Islamic University of Technology
Lab Task 08
CSE 4308 - DBMS Lab

Submitted To :

MD Bakhtiar Hasan
Ast. Professor, CSE Department
Islamic University of Technology

Zannatun Naim Sristy
Lecturer, CSE Department
Islamic University of Technology

Submitted By :

Sian Ashsad
ID : 200042151
Prog. : SWE
Dept. : CSE

Task 1 :

Working code :

```
import java.sql.*;

public class Task01 {
   static final String JDBC_DRIVER = "oracle.jdbc.driver.OracleDriver";
   static final String DB_URL= "jdbc:oracle:thin:@localhost:1521:xe";
   static final String USER="s200042151";
   static final String PASS="cse4308";
   public static void main (String args[])
   {
       Connection conn=null;
       Statement stmt=null;
       try
       {
           Class.forName(JDBC_DRIVER);
           System.out.println("Connecting to database");
           conn=DriverManager.getConnection(DB_URL, USER, PASS);
           System.out.println("Creating statement");
           stmt=conn.createStatement();
           String sql;
           sql="SELECT COUNT(t_id) as Transaction_45 FROM TRANSACTIONS WHERE a_id =
'45'";
           System.out.println("Executing the query: " + sql);
           ResultSet rs=stmt.executeQuery(sql);
           while(rs.next())
           {
               int transaction_45=rs.getInt("Transaction_45");

               System.out.print("Number of transactions done by user 45 : " +
transaction_45);

           }

           rs.close();
           stmt.close();
           conn.close();
           System.out.println("\nThank you for banking with us!");
       }
       catch(SQLException se)
       {
           se.printStackTrace();
       }
```

```
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Report :

Analysis : In this task, we need to count the total number of transactions conducted under the account 45 using Java code.

Explanation of solution : First we establish a connection with the oracle server through the try block of the java code. Connection is done using the DriverManager.getConnection method. Then a statement is created. Necessary query is passed through the statement.
This query counts the total number of transactions conducted under the account 45 using the count aggregate function.
The result of the query is saved in an integer and then printed out in the console.

Findings : Connection of oracle database using Java code was a new knowledge. It's relatively easy to implement.

Problems : Adhering to proper guidance, there weren't any problems completing this task.

Task 2 :

Working code :

```
import java.sql.*;

public class Task02 {
   static final String JDBC_DRIVER = "oracle.jdbc.driver.OracleDriver";
   static final String DB_URL= "jdbc:oracle:thin:@localhost:1521:xe";
   static final String USER="s200042151";
   static final String PASS="cse4308";
   public static void main (String args[])
   {
       Connection conn=null;
       Statement stmt=null;
       try
       {
           Class.forName(JDBC_DRIVER);
           System.out.println("Connecting to database");
           conn=DriverManager.getConnection(DB_URL, USER, PASS);
           System.out.println("Creating statement");
           stmt=conn.createStatement();
           String sql;
           sql="SELECT COUNT(t_id) as Totaldebit FROM TRANSACTIONS WHERE type =
'1'";
           System.out.println("Executing the query: " + sql);
           ResultSet rs=stmt.executeQuery(sql);
           while(rs.next())
           {
               int totalDebit=rs.getInt("Totaldebit");

               System.out.print("Total number of debits : " + totalDebit);

           }

           rs.close();
           stmt.close();
           conn.close();
           System.out.println("\nThank you for banking with us!");
       }
       catch(SQLException se)
       {
           se.printStackTrace();
       }
       catch(Exception e)
       {
```

```
            e.printStackTrace();
        }
    }
}
```

Report :

Analysis : In this task, we need to find the number of debits using Java code.

Explanation of solution : The connection procedure is similar to task 1, except the query is different. Here the count aggregate function is used to figure out the total number of debits through the attribute “TYPE”.
Afterwards, the value was taken into an integer and shown in the console.

Findings : The data that passes into the java code is controlled by the SQL query. Merely changing the query can result in a totally different scenario of data processing.

Task 3 :

Working code :

```
import java.sql.*;
public class Task03 {

   static final String JDBC_DRIVER = "oracle.jdbc.driver.OracleDriver";
   static final String DB_URL= "jdbc:oracle:thin:@localhost:1521:xe";
   static final String USER="s200042151";
   static final String PASS="cse4308";
   public static void main (String args[])
   {
       Connection conn=null;
       Statement stmt=null;
       try
       {
           Class.forName(JDBC_DRIVER);
           System.out.println("Connecting to database");
           conn=DriverManager.getConnection(DB_URL, USER, PASS);
           System.out.println("Creating statement");
           stmt=conn.createStatement();
           String sql;
           sql="SELECT a_id,amount,type,EXTRACT( YEAR FROM DTM) as year FROM
TRANSACTIONS  GROUP BY a_id,amount,type,EXTRACT( YEAR FROM DTM) HAVING EXTRACT(
YEAR FROM DTM) = '2020'";
           System.out.println("Executing the query: " + sql);
           System.out.print("Transactions that occurred in 2020 : \n");
           ResultSet rs=stmt.executeQuery(sql);
           while(rs.next())
           {

               int account=rs.getInt("a_id");
               int Amount=rs.getInt("amount");
               String type=rs.getString("type");

               System.out.print(Amount + " taka has been");
               if(type.charAt(0)=='0')
                   System.out.print(" deposited to");
               else
                   System.out.print(" taken out from ");
               System.out.println(" account " + account);
```

```
            }

            rs.close();
            stmt.close();
            conn.close();
            System.out.println("\nThank you for banking with us!");
        }
        catch(SQLException se)
        {
            se.printStackTrace();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

}
```

Report :

Analysis : In this task we have to list the transactions that occurred in the year 2020.

Explanation of solution : The connection to Oracle database is similar to that of task 1 and 2. The query returns the account id, type, and year of the transactions made in the year 2020 from the transactions table. The aforementioned attributes are taken in variables of suitable types. Afterwards the transactions are shown in the console.

Findings : Extracting the year from a DATE attribute requires the EXTRACT aggregate function. Therefore group by is needed for additional attributes alongside extract function. The “year” operator is used inside the extract clause to fetch the year.

Problems : The only problem faced here was to find out how to extract the year from the DATE attribute in the transactions table.

Task 4 :

Working code :

```
import java.sql.*;
import java.util.*;
public class Task04 {
   static final String JDBC_DRIVER = "oracle.jdbc.driver.OracleDriver";
   static final String DB_URL= "jdbc:oracle:thin:@localhost:1521:xe";
   static final String USER="s200042151";
   static final String PASS="cse4308";

   public static void main (String args[])
   {
       Connection conn=null;
       Statement stmt=null;
       try
       {
           Class.forName(JDBC_DRIVER);
           System.out.println("Connecting to database");
           conn=DriverManager.getConnection(DB_URL, USER, PASS);
           System.out.println("Creating statement");
           stmt=conn.createStatement();
           String sql;
           sql="SELECT A_ID, AMOUNT,TYPE FROM TRANSACTIONS";
           System.out.println("Executing the query: " + sql);
           ResultSet rs=stmt.executeQuery(sql);
           int CIP=0,VIP=0,OP=0;

           HashMap<Integer,Long>Transactions=new HashMap<Integer, Long>();

           HashMap<Integer,Long>account_balance=new HashMap<Integer, Long>();
           while(rs.next())
           {
               int ID=rs.getInt("A_ID");
               Long amount=rs.getLong("AMOUNT");
               String type=rs.getString("TYPE");

               if(Transactions.get(ID)==null){
                   Transactions.put(ID,amount);
               }
               else{
                   Transactions.replace(ID,Transactions.get(ID)+amount);
               }
               if(account_balance.get(ID)==null){
```

```
                    if(type.equals("0")){
                        account_balance.put(ID,-amount);
                    }
                    else{
                        account_balance.put(ID,amount);
                    }
                }
                else{
                    if(type.equals("0")){
                        account_balance.replace(ID,account_balance.get(ID)-amount);
                    }
                    else{
                        account_balance.replace(ID,account_balance.get(ID)+amount);
                    }

                }
            }
            int exception=0;
            for(int i=1;i<=150;i++){
                if(Transactions.get(i)>5000000 && account_balance.get(i)>1000000){
                    CIP++;
                }
                else if(Transactions.get(i)>2500000 && Transactions.get(i)<4500000
&& account_balance.get(i)>500000 && account_balance.get(i)<900000){
                    VIP++;
                }
                else if(Transactions.get(i)<1000000 &&
account_balance.get(i)<100000){
                    OP++;
                }
                else{
                    exception++;
                }
            }
            System.out.printf("CIP: " + CIP + ", VIP: " + VIP + " , OP: " + OP + ",
No Category: " + exception + "\n");


            rs.close();
            stmt.close();
            conn.close();
            System.out.println("\nThank you for banking with us!");
        }
        catch(SQLException se)
        {
            se.printStackTrace();
        }
```

```
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Report :

Analysis : In this task, we need to count the number of CIP, VIP and OPs from the transactions table. Additionally, we have to show the number of people that do not fall in any of the categories.

Explanation of solution : The solution focused on the java code to solve the problem. The query is the same as the query provided in the solution.java file. Two hashmaps are used in order to keep track of the transactions and balance of each account. The hashmaps have two attributes : Integer for storing the account id and Long for storing the transactions/balance.
The account id is stored in an integer while the amount is stored in a long variable. Type attribute is stored in a string variable.
The conditions imply that, in case of transactions, the amount will keep on increasing per transaction under the same account holder and stored in the Transactions hashmap.
On the other hand, the balance of an account holder will decrease if there's a credit in their transaction (denoted when the type attribute is equal to 0) and increase if there's a debit. The balance will be stored in the account_balance hashmap similar to Transactions hashmap.
Afterwards, both the Transactions and account_balance hashmaps are iterated to find out the number of CIP, VIP, OPs and accounts that are not under any categories. Finally the result is shown in the console.

Findings : Functionalities of map in Java was a new learning. Hashmap in java is similar to Map in Cpp which made completing this task easier as I am familiar with the concept.

Problems : At first I thought of completing the task using SQL query. But doing so would've made things very complicated as setting appropriate variables in a query is quite difficult. Solving this problem in Java code instead proved to be much easier.

However it took me some time to reach this conclusion. There were no further problems other than implementing the provided logic in the Java code.