|  |
| --- |
| Hands-on Exercise Objective |
| After completing the hands-on exercises, you will be able to:   * Use JDBC for performing DML related operations in Java applications. |

|  |
| --- |
| Problem Statement: Mike is a software developer in a Marketing company. The business analysts have provided mike a requirement to develop a application which stores the department and employee information. The application should also provide a feature for users to query the records based on employee id.  The following are the information which needs to be stored   * Employees information such as employee id, employee name, salary, Employee address, contact number, and the department where he works. * Department information such as of the firm such as department id, department name, department head and number of employees in the department.   **Additional Requirement:**   1. Duplicate Employee data should not be stored in the system. 2. Duplicate department data should not be stored in the system. 3. Salary should be between 1000 and 30001.   **Problem # 1** **Creating Tables :** Create following tables using Oracle Client and DDL’s.   1. Create ***Department*** table    1. Department\_ID – Primary Key - Number    2. Department\_Name - Varchar    3. Department\_Head – Varchar    4. Department\_Description - Varchar 2. Create Employee table    1. Employee\_Id- Primarykey- Number    2. Employee\_Name- Varchar    3. Employee\_Address- Varchar    4. Employee\_Salary- Decimal Number    5. Employee\_Contact\_No- Number    6. Department\_Id- Number (Foreign Key) 3. Create a constraint on salary to ensure salary between 1000 and 1000000.   **Problem # 2 Loading tables using DML:**  Create a java program ***EmployeeUploader.java*** to insert data into the above mentioned tables. Develop the following methods,   * ***storeDepartmentDetails()*** – This method should connect to oracle database and load the department table with the department details. **Important:** All the data should come as input parameters to the method or as a value object. * ***storeEmployeeDetails()*** – This method should connect to oracle database and load the Employee table with the employee details. **Important:** All the data should come as input parameters to the method or as a value object. * Develop a main method which will invoke the above two methods by passing the respective values to be loaded int the database. The data can be hard coded in the main method, it can changed every time and run to load different data in the table.   **NOTE:** Use the data mentioned in [Appendix section](#Appendix) to load the tables  .  **Problem # 3 Retrieving tables using Select query:**  Add a method in ***EmployeeUploader.java*** to retrieve the data based on employee id. Develop a method ***retrieveEmployeeDetails*** whose parameter is employee id, this should query the database and pint the following details ins console. In case the employee record is not present display a message “***Employee Id not Present***”   * Employee Id * Employee Name * Employee Contact Number * Employee Address * Department Name * Department Head.   Invoke the above method from main method and retrieve the employee details of data loaded as part of ***problem# 2***.  **Problem # 4: Creating and Invoking Stored Procedure:**   * Develop a stored procedure which calculates PF for a given employee id as mentioned below and returns the PF amount.   + If salary is between 1000 and 10000 then     - PF = Salary \* 5%   + If salary is between 10000 and 100000 then     - PF = Salary \* 6%   + If salary is greater than 100000 then     - PF = Salary \* 7% * Add a method in ***EmployeeUploader.java*** which invokes the stored procedure mentioned above by passing employee id and display the PF amount returned in the console. Develop a method ***calculatePF*** the parameter is employee id.   **Method Output:**  Employee PF amount is: <PF Amount>  **Problem # 5: Exception Scenarios:**   * Use the ***storeEmployeeDetails()***  method to load duplicate Employee data, employee already exists as mentioned in [Appendix 2](#_Appendix_2:) and observe the error you are getting from oracle engine. * Use the ***storeDepartmentDetails()***  method to load inconsistent data, department does not exist, as mentioned in [Appendix 3](#_Appendix_3:) and observe the error you are getting from oracle engine. * Use the ***storeEmployeeDetails()***  method to load invalid Employee data, salary < 1000 as mentioned in Appendix 4 and observe the error you are getting from oracle engine. |

# Appendix 1:

## Department Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Department\_ID | Department\_Name | Department\_Head | Department\_Description |
| 1 | Accounts | Ramesh | Accounts Dept |
| 2 | Admin | Vijay | Admin Dept |
| 3 | Sales | Vinod | Sales Dept |
| 4 | HR | Mahesh | HR Dept |

## Employee Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Employee\_Id | Employee\_Name | Employee\_Salary | Employee\_Contact\_No | Employee\_Address | Department\_ID |
| 087 | Vikram | 12000 | 9878761212 | Address 1 | 2 |
| 110 | Ajay | 18000 | 9654376143 | Address 2 | 1 |
| 098 | Rajesh | 11000 | 9965322212 | Address 3 | 4 |
| 067 | Ram | 19000 | 8078343732 | Address 4 | 3 |
| 045 | Vimal | 27000 | 9932113221 | Address 5 | 4 |
| 987 | Kiran | 21000 | 7076337238 | Address 6 | 2 |

# Appendix 2:

## Employee Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Employee\_Id | Employee\_Name | Employee\_Salary | Employee\_Contact\_No | Employee\_Address | Department\_ID |
| **087** | Jack | 12000 | 994234651 | Address 1 | 2 |

# Appendix 3:

## Department Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Employee\_Id | Employee\_Name | Employee\_Salary | Employee\_Contact\_No | Employee\_Address | Department\_ID |
| 123 | Ron | 12000 | 972234651 | Address 1 | **99** |

# Appendix 4:

## Employee Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Employee\_Id | Employee\_Name | Employee\_Salary | Employee\_Contact\_No | Employee\_Address | Department\_ID |
| 124 | Jim | **500** | 923234651 | Address 1 | 2 |