Project Title: Data Exploration with Azure SQL Database – Customer, Account, and Loan Feeds

1. Setting Up Azure SQL Database

Create an Azure SQL Database in the Azure portal.
Define a new database and server.
Name the database `CustomerAccountLoanDB`.

Step 1: Open azure portal
Step 2: Go to search and type “SQL database” and select “create”
Step 3: **Basics**

- Resource group - Create New - “projectweek2”,
- Database name: “CustomerAccountLoanDB”
- Server: - Create New
- Server name: “shreeserverproject2”
- Location: “Central US”
- Authentication method: select “ SQL Authentication”
- Server admin login: “shreeserverproject2”
- Password: ”****”
- Confirm Password: “*****” and select “OK”
- Back to Basics page
- Want to use SQL elastic pool?: “NO”
- Workload environment: “Production”
- Compute + storage: select “configure database”
- Service tier: “Basics (for less demanding workloads)” and select “Apply”
- Back to Basics page
- Backup storage redundancy: “Locally-redundant backup storage”
- Next Networking

Step 4: **Networking**

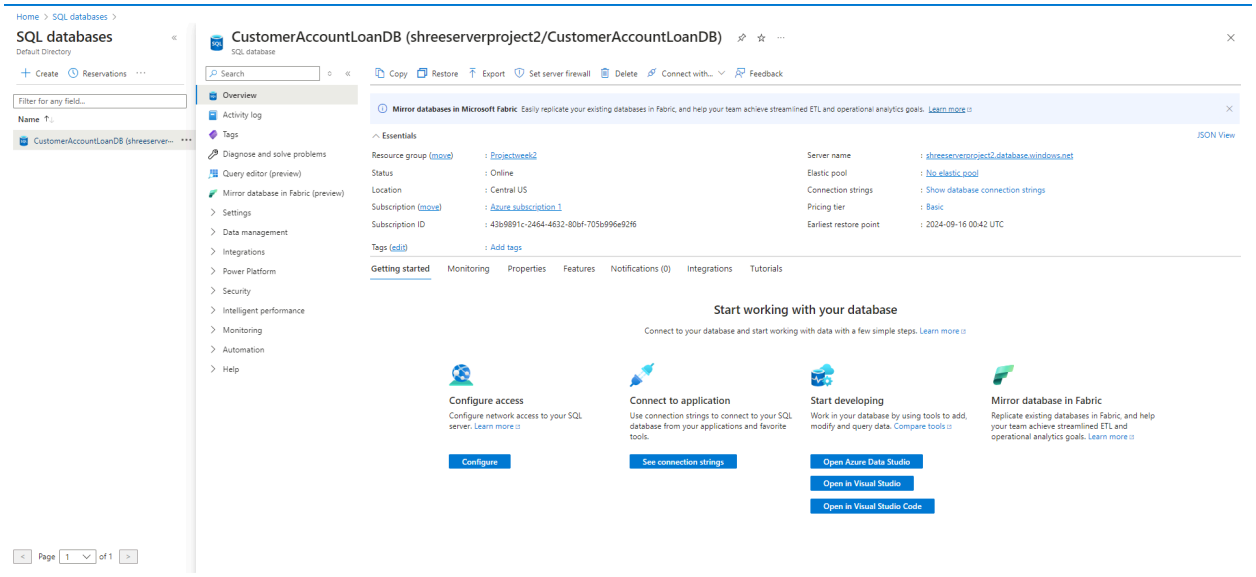
- Connectivity method: “Public Endpoint”
- Allow Azure services and resources to access this server: “Yes”
- Add current client IP address: “Yes”
- Connection policy: “Default - Uses Redirect policy for all client connections originating inside of Azure (except Private Endpoint connections) and Proxy for all client connections originating outside Azure”
- Next Security (no changes - keep it default)
- Next Additional settings

Step 5: **Additional settings**

- Use existing data: “Sample”

- Next Tags (no changes - keep it default)
- Next Review + create
- Select “create”

Refer the below screenshot once the SQL database is created.



2. Data Organization

Step 1: Select “Query Editor” in SQL dashboard: CustomerAccountLoanDB

- User name: “shreeserverproject2”
- Password: “*****”

Step 2: Select “Tables”

Step 3: In the query dashboard create a customer table.

- Create a customer table:

```
CREATE TABLE customers (
            customer_id INT PRIMARY KEY,
            first_name VARCHAR(50),
            last_name VARCHAR(50),
            address VARCHAR(100),
            city VARCHAR(50),
            state VARCHAR(50),
            zip VARCHAR(20)
        );
```

- Create a Account table:

```
CREATE TABLE accounts (
            account_id INT PRIMARY KEY,
            customer_id INT,
            account_type VARCHAR(50),
            balance DECIMAL(10, 2),
            FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
        );
```

- Create Transaction table:

```
 CREATE TABLE transactions (
            transaction_id INT PRIMARY KEY,
            account_id INT,
            transaction_date DATE,
            transaction_amount DECIMAL(10, 2),
            transaction_type VARCHAR(50),
            FOREIGN KEY (account_id) REFERENCES accounts(account_id)
        );
```

- Create Loan Table:

```
 CREATE TABLE loans (
            loan_id INT PRIMARY KEY,
            customer_id INT,
            loan_amount DECIMAL(10, 2),
            interest_rate DECIMAL(5, 2),
            loan_term INT,
            FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
        );
```

- Create Loan table payment:

```
 CREATE TABLE loan_payments (
            payment_id INT PRIMARY KEY,
            loan_id INT,
            payment_date DATE,
            payment_amount DECIMAL(10, 2),
            FOREIGN KEY (loan_id) REFERENCES loans(loan_id)
        );
```

3. Data Insertion

Step 1: Insert sample data into all Tables(customer, account, transaction, loan, loan payment).

- Insert 10 values into the customer table.

```
INSERT INTO customers (customer_id, first_name, last_name, address, city, state, zip) VALUES
(1, 'John', 'Doe', '123 Elm St', 'Springfield', 'IL', '62701'),
(2, 'Jane', 'Smith', '456 Oak St', 'Chicago', 'IL', '60614'),
(3, 'Emily', 'Johnson', '789 Pine St', 'Dallas', 'TX', '75201'),
(4, 'Michael', 'Williams', '101 Maple St', 'Seattle', 'WA', '98101'),
(5, 'Sarah', 'Brown', '202 Birch St', 'New York', 'NY', '10001'),
(6, 'David', 'Jones', '303 Cedar St', 'Los Angeles', 'CA', '90001'),
(7, 'Laura', 'Garcia', '404 Willow St', 'San Francisco', 'CA', '94101'),
(8, 'James', 'Martinez', '505 Redwood St', 'Houston', 'TX', '77001'),
(9, 'Olivia', 'Davis', '606 Fir St', 'Boston', 'MA', '02101'),
(10, 'Daniel', 'Rodriguez', '707 Spruce St', 'Philadelphia', 'PA', '19101');
```

- Insert 10 values into the Account table.

```
INSERT INTO accounts (account_id, customer_id, account_type, balance) VALUES
(1, 1, 'Checking', 1000.00),
(2, 1, 'Savings', 5000.00),
(3, 2, 'Checking', 1500.00),
(4, 2, 'Investment', 7500.00),
(5, 3, 'Savings', 2000.00),
(6, 4, 'Checking', 3000.00),
(7, 5, 'Checking', 2500.00),
(8, 6, 'Savings', 6000.00),
(9, 7, 'Investment', 8000.00),
(10, 8, 'Checking', 1200.00);
```

- Insert 10 values into the Transaction table.

```
INSERT INTO transactions (transaction_id, account_id, transaction_date, transaction_amount, transaction_type) VALUES
(1, 1, '2024-09-01', 200.00, 'Deposit'),
(2, 1, '2024-09-03', -100.00, 'Withdrawal'),
```

```
(3, 2, '2024-09-02', 300.00, 'Deposit'),
(4, 2, '2024-09-04', -50.00, 'Withdrawal'),
(5, 3, '2024-09-05', 150.00, 'Deposit'),
(6, 4, '2024-09-06', -200.00, 'Withdrawal'),
(7, 5, '2024-09-07', 250.00, 'Deposit'),
(8, 6, '2024-09-08', -300.00, 'Withdrawal'),
(9, 7, '2024-09-09', 400.00, 'Deposit'),
(10, 8, '2024-09-10', -150.00, 'Withdrawal');
```

- Insert 10 values into the loans table.

```
INSERT INTO loans (loan_id, customer_id, loan_amount, interest_rate,
loan_term) VALUES
(1, 1, 5000.00, 3.50, 12),
(2, 2, 7500.00, 4.00, 24),
(3, 3, 6000.00, 3.75, 18),
(4, 4, 10000.00, 4.25, 36),
(5, 5, 12000.00, 4.50, 48),
(6, 6, 8000.00, 3.90, 24),
(7, 7, 9500.00, 4.10, 30),
(8, 8, 11000.00, 4.00, 42),
(9, 9, 13000.00, 4.20, 54),
(10, 10, 7000.00, 3.85, 20);
```

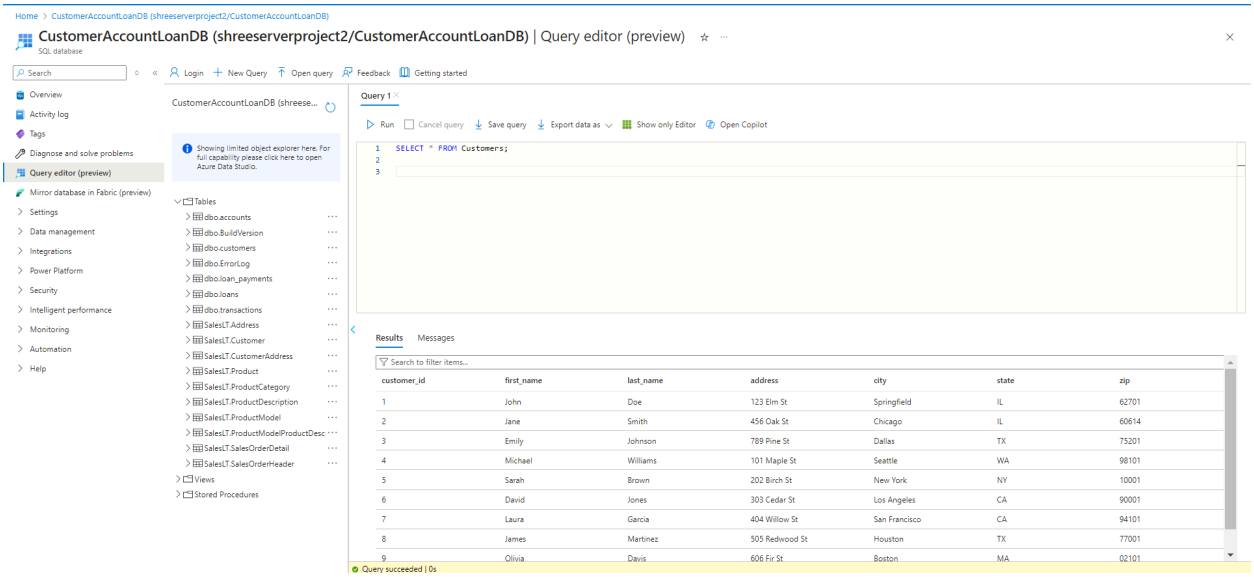
- Insert 10 values into the loan_payments table.

```
INSERT INTO loan_payments (payment_id, loan_id, payment_date, payment_amount)
VALUES
(1, 1, '2024-01-15', 250.00),
(2, 1, '2024-02-15', 200.00),
(3, 2, '2024-01-20', 150.00),
(4, 2, '2024-02-20', 400.00),
(5, 3, '2024-01-25', 400.00),
(6, 3, '2024-02-25', 200.00),
(7, 4, '2024-03-01', 250.00),
(8, 4, '2024-04-01', 200.00),
(9, 5, '2024-05-10', 400.00),
(10, 5, '2024-06-10', 400.00);
```

4. Data Exploration:

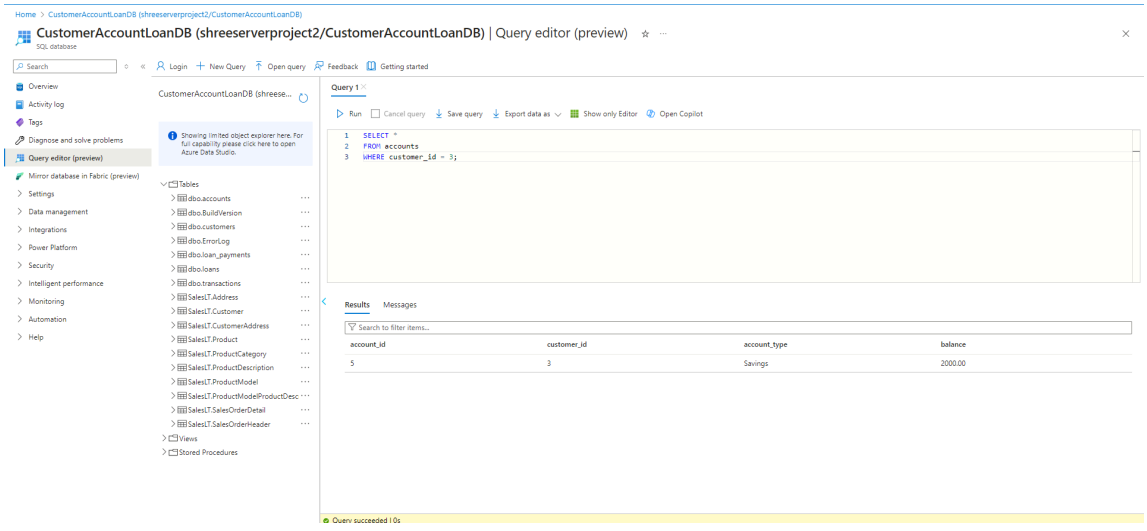
- Write a query to retrieve all customer information.

```
SELECT * FROM Customers;
```



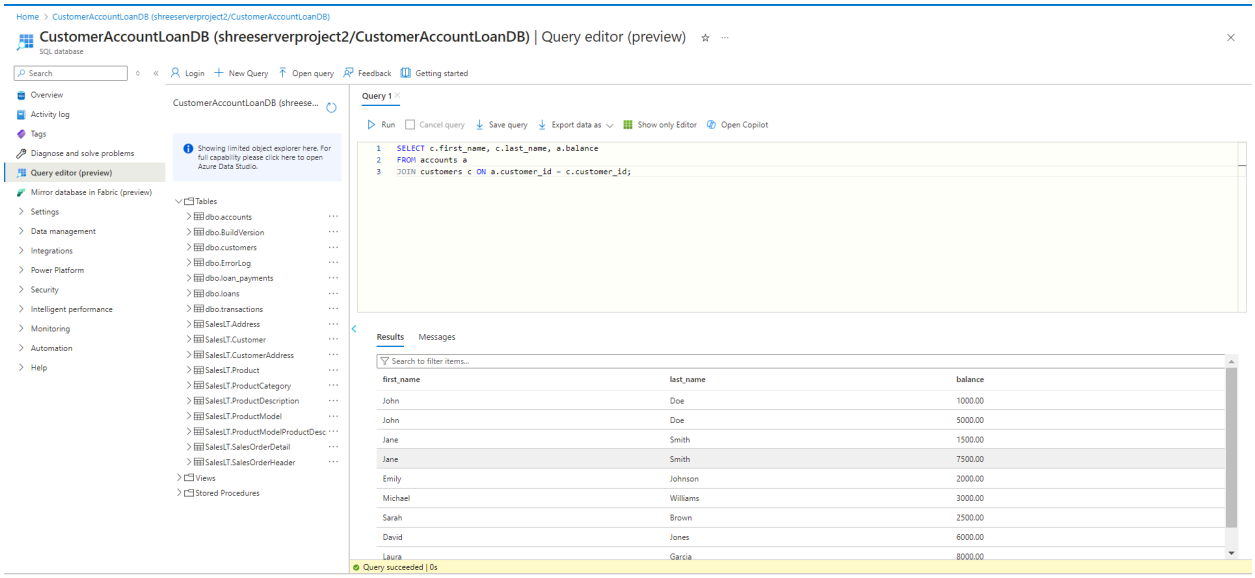
- Query accounts for a specific customer.

```
SELECT *
FROM accounts
WHERE customer_id = 3;
```



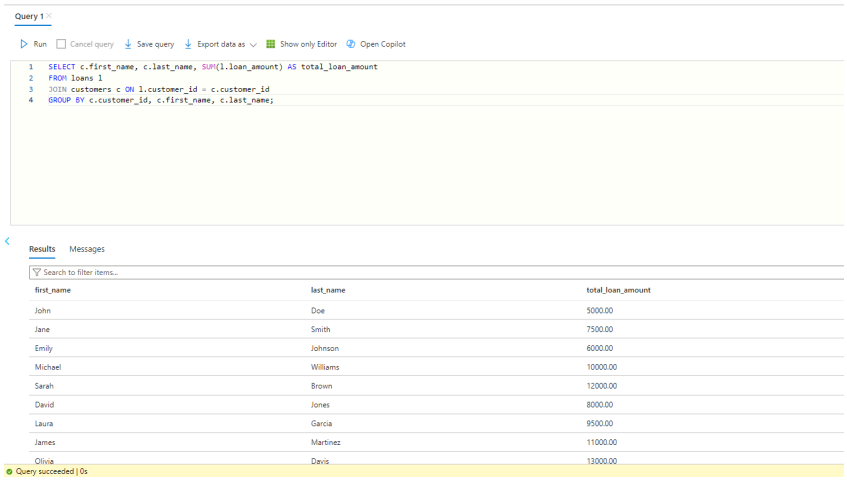
- Find the customer name and account balance for each account

```
SELECT c.first_name, c.last_name, a.balance
FROM accounts a
JOIN customers c ON a.customer_id = c.customer_id;
```



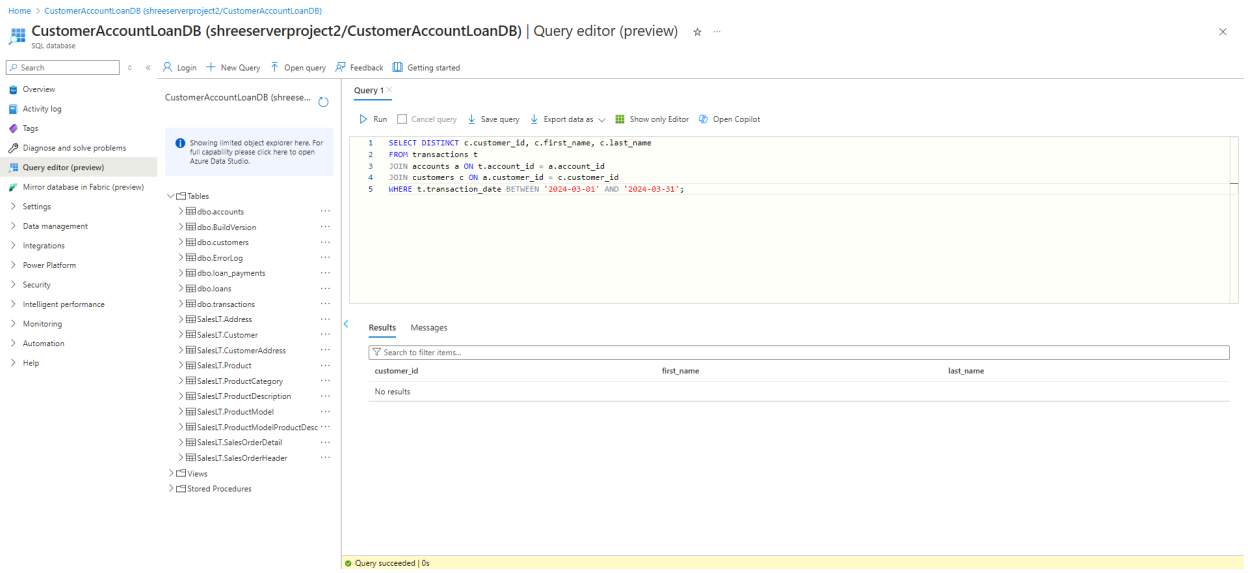
- Analyze customer loan balances

```
SELECT c.first_name, c.last_name, SUM(l.loan_amount) AS total_loan_amount
FROM loans l
JOIN customers c ON l.customer_id = c.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name;
```



- List all customers who have made a transaction in the 2024-03

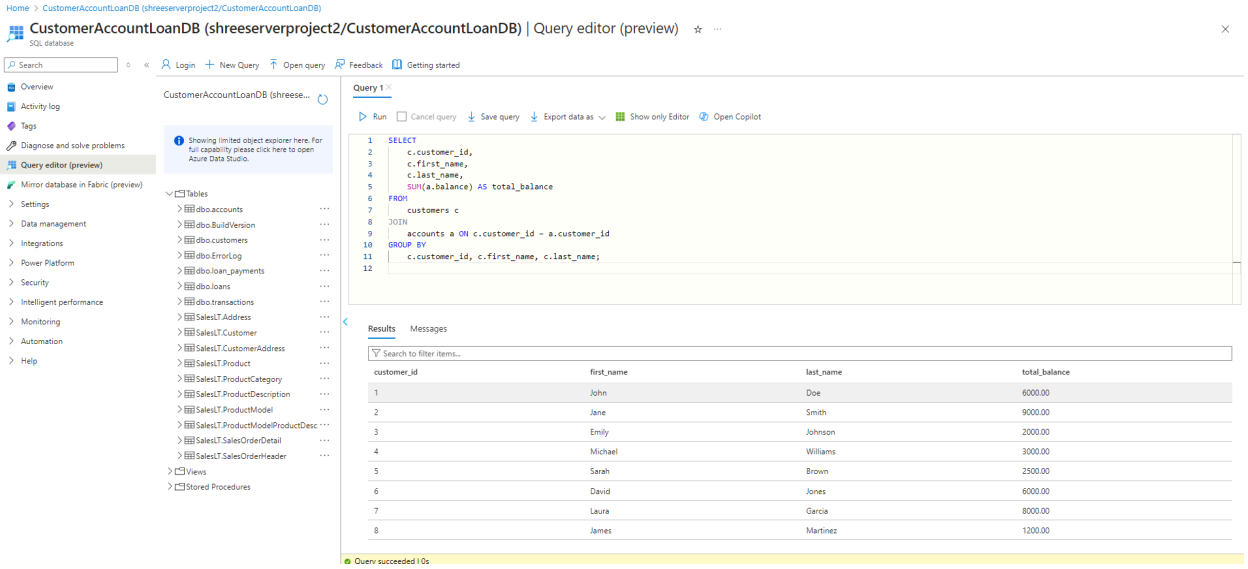
```sql
SELECT DISTINCT c.customer_id, c.first_name, c.last_name
FROM transactions t
JOIN accounts a ON t.account_id = a.account_id
JOIN customers c ON a.customer_id = c.customer_id
WHERE t.transaction_date BETWEEN '2024-03-01' AND '2024-03-31';
```



5. Aggregation and Insights

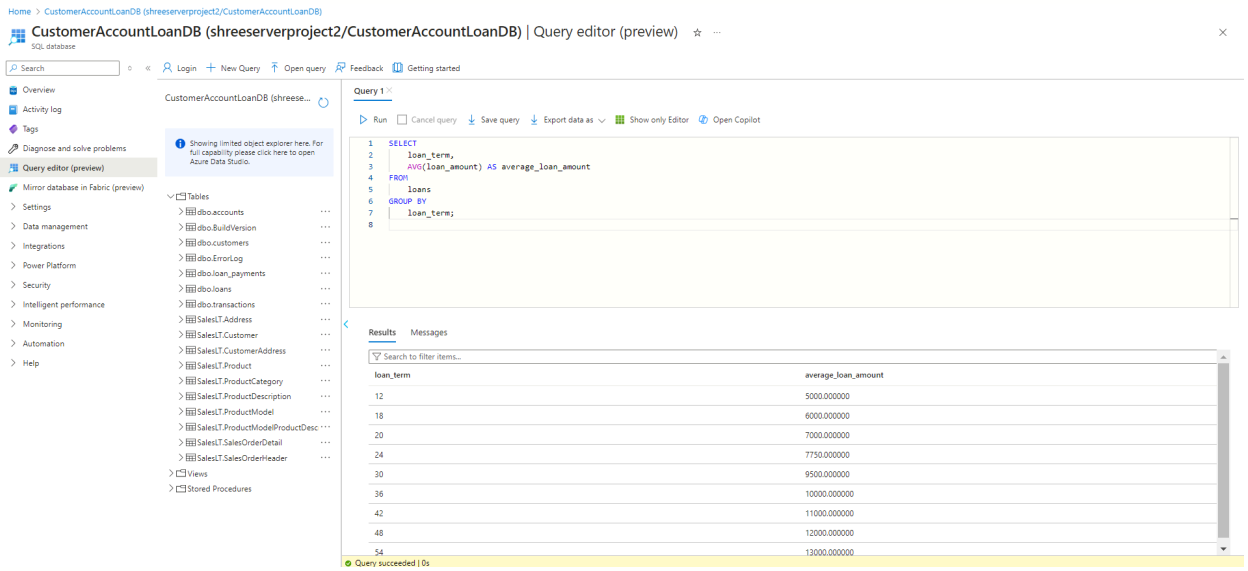
- Calculate the total balance across all accounts for each customer:

```sql
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    SUM(a.balance) AS total_balance
FROM
    customers c
JOIN
    accounts a ON c.customer_id = a.customer_id
GROUP BY
    c.customer_id, c.first_name, c.last_name;
```



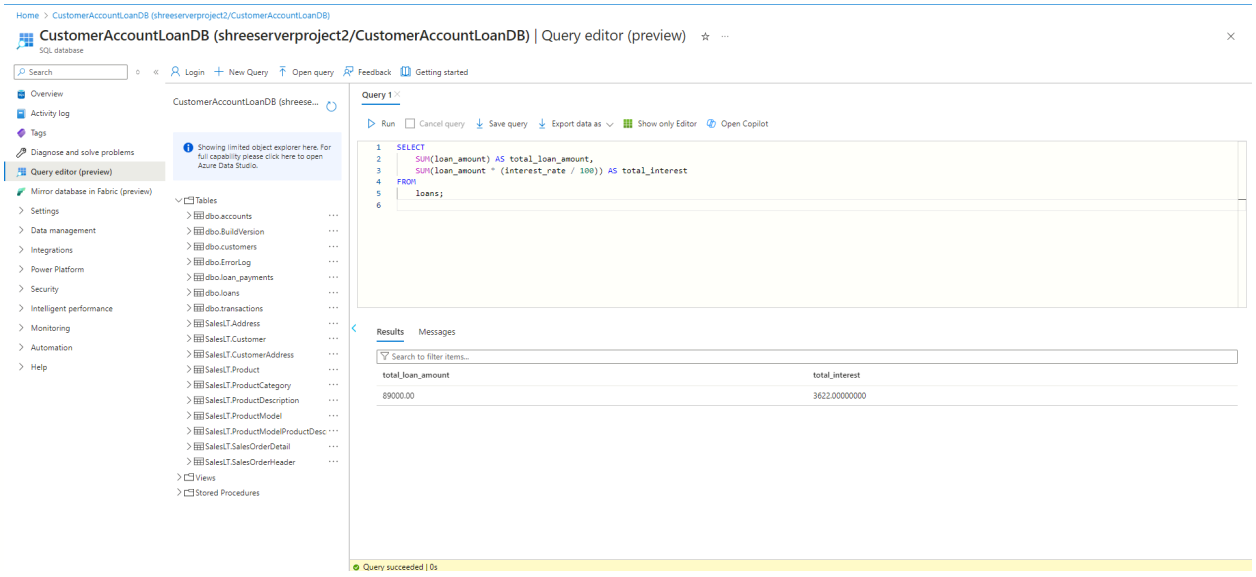
- Calculate the average loan amount for each loan term:

```
SELECT
    loan_term,
    AVG(loan_amount) AS average_loan_amount
FROM
    loans
GROUP BY
    loan_term;
```



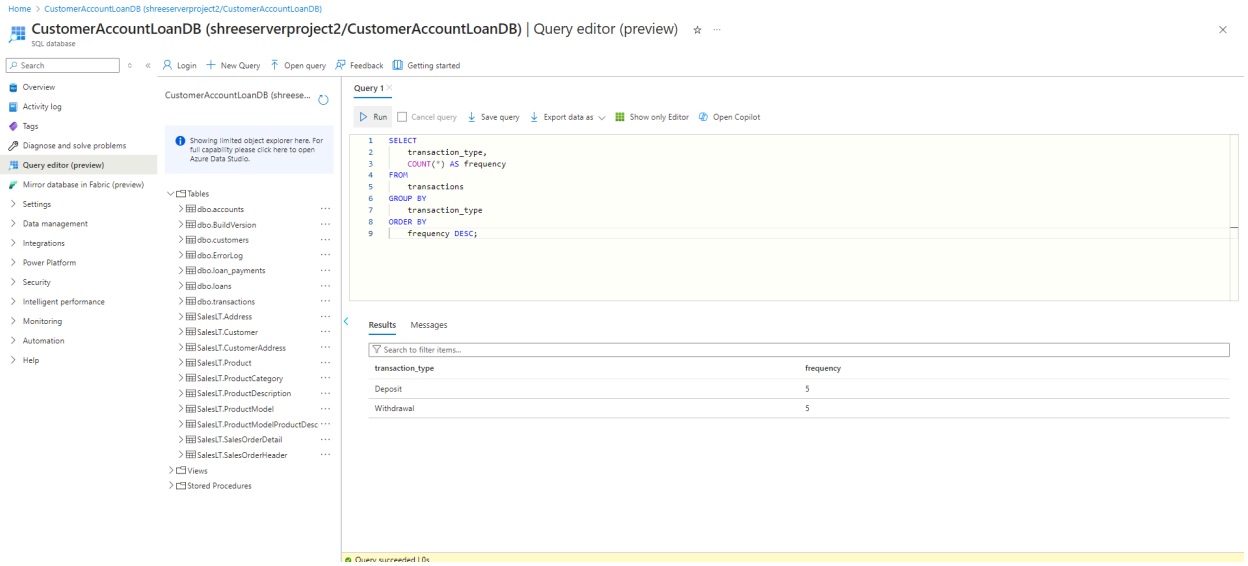
- Find the total loan amount and interest across all loans:

```sql
SELECT
    SUM(loan_amount) AS total_loan_amount,
    SUM(loan_amount * (interest_rate / 100)) AS total_interest
FROM
    loans;
```



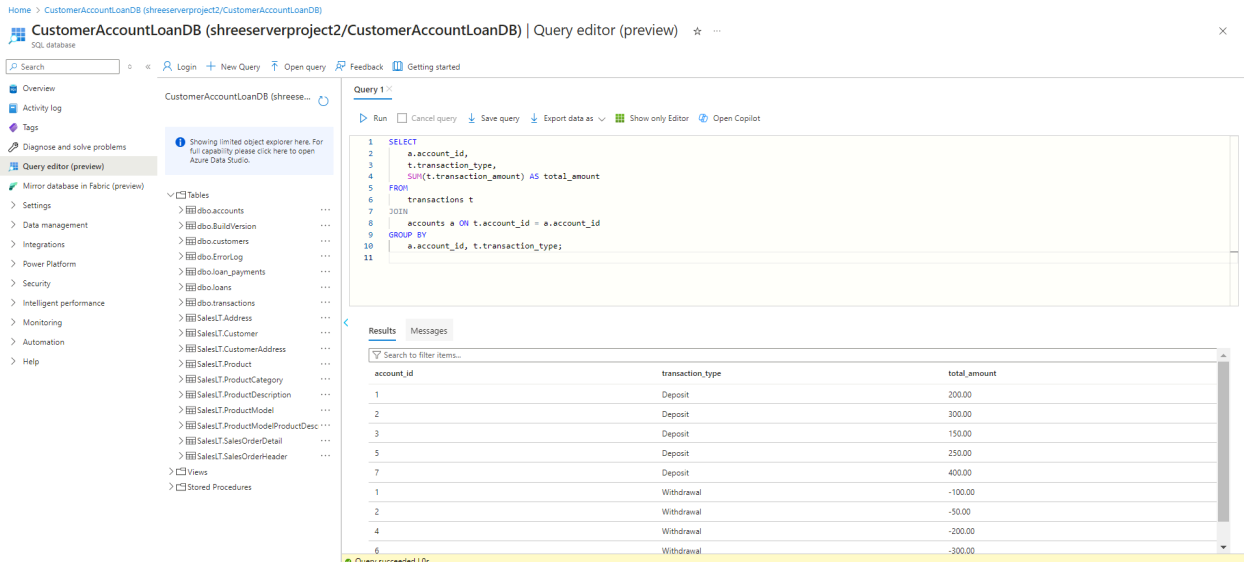
- Find the most frequent transaction type

```sql
SELECT
    transaction_type,
    COUNT(*) AS frequency
FROM
    transactions
GROUP BY
    transaction_type
ORDER BY
    frequency DESC;
```



- Analyze transactions by account and transaction type:

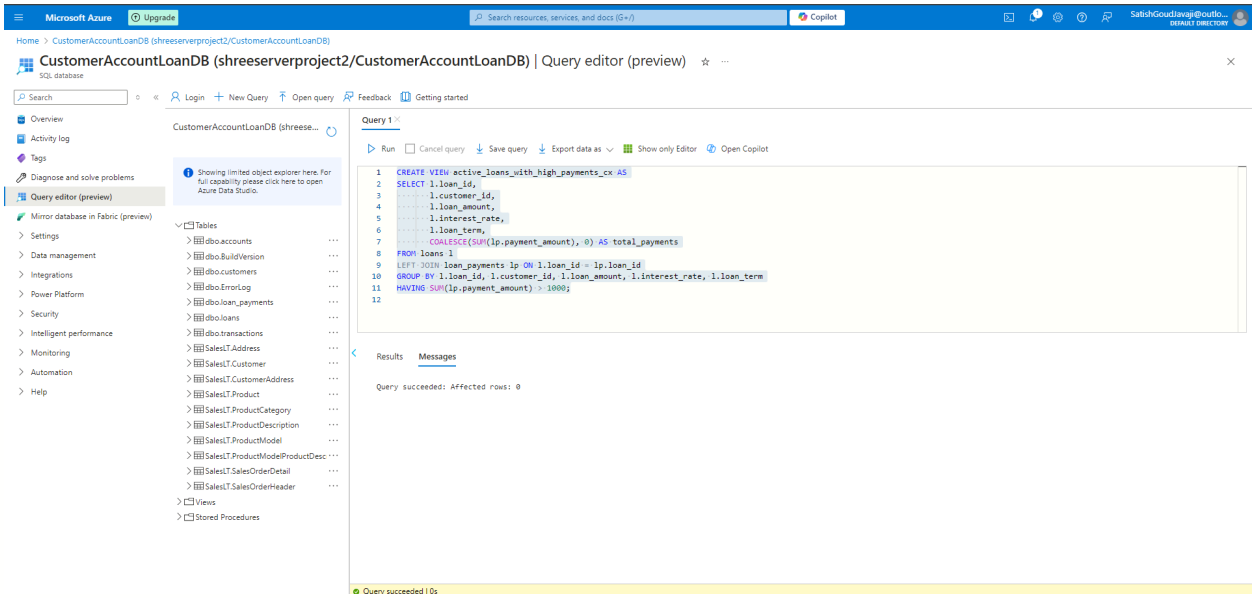
```
SELECT
    a.account_id,
    t.transaction_type,
    SUM(t.transaction_amount) AS total_amount
FROM
    transactions t
JOIN
    accounts a ON t.account_id = a.account_id
GROUP BY
    a.account_id, t.transaction_type;
```

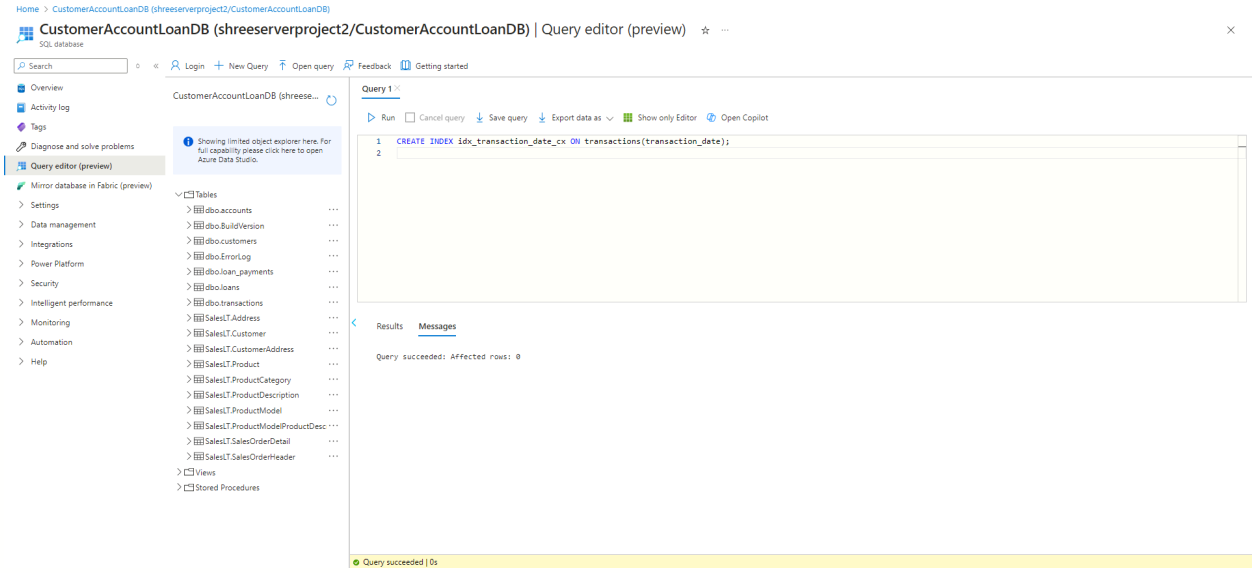


6. Advanced Analysis

- Create a view of active loans with payments greater than $1000:

```
CREATE VIEW active_loans_with_high_payments_cx AS
SELECT l.loan_id,
       l.customer_id,
       l.loan_amount,
       l.interest_rate,
       l.loan_term,
       COALESCE(SUM(lp.payment_amount), 0) AS total_payments
FROM loans l
LEFT JOIN loan_payments lp ON l.loan_id = lp.loan_id
GROUP BY l.loan_id, l.customer_id, l.loan_amount, l.interest_rate,
l.loan_term
HAVING SUM(lp.payment_amount) > 1000;
```



- Create an index on `transaction_date` in the `transactions` table for performance optimization:

```
CREATE INDEX idx_transaction_date_cx ON transactions(transaction_date);
```