Bank Management System Schema (PostgreSQL & SQLite)

PostgreSQL Schema

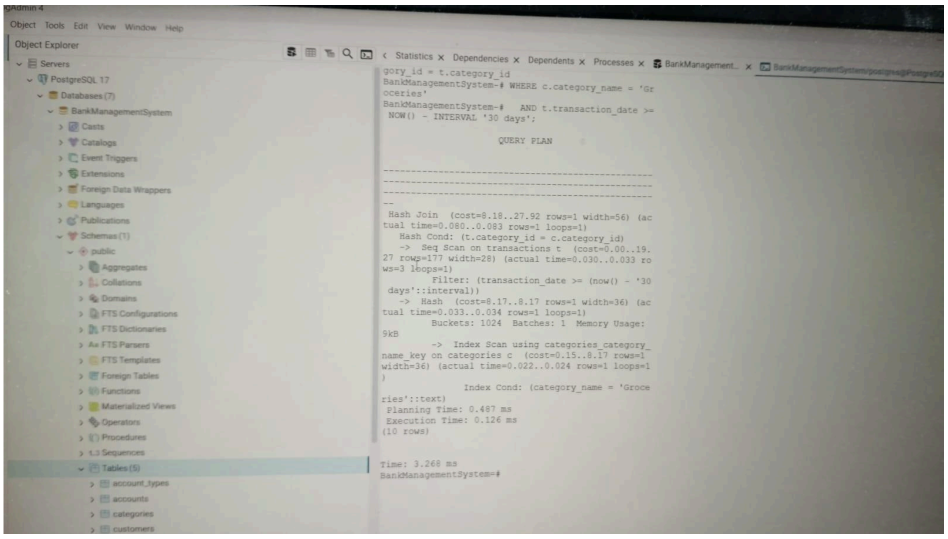
```
-- Enable uuid-ossp extension
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE IF NOT EXISTS customers (
  customer_id      UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  first_name       TEXT NOT NULL,
  last_name        TEXT NOT NULL,
  date_of_birth    DATE,
  phone_number     TEXT UNIQUE,
  email            TEXT UNIQUE,
  adhaar_id        TEXT UNIQUE
);

CREATE TABLE IF NOT EXISTS account_types (
  account_type_id    SERIAL PRIMARY KEY,
  type_name          TEXT NOT NULL UNIQUE,
  interest_rate      NUMERIC(5,2) NOT NULL,
  withdrawal_limit   NUMERIC(12,2),
  overdraft_limit    NUMERIC(12,2)
);

CREATE TABLE IF NOT EXISTS accounts (
  account_id              UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  customer_id             UUID NOT NULL REFERENCES customers(customer_id),
  account_type_id         INT  NOT NULL REFERENCES
account_types(account_type_id),
  balance                 NUMERIC(12,2) NOT NULL DEFAULT 0,
  open_date               DATE NOT NULL DEFAULT CURRENT_DATE,
  status                  TEXT NOT NULL CHECK (status IN
('OPEN','BLOCKED','CLOSED')),
  last_interest_calc_date DATE
);

CREATE TABLE IF NOT EXISTS categories (
  category_id    SERIAL PRIMARY KEY,
  category_name  TEXT NOT NULL UNIQUE,
  is_income      BOOLEAN NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS transactions (
  transaction_id    UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  account_id        UUID NOT NULL REFERENCES accounts(account_id),
  transaction_type  TEXT NOT NULL CHECK (transaction_type IN
('DEPOSIT','WITHDRAW','TRANSFER')),
  amount            NUMERIC(12,2) NOT NULL CHECK (amount > 0),
  transaction_date  TIMESTAMPTZ NOT NULL DEFAULT NOW(),
  description       TEXT,
  category_id       INT REFERENCES categories(category_id)
);
```

Optimization Snapshots

Before Optimization

```
-- EXPLAIN ANALYZE
Hash Join  (cost=8.18..27.92 rows=1 width=56) (actual time=0.080..0.083 rows=1
loops=1)
  Hash Cond: (t.category_id = c.category_id)
  ->  Seq Scan on transactions t  (cost=0.00..19.27 rows=177 width=28) (actual
time=0.030..0.033 rows=3 loops=1)
        Filter: (transaction_date >= now() - '30 days'::interval)
  ->  Hash  (cost=8.17..8.17 rows=1 width=36) (actual time=0.033..0.034 rows=1
loops=1)
        ->  Index Scan using categories_category_name_key on categories c
(cost=0.15..8.17 rows=1 width=36) (actual time=0.022..0.024 rows=1 loops=1)
              Index Cond: (category_name = 'Groceries'::text)
```
Planning Time: 0.487 ms
Execution Time: 0.126 ms
Time: 3.268 ms

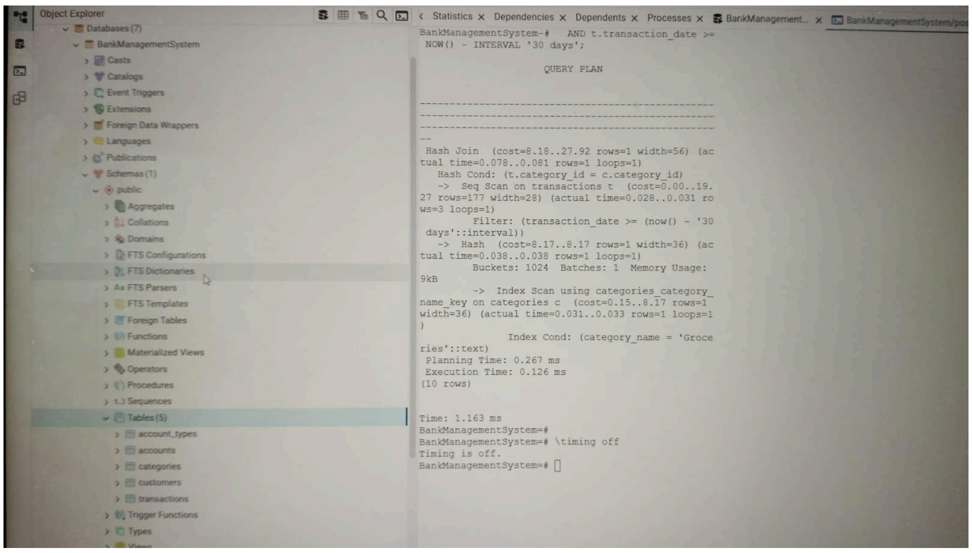


```
gory_id = t.category_id
BankManagementSystem-# WHERE c.category_name = 'Gr
oceries'
BankManagementSystem-#   AND t.transaction_date >=
 NOW() - INTERVAL '30 days';

                         QUERY PLAN

-----------------------------------------------------
-----------------------------------------------------
-----------------------------------------------------
--
 Hash Join  (cost=8.18..27.92 rows=1 width=56) (ac
tual time=0.080..0.083 rows=1 loops=1)
   Hash Cond: (t.category_id = c.category_id)
   ->  Seq Scan on transactions t  (cost=0.00..19.
27 rows=177 width=28) (actual time=0.030..0.033 ro
ws=3 loops=1)
         Filter: (transaction_date >= (now() - '30
 days'::interval))
   ->  Hash  (cost=8.17..8.17 rows=1 width=36) (ac
tual time=0.033..0.034 rows=1 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage:
9kB
         ->  Index Scan using categories_category_
name_key on categories c  (cost=0.15..8.17 rows=1
width=36) (actual time=0.022..0.024 rows=1 loops=1
)
               Index Cond: (category_name = 'Groce
ries'::text)
 Planning Time: 0.487 ms
 Execution Time: 0.126 ms
(10 rows)


Time: 3.268 ms
BankManagementSystem=#
```

After Optimization

```
-- EXPLAIN ANALYZE
Hash Join  (cost=8.18..27.92 rows=1 width=56) (actual time=0.078..0.081 rows=1 loops=1)
  Hash Cond: (t.category_id = c.category_id)
  ->  Seq Scan on transactions t  (cost=0.00..19.27 rows=177 width=28) (actual time=0.028..0.031 rows=3 loops=1)
        Filter: (transaction_date >= now() - '30 days'::interval)
  ->  Hash  (cost=8.17..8.17 rows=1 width=36) (actual time=0.031..0.033 rows=1 loops=1)
        ->  Index Scan using categories_category_name_key on categories c  (cost=0.15..8.17 rows=1 width=36) (actual time=0.031..0.033 rows=1 loops=1)
              Index Cond: (category_name = 'Groceries'::text)
```

Planning Time: 0.267 ms
Execution Time: 0.126 ms
Time: 1.163 ms



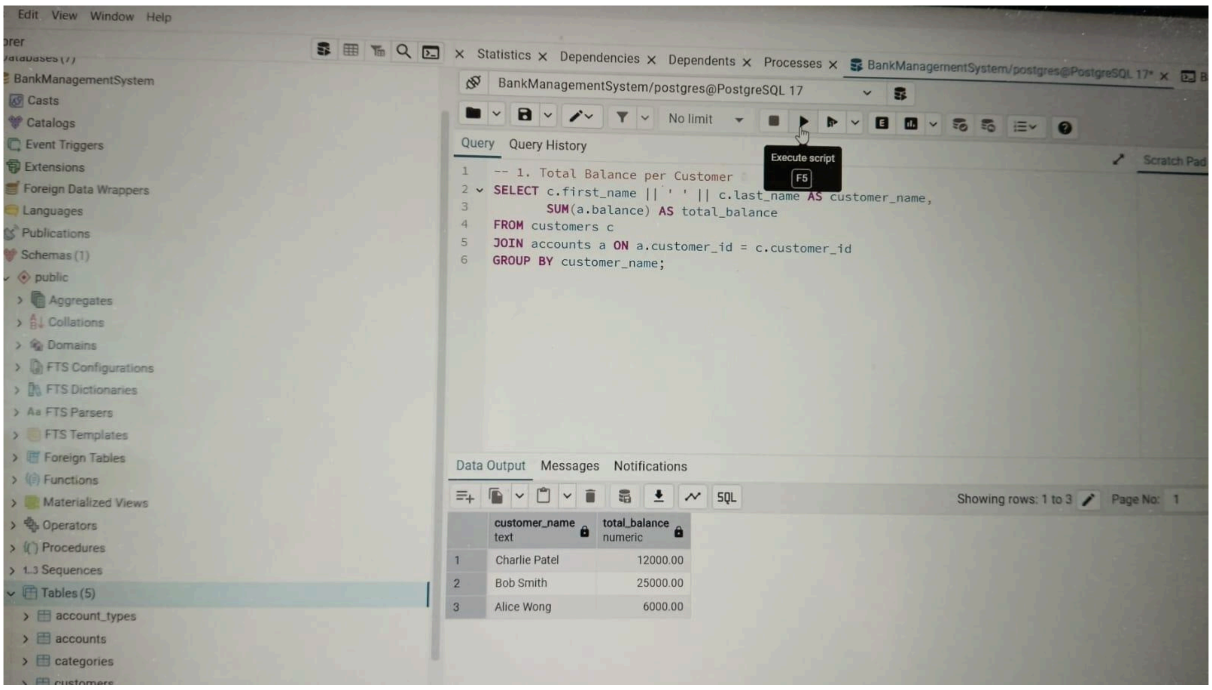
```
BankManagementSystem-#   AND t.transaction_date >=
 NOW() - INTERVAL '30 days';

                     QUERY PLAN


---------------------------------------------------
---------------------------------------------------
---------------------------------------------------
--
 Hash Join  (cost=8.18..27.92 rows=1 width=56) (ac
tual time=0.078..0.081 rows=1 loops=1)
   Hash Cond: (t.category_id = c.category_id)
   ->  Seq Scan on transactions t  (cost=0.00..19.
27 rows=177 width=28) (actual time=0.028..0.031 ro
ws=3 loops=1)
         Filter: (transaction_date >= (now() - '30
 days'::interval))
   ->  Hash  (cost=8.17..8.17 rows=1 width=36) (ac
tual time=0.038..0.038 rows=1 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage:
9kB
         ->  Index Scan using categories_category_
name_key on categories c  (cost=0.15..8.17 rows=1
width=36) (actual time=0.031..0.033 rows=1 loops=1
)
               Index Cond: (category_name = 'Groce
ries'::text)
 Planning Time: 0.267 ms
 Execution Time: 0.126 ms
(10 rows)


Time: 1.163 ms
BankManagementSystem=#
BankManagementSystem=# \timing off
Timing is off.
BankManagementSystem=#
```

SQLite-Compatible Schema

```
-- SQLite schema

CREATE TABLE IF NOT EXISTS customers (
  customer_id INTEGER PRIMARY KEY AUTOINCREMENT,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL,
  date_of_birth TEXT,
  phone_number TEXT UNIQUE,
  email TEXT UNIQUE,
  adhaar_id TEXT UNIQUE
);

CREATE TABLE IF NOT EXISTS account_types (
  account_type_id INTEGER PRIMARY KEY AUTOINCREMENT,
  type_name TEXT NOT NULL UNIQUE,
  interest_rate REAL NOT NULL,
  withdrawal_limit REAL,
  overdraft_limit REAL
);

CREATE TABLE IF NOT EXISTS accounts (
  account_id TEXT PRIMARY KEY,
  customer_id TEXT NOT NULL,
  account_type_id INTEGER NOT NULL,
  balance REAL NOT NULL DEFAULT 0,
  open_date TEXT NOT NULL DEFAULT (date('now')),
  status TEXT NOT NULL CHECK (status IN ('OPEN','BLOCKED','CLOSED')),
  last_interest_calc_date TEXT,
  FOREIGN KEY(customer_id) REFERENCES customers(customer_id),
  FOREIGN KEY(account_type_id) REFERENCES account_types(account_type_id)
);

CREATE TABLE IF NOT EXISTS categories (
  category_id INTEGER PRIMARY KEY AUTOINCREMENT,
  category_name TEXT NOT NULL UNIQUE,
  is_income INTEGER NOT NULL CHECK (is_income IN (0,1))
);

CREATE TABLE IF NOT EXISTS transactions (
  transaction_id TEXT PRIMARY KEY,
  account_id TEXT NOT NULL,
  transaction_type TEXT NOT NULL CHECK (transaction_type IN
('DEPOSIT','WITHDRAW','TRANSFER')),
  amount REAL NOT NULL CHECK (amount > 0),
  transaction_date TEXT NOT NULL DEFAULT (datetime('now')),
  description TEXT,
  category_id INTEGER,
  FOREIGN KEY(account_id) REFERENCES accounts(account_id),
  FOREIGN KEY(category_id) REFERENCES categories(category_id)
);
```

Postgresql Queries:

1. Total Balance per Customer

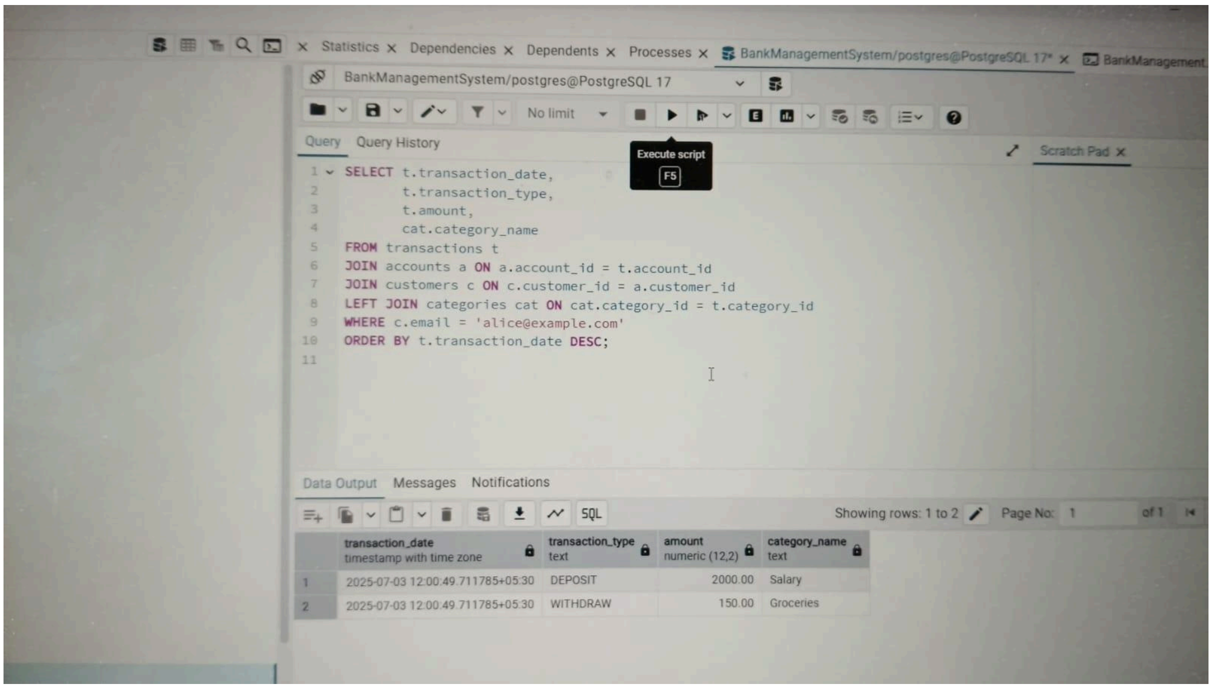


```sql
-- 1. Total Balance per Customer
SELECT c.first_name || ' ' || c.last_name AS customer_name,
       SUM(a.balance) AS total_balance
FROM customers c
JOIN accounts a ON a.customer_id = c.customer_id
GROUP BY customer_name;
```

Showing rows: 1 to 3

	customer_name text	total_balance numeric
1	Charlie Patel	12000.00
2	Bob Smith	25000.00
3	Alice Wong	6000.00

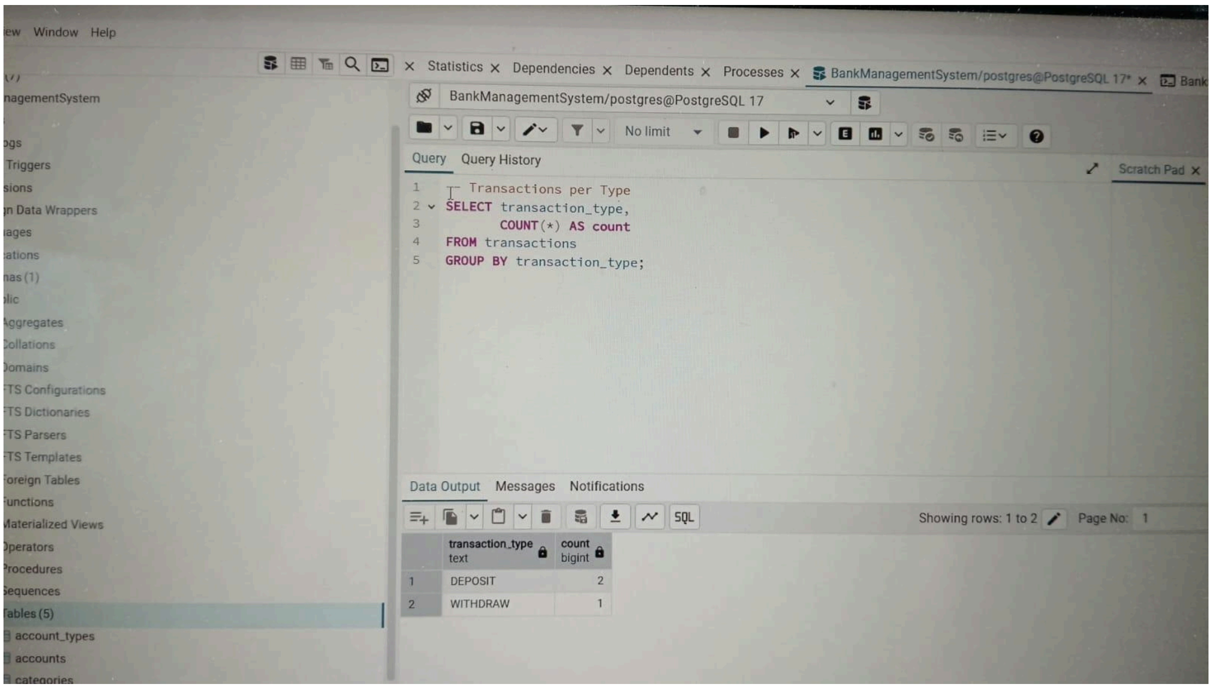
2. Transactions for a Specific Customer



```
SELECT t.transaction_date,
       t.transaction_type,
       t.amount,
       cat.category_name
FROM transactions t
JOIN accounts a ON a.account_id = t.account_id
JOIN customers c ON c.customer_id = a.customer_id
LEFT JOIN categories cat ON cat.category_id = t.category_id
WHERE c.email = 'alice@example.com'
ORDER BY t.transaction_date DESC;
```

	transaction_date timestamp with time zone	transaction_type text	amount numeric (12,2)	category_name text
1	2025-07-03 12:00:49.711785+05:30	DEPOSIT	2000.00	Salary
2	2025-07-03 12:00:49.711785+05:30	WITHDRAW	150.00	Groceries

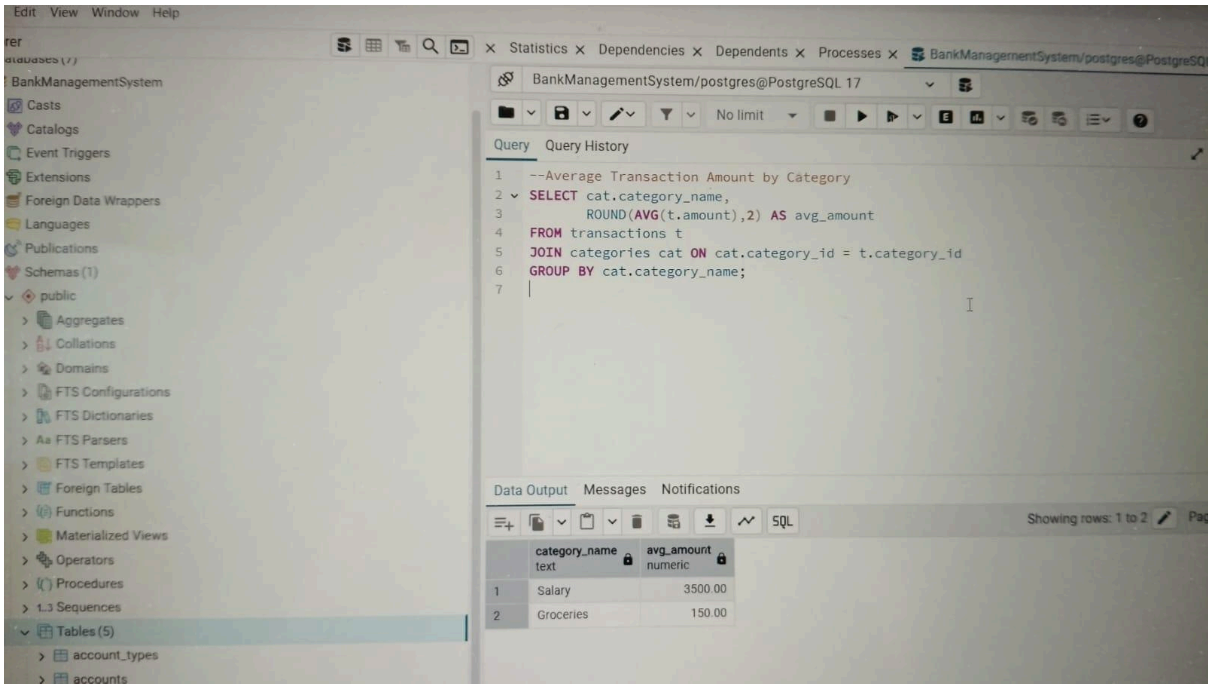
3. Count of Transactions by Type



```
-- Transactions per Type
SELECT transaction_type,
       COUNT(*) AS count
FROM transactions
GROUP BY transaction_type;
```

	transaction_type text	count bigint
1	DEPOSIT	2
2	WITHDRAW	1

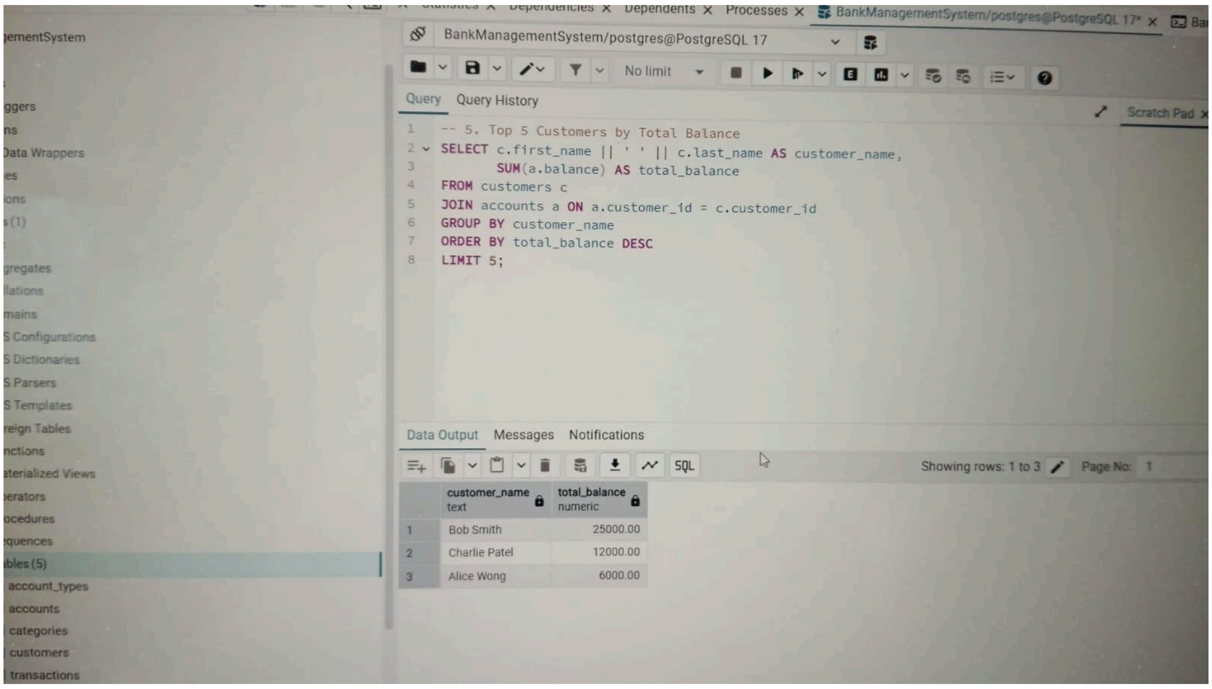
4. Average Transaction Amount by Category



```
--Average Transaction Amount by Category
SELECT cat.category_name,
       ROUND(AVG(t.amount),2) AS avg_amount
FROM transactions t
JOIN categories cat ON cat.category_id = t.category_id
GROUP BY cat.category_name;
```

	category_name text	avg_amount numeric
1	Salary	3500.00
2	Groceries	150.00

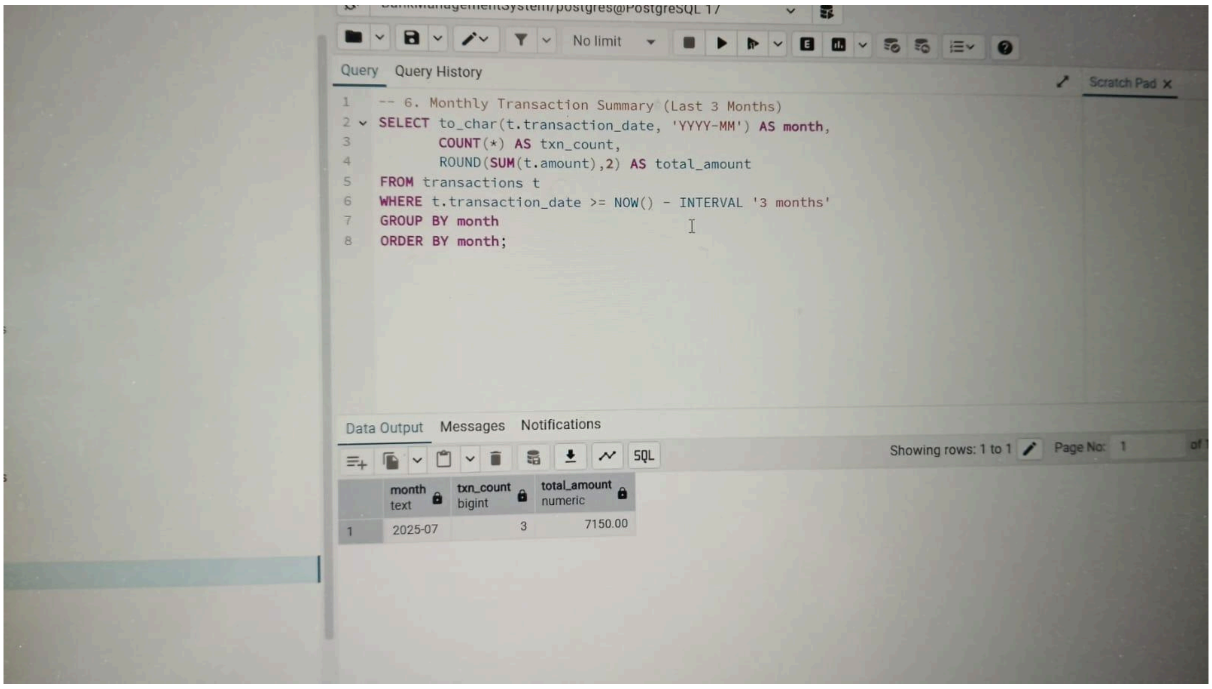
5. Top 5 Customers by Total Balance



```
-- 5. Top 5 Customers by Total Balance
SELECT c.first_name || ' ' || c.last_name AS customer_name,
       SUM(a.balance) AS total_balance
FROM customers c
JOIN accounts a ON a.customer_id = c.customer_id
GROUP BY customer_name
ORDER BY total_balance DESC
LIMIT 5;
```

	customer_name text	total_balance numeric
1	Bob Smith	25000.00
2	Charlie Patel	12000.00
3	Alice Wong	6000.00

6. Monthly Transaction Summary (Last 3 Months)



```sql
-- 6. Monthly Transaction Summary (Last 3 Months)
SELECT to_char(t.transaction_date, 'YYYY-MM') AS month,
       COUNT(*) AS txn_count,
       ROUND(SUM(t.amount),2) AS total_amount
FROM transactions t
WHERE t.transaction_date >= NOW() - INTERVAL '3 months'
GROUP BY month
ORDER BY month;
```

	month text	txn_count bigint	total_amount numeric
1	2025-07	3	7150.00