```
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;
```

**-- Create the employees table**

```
CREATE TABLE employees (
   employee_id INT AUTO_INCREMENT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   hire_date DATE NOT NULL,
   salary DECIMAL(10, 2) NOT NULL
);
```

**-- Insert sample data into the employees table**

```
INSERT INTO employees (first_name, last_name, email, hire_date, salary) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00),
('Charlie', 'Davis', 'charlie.davis@example.com', '2023-05-10', 52000.00);
```

**-- Create an index on the last_name column**

```
CREATE INDEX idx_last_name ON employees (last_name);
```

**-- Create a view to show employees with a salary greater than 50,000**

```
CREATE VIEW employee_salaries AS
SELECT employee_id, first_name, last_name, salary
FROM employees
WHERE salary > 50000;
```

**-- Create a sequence-like behavior using an auto-incrementing column**

```
CREATE TABLE sequence_example (
   seq_id INT AUTO_INCREMENT PRIMARY KEY
);
```

**-- Insert a dummy row to get the next sequence value**

```
INSERT INTO sequence_example () VALUES ();
SELECT LAST_INSERT_ID() AS next_val;
```

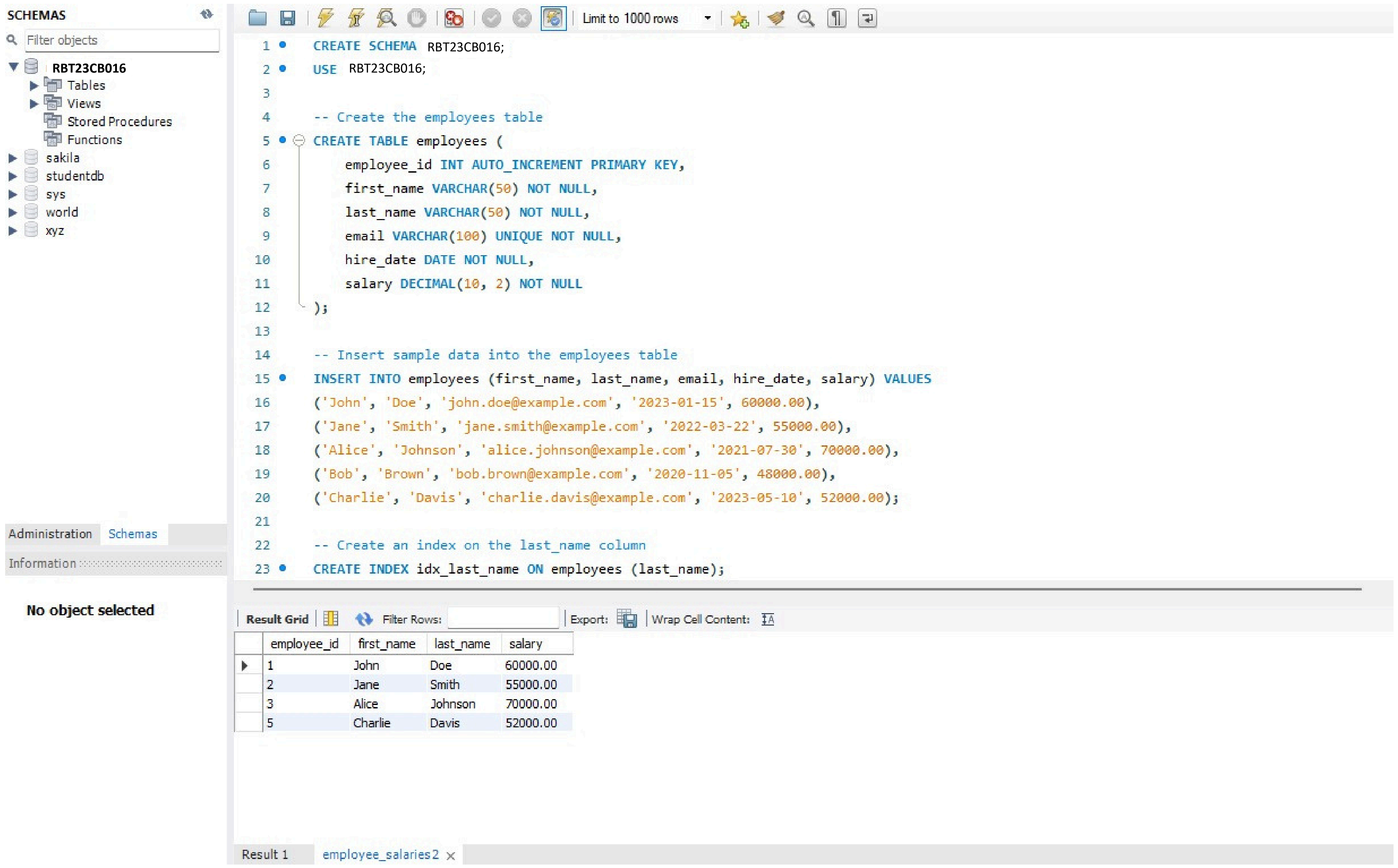
**-- Create a view that acts as a synonym for the employees table**

```
CREATE VIEW emp AS
SELECT * FROM employees;
```

**-- Select from the employee_salaries view to see the results**

```
SELECT * FROM employee_salaries;
```

1

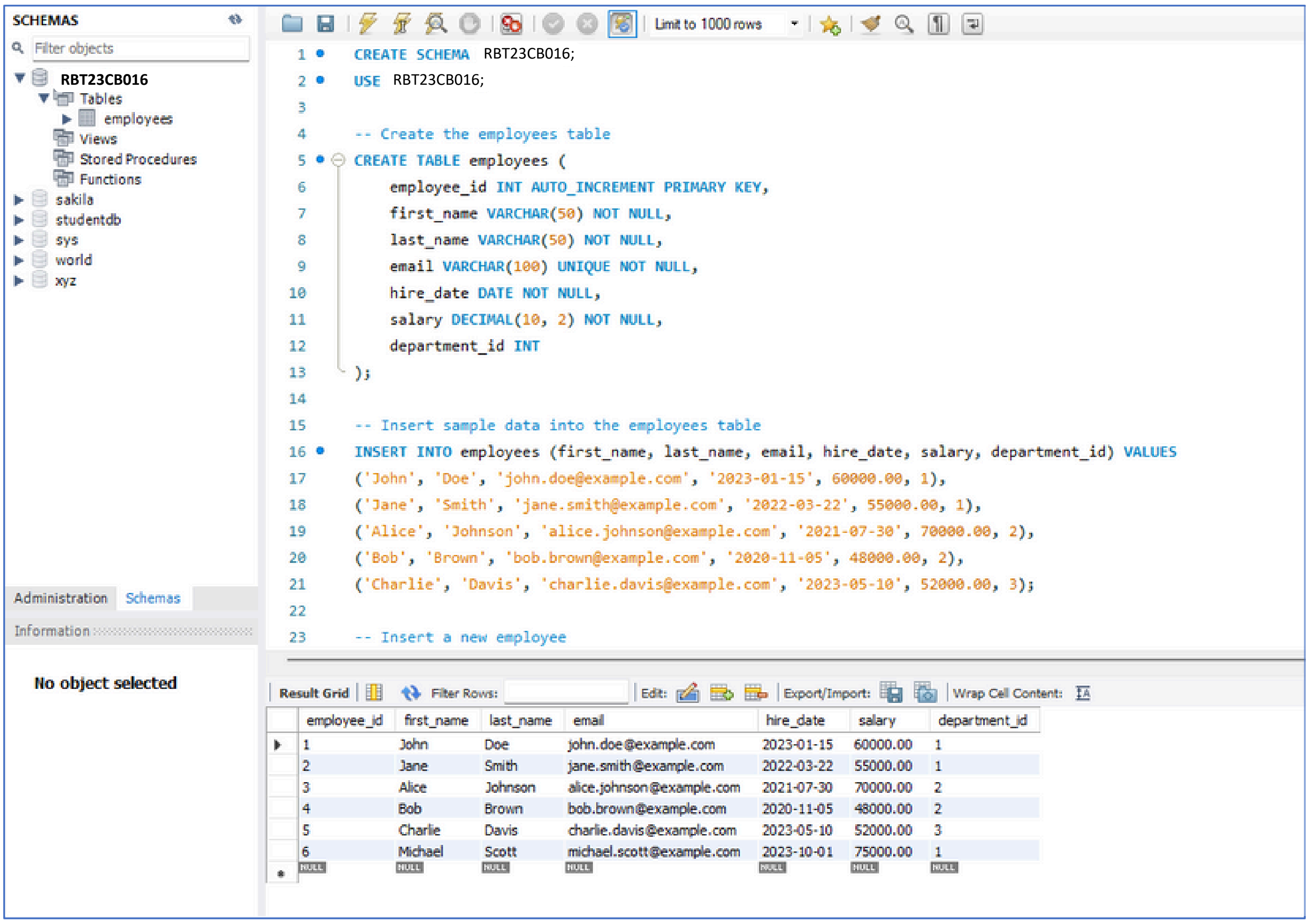


```sql
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;

-- Create the employees table
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL
);

-- Insert sample data into the employees table
INSERT INTO employees (first_name, last_name, email, hire_date, salary) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00),
('Charlie', 'Davis', 'charlie.davis@example.com', '2023-05-10', 52000.00);

-- Create an index on the last_name column
CREATE INDEX idx_last_name ON employees (last_name);
```

| employee_id | first_name | last_name | salary |
|---|---|---|---|
| 1 | John | Doe | 60000.00 |
| 2 | Jane | Smith | 55000.00 |
| 3 | Alice | Johnson | 70000.00 |
| 5 | Charlie | Davis | 52000.00 |

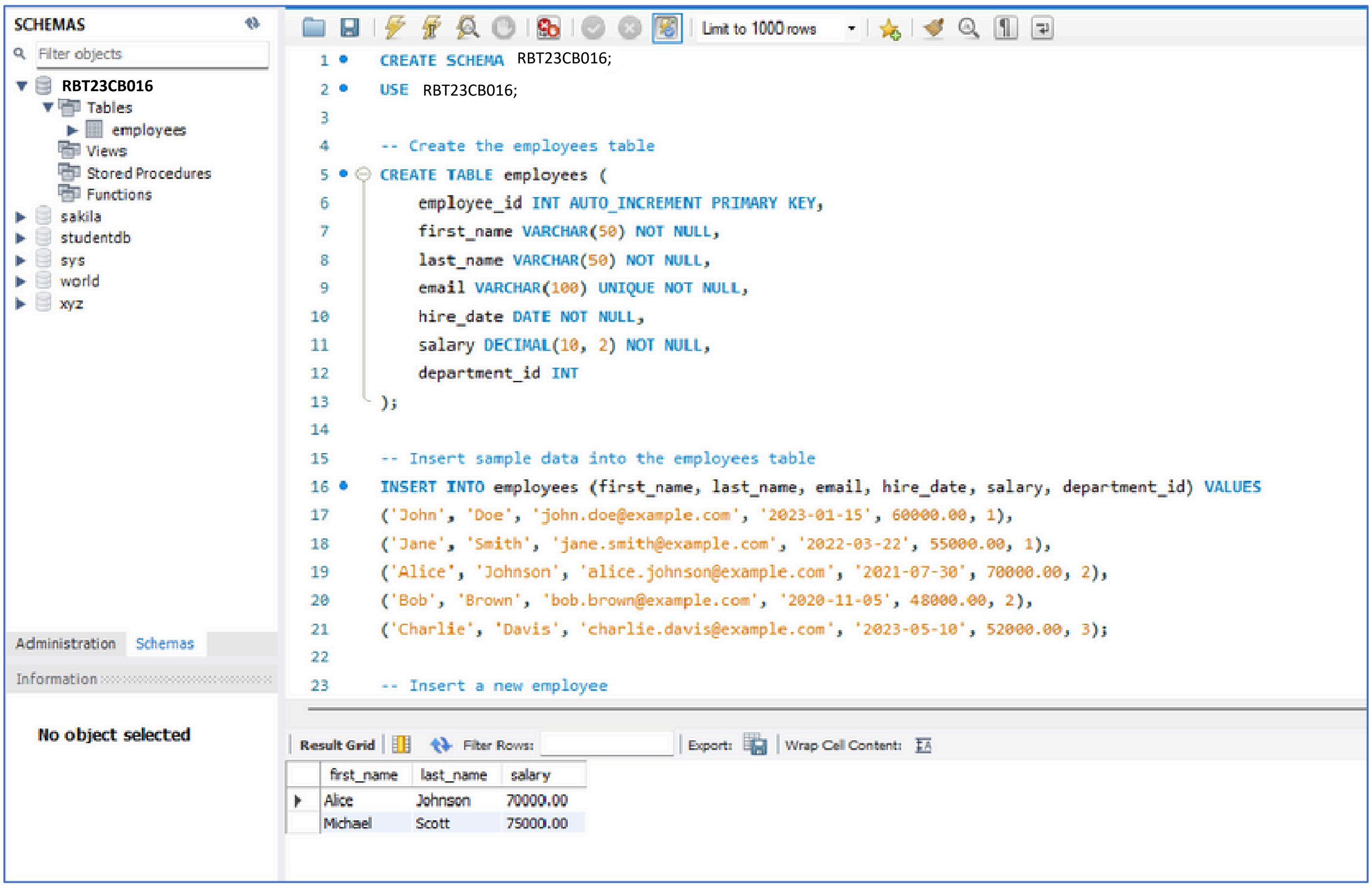
```
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;

-- Create the employees table
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT
);

-- Insert sample data into the employees table
INSERT INTO employees (first_name, last_name, email, hire_date, salary, department_id) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00, 1),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00, 1),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00, 2),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00, 2),
('Charlie', 'Davis', 'charlie.davis@example.com', '2023-05-10', 52000.00, 3);

-- Insert a new employee
INSERT INTO employees (first_name, last_name, email, hire_date, salary, department_id)
VALUES ('Michael', 'Scott', 'michael.scott@example.com', '2023-10-01', 75000.00, 1);

-- Select all employees
SELECT * FROM employees;

-- Select employees with salary greater than 60,000
SELECT first_name, last_name, salary
FROM employees
WHERE salary > 60000;

-- Update employee salary for those in department 1
UPDATE employees
SET salary = salary * 1.10
WHERE department_id = 1;

-- Delete an employee by email
DELETE FROM employees
WHERE email = 'john.doe@example.com';

-- Select employees hired in the last year
SELECT first_name, last_name, hire_date
FROM employees
WHERE hire_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR);
```

```
-- Select distinct departments
SELECT DISTINCT department_id
FROM employees;

-- Select employees with salary in a specific range
SELECT first_name, last_name, salary
FROM employees
WHERE salary BETWEEN 50000 AND 70000;

-- Use a set operator to find employees in two different departments
SELECT first_name, last_name
FROM employees
WHERE department_id = 1
UNION
SELECT first_name, last_name
FROM employees
WHERE department_id = 2;

-- Count employees in each department
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id;
```



```sql
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;

-- Create the employees table
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT
);

-- Insert sample data into the employees table
INSERT INTO employees (first_name, last_name, email, hire_date, salary, department_id) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00, 1),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00, 1),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00, 2),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00, 2),
('Charlie', 'Davis', 'charlie.davis@example.com', '2023-05-10', 52000.00, 3);

-- Insert a new employee
```

| employee_id | first_name | last_name | email | hire_date | salary | department_id |
|---|---|---|---|---|---|---|
| 1 | John | Doe | john.doe@example.com | 2023-01-15 | 60000.00 | 1 |
| 2 | Jane | Smith | jane.smith@example.com | 2022-03-22 | 55000.00 | 1 |
| 3 | Alice | Johnson | alice.johnson@example.com | 2021-07-30 | 70000.00 | 2 |
| 4 | Bob | Brown | bob.brown@example.com | 2020-11-05 | 48000.00 | 2 |
| 5 | Charlie | Davis | charlie.davis@example.com | 2023-05-10 | 52000.00 | 3 |
| 6 | Michael | Scott | michael.scott@example.com | 2023-10-01 | 75000.00 | 1 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |



```sql
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;

-- Create the employees table
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT
);

-- Insert sample data into the employees table
INSERT INTO employees (first_name, last_name, email, hire_date, salary, department_id) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00, 1),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00, 1),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00, 2),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00, 2),
('Charlie', 'Davis', 'charlie.davis@example.com', '2023-05-10', 52000.00, 3);

-- Insert a new employee
```

| first_name | last_name | salary |
|---|---|---|
| Alice | Johnson | 70000.00 |
| Michael | Scott | 75000.00 |

```
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;
```

**-- Create the departments table**

```
CREATE TABLE departments (
   department_id INT AUTO_INCREMENT PRIMARY KEY,
   department_name VARCHAR(50) NOT NULL
);
```

**-- Create the employees table**

```
CREATE TABLE employees (
   employee_id INT AUTO_INCREMENT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   hire_date DATE NOT NULL,
   salary DECIMAL(10, 2) NOT NULL,
   department_id INT,
   FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

**-- Insert sample data into the departments table**

```
INSERT INTO departments (department_name) VALUES
('Sales'),
('Marketing'),
('HR'),
('IT');
```

**-- Insert sample data into the employees table**

```
INSERT INTO employees (first_name, last_name, email, hire_date, salary, department_id) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00, 1),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00, 1),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00, 2),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00, 2),
('Charlie', 'Davis', 'charlie.davis@example.com', '2023-05-10', 52000.00, 3),
('Michael', 'Scott', 'michael.scott@example.com', '2023-10-01', 75000.00, 1);
```

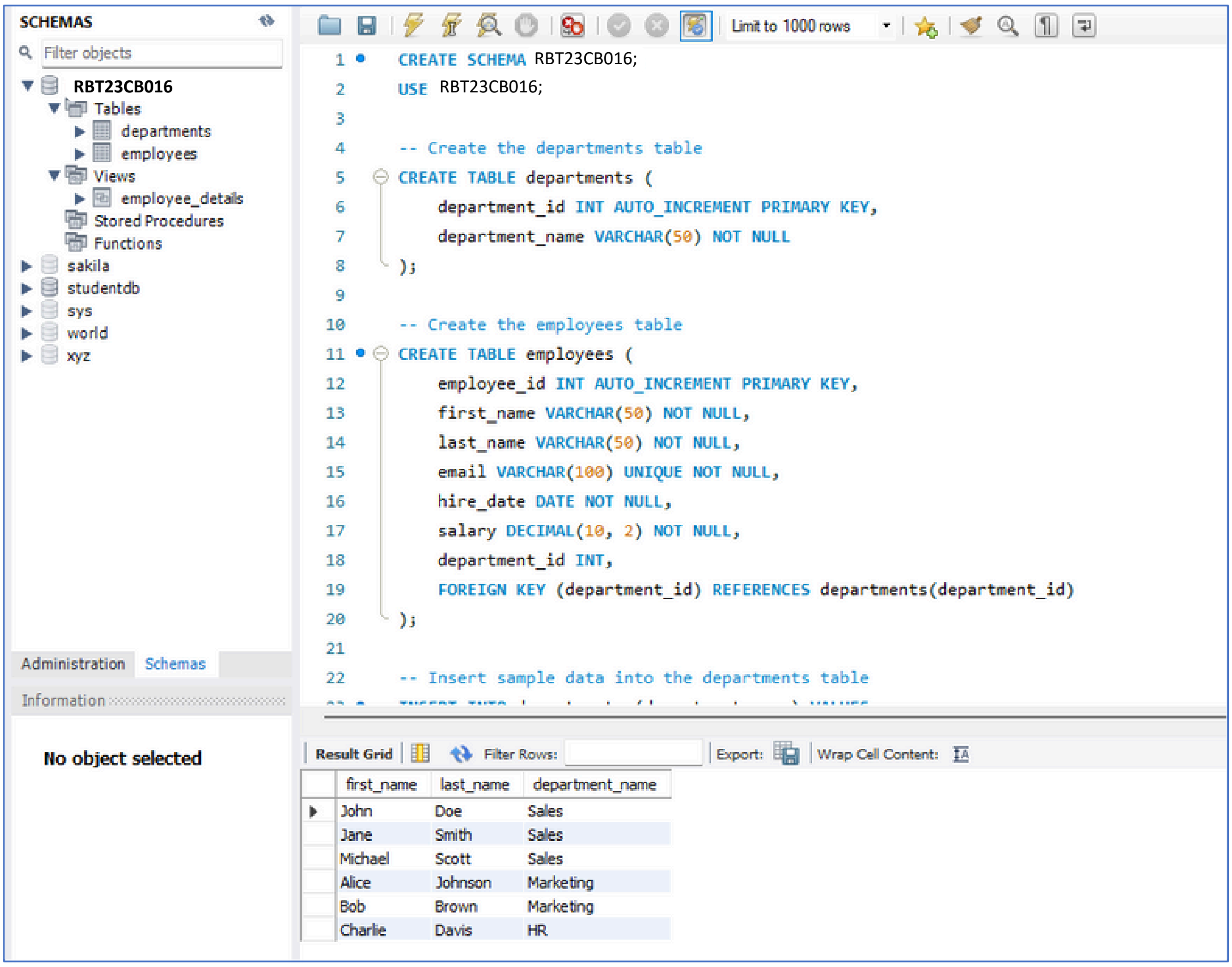
**-- 1. Inner Join: Select employees with their department names**

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
INNER JOIN departments d ON e.department_id = d.department_id;
```

**-- 2. Left Join: Select all employees and their department names (including those without a department)**

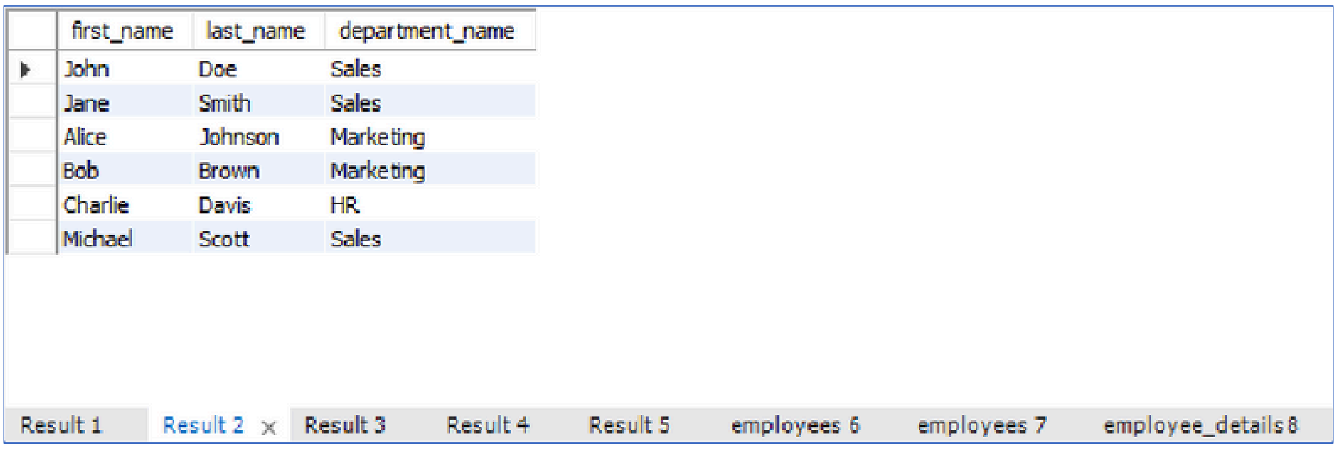
```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

```
-- 3. Right Join: Select all departments and their employees (including departments without employees)
SELECT d.department_name, e.first_name, e.last_name
FROM departments d
RIGHT JOIN employees e ON d.department_id = e.department_id;

-- 4. Full Outer Join: MySQL does not support FULL OUTER JOIN directly, but we can simulate it using UNION
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id
UNION
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.department_id;

-- 5. Cross Join: Select all combinations of employees and departments
SELECT e.first_name, d.department_name
FROM employees e
CROSS JOIN departments d;

-- 6. Subquery: Select employees with a salary greater than the average salary
SELECT first_name, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);

-- 7. Subquery with IN: Select employees in departments with a specific name
SELECT first_name, last_name
FROM employees
WHERE department_id IN (SELECT department_id FROM departments WHERE department_name = 'Sales');

-- 8. Create a View: Create a view to show employee details with department names
CREATE VIEW employee_details AS
SELECT e.first_name, e.last_name, e.salary, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;

-- 9. Select from the View: Retrieve data from the created view
SELECT * FROM employee_details;

-- 10. Update Employee Salary: Increase salary for employees in the IT department
UPDATE employees
SET salary = salary * 1.10
WHERE department_id = (SELECT department_id FROM departments WHERE department_name = 'IT');
```

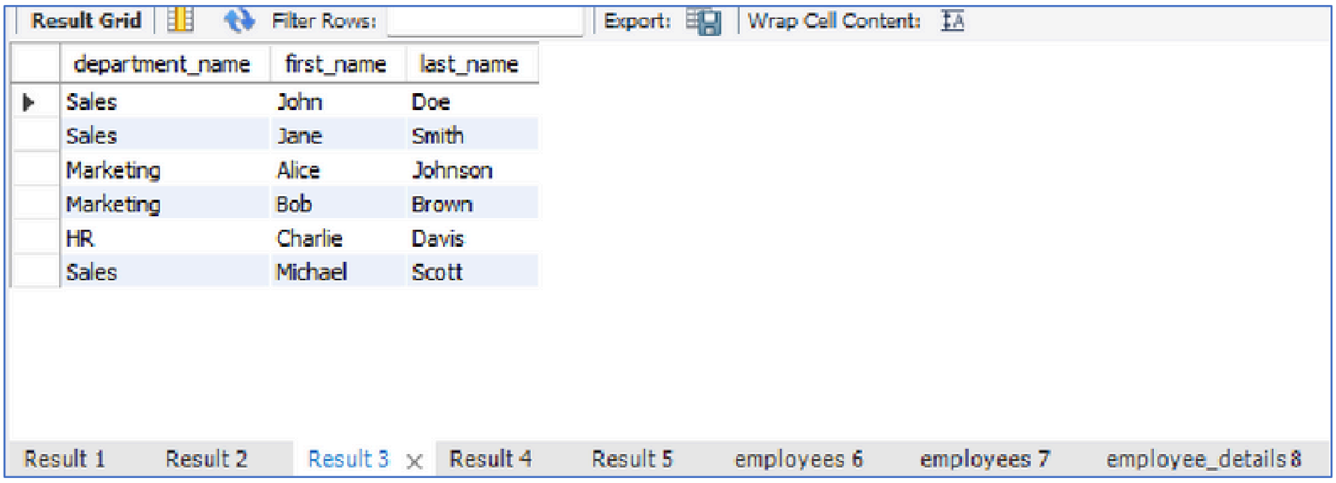


```sql
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;

-- Create the departments table
CREATE TABLE departments (
    department_id INT AUTO_INCREMENT PRIMARY KEY,
    department_name VARCHAR(50) NOT NULL
);

-- Create the employees table
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);

-- Insert sample data into the departments table
```

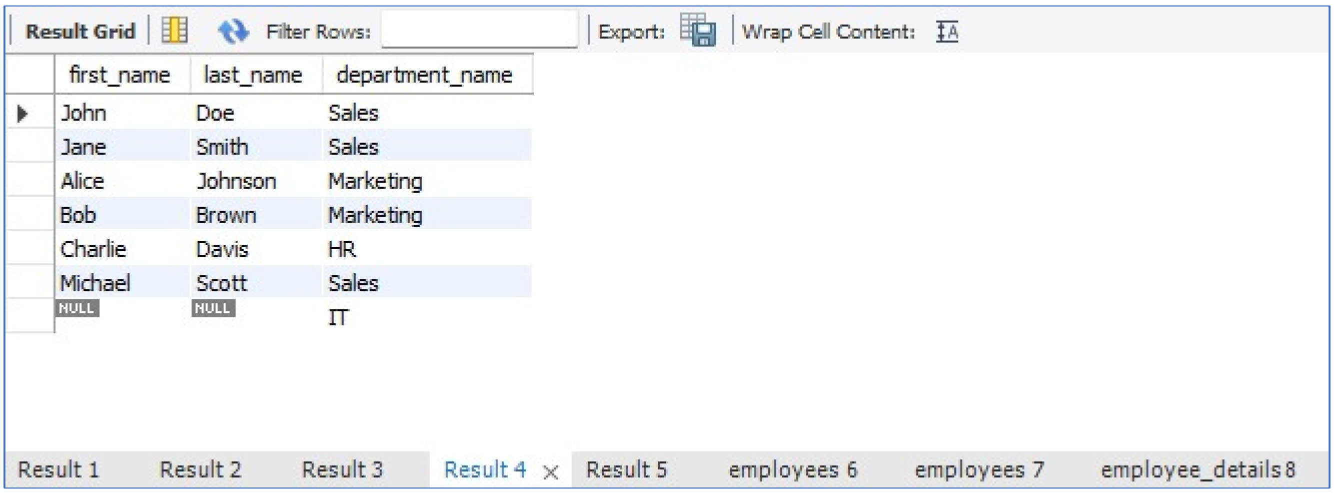
| first_name | last_name | department_name |
|---|---|---|
| John | Doe | Sales |
| Jane | Smith | Sales |
| Michael | Scott | Sales |
| Alice | Johnson | Marketing |
| Bob | Brown | Marketing |
| Charlie | Davis | HR |



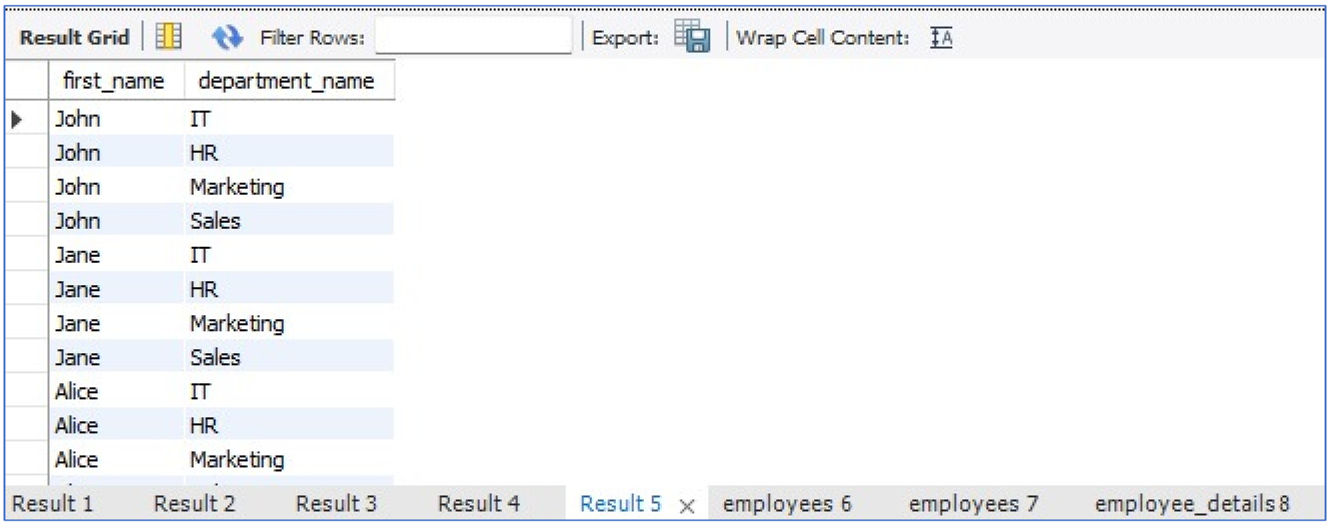
| first_name | last_name | department_name |
|---|---|---|
| John | Doe | Sales |
| Jane | Smith | Sales |
| Alice | Johnson | Marketing |
| Bob | Brown | Marketing |
| Charlie | Davis | HR |
| Michael | Scott | Sales |



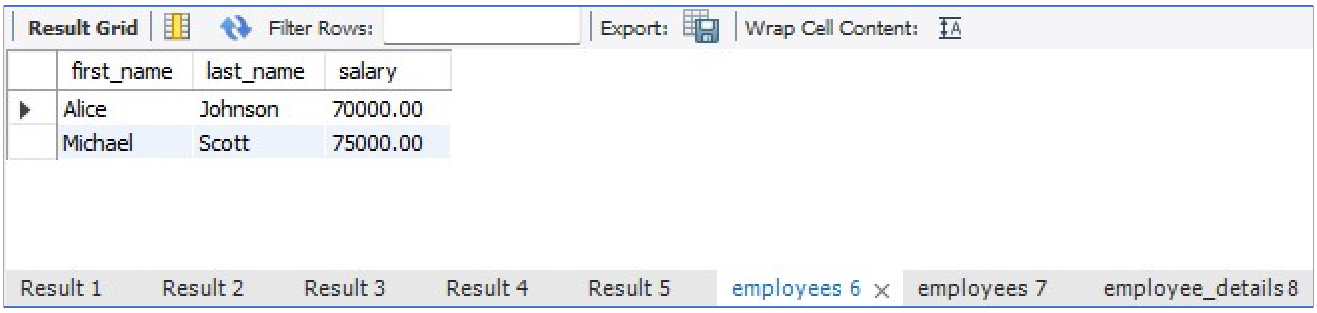
| department_name | first_name | last_name |
|---|---|---|
| Sales | John | Doe |
| Sales | Jane | Smith |
| Marketing | Alice | Johnson |
| Marketing | Bob | Brown |
| HR | Charlie | Davis |
| Sales | Michael | Scott |



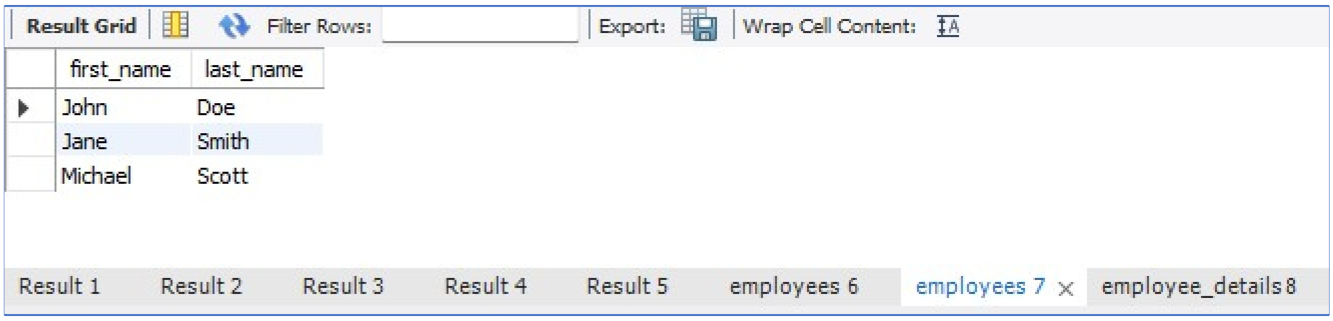
| first_name | last_name | department_name |
|---|---|---|
| John | Doe | Sales |
| Jane | Smith | Sales |
| Alice | Johnson | Marketing |
| Bob | Brown | Marketing |
| Charlie | Davis | HR |
| Michael | Scott | Sales |
| NULL | NULL | IT |



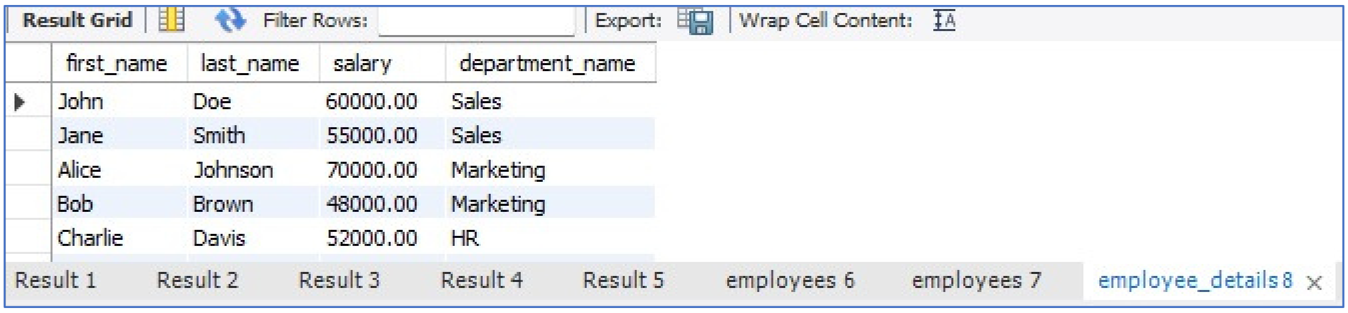
| first_name | department_name |
|---|---|
| John | IT |
| John | HR |
| John | Marketing |
| John | Sales |
| Jane | IT |
| Jane | HR |
| Jane | Marketing |
| Jane | Sales |
| Alice | IT |
| Alice | HR |
| Alice | Marketing |



| first_name | last_name | salary |
|---|---|---|
| Alice | Johnson | 70000.00 |
| Michael | Scott | 75000.00 |

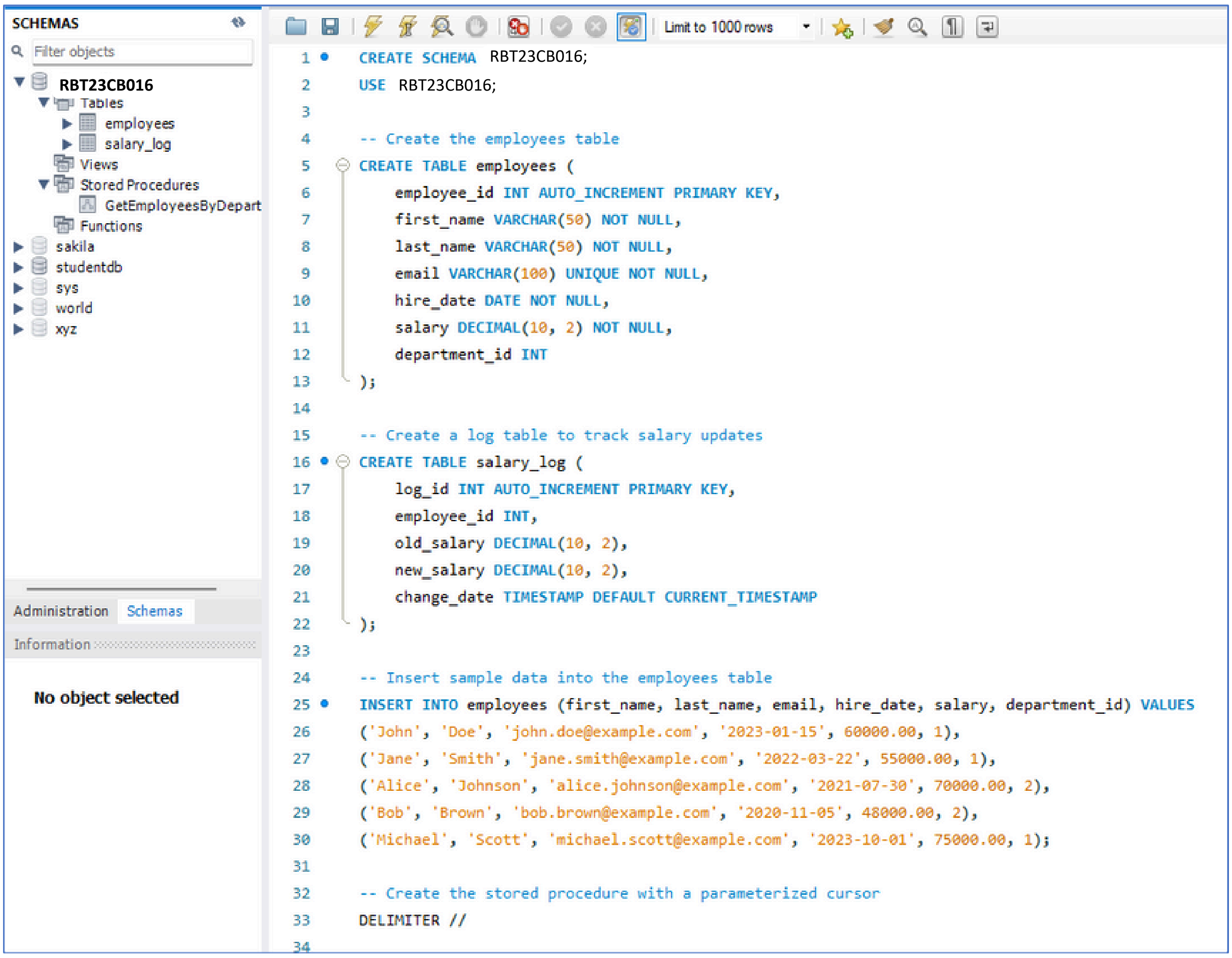


| first_name | last_name |
|---|---|
| John | Doe |
| Jane | Smith |
| Michael | Scott |

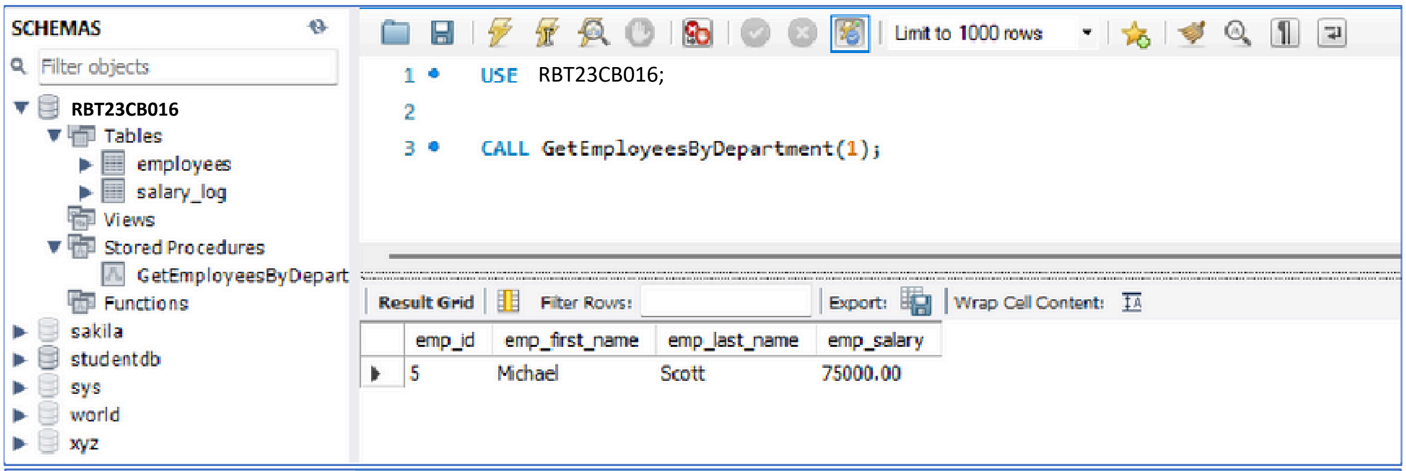


| first_name | last_name | salary | department_name |
|---|---|---|---|
| John | Doe | 60000.00 | Sales |
| Jane | Smith | 55000.00 | Sales |
| Alice | Johnson | 70000.00 | Marketing |
| Bob | Brown | 48000.00 | Marketing |
| Charlie | Davis | 52000.00 | HR |

```
CREATE SCHEMA RBT23CB016;
USE RBT23CB016;


-- Create the employees table
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT
);

-- Create a log table to track salary updates
CREATE TABLE salary_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id INT,
    old_salary DECIMAL(10, 2),
    new_salary DECIMAL(10, 2),
    change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Insert sample data into the employees table
INSERT INTO employees (first_name, last_name, email, hire_date, salary, department_id) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00, 1),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00, 1),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00, 2),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00, 2),
('Michael', 'Scott', 'michael.scott@example.com', '2023-10-01', 75000.00, 1);

-- Create the stored procedure with a parameterized cursor
DELIMITER //

CREATE PROCEDURE GetEmployeesByDepartment(IN dept_id INT)
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE emp_id INT;
    DECLARE emp_first_name VARCHAR(50);
    DECLARE emp_last_name VARCHAR(50);
    DECLARE emp_salary DECIMAL(10, 2);

    -- Declare a cursor for employees in the specified department
    DECLARE emp_cursor CURSOR FOR
        SELECT employee_id, first_name, last_name, salary
        FROM employees
        WHERE department_id = dept_id;

    -- Declare a NOT FOUND handler
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    -- Open the cursor
    OPEN emp_cursor;

    -- Loop through the cursor
    read_loop: LOOP
        FETCH emp_cursor INTO emp_id, emp_first_name, emp_last_name, emp_salary;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Output the employee details (for demonstration purposes)
        SELECT emp_id, emp_first_name, emp_last_name, emp_salary;
    END LOOP;

    -- Close the cursor
    CLOSE emp_cursor;
END //

DELIMITER ;

-- Create the trigger to log salary updates
DELIMITER //

CREATE TRIGGER after_salary_update
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
    -- Insert a record into the salary_log table
    INSERT INTO salary_log (employee_id, old_salary, new_salary)
    VALUES (OLD.employee_id, OLD.salary, NEW.salary);
END //

DELIMITER ;


CALL GetEmployeesByDepartment(1);



UPDATE employees SET salary = 80000 WHERE employee_id = 1;
```

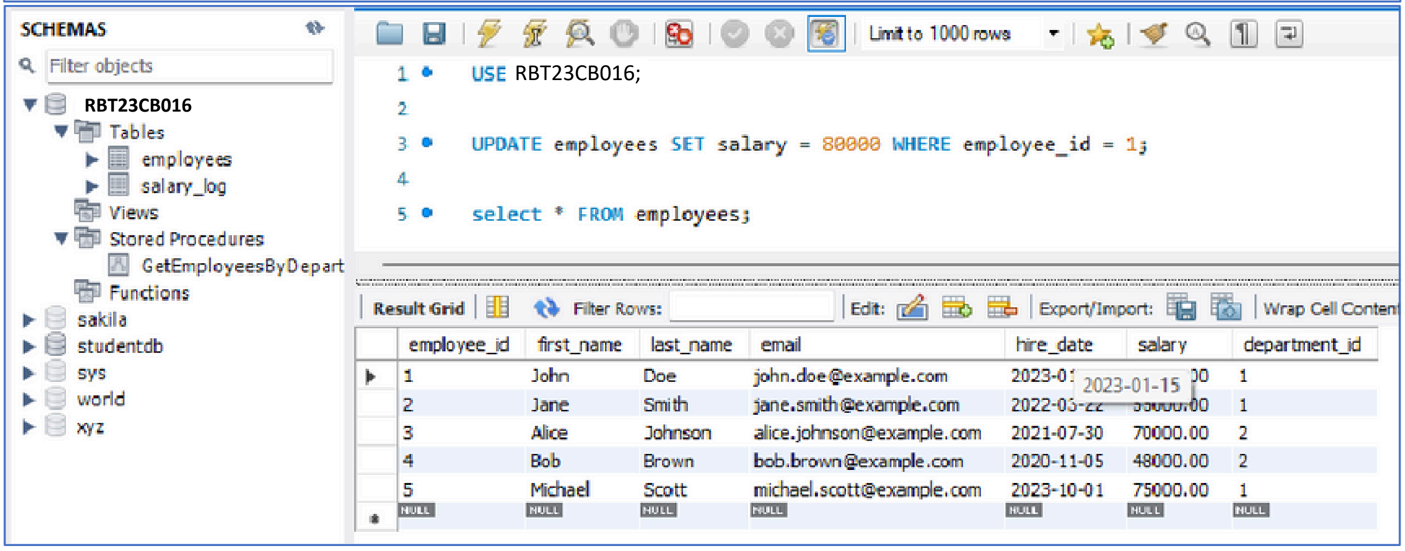


```sql
CREATE SCHEMA  RBT23CB016;
USE  RBT23CB016;

-- Create the employees table
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    hire_date DATE NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    department_id INT
);

-- Create a log table to track salary updates
CREATE TABLE salary_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id INT,
    old_salary DECIMAL(10, 2),
    new_salary DECIMAL(10, 2),
    change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Insert sample data into the employees table
INSERT INTO employees (first_name, last_name, email, hire_date, salary, department_id) VALUES
('John', 'Doe', 'john.doe@example.com', '2023-01-15', 60000.00, 1),
('Jane', 'Smith', 'jane.smith@example.com', '2022-03-22', 55000.00, 1),
('Alice', 'Johnson', 'alice.johnson@example.com', '2021-07-30', 70000.00, 2),
('Bob', 'Brown', 'bob.brown@example.com', '2020-11-05', 48000.00, 2),
('Michael', 'Scott', 'michael.scott@example.com', '2023-10-01', 75000.00, 1);

-- Create the stored procedure with a parameterized cursor
DELIMITER //
```



```sql
USE  RBT23CB016;

CALL GetEmployeesByDepartment(1);
```

| emp_id | emp_first_name | emp_last_name | emp_salary |
|---|---|---|---|
| 5 | Michael | Scott | 75000.00 |

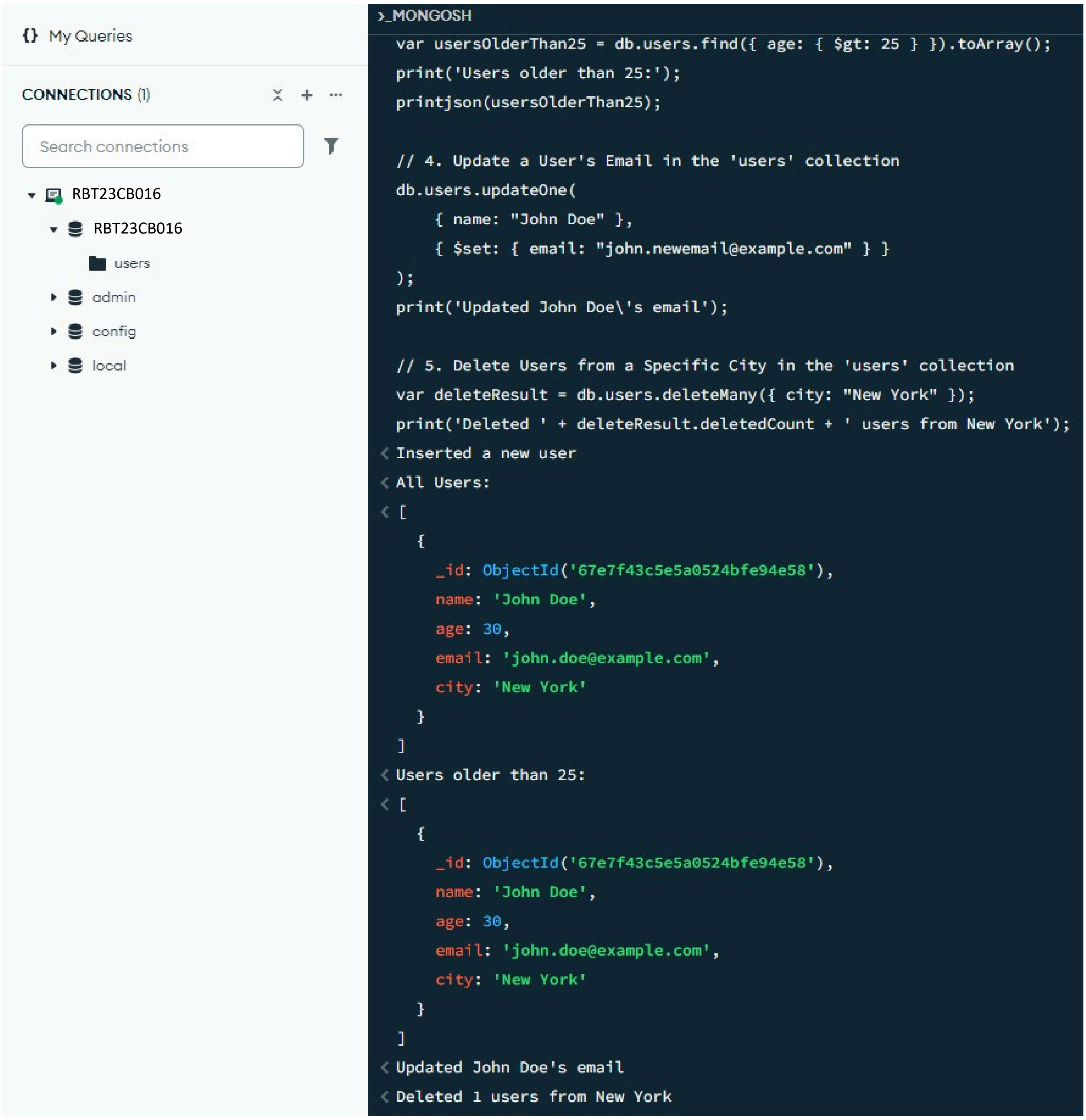


```sql
USE RBT23CB016;

UPDATE employees SET salary = 80000 WHERE employee_id = 1;

select * FROM employees;
```

| employee_id | first_name | last_name | email | hire_date | salary | department_id |
|---|---|---|---|---|---|---|
| 1 | John | Doe | john.doe@example.com | 2023-01-15 | [illegible] | 1 |
| 2 | Jane | Smith | jane.smith@example.com | 2022-03-22 | 55000.00 | 1 |
| 3 | Alice | Johnson | alice.johnson@example.com | 2021-07-30 | 70000.00 | 2 |
| 4 | Bob | Brown | bob.brown@example.com | 2020-11-05 | 48000.00 | 2 |
| 5 | Michael | Scott | michael.scott@example.com | 2023-10-01 | 75000.00 | 1 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

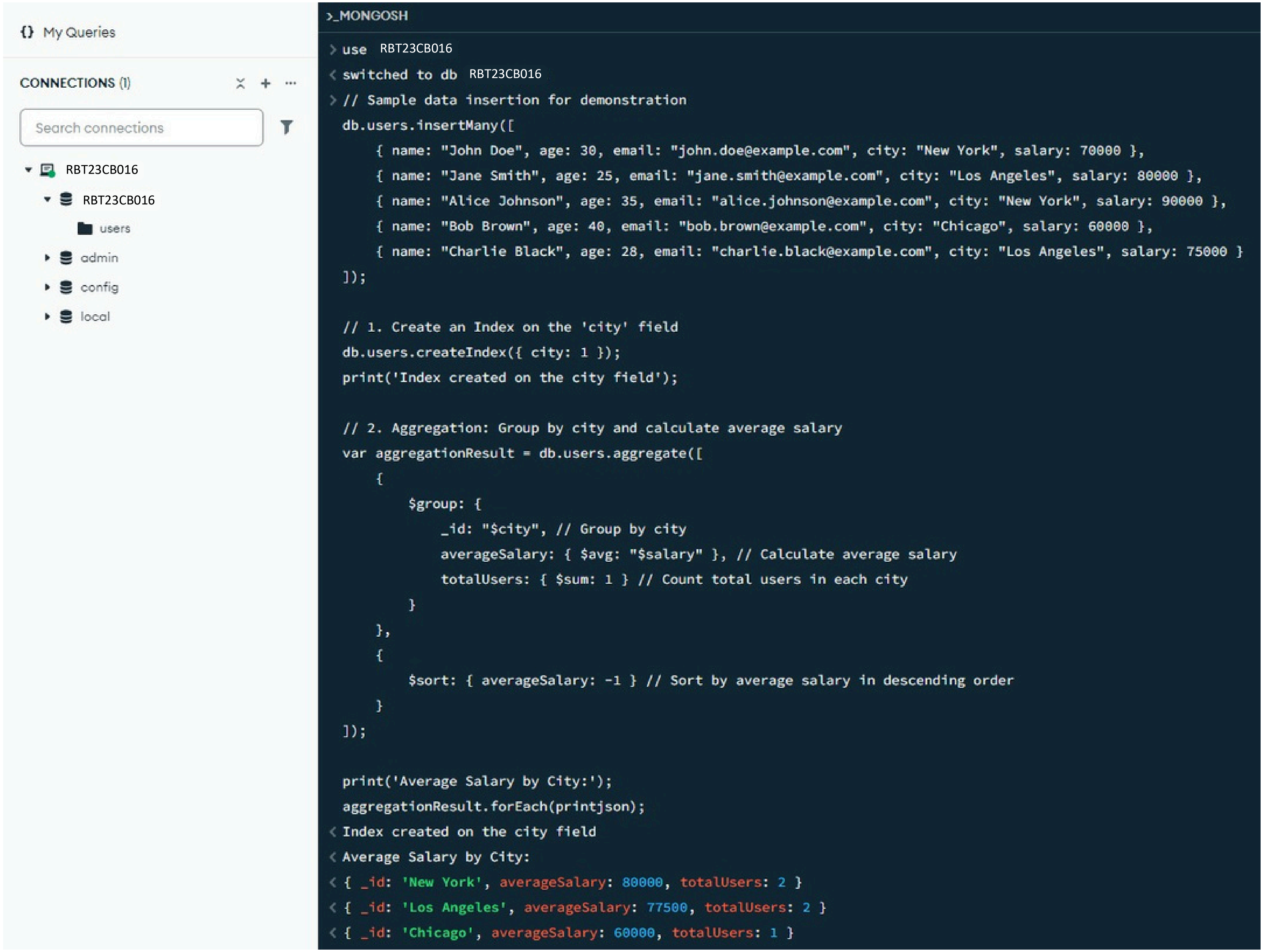
```
use RBT23CB016
 // 1. Insert a New User into the 'users' collection
db.users.insertOne({
   name: "John Doe",
   age: 30,
   email: "john.doe@example.com",
   city: "New York"
});
print('Inserted a new user');

// 2. Find All Users in the 'users' collection
var allUsers = db.users.find({}).toArray();
print('All Users:');
printjson(allUsers);

// 3. Find Users by Age in the 'users' collection
var usersOlderThan25 = db.users.find({ age: { $gt: 25 } }).toArray();
print('Users older than 25:');
printjson(usersOlderThan25);

// 4. Update a User's Email in the 'users' collection
db.users.updateOne(
   { name: "John Doe" },
   { $set: { email: "john.newemail@example.com" } }
);
print('Updated John Doe\'s email');

// 5. Delete Users from a Specific City in the 'users' collection
var deleteResult = db.users.deleteMany({ city: "New York" });
print('Deleted ' + deleteResult.deletedCount + ' users from New York');
```

5



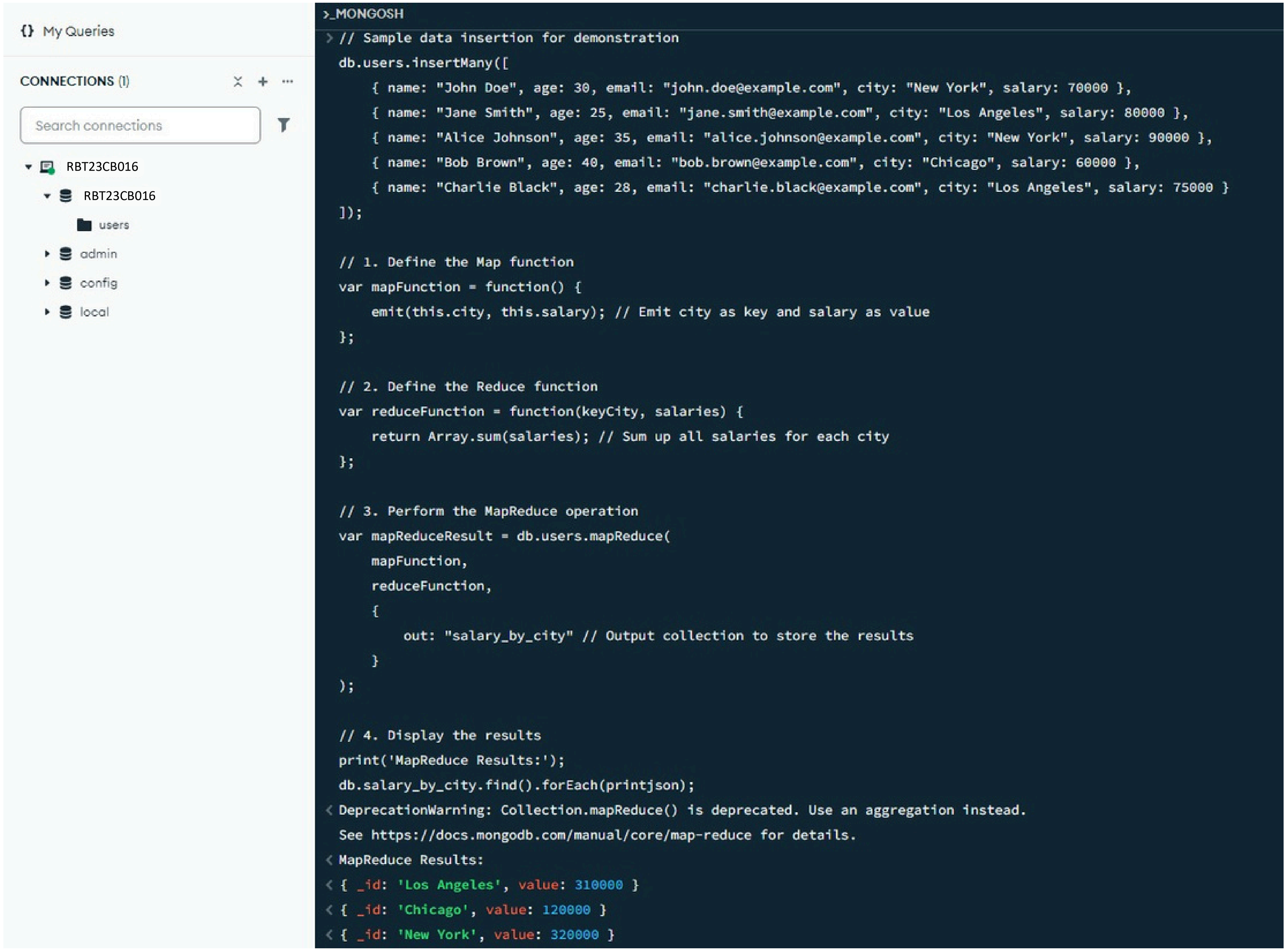
```
var usersOlderThan25 = db.users.find({ age: { $gt: 25 } }).toArray();
print('Users older than 25:');
printjson(usersOlderThan25);

// 4. Update a User's Email in the 'users' collection
db.users.updateOne(
    { name: "John Doe" },
    { $set: { email: "john.newemail@example.com" } }
);
print('Updated John Doe\'s email');

// 5. Delete Users from a Specific City in the 'users' collection
var deleteResult = db.users.deleteMany({ city: "New York" });
print('Deleted ' + deleteResult.deletedCount + ' users from New York');
< Inserted a new user
< All Users:
< [
    {
      _id: ObjectId('67e7f43c5e5a0524bfe94e58'),
      name: 'John Doe',
      age: 30,
      email: 'john.doe@example.com',
      city: 'New York'
    }
  ]
< Users older than 25:
< [
    {
      _id: ObjectId('67e7f43c5e5a0524bfe94e58'),
      name: 'John Doe',
      age: 30,
      email: 'john.doe@example.com',
      city: 'New York'
    }
  ]
< Updated John Doe's email
< Deleted 1 users from New York
```

```
use RBT23CB016;

// Sample data insertion for demonstration
db.users.insertMany([
   { name: "John Doe", age: 30, email: "john.doe@example.com", city: "New York", salary: 70000 },
   { name: "Jane Smith", age: 25, email: "jane.smith@example.com", city: "Los Angeles", salary: 80000 },
   { name: "Alice Johnson", age: 35, email: "alice.johnson@example.com", city: "New York", salary: 90000 },
   { name: "Bob Brown", age: 40, email: "bob.brown@example.com", city: "Chicago", salary: 60000 },
   { name: "Charlie Black", age: 28, email: "charlie.black@example.com", city: "Los Angeles", salary: 75000 }
]);

// 1. Create an Index on the 'city' field
db.users.createIndex({ city: 1 });
print('Index created on the city field');

// 2. Aggregation: Group by city and calculate average salary
var aggregationResult = db.users.aggregate([
   {
      $group: {
         _id: "$city", // Group by city
         averageSalary: { $avg: "$salary" }, // Calculate average salary
         totalUsers: { $sum: 1 } // Count total users in each city
      }
   },
   {
      $sort: { averageSalary: -1 } // Sort by average salary in descending order
   }
]);

print('Average Salary by City:');
aggregationResult.forEach(printjson);
```



```
> use  RBT23CB016
< switched to db  RBT23CB016
> // Sample data insertion for demonstration
  db.users.insertMany([
      { name: "John Doe", age: 30, email: "john.doe@example.com", city: "New York", salary: 70000 },
      { name: "Jane Smith", age: 25, email: "jane.smith@example.com", city: "Los Angeles", salary: 80000 },
      { name: "Alice Johnson", age: 35, email: "alice.johnson@example.com", city: "New York", salary: 90000 },
      { name: "Bob Brown", age: 40, email: "bob.brown@example.com", city: "Chicago", salary: 60000 },
      { name: "Charlie Black", age: 28, email: "charlie.black@example.com", city: "Los Angeles", salary: 75000 }
  ]);

  // 1. Create an Index on the 'city' field
  db.users.createIndex({ city: 1 });
  print('Index created on the city field');

  // 2. Aggregation: Group by city and calculate average salary
  var aggregationResult = db.users.aggregate([
      {
          $group: {
              _id: "$city", // Group by city
              averageSalary: { $avg: "$salary" }, // Calculate average salary
              totalUsers: { $sum: 1 } // Count total users in each city
          }
      },
      {
          $sort: { averageSalary: -1 } // Sort by average salary in descending order
      }
  ]);

  print('Average Salary by City:');
  aggregationResult.forEach(printjson);
< Index created on the city field
< Average Salary by City:
< { _id: 'New York', averageSalary: 80000, totalUsers: 2 }
< { _id: 'Los Angeles', averageSalary: 77500, totalUsers: 2 }
< { _id: 'Chicago', averageSalary: 60000, totalUsers: 1 }
```

```
use RBT23CB031
// Sample data insertion for demonstration
db.users.insertMany([
  { name: "John Doe", age: 30, email: "john.doe@example.com", city: "New York", salary: 70000 },
  { name: "Jane Smith", age: 25, email: "jane.smith@example.com", city: "Los Angeles", salary: 80000 },
  { name: "Alice Johnson", age: 35, email: "alice.johnson@example.com", city: "New York", salary: 90000 },
  { name: "Bob Brown", age: 40, email: "bob.brown@example.com", city: "Chicago", salary: 60000 },
  { name: "Charlie Black", age: 28, email: "charlie.black@example.com", city: "Los Angeles", salary: 75000 }
]);

// 1. Define the Map function
var mapFunction = function() {
  emit(this.city, this.salary); // Emit city as key and salary as value
};

// 2. Define the Reduce function
var reduceFunction = function(keyCity, salaries) {
  return Array.sum(salaries); // Sum up all salaries for each city
};

// 3. Perform the MapReduce operation
var mapReduceResult = db.users.mapReduce(
  mapFunction,
  reduceFunction,
  {
    out: "salary_by_city" // Output collection to store the results
  }
);

// 4. Display the results
print('MapReduce Results:');
db.salary_by_city.find().forEach(printjson);
```



```
> // Sample data insertion for demonstration
  db.users.insertMany([
      { name: "John Doe", age: 30, email: "john.doe@example.com", city: "New York", salary: 70000 },
      { name: "Jane Smith", age: 25, email: "jane.smith@example.com", city: "Los Angeles", salary: 80000 },
      { name: "Alice Johnson", age: 35, email: "alice.johnson@example.com", city: "New York", salary: 90000 },
      { name: "Bob Brown", age: 40, email: "bob.brown@example.com", city: "Chicago", salary: 60000 },
      { name: "Charlie Black", age: 28, email: "charlie.black@example.com", city: "Los Angeles", salary: 75000 }
  ]);

  // 1. Define the Map function
  var mapFunction = function() {
      emit(this.city, this.salary); // Emit city as key and salary as value
  };

  // 2. Define the Reduce function
  var reduceFunction = function(keyCity, salaries) {
      return Array.sum(salaries); // Sum up all salaries for each city
  };

  // 3. Perform the MapReduce operation
  var mapReduceResult = db.users.mapReduce(
      mapFunction,
      reduceFunction,
      {
          out: "salary_by_city" // Output collection to store the results
      }
  );

  // 4. Display the results
  print('MapReduce Results:');
  db.salary_by_city.find().forEach(printjson);
< DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
  See https://docs.mongodb.com/manual/core/map-reduce for details.
< MapReduce Results:
< { _id: 'Los Angeles', value: 310000 }
< { _id: 'Chicago', value: 120000 }
< { _id: 'New York', value: 320000 }
```