DATABASE SYSTEMS LABORATORY

10B17CI372

LABORATORY MANUAL

III SEMESTER B. Tech. CSE , IT & BI

ACADEMIC YEAR: 2015 –2016 (ODD)

Department of Computer Science and Engineering

Jaypee University of Information Technology,Waknaghat

HARDWARE REQUIREMENTS:

INTEL PENTIUM 915 GV
80GB HDD
512MB DDR

SOFTWARE REQUIREMENTS:

ORACLE 8i,9i. MY SQL, DB2.

EVALUATION SCHEME:

1.	Mid Term Exam (Viva and Written Exam):	10
2.	End term Exam (Viva and Written Exam):	10
3.	Lab Records:	7.5
4.	Regular Assessment (Quality and quantity of experiment performed, Learning laboratory skills):	15
5.	Attendance and discipline in lab:	7.5
	Total:	50

LIST OF EXPERIMENTS

1. Data Definition, Table Creation, Constraints,
2. Insert, Select Commands, Update and Delete Commands.
3. Nested Queries and Join Queries
4. Views
5. High level programming language extensions (Control structures, Procedures and Functions).
6. Front end Tools
7. Forms
8. Triggers
9. Menu Design
10. Reports.
11. Database Design and implementation (Mini Project).

CONTENTS

S. No	LIST OF EXPRIEMENTS	Page No.
1	Data Definition Language Commands	5
2	Data Manipulation Language Commands	10
3	Data Control Language, Transfer Control Language Commands	17
4	In Built Functions	20
5	Nested Queries And Join Queries	26
6	Set operators	31
7	Views	33
8	Control Structure	36
9	Procedure and Function	46
10	Trigger	56
11	Front End Tools	61
12	Form	62
13	Menu Design	70
14	Report Generation	72
15	Database Design And Implementation Payroll Processing	74
Beyond the Syllabus		
16	Banking System (mini Project)	78

Exercise Number: 1

Title of the Exercise : DATA DEFINITION LANGUAGE (DDL) COMMANDS

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To practice and implement data definition language commands and constraints.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**DDL COMMAND** It is used to communicate with database. DDL is used to: o Create an object o Alter the structure of an object o To drop the object created.
2	The commands used are: Create, Alter, Drop, Truncate
3	**INTEGRITY CONSTRAINT** An integrity constraint is a mechanism used by oracle to prevent invalid data entry into the table. It has enforcing the rules for the columns in a table. The types of the integrity constraints are: a) Domain Integrity b) Entity Integrity c) Referential Integrity
4	**a) Domain Integrity** This constraint sets a range and any violations that take place will prevent the user from performing the manipulation that caused the breach. It includes: **Not Null constraint:** While creating tables, by default the rows can have null value .the enforcement of not null constraint in a table ensure that the table contains values. **Principle of null values:** o Setting null value is appropriate when the actual value is unknown, or when a value would not be meaningful. o A null value is not equivalent to a value of zero. o A null value will always evaluate to null in any expression. o When a column name is defined as not null, that column becomes a mandatory i.e., the user has to enter data into it. o Not null Integrity constraint cannot be defined using the alter table command when the table contain rows.
5	**Check Constraint:** Check constraint can be defined to allow only a particular range of values .when the manipulation violates this constraint, the record will be rejected. Check condition cannot contain sub queries.

6	**b) Entity Integrity** Maintains uniqueness in a record. An entity represents a table and each row of a table represents an instance of that entity. To identify each row in a table uniquely we need to use this constraint. There are 2 entity constraints: **Unique key constraint** It is used to ensure that information in the column for each record is unique, as with telephone or drivers license numbers. It prevents the duplication of value with rows of a specified column in a set of column. A column defined with the constraint can allow null value. If unique key constraint is defined in more than one column i.e., combination of column cannot be specified. Maximum combination of columns that a composite unique key can contain is 16. **Primary Key Constraint** A primary key avoids duplication of rows and does not allow null values. It can be defined on one or more columns in a table and is used to uniquely identify each row in a table. These values should never be changed and should never be null. A table should have only one primary key. If a primary key constraint is assigned to more than one column or combination of column is said to be composite primary key, which can contain 16 columns.
7	**c) Referential Integrity** It enforces relationship between tables. To establish parent-child relationship between 2 tables having a common column definition, we make use of this constraint. To implement this, we should define the column in the parent table as primary key and same column in the child table as foreign key referring to the corresponding parent entry. **Foreign key** A column or combination of column included in the definition of referential integrity, which would refer to a referenced key. **Referenced key** It is a unique or primary key upon which is defined on a column belonging to the parent table.

c) SQL Commands:

CREATE TABLE

It is used to create a table

Syntax: Create table tablename (column_name1 data_ type constraints, column_name2 data_ type constraints …)

Example:

Create table Emp (EmpNo number(5), EName VarChar(15), Job Char(10) constraint un unique, DeptNo number(3) CONSTRAINT FKey2 REFERENCES DEPT(DeptNo));

Create table stud (sname varchar2(20) not null, rollno number(10) not null,dob date not null);

Rules:

1. Oracle reserved words cannot be used.
3. Underscore, numerals, letters are allowed but not blank space.
3. Maximum length for the table name is 30 characters.
4. 2 different tables should not have same name.
5. We should specify a unique column name.
6. We should specify proper data type along with width.
7. We can include “not null” condition when needed. By default it is „null‟.

ALTER TABLE

Alter command is used to:

1. Add a new column.
3. Modify the existing column definition.

3. To include or drop integrity constraint.

Syntax: alter table tablename add/modify (attribute datatype(size));

Example:

1. Alter table emp add (phone_no char (20));
2. Alter table emp modify(phone_no number (10));
3. ALTER TABLE EMP ADD CONSTRAINT Pkey1 PRIMARY KEY (EmpNo);

DROP TABLE

It will delete the table structure provided the table should be empty.

Example: drop table prog20; Here prog20 is table name

TRUNCATE TABLE

If there is no further use of records stored in a table and the structure has to be retained then the records alone can be deleted.

Syntax: TRUNCATE TABLE <TABLE NAME>;

Example: Truncate table stud;

DESC

This is used to view the structure of the table.

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)

DOMAIN INTEGRITY

Example: Create table cust(custid number(6) not null, name char(10));
Alter table cust modify (name not null);

CHECK CONSTRAINT

Example: Create table student (regno number (6), mark number (3) constraint b check (mark >=0 and mark <=100));
Alter table student add constraint b2 check (length(regno<=4));

ENTITY INTEGRITY

a) Unique key constraint

Example: Create table cust(custid number(6) constraint uni unique, name char(10));
Alter table cust add(constraint c unique(custid));

b) **Primary Key Constraint**

Example: Create table stud(regno number(6) constraint primary key, name char(20));

d) 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:

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

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 ch 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.
SQL> drop table dept;
Table dropped.
```

e) Result:

Thus the data definition language commands was performed and implemented successfully

QUESTIONS AND ANSWERS

1. **Define the terms**

   DDL:

   Data base schema is specified by a set of definitions expressed by a special language called a data definition language.

2. **What are the categories of SQL command?**

   SQL commands are divided in to the following categories:

   Data Delimitation language

   Data manipulation language

   Data control language

   Transaction Control Language

3. **What is integrity constraint?**

   An integrity constraint is a mechanism used by oracle to prevent invalid data entry into the table. It has enforcing the rules for the columns in a table.

4. **List the types of constraint.**

   a) Domain Integrity

   b) Entity Integrity

   c) Referential Integrity

5. **Primary Key Constraint**

   A primary key avoids duplication of rows and does not allow null values. It can be defined on one or more columns in a table and is used to uniquely identify each row in a table. These values should never be changed and should never be null.

6. **Referential Integrity**

   It enforces relationship between tables. To establish parent-child relationship between 2 tables having a common column definition, we make use of this constraint. To implement this, we should define the column in the parent table as primary key and same column in the child table as foreign key referring to the corresponding parent entry.

Exercise Number: 2

Title of the Exercise : DATA MANIPULATION LANGUAGE (DML) COMMANDS
Date of the Exercise :

AIM OF THE EXPERIMENT

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

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**DML COMMAND** 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
2	**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.
3	**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.
4	**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.
5	**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.

c) SQL Commands:

INSERT COMMAND

Inserting a single row into a table:

Syntax: insert into <table name> values (value list)
Example: insert into s values(„s3",‟‟sup3‟‟,‟‟blore‟‟,10)

Inserting more than one record using a single insert commands:

Syntax: insert into <table name> values (&col1, &col2, ….)
Example: Insert into stud values(&reg, „&name‟‟, &percentage);

Skipping the fields while inserting:

Insert into <tablename(coln names to which datas to b inserted)> values (list of values);
Other way is to give null while passing the values.

SELECT COMMANDS

Selects all rows from the table

Syntax: Select * from tablename;

Example; Select * from IT;

The retrieval of specific columns from a table:

It retrieves the specified columns from the table

Syntax: Select column_name1, …..,column_namen from table name;

Example: Select empno, empname from emp;

Elimination of duplicates from the select clause:

It prevents retriving the duplicated values .Distinct keyword is to be used.

Syntax: Select DISTINCT col1, col2 from table name;

Example: Select DISTINCT job from emp;

Select command with where clause:

To select specific rows from a table we include „where" clause in the select command. It can appear only after the „from" clause.

Syntax: Select column_name1, …..,column_namen from table name where condition;

Example: Select empno, empname from emp where sal>4000;

Select command with order by clause:

Syntax: Select column_name1, …..,column_namen from table name where condition order by colmnname;

Example: Select empno, empname from emp order by empno;

Select command to create a table:

Syntax: create table tablename as select * from existing_tablename;

Example: create table emp1 as select * from emp;

Select command to insert records:

Syntax: insert into tablename (select columns from existing_tablename);

Example: insert into emp1 (select * from emp);

UPDATE COMMAND

Syntax:update tablename set field=values where condition;

Example: Update emp set sal = 10000 where empno=135;

DELETE COMMAND

Syntax: Delete from table where conditions;

Example:delete from emp where empno=135;

d) Queries:

Q1: Insert a single record into dept table.

Solution:

1.Decide the data to add in dept.

2.Add to dept one row at a time using the insert into syntax.

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
         2 Arjun              ASP                   2      15000
         3 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.
```

Q6: Delete only those who are working as lecturer

Ans:

```
SQL> select * from emp;
     EMPNO ENAME    JOB              DEPTNO       SAL
---------- -------------------- ------------- ---------- ----------
         1 Mathi                AP                1      10000
         2 Arjun                ASP               2      15000
         3 Gugan                ASP               1      15000
         4 Karthik              Prof              2      30000
         5 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
         2 Arjun                ASP               2      15000
         3 Gugan                ASP               1      15000
         4 Karthik              Prof              2      30000
         5 Akalya               AP                1      10000
```

Q7: 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
         2 Arjun                ASP               2      15000
         3 Gugan                ASP               1      15000
         4 Karthik              Prof              2      30000
```

Q8: 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
         2 Arjun                ASP               2      15000
         3 Gugan                ASP               1      15000
```

1	Mathi	AP	1	10000
5	Akalya	AP	1	10000

Q9: Display only those employees whose deptno is 30.

Solution:

1. 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

Q10: 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

e) Result:

Thus the DML commands using from where clause was performed successfully and executed.

QUESTIONS AND ANSWERS

1. **What is DML?**
   DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects.
2. **What are DML command?**
   Some of the commands are Insert, Select, Update, Delet
3. **Give the general form of SQL Queries? Select**
   A1, A2……., An
   From R,1R2……, R
   m Where P
4. **What is the use of rename operation?**
   Rename operation is used to rename both relations and an attributes. It uses the as clause, taking the form: Old-name as new-name
5. **Define tuple variable?**
   Tuple variables are used for comparing two tuples in the same relation. The tuple variables are defined in the from clause by way of the as clause.
6. **Write the syntax to retrieve specific columns from a table:**

Syntax: Select column_name1, …..,column_namen from table name;

Exercise Number: 3

Title of the Exercise : DATA CONTROL LANGUAGE (DCL), TRANSACTION CONTROL LANGUAGE (TCL) COMMANDS

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To study the various data language commands (DCL, TCL) and implements them on the database.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**DCL COMMAND** The DCL language is used for controlling the access to the table and hence securing the database. DCL is used to provide certain privileges to a particular user. Privileges are rights to be allocated.
2	The privilege commands are namely, Grant and Revoke
3	The various privileges that can be granted or revoked are, Select Insert Delete Update References Execute All
4	**GRANT COMMAND:** It is used to create users and grant access to the database. It requires database administrator (DBA) privilege, except that a user can change their password. A user can grant access to their database objects to other users.
5	**REVOKE COMMAND:** Using this command , the DBA can revoke the granted database privileges from the user.
6	**TCL COMMAND** **COMMIT:** command is used to save the Records. **ROLL BACK:** command is used to undo the Records. **SAVE POINT** command is used to undo the Records in a particular transaction.

c)SQL Commands

DCL Commands

GRANT COMMAND

Grant < database_priv [database_priv…..] > to <user_name> identified by <password> [,<password…..];

Grant <object_priv> | All on <object> to <user | public> [With Grant Option];

REVOKE COMMAND

Revoke <database_priv> from <user [, user] >;

Revoke <object_priv> on <object> from < user | public >;

<database_priv> -- Specifies the system level priveleges to be granted to the users or roles. This includes create / alter / delete any object of the system.

<object_priv> -- Specifies the actions such as alter / delete / insert / references / execute / select / update for tables.

<all> -- Indicates all the priveleges.

[With Grant Option] – Allows the recipient user to give further grants on the objects.

The priveleges can be granted to different users by specifying their names or to all users by using the "Public" option.

TCL COMMANDS:

Syntax:

SAVEPOINT: SAVEPOINT <SAVE POINT NAME>;

ROLLBACK: ROLL BACK <SAVE POINT NAME>;

COMMIT: Commit;

d) Queries:

Tables Used:

Consider the following tables namely "DEPARTMENTS" and "EMPLOYEES"

Their schemas are as follows ,

Departments (dept _no , dept_ name , dept_location);

Employees (emp_id , emp_name , emp_salary);

Q1: Develop a query to grant all privileges of employees table into departments table

Ans:

```
SQL> Grant all on employees to departments;
Grant succeeded.
```

Q2: Develop a query to grant some privileges of employees table into departments table

Ans:

```
SQL> Grant select, update , insert on departments to departments with grant option;
Grant succeeded.
```

Q3: Develop a query to revoke all privileges of employees table from departments table

Ans:

```
SQL> Revoke all on employees from departments;
Revoke succeeded.
```

Q4: Develop a query to revoke some privileges of employees table from departments table

Ans:

```
SQL> Revoke select, update , insert on departments from departments;
Revoke succeeded.
```

Q5: Write a query to implement the save point

Ans:

```
SQL> SAVEPOINT S1;
Savepoint created.

SQL> select * from emp;
    EMPNO ENAME   JOB             DEPTNO       SAL
---------- -------------------- ------------- ---------- ----------
         1 Mathi              AP              1      10000
         2 Arjun              ASP             2      15000
         3 Gugan              ASP             1      15000
         4 Karthik            Prof            2      30000

SQL> INSERT INTO EMP VALUES(5,'Akalya','AP',1,10000);
1 row created.
```

```
SQL> select * from emp;
    EMPNO ENAME   JOB            DEPTNO       SAL
---------- -------------------- ------------ ---------- ----------
         1 Mathi                AP                    1      10000
         2 Arjun                ASP                   2      15000
         3 Gugan                ASP                   1      15000
         4 Karthik              Prof                  2      30000
         5 Akalya               AP                    1      10000
```

Q6: Write a query to implement the rollback

Ans:

```
SQL> rollback s1;
SQL> select * from emp;
    EMPNO ENAME   JOB            DEPTNO       SAL
---------- -------------------- ------------ ---------- ----------
         1 Mathi                AP                    1      10000
         2 Arjun                ASP                   2      15000
         3 Gugan                ASP                   1      15000
         4 Karthik              Prof                  2      30000
```

Q7: Write a query to implement the commit

Ans:

```
SQL> COMMIT;
Commit complete.
```

e)Result

The DCL,TCL commands was performed successfully and executed.

QUESTIONS AND ANSWERS

1. **Define DCL?**

   The DCL language is used for controlling the access to the table and hence securing the database. DCL is used to provide certain privileges to a particular user. Privileges are rights to be allocated.

2. **List the DCL commands used in data bases**

   The privilege commands are namely, Grant and Revoke

3. **What type of privileges can be granted?**

   The various privileges that can be granted or revoked are,

   - ✓ Select
   - ✓ Insert
   - ✓ Delete
   - ✓ Update
   - ✓ References
   - ✓ Execute
   - ✓ All

4. **Write the syntax for grant command**

   Grant < database_priv [database_priv…..] > to <user_name> identified by <password> [,<password…..];

   Grant <object_priv> | All on <object> to <user | public> [With Grant Option];

5. **What are TCL commands?**

   *Commit *Rollback *save point

Exercise Number: 4

Title of the Exercise : IN BUILT FUNCTIONS

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To perform nested Queries and joining Queries using DML command.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Function is a group of code that accepts zero or more arguments and both return one or more results. Both are used to manipulate individual data items. Operators differ from functional in that they follow the format of function name (arg..). An argument is a user defined variables or constants. Most operators accept at most 2 arguments while the structure of functions permit to accept 3 or more arguments. Function can be classifies into **single row function and group functions**.
2	**Single Row functions** A single row function or scalar function returns only one value for every row queries in table. Single row function can appear in a select command and can also be included in a where clause. The single row function can be broadly classified as, o Date Function o Numeric Function o Character Function o Conversion Function o Miscellaneous Function The example that follows mostly uses the symbol table "**dual**". It is a table, which is automatically created by oracle along with the data dictionary.
3	**Date Function** They operate on date values and produce outputs, which also belong to date data type except for months, between, date function returns a number.
4	**Group Functions** A group function returns a result based on group of rows

c) SQL Commands:

DATE FUNCTION

1. Add_month

This function returns a date after adding a specified date with specified number of months.

Syntax: Add_months(d,n); where d-date n-number of months

Example: Select add_months(sysdate,2) from dual;

2. last_day

It displays the last date of that month.

Syntax: last_day (d); where d-date

Example: Select last_day („1-jun-2009") from dual;

3. Months_between

It gives the difference in number of months between d1 & d2.

Syntax: month_between (d1,d2); where d1 & d2 -dates

Example: Select month_between („1-jun-2009","1-aug-2009") from dual;

4. next_day

It returns a day followed the specified date.

Syntax: next_day (d,day);

Example: Select next_day (sysdate,"wednesday") from dual

5. round

This function returns the date, which is rounded to the unit specified by the format model.

Syntax : round (d,[fmt]);

where d- date, [fmt] – optional. By default date will be rounded to the nearest day

Example: Select round (to_date(„1-jun-2009","dd-mm-yy"),"year") from dual; Select round („1-jun-2009","year") from dual;

NUMERICAL FUNCTIONS

Command	Query	Output
Abs(n)	Select abs(-15) from dual;	15
Ceil(n)	Select ceil(55.67) from dual;	56
Exp(n)	Select exp(4) from dual;	54.59
Floor(n)	Select floor(100.2) from dual;	100
Power(m,n)	Select power(4,2) from dual;	16
Mod(m,n)	Select mod(10,3) from dual;	1
Round(m,n)	Select round(100.256,2) from dual;	100.26
Trunc(m,n)	Select trunc(100.256,2) from dual;	100.23
Sqrt(m,n)	Select sqrt(16) from dual;	4

CHARACTER FUNCTIONS

Command	Query	Output
initcap(char);	select initcap("hello") from dual;	Hello
lower (char);	select lower („HELLO") from dual;	hello
upper (char);	select upper („hello") from dual;	HELLO
ltrim (char,[set]);	select ltrim („cseit", „cse") from dual; select	it
rtrim (char,[set]);	rtrim („cseit", „it") from dual;	cse
replace (char,search string, replace string);	select replace(„jack and jue",„j",„bl") from dual;	black and blue
substr (char,m,n);	select substr („information", 3, 4) from dual;	Form

CONVERSION FUNCTION

1. to_char()

Syntax: to_char(d,[format]);

This function converts date to a value of varchar type in a form specified by date format. If format is negelected then it converts date to varchar2 in the default date format.

Example: select to_char (sysdate, "dd-mm-yy") from dual;

2. to_date()

Syntax: to_date(d,[format]);

This function converts character to date data format specified in the form character.

Example: select to_date(„aug 15 2009","mm-dd-yy") from dual;

Miscellaneous Functions

1. uid – This function returns the integer value (id) corresponding to the user currently logged in.

Example: select uid from dual;

2. user – This function returns the logins user name.

Example: select user from dual;

3. nvl – The null value function is mainly used in the case where we want to consider null values as zero.

Syntax; nvl(exp1, exp2)

If exp1 is null, return exp2. If exp1 is not null, return exp1.

Example: select custid, shipdate, nvl(total,0) from order;

4. vsize: It returns the number of bytes in expression.

Example: select vsize(„tech") from dual;

GROUP FUNCTIONS

A group function returns a result based on group of rows.

1. avg - Example: select avg (total) from student;

2. max - Example: select max (percentagel) from student;

2.min - Example: select min (marksl) from student;

4. sum - Example: select sum(price) from product;

COUNT FUNCTION

In order to count the number of rows, count function is used.

1. count(*) – It counts all, inclusive of duplicates and nulls.

Example: select count(*) from student;

2. count(col_name)– It avoids null value.

Example: select count(total) from order;

2. count(distinct col_name) – It avoids the repeated and null values.

Example: select count(distinct ordid) from order;

GROUP BY CLAUSE

This allows us to use simultaneous column name and group functions.

Example: Select max(percentage), deptname from student group by deptname;

HAVING CLAUSE

This is used to specify conditions on rows retrieved by using group by clause.

Example: Select max(percentage), deptname from student group by deptname having count(*)>=50;

SPECIAL OPERATORS:

In / not in – used to select a equi from a specific set of values

Any - used to compare with a specific set of values

Between / not between – used to find between the ranges

Like / not like – used to do the pattern matching

d) Queries:

Q1: **Display all the details of the records whose employee name starts with „A'.**

Solution:

1. Use SELECT FROM WHERE syntax. 2. select should include all in the given format.

3. from should include employee 4. where should include condition on empname like „A%".

Ans:

```
SQL> select * from emp where ename like 'A%';
     EMPNO ENAME              JOB              DEPTNO       SAL
---------- -------------------- ------------- ---------- ----------
```

```
2 Arjun          ASP          2     15000
5 Akalya         AP           1     10000
```

Q2: Display all the details of the records whose employee name does not starts with ‗A'.
Ans:

```
SQL> select * from emp where ename not like 'A%';


 EMPNO ENAME            JOB            DEPTNO       SAL
---------- -------------------- ------------- ---------- ----------
     1 Mathi          AP           1     10000
     3 Gugan          ASP          1     15000
     4 Karthik        Prof         2     30000
```

Q3: Display the rows whose salary ranges from 15000 to 30000.
Ans:

```
SQL> select * from emp where sal between 15000 and 30000;

   EMPNO ENAME            JOB            DEPTNO       SAL
---------- -------------------- ------------- ---------- ----------
     2 Arjun          ASP          2     15000
     3 Gugan          ASP          1     15000
     4 Karthik        Prof         2     30000
```

Q4: Calculate the total and average salary amount of the emp table.
Ans:

```
SQL> select sum(sal),avg(sal) from emp;
  SUM(SAL)   AVG(SAL)
---------- ----------
    80000      16000
```

Q5: **Count the total records in the emp table.**
Ans:

```
  SQL>select * from emp;
 EMPNO ENAME            JOB            DEPTNO       SAL
------ -------------------- ------------- ---------- ----------
   1 Mathi          AP           1     10000
   2 Arjun          ASP          2     15000
   3 Gugan          ASP          1     15000
   4 Karthik        Prof         2     30000
   5 Akalya         AP           1     10000

SQL> select count(*) from emp;
  COUNT(*)
---------
       5
```

Q6: Determine the max and min salary and rename the column as max_salary and min_salary.
Solution:
1. Use the MIN & MAX aggregate function in select clause.
2. Rename the column as min_sal & max_sal.
Ans:

```
SQL> select max(sal) as max_salary, min(sal) as min_salary from emp;
MAX_SALARY MIN_SALARY
----------          ----------
    30000                10000
```

Q7: Display the month between ―1-jun-10‖and 1-aug-10 in full.
Ans:

SQL>Select month_between („1-jun-2010‟,‟1-aug-2010‟) from dual;

Q8: Display the last day of that month in ―05-Oct-09‖.

Ans:

```
SQL> Select last_day ('1-jun-2009') from dual;
LAST_DAY(
---------
30-JUN-09
```

Q9: Find how many job titles are available in employee table.

Solution:

1. Use select from clause.
2. Use count function to get the result.

Ans:

```
SQL> select count(job) from emp;
COUNT(JOB)
----------
         4
SQL> select count(distinct job) from emp;
COUNT(DISTINCTJOB)
------------------
                 2
```

Q10: What is the difference between maximum and minimum salaries of employees in the organization?

Solution:

1. Use select from clause.
2. Use function max(),min() and find the difference between them to get the result.

Ans:

```
SQL> select max(sal), min(sal) from emp;
MAX(SAL)   MIN(SAL)
---------- ----------
    20000      10000
```

d) Result:

Thus the nested Queries and join Queries was performed successfully and executed.

QUESTIONS AND ANSWERS

1. **Define function?**

   Function is a group of code that accepts zero or more arguments and both return one or more results. Both are used to manipulate individual data items.

2. **Write the two types of functions**

   i. Single row functions
   ii. Group functions

3. **What are single row functions?**

   A single row function or scalar function returns only one value for every row queries in table. Single row function can appear in a select command and can also be included in a where clause. The single row function can be broadly classified as,

   o Date Function
   o Numeric Function
   o Character Function
   o Conversion Function
   o Miscellaneous Function

4. **List some character funcitons**

initcap(char);

lower (char);

upper (char);

ltrim (char,[set]); rtrim (char,[set]);

Exercise Number: 5

Title of the Exercise : NESTED QUERIES AND JOIN QUERIES

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To perform nested Queries and joining Queries using DML command.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**Nested Queries:** Nesting of queries one within another is known as a nested queries. **Sub queries** The query within another is known as a sub query. A statement containing sub query is called parent statement. The rows returned by sub query are used by the parent statement.
2	**Types** **1. Sub queries that return several values** Sub queries can also return more than one value. Such results should be made use along with the operators in and any. **2. Multiple queries** Here more than one sub query is used. These multiple sub queries are combined by means of „and" & „or" keywords **3. Correlated sub query** A sub query is evaluated once for the entire parent statement whereas a correlated Sub query is evaluated once per row processed by the parent statement.
3	**Relating Data through Join Concept** The purpose of a join concept is to combine data spread across tables. A join is actually performed by the „where" clause which combines specified rows of tables. Syntax; select columns from table1, table2 where logical expression; **Types of Joins** 1. Simple Join 2. Self Join 3. Outer Join 4. Inner Join
4	1. Simple Join **a) Equi-join:** A join, which is based on equalities, is called equi-join. **b) Non Equi-join:** It specifies the relationship between **Table Aliases** Table aliases are used to make multiple table queries shorted and more readable. We give an alias name to the table in the „from" clause and use it instead of the name throughout the query.
5	**Self join:** Joining of a table to itself is known as self-join. It joins one row in a table to another. It can compare each row of the table to itself and also with other rows of the same table.
6	**Outer Join:** It extends the result of a simple join. An outer join returns all the rows returned by simple join as well as those rows from one table that do not match any row from the table. The symbol (+) represents outer join. **Inner join:** Inner join returns the matching rows from the tables that are being joined

c) SQL Commands:

Nested Queries:

Example: select ename, eno, address where salary >(select salary from employee where ename ="jones");

1.Subqueries that return several values

Example: select ename, eno, from employee where salary <any (select salary from employee where deptno =10");

3.Correlated subquery

Example: select * from emp x where x.salary > (select avg(salary) from emp where deptno =x.deptno);

Simple Join

a) Equi-join

Example: select * from item, cust where item.id=cust.id;

b) Non Equi-join

Example: select * from item, cust where item.id<cust.id;

Self join

Example: select * from emp x ,emp y where x.salary >= (select avg(salary) from x.emp where x. deptno =y.deptno);

Outer Join

Example: select ename, job, dname from emp, dept where emp.deptno (+) = dept.deptno;

d) Queries:

Q1: Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with „M'.

Solution:

1. Use select from clause.
2. Use like operator to match job and in select clause to get the result.

Ans:

```
SQL> select ename,sal from emp where sal>(select min(sal) from emp where job like 'A%');
ENAME                SAL
-------------------- ----------
Arjun                12000
Gugan                20000
Karthik              15000
```

Q2: Issue a query to find all the employees who work in the same job as Arjun.

Ans:

```
SQL> select * from emp;
    EMPNO ENAME              JOB          DEPTNO       SAL
---------- -------------------- ---------- ---------- ----------
         1 Mathi              AP               1      10000
         2 Arjun              ASP              2      12000
         3 Gugan               ASP              2      20000
         4 Karthik            AP               1      15000
SQL> select ename from emp where job=(select job from emp where ename='Arjun');
ENAME
--------------
Arjun
Gugan
```

Q3: Issue a query to display information about employees who earn more than any employee in dept 1.
Ans:

```
SQL> select * from emp where sal>(select max(sal) from emp where empno=1);

    EMPNO ENAME              JOB          DEPTNO       SAL
---------- -------------------- ---------- ---------- ----------
         2 Arjun                ASP               2      12000
         3 Gugan                 ASP               2      20000
         4 Karthik              AP                1      15000
```

JOINS

Tables used

```
SQL> select * from emp;

    EMPNO ENAME              JOB          DEPTNO       SAL
---------- -------------------- ---------- ---------- ----------
         1 Mathi                AP                1      10000
         2 Arjun                ASP               2      12000
         3 Gugan                 ASP               2      20000
         4 Karthik              AP                1      15000

SQL> select * from dept;

   DEPTNO DNAME        LOC
---------- -------------- -------------
         1 ACCOUNTING     NEW YORK
         2 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

EQUI-JOIN

Q4: Display the employee details, departments that the departments are same in both the emp and dept.
Solution:
1. Use select from clause. 2. Use equi join in select clause to get the result.
Ans:

```
SQL> select * from emp,dept where emp.deptno=dept.deptno;
   EMPNO ENAME      JOB    DEPTNO   SAL     DEPTNO DNAME          LOC
---------- ------------------ ---------- ---------- ---------- ----------  -------------- -------------
         1 Mathi              AP                1      10000          1  ACCOUNTING     NEW YORK
         2 Arjun              ASP               2      12000          2   RESEARCH       DALLAS
         3 Gugan               ASP               2      20000          2 RESEARCH       DALLAS
         4 Karthik            AP                1      15000          1  ACCOUNTING     NEW YORK
```

NON-EQUIJOIN

Q5: Display the employee details, departments that the departments are not same in both the emp and dept.
Solution:
1.Use select from clause. 2. Use non equi join in select clause to get the result.

Ans:

SQL> select * from emp,dept where emp.deptno!=dept.deptno;

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
2	Arjun	ASP	2	12000	1	ACCOUNTING	NEW YORK
3	Gugan	ASP	2	20000	1	ACCOUNTING	NEW YORK
1	Mathi	AP	1	10000	2	RESEARCH	DALLAS

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
4	Karthik	AP	1	15000	2	RESEARCH	DALLAS
1	Mathi	AP	1	10000	30	SALES	CHICAGO
2	Arjun	ASP	2	12000	30	SALES	CHICAGO

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
3	Gugan	ASP	2	20000	30	SALES	CHICAGO
4	Karthik	AP	1	15000	30	SALES	CHICAGO
1	Mathi	AP	1	10000	40	OPERATIONS	BOSTON

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
2	Arjun	ASP	2	12000	40	OPERATIONS	BOSTON
3	Gugan	ASP	2	20000	40	OPERATIONS	BOSTON
4	Karthik	AP	1	15000	40	OPERATIONS	BOSTON

12 rows selected.

LEFTOUT-JOIN

Tables used

SQL> select * from stud1;

Regno	Name	Mark2	Mark3	Result
101	john	89	80	pass
102	Raja	70	80	pass
103	Sharin	70	90	pass
104	sam	90	95	pass

SQL> select * from stud2;

NAME	GRA
john	s
raj	s
sam	a
sharin	a

Q6: Display the Student name and grade by implementing a left outer join.

Ans: SQL> select stud1.name,grade from stud1 left outer join stud2 on stud1.name=stud2.name;

Name	Gra
john	s
raj	s
sam	a
sharin	a

smith null

RIGHTOUTER-JOIN

Q7: Display the Student name, register no, and result by implementing a right outer join.

Ans:

SQL> select stud1.name, regno, result from stud1 right outer join stud2 on stud1.name = stud2.name;

Name	Regno	Result
john	101	pass
raj	102	pass
sam	103	pass
sharin	104	pass

Rollno	Name	Mark1	Mark2	Total
1	sindu	90	95	185
2	arul	90	90	180

FULLOUTER-JOIN

Q8: Display the Student name register no by implementing a full outer join.

Ans:

SQL> select stud1.name, regno from stud1 full outer join stud2 on (stud1.name= stud2.name);

Name	Regno
john	101
raj	102
sam	103
sharin	104

SELFJOIN

Q9: Write a query to display their employee names

Ans:

SQL> select distinct ename from emp x, dept y where x.deptno=y.deptno;

ENAME

Arjun
Gugan
Karthik
Mathi

Q10: Display the details of those who draw the salary greater than the average salary.

Ans:

SQL> select distinct * from emp x where x.sal >= (select avg(sal) from emp);

EMPNO	ENAME	JOB	DEPTNO	SAL
3	Gugan	ASP	2	20000
4	Karthik	AP	1	15000
11	kavitha	designer	12	17000

e) Result:

Thus the nested Queries and join Queries was performed successfully and executed.

QUESTIONS AND ANSWERS

1. **What is the use of sub Queries?**

   A sub Queries is a select-from-where expression that is nested with in another Queries. A common use of sub Queries is to perform tests for set membership, make set comparisons, and determine set cardinality

Exercise Number: 6

Title of the Exercise : SET OPERATORS
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To perform set operations using DML Commands.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**Set Operators:** The Set operator combines the result of 2 queries into a single result. The following are the operators: · Union · Union all · Intersect · Minus
2	**The rules to which the set operators are strictly adhere to :** · The queries which are related by the set operators should have a same number of column and column definition. · Such query should not contain a type of long. · Labels under which the result is displayed are those from the first select statement.

c) SQL commands:

Union: Returns all distinct rows selected by both the queries

Syntax:

Query1 Union Query2;

Union all: Returns all rows selected by either query including the duplicates.

Syntax:

Query1 Union all Query2;

Intersect: Returns rows selected that are common to both queries.

Syntax:

Query1 Intersect Query2;

Minus: Returns all distinct rows selected by the first query and are not by the second

Syntax:

Query1 minus Query2;

d) Queries:

Q1: Display all the dept numbers available with the dept and emp tables avoiding duplicates.

Solution:

1. Use select from clause. 2. Use union select clause to get the result.

Ans:

SQL> select deptno from emp union select deptno from dept;

```
   DEPTNO
----------
         1
         2
        12
        30
        40
```

Q2: **Display all the dept numbers available with the dept and emp tables.**
Solution:
1. Use select from clause. 2. Use union all in select clause to get the result.
Ans:

```
SQL>  select deptno from emp union all select deptno from dept;

   DEPTNO
----------
         1
         2
         2
         1
        12
         1
         2
        30
        40
9 rows selected.
```

Q3: **Display all the dept numbers available in emp and not in dept tables and vice versa.**
Solution:
1. Use select from clause.
2. Use minus in select clause to get the result.
Ans:

```
SQL> select deptno from emp minus select deptno from dept;

   DEPTNO
----------
        12
SQL> select deptno from dept minus select deptno from emp;
   DEPTNO
----------
        30
        40
```

e) Result:

Thus the set operations using DML Commands was successfully performed and executed.

QUESTIONS AND ANSWERS

1. List the set operations of SQL?

1)Union 2)Intersect operation 3)The except operation(minus)

2. Which command returns all distinct rows selected by both the queries?

Union

Exercise Number: 7

Title of the Exercise : VIEWS
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To create and manipulate various database objects of the Table using views

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**Views:** A view is the tailored presentation of data contained in one or more table and can also be said as restricted view to the data‟s in the tables. A view is a “virtual table” or a “stored query” which takes the output of a query and treats it as a table. The table upon which a view is created is called as base table.
2	A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables. The view is stored as a SELECT statement in the data dictionary
3	**Advantages of a view:** a. Additional level of table security. b. Hides data complexity. c. Simplifies the usage by combining multiple tables into a single table. d. Provides data‟s in different perspective.
4	**Types of view:** Horizontal -> enforced by where cause Vertical -> enforced by selecting the required columns

c) SQL Commands

Creating and dropping view:

Syntax:

Create [or replace] view <view name> [column alias names] as <query> [with <options> conditions];

Drop view <view name>;

Example:

Create or replace view empview as select * from emp;

Drop view empview;

d)

Queries:

Tables used:

```
SQL> select * from emp;
    EMPNO ENAME      JOB           DEPTNO       SAL
--------- -------------------- ---------- ---------- ----------
        1 Mathi              AP             1      10000
```

2	Arjun	ASP	2	12000
3	Gugan	ASP	2	20000
4	Karthik	AP	1	15000

Q1: The organization wants to display only the details of the employees those who are ASP. (Horizontal portioning)

Solution:

1. Create a view on emp table named managers
2. Use select from clause to do horizontal partioning

Ans:

```
SQL> create view empview as select * from emp where job='ASP';
View created.
SQL> select * from empview;
    EMPNO ENAME    JOB       DEPTNO   SAL
---------- -------------------- ---------- ---------- ----------
         2 Arjun              ASP              2      12000
         3 Gugan              ASP              2      20000
```

Q2: The organization wants to display only the details like empno, empname, deptno, deptname of the employees. (Vertical portioning)

Solution:

1. Create a view on emp table named general 2. Use select from clause to do vertical partioning

Ans:

```
SQL> create view empview1 as  select ename,sal from emp;
View created.
```

Q3: **Display all the views generated.**

Ans:

```
SQL> select * from tab;
TNAME                TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                  TABLE
EMP                   TABLE
EMPVIEW               VIEW
EMPVIEW1              VIEW
```

Q4: Execute the DML commands on the view created.

Ans:

```
SQL> select * from empview;
    EMPNO ENAME              JOB          DEPTNO       SAL
---------- -------------------- ---------- ---------- ----------
         2 Arjun              ASP              2      12000
         3 Gugan              ASP              2      20000
```

Q5: Drop a view.

Ans: SQL> drop view empview1;

View dropped.

e) Result:

Thus the creation and manipulate various database objects of the Table using views was successfully executed.

QUESTIONS AND ANSWERS

1. What is a view?

A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed.

2. List any two advantages of view?

1. Hides data complexity.
2. Simplifies the usage by combining multiple tables into a single table.

Exercise Number: 8

Title of the Exercise : CONTROL STRUCTURE

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To create PL/SQL programs to implement various types of control structure.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) PL/SQL Syntax:

PL/SQL can also process data using flow of statements. The flow of control statements are classified into the following categories.

- Conditional control –Branching
- Iterative control – looping
- Sequential control - Selection

BRANCHING in PL/SQL:

Sequence of statements can be executed on satisfying certain condition. If statements are being used and different forms of if are:

1. Simple IF 2. If then else 3. Else if 4. Nested if

SELECTION IN PL/SQL (Sequential Controls)

1. Simple case 2. Searched case

ITERATIONS IN PL/SQL

Sequence of statements can be executed any number of times using loop construct. It is broadly classified into:

1.Simple Loop 2. For Loop 3. While Loop

SIMPLE IF:

Syntax:

```
IF condition THEN
statement1;
statement2;
END IF;
```

IF-THEN-ELSE STATEMENT:

Syntax:

```
IF condition THEN
statement1;
ELSE
statement2;
END IF;
```

ELSIF STATEMENTS:

Syntax:

```
IF condition1 THEN
statement1;
ELSIF condition2 THEN
statement2;
ELSIF condition3 THEN
statement3;
ELSE
```

```
        statement;
        END IF;
```

NESTED IF:

Syntax:

```
        IF condition THEN
        statement1;
        ELSE
        IF condition THEN
        statement2;
        ELSE
        statement3;
        END IF;
        END IF;
        ELSE
        statement3;
        END IF;
```

SELECTION IN PL/SQL (Sequential Controls)

SIMPLE CASE

Syntax:

```
        CASE SELECTOR
        WHEN Expr1 THEN statement1;
        WHEN Expr2 THEN statement2;
        :
        ELSE
        Statement n;
        END CASE;
```

SEARCHED CASE:

Syntax:

```
        CASE
        WHEN searchcondition1 THEN statement1;
        WHEN searchcondition2 THEN statement2;
        :: ELSE
        statementn;
        END CASE;
```

ITERATIONS IN PL/SQL

SIMPLE LOOP

Syntax:

```
        LOOP
        statement1;
        EXIT [ WHEN Condition];
        END LOOP;
```

Example:

```
    Declare
    A number:=10;
    Begin
    Loop
    a := a+25;
    exit when a=250; end loop;
    dbms_output.put_line(to_char(a));
    end;
/
```

WHILE LOOP

Syntax

```
WHILE condition LOOP
statement1;
statement2;
END LOOP;
```

Example:

```
Declare
i number:=0;
j number:=0;
begin
while i<=100 Loop
j := j+i;
i := i+2;
end loop;
dbms_output.put_line(„the value of j is‟ ||j);
end;
/
```

FOR LOOP

Syntax:

```
FOR counter IN [REVERSE]
LowerBound..UpperBound
LOOP
statement1;
statement2;
END LOOP;
```

Example:

```
Begin
For I in 1..2
Loop
Update emp set field = value where condition;
End loop;
End;
/
```

Q1: write a pl/sql program to swap two numbers

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Declare three variables and read variables through a and b
2	Swap the values of a and b using temporary variables
3	Display the swapped results

d) Program:

```
SQL>edit swapping.sql
declare
a number(10);
b  number(10);
c  number(10);
begin
dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
dbms_output.put_line(a);
dbms_output.put_line(b);
```

```
a:=&a;
b:=&b;
c:=a;
a:=b;
b:=c;
dbms_output.put_line('THE VALUES OF A AND B ARE');
dbms_output.put_line(a);
dbms_output.put_line(b);
end;
```

e)output:

```
SQL> @ swapping.sql
19 /
Enter value for a: 5
old 6: a:=&a;
new 6: a:=5;
Enter value for b: 3
old 7: b:=&b;
new 7: b:=3;
THE PREV VALUES OF A AND B WERE
53
THE VALUES OF A AND B ARE
35
```

PL/SQL procedure successfully completed.

Q2: Write a pl/sql program to find the largest of three numbers

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read three numbers through a, b & c
2	Find the biggest among three using nested if statement
3	Display the biggest no as result

d)Program:

```
SQL>set server output on;
SQL>edit biggest.sql
declare
a number;
b number;
c number;
begin
a:=&a;
b:=&b;
c:=&c;
if a>b then
if a>c then
dbms_output.put_line ('biggest is:' ||to_char(a));
else
dbms_output.put_line('biggest is :' ||to_char(c));
end if;
elsif b>c then
dbms_output.put_line('biggest is :' ||to_char(b));
else
```

```
dbms_output.put_line('biggest is :' ||to_char(c));
end if;
end;
```

e)output:

```
SQL>@biggest.sql
/
        Enter value for a: 5
        old 6: a:=&a;
        new 6: a:=5;
        Enter value for b: 5
        old 6: b:=&b;
        new 6: b:=8;
        Enter value for c: 8
        old 6: c:=&c;
        new 6: c:=4;
        biggest is : 8
```

Q3: write a pl/sql program to find the total and average of 6 subjects and display the grade

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read six numbers and calculate total and average
2	Find whether the student is pass or fail using if statement
3	Find the grade using nested elseif statement
4	Display the Grade, Percentage and Total of the student

d)Program:

```
SQL> edit grade.sql
        declare
        java number(10);
        dbms number(10);
        co   number(10);
        se   number(10);
        es   number(10);
        ppl number(10);
        total number(10);
        avgs number(10);
        per number(10);
        begin
        dbms_output.put_line('ENTER THE MARKS');
        java:=&java;
        dbms:=&dbms;
        co:=&co;
        se:=&se;
        es:=&es;
        ppl:=&ppl;
        total:=(java+dbms+co+se+es+ppl);
        per:=(total/600)*100;
        if java<50 or dbms<50 or co<50 or se<50 or es<50 or ppl<50 then
        dbms_output.put_line('FAIL'); if
        per>75 then
        dbms_output.put_line('GRADE A');
        elsif per>65 and per<75 then
        dbms_output.put_line('GRADE B');
```

```
elsif per>55 and per<65 then

dbms_output.put_line('GRADE C');
else
dbms_output.put_line('INVALID INPUT');
end if;
dbms_output.put_line('PERCENTAGE IS '||per);
dbms_output.put_line('TOTAL IS '||total);
end;
```

e)output:

```
SQL> @ grade.sql
31 /
Enter value for java: 80
old 12: java:=&java;
new 12: java:=80;
Enter value for dbms: 70
old 13: dbms:=&dbms;
new 13: dbms:=70;
Enter value for co: 89
old 14: co:=&co;
new 14: co:=89;
Enter value for se: 72
old 15: se:=&se;
new 15: se:=72;
Enter value for es: 76
old 16: es:=&es;
new 16: es:=76;
Enter value for ppl: 71
old 17: ppl:=&ppl;
new 17: ppl:=71;
GRADE A
PERCENTAGE IS 76
TOTAL IS 458
PL/SQL procedure successfully completed.
```

Q4: Write a pl/sql program to find the sum of digits in a given number

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number. Separate the digits using modular function
2	Sum the digits separated by mod function
3	Display the sum of digits

d)Program:

```
SQL>edit sumofdigits.sql
declare
a number;
d number:=0; sum1
number:=0; begin
a:=&a;
while a>0 loop
d:=mod(a,10);
```

```
sum1:=sum1+d;
a:=trunc(a/10);
end loop;
dbms_output.put_line('sum is'|| sum1);
end;
```

e)output:

```
SQL> @ sumofdigits.sql
        16 /
```

Q5: write a pl/sql program to display the number in reverse order

c)Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number. Separate the digits using modular function
2	Reverse the digits separated by taking remainder from mod function
3	Display the reverse of the digits

d)Program:

```
SQL>edit reverse.sql
declare
a number;
rev number;
d number;
begin
a:=&a;
rev:=0;
while a>0
loop
d:=mod(a,10);
rev:=(rev*10)+d;
a:=trunc(a/10);
end loop;
dbms_output.put_line('no is'|| rev);
end;
```

e)output:

```
SQL> @ reverse.sql
16 /
Enter value for a: 536
        old 6: a:=&a;
        new 6: a:=536;
no is 635
PL/SQL procedure successfully completed.
```

Q6: Write a PL / SQL program to check whether the given number is prime or not

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read the number
2	Using mod function find the given number is prime or not
3	Display the result

d)Program:

```
SQL>edit prime.sql
declare
a number;        c number:=0;              i number;
begin
a:=&a;
for i in 1..a
loop
if mod(a,i)=0 then
c:=c+1;
end if;
end loop;
if c=2 then
dbms_output.put_line(a ||'is a prime number');
else
dbms_output.put_line(a ||'is not a prime number');
end if;
end;
```

e)output:

```
SQL> @ prime.sql
19 /
Enter value for a: 11
old 6: a:=&a;
new 6: a:=11;
11is a prime number
PL/SQL procedure successfully completed.
```

Q7: Write a PL/SQL program to find the factorial of a given number

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number for calculating factorial value.
2	Calculate the factorial of a given number using for loop
3	Display the factorial value of a given number.

d)Program:

```
SQL>edit fact.sql
declare
n number;f number:=1;
begin
n:=&n;
for i in 1..n
loop
f:=f*i;
end loop;
dbms_output.put_line('the factorial is'|| f);
end;
```

e)output:

```
SQL> @ fact.sql
12 /
Enter value for n: 5
old 5: n:=&n;
new 5: a:=5;
the factorial is 120
```

Q8: write a pl/sql code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns radius & area

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Create a table named areas with radius and area
2	Initialize values to pi, radius and area
3	Calculate the area using while loop. Display the result.

d)Program:

```
SQL> create table areas(radius number(10),area number(6,2));
Table created.
PROGRAM
declare
pi constant number(4,2):=3.14;
radius number(5):=3; area number(6,2);
begin
while radius<7
loop
area:=pi*power(radius,2);
insert into areas values(radius,area);
radius:=radius+1;
end loop;
end;
```

e)output:

```
SQL> @ AREAOFCIRCLE.SQL
13 /
PL/SQL procedure successfully completed.
SQL> SELECT * FROM AREAS;
RADIUS AREA
---------- ----------
3 28.26
4 50.24
5 78.5
6 113.04
```

Q9: write a PL/SQL code block that will accept an account number from the user, check if the users balance is less than minimum balance, only then deduct rs.100/- from the balance. This process is fired on the acct table.

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Develop a query to Create the table acct and insert values into them
2	Develop a PL/SQL program to read the account number.
3	Check the balance for the account no. check if the users balance is less than minimum balance, only then deduct rs.100/- from the balance
4	Update the balance changes into the acct table.

d)Program:

```
SQL> create table acct(name varchar2(10),cur_bal number(10),acctno number(6,2));
SQL> insert into stud values('&sname',&rollno,&marks);
SQL> select * from acct;
ACCTNO NAME CUR_BAL
---------- ---------- ----------
777 sirius 10000
765 john 1000
855 sam 500
353 peter 800
declare
mano number(5);
mcb number(6,2);
minibal constant number(7,2):=1000.00;
fine number(6,2):=100.00;
begin
mano:=&mano;
select cur_bal into mcb from acct where acctno=mano;
if mcb<minibal then
update acct set cur_bal=cur_bal-fine where acctno=mano;
end if;
end;
```

e)output:

```
SQL> @ BANKACC.sql
13 /
Enter value for mano: 855
old 7: mano:=&mano;
new 7: mano:=855;
        PL/SQL procedure successfully completed.
```

f)Result:

Thus the above creation of PL/SQL programs to implement various types of control structure was successfully executed.

QUESTIONS AND ANSWERS

1. What is meant by branching in PL/SQL:

Sequence of statements can be executed on satisfying certain condition. If statements are being used and different forms of if are:

1. Simple IF 2. If then else 3. Else if 4. Nested if

2. What are selection statements?

1. Switch case statement

3. Define iterations IN PL/SQL

Sequence of statements can be executed any number of times using loop construct.

4. Classify the iteration statements `in PL/SQL

It is broadly classified into:

1.Simple Loop
2. For Loop
3. While Loop

Exercise Number: 9

Title of the Exercise : PROCEDURE AND FUNCTION
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To develop procedures and function for various operations.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) PL/SQL syntax:

A procedure is a block that can take parameters (sometimes referred to as arguments) and be invoked.

Procedures promote reusability and maintainability. Once validated, they can be used in number of applications. If the definition changes, only the procedure are affected, this greatly simplifies maintenance.

Modularized program development:

· Group logically related statements within blocks.
· Nest sub-blocks inside larger blocks to build powerful programs.
· Break down a complex problem into a set of manageable well defined logical modules and implement the modules with blocks.

KEYWORDS AND THEIR PURPOSES

REPLACE: It recreates the procedure if it already exists.
PROCEDURE: It is the name of the procedure to be created.
ARGUMENT: It is the name of the argument to the procedure. Parenthesis can be omitted if no arguments are present.
IN: Specifies that a value for the argument must be specified when calling the procedure ie., used to pass values to a sub-program. This is the default parameter.
OUT: Specifies that the procedure passes a value for this argument back to it‟s calling environment after execution ie. used to return values to a caller of the sub-program.
INOUT: Specifies that a value for the argument must be specified when calling the procedure and that procedure passes a value for this argument back to it‟s calling environment after execution.
RETURN: It is the data type of the function‟s return value because every function must return a value, this clause is required.

PROCEDURES

Syntax :

```
create or replace procedure <procedure name> (argument {in,out,inout} datatype ) {is,as}
variable  declaration;
constant  declaration;
begin
PL/SQL subprogram body;
exception
exception PL/SQL block;
end;
```

FUNCTIONS

Syntax:

```
create or replace function <function name> (argument in datatype,……) return datatype {is,as}
variable  declaration;
constant  declaration;
begin
PL/SQL subprogram body;
exception
exception PL/SQL block;
end;
```

Tables used:

```
SQL> select * from ititems;
ITEMID  ACTUALPRICE    ORDID    PRODID
---------  -----------  --------  ---------
     101        2000       500       201
     102        3000      1600       202
     103        4000       600       202
```

PROGRAM FOR GENERAL PROCEDURE – SELECTED RECORD'S PRICE IS INCREMENTED BY 500 , EXECUTING THE PROCEDURE CREATED AND DISPLAYING THE UPDATED TABLE

```
SQL> create procedure itsum(identity number, total number) is price number;
  2  null_price exception;
  3  begin
  4  select actualprice into price from ititems where itemid=identity;
  5  if price is null then
  6  raise null_price;
  7  else
  8  update ititems set actualprice=actualprice+total where itemid=identity;
  9  end if;
 10  exception
 11  when null_price then
 12  dbms_output.put_line('price is null');
 13  end;
 14  /
Procedure created.

SQL> exec itsum(101, 500);
PL/SQL procedure successfully completed.

SQL> select * from ititems;
 ITEMID   ACTUALPRICE     ORDID     PRODID
---------  -----------  ---------  ---------
     101        2500       500       201
     102        3000      1600       202
     103        4000       600       202
```

PROCEDURE FOR _IN' PARAMETER – CREATION, EXECUTION

```
SQL> set serveroutput on;
SQL> create procedure yyy (a IN number) is price number;
  2  begin
  3  select actualprice into price from ititems where itemid=a;
```

```
  4  dbms_output.put_line('Actual price is ' || price);
  5  if price is null then
  6  dbms_output.put_line('price is null');
  7  end if;
  8  end;
  9  /
Procedure created.

SQL> exec yyy(103);
Actual price is 4000
PL/SQL procedure successfully completed.
```

PROCEDURE FOR „OUT‘ PARAMETER – CREATION, EXECUTION

```
SQL> set serveroutput on;
SQL> create procedure zzz (a in number, b out number) is identity number;
  2  begin
  3  select ordid into identity from ititems where itemid=a;
  4  if identity<1000 then
  5   b:=100;
  6  end if;
  7  end;
  8  /
Procedure created.
SQL> declare
  2  a number;
  3  b number;
  4  begin
  5  zzz(101,b);
  6  dbms_output.put_line('The value of b is '|| b);
  7  end;
  8  /
The value of b is 100
PL/SQL procedure successfully completed.
```

PROCEDURE FOR „INOUT‘ PARAMETER – CREATION, EXECUTION

```
SQL> create procedure itit ( a in out number) is
  2  begin
  3  a:=a+1;
  4  end;
  5  /
  Procedure created.

SQL> declare
  2  a number:=7;
  3  begin
  4  itit(a);
  5  dbms_output.put_line(„The updated value is „||a);
  6  end;
  7  /
The updated value is 8
PL/SQL procedure successfully completed.
```

Tables used:

```
SQL>select * from ittrain;
    TNO      TFARE
 ---------   ------------
   1001        550
   1002        600
```

PROGRAM FOR FUNCTION AND IT'S EXECUTION

```
SQL> create function trainfn (trainnumber number) return number is
  2  trainfunction ittrain.tfare % type;
  3  begin
  4  select tfare into trainfunction from ittrain where tno=trainnumber;
  5  return(trainfunction);
  6  end;
  7  /
Function created.

SQL> declare
  2  total number;
  3  begin
  4  total:=trainfn (1001);
  5  dbms_output.put_line('Train fare is Rs. '||total);
  6  end;
  7  /
Train fare is Rs.550
PL/SQL procedure successfully completed.
```

FACTORIAL OF A NUMBER USING FUNCTION — PROGRAM AND EXECUTION

```
 SQL>  create function itfact (a number) return number is
 2   fact number:=1;
 3   b number;
 4   begin
 5   b:=a;
 6   while b>0
 7   loop
 8   fact:=fact*b;
 9   b:=b-1;
10   end loop;
11  return(fact);
12  end;
13  /
Function created.

SQL> declare
  2  a number:=7;
  3  f number(10);
  4  begin
  5  f:=itfact(a);
  6  dbms_output.put_line(„The factorial of the given number is"||f);
  7 end;
  8  /
```

The factorial of the given number is 5040

Q1: Write a procedure to calculate total for the all the students and pass regno, mark1, & mark2 as arguments.

c)Procedure for doing the experiment:

Step no.	Details of the step
1	Develop a query to create a table named itstudent2 and insert values into them
2	Develop a procedure p1 with regno, mark1, & mark2 as arguments.
3	Calculate the total and update the total value into the itstudent2 table

d)Program:

```
SQL>  create table itstudent2(regno number(3),name varchar(9),mark1
number(3),mark2 number(3));
Table created.
SQL> insert into itstudent2
  2
values(&a,'&b',&c,&d);
Enter value for a: 110
Enter value for b: arun
Enter value for c: 99            Enter value for d: 100
old   2: values(&a,'&b',&c,&d)
new   2: values(110,'arun',99,100)
1 row
created.
SQL> /
Enter value for a: 112           Enter value for b: siva          Enter value for c: 99     Enter
value for d: 90
old   2: values(&a,'&b',&c,&d)
new   2: values(112,'siva',99,90)
1 row created.
SQL> select * from itstudent2;
    REGNO NAME          MARK1      MARK2
       110 arun            99        100
       112 siva            99        90
SQL> alter table itstudent2 add(total number(5));              Table altered.
SQL> select * from itstudent2;
REGNO NAME          MARK1      MARK2      TOTAL
       110  arun            99        100
       112  siva            99        90
SQL> create or replace procedure p1(sno number,mark1 number,mark2 number) is
  2  tot number(5);
  3  begin
  4  tot:=mark1+mark2;
  5  update itstudent2 set total=tot where regno=sno;
  6  end;
  7  /
Procedure created.
SQL> declare
  2  cursor c1 is select * from itstudent2;
  3  rec itstudent2 % rowtype;
  4  begin
```

```
 5  open c1;
 6  loop
 7  fetch c1 into rec;
 8  exit when c1%notfound;
 9  p1(rec.regno,rec.mark1,rec.mark2);
10  end loop;
11  close c1;
12  end;
13  /
```

PL/SQL procedure successfully completed.

e)Output:

```
SQL> select * from itstudent2;
REGNO NAME          MARK1      MARK2      TOTAL
--------- --------- ---------- ---------- ----------
      110 arun             99        100        199
      112 va               99         90        189
```

Q2: Write a PL/SQL procedure called MULTI_TABLE that takes two numbers as parameter and displays the multiplication of the first parameter till the second parameter.

Ans.

//p2.sql

```
create or replace procedure multi_table (a number, b number) as
mul number;
begin
for i in 1. .b
loop
mul : = a * i;
dbms_output.put_line (a || „*" || i || „=" || mul);
end loop;
end;
```

//pq2.sql

```
declare
a number;  b number;
begin
a:=&a;              b:=&b;                multi_table(a,b);
end;
```

e)Output:

```
SQL> @p2.sql;
  Procedure created.
SQL> @pq2.sql;
Enter value for a: 4
old   5: a:=&a;             new   5: a:=4;
Enter value for b: 3
old   6: b:=&b;             new   6: b:=3;
4*1=4
4*2=8
4*3=12
```

Q3: Consider the EMPLOYEE (EMPNO, SALARY, ENAME) Table.
Write a procedure raise_sal which increases the salary of an employee. It accepts an employee number and salary increase amount. It uses the employee number to find the current salary from the EMPLOYEE table and update the salary.

Ans:

//p3.sql

```
create or replace procedure raise_sal( mempno employee . empno % type, msal_percent
number ) as
begin
```

```
update employee set salary = salary + salary*msal_percent /100 where empno = mempno;
end;
/
```

//pq3.sql

```
declare
 cursor c1 is select * from emp;
  rec emp % rowtype;
  begin
  open c1;
  loop
  fetch c1 into rec;
  exit when c1%notfound;
  raisal(rec.empno,10);
  end  loop;
  close   c1;
  end;
  /
```

e)Output:

```
SQL> @p3.sql;
  Procedure created.
SQL> select * from emp;
    EMPNO ENAME    JOB             DEPTNO       SAL
---------- -------------------- ------------- ---------- ----------
         1 Mathi               AP                 1     10000
         2 Arjun               ASP                2     15000
         3 Gugan               ASP                1     15000
         4 Karthik             Prof               2     30000
         5 Akalya              AP                 1     10000
SQL> @pq3.sql;
PL/SQL procedure successfully completed.
SQL> select * from emp;
    EMPNO ENAME   JOB         DEPTNO       SAL
---------- -------------------- ------------- ---------- ----------
         1 Mathi               AP                 1     11000
         2 Arjun               ASP                2     16500
         3 Gugan               ASP                1     16500
         4 Karthik             Prof               2     33000
         5 Akalya              AP                 1     11000
```

Q4: Write a PL/SQL function CheckDiv that takes two numbers as arguments and returns the values 1 if the first argument passed to it is divisible by the second argument, else will return the value 0;

Ans:

//p4.sql

```
create or replace function checkdiv (n1 number, n2 number) return number as res
number;
begin
if mod (n1, n2) = 0 then
res := 1;
else
res:= 0;
end if;
return res;
end;
/
```

//pq4.sql

```
declare
a number;
b number;
begin
a:=&a;          b:=&b;
dbms_output.put_line(„result="||checkdiv(a,b));
end;
/
```

e)Output:

```
SQL> @p4.sql;
        Function created.
SQL> @pq4.sql;
        Enter value for a: 4
        old   5: a:=&a;          new   5: a:=4;
        Enter value for b: 2
        old   6: b:=&b;          new   6: b:=2;
        result=1
```

Q5: Write a PL/SQL function called POW that takes two numbers as argument and return the value of the first number raised to the power of the second .

Ans:

//p5.sql

```
create or replace function pow (n1 number, n2 number) return number as
res number;
begin
select power ( n1, n2) into res from dual;          return res;
end;
```

or

```
create or replace function pow (n1 number, n2 number) return number as
res number : =1;
begin
for res in 1..n2
loop
res : = n1 * res;
end loop;
return res;
end;
```

//pq5.sql

```
declare
a  number;
b  number;
begin
a:=&a;          b:=&b;
dbms_output.put_line('power(n1,n2)='||pow(a,b));
end;
/
```

e)Output:

```
SQL> @p5.sql;
        Function created.
SQL> @ pq5.sql;
        Enter value for a: 2
        old   5: a:=&a;
        new   5: a:=2;
        Enter value for b: 3
        old   6: b:=&b;
```

```
new   6: b:=3;
power(n1,n2)=8
```

Q6: Write a PL/SQL function ODDEVEN to return value TRUE if the number passed to it is EVEN else will return FALSE.

Ans:

//p6.sql

```
create or replace function oddeven (n number) return boolean as
begin
if mod (n, 2) = 0 then          return true;
else
return false;
end if;
end;
/
```

//pq6.sql

```
declare
a number;        b boolean;
begin
a:=&a;                    b:=oddeven(a);
if b then
dbms_output.put_line('The given number is Even');
else
dbms_output.put_line('The given number is Odd');
end if;
end;
/
```

e)Output:

```
SQL> @p6.sql;
        Function created.
SQL> @pq6.sql;
        Enter value for a: 5
        old   5: a:=&a;                    new   5: a:=5;
        The given number is Odd
```

f)Result:

Thus the procedures and function for various operations was developed and executed successfully.

QUESTIONS AND ANSWERS

1. What is procedure? Write its advantages.

A procedure is a block that can take parameters (sometimes referred to as arguments) and be invoked. **Advantages:**

- Procedures promote reusability and maintainability.
- They can be used in number of applications.
- If the definition changes, only the procedure are affected, this greatly simplifies maintenance.

2. List the three types of argument passed in to the procedure

IN: Specifies that a value for the argument must be specified when calling the procedure

OUT: Specifies that the procedure passes a value for this argument back to it"s calling environment after execution ie. used to return values to a caller of the sub-program.

INOUT: Specifies that a value for the argument must be specified when calling the procedure and that procedure passes a value for this argument back to it"s calling environment after execution.

3. Is the function return value?

Yes, function‟s return value because every function must return a value, this clause is required.

Exercise Number: 10

Title of the Exercise : TRIGGER
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To create triggers for various events such as insertion, updation, etc.,

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) PL/SQL Syntax:

TRIGGER

A Trigger is a stored procedure that defines an action that the database automatically take when some database-related event such as Insert, Update or Delete occur.

TRIGGER VS. PROCEDURE VS CURSOR

TRIGGER	PROCEDURES	CURSORS
These are named PL/SQL blocks.	These are named PL/SQL blocks.	These are named PL/SQL blocks.
These are invoked automatically.	User as per need invokes these.	These can be created both explicitly and implicitly.
These can‟t take parameters.	These can take parameters.	These can take parameters.
These are stored in database.	These are stored in database.	These are not stored in database.

TYPES OF TRIGGERS

The various types of triggers are as follows,

- **Before**: It fires the trigger before executing the trigger statement.
- **After**: It fires the trigger after executing the trigger statement.
- **For each row**: It specifies that the trigger fires once per row.
- **For each statement**: This is the default trigger that is invoked. It specifies that the trigger fires once per statement.

VARIABLES USED IN TRIGGERS

- :new
- :old

These two variables retain the new and old values of the column updated in the database. The values in these variables can be used in the database triggers for data manipulation

Row Level Trigger vs. Statement Level Trigger:

Row Level Trigger	Statement Level Trigger
These are fired for each row affected by the DML statement.	These are fired once for the statement instead of the no of rows modified by it.
These are used for generating/checking the values begin inserted or updated.	These are used for generated the summary information.

Before trigger vs. after trigger

Before Triggers	After Triggers
Before triggers are fired before the DML statement is actually executed.	After triggers are fired after the DML statement has finished execution.

Sytax:

Create or replace trigger <trg_name> Before /After Insert/Update/Delete
[of column_name, column_name….]
on <table_name>
[for each row]
[when condition]
begin
---statement
end;

Q1: Create a trigger that insert current user into a username column of an existing table
c) Procedure for doing the experiment:

Step no.	Details of the step
1	Create a table itstudent4 with name and username as arguments
2	Create a trigger for each row that insert the current user as user name into a table
3	Execute the trigger by inserting value into the table

d)Program:

```
SQL> create table itstudent4(name varchar2(15),username varchar2(15));
Table created.
SQL> create or replace trigger itstudent4 before insert on itstudent4 for each row
  2  declare
  3  name varchar2(20);
  4  begin
  5  select user into name from dual;
  6  :new.username:=name;
  7  end;
  8   /
Trigger created.
```

e)Output:

```
SQL> insert into itstudent4 values('&name','&username');
Enter value for name: akbar
Enter value for username: ranjani
old   1: insert into itstudent4 values('&name','&username')
new   1: insert into itstudent4 values('akbar','ranjani')
1 row created.
SQL> /
Enter value for name: suji
Enter value for username: priya
old   1: insert into itstudent4 values('&name','&username')
```

```
new   1: insert into itstudent4 values('suji','priya')
1 row created.
SQL> select * from itstudent4;

NAME          USERNAME
--------------- ---------------
akbar         SCOTT
suji          SCOTT
```

Q2: Create a Simple Trigger that does not allow Insert Update and Delete Operations on the Table

d)Program:

Table used:

```
SQL> select * from itempls;
ENAME          EID    SALARY
---------- --------- ---------
xxx             11     10000
yyy             12     10500
zzz            13     15500
```

Trigger:

```
SQL> create trigger ittrigg before insert or update or delete on itempls for each row
  2  begin
  3  raise_application_error(-20010,'You cannot do manipulation');
  4  end;
  5
  6  /
Trigger created.
```

e)Output:

```
SQL> insert into itempls values('aaa',14,34000);
insert into itempls values('aaa',14,34000)
            *
ERROR at line 1:
ORA-20010: You cannot do manipulation
ORA-06512: at "STUDENT.ITTRIGG", line 2
ORA-04088: error during execution of trigger 'STUDENT.ITTRIGG'

SQL> delete from itempls where ename='xxx';
delete from itempls where ename='xxx'
            *
ERROR at line 1:
ORA-20010: You cannot do manipulation
ORA-06512: at "STUDENT.ITTRIGG", line 2
ORA-04088: error during execution of trigger 'STUDENT.ITTRIGG'

SQL> update itempls set eid=15 where ename='yyy';
update itempls set eid=15 where ename='yyy'
       *
ERROR at line 1:
ORA-20010: You cannot do manipulation
ORA-06512: at "STUDENT.ITTRIGG", line 2
ORA-04088: error during execution of trigger 'STUDENT.ITTRIGG'
```

Q3: Create a Trigger that raises an User Defined Error Message and does not allow updating and Insertion

d)Program:

Table used:

```
SQL> select * from itempls;
ENAME          EID    SALARY
---------- --------- ---------
xxx             11     10000
yyy             12     10500
zzz             13     15500
```

Trigger:

```
SQL> create trigger ittriggs before insert or update of salary on itempls for each row
  2  declare
  3  triggsal itempls.salary%type;
  4  begin
  5  select salary into triggsal from itempls where eid=12;
  6  if(:new.salary>triggsal or :new.salary<triggsal) then
  7  raise_application_error(-20100,'Salary has not been changed');
  8  end if;
  9  end;
 10  /
Trigger created.
```

e)Output:

```
SQL> insert into itempls values ('bbb',16,45000);
insert into itempls values ('bbb',16,45000)
            *
ERROR at line 1:
ORA-04098: trigger 'STUDENT.ITTRIGGS' is invalid and failed re-validation
SQL> update itempls set eid=18 where ename='zzz';
update itempls set eid=18 where ename='zzz'
            *
ERROR at line 1:
ORA-04298: trigger 'STUDENT.ITTRIGGS' is invalid and failed re-validation
```

Q4: develop a query to Drop the Created Trigger

Ans:

```
SQL> drop trigger ittrigg;
Trigger dropped.
```

f)Result:

Thus the creation of triggers for various events such as insertion, updation, etc., was performed and executed successfully.

QUESTIONS AND ANSWERS

1. **What is the need for triggers? Or List the requirements needed to design a trigger.**
   - Specifying when a trigger is to be executed.
   - Specify the actions to be taken when the trigger executes.
2. **What is trigger?**

   Triggers are statements that are executed automatically by the system as the side effect of a modification to the database. The triggers can be initiated before the event or after the event.

Exercise Number: 11

Title of the Exercise : FRONT END TOOLS
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To study about various Visual Basic (front end) tools

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	S/W Name	VB

b) Procedure

Object

- An object is a type of user interface element you create on a Visual Basic form by using a toolbox control.
- In fact, in Visual Basic, the form itself is also an object.
- You can move, resize, and customize objects by setting object properties.
- A property is a value or characteristic held by a Visual Basic object, such as Caption or Fore Color.
- Properties can be set at design time by using the Properties window or at run time by using statements in the program code.

  Object. Property = Value
- Where

  Object is the name of the object you"re customizing.
  Property is the characteristic you want to change.
  Value is the new property setting.
- For example,

  Command1.Caption = "Hello"

1.The Form Object

- The Form is where the user interface is drawn. It is central to the development of Visual Basic applications.
- **Form Properties:**
  - Appearance Selects 3-D or flat appearance.
  - BackColor Sets the form background color.
  - BorderStyle Sets the form border to be fixed or sizeable.
  - Caption sets the form window title.
  - Enabled If True, allows the form to respond to mouse and keyboard events; if False, disables form.
  - Font Sets font type, style, size.
  - ForeColor Sets color of text or graphics.
  - Picture Places a bitmap picture in the form.
  - Visible If False, hides the form.
- **Form Events:**
- Activate Form_Activate event is triggered when form becomes the active window.
- Click Form_Click event is triggered when user clicks on form.
- DblClick Form_DblClick event is triggered when user doubleclicks on form.
- Load Form_Load event occurs when form is loaded.This is a good place to initialize variables and set any runtime properties.
- **Form Methods:**
- Cls Clears all graphics and text from form. Does not clear any objects.
- Print Prints text string on the form.

- **Examples**

```
frmExample.Cls ' clears the form
frmExample.Print "This will print on the form"
```

2. CommandButton

- The Command Button control is use to create buttons with a variety of uses on a form.
- A command button is the most basic way to get user input while a program is running.
- By clicking a command button, the user requests that a specific action be taken in the program.
- Visual Basic terms, clicking a command button creates an event, which must be processed in your program.
- **Command Button Properties:**
- Appearance Selects 3-D or flat appearance.
- Cancel Allows selection of button with Esc key (only one button on a form can have this property True).
- Caption String to be displayed on button.
- Default Allows selection of button with Enter key (only one button on a form can have this property True).
- Font Sets font type, style, size.
- **Command Button Events:**
- Click Event triggered when button is selected either by clicking on it or by pressing the access key.

3. Label Boxes

- Label, the simplest control in the Visual Basic toolbox, displays formatted text on a user interface form. Typical uses for the Label control include:
- Help text
- Program splash screen headings
- Formatted output, such as names, times, and dates
- Descriptive labels for other objects, including text boxes and list boxes.
- **Label Properties:**
- Alignment Aligns caption within border.
- Appearance Selects 3-D or flat appearance.
- AutoSize -If True, the label is resized to fit the text specified by the caption property. If False, the label will remain the size defined at design time and the text may be clipped.
- BorderStyle Determines type of border.
- Caption String to be displayed in box.
- Font Sets font type, style, size.
- **Label Events:**
- Click Event triggered when user clicks on a label.
- DblClick Event triggered when user double-clicks on a label.

4. Textbox

- A Textbox is used to display information entered at design time, by a user at run-time, or assigned within code.
- The displayed text may be edited.
- The Textbox control is one of the most versatile tools in the Visual Basic toolbox.
- This control performs two functions:
- Displaying output (such as operating instructions or the contents of a file) on a form.
- Receiving text (such as names and phone numbers) as user input.
- **Text Box Properties:**

1. Appearance : Selects 3-D or flat appearance.
2. BorderStyle : Determines type of border.
3. Font : Sets font type, style, size.
4. MaxLength : Limits the length of displayed text (0 value indicates unlimited length).
5. MultiLine : Specifies whether text box displays single line or multiple lines.

6. PasswordChar :Hides text with a single character.
7. ScrollBars :Specifies type of displayed scroll bar(s).
8. SelLength :Length of selected text (run-time only).
9. SelStart :Starting position of selected text (run-time only).
10. SelText :Selected text (run-time only).
11. Tag : Stores a string expression.
12. Text :Displayed text

- **Text Box Events:**

1. Change :Triggered every time the Text property changes.
2. LostFocus :Triggered when the user leaves the text box. This is a good place to examine the contents of a text box after editing.
3. KeyPress :Triggered whenever a key is pressed. Used for key trapping, as seen in last class.

5. Check Boxes

- Check boxes provide a way to make choices from a list of potential candidates.
- Some, all, or none of the choices in a group may be selected
- **Check Box Properties:**
- Caption :Identifying text next to box.
- Font :Sets font type, style, size.
- Value :Indicates if unchecked (0, vbUnchecked), checked (1,vbChecked), or grayed out (2, vbGrayed).
- **Check Box Events:**
- Click :Triggered when a box is clicked. Value property is automatically changed by Visual Basic.

6. Option Buttons

- Option buttons provide the capability to make a mutually exclusive choice among a group of potential candidate choices.
- Hence, option buttons work as a group, only one of which can have a True (or selected) value.
- **Option Button Properties:**
- Caption :Identifying text next to button.
- Font :Sets font type, style, size.
- Value :Indicates if selected (True) or not (False). Only one option button in a group can be True. One button in each group of option buttons should always be initialized to True at design time.

7. List Boxes

- A list box displays a list of items from which the user can select one or more items.
- If the number of items exceeds the number that can be displayed, a scroll bar is automatically added.
- **List Box Properties:**

1. Appearance :Selects 3-D or flat appearance.
2. List :Array of items in list box.
3. ListCount :Number of items in list.
4. ListIndex :The number of the most recently selected item in list.If no item is selected, ListIndex = -1.
5. MultiSelect :Controls how items may be selected (0-no multiple selection allowed, 1-multiple selection allowed, 2-group selection allowed).
6. Selected :Array with elements set equal to True or False, depending on whether corresponding list item is selected.
7. Sorted :True means items are sorted in 'Ascii' order, else items appear in order added.
8. Text : Text of most recently selected item.

- **List Box Events:**
- Click :Event triggered when item in list is clicked.
- DblClick :Event triggered when item in list is double-clicked.Primary way used to process selection.

- **List Box Methods:**
- AddItem :Allows you to insert item in list.
- Clear :Removes all items from list box.
- RemoveItem :Removes item from list box, as identified by index of item to remove.
- **Examples**
- lstExample.AddItem "This is an added item" ' adds text string to list
- lstExample.Clear ' clears the list box
- lstExample.RemoveItem 4 ' removes lstExample.List(4) from list box

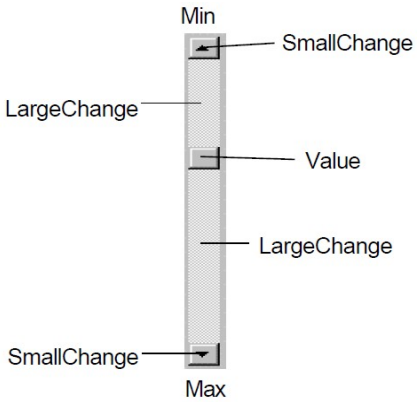
8.Combo Boxes

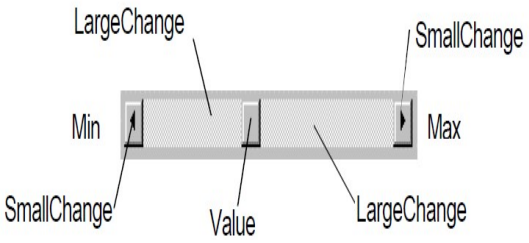
- The combo box is similar to the list box.
- The differences are a combo box includes a text box on top of a list box and only allows selection of one item.
- In some cases, the user can type in an alternate response.
- **Combo Box Properties:**
- Combo box properties are nearly identical to those of the list box, with the deletion of the MultiSelect property and the addition of a Style property.
- Appearance Selects 3-D or flat appearance.
- List Array of items in list box portion.
- ListCount Number of items in list.
- ListIndex The number of the most recently selected item in list.If no item is selected, ListIndex = -1.
- Sorted True means items are sorted in 'Ascii' order, else items appear in order added.
- Style change Selects the combo box form.Style = 0, Dropdown combo; user can selection.Style = 1, Simple combo; user can change selection. Style = 2, Dropdown combo; user cannot change selection.
- Text Text of most recently selected item.
- **Combo Box Events:**
- Click Event triggered when item in list is clicked.
- DblClick Event triggered when item in list is double-clicked. Primary way used to process selection.
- **Combo Box Methods:**
- AddItem Allows you to insert item in list.
- Clear Removes all items from list box.
- RemoveItem Removes item from list box, as identified by index of item to remove.
- **Examples**
- cboExample.AddItem "This is an added item" ' adds text string to list
- cboExample.Clear ' clears the combo box
- cboExample.RemoveItem 4 ' removes cboExample.List(4) from list box

9.Horizontal and Vertical Scroll Bars

- Horizontal and vertical scroll bars are widely used in Windows applications.
- Scroll bars provide an intuitive way to move through a list of information and make great input devices.
- **Scroll Bar Properties:**

1. LargeChange Increment added to or subtracted from the scroll bar Value property when the bar area is clicked.
2. Max The value of the horizontal scroll bar at the far right and the value of the vertical scroll bar at the bottom.Can range from -32,768 to 32,767.
3. Min The other extreme value - the horizontal scroll bar at the left and the vertical scroll bar at the top. Can range from -32,768 to 32,767.
4. SmallChange The increment added to or subtracted from the scroll bar Value property when either of the scroll arrows is clicked.
5. Value The current position of the scroll box (thumb) within the scroll bar. If you set this in code, Visual Basic moves the scroll box to the proper position.





10.Picture Boxes

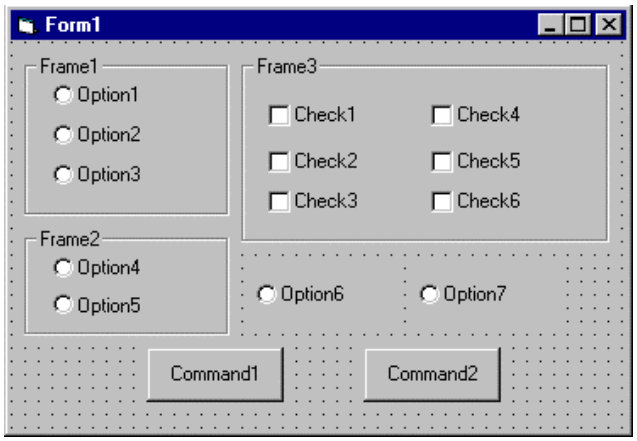
- The picture box allows you to place graphics information on a form.
- It is best suited for dynamic environments - for example, when doing animation.
- Picture boxes lie in the top layer of the form display.
- They behave very much like small forms within a form, possessing most of the same properties as a form.
- **Picture Box Properties:**

1. AutoSize If True, box adjusts its size to fit the displayed graphic.
2. Font Sets the font size, style, and size of any printing done in the picture box.
3. Picture Establishes the graphics file to display in the picture box.

- **Picture Box Events:**
  - Click Triggered when a picture box is clicked.
  - DblClick Triggered when a picture box is double-clicked.
- **Example**
- picExample.Picture = LoadPicture("c:\pix\sample.bmp")

11. Frames

- Frames provide a way of grouping related controls on a form. And, in the case of option buttons, frames affect how such buttons operate.
- Option buttons within a frame work as a group, independently of option buttons in other frames.
- Option buttons on the form, and not in frames, work as another independent group.
- That is, the form is itself a frame by default.
- It is important to note that an independent group of option buttons is defined by physical location within frames, not according to naming convention.
- That is, a control array of option buttons does not work as an independent group just because it is a control array.
- It would only work as a group if it were the only group of option buttons within a frame or on the form.



12.Shape Tool

- The shape tool can create circles, ovals, squares, rectangles, and rounded squares and rectangles.
- Colors can be used and various fill patterns are available.
- **Shape Tool Properties:**

1. BackColor Determines the background color of the shape (only used when FillStyle not Solid.
2. BackStyle Determines whether the background is transparent or opaque.
3. BorderColor Determines the color of the shape's outline.
4. BorderStyle Determines the style o the shape's outline. The border can be transparent, solid, dashed, dotted, and combinations.
5. BorderWidth Determines the width of the shape border line.
6. fillColor Defines the interior color of the shape.
7. FillStyle Determines the interior pattern of a shape. Some choices are: solid, transparent, cross, etc.
8. Shape Determines whether the shape is a square, circle, or some other choice.

13. The Line Control

- Like the shape control, the line control is a graphical control.
- Line is use it to display horizontal, vertical, or diagonal lines in a form.
- We can use these controls at design time as a design element or at runtime to alter the original line you drew.
- It can also change a line at runtime by changing its X1, X2, Y1, and Y2 properties.
- **Line Tool Properties:**
- BorderColor Determines the line color.
- BorderStyle Determines the line 'shape'. Lines can be transparent, solid, dashed, dotted,and combinations.
- BorderWidth Determines line width.

14.The Timer Control

- You use a timer control when you want to execute code at specific intervals.
- Many times, especially in using graphics, we want to repeat certain operations at regular intervals.
- The timer tool allows such repetition. The timer tool does not appear on the form while the application is running.
- **The Timer Properties:**
- Enabled Used to turn the timer on and off. When on, it continues to operate until the Enabled property is set to False.

  Interval Number of milliseconds between each invocation of the Timer Event

c) Result:

Thus the study of various Visual Basic tools was done.

QUESTIONS AND ANSWERS:

1.What is meant by front end tools?

Front end tools: are the tools which are used to design the user interface of the application Example Visual basic, .Net,Java,C,C++ etc.

2.What is meant by back end tools?

Back End tools: are the tools which are used to design data base(storage design) of the application Example SQL,Ms-Access,Oracle,DB2,SyBase etc.

Exercise Number: 12

Title of the Exercise : FORM
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To design and implement a form using visual basic

FACILITIES REQUIRED AND PROCEDURE

d) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	S/W Name	VB

e) Procedure for doing the experiment:

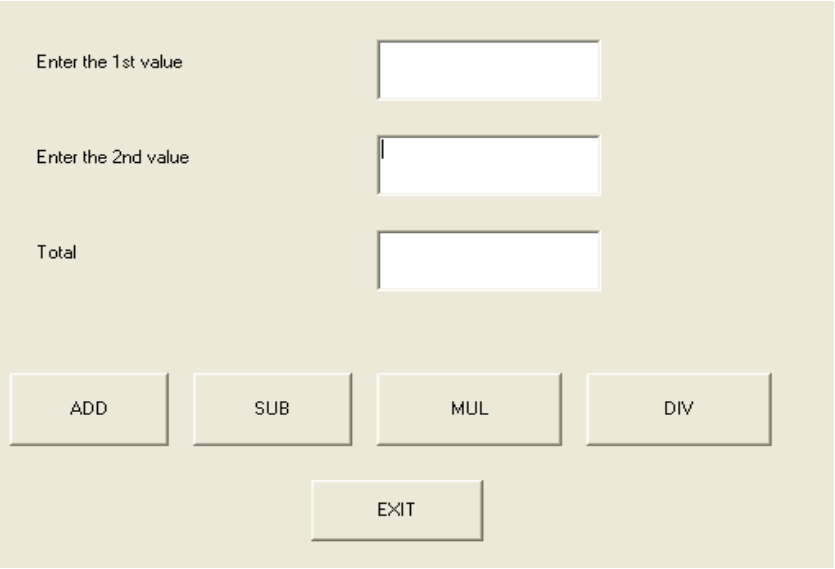
Step no.	Details of the step
1	Create a new project in visual basic using the option file---> new project.
2	In the form use the front end tools in the toolbox like textbox, label, and command button and create a front end Design for the simple calculator.
3	Open the properties window for the tools and select properties. Now the properties window is opened. Set properties for each tool in the form like caption, name, etc. Delete the default values of the text box
4	Double click each and every tool to open the project code window. write the code for the events of the tools
5	Double click the buttons and write the code for buttons
6	Write the code for the simple operations in the calculator like Addition, subtraction, multiplication and division. The code is Automatically compiled at the end of each line while pressing the Enter key.
7	Now execute the code by click the F5 button in the keyboard or select Run--->start. After successfully executing the project create the executable file by Select the option file---> make file.exe.

f) Program:

```
Private Sub Command1_Click()
Text3.Text = Val(Text1.Text) + Val(Text2.Text)
End Sub

Private Sub Command2_Click()
Text3.Text = Val(Text1.Text) - Val(Text2.Text)
End Sub

Private Sub Command3_Click()
Text3.Text = Val(Text1.Text) * Val(Text2.Text)
End Sub

Private Sub Command4_Click()
```

```
Text3.Text = Val(Text1.Text) / Val(Text2.Text)
End Sub


Private Sub Command5_Click()
End
End Sub
```

g) Output:



h) Result:

Thus the design and implementation of a form using visual basic was performed.

QUESTIONS AND ANSWERS

1. What does meant by ByRef argument in Procedure inVB?

ByRef sends a reference indicating where the value is stored in memory, allowing the called procedure to actually change the argument"s original value.

2. Give the steps required to convert a project into an .exe file

1. After completing project save it.
2. Then click on file menu.
3. In file menu there is an sub menu „Make Project".
4. After clicking on Make Project sub menu it will open a save dialog box to locate the path where .exe file will be created. Select the location and click on Ok then your project will be converted into .exe file at specified location.

3. What are objects and properties? How are they related to each other?

Objects:

Think of objects as a thing, or a noun. Examples of objects are forms and controls. Forms are the windows and dialog boxes you place on the screen; controls are the elements that you place inside a form, such as text boxes, command buttons, and list boxes.

Properties:

Properties tell something about an object, such as name, color, size, location, or how it will behave. You can think of properties as adjectives that describes objects. When you refer to a

property, you must first name the object, add a period, and then name the property. For example, refer to caption property of form called Form1 as Form1.Caption (say Form1.Caption).

Exercise Number: 13

Title of the Exercise : MENU DESIGN
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To design and implement a menu design using Visual Basic

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	S/W Name	VB

b) Procedure for doing the experiment:

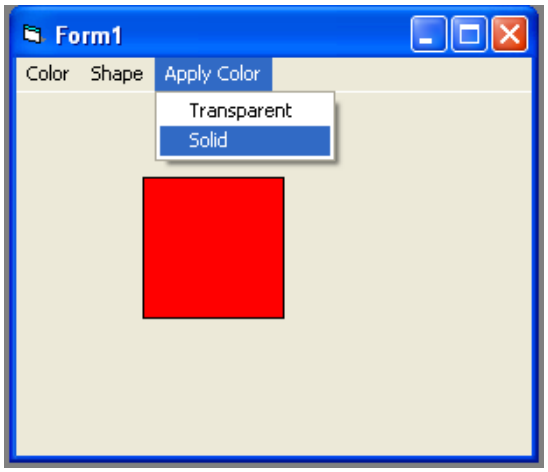
Step no.	Details of the step
1	Create a new project in VB
2	In the form window place the controls like Labels , text boxes in form
3	Go to **Tools → Menu Editor** or right click on the form and select Menu Editor.
4	In the Menu Editor dialog box enter the values for **caption**, **name** and **shortcut.**
5	Select **next** to add a new menu item and **delete** to remove a menu item.
6	To add a **sub menu** select the right button in the Menu Editor dialog box and then add the sub menus caption, name and shortcut.
7	Use "-" in the name field to add a separator line inside a menu. After adding all menu items select the **OK** button.
8	Double click the menu items in the form window to add the corresponding codes. Similarly the codes are given to the text boxes and labels.
9	Execute the project by selecting Run →Start. After the debugging and successful execution of the project save the project and make an executable file(*.exe) of it using the File menu.

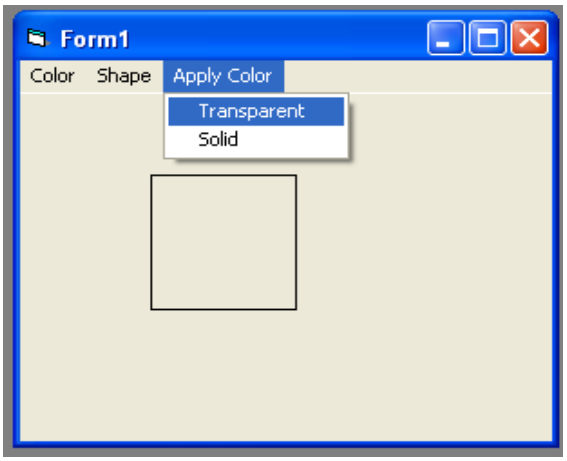
c) Program:

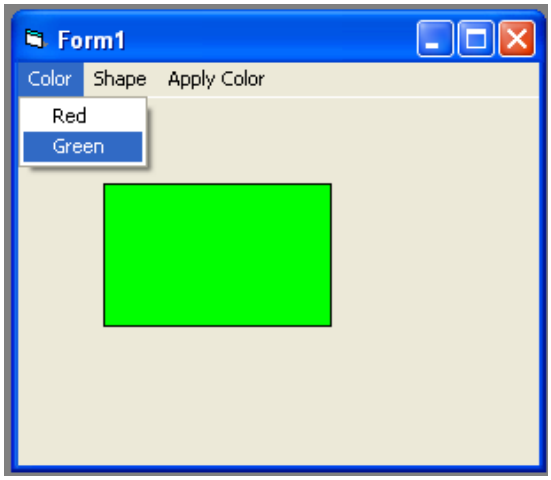
```
Private Sub Green_Click(Index As Integer)
Shape1.FillColor = RGB(0, 255, 0)
End Sub
Private Sub Oval_Click(Index As Integer)
Shape1.Shape = 2
End Sub
Private Sub Rect_Click(Index As Integer)
Shape1.Shape = 0
End Sub
Private Sub Red_Click(Index As Integer)
Shape1.FillColor = RGB(255, 0, 0)
End Sub
Private Sub Solid_Click(Index As Integer)
Shape1.FillStyle = 0
End Sub
```

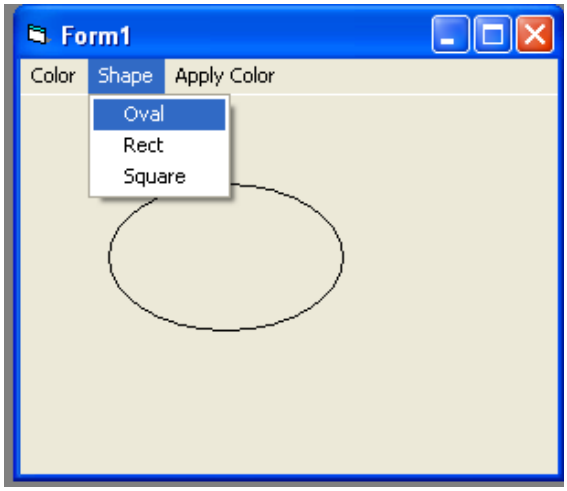
```
    Private Sub Square_Click(Index As Integer)
    Shape1.Shape = 1
    End Sub
    Private Sub Transparent_Click(Index As Integer)
    Shape1.FillStyle = 2
End Sub
```

d) Output:









e) Result:

Thus the menu design using Visual Basic was successfully designed and implemented.

QUESTIONS AND ANSWERS

1. Define menus

Menus consist of a menu bar with menu names, each of which drops down to display a list of menu commands. You can use menu commands in place of or addition to command buttons to active procedures. Menu commands are actually controls; they have properties and events. Each event has name property and click event, similar to action command buttons.

2. Define submenus.

The drop down list of commands below a menu name is called a menu. When a command on the menu has another list of commands that pops up, the new list is called submenu. The filled triangle to the right of the command indicates a menu command has a sub menu.

Exercise Number: 14

Title of the Exercise : REPORT GENERATION

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To generate report for employee details using VB.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

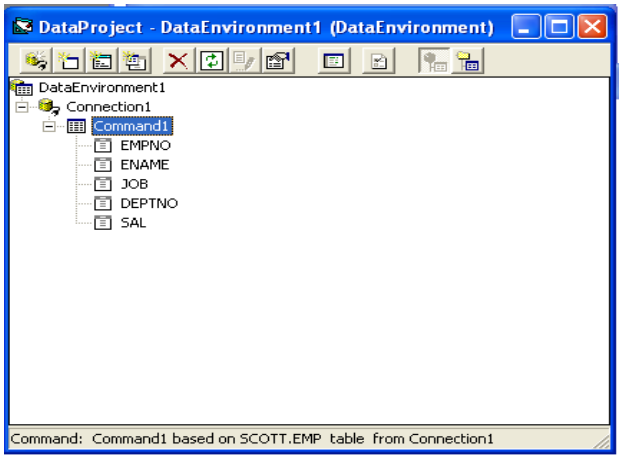
b) Procedure for doing the experiment:

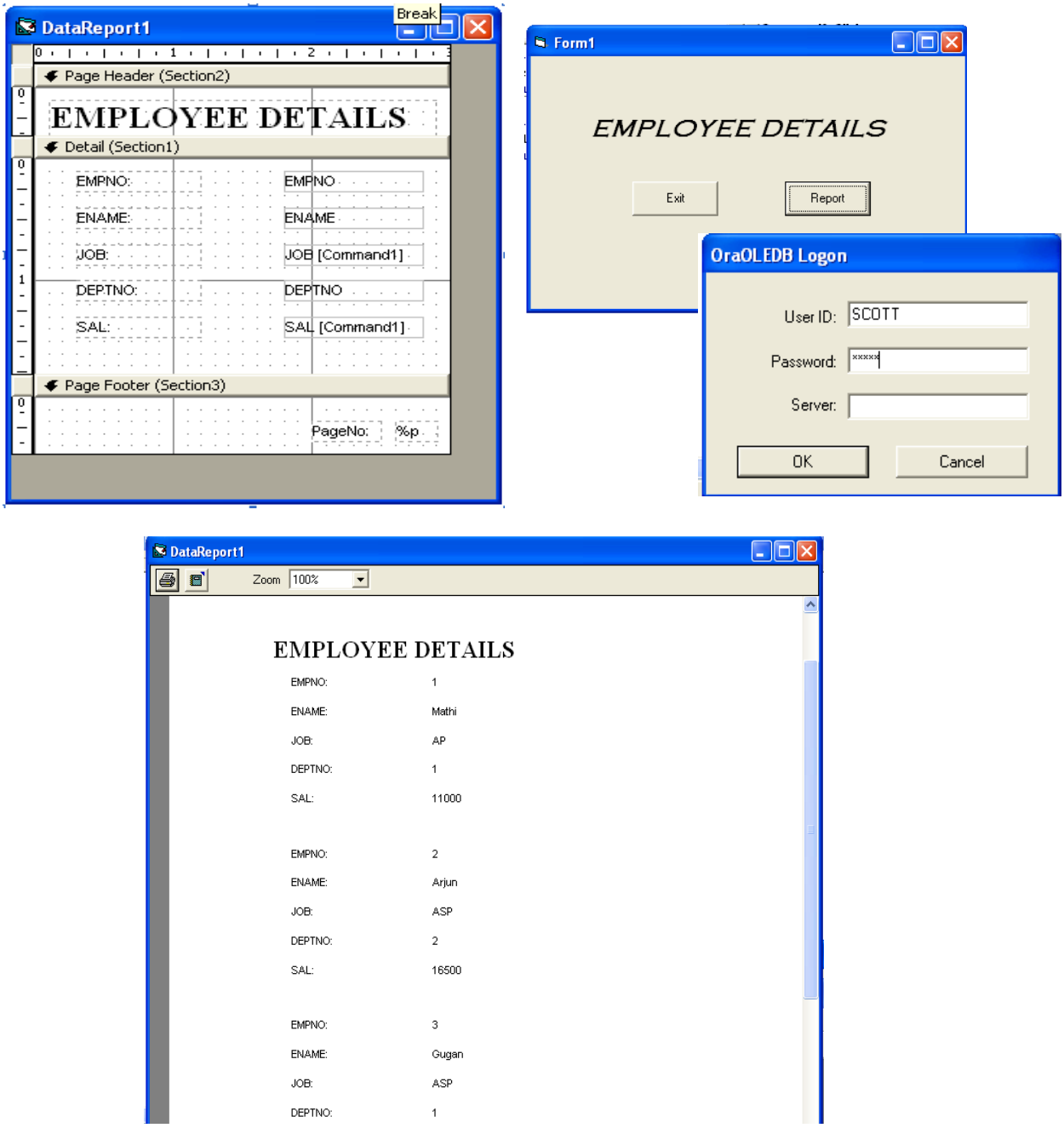
Step no.	Details of the step
1	Create a new VB project as data project.
2	Right click connection1 of the data environment click the property, set the provider as Oracle provider for OLEDB and click next then type user name as scott and password as tiger. Check whether the connection is established using test connection button.
3	Right click connection, click the Add command for the connection1 and set database object as table and object name as scott.emp
4	Drag command1and drop it in the data reports detail section1. Two items will appear for each object. The first label for heading information. Move this label into page header section. The label is used to store the actual value of the object.
5	In data report1 properties set data source data environment1 and the data member as command1. Place a button in the form write coding to view the report.

c) Program:

```
Private Sub Command1_Click()
DataReport1.Show
End Sub
Private Sub Command2_Click()
Unload Me
End Sub
```

d) Output:





e) Result:

Thus the report generation of employee details using Visual Basic was successfully developed.

QUESTIONS AND ANSWERS

1. Name the at least three types of common dialog boxes

1. Color Dialog Box
2. Font Dialog Box
3. Save Dialog Box

2. What does meant by ByVal in procedure in VB?

The ByVal sends a copy of the argument"s value to the procedure cannot alter the original value.

Exercise Number: 15

Title of the Exercise : DATABASE DESIGN AND IMPLEMENTATION

PAY ROLL PROCESSING

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To create a database for payroll processing system using SQL and implement it using VB

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Create a database for payroll processing which request the using SQL
2	Establish ODBC connection
3	In the administrator tools open data source ODBC
4	Click add button and select oracle in ORA home 90,click finish
5	A window will appear given the data source home as oracle and select TNS source name as lion and give the used id as SWTT
6	**ADODC CONTROL FOR SALARY FORM:-**
7	The above procedure must be follow except the table ,A select the table as salary
8	Write appropriate Program in form each from created in VB from each from created in VB form project.

c) Program:

SQL>create table emp(eno number primary key,enamr varchar(20),age number,addr varchar(20),DOB date,phno number(10));
Table created.
SQL>create table salary(eno number,edesig varchar(10),basic number,da number,hra number,pf number,mc number,met number,foreign key(eno) references emp);
Table created.

TRIGGER to calculate DA,HRA,PF,MC

```
SQL> create or replace trigger employ
  2 after insert on salary
  3 declare
  4 cursor cur is select eno,basic from salary;
  5  begin
  6 for cur1 in cur loop
  7 update salary set
  8 hra=basic*0.1,da=basic*0.07,pf=basic*0.05,mc=basic*0.03 where hra=0;
  9 end loop;
```

```
 10  end;
 11  /
Trigger created.
```

PROGRAM FOR FORM 1

```
Private Sub emp_Click()
Form2.Show
End Sub
Private Sub exit_Click()
Unload Me
End Sub
Private Sub salary_Click()
Form3.Show
End Sub
```

PROGRAM FOR FORM 2

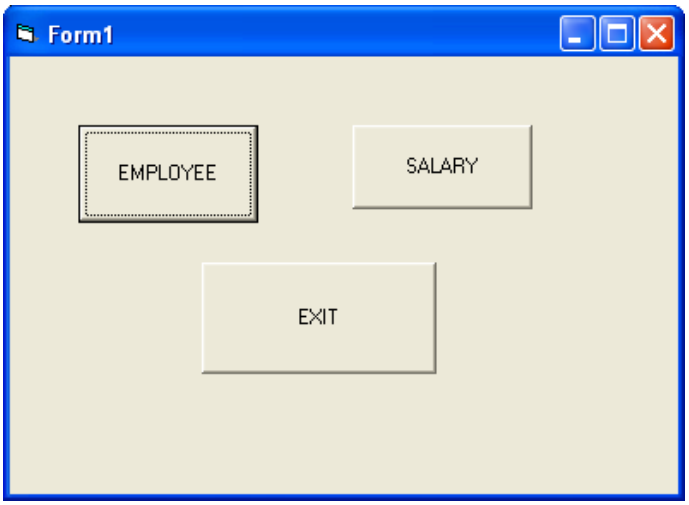
```
Private Sub add_Click()
Adodc1.Recordset.AddNew
MsgBox "Record added"
End Sub
Private Sub clear_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
End Sub
Private Sub delte_Click()
Adodc1.Recordset.Delete
MsgBox "Record Deleted"
If Adodc1.Recordset.EOF = True Then
Adodc1.Recordset.MovePrevious
End If
End Sub
Private Sub exit_Click()
Unload Me
End Sub
Private Sub main_Click()
Form1.Show
End Sub
Private Sub modify_Click()
Adodc1.Recordset.Update
End Sub
```

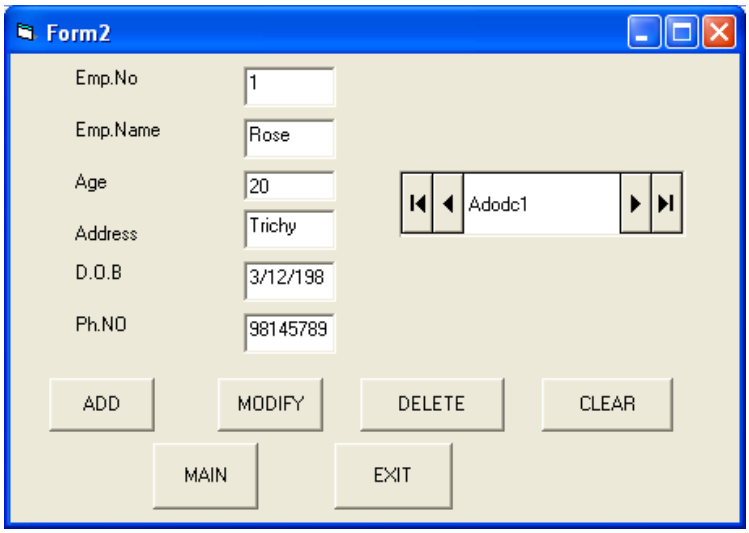
PROGRAM FOR FORM 3

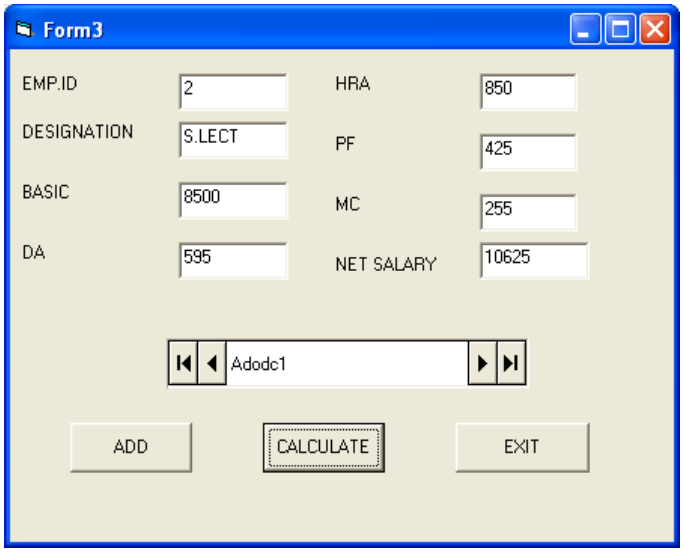
```
Private Sub add_Click()
Adodc1.Recordset.AddNew
MsgBox "Record added"
End Sub
Private Sub clear_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
```

```
End Sub
Private   Sub   delte_Click()
Adodc1.Recordset.Delete
MsgBox "Record Deleted"
If Adodc1.Recordset.EOF = True Then
Adodc1.Recordset.MovePrevious
End If
End Sub
Private Sub exit_Click()
Unload Me
End Sub
Private Sub main_Click()
Form1.Show

End Sub
Private Sub modify_Click()
Adodc1.Recordset.Update
End Sub
```

d) Output:







e) Result:

Thus the design and implementation of payroll processing system using SQL, VB was successfully done.

QUESTIONS AND ANSWERS

1. What is the purpose of Visual Basic file types: .vbp, .frm, .bas, and .ocx?

Visual Project consist of at least two, and usually more, files as follows:

.vbp file: This file is called the project file, is a small text file that holds the names of the other files in the project, as well as some information about the VB environment.

.frm file: Each of your form in the project is saved in a file with extension. To begin your project your project will have only one form. Later you can expect your projects to have several forms, with one .frm file for each form. A from holds the description of all objects and their properties for each form, as well as the basic code that you have written to respond to the events. These are also referred as form modules.

.bas file: Optionally your project can have this file. These file holds basic statements that can be accessed from any form. As soon as you begin .bas file are called standard code modules.

.ocx file: additional controls, called custom controls, are stored with a file .ocx extension. If you include controls in your projects that are not part of the standard control set, the .ocx file names will be included in the project.

2. What is ODBC?

Open Data Base Connectivity is an administrative tool which is used to connect different types of databases through data source name.

3. What is use of ADODC?

ActiveX Data Object Data Control is a data control which is used to connect Visual basic controls to a database. That is, it connects the front end data with the back end.

Exercise Number: 16

Title of the Exercise : BANKING SYSTEM

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To develop a mini project for banking system.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows XP
3	Front end	VB/VC ++/JAVA
4	Back end	Oracle11g,my SQL, DB2

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Create the DB for banking system source request the using SQL
2	Establishing ODBC connection
3	Click add button and select oracle in ORA home 90 click finished
4	A window will appear give the data source name as oracle and give the user id as scott
5	Now click the test connection a window will appear with server and user name give user as scott and password tiger Click ok
6	**VISUAL BASIC APPLICATION:-** • Create standard exe project in to and design ms from in request format • To add ADODC project select component and check ms ADO data control click ok • Now the control is added in the tool book • Create standard exe project in to and design ms from in request format
7	**ADODC CONTEOL FOR ACCOUNT FROM:-** Click customs and property window and window will appear and select ODBC data source name as oracle and click apply as the some window.

c) Program:

CREATE A TABLE IN ORACLE

```
SQL>create table account(cname   varchar(20),accno  number(10),balance  number);
Table Created
SQL> insert into account values('&cname',&accno,&balance);
Enter value for cname: Mathi
Enter value for accno: 1234
Enter value for balance: 10000
old   1: insert into account values('&cname',&accno,&balance)
new   1: insert into emp values('Mathi',1234,10000)         1 row created.
```

SOURCE CODE FOR FORM1

```
Private Sub ACCOUNT_Click()
Form2.Show
End Sub
Private Sub EXIT_Click()
Unload Me
End Sub
```

```
Private Sub TRANSACTION_Click()
Form3.Show
End Sub
```

SOURCE CODE FOR FORM 2

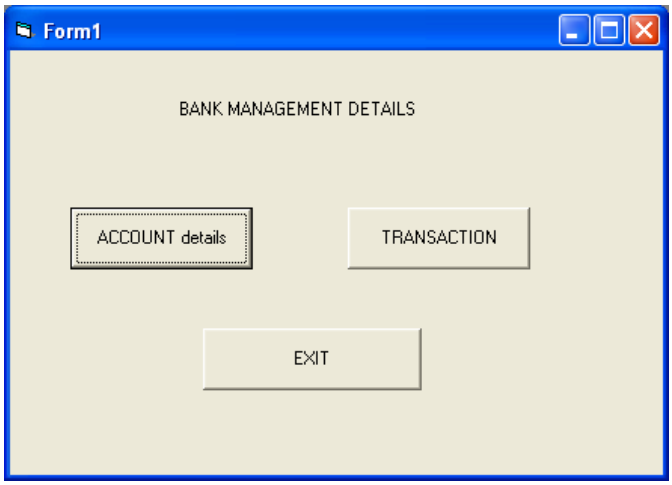
```
Private Sub CLEAR_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
End Sub

Private Sub DELETE_Click()
Adodc1.Recordset.DELETE
MsgBox "record deleted"
Adodc1.Recordset.MoveNext
If Adodc1.Recordset.EOF = True Then
Adodc1.Recordset.MovePrevious
End If
End Sub
Private Sub EXIT_Click()
Unload Me
End Sub
Private Sub HOME_Click()
Form1.Show
End Sub
Private Sub INSERT_Click()
Adodc1.Recordset.AddNew
End Sub
Private Sub TRANSACTION_Click()
Form3.Show
End Sub
Private Sub UPDATE_Click()
Adodc1.Recordset.UPDATE
MsgBox "record updated successfully"
End Sub
```

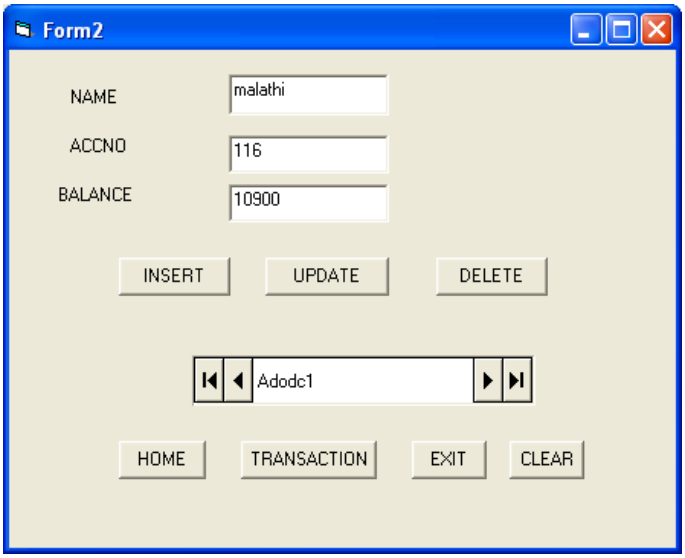
SOURCE CODE FOR FORM 3

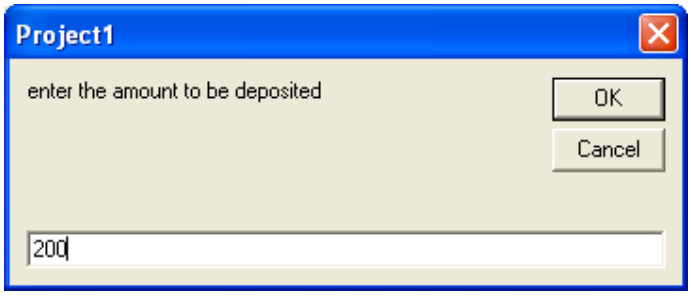
```
Private Sub ACCOUNT_Click()
Form2.Show
End Sub

Private Sub CLEAR_Click()
Text1.Text = ""
Text2.Text = ""
End Sub

Private Sub DEPOSIT_Click()
Dim s As String
s = InputBox("enter the amount to be deposited")
Text2.Text = Val(Text2.Text) + Val(s)
A = Text2.Text
MsgBox "CURRENT BALANCE IS Rs" + Str(A)
Adodc1.Recordset.Save
Adodc1.Recordset.UPDATE
```

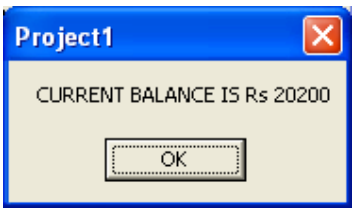
```
End Sub
Private Sub EXIT_Click()
Unload Me
End Sub

Private Sub HOME_Click()
Form1.Show
End Sub

Private Sub WITHDRAW_Click()
Dim s As String
s = InputBox("enter the amount to be deleted")
Text2.Text = Val(Text2.Text) - Val(s)
A = Text2.Text
MsgBox "current balance is Rs" + Str(A)
Adodc1.Recordset.Save
Adodc1.Recordset.UPDATE
End Sub
```

d) Output:









e) Result:

Thus the mini project for banking system by using VB,SQL was done.

QUESTIONS AND ANSWERS

1. What is a purpose of image control and how can it be created?

An image control holds a picture. You can set an image picture property to file with an extension of .bmp, GIF, .JPEG etc.

First place the image control on a form and then select its picture property in the properties window. Click on the properties button to display load picture dialog box were you can select a filename. You can use any picture file (with the property format) that you have available.

Set the Stretch property of image control to true to make the picture enlarge to filled the control You can set the visible property to false to make the image disappear. The letter prefix for naming an image is image.

For Example, to make an image invisible at run time, use this code statement:

imgLogo. Visible = False

2. How can you make groups of option buttons?

Option Button allows user to select one of several choices. Unlike CheckBoxes,The user can select one and only one option button at a time.

After selecting one option button if the user click another option button. VB deselect the previously selected option button and select the one the user click not that option but is also called as Radio Button.