

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**

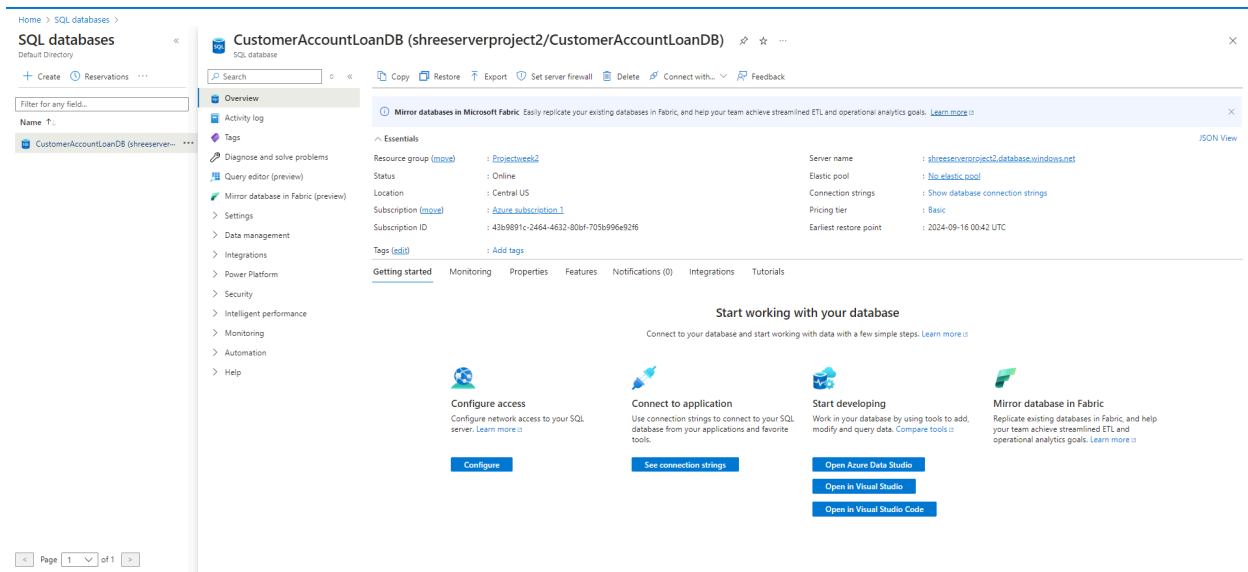
- Connectivity method: “Public Endpoint”
- Allow Azure services and resources to access this server: “Yes”
- Add current client IP address: “Yes”
- Connection policy: “Default - Uses Redirect policy for all client connections originating inside of Azure (except Private Endpoint connections) and Proxy 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;
```

The screenshot shows the Azure Data Studio interface with the 'Query editor (preview)' tab active. The query 'SELECT * FROM Customers;' has been executed successfully, returning 9 rows of customer data. The results are displayed in a table with columns: customer_id, first_name, last_name, address, city, state, and zip.

customer_id	first_name	last_name	address	city	state	zip
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

- Query accounts for a specific customer.

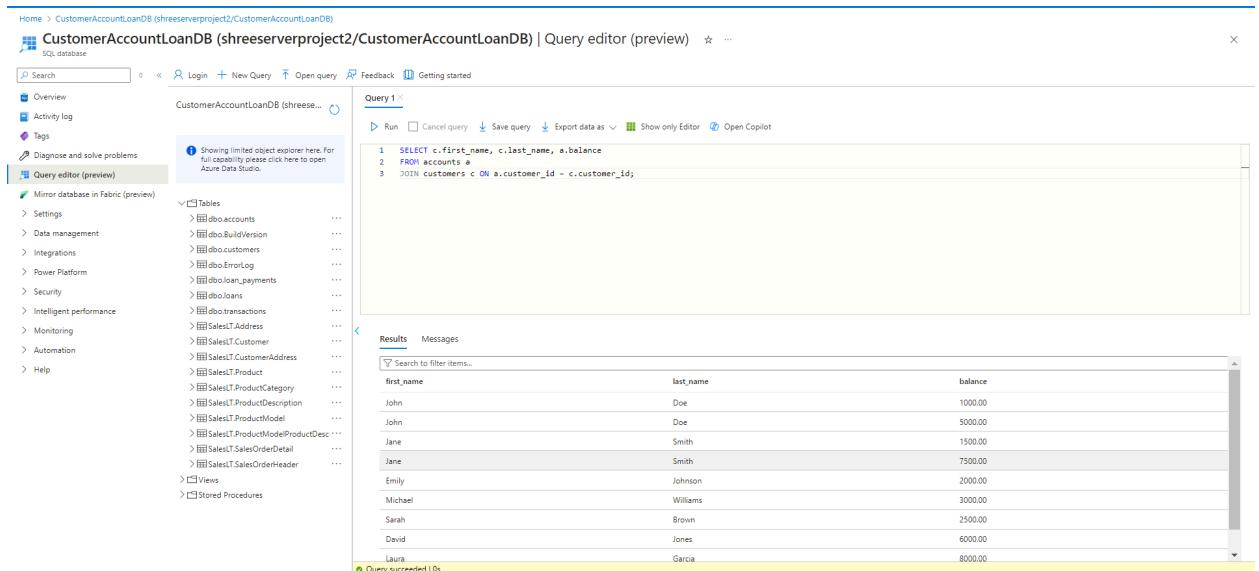
```
SELECT *  
FROM accounts  
WHERE customer_id = 3;
```

The screenshot shows the Azure Data Studio interface with the 'Query editor (preview)' tab active. The query 'SELECT * FROM accounts WHERE customer_id = 3;' has been executed successfully, returning 1 row of account data for customer_id 3. The results are displayed in a table with columns: account_id, customer_id, account_type, and balance.

account_id	customer_id	account_type	balance
5	3	Savings	2000.00

- 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;
```



CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB) | Query editor (preview)

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Query 1

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

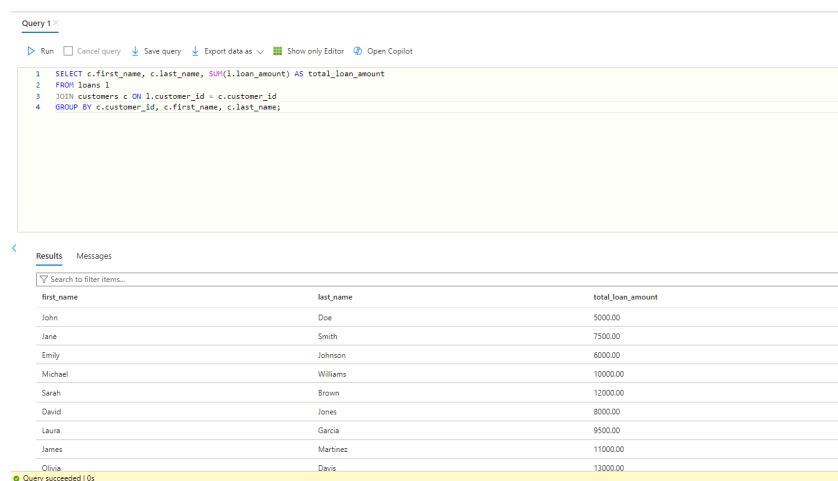
Results

first_name	last_name	balance
John	Doe	1000.00
John	Doe	5000.00
Jane	Smith	1500.00
Jane	Smith	7500.00
Emily	Johnson	2000.00
Michael	Williams	3000.00
Sarah	Brown	2500.00
David	Jones	6000.00
Laura	Garcia	8000.00

Query succeeded | 0s

- 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;
```



Query 1

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

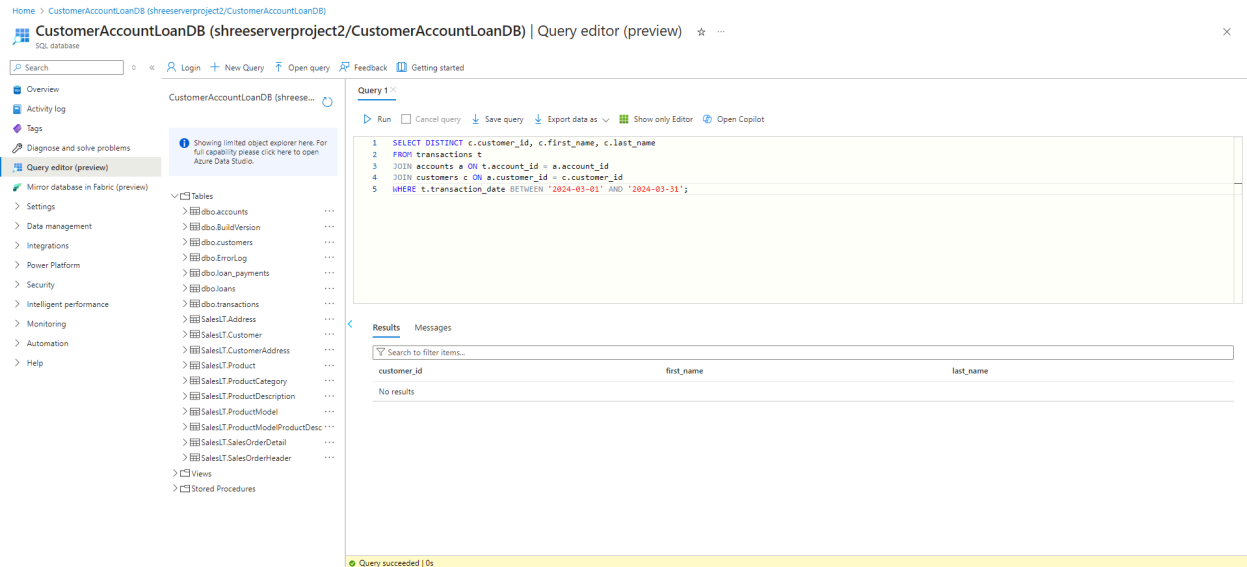
Results

first_name	last_name	total_loan_amount
John	Doe	5000.00
Jane	Smith	7500.00
Emily	Johnson	6000.00
Michael	Williams	10000.00
Sarah	Brown	12000.00
David	Jones	8000.00
Laura	Garcia	9500.00
James	Martinez	11000.00
Olivia	Davis	13000.00

Query succeeded | 0s

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

```
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:

```
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;
```


Home > CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB) | Query editor (preview) ☆

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Mirror database in Fabric (preview)

Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help

CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB)

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Tables

- dbo.accounts
- dbo.BuildVersion
- dbo.customers
- dbo.ErrorLog
- dbo.loan_payments
- dbo.loans
- dbo.transactions
- SalesLT.Address
- SalesLT.Customer
- SalesLT.CustomerAddress
- SalesLT.Product
- SalesLT.ProductCategory
- SalesLT.ProductDescription
- SalesLT.ProductModelProductDesc
- SalesLT.ProductOrderDetail
- SalesLT.SalesOrderHeader
- Views
- Stored Procedures

Query 1

Run Cancel query Save query Export data as Show only Editor Open Copilot

```

1 SELECT
2     c.customer_id,
3     c.first_name,
4     c.last_name,
5     SUM(a.balance) AS total_balance
6 FROM
7     customers c
8 JOIN
9     accounts a ON c.customer_id = a.customer_id
10 GROUP BY
11     c.customer_id, c.first_name, c.last_name;
12

```

Results Messages

Search to filter items...

customer_id	first_name	last_name	total_balance
1	John	Doe	6000.00
2	Jane	Smith	9000.00
3	Emily	Johnson	2000.00
4	Michael	Williams	3000.00
5	Sarah	Brown	2500.00
6	David	Jones	6000.00
7	Laura	Garcia	8000.00
8	James	Martinez	1200.00

Query succeeded | 0s

- 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;

```

Home > CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB) | Query editor (preview) ☆

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Mirror database in Fabric (preview)

Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help

CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB)

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Tables

- dbo.accounts
- dbo.BuildVersion
- dbo.customers
- dbo.ErrorLog
- dbo.loan_payments
- dbo.loans
- dbo.transactions
- SalesLT.Address
- SalesLT.Customer
- SalesLT.CustomerAddress
- SalesLT.Product
- SalesLT.ProductCategory
- SalesLT.ProductDescription
- SalesLT.ProductModelProductDesc
- SalesLT.ProductOrderDetail
- SalesLT.SalesOrderHeader
- Views
- Stored Procedures

Query 1

Run Cancel query Save query Export data as Show only Editor Open Copilot

```

1 SELECT
2     loan_term,
3     AVG(loan_amount) AS average_loan_amount
4 FROM
5     loans
6 GROUP BY
7     loan_term;
8

```

Results Messages

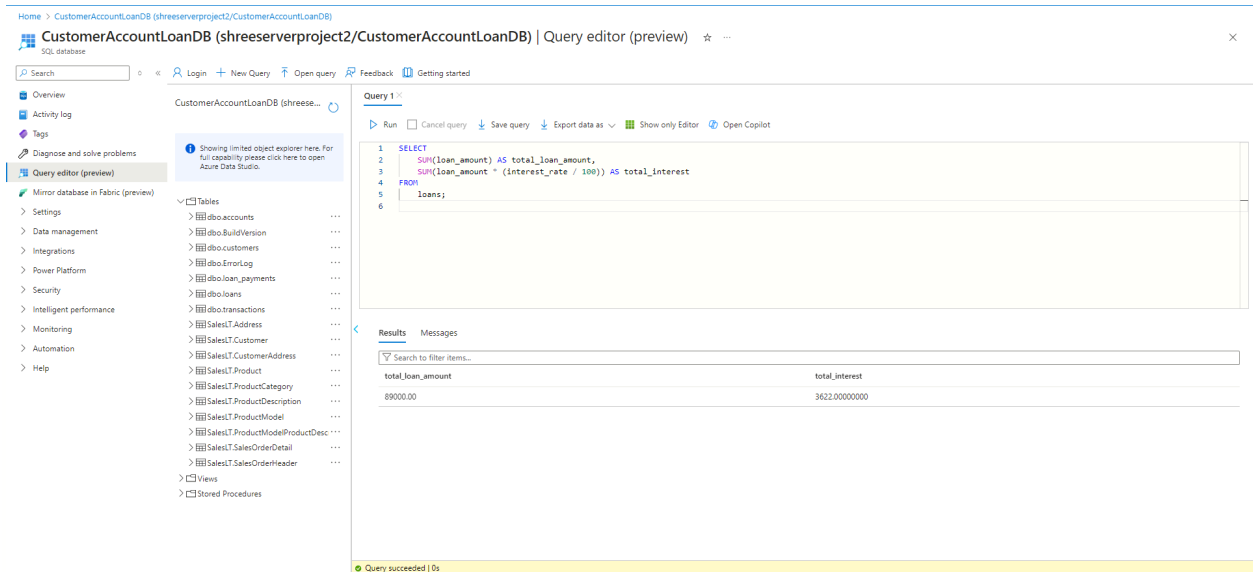
Search to filter items...

loan_term	average_loan_amount
12	5000.000000
18	6000.000000
20	7000.000000
24	7750.000000
30	9500.000000
36	10000.000000
42	11000.000000
48	12000.000000
54	13000.000000

Query succeeded | 0s

- 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;
```



The screenshot shows the Azure Data Studio interface with the 'Query editor (preview)' window open. The query editor displays the following SQL query:

```
1 SELECT
2     SUM(loan_amount) AS total_loan_amount,
3     SUM(loan_amount * (interest_rate / 100)) AS total_interest
4 FROM
5     loans;
```

The 'Results' tab shows the output of the query:

total_loan_amount	total_interest
89000.00	3622.00000000

A status bar at the bottom indicates 'Query succeeded | 0s'.

- Find the most frequent transaction type

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

Home > CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB) | Query editor (preview) ☆ ✕

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Mirror database in Fabric (preview)

Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help

Tables
dbo.accounts
dbo.BuildVersion
dbo.customers
dbo.ErrorLog
dbo.loan_payments
dbo.loans
dbo.transactions
SalesLT.Address
SalesLT.Customer
SalesLT.CustomerAddress
SalesLT.Product
SalesLT.ProductCategory
SalesLT.ProductDescription
SalesLT.ProductModel
SalesLT.ProductModelProductDesc
SalesLT.ProductOrderDetail
SalesLT.SalesOrderHeader
Views
Stored Procedures

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Query 1

Run Cancel query Save query Export data as Show only Editor Open Copilot

```
1 SELECT
2   transaction_type,
3   COUNT(*) AS frequency
4 FROM
5   transactions
6 GROUP BY
7   transaction_type
8 ORDER BY
9   frequency DESC;
```

Results Messages

Search to filter items...

transaction_type	frequency
Deposit	5
Withdrawal	5

Query succeeded 1.0s

- 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;
```

Home > CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB) | Query editor (preview) ☆

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Mirror database in Fabric (preview)

Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help

Tables
dbo.accounts
dbo.BuildVersion
dbo.customers
dbo.ErrorLog
dbo.loan_payments
dbo.loans
dbo.transactions
SalesLT.Address
SalesLT.Customer
SalesLT.CustomerAddress
SalesLT.Product
SalesLT.ProductCategory
SalesLT.ProductDescription
SalesLT.ProductModel
SalesLT.ProductModelProductDesc
SalesLT.SalesOrderDetail
SalesLT.SalesOrderHeader

Views
Stored Procedures

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Query 1

Run Cancel query Save query Export data as Show only Editor Open Copilot

```

1 SELECT
2     a.account_id,
3     t.transaction_type,
4     SUM(t.transaction_amount) AS total_amount
5 FROM
6     transactions t
7 JOIN
8     accounts a ON t.account_id = a.account_id
9 GROUP BY
10    a.account_id, t.transaction_type;
11

```

Results Messages

Search to filter items...

account_id	transaction_type	total_amount
1	Deposit	200.00
2	Deposit	300.00
3	Deposit	150.00
5	Deposit	250.00
7	Deposit	400.00
1	Withdrawal	-100.00
2	Withdrawal	-50.00
4	Withdrawal	-200.00
6	Withdrawal	-300.00

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;

```

Microsoft Azure | Upgrade | Search resources, services, and docs (G+)

Home > CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB) | Query editor (preview) ☆

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Mirror database in Fabric (preview)

Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help

Tables
dbo.accounts
dbo.BuildVersion
dbo.customers
dbo.ErrorLog
dbo.loan_payments
dbo.loans
dbo.transactions
SalesLT.Address
SalesLT.Customer
SalesLT.CustomerAddress
SalesLT.Product
SalesLT.ProductCategory
SalesLT.ProductDescription
SalesLT.ProductModel
SalesLT.ProductModelProductDesc
SalesLT.SalesOrderDetail
SalesLT.SalesOrderHeader
Views
Stored Procedures

Query 1

Run Cancel query Save query Export data as Show only Editor Open Copilot

```
1 CREATE VIEW active_loans_with_high_payments_cx AS
2 SELECT l.loan_id,
3        l.customer_id,
4        l.loan_amount,
5        l.interest_rate,
6        l.loan_term,
7        COALESCE(SUM(lp.payment_amount), 0) AS total_payments
8 FROM loans l
9 LEFT JOIN loan_payments lp ON l.loan_id = lp.loan_id
10 GROUP BY l.loan_id, l.customer_id, l.loan_amount, l.interest_rate, l.loan_term
11 HAVING SUM(lp.payment_amount) > 1000;
```

Results Messages

Query succeeded: Affected rows: 0

Query succeeded | 0s

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

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

Home > CustomerAccountLoanDB (shreeserverproject2/CustomerAccountLoanDB) | Query editor (preview) ☆

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Mirror database in Fabric (preview)

Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help

Tables
dbo.accounts
dbo.BuildVersion
dbo.customers
dbo.ErrorLog
dbo.loan_payments
dbo.loans
dbo.transactions
SalesLT.Address
SalesLT.Customer
SalesLT.CustomerAddress
SalesLT.Product
SalesLT.ProductCategory
SalesLT.ProductDescription
SalesLT.ProductModel
SalesLT.ProductModelProductDesc
SalesLT.SalesOrderDetail
SalesLT.SalesOrderHeader
Views
Stored Procedures

Query 1

Run Cancel query Save query Export data as Show only Editor Open Copilot

```
1 CREATE INDEX idx_transaction_date_cx ON transactions(transaction_date);
2
```

Results Messages

Query succeeded: Affected rows: 0

Query succeeded | 0s

