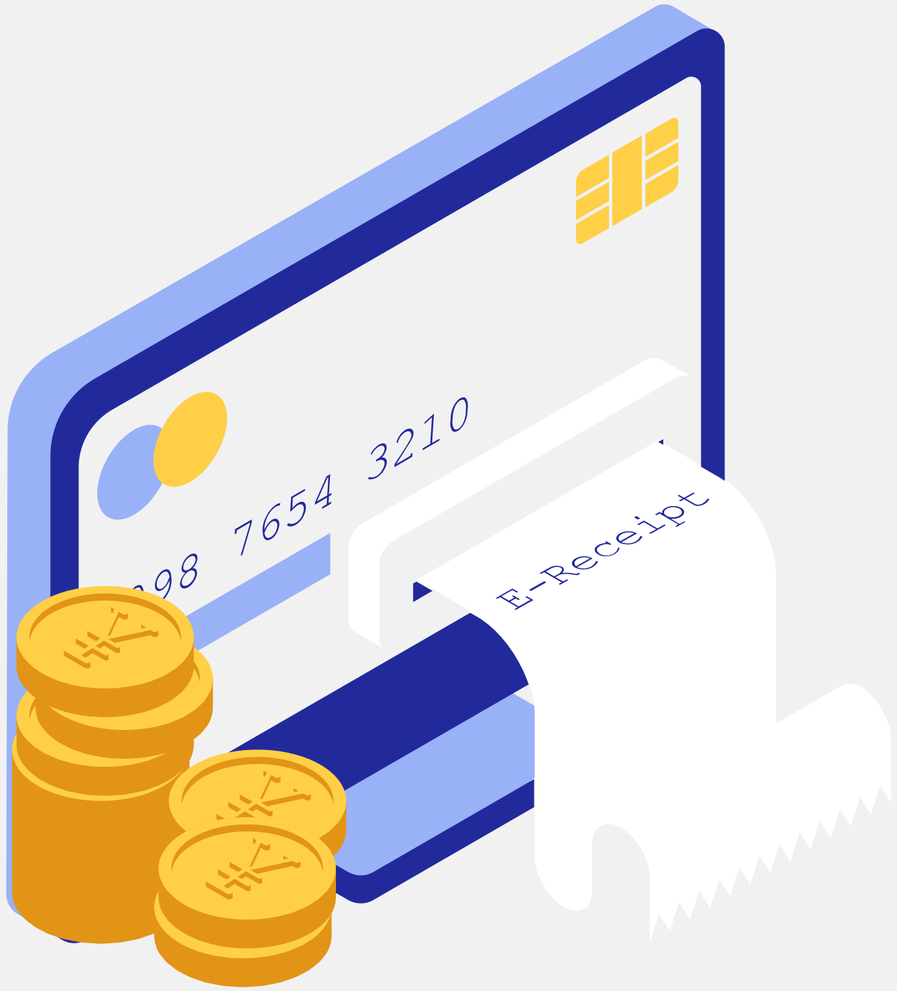


BankHub – Team 11

DAMG 6210 - Data Management and Database Design
Instructor – Wu-Ping Simon Wang

1) Darshan Gohil
2) Sagar Jayantilal Satra
3) Anusha Prakash
4) Nikhil Choudhari

Business Problems

- Fragmented Account Access: Customers face difficulties accessing accounts across multiple banks and consolidating their financial history for different banking products.

- Decentralized Compliance Mechanisms: There is a lack of unified compliance checks for financial products, leading to inconsistencies and inefficiencies in regulatory adherence.

- Absence of a Centralized Comparison Platform: Customers do not have a single, insightful platform to fetch and compare similar financial products offered by diverse banks operating under various government regulations.

- Lack of Awareness About Products and Benefits: Customers often remain unaware of the full range of financial products and their associated benefits, resulting in underutilization of available options and missed opportunities for better financial management.

Purpose of BankHub

- Centralize customer data and bank services for seamless access across multiple banking products
- Enable data aggregation for banks to streamline financial insights and product offerings
- Provide governments with tools to monitor and enforce financial compliance
- Enhance transparency and efficiency in banking operations for all stakeholders

Business Rules and Design Decisions

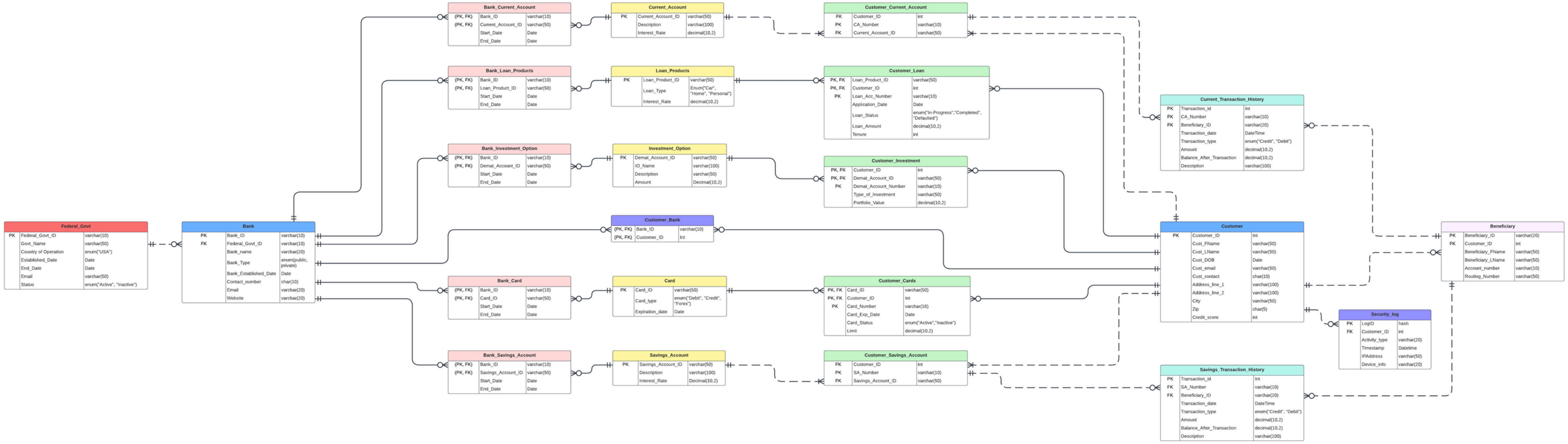
- **Customer Management:** Each customer can have multiple bank accounts across different banks, but each account is tied to only one customer
- **Beneficiary Relationship:** A customer can have multiple beneficiaries associated with their different accounts
- **Banking System:** Each bank can have multiple branches, and banks can offer various products such as loans, credit cards, and savings accounts
- **Transactions:** Customers conduct transactions that are tied to specific accounts and banks. Each transaction will record the type, amount, and resulting balance
- **Compliance and Regulation:** Banks are regulated by the federal government, which enforces regulations and assigns compliance role

Database Design Considerations

Design Decisions:-

- Data centralization
- Enforcing 3NF for scalability
- Encrypting sensitive data for compliance and a secure environment
- 3 Major Entities - Customer, Bank, and Federal_Bank represent real-world stakeholders.
- Relationships(e.g., Customer-Transaction-History, Federal-Bank) enforce data monitoring
- Constraints like encrypted fields ensure security

ERD



Design Document

Data Definition Language - Table Creation

Bank Table Creation

```
CREATE TABLE BankSchema.Bank (
    Bank_ID VARCHAR(10) PRIMARY KEY,
    Federal_Govt_ID VARCHAR(10) NOT NULL,
    Bank_Name VARCHAR(20) NOT NULL,
    Bank_Type VARCHAR(10) NOT NULL,
    Bank_Established_Date DATE,
    Contact_number CHAR(10),
    Email VARCHAR(20),
    Website VARCHAR(20),
    FOREIGN KEY (Federal_Govt_ID) REFERENCES BankSchema.FederalGovt(Federal_Govt_ID),
    CONSTRAINT chk_email_format CHECK (BankSchema.ValidateEmail(Email) = 1),
    CONSTRAINT chk_bank_type CHECK (Bank_Type IN ('Public', 'Private'))
);
```

Customer Table Creation

```
CREATE TABLE CustomerSchema.Customer (
    Customer_ID INT IDENTITY(1,1) PRIMARY KEY,       -- Auto-incrementing primary key
    Cust_FName VARCHAR(50) NOT NULL,                -- First name (Unicode support)
    Cust_LName VARCHAR(50) NOT NULL,                -- Last name (Unicode support)
    Cust_DOB DATE NOT NULL,                          -- Date of Birth
    Cust_email NVARCHAR(50) NOT NULL UNIQUE,         --
    Cust_contact CHAR(10) NOT NULL,                  -- 10-digit phone number
    Address_line_1 VARCHAR(100) NOT NULL,           -- Address line 1 (Unicode support)
    Address_line_2 VARCHAR(100) NULL,               -- Optional address line 2
    City VARCHAR(50) NOT NULL,                      -- City (Unicode support)
    Zip CHAR(5) NOT NULL,                            -- 5-digit ZIP code
    Credit_score INT NULL
);
```

Card Product Table Creation

```
CREATE TABLE BankSchema.Card (
    Card_ID VARCHAR(50) PRIMARY KEY,
    Card_Type VARCHAR(10) NOT NULL,
    Expiration_Date DATE NOT NULL,
    CONSTRAINT chk_card_type CHECK (Card_Type IN ('Debit', 'Credit', 'Forex')),
    CONSTRAINT chk_expiration_date CHECK (Expiration_Date > GETDATE()) -- Ensure
);
```

Beneficiary Table Creation

```
CREATE TABLE CustomerSchema.Beneficiary (
    Customer_ID INT,
    Beneficiary_ID VARCHAR(20) PRIMARY KEY,
    Beneficiary_FName VARCHAR(50) NOT NULL,
    Beneficiary_LName VARCHAR(50) NOT NULL,
    Account_number VARCHAR(10),
    Routing_Number VARCHAR(50) NOT NULL,
    FOREIGN KEY (Customer_ID) REFERENCES CustomerSchema.Customer(Customer_ID)
);
```

DML - Data Insertions

Customer Table Insertion

```
INSERT INTO CustomerSchema.Customer (
    Cust_FName,
    Cust_LName,
    Cust_DOB,
    Cust_email,
    Cust_contact,
    Address_line_1,
    Address_line_2,
    City,
    Zip
)
VALUES
    ('Michael', 'Johnson', '1990-11-10', 'michael.johnson@yahoo.com', '9876543210', '55 Elmwood Ave', NULL, 'Boston'
    ('Jessica', 'Williams', '1985-05-05', 'jessica.williams@outlook.com', '4157896543', '202 Ocean Blvd', 'Suite 5',
    ('Daniel', 'Smith', '1992-07-18', 'daniel.smith@gmail.com', '3216549870', '12 Cedar Lane', NULL, 'Chicago', '606
    ('Sophia', 'Hernandez', '1989-03-12', 'sophia.hernandez@hotmail.com', '2027894561', '456 Pine St', 'Unit 6A', 'S
    ('Liam', 'Garcia', '1994-01-15', 'liam.garcia@gmail.com', '7147891234', '789 Birch Rd', NULL, 'Houston', '77001'
    ('Olivia', 'Martinez', '1997-09-22', 'olivia.martinez@aol.com', '3056543217', '101 Redwood Dr', 'Bldg 2', 'Miami
    ('Ethan', 'Rodriguez', '1991-08-05', 'ethan.rodriguez@gmail.com', '8187894321', '303 Willow Way', 'Ste 4A', 'Dal
    ('Emma', 'Lopez', '1988-06-14', 'emma.lopez@example.com', '6194567890', '123 Maple Ct', NULL, 'San Diego', '9210
    ('James', 'Taylor', '1984-02-28', 'james.taylor@example.com', '7039876543', '405 Spruce Circle', 'Apt 7C', 'Phil

Select * from CustomerSchema.Customer;
```

Card Product Table Insertion

```
INSERT INTO BankSchema.Card (Card_ID, Card_Type, Expiration_Date)
VALUES
('AMEX_Gold', 'Debit', '2025-12-31'),
('BBVA_ClearSpend', 'Forex', '2026-11-15'),
('CapitalOne_Venture', 'Credit', '2025-09-15'),
('Chase_Sapphire', 'Credit', '2026-06-15'),
('Discover_It', 'Debit', '2025-07-20'),
('FifthThird_Accelerate', 'Forex', '2026-07-25'),
('HSBC_Advance', 'Debit', '2027-03-01'),
('KeyBank_Preferred', 'Credit', '2025-12-31'),
('MasterCard_Black', 'Credit', '2026-11-30'),
('TDBank_Everyday', 'Debit', '2027-10-31'),
('Visa_Platinum', 'Forex', '2027-03-10');

Select * from BankSchema.Card;
```

Bank Table Insertion

```
INSERT INTO BankSchema.Bank (Bank_ID, Federal_Govt_ID, Bank_Name, Bank_Type, B
VALUES
('B001', 'FG001', 'Bank of America', 'Private', '1904-10-17', '9876543210', 'i
('B002', 'FG002', 'JPMorgan Chase', 'Private', '1871-12-01', '1234567890', 'co
('B003', 'FG003', 'Wells Fargo', 'Private', '1852-03-18', '1122334455', 'suppo
('B004', 'FG001', 'Citibank', 'Private', '1812-06-16', '2233445566', 'service@
('B005', 'FG002', 'Goldman Sachs', 'Private', '1869-11-01', '3344556677', 'inf
('B006', 'FG003', 'Morgan Stanley', 'Private', '1935-09-16', '4455667788', 'su
('B008', 'FG002', 'U.S. Bank', 'Public', '1863-07-13', '6677889900', 'info@usb
('B009', 'FG003', 'Truist Financial', 'Private', '1872-09-15', '7788990011', '
('B010', 'FG001', 'Capital One', 'Private', '1994-07-21', '8899001122', 'info@
('B011', 'FG002', 'TD Bank', 'Private', '1855-02-01', '9900112233', 'contact@t
('B012', 'FG003', 'Fifth Third Bank', 'Private', '1858-06-17', '1011121314', '
('B013', 'FG001', 'KeyBank', 'Private', '1825-11-12', '1112131415', 'support@k
('B014', 'FG002', 'HSBC Bank USA', 'Private', '1865-01-01', '1213141516', 'inf
('B015', 'FG003', 'BBVA USA', 'Private', '1964-03-01', '1314151617', 'contact@

Select * from BankSchema.Bank;
```

Beneficiary Table Insertion

```
INSERT INTO CustomerSchema.Beneficiary (Customer_ID, Beneficiary_FName, Beneficiary_LName, Account_number, 
VALUES
(3, 'Daniel', 'Smith', '6-CA003', '123456789'),  -- Customer 3 (Emily) transfers to Customer 6 (Daniel) wit
(6, 'Emily', 'Brown', '3-SA002', '987654321'),  -- Customer 6 (Daniel) transfers to Customer 3 (Emily) with
(4, 'Jessica', 'Williams', '5-SA001', '112233445'),  -- Customer 4 (James) transfers to Customer 5 (Olivia)
(5, 'Michael', 'Johnson', '4-SA003', '223344556'),  -- Customer 5 (Olivia) transfers to Customer 4 (James) 
(3, 'Olivia', 'Martinez', '9-CA010', '334455667'),  -- Customer 3 (Emily) transfers to Customer 9 (Sophia) 
(9, 'Emily', 'Brown', '3-SA001', '445566778'),  -- Customer 9 (Sophia) transfers to Customer 3 (Emily) with
(5, 'Sophia', 'Hernandez', '7-SA006', '556677889'),  -- Customer 5 (Olivia) transfers to Customer 7 (Olivia
(7, 'Jessica', 'Williams', '5-SA002', '667788990'),  -- Customer 7 (Olivia) transfers to Customer 5 (Olivia
(6, 'Liam', 'Garcia', '8-CA002', '778899001'),  -- Customer 6 (Daniel) transfers to Customer 8 (Olivia) wit
(8, 'Daniel', 'Smith', '6-CA004', '889900112'),  -- Customer 8 (Olivia) transfers to Customer 6 (Daniel) wi
(10, 'Emma', 'Lopez', '11-SA006', '990011223'),  -- Customer 10 (Sophia) transfers to Customer 11 (Olivia) 
(11, 'Ethan', 'Rodriguez', '10-CA010', '101112233'),  -- Customer 11 (Olivia) transfers to Customer 10 (Sop
(4, 'Liam', 'Garcia', '8-SA001', '112233445'),  -- Customer 4 (James) transfers to Customer 8 (Olivia) with
(8, 'Michael', 'Johnson', '4-SA004', '223344556'),  -- Customer 8 (Olivia) transfers to Customer 4 (James) 
(5, 'Olivia', 'Martinez', '9-CA006', '334455667'),  -- Customer 5 (Olivia) transfers to Customer 9 (Sophia)
(9, 'Jessica', 'Williams', '5-SA001', '445566778'),  -- Customer 9 (Sophia) transfers to Customer 5 (Olivia
(3, 'Emma', 'Lopez', '11-CA002', '556677889'),  -- Customer 3 (Emily) transfers to Customer 11 (Olivia) wit
(11, 'Emily', 'Brown', '3-CA002', '667788990'),  -- Customer 11 (Olivia) transfers to Customer 3 (Emily) wi
(12, 'Olivia', 'Martinez', '9-SA010', '778899001'),  -- Customer 12 (Olivia) transfers to Customer 9 (Sophi
(9, 'James', 'Taylor', '12-CA008', '889900112'),  -- Customer 9 (Sophia) transfers to Customer 12 (Olivia) 
-- People registering themselves for initial deposit into their own Current account
(7, 'Sophia', 'Hernandez', '7-CA004','456789'),
```

Triggers

```
CREATE TRIGGER trg_Update_Balance
ON CustomerSchema.Current_Transaction_History
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    -- Update the balance for the latest transaction
    WITH LatestBalances AS (
        SELECT
            I.Transaction_ID,
            I.CA_Number,
            I.Transaction_Type,
            I.Amount,
            ISNULL(
                (
                    SELECT TOP 1 Balance_After_Transaction
                    FROM CustomerSchema.Current_Transaction_History
                    WHERE CA_Number = I.CA_Number
                    AND Transaction_ID < I.Transaction_ID
                    ORDER BY Transaction_ID DESC
                ),
                0
            ) AS Previous_Balance
        FROM
            INSERTED I
    )
    UPDATE CTH
    SET
```

- Created triggers on the field "Balance_After_Transaction" in the Current and Saving Account Transaction History tables, to update the balance of the Customer account after every Credit and Debit Transaction.

```
CREATE TRIGGER trg_BeneficiaryID_AutoGenerate
ON CustomerSchema.Beneficiary
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @MaxID INT;
    DECLARE @NewBeneficiaryID VARCHAR(20);

    -- Get the highest numerical part of the Beneficiary_ID and increment it
    SELECT @MaxID = MAX(CAST(SUBSTRING(Beneficiary_ID, 5, LEN(Beneficiary_ID)) AS INT))
    FROM CustomerSchema.Beneficiary;

    -- If no data exists, start from 1, else increment the existing max ID
    IF @MaxID IS NULL
    BEGIN
        SET @MaxID = 1;
    END
    ELSE
    BEGIN
        SET @MaxID = @MaxID + 1;
    END

    -- Generate the new Beneficiary_ID in the format 'Ben-001', 'Ben-002', etc.
    SET @NewBeneficiaryID = 'Ben-' + RIGHT('000' + CAST(@MaxID AS VARCHAR(3)), 3);

    -- Ensure the new Beneficiary_ID is unique
```

- Created triggers on the field "Beneficiary_ID" in the Beneficiary table, to autogenerate the Beneficiary_ID incrementally.

Table Level Check Constraints

- Table-level check constraints can be applied to check data integrity rules on the data

- These constraints ensure that the data entered into the table meets specific conditions or requirements

- In our project, Table-level check constraints are applied to validate email, phone number, date of birth, first name, last name, address, start and end dates on bank products, IPAddress etc.

```
CREATE FUNCTION BankSchema.ValidateEmail(@Email NVARCHAR(50))
RETURNS BIT
AS
BEGIN
    RETURN CASE
        WHEN @Email LIKE '%@%.%' THEN 1  -- Basic email validation
        ELSE 0
    END
END;
GO


CREATE FUNCTION CustomerSchema.ValidateAmount(@Amount DECIMAL(10, 2))
RETURNS BIT
AS
BEGIN
    RETURN CASE
        WHEN @Amount > 0 THEN 1
        ELSE 0
    END
END;
GO
```

Table Level Check Constraints

```
CREATE FUNCTION CustomerSchema.ValidatePhoneNumber(@PhoneNumber NVARCHAR(10))
RETURNS BIT
AS
BEGIN
    RETURN CASE
        WHEN ISNUMERIC(@PhoneNumber) = 1 AND LEN(@PhoneNumber) = 10 THEN 1
        ELSE 0
    END
END;
```

```
CREATE FUNCTION CustomerSchema.ValidateCity(@City NVARCHAR(50))
RETURNS BIT
AS
BEGIN
    RETURN CASE
        WHEN @City LIKE '%[^a-zA-Z ]%' THEN 0
        ELSE 1
    END
END;
```

Encryption and Decryption

```sql
-- create certificate
CREATE CERTIFICATE SecurityCert
WITH SUBJECT = 'Encryption for Security Log Data';

-- Create a Symmetric Key:
CREATE SYMMETRIC KEY SecurityKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE SecurityCert;

-- Encrypt data in col temp_IPAddress
UPDATE CustomerSchema.Security_Log
SET Temp_IPAddress = EncryptByKey(Key_GUID('SecurityKey'), CONVERT(NVARCHAR(MAX), B.IPAddress))
FROM CustomerSchema.Security_Log AS SL
JOIN Temp_Security_Log_Backup AS B
ON SL.LogID = B.LogID;
```

LogID	Cl	Activity_Typ	Timestamp	Device_Info	IPAddress
9098AD-5533-4617-A823-1BB405DE77D5	8	Logout	4-11-22 20:15:00.000	Linux Laptop	^Yú÷ C ¥4u·z"á ªù# äÓ|z ... [84]
D16E4F-CF62-4C43-8CA8-E2C78C7FBA5A	5	Login	-12-05 09:30:00.000	MacBook Pro	^Yú÷ C ¥4u·z"á $Í ú D c%³1... [84]
349C36-6C4F-41D6-83EA-BC147D49D4B5	10	Two-Factor Auth	4-11-23 07:35:25.000	Samsung Galaxy S21	^Yú÷ C ¥4u·z"á ™ ÅP'®Ù }B ... [68]
97C2A8-6F15-463B-AAA8-863471B9B874	5	Purchase	24-11-22 18:30:15.000	MacBook Pro	^Yú÷ C ¥4u·z"á Éc[Q ß þÊD... [84]
A546E8-90B6-40A2-8015-6D865EAD3C8F	4	Password Change	4-11-21 14:45:30.000	iPhone 12	^Yú÷ C ¥4u·z"á ð]äUy ÝÄD ¶ ... [68]

- Data encryption and decryption in SQL are used to secure sensitive data stored in databases

- In BankHub, the key data is the IPAddress of the customer, which should not be visible to the employees. The IPAddress is added for the security_log table to check for any security discrepancies

- We used EncryptByKey() and DecryptByKey() functions

SQL View - Transaction Summary Overall

```sql
CREATE VIEW CustomerSchema.VW_Current_Transaction_History AS
SELECT
    t.Transaction_ID,
    t.CA_Number,
    t.Beneficiary_ID,
    b.Beneficiary_FName,
    b.Beneficiary_LName,
    t.Transaction_Date,
    t.Transaction_Type,
    t.Amount,
    t.Balance_After_Transaction,
    t.Description
FROM
    CustomerSchema.Current_Transaction_History t
LEFT JOIN CustomerSchema.Beneficiary b
    ON t.Beneficiary_ID = b.Beneficiary_ID;

select * from CustomerSchema.VW_Current_Transaction_History;
```

Results | Messages

	Transaction_ID	CA_Number	Beneficiary_ID	Beneficiary_FName	Beneficiary_LName	Transaction_Date	Transaction_Type	Amount	Balance_After_Transaction	Description
102	27	10-CA009	Ben-027	Ethan	Rodriguez	2024-12-11 10:00:00.000	Credit	2000....	2000.00	Deposit into own account
103	28	11-CA001	Ben-028	Emma	Lopez	2024-06-11 09:00:00.000	Credit	2000....	2000.00	Deposit into own account
104	29	12-CA007	Ben-029	James	Taylor	2024-06-15 09:00:00.000	Credit	2000....	2000.00	Deposit into own account
105	5	7-CA004	Ben-030	Liam	Garcia	2024-11-01 10:00:00.000	Debit	200.00	800.00	Transferring to Liam
106	14	7-CA004	Ben-030	Liam	Garcia	2024-10-01 10:00:00.000	Credit	200.00	1000.00	Transferring to Liam
107	15	7-CA004	Ben-030	Liam	Garcia	2024-10-01 10:00:00.000	Debit	200.00	800.00	Transferring to Liam
108	16	7-CA004	Ben-030	Liam	Garcia	2024-10-01 10:00:00.000	Debit	200.00	600.00	Transferring to Liam
109	17	7-CA004	Ben-030	Liam	Garcia	2024-10-01 10:00:00.000	Credit	200.00	800.00	Transferring to Liam
110	18	7-CA004	Ben-030	Liam	Garcia	2024-10-01 10:00:00.000	Credit	2200....	3000.00	Transferring to Liam
111	38	4-CA003	Ben-041	Michael	Johnson	2024-01-01 10:00:00.000	Credit	2000....	2000.00	Deposit into own account
112	41	4-CA003	Ben-041	Michael	Johnson	2024-01-15 11:00:00.000	Credit	500.00	1800.00	Self Deposit
113	43	4-CA003	Ben-041	Michael	Johnson	2024-01-15 11:00:00.000	Credit	770.00	2070.00	Self Deposit
114	45	4-CA003	Ben-041	Michael	Johnson	2024-02-01 11:00:00.000	Credit	5000....	6950.00	Salary
115	50	4-CA003	Ben-041	Michael	Johnson	2024-02-17 11:00:00.000	Credit	500.00	5176.00	From Liam
116	57	4-CA003	Ben-041	Michael	Johnson	2024-03-21 11:00:00.000	Credit	550.00	8669.00	From Liam
117	58	4-CA003	Ben-041	Michael	Johnson	2024-03-23 11:00:00.000	Credit	1200....	9869.00	Bonus
118	53	4-CA003	Ben-041	Michael	Johnson	2024-03-01 11:00:00.000	Credit	5500....	9478.00	Salary
119	61	4-CA003	Ben-041	Michael	Johnson	2024-04-01 11:00:00.000	Credit	6000....	8559.00	Salary
120	69	4-CA003	Ben-041	Michael	Johnson	2024-04-14 11:00:00.000	Credit	600.00	7053.00	From Someone
121	74	4-CA003	Ben-041	Michael	Johnson	2024-05-01 11:00:00.000	Credit	6000....	8413.00	Salary
122	78	4-CA003	Ben-041	Michael	Johnson	2024-05-07 11:00:00.000	Credit	3000....	10723.00	Bonus on shares
123	114	4-CA003	Ben-041	Michael	Johnson	2024-05-07 14:25:00.000	Debit	150.75	10371.75	Fuel
124	124	4-CA003	Ben-041	Michael	Johnson	2024-05-13 13:40:00.000	Debit	245.60	11305.80	Shopping
125	121	4-CA003	Ben-041	Michael	Johnson	2024-05-10 11:10:00.000	Credit	2300....	12461.85	Bonus
126	127	4-CA003	Ben-041	Michael	Johnson	2024-05-16 14:25:00.000	Debit	77.45	10137.65	Fuel
127	153	4-CA003	Ben-041	Michael	Johnson	2024-06-01 09:00:00.000	Credit	6000....	15239.10	Salary
128	130	4-CA003	Ben-041	Michael	Johnson	2024-05-28 15:30:00.000	Debit	88.10	10109.10	Dining
129	158	4-CA003	Ben-041	Michael	Johnson	2024-06-15 09:30:00.000	Credit	1200....	14707.90	Bonus
130	162	4-CA003	Ben-041	Michael	Johnson	2024-06-28 16:55:00.000	Credit	450.00	13736.65	Gift
131	163	4-CA003	Ben-041	Michael	Johnson	2024-07-01 09:00:00.000	Credit	6000....	19736.65	Salary
132	167	4-CA003	Ben-041	Michael	Johnson	2024-07-15 08:45:00.000	Credit	1250....	19495.40	Bonus
133	172	4-CA003	Ben-041	Michael	Johnson	2024-07-30 16:55:00.000	Credit	450.00	18154.05	Freelance Income
134	173	4-CA003	Ben-041	Michael	Johnson	2024-08-01 09:00:00.000	Credit	6000....	24154.05	Salary

Query executed successfully. | boyce.coe.neu.edu (15.0 RTM) | INFO6210 (87) | BankHub | 00:00:00 | 148 rows

This view provides a comprehensive and user-friendly summary of current transactions along with optional beneficiary details for further analysis or reporting.

SQL View - Monthly Spendings

Results | Messages

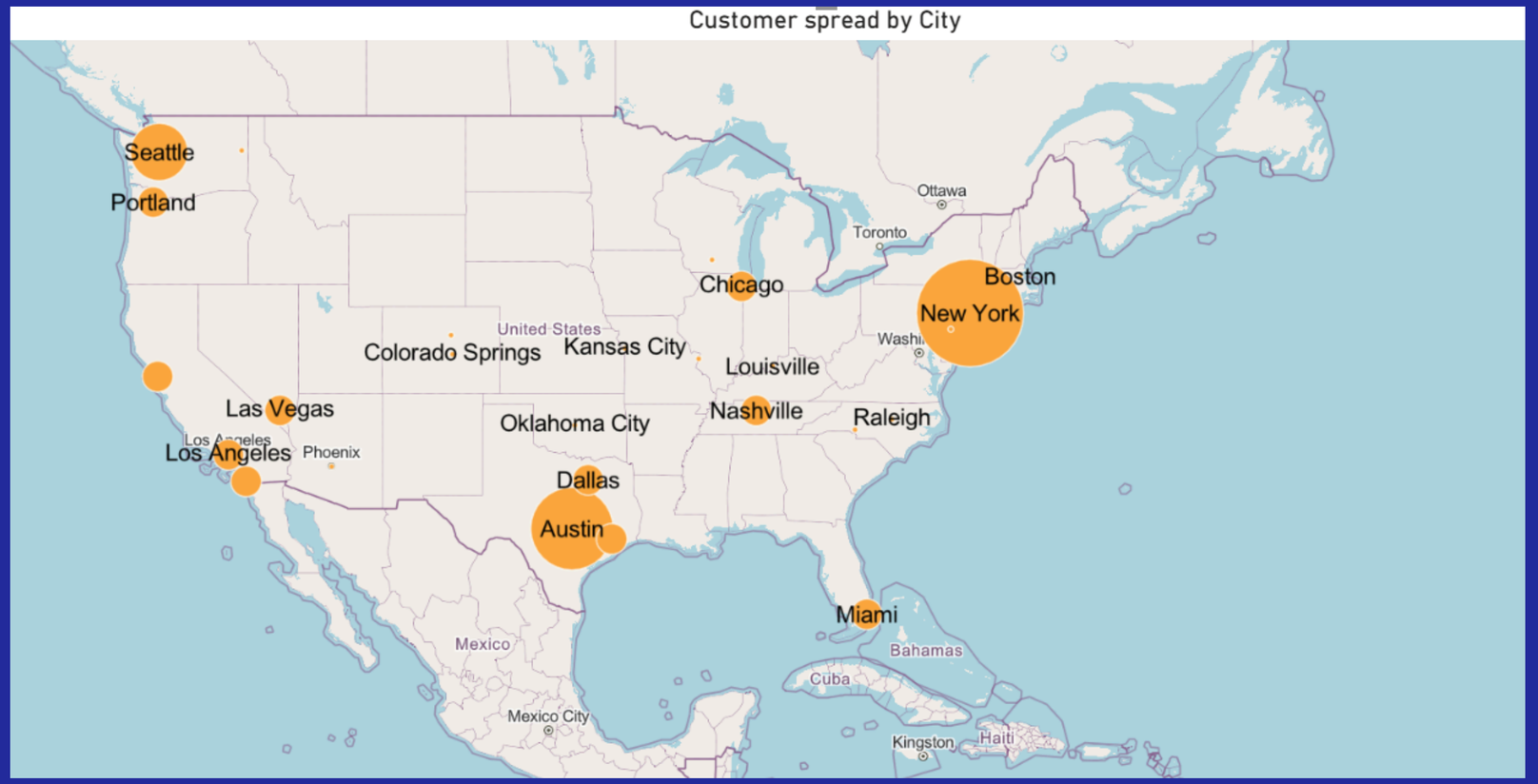
	CA_Number	Transaction_Month	Avg_Debit
1	4-CA003	2024-01	330.000000
2	4-CA003	2024-02	578.666666
3	4-CA003	2024-03	1733.800000
4	4-CA003	2024-04	613.272727
5	4-CA003	2024-05	286.700000
6	4-CA003	2024-06	450.350000
7	4-CA003	2024-07	468.942857
8	4-CA003	2024-08	491.900000
9	4-CA003	2024-09	490.806250
10	4-CA003	2024-10	470.371428
11	7-CA004	2024-10	200.000000
12	12-CA007	2024-11	1000.000000
13	3-CA001	2024-11	275.000000
14	4-CA003	2024-11	497.478571
15	5-CA006	2024-11	330.000000
16	7-CA004	2024-11	200.000000
17	8-CA001	2024-11	530.000000
18	9-CA005	2024-11	500.000000
19	4-CA003	2024-12	473.200000

```sql
CREATE VIEW CustomerSchema.VW_Monthly_Avg_Debit AS
SELECT
    t.CA_Number,
    FORMAT(t.Transaction_Date, 'yyyy-MM') AS Transaction_Month,
    AVG(t.Amount) AS Avg_Debit
FROM
    CustomerSchema.Current_Transaction_History t
WHERE
    t.Transaction_Type = 'Debit'
GROUP BY
    t.CA_Number, FORMAT(t.Transaction_Date, 'yyyy-MM');

SELECT * FROM CustomerSchema.VW_Monthly_Avg_Debit;
```

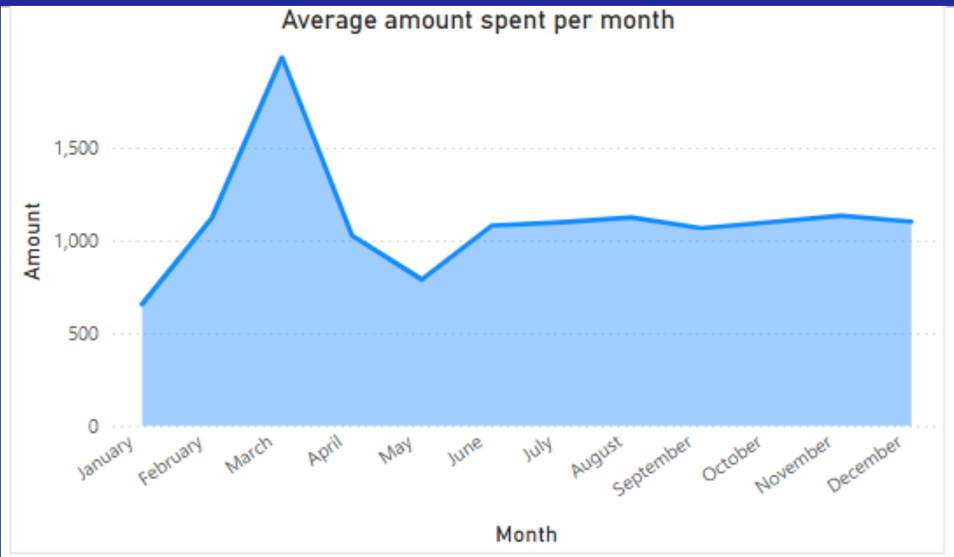
- This view enables tracking and analysis of average monthly debit trends for each account
- Aids in financial insights and decision-making

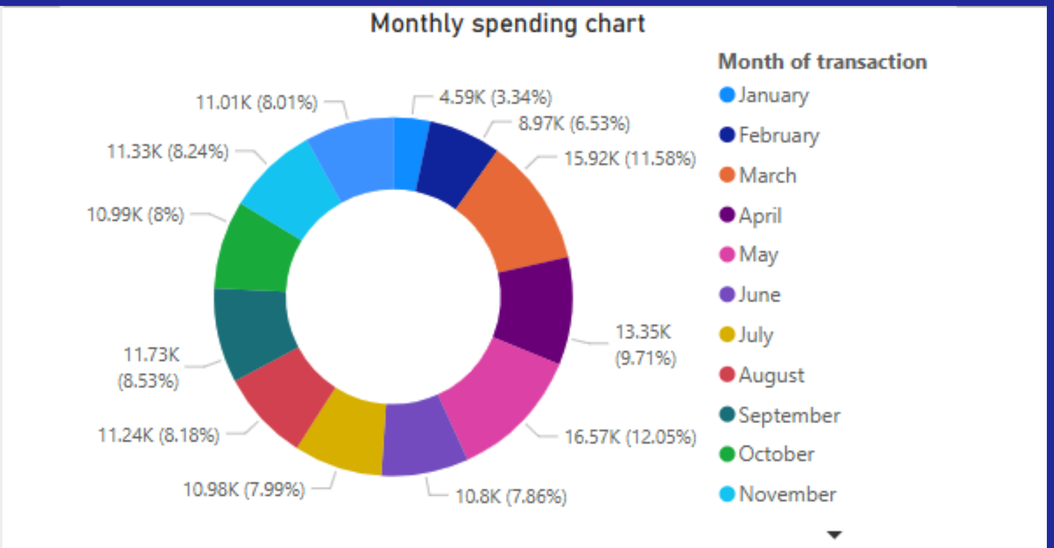
Analysis and Visualization with Power BI

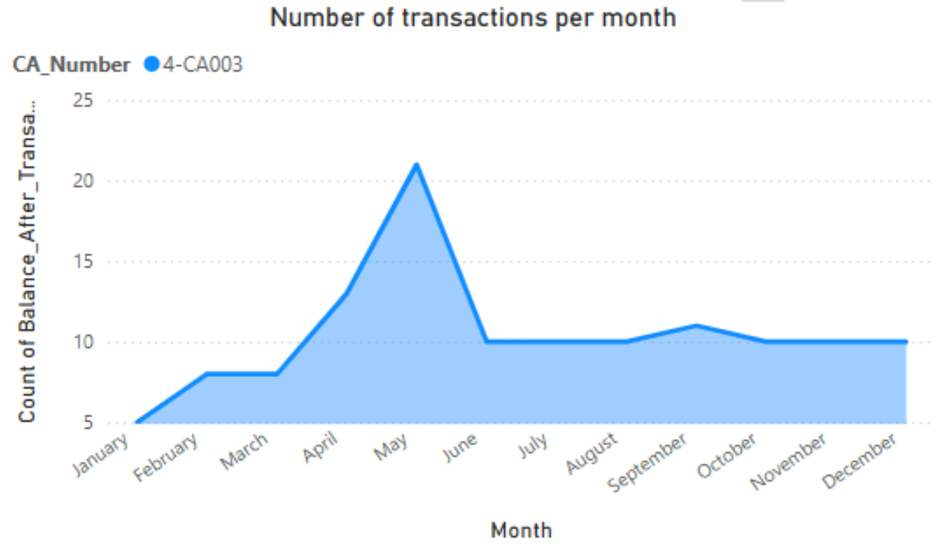


This visualization shows the customer concentration in various cities

Customer Spending Analysis by Month







29,481.00	Qtr 1	4-CA003
Sum of Amount	Quarter	CA_Number
40,722.35	Qtr 2	4-CA003
Sum of Amount	Quarter	CA_Number
33,952.35	Qtr 3	4-CA003
Sum of Amount	Quarter	CA_Number
33,337.35	Qtr 4	4-CA003
Sum of Amount	Quarter	CA_Number

This visual helps a customer access his monthly spending patterns, number of transactions his account has and the distribution of his spending per month

Conclusion

- The centralized relational banking database addresses fragmented financial data, enabling seamless customer access, data aggregation for banks, and government oversight
- Designed in 3NF, it ensures data integrity, scalability, and robustness for modern banking needs
- Demonstrated potential to enhance transparency, compliance, and operational efficiency using artificial data
- Future steps: Integrate real-world data and refine compliance checks to validate and stress-test the system for broader adoption

Questions ?!

Feel free to ask all your doubts or if you need further clarifications

Thank You!

GROUP 11