**Final Project summary**

**Credit Card Billing System**

**Group: Final Project 100**

Balaji Gewini Vijayabaskar

Nagalakshmi Cherukuri

/\*

\*\* Author: Balaji Gewini Vijayabaskar, Nagalakshmi Cherukuri

\*\* Course: IFT/530

\*\* SQL Server Version: Microsoft SQL Server 2022(RTM)

\*\* History

\*\* Date Created Comments

\*\* 04/30/2025 Credit Card Billing System (Final Project)

\*/

‘Create database’

A screenshot of a computer

AI-generated content may be incorrect.

‘Create the tables using SQL script you have defined in the Final project.’

‘Code:’

/\*

\*\* Author: Balaji Gewini Vijayabaskar, Nagalakshmi Cherukuri

\*\* Course: IFT/530

\*\* SQL Server Version: Microsoft SQL Server 2022(RTM)

\*\* History

\*\* Date Created Comments

\*\* 04/30/2025 Credit Card Billing System (Final Project)

\*/

CREATE DATABASE Final\_Project\_100;

USE Final\_Project\_100;

-- Drop existing tables in dependency order

IF EXISTS (SELECT \* FROM sysobjects WHERE id = OBJECT\_ID('LateFees') AND xtype = 'U')

DROP TABLE LateFees;

IF EXISTS (SELECT \* FROM sysobjects WHERE id = OBJECT\_ID('Payments') AND xtype = 'U')

DROP TABLE Payments;

IF EXISTS (SELECT \* FROM sysobjects WHERE id = OBJECT\_ID('BillingStatements') AND xtype = 'U')

DROP TABLE BillingStatements;

IF EXISTS (SELECT \* FROM sysobjects WHERE id = OBJECT\_ID('Transactions') AND xtype = 'U')

DROP TABLE Transactions;

IF EXISTS (SELECT \* FROM sysobjects WHERE id = OBJECT\_ID('Rewards') AND xtype = 'U')

DROP TABLE Rewards;

IF EXISTS (SELECT \* FROM sysobjects WHERE id = OBJECT\_ID('CreditCards') AND xtype = 'U')

DROP TABLE CreditCards;

IF EXISTS (SELECT \* FROM sysobjects WHERE id = OBJECT\_ID('Customers') AND xtype = 'U')

DROP TABLE Customers;

-- Customers Table

CREATE TABLE Customers (

customer\_id INT IDENTITY(1,1) PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

phone VARCHAR(15) NOT NULL UNIQUE,

email VARCHAR(100),

address VARCHAR(300)

);

-- CreditCards Table

CREATE TABLE CreditCards (

card\_id INT IDENTITY(1,1) PRIMARY KEY,

customer\_id INT NOT NULL,

card\_number VARCHAR(16) NOT NULL UNIQUE,

card\_type VARCHAR(20) NOT NULL CHECK (card\_type IN ('Visa', 'MasterCard', 'Amex')),

credit\_limit DECIMAL(10,2) NOT NULL,

expiry\_date DATE NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

-- Transactions Table (Fact Table)

CREATE TABLE Transactions (

txn\_id INT IDENTITY(1,1) PRIMARY KEY,

card\_id INT NOT NULL,

txn\_date DATETIME NOT NULL,

merchant\_name VARCHAR(100) NOT NULL,

amount DECIMAL(10,2) NOT NULL CHECK (amount >= 0),

txn\_type VARCHAR(15) NOT NULL CHECK (txn\_type IN ('DEBIT', 'REFUND')),

FOREIGN KEY (card\_id) REFERENCES CreditCards(card\_id)

);

-- BillingStatements Table (Fact Table)

CREATE TABLE BillingStatements (

statement\_id INT IDENTITY(1,1) PRIMARY KEY,

card\_id INT NOT NULL,

billing\_period VARCHAR(20) NOT NULL,

total\_due DECIMAL(10,2) NOT NULL,

due\_date DATE NOT NULL,

min\_due DECIMAL(10,2) NOT NULL,

generated\_on DATE NOT NULL,

FOREIGN KEY (card\_id) REFERENCES CreditCards(card\_id)

);

-- Payments Table

CREATE TABLE Payments (

payment\_id INT IDENTITY(1,1) PRIMARY KEY,

statement\_id INT NOT NULL,

paid\_on DATE NOT NULL,

paid\_amount DECIMAL(10,2) NOT NULL CHECK (paid\_amount >= 0),

payment\_method VARCHAR(50) NOT NULL,

FOREIGN KEY (statement\_id) REFERENCES BillingStatements(statement\_id)

);

-- LateFees Table

CREATE TABLE LateFees (

fee\_id INT IDENTITY(1,1) PRIMARY KEY,

statement\_id INT NOT NULL,

fee\_amount DECIMAL(10,2) NOT NULL,

reason VARCHAR(200) NOT NULL,

applied\_on DATE NOT NULL,

FOREIGN KEY (statement\_id) REFERENCES BillingStatements(statement\_id)

);

-- Rewards Table

-- Rewards Table

CREATE TABLE Rewards (

reward\_id INT IDENTITY(1,1) PRIMARY KEY,

card\_id INT NOT NULL,

points\_earned INT NOT NULL DEFAULT 0,

points\_redeemed INT NOT NULL DEFAULT 0,

last\_updated DATE NOT NULL,

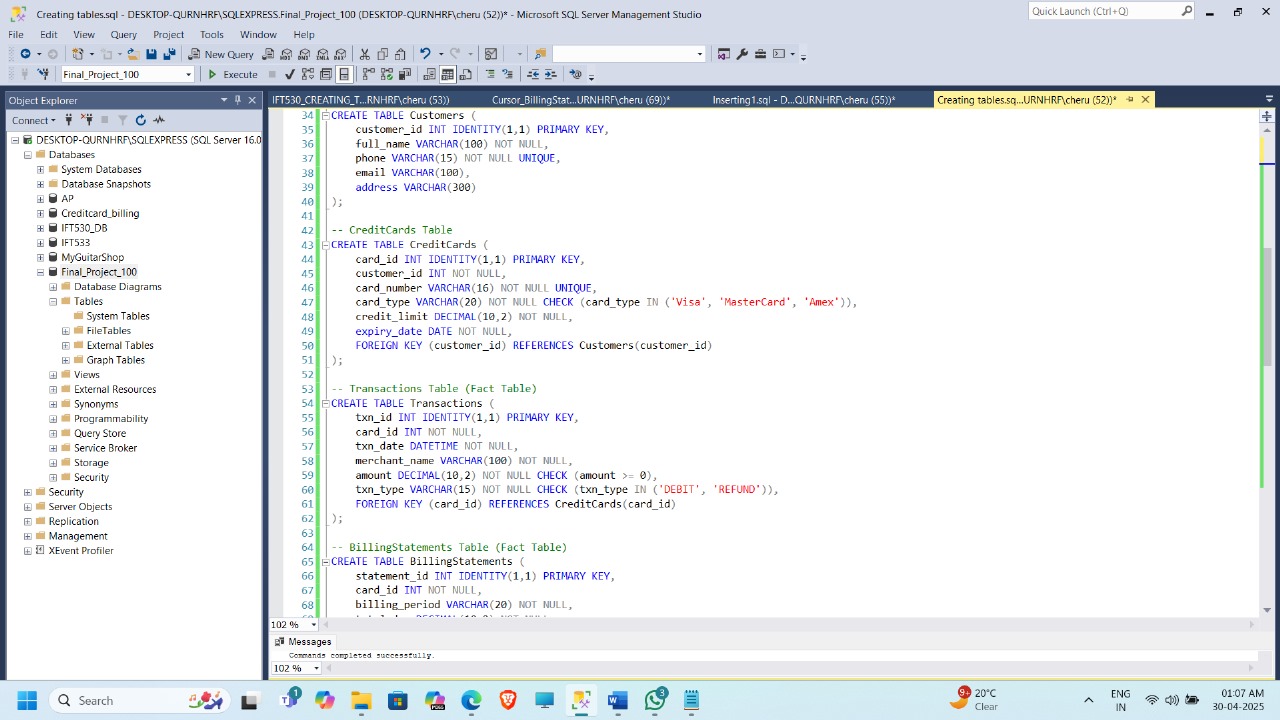
FOREIGN KEY (card\_id) REFERENCES CreditCards(card\_id)

);

‘Image:’

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

‘Generate the ERD diagram and paste it in a word document.’

‘Image’

A computer screen shot of a computer screen

AI-generated content may be incorrect.

‘Populate the table:’

‘code:’

/\*

\*\* Author: Balaji Gewini Vijayabaskar, Nagalakshmi Cherukuri

\*\* Course: IFT/530

\*\* SQL Server Version: Microsoft SQL Server 2022(RTM)

\*\* History

\*\* Date Created Comments

\*\* 04/30/2025 Credit Card Billing System (Final Project)

\*/

use Final\_Project\_100;

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 1', '9876543201', 'customer1@mail.com', '1 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 2', '9876543202', 'customer2@mail.com', '2 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 3', '9876543203', 'customer3@mail.com', '3 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 4', '9876543204', 'customer4@mail.com', '4 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 5', '9876543205', 'customer5@mail.com', '5 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 6', '9876543206', 'customer6@mail.com', '6 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 7', '9876543207', 'customer7@mail.com', '7 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 8', '9876543208', 'customer8@mail.com', '8 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 9', '9876543209', 'customer9@mail.com', '9 Main St, City');

INSERT INTO Customers (full\_name, phone, email, address) VALUES ('Customer 10', '9876543210', 'customer10@mail.com', '10 Main St, City');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (1, '4111111111111101', 'Visa', 8385, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (2, '4111111111111102', 'Amex', 8354, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (3, '4111111111111103', 'Visa', 2904, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (4, '4111111111111104', 'Amex', 9402, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (5, '4111111111111105', 'Visa', 6966, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (6, '4111111111111106', 'MasterCard', 3473, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (7, '4111111111111107', 'Amex', 3907, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (8, '4111111111111108', 'MasterCard', 6853, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (9, '4111111111111109', 'MasterCard', 2179, '2027-12-31');

INSERT INTO CreditCards (customer\_id, card\_number, card\_type, credit\_limit, expiry\_date) VALUES (10, '4111111111111110', 'MasterCard', 5876, '2027-12-31');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (7, '2025-04-02 00:00:00', 'Merchant 1', 110.59, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (10, '2025-04-03 00:00:00', 'Merchant 2', 94.48, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (3, '2025-04-04 00:00:00', 'Merchant 3', 166.04, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (8, '2025-04-05 00:00:00', 'Merchant 4', 75.84, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (10, '2025-04-06 00:00:00', 'Merchant 5', 175.48, 'DEBIT');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (1, '2025-04-07 00:00:00', 'Merchant 6', 273.88, 'DEBIT');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (10, '2025-04-08 00:00:00', 'Merchant 7', 207.64, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (3, '2025-04-09 00:00:00', 'Merchant 8', 128.99, 'DEBIT');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (9, '2025-04-10 00:00:00', 'Merchant 9', 396.19, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (8, '2025-04-11 00:00:00', 'Merchant 10', 131.93, 'DEBIT');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (9, '2025-04-12 00:00:00', 'Merchant 11', 113.53, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (4, '2025-04-13 00:00:00', 'Merchant 12', 58.06, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (3, '2025-04-14 00:00:00', 'Merchant 13', 486.93, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (4, '2025-04-15 00:00:00', 'Merchant 14', 116.3, 'DEBIT');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (3, '2025-04-16 00:00:00', 'Merchant 15', 452.32, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (3, '2025-04-17 00:00:00', 'Merchant 16', 363.77, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (2, '2025-04-18 00:00:00', 'Merchant 17', 223.39, 'REFUND');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (4, '2025-04-19 00:00:00', 'Merchant 18', 79.01, 'DEBIT');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (1, '2025-04-20 00:00:00', 'Merchant 19', 109.64, 'DEBIT');

INSERT INTO Transactions (card\_id, txn\_date, merchant\_name, amount, txn\_type) VALUES (4, '2025-04-21 00:00:00', 'Merchant 20', 494.36, 'DEBIT');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (1, 'Apr 2025', 105.51, '2025-04-25', 39.54, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (2, 'Apr 2025', 642.87, '2025-04-25', 66.23, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (3, 'Apr 2025', 389.84, '2025-04-25', 57.37, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (4, 'Apr 2025', 560.19, '2025-04-25', 76.61, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (5, 'Apr 2025', 896.66, '2025-04-25', 62.57, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (6, 'Apr 2025', 718.33, '2025-04-25', 99.07, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (7, 'Apr 2025', 640.49, '2025-04-25', 53.78, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (8, 'Apr 2025', 179.92, '2025-04-25', 29.28, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (9, 'Apr 2025', 395.71, '2025-04-25', 81.91, '2025-04-01');

INSERT INTO BillingStatements (card\_id, billing\_period, total\_due, due\_date, min\_due, generated\_on) VALUES (10, 'Apr 2025', 935.92, '2025-04-25', 88.49, '2025-04-01');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (6, '2025-04-11', 370.88, 'Card');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (8, '2025-04-12', 103.47, 'Card');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (9, '2025-04-13', 321.55, 'Card');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (4, '2025-04-14', 356.23, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (1, '2025-04-15', 179.49, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (1, '2025-04-16', 481.75, 'Bank');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (10, '2025-04-17', 438.42, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (3, '2025-04-18', 350.3, 'Card');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (2, '2025-04-19', 61.06, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (8, '2025-04-20', 179.37, 'Bank');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (6, '2025-04-21', 318.86, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (8, '2025-04-22', 457.92, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (10, '2025-04-23', 455.48, 'Card');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (1, '2025-04-24', 479.1, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (4, '2025-04-25', 159.39, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (10, '2025-04-26', 197.41, 'Bank');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (6, '2025-04-27', 430.05, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (6, '2025-04-28', 51.89, 'Bank');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (1, '2025-04-29', 373.28, 'UPI');

INSERT INTO Payments (statement\_id, paid\_on, paid\_amount, payment\_method) VALUES (4, '2025-04-30', 418.13, 'UPI');

INSERT INTO LateFees (statement\_id, fee\_amount, reason, applied\_on) VALUES (2, 30.01, 'Late payment', '2025-04-6');

INSERT INTO LateFees (statement\_id, fee\_amount, reason, applied\_on) VALUES (4, 27.44, 'Late payment', '2025-04-7');

INSERT INTO LateFees (statement\_id, fee\_amount, reason, applied\_on) VALUES (3, 37.35, 'Late payment', '2025-04-8');

INSERT INTO LateFees (statement\_id, fee\_amount, reason, applied\_on) VALUES (7, 15.72, 'Late payment', '2025-04-9');

INSERT INTO LateFees (statement\_id, fee\_amount, reason, applied\_on) VALUES (8, 38.42, 'Late payment', '2025-04-10');

INSERT INTO LateFees (statement\_id, fee\_amount, reason, applied\_on)

VALUES

(2, 30.01, 'Late payment', '2025-04-06'),

(4, 27.44, 'Late payment', '2025-04-07'),

(3, 37.35, 'Late payment', '2025-04-08'),

(7, 15.72, 'Late payment', '2025-04-09'),

(8, 38.42, 'Late payment', '2025-04-10'),

(1, 25.00, 'Late payment', '2025-04-11'),

(2, 32.75, 'Late payment', '2025-04-12'),

(3, 29.99, 'Late payment', '2025-04-13'),

(4, 33.50, 'Late payment', '2025-04-14'),

(5, 40.00, 'Late payment', '2025-04-15'),

(6, 30.00, 'Late payment', '2025-04-16'),

(1, 15.00, 'Late payment', '2025-05-06'),

(2, 18.00, 'Late payment', '2025-05-07'),

(3, 12.50, 'Late payment', '2025-05-08'),

(4, 20.00, 'Late payment', '2025-05-09'),

(5, 22.75, 'Late payment', '2025-05-10'),

(6, 19.00, 'Late payment', '2025-05-11'),

(7, 17.25, 'Late payment', '2025-05-12'),

(8, 23.10, 'Late payment', '2025-05-13'),

(9, 21.45, 'Late payment', '2025-05-14');

GO

INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (1, 270, 160, '2025-04-10');

INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (2, 867, 265, '2025-04-10');

INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (3, 990, 479, '2025-04-10');

INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (4, 825, 206, '2025-04-10');

INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (5, 601, 115, '2025-04-10');

INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (6, 641, 287, '2025-04-10');

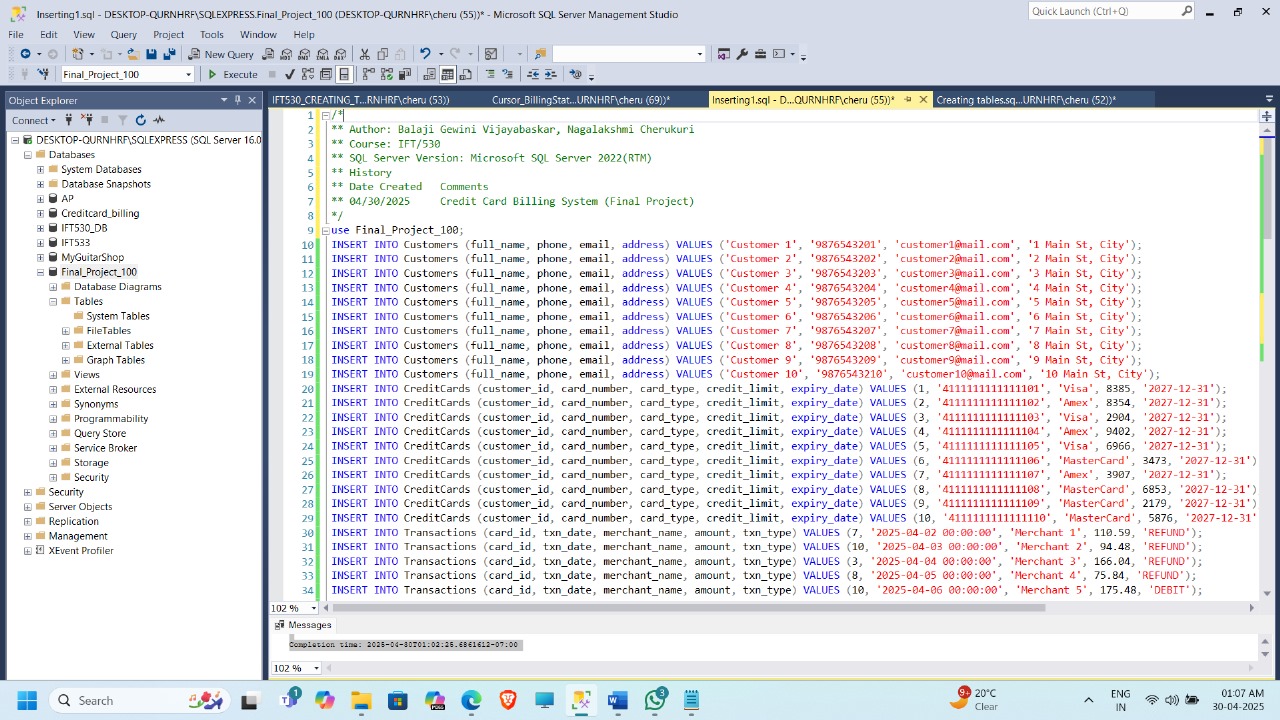
INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (7, 935, 24, '2025-04-10');

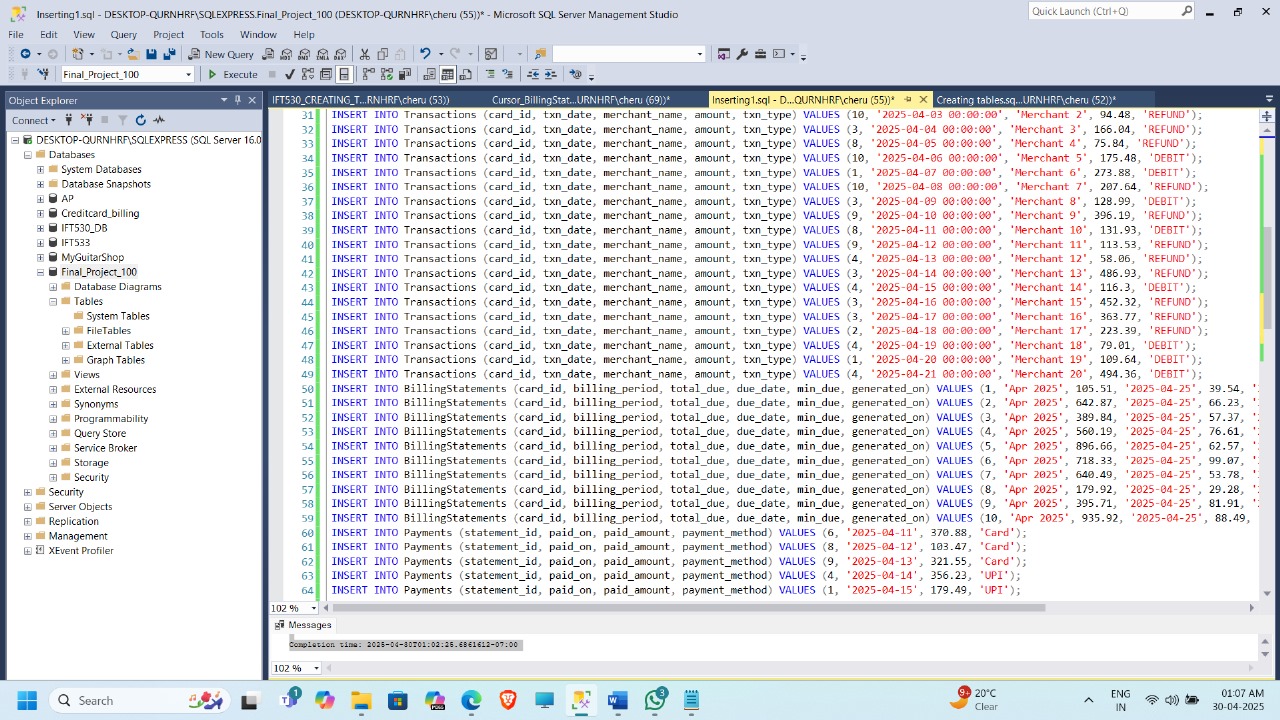
INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (8, 394, 149, '2025-04-10');

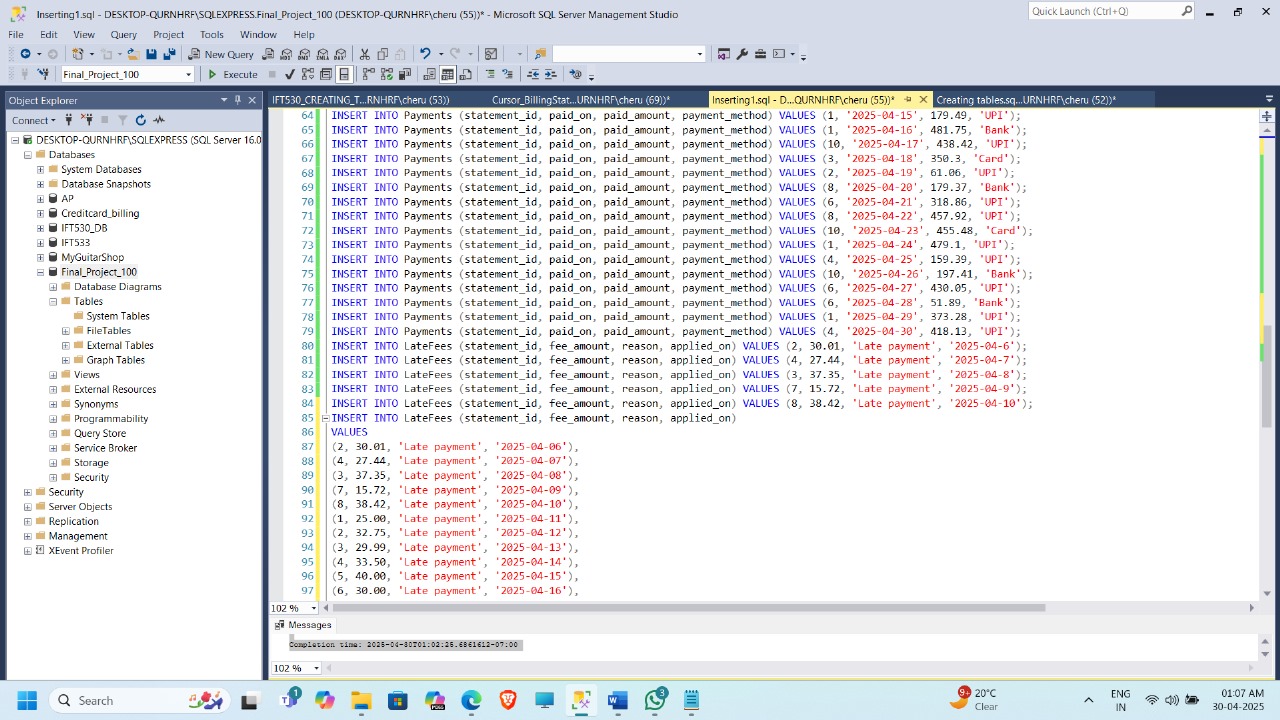
INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (9, 615, 416, '2025-04-10');

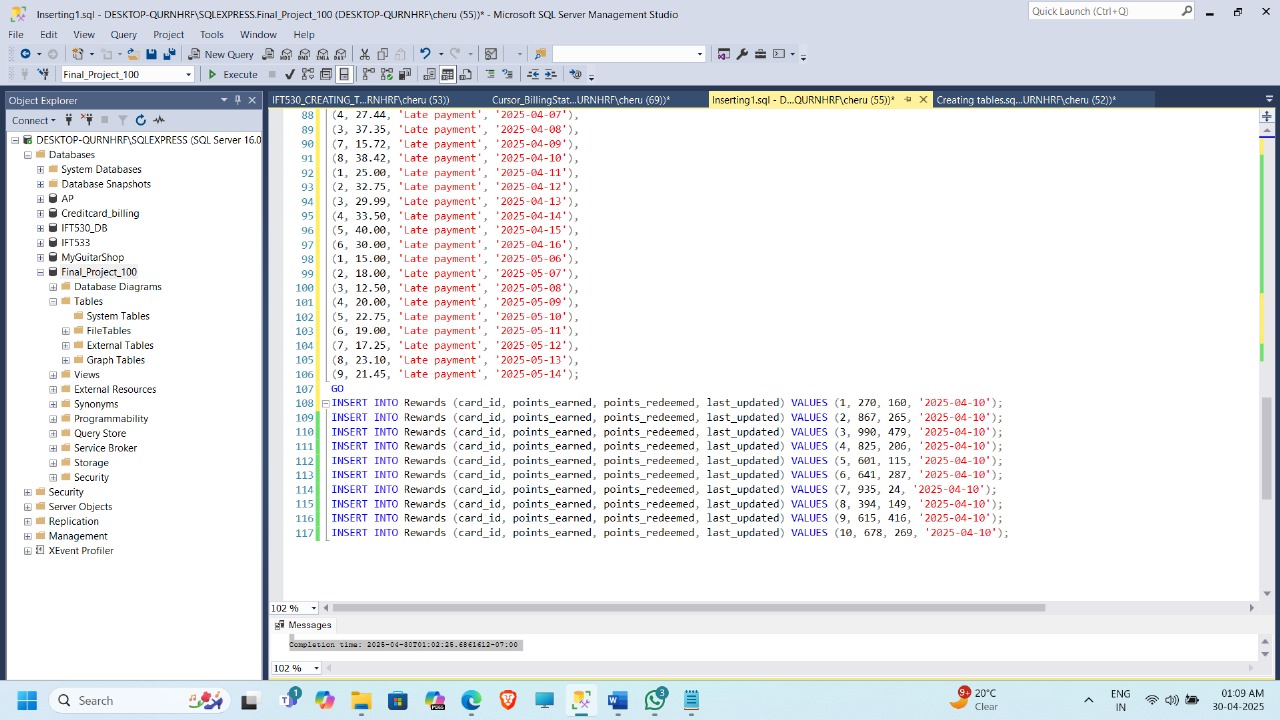
INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated) VALUES (10, 678, 269, '2025-04-10');

‘Image:’









‘Create three queries and convert them into views’

‘code’

/\*

\*\* Author: Balaji Gewini Vijayabaskar, Nagalakshmi Cherukuri

\*\* Course: IFT/530

\*\* SQL Server Version: Microsoft SQL Server 2022(RTM)

\*\* History

\*\* Date Created Comments

\*\* 04/30/2025 Credit Card Billing System (Final Project)

\*/

USE Final\_Project\_100;

-- Drop existing views if they exist

IF OBJECT\_ID('vw\_HighOutstandingBalances', 'V') IS NOT NULL

DROP VIEW vw\_HighOutstandingBalances;

GO

CREATE VIEW vw\_HighOutstandingBalances AS

SELECT

c.full\_name,

cc.card\_number,

bs.billing\_period,

bs.total\_due

FROM

Customers c

JOIN CreditCards cc ON c.customer\_id = cc.customer\_id

JOIN BillingStatements bs ON cc.card\_id = bs.card\_id

WHERE

bs.total\_due > 500;

GO

SELECT \* FROM vw\_HighOutstandingBalances;

GO

-- View 2: Recent Late Fees

IF OBJECT\_ID('vw\_RecentLateFees', 'V') IS NOT NULL

DROP VIEW vw\_RecentLateFees;

GO

CREATE VIEW vw\_RecentLateFees AS

SELECT

c.full\_name,

cc.card\_number,

bs.billing\_period,

lf.fee\_amount,

lf.applied\_on

FROM

Customers c

JOIN CreditCards cc ON c.customer\_id = cc.customer\_id

JOIN BillingStatements bs ON cc.card\_id = bs.card\_id

JOIN LateFees lf ON bs.statement\_id = lf.statement\_id

WHERE

lf.applied\_on >= DATEADD(DAY, -30, GETDATE());

GO

SELECT \* FROM vw\_RecentLateFees;

GO

-- View 3: Reward Summary

IF OBJECT\_ID('vw\_RewardSummary', 'V') IS NOT NULL

DROP VIEW vw\_RewardSummary;

GO

CREATE VIEW vw\_RewardSummary AS

SELECT

c.full\_name,

cc.card\_number,

r.points\_earned,

r.points\_redeemed,

(r.points\_earned - r.points\_redeemed) AS remaining\_points

FROM

Customers c

JOIN CreditCards cc ON c.customer\_id = cc.customer\_id

JOIN Rewards r ON cc.card\_id = r.card\_id

WHERE

r.points\_earned > 0;

GO

SELECT \* FROM vw\_RewardSummary;

GO

View 1 :

Explanation of Usefulness:

This view filters for customers owing more than $500, enabling: Follow-up to be given priority in collections units. Customer debt trends to be tracked by risk analysts. Customer service teams to step in proactively to ask for payment support or restructuring plans.

‘Image’

A screenshot of a computer

AI-generated content may be incorrect.

View 2 :

Explanation of Usefulness

This perspective calls out customers who have recently incurred late fees. It is valuable to: Customer service representatives to notify and educate customers about charges. Operations teams to measure whether payment reminders are working. Management to track trends in late payments and plan for optimization.

A screenshot of a computer

AI-generated content may be incorrect.

View 3 :

Explanation of Usefulness:

This view presents a clean picture of a customer's reward position. It can be used by: Marketing teams to market highest reward earners for promotions. Customers (via dashboards) to track and redeem points. Finance to predict reward liabilities.

A screenshot of a computer

AI-generated content may be incorrect.

‘Create an audit table:’

‘code’

/\*

\*\* Author: Balaji Gewini Vijayabaskar, Nagalakshmi Cherukuri

\*\* Course: IFT/530

\*\* SQL Server Version: Microsoft SQL Server 2022(RTM)

\*\* History

\*\* Date Created Comments

\*\* 04/30/2025 Credit Card Billing System (Final Project)

\*/

USE Final\_Project\_100;

GO

-- Step 1: Create the Audit Table

IF OBJECT\_ID('Rewards\_Audit', 'U') IS NOT NULL

DROP TABLE Rewards\_Audit;

GO

CREATE TABLE Rewards\_Audit (

audit\_id INT IDENTITY(1,1) PRIMARY KEY,

reward\_id INT,

card\_id INT,

points\_earned INT,

points\_redeemed INT,

last\_updated DATE,

action\_type VARCHAR(10), -- 'INSERT', 'UPDATE', 'DELETE'

changed\_on DATETIME DEFAULT GETDATE()

);

GO

-- Step 2: Create Trigger for INSERT

IF OBJECT\_ID('trg\_Reward\_Insert', 'TR') IS NOT NULL

DROP TRIGGER trg\_Reward\_Insert;

GO

CREATE TRIGGER trg\_Reward\_Insert

ON Rewards

AFTER INSERT

AS

BEGIN

INSERT INTO Rewards\_Audit (reward\_id, card\_id, points\_earned, points\_redeemed, last\_updated, action\_type)

SELECT reward\_id, card\_id, points\_earned, points\_redeemed, last\_updated, 'INSERT'

FROM inserted;

END;

GO

-- Step 3: Create Trigger for UPDATE

IF OBJECT\_ID('trg\_Reward\_Update', 'TR') IS NOT NULL

DROP TRIGGER trg\_Reward\_Update;

GO

CREATE TRIGGER trg\_Reward\_Update

ON Rewards

AFTER UPDATE

AS

BEGIN

INSERT INTO Rewards\_Audit (reward\_id, card\_id, points\_earned, points\_redeemed, last\_updated, action\_type)

SELECT reward\_id, card\_id, points\_earned, points\_redeemed, last\_updated, 'UPDATE'

FROM inserted;

END;

GO

-- Step 4: Create Trigger for DELETE

IF OBJECT\_ID('trg\_Reward\_Delete', 'TR') IS NOT NULL

DROP TRIGGER trg\_Reward\_Delete;

GO

CREATE TRIGGER trg\_Reward\_Delete

ON Rewards

AFTER DELETE

AS

BEGIN

INSERT INTO Rewards\_Audit (reward\_id, card\_id, points\_earned, points\_redeemed, last\_updated, action\_type)

SELECT reward\_id, card\_id, points\_earned, points\_redeemed, last\_updated, 'DELETE'

FROM deleted;

END;

GO

-- Step 5: Test Scripts

-- Insert new reward (will trigger INSERT)

INSERT INTO Rewards (card\_id, points\_earned, points\_redeemed, last\_updated)

VALUES (1, 100, 10, '2025-04-28');

-- Update reward (will trigger UPDATE)

UPDATE Rewards

SET points\_earned = 10

WHERE card\_id = 1;

-- Delete reward (will trigger DELETE)

DELETE FROM Rewards

WHERE card\_id = 1;

-- View audit logs

SELECT \* FROM Rewards\_Audit;

GO

‘image’

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

‘stored procedures and User Defined Function’

‘code:’

/\*

\*\* Author: Balaji Gewini Vijayabaskar, Nagalakshmi Cherukuri

\*\* Course: IFT/530

\*\* SQL Server Version: Microsoft SQL Server 2022(RTM)

\*\* History

\*\* Date Created Comments

\*\* 04/30/2025 Credit Card Billing System (Final Project)

\*/

USE Final\_Project\_100;

GO

-- USER DEFINED FUNCTION: Get Total Rewards by Card ID

IF OBJECT\_ID('dbo.fn\_GetTotalRewardPoints', 'FN') IS NOT NULL

DROP FUNCTION dbo.fn\_GetTotalRewardPoints;

GO

CREATE FUNCTION dbo.fn\_GetTotalRewardPoints (@card\_id INT)

RETURNS INT

AS

BEGIN

DECLARE @totalPoints INT;

SELECT @totalPoints = ISNULL(SUM(points\_earned - points\_redeemed), 0)

FROM Rewards

WHERE card\_id = @card\_id;

RETURN @totalPoints;

END;

GO

-- STORED PROCEDURE: Get Customer Billing Summary

IF OBJECT\_ID('dbo.sp\_GetCustomerBillingSummary', 'P') IS NOT NULL

DROP PROCEDURE dbo.sp\_GetCustomerBillingSummary;

GO

CREATE PROCEDURE dbo.sp\_GetCustomerBillingSummary

@customer\_id INT

AS

BEGIN

SELECT

c.full\_name,

cc.card\_number,

bs.billing\_period,

bs.total\_due,

bs.min\_due,

bs.due\_date

FROM Customers c

JOIN CreditCards cc ON c.customer\_id = cc.customer\_id

JOIN BillingStatements bs ON cc.card\_id = bs.card\_id

WHERE c.customer\_id = @customer\_id;

END;

GO

-- TESTING

-- Test the UDF

SELECT dbo.fn\_GetTotalRewardPoints(0) AS TotalRewardPoints;

GO

-- Test the stored procedure

EXEC dbo.sp\_GetCustomerBillingSummary @customer\_id = 1;

GO

‘Image:’

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

‘cursor for your database.’

‘code:’

/\*

\*\* Author: Balaji Gewini Vijayabaskar, Nagalakshmi Cherukuri

\*\* Course: IFT/530

\*\* SQL Server Version: Microsoft SQL Server 2022(RTM)

\*\* History

\*\* Date Created Comments

\*\* 04/30/2025 Credit Card Billing System (Final Project)

\*/

USE Final\_Project\_100;

GO

-- CURSOR: Print all billing statements with total\_due > 0

-- DROP existing procedure if it exists

IF OBJECT\_ID('dbo.sp\_ProcessBillingDue', 'P') IS NOT NULL

DROP PROCEDURE dbo.sp\_ProcessBillingDue;

GO

-- CREATE procedure that uses a CURSOR

CREATE PROCEDURE dbo.sp\_ProcessBillingDue

AS

BEGIN

DECLARE @statement\_id INT,

@total\_due DECIMAL(10,2),

@due\_date DATE;

DECLARE billing\_cursor CURSOR FOR

SELECT statement\_id, total\_due, due\_date

FROM BillingStatements

WHERE total\_due > 150;

OPEN billing\_cursor;

FETCH NEXT FROM billing\_cursor INTO @statement\_id, @total\_due, @due\_date;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Statement ID: ' + CAST(@statement\_id AS VARCHAR) +

' | Due: $' + CAST(@total\_due AS VARCHAR) +

' | Due Date: ' + CAST(@due\_date AS VARCHAR);

-- You can also add logic to send alerts, update flags, etc. here.

FETCH NEXT FROM billing\_cursor INTO @statement\_id, @total\_due, @due\_date;

END

CLOSE billing\_cursor;

DEALLOCATE billing\_cursor;

END;

GO

-- Run the procedure to test the cursor

EXEC dbo.sp\_ProcessBillingDue;

GO

‘Image’

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.