IMPLEMENTATION AND RESULT

ROLL NO:23ADR069
NAME :JANANI K
PROJECT TITLE: EXPENSE TRACKER APPLICATION USING JAVA

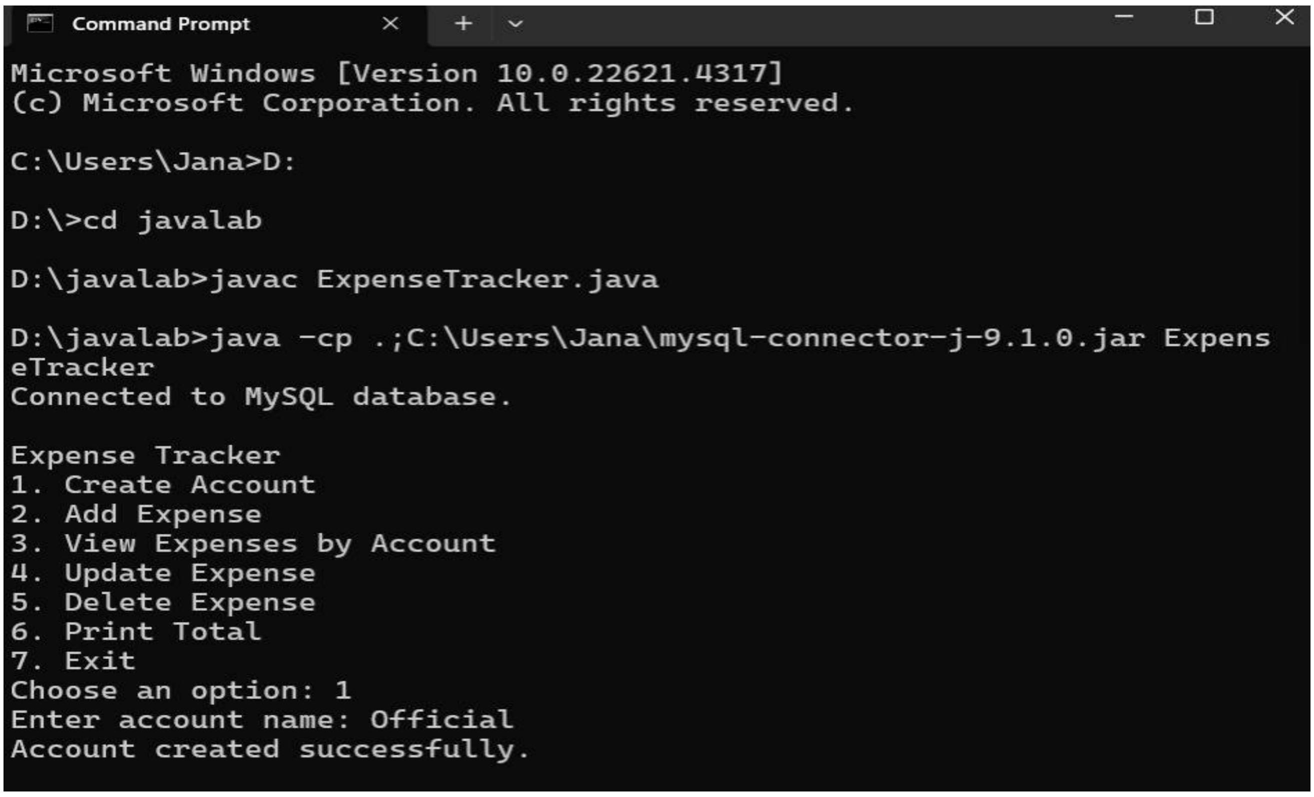
IMPLEMENTATION:

```
import java.sql.*;
import java.util.Scanner;

// Interface for common database operations
interface DatabaseOperations {
   void createTable();
   void add(Scanner scanner);
   void delete(Scanner scanner);
   void view(Scanner scanner);
}

// Abstract base class to manage the database connection
abstract class BaseEntity implements DatabaseOperations {
   protected static final String URL = "jdbc:mysql://localhost:3306/expense_tracker";
   protected static final String USER = "root";
   protected static final String PASSWORD = "21-Apr-06";
   protected static Connection conn;

   static {
      try {
         conn = DriverManager.getConnection(URL, USER, PASSWORD);
         System.out.println("Connected to MySQL database.");
      } catch (SQLException e) {
         System.out.println("Connection failed: " + e.getMessage());
      }
   }

   protected void closeConnection() {
      try {
         if (conn != null && !conn.isClosed()) {
            conn.close();
            System.out.println("Database connection closed.");
         }
      } catch (SQLException e) {
         System.out.println("Error closing connection: " + e.getMessage());
      }
   }
}

// Account class implementing database operations specific to accounts
class Account extends BaseEntity {

   @Override
   public void createTable() {
      String createAccountsTable = "CREATE TABLE IF NOT EXISTS accounts ("
            + "id INT AUTO_INCREMENT PRIMARY KEY, "
            + "name VARCHAR(255) NOT NULL);";

        try (Statement stmt = conn.createStatement()) {
            stmt.execute(createAccountsTable);
            System.out.println("Accounts table created.");
        } catch (SQLException e) {
            System.out.println("Error creating accounts table: " + e.getMessage());
        }
    }

    @Override
    public void add(Scanner scanner) {
        System.out.print("Enter account name: ");
        String name = scanner.nextLine();

        String sql = "INSERT INTO accounts(name) VALUES(?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.executeUpdate();
            System.out.println("Account created successfully.");
        } catch (SQLException e) {
            System.out.println("Error creating account: " + e.getMessage());
        }
    }

    @Override
    public void delete(Scanner scanner) {
        System.out.println("Delete operation for Account not implemented.");
    }

    @Override
    public void view(Scanner scanner) {
        String sql = "SELECT * FROM accounts";
        try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
            System.out.println("\nAccounts:");
            while (rs.next()) {
                System.out.printf("%d - %s%n", rs.getInt("id"), rs.getString("name"));
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving accounts: " + e.getMessage());
        }
    }

    public int chooseAccount(Scanner scanner) {
        view(scanner);
        System.out.print("Enter account ID to select an account: ");
        return scanner.nextInt();
    }
}

// Expense class implementing database operations specific to expenses
class Expense extends BaseEntity {

    @Override
    public void createTable() {
```

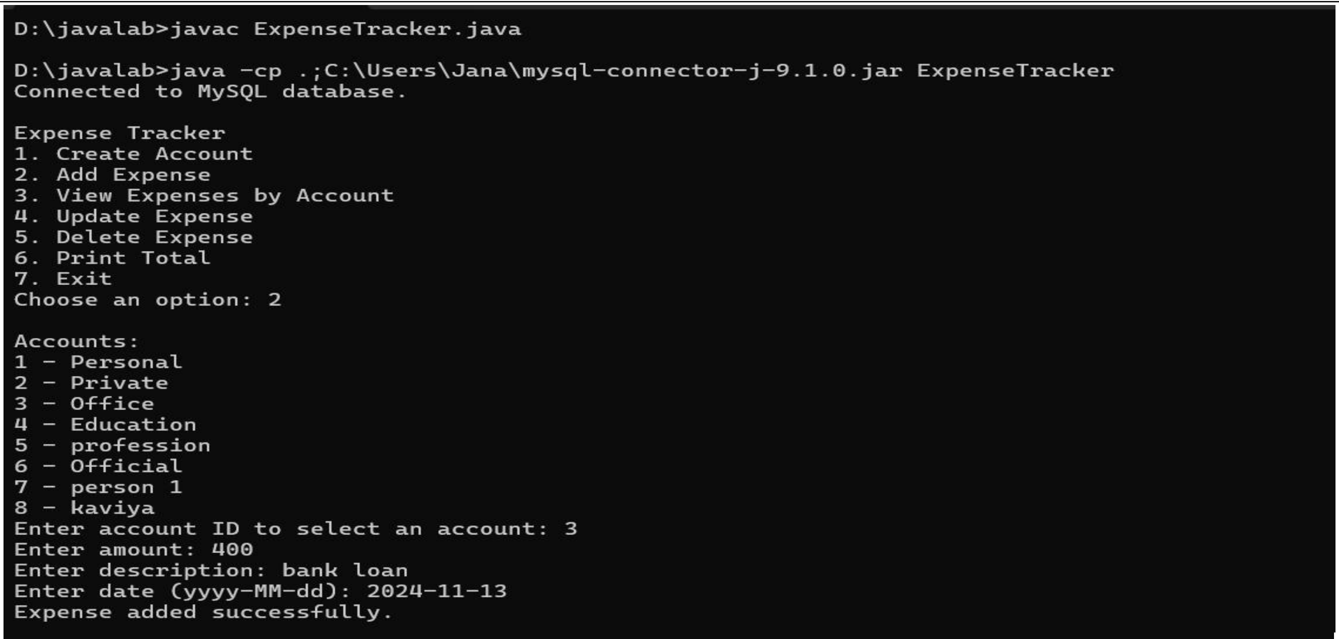
```java
    String createExpensesTable = "CREATE TABLE IF NOT EXISTS expenses ("
            + "id INT AUTO_INCREMENT PRIMARY KEY, "
            + "account_id INT, "
            + "amount DOUBLE NOT NULL, "
            + "description VARCHAR(255) NOT NULL, "
            + "date DATE NOT NULL, "
            + "FOREIGN KEY (account_id) REFERENCES accounts(id));";

    try (Statement stmt = conn.createStatement()) {
        stmt.execute(createExpensesTable);
        System.out.println("Expenses table created.");
    } catch (SQLException e) {
        System.out.println("Error creating expenses table: " + e.getMessage());
    }
}

@Override
public void add(Scanner scanner) {
    Account account = new Account();
    int accountId = account.chooseAccount(scanner);
    if (accountId == -1) {
        System.out.println("No account selected.");
        return;
    }

    System.out.print("Enter amount: ");
    double amount = scanner.nextDouble();
    scanner.nextLine();

    System.out.print("Enter date (yyyy-MM-dd): ");
    String date = scanner.nextLine();

    double totalForDate = getTotalExpenseForDate(accountId, date);
    if (totalForDate + amount > ExpenseTracker.DAILY_EXPENSE_LIMIT) {
        System.out.println("Overspending! You have exceeded the daily expense limit.");
        return;
    }

    System.out.print("Enter description: ");
    String description = scanner.nextLine();

    String sql = "INSERT INTO expenses(account_id, amount, description, date) VALUES(?, ?, ?, ?)";
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, accountId);
        pstmt.setDouble(2, amount);
        pstmt.setString(3, description);
        pstmt.setString(4, date);
        pstmt.executeUpdate();
        System.out.println("Expense added successfully.");
    } catch (SQLException e) {
        System.out.println("Error adding expense: " + e.getMessage());
    }
}

private double getTotalExpenseForDate(int accountId, String date) {
    String sql = "SELECT SUM(amount) AS total FROM expenses WHERE account_id = ? AND date
= ?";
    double total = 0.0;

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, accountId);
        pstmt.setString(2, date);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            total = rs.getDouble("total");
        }
    } catch (SQLException e) {
        System.out.println("Error retrieving total expenses for the date: " + e.getMessage());
    }
    return total;
}

@Override
public void delete(Scanner scanner) {
    System.out.print("Enter date (yyyy-MM-dd) to delete expenses: ");
    String dateStr = scanner.nextLine();

    String sql = "DELETE FROM expenses WHERE date = ?";
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, dateStr);
        int rowsAffected = pstmt.executeUpdate();

        if (rowsAffected > 0) {
            System.out.println("Expenses for the date deleted successfully.");
        } else {
            System.out.println("No expense found for the specified date.");
        }
    } catch (SQLException e) {
        System.out.println("Error deleting expenses: " + e.getMessage());
    }
}

@Override
public void view(Scanner scanner) {
    Account account = new Account();
    int accountId = account.chooseAccount(scanner);
    if (accountId == -1) {
        System.out.println("No account selected.");
        return;
    }

    String sql = "SELECT * FROM expenses WHERE account_id = ?";
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, accountId);
        ResultSet rs = pstmt.executeQuery();

        System.out.println("\nID | Amount | Description | Date");
        System.out.println("-------------------------------------------");
        while (rs.next()) {
```

```java
                System.out.printf("%d | %.2f | %s | %s%n",
                        rs.getInt("id"),
                        rs.getDouble("amount"),
                        rs.getString("description"),
                        rs.getDate("date").toString());
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving expenses: " + e.getMessage());
        }
    }
}

// Main application class
public class ExpenseTracker {
    public static double DAILY_EXPENSE_LIMIT;

    public static void main(String[] args) {
        Account account = new Account();
        Expense expense = new Expense();

        // Create tables
        account.createTable();
        expense.createTable();

        Scanner scanner = new Scanner(System.in);

        // Set daily expense limit
        setExpenseLimit(scanner);

        int choice;

        do {
            System.out.println("\nExpense Tracker");
            System.out.println("1. Create Account");
            System.out.println("2. Add Expense");
            System.out.println("3. View Expenses by Account");
            System.out.println("4. Delete Expense");
            System.out.println("5. Update Expense Limit");
            System.out.println("6. Exit");
            System.out.print("Choose an option: ");
            choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1 -> account.add(scanner);
                case 2 -> expense.add(scanner);
                case 3 -> expense.view(scanner);
                case 4 -> expense.delete(scanner);
                case 5 -> setExpenseLimit(scanner);
                case 6 -> System.out.println("Exiting...");
                default -> System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 6);

        scanner.close();
        expense.closeConnection();
    }

    private static void setExpenseLimit(Scanner scanner) {
        System.out.print("Enter expense limit: ");
        DAILY_EXPENSE_LIMIT = scanner.nextDouble();
        scanner.nextLine();
        System.out.println("Expense limit set to: " + DAILY_EXPENSE_LIMIT);
    }
}
```

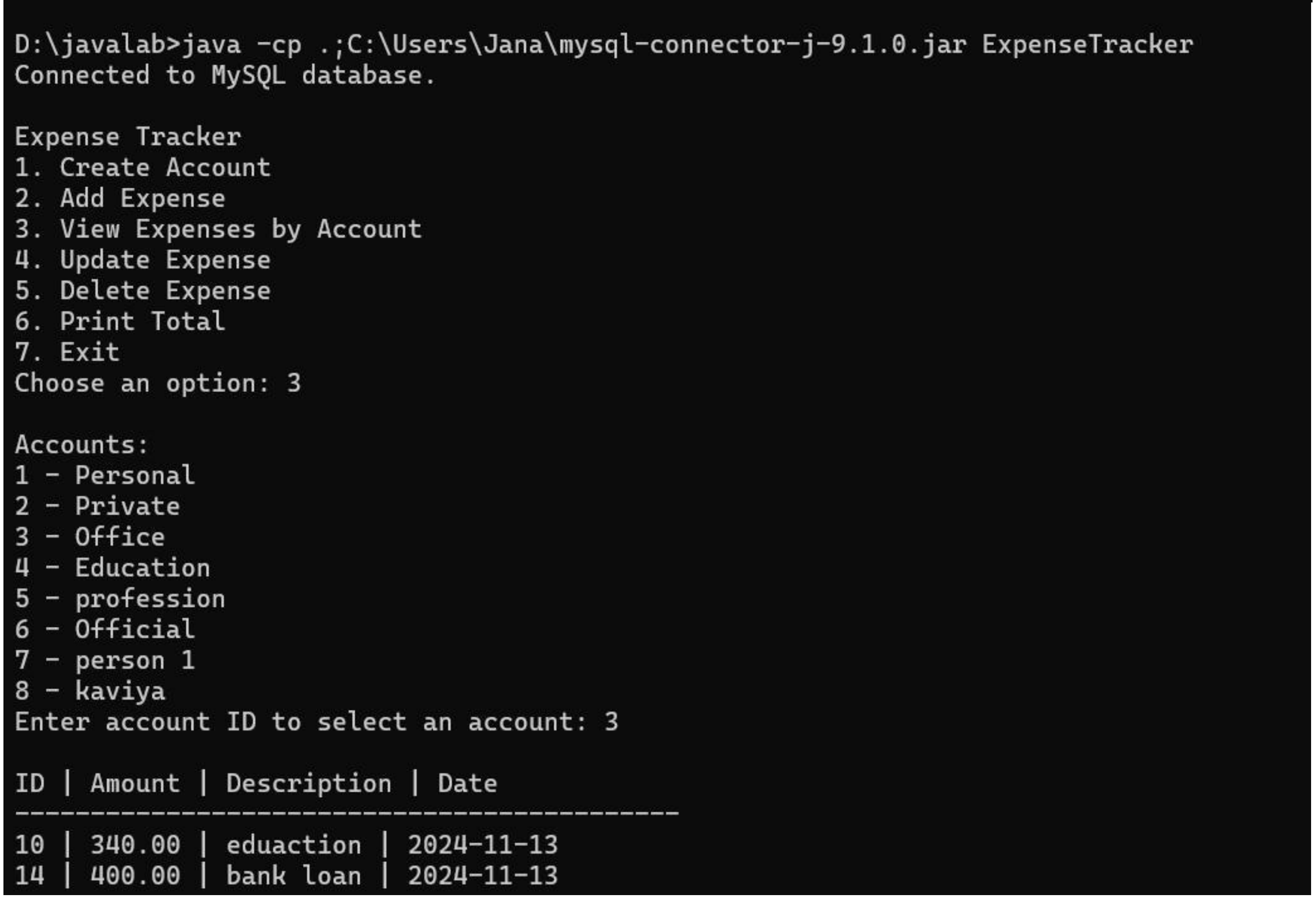
RESULT:



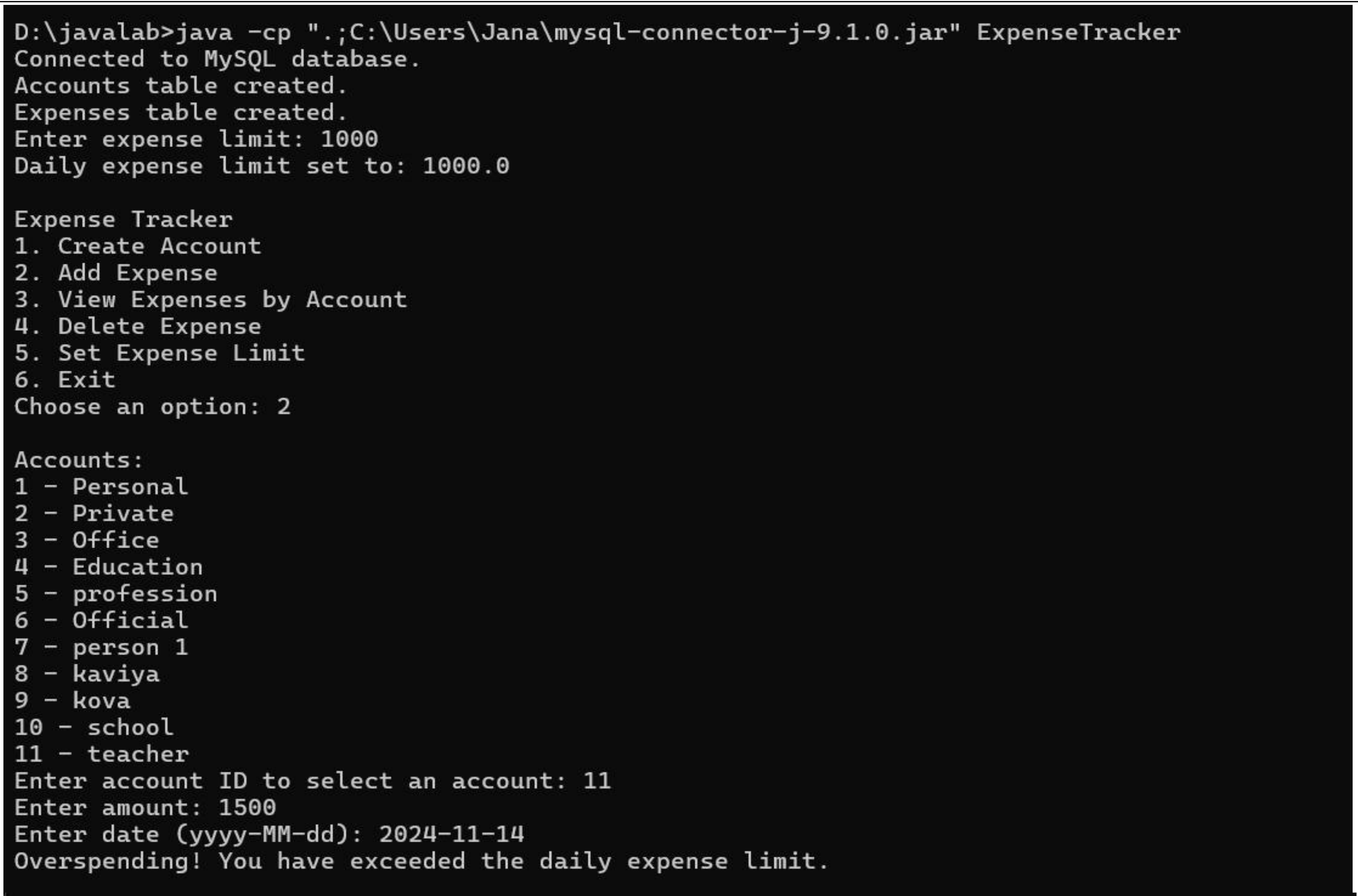
```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jana>D:

D:\>cd javalab

D:\javalab>javac ExpenseTracker.java

D:\javalab>java -cp .;C:\Users\Jana\mysql-connector-j-9.1.0.jar Expens
eTracker
Connected to MySQL database.

Expense Tracker
1. Create Account
2. Add Expense
3. View Expenses by Account
4. Update Expense
5. Delete Expense
6. Print Total
7. Exit
Choose an option: 1
Enter account name: Official
Account created successfully.
```



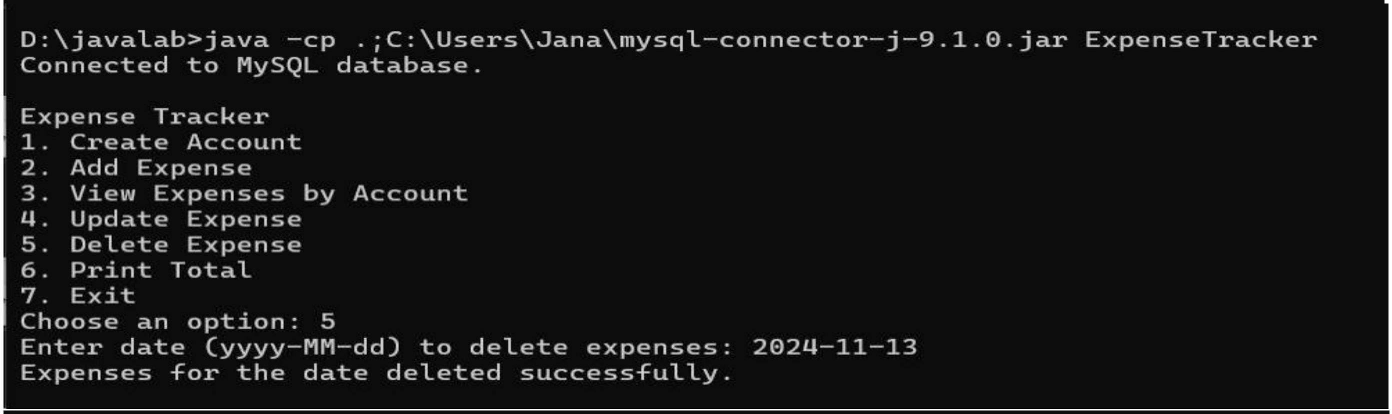
```
D:\javalab>javac ExpenseTracker.java

D:\javalab>java -cp .;C:\Users\Jana\mysql-connector-j-9.1.0.jar ExpenseTracker
Connected to MySQL database.

Expense Tracker
1. Create Account
2. Add Expense
3. View Expenses by Account
4. Update Expense
5. Delete Expense
6. Print Total
7. Exit
Choose an option: 2

Accounts:
1 - Personal
2 - Private
3 - Office
4 - Education
5 - profession
6 - Official
7 - person 1
8 - kaviya
Enter account ID to select an account: 3
Enter amount: 400
Enter description: bank loan
Enter date (yyyy-MM-dd): 2024-11-13
Expense added successfully.
```



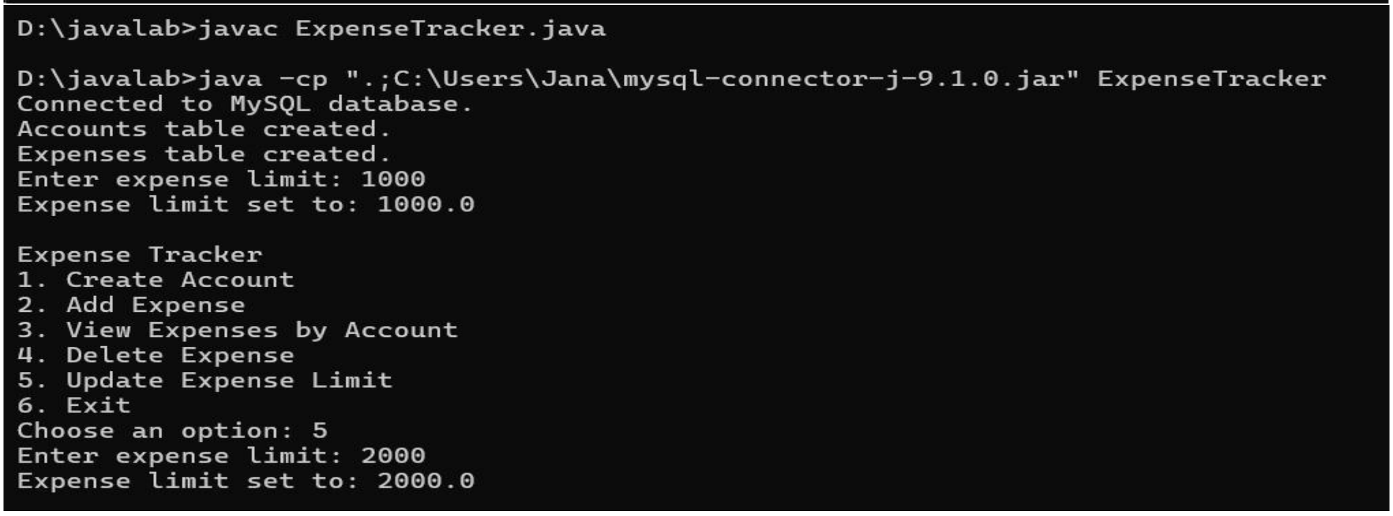
```
D:\javalab>java -cp .;C:\Users\Jana\mysql-connector-j-9.1.0.jar ExpenseTracker
Connected to MySQL database.

Expense Tracker
1. Create Account
2. Add Expense
3. View Expenses by Account
4. Update Expense
5. Delete Expense
6. Print Total
7. Exit
Choose an option: 3

Accounts:
1 - Personal
2 - Private
3 - Office
4 - Education
5 - profession
6 - Official
7 - person 1
8 - kaviya
Enter account ID to select an account: 3

ID | Amount | Description | Date
--------------------------------------------
10 | 340.00 | eduaction | 2024-11-13
14 | 400.00 | bank loan | 2024-11-13
```



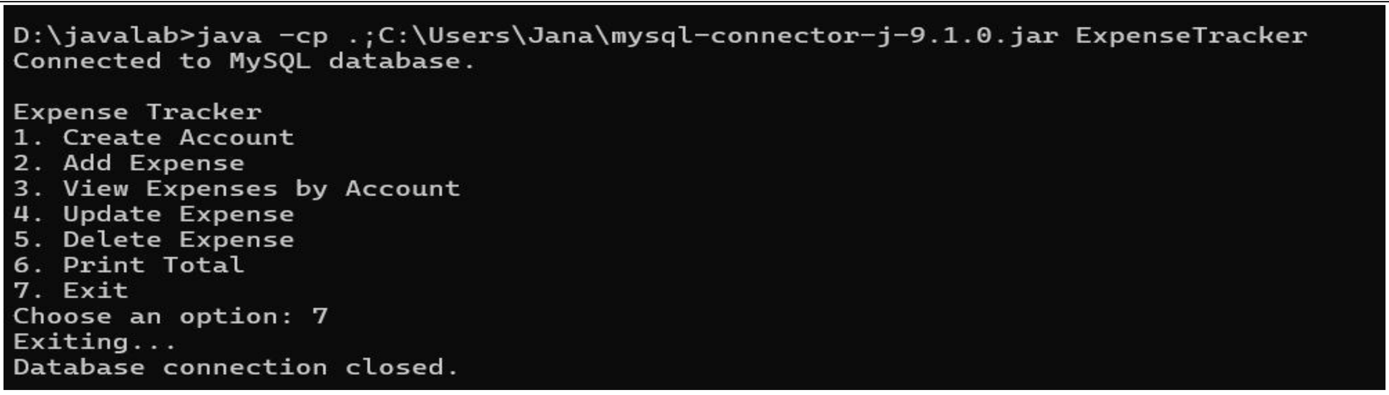
```
D:\javalab>java -cp ".;C:\Users\Jana\mysql-connector-j-9.1.0.jar" ExpenseTracker
Connected to MySQL database.
Accounts table created.
Expenses table created.
Enter expense limit: 1000
Daily expense limit set to: 1000.0

Expense Tracker
1. Create Account
2. Add Expense
3. View Expenses by Account
4. Delete Expense
5. Set Expense Limit
6. Exit
Choose an option: 2

Accounts:
1 - Personal
2 - Private
3 - Office
4 - Education
5 - profession
6 - Official
7 - person 1
8 - kaviya
9 - kova
10 - school
11 - teacher
Enter account ID to select an account: 11
Enter amount: 1500
Enter date (yyyy-MM-dd): 2024-11-14
Overspending! You have exceeded the daily expense limit.
```



```
D:\javalab>java -cp .;C:\Users\Jana\mysql-connector-j-9.1.0.jar ExpenseTracker
Connected to MySQL database.

Expense Tracker
1. Create Account
2. Add Expense
3. View Expenses by Account
4. Update Expense
5. Delete Expense
6. Print Total
7. Exit
Choose an option: 5
Enter date (yyyy-MM-dd) to delete expenses: 2024-11-13
Expenses for the date deleted successfully.
```

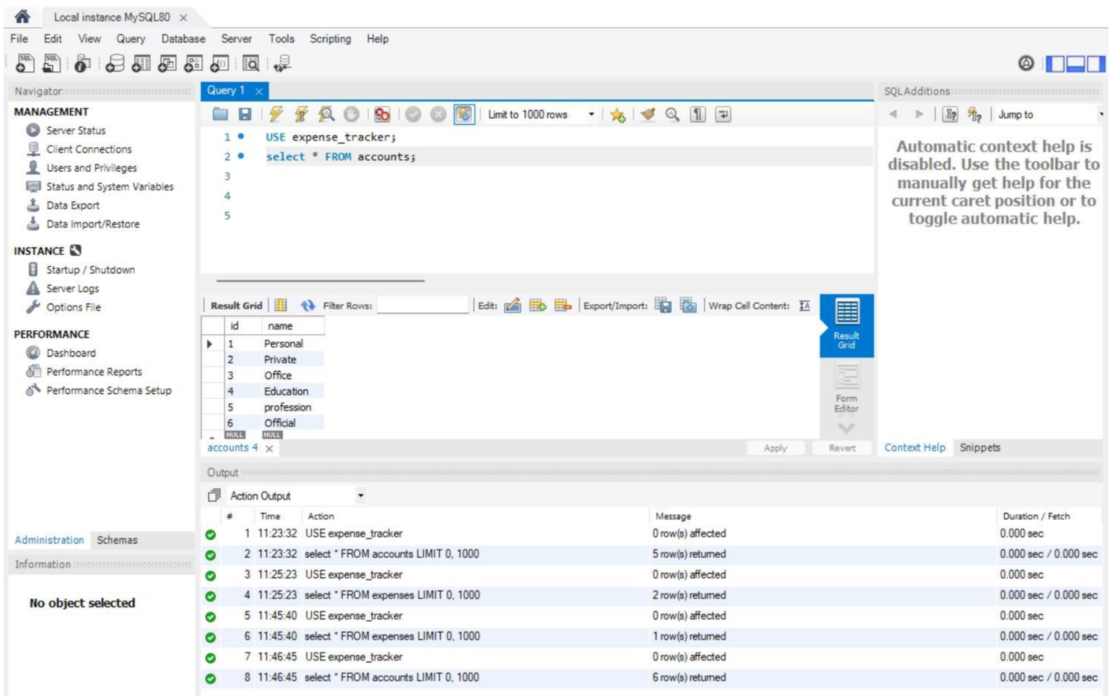


```
D:\javalab>javac ExpenseTracker.java

D:\javalab>java -cp ".;C:\Users\Jana\mysql-connector-j-9.1.0.jar" ExpenseTracker
Connected to MySQL database.
Accounts table created.
Expenses table created.
Enter expense limit: 1000
Expense limit set to: 1000.0

Expense Tracker
1. Create Account
2. Add Expense
3. View Expenses by Account
4. Delete Expense
5. Update Expense Limit
6. Exit
Choose an option: 5
Enter expense limit: 2000
Expense limit set to: 2000.0
```



```
D:\javalab>java -cp .;C:\Users\Jana\mysql-connector-j-9.1.0.jar ExpenseTracker
Connected to MySQL database.

Expense Tracker
1. Create Account
2. Add Expense
3. View Expenses by Account
4. Update Expense
5. Delete Expense
6. Print Total
7. Exit
Choose an option: 7
Exiting...
Database connection closed.
```

DATABASE OUTPUT:

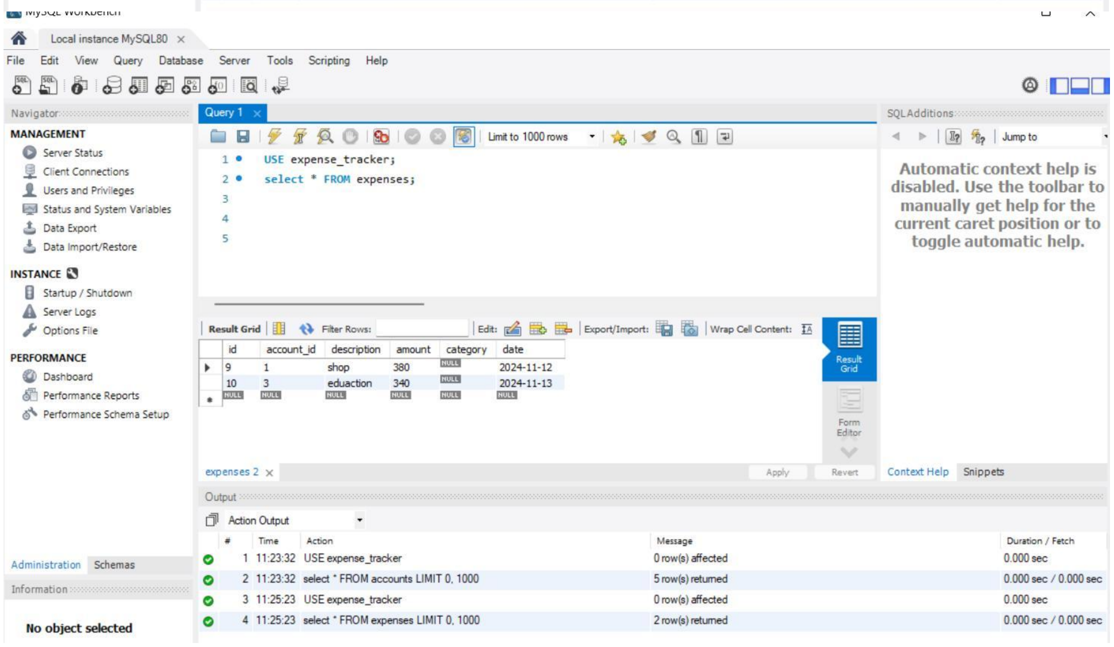


```sql
USE expense_tracker;
select * FROM accounts;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

id	name
1	Personal
2	Private
3	Office
4	Education
5	profession
6	Official
NULL	NULL

#	Time	Action	Message	Duration / Fetch
1	11:23:32	USE expense_tracker	0 row(s) affected	0.000 sec
2	11:23:32	select * FROM accounts LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	11:25:23	USE expense_tracker	0 row(s) affected	0.000 sec
4	11:25:23	select * FROM expenses LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
5	11:45:40	USE expense_tracker	0 row(s) affected	0.000 sec
6	11:45:40	select * FROM expenses LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
7	11:46:45	USE expense_tracker	0 row(s) affected	0.000 sec
8	11:46:45	select * FROM accounts LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec



```sql
USE expense_tracker;
select * FROM expenses;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

id	account_id	description	amount	category	date
9	1	shop	380	NULL	2024-11-12
10	3	eduaction	340	NULL	2024-11-13
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message	Duration / Fetch
1	11:23:32	USE expense_tracker	0 row(s) affected	0.000 sec
2	11:23:32	select * FROM accounts LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
3	11:25:23	USE expense_tracker	0 row(s) affected	0.000 sec
4	11:25:23	select * FROM expenses LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec