world. Perform SQL queries to get the key insights of a customer.

Dataset:

The 3 key datasets for this case study are:

a. **Continent:** The Continent table has two attributes i.e., region_id and region_name, where region_name consists of different continents such as Asia, Europe, Africa etc., assigned with the unique region id.

b. **Customers:** The Customers table has four attributes named customer_id, region_id, start_date and end_date which consists of 3500 records.

c. **Transaction:** Finally, the Transaction table contains around 5850 records and has four attributes named customer_id, txn_date, txn_type and txn_amount.

```sql
use case_study

select * from dbo.continent
select * from Customers
select * from dbo.trans
```

```sql
--1. Display the count of customers in each region who have done the
transaction in the year 2020.
SELECT c.region_id, r.region_name, COUNT(DISTINCT c.customer_id) AS
customer_count
FROM Customers c
JOIN Continent r
ON c.region_id = r.region_id
JOIN trans t
ON c.customer_id = t.customer_id
WHERE YEAR(t.txn_date) = 2020
GROUP BY c.region_id, r.region_name
order by customer_count desc

--2. Display the maximum and minimum transaction amount of
eachtransaction type.
select * from dbo.continent
select * from Customers
select * from trans

SELECT txn_type, MAX(txn_amount) AS max_amount, MIN(txn_amount) AS
min_amount
FROM Trans
GROUP BY txn_type;

--3. Display the customer id, region name and transaction amount where
transaction type is deposit and transaction amount > 2000.

SELECT c.customer_id, r.region_name, t.txn_amount
FROM Customers c
JOIN Continent r
ON c.region_id = r.region_id
JOIN Trans t
ON c.customer_id = t.customer_id
WHERE t.txn_type = 'deposit' AND t.txn_amount > 2000;
```

```sql
--4. Find duplicate records in the Customer table.
SELECT    c1.customer_id, c1.region_id, c1.start_date, c1.end_date
FROM Customers c1
INNER JOIN Customers c2
   ON c1.customer_id = c2.customer_id AND c1.region_id = c2.region_id AND
c1.start_date = c2.start_date
   AND c1.end_date = c2.end_date AND c1.customer_id <> c2.customer_id
        -- to avoid comparing the same record
GROUP BY c1.customer_id, c1.region_id, c1.start_date, c1.end_date
HAVING COUNT(*) > 1;

--5. Display the customer id, region name, transaction type and transaction
amount for the minimum transaction amount in deposit.

select * from dbo.continent
select * from Customers
select * from trans


SELECT c.customer_id, r.region_name, t.txn_type, t.txn_amount
FROM Customers c
JOIN Continent r
ON c.region_id = r.region_id
JOIN Trans t
ON c.customer_id = t.customer_id
WHERE t.txn_type = 'deposit'
   AND t.txn_amount = ( SELECT MIN(txn_amount) FROM Trans WHERE
txn_type = 'deposit' );


--6. create a stored procedure to display details of customers in the
Transaction table where the transaction date is greater than Jun 2020.


CREATE PROCEDURE GetCustomersAfterJune2020
AS
BEGIN
   SELECT c.customer_id, r.region_name, t.txn_date, t.txn_type, t.txn_amount
   FROM Customers c
   INNER JOIN Continent r
```

```sql
        ON c.region_id = r.region_id
    INNER JOIN Trans t
        ON c.customer_id = t.customer_id
    WHERE t.txn_date > '2020-06-30';
END
GO

EXEC GetCustomersAfterJune2020;

--7. Create a stored procedure to insert a record in the Continent table.
select * from dbo.continent
select * from Customers
select * from trans


CREATE PROCEDURE InsertContinent
    @region_id INT, @region_name VARCHAR(50)
AS
BEGIN
    INSERT INTO Continent (region_id, region_name)
    VALUES (@region_id, @region_name)
END
GO

EXEC InsertContinent @region_id = 7, @region_name = 'Antarctica'

select * from dbo.continent

--8. Create a stored procedure to display the details of transactions that
happened on a specific day.


CREATE PROCEDURE GetTransactionsByDate
    @TxnDate DATE
AS
BEGIN
    SELECT
        c.customer_id, r.region_name, t.txn_date, t.txn_type, t.txn_amount
    FROM Customers c
    INNER JOIN Continent r
```

```
        ON c.region_id = r.region_id
    INNER JOIN Trans t
        ON c.customer_id = t.customer_id
    WHERE t.txn_date = @TxnDate;
END
GO

EXEC GetTransactionsByDate @TxnDate = '2022-05-15'

--9. Create a user defined function to add 10% of the transaction amount in a
table.

CREATE FUNCTION AddTenPercent (@txn_amount DECIMAL(10,2))
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @result DECIMAL(10,2)
    SET @result = @txn_amount + (@txn_amount * 0.1)
    RETURN @result
END

--call the function

SELECT customer_id, txn_date, txn_type, txn_amount,
dbo.AddTenPercent(txn_amount) AS txn_amount_with_10_percent
FROM Trans

--10. Create a user defined function to find the total transaction amount for a
given transaction type.


CREATE FUNCTION GetTotalTransactionAmount
( @txn_type VARCHAR(50) )
RETURNS DECIMAL(18,2)
AS
BEGIN
    DECLARE @total_amount DECIMAL(18,2)

    SELECT @total_amount = SUM(txn_amount)
    FROM Trans
```

```sql
    WHERE txn_type = @txn_type

    RETURN ISNULL(@total_amount, 0)
END
GO


SELECT dbo.GetTotalTransactionAmount('deposit') AS total_deposit_amount;

--11. Create a table value function which comprises the columns
customer_id,region_id ,txn_date , txn_type , txn_amount which will retrieve
data from the above table.

CREATE FUNCTION GetCustomerTransactionDetails()
RETURNS @CustomerTransactionDetails TABLE
( customer_id INT, region_id INT, region_name VARCHAR(50), txn_date DATE,
txn_type VARCHAR(50), txn_amount DECIMAL(10,2) )
AS
BEGIN
    INSERT INTO @CustomerTransactionDetails
    SELECT c.customer_id, c.region_id, r.region_name, t.txn_date, t.txn_type,
t.txn_amount
    FROM Customers c
    INNER JOIN Continent r
        ON c.region_id = r.region_id
    INNER JOIN Trans t
        ON c.customer_id = t.customer_id;
RETURN;
END
GO

SELECT * FROM dbo.GetCustomerTransactionDetails();

--12. Create a TRY...CATCH block to print a region id and region name in a
single column.

BEGIN TRY
    SELECT CONCAT(region_id, ' --- ', region_name) AS 'Region ID - Region Name'
    FROM Continent;
END TRY
```

```sql
BEGIN CATCH
   PRINT 'An error occurred: ' + ERROR_MESSAGE();
END CATCH

--13. Create a TRY...CATCH block to insert a value in the Continent table.

BEGIN TRY
   INSERT INTO Continent (region_id, region_name)
   VALUES (8, 'Antarctica');
   PRINT 'Record inserted successfully.';
END TRY
BEGIN CATCH
   PRINT 'Error occurred: ' + ERROR_MESSAGE();
END CATCH


--14. Create a trigger to prevent deleting a table in a database.

CREATE TRIGGER tr_PreventDeleteTable
ON DATABASE
FOR DROP_TABLE
AS
BEGIN
   PRINT 'Deleting tables is not allowed in this database.'
   ROLLBACK TRANSACTION
END
GO

DROP TABLE Trans; -- This will be prevented by the trigger
select* from trans

drop trigger tr_PreventDeleteTable

--15. Create a trigger to audit the data in a table.

create table customer_audit(id int identity(1,1), AuditData varchar(50))

select * from customer_audit

CREATE TRIGGER  trg_audit
```

```
ON customers
FOR INSERT
as begin
Declare @id int
select @id = customer_id from inserted
insert into customer_audit values
('New customer with ID = ' + cast(@id as varchar(5)) + ' ' +'is added at')
end

insert into Customers values(2003,2,'2001-11-14','2001-11-14')

select * from customer_audit
--select * from Customers

--drop trigger trg_audit
--drop table customer_audit


--16. Create a trigger to prevent login of the same user id in multiple pages.

CREATE TABLE ActiveSessions (
   user_id INT PRIMARY KEY,
   session_id VARCHAR(50) NOT NULL,
   login_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP );

         --select * from ActiveSessions
         CREATE TRIGGER prevent_multiple_logins_trigger
ON ActiveSessions
INSTEAD OF INSERT
AS BEGIN
   SET NOCOUNT ON;
   IF EXISTS ( SELECT 1 FROM ActiveSessions
      WHERE user_id = (SELECT user_id FROM inserted) )
   BEGIN
      RAISERROR ('User is already logged in on another session.', 16, 1);
      ROLLBACK TRANSACTION;
   END
   ELSE
   BEGIN
      INSERT INTO ActiveSessions
```

```
        SELECT user_id, session_id, login_time
        FROM inserted;
    END
END
GO

-- add input
INSERT INTO ActiveSessions (user_id, session_id)
VALUES (1, 'session123');

INSERT INTO ActiveSessions (user_id, session_id)    -- value not added
VALUES (1, 'session456');

SELECT * FROM ActiveSessions;



--17a. Display top n customers on the basis of transaction type.

SELECT top 4 c.customer_id, COUNT(t.txn_type) AS total_transactions,
t.txn_type
FROM Customers c
JOIN trans t
ON c.customer_id = t.customer_id
GROUP BY c.customer_id, t.txn_type
ORDER BY total_transactions DESC

--b. Display top n customers on the basis of each transaction type.

WITH cte AS (             -- common table expression (CTE)
    SELECT customer_id, txn_type, SUM(txn_amount) AS total_amount,
        RANK() OVER (PARTITION BY txn_type ORDER BY SUM(txn_amount) DESC)
AS rnk
    FROM Trans
    GROUP BY  customer_id, txn_type )

SELECT customer_id, txn_type, total_amount
FROM cte
WHERE rnk <= COALESCE(4, 5) -- Replace @n with the desired value of n or use
5 as the default
```

```
ORDER BY txn_type, total_amount DESC;

select * from trans
```