CASE STUDY: FINANCE MANAGEMNT SYSTEM

SQL CODE AND OUTPUTS:(compiled at MySQL workbench)

CODE:

```sql
CREATE DATABASE FinanceDB;
USE FinanceDB;

CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL
);
```

OUTPUT:

✔	2	15:18:19	CREATE DATABASE FinanceDB	1 row(s) affected
✔	3	15:18:19	USE FinanceDB	0 row(s) affected
✔	4	15:18:19	CREATE TABLE Users (user_id INT PRIMARY KEY AUTO_INCREMENT, username VARCHAR(50) UN...	0 row(s) affected

CODE:

```sql
CREATE TABLE ExpenseCategories (
    category_id INT PRIMARY KEY AUTO_INCREMENT,
    category_name VARCHAR(50) UNIQUE NOT NULL
);
```

OUTPUT:

✔	5	15:18:19	CREATE TABLE ExpenseCategories (category_id INT PRIMARY KEY AUTO_INCREMENT, category_n...	0 row(s) affected

CODE:

```sql
CREATE TABLE Expenses (
    expense_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    category_id INT,
    date DATE NOT NULL,
    expense_name VARCHAR(100) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (category_id) REFERENCES ExpenseCategories(category_id) ON DELETE SET NULL
);
```

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	15:18:36	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:

```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 getUsername() { return username; }
    public String getPassword() { return password; }
    public String getEmail() { return email; }

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

```
package dao;

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();
    }

    // Check if User Exists
    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();
            return false;
        }
    }

    private boolean categoryExists(int categoryId) {
        String query = "SELECT 1 FROM ExpenseCategories WHERE category_id =
?";
        try (PreparedStatement pstmt = connection.prepareStatement(query)) {
            pstmt.setInt(1, categoryId);
            try (ResultSet rs = pstmt.executeQuery()) {
                return rs.next();
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    private boolean expenseExists(int expenseId) {
        String query = "SELECT 1 FROM Expenses WHERE expense_id = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(query)) {
            pstmt.setInt(1, expenseId);
            try (ResultSet rs = pstmt.executeQuery()) {
                return rs.next();
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean createUser(User user) {
        String query = "INSERT INTO Users (username, password, email) VALUES
(?, ?, ?)";
        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();
            return false;
        }
```

```java
    }

    public boolean addExpense(Expense expense) {
        if (!userExists(expense.getUserId())) {
            System.out.println("Error: User ID does not exist.");
            return false;
        }
        if (!categoryExists(expense.getCategoryId())) {
            System.out.println("Error: Category ID does not exist.");
            return false;
        }

        String query = "INSERT INTO Expenses (user_id, expense_name, amount,
category_id, date) VALUES (?, ?, ?, ?, ?)";
        try (PreparedStatement pstmt = connection.prepareStatement(query)) {
            pstmt.setInt(1, expense.getUserId());
            pstmt.setString(2, expense.getName());
            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();
            return false;
        }
    }

    public boolean deleteUser(int userId) {
        if (!userExists(userId)) {
            System.out.println("Error: User does not exist.");
            return false;
        }

        String query = "DELETE FROM Users WHERE user_id = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(query)) {
            pstmt.setInt(1, userId);
            return pstmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean deleteExpense(int expenseId) {
        if (!expenseExists(expenseId)) {
            System.out.println("Error: Expense does not exist.");
            return false;
        }

        String query = "DELETE FROM Expenses WHERE expense_id = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(query)) {
            pstmt.setInt(1, expenseId);
            return pstmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
```

```
        }
    }

    public boolean updateExpense(int userId, Expense expense) {
        if (!userExists(userId)) {
            System.out.println("Error: User ID does not exist.");
            return false;
        }
        if (!expenseExists(expense.getId())) {
            System.out.println("Error: Expense ID does not exist.");
            return false;
        }
        if (!categoryExists(expense.getCategoryId())) {
            System.out.println("Error: Category ID does not exist.");
            return false;
        }

        String query = "UPDATE Expenses SET expense_name = ?, amount = ?,
category_id = ? WHERE expense_id = ? AND user_id = ?";
        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();
            return false;
        }
    }


    public List<Expense> getExpensesByUser(int userId) {
        List<Expense> expenses = new ArrayList<>();
        String query = "SELECT * FROM Expenses WHERE user_id = ?";

        try (PreparedStatement pstmt = connection.prepareStatement(query)) {
            pstmt.setInt(1, userId);
            try (ResultSet rs = pstmt.executeQuery()) {
                while (rs.next()) {
                    Expense expense = new Expense(
                            rs.getInt("expense_id"),
                            rs.getInt("user_id"),
                            rs.getDouble("amount"),
                            rs.getInt("category_id"),
                            rs.getDate("date"),
                            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.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnUtil {

    private static final String fileName = "db.properties";

    public static Connection getConnection() {
        Connection con = null;
        String connString = null;

        try {
            connString = DBPropertyUtil.getConnectionString(fileName);
        } catch (IOException e) {
            System.out.println("Connection String Creation Failed");
            e.printStackTrace();
        }

        if (connString != null) {
            try {
                con = DriverManager.getConnection(connString);
            } catch (SQLException e) {
                System.out.println("Error While Establishing DB
Connection...");
                e.printStackTrace();
            }
        }

        return con;
    }
}
```

DBPropertyUtil.java:

```
package util;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class DBPropertyUtil {

    public static String getConnectionString(String fileName) throws
IOException {
        String connStr = null;
        Properties props = new Properties();

        FileInputStream fis = new FileInputStream("resources/" + fileName);
        props.load(fis);

        String user = props.getProperty("user");
        String password = props.getProperty("password");
        String protocol = props.getProperty("protocol");
        String system = props.getProperty("system");
        String port = props.getProperty("port");
        String database = props.getProperty("database");

        connStr = protocol + "//" + system + ":" + port + "/" + database +
                "?user=" + user + "&password=" + password;

        return connStr;
    }
}
```

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

```
package main;

import dao.FinanceRepositoryImpl;
import entity.Expense;
import entity.User;
import java.util.*;

public class FinanceApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        FinanceRepositoryImpl financeRepo = new FinanceRepositoryImpl();

        while (true) {
            System.out.println("\nFinance Management System");
            System.out.println("1. Add User");
            System.out.println("2. Add Expense");
            System.out.println("3. Delete User");
            System.out.println("4. Delete Expense");
            System.out.println("5. Update Expense");
            System.out.println("6. View Expenses");
            System.out.println("7. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();

            if (choice == 1) { // Add User
                System.out.print("Enter Username: ");
                String username = scanner.next();
                System.out.print("Enter Password: ");
                String password = scanner.next();
                System.out.print("Enter Email: ");
                String email = scanner.next();
                User user = new User(0, username, password, email);
                if (financeRepo.createUser(user)) {
                    System.out.println("User created successfully!");
```

```
                } else {
                    System.out.println("User creation failed!");
                }
            }
            else if (choice == 2) { // Add Expense
                System.out.print("Enter User ID: ");
                int userId = scanner.nextInt();
                System.out.print("Enter Expense Name: ");
                String name = scanner.next();
                System.out.print("Enter Amount: ");
                double amount = scanner.nextDouble();
                System.out.print("Enter Category ID: ");
                int categoryId = scanner.nextInt();
                Date date = new Date(); // Current date

                Expense expense = new Expense(0, userId, amount, categoryId,
date, name);
                if (financeRepo.addExpense(expense)) {
                    System.out.println("Expense added successfully!");
                } else {
                    System.out.println("Expense addition failed!");
                }
            }
            else if (choice == 3) { // Delete User
                System.out.print("Enter User ID to Delete: ");
                int userId = scanner.nextInt();
                if (financeRepo.deleteUser(userId)) {
                    System.out.println("User deleted successfully!");
                } else {
                    System.out.println("User deletion failed!");
                }
            }else if (choice == 4) { // Delete Expense
                System.out.print("Enter Expense ID to Delete: ");
                int expenseId = scanner.nextInt();

                if (financeRepo.deleteExpense(expenseId)) {
                    System.out.println("Expense deleted successfully!");
                } else {
                    System.out.println("Expense deletion failed!");
                }
            }else if (choice == 5) { // Update Expense
                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: ");
                String name = scanner.next();
                System.out.print("Enter New Amount: ");
                double amount = scanner.nextDouble();
                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)) {
                    System.out.println("Expense updated successfully!");
```

```
                } else {
                    System.out.println("Expense update failed!");
                }
            }

            else if (choice == 6) {
                System.out.print("Enter User ID to View Expenses: ");
                int userId = scanner.nextInt();
                List<Expense> expenses =
financeRepo.getExpensesByUser(userId);

                if (expenses.isEmpty()) {
                    System.out.println("No expenses found for this user.");
                } else {
                    System.out.println("Expenses:");
                    for (Expense exp : expenses) {
                        System.out.println("ID: " + exp.getId() + ", Name: "
+ exp.getName() + ", Amount: " + exp.getAmount());
                    }
                }
            }


            else if (choice == 7) { // Exit
                break;
            }
        }
        scanner.close();
    }
}
```

UNIT TESTING:

UserTest.java

```
package Test;

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:

```java
package Test;

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:

```java
package Test;

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:

```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

```
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: 1
Enter Username: seetha
Enter Password:
see12
Enter Email: see@gmail.com
User created successfully!
```

SQL OUTPUT:

Result Grid | Filter Rows: | Edit:

user_id	username	password	email
1	abi	abi12	abi@gmail.com
2	testuser	pass123	testuser@example.com
9	seetha	see12	see@gmail.com
NULL	NULL	NULL	NULL

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:

Result Grid | Filter Rows: | Edit: | Export/Import:

expense_id	user_id	amount	category_id	date	expense_name
1	1	2000.00	1	2025-04-10	food
2	1	1000.00	1	2025-04-10	Test Expense
3	1	1000.00	1	2025-04-10	Test Expense
4	1	1000.00	1	2025-04-10	Test Expense
5	1	1000.00	1	2025-04-10	Food
6	2	4000.00	2	2025-04-11	transport
NULL	NULL	NULL	NULL	NULL	NULL

3.Delete User:

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

SQL OUTPUT:

Result Grid | Filter Rows: | Edit:

user_id	username	password	email
1	abi	abi12	abi@gmail.com
2	testuser	pass123	testuser@example.com
NULL	NULL	NULL	NULL

4.Delete Expense

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

SQL OUTPUT:

Result Grid | Filter Rows: | Edit: | Export/Impo

expense_id	user_id	amount	category_id	date	expense_name
1	1	2000.00	1	2025-04-10	food
2	1	1000.00	1	2025-04-10	Test Expense
3	1	1000.00	1	2025-04-10	Test Expense
4	1	1000.00	1	2025-04-10	Test Expense
5	1	1000.00	1	2025-04-10	Food
NULL	NULL	NULL	NULL	NULL	NULL

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:

Result Grid | Filter Rows: | Edit: | Export/I

expense_id	user_id	amount	category_id	date	expense_name
1	1	3000.00	3	2025-04-10	entertainment
2	1	1000.00	1	2025-04-10	Test Expense
3	1	1000.00	1	2025-04-10	Test Expense
4	1	1000.00	1	2025-04-10	Test Expense
5	1	1000.00	1	2025-04-10	Food
NULL	NULL	NULL	NULL	NULL	NULL

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
```

7.EXIT

```
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
```