# Data Exploration with Azure SQL Database – Customer, Account, and Loan Feeds

# 1. Setting Up Azure SQL Database

In this section, we will create an Azure SQL Database using the Azure portal. Follow the steps below to set up a new database and server.

## Step 1: Open the Azure portal to create a database and for working on it.

## Step 2: Go to the search bar for 'SQL database', and select 'Create'.

## Step 3: Basics

* a. Resource group: Create a new group named ‘resourcegrp1’.
* b. Database name: Enter ‘db1’.
* c. Server: Create a new server with the following details:
* - Server name: “dbserver1”.
* - Location: 'Central US'.
* - Authentication method: Select 'SQL Authentication'.
* - Server admin login: Enter 'bellamkonda'.
* - Password: Enter a secure password.
* - Confirm Password: Re-enter the password, then select 'OK'.
* d. Back to the Basics page:
* - Want to use SQL elastic pool?: Select 'No'.
* - Workload environment: Choose 'Production'.
* - Compute + storage: Select 'Configure database'.
* - Service tier: Select 'Basic (for less demanding workloads)', then click 'Apply'.
* - Backup storage redundancy: Choose 'Locally-redundant backup storage'.

## Step 4: Networking

* a. Connectivity method: Select 'Public Endpoint'.
* b. Allow Azure services and resources to access this server: Choose 'Yes'.
* c. Add current client IP address: Select 'Yes'.
* d. Connection policy: Choose '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.

## Step 5: Security

* No changes needed - keep the default settings.

## Step 6: Additional Settings

* a. Use existing data: Select 'Sample'.

## Step 7: Tags

* No changes needed - keep the default settings.

## Step 8: Review + Create

* a. Review your settings, then select 'Create'.

Congratulations! You just created a Database in Azure Portal.

Now go to Query Editor to perform some basic operations like Creating, Inserting and Querying in database.

## Step 2: Data Organization

Here, in this step we organize data by creating tables as per our requirement.

#### Table 1: - Customer Feed:

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)  
);

#### Table 2: - Account Feed:

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)  
);

#### Table 3: - Transaction Feed:

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)  
);

#### Table 4: - Loan Feed:

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)  
);

#### Table 5: - Loan Payment Feed:

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)  
);

## Step 3: Data Insertion

* Populate tables with sample data using `INSERT INTO` statements for each table.
* Ensure data consistency and relationships, ensuring each foreign key points to valid primary keys.
* Historical Data is available in many open resources like Kaggle. By using that we can populate the data to our tables.

**Insert into customers Table:**

INSERT INTO customers (customer\_id, first\_name, last\_name, address, city, state, zip) VALUES

(1, 'Rajesh', 'Sharma', '123 MG Road', 'Mumbai', 'Maharashtra', '400001'),

(2, 'Anjali', 'Verma', '456 Sector 5', 'Gurgaon', 'Haryana', '122001'),

(3, 'Ravi', 'Kumar', '789 Green Park', 'Delhi', 'Delhi', '110016'),

(4, 'Priya', 'Mehta', '321 Ring Road', 'Ahmedabad', 'Gujarat', '380001'),

(5, 'Arjun', 'Desai', '654 Hill View', 'Bangalore', 'Karnataka', '560001'),

(6, 'Sneha', 'Gupta', '111 Park Street', 'Kolkata', 'West Bengal', '700001'),

(7, 'Mohan', 'Reddy', '222 Jubilee Hills', 'Hyderabad', 'Telangana', '500033'),

(8, 'Neha', 'Patel', '333 SG Highway', 'Surat', 'Gujarat', '395007'),

(9, 'Karan', 'Malhotra', '444 Marine Drive', 'Mumbai', 'Maharashtra', '400002'),

(10, 'Aditi', 'Nair', '555 Palm Beach Road', 'Navi Mumbai', 'Maharashtra', '400703');

A screenshot of a computer

Description automatically generated

**Insert into accounts Table:**

INSERT INTO accounts (account\_id, customer\_id, account\_type, balance) VALUES

(1, 1, 'Savings', 50000.00),

(2, 2, 'Current', 150000.00),

(3, 3, 'Savings', 75000.00),

(4, 4, 'Current', 50000.00),

(5, 5, 'Savings', 200000.00),

(6, 6, 'Savings', 100000.00),

(7, 7, 'Current', 30000.00),

(8, 8, 'Savings', 120000.00),

(9, 9, 'Current', 95000.00),

(10, 10, 'Savings', 80000.00);

A screenshot of a computer

Description automatically generated

**Insert into transactions Table:**

INSERT INTO transactions (transaction\_id, account\_id, transaction\_date, transaction\_amount, transaction\_type) VALUES

(1, 1, '2024-03-01', 5000.00, 'Credit'),

(2, 2, '2024-09-02', 20000.00, 'Debit'),

(3, 3, '2024-09-03', 10000.00, 'Credit'),

(4, 4, '2024-03-04', 5000.00, 'Debit'),

(5, 5, '2024-09-05', 50000.00, 'Credit'),

(6, 6, '2024-09-06', 20000.00, 'Debit'),

(7, 7, '2024-09-07', 1000.00, 'Credit'),

(8, 8, '2024-09-08', 30000.00, 'Debit'),

(9, 9, '2024-09-09', 5000.00, 'Credit'),

(10, 10, '2024-09-10', 10000.00, 'Debit');

A computer screen with a computer screen

Description automatically generated

**Insert into loans Table:**

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_term) VALUES

(1, 1, 100000.00, 8.5, 60),

(2, 2, 200000.00, 9.0, 72),

(3, 3, 150000.00, 8.0, 48),

(4, 4, 250000.00, 7.5, 84),

(5, 5, 300000.00, 9.2, 60),

(6, 6, 120000.00, 8.8, 36),

(7, 7, 180000.00, 7.9, 48),

(8, 8, 220000.00, 9.1, 60),

(9, 9, 160000.00, 8.3, 72),

(10, 10, 190000.00, 8.6, 60);

A screenshot of a computer

Description automatically generated

**Insert into loan\_payments Table:**

INSERT INTO loan\_payments (payment\_id, loan\_id, payment\_date, payment\_amount) VALUES

(1, 1, '2024-09-11', 2000.00),

(2, 2, '2024-09-12', 3000.00),

(3, 3, '2024-09-13', 2500.00),

(4, 4, '2024-09-14', 4000.00),

(5, 5, '2024-09-15', 5000.00),

(6, 6, '2024-09-16', 1500.00),

(7, 7, '2024-09-17', 2200.00),

(8, 8, '2024-09-18', 3500.00),

(9, 9, '2024-09-19', 2800.00),

(10, 10, '2024-09-20', 3000.00);

A screenshot of a computer

Description automatically generated

Now we are inserted some sample data successfully into the tables we are created as per the screenshots.

# Step 4: Data Exploration

4.1Write query to retrieve all customer information:

select \* from Customers;

A computer screen shot of a computer

Description automatically generated

4.2: Query accounts for a specific customer:

select \* from accounts where customer\_id=5;

A screenshot of a computer

Description automatically generated

4.3: Find the customer name and account balance for each account

select concat(first\_name,' ',last\_name) as customer\_name,a.account\_type, balance

 from customers c join accounts a on c.customer\_id=a.customer\_id ;

A screenshot of a computer

Description automatically generated

4.4: Analyze customer loan balances:

select concat(first\_name,' ',last\_name) as customer\_name,l.loan\_amount

 from customers c join loans l on c.customer\_id=l.customer\_id ;

A screenshot of a computer

Description automatically generated

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

select distinct concat(first\_name, last\_name) as customername , transaction\_date

from customers c

join accounts a on   c.customer\_id=a.customer\_id

join transactions t on a.account\_id=t.account\_id

    where t.transaction\_date between '2024-03-01' and '2024-03-31';

A computer screen with text boxes

Description automatically generated

## Step 5:  Aggregation and Insights

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

SELECT concat(first\_name, last\_name) as customername , 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;

A screenshot of a computer

Description automatically generated

5.2: Calculate the average loan amount for each loan term:

SELECT loan\_term, AVG(loan\_amount) AS average\_loan\_amount

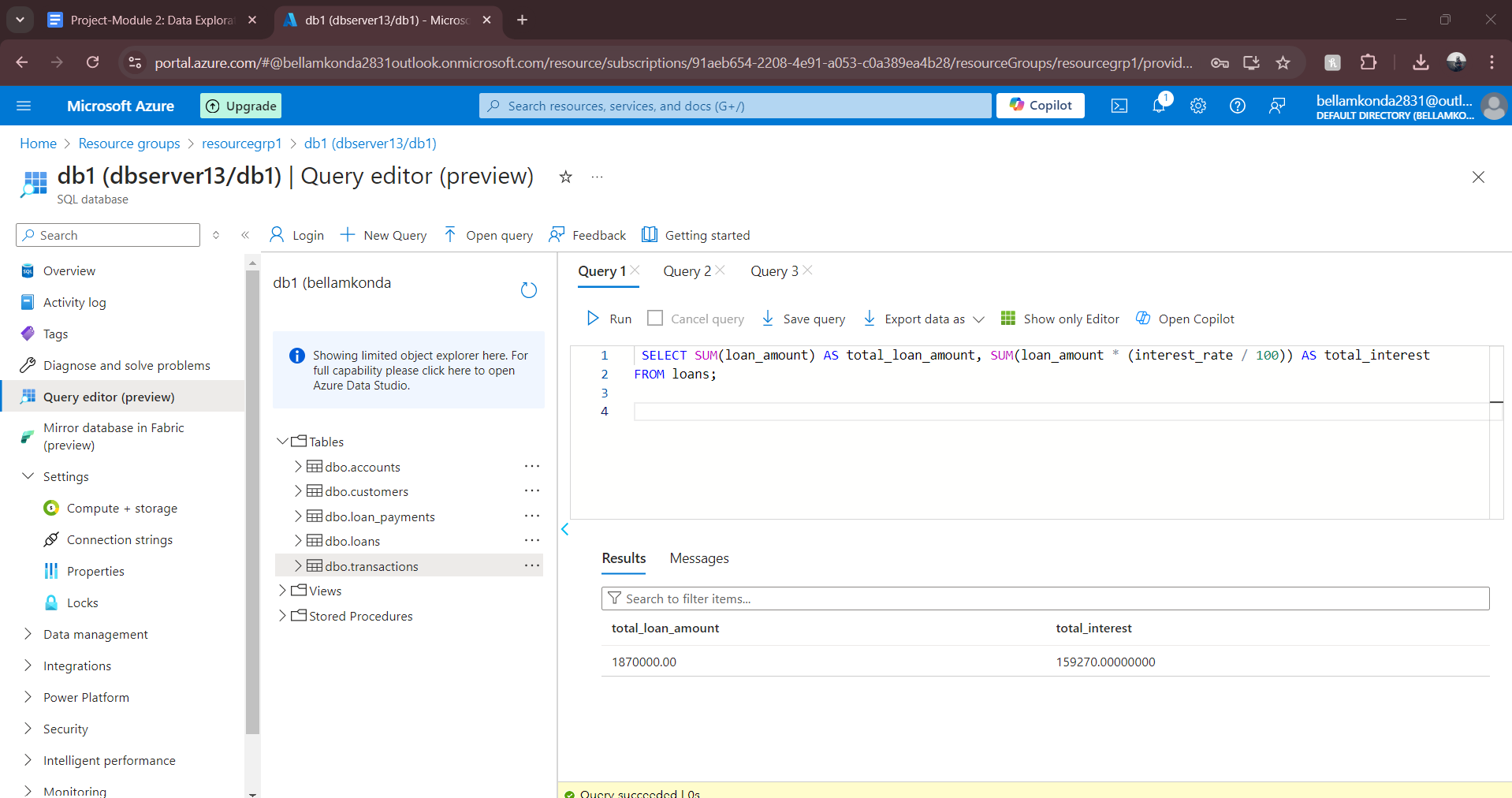
FROM loans

GROUP BY loan\_term;

## A screenshot of a computer Description automatically generated

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

 SELECT SUM(loan\_amount) AS total\_loan\_amount, SUM(loan\_amount \* (interest\_rate / 100)) AS total\_interest FROM loans;



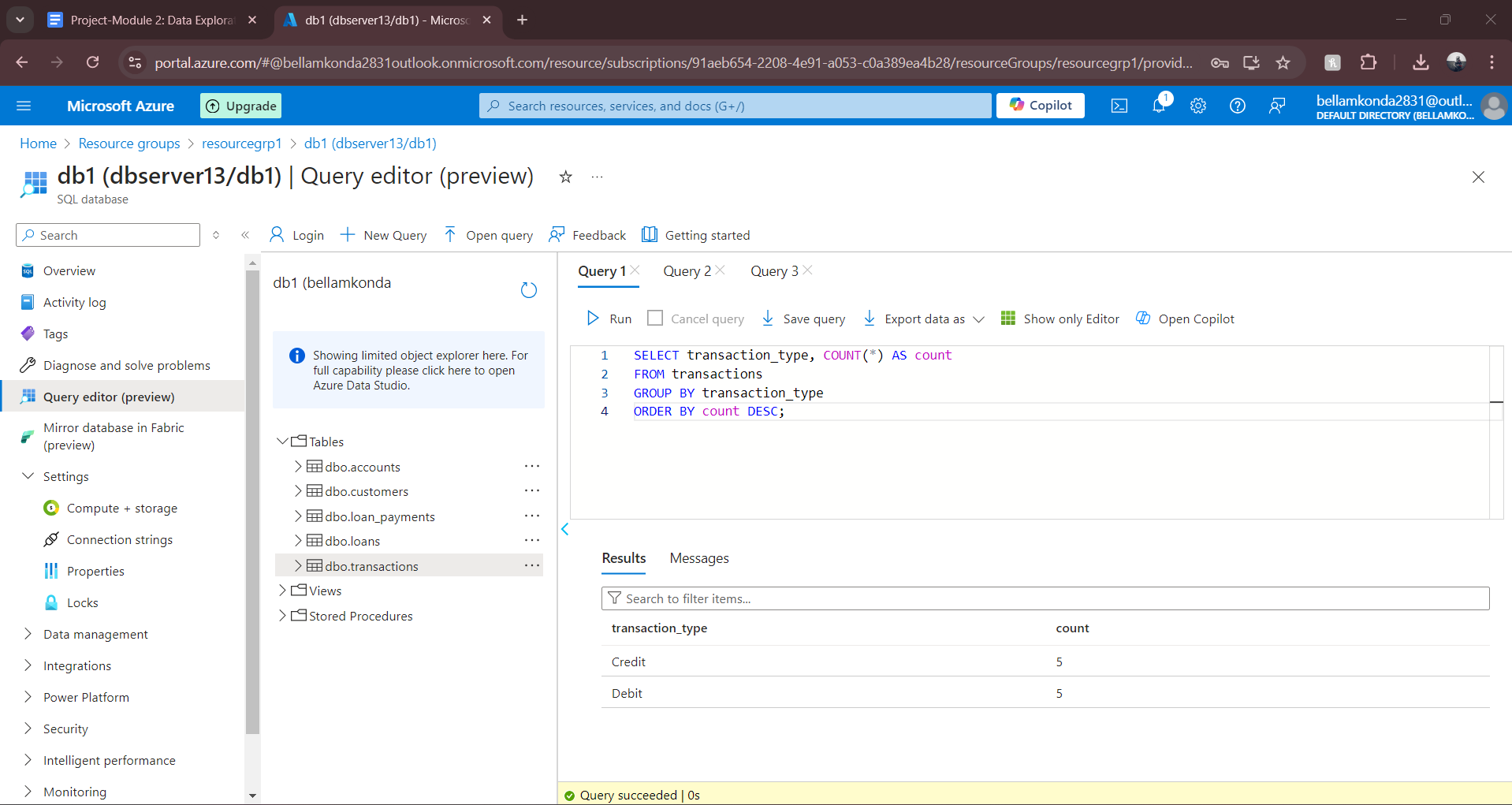
5.4: Find the most frequent transaction type

SELECT transaction\_type, COUNT(\*) AS count

FROM transactions

GROUP BY transaction\_type

ORDER BY count DESC;

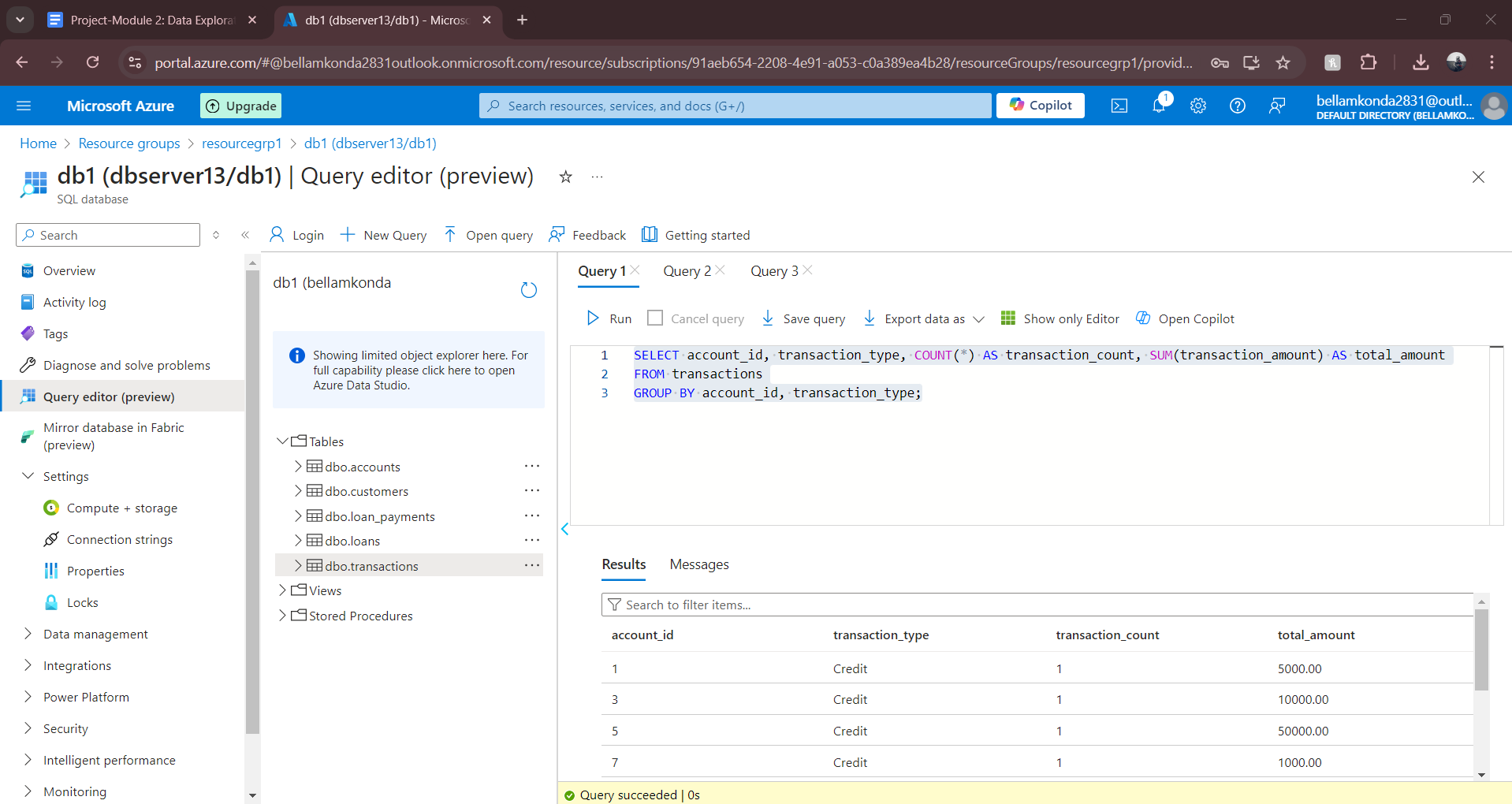


5.5: Analyze transactions by account and transaction type:

SELECT account\_id, transaction\_type, COUNT(\*) AS transaction\_count, SUM(transaction\_amount) AS total\_amount

FROM transactions

GROUP BY account\_id, transaction\_type;



## Step 6: Advanced Analysis

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

CREATE VIEW active\_loans\_with\_high\_payments AS

SELECT l.loan\_id, c.first\_name, c.last\_name, l.loan\_amount, lp.payment\_amount

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

JOIN loan\_payments lp ON l.loan\_id = lp.loan\_id

WHERE lp.payment\_amount > 1000;

A screenshot of a computer

Description automatically generated

   6.2: Create an index on `transaction\_date` in the `transactions` table for performance optimization:

CREATE INDEX idx\_transaction\_date ON transactions(transaction\_date);

A screenshot of a computer

Description automatically generated

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF PROJECT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*