

**UNIVERSITY OF DAYTON**

**DATEBASE DESIGN**

**OF**

**BANK MANAGEMENT SYSTEM**

**FINAL REPORT**

**BY**

KAKANURU HARSHITHA 101790901

KUNDURI SAI NIKHIL REDDY 101791579

GANNE ESWAR CHANDRA VIDYA SAGAR 101792457

GARIKAPATI CHANDANIKA 101791574

MUNIGANTI VENU 101792514

UNDER THE ESTEEMED GUIDANCE

OF

DR. EMAN ELRIFAEI

**DATABASE MANAGEMENT SYSTEMS**

**CPS 542**

**DEPARTMENT OF COMPUTER SCIENCE**

**INTRODUCTION**

A bank management system is an essential piece of software that keeps banking organizations running smoothly. It includes a number of vital components, each of which contributes in a unique way to maintaining operational effectiveness and promoting smooth customer relations.

Customers, who are essential to the banking process, are at the center of the system. Clients are recognized by their individual IDs and full names; contact information, including phone numbers and addresses, is kept on file for correspondence and validation. These particulars serve as the foundation for handling client accounts and offering tailored services.

Branches act as the actual points of contact for clients to obtain banking services. Each branch is distinguished from the others by its name, contact details, and hours of operation. Furthermore, information is kept on branch types (such as corporate and retail) and addresses in order to properly classify branches and guarantee client accessibility.

Customers interact with the bank's services through their accounts, which are their financial conduits. Every account is given a special number and categorized according to its kind, such checking, or savings. Funds included therein are represented by account balances, and their operating status is indicated by status indicators. This organization gives clients effective financial management tools and gives the bank vital information for financial supervision and service delivery.

**ENTITIES AND ITS ATTRIBUTES**

**Customer:** Represents people with individual identity numbers such as SSN, State ID, and complete names, phone numbers, and addresses that include the apartment number, street, city, state, and zip code.

**Attributes:**

**Customer ID :** Unique Id of a customer. **First Name :** Customer first name

**Last Name :** Customer last name **Phone Number:** Customer Mobile number

**Identification Number :** Unique ID number such as SSN, State ID

**Composite Attribute :** In a database or data model, a composite attribute is a kind of attribute that is made up of several related sub-attributes combined together. These sub-characteristics are regarded as a unified entity.

**Address :** Address of the customer

* Apt Number
* Street
* City
* State
* Zip code

**Derived Attribute:** An attribute in a database or data model that is not explicitly saved, but instead has a value that may be computed or derived from other properties, is called a derived attribute.

**Birth Year:** Year of the birth of Customer.

* **Age**

**Account:** contains information on account numbers, kinds, balances, statuses, and opening dates for financial holding management.

**Attributes:**

**Account Number:** Unique Account Number **Account type:** Type of account checking, savings etc.. **Balance:** Indicates the total amount in account.

**Account Status:** Indicates the status of account i.e. active etc.…

**Open Date :** open date of account

**Services:** Keeps track of transactional activity details such as service IDs, types, amounts, dates, and statuses.

**Attributes:**

**Service ID:** Unique service ID is generated for specific service to perform.  
 **Service Type:** type of service such as deposit, withdrawal etc. **Amount:** On what amount the service is performed

**Date:** the date of services is done

**Status:** status of that service

**Employees:** maintains a record of each employee's ID, full name, title, pay, and contacts, including emails and phone numbers.

**Attributes:**

**Employee ID:** Unique ID of employee **First Name:** First name of employee

**Last Name:** Last name of employee **Designation:** Role of employee

**Salary:** salary of individual

**Composite Attribute:**

**Contact:** contact information of employee

* **Phone Number**
* **Email**

**Details:** keeps track of information for transaction records, including IDs, descriptions, dates, locations, and amounts.

**Attributes:**

**Details ID:** Detail ID is generated for that particular service **Description:** description of that service **Date:** specifies the date the details is stored

**Location:** stores the location

**Branch :** Branches are arranged according to their IDs, names, types, operating hours, contacts (phone and email), and addresses.

**Attributes:**

**Branch ID:** specifies every branch with unique ID **Branch Name:** name of the branch **Operating Hours:** operating hours of that branch

**Branch Type: s**pecifies type of branch.

**Composite Attribute :**

**Branch Contact:** branch contact information

* **Phone Number**
* **Email**

**Address :** address of branches

* **Apt Number**
* **Street**
* **City**
* **State**
* **Zip code**

**Feedback:** Records comments, ratings, dates, and IDs for customer evaluations in feedback.

**Attributes:**

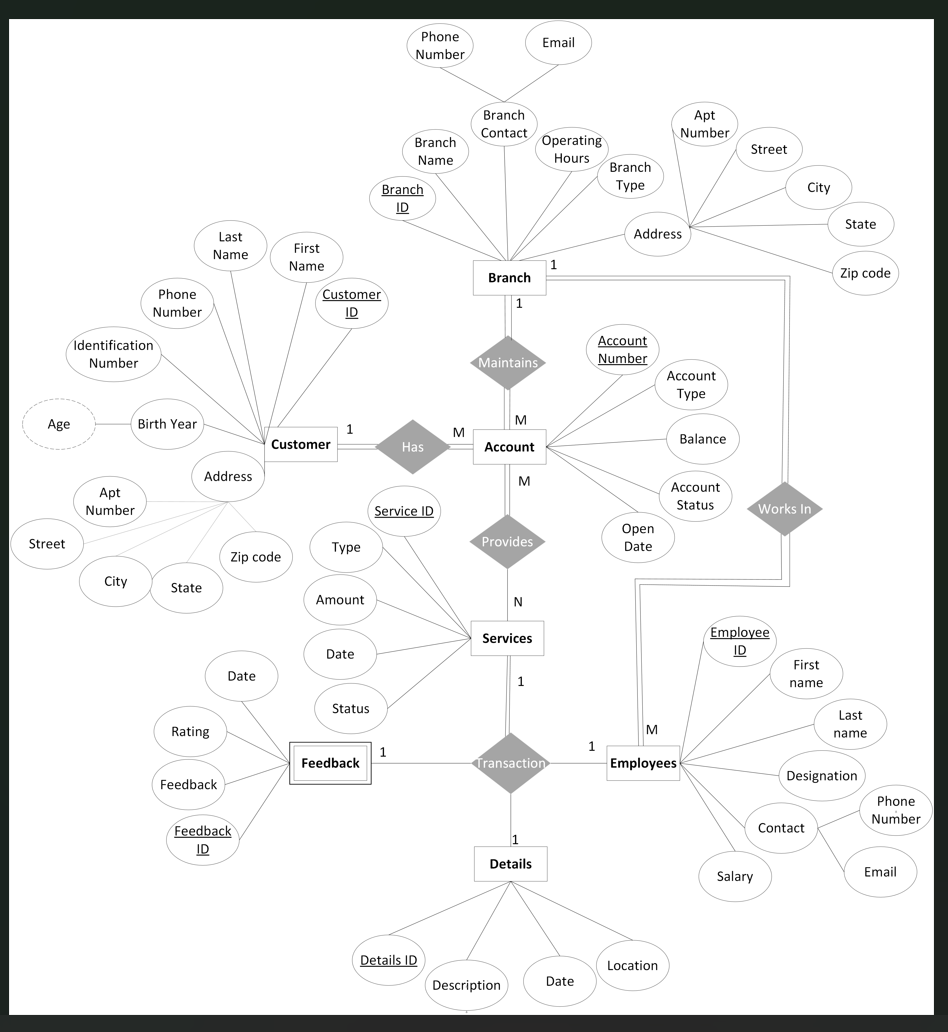
**Feedback ID:** feedback ID is generated for every feedback **Rating:** rating of service. **Feedback:** actual comments

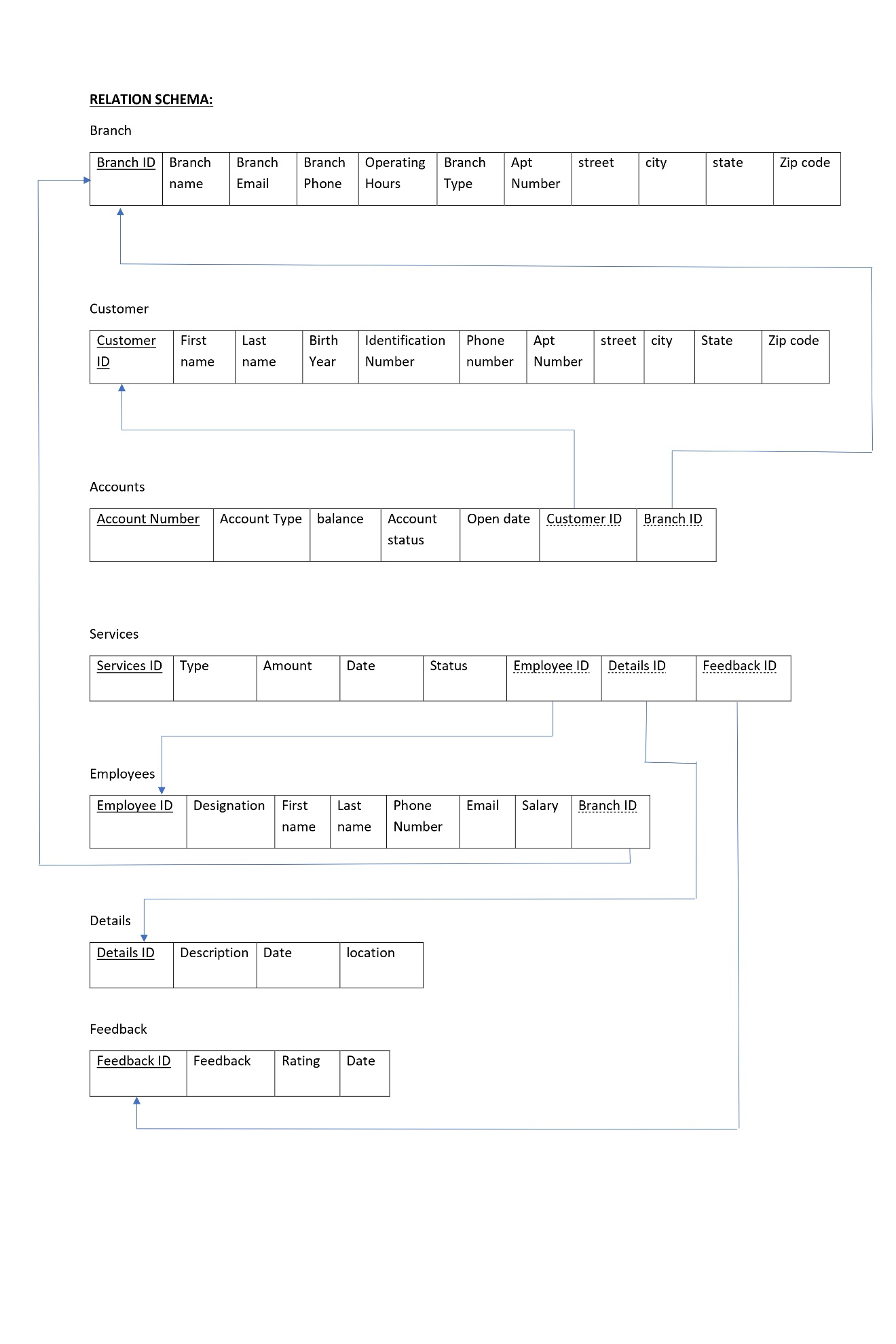
**Date:** recorded date

**DESIGN ASSUMPTIONS:**

* The bank provides a range of services to cater to the diverse needs of its clients, including checking, savings, and loan accounts.
* The system of the bank manages a variety of services, including loan payments, transfers, withdrawals, and deposits, making sure that clients may conveniently complete all required transactions.
* The bank stores all of its data from several branches in a single central database, which simplifies system design, implementation, and management.

**ER DIAGRAM:**





**RELATIONS:**

Customer has Account ( one to many )

Branch Maintains Account ( one to many )

Employee Works in Branch ( many to one )

Account Provides services ( many to many )

Services transaction details ( one to one )

Services transaction Feedback ( one to one )

Services transaction Employees ( one to one )

**NORMALIZATION OF ENTITIES:**

**Normalization:** The process of normalizing a relational database involves organizing it according to several normal forms in order to get rid of dependencies and redundancies.

**First Normal Form (1NF):**

Atomic values are present in every column, hence there are no repeated arrays or groups.

For instance, if a table contains the characteristics A and B and A decides B, then the table is in 1NF.

**Second Normal Form (2NF):**

Since there are no partial dependencies, every non-prime attribute depends entirely on the primary key.

For instance, A table in 2NF has the following properties: A, B, and C. A is the primary key, and B depends on A.

**Third Normal Form (3NF):**

Non-key characteristics solely depend on the primary key and not on other non-key attributes because there are no transitive dependencies.

For instance, A table is in 3NF if it has the following attributes: A is the primary key, B depends on A, and C depends on B.

**Boyce-Codd Normal Form (BCNF):**

There are no non-trivial functional dependencies on candidate keys because every determinant is a candidate key.

For instance, A table is in BCNF if each determinant in its functional dependencies is a potential key.

**Customer :**

Assumptions: Phone number is not multi-valued attribute, i.e each customer has utmost 1 phone number.

**Primary Key:{Customer ID}**

**Customer(Customer ID, First Name, Last Name, Phone Number, Identification Number ,Birth Year, apt number, street, city , state, zipcode)**

**First Normal Form (1NF):**

Customer relation is in 1NF as each column contains atomic values and each column has values that are of same type.

Customer(Customer ID, First Name, Last Name, phone\_number, identification\_number , apt number, street, city , state, zipcode)

**Second Normal Form (2NF):**

Customer relation is in 2NF because it is in 1NF and all non-key attributes are fully functionally dependent on the primary key(Customer ID)

Customer(Customer ID,First Name, Last Name, phone\_number, identification\_number , apt number, street, city , state, zipcode)

**Third Normal Form (3NF):**

Customer relation is not in 3NF because there is transitive dependency.

Here non-key attribute Zipcode determines other non-key attributes city, state.

CustomerID->All attributes

zipcode->city, state.

To achieve 3NF , perform projection on original table.

**Resulting Schema:**

Customer(Customer ID, First Name, Last Name, phone\_number, Identification Number, Birth Year, apt number, street, *zipcode*)

Zips(zipcode,city,state)

**Boyce-Codd Normal Form (BCNF):**

A relation is in Boyce-Codd normal form (BCNF) if, whenever a non-trivial functional dependency X → A exists, then X is a super key. Hence Customer, Zips relations are in BCNF.

**Branch:**

**Primary Key: BranchID**

**Branch(BranchID, Branch Name, Branch Email, Phone Number, Operating Hours, Branch Type, Apt Number, Street, city, state, zipcode)**

**First Normal Form (1NF):**

Branch relation is in 1NF as each column contains atomic values and each column has values that are of same type.

**Second Normal Form (2NF):**

Branch relation is in 2NF as all non-key attributes are fully functionally dependent on the primary key.

BranchID ->All attributes

**Third Normal Form (3NF):**

Branch relation is not in 3NF because there is transitive dependency.

Here non-key attribute Zipcode determines other non-key attributes.

BranchID->All attributes

zipcode->city, state.

To achieve 3NF , perform projection on Branch table.

**Resulting Schema:**

Branch(BranchID, Branch Name, Branch Email, Phone Number, Operating Hours, Branch Type, Apt Number, Street, City, State, *zipcode*)

Zips(zipcode, city, state)

**Boyce-Codd Normal Form (BCNF):**

A relation is in Boyce-Codd normal form (BCNF) if, whenever a non-trivial functional dependency X → A exists, then X is a super key. Hence Customer, Zips relations are in BCNF

**Accounts:**

**Primary Key: Account Number**

**Accounts(Account Number, Account Type, Balance, Account Status, Open Date, *CustomerID*, *BranchID*)**

**First Normal Form (1NF):**

Accounts relation is in 1NF as each column contains atomic values and each column has values that are of same type.

**Second Normal Form (2NF):**

Accounts relation is in 2NF as it is in 1NF and all non-key attributes are fully functionally dependent on the primary key(Account Number)

**Third Normal Form (3NF):**

Accounts relation is in 3NF as it is in 2NF and there are no transitive dependencies.

**Boyce-Codd Normal Form (BCNF):**

A relation is in Boyce-Codd normal form (BCNF) if, whenever a non-trivial functional dependency X → A exists, then X is a super key. Hence Account relation is in BCNF.

**Services:**

**Primary Key: Services ID**

**Services(Services ID, Type, Amount, Date, Status, *Employee\_ID, Details\_ID, FeedbackID)***

**First Normal Form (1NF):**

Services relation is in 1NF as each column contains atomic values and each column has values that are of same type.

**Second Normal Form (2NF):**

Services relation is in 2NF as it is in 1NF and all non-key attributes are fully functionally dependent on the primary key(Account Number)

**Third Normal Form (3NF):**

Services relation is in 3NF as it is in 2NF and there are no transitive dependencies.

**Boyce-Codd Normal Form (BCNF):**

A relation is in Boyce-Codd normal form (BCNF) if, whenever a non-trivial functional dependency X → A exists, then X is a super key. Hence Services relation is in BCNF.

**Employees:**

**Employees(Employee\_ID, Designation, First Name, Last Name Salary, Phone Number, Email, *BranchID*)**

**First Normal Form (1NF):**

Employee relation is in 1NF as each column contains atomic values and each column has values that are of same type.

**Second Normal Form (2NF):**

Employee relation is in 2NF as it is in 1NF and all non-key attributes are fully functionally dependent on the primary key(Account Number)

**Third Normal Form (3NF):**

Employee relation is in 3NF as it is in 2NF and there are no transitive dependencies.

**Boyce-Codd Normal Form (BCNF):**

A relation is in Boyce-Codd normal form (BCNF) if, whenever a non-trivial functional dependency X → A exists, then X is a super key. Hence Employee relation is in BCNF.

**Details:**

**Details(Details\_ID, Description, Date, Location)**

**First Normal Form (1NF):**

1NF: Details relation is in 1NF as each column contains atomic values and each column has values that are of same type.

**Second Normal Form (2NF):**

Details relation is in 2NF as it is in 1NF and all non-key attributes are fully functionally dependent on the primary key(Account Number)

**Third Normal Form (3NF):**

Details relation is in 3NF as it is in 2NF and there are no transitive dependencies.

**Boyce-Codd Normal Form (BCNF):**

A relation is in Boyce-Codd normal form (BCNF) if, whenever a non-trivial functional dependency X → A exists, then X is a super key. Hence Details relation is in BCNF.

**Feedback:**

Feedback(FeedbackID, Feedback, Rating, Date)

**First Normal Form (1NF):**

Details relation is in 1NF as each column contains atomic values and each column has values that are of same type.

**Second Normal Form (2NF):**

Details relation is in 2NF as it is in 1NF and all non-key attributes are fully functionally dependent on the primary key(Account Number)

**Third Normal Form (3NF):**

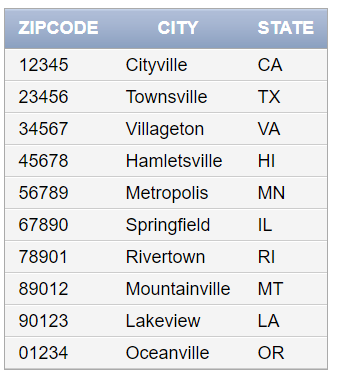
Details relation is in 3NF as it is in 2NF and there are no transitive dependencies.

**Boyce-Codd Normal Form (BCNF):**

A relation is in Boyce-Codd normal form (BCNF) if, whenever a non-trivial functional dependency X → A exists, then X is a super key. Hence Details relation is in BCNF.

**AFTER NORMALIZATION TABLE SCHEMA:**

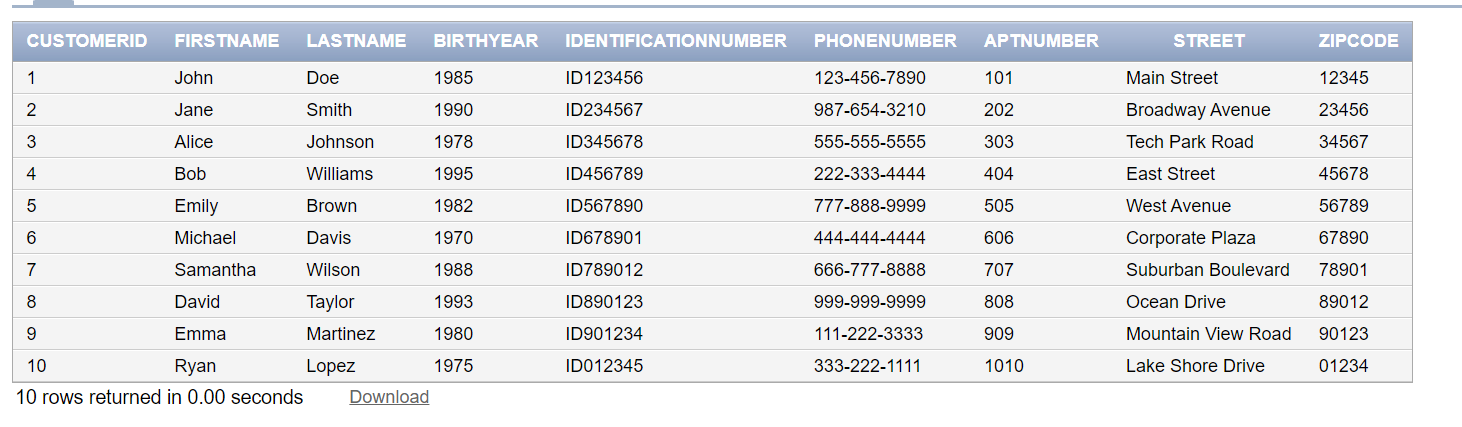
**Zips:**

****

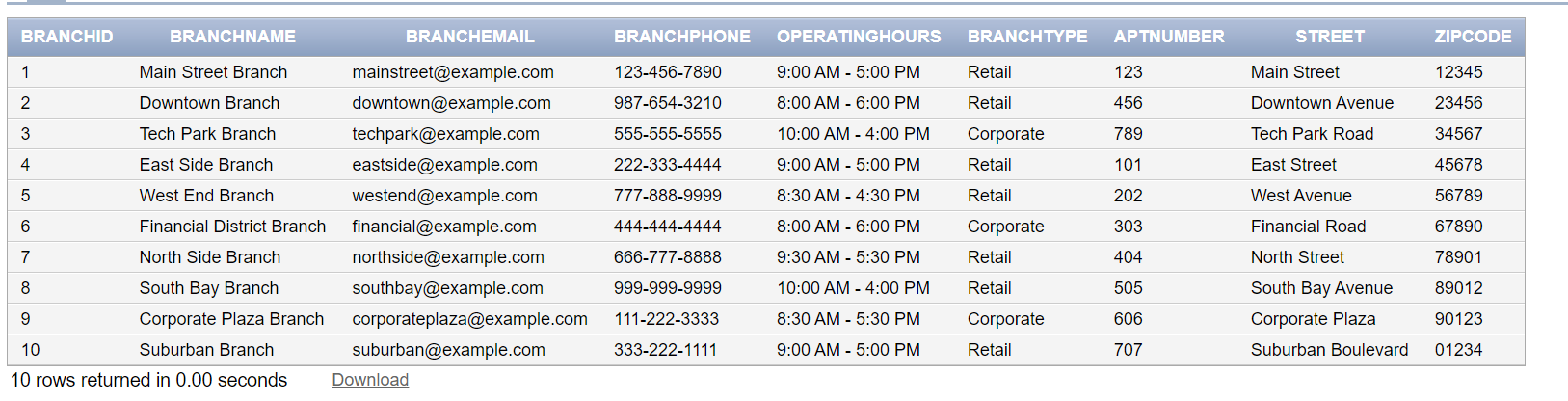
**Feedback:**

****

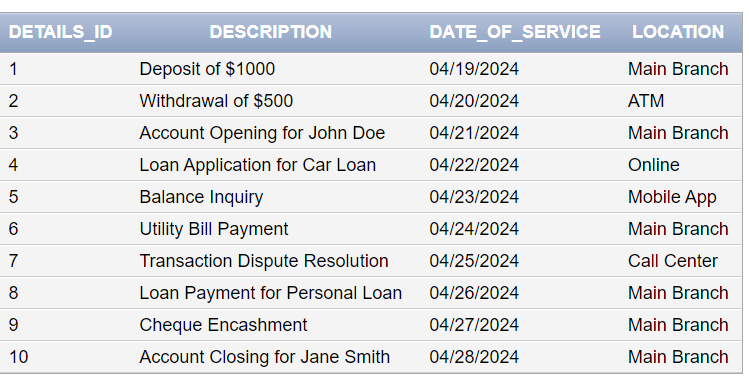
**Customer:**

****

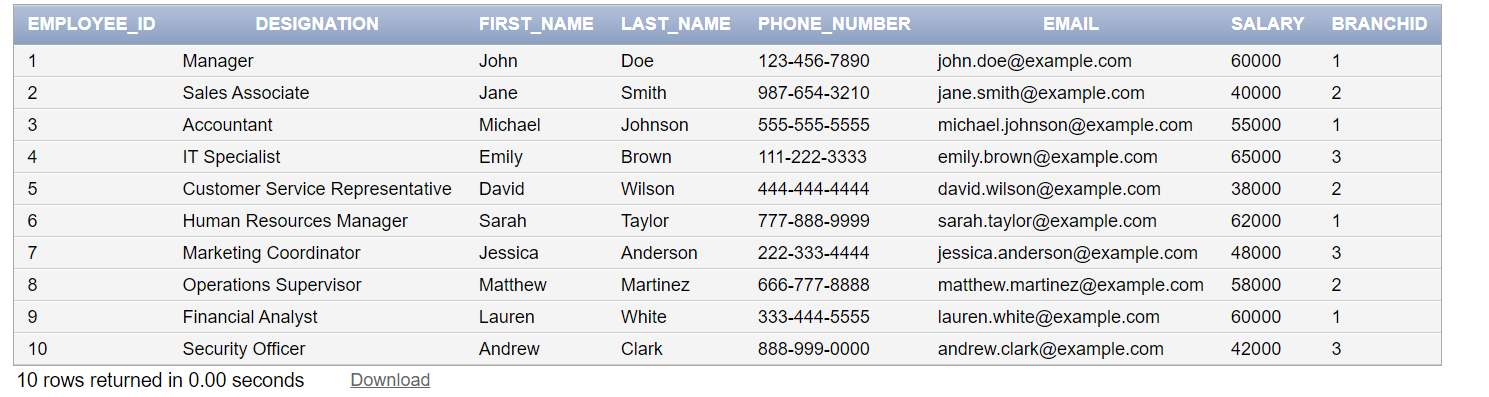
**Branch:**

****

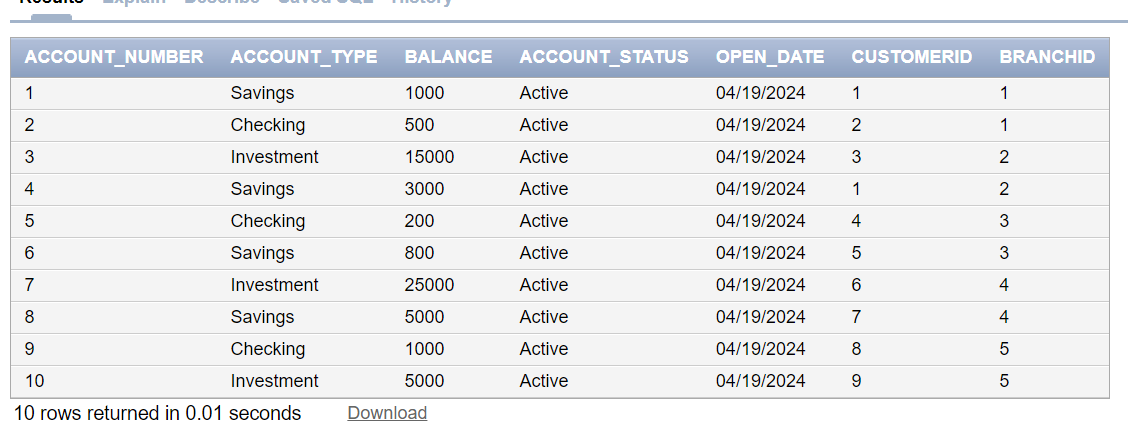
**Details:**

****

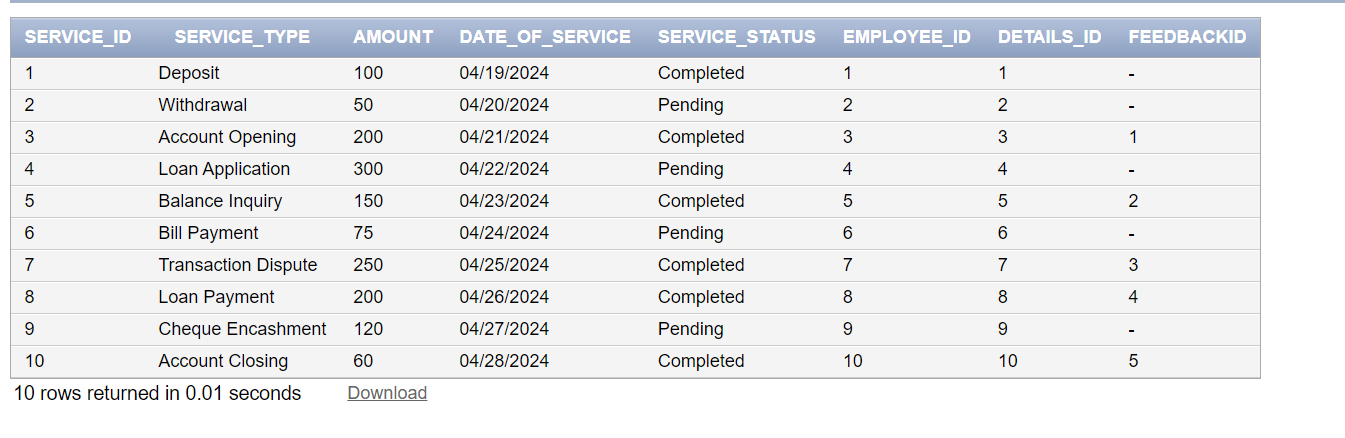
**Employees:**

****

**Accounts:**

****

**Services:**

****

**PHYSICAL MODEL:**

**CREATE TABLES ----**

**Customer Entity:**

CREATE TABLE Customer (

    CustomerID NUMBER PRIMARY KEY,

    FirstName VARCHAR2(50),

    LastName VARCHAR2(50),

    BirthYear NUMBER(4),

    IdentificationNumber VARCHAR2(20),

    PhoneNumber VARCHAR2(20),

    AptNumber VARCHAR2(20),

    Street VARCHAR2(255),

    Zipcode VARCHAR2(20),

FOREIGN KEY (Zipcode) REFERENCES Zips(Zipcode)

);

**Zipcode Table:**

CREATE TABLE Zips (

    Zipcode VARCHAR(10) PRIMARY KEY,

    City VARCHAR(255),

    State VARCHAR(255)

);

**Branch Entity:**

CREATE TABLE Branch (

    BranchID NUMBER PRIMARY KEY,

    BranchName VARCHAR2(255),

    BranchEmail VARCHAR2(255),

    BranchPhone VARCHAR2(20),

    OperatingHours VARCHAR2(100),

    BranchType VARCHAR2(50),

    AptNumber VARCHAR2(50),

    Street VARCHAR2(255),

    Zipcode VARCHAR2(20),

FOREIGN KEY (Zipcode) REFERENCES Zips(Zipcode)

);

**Employees Entity:**

CREATE TABLE Employees (

    Employee\_ID NUMBER PRIMARY KEY,

    Designation VARCHAR2(50),

    First\_Name VARCHAR2(50),

    Last\_Name VARCHAR2(50),

    Phone\_Number VARCHAR2(20),

    Email VARCHAR2(100),

    Salary NUMBER,

    BranchID NUMBER,

    FOREIGN KEY (BranchID) REFERENCES Branches(BranchID)

);

**Accounts Entity:**

CREATE TABLE Accounts (

    Account\_Number NUMBER PRIMARY KEY,

    Account\_Type VARCHAR2(50),

    Balance NUMBER,

    Account\_Status VARCHAR2(20),

    Open\_Date DATE,

    CustomerID NUMBER,

    BranchID NUMBER,

    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

    FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)

);

**Services Entity:**

CREATE TABLE Bank\_Services (

    Service\_ID NUMBER PRIMARY KEY,

    Service\_Type VARCHAR2(50),

    Amount NUMBER,

    Date\_of\_Service DATE,

    Service\_Status VARCHAR2(20),

    Employee\_ID NUMBER,

    Details\_ID NUMBER,

    FeedbackID NUMBER,

    FOREIGN KEY (Employee\_ID) REFERENCES Employees(Employee\_ID),

    FOREIGN KEY (Details\_ID) REFERENCES Service\_Details(Details\_ID),

    FOREIGN KEY (FeedbackID) REFERENCES Feedback(FeedbackID)

);

**Details Entity:**

CREATE TABLE Service\_Details (

    Details\_ID NUMBER PRIMARY KEY,

    Description VARCHAR2(100),

    Date\_of\_Service DATE,

    Location VARCHAR2(100)

);

**Feedback Entity:**

CREATE TABLE Feedback (

    FeedbackID NUMBER PRIMARY KEY,

    FeedbackText VARCHAR2(255),

    Rating NUMBER,

    FeedbackDate DATE

);

**INSERTING DATA INTO TABLES**

**Customer Entity**

INSERT INTO Customer (CustomerID, FirstName, LastName, BirthYear, IdentificationNumber, PhoneNumber, AptNumber, Street, Zipcode)

VALUES

    (1, 'John', 'Doe', 1985, 'ID123456', '123-456-7890', '101', 'Main Street','12345'),

    (2, 'Jane', 'Smith', 1990, 'ID234567', '987-654-3210', '202', 'Broadway Avenue', '23456'),

    (3, 'Alice', 'Johnson', 1978, 'ID345678', '555-555-5555', '303', 'Tech Park Road', '34567'),

    (4, 'Bob', 'Williams', 1995, 'ID456789', '222-333-4444', '404', 'East Street', '45678'),

    (5, 'Emily', 'Brown', 1982, 'ID567890', '777-888-9999', '505', 'West Avenue', '56789'),

    (6, 'Michael', 'Davis', 1970, 'ID678901', '444-444-4444', '606', 'Corporate Plaza', '67890'),

    (7, 'Samantha', 'Wilson', 1988, 'ID789012', '666-777-8888', '707', 'Suburban Boulevard', '78901'),

    (8, 'David', 'Taylor', 1993, 'ID890123', '999-999-9999', '808', 'Ocean Drive', '89012'),

    (9, 'Emma', 'Martinez', 1980, 'ID901234', '111-222-3333', '909', 'Mountain View Road', '90123'),

    (10, 'Ryan', 'Lopez', 1975, 'ID012345', '333-222-1111', '1010', 'Lake Shore Drive', '01234');

**Branch Entity:**

INSERT INTO Branch (BranchID, BranchName, BranchEmail, BranchPhone, OperatingHours, BranchType, AptNumber, Street, Zipcode)

VALUES

    (1, 'Main Street Branch', 'mainstreet@example.com', '123-456-7890', '9:00 AM - 5:00 PM', 'Retail', '123', 'Main Street', '12345'),

    (2, 'Downtown Branch', 'downtown@example.com', '987-654-3210', '8:00 AM - 6:00 PM', 'Retail', '456', 'Downtown Avenue', '23456'),

    (3, 'Tech Park Branch', 'techpark@example.com', '555-555-5555', '10:00 AM - 4:00 PM', 'Corporate', '789', 'Tech Park Road', '34567'),

    (4, 'East Side Branch', 'eastside@example.com', '222-333-4444', '9:00 AM - 5:00 PM', 'Retail', '101', 'East Street', '45678'),

    (5, 'West End Branch', 'westend@example.com', '777-888-9999', '8:30 AM - 4:30 PM', 'Retail', '202', 'West Avenue', '56789'),

    (6, 'Financial District Branch', 'financial@example.com', '444-444-4444', '8:00 AM - 6:00 PM', 'Corporate', '303', 'Financial Road', '67890'),

    (7, 'North Side Branch', 'northside@example.com', '666-777-8888', '9:30 AM - 5:30 PM', 'Retail', '404', 'North Street', '78901'),

    (8, 'South Bay Branch', 'southbay@example.com', '999-999-9999', '10:00 AM - 4:00 PM', 'Retail', '505', 'South Bay Avenue', '89012'),

    (9, 'Corporate Plaza Branch', 'corporateplaza@example.com', '111-222-3333', '8:30 AM - 5:30 PM', 'Corporate', '606', 'Corporate Plaza', '90123'),

    (10, 'Suburban Branch', 'suburban@example.com', '333-222-1111', '9:00 AM - 5:00 PM', 'Retail', '707', 'Suburban Boulevard', '01234');

**Zips Table:**

INSERT INTO Zips (Zipcode, city, state)

SELECT '12345', 'Cityville', 'CA' FROM DUAL

UNION ALL

SELECT '23456', 'Townsville', 'TX' FROM DUAL

UNION ALL

SELECT '34567', 'Villageton', 'VA' FROM DUAL

UNION ALL

SELECT '45678', 'Hamletsville', 'HI' FROM DUAL

UNION ALL

SELECT '56789', 'Metropolis', 'MN' FROM DUAL

UNION ALL

SELECT '67890', 'Springfield', 'IL' FROM DUAL

UNION ALL

SELECT '78901', 'Rivertown', 'RI' FROM DUAL

UNION ALL

SELECT '89012', 'Mountainville', 'MT' FROM DUAL

UNION ALL

SELECT '90123', 'Lakeview', 'LA' FROM DUAL

UNION ALL

SELECT '01234', 'Oceanville', 'OR' FROM DUAL;

**Employees Entity:**

INSERT INTO Employees (Employee\_ID, Designation, First\_Name, Last\_Name, Phone\_Number, Email, Salary, BranchID)

SELECT

    1, 'Manager', 'John', 'Doe', '123-456-7890', 'john.doe@example.com', 60000, 1 FROM DUAL

UNION ALL

SELECT

    2, 'Sales Associate', 'Jane', 'Smith', '987-654-3210', 'jane.smith@example.com', 40000, 2 FROM DUAL

UNION ALL

SELECT

    3, 'Accountant', 'Michael', 'Johnson', '555-555-5555', 'michael.johnson@example.com', 55000, 1 FROM DUAL

UNION ALL

SELECT

    4, 'IT Specialist', 'Emily', 'Brown', '111-222-3333', 'emily.brown@example.com', 65000, 3 FROM DUAL

UNION ALL

SELECT

    5, 'Customer Service Representative', 'David', 'Wilson', '444-444-4444', 'david.wilson@example.com', 38000, 2 FROM DUAL

UNION ALL

SELECT

    6, 'Human Resources Manager', 'Sarah', 'Taylor', '777-888-9999', 'sarah.taylor@example.com', 62000, 1 FROM DUAL

UNION ALL

SELECT

    7, 'Marketing Coordinator', 'Jessica', 'Anderson', '222-333-4444', 'jessica.anderson@example.com', 48000, 3 FROM DUAL

UNION ALL

SELECT

    8, 'Operations Supervisor', 'Matthew', 'Martinez', '666-777-8888', 'matthew.martinez@example.com', 58000, 2 FROM DUAL

UNION ALL

SELECT

    9, 'Financial Analyst', 'Lauren', 'White', '333-444-5555', 'lauren.white@example.com', 60000, 1 FROM DUAL

UNION ALL

SELECT

    10, 'Security Officer', 'Andrew', 'Clark', '888-999-0000', 'andrew.clark@example.com', 42000, 3 FROM DUAL;

UNION ALL

**Accounts Entity:**

INSERT INTO Accounts (Account\_Number, Account\_Type, Balance, Account\_Status, Open\_Date, CustomerID, BranchID)

VALUES

    (1, 'Savings', 1000.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 1, 1),

    (2, 'Checking', 500.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 2, 1),

    (3, 'Investment', 15000.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 3, 2),

    (4, 'Savings', 3000.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 1, 2),

    (5, 'Checking', 200.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 4, 3),

    (6, 'Savings', 800.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 5, 3),

    (7, 'Investment', 25000.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 6, 4),

    (8, 'Savings', 5000.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 7, 4),

    (9, 'Checking', 1000.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 8, 5),

    (10, 'Investment', 5000.00, 'Active', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 9, 5);

**Services Entity:**

INSERT INTO Bank\_Services (Service\_ID, Service\_Type, Amount, Date\_of\_Service, Service\_Status, Employee\_ID, Details\_ID, FeedbackID)

SELECT

    1, 'Deposit', 100.00, TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 'Completed', 1, 1, NULL FROM DUAL

UNION ALL

SELECT

    2, 'Withdrawal', 50.00, TO\_DATE('2024-04-20', 'YYYY-MM-DD'), 'Pending', 2, 2, NULL FROM DUAL

UNION ALL

SELECT

    3, 'Account Opening', 200.00, TO\_DATE('2024-04-21', 'YYYY-MM-DD'), 'Completed', 3, 3, 1 FROM DUAL

UNION ALL

SELECT

    4, 'Loan Application', 300.00, TO\_DATE('2024-04-22', 'YYYY-MM-DD'), 'Pending', 4, 4, NULL FROM DUAL

UNION ALL

SELECT

    5, 'Balance Inquiry', 150.00, TO\_DATE('2024-04-23', 'YYYY-MM-DD'), 'Completed', 5, 5, 2 FROM DUAL

UNION ALL

SELECT

    6, 'Bill Payment', 75.00, TO\_DATE('2024-04-24', 'YYYY-MM-DD'), 'Pending', 6, 6, NULL FROM DUAL

UNION ALL

SELECT

    7, 'Transaction Dispute', 250.00, TO\_DATE('2024-04-25', 'YYYY-MM-DD'), 'Completed', 7, 7, 3 FROM DUAL

UNION ALL

SELECT

    8, 'Loan Payment', 200.00, TO\_DATE('2024-04-26', 'YYYY-MM-DD'), 'Completed', 8, 8, 4 FROM DUAL

UNION ALL

SELECT

    9, 'Cheque Encashment', 120.00, TO\_DATE('2024-04-27', 'YYYY-MM-DD'), 'Pending', 9, 9, NULL FROM DUAL

UNION ALL

SELECT

    10, 'Account Closing', 60.00, TO\_DATE('2024-04-28', 'YYYY-MM-DD'), 'Completed', 10, 10, 5 FROM DUAL;

**Details Entity:**

INSERT INTO Service\_Details (Details\_ID, Description, Date\_of\_Service, Location)

SELECT

    1, 'Deposit of $1000', TO\_DATE('2024-04-19', 'YYYY-MM-DD'), 'Main Branch' FROM DUAL

UNION ALL

SELECT

    2, 'Withdrawal of $500', TO\_DATE('2024-04-20', 'YYYY-MM-DD'), 'ATM' FROM DUAL

UNION ALL

SELECT

    3, 'Account Opening for John Doe', TO\_DATE('2024-04-21', 'YYYY-MM-DD'), 'Main Branch' FROM DUAL

UNION ALL

SELECT

    4, 'Loan Application for Car Loan', TO\_DATE('2024-04-22', 'YYYY-MM-DD'), 'Online' FROM DUAL

UNION ALL

SELECT

    5, 'Balance Inquiry', TO\_DATE('2024-04-23', 'YYYY-MM-DD'), 'Mobile App' FROM DUAL

UNION ALL

SELECT

    6, 'Utility Bill Payment', TO\_DATE('2024-04-24', 'YYYY-MM-DD'), 'Main Branch' FROM DUAL

UNION ALL

SELECT

    7, 'Transaction Dispute Resolution', TO\_DATE('2024-04-25', 'YYYY-MM-DD'), 'Call Center' FROM DUAL

UNION ALL

SELECT

    8, 'Loan Payment for Personal Loan', TO\_DATE('2024-04-26', 'YYYY-MM-DD'), 'Main Branch' FROM DUAL

UNION ALL

SELECT

    9, 'Cheque Encashment', TO\_DATE('2024-04-27', 'YYYY-MM-DD'), 'Main Branch' FROM DUAL

UNION ALL

SELECT

    10, 'Account Closing for Jane Smith', TO\_DATE('2024-04-28', 'YYYY-MM-DD'), 'Main Branch' FROM DUAL;

**Feedback Entity:**

INSERT INTO Feedback (FeedbackID, FeedbackText, Rating, FeedbackDate)

SELECT

    1, 'Great service, very helpful staff!', 5, TO\_DATE('2024-04-15', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    2, 'The website needs improvement.', 3, TO\_DATE('2024-04-16', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    3, 'Poor customer service experience.', 2, TO\_DATE('2024-04-17', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    4, 'Excellent service, highly recommended!', 5, TO\_DATE('2024-04-18', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    5, 'Average experience, room for improvement.', 3, TO\_DATE('2024-04-19', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    6, 'Very satisfied with the product quality.', 4, TO\_DATE('2024-04-20', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    7, 'Disappointed with the delivery time.', 2, TO\_DATE('2024-04-21', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    8, 'Prompt response from customer support.', 4, TO\_DATE('2024-04-22', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    9, 'Product was damaged upon arrival.', 1, TO\_DATE('2024-04-23', 'YYYY-MM-DD') FROM DUAL

UNION ALL

SELECT

    10, 'Fast and easy checkout process.', 5, TO\_DATE('2024-04-24', 'YYYY-MM-DD') FROM DUAL;

**Queries:**

* **Get the List of customers from the bank as we have two tables to populate the data from it.**

SELECT

C.CustomerID,

C.FirstName,

C.LastName,

C.BirthYear,

C.IdentificationNumber,

C.PhoneNumber,

C.AptNumber,

C.Street,

Z.City,

Z.State,

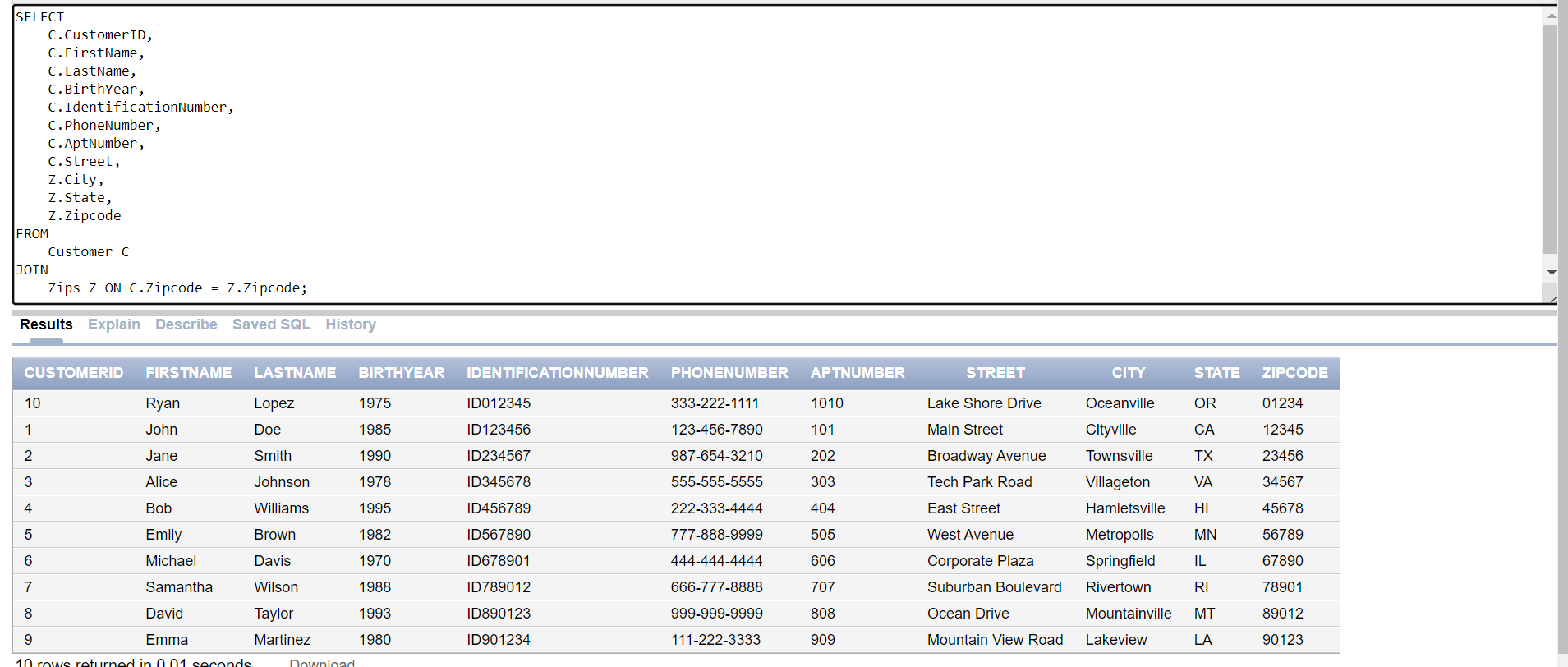
Z.Zipcode

FROM

Customer C

JOIN

Zips Z ON C.Zipcode = Z.Zipcode;

****

* **Get the Customer details who are with branch Type of “retail”**

SELECT

C.CustomerID,

C.FirstName,

C.LastName,

C.PhoneNumber,

B.BranchName,

Z.City,

Z.State

FROM

Customer C

JOIN

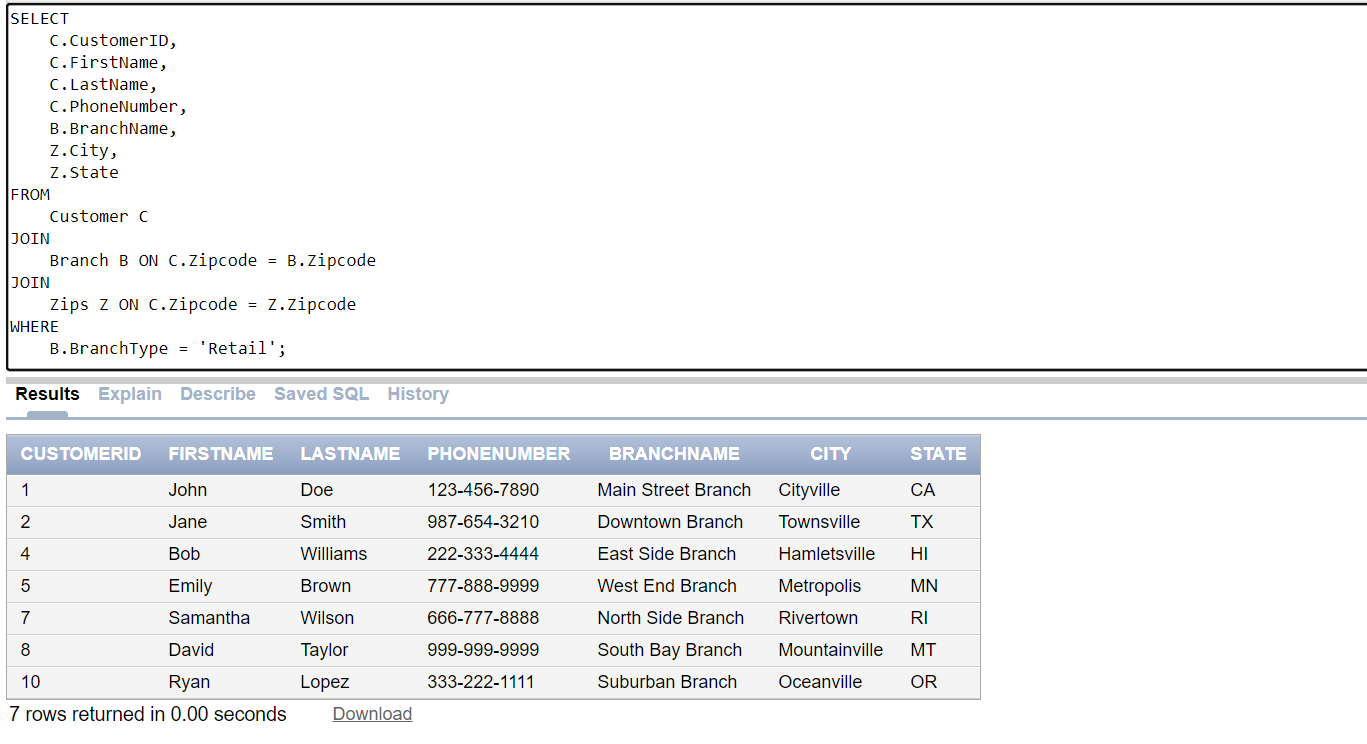
Branch B ON C.Zipcode = B.Zipcode

JOIN

Zips Z ON C.Zipcode = Z.Zipcode

WHERE

B.BranchType = 'Retail';



* **Find The total balance of all active accounts in each branch**

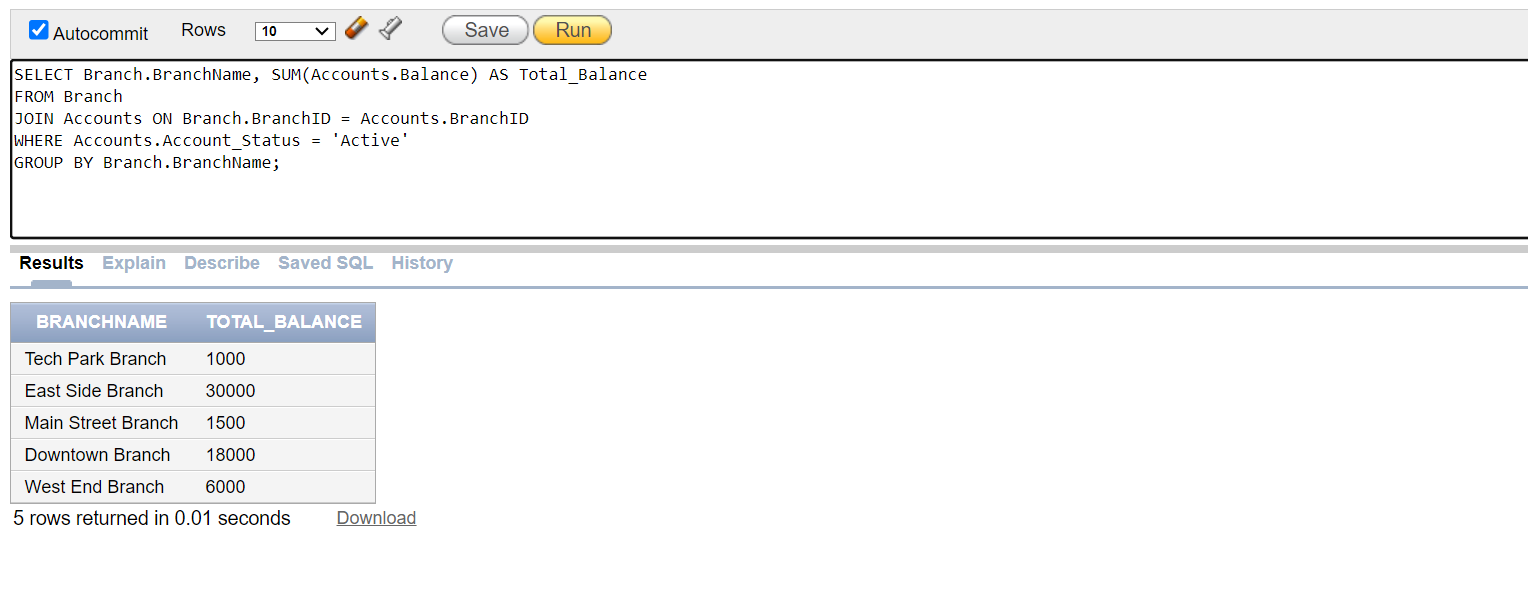
SELECT Branch.BranchName, SUM(Accounts.Balance) AS Total\_Balance

FROM Branch

JOIN Accounts ON Branch.BranchID = Accounts.BranchID

WHERE Accounts.Account\_Status = 'Active'

GROUP BY Branch.BranchName;



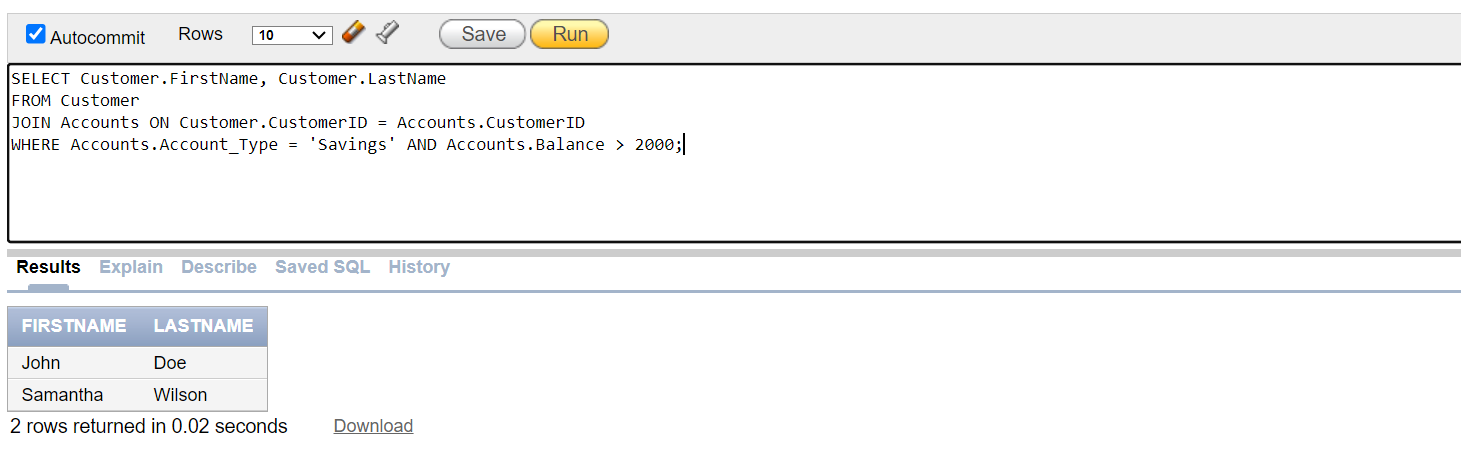
* **List of all customers who have a savings account with a balance greater than $2000**

SELECT Customer.FirstName, Customer.LastName

FROM Customer

JOIN Accounts ON Customer.CustomerID = Accounts.CustomerID

WHERE Accounts.Account\_Type = 'Savings' AND Accounts.Balance > 2000;



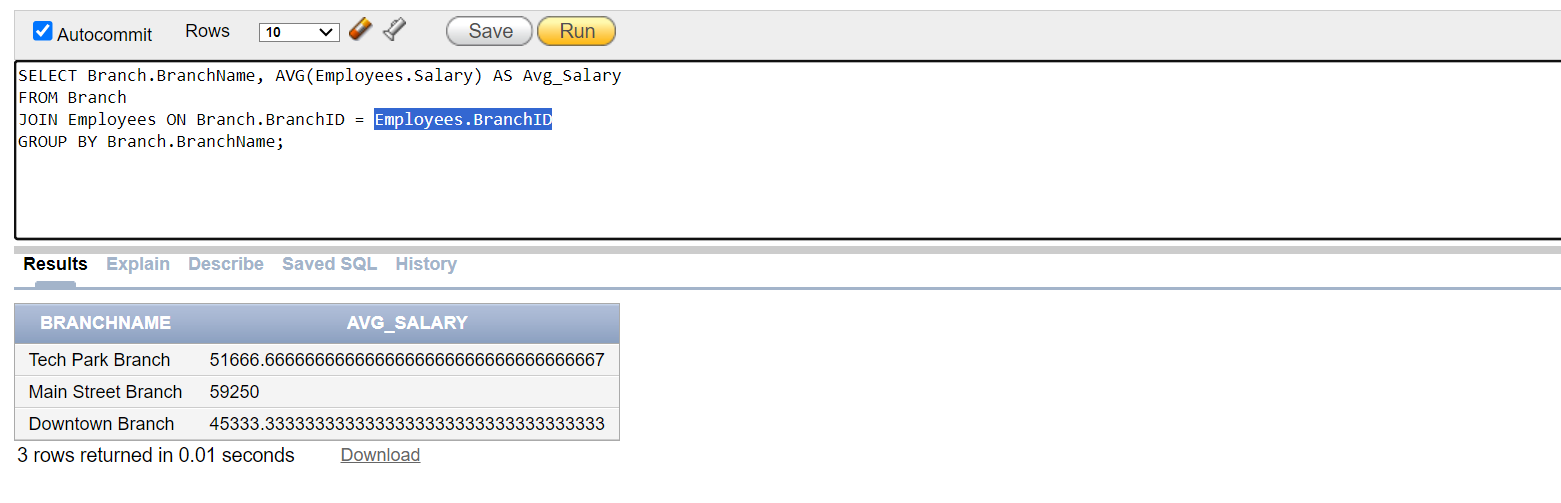
* **Find the average salary of employees in each branch**

SELECT Branch.BranchName, AVG(Employees.Salary) AS Avg\_Salary

FROM Branch

JOIN Employees ON Branch.BranchID = Employees.Branch\_ID

GROUP BY Branch.BranchName;



* **Retrieve details of all services provided by Sales Associates (Designation = ‘Sales Associate’)**

SELECT Bank\_Services.Service\_Type, Bank\_Services.Amount, Employees.First\_Name, Employees.Last\_Name

FROM Bank\_Services

JOIN Employees ON Bank\_Services.Employee\_ID = Employees.Employee\_ID

WHERE Employees.Designation = 'Sales Associate';

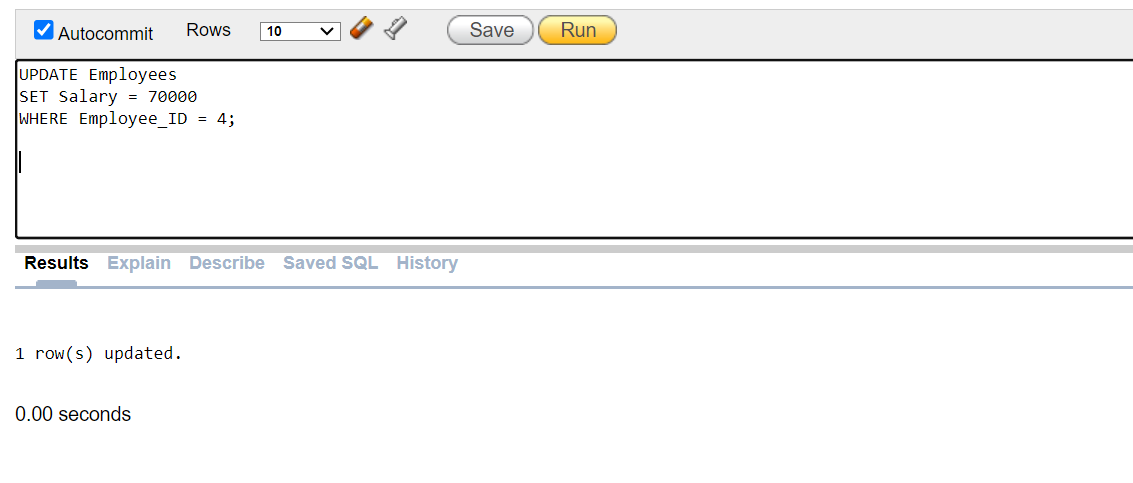


* **Update the salary of the IT Specialist(Employee\_ID = 4) to $70000**

UPDATE Employees

SET Salary = 70000

WHERE Employee\_ID = 4;



* **Delete the record of a customer who has an inactive account**

DELETE FROM Customer

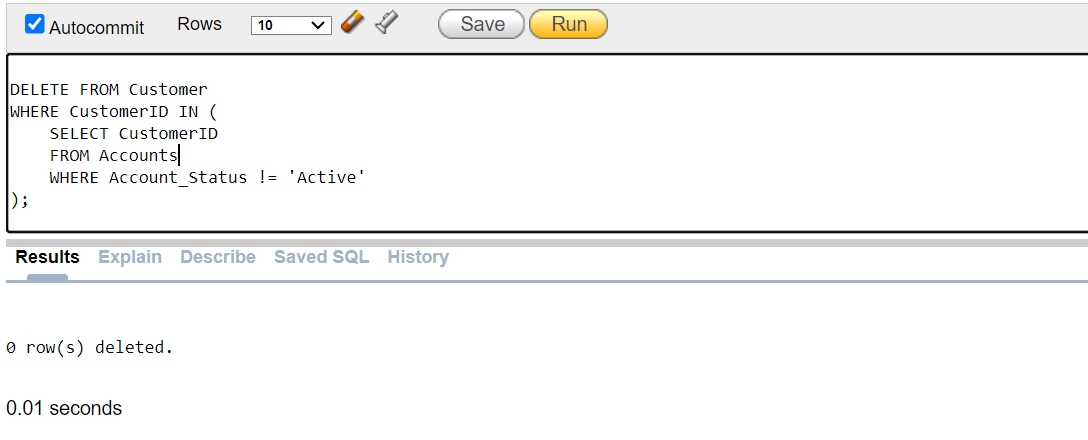
WHERE CustomerID IN (

SELECT CustomerID

FROM Accounts

WHERE Account\_Status != 'Active'

);



* **Find the total number of services completed by each employee**

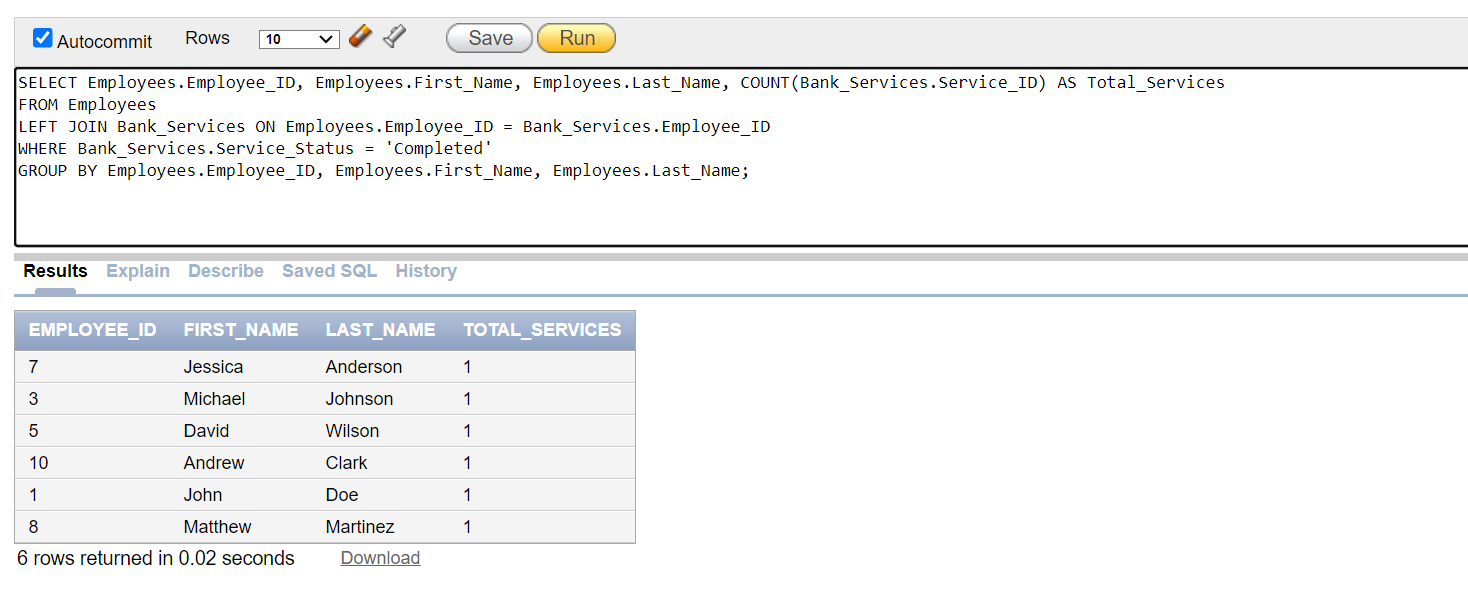
SELECT Employees.Employee\_ID, Employees.First\_Name, Employees.Last\_Name, COUNT(Bank\_Services.Service\_ID) AS Total\_Services

FROM Employees

LEFT JOIN Bank\_Services ON Employees.Employee\_ID = Bank\_Services.Employee\_ID

WHERE Bank\_Services.Service\_Status = 'Completed'

GROUP BY Employees.Employee\_ID, Employees.First\_Name, Employees.Last\_Name;



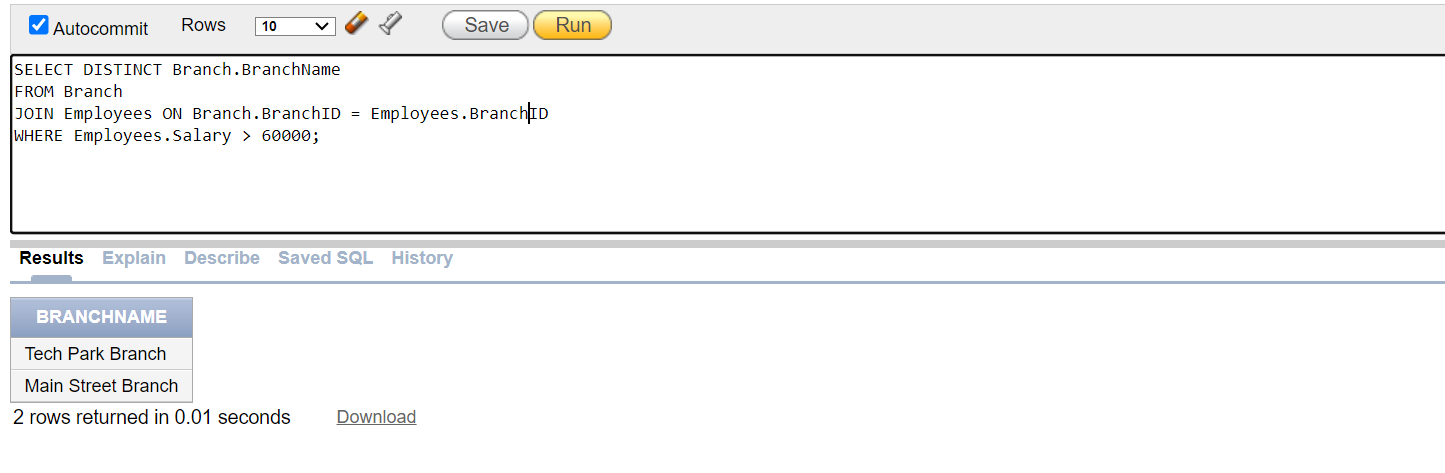
* **List of all branches with at least one employee earning more than $60000.**

SELECT DISTINCT Branch.BranchName

FROM Branch

JOIN Employees ON Branch.BranchID = Employees.Branch\_ID

WHERE Employees.Salary > 60000;



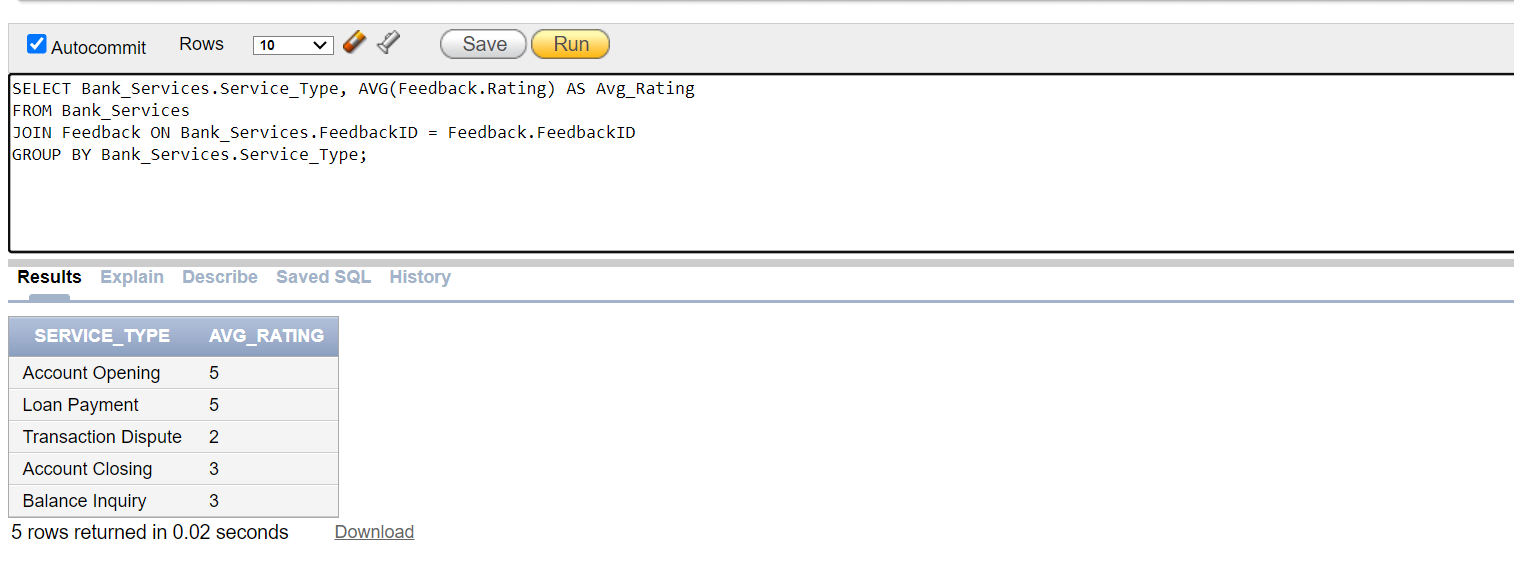
* **Find the average rating of feedback for each service type:**

SELECT Bank\_Services.Service\_Type, AVG(Feedback.Rating) AS Avg\_Rating

FROM Bank\_Services

JOIN Feedback ON Bank\_Services.FeedbackID = Feedback.FeedbackID

GROUP BY Bank\_Services.Service\_Type;



* **Find the branch with the highest number of active accounts**

SELECT BranchName, Active\_Accounts

FROM (

SELECT Branch.BranchName, COUNT(Accounts.Account\_Number) AS Active\_Accounts

FROM Branch

JOIN Accounts ON Branch.BranchID = Accounts.BranchID

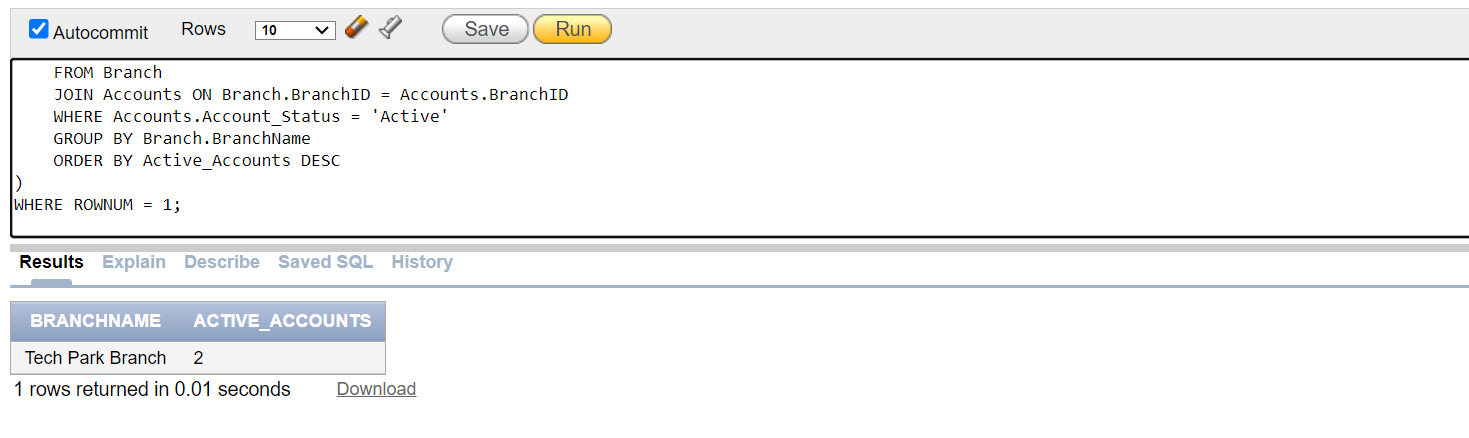
WHERE Accounts.Account\_Status = 'Active'

GROUP BY Branch.BranchName

ORDER BY Active\_Accounts DESC

)

WHERE ROWNUM = 1;



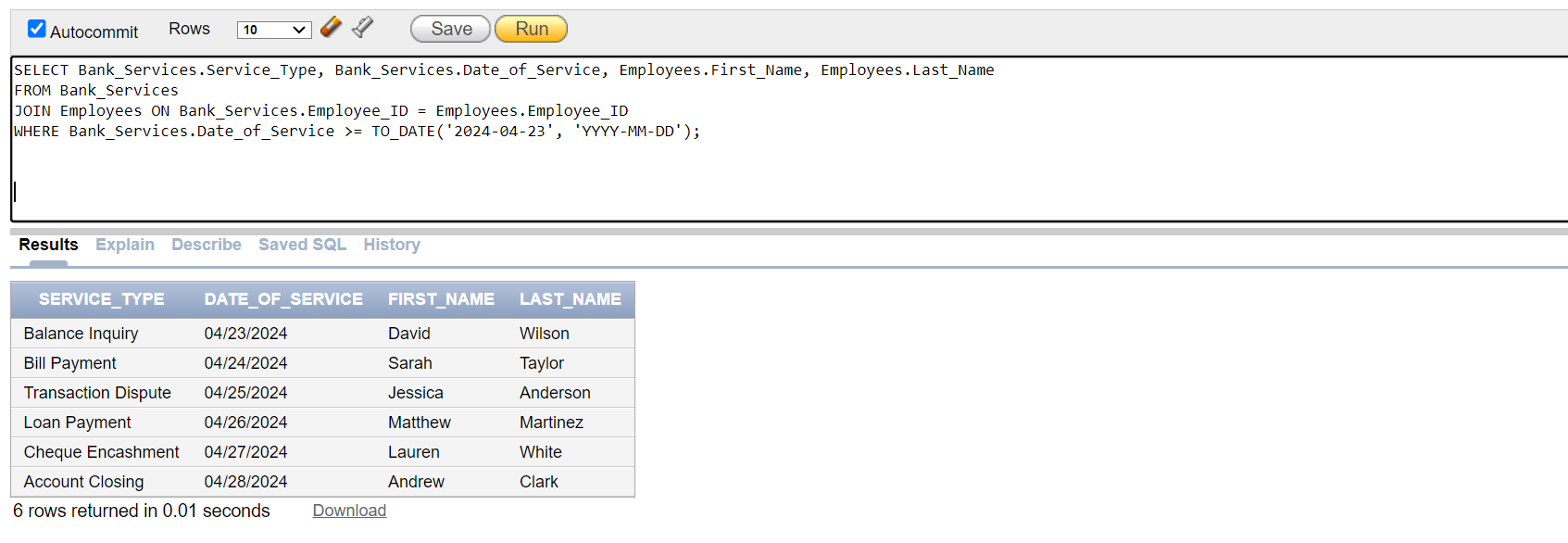
* **List all services provided on or after April 23, 2024 along with the employee details**

SELECT Bank\_Services.Service\_Type, Bank\_Services.Date\_of\_Service, Employees.First\_Name, Employees.Last\_Name

FROM Bank\_Services

JOIN Employees ON Bank\_Services.Employee\_ID = Employees.Employee\_ID

WHERE Bank\_Services.Date\_of\_Service >= TO\_DATE('2024-04-23', 'YYYY-MM-DD');



* **Find the total amount of transactions completed by each branch on April 24, 2024.**

SELECT Branch.BranchName, SUM(Bank\_Services.Amount) AS Total\_Amount

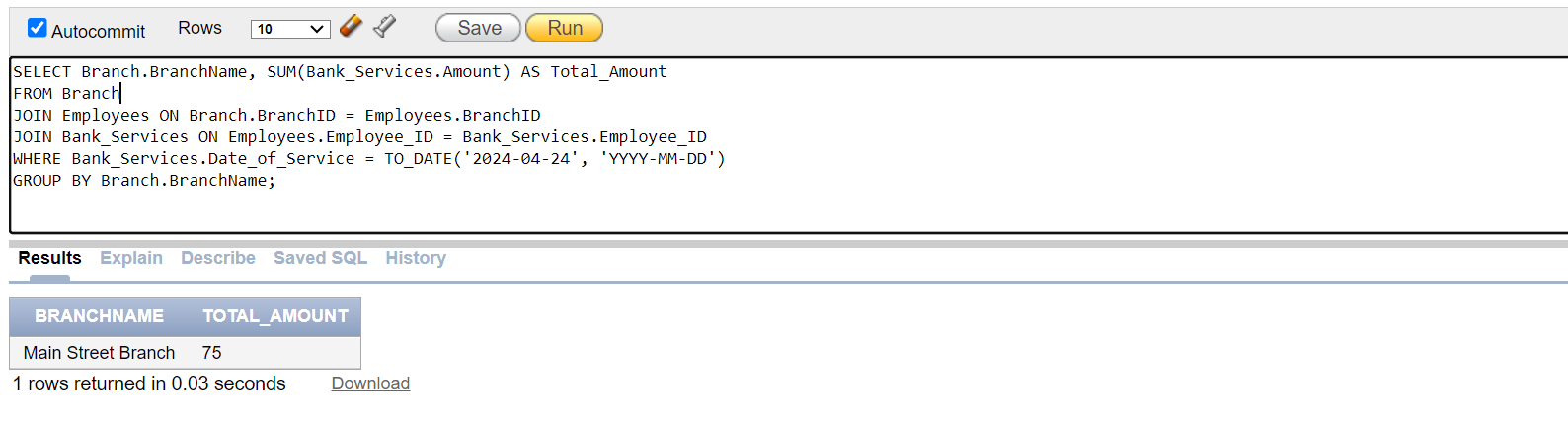
FROM Branch

JOIN Employees ON Branch.BranchID = Employees.Branch\_ID

JOIN Bank\_Services ON Employees.Employee\_ID = Bank\_Services.Employee\_ID

WHERE Bank\_Services.Date\_of\_Service = TO\_DATE('2024-04-24', 'YYYY-MM-DD')

GROUP BY Branch.BranchName;



* **Get the detailed transaction statement information including account and customer details branch information and transaction types for customer ‘John Doe’**

SELECT

Customer.FirstName || ' ' || Customer.LastName AS Account\_Holder\_Name,

Accounts.Account\_Number,

Accounts.Account\_Type,

Accounts.Balance AS Current\_Balance,

Branch.BranchName AS Branch\_Name,

Bank\_Services.Service\_Type,

Bank\_Services.Amount AS Transaction\_Amount,

Bank\_Services.Date\_of\_Service AS Transaction\_Date,

Service\_Details.Description AS Transaction\_Description

FROM

Customer,

Accounts,

Branch,

Employees,

Bank\_Services,

Service\_Details

WHERE

Customer.CustomerID = Accounts.CustomerID

AND Accounts.BranchID = Branch.BranchID

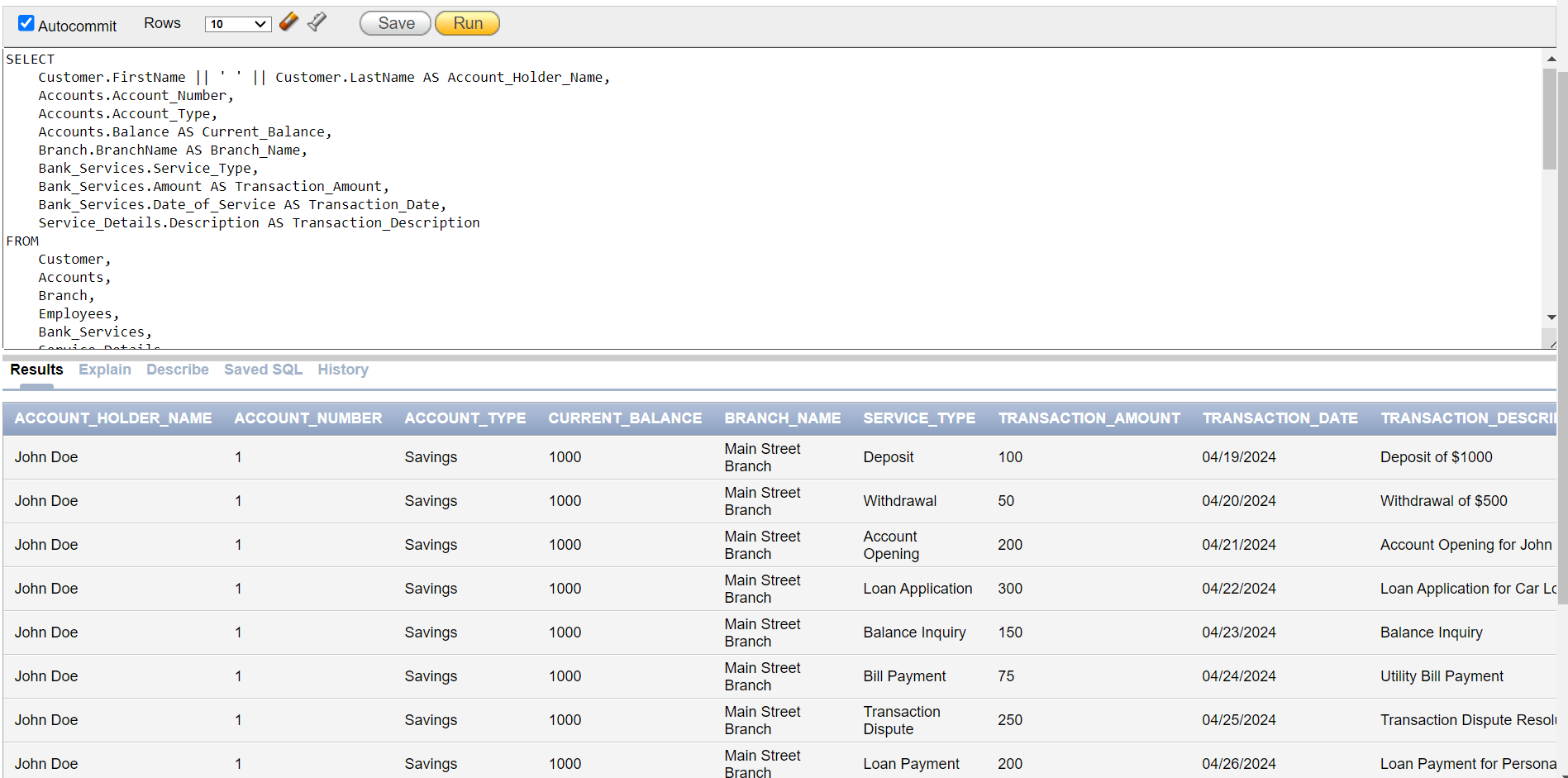
AND Accounts.BranchID = Employees.BranchID

AND Employees.Designation = 'Manager'

AND Bank\_Services.Details\_ID(+) = Service\_Details.Details\_ID

AND Customer.FirstName = 'John'

AND Customer.LastName = 'Doe';



**OBSERVATION:**

**Project Progression and Learning:**

The journey of creating our bank management system was nothing short of an adventure in the world of databases. From the early stages of brainstorming and conceptualizing to the late nights of coding and debugging, each phase brought its own set of challenges and learning opportunities.

We started with a broad vision of what the system should achieve, gradually narrowing down our focus as we delved deeper into the complexities of database design. The iterative process of refining our entity-relationship diagrams, normalizing the database, and implementing the physical schema was both demanding and enlightening. It taught us the importance of attention to detail and the value of collaboration, as we constantly sought feedback from each other to improve our design.

**Unexpected Events:**

Just when we thought we had everything figured out, we stumbled upon the maze of transitive dependencies. It was like hitting a roadblock in our journey, forcing us to backtrack and rethink our entire approach. While it was frustrating at first, this unexpected challenge turned out to be a blessing in disguise. It pushed us to revisit our initial assumptions, leading to a more robust and efficient database design.

**Beyond the Course:**

While the course provided us with a solid theoretical foundation, the project exposed us to the practical realities of database management. We learned about the importance of data integrity, the complexities involved in designing a schema that can evolve over time, and the art of writing efficient SQL queries. These were lessons that went beyond the scope of our coursework, offering us a glimpse into the real-world challenges and opportunities in the field of database management.

**Database Product Experience:**

Using Oracle SQL for our project was a game-changer. Its powerful features and intuitive interface made the process of creating and managing our database a breeze. From defining data types and constraints to executing complex queries and transactions, Oracle SQL proved to be a reliable ally throughout our project. Its scalability and robustness allowed us to focus on the design and implementation of our database without getting bogged down by technical limitations.

**Conclusion:**

In retrospect, our bank management system project was more than just an academic exercise; it was a voyage of discovery and growth. We learned valuable lessons about teamwork, problem-solving, and the intricacies of database design. We discovered that sometimes the most challenging problems lead to the most rewarding solutions, and that flexibility and adaptability are key to success in any project.

Using Oracle SQL was a delightful experience that enhanced our productivity and creativity, empowering us to bring our vision to life. It reinforced our belief in the power of technology to transform ideas into reality and left us eager to explore further the endless possibilities that the world of databases has to offer.

**STUDENT LOG:**

