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.

--1. Display the count of customers in each region who have done the transaction in the year 2020.

```
SELECT
COUNT(CS.customer_id) AS 'COUNT OF CUSTOMER',
region_name FROM [Customers] CS
INNER JOIN  [Transaction] TR
ON CS.customer_id = TR.customer_id
INNER JOIN [Continent] CT
ON CS.region_id = CT.region_id
WHERE YEAR(txn_date) = 2020
GROUP BY region_name
```

```
--WITH CTE AS
--(
--SELECT customer_id FROM [Transaction] WHERE
--customer_id IN (SELECT customer_id FROM Customers
```

```
--WHERE REGION_ID = 2)
--)
--SELECT COUNT(*) AS TOTAL FROM CTE;

--DECLARE
--@i INT = 1;
--WHILE (@i <= 5)
--BEGIN
--WITH CTE AS
--(
--SELECT customer_id FROM [Transaction] WHERE
--customer_id IN (SELECT customer_id FROM Customers
--WHERE REGION_ID = @i) and year (txn_date) = 2020
--)
--SELECT COUNT(*) AS TOTAL FROM CTE
----PRINT' ITERATION' ;
--SET @i = @i+1;
--END;

--CREATE PROCEDURE TOTALTRANSACTION1 @TOTALTXN INT AS

--WITH CTE AS
--(
--SELECT customer_id FROM [Transaction] WHERE
--customer_id IN (SELECT customer_id FROM Customers
--WHERE REGION_ID = @TOTALTXN) and year (TXN_DATE) = 2020
--)

--SELECT COUNT(*) AS TOTAL FROM CTE

--EXEC TOTALTRANSACTION1 @TOTALTXN = 2;
```

```
--CREATE PROCEDURE TOTALTRANSACTIONS @TOTALTXNS VARCHAR(20) AS
--WITH CTE AS
--(
--SELECT customer_id FROM [Transaction] WHERE
--customer_id IN (SELECT customer_id FROM Customers
--WHERE REGION_ID = (SELECT REGION_ID FROM Continent
--WHERE REGION_NAME = @TOTALTXNS)) and year (txn _date) = 2020
--)
--SELECT COUNT(*) AS TOTAL FROM CTE

--EXEC TOTALTRANSACTIONS @TOTALTXNS = 'AUSTRALIA';

--SELECT
--COUNT (*) AS TOTAL,
--CU. REGION_ID
--FROM Customers CU INNER JOIN [Transaction] T ON T.customer_id = CU.customer_id
--AND(T.txn_date BETWEEN CU.start_date AND CU.end_date)
--WHERE YEAR(T. TXN_DATE) = 2020
--GROUP BY region_id
--ORDER BY region_id;

--SELECT
--*
--FROM Customers CU INNER JOIN [Transaction] T ON T.customer_id = CU.customer_id
--AND(T.txn_date BETWEEN CU.start_date AND CU.end_date)
--WHERE T. customer_id = 1
--SELECT
--*
--FROM Customers CU INNER JOIN [Transaction] T ON T.customer_id = CU.customer_id
--WHERE T.customer_id = 1
```

--2. Display the maximum and minimum transaction amount of each transaction type.

```
SELECT
TXN_TYPE,
MAX(TXN_AMOUNT) AS MAX,
MIN(TXN_AMOUNT) AS MIN
FROM [Transaction]
GROUP BY TXN_TYPE
```

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

```
SELECT
DISTINCT(CS.customer_id),
region_name,
txn_type,
txn_amount
FROM [Customers] CS
INNER JOIN  [Transaction] TR
ON CS.customer_id = TR.customer_id
INNER JOIN [Continent] CT
ON CS.region_id = CT.region_id
WHERE txn_type = 'DEPOSIT' AND txn_amount >2000
ORDER BY CS.customer_id
```

```
--SELECT CU.CUSTOMER_ID,CO.REGION_NAME,T.TXN_AMOUNT
--FROM Customers CU INNER JOIN Continent CO
--ON CO.region_id = CU.region_id INNER JOIN [Transaction] T
```

```
--ON T.customer_id = CU.customer_id AND(T.txn_date BETWEEN CU.start_date
--AND CU.end_date)WHERE T.TXN_TYPE = 'DEPOSIT' AND T.txn_amount > 2000
```

--4. Find duplicate records in the Customer table.

```
SELECT START_DATE FROM Customers
GROUP BY START_DATE
HAVING COUNT(*) > 1;
```

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

```
SELECT
DISTINCT(CS.customer_id),
region_name,
txn_type,
MIN(txn_amount) AS 'MINIMUM AMOUNT DEPOSIT'
FROM [Customers] CS
INNER JOIN  [Transaction] TR
ON CS.customer_id = TR.customer_id
INNER JOIN [Continent] CT
ON CS.region_id = CT.region_id
WHERE  txn_type = 'DEPOSIT'
GROUP BY region_name,CS.customer_id,txn_type
ORDER BY CS.customer_id

-- WRONG AS PER REQUIRMENT
--SELECT
--CU.CUSTOMER_ID,
--CU.REGION_ID,
--CU.START_DATE,
```

```
--CU.END_DATE,

--CO.REGION_NAME,

--T.TXN_AMOUNT

--FROM Customers CU

--INNER JOIN Continent CO

--ON CO.region_id = CU.region_id

--INNER JOIN [Transaction] T

--ON T.customer_id = CU.customer_id AND (T.txn_date BETWEEN CU.start_date AND CU.end_date)

--WHERE T.TXN_AMOUNT IN (SELECT MIN(TXN_AMOUNT) FROM [Transaction])
```

--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 DETAILS @MONTH_TXN_DATE INT, @YEAR_TXN INT

AS

SELECT * FROM Customers C

INNER JOIN [Transaction] T

ON C.customer_id = T.customer_id

WHERE MONTH(txn_date) > @MONTH_TXN_DATE AND YEAR(txn_date) > @YEAR_TXN


EXEC DETAILS @MONTH_TXN_DATE = 1,@YEAR_TXN = 2016;


   -- BOTH QUERY RIGHT QUERY

--CREATE PROCEDURE ALLDETAILS @MONTHOFTXNDATE INT, @YEAROFTXNDATE INT

--AS

--SELECT * FROM Customers CU

--INNER JOIN [Transaction] T

--ON T.customer_id = CU.customer_id
```

```
--AND (T.txn_date BETWEEN CU.start_date AND CU.end_date)

--WHERE MONTH(TXN_DATE) > @MONTHOFTXNDATE AND YEAR(TXN_DATE) >= @YEAROFTXNDATE;


-- EXECUTION PROCESS

--EXEC ALLDETAILS @MONTHOFTXNDATE = 1, @YEAROFTXNDATE = 2019



--7. Create a stored procedure to insert a record in the Continent table.


CREATE PROCEDURE INSERT_DATA_IN_CONTINENT

@REGION_ID INT ,@REGION_NAME VARCHAR(20)

AS

INSERT INTO Continent VALUES(@REGION_ID,@REGION_NAME);


-- EXECUTION PROCESS

Exec INSERT_DATA_IN_CONTINENT @REGION_ID = 6 ,@REGION_NAME = 'Russia'



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


   ALTER PROCEDURE DATE_WISE_DETAILS @YEAR INT, @MONTH INT, @DAY INT

AS

SELECT * FROM [Transaction]

WHERE txn_date = (SELECT DATEFROMPARTS(@YEAR, @MONTH, @DAY) )


-- FORMAT SHOULD BE SAME AS PER DATE DON'T FORGET

EXEC DATE_WISE_DETAILS @YEAR = 2020, @MONTH = 4 , @DAY = 20



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

```
CREATE FUNCTION INCREMENT (@ADD INT)
RETURNS TABLE
AS RETURN
(
SELECT
(txn_amount + ((@ADD) * txn_amount)) AS AMOUNT
FROM [Transaction]
)
SELECT * FROM
INCREMENT(1);
SELECT * FROM [Transaction]
```

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

```
CREATE FUNCTION TOTAL_AMOUNT_BY_TYPE(@TRANSACTION_TYPE VARCHAR(10))
RETURNS TABLE
AS
RETURN
(
SELECT
TXN_TYPE,
SUM(txn_amount) AS 'TOTAL AMOUNT'
FROM [Transaction]
WHERE txn_type = @TRANSACTION_TYPE
GROUP BY txn_type
)

SELECT * FROM TOTAL_AMOUNT_BY_TYPE('DEPOSIT')
```

--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.

```
ALTER FUNCTION TRANSACTION_DETAILS()
RETURNS TABLE
AS
RETURN
(
SELECT
DISTINCT(T.txn_date),
C.customer_id,
C.region_id,
txn_amount
txn_type
FROM Customers C
INNER JOIN [Transaction] T
ON C.customer_id = T.customer_id
)

--CREATE FUNCTION TDETAILS()
--RETURNS TABLE
--AS
--RETURN
--(SELECT
--CU.CUSTOMER_ID,
--CU.REGION_ID,
--T.TXN_DATE,
--T.TXN_TYPE,
--T.TXN_AMOUNT
--FROM Customers CU INNER JOIN [Transaction] T ON T.customer_id = CU.customer_id
--AND (T.txn_date BETWEEN CU.start_date AND CU.end_date))
```

```
SELECT * FROM TDETAILS()
SELECT * FROM TRANSACTION_DETAILS()


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

   BEGIN TRY
SELECT REGION_ID+"+ REGION_NAME AS COMBINED_COLUMN FROM Continent
END TRY
BEGIN CATCH
SELECT ERROR_MESSAGE() AS ERROR
END CATCH;

SELECT * FROM Customers
SELECT * FROM Continent
SELECT * FROM [Transaction]

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

   BEGIN TRY
INSERT INTO Continent VALUES
(7, 'INDIA')
END TRY
BEGIN CATCH
SELECT ERROR_MESSAGE() AS ERROR
END CATCH;

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

CREATE TRIGGER TRG_DELETE
ON CONTINENT
FOR DELETE
```

```
AS
BEGIN
ROLLBACK
PRINT '*****************************************'
PRINT 'YOU CANNOT DELETE FROM THIS TABLE'
PRINT '*****************************************'
END

DELETE FROM Continent
WHERE region_id = 6

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

 SELECT * FROM Continent;
 CREATE TABLE CONTINENT_AUDIT
 (
 REGION_ID INT,
 REGION_NAME VARCHAR(20),
 INSERTED_BY VARCHAR(50)
 );
 CREATE TRIGGER TRG_CONTINET
 ON CONTINENT
 FOR INSERT, UPDATE, DELETE
 AS
 BEGIN
 DECLARE @ID INT, @NAME VARCHAR(20)
 SELECT @ID = REGION_ID, @NAME = REGION_NAME FROM inserted
 INSERT INTO CONTINENT_AUDIT (REGION_ID, REGION_NAME, INSERTED_BY)
 VALUES (@ID, @NAME, ORIGINAL_LOGIN())
 PRINT 'INSERT TRIGGER EXECUTED'
 END;
 SELECT * FROM CONTINENT_AUDIT;
```

```
INSERT INTO CONTINENT VALUES(6, 'RUSSIA');
DELETE FROM Continent
WHERE REGION_ID = 6;
UPDATE CONTINENT
SET REGION_NAME = 'INDIA'
WHERE region_id = 6;
ENABLE TRIGGER TRG_DELETE ON CONTINENT;
```

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

```
CREATE TRIGGER PREVENT_MULTIPLE_LOGINS
ON ALL SERVER
FOR LOGON
AS
BEGIN
DECLARE @SESSION_COUNT INT
SELECT @SESSION_COUNT = COUNT(*)
FROM SYS.DM_EXEC_SESSIONS
WHERE is_user_process = 1
AND LOGIN_NAME = ORIGINAL_LOGIN()
IF @SESSION_COUNT > 1
BEGIN
PRINT 'MULTIPLE LOGINS NOT ALLOWED'
ROLLBACK
END
END;
DISABLE TRIGGER PREVENT_MULTIPLE_LOGINS ON ALL SERVER;
```

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

```
SELECT TOP 10000 * FROM [Transaction]
WHERE TXN_TYPE = 'DEPOSIT'
```

```
ORDER BY TXN_AMOUNT DESC;
--18. Create a pivot table to display the total purchase, withdrawal and deposit for all the customers.

SELECT * FROM
(
SELECT CUSTOMER_ID, TXN_TYPE, TXN_AMOUNT FROM [Transaction]) AS T
PIVOT
(
SUM(TXN_AMOUNT)
FOR TXN_TYPE IN (PURCHASE, DEPOSIT, WITHDRAWAL)
) AS P

SELECT * FROM Customers
SELECT * FROM Continent
SELECT * FROM [Transaction]
```