# CASE STUDY: FINANCE MANAGEMNT SYSTEM

# SQL CODE AND OUTPUTS:(compiled at MySQL workbench)

# **CODE:**

#### **OUTPUT:**

```
    2 15:18:19 CREATE DATABASE FinanceDB
    3 15:18:19 USE FinanceDB
    4 15:18:19 CREATE TABLE Users ( user_id INT PRIMARY KEY AUTO_INCREMENT, username VARCHAR(50) UN... 0 row(s) affected
```

#### CODE:

```
11 • CREATE TABLE ExpenseCategories (

12 category_id INT PRIMARY KEY AUTO_INCREMENT,

13 category_name VARCHAR(50) UNIQUE NOT NULL

14 );
```

# **OUTPUT:**

5 15:18:19 CREATE TABLE ExpenseCategories ( category\_id INT PRIMARY KEY AUTO\_INCREMENT, category\_n... 0 row(s) affected

## **CODE:**

```
16 • ○ CREATE TABLE Expenses (
17
           expense_id INT PRIMARY KEY AUTO_INCREMENT,
           user id INT NOT NULL,
18
19
           amount DECIMAL(10,2) NOT NULL,
20
           category_id INT,
21
           date DATE NOT NULL,
           expense_name VARCHAR(100) NOT NULL,
22
           FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
23
           FOREIGN KEY (category_id) REFERENCES ExpenseCategories(category_id) ON DELETE SET NULL
     ز( ا
25
```

#### **OUTPUT:**

```
6 15:18:19 CREATE TABLE Expenses ( expense_id INT PRIMARY KEY AUTO_INCREMENT, user_id INT NOT NU... 0 row(s) affected
```

#### CODE:

```
INSERT INTO ExpenseCategories (category_name) VALUES ('Food'), ('Transport'), ('Entertainment');
```

#### **OUTPUT:**

7 15:10:20 INICEDT INITO Eve

7 15:18:36 INSERT INTO ExpenseCategories (category\_name) VALUES ('Food'), ('Transport'), ('Entertainment')

3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters, setters)

#### **ENTITY CLASS:**

### User.java:

```
package entity;

public class User {
    private int userId;
    private String username;
    private String password;
    private String email;

public User() {}

public User(int userId, String username, String password, String email) {
        this.userId = userId;
        this.username = username;
        this.password = password;
        this.email = email;
    }

public int getUserId() { return userId; }

public String getPassword() { return username; }

public String getEmail() { return email; }

public void setUserId(int userId) { this.userId = userId; }

public void setUserIname(String username) { this.username = username; }
```

```
public void setPassword(String password) { this.password = password; }
  public void setEmail(String email) { this.email = email; }
}
```

# Expense.java

```
package entity;
import java.util.Date;

public class Expense {
    private int id;
    private int userId;
    private String name;
    private double amount;
    private int categoryId;
    private Date date;

    public Expense(int id, int userId, double amount, int categoryId, Date date, String name) {
        this.id = id;
        this.userId = userId;
        this.amount = amount;
        this.categoryId = categoryId;
        this.date = date;
        this.name = name;
    }

    public int getId() { return id; }
    public int getUserId() { return name; }
    public double getAmount() { return amount; }
    public int getCategoryId() { return categoryId; }
    public Date getDate() { return date; }
}
```

# ExpenseCategory.java

```
package entity;

public class ExpenseCategory {
    private int categoryId;
    private String categoryName;

    // Constructors
    public ExpenseCategory() {}

    public ExpenseCategory(int categoryId, String categoryName) {
        this.categoryId = categoryId;
        this.categoryName = categoryName;
    }

    // Getters
```

```
public int getCategoryId() {
    return categoryId;
}

public String getCategoryName() {
    return categoryName;
}

// Setters
public void setCategoryId(int categoryId) {
    this.categoryId = categoryId;
}

public void setCategoryName(String categoryName) {
    this.categoryName = categoryName;
}
```

## **DAO PACKAGE:**

# IFinanceRepository.java

```
import entity.Expense;
import entity.User;
import java.util.List;

public interface IFinanceRepository {
    boolean createUser(User user);
    boolean createExpense(Expense expense);
    boolean deleteUser(int userId);
    boolean deleteExpense(int expenseId);
    boolean updateExpense(int userId, Expense expense);
    List<Expense> getAllExpenses(int userId);
}
```

# FinanceRepositoryImpl.java:

```
package dao;
import util.DBConnUtil;
import entity.Expense;
import entity.User;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class FinanceRepositoryImpl {
    private Connection connection;

    public FinanceRepositoryImpl() {
```

```
connection = DBConnUtil.getConnection();
private boolean userExists(int userId) {
    String query = "SELECT 1 FROM Users WHERE user id = ?";
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setInt(1, userId);
        try (ResultSet rs = pstmt.executeQuery()) {
            return rs.next();
    } catch (SQLException e) {
        e.printStackTrace();
    String query = "SELECT 1 FROM ExpenseCategories WHERE category id =
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setInt(1, categoryId);
        try (ResultSet rs = pstmt.executeQuery()) {
    } catch (SQLException e) {
        e.printStackTrace();
private boolean expenseExists(int expenseId) {
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setInt(1, expenseId);
        try (ResultSet rs = pstmt.executeQuery()) {
            return rs.next();
    } catch (SQLException e) {
        e.printStackTrace();
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setString(1, user.getUsername());
        pstmt.setString(2, user.getPassword());
pstmt.setString(3, user.getEmail());
        return pstmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
```

```
public boolean addExpense (Expense expense) {
    if (!userExists(expense.getUserId())) {
        System.out.println("Error: User ID does not exist.");
    if (!categoryExists(expense.getCategoryId())) {
    String query = "INSERT INTO Expenses (user id, expense name, amount,
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setInt(1, expense.getUserId());
        pstmt.setDouble(3, expense.getAmount());
        pstmt.setInt(4, expense.getCategoryId());
        pstmt.setDate(5, new java.sql.Date(expense.getDate().getTime()));
        return pstmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
public boolean deleteUser(int userId) {
    if (!userExists(userId)) {
        System.out.println("Error: User does not exist.");
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setInt(1, userId);
        return pstmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
public boolean deleteExpense(int expenseId) {
    if (!expenseExists(expenseId)) {
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setInt(1, expenseId);
        return pstmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
```

```
public boolean updateExpense(int userId, Expense expense) {
    if (!userExists(userId)) {
        System.out.println("Error: User ID does not exist.");
    if (!expenseExists(expense.getId())) {
        System.out.println("Error: Expense ID does not exist.");
    if (!categoryExists(expense.getCategoryId())) {
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setString(1, expense.getName());
        pstmt.setDouble(2, expense.getAmount());
        pstmt.setInt(3, expense.getCategoryId());
        pstmt.setInt(4, expense.getId());
        pstmt.setInt(5, userId);
        return pstmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
public List<Expense> getExpensesByUser(int userId) {
    List<Expense> expenses = new ArrayList<>();
    try (PreparedStatement pstmt = connection.prepareStatement(query)) {
        pstmt.setInt(1, userId);
        try (ResultSet rs = pstmt.executeQuery()) {
                Expense expense = new Expense (
                        rs.getInt("user id"),
                        rs.getString("expense name")
                expenses.add(expense);
    } catch (SQLException e) {
        e.printStackTrace();
    return expenses;
```

```
// Close connection method
public void closeConnection() {
    if (connection != null) {
        try {
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

# UTIL PACKAGE: DBConnUtil.java

```
package util;
import java.io.IOException;
import java.sql.SQLException;
            connString = DBPropertyUtil.getConnectionString(fileName);
        } catch (IOException e) {
            System.out.println("Connection String Creation Failed");
            e.printStackTrace();
                con = DriverManager.getConnection(connString);
            } catch (SQLException e) {
                e.printStackTrace();
```

# DBPropertyUtil.java:

# db.properties:

```
protocol=jdbc:mysql:
    system=localhost
port=3306
database=FinanceDB
user=root
password=Sandhiya_21
```

#### **EXCEPTION PACKAGE:**

## **UserNotFoundException:**

```
package exception;

public class UserNotFoundException extends Exception {
    public UserNotFoundException(String message) {
```

```
super(message);
}
```

## **ExpenseNotFoundException:**

```
package exception;

public class ExpenseNotFoundException extends Exception {
    public ExpenseNotFoundException(String message) {
        super(message);
    }
}
```

#### MAIN:

# FinanceApp.java

```
System.out.println("User creation failed!");
               System.out.print("Enter Amount: ");
               System.out.print("Enter Category ID: ");
                int categoryId = scanner.nextInt();
                Date date = new Date(); // Current date
                Expense expense = new Expense(0, userId, amount, categoryId,
                if (financeRepo.addExpense(expense)) {
                   System.out.println("Expense added successfully!");
                    System.out.println("Expense addition failed!");
                if (financeRepo.deleteUser(userId)) {
                    System.out.println("User deletion failed!");
                System.out.print("Enter Expense ID to Delete: ");
                int expenseId = scanner.nextInt();
                if (financeRepo.deleteExpense(expenseId)) {
                    System.out.println("Expense deletion failed!");
               System.out.print("Enter User ID: ");
                int userId = scanner.nextInt();
               System.out.print("Enter Expense ID to Update: ");
               int expenseId = scanner.nextInt();
               System.out.print("Enter New Expense Name: ");
               System.out.print("Enter New Amount: ");
                System.out.print("Enter New Category ID: ");
                int categoryId = scanner.nextInt();
                Expense expense = new Expense(expenseId, userId, amount,
categoryId, new Date(), name);
                if (financeRepo.updateExpense(userId, expense)) {
```

#### **UNIT TESTING:**

# UserTest.java

```
import dao.FinanceRepositoryImpl;
import entity.User;
import org.junit.Test;
import static org.junit.Assert.*;

public class UserTest {

    FinanceRepositoryImpl financeRepo = new FinanceRepositoryImpl();

    @Test
    public void testUserCreatedSuccessfully() {

        FinanceRepositoryImpl repo = new FinanceRepositoryImpl();

        String randomUsername = "testuser" + System.currentTimeMillis();

        User user = new User(0, randomUsername, "junitpass", randomUsername +

"@example.com");

        assertTrue(repo.createUser(user));
    }
}
```

# ExpenseTest.java:

```
import dao.FinanceRepositoryImpl;
import entity.Expense;
import org.junit.Test;
import java.util.Date;
import static org.junit.Assert.*;

public class ExpenseTest {

    FinanceRepositoryImpl financeRepo = new FinanceRepositoryImpl();

    @Test
    public void testExpenseCreatedSuccessfully() {
        Expense expense = new Expense(0, 1, 1000.0, 1, new Date(), "Food");
        boolean result = financeRepo.addExpense(expense);
        assertTrue(result);
    }
}
```

# ExpenseSearchTest.java:

```
import dao.FinanceRepositoryImpl;
import entity.Expense;
import org.junit.Test;

import java.util.List;

import static org.junit.Assert.*;

public class ExpenseSearchTest {

    FinanceRepositoryImpl financeRepo = new FinanceRepositoryImpl();

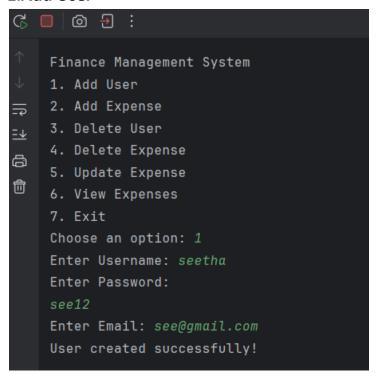
    @Test
    public void testSearchExpensesByUser() {
        int userId = 1; // Existing user
        List<Expense> expenses = financeRepo.getExpensesByUser(userId);
        assertNotNull("Expenses list should not be null", expenses);
    }
}
```

# ExceptionTest.java:

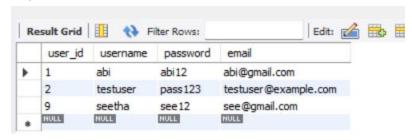
```
package Test;
import dao.FinanceRepositoryImpl;
import entity.Expense;
import org.junit.Test;
import java.util.Date;
import static org.junit.Assert.*;
public class ExceptionTest {
    FinanceRepositoryImpl financeRepo = new FinanceRepositoryImpl();
    @Test
    public void testExceptionForInvalidUserOnExpenseCreation() {
        Expense expense = new Expense(0, -1, 500.0, 1, new Date(), "Invalid User");
        boolean result = financeRepo.addExpense(expense);
        assertFalse(result);
    }
    @Test
    public void testExceptionForInvalidCategory() {
        Expense expense = new Expense(0, 1, 500.0, 999, new Date(), "Invalid Category");
        boolean result = financeRepo.addExpense(expense);
        assertFalse(result);
    }
}
```

#### **OUTPUT:**

## 1.Add User



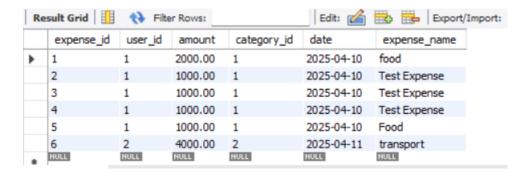
# **SQL OUTPUT:**



# 2.Add Expense:

```
Choose an option: 2
Enter User ID: 2
Enter Expense Name: transport
Enter Amount: 4000
Enter Category ID: 2
Expense added successfully!
```

# **SQL OUTPUT:**



## 3.Delete User:

```
Choose an option: 3
Enter User ID to Delete: 9
User deleted successfully!
```

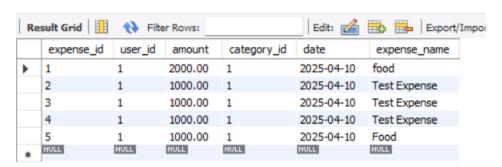
# **SQL OUTPUT:**



# **4.Delete Expense**

```
Choose an option: 4
Enter Expense ID to Delete: 6
Expense deleted successfully!
```

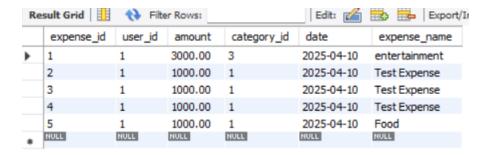
## **SQL OUTPUT:**



# **5.Update Expense:**

```
Choose an option: 5
Enter User ID: 1
Enter Expense ID to Update: 1
Enter New Expense Name: entertainment
Enter New Amount: 3000
Enter New Category ID: 3
Expense updated successfully!
```

# **SQL OUTPUT:**



# **6.View Expenses:**

```
Choose an option: 6
Enter User ID to View Expenses: 1
Expenses:
ID: 1, Name: entertainment, Amount: 3000.0
ID: 2, Name: Test Expense, Amount: 1000.0
ID: 3, Name: Test Expense, Amount: 1000.0
ID: 4, Name: Test Expense, Amount: 1000.0
ID: 5, Name: Food, Amount: 1000.0
```

# Finance Management System

- 1. Add User
- 2. Add Expense
- 3. Delete User
- 4. Delete Expense
- 5. Update Expense
- 6. View Expenses
- 7. Exit

Choose an option: 7

Process finished with exit code 0