

November 8, 2022

-- TABLES

DROP TABLE IF EXISTS Bank;

DROP TABLE IF EXISTS Loans;

DROP TABLE IF EXISTS Accounts;

DROP TABLE IF EXISTS Customers;

DROP TABLE IF EXISTS Cards;

DROP TABLE IF EXISTS Services;

DROP TABLE IF EXISTS Transactions;

CREATE TABLE Bank (

id INT IDENTITY(1,1) PRIMARY KEY,

bank_name VARCHAR(255),

address VARCHAR(255),

);

CREATE TABLE Loans (

id INT IDENTITY(1,1) PRIMARY KEY,

amount DECIMAL(8,2),

bank_id INT FOREIGN KEY (bank_id) REFERENCES Bank (id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE Accounts (

id INT IDENTITY(1,1) PRIMARY KEY,

acc_no VARCHAR(255),

acc_type VARCHAR(255),

```
balance SMALLMONEY,  
bank_id INT FOREIGN KEY (bank_id) REFERENCES Bank (id)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION  
);
```

```
CREATE TABLE Customers (  
    id INT IDENTITY(1,1) PRIMARY KEY,  
    first_name VARCHAR(255),  
    last_name VARCHAR(255),  
    phone_no VARCHAR(255),  
    email_addr VARCHAR(255),  
    address VARCHAR(255),  
    acc_id INT FOREIGN KEY (acc_id) REFERENCES Accounts (id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION,  
    loan_id INT FOREIGN KEY (loan_id) REFERENCES Loans (id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION,  
    join_dt DATE  
);
```

```
CREATE TABLE Cards (  
    id INT IDENTITY(1,1) PRIMARY KEY,  
    card_type VARCHAR(255) DEFAULT 'DEBIT',  
    acc_id INT FOREIGN KEY (acc_id) REFERENCES Accounts (id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
);
```

```
CREATE TABLE Services (  
    id INT IDENTITY(1,1) PRIMARY KEY,  
    serv_type VARCHAR(255),  
    status VARCHAR(255),  
);
```

```
CREATE TABLE Transactions (  
    id INT IDENTITY(1,1) PRIMARY KEY,  
    tran_dt DATE,  
    tran_amt SMALLMONEY,  
    cust_id INT FOREIGN KEY (cust_id) REFERENCES Customers (id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION,  
    acc_id INT FOREIGN KEY (acc_id) REFERENCES Accounts (id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION,  
    loan_id INT FOREIGN KEY (loan_id) REFERENCES Loans (id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
);
```

```
-- INSERT INTO TABLES
```

```
INSERT INTO
```

```
    Bank (bank_name, address)
```

```
VALUES
```

```
    ('NCB', 'Kingston, Jamaica');
```

```
INSERT INTO
```

```
    Loans ( amount, bank_id)
```

```
VALUES
```

```
    (12000, 1),
```

```
    (1000, 1);
```

```
INSERT INTO
```

```
    Accounts (acc_no, acc_type, balance, bank_id)
```

```
VALUES
```

```
    ('acc-001', 'SAVINGS', 1000, 1),
```

```
    ('acc-002', 'SAVINGS', 300, 1),
```

```
    ('acc-003', 'CURRENT', 1000, 1);
```

```
INSERT INTO
```

```
    Customers (first_name, last_name, phone_no, email_addr, address, acc_id, loan_id, join_dt)
```

```
VALUES
```

```
    ('ezra', 'muir', '(876) 356-7600', 'ezra@gmail.com', 'abc avenue', 1, 1, '2022-01-28'),
```

```
    ('trizzel', 'white', '(876) 000-7600', 'name@email.com', 'xyz harbour', 2, 2, '2009-03-20'),
```

```
    ('jada', 'kingdom', '(876) 344-7000', 'jada@gmail.com', 'abc avenue', 3, 2, '2010-05-24');
```

```
INSERT INTO
```

```
    Cards (card_type, acc_id)
```

```
VALUES
```

```
('DEBIT', 3),  
( 'DEBIT', 1),  
( 'DEBIT', 2);
```

INSERT INTO

Services (serv_type, status)

VALUES

```
('account', 'ACTIVE'),  
( 'loan', 'CLOSED'),  
( 'card', 'INACTIVE');
```

INSERT INTO

Transactions (tran_dt, tran_amt, cust_id, acc_id, loan_id)

VALUES

```
('2021-12-31', 100001, 1, 1, NULL),  
( '2022-02-23', 6000, 1, 1, NULL),  
( '2015-06-19', 700, 1, 1, NULL),  
( '2022-08-31', 90, 2, 2, 2),  
( '2020-10-30', 300, 3, 3, 2),  
( '2015-06-25', 200, 3, 3, 2);
```

-- QUERIES -- -----

SELECT * FROM Bank;

SELECT * FROM Loans;

SELECT * FROM Accounts;

SELECT * FROM Customers;

SELECT * FROM Cards;

SELECT * FROM Services;

SELECT * FROM Transactions;

-- Question 5 - Generate a SQL query for eligible customers for the LOAN

DECLARE @month_gap DATE = DATEADD(month, -24, GETDATE())

SELECT

*

FROM

Customers c

INNER JOIN Loans l

ON c.loan_id = l.id

WHERE

c.join_dt < @month_gap

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
145 -- Question 5 - Generate a SQL query for eligible customers for the LOAN
146 DECLARE @month_gap DATE = DATEADD(month, -24, GETDATE())
147 SELECT
148 *
149 FROM
150 Customers c
151 INNER JOIN Loans l
152 ON c.loan_id = l.id
153 WHERE
154 c.join_dt < @month_gap
155
```

The Results pane shows the following data:

id	first_name	last_name	phone_no	email_addr	address	acc_id	loan_id	join_dt	id	amount	bank_id
1	bradley	white	(876) 600-7600	name@email.com	xyz harbour	2	2	2009-03-20	2	1000.00	1
2	jade	kingdom	(876) 344-7000	jade@gmail.com	abc avenue	3	2	2010-05-24	2	1000.00	1

The status bar at the bottom indicates: Query executed successfully. DESKTOP-VOT1DO6\MSSQLSERVER... DESKTOP-VOT1DO6\Ezra M... BankDB 00:00:00 2 rows

-- Question 6 - Generate a SQL query for eligible customers for the Credit card service

DECLARE @year_gap DATE = DATEADD(MONTH, -12, GETDATE())

DECLARE @current_dt DATE = CAST(GETDATE() AS DATE)

DECLARE @card_grant INT = 100000

SELECT

*

FROM

Customers c

INNER JOIN Cards cs

ON c.acc_id = cs.acc_id

INNER JOIN Transactions t

ON t.acc_id = cs.acc_id

WHERE

t.tran_dt > @year_gap AND t.tran_dt <= @current_dt AND t.tran_amt > @card_grant;

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'BankDB' database selected. The right pane shows a SQL query window with the following code:

```
-- Question 6 - Generate a SQL query for eligible customers for the Credit card service
DECLARE @year_gap DATE = DATEADD(MONTH, -12, GETDATE())
DECLARE @current_dt DATE = CAST(GETDATE() AS DATE)
DECLARE @card_grant INT = 100000
SELECT
*
FROM
Customers c
INNER JOIN Cards cs
ON c.acc_id = cs.acc_id
INNER JOIN Transactions t
ON t.acc_id = cs.acc_id
WHERE
t.tran_dt > @year_gap AND t.tran_dt <= @current_dt AND t.tran_amt > @card_grant;
```

Below the query window, the 'Results' pane shows a single row of data:

id	first_name	last_name	phone_no	email_addr	address	acc_id	loan_id	prin_dt	id	card_type	acc_id	id	tran_dt	tran_amt	cust_id	acc_id	loan_id
1	Ezra	Muir	(876) 356-7800	ezra@gmail.com	abc avenue	1	1	2022-01-28	2	DEBIT	1	2	2021-12-31	100001.00	1	1	NULL

The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

-- Question 7 - Generate a SQL query for customer who doesn't do transaction for the past 6 months.

```
DECLARE @month_gap2 DATE = DATEADD(MONTH, -6, GETDATE())
```

```
SELECT
```

```
    c.id,  
    t.cust_id 'Customer ID',  
    c.loan_id,  
    c.first_name,  
    c.last_name,  
    c.join_dt,  
    t.id,  
    t.tran_dt
```

```
FROM
```

```
    Customers c
```

```
    INNER JOIN Transactions t
```

```
    ON c.id = t.cust_id
```

```
WHERE
```

```
    t.tran_dt < @month_gap2
```

```
ORDER BY c.id, t.tran_dt;
```

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'BankDB' database selected. The right pane shows a SQL query window with the following code:

```
-- Question 7 - Generate a SQL query for customer who doesn't do transaction for the past 6 months  
DECLARE @month_gap2 DATE = DATEADD(MONTH, -6, GETDATE())  
SELECT  
    c.id,  
    t.cust_id 'Customer ID',  
    c.loan_id,  
    c.first_name,  
    c.last_name,  
    c.join_dt,  
    t.id,  
    t.tran_dt  
FROM  
    Customers c  
    INNER JOIN Transactions t  
    ON c.id = t.cust_id  
WHERE  
    t.tran_dt < @month_gap2  
ORDER BY c.id, t.tran_dt;
```

Below the query window, the 'Results' pane shows the output of the query. The results are displayed in a table with 5 rows and 8 columns. The columns are: id, Customer ID, loan_id, first_name, last_name, join_dt, id, and tran_dt. The data is as follows:

id	Customer ID	loan_id	first_name	last_name	join_dt	id	tran_dt
1	1	1	ezra	muir	2022-01-28	4	2015-08-19
2	1	1	ezra	muir	2022-01-28	2	2021-12-31
3	1	1	ezra	muir	2022-01-28	3	2022-02-28
4	3	3	jade	kingdom	2010-05-24	7	2015-06-25
5	3	3	jade	kingdom	2010-05-24	6	2020-10-30

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-VOT1D06\MSSQLSERVER... | BankDB | 00:00:00 | 5 rows'.

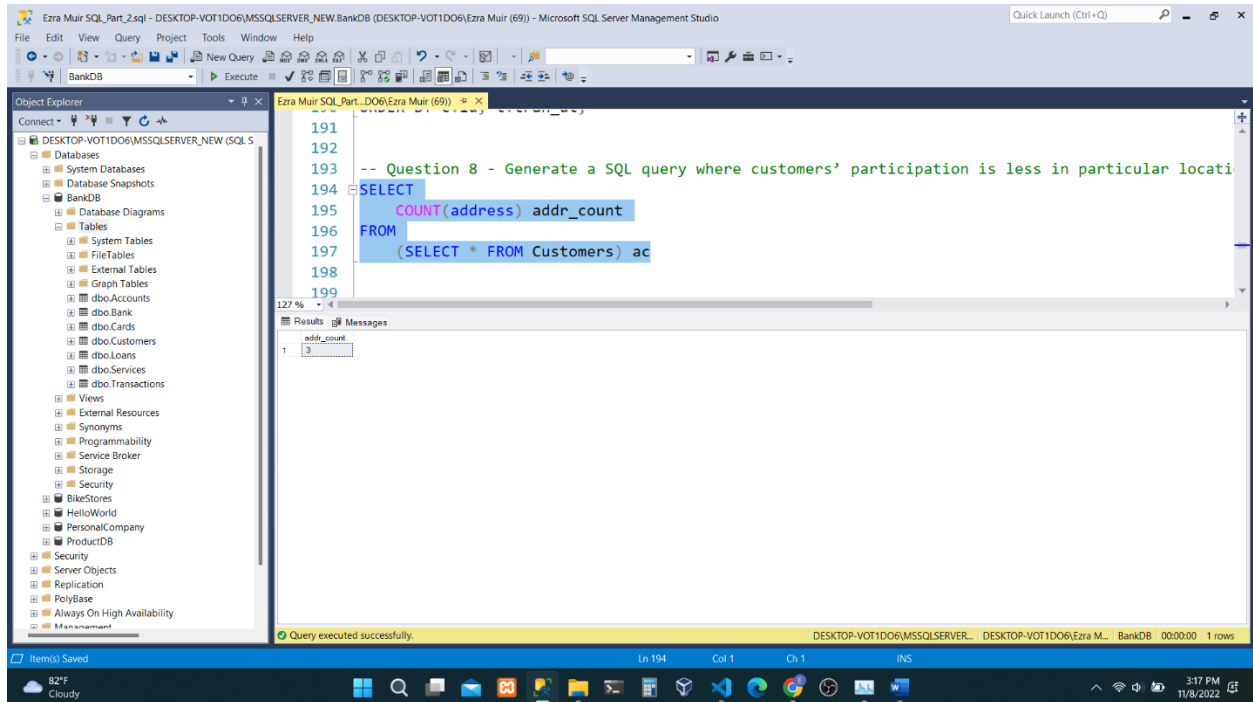
-- Question 8 - Generate a SQL query where customers' participation is less in particular location.

SELECT

COUNT(address) addr_count

FROM

(SELECT * FROM Customers) ac



-- Question 9 - Generate a SQL query where customer and loan and outstanding amount.

SELECT

c.first_name + ' ' + c.last_name AS 'Customer Name',

t.tran_dt,

'Outstanding Balance' = l.amount - sum(t.tran_amt)

OVER (

PARTITION BY a.acc_no

ORDER BY a.acc_no rows

BETWEEN UNBOUNDED PRECEDING AND CURRENT row

)

FROM

Transactions t

INNER JOIN Accounts a

ON t.acc_id = a.id

INNER JOIN Loans l

ON l.id = t.loan_id

INNER JOIN Customers c

ON c.id = t.cust_id

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'BankDB' database selected. The right pane shows a SQL query window with the following code:

```
-- Question 9 - Generate a SQL query where customer and loan and outstanding amount.
SELECT
    c.first_name + ' ' + c.last_name AS 'Customer Name',
    t.tran_dt,
    'Outstanding Balance' = l.amount - sum(t.tran_amt)
    OVER (
        PARTITION BY a.acc_no
        ORDER BY a.acc_no rows
        BETWEEN UNBOUNDED PRECEDING AND CURRENT row
    )
FROM
    Transactions t
    INNER JOIN Accounts a
        ON t.acc_id = a.id
    INNER JOIN Loans l
        ON l.id = t.loan_id
    INNER JOIN Customers c
        ON c.id = t.cust_id;
```

Below the query window, the 'Results' pane shows the output of the query:

	Customer Name	tran_dt	Outstanding Balance
1	Isaac White	2022-09-31	910.0000
2	Jade Kingdom	2020-10-30	700.0000
3	Jade Kingdom	2019-06-25	500.0000

The status bar at the bottom indicates 'Query executed successfully.' and '3 rows'.

-- Question 10 - Generate a SQL query with all details like Customer, account, type, Balance amount based on the transaction.

SELECT

```
t.id 'Transaction ID',  
c.first_name 'First Name',  
c.last_name 'Last Name',  
a.acc_type 'Account Type',  
a.balance 'Balance'
```

FROM

Customers c

INNER JOIN Accounts a

ON c.acc_id = a.id

INNER JOIN Transactions t

ON t.acc_id = a.id;

The screenshot displays the Microsoft SQL Server Management Studio interface. The title bar indicates the connection to 'DESKTOP-VOT1DO6\MSSQLSERVER_NEW.BankDB (DESKTOP-VOT1DO6\Ezra Muir (69))'. The Object Explorer on the left shows the database structure, including 'BankDB' with tables like 'dbo.Accounts' and 'dbo.Transactions'. The main query editor contains the following SQL code:

```
-- Question 10 - Generate a SQL query with all details like Customer, account, type, Balance amo  
220  
221 SELECT  
222 t.id 'Transaction ID',  
223 c.first_name 'First Name',  
224 c.last_name 'Last Name',  
225 a.acc_type 'Account Type',  
226 a.balance 'Balance'  
227 FROM  
228 Customers c  
229 INNER JOIN Accounts a  
230 ON c.acc_id = a.id  
231 INNER JOIN Transactions t  
232 ON t.acc_id = a.id;  
233
```

Below the query editor, the 'Results' pane shows the output of the query, which consists of 6 rows of data:

Transaction ID	First Name	Last Name	Account Type	Balance
1	eza	muir	SAVINGS	1000.00
2	eza	muir	SAVINGS	1000.00
3	eza	muir	SAVINGS	1000.00
4	truzel	white	SAVINGS	300.00
5	jada	kingdom	CURRENT	1000.00
6	jada	kingdom	CURRENT	1000.00

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-VOT1DO6\MSSQLSERVER_NEW.BankDB 00:00:00 6 rows'.

-- Question 11 - Generate a SQL query with loan details and history of payments done by the customer and outstanding amount to be paid.

SELECT

```
c.first_name 'First Name',  
c.last_name 'Last Name',  
  
l.id 'Loan ID',  
  
l.amount 'Loan Amount',  
  
t.id 'Transaction ID',  
  
t.acc_id 'Account ID',  
  
t.cust_id 'Customer ID',  
  
t.loan_id 'Loan ID',  
  
t.tran_amt 'Transaction Amount',  
  
t.tran_dt 'Transaction Date'
```

FROM

Customers c

INNER JOIN Loans l

ON c.loan_id = l.id

INNER JOIN Transactions t

ON t.id = l.id

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
-- Question 11 - Generate a SQL query with loan details and history of payments done by the cust  
SELECT  
c.first_name 'First Name',  
c.last_name 'Last Name',  
l.id 'Loan ID',  
l.amount 'Loan Amount',  
t.id 'Transaction ID',  
t.acc_id 'Account ID',  
t.cust_id 'Customer ID',  
t.loan_id 'Loan ID',  
t.tran_amt 'Transaction Amount',  
t.tran_dt 'Transaction Date'  
FROM  
Customers c  
INNER JOIN Loans l  
ON c.loan_id = l.id  
INNER JOIN Transactions t  
ON t.id = l.id;
```

The Results pane shows the following data:

	First Name	Last Name	Loan ID	Loan Amount	Transaction ID	Account ID	Customer ID	Loan ID	Transaction Amount	Transaction Date
1	Suzanne	White	2	1000.00	1	1	1	NULL	100001.00	2021-12-31
2	Jada	Kingdom	2	1000.00	2	1	1	NULL	100001.00	2021-12-31

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-VOT1D06\MSSQLSERVER... DESKTOP-VOT1D06\Ezra M... BankDB 00:00:00 2 rows".