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

Objective:

Trainees will explore and manipulate multiple related datasets using Azure SQL Database. The focus will be on organizing the datasets, identifying data types, and exploring their relationships and contents.

Tools Required:

- Azure SQL Database
- SQL Management tools (Azure Data Studio or SQL Server Management Studio)
- Dataset - https://kaggle.com/datasets/9234c6c4d25b6eb7c3dbb15a0e33d65ae68a405d42acba8db1248defee7aff9c
- GitHub

Project Tasks:

1. Setting Up Azure SQL Database

- Step 1.1: Create an Azure SQL Database in the Azure portal.
  - Define a new database and server.
- Step 1.2: Name the database `CustomerAccountLoanDB`.

2. Data Organization

- Step 2.1: Create tables for the provided feeds:
  - 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)
       );
```

- 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)
);
```

- 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)
);
```

- 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)
);
```

- 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)
);
```

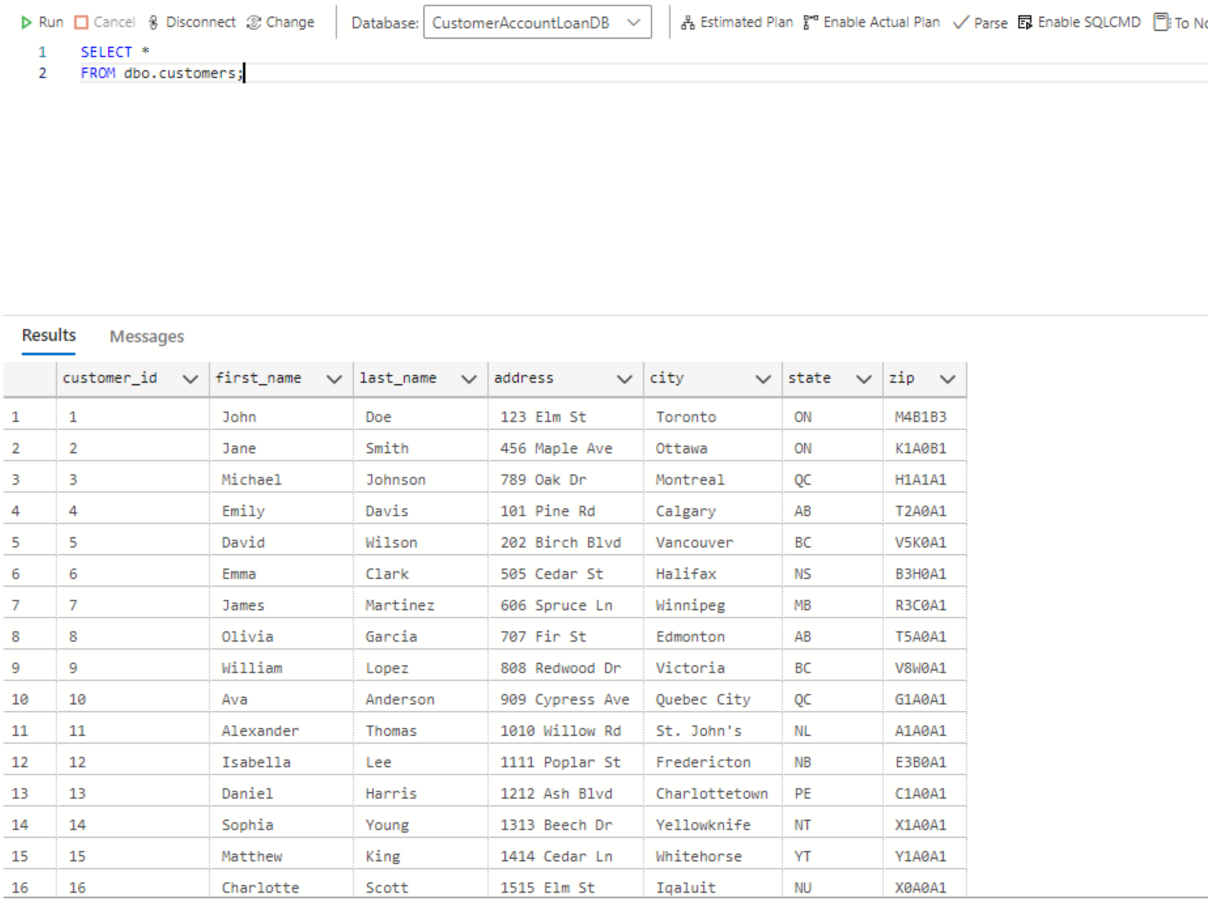
3. Data Insertion

- Step 3.1: Populate tables with sample data using `INSERT INTO` statements for each table.
- Step 3.2: Ensure data consistency and relationships, ensuring each foreign key points to valid primary keys.

Data available @
https://kaggle.com/datasets/9234c6c4d25b6eb7c3dbb15a0e33d65ae68a405d42acba8db1248defee7aff9c

4. Data Exploration

- Step 4.1: Write query to retrieve all customer information:

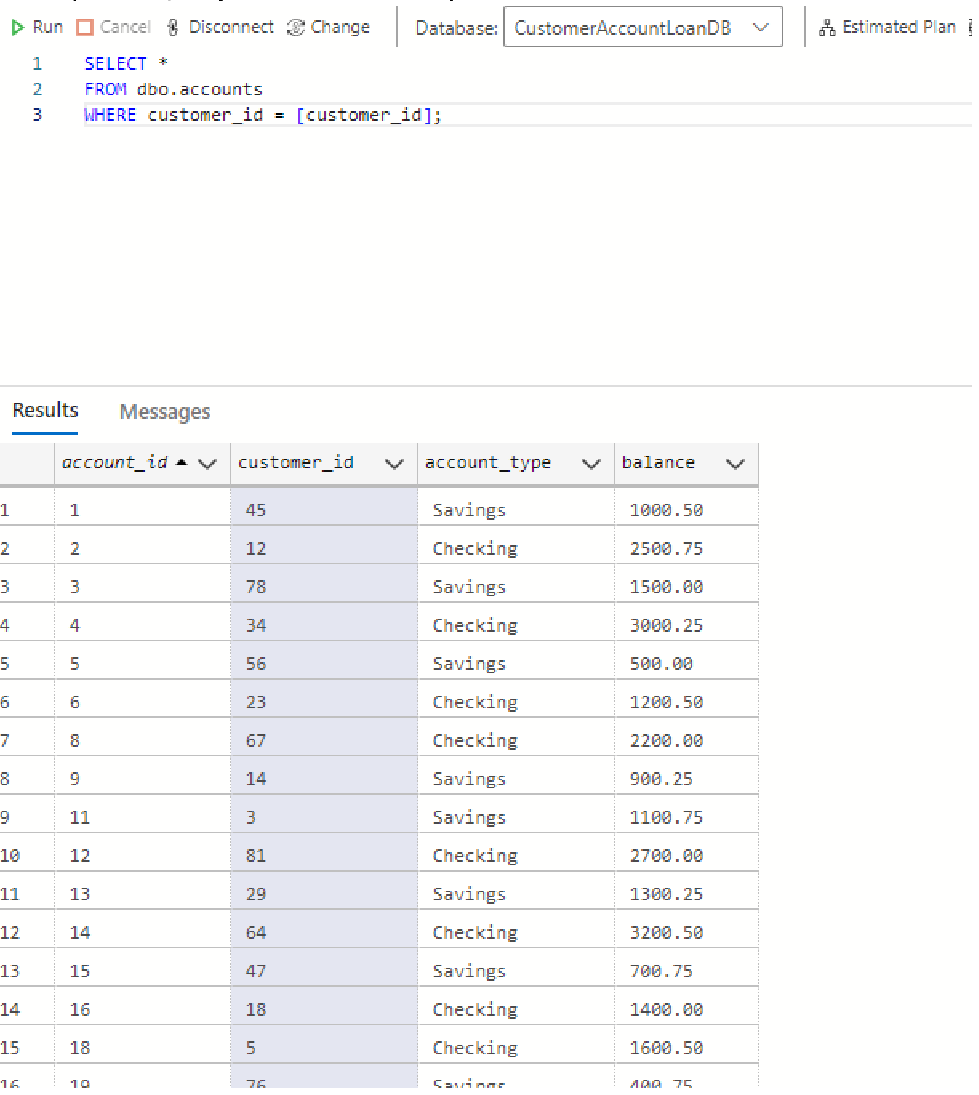


Database: CustomerAccountLoanDB

```
SELECT *
FROM dbo.customers;
```

	customer_id	first_name	last_name	address	city	state	zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1
9	9	William	Lopez	808 Redwood Dr	Victoria	BC	V8W0A1
10	10	Ava	Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1
11	11	Alexander	Thomas	1010 Willow Rd	St. John's	NL	A1A0A1
12	12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1
13	13	Daniel	Harris	1212 Ash Blvd	Charlottetown	PE	C1A0A1
14	14	Sophia	Young	1313 Beech Dr	Yellowknife	NT	X1A0A1
15	15	Matthew	King	1414 Cedar Ln	Whitehorse	YT	Y1A0A1
16	16	Charlotte	Scott	1515 Elm St	Iqaluit	NU	X0A0A1

- Step 4.2: Query accounts for a specific customer:

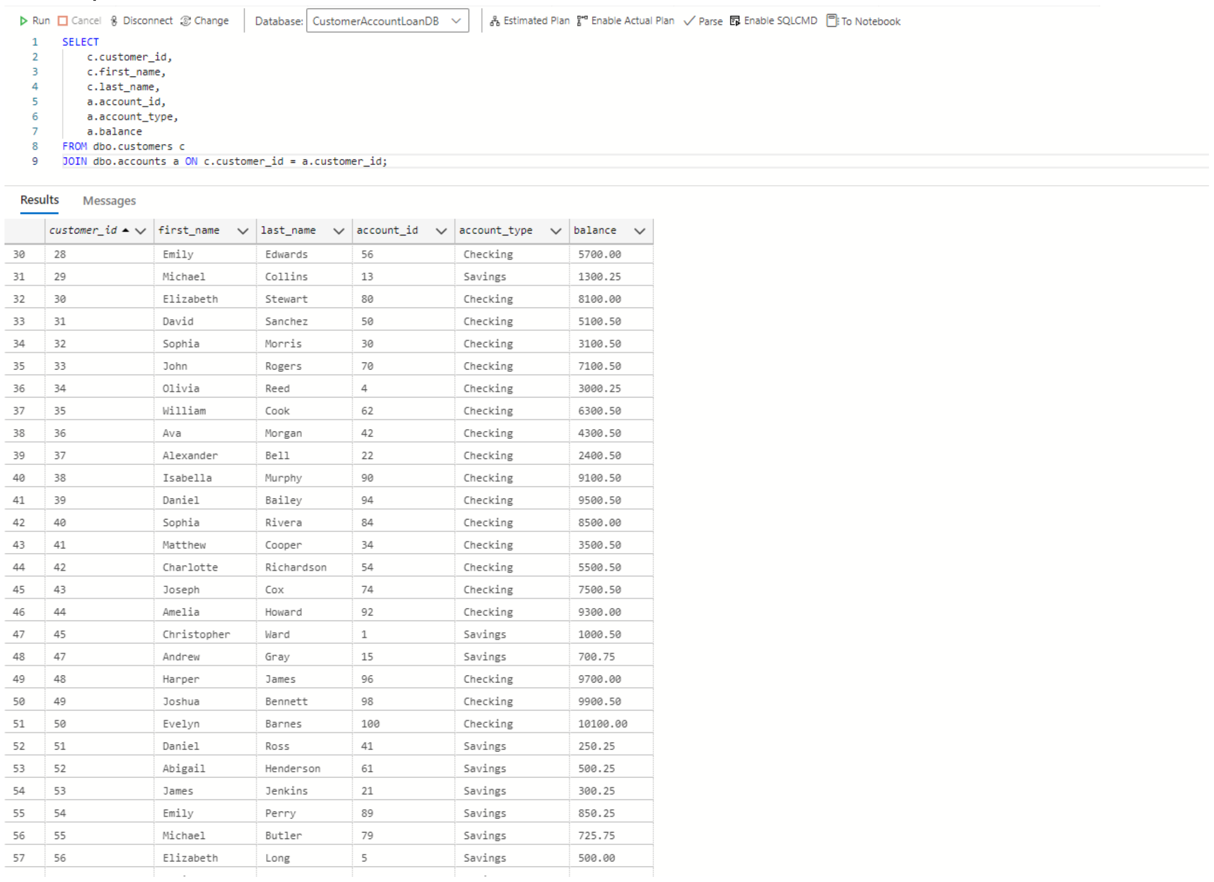


Database: CustomerAccountLoanDB

```sql
SELECT *
FROM dbo.accounts
WHERE customer_id = [customer_id];
```

	account_id	customer_id	account_type	balance
1	1	45	Savings	1000.50
2	2	12	Checking	2500.75
3	3	78	Savings	1500.00
4	4	34	Checking	3000.25
5	5	56	Savings	500.00
6	6	23	Checking	1200.50
7	8	67	Checking	2200.00
8	9	14	Savings	900.25
9	11	3	Savings	1100.75
10	12	81	Checking	2700.00
11	13	29	Savings	1300.25
12	14	64	Checking	3200.50
13	15	47	Savings	700.75
14	16	18	Checking	1400.00
15	18	5	Checking	1600.50
16	19	76	Savings	400.75

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

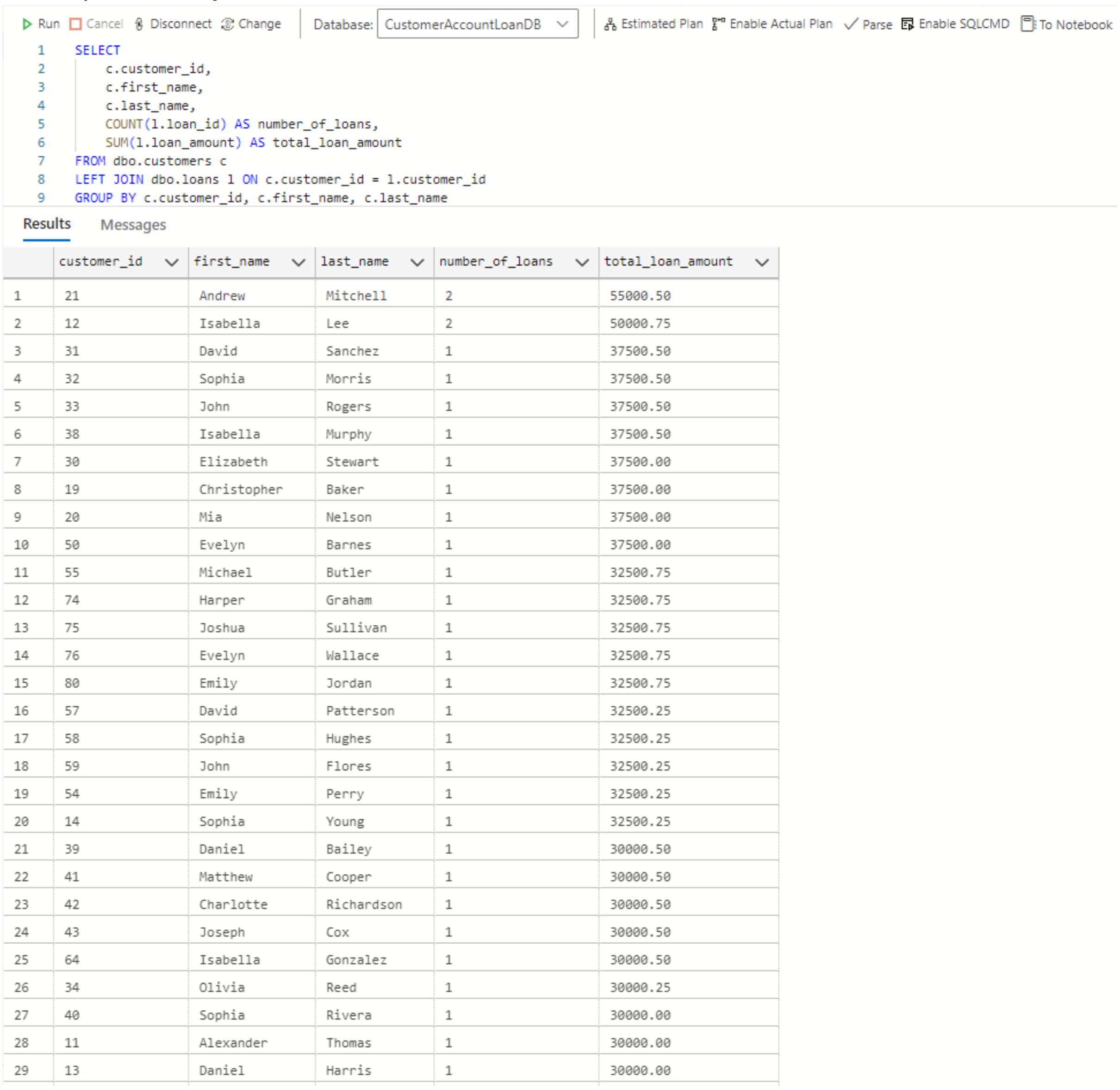


Database: CustomerAccountLoanDB

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

	customer_id	first_name	last_name	account_id	account_type	balance
30	28	Emily	Edwards	56	Checking	5700.00
31	29	Michael	Collins	13	Savings	1300.25
32	30	Elizabeth	Stewart	80	Checking	8100.00
33	31	David	Sanchez	50	Checking	5100.50
34	32	Sophia	Morris	30	Checking	3100.50
35	33	John	Rogers	70	Checking	7100.50
36	34	Olivia	Reed	4	Checking	3000.25
37	35	William	Cook	62	Checking	6300.50
38	36	Ava	Morgan	42	Checking	4300.50
39	37	Alexander	Bell	22	Checking	2400.50
40	38	Isabella	Murphy	90	Checking	9100.50
41	39	Daniel	Bailey	94	Checking	9500.50
42	40	Sophia	Rivera	84	Checking	8500.00
43	41	Matthew	Cooper	34	Checking	3500.50
44	42	Charlotte	Richardson	54	Checking	5500.50
45	43	Joseph	Cox	74	Checking	7500.50
46	44	Amelia	Howard	92	Checking	9300.00
47	45	Christopher	Ward	1	Savings	1000.50
48	47	Andrew	Gray	15	Savings	700.75
49	48	Harper	James	96	Checking	9700.00
50	49	Joshua	Bennett	98	Checking	9900.50
51	50	Evelyn	Barnes	100	Checking	10100.00
52	51	Daniel	Ross	41	Savings	250.25
53	52	Abigail	Henderson	61	Savings	500.25
54	53	James	Jenkins	21	Savings	300.25
55	54	Emily	Perry	89	Savings	850.25
56	55	Michael	Butler	79	Savings	725.75
57	56	Elizabeth	Long	5	Savings	500.00

- Step 4.4: Analyze customer loan balances:

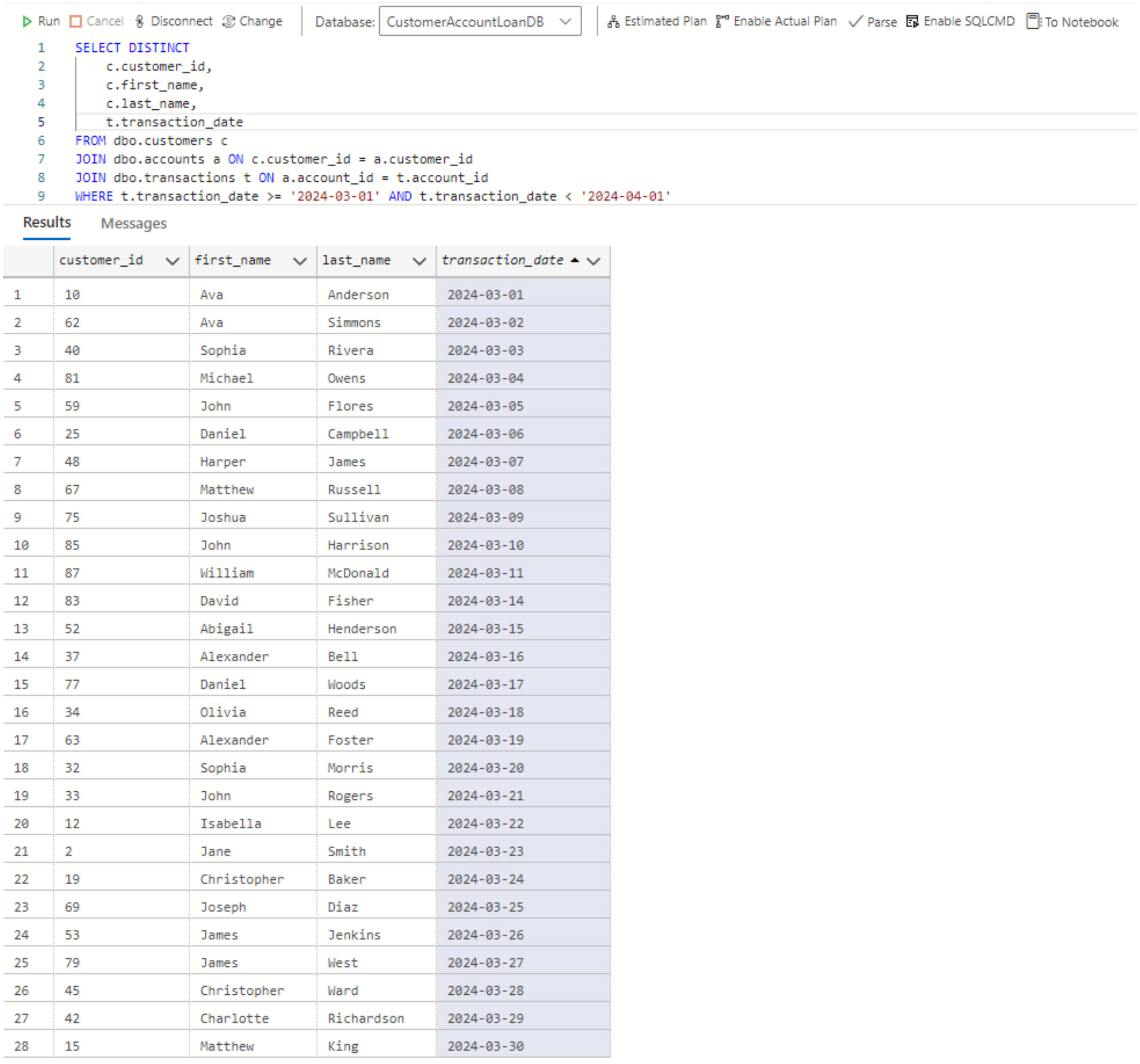


Database: CustomerAccountLoanDB

```sql
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    COUNT(l.loan_id) AS number_of_loans,
    SUM(l.loan_amount) AS total_loan_amount
FROM dbo.customers c
LEFT JOIN dbo.loans l ON c.customer_id = l.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
```

	customer_id	first_name	last_name	number_of_loans	total_loan_amount
1	21	Andrew	Mitchell	2	55000.50
2	12	Isabella	Lee	2	50000.75
3	31	David	Sanchez	1	37500.50
4	32	Sophia	Morris	1	37500.50
5	33	John	Rogers	1	37500.50
6	38	Isabella	Murphy	1	37500.50
7	30	Elizabeth	Stewart	1	37500.00
8	19	Christopher	Baker	1	37500.00
9	20	Mia	Nelson	1	37500.00
10	50	Evelyn	Barnes	1	37500.00
11	55	Michael	Butler	1	32500.75
12	74	Harper	Graham	1	32500.75
13	75	Joshua	Sullivan	1	32500.75
14	76	Evelyn	Wallace	1	32500.75
15	80	Emily	Jordan	1	32500.75
16	57	David	Patterson	1	32500.25
17	58	Sophia	Hughes	1	32500.25
18	59	John	Flores	1	32500.25
19	54	Emily	Perry	1	32500.25
20	14	Sophia	Young	1	32500.25
21	39	Daniel	Bailey	1	30000.50
22	41	Matthew	Cooper	1	30000.50
23	42	Charlotte	Richardson	1	30000.50
24	43	Joseph	Cox	1	30000.50
25	64	Isabella	Gonzalez	1	30000.50
26	34	Olivia	Reed	1	30000.25
27	40	Sophia	Rivera	1	30000.00
28	11	Alexander	Thomas	1	30000.00
29	13	Daniel	Harris	1	30000.00

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

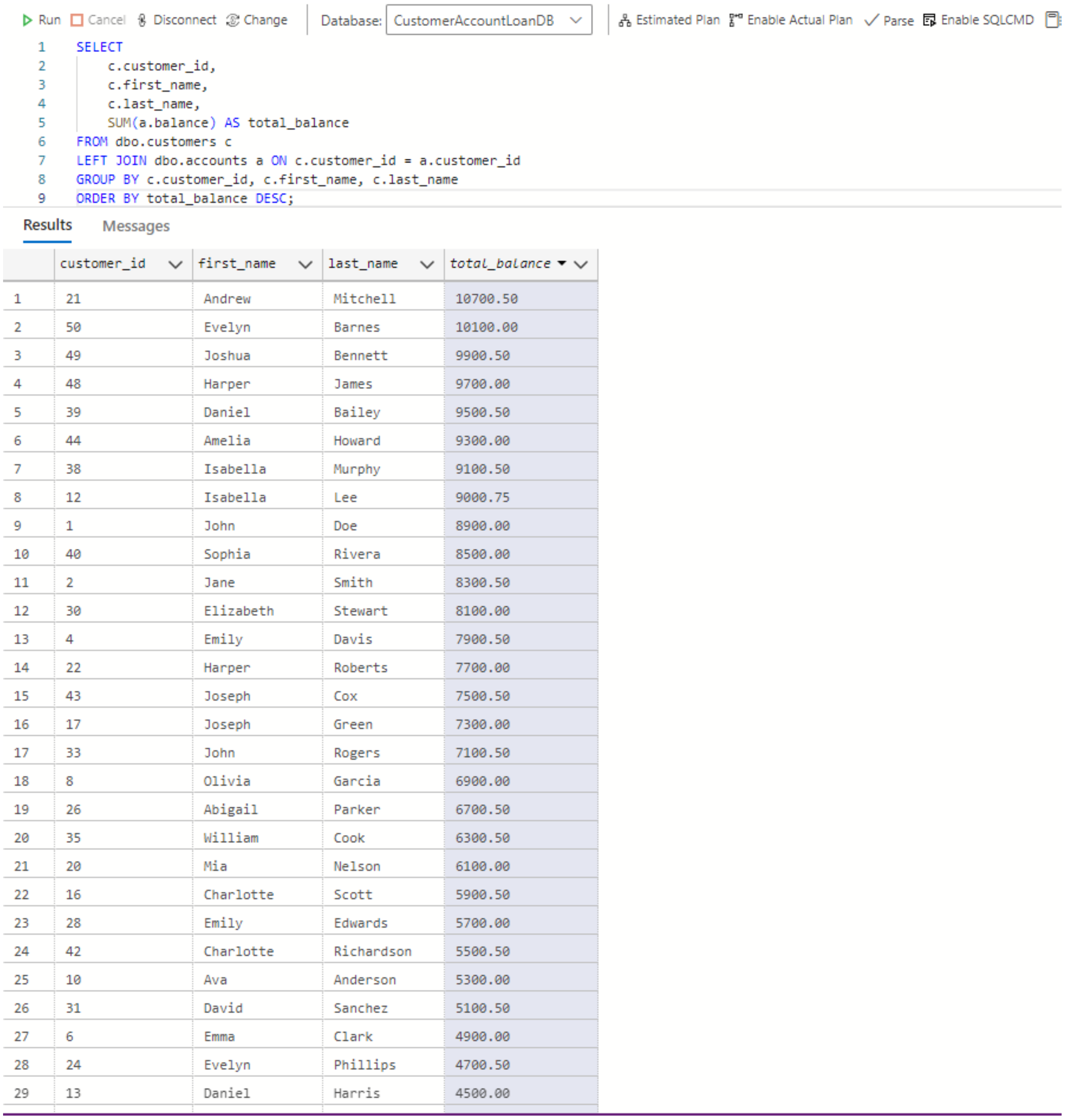


```sql
SELECT DISTINCT
    c.customer_id,
    c.first_name,
    c.last_name,
    t.transaction_date
FROM dbo.customers c
JOIN dbo.accounts a ON c.customer_id = a.customer_id
JOIN dbo.transactions t ON a.account_id = t.account_id
WHERE t.transaction_date >= '2024-03-01' AND t.transaction_date < '2024-04-01'
```

	customer_id	first_name	last_name	transaction_date
1	10	Ava	Anderson	2024-03-01
2	62	Ava	Simmons	2024-03-02
3	40	Sophia	Rivera	2024-03-03
4	81	Michael	Owens	2024-03-04
5	59	John	Flores	2024-03-05
6	25	Daniel	Campbell	2024-03-06
7	48	Harper	James	2024-03-07
8	67	Matthew	Russell	2024-03-08
9	75	Joshua	Sullivan	2024-03-09
10	85	John	Harrison	2024-03-10
11	87	William	McDonald	2024-03-11
12	83	David	Fisher	2024-03-14
13	52	Abigail	Henderson	2024-03-15
14	37	Alexander	Bell	2024-03-16
15	77	Daniel	Woods	2024-03-17
16	34	Olivia	Reed	2024-03-18
17	63	Alexander	Foster	2024-03-19
18	32	Sophia	Morris	2024-03-20
19	33	John	Rogers	2024-03-21
20	12	Isabella	Lee	2024-03-22
21	2	Jane	Smith	2024-03-23
22	19	Christopher	Baker	2024-03-24
23	69	Joseph	Diaz	2024-03-25
24	53	James	Jenkins	2024-03-26
25	79	James	West	2024-03-27
26	45	Christopher	Ward	2024-03-28
27	42	Charlotte	Richardson	2024-03-29
28	15	Matthew	King	2024-03-30

5. Aggregation and Insights

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

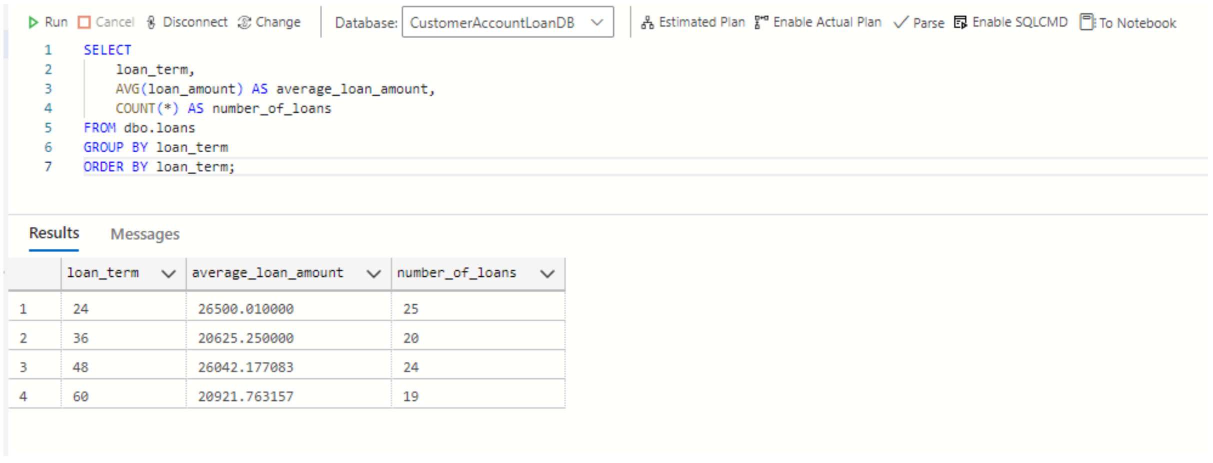


Database: CustomerAccountLoanDB

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

	customer_id	first_name	last_name	total_balance
1	21	Andrew	Mitchell	10700.50
2	50	Evelyn	Barnes	10100.00
3	49	Joshua	Bennett	9900.50
4	48	Harper	James	9700.00
5	39	Daniel	Bailey	9500.50
6	44	Amelia	Howard	9300.00
7	38	Isabella	Murphy	9100.50
8	12	Isabella	Lee	9000.75
9	1	John	Doe	8900.00
10	40	Sophia	Rivera	8500.00
11	2	Jane	Smith	8300.50
12	30	Elizabeth	Stewart	8100.00
13	4	Emily	Davis	7900.50
14	22	Harper	Roberts	7700.00
15	43	Joseph	Cox	7500.50
16	17	Joseph	Green	7300.00
17	33	John	Rogers	7100.50
18	8	Olivia	Garcia	6900.00
19	26	Abigail	Parker	6700.50
20	35	William	Cook	6300.50
21	20	Mia	Nelson	6100.00
22	16	Charlotte	Scott	5900.50
23	28	Emily	Edwards	5700.00
24	42	Charlotte	Richardson	5500.50
25	10	Ava	Anderson	5300.00
26	31	David	Sanchez	5100.50
27	6	Emma	Clark	4900.00
28	24	Evelyn	Phillips	4700.50
29	13	Daniel	Harris	4500.00

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

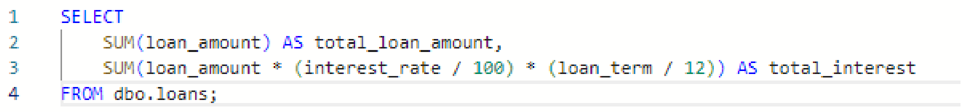


Database: CustomerAccountLoanDB

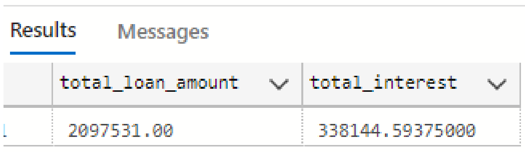
```sql
SELECT
    loan_term,
    AVG(loan_amount) AS average_loan_amount,
    COUNT(*) AS number_of_loans
FROM dbo.loans
GROUP BY loan_term
ORDER BY loan_term;
```

	loan_term	average_loan_amount	number_of_loans
1	24	26500.010000	25
2	36	20625.250000	20
3	48	26042.177083	24
4	60	20921.763157	19

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

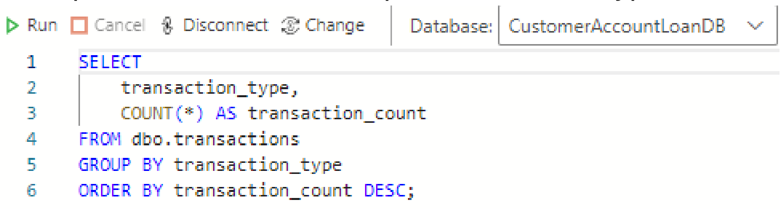


```sql
SELECT
    SUM(loan_amount) AS total_loan_amount,
    SUM(loan_amount * (interest_rate / 100) * (loan_term / 12)) AS total_interest
FROM dbo.loans;
```



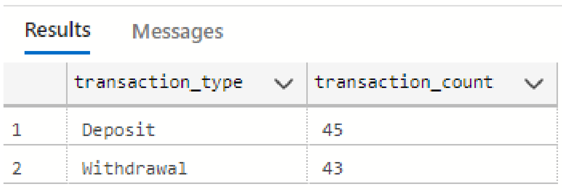
	total_loan_amount	total_interest
1	2097531.00	338144.59375000

- Step 5.4: Find the most frequent transaction type



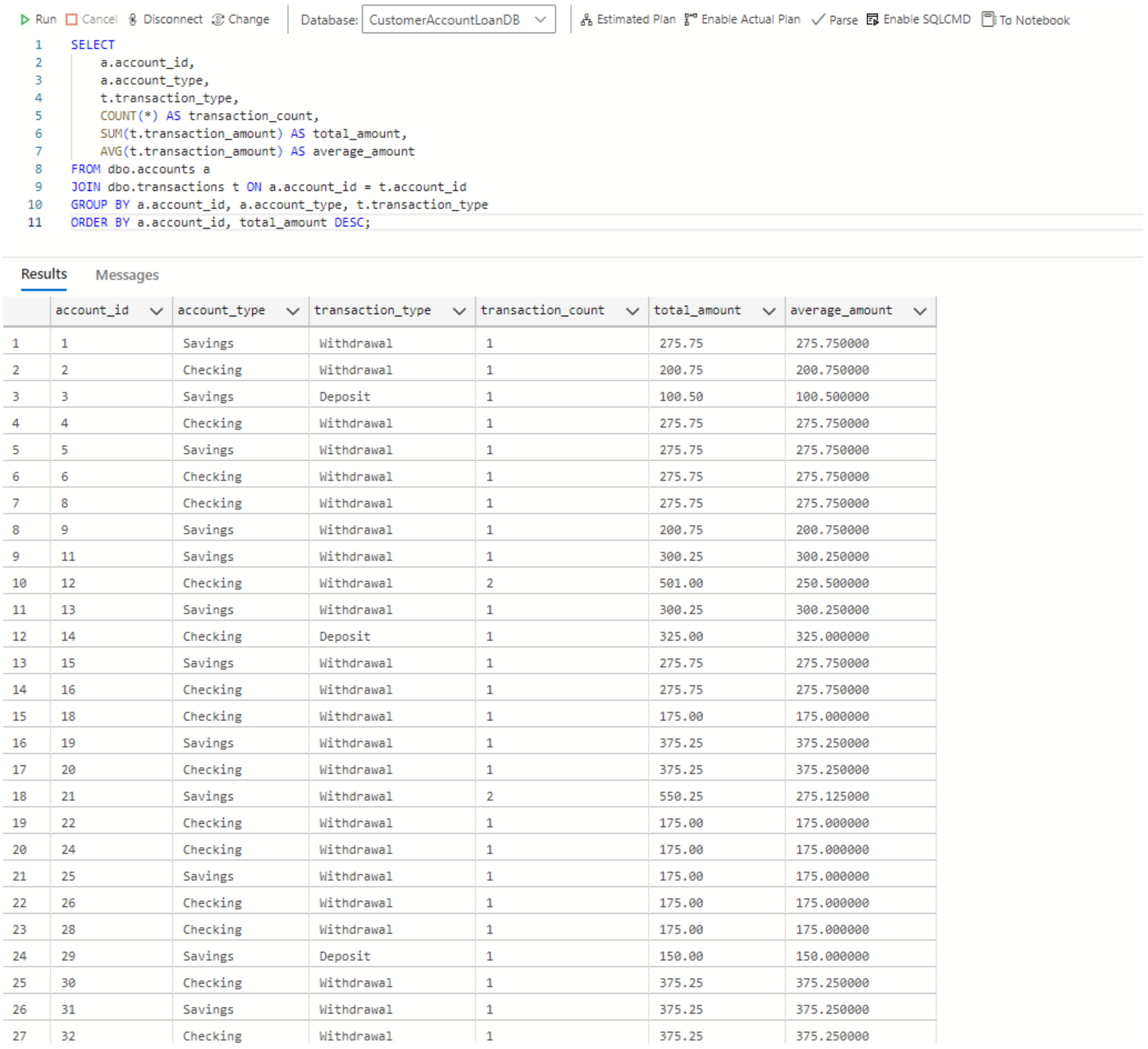
Database: CustomerAccountLoanDB

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



	transaction_type	transaction_count
1	Deposit	45
2	Withdrawal	43

- Step 5.5: Analyze transactions by account and transaction type:



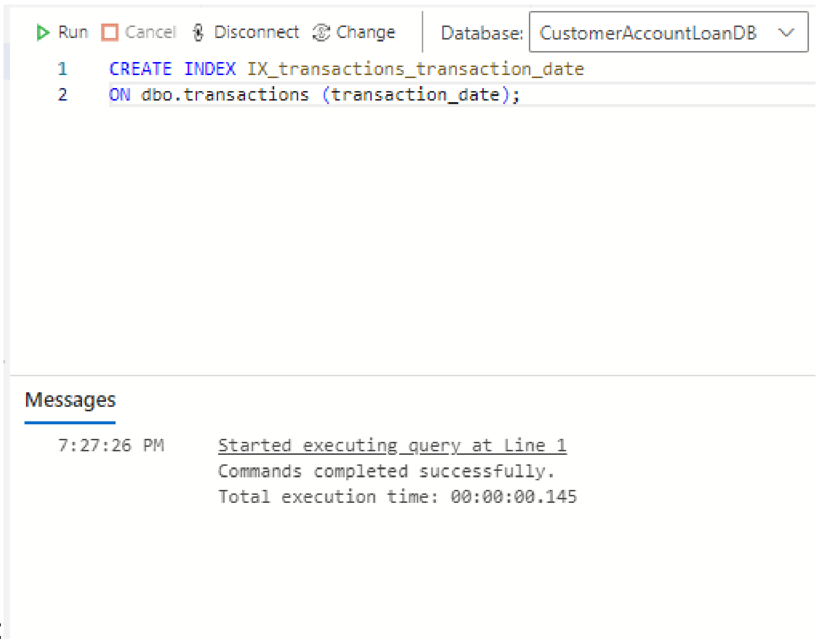
```sql
SELECT
    a.account_id,
    a.account_type,
    t.transaction_type,
    COUNT(*) AS transaction_count,
    SUM(t.transaction_amount) AS total_amount,
    AVG(t.transaction_amount) AS average_amount
FROM dbo.accounts a
JOIN dbo.transactions t ON a.account_id = t.account_id
GROUP BY a.account_id, a.account_type, t.transaction_type
ORDER BY a.account_id, total_amount DESC;
```

	account_id	account_type	transaction_type	transaction_count	total_amount	average_amount
1	1	Savings	Withdrawal	1	275.75	275.750000
2	2	Checking	Withdrawal	1	200.75	200.750000
3	3	Savings	Deposit	1	100.50	100.500000
4	4	Checking	Withdrawal	1	275.75	275.750000
5	5	Savings	Withdrawal	1	275.75	275.750000
6	6	Checking	Withdrawal	1	275.75	275.750000
7	8	Checking	Withdrawal	1	275.75	275.750000
8	9	Savings	Withdrawal	1	200.75	200.750000
9	11	Savings	Withdrawal	1	300.25	300.250000
10	12	Checking	Withdrawal	2	501.00	250.500000
11	13	Savings	Withdrawal	1	300.25	300.250000
12	14	Checking	Deposit	1	325.00	325.000000
13	15	Savings	Withdrawal	1	275.75	275.750000
14	16	Checking	Withdrawal	1	275.75	275.750000
15	18	Checking	Withdrawal	1	175.00	175.000000
16	19	Savings	Withdrawal	1	375.25	375.250000
17	20	Checking	Withdrawal	1	375.25	375.250000
18	21	Savings	Withdrawal	2	550.25	275.125000
19	22	Checking	Withdrawal	1	175.00	175.000000
20	24	Checking	Withdrawal	1	175.00	175.000000
21	25	Savings	Withdrawal	1	175.00	175.000000
22	26	Checking	Withdrawal	1	175.00	175.000000
23	28	Checking	Withdrawal	1	175.00	175.000000
24	29	Savings	Deposit	1	150.00	150.000000
25	30	Checking	Withdrawal	1	375.25	375.250000
26	31	Savings	Withdrawal	1	375.25	375.250000
27	32	Checking	Withdrawal	1	375.25	375.250000

6. Advanced Analysis

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

- Step 6.2: Create an index on `transaction_date` in the `transactions` table for performance optimization:



```sql
CREATE INDEX IX_transactions_transaction_date
ON dbo.transactions (transaction_date);
```

```
7:27:26 PM     Started executing query at Line 1
               Commands completed successfully.
               Total execution time: 00:00:00.145
```

Deliverables:

- A SQL script with the table creation and queries for data exploration.
- Screenshots of the queries and their results.
- Upload the SQL file (query.sql) and the document to GitHub.