**Contents**

**Course Code: 15IT302J**

**Course Title: Database Management System**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Exp. No.** | **Date** | **Exp. No.** | **Title** | **Sign** |
| 1 | 14 July | 1 | DDL Commands |  |
| 2 | 21 July | 2 | Working with Data Manipulation Commands |  |
| 3 | 28 July | 3 | SELECT Commands |  |
| 4 | 04 Aug | 4 | View Commands |  |
| 5 | 11 Aug | 5 | Working with Constraints |  |
| 6 | 22 Aug | 6 | SQL Functions |  |
| 7 | 30 Aug | 7 | Working with various joints |  |
| 8 | 6 Sept | 8 | Nested Queries with single and multiple tables |  |
| 9 | 13 Sept – 3 Oct | 9 | Application of PL/SQL Programs |  |
| 10 | 10 Oct | 10 | Mini Project |  |

|  |  |
| --- | --- |
| **EX.NO:1** | **CREATING DATABASE TABLE** |

**AIM**

To create a DDL to perform creation of table, alter, modify and drop column.

**DDL COMMANDS**

1. The Create Table Command: - it defines each column of the table uniquely. Each column has minimum of three attributes, a name , data type and size.

**Syntax:**

Create table <table name> (<col1> <datatype>(<size>),<col2> <datatype><size>)); Ex:create table emp(empno number(4) primary key, ename char(10));

2.Modifying the structure of tables. a)

Add new columns

**Syntax:**

Alter table <tablename> add(<new col><datatype(size),<new col>datatype(size)); Ex:alter table emp add(sal number(7,2));

1. Dropping a column from a table.

**Syntax:**

Alter table <tablename> drop column <col>;

Ex:alter table emp drop column sal;

1. Modifying existing columns.

**Syntax:**

Alter table <tablename> modify(<col><newdatatype>(<newsize>));

Ex:alter table emp modify(ename varchar2(15));

1. Renaming the tables

**Syntax:**

Rename <oldtable> to <new table>;

Ex:rename emp to emp1;

1. truncating the tables.

**Syntax:**

Truncate table <tablename>;

Ex:trunc table emp1;

1. Destroying tables.

**Syntax:**

Drop table <tablename>;

Ex:drop table emp;

**CREATION OF TABLE:**

**SYNTAX:**

create table<tablename>(column1 datatype,column2 datatype...);

**EXAMPLE:**

SQL>create table std(sno number(5),sname varchar(20),age number(5),sdob date,sm1 number(4,2),sm2 number(4,2),sm3 number(4,4)); Table created.

SQL>insert into std values(101,‟AAA‟,16,‟03-jul-88‟,80,90,98); 1 row created.

SQL>insert into std values(102,‟BBB‟,18,‟04-aug-89‟,88,98,90); 1 row created.

|  |  |
| --- | --- |
| **OUTPUT:** |  |
| Select \* from std; |  |
| SNO SNAME AGE SDOBSM1 | SM2 SM3 |
| 101 AAA 16 03-jul-88 80 | 90 98 |
| 102 BBB 18 04-aug-89 88 | 98 90 |

**ALTER TABLE WITH ADD:**

SQL>create table student(id number(5),name varchar(10),game varchar(20));

Table created.

SQL>insert into student values(1,‟mercy‟,‟cricket‟); 1 row created.

**SYNTAX:**

alter table<tablename>add(col1 datatype,col2 datatype..);

EXAMPLE:

SQL>alter table student add(age number(4));

SQL>insert into student values(2,‟sharmi‟,‟tennis‟,19);

**OUTPUT:**

ALTER: select \* from student;

ID NAME GAME

1 Mercy Cricket

ADD: select \* from student;

ID NAME GAME AGE

1. Mercy cricket

1. Sharmi Tennis 19

**ALTER TABLE WITH MODIFY:**

**SYNTAX:**

Alter table<tablename>modify(col1 datatype,col2 datatype..);

**EXAMPLE:**

SQL>alter table student modify(id number(6),game varchar(25));

**OUTPUT:**

MODIFY

desc student;

NAME NULL? TYPE

Id Number(6)

Name Varchar(20)

Game Varchar(25)

Age Number(4)

**DROP:**

**SYNTAX:** drop table<tablename>;

EXAMPLE:

SQL>drop table student;

SQL>Table dropped.

**TRUNCATE TABLE**

**SYNTAX:** TRUNCATE TABLE <TABLE NAME>;

Example: Truncate table stud;

**DESC**

Example: desc emp;

Name Null? Type

--------------------------------- --------

EmpNo NOT NULL number(5)

EName VarChar(15)

Job NOT NULL Char(10)

DeptNo NOT NULL number(3)

PHONE\_NO number (10)

**Queries:**

Q1. Create a table called EMP with the following structure.

Name Type

---------- ----------------------

EMPNO NUMBER(6)

ENAME VARCHAR2(20)

JOB VARCHAR2(10)

DEPTNO NUMBER(3)

SAL NUMBER(7,2)

Allow NULL for all columns except ename and job.

**Solution:**

1. Understand create table syntax.
2. Use the create table syntax to create the said tables.
3. Create primary key constraint for each table as understand from logical table structure.

Ans:

1. SQL> create table emp(empno number(6),ename varchar2(20)not null,job varchar2(10) not null, deptno number(3),sal number(7,2));
2. Table created.

Q2: Add a column experience to the emp table. experience numeric null allowed.

**Solution:**

1. Learn alter table syntax.
2. Define the new column and its data type.
3. Use the alter table syntax.

Ans: SQL> alter table emp add(experience number(2)); Table altered.

Q3: Modify the column width of the job field of emp table. **Solution:**

1. Use the alter table syntax.
2. Modify the column width and its data type. Ans: SQL> alter table emp modify(job varchar2(12)); Table altered.

SQL> alter table emp modify(job varchar(13)); Table altered.

Q4: Create dept table with the following structure.

Name Type

------------ ---------------------

DEPTNO NUMBER(2)

DNAME VARCHAR2(10)

LOC VARCHAR2(10)

Deptno as the primarykey

**Solution:**

1. Understand create table syntax.
2. Decide the name of the table.
3. Decide the name of each column and its data type.
4. Use the create table syntax to create the said tables.
5. Create primary key constraint for each table as understand from logical table structure. Ans:

SQL> create table dept(deptno number(2) primary key,dname varchar2(10),loc varchar2(10)); Table created.

Q5: create the emp1 table with ename and empno, add constraints to check the empno value while entering (i.e) empno > 100.

**Solution:**

1. Learn alter table syntax.
2. Define the new constraint [columns name type]
3. Use the alter table syntax for adding constraints.

Ans:

SQL> create table emp1(ename varchar2(10),empno number(6) constraint check(empno>100)); Table created.

Q6: drop a column experience to the emp table.

**Solution:**

1. Learn alter table syntax. Use the alter table syntax to drop the column.

Ans:

SQL> alter table emp drop column experience; Table altered.

Q7: Truncate the emp table and drop the dept table

**Solution:**

1. Learn drop, truncate table syntax.

Ans: SQL> truncate table emp; Table truncated.

**RESULT:**

Thus the DDL commands have been executed successfully.

|  |  |
| --- | --- |
| **EX.NO:2** | **WORKING WITH DATA MANIPULATION COMMANDS** |

**AIM**

To study the various DML commands and implement them on the database.

**DML COMMANDS**

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are Insert, Select, Update, Delete.

**Insert Command** This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.

**Select Commands** It is used to retrieve information from the table. It is generally referred to as querying the table. We can either display all columns in a table or only specify column from the table.

**Update Command** It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.

**Delete command** After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause.

**SYNTAX:**

INSERT INTO table\_name*(*column1*,* column2*,* column3*, ...)*VALUES *(*value1*,* value2*,* value3*, ...);*

INSERT INTO table\_nameVALUES *(*value1*,* value2*,* value3*, ...);*

Q1: Insert a single record into dept table.

Ans: SQL> insert into dept values (1,'IT','Tholudur');

1 row created.

Q2: Insert more than a record into emp table using a single insert command.

Ans: SQL> insert into emp values(&empno,'&ename','&job',&deptno,&sal);

Enter value for empno: 1

Enter value for ename: Mathi

Enter value for job: AP

Enter value for deptno: 1

Enter value for sal: 10000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)

new 1: insert into emp values(1,'Mathi','AP',1,10000)

1 row created.

SQL> / Enter value for empno: 2

Enter value for ename: Arjun

Enter value for job: ASP

Enter value for deptno: 2

Enter value for sal: 12000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)

new 1: insert into emp values(2,'Arjun','ASP',2,12000)

1 row created.

SQL> / Enter value for empno: 3

Enter value for ename: Gugan

Enter value for job: ASP

Enter value for deptno: 1

Enter value for sal: 12000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)

new 1: insert into emp values(3,'Gugan','ASP',1,12000)

1 row created.

Q3: Update the emp table to set the salary of all employees to Rs15000/- who are working as ASP

Ans: SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

---------- -------------------- ------------- ---------- ----------

1. Mathi AP 1 10000
2. Arjun ASP 2 12000
3. Gugan ASP 1 12000

SQL> update emp set sal=15000 where job='ASP'; 2 rows updated.

SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

---------- -------------------- ------------- ---------- ----------

1. Mathi AP 1 10000

1. Arjun ASP 2 15000

1. Gugan ASP 1 15000

Q4: Create a pseudo table employee with the same structure as the table emp and insert rows into the table using select clauses.

Ans: SQL> create table employee as select \* from emp;

Table created.

SQL> desc employee;

Name Null? Type

----------------------------------------- -------- ----------------------------

EMPNO NUMBER(6)

ENAME NOT NULL VARCHAR2(20)

JOB NOT NULL VARCHAR2(13)

DEPTNO NUMBER(3)

SAL NUMBER(7,2)

Q5: select employee name, job from the emp table

Ans: SQL> select ename, job from emp;

ENAME JOB

-------------------- -------------

Mathi AP

Arjun ASP

Gugan ASP

Karthik Prof Akalya AP suresh lect

6 rows selected.

**RESULT:**

Thus the DML commands have been executed successfully.

|  |  |  |
| --- | --- | --- |
| **EX.NO:3** |  | **BASIC SELECT STATEMENTS** |

**AIM**

To study the various Basic Select statement on the database.

The SELECT statement is used to select data from a database.

**SYNTAX**:

SELECT column1*,* column2, ...FROM table\_name*;*

Q1: Delete only those who are working as lecturer

Ans: SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

---------- -------------------- ------------- ---------- ---------- 1 Mathi AP 1 10000

1. Arjun ASP 2 15000
2. Gugan ASP 1 15000
3. Karthik Prof 2 30000
4. Akalya AP 1 10000 6 suresh lect 1 8000 6 rows selected.

SQL> delete from emp where job='lect'; 1 row deleted.

SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

- --------- -------------------- ------------- ---------- ---------- 1 Mathi AP 1 10000

1. Arjun ASP 2 15000
2. Gugan ASP 1 15000
3. Karthik Prof 2 30000
4. Akalya AP 1 10000

Q2: List the records in the emp table orderby salary in ascending order.

Ans: SQL> select \* from emp order by sal;

EMPNO ENAME JOB DEPTNO SAL

---------- -------------------- ------------- ---------- ---------- 1 Mathi AP 1 10000

5 Akalya AP 1 10000

1. Arjun ASP 2 15000
2. Gugan ASP 1 15000
3. Karthik Prof 2 30000

Q3: List the records in the emp table orderby salary in descending order.

Ans: SQL> select \* from emp order by sal desc;

EMPNO ENAME JOB DEPTNO SAL

---------- -------------------- ------------- ---------- ---------- 4 Karthik Prof 2 30000

1. Arjun ASP 2 15000
2. Gugan ASP 1 15000

1 Mathi AP 1 10000

5 Akalya AP 1 10000

Q4: Display only those employees whose deptno is 30. Solution: Use SELECT FROM WHERE syntax.

Ans: SQL> select \* from emp where deptno=1;

EMPNO ENAME JOB DEPTNO SAL

---------- -------------------- ------------- ---------- ---------- 1 Mathi AP 1 10000

3 Gugan ASP 1 15000

5 Akalya AP 1 10000

Q5: Display deptno from the table employee avoiding the duplicated values.

Solution:

1. . Use SELECT FROM syntax.
2. .Select should include distinct clause for the deptno.

Ans: SQL> select distinct deptno from emp;

DEPTNO

----------

1

2

1. To select all stores with sales above $1,000 in Table Store\_Information

SQL>SELECT Store\_Name FROM Store\_Information WHERE Sales > 1000;

Store\_Name

Los Angeles

1. To select all distinct stores in Table Store\_Information, we key in, SQL>SELECT DISTINCT Store\_Name FROM Store\_Information;

Result:

Store\_Name

Los Angeles

San Diego

Boston

1. If we want to select all stores with sales greater than $1,000 or all stores with sales less than $500 but greater than $275 in Table Store\_Information, we key in,

SQL>SELECT Store\_Name FROM Store\_Information WHERE Sales > 1000 OR (Sales < 500 AND Sales > 275);

Result:

Store\_Name

Los Angeles

San Francisco

1. To select all records for the Los Angeles and the San Diego stores in Table Store\_Information, we key in,

SQL>SELECT \* FROM Store\_Information WHERE Store\_Name IN ('Los Angeles', 'San Diego');

Result:

|  |  |  |
| --- | --- | --- |
| Store\_Name | Sales | Txn\_Date |
| Los Angeles | 1500 | Jan-05-1999 |
| San Diego | 250 | Jan-07-1999 |

1. To select view all sales information between January 6, 1999, and January 10, 1999, we key in,

SQL>SELECT \* FROM Store\_Information WHERE Txn\_Date BETWEEN 'Jan-06-1999' AND

'Jan-10-1999';

Note that date may be stored in different formats in different databases. This tutorial simply choose one of the formats.

Result:

Store\_Name Sales Txn\_Date

|  |  |  |
| --- | --- | --- |
| San Diego | 250 | Jan-07-1999 |
| San Francisco | 300 | Jan-08-1999 |
| Boston | 700 | Jan-08-1999 |

1. We want to find all stores whose name contains 'AN'. To do so, we key in,

SQL>SELECT \* FROM Store\_Information WHERE Store\_Name LIKE '%AN%';

Result:

|  |  |  |
| --- | --- | --- |
| Store\_Name | Sales | Txn\_Date |
| LOS ANGELES | 1500 | Jan-05-1999 |
| SAN DIEGO | 250 | Jan-07-1999 |
| SAN FRANCISCO | 300 | Jan-08-1999 |

1. To list the contents of Table Store\_Information by Sales in descending order, we key in,

SQL> SELECT Store\_Name, Sales, Txn\_Date FROM Store\_Information ORDER BY Sales DESC;

Result:

|  |  |  |
| --- | --- | --- |
| Store\_Name | Sales | Txn\_Date |
| Los Angeles | 1500 | Jan-05-1999 |
| Boston | 700 | Jan-08-1999 |
| San Francisco | 300 | Jan-08-1999 |
| San Diego | 250 | Jan-07-1999 |

1. To see only the stores with sales over $1,500, we would type,

SELECT Store\_Name, SUM(Sales)

FROM Store\_Information GROUP BY Store\_Name HAVING SUM(Sales) > 1500;

Result:

|  |  |
| --- | --- |
| Store\_Name | SUM(Sales) |
| Los Angeles | 1800 |

**RESULT:**

Thus the Basic select commands have been executed successfully.

|  |  |  |  |
| --- | --- | --- | --- |
| **EX.NO.4** |  |  | **VIEWS** |

**AIM**

To study the various SQL view operations on the database.

1. CREATE VIEW command is used to define a view.
2. INSERT command is used to insert a new row into the view.
3. DELETE command is used to delete a row from the view.
4. UPDATE command is used to change a value in a tuple without changing all values in the tuple.
5. DROP command is used to drop the view table

Commands Execution

Creation Of Table

-------------------------------- Sql> Create Table Employee (

Employee\_Namevarchar2(10),

Employee\_Nonumber(8),

Dept\_Name Varchar2(10),

Dept\_No Number (5),Date\_Of\_Join Date); Table Created.

Table Description

-------------------------------

Sql> Desc Employee;

Name Null? Type

------------------------------- -------- ------------------------

Employee\_Name Varchar2(10)

Employee\_No Number(8)

Dept\_Name Varchar2(10)

Dept\_No Number(5)

Date\_Of\_Join Date

Suntax For Creation Of View

--------------------------------------------------

Sql> Create <View> <View Name> As Select

<Column\_Name\_1>, <Column\_Name\_2> From <Table Name>;

Creation Of View

------------------------------

Sql> Create View Empview As Select

Employee\_Name,Employee\_No,Dept\_Name,Dept\_No,Date\_Of\_Join From Employee; View Created.

Description Of View

--------------------------------

Sql> Desc Empview;

Name Null? Type

----------------------------------------- -------- ----------------------------

Employee\_Name Varchar2(10)

Employee\_No Number(8)

Dept\_Name Varchar2(10)

Dept\_No Number(5)

Display View:

----------------------

Sql> Select \* From Empview;

Employee\_N Employee\_No Dept\_Name Dept\_No

---------- ----------- ---------- ----------

Ravi 124 Ece 89

Vijay 345 Cse 21

Raj 98 It 22

Giri 100 Cse 67

Insertion Into View ---------------------------------- Insert Statement: Syntax:

Sql> Insert Into <View\_Name> (Column Name1,………)

Values(Value1,….);

Sql> Insert Into Empview Values ('Sri', 120,'Cse', 67,'16-Nov-1981'); 1 Row Created.

Sql> Select \* From Empview;

Employee\_N Employee\_No Dept\_Name Dept\_No

---------- ----------- ---------- ----------

Ravi 124 Ece 89

Vijay 345 Cse 21

Raj 98 It 22

Giri 100 Cse 67

Sri 120 Cse 67

Sql> Select \* From Employee;

Employee\_N Employee\_No Dept\_Name Dept\_No Date\_Of\_J

---------- ----------- ---------- ---------- ---------

Ravi 124 Ece 89 15-Jun-05

Vijay 345 Cse 21 21-Jun-06

Raj 98 It 22 30-Sep-06

Giri 100 Cse 67 14-Nov-81 Sri 120 Cse 67 16-Nov-81 Deletion Of View:

Delete Statement: Syntax:

Sql> Delete <View\_Nmae>Where <Column Nmae> =’Value’; Sql> Delete From Empview Where Employee\_Name='Sri'; 1 Row Deleted.

Sql> Select \* From Empview;

Employee\_N Employee\_No Dept\_Name Dept\_No

---------- ----------- ---------- ----------

Ravi 124 Ece 89

Vijay 345 Cse 21

Raj 98 It 22 Giri 100 Cse 67 Update Statement:

Syntax:

Aql>Update <View\_Name> Set< Column Name> = <Column Name>

+<View> Where <Columnname>=Value;

Sql> Update Empkaviview Set Employee\_Name='Kavi' Where Employee\_Name='Ravi'; 1 Row Updated.

Sql> Select \* From Empkaviview;

Employee\_N Employee\_No Dept\_Name Dept\_No

---------- ----------- ---------- ----------

Kavi 124 Ece 89

Vijay 345 Cse 21

Raj 98 It 22

Giri 100 Cse 67 Drop A View:

Syntax:

Sql> Drop View <View\_Name>

Example

Sql>Drop View Empview;

View Droped

Create A View With Selected Fields:

Syntax:

Sql>Create [Or Replace] View <View Name>As Select <Column

Name1>…..From <Table Anme>; Example-2:

Sql> Create Or Replace View Empl\_View1 As Select Empno, Ename, Salary From Empl;

Sql> Select \* From Empl\_View1; Example-3:

Sql> Create Or Replace View Empl\_View2 As Select \* From Empl Where

Deptno=10;

Sql> Select \* From Empl\_View2; Note: Replace Is The Keyboard To Avoid The Error “Ora\_0095:Name Is Already Used By An Existing

Abject”.

Changing The Column(S) Name M The View During As Select Statement:

Type-1:

Sql> Create Or Replace View Emp\_Totsal(Eid,Name,Sal) As Select Empno,Ename,Salary From Empl; View Created.

Empno Ename Salary

---------- -------------------- ---------- ----------

7369 Smith 1000

7499 Mark 1050

7565 Will 1500

7678 John 1800

7578 Tom 1500

7548 Turner 1500 6 Rows Selected.

View Created.

Empno Ename Salary Mgrno Deptno

---------- -------------------- ---------- ---------- ----------------------------

7578 Tom 1500 7298 10

7548 Turner 1500 7298 10 View Created.

Sql> Select \* From Emp\_Totsal; Type-2:

Sql> Create Or Replace View Emp\_Totsal As Select Empno "Eid",Ename

"Name",Salary "Sal" From Empl; Sql> Select \* From Emp\_Totsal; Example For Join View:

Type-3:

Sql> Create Or Replace View Dept\_Emp As Select A.Empno "Eid",A.Ename

"Empname",A.Deptno "Dno",B.Dnam

E "D\_Name",B.Loc "D\_Loc" From Empl A,Depmt B Where

A.Deptno=B.Deptno;

Sql> Select \* From Dept\_Emp;

Eid Name Sal

---------- -------------------- ---------- ----------

7369 Smith 1000

7499 Mark 1050

7565 Will 1500

7678 John 1800

7578 Tom 1500

7548 Turner 1500 6 Rows Selected. View Created.

Eid Name Sal

---------- -------------------- ---------- ----------

7369 Smith 1000

7499 Mark 1050

7565 Will 1500

7678 John 1800

7578 Tom 1500

7548 Turner 1500 6 Rows Selected.

View Created.

Eid Empname Dno D\_Name D\_Loc

---------- -------------------- ---------- ---------- ------------------------

7578 Tom 10 Account New York

7548 Turner 10 Account New York

7369 Smith 20 Sales Chicago

7678 John 20 Sales Chicago

7499 Mark 30 Research Zurich

7565 WILL 30 RESEARCH ZURICH

**RESULT:**

Thus the SQL views have been executed successfully.

|  |  |  |
| --- | --- | --- |
| **EX.NO:5** |  | **INTEGRITY AND CONSTRAINTS** |

**AIM**

To study the various Integrity and Constraints on the database.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

* [**NOT NULL**](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [**UNIQUE**](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [**PRIMARY KEY**](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [**FOREIGN KEY**](https://www.w3schools.com/sql/sql_foreignkey.asp) - Uniquely identifies a row/record in another table
* [**CHECK**](https://www.w3schools.com/sql/sql_check.asp) - Ensures that all values in a column satisfies a specific condition
* [**DEFAULT**](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column when no value is specified  [**INDEX**](https://www.w3schools.com/sql/sql_create_index.asp) - Use to create and retrieve data from the database very quickly

Q1. Sql>Create Table Employee(Empno Number(4)

**Primary Key,**

Ename Varchar2(10),

Job Varchar2(6),

Sal Number(5),

Deptno Number(7));

Column Level Constraints Using Primary Key With Naming Convention

Q2. Sql>Create Table Employee (Empno Number(4)

Constraint **Emp\_Empno\_Pk Primary Key**,

Ename Varchar2(10),

Job Varchar2(6),

Sal Number(5),

Deptno Number(7));

Q3: Write a query to create foreign key constraints with Table level with alter command.

Sql>Create Table Dept (Deptno Number (2) **Primary Key**,

Dname Varchar2 (20),

Location Varchar2 (15));

Sql>Create Table Emp4 (Empno Number (3),

Deptno Number (2) **References Dept (Deptno)**,

Design Varchar2 (10));

Column Level Foreign Key Constraint With Naming Conversions:

Q4. Sql>Create Table Dept

(Deptno Number (2) **Primary Key,**

Dname Varchar2 (20),

Location Varchar2 (15));

Sql>Create Table Emp5

(Empno Number (3),

Deptno Number (2),

Design Varchar2 (10) **Constraint Enp2\_Deptno\_Fk Foreign Key**

(Dept No) Referencesdept (Deptno));

Q5. Write a query to create Check constraints with table level using alter command

Sql>Create Table Emp7

(Empno Number (3),

Ename Varchar2 (20),

Design Varchar2 (15),

Sal Number (5) **Constraint Emp7\_Sal\_Ck Check (Sal>500 And Sal <10001**

Dweptno Number (2));

Q6. Write a query to create unique constraints with column level

Sql>Create Table Emp10 (Empno Number (3),

Ename Varchar2 (20),

Desgin Varchar2 (15) **Constraint Emp10\_Design\_Uk Unique**,

Sal Number (5));

Q7. Write a query to create Not Null constraints with column level

Sql>Create Table Emp13

(Empno Number (4),

Ename Varchar2 (20) **Constraint Emp13\_Ename\_Nn Not Null,**

Design Varchar2 (20),

Sal Number (3));

Q8. Write a query to create Null constraints with column level.

Sql>Create Table Emp13

(Empno Number (4),

Ename Varchar2 (20) **Constraint Emp13\_Ename\_Nn Null,**

Design Varchar2 (20),

Sal Number (3));

Q9. Write a query to disable the constraints

Sql>Alter Table Emp13 Disable Constraint Emp13\_Ename\_Nn Null;

Q10. Write a query to enable the constraints

Sql>Alter Table Emp13 Enable Constraint Emp13\_Ename\_Nn Null;

**RESULT:**

Thus the Integrity and Constraints have been executed successfully .

|  |  |  |
| --- | --- | --- |
| **EX.NO:6** |  | **SQL FUNCTIONS** |

**AIM**

To study the various SQL Functions operations on the database.

# CHARACTER/STRING FUNCTION:

**Upper**: Converts the text to upper-case

1. SQL> select upper('welcome') from dual;

-----------

WELCOME

2.SQL> select upper('hai') from dual;

---

HAI

**Lower**: Converts the text to lower-case

3.SQL> select lower('HAI') from dual;

LOW

--- hai

4. SQL> select initcap(„hello world') from dual;

INITCAP('Hello

--------------

Hello World

**Ltrm:**The Oracle LTRIM function removes all specified characters from the left-hand side of a string.

5.SQL> select ltrim(' hai') from dual;

LTR

--- hai

**Rtrim:**The Oracle RTRIM function removes all specified characters from the right-hand side of a string

6.SQL> select rtrim('hai')from dual;

RTR

--- hai

7 .SQL> select rtrim(' hai ')from dual;

RTRIM('

------- hai SQL> select concat('SRM',' university')from dual;

------------------------

SRM university

**Length:**The Oracle LENGTH function returns the length of the specified string.

8.SQL> select length('SRM‟)from dual;

LENGTH('SRM')

----------------------

12

**Replace:**Replace a sequence of characters in a string with another set of characters.

9. SQL> select replace('SRM university', 'SRM','Anna')from dual;

----------------

Anna university

**Substr**:Extract a substring from a string.

10.SQL> select substr('SRM', 7,6)from dual;

SUBSTR

------

lingam

**Rpad:**The RPAD() function returns a string that is right-padded with a specified string to a certain length.

11.SQL> select rpad('hai',3,'\*')from dual;

RPAD('

------

hai\*\*\*

**Lpad**:The Oracle LPAD function pads the left-side of a string with a specific set of characters

12.SQL> select lpad('hai',3,'\*')from dual;

LPAD('

------

\*\*\*hai

13.SQL> select replace('Dany','y','ie')from dual;

REPLACE

-------

Danie

14.SQL> select translate('cold','ld','ol')from dual;

TRANSL

------ cool

**DATE & TIME FUNCTION**

**SYSDATE** :The Oracle SYSDATE function returns the current system date and time on your local database.

15.SQL> select sysdate from dual;

SYSDATE

---------

07-APR-10

16.SQL> select round(sysdate)from dual;

ROUND(SYS

---------

07-APR-10

17.SQL> select add\_months(sysdate,3)from dual;

ADD\_MONTH

---------

07-JUL-10

18.SQL> select last\_day(sysdate)from dual;

LAST\_DAY(

---------

30-APR-10

19.SQL> select sysdate+20 from dual;

SYSDATE+2

---------

27-APR-10

20. SQL> select next\_day(sysdate,'tuesday')from dual;

NEXT\_DAY(

---------

13-APR-10

# NUMERIC FUNCTION

Return a number rounded to a certain number of decimal places

21.SQL> select round(15.6789)from dual;

ROUND(15.6789)

--------------

16

The Oracle CEIL function returns the smallest integer value that is greater than or equal to a number.

22. SQL> select ceil(23.20)from dual;

CEIL(23.20)

-----------

24

Return the largest integer value that is greater than or equal to a number

23. SQL> select floor(34.56)from dual;

FLOOR(34.56)

-----------

34

he Oracle/PLSQL TRUNC function returns a number truncated to a certain number of decimal places

24. SQL> select trunc(15.56743)from dual;

TRUNC(15.56743)

---------------

15

The Oracle/PLSQL SIGN function returns a value indicating the sign of a number

25. SQL> select sign(-345)from dual;

SIGN(-345)

----------

-1

Return the absolute value of a number

26. SQL> select abs(-70)from dual; ABS(-70)

---------

70

# MATH FUNCTION:

Returns the value of the specified expression to the specified power.

26. SQL> select power(10,12) from dual;

POWER(10,12)

------------

1.000E+12

SQL MOD() function is used to get the remainder from a division

27. SQL> select mod(11,5) from dual;

MOD(11,5)

---------

1

SQL EXP() returns the exponential value of the argument

28.SQL> select exp(10) from dual;

EXP(10)

---------

22026.466

SQL SQRT() returns the square root of a given value in the argument

29. SQL> select sqrt(225) from dual;

SQRT(225)

---------

15

**RESULT:**

Thus the SQL Functions have been executed successful

|  |  |  |
| --- | --- | --- |
| **EX.NO:7** |  | **JOINING TABLES** |

**AIM**

To study the various Join operations on the database.

SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.

The INNER JOIN keyword return rows when there is at least one match in both tables.

The LEFT JOIN keyword returns all rows from the left table (table\_name1), even if there are no matches in the right table (table\_name2).

The RIGHT JOIN keyword Return all rows from the right table (table\_name2), even if there are no matches in the left table (table\_name1).

The FULL JOIN keyword return rows when there is a match in one of the tables.

**LEFT JOIN or LEFT OUTTER JOIN**

Table:1 - ORDERS

SQL> CREATE table orders(O\_Id number(5),

Orderno number(5),

P\_Id number(3));

Table created.

SQL> DESC orders;

Name Null? Type

--------------------- -------- -----------

O\_ID NUMBER(5)

ORDERNO NUMBER(5)

P\_ID NUMBER(3)

INSERTING VALUES INTO ORDERS

* SQL> INSERT into orders values(&O\_Id,&Orderno,&P\_Id);

Enter value for o\_id: 1

Enter value for orderno: 77895

Enter value for p\_id: 3

old 1: INSERT into orders values(&O\_Id,&Orderno,&P\_Id) new 1: INSERT into orders values(1,77895,3)

1 row created.

* SQL> INSERT into orders values(&O\_Id,&Orderno,&P\_Id);

Enter value for o\_id: 2

Enter value for orderno: 44678 Enter value for p\_id: 3

old 1: INSERT into orders values(&O\_Id,&Orderno,&P\_Id) new 1: INSERT into orders values(2,44678,3) 1 row created.

* SQL> INSERT into orders values(&O\_Id,&Orderno,&P\_Id);

Enter value for o\_id: 3

Enter value for orderno: 22456 Enter value for p\_id: 1

old 1: INSERT into orders values(&O\_Id,&Orderno,&P\_Id) new 1: INSERT into orders values(3,22456,1)

1 row created.

* SQL> INSERT into orders values(&O\_Id,&Orderno,&P\_Id);

Enter value for o\_id: 4

Enter value for orderno: 24562 Enter value for p\_id: 1

old 1: INSERT into orders values(&O\_Id,&Orderno,&P\_Id) new 1: INSERT into orders values(4,24562,1)

1 row created.

* SQL> INSERT into orders values(&O\_Id,&Orderno,&P\_Id);

Enter value for o\_id: 5

Enter value for orderno: 34764 Enter value for p\_id: 15

old 1: INSERT into orders values(&O\_Id,&Orderno,&P\_Id)

new 1: INSERT into orders values(5,34764,15)

1. row created.

TABLE SECTION:

SQL> SELECT \* FROM orders;

O\_ID ORDERNO P\_ID

---------- ---------- ---------- 1 77895 3

1. 44678 3
2. 22456 1
3. 24562 1
4. 34764 15

TABLE -2: PERSONS

* SQL> CREATE table persons(p\_Id number(5),

LASTNAME varchar2(10),

Firstname varchar2(15), Address varchar2(20),

city varchar2(10));

Table created.

* SQL> INSERT into persons values(&p\_Id,'&Lastname','&firstname','&Address','&city'); Enter value for p\_id: 1

Enter value for lastname: Hansen

Enter value for firstname: Ola

Enter value for address: Timoteivn 10

Enter value for city: sadnes

old 1: INSERT into persons values(&p\_Id,'&Lastname','&firstname','&Address','&city') new 1: INSERT into persons values(1,'Hansen','Ola','Timoteivn 10','sadnes')

1 row created.

* SQL> INSERT into persons values(&p\_Id,'&Lastname','&firstname','&Address','&city'); Enter value for p\_id: 2

Enter value for lastname: Svendson

Enter value for firstname: Tove

Enter value for address: Borgn 23

Enter value for city: Sandnes

old 1: INSERT into persons values(&p\_Id,'&Lastname','&firstname','&Address','&city') new 1: INSERT into persons values(2,'Svendson','Tove','Borgn 23','Sandnes')

1 row created.

* SQL> INSERT into persons values(&p\_Id,'&Lastname','&firstname','&Address','&city'); Enter value for p\_id: 3

Enter value for lastname: Pettersen

Enter value for firstname: Kari

Enter value for address: Storgt 20 Enter value for city: Stavanger

old 1: INSERT into persons values(&p\_Id,'&Lastname','&firstname','&Address','&city') new 1: INSERT into persons values(3,'Pettersen','Kari','Storgt 20','Stavanger')

1 row created.

* SQL> SELECT \* FROM persons;

P\_ID LASTNAME FIRSTNAME ADDRESS CITY

---------- ---------- --------------- ------------------- ---------- 1 Hansen Ola Timoteivn 10 sandnes 2 Svendson Tove Borgn 23 Sandnes

3 Pettersen Kari Storgt 20 Stavanger

**LEFT JOIN SYNTAX**

SQL> SELECT column\_name(s)

FROM table\_name1

LEFT JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

**LEFT JOIN EXAMPLE**

SQL> SELECT persons.lastname,persons.firstname,orders.orderno

FROM persons

LEFT JOIN orders

ON persons.p\_Id = orders.p\_Id

ORDER BY persons.lastname;

OUTPUT

LASTNAME FIRSTNAME ORDERNO

------------------ ------------------ ---------------

Hansen Ola 22456

Hansen Ola 24562

Pettersen Kari 77895

Pettersen Kari 44678

Svendson Tove

**FULL OUTTER JOIN**

SQL> SELECT \* FROM persons;

P\_ID LASTNAME FIRSTNAME ADDRESS CITY

---------- --------------- -------------------- ---------------- ---------- 1 Hansen Ola Timoteivn 10 sandnes 2 Svendson Tove Borgn 23 Sandnes

1. Pettersen Kari Storgt 20 Stavanger

SQL> SELECT \* FROM orders; O\_ID ORDERNO P\_ID

---------- ---------- ---------- 1 77895 3

1. 44678 3
2. 22456 1

1. 24562 1
2. 34764 15

**FULL OUTER JOIN SYNTAX**

SQL>SELECT column\_name(s)

FROM table\_name1

FULL JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

**FULL OUTER JOIN EXAMPLE**

SQL> SELECT persons.lastname,persons.firstname,orders.orderno

FROM persons

FULL OUTER JOIN orders

ON persons.p\_Id = orders.p\_Id

ORDER BY persons.lastname;

**RIGHT OUTTER JOIN SYNTAX**

SQL>SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

RIGHT JOIN Orders

ON Persons.P\_Id=Orders.P\_Id

ORDER BY Persons.LastName

**RIGHT OUTTER JOIN EXAMPLE**

SQL> SELECT persons.lastname,persons.firstname,orders.orderno

FROM persons

RIGHT OUTER JOIN orders

ON persons.p\_Id = orders.p\_Id

ORDER BY persons.lastname;

LASTNAME FIRSTNAME ORDERNO

------------------- ------------------ ---------------

Hansen Ola 24562

Hansen Ola 22456

Pettersen Kari 44678

Pettersen Kari 77895

**INNER JOIN**

**INNTER JOIN SYNTAX**

SQL>SELECT column\_name(s)

FROM table\_name1

INNER JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

**INNTER JOIN EXAMPLE**

SQL> SELECT persons.lastname,persons.firstname,orders.orderno

1. FROM persons
2. INNER JOIN orders
3. ON persons.p\_Id = orders.p\_Id
4. ORDER BY persons.lastname;

LASTNAME FIRSTNAME ORDERNO

------------------ ------------------ ---------------

Hansen Ola 22456

Hansen Ola 24562

Pettersen Kari 77895

Pettersen Kari 44678

LASTNAME FIRSTNAME ORDERNO

------------- --------------- ----------

Hansen Ola 22456

Hansen Ola 24562

Pettersen Kari 77895

Pettersen Kari 44678 Svendson Tove 34764 6 rows selected.

**RESULT:**

Thus the SQL Functions have been executed successful

|  |  |  |  |
| --- | --- | --- | --- |
| **EX.NO:8** |  |  | **NESTED QUERIES** |

**AIM**

To study the various SQL sub queries operations on the database.

Sub Query can have more than one level of nesting in one single query. A SQL nested query is a SELECT query that is nested inside a SELECT, UPDATE, INSERT, or DELETE SQL query.

1. Select Command Is Used To Select Records From The Table.
2. Where Command Is Used To Identify Particular Elements.
3. Having Command Is Used To Identify Particular Elements.
4. Min (Sal) Command Is Used To Find Minimum Salary.

**Syntax For Creating A Table:**

Sql: Create <Obj.Type> <Obj.Name> (Column Name.1 <Datatype>

(Size), Column Name.1 <Datatype> (Size) ……………………………);

Sql> Create Table Emp2(Empno Number(5),

Ename Varchar2(20),

Job Varchar2(20),

Sal Number(6),

Mgrno Number(4),

Deptno Number(3));

**Syntax For Insert Records In To A Table:**

Sql :> Insert Into <Table Name> Values< Val1, ‘Val2’,…..);

Insertion

Sql> Insert Into Emp2 Values(1001,'Mahesh','Programmer',15000,1560,200); 1 Row Created.

Sql> Insert Into Emp2 Values (1002,'Manoj','Tester',12000,1560,200); 1 Row Created.

Sql> Insert Into Emp2 Values(1003,'Karthik','Programmer',13000,1400,201); 1 Row Created.

Sql> Insert Into Emp2 Values(1004,'Naresh','Clerk',1400,1400,201); 1 Row Created.

Sql> Insert Into Emp2 Values(1005,'Mani','Tester',13000,1400,200); 1 Row Created.

Sql> Insert Into Emp2 Values(1006,'Viki','Designer',12500,1560,201); 1 Row Created.

Sql> Insert Into Emp2 Values(1007,'Mohan','Designer',14000,1560,201);1 Row Created. Sql> Insert Into Emp2 Values(1008,'Naveen','Creation',20000,1400,201); 1 Row Created.

Sql> Insert Into Emp2 Values(1009,'Prasad','Dir',20000,1560,202); 1 Row Created.

Sql> Insert Into Emp2 Values(1010,'Agnesh','Dir',15000,1400,200); 1 Row Created.

**Syntax For Select Records From The Table:**

Sql> Select \* From <Table Name>;

Sql> Select \*From Emp2;

Empno Ename Job Sal Mgrno Dptno

---------- ---------- ---------- ---------- ---------- ----------

1. Mahesh Programmer 15000 1560 200
2. Manoj Tester 12000 1560 200
3. Karthik Programmer 13000 1400 201
4. Naresh Clerk 1400 1400 201
5. Mani Tester 13000 1400 200
6. Viki Designer 12500 1560 201
7. Mohan Designer 14000 1560 201
8. Naveen Creation 20000 1400 201
9. Prasad Dir 20000 1560 202
10. Agnesh Dir 15000 1400 200

Table- 2

**Syntax For Creating A Table:**

Sql: Create <Obj.Type> <Obj.Name> (Column Name.1 <Datatype> (Size), Column Name.1 <Datatype> (Size) ……………………………);

Sql> Create Table Dept2(Deptno Number(3),

Deptname Varchar2(10), Location Varchar2(15)); Table Created.

**Syntax For Insert Records In To A Table:**

Sql :> Insert Into <Table Name> Values< Val1, ‘Val2’,…..);

**Insertion**

Sql> Insert Into Dept2 Values(107,'Develop','Adyar'); 1 Row Created.

Sql> Insert Into Dept2 Values(201,'Debug','Uk'); 1 Row Created.

Sql> Insert Into Dept2 Values(200,'Test','Us');

Sql> Insert Into Dept2 Values(201,'Test','Ussr'); 1 Row Created.

Sql> Insert Into Dept2 Values(108,'Debug','Adyar'); 1 Row Created.

Sql> Insert Into Dept2 Values(109,'Build','Potheri'); 1 Row Created.

Syntax For Select Records From The Table:

Sql> Select \* From <Table Name>;

Sql> Select \*From Dept2;

Deptno Deptname Location

---------- ---------- ---------------

107 Develop Adyar

201 Debug Uk

1. Test Us
2. Test Ussr

108 Debug Adyar 109 Build Potheri 6 Rows Selected.

**General Syntax For Nested Query:**

Select "Column\_Name1"

From "Table\_Name1"

Where "Column\_Name2" [Comparison Operator]

(Select "Column\_Name3"

From "Table\_Name2"

Where [Condition])

**Syntax Nested Query Statement:**

Sql> Select <Column\_Name> From Frorm <Table \_1> Where

<Column\_Name> <Relational \_Operation> ‘Value’

(Select (Aggrecate Function) From <Table\_1> Where <Column Name> = ‘Value’

(Select <Column\_Name> From <Table\_2> Where <Column\_Name=

‘Value’));

Nested Query Statement:

Sql> Select Ename From Emp2 Where Sal>

(Select Min(Sal) From Emp2 Where Dptno=

(Select Deptno From Dept2 Where Location='Uk'));

**Nested Query Output:**

Ename

----------

Mahesh

Manoj

Karthik

Mani

Viki

Mohan

Naveen

Prasad

Agnesh

**RESULT:**

Thus the SQL sub query has been executed successfully

|  |  |  |  |
| --- | --- | --- | --- |
| **EX.NO:9** |  |  | **BASICS OF PL/SQL** |

**AIM** : To study the various basic PL/SQL view operations on the database.

1. **Write a PL/SQL program to check whether the given number is prime or not.**

**PROGRAM**

set serveroutput on

declare

num int;

r ,flag int;

begin

num:='&num';

for i in 2..num-1

loop

r=mod(num,i);

if (r=0)

then

falag=1;

end if;

end loop;

if(flag=1) then

dbms\_output.put\_line('The number is not prime');

else

dbms\_output.put\_line('The number is prime');

end if;

end;

**OUTPUT**

Enter the value for num:4

Old **7:**num:=’&num’;

New 7:num:=’&num’;

The number is not prime

1. **Write a PL/SQL program to find the factorial of a given number.**

**PROGRAM**

set serveroutput on

declare

n number(3);

fact number(5);

begin

n:=&n;

if(n=0)then

dbms\_output.put\_line('fact(0) is 1');

end if;

fact:=1;

for i in 1..n

loop

fact:=fact\*i;

end loop;

dbms\_output.put\_line('fact(n) is'||fact);

end;

**OUTPUT**

Enter the value for n:4

Old 6:n:=&n;

New 6:n:=&4;

Fact(n) is 24

1. **CURSOR PROGRAM TO DISPLAY ALL THE EMPLOYEE HAVING SALARY GREATER THAN 10,000**

**PROGRAM**

set serveroutput on;

declare

cursor empdet is select \* from curex where sal>10000;

c\_var curex%rowtype;

i number(2);

begin

if not empdet%isopen then

open empdet;

end if;

fetch empdet into c\_var;

while empdet%found

loop

dbms\_output.put\_line('The employee having salary more than 10000 are: '|| c\_var.name || ',' || c\_var.id );

fetch empdet into c\_var;

end loop;

close empdet;

end;

/

**OUTPUT:**

SQL> select \* from curex;

NAME ID DOB DOJ SAL

--------------- ---------- --------- --------- ----------

ramesh 301 12-OCT-78 13-JUN-90 25000

ramu 302 14-JAN-80 15-OCT-99 34000

rajesh 303 25-MAR-67 23-JAN-88 50000

vicky 304 12-JAN-77 13-OCT-90 40000

dharmendra 305 14-AUG-88 12-JUN-00 4000

The employee having salary more than 10000 are: ramesh,301

The employee having salary more than 10000 are: ramu,302

The employee having salary more than 10000 are: rajesh,303

The employee having salary more than 10000 are: vicky,304

PL/SQL procedure successfully completed.

**4. PL/SQL BLOCK THAT HANDLES EXCEPTIONS.**

**PROGRAM**

SQL> set serveroutput on;

SQL> declare

incorrect\_id exception ;

cursor a is select id from curex;

a\_var curex.id%type;

nme varchar2(10);

n number(3);

begin

select '&name',&id into nme,n from dual;

open a;

fetch a into a\_var;

while a%found

loop

if a\_var=n then

raise incorrect\_id;

end if;

fetch a into a\_var;

end loop;

insert into curex (name,id) values (nme,n);

exception

when incorrect\_id then

dbms\_output.put\_line('empno already exists');

end;

/

**OUTPUT:**

SQl>@exceptn.sql

Enter value for name: ramu

Enter value for id: 302

old 8: select '&name',&id into nme,n from dual;

new 8: select 'ramu',302 into nme,n from dual;

empno already exists

PL/SQL procedure successfully completed.

SQL> select \* from curex;

NAME ID DOB DOJ SAL

--------------- ---------- --------- --------- ----------

ramesh 301 12-OCT-78 13-JUN-90 27000

ramu 302 14-JAN-80 15-OCT-99 36000

rajesh 303 25-MAR-67 23-JAN-88 52000

vicky 304 12-JAN-77 13-OCT-90 42000

dharmendra 305 14-AUG-88 12-JUN-00 36000

SQl>@exceptn.sql

Enter value for name: amitava

Enter value for id: 306

old 8: select '&name',&id into nme,n from dual;

new 8: select 'amitava',306 into nme,n from dual;

PL/SQL procedure successfully completed.

SQL> select \* from curex;

NAME ID DOB DOJ SAL

--------------- ---------- --------- --------- ----------

ramesh 301 12-OCT-78 13-JUN-90 27000

ramu 302 14-JAN-80 15-OCT-99 36000

rajesh 303 25-MAR-67 23-JAN-88 52000

vicky 304 12-JAN-77 13-OCT-90 42000

dharmendra 305 14-AUG-88 12-JUN-00 36000

amitava 306

1. rows selected.
2. **PROCEDURE TO FIND THE PRICE OF A PARTICULAR PRODUCT**

create or replace procedure prod (pid in number,quantity in number,price out number) is

uprice number;

begin

select up into uprice from product where product.pid=prod.pid;

price:=quantity\*uprice;

end;

**PROGRAM**

set serveroutput on

declare

price number;

begin

prod(&pid,&quantity,price);

dbms\_output.put\_line('price='||price);

end;

**OUTPUT**

SQL> select \* from product;

PID PNAME UP

---------- --------------- ----------

101 maggi 10

102 milkybar 15

103 dairymilk 10

104 5star 5

SQL> @ prd.sql;

7 /

Procedure created.

SQL> @ pg3.sql;

7 /

Enter value for pid: 102

Enter value for quantity: 2

old 4: prod(&pid,&quantity,price);

new 4: prod(102,2,price);

price=30

PL/SQL procedure successfully completed.

**7. FUNCTION TO DETERMINE THE SENIOR EMPLOYEE BASED ON EMPLOYMENT DATE USING FUNCTION**

**PROGRAM:**

create or replace function find\_senior

return varchar2 is

cursor c is select min(doj) from curex;

c\_var date;

cursor d is select name from curex where doj= c\_var;

d\_var curex.name%type;

begin

open c;

fetch c into c\_var;

close c;

open d;

fetch d into d\_var;

close d;

return d\_var;

end find\_senior;

/

Function created.

SQL> set serveroutput on

SQL> declare

nm varchar2(20);

begin

nm:=find\_senior;

dbms\_output.put\_line('The senior Employee is '||nm);

end;

/

**OUTPUT:**

SQL> select \* from curex;

NAME ID DOB DOJ SAL

--------------- ---------- --------- --------- ----------

ramesh 301 12-OCT-78 13-JUN-90 27000

ramu 302 14-JAN-80 15-OCT-99 36000

rajesh 303 25-MAR-67 23-JAN-88 52000

vicky 304 12-JAN-77 13-OCT-90 42000

dharmendra 305 14-AUG-88 12-JUN-00 36000

SQL> @senior.sql;

'The senior Employee is rajesh

PL/SQL procedure successfully completed

**8. PHONE NUMBER VALIDATION TRIGGER**

**PROGRAM:**

create or replace trigger tg2 before insert on test

for each row

begin

if (:new.ph is null) then

:new.ph:='NULL';

end if;

end;

**OUTPUT**

SQL> @t4.sql;

9 /

Trigger created.

SQL> insert into test values('&name','&ph');

Enter value for name: sara

Enter value for ph: 999999

old 1: insert into test values('&name','&ph')

new 1: insert into test values('sara','999999')

1 row created.

SQL> /

Enter value for name: ram

Enter value for ph:

old 1: insert into test values('&name','&ph')

new 1: insert into test values('ram','')

1 row created.

SQL> select \* from test;

NAME PH

---------- ----------

sara 999999

ram NULL

**9. WRITE A PL-SQL CURSOR PROGRAM TO COMPUTE THE RETIREMENT AGE OF THE YOUNGEST EMPLOYEE AND ALSO LIST OF EMPLOYEE NAME.**

SQL> select \* from curex;

NAME ID DOB DOJ SAL

--------------- ---------- --------- --------- ----------

ramesh 301 12-OCT-78 13-JUN-90 25000

ramu 302 14-JAN-80 15-OCT-99 34000

rajesh 303 25-MAR-67 23-JAN-88 50000

vicky 304 12-JAN-77 13-OCT-90 40000

dharmendra 305 14-AUG-88 12-JUN-00 34000

**PROGRAM:**

SQL> set serveroutput on;

SQL> declare

new\_bday date;

nm curex.name%type;

eid curex.id%type;

ra date;

cursor find\_new\_bday is select max(dob)from curex;

cursor emp\_det is select name,id from curex where dob=new\_bday;

begin

if not find\_new\_bday%isopen then

open find\_new\_bday;

end if;

fetch find\_new\_bday into new\_bday;

close find\_new\_bday;

open emp\_det;

fetch emp\_det into nm,eid;

close emp\_det;

dbms\_output.put\_line('The youngest employee is ' || nm ||',' || eid);

ra:=add\_months(new\_bday,60\*12);

dbms\_output.put\_line('Retirement date is' || ra);

end;

/

**OUTPUT:**

The youngest employee is dharmendra,305

Retirement date is 14-AUG-48

PL/SQL procedure successfully completed.

**10. WRITE A PL-SQL CURSOR PROGRAM TO GENERATE EMPLOYEE DETAILS AND GIVE BONUS OF Rs1000 FOR ALL EMPLOYEES WHO HAVE JOINED BEFORE 01-JAN-2001**

**PROGRAM:**

SQL> set serveroutput on;

SQL> declare

cursor a is select name,dob,sal from curex for update;

a\_var a%rowtype;

begin

if not a%isopen then

open a;

end if;

dbms\_output.put\_line('hai');

fetch a into a\_var;

while a%found

loop

update curex set sal=sal+1000 where current of a;

fetch a into a\_var;

end loop;

close a;

dbms\_output.put\_line('Emp details are');

commit;

open a;

fetch a into a\_var;

while a%found

loop

dbms\_output.put\_line(a\_var.name|| a\_var.dob || a\_var.sal);

fetch a into a\_var;

end loop;

close a;

end;

/

**OUTPUT**

Hai

Emp details r

ramesh 12-OCT-78 27000

ramu 14-JAN-80 36000

rajesh 25-MAR-67 52000

vicky 12-JAN-77 42000

dharmendra14-AUG-88 36000

PL/SQL procedure successfully complete

**11. TO DISPLAY EMPLOYEE WHO ARE EMPLOYED BETWEEN ANY 2 INPUT DATES USING PROCEDURE**

**PROGRAM:**

SQL> create or replace procedure find\_emp

( x in date,y in date) is

cursor c is select name,dob from curex;

c\_var c%rowtype;

begin

open c;

fetch c into c\_var;

while c%found

loop

if c\_var.dob>=x and c\_var.dob<=y then

dbms\_output.put\_line(c\_var.name);

end if;

fetch c into c\_var;

end loop;

close c;

end find\_emp;

/

Procedure created.

SQL> set serveroutput on

SQL> declare

a date;

b date;

begin

select '&a' into a from dual;

select '&b' into b from dual;

find\_emp(a,b);

end;

/

**OUTPUT:**

Enter value for a: 1-jan-80

old 8: select '&a' into a from dual;

new 8: select '1-jan-80' into a from dual;

Enter value for b: 1-jan-90

old 12: select '&b' into b from dual;

new 12: select '1-jan-90' into b from dual;

ramu

dharmendra

PL/SQL procedure successfully completed.

**RESULT:**

Thus the basics of pl/sql have been executed successfully.

|  |  |  |  |
| --- | --- | --- | --- |
| **EX.NO:10** |  |  | **PROJECT**  **TITLE:TELECOMMUNICATION** |

**ABSTRACT**

Telecommunication is the transmission of signs, signals, messages, words, writings, images and sounds or intelligence of any nature by wire, radio, optical or other electromagnetic systems. Telecommunication occurs when the exchange of information between communication participants includes the use of technology. Customer care is a very important aspect for customer satisfaction for the tele-companies and thus is a growing field.

The project focuses on developing an easy-to-use GUI for customer care executives so that their performance could be maximized. Using this project ,the executive will be able to do all the task assigned to him with ease.

**Algorithm/Procedure:**

1.Import all the moduels and jdbc driver and libraries to connect with sql.

2.create classes and define the functions in the classes.

3.Using java swing make the GUI and connect it with sql tables.

4.Now take the input from the user and store it in database.

5.Show the result.

**Modules:**

**Login:**

1. Create database if it does not exist already.

2. If database exists, connect to the database.

3. Create a table if it does not exist.

4. Fetch username and password from the users.

5. Send a query for checking if the credentials match.

6. If yes, them proceed to the home page and start a session.

7. If no, then say “Invalid Credentials”.

**Register/Add new employee:**

1. Create database if it does not exist already.
2. If database exists, connect to the database.
3. Create a table if it does not exist.
4. Fetch details from the users using a form.
5. Send a query with the fetched details to store in the database.
6. If id already exists, send “Email address already exists”
7. Else, if the user is new, feed the data to the table.

**Main menu:-**

1.Displays all the contents and functionalities

2.Provides a central control

Add Client:-

1.Adds a new customer to the database.

2.Gets the customer details and stores in the database

**Add new Offers:-**

1.Adds new Offers

2.Takes the details and store the offer for the particular mobile number in the database.

**Show Offers:**

1.Retrieve the offer data from database.

2.Displays all the offers available for a particular mobile number

**Update/show client details**

1.Retrieves the client data from database.

2.displays the retrieved data in the form

3.Update the edited data in the database.

**Code:**

**1.Login:**

try{

Integer nempno=Integer.parseInt(jTextField1.getText());

String Query="Select pass from employee where empno="+(nempno)+"; ";

Class.forName("com.mysql.jdbc.Driver");

Connection con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement st=con.createStatement();

ResultSet rs=st.executeQuery(Query);

rs.next();

String npass=rs.getString("pass");

String pass=jPasswordField1.getText();

MainMenu m=new MainMenu();

if(npass.equals(pass))

{

this.dispose();

//JFrame.add(m);

m.setVisible(true);

}

else

{

JOptionPane.showMessageDialog(null,"Wrong Password Try Again");

jPasswordField1.setText("");

}

con.close();

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,"No ID Found");

e.printStackTrace();

}

**Add employee/Register:**

try{

Integer nempno=Integer.parseInt(jTextField1.getText());

String npass=jTextField2.getText();

String nname=jTextField3.getText();

String ndept=jTextField4.getText();

String ndesg=jTextField5.getText();

String Query="insert into employee(empno,pass,name,dept,desg) values ("+(nempno)+",'" + (npass) + "','" + (nname) + "','" + (ndept) + "','" + (ndesg) + "')";

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement smt=con.createStatement();

System.out.println("rows effected");

int rowsEffected = smt.executeUpdate(Query);

System.out.println(rowsEffected + "rows effected");

JOptionPane.showMessageDialog(null,"record added");

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

jTextField4.setText("");

jTextField5.setText("");

}

catch(Exception e)

{

JOptionPane.showMessageDialog(this, e.getMessage());

e.printStackTrace();

}

**ADD new client:**

try{

Integer nphno=Integer.parseInt(jTextField1.getText());

String naddrprf=jTextField3.getText();

String nname=jTextField2.getText();

String nstate=jTextField8.getText();

String ndor=jTextField5.getText();

String naddr=jTextArea1.getText();

String nmail=jTextField6.getText();

jTextField8.setText("");

String Query="insert into client(mobno,name,address\_proof,state,reg\_date,address,email) values ("+(nphno)+",'" + (nname) + "','" + (naddrprf) + "','" + (nstate) + "','" + (ndor) + "','" + (naddr) + "','" + (nmail) + "')";

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement smt=con.createStatement();

System.out.println("rows effected");

int rowsEffected = smt.executeUpdate(Query);

System.out.println(rowsEffected + "rows effected");

JOptionPane.showMessageDialog(null,"record added");

jTextField3.setText("");

jTextField1.setText("");

jTextField2.setText("");

jTextField4.setText("");

jTextField5.setText("");

jTextField6.setText("");

jTextArea1.setText("");

}

catch(Exception e)

{

JOptionPane.showMessageDialog(this, e.getMessage());

e.printStackTrace();}

**3.ADD offers**

try{

Integer nmobno=Integer.parseInt(jTextField1.getText());

String nplan=jTextField2.getText();

String nvld=jTextField3.getText();

Integer nprice=Integer.parseInt(jTextField4.getText());

String ndscp=jTextArea1.getText();

String Query="insert into offers(mobno,plan,validity,price,description) values ("+(nmobno)+",'" + (nplan) + "','" + (nvld) + "'," + (nprice) + ",'" + (ndscp) + "')";

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement smt=con.createStatement();

System.out.println("rows effected");

int rowsEffected = smt.executeUpdate(Query);

System.out.println(rowsEffected + "rows effected");

JOptionPane.showMessageDialog(null,"record added");

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

jTextField4.setText("");

jTextArea1.setText("");

}

catch(Exception e)

{

JOptionPane.showMessageDialog(this, e.getMessage());

e.printStackTrace();

}

**4.Show offers:**

try{

Integer nmobno=Integer.parseInt(jTextField1.getText());

String Query="Select \* from offers where mobno="+(nmobno)+"; ";

Class.forName("com.mysql.jdbc.Driver");

Connection con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement st=con.createStatement();

ResultSet rs=st.executeQuery(Query);

DefaultListModel m = new DefaultListModel();

jList1.setModel(m);

DefaultListModel m2 = new DefaultListModel();

jList2.setModel(m2);

System.out.println("nprice");

while(rs.next())

{

String nplan=rs.getString("plan");

String nvald=rs.getString("validity");

String nprice=rs.getString("price");

System.out.println("nprice");

m.addElement(nplan+"--"+nvald+"--"+nprice);

m2.addElement(nplan);

}

con.close();

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,e.getMessage());

e.printStackTrace();

}

}

private void jList1ValueChanged(javax.swing.event.ListSelectionEvent evt) {

}

private void jList2ValueChanged(javax.swing.event.ListSelectionEvent evt) {

String s=jList2.getSelectedValue();

try{

String Query="Select \* from offers where plan like '"+(s)+"'; ";

Class.forName("com.mysql.jdbc.Driver");

Connection con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement st=con.createStatement();

ResultSet rs=st.executeQuery(Query);

while(rs.next())

{

String ndsc=rs.getString("description");

jTextArea2.setText(ndsc);

}

con.close();

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,e.getMessage());

e.printStackTrace();

}

**5.Show client details:**

try{

Integer nmobno=Integer.parseInt(jTextField2.getText());

jTextField2.setEditable(false);

String Query="Select \* from client where mobno="+(nmobno)+"; ";

Class.forName("com.mysql.jdbc.Driver");

Connection con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement st=con.createStatement();

ResultSet rs=st.executeQuery(Query);

// DefaultListModel m =(DefaultListModel) jList1.getModel();

rs.next();

String nname=rs.getString("name");

String naddrprf=rs.getString("address\_proof");

String nstate=rs.getString("state");

String ndor=rs.getString("reg\_date");

String naddr=rs.getString("address");

String nmail=rs.getString("email");

jTextField1.setText(naddrprf);

jTextField2.setText(""+nmobno);

jTextField3.setText(nname);

jTextField4.setText(nstate);

jTextField5.setText(ndor);

jTextArea1.setText(naddr);

jTextField6.setText(nmail);

con.close();

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,e.getMessage());

e.printStackTrace();

}

**6.Update client details:-**

try{

Integer nphno=Integer.parseInt(jTextField2.getText());

String naddrprf=jTextField1.getText();

String nname=jTextField3.getText();

String nstate=jTextField4.getText();

String ndor=jTextField5.getText();

String naddr=jTextArea1.getText();

String nmail=jTextField6.getText();

String Query="update client set name='" + (nname) + "', address\_proof='" + (naddrprf) + "',state='" + (nstate) + "',reg\_date='" + (ndor) + "',address='" + (naddr) + "',email='" + (nmail) + "' where mobno="+(nphno)+";";

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement smt=con.createStatement();

System.out.println("rows effected");

int rowsEffected = smt.executeUpdate(Query);

System.out.println(rowsEffected + "rows effected");

JOptionPane.showMessageDialog(null,"record added");

jTextField3.setText("");

jTextField1.setText("");

jTextField2.setText("");

jTextField4.setText("");

jTextField5.setText("");

jTextField6.setText("");

jTextArea1.setText("");

}

catch(Exception e)

{

JOptionPane.showMessageDialog(this, e.getMessage());

e.printStackTrace();

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

try{

Integer nmobno=Integer.parseInt(jTextField2.getText());

jTextField2.setEditable(false);

String Query="Select \* from client where mobno="+(nmobno)+"; ";

Class.forName("com.mysql.jdbc.Driver");

Connection con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost/dbms","root","20oct1997");

Statement st=con.createStatement();

ResultSet rs=st.executeQuery(Query);

// DefaultListModel m =(DefaultListModel) jList1.getModel();

rs.next();

String nname=rs.getString("name");

String naddrprf=rs.getString("address\_proof");

String nstate=rs.getString("state");

String ndor=rs.getString("reg\_date");

String naddr=rs.getString("address");

String nmail=rs.getString("email");

jTextField1.setText(naddrprf);

jTextField2.setText(""+nmobno);

jTextField3.setText(nname);

jTextField4.setText(nstate);

jTextField5.setText(ndor);

jTextArea1.setText(naddr);

jTextField6.setText(nmail);

con.close();

}

catch(Exception e)

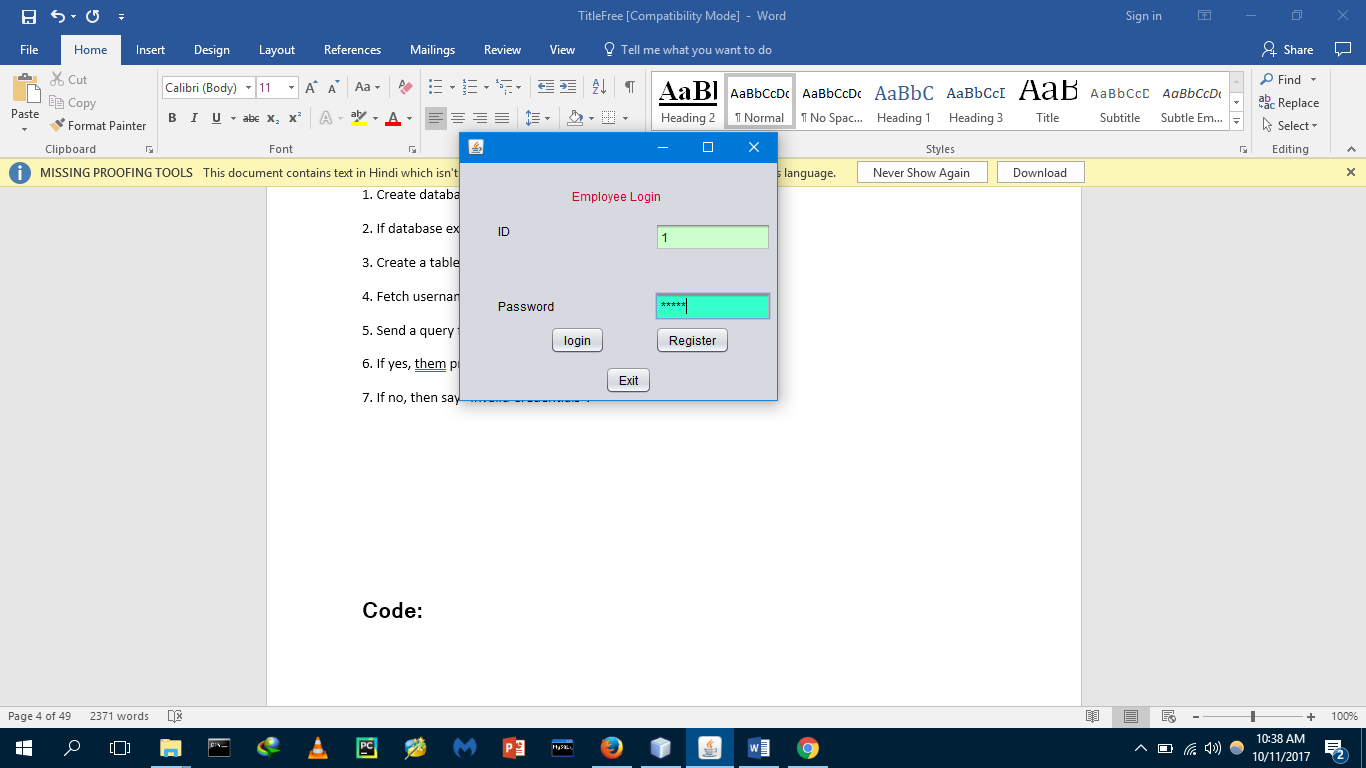
{

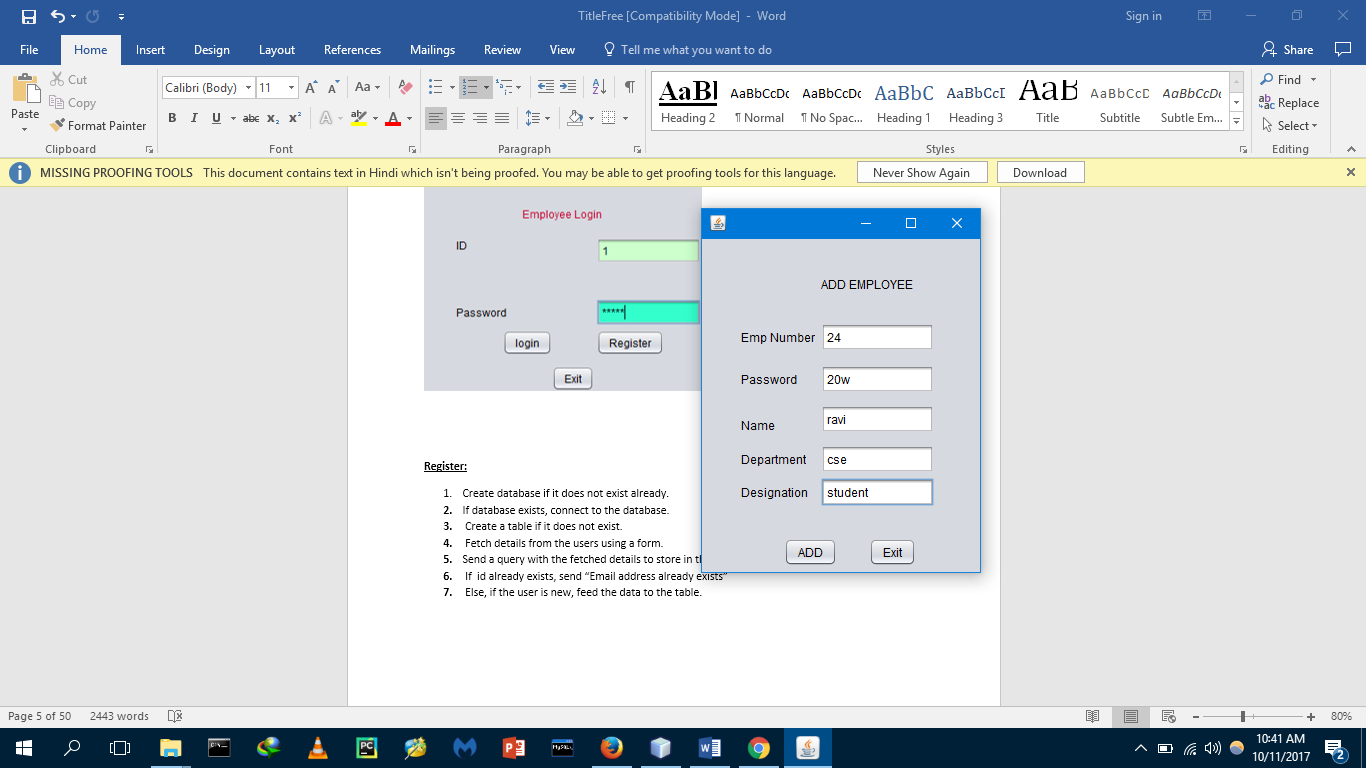
JOptionPane.showMessageDialog(null,e.getMessage());

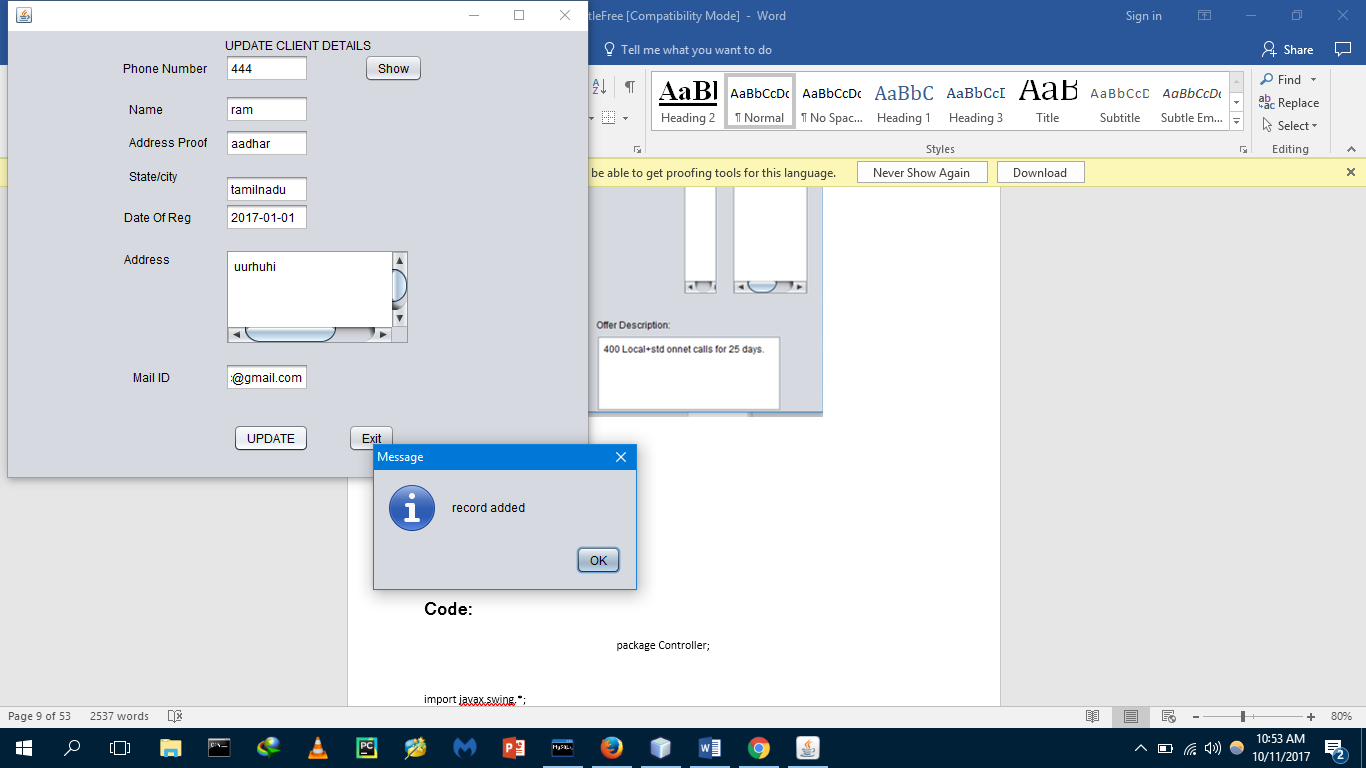
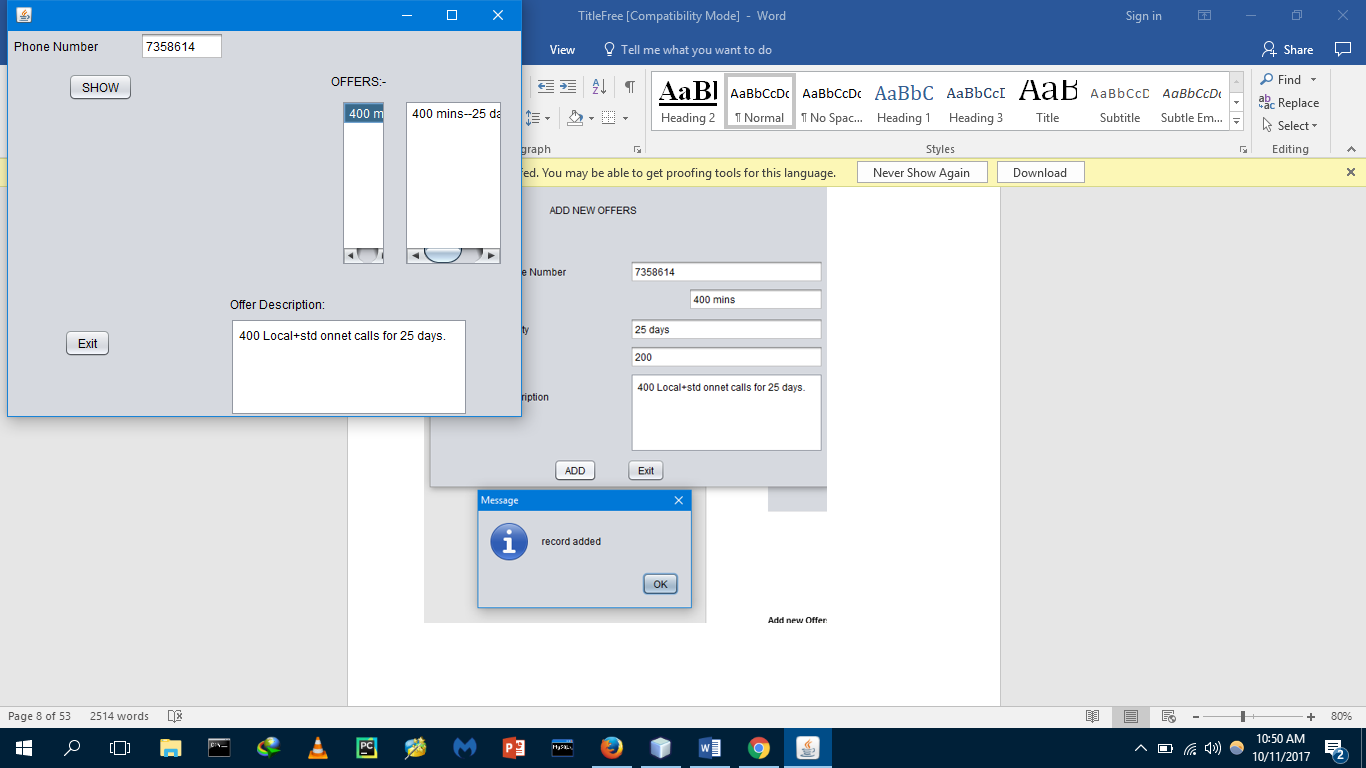
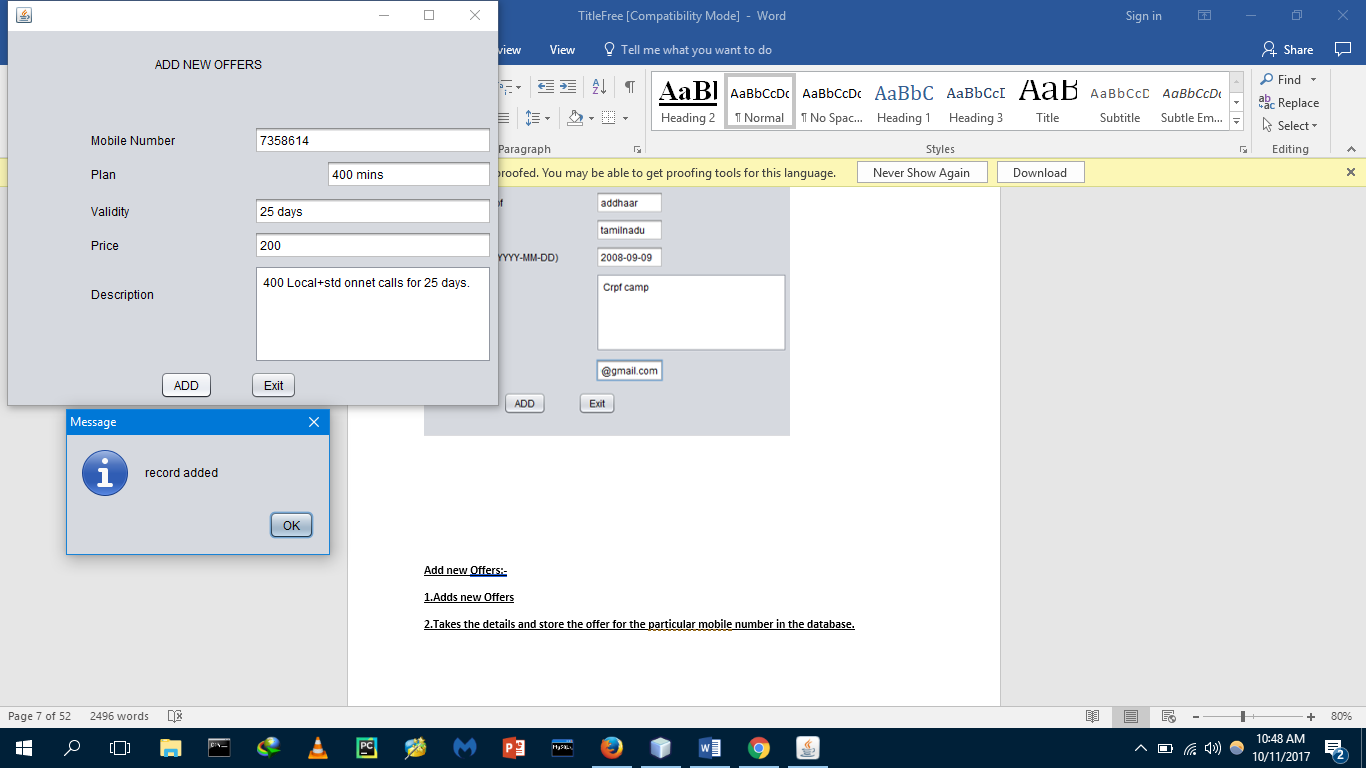
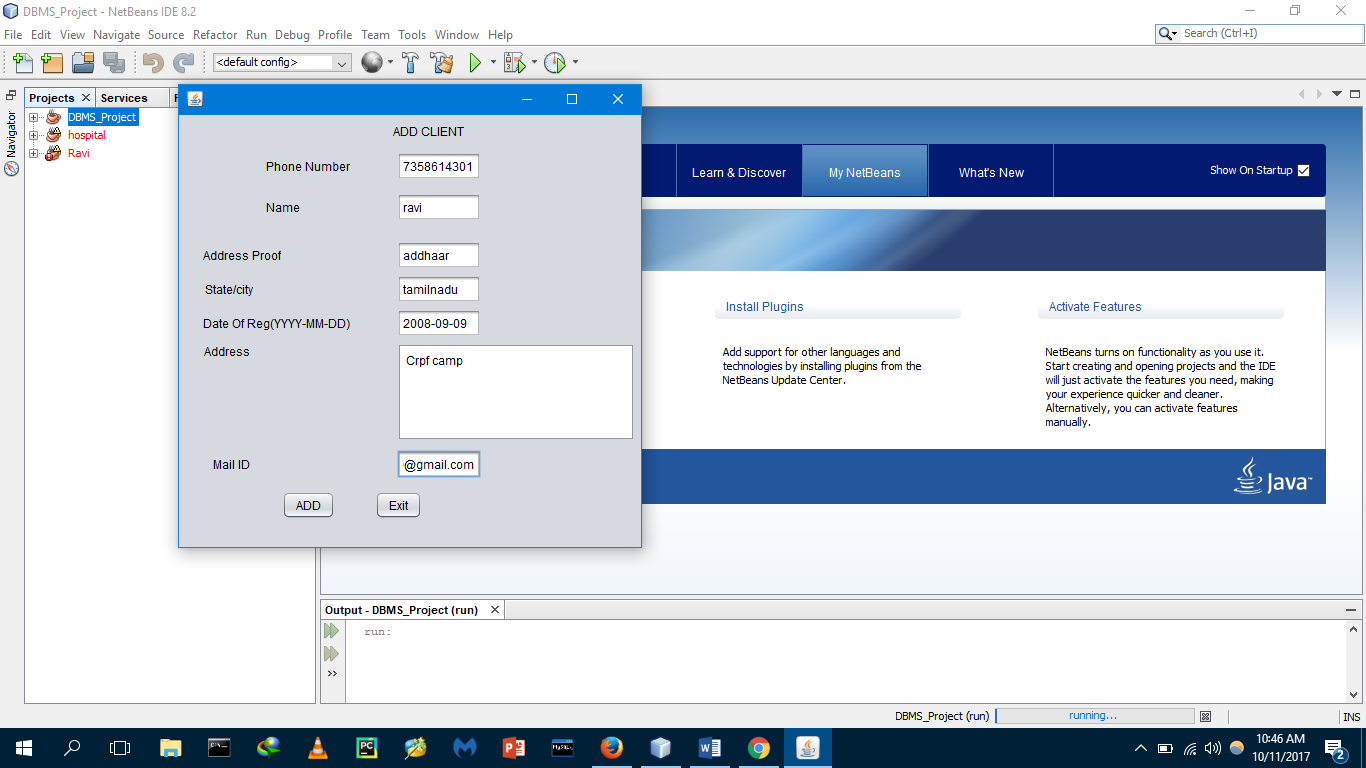
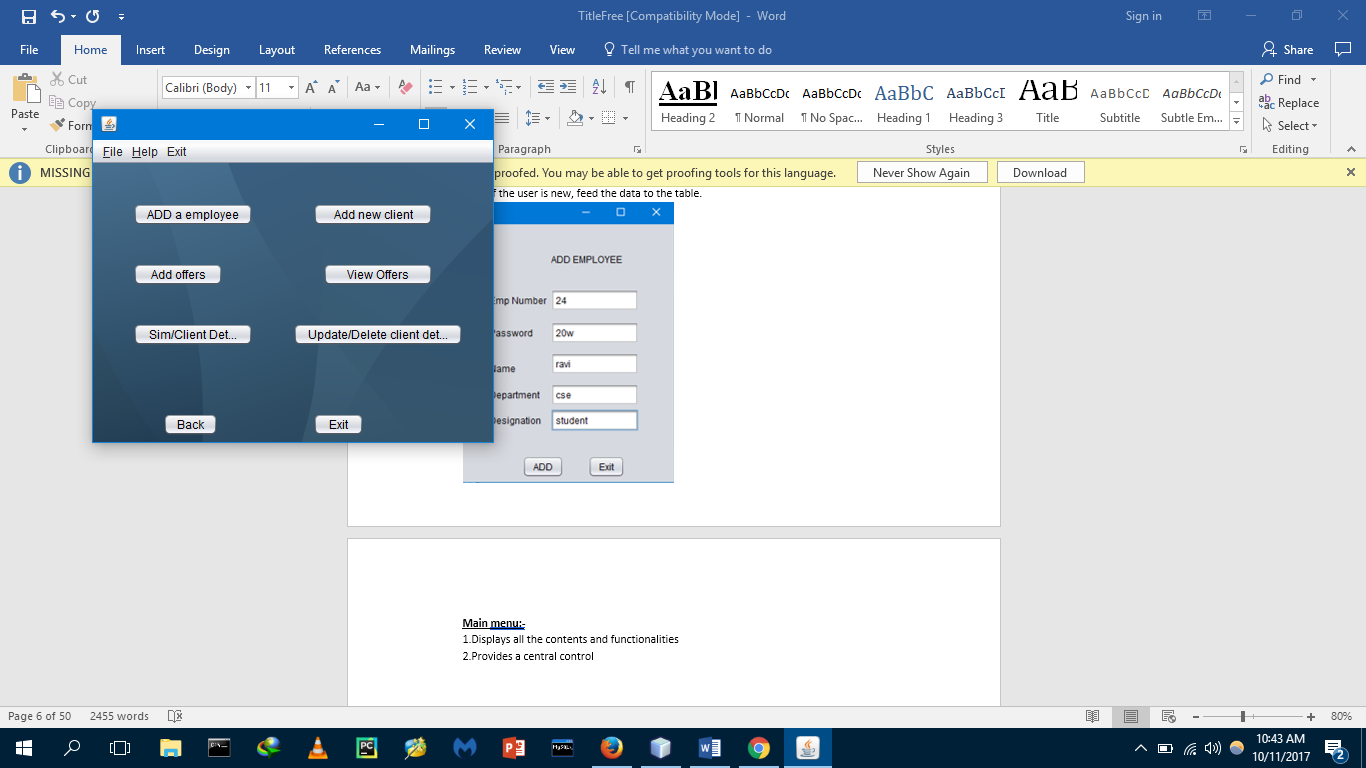
e.printStackTrace();

}

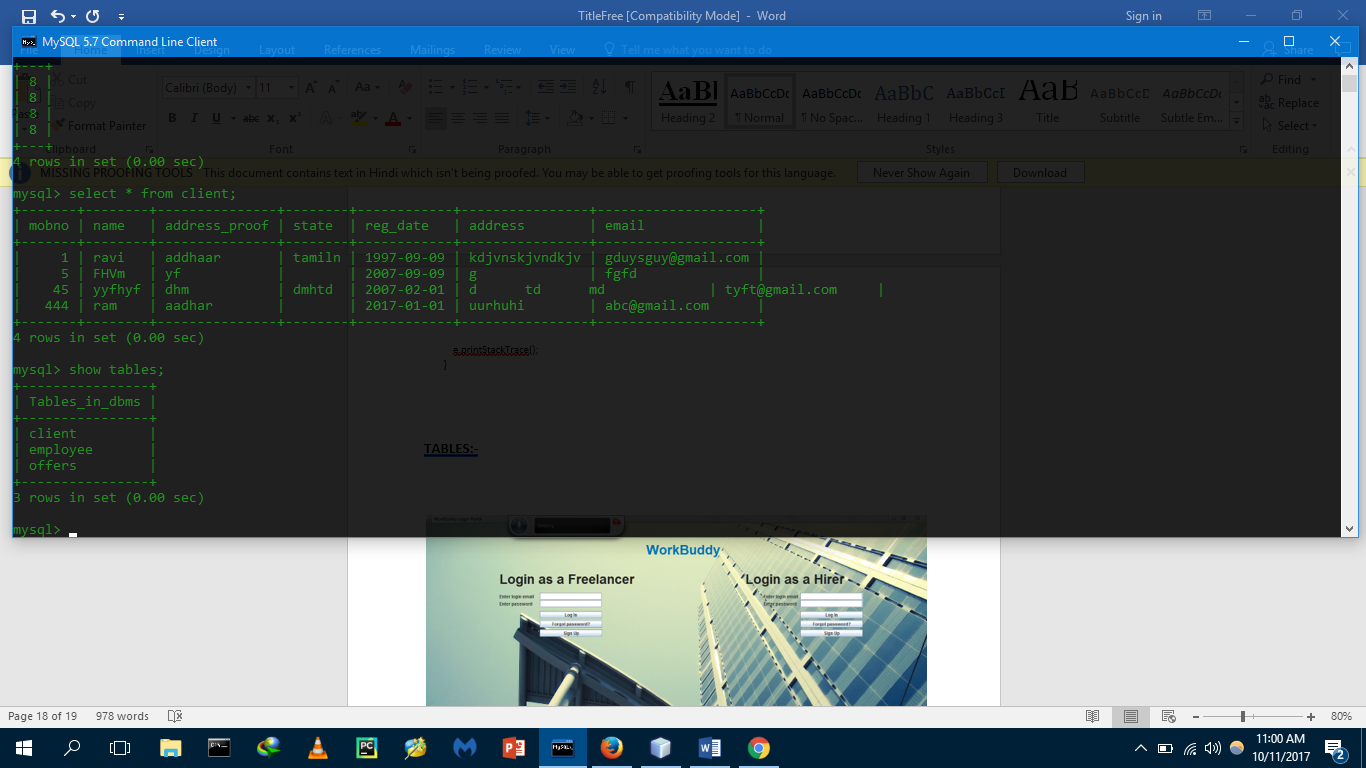
**OUTPUT SCREENSHOTS:-**



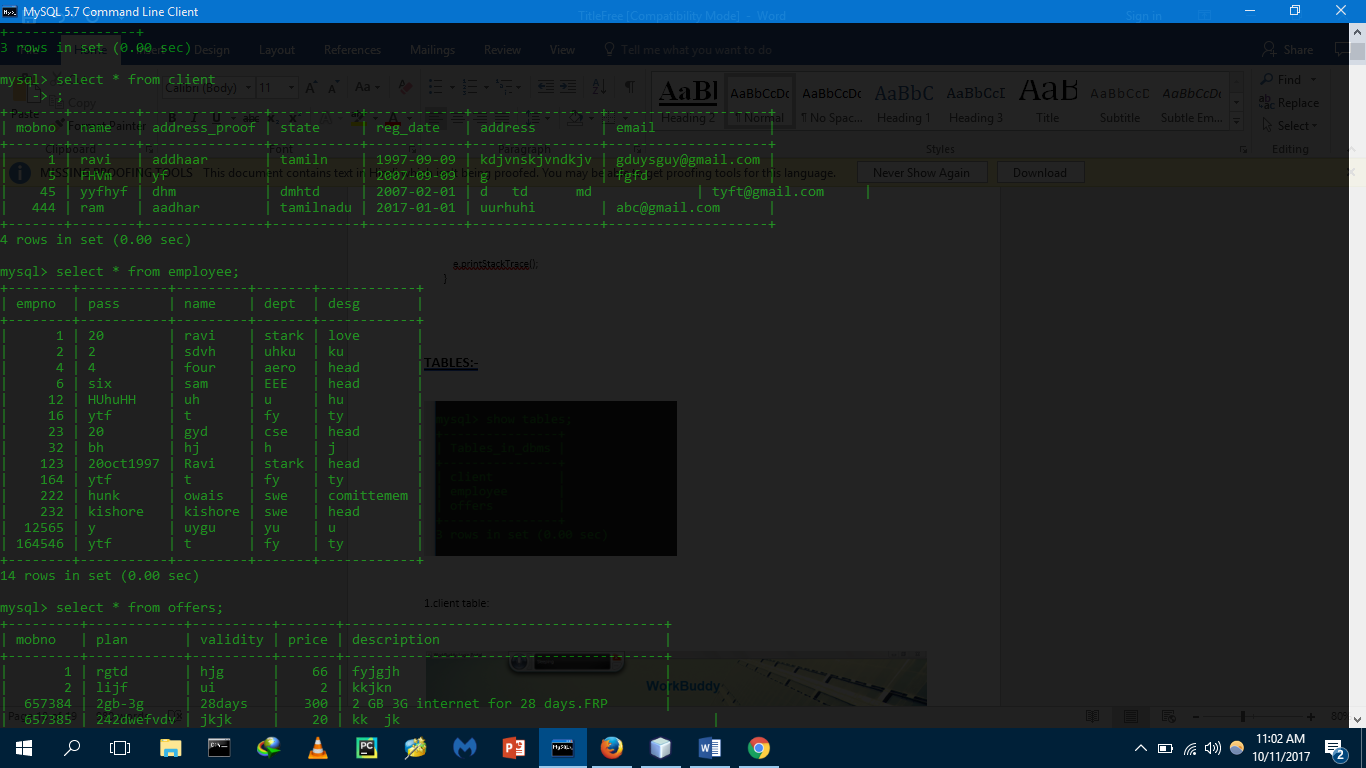




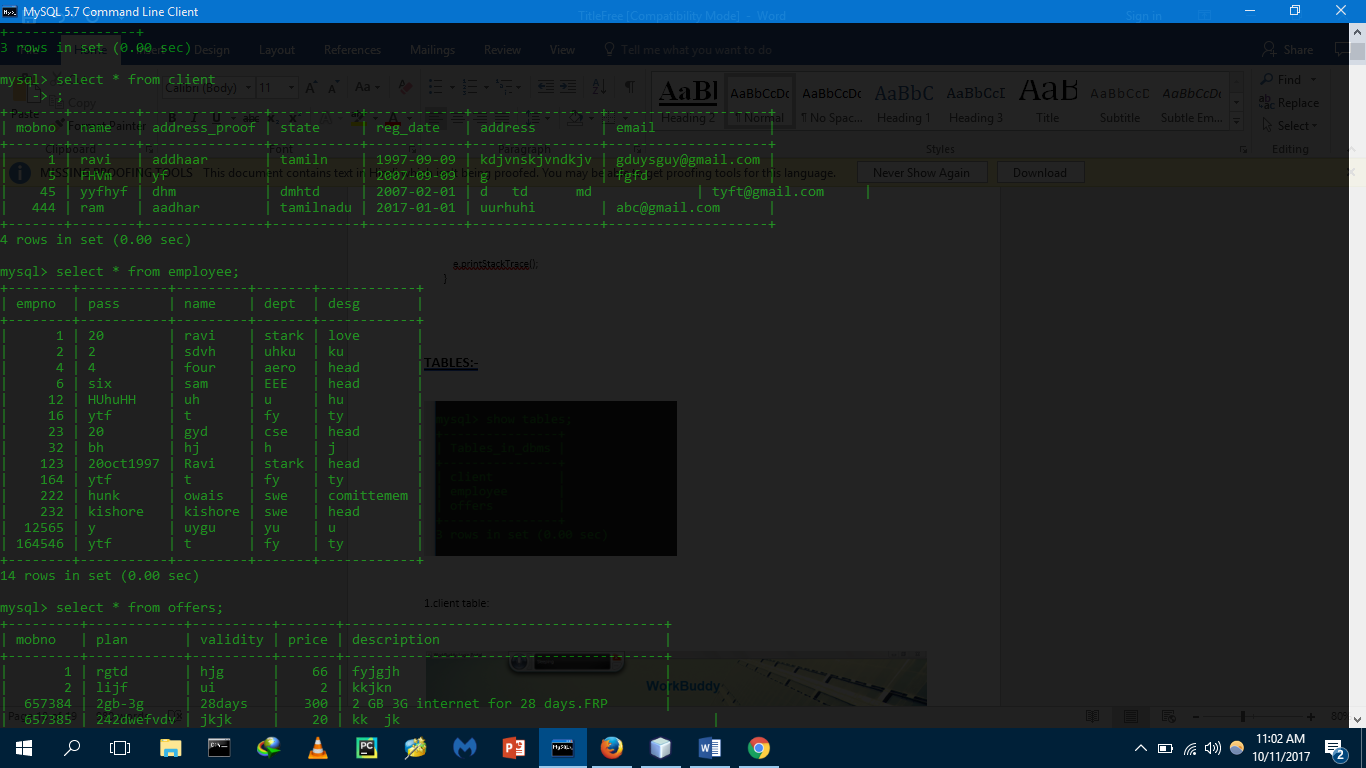
**TABLES:-**



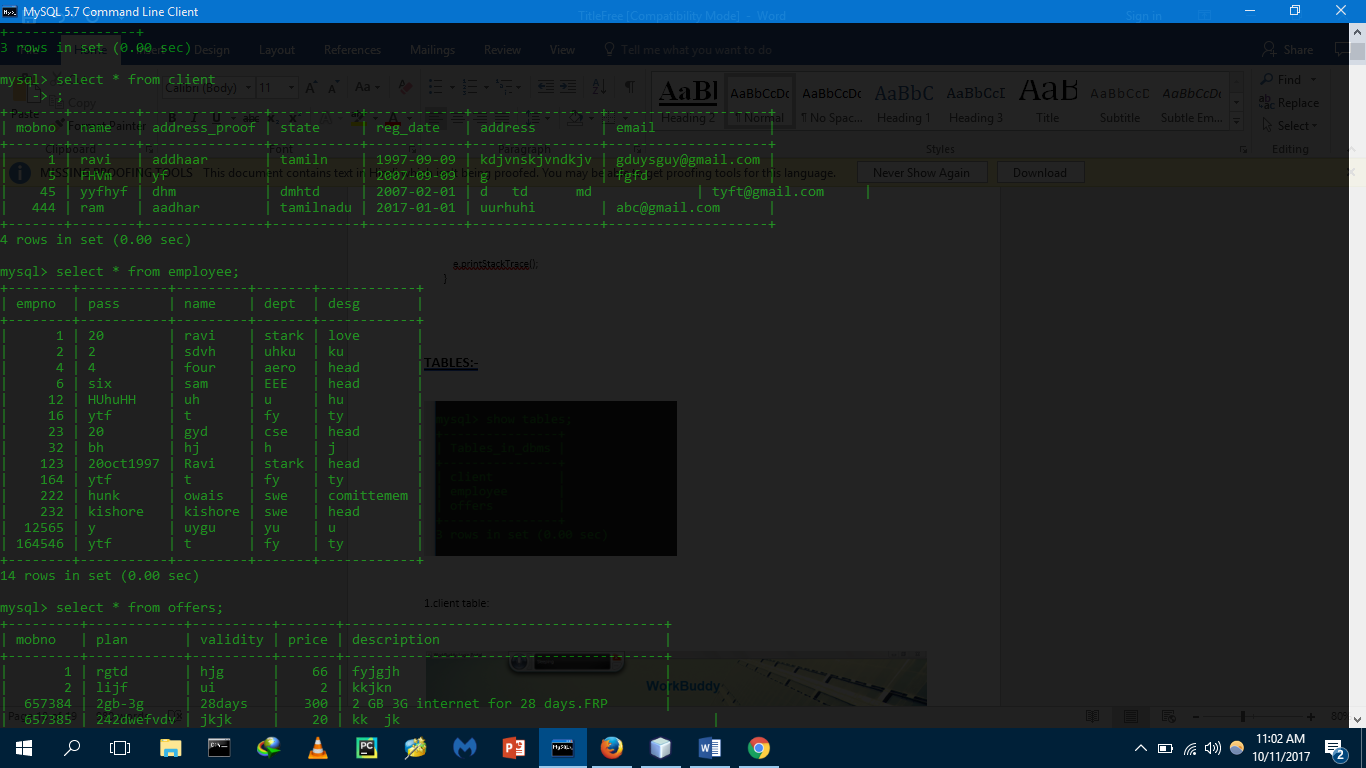
1.client table:



2.Employee table:



Offers table:



**CONCLUSION:**

In all, this telecommunication app is an excellent example to see how the data is stored in a relational database with the Login feature.

This project report contains:

• Registering a new user

• Checking the existence of a user

• Checking the repetition of user

• Authentication using the user login credentials

• Forgot password functionality

This app also stores simple day to day tasks and enables the customer care executives to do their task efficiently.

The features are as followed:

• See offers of a new client

• Add a new customer and his/her details

• Add new offers for customers

• Update customer details

• See Customer details

This project has many day to day applications, and can be evolved to serve many other purposes such as task status, reminder updates, etc.