Table of Content

Department of AIML		
Average Performance	**25**	
Average Record	**15**	
Average Viva	**10**	
Total	**50**	

Ex. No: 01 Date:	Creation of a database and SQL queries to retrieve information from the database

Aim:

To Create database and to retrieve table using DDL Query.

DDL Commands

- Create Table
- Alter Table
- Drop Table
- Truncate Table
- Rename

Table Creation

Table is a primary object of database, used to store data in form of rows and columns. It is created using following command:

Syntax: CREATE TABLE <table_name> (column_name data_ type (constraints), …)

Desc Command

Describe command is external command of Oracle. The describe command is used to view the structure of table as follows.

Syntax: desc <table_name>

Alter Table:

Add a column

Syntax: Alter table <table_name> add (new_column_name data_ type (constraints), …)

Modify a Column

Syntax: Alter Table <Table Name>Modify Column <Column Name> <New Datatype>(<New Size>);

Drop Table:

Drop A Column:

Syntax: Alter Table <Table Name>Drop Column <Column Name>;

Drop Entire Table

Syntax:

Drop table <Table_Name>

Truncate Table

If there is no further use of records stored in a table and the structure is required then only data can be deleted using truncate command. Truncate command will delete all the records permanently of specified table

Syntax:

Truncate table <table name>

Rename a table

Rename command is used to give new names for existing tables.

Syntax:

Rename table old table_name TO new table_name;

Constraints

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax

```
CREATE TABLE table_name (
   column1 datatype constraint,
   column2 datatype constraint,
   column3 datatype constraint,
   ....
);
```

Add primary Key to table

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

Example

CREATE TABLE Persons (ID int NOT NULL PRIMARY KEY, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int);	CREATE TABLE Persons (ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int, CONSTRAINT PK_Person PRIMARY KEY (ID,LastName));

Add primary key to Existing Table

Example

- ALTER TABLE Persons ADD PRIMARY KEY (ID);
- ALTER TABLE Persons ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);

DROP a PRIMARY KEY Constraint

Example

- ALTER TABLE Persons DROP PRIMARY KEY;
- ALTER TABLE Persons DROP CONSTRAINT PK_Person;

FOREIGN KEY on CREATE TABLE

Example

```
CREATE TABLE Orders (
   OrderID int NOT NULL PRIMARY KEY,
   OrderNumber int NOT NULL,
   PersonID
int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

```
CREATE TABLE Orders (
   OrderID int NOT NULL,
   OrderNumber int NOT NULL,
   PersonID int,
   PRIMARY KEY (OrderID),
   CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
   REFERENCES Persons(PersonID)
);
```

FOREIGN KEY on ALTER TABLE

Example

- ALTER TABLE Orders ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

- ALTER TABLE Orders ADD CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

DROP a FOREIGN KEY Constraint

Example

- ALTER TABLE Orders DROP FOREIGN KEY FK_PersonOrder;
- ALTER TABLE Orders DROP CONSTRAINT FK_PersonOrder;

Datatype

CHAR(size)	It is used to store character data within the predefined length. It can be stored up to 2000 bytes.
NCHAR(size)	It is used to store national character data within the predefined length. It can be stored up to 2000 bytes.
VARCHAR2(size)	It is used to store variable string data within the predefined length. It can be stored up to 4000 byte.
VARCHAR(SIZE)	It is the same as VARCHAR2(size). You can also use VARCHAR(size), but it is suggested to use VARCHAR2(size)
NVARCHAR2(size)	It is used to store Unicode string data within the predefined length. We have to must specify the size of NVARCHAR2 data type. It can be stored up to 4000 bytes.
NUMBER(p, s)	It contains precision p and scale s. The precision p can range from 1 to 38, and the scale s can range from -84 to 127.
FLOAT(p)	It is a subtype of the NUMBER data type. The precision p can range from 1 to 126.
BINARY_FLOAT	It is used for binary precision(32-bit). It requires 5 bytes, including length byte.
BINARY_DOUBLE	It is used for double binary precision (64-bit). It requires 9 bytes, including length byte.
DATE	It is used to store a valid date-time format with a fixed length. Its range varies from January 1, 4712 BC to December 31, 9999 AD.
TIMESTAMP	It is used to store the valid date in YYYY-MM-DD with time hh:mm:ss format.
BLOB	It is used to specify unstructured binary data. Its range goes up to 2^{32}-1 bytes or 4

	GB.
BFILE	It is used to store binary data in an external file. Its range goes up to 2^{32}-1 bytes or 4 GB.
CLOB	It is used for single-byte character data. Its range goes up to 2^{32}-1 bytes or 4 GB.
NCLOB	It is used to specify single byte or fixed length multibyte national character set (NCHAR) data. Its range is up to 2^{32}-1 bytes or 4 GB.
RAW(size)	It is used to specify variable length raw binary data. Its range is up to 2000 bytes per row. Its maximum size must be specified.
LONG RAW	It is used to specify variable length raw binary data. Its range up to 2^{31}-1 bytes or 2 GB, per row.

Activity 1

Able to apply SQL commands to create database and table structure.

Activity 2

A database named studentDB will developed by an application software house company. There are 6 tables in the database. Relationship scheme for the tables is as below:

Student(StudID, StudName, StudAddress, StudBirthDate, CourseID);

Course(CourseID, CourseName, LectID);

Lecturer(LectID, LectName, LectTelNo, DepartID);

Subject(SubID, SubName);

Department(DepartID, DepartName);

Stud_Sub (StudSubID, StudID, SubID, Mark,Grade)

(Primary Key showed by underlined and bold while in italic and dotted lined is Foreign Key)

Data Dictionary for the tables is as below:

Tables Name	Attributes Name	Data Types	PK/FK	FK refer to
STUDENT	StudID	VARCHAR(5)	PK	
	StudName	VARCHAR(20)		
	StudAddress	VARCHAR(30)		
	StudBirthDate	DATE		
	CourseID	VARCHAR(5)	FK	COURSE
COURSE	CourseID	VARCHAR(5)	PK	
	CourseName	VARCHAR(10)		
	LectID	VARCHAR(5)	FK	LECTURER
LECTURER	LectID	VARCHAR(5)	PK	
	LectName	VARCHAR(20)		
	LectTelNo	INT(10)		
	DepartID	VARCHAR(5)	FK	DEPARTMENT
DEPARTMENT	DepartID	VARCHAR(5)	PK	
	DepartName	VARCHAR(30)		
SUBJECT	SubID	VARCHAR(5)	PK	
	SubName	VARCHAR(20)		
STUD_SUB	StudSubID	VARCHAR(5)	PK	
	StudID	VARCHAR(5)	FK	STUDENT
	SubID	VARCHAR(5)	FK	SUBJECT
	Mark	INT(3)		
	Grade	CHAR(2)		

Refer to the information above, write SQL statements for the questions below:

1. Create database studentDB.
2. Create 6 tables from data dictionary given above.
3. Add one column StudTelNo in STUDENT table and the data type is INT(9).
4. Alter data type of CourseName column for COURSE table to VARCHAR(30).
5. Drop table Stud_Sub from database

Viva Questions

1. Define data and information.
2. Define Data base management system.
3. What is SQL?
4. What is the syntax for creating a table?
5. List the components of SQL.
6. Define DDL? What are the DDL commands?
7. List out the uses of alter command.
8. What is Syntax for truncate a table?
9. What is the use drop table command?

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result: Thus, the database has been created and retrieved the information using SQL Queries.

Ex. No: 02 Date:	Performing Insertion, Deletion, Modifying, Altering, Updating and Viewing records based on conditions.

Aim: To Perform Insertion, Deletion, Modifying, Altering, Updating and Viewing records based on conditions.

DML Commands

- **Insert**
- **Select**
- **Update**
- **Delete**

Insert a data into table

Insert command is used to insert rows into the table.

Syntax

- INSERT INTO tablename values (columnname1, columnname2,….columnname n)
- INSERT INTO tablename (columnname1, columnname2,….columnname n) VALUES(Value1,Value2,..Value n);
- INSERT INTO table _name values ('&column_name1 ',&Column_name2, '&Column_name3' , '&Column_Name'…….);

To retrieve / display data from tables

Select command is used to select values or data from table

Syntax: SELECT * FROM TABLENAME;

Elimination of duplicates from the select statement

SELECT DISTINCT columnname 1, columnname 2,…. columnname n FROM tablename;

The retrieving of specific columns from a table

SELECT columnname 1, columnname 2,…. columnname n FROM tablename;

Update a data in the table

This SQL command is used to modify the values in an existing table.

Syntax

UPDATE tablename SET column1= expression1, column2= expression 2,... WHERE somecolumn=somevalue;

Note:

- An expression consists of either a constant (new value), an arithmetic or string operation or an SQL query. Note that the new value to assign to must matching data type.

- An update statement used without a where clause results in changing respective attributes of all tuples in the specified table.

Delete rows from a table

Syntax

DELETE FROM table_name WHERE condition;

Task:

Execute the Data Manipulation commands for Experiment No. 01, which you completed in the previous class. Refer above mentioned syntax

Viva Questions

1. What are the DML commands?
2. How the data or values to be entered into a table?
3. What is the use of DELETE command?
4. How the data or values to be updated on a table?
5. List out the uses of SELECT command?
6. How the data or values are retrieved from a table?
7. Define DML? What are the DML commands?

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result: Thus, the database has been created and executed the DML commands using SQL Queries.

Ex. No: 03 Date:	Creation of Views, Synonyms, Sequence, Indexes, Save point

Aim:

To implement the Views, Synonyms, Sequence, Indexes, save point using SQL

VIEWS

Manipulation Operation in View

1. Create view
2. Insert in view
3. Delete in view
4. Update of view
5. Drop of view

Syntax

Create View: This command is used to create a view.

```
CREATE VIEW view_name AS SELECT column_name(s)
FROM table_name
WHERE condition;
```

Update View: This command is used to update a view.

```
CREATE OR REPLACE VIEW view_name AS SELECT column_name(s)
FROM table_name
WHERE condition;
```

Drop View: This command is used to drop the view table

```
DROP VIEW view_name;
```

Inserting a row from View

```
INSERT INTO view_name(column1, column2 , column3,..) VALUES(value1, value2, value3..);
```

Deleting a row from a View

```
DELETE FROM view_name WHERE condition;
```

SYNONYM

Provides an alternative name for another database object, referred to as the base object, that can exist on a local or remote server**.**

Synonyms are used to create alternate names for tables, views, sequences ... etc.

The syntax for this is:

```
CREATE [PUBLIC] SYNONYM synname FOR objectname;
CREATE SYNONYM Emp FOR Employee;
CREATE SYNONYM Cstd FOR Customer_Details;
SELECT * FROM Cstd;
```

Viewing the details of a user synonyms

```
SELECT synonym_name, table_name, table_owner FROM USER_SYNONYMS;
```

Dropping a Synonym- A Synonym can be dropped as:

Syntax: DROP SYNONYM synonymname;

Example:

```
DROP SYNONYM Emp;
```

SEQUENCE

Sequence is a set of integers 1, 2, 3, … that are generated and supported by some database systems to produce unique values on demand.

- A sequence is a user defined schema bound object that generates a sequence of numeric values.
- Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provides an easy way to generate them.
- The sequence of numeric values is generated in an ascending or descending order at defined intervals and can be configured to restart when exceeds max_value.

Syntax:

```
CREATE SEQUENCE sequence_name
START WITH initial_value
INCREMENT BY increment_value
MINVALUE minimum value
MAXVALUE maximum value
CYCLE|NOCYCLE ;
```

Example:

```
CREATE SEQUENCE sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle;
```

Table Creation

```
CREATE TABLE students
(
ID number(10),
NAME char(20)
);
```

insert values into table

```
INSERT into students VALUES (sequence_1.nextval,'Ramesh');
INSERT into students VALUES(sequence_1.nextval,'Suresh');
```

INDEXES

Index is used for faster retrieval of rows from a table. It can be used implicitly or explicitly.

Mainly, index is of two types:

Simple Index - It is created on a single column.

Syntax

CREATE INDEX indexname ON tablename(column);

Example

CREATE INDEX idx ON Student (cgpa);

Complex Index- It is created on more than one column.

Syntax

CREATE INDEX indexname ON tablename(columns);

Example

CREATE INDEX ids ON Student (first, last);

Unique Index - unique index does not allow any duplicate values to be inserted into the table.

Syntax

CREATE UNIQUE INDEX index_name on table_name (column_name);

Drop Index

Syntax

DROP INDEX index_name;

Display

SHOW INDEX FROM tablename;

SAVE POINT - A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

Syntax

SAVEPOINT <SAVE POINT NAME>;

ROLLBACK- The ROLLBACK command is used to undo a group of transactions.

Syntax

ROLL BACK ; or ROLLBACK <SAVE POINT NAME>;

Viva Questions

1. What do database languages do?
2. What is SQL?
3. Difference between CHAR and VARCHAR?
4. What is the difference between primary key and candidate key?
5. What happens when the column is set to AUTO INCREMENT and if you reach maximum value in the table?
6. How can you see all indexes defined for a table?

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result: Thus, the implementation of view in SQL is performed and output is verified.

Ex. No: 04 Date:	Creating an Employee database to set various constraints.

Aim:

To create Employee database to set various constraints using SQL

SQL Constraints:

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

Syntax

CREATE TABLE table_name *(*

column1 datatype *constraint,*

column2 datatype *constraint,*

column3 datatype *constraint,*

....

);

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Prevents actions that would destroy links between tables
- **CHECK** - Ensures that the values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column if no value is specified
- **CREATE INDEX** - Used to create and retrieve data from the database very quickly

NOT NULL Constraint

- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.

UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.

- Can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

PRIMARY KEY Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist' of single or multiple columns (fields).

FOREIGN KEY Constraint

- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table

CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

DEFAULT Constraint

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

CREATE INDEX Statement

- The CREATE INDEX statement is used to create indexes in tables.
- Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

Viva Question

1. To include integrity constraint in a existing relation use:
   A. Create table
   B. Modify table
   C. Alter table
   D.

2. Which of the following is not a integrity constraint?
   A. Not null
   B. Positive
   C. Unique
   D. Check 'predicate'

3. Foreign key is the one in which the _____ of one relation is referenced in another relation.
   A. Foreign key
   B. Primary key
   C. References
   D. Check constraint

4. Constraints regarding integrity are ___ that must be followed.
   A. Data
   B. Rules
   C. Tables
   D. None

5. Information ___ is maintained by Integrity Constraints.
   E. Quantity
   F. Data
   G. Quality
   H. Flow

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result

Thus, the Employee database has been created with various constraints and output is verified.

Ex. No: 05 Date:	Creating relationship between the databases

Aim

To create a databases and implement the relationships between the databases.

Definition

A r**elationship** between two database tables presupposes that one of them has a foreign key that references the primary key of another table.

An **entity-relationship diagram,** also known as ERD, ER diagram, or ER model, comprises a graphical representation of how entities relate to each other within a database. ER models are widely used in database design as they are fairly abstract and are easy to view and analyze.

Types of relationships in a database

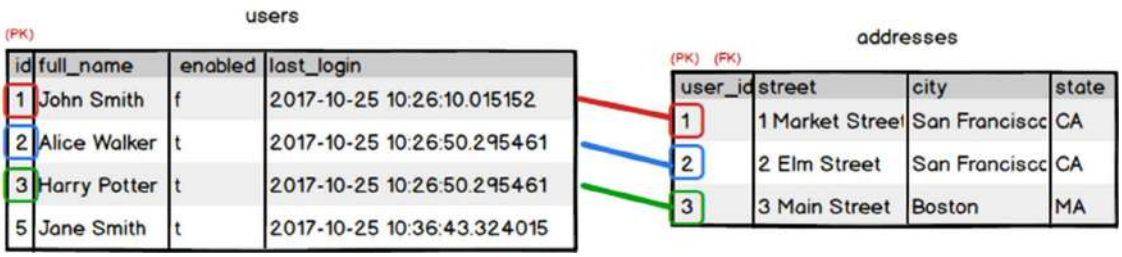
There are 3 main types of relationship in a database:

- one-to-one
- one-to-many
- many-to-many.

One to One

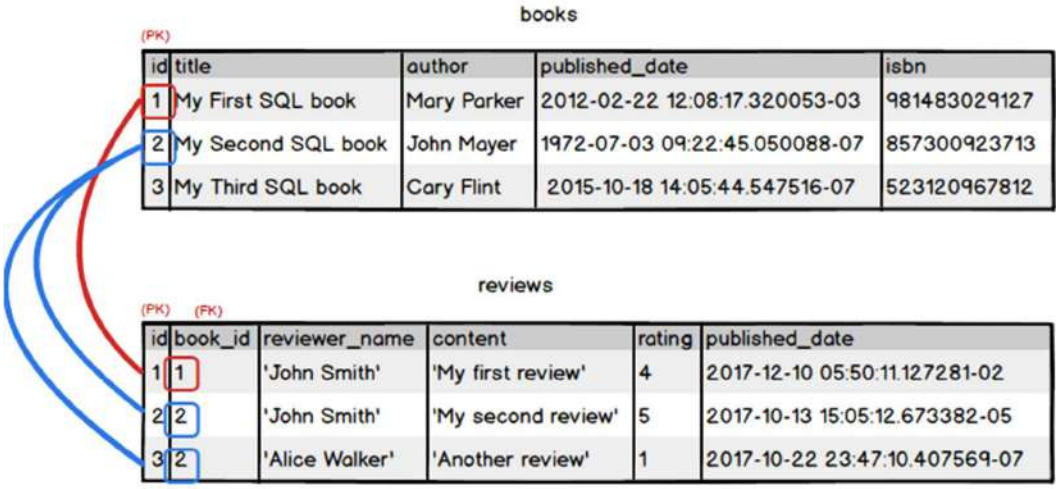
A one-to-one relationship between two entities exists when a particular entity instance exists in one table, and it can have only one associated entity instance in another table.

Example: A user can have only one address, and an address belongs to only one user.



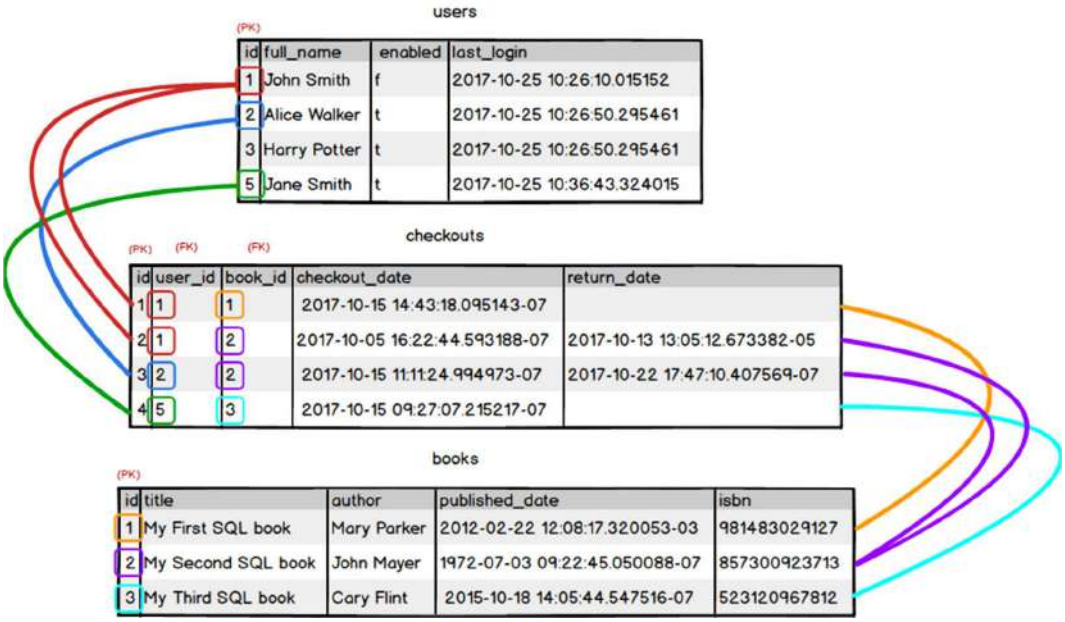
One to Many

A one-to-many relationship exists between two entities if an entity instance in one of the tables can be associated with multiple records (entity instances) in the other table. The opposite relationship does not exist; that is, each entity instance in the second table can only be associated with one entity instance in the first table.



Many-to-Many

A many-to-many relationship exists between two entities if for one entity instance there may be multiple records in the other table, and vice versa.



Viva Question

1. What is a relation in a database?
2. What is data redundancy?
3. What are the types of keys in relational databases?
4. What is Normalization?
5. What is data abstraction? Explain the levels of abstraction.

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result

Thus, the databases has been created with the relationships using SQL.

Ex. No: 06 Date:	Study of PL/SQL

Aim:

To study about PL/ SQL

PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java**.**

The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database. Following are certain notable facts about PL/SQL −

- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in, interpreted and OS independent programming environment.
- PL/SQL can also directly be called from the command-line SQL*Plus interface.
- Direct call can also be made from external programming language calls to database.
- PL/SQL's general syntax is based on that of ADA and Pascal programming language.
- Apart from Oracle, PL/SQL is available in TimesTen in-memory database and IBM DB2.

Features of PL/SQL

PL/SQL has the following features −

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports the development of web applications and server pages.

Advantages of PL/SQL

PL/SQL has the following advantages −

- SQL is the standard database language and PL/SQL is strongly integrated with SQL. PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. In Dynamic SQL, SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.
- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.

- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for developing Web Applications and Server Pages.

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result
Thus, the PL/SQL has been studied.

Ex. No: 07 Date:	PL/SQL block to satisfy some conditions by accepting input from the user.

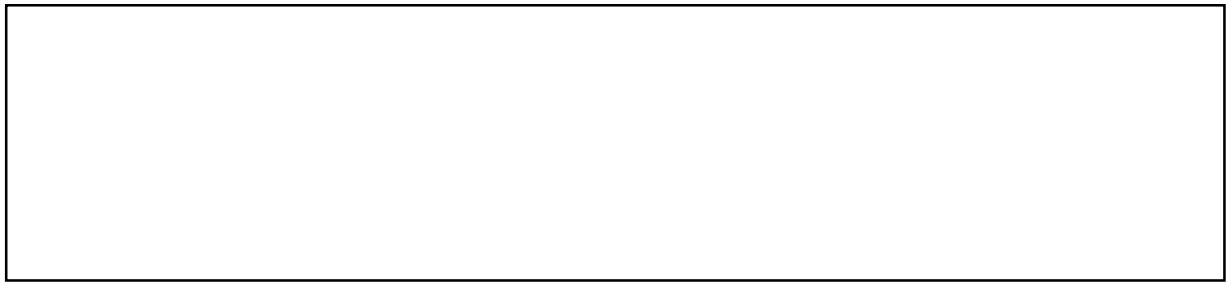
Aim

To implement simple control structure and cursor programs using PL/SQL.

Program:

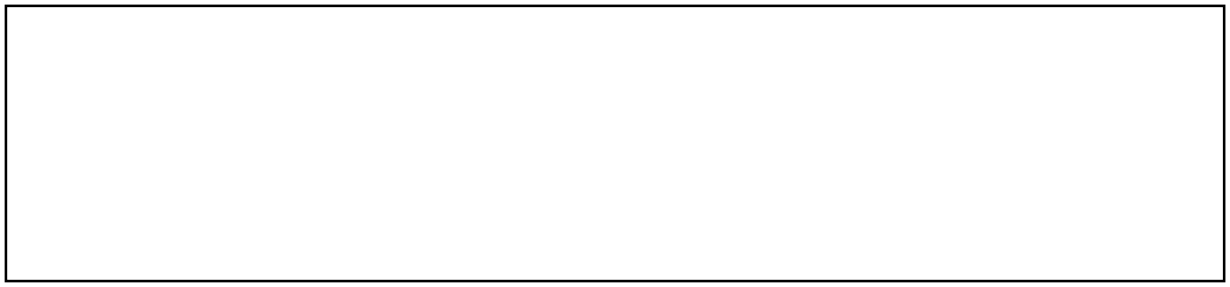
REVERSING THE NUMBERS

```
SQL>  set serveroutput on
SQL> declare
 given_number varchar(5):='567';
 str_length number(2);
 inverted_number varchar(5);
 begin
 str_length:=length(given_number);
 for cntr in reverse 1..str_length
 loop
 inverted_number:=inverted_number||substr(given_number,cntr,1);
end loop;
dbms_output.put_line('the given no is'||given_number);
dbms_output.put_line('the inverted number is'|| inverted_number);
end;
SQL> /
```



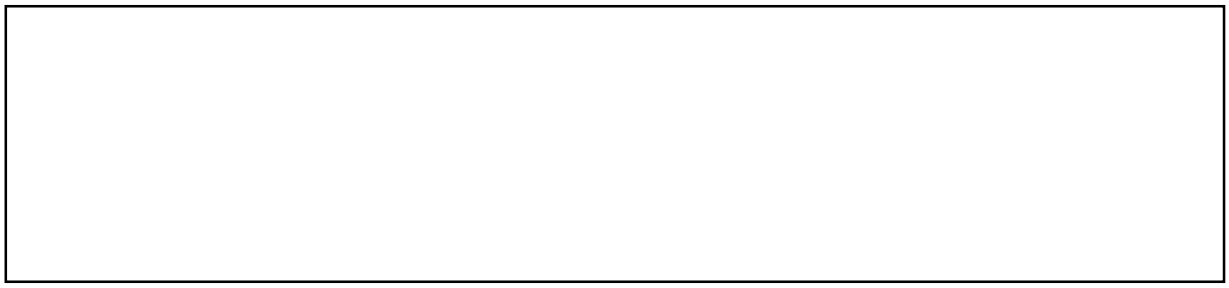
SUM OF 100 NUMBERS

```
SQL>  set serveroutput on
SQL> declare
  a number;
  s1 number default 0;
  begin
  a:=1;
  loop
  s1:=s1+a;
 exit when (a=100);
  a:=a+1;
 end loop;
dbms_output.put_line('sum between 1 and 100 is'|| s1);
 end;
SQL> /
```



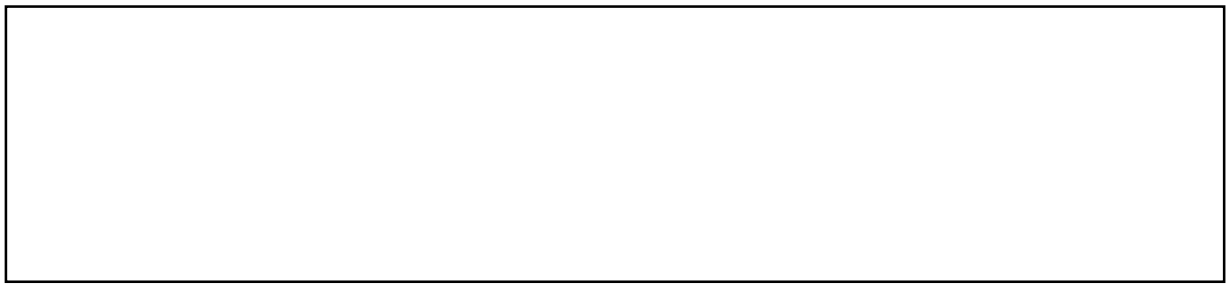
SUM OF ODD NUMBER USING USER INPUT

```
SQL>  set serveroutput on
SQL> declare
  n number;
  sum1 number default 0;
 endvalue number;
  begin
  endvalue:=&endvalue;
  n:=1;
  for n in 1..endvalue
  loop
 if mod(n,2)=1
 then
 sum1:=sum1+n;
 end if;
 end loop;
 dbms_output.put_line('sum ='||sum1);
 end;
SQL> /
```



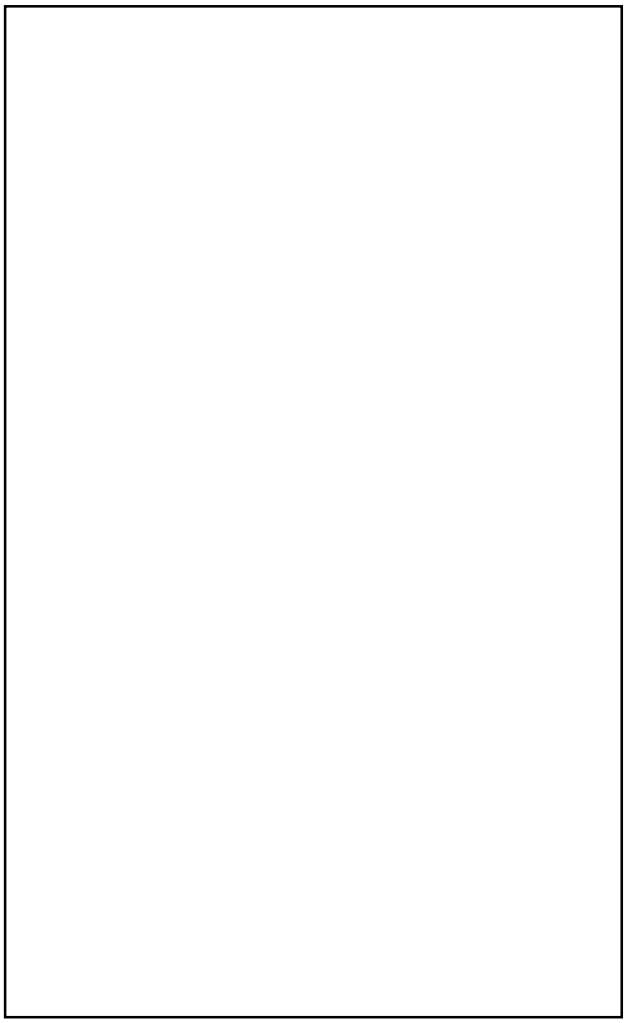
SUM OF ODD NO USING WHILE LOOP

```
SQL>  set serveroutput on
SQL>  declare
n number;
  sum1 number default 0;
endvalue number;
  begin
   endvalue:=&endvalue;
    n:=1;
   while(n<endvalue)
   loop
   sum1:=sum1+n;
   n:=n+2;
 end loop;
dbms_output.put_line('sum of odd no. bt 1 and' ||endvalue||'is'||sum1);
end;
/
```



CALCULATE THE NET SALARY

```
SQL>  set serveroutput on
SQL>  declare
  ename varchar2(15);
 basic number;
da number;
hra number;
pf number;
netsalary number;
begin
ename:=&ename;
 basic:=&basic;
 da:=basic*(41/100);
 hra:=basic*(15/100);
 if(basic<3000)
 then
 pf:=basic*(5/100);
 elsif(basic>=3000 and basic<=5000)
 then
 pf:=basic*(7/100);
  elsif(basic>=5000 and basic<=8000)
 then
  pf:=basic*(8/100);
 else
  pf:=basic*(10/100);
end if;
netsalary:=basic+da+hra-pf;
dbms_output.put_line('employee name:'||ename);
dbms_output.put_line('providend fund:'||pf);
dbms_output.put_line('net salary:'||netsalary);
 end;
```

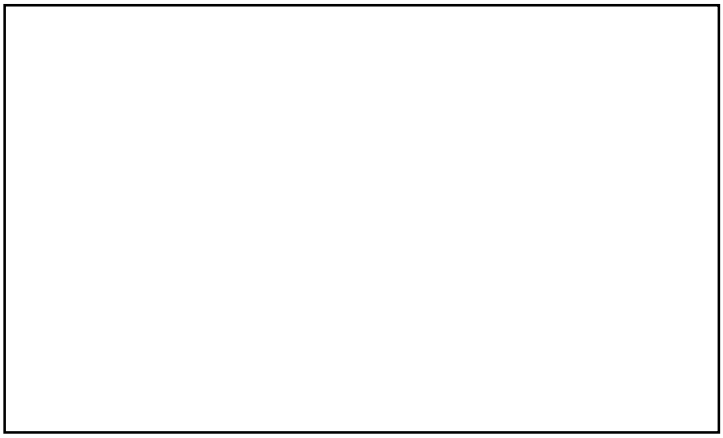


EXAMPLE FOR LOOP

```
SQL>  set serveroutput on
SQL>  declare
 begin
for i in 1..10
loop
dbms_output.put_line(to_char(i));
end loop;
end;
 /
```

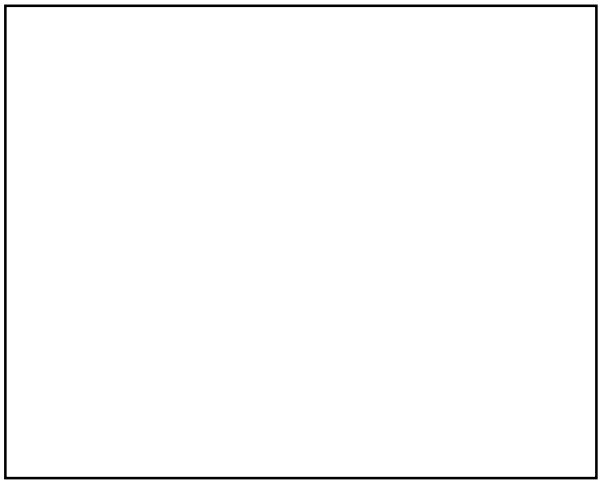
EXAMPLE FOR WHILE

```
SQL> set serveroutput on
SQL> declare
i number:=0;
   j number:=0;
   begin
   while i<=100 loop
    j:=j+1;
   i:=i+2;
   end loop;
  dbms_output.put_line(to_char(i));
  end;
  /
```



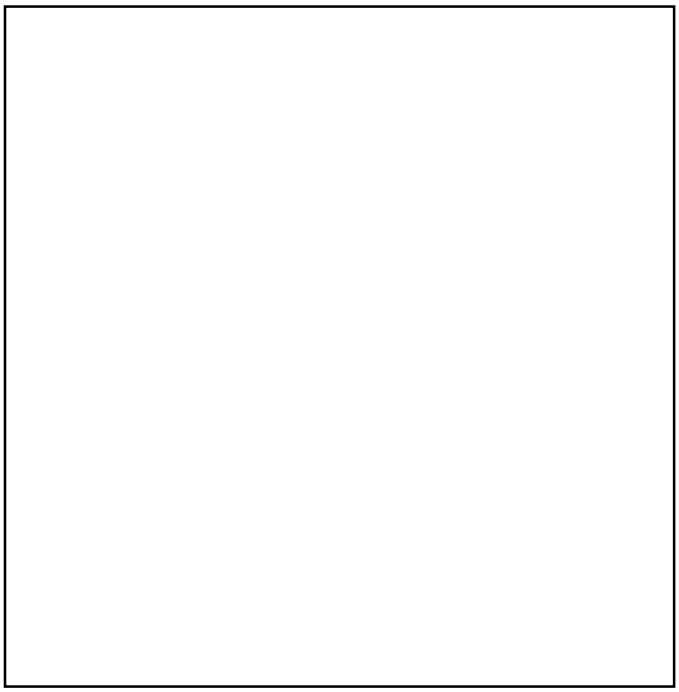
EXAMPLE FOR LOOP USING EXIT

```
SQL> set serveroutput on
SQL>  declare
 a number:=100;
  begin
  loop
  a:=a+25;
  exit when a=250;
  end loop;
  dbms_output.put_line(to_char(a));
  end;   /
```



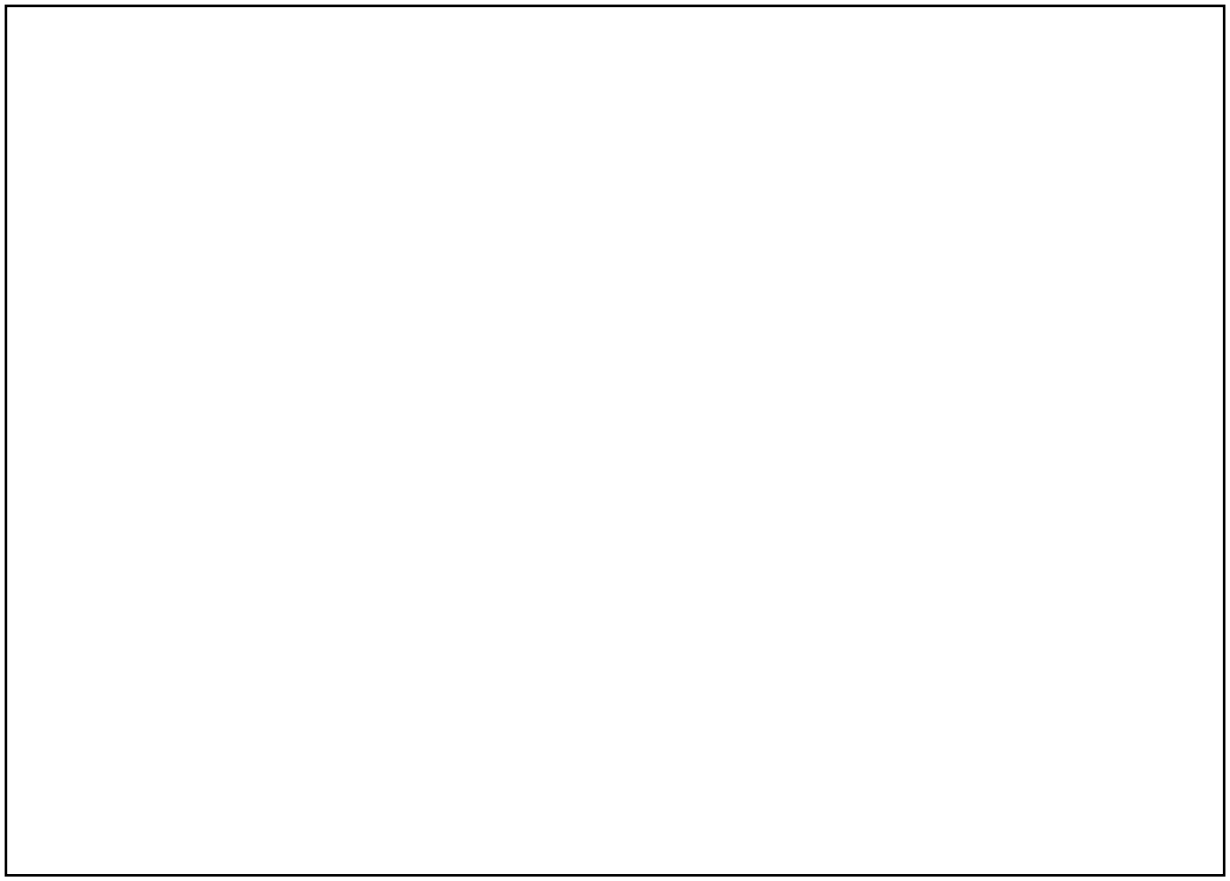
PRIME OR NOT

```
SQL> set serveroutput on
SQL>  declare
   no number(3):=&no;
   a number(4);
   b number(2);
   begin
   for i in 2..no-1
    loop
   a:=no MOD i;
   if a=0
   then
  GOTO out;
  end if;
end loop;
<<out>>
   if a=1
   then
  dbms_output.put_line(no||'is a prime');
 else
dbms_output.put_line(no||'is not a prime');
   end if;
  end;  /
```



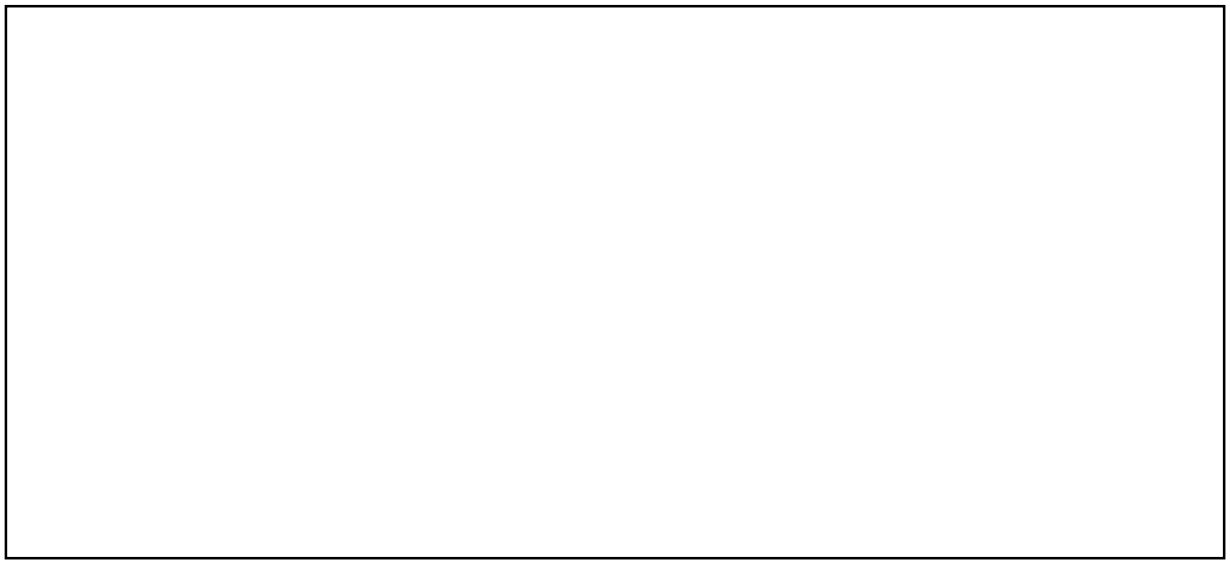
AREA CALCULATION

```
SQL> set serveroutput on
SQL> declare
pi constant number(4,2):=3.14;
radius number(5);
area number(14,2);
begin
radius:=3;
while radius<=7
loop
area:=pi*power(radius,2);
 insert into areas values(radius,area);
 radius:=radius+1;
 end loop;
 end;
 /
```



Implicit cursors:

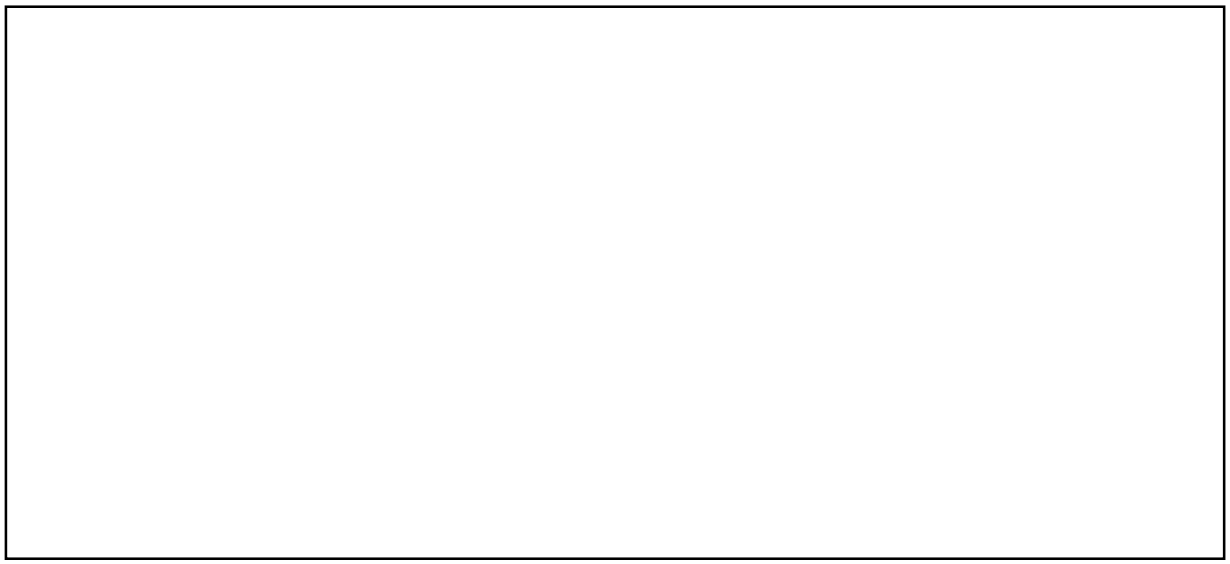
```
SQL> DECLARE
ena EMP.ENAME%TYPE;
esa EMP.SAL%TYPE;
BEGIN
SELECT ENAME,SAL INTO ENA,ESA FROM EMP
WHERE EMPNO = &EMPNO;
DBMS_OUTPUT.PUT_LINE('NAME :' || ENA);
DBMS_OUTPUT.PUT_LINE('SALARY :' || ESA);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Employee no does not exits');
END;
```



.

Explicit Cursors:

```
SQL> DECLARE
ena EMP.ENAME%TYPE;
esa EMP.SAL%TYPE;
CURSOR c1 IS SELECT ename,sal FROM EMP;
 BEGIN
OPEN c1;
FETCH c1 INTO ena,esa;
DBMS_OUTPUT.PUT_LINE(ena || ' salry is $ ' || esa);
FETCH c1 INTO ena,esa;
DBMS_OUTPUT.PUT_LINE(ena || ' salry is $ ' || esa);
FETCH c1 INTO ena,esa;
DBMS_OUTPUT.PUT_LINE(ena || ' salry is $ ' || esa);
CLOSE c1;
END;
/
```



Viva Questions

1. What are the advantages and disadvantages of DBMS?
2. Difference between file system and database system.
3. Define DBA
4. Name different types of Attributes
5. Define tuples
6. Define Functional dependency
7. Define Normalization

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result

Thus, the simple control structure and cursor programs are implemented successfully using PL/SQL.

Ex. No: 08 Date:	PL/SQL block that handles all types of exceptions

Aim:

To implement the PL/SQL block that handles all types of exceptions.

Exceptions

An exception is an error which disrupts the normal flow of program instructions. PL/SQL provides us the exception block which raises the exception thus helping the programmer to find out the fault and resolve it.

There are two types of exceptions defined in PL/SQL

1. User defined exception.
2. System defined exceptions.

Various System Defined Exceptions

Exception Name	Oracle Server Error	Description
DUP_VAL_ON_INDEX	ORA-00001	Attempted to insert a duplicate value
INVALID_NUMBER	ORA-01722	Conversion of character string to number fails
NO_DATA_FOUND	ORA-01403	Single row SELECT returned no data
TIMEOUT_ON_RESOURCE	ORA-00051	Time-out occurred while Oracle is waiting for a resource
TOO_MANY_ROWS	ORA-01422	Single-row SELECT returned more than one row
VALUE_ERROR	ORA-06502	Arithmetic, conversion, truncation, or size-constraint error returned
ZERO_DIVIDE	ORA-01476	Attempted to divide by zero

Syntax

```
WHEN exception THEN
   statement;
DECLARE
declaration Section
BEGIN
executable command(s);
EXCEPTION
WHEN exception1 THEN
statement1;
WHEN exception2 THEN
```

statement2;

[WHEN others THEN]

/* default exception handling code */

END;

SYSTEM DEFINED EXCEPTIONS

These exceptions are predefined in PL/SQL which get raised WHEN certain database rule is violated.

System-defined exceptions are further divided into two categories:

1. Named system exceptions.
2. Unnamed system exceptions.

Named system exceptions: They have a predefined name by the system like ACCESS_INTO_NULL, DUP_VAL_ON_INDEX, LOGIN_DENIED etc.

Create a table (Student) with following details

S_id, S_Name,Mark

1. NO_DATA_FOUND: It is raised WHEN a SELECT INTO statement returns *no* rows

Program

```
DECLARE
  temp varchar(20);

BEGIN
  SELECT s_id into temp from student where s_name='akash';

exception
  WHEN no_data_found THEN
    dbms_output.put_line('ERROR');
    dbms_output.put_line('there is no name as');
    dbms_output.put_line(' akash in geeks table');
end;
```

Output

2. ZERO_DIVIDE = raises exception WHEN dividing with zero

Program

```
DECLARE
  a int:=10;
  b int:=0;
  answer int;

BEGIN
  answer:=a/b;
  dbms_output.put_line('the result after division is'||answer);

exception
  WHEN zero_divide THEN
    dbms_output.put_line('dividing by zero please check the values again');
    dbms_output.put_line('the value of a is '||a);
    dbms_output.put_line('the value of b is '||b);
END;
```

Output

User defined exceptions:

This type of users can create their own exceptions according to the need and to raise these exceptions explicitly raise command is used.

Description

Divide non-negative integer x by y such that the result is greater than or equal to 1.
From the given question we can conclude that there exist two exceptions

- Division be zero.
- If result is greater than or equal to 1 means y is less than or equal to x.

Program

```
DECLARE
  x int:=&x; /*taking value at run time*/
  y int:=&y;
  div_r float;
  exp1 EXCEPTION;
  exp2 EXCEPTION;

BEGIN
  IF y=0 then
     raise exp1;

  ELSIF y > x then
    raise exp2;

  ELSE
    div_r:= x / y;
    dbms_output.put_line('the result is '||div_r);

  END IF;

EXCEPTION
  WHEN exp1 THEN
    dbms_output.put_line('Error');
    dbms_output.put_line('division by zero not allowed');

  WHEN exp2 THEN
    dbms_output.put_line('Error');
    dbms_output.put_line('y is greater than x please check the input');

END;
```

Output

Raise_Application_Error

It is used to display user-defined error messages with error number whose range is in between -20000 and -20999. When RAISE_APPLICATION_ERROR executes it returns error message and error code which looks same as Oracle built-in error.

Program

```
DECLARE
   myex EXCEPTION;
   n NUMBER :=10;

BEGIN
   FOR i IN 1..n LOOP
   dbms_output.put_line(i*i);
      IF i*i=36 THEN
      RAISE myex;
      END IF;
   END LOOP;

EXCEPTION
   WHEN myex THEN
      RAISE_APPLICATION_ERROR(-20015, 'Welcome to DBMS LAB');

END;
```

Output

1.

Viva Question

2. What are Errors?
3. What are Exceptions?
4. What is the difference between Errors and Exceptions?
5. How do you differentiate between errors in SQL and PL/SQL?
6. What is basic syntax of Oracle exception handling block?
7. What are the types of exceptions in Oracle PL/SQL?
8. List some Pre-defined Oracle exceptions?

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result

Thus, the implementation of PL/SQL block Programs using exception handling is implemented.

Ex. No: 09 Date:	**Creation of Procedures**

Aim

To create a Procedures using SQL

Procedures in PL/SQL

A subprogram is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the calling program.

A subprogram can be created

- At the schema level
- Inside a package
- Inside a PL/SQL block

PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms −

- Functions − These subprograms return a single value; mainly used to compute and return a value.
- Procedures − These subprograms do not return a value directly; mainly used to perform an action.

Syntax

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
  < procedure_body >
END procedure_name;
```

Executing a Standalone Procedure

A standalone procedure can be called in two ways −

- Using the EXECUTE keyword
  - **Syntax :** EXECUTE Procedure_Name or exec Procedure_Name
- Calling the name of the procedure from a PL/SQL block
  - **Syntax:**

```
BEGIN
  greetings;
END;
/
```

Deleting a Standalone Procedure

Syntax

```
DROP PROCEDURE procedure-name;
```

Simple Program

```
CREATE OR REPLACE PROCEDURE dbmslab
AS
BEGIN
  dbms_output.put_line('WELCOME TO DBMS LAB');
END;
/
```

Output

Program 2

create table stud(rno number(2),mark1 number(3),mark2 number(3),total number(3),primary key(rno));

create or replace procedure studd(rnum number) is

m1 number;

m2 number;

total number;

begin

select mark1,mark2 into m1,m2 from stud where rno=rnum;

if m1<m2 then

update stud set total=m1+m2 where rno=rnum;

end if;

end;

/

Output

Viva Question

1. What are the different schema objects that can be created in PL/SQL?
2. What are subprograms in PL/SQL and what are the advantages of subprograms?
3. What is a Stored Procedure in PL/SQL?
4. What is the syntax to create a Stored Procedure in PL/SQL?

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result

Thus, the report is generated using the PL/SQL procedure**.**

Ex. No: 10 Date:	**Creation of database triggers and functions**

Aim

To create a database triggers and functions using SQL

Trigger

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs.

Syntax

create trigger [trigger_name]

[before | after]

{insert | update | delete}

on [table_name]

[for each row]

[trigger_body]

Create a Student Report Database (S_Id, S_Name, Subject1, Subject 2, Subject 3, total, Percentage), in which student marks assessment is recorded. In such schema, create a trigger so that the total and percentage of specified marks is automatically inserted whenever a record is insert.

Table Creation

Execute Trigger

create trigger stud_marks

before INSERT

on

Student

for each row

set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total * 60 / 100;

Insert the values into the table

Function

The PL/SQL Function is very similar to PL/SQL Procedure. The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value. Except this, all the other things of PL/SQL procedure are true for PL/SQL function too.

Syntax

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
  < function_body >
END [function_name];
```

Create a function

```
create or replace function adder(n1 in number, n2 in number)
return number
is
n3 number(8);
begin
n3 :=n1+n2;
return n3;
end;
/
```

Call the function

```
DECLARE
  n3 number(2);
BEGIN
  n3 := adder(11,22);
  dbms_output.put_line('Addition is: ' || n3);
END;
/
```

Output:

Factorial of given number using function

```
DECLARE
  num number;
  factorial number;

FUNCTION fact(x number)
RETURN number
IS
  f number;
BEGIN
  IF x=0 THEN
    f := 1;
  ELSE
    f := x * fact(x-1);
  END IF;
RETURN f;
END;

BEGIN
  num:= 6;
  factorial := fact(num);
  dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
/
```

Output

Function using Table

Create the table with following Attributes

Customers			
Id	**Name**	**Department**	**Salary**
1	alex	web developer	35000
2	ricky	program developer	45000
3	mohan	web designer	35000
4	dilshad	database manager	44000

Function Creation

```
CREATE OR REPLACE FUNCTION totalcustomers
RETURN number IS
  total number(2) := 0;
BEGIN
  SELECT count(*) into total
  FROM customers;
   RETURN total;
END;
/
```

Output

Call the function

```
DECLARE
  c number(2);
BEGIN
  c := totalcustomers();
  dbms_output.put_line('Total no. of Customers: ' || c);
END;
/
```

Output

Viva Questions

1. What is a Trigger?
2. What is a triggering event?
3. Types of trigger based on triggering event?
4. What are some use cases of Triggers??
5. Difference between Triggers and Constraints?

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result: Thus, the PL/ SQL commands for triggers and functions are executed and verified successfully.

Ex. No: 11 Date:	Installing and Configuring - MongoDB

Aim:

To install and configure the mongodb shell in windows.

Introduction:

MongoDB is a powerful, highly scalable, free and open-source NoSQL based database. MongoDB was initially released approximately 9 years ago on the 11th of February, 2009 and has since then achieved the position of the leading NoSQL database. The company MongoDB Inc. (New York, United States) maintains and manages the development of MongoDB. They also provide the commercial version of MongoDB which includes support also. The source code of MongoDB is available on Github.

Over the years, MongoDB has become a popular choice of a highly scalable database and it is currently being used as the backend data store of many well-known organizations like IBM, Twitter, Zendesk, Forbes, Facebook, Google, and a gazillion others. MongoDB has also caught the eyes of the open-source community and a lot of developers work on various open-source projects based on MongoDB.

MongoDB is a document-based data store which means that it stores the information in rather an unstructured format as compared to structured tables like in MySQL or PostgreSQL. This essentially means that the data stored in MongoDB is “schema-less”. Therefore, MongoDB provides a fast and scalable data storage service which makes it a popular choice in the performance-critical application. Moreover, the fact that MongoDB has been written in C++ makes it even faster as compared to a lot of other databases.

MongoDB should not be used in applications that require table joins simply because it doesn’t support joins (like in SQL). This is attributed to the fact that the data stored in MongoDB is not structured and therefore, performing joins is a highly time-consuming process that may lead to slow performance.

Here is how a simple MongoDB based Student model will look like:

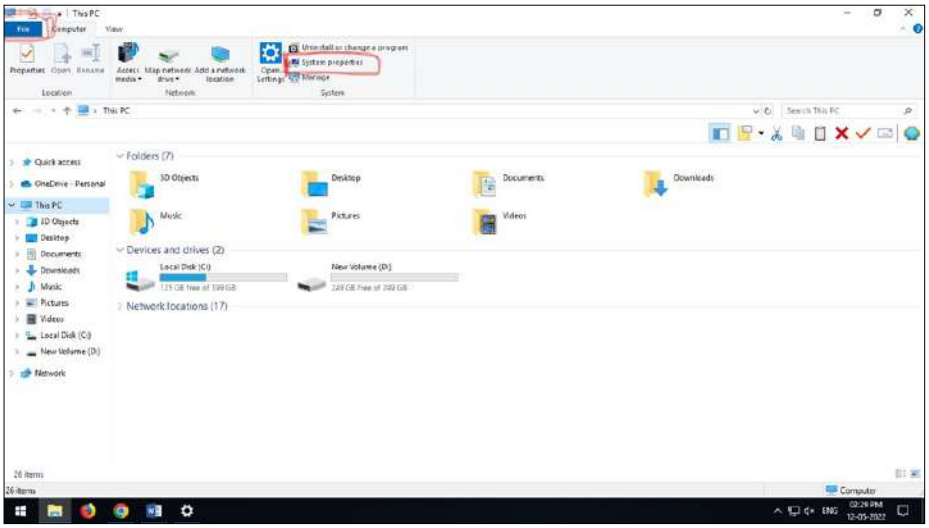
```
{
 _id: STUDENT_IDname: STUDENT_NAME,
 data_of_birth: STUDENT_DATE_OF_BIRTH,
 courses: [
  {
   code: COURSE_CODE,
   instructor: COURSE_INSTRUCTOR,
```

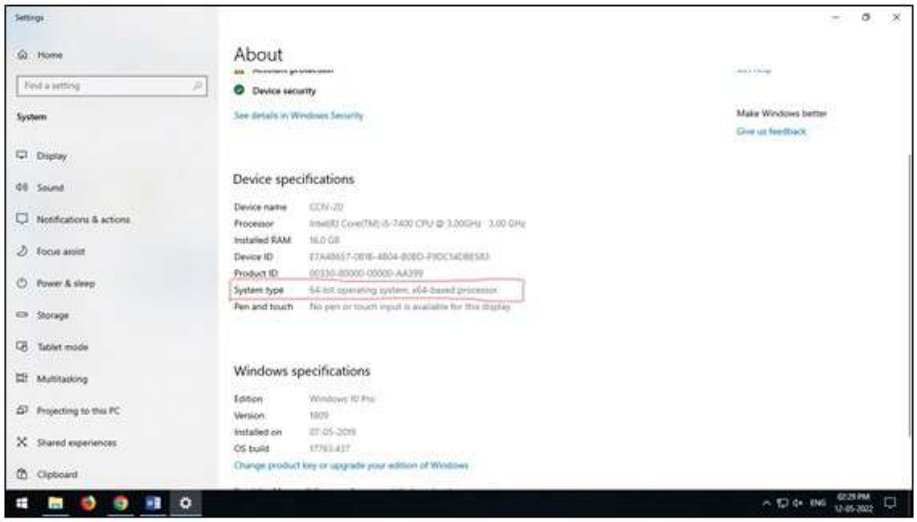
```
},
{
  code: COURSE_CODE,
  instructor: COURSE_INSTRUCTOR,
}        ]       }
```

PRE-REQUISITE INFORMATION ON WINDOWS:

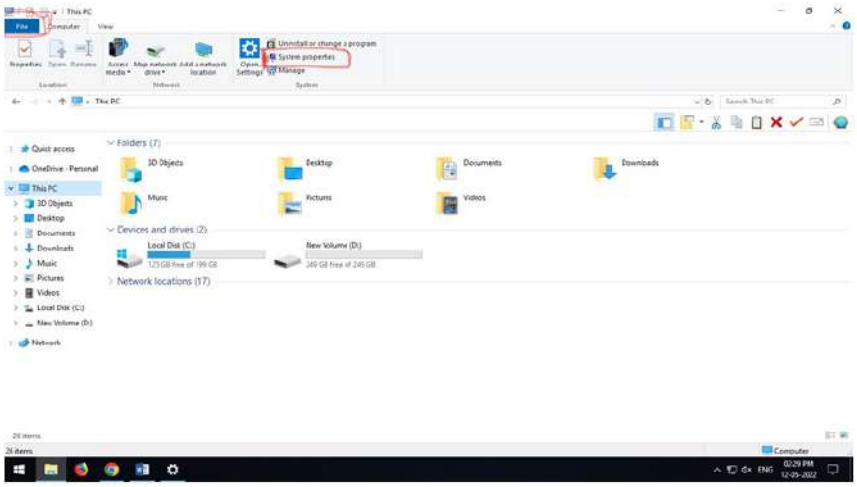
The users of Windows must know that their windows desktop has got one of the two versions i.e. 32-bit & 64-bit.

This information could be found out in the properties of one's "My Computer" or "This PC" on their device i.e. either their windows is 32-bit or 64-bit.

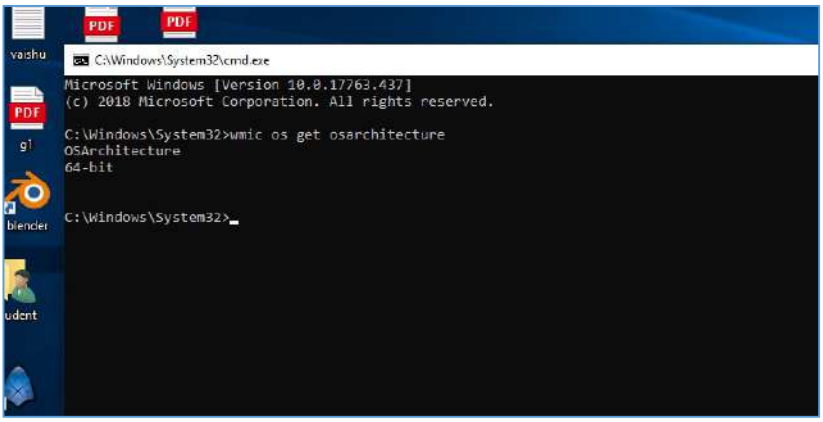




Meanwhile, in order to check the window version, one can also use command prompt in the way as narrated in the snippet below:



The command is **C:\>wmic os get osarchitecture**



Command to find the Operating System architecture

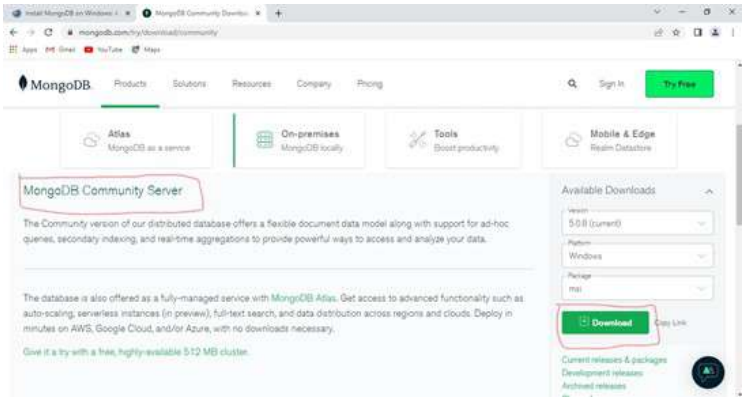
MongoDB is available in both the versions which support their respective 32-bit & 64-bit windows.

For instance, 32-bit windows users have got the advantage of having qualitative development and testing environments. Meanwhile, if one must get into production environments, then they must have to adopt 64-bit windows, because in this case, using 32-bit would limit the data usage which would be stored in MongoDB. Therefore, the 32 –bit windows version supports the MongoDB version which has the database size lesser than 2 GigaBytes.

MONGODB DOWNLOAD ON WINDOWS:

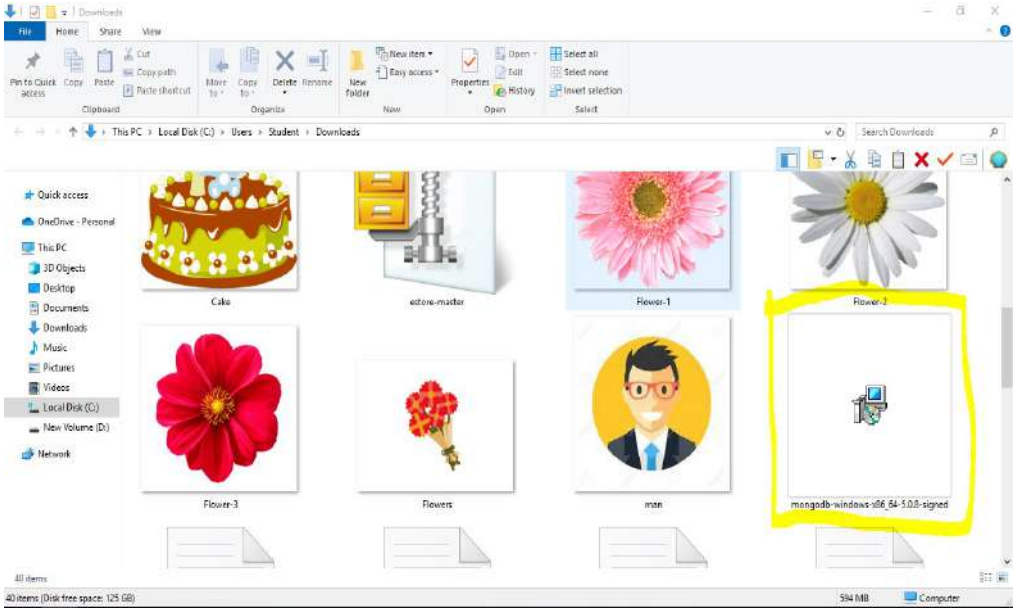
Click on the following link to download MongoDB on Windows

https://www.mongodb.com/try/download/community

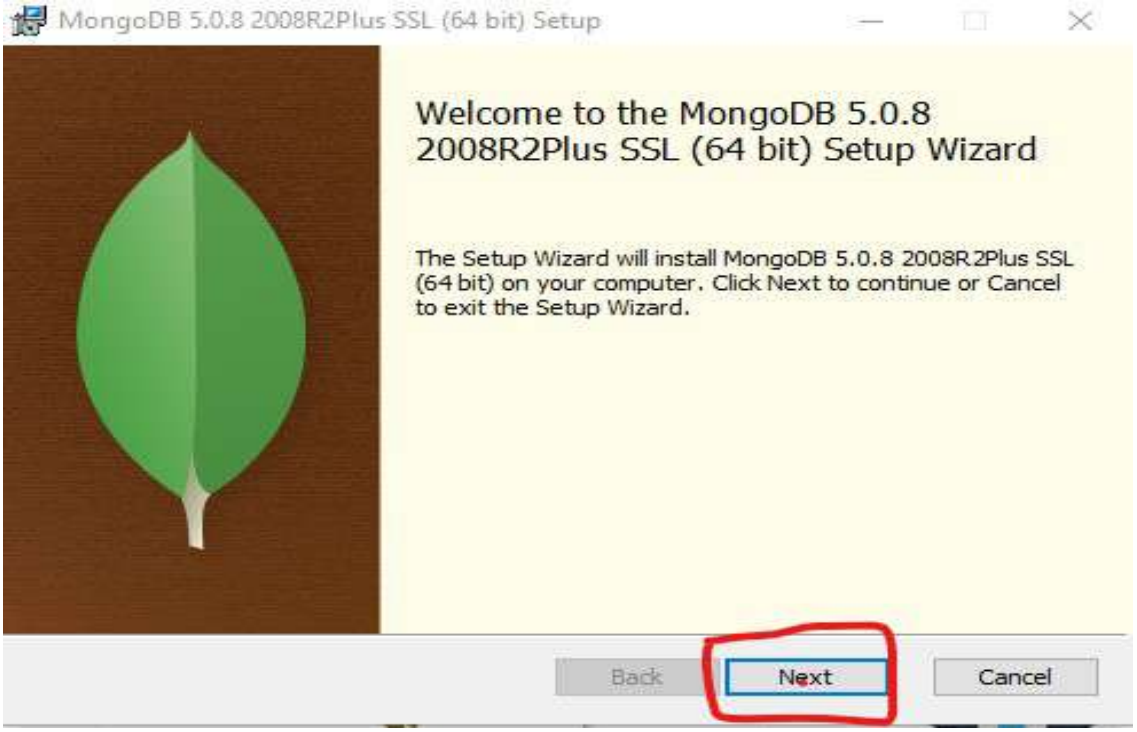


To get the Download File of MongoDB from website

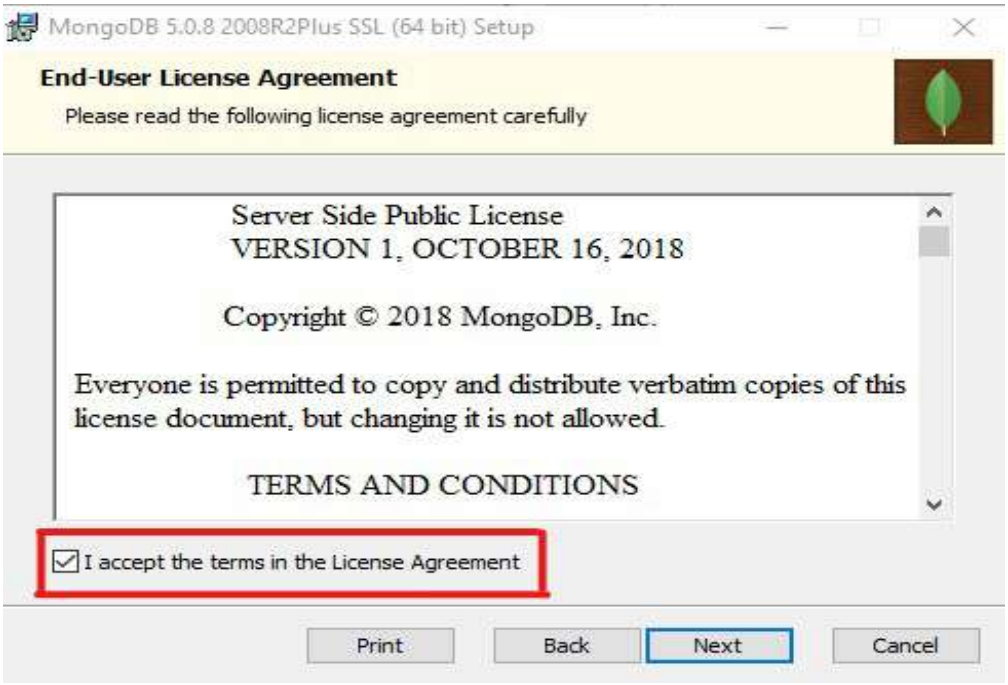
Step 1: Open the file. We have installed for 64-bit version with the name as "mongoDB-windows-x86_64-5.0.8-signed.msi". Click on the file where you've saved it to start the wizard.



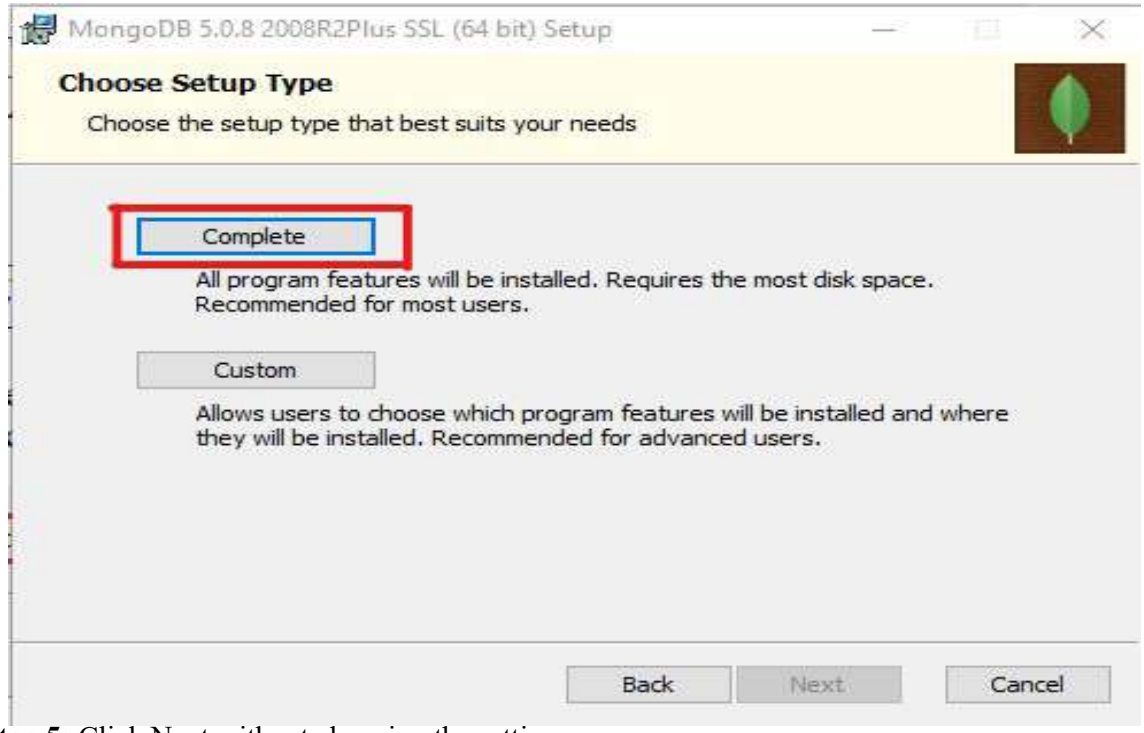
Step 2: Click “Next”.



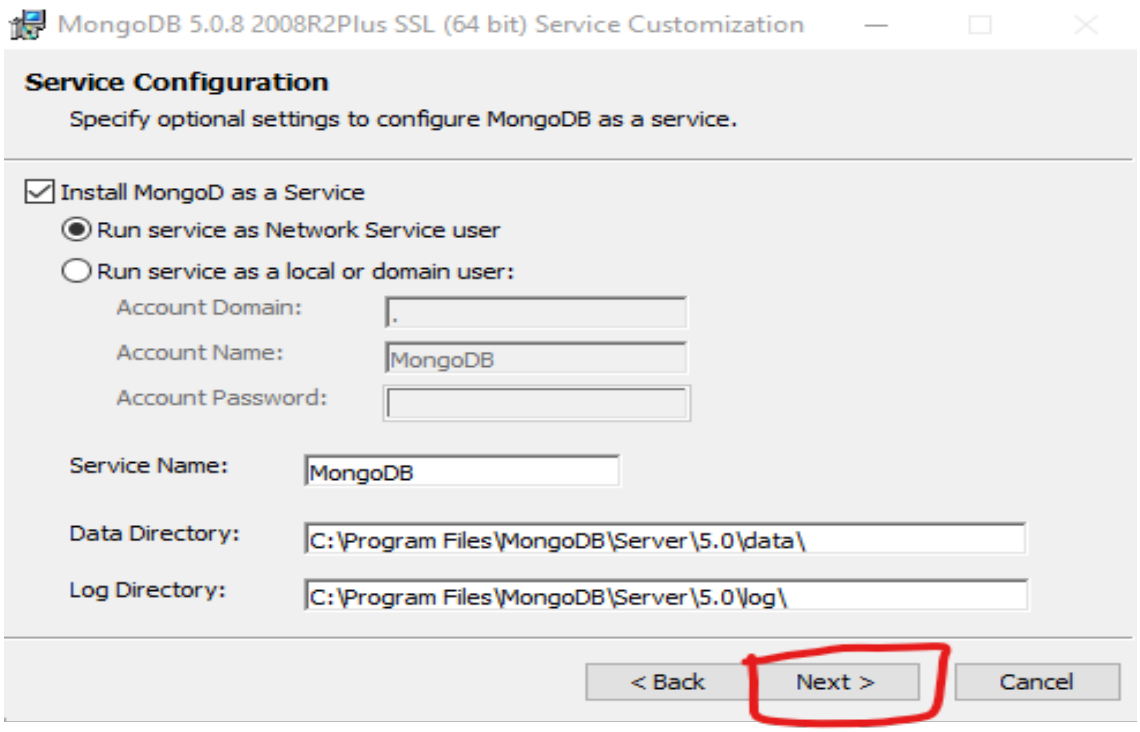
Step 3: Tick the check box next to ‘I accept the terms in the License Agreement’ and again click on ‘Next’.



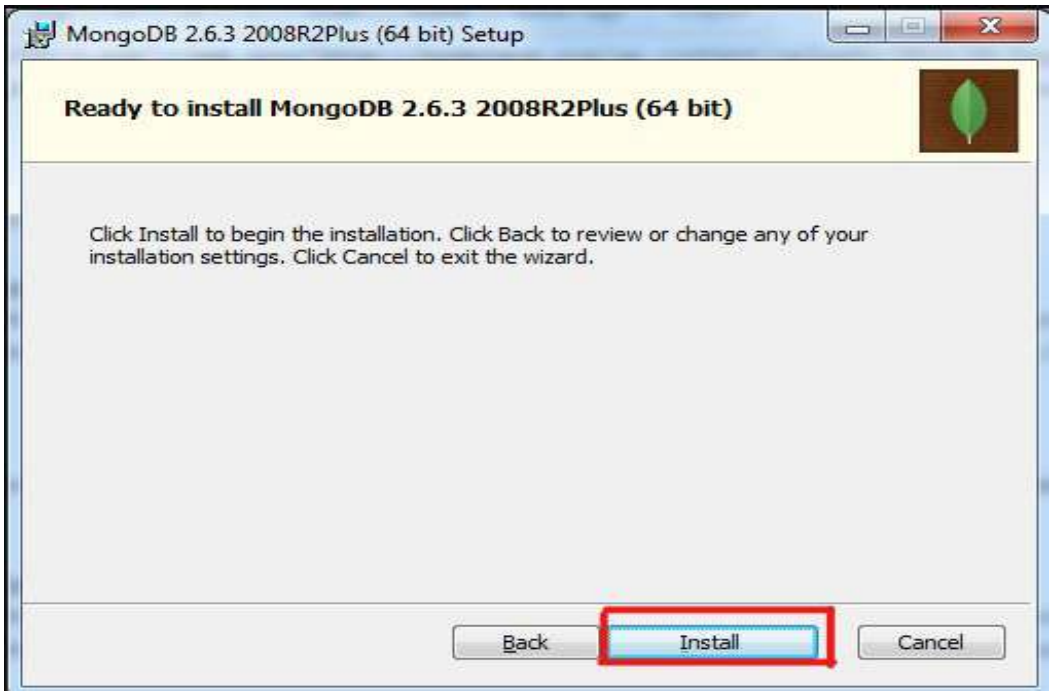
Step 4: Click "Complete" to install all the features of MongoDB. As for "Custom", this option would be used to install only the specific components of MongoDB and also if a user wants to change the location of where the installation must be done.

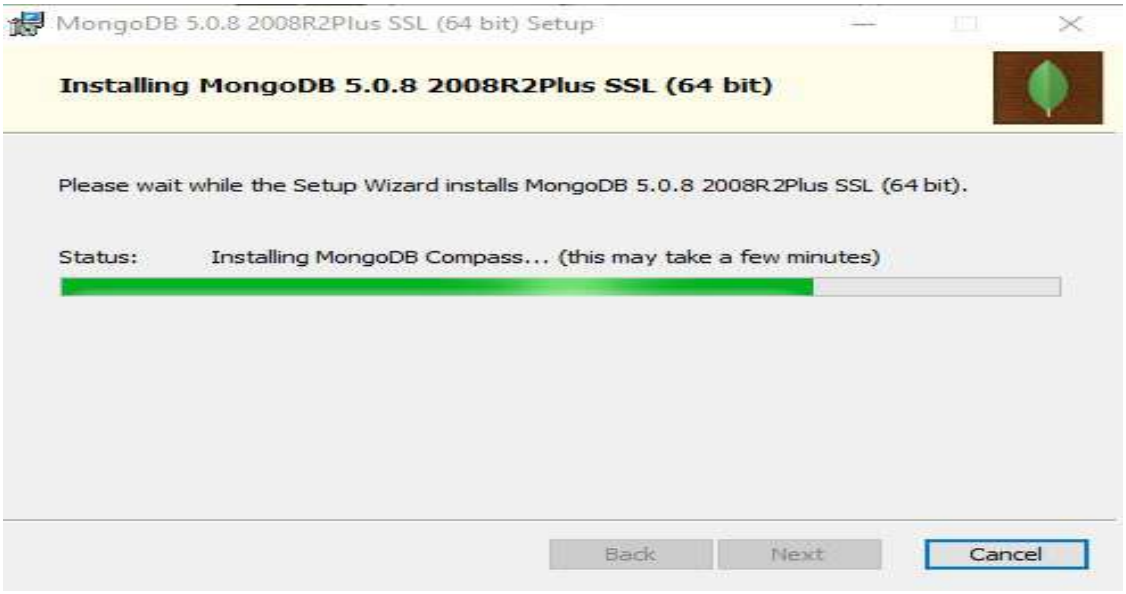


Step 5: Click Next without changing the settings.

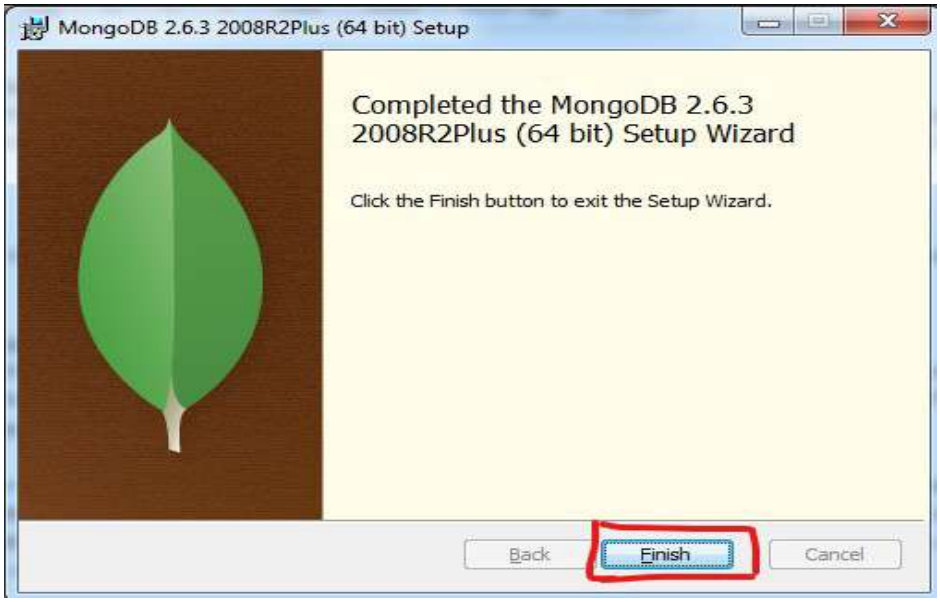


Step 6: Click “Install” to begin the installation drive.





Step 7: After the installation has been finished. Simply, click “Finish“.



CONFIGURATION THROUGH MONGO SHELL

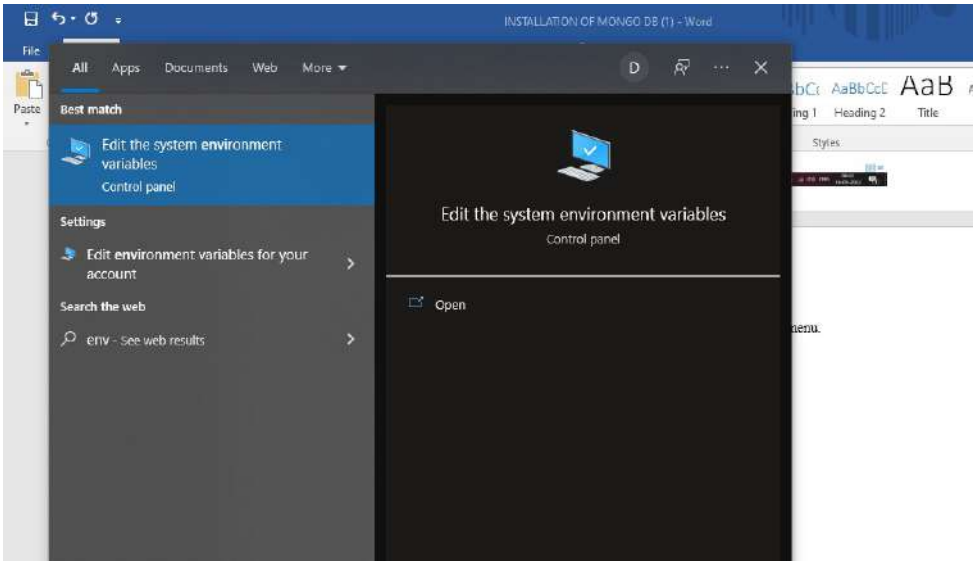
After the whole installation is done, the user must configure it.

Follow the below steps:

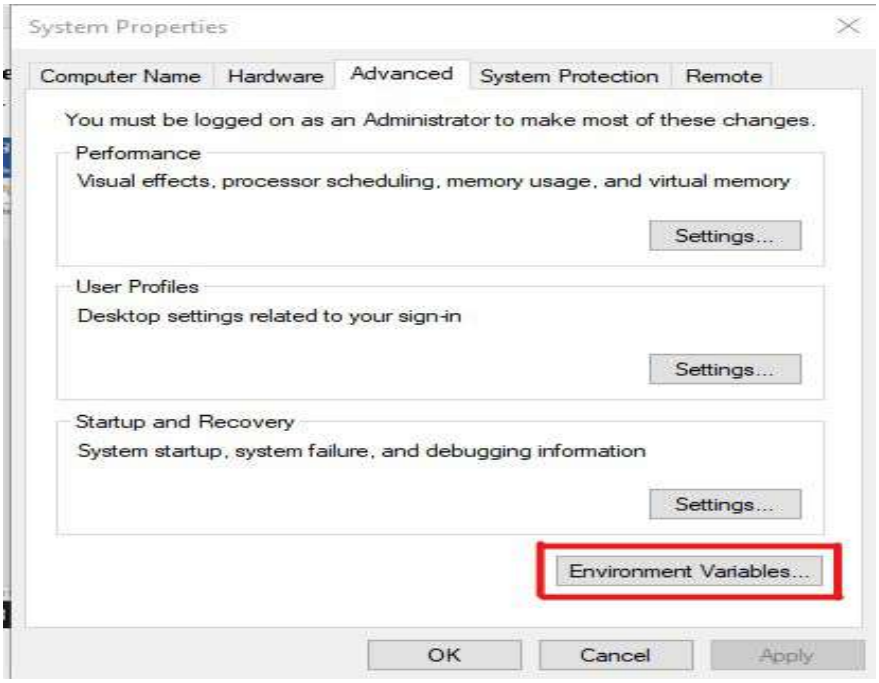
If you use “mongo” in the command prompt without configuring it, then it would give an error. Hence, configure it first.

Step 1: Go to the local disk C and get into “Program Files“. There you’ll find a folder named “MongoDB“. Inside that their will be a folder named server. Click that and you can find the version folder go inside that then there will be a folder named bin. Double click that folder. You can find several files related to mongodb their. Then copy the path.

Step 2: Paste the copied path in the environmental variable setup. Type env in start menu. Then click on the "**Edit the system environmental variable".**

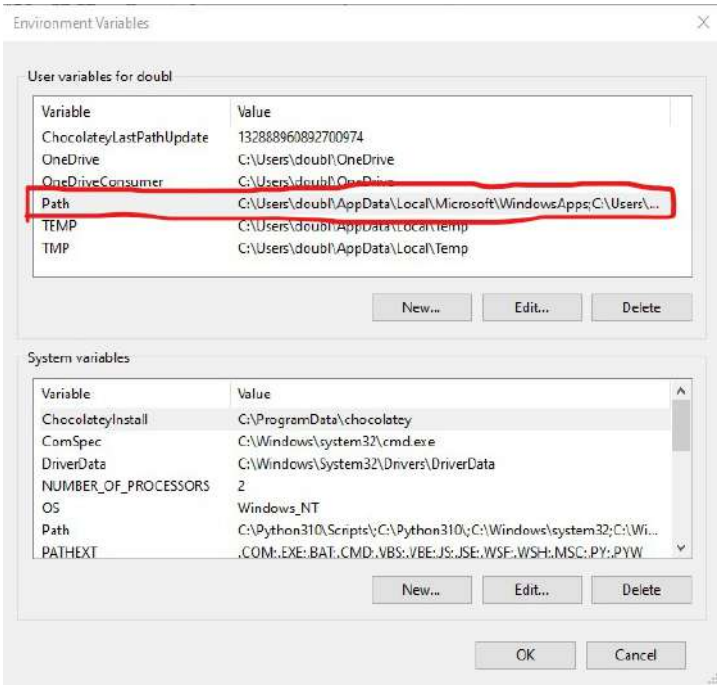


Step 3: Then the system properties dialog box appear click on environmental variable in that.

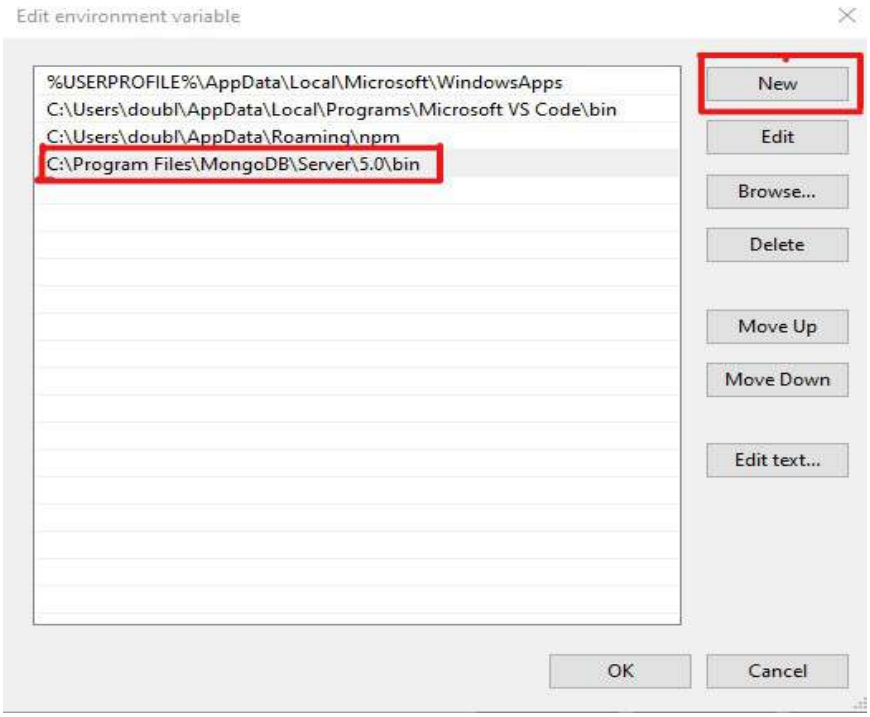


Step 4: Then the environmental variable dialog box appear. In that there are two different kinds of variable. One was **"User Variable, System Variable"**.

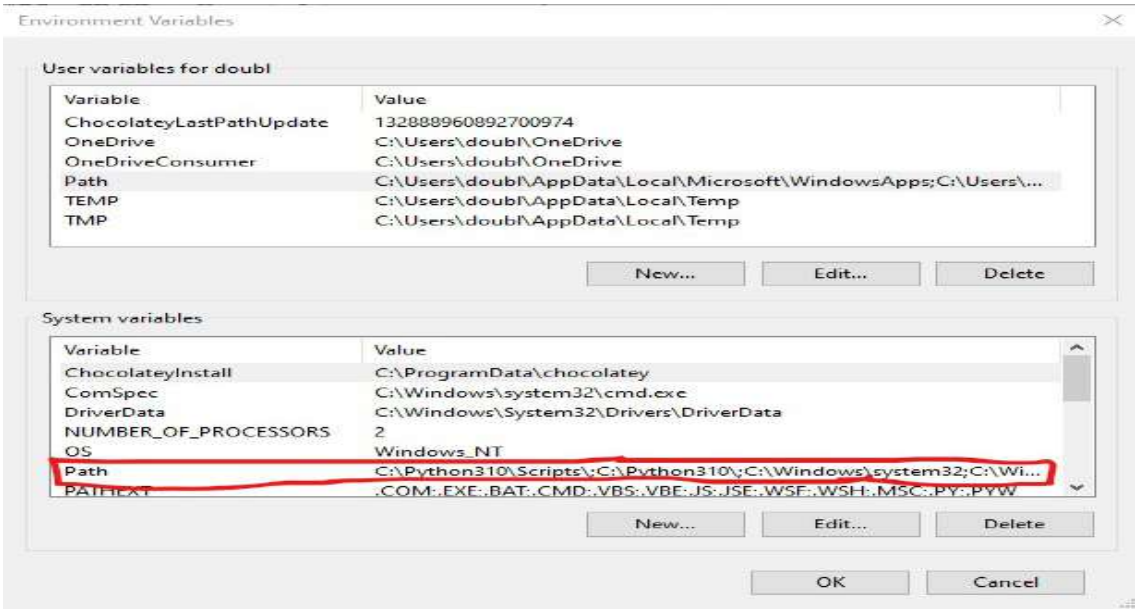
In user variable click on the path and click edit



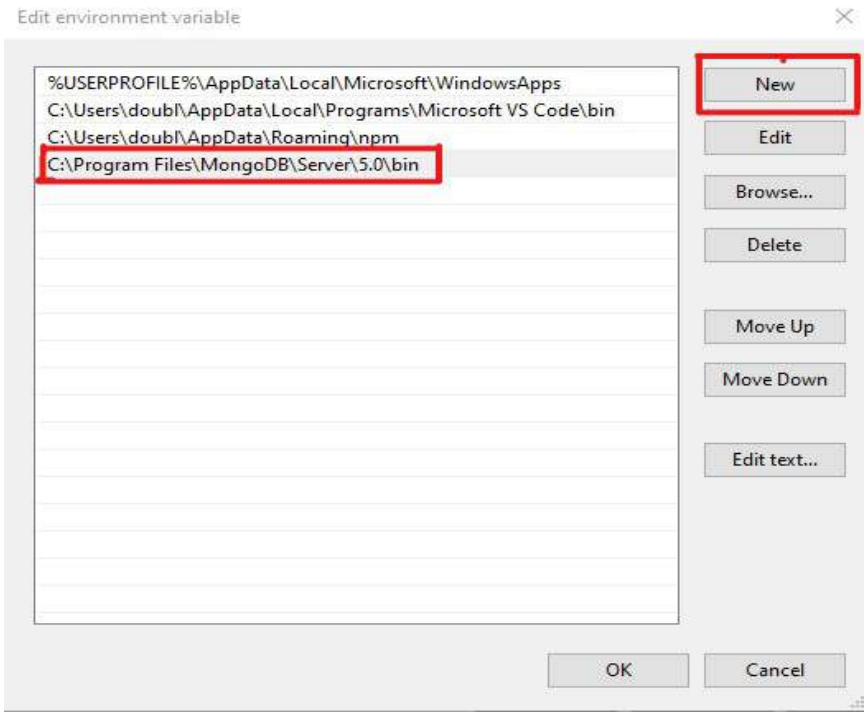
Step 5: Click new in that paste the copied path in the variable and Click Ok



Step 6: Then click on the ‘system variable’. And click on path. Click on edit.



Step 7: Click new in that, Paste the copied path in the variable and Click Ok

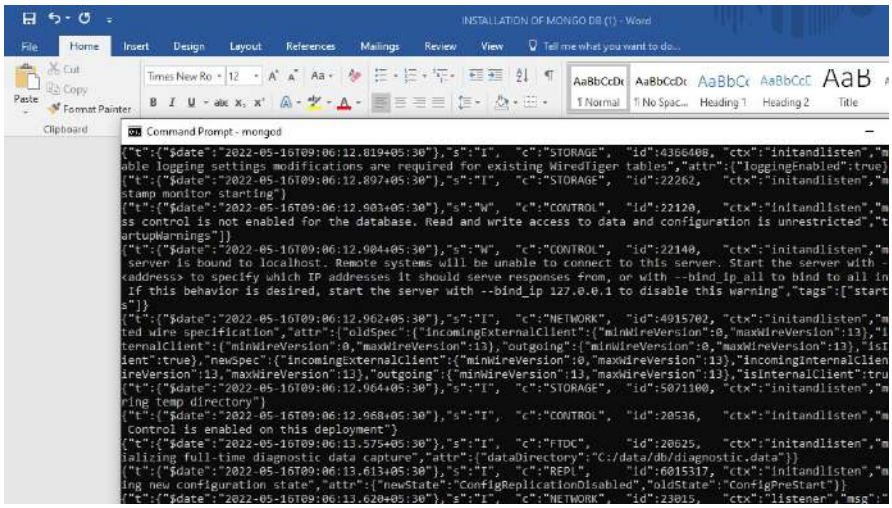


Step 8: Then click ok in all the window.

Step 9: Then create a folder in local disk:C, named “data”, in that folder create another folder called db

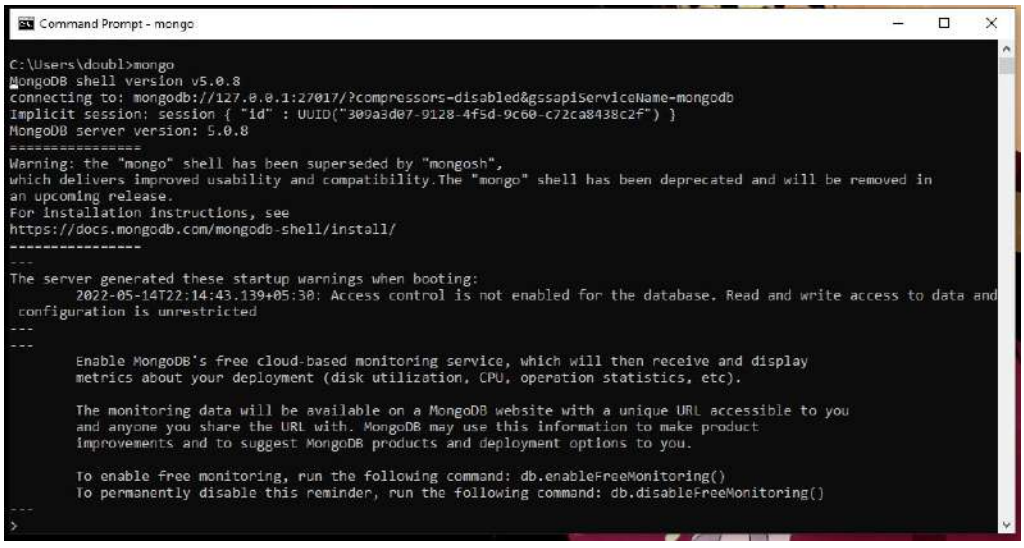
Step 10: Then open command prompt and type “mongod” to start the server.

C:\Users\doubl>mongod



Step 11: Open another command prompt and type ‘mongo’ to start the client server.

C:\Users\doubl>mongo



Viva Questions

1. What is MongoDB ?
2. What are some of the advantages of MongoDB?
3. What is a Collection in MongoDB?
4. What are Databases in MongoDB?
5. What is the Mongo Shell?

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result:

Thus, the installation and configuration of MongoDB has been installed and configured.

Ex. No: 12 Date:	Creation, Insertion, Updation, Retrieval and Deletion operations on Mongo DB

Aim:

To create the database and perform the manipulation operations using Mongo DB

Operation

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result:

Thus, the database has been created and manipulation operation using Mongodb

Ex. No: 13 Date:	Accessing Databases from Programs using JDBC

Aim:

To Accessing the databases from program using Java Database Connectivity (JDBC)

Accessing a Database

The process of retrieving information from a database via JDBC involves these five basic steps:

1. Register the JDBC driver with the driver manager.
2. Establish a database connection.
3. Execute an SQL statement.
4. Process the results.
5. Close the database connection.

Register the JDBC Driver

JDBC driver can be used to establish a database connection, it must first be registered with the driver manager. The driver manager's job is to maintain a reference to all driver objects that are available to JDBC clients. A JDBC driver automatically registers itself with the driver manager when it is loaded. To load a JDBC driver, use the Class.forName().newInstance() method call as demonstrated here:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
```

`Class.forName()` is a static method that instructs the Java virtual machine to dynamically locate, load, and link the specified class (if not already loaded).

If the class cannot be located, a `ClassNotFoundException` is thrown.

The `newInstance()` method indicates that a new instance of this class should be created.

Establish a Database Connection

Once the driver is loaded, we can use it to establish a connection to the database. A JDBC connection is identified by a database URL that tells the driver manager which driver and data source to use. The standard syntax for a database URL is shown here:

jdbc: SUBPROTOCOL: SUBNAME

The first part of the URL indicates that JDBC is being used to establish the connection. The SUBPROTOCOL is the name of a valid JDBC driver or other database connectivity solution. The SUBNAME is typically a logical name, or alias, that maps to a physical database.

JDBC URLs for Common Data Sources

Database	JDBC Database URL
ODBC Data Source	dbc:odbc:DATA_SOURCE_NAME
MySQL	dbc:mysql://SERVER[:PORT]/DATABASE_NAME
Oracle	dbc:oracle:thin:@SERVER:PORT:INSTANCE_NAME

To establish a database connection, use the `DriverManager` object's `getConnection ()` method.

`getConnection ()` is a static method that can be used like this:

```
Connection dbConn = DriverManager.getConnection(
      "jdbc:mysql://localhost/phonebook");
```

Or, for databases that require authentication, the connection can be established like this:

```
String username = "Erin", password = "secret";
Connection dbConn = DriverManager.getConnection(
       "jdbc:mysql://localhost/phonebook", username, password);
```

Within its getConnection () method, the DriverManager queries each registered driver until it finds one that recognizes the specified database URL. Once the correct driver is located, the DriverManager uses it to create the Connection object. The DriverManager and Connection objects are contained in the java.sql package. Be sure to import this package when using JDBC.

Execute an SQL Statement

Once established, the database connection can be used to submit SQL statements to the database. An SQL statement performs some operation on the database such as retrieving, inserting, updating, or deleting rows. To execute an SQL command, a Statement object must be created using the Connection object's **createstatement () method.** The Statement object provides methods to perform various operations against the database.

```
Statement stmt = dbConn.createStatement();
```

Using the Statement object's executeQuery () method, information can be retrieved from the database. The executeQuery () method accepts an SQL SELECT statement and returns a ResultSet object containing the database rows extracted by the query. For inserts, updates, or deletes, use the executeUpdate () method. The ResultSet object can be created like this:

```
ResultSet rs = stmt.executeQuery("select * from employee");
```

Process the Results

To process the results, Can traverse the rows of the result set using the ResultSet object's next() and previous() methods (the previous() method is only available in JDBC 2.0 and later using certain types of result sets). The following sample code creates a Statement object, executes a query, and iterates through the result set. The ResultSet object's getString() method is used to extract the value of specific fields.

Statement stmt = dbConn.createStatement();

```
ResultSet rs = stmt.executeQuery("select * from employee");

while (rs.next())
{
       System.out.println(rs.getString("FIRST_NAME") +" " +
            rs.getString("LAST_NAME"));
}
```

It is important to keep in mind that the ResultSet object is tied to the Statement object that created it. If the ResultSet object's Statement is closed or used to execute another query, the ResultSet is closed automatically.

Close the Database Connection

Because database connections are a valuable and limited resource, you should close the connection when processing is complete. The Connection object provides a simple close() method for this purpose.

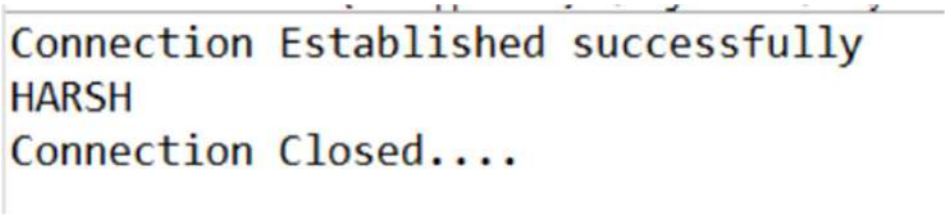
```
try
{
//...create statements and result sets here...

       rs.close(); //close ResultSet after use
stmt.close(); //close Statement after use
}
catch (SQLException e)
{
       //...handle exception here...
}
finally
{
try
{
if (dbConn "` null)
{
dbConn.close(); //close Connection in finally block
}
       }
catch (SQLException e)
{
 //...handle exception here...
}
}
```

Program

```
import java.sql.*;
class OracleCon{
public static void main(String args[]){
```

```
try{
//step1 load the driver class
Class.forName("oracle.jdbc.driver.OracleDriver");

//step2 create  the connection object
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

//step3 create the statement object
Statement stmt=con.createStatement();

//step4 execute query
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

//step5 close the connection object
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

Output:

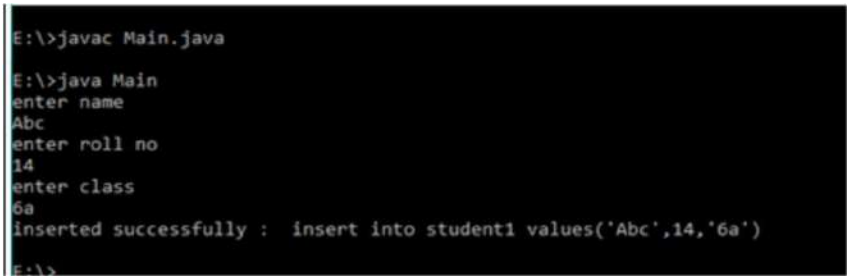


Program 2

```
import java.sql.*;
// Importing required classes
import java.util.*;

// Main class
class Main {

    // Main driver method
    public static void main(String a[])
    {

        // Creating the connection using Oracle DB
        // Note: url syntax is standard, so do grasp
        String url = "jdbc:oracle:thin:@localhost:1521:xe";

        // Username and password to access DB
        // Custom initialization
        String user = "system";
        String pass = "12345";

        // Entering the data
        Scanner k = new Scanner(System.in);

        System.out.println("enter name");
        String name = k.next();

        System.out.println("enter roll no");
        int roll = k.nextInt();

        System.out.println("enter class");
        String cls = k.next();

        // Inserting data using SQL query
        String sql = "insert into student1 values('" + name
                + "'," + roll + ",'" + cls + "')";

        // Connection class object
        Connection con = null;

        // Try block to check for exceptions
        try {

            // Registering drivers
            DriverManager.registerDriver(
                new oracle.jdbc.OracleDriver());

            // Reference to connection interface
```

```
            con = DriverManager.getConnection(url, user,
                                              pass);

            // Creating a statement
            Statement st = con.createStatement();

            // Executing query
            int m = st.executeUpdate(sql);
            if (m == 1)
               System.out.println(
                  "inserted successfully : " + sql);
            else
               System.out.println("insertion failed");

            // Closing the connections
            con.close();
         }

         // Catch block to handle exceptions
         catch (Exception ex) {
            // Display message when exceptions occurs
            System.err.println(ex);
         }
      }
}
```

Output



Department of AIML		
Average Performance	**25**	
Average Record	**15**	
Average Viva	**10**	
Total	**50**	

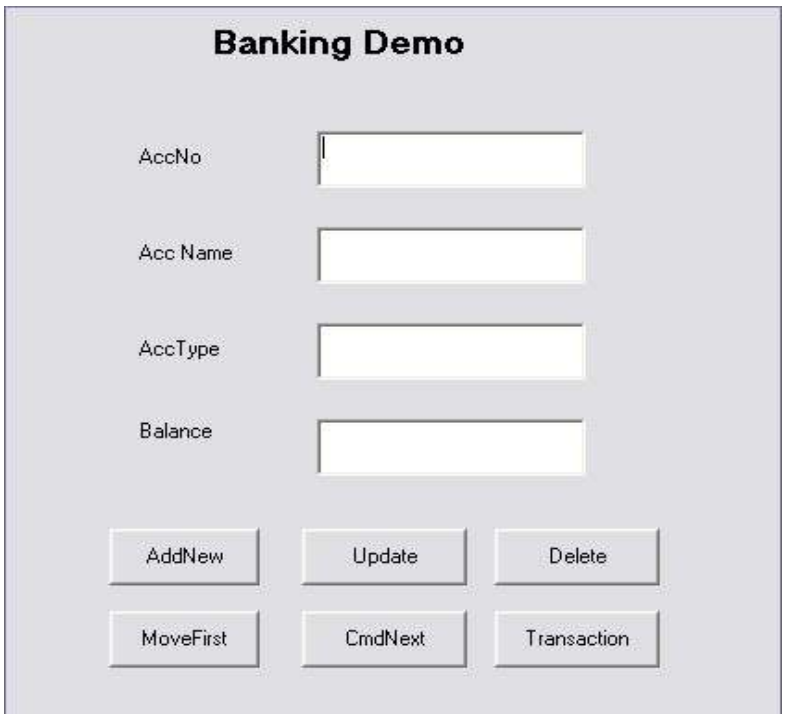
Result:

Thus, the databases have been accesses from program using Java Database Connectivity

Ex. No: 14 Date:	Mini Project

Banking Management System

Design Window



Adding Library

Project = > References => Microsoft ActiveX Data Objects 2.0 Library

Coding

```
Dim Con As New ADODB.Connection
Dim Rs As New ADODB.Recordset

Private Sub CmdAddNew_Click()

Rs.AddNew
txtClear
Text1.SetFocus

End Sub

Private Sub CmdDelete_Click()
```

```
On Error Resume Next

If Rs.EOF Then
MsgBox "No Records"
Else
Rs.Delete
Rs.MoveNext
DisplayText
MsgBox "Record Deleted"

End If

End Sub

Private Sub CmdFirst_Click()
Rs.MoveFirst
DisplayText
End Sub

Private Sub CmdNext_Click()
Rs.MoveNext
If Not Rs.EOF Then
DisplayText
Else
MsgBox "End Of The Record"
Rs.MovePrevious
End If
End Sub

Private Sub CmdUpdate_Click()

Rs(0).Value = Val(Text1.Text)
Rs(1).Value = Text2.Text
Rs(2).Value = Text3.Text
Rs(3).Value = Val(Text4.Text)

Rs.Update

MsgBox "Record Updated"

End Sub

Private Sub Form_Load()

Con.Open "Provider=MSDAORA.1;User ID=cse101;Password=cse101;Data
Source=ibmrec;Persist Security Info=False"
Con.CursorLocation = adUseClient

Rs.Open "select * from Bank", Con, adOpenKeyset, adLockOptimistic
```

End Sub

Sub txtClear()

```
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
```

End Sub

Sub DisplayText()

```
Text1.Text = Rs(0).Value
Text2.Text = Rs(1).Value
Text3.Text = Rs(2).Value
Text4.Text = Rs(3).Value
```

End Sub

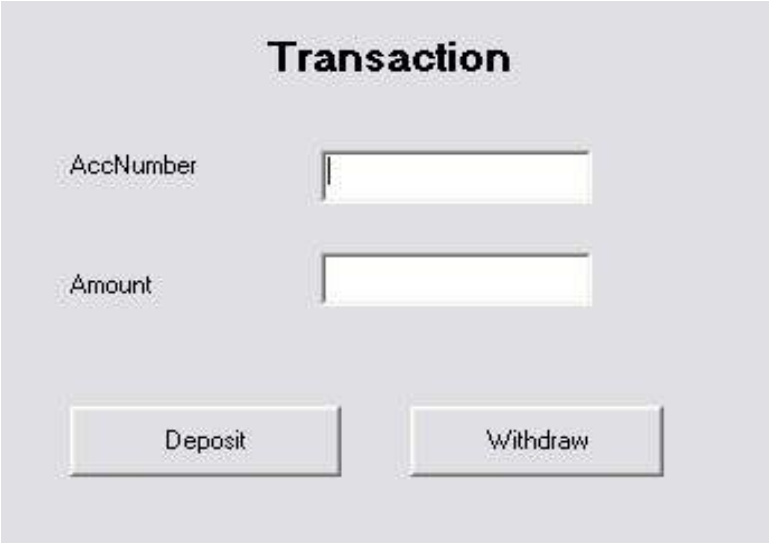
Private Sub CmdTransact_Click()

```
Form2.Show
```

End Sub

Transaction Window

Design



Coding

```
Dim Con As New ADODB.Connection
Dim Rs As New ADODB.Recordset
```

```
Private Sub CmdDeposit_Click()

Dim AccNo As Integer
AccNo = Val(Text1.Text)
Rs.MoveFirst

Do While Not Rs.EOF

   If Rs(0).Value = AccNo Then
      Rs(3).Value = Rs(3).Value + Val(Text2.Text)
      Rs.Update
      MsgBox "Amount Deposited"
   End If

   Rs.MoveNext

Loop

End Sub

Private Sub CmdWithdraw_Click()

Dim AccNo As Integer, Bal As Double
Rs.MoveFirst

AccNo = Val(Text1.Text)

Do While Not Rs.EOF

   If Rs(0).Value = AccNo Then

      Bal = Rs(3).Value - Val(Text2.Text)

      If Bal >= 500 Then
         Rs(3).Value = Rs(3).Value - Val(Text2.Text)
         Rs.Update
         MsgBox "Amount Withdrew"
      Else
         MsgBox "Balance Problem"
      End If
   End If
      Rs.MoveNext
Loop

End Sub

Private Sub Form_Load()
```

```
Con.Open "Provider=MSDAORA.1;User ID=cse101;Password=cse101;Data
Source=ibmrec;Persist Security Info=False"
Con.CursorLocation = adUseClient

Rs.Open "select * from Bank", Con, adOpenKeyset, adLockOptimistic
```

End Sub

Employees Management System

Design Window

Adding Library

Project = > References => Microsoft ActiveX Data Objects 2.0 Library

Coding

```
Dim Con As New ADODB.Connection
Dim Rs As New ADODB.Recordset
```

Private Sub CmdAddNew_Click()

```
Rs.AddNew
txtClear
Text1.SetFocus

End Sub

Private Sub CmdDelete_Click()

On Error Resume Next

If Rs.EOF Then
MsgBox "No Records"
Else
Rs.Delete
Rs.MoveNext
DisplayText
MsgBox "Record Deleted"
End If

End Sub

Private Sub CmdMoveFirst_Click()

Rs.MoveFirst
DisplayText

End Sub


Private Sub CmdMoveNext_Click()

Rs.MoveNext
If Not Rs.EOF Then
DisplayText
Else
MsgBox "End Of The Record"
End If

End Sub

Private Sub CmdUpdate_Click()

Rs(0).Value = Val(Text1.Text)
Rs(1).Value = Text2.Text
Rs(2).Value = Text3.Text
Rs(3).Value = Text4.Text

Rs.Update
```

```
MsgBox "Record Updated"

End Sub

Private Sub Form_Load()

Con.Open "Provider=MSDAORA.1;User ID=cse101;Password=cse101;Data
Source=ibmrec;Persist Security Info=False"
Con.CursorLocation = adUseClient

Rs.Open "select * from employees", Con, adOpenKeyset, adLockOptimistic

End Sub

Sub txtClear()

Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""

End Sub

Sub DisplayText()

Text1.Text = Rs(0).Value
Text2.Text = Rs(1).Value
Text3.Text = Rs(2).Value
Text4.Text = Rs(3).Value

End Sub
```

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result: Thus, the mini project has been completed and the output is verified

Ex. No: 15 Date:	Content Beyond Syllabus: Oracle Installation

Aim:

To install and configure the Oracle in windows

Introduction

The Oracle Universal Installer (OUI) is used to install the Oracle Database software. OUI is a graphical user interface utility that enables you to:

- View the Oracle software that is installed on your machine
- Install new Oracle Database software
- Delete Oracle software that is no longer required

During the installation process, OUI will start the Oracle Database Configuration Assistant (DBCA) which can install a precreated default database that contains example schemas or can guide you through the process of creating and configuring a customized database. If you do not create a database during installation, you can invoke DBCA after you have installed the software, to create one or more databases.

Hardware and Software Requirements

Before installing the software, OUI performs several automated checks to ensure that your computer fulfills the basic hardware and software requirements for an Oracle Database installation. If your computer does not meet the requirements, an error message is displayed. Some of the requirements to install the software are:

- Minimum 2 GB of physical memory
- Sufficient virtual memory (swap)
- At least 10 GB of free disk space

Downloading an oracle database software

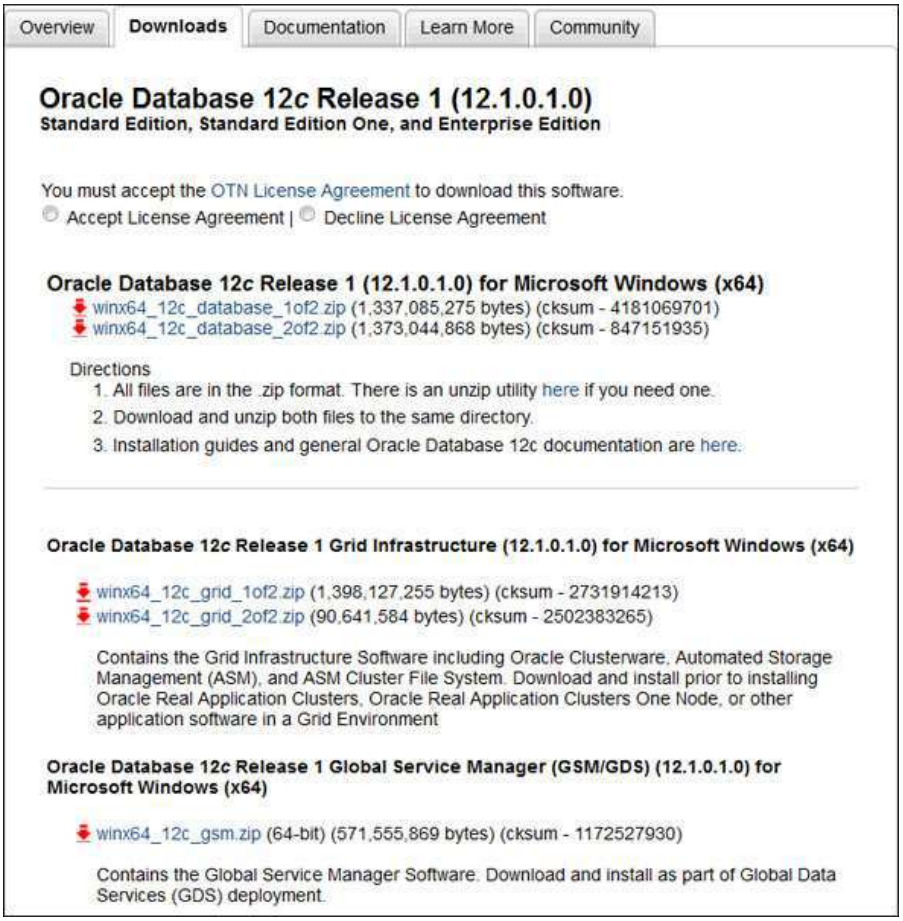
Step1: Open a web browser of your choice and navigate **to http://otn.oracle.com/windows**. By default, the page displays the **what's New** tab, showcasing news about Oracle on Windows.

Note: In this OBE, we use Internet Explorer to download the software.

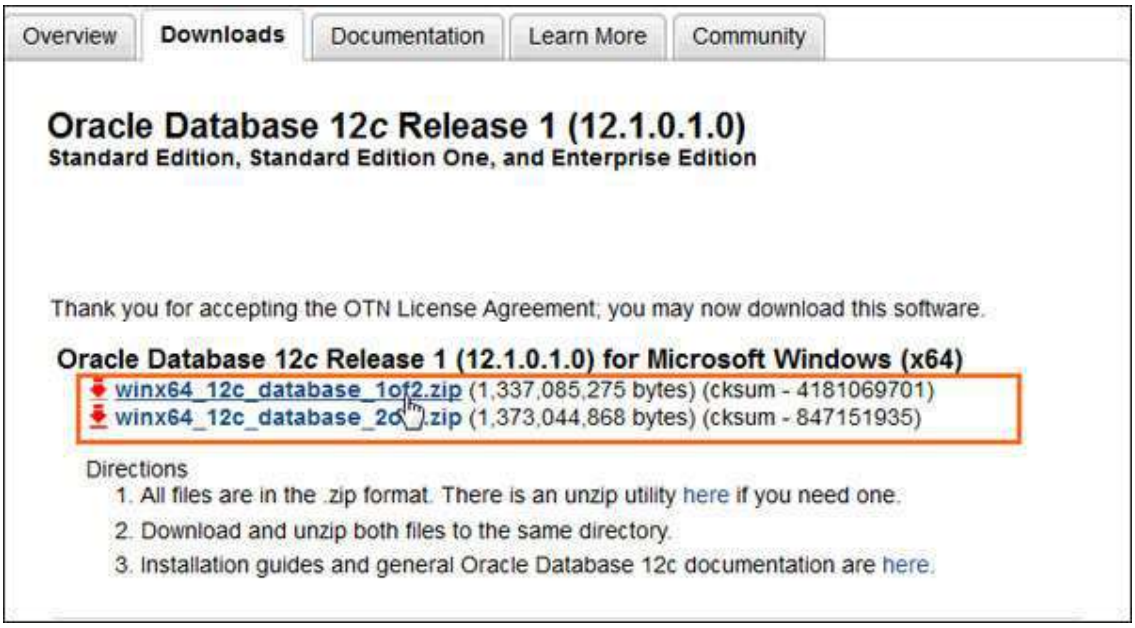
Step 2: Click the **Downloads** tab.

Step 3: Click on the latest version of **Oracle Database 12c (x64)**.

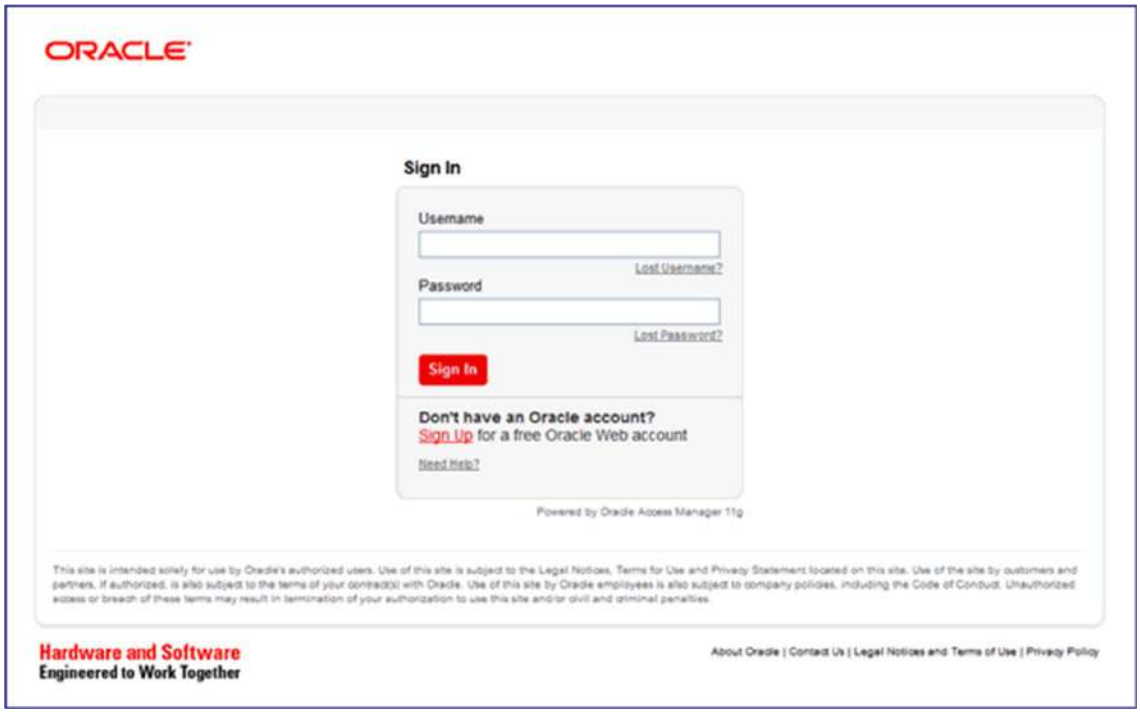
Step 4: The software downloads page displays the files required to download the database. It lists a set of software files to install software such as, **Oracle Database Grid Infrastructure** and **Oracle Database Gateways** with the Oracle database. Each set displays a short description of what the file includes. You can choose to download any set of software depending on your requirement.



Step 5: In this OBE, we install Oracle Database 12c. Accept the license agreement and click the files under **Oracle Database 12c Release ... for Microsoft Windows (x64)** to download.

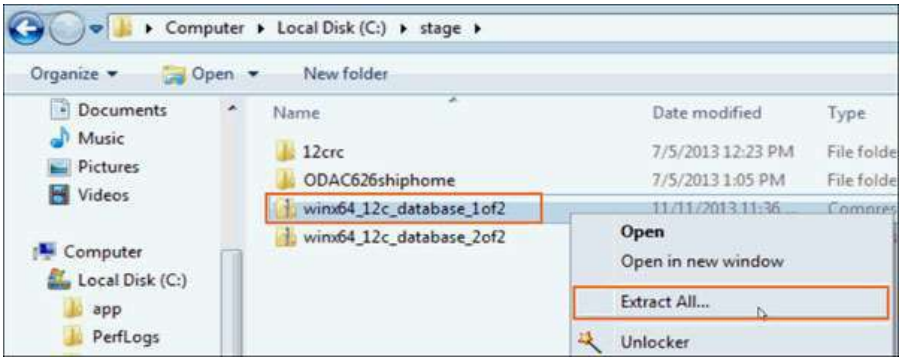


Step 6: Log into your Oracle web account. If you do not have an Oracle account, click the "Sign Up" link to create one. Then choose the location where you want to download the .zip files.

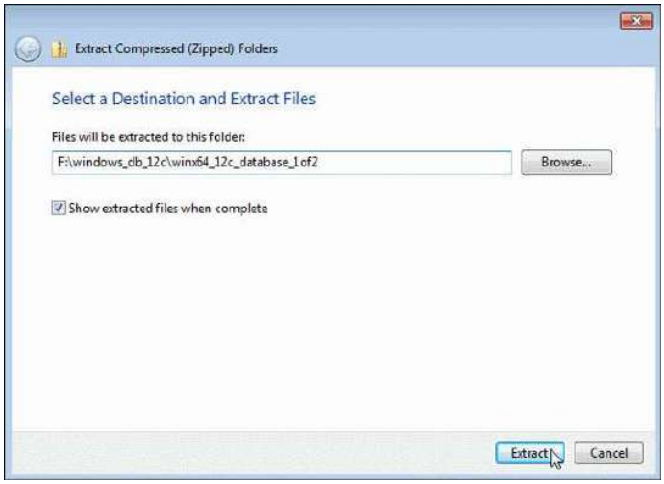


Step 7: After downloading the files, use the default built-in extraction tool provided by Windows, or tools such as 7-zip to extract the .zip files. In this OBE, we use the built-in extractor to extract the software files.

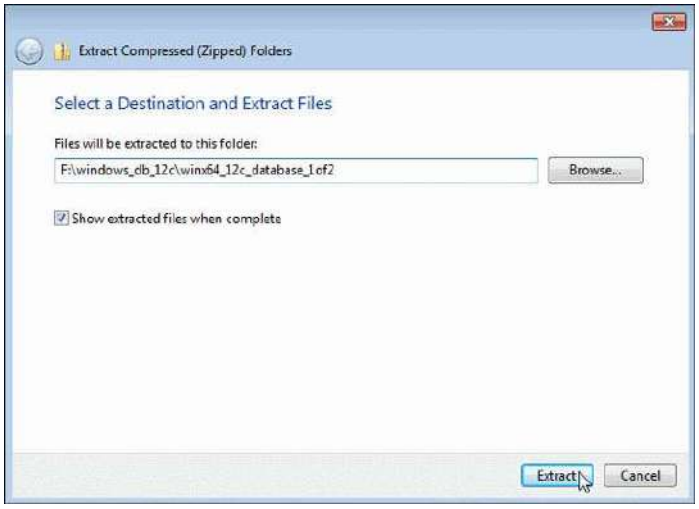
Right click **winx64_12c_database_1of2** and select **Extract All...**



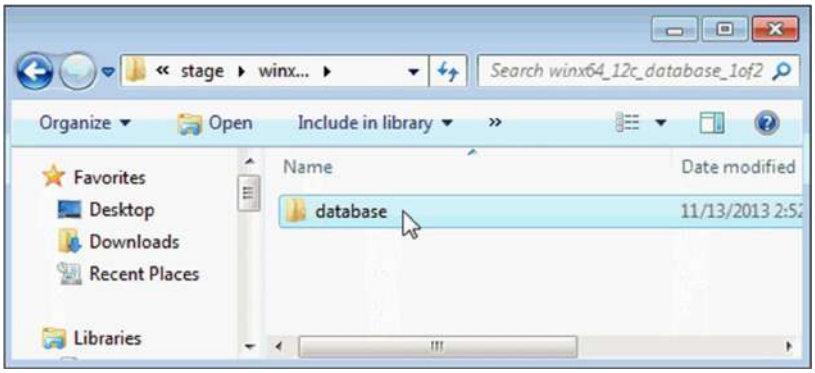
Step 8: Choose the folder of your choice and click **Extract.** In this OBE, we extract the file to the **F:\windows_db_12c\winx64_12c_database_1of2** folder.



Step 9: The file winx64_12c_database_2of2.zip must be extracted into the same folder where the first file was extracted. Right click **winx64_12c_database_2of2.zip**, select the **Extract All...** option and specify the same location where the first file was extracted. In this OBE, we extract the files to the **F:\windows_db_12c\winx64_12c_database_1of2** folder. Click **Extract.**



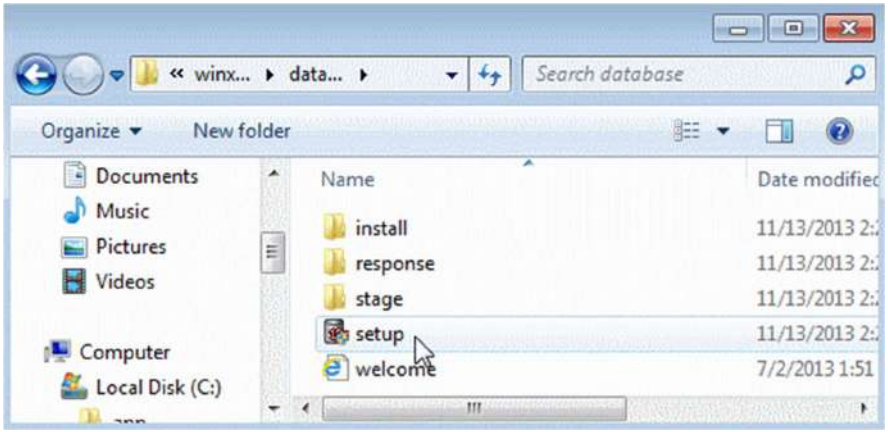
Step 10: The software files are extracted. Expand the winx64_12c_database_1 folder.



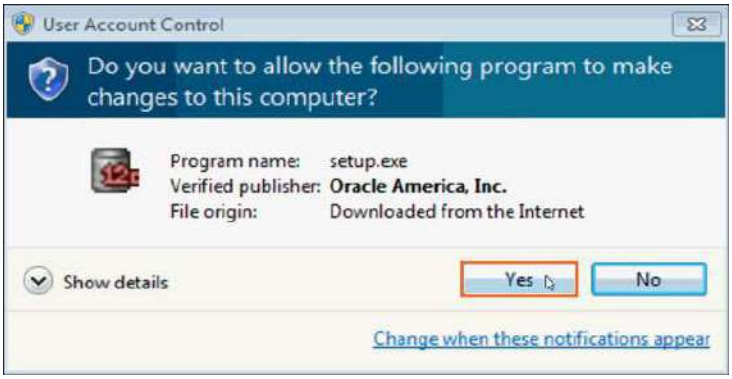
Installing Oracle Database Software

Step 1: Expand the **database** folder that you extracted in the previous section.

Double-click **setup**.



Step 2: Click **Yes** in the User Account Control window to continue with the installation.

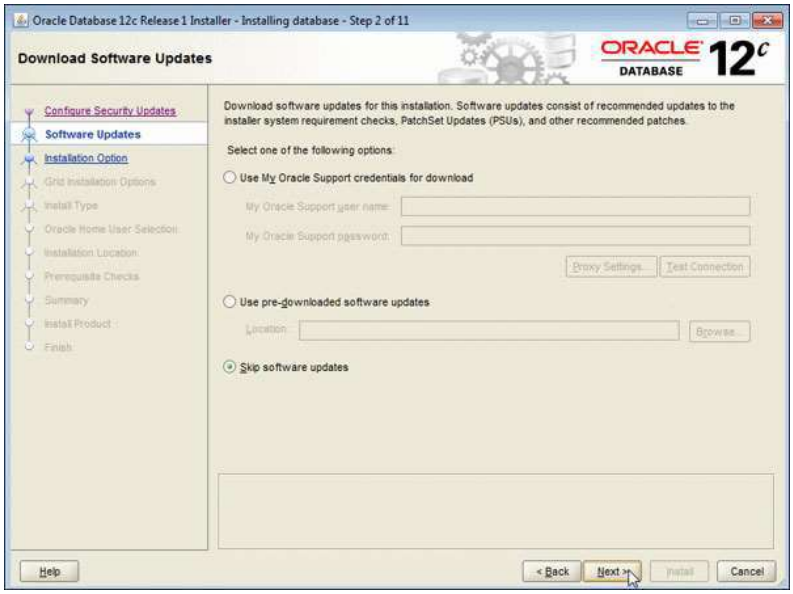


Step 3: The **Configure Security Updates** window appears. Enter your email address and My Oracle Support password to receive security issue notifications via email. If you do not wish to receive notifications via email, deselect "**I wish to receive security updates via My Oracle Support**". Click **Next** to continue. Click "**Yes**" in the confirmation window to confirm your preference.

Step 4: The **Download Software Updates** window appears with the following options:

- Select "Use My Oracle Support credentials for download" to download and apply the latest software updates.
- Select "Use pre-downloaded software updates" to apply software updates that you previously downloaded.
- Select "Skip software updates" if do not want to apply any updates.

Accept the default and click **Next**.

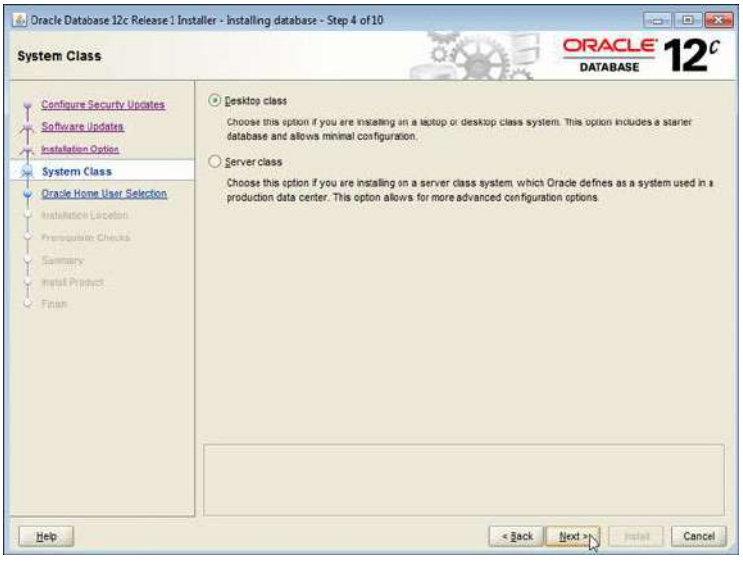


Step 5: The **Select Installation Option** window appears with the following options:

- Select "Create and configure a database" to install the database, create database instance and configure the database.
- Select "Install database software only" to only install the database software.
- Select "Upgrade an existing database" to upgrade the database that is already installed.

In this OBE, we create and configure the database. Select the **Create and configure a database** option and click **Next**.

Step 6: The **System Class** window appears. Select Desktop Class or Server Class depending on the type of system you are using. In this OBE, we will perform the installation on a desktop/laptop. Select **Desktop class** and click **Next**.



Step 7: The **Oracle Home User Selection** window appears. Starting with Oracle Database 12c Release 1 (12.1), Oracle Database on Microsoft Windows supports the use of an Oracle Home User, specified at the time of installation. This Oracle Home User is used to run the Windows services for a Oracle Home, and is similar to the Oracle User on Oracle Database on Linux. This user is associated with an Oracle Home and cannot be changed to a different user post installation.

Note:

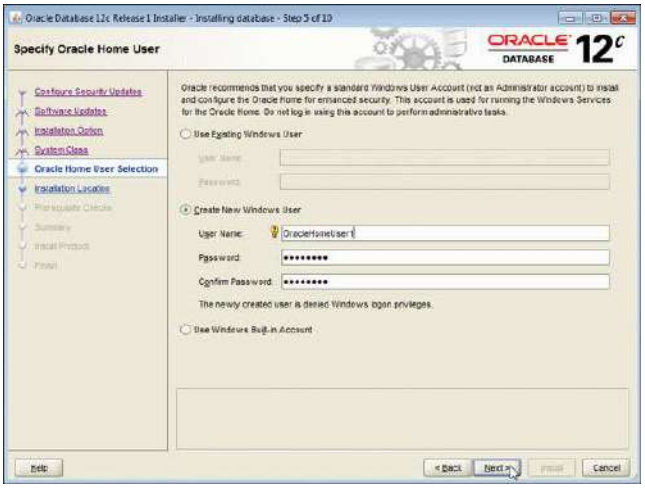
- Different Oracle homes on a system can share the same Oracle Home User or use different Oracle Home Users.
- The Oracle Home User is different from an Oracle Installation User. The Oracle Installation User is the user who requires administrative privileges to install Oracle products. The Oracle Home User is used to run the Windows services for the Oracle Home.

The window provides the following options:

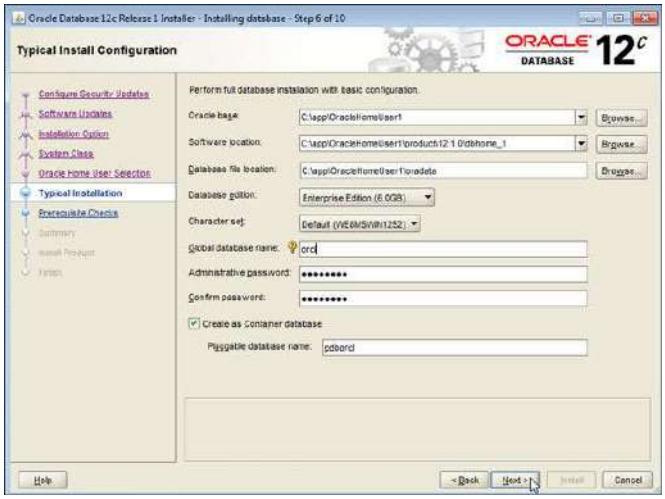
- If you select "Use Existing Windows User", the user credentials provided must be a standard Windows user account (not an administrator).
  - If this is a single instance database installation, the user can be a local user, a domain user, or a managed services account.
  - If this is an Oracle RAC database installation, the existing user must be a Windows domain user. The Oracle installer will display an error if this user has administrator privileges.
- If you select "Create New Windows User", the Oracle installer will create a new standard Windows user account. This user will be assigned as the Oracle Home User. Please note that this user will not have login privileges. This option is not available for an Oracle RAC Database installation.
- If you select "Use Windows Built-in Account", the system uses the Windows Built-in account as the Oracle Home User.

Select the **Create New Windows User** option. Enter the user name as **OracleHomeUser1** and password as **Welcome1**. Click **Next**.

Note: Remember the Windows User password. It will be required later to administer or manage database services.

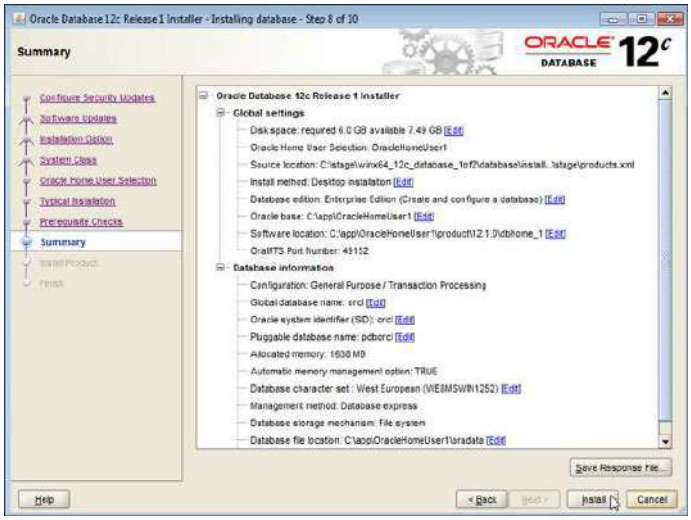


Step 8: The **Typical Install Configuration** window appears. Click on a text field and then the balloon icon ()to know more about the field. Note that by default, the installer creates a container database along with a pluggable database called "`pdborcl`". The pluggable database contains the sample `HR` schema. Change the Global database name to **orcl**. Enter the "Administrative password" as **Oracle_1**. This password will be used later to log into administrator accounts such as `SYS` and `SYSTEM`. Click **Next**.

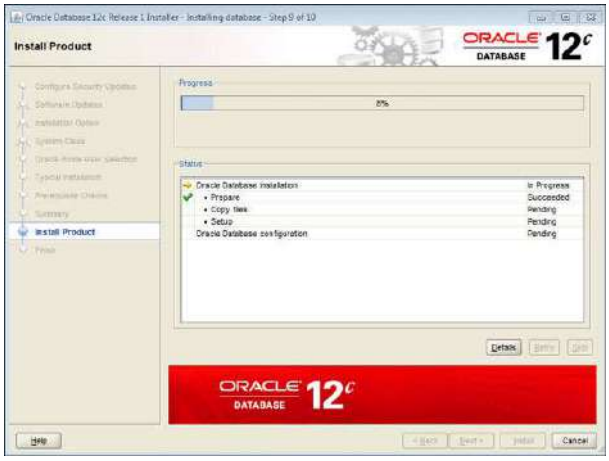


Step 9: The prerequisite checks are performed and a **Summary** window appears. Review the settings and click **Install**.

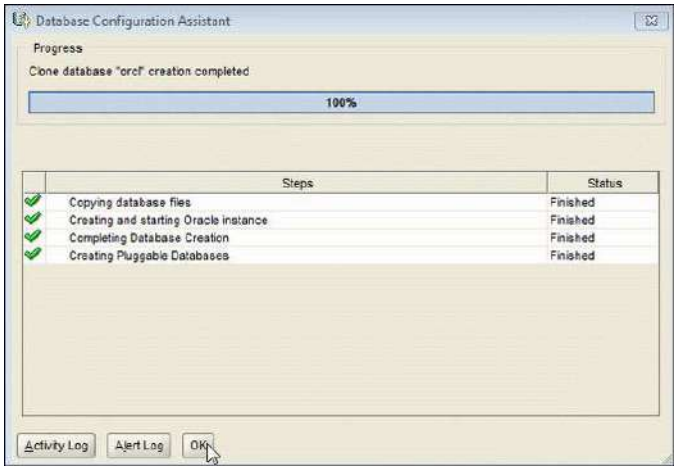
Note: Depending on your firewall settings, you may need to grant permissions to allow java to access the network.



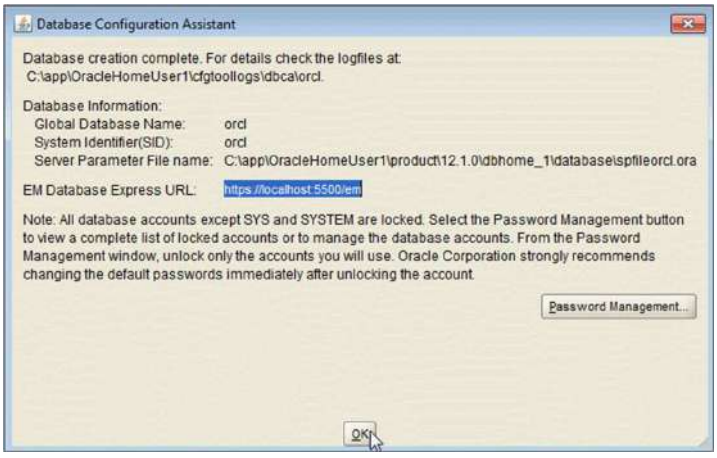
Step 10: The progress window appears.



Step 11: The Database Configuration Assistant creates the database.



Step 12: After the Database Configuration Assistant creates the database, you can navigate to https://localhost:5500/em as a `SYS` user to manage the database using Enterprise Manager Database Express. You can click "Password Management..." to unlock accounts. Click **OK** to continue.

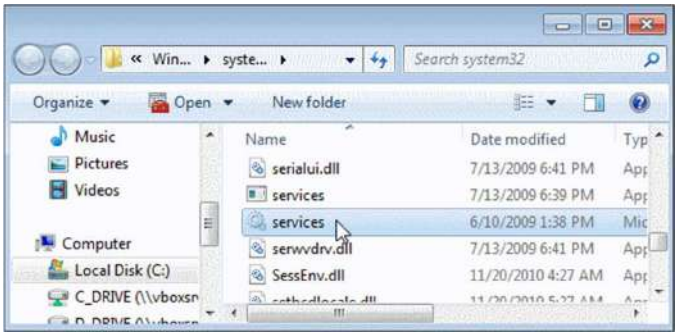


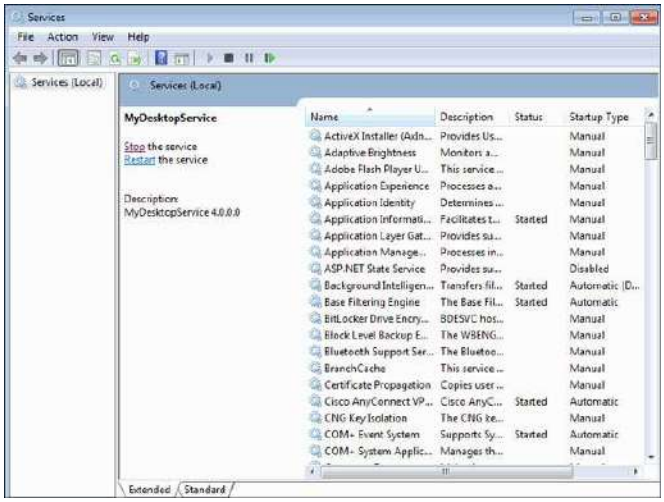
Step 13: The **Finish** window appears. Click **Close** to exit the Oracle Universal Installer.

Verifying the Installation

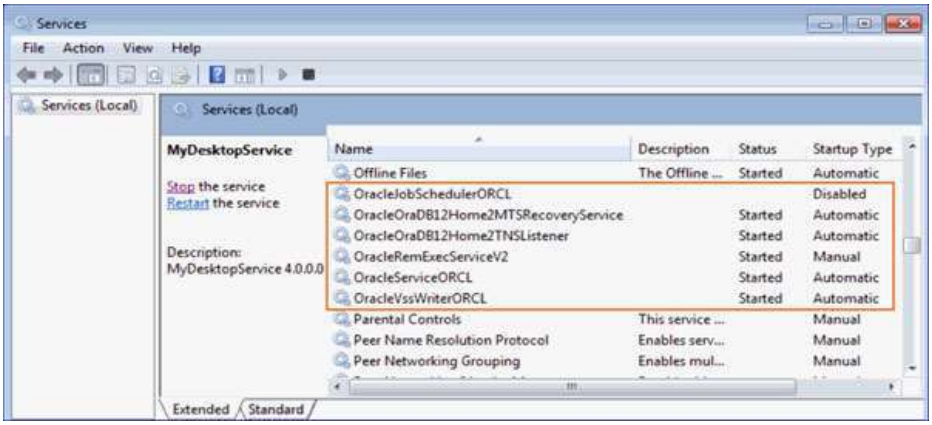
View Oracle Services

Step 1: Navigate to C:\Windows\system32 **using Windows Explorer. Double-click** services**. The** Services **window appears, displaying a list of services.**



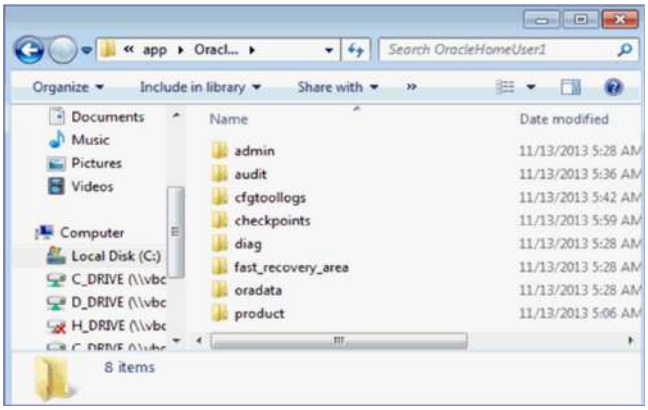


Step 2: Scroll down to view a list Oracle services. You see that most of the Oracle services are started successfully by the database.

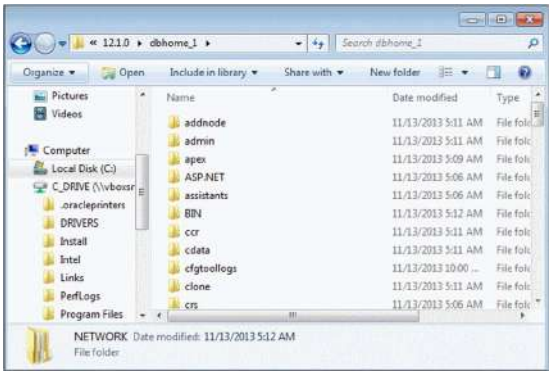


View Oracle Home on the File System

Step 1: Navigate to the **C:\app\OracleHomeUser1** folder. This folder contains database files (in `oradata` folder) and the Oracle Database software (in the `product` folder).

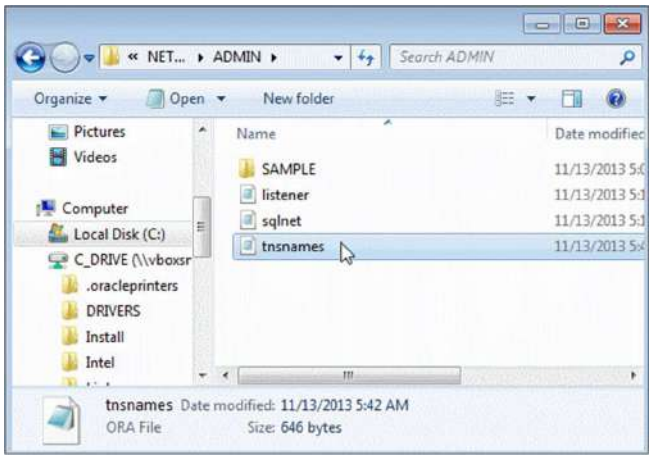


Step 2: Navigate to **`C:\app\OracleHomeUser1\product\12.1.0\dbhome_1`** folder. This folder is the new "Oracle Home" created by the installer and contains software files related to the database.

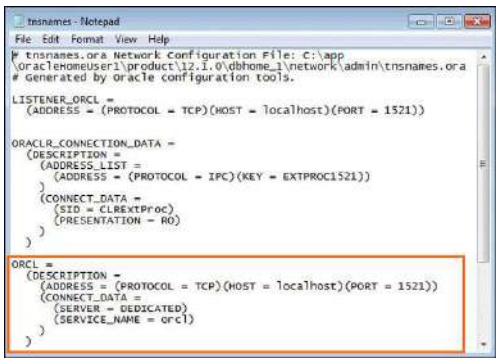


View the tnsnames.ora File

Step 1: Navigate to C:\app\OracleHomeUser1\product\12.1.0\dbhome_1\NETWORK\ADMIN**. Double-click** tnsnames.ora **to view the network configuration settings.**

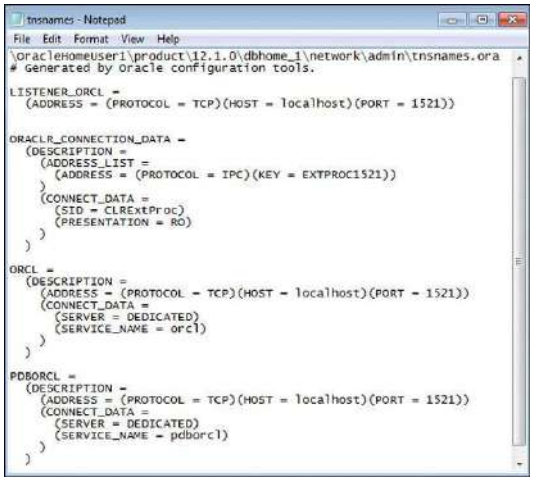


Step 2: You see that a connect alias called "ORCL" has been created. This "ORCL" alias points to the container database with the service name "ORCL".



Step 3: Create a database connect alias called "PDBORCL" and specify the network configuration settings to access the pluggable database "PDBORCL" that we created during installation. Copy the following code and paste it in the **tnsnames.ora** file. If necessary, modify the host and port to match the values in the ORCL alias.

```
PDBORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = pdborcl)
    )
  )
```



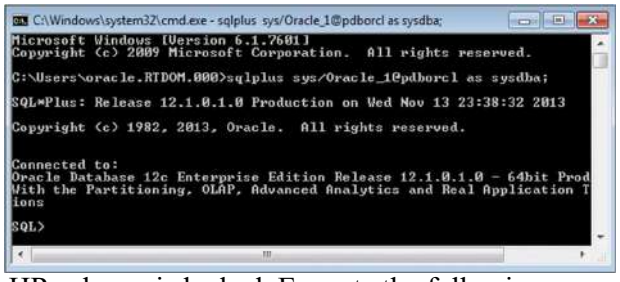
Step 4: Save and close the file.

Connecting to Oracle Database Using SQL*Plus

Step 1: Open a command prompt and execute the following command. This command uses SQL*Plus to connect to the pluggable database as a system administrator:

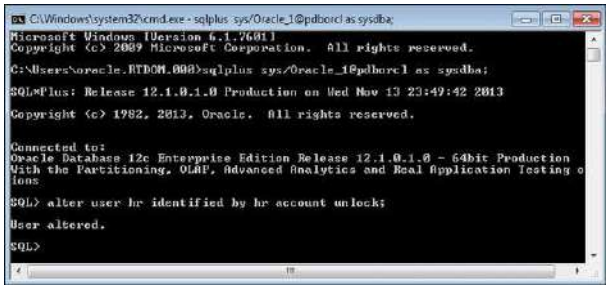
sqlplus sys/Oracle_1@pdborcl as sysdba;

Note: If you had chosen a different administrative password during installation, replace Oracle_1 with the appropriate password in the command.



Step 2: By default, the HR schema is locked. Execute the following command to unlock the HR schema.

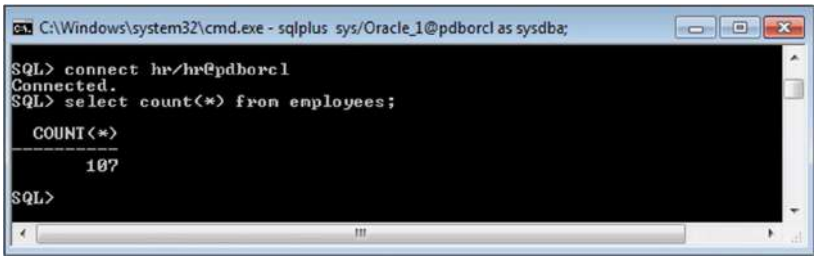
alter user hr identified by hr account unlock;



Step 3: Execute the following commands to connect to the HR schema in the pluggable database and query the EMPLOYEES table.

connect hr/hr@pdborcl

select count(*) from employees;



Step 4: The EMPLOYEES table contains 107 rows. This verifies that the HR schema was created successfully when the database was installed.

Department of AIML		
Performance	**25**	
Record	**15**	
Viva	**10**	
Total	**50**	

Result: Thus, the oracle has been installed, configured and verified by executing the SQL Statements