



MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY **(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)**

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015
Maisammaguda, Dhulapally, Komapally, Secunderabad - 500100, Telangana State, India

LABORATORY MANUAL & RECORD

Name:

Roll No: Branch:

Year: Sem:





MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY **(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)**

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015
Maisammaguda, Dhulapally, Komapally, Secunderabad - 500100, Telangana State, India

Certificate

Certified that this is the Bonafide Record of the Work Done by
Mr./Ms.....Roll.No.....of
B.Tech.....year..... Semester for Academic year.....
in.....Laboratory.

Date:

Faculty Incharge

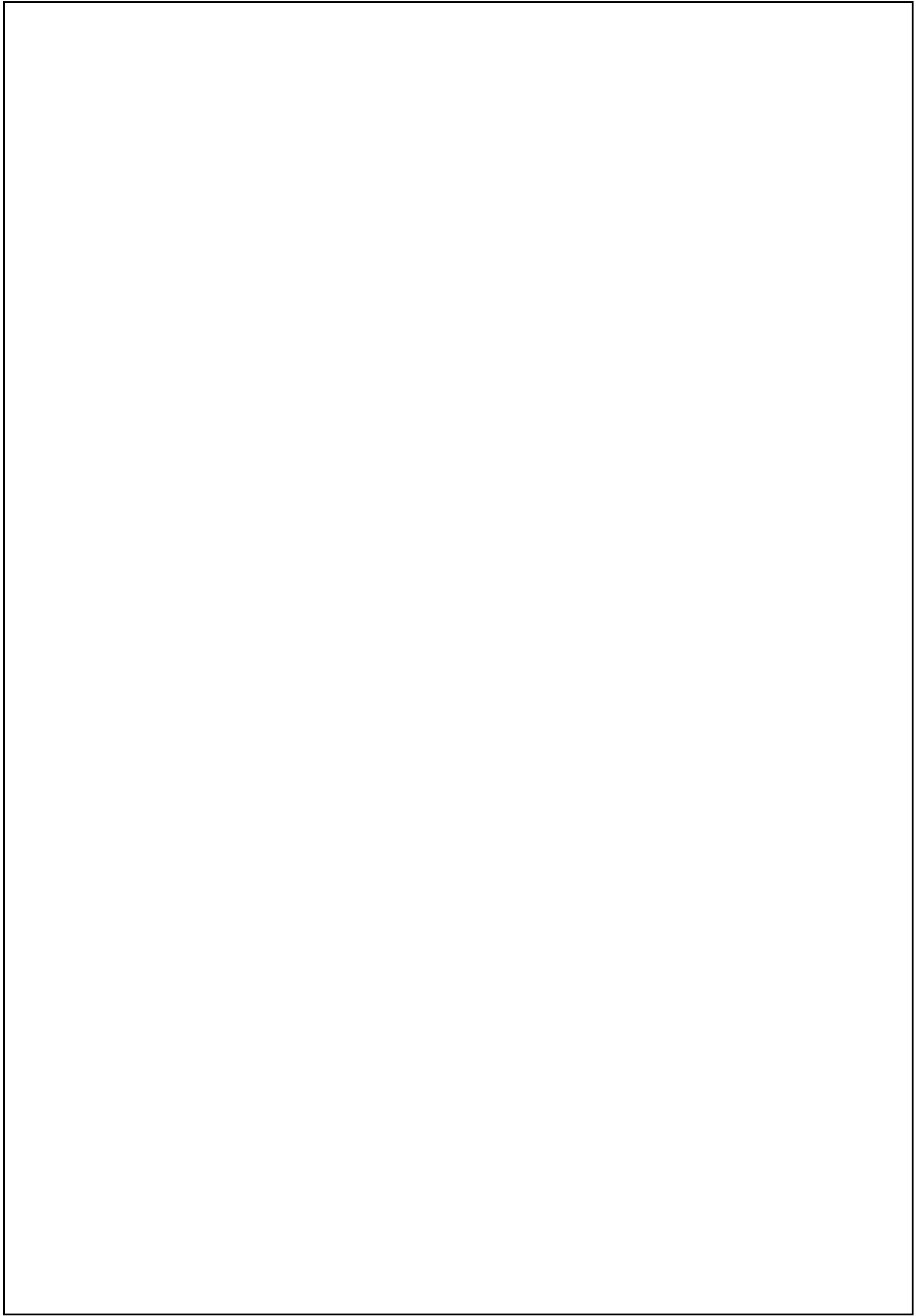
HOD

Internal Examiner

External Examiner

INDEX

[illegible]



DATABASE MANAGEMENT SYSTEMS
LAB MANUAL
(R22A0504)

B.TECH



(II YEAR – II SEM)
(2023-24)



DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
(CSE-AIML, AIML & AIDS)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC - 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

Department of Computer Science & Engineering

(Artificial Intelligence & Machine Learning)

Vision

To be a premier centre for academic excellence and research through innovative interdisciplinary collaborations and making significant contributions to the community, organizations, and society as a whole.

Mission

- To impart cutting-edge Artificial Intelligence technology in accordance with industry norms.
- To instill in students a desire to conduct research in order to tackle challenging technical problems for industry.
- To develop effective graduates who are responsible for their professional growth, leadership qualities and are committed to lifelong learning.

Quality Policy

- To provide sophisticated technical infrastructure and to inspire students to reach their full potential.
- To provide students with a solid academic and research environment for a comprehensive learning experience.
- To provide research development, consulting, testing, and customized training to satisfy specific industrial demands, thereby encouraging self-employment and entrepreneurship among students.

Department of Computer Science & Engineering
(Artificial Intelligence & Machine Learning)

Programme Educational Objectives (PEO):

PEO1: To possess knowledge and analytical abilities in areas such as maths, science, and fundamental engineering.

PEO2: To analyse, design, create products, and provide solutions to problems in Computer Science and Engineering.

PEO3: To leverage the professional expertise to enter the workforce, seek higher education, and conduct research on AI-based problem resolution.

PEO4: To be solution providers and business owners in the field of computer science and engineering with an emphasis on artificial intelligence and machine learning.

Programme Specific Outcomes (PSO):

After successful completion of the program a student is expected to have specific abilities to:

PSO1: To understand and examine the fundamental issues with AI and ML applications.

PSO2: To apply machine learning, deep learning, and artificial intelligence approaches to address issues in social computing, healthcare, vision, language processing, speech recognition, and other domains.

PSO3: Use cutting-edge AI and ML tools and technology to further your study and research.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – PROFESSIONALISM & CITIZENSHIP

To create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with due consideration for ethical, ecological and economic issues.

PEO2 – TECHNICAL ACCOMPLISHMENTS

To provide knowledge based services to satisfy the needs of society and the industry by providing hands on experience in various technologies in core field.

PEO3 – INVENTION, INNOVATION AND CREATIVITY

To make the students to design, experiment, analyze, interpret in the core field with the help of other multi-disciplinary concepts wherever applicable.

PEO4 – PROFESSIONAL ETHICS

To educate the students to disseminate research findings with good soft skills and become a successful entrepreneur.

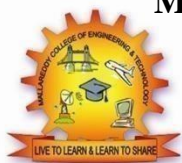
PEO5 – HUMAN RESOURCE DEVELOPMENT

To graduate the students in building national capabilities in technology, education and research.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100
DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
(CSE-AIML, AIML & AIDS)

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
 - a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
 - b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.
 - c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly

HEAD OF THE DEPARTMENT

PRINCIPAL

INDEX

S. No	Topic	Sign
1	Introduction SQL-SQL*Plus	
2	Concept design with ER model	
3	Relational model	
4	Normalization	
5	Practicing DDL commands	
6	Practicing DML commands	
7 a.	Querying	
7 b.	Nested queries	
8	Aggregate functions, GROUPBY, HAVING,VIEWS	
9	Triggers	
10	Procedures	
11	Usage of cursors	
12	Installation of Mysql/MongoDB	

INTRODUCTION

Database Management System

This model is like a hierarchical tree structure, used to construct a hierarchy of records in the form of nodes and branches. The data elements present in the structure have Parent-Child relationship. Closely related information in the parent-child structure is stored together as a logical unit. A parent unit may have many child units, but a child is restricted to have only one parent.

The drawbacks of this model are:

The hierarchical structure is not flexible to represent all the relationship proportions,

which occur in the real world.

It cannot demonstrate the overall data model for the enterprise because of the non-availability of actual data at the time of designing the data model.

It cannot represent the Many-to-Many relationship.

Network Model

It supports the One-To-One and One-To-Many types only. The basic objects in this model are Data Items, Data Aggregates, Records and Sets.

It is an improvement on the Hierarchical Model. Here multiple parent-child relationships are used. Rapid and easy access to data is possible in this model due to multiple access paths to the data elements.

Relational Model

Does not maintain physical connection between relations

Data is organized in terms of rows and columns in a table

The position of a row and/or column in a table is of no importance

The intersection of a row and column must give a single value

Features of an RDBMS

The ability to create multiple relations and enter data into them

An attractive query language

Retrieval of information stored in more than one table

An RDBMS product has to satisfy at least Seven of the 12 rules of Codd to be accepted as a full-fledged RDBMS.

Relational Database Management System

RDBMS is acronym for Relation Database Management System. Dr. E. F. Codd first introduced the Relational Database Model in 1970. The Relational model allows data to be represented in a simple row- column. Each data field is considered as a column and each record is considered as a row. Relational Database is more or less similar to Database Management System. In relational model there is relation between their data elements. Data is stored in tables. Tables have columns, rows and names. Tables can be related to each other if each has a column with a common type of information. The most famous RDBMS packages are Oracle, Sybase and Informix.

Simple example of Relational model is as follows :

Student Details Table

<u>Roll no</u>	<u>Sname</u>	<u>S Address</u>
1	Rahul	Satelite
2	Sachin	Ambawadi
3	Saurav	Naranpura

Student Marksheet Table

<u>Rollno</u>	<u>Sub1</u>	<u>Sub2</u>	<u>Sub3</u>
1	78	89	94
2	54	65	77
3	23	78	46

Here, both tables are based on students details. Common field in both tables is Rollno. So we can say both tables are related with each other through Rollno column.

Degree of Relationship

One to One (1:1)

One to Many or Many to One (1:M / M: 1)

Many to Many (M: M)

The Degree of Relationship indicates the link between two entities for a specified occurrence of each.

One to One Relationship: (1:1)

1 1

Student Has Roll No.

One student has only one Rollno. For one occurrence of the first entity, there can be, at the most one related occurrence of the second entity, and vice-versa.

One to Many or Many to One Relationship: (1:M/M: 1)

1 M

Course Contains Students

As per the Institutions Norm, One student can enroll in one course at a time however, in one course, there can be more than one student.

For one occurrence of the first entity there can exist many related occurrences of the second entity and for every occurrence of the second entity there exists only one associated occurrence of the first.

Many to Many Relationship: (M:M)

M M

Students Appears Tests

The major disadvantage of the relational model is that a clear-cut interface cannot be determined. Reusability of a structure is not possible. The Relational Database now accepted model on which major database system are built.

Oracle has introduced added functionality to this by incorporated object-oriented capabilities. Now it is known as Object Relational Database Management System (ORDBMS). Object-oriented concept is added in Oracle8.

Some basic rules have to be followed for a DBMS to be relational. They are known as Codd's rules, designed in such a way that when the database is ready for use it encapsulates the relational theory to its full potential. These twelve rules are as follows.

E. F. Codd Rules

1. The Information Rule

All information must be store in table as data values.

2. The Rule of Guaranteed Access

Every item in a table must be logically addressable with the help of a table name.

3. The Systematic Treatment of Null Values

The RDBMS must be taken care of null values to represent missing or inapplicable information.

4. The Database Description Rule

A description of database is maintained using the same logical structures with which data was defined by the RDBMS.

5. Comprehensive Data Sub Language

According to the rule the system must support data definition, view definition, data manipulation, integrity constraints, authorization and transaction management operations.

6. The View Updating Rule

All views that are theoretically updatable are also updatable by the system.

7. The Insert and Update Rule

This rule indicates that all the data manipulation commands must be operational on sets of rows having a relation rather than on a single row.

8. The Physical Independence Rule

Application programs must remain unimpaired when any changes are made in storage representation or access methods.

9. The Logical Data Independence Rule

The changes that are made should not affect the user's ability to work with the data. The change can be splitting table into many more tables.

10. The Integrity Independence Rule

The integrity constraints should store in the system catalog or in the database.

11. The Distribution Rule

The system must be access or manipulate the data that is distributed in other systems.

12. The Non-subversion Rule

If a RDBMS supports a lower level language then it should not bypass any integrity constraints defined in the higher level.

Object Relational Database Management System

Oracle8 and later versions are supported object-oriented concepts. A structure once created can be reused is the fundamental of the OOP's concept. So we can say Oracle8 is supported Object Relational model, Object - oriented model both. Oracle products are based on a concept known as a client-server technology. This concept involves segregating the processing of an application between two systems. One performs all activities related to the database (server) and the other performs activities that help the user to interact with the application (client). A client or front-end database application also interacts with the database by requesting and receiving information from database server. It acts as an interface between the user and the database.

The database server or back end is used to manage the database tables and also respond to client requests.

Introduction to ORACLE

ORACLE is a powerful RDBMS product that provides efficient and effective solutions for major database features. This includes:

- Large databases and space management control

- Many concurrent database users

- High transaction processing performance

- High availability

- Controlled availability

- Industry accepted standards

- Manageable security

- Database enforced integrity

- Client/Server environment

- Distributed database systems

- Portability

Compatibility

Connectivity

An ORACLE database system can easily take advantage of distributed processing by using its Client/ Server architecture. In this architecture, the database system is divided into two parts:

A front-end or a client portion

The client executes the database application that accesses database information and interacts with the user.

A back-end or a server portion

The server executes the ORACLE software and handles the functions required for concurrent, shared data access to ORACLE database.

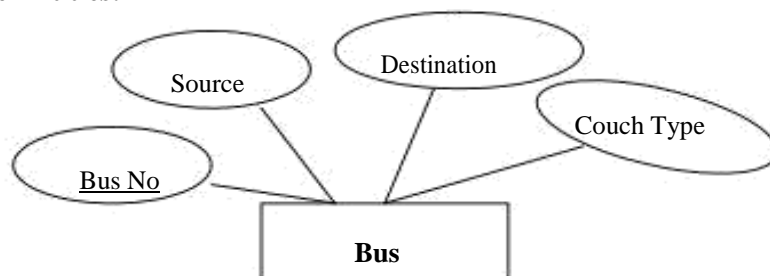
AIM: Analyze the problem and come with the entities in it. Identify what Data has to be persisted in the databases.

The Following are the entities:

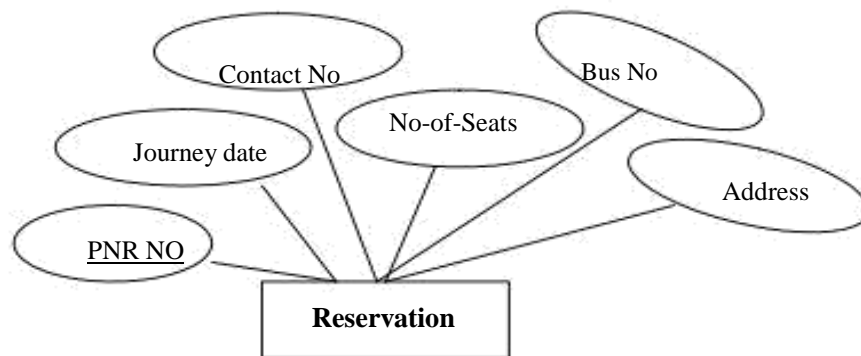
1. Bus
2. Reservation
3. Ticket
4. Passenger
5. Cancellation

The attributes in the Entities:

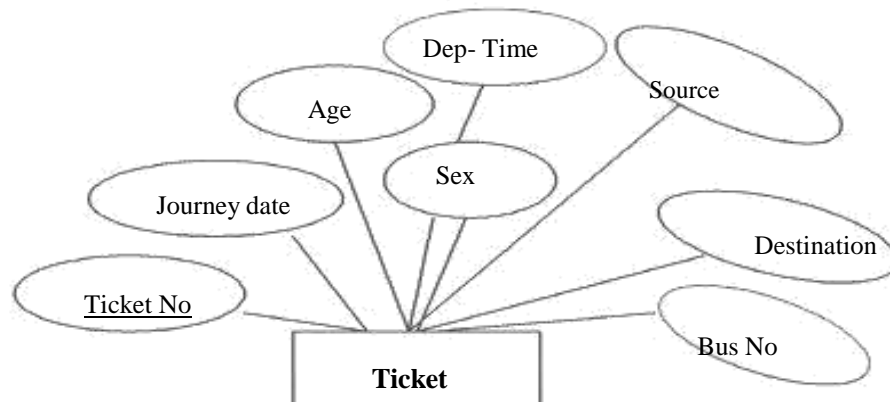
Bus:(Entity)



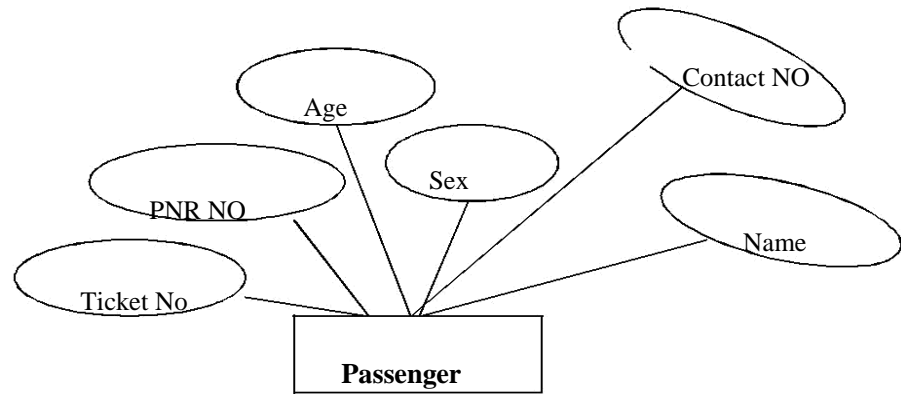
Reservation (Entity)



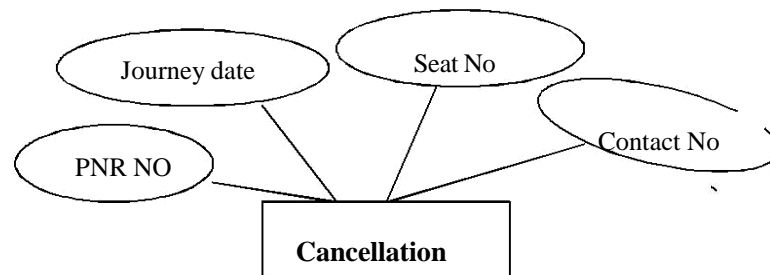
Ticket :(Entity)



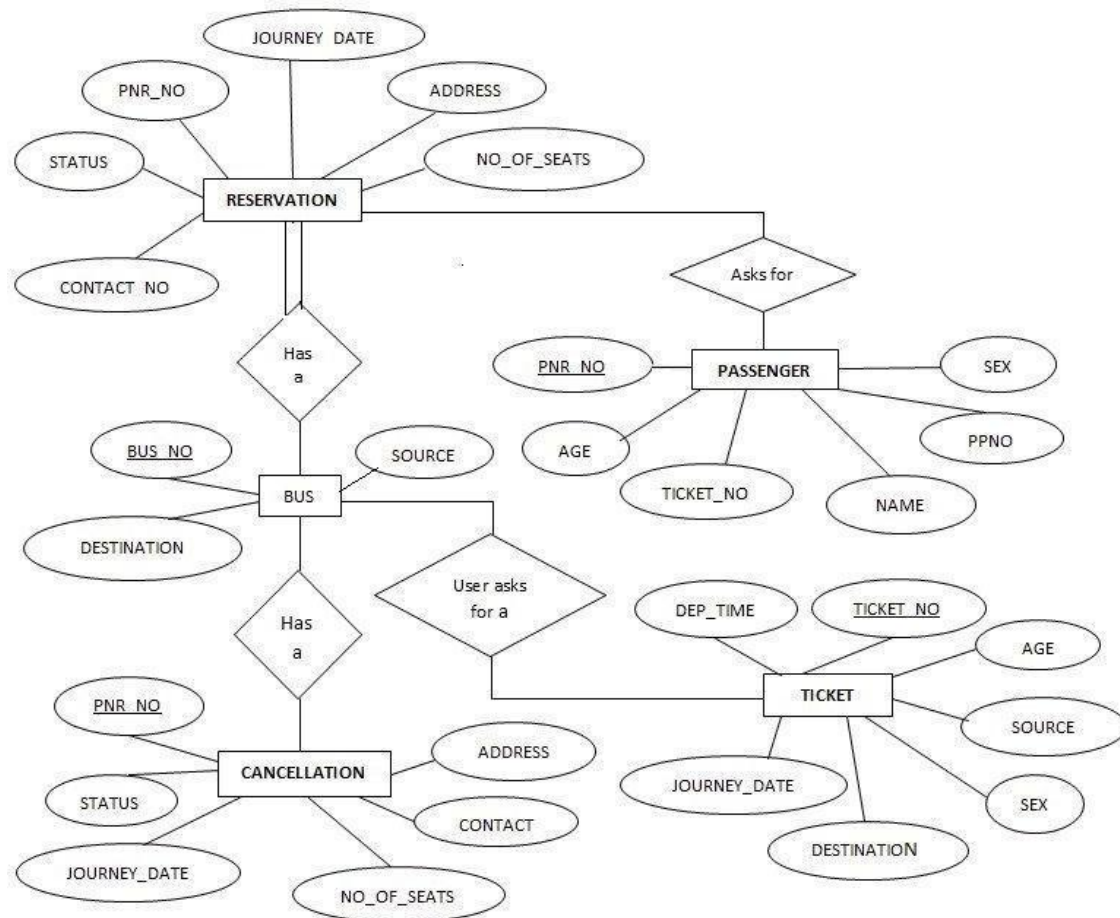
Passenger:



Cancellation (Entity)



1. Concept design with E-R Model:



What is SQL and SQL*Plus

Oracle was the first company to release a product that used the English-based Structured Query Language or SQL. This language allows end users to manipulate information of table(primary database object). To use SQL you need not to require any programming experience. SQL is a standard language common to all relational databases. SQL is database language used for storing and retrieving data from the database. Most Relational Database Management Systems provide extension to SQL to make it easier for application developer. A table is a primary object of database used to store data. It stores data in form of rows and columns.

SQL*Plus is an Oracle tool (specific program) which accepts SQL commands and PL/SQL blocks and executes them. SQL *Plus enables manipulations of SQL commands and PL/SQL blocks. It also performs additional tasks such as calculations, store and print query results in the form of reports, list column definitions of any table, access and copy data between SQL databases and send messages to and accept responses from the user. SQL *Plus is a character based interactive tool, that runs in a GUI environment. It is loaded on the client machine.

To communicate with Oracle, SQL supports the following categories of commands:

1. Data Definition Language

Create, Alter, Drop and Truncate

2. Data Manipulation Language

Insert, Update, Delete and Select

3. Transaction Control Language

Commit, Rollback and Save point

4. Data Control Language

Grant and Revoke

AIM: Installation of MySQL and practicing DDL & DML commands.

1. Steps for installing MySQL

Step1

1

Make sure you already downloaded the **MySQL essential 5.0.45 win32.msi file**. Double click on the .msi file.

Step2

2

This is MySQL Server 5.0 setup wizard. The setup wizard will install MySQL Server 5.0 release 5.0.45 on your computer. To continue, click **next**.



Step3

3

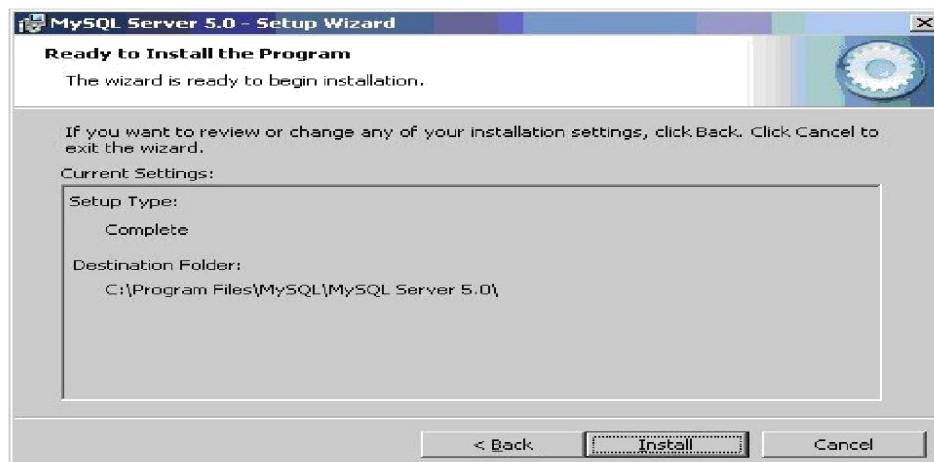
Choose the setup type that best suits your needs. For common program features select *Typical* and it's recommended for general use. To continue, click **next**.



Step4

4

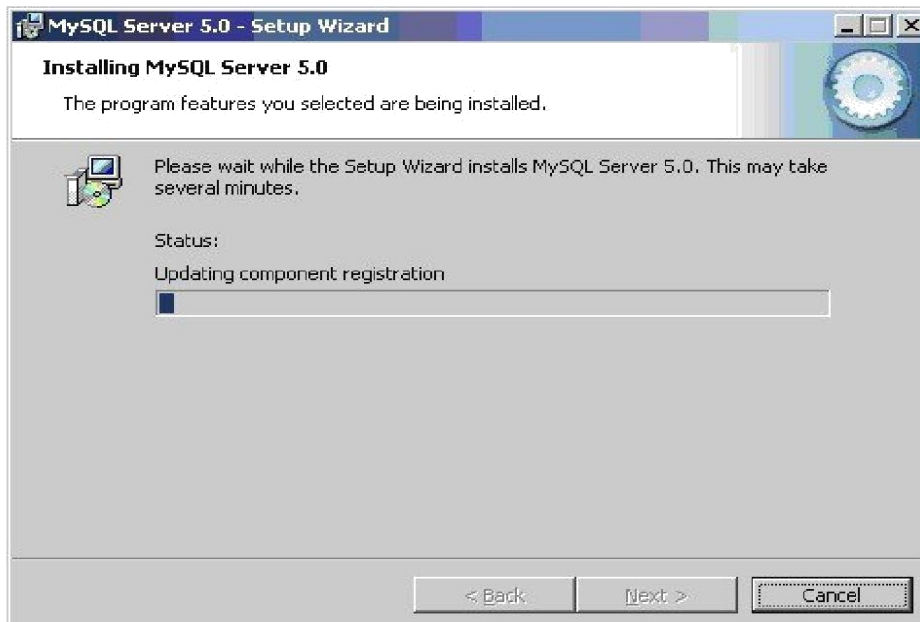
This wizard is ready to begin installation. Destination folder will be in **C:\Program Files\MySQL\MySQL Server 5.0**. To continue, click **next**.



Step5

5

The program features you selected are being installed. Please wait while the setup wizard installs MySQL 5.0. This may take several minutes.



Step6

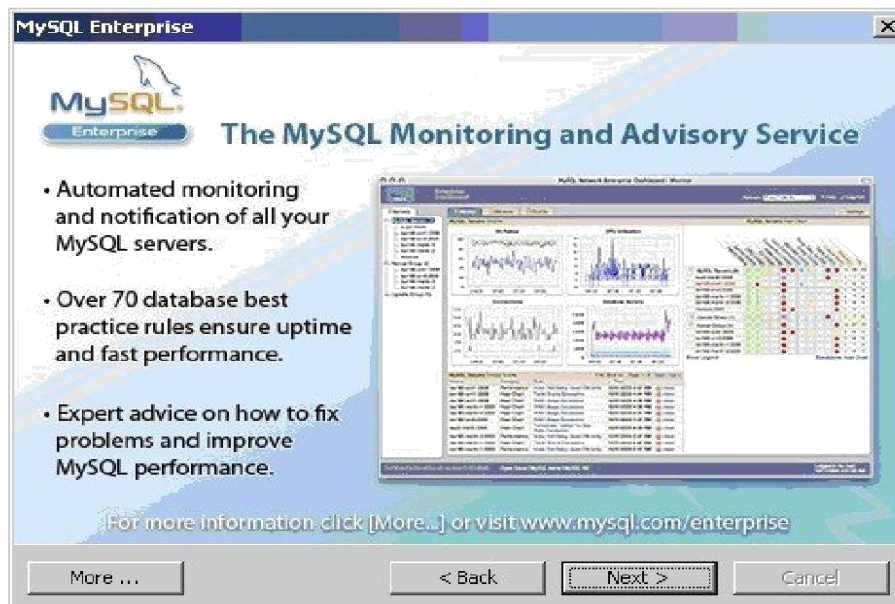
To continue, click **next**.



Step7

To continue, click **next**.

7



Step8

8

Wizard Completed. Setup has finished installing MySQL 5.0. **Check** the configure the MySQL server now to continue. Click **Finish** to exit the wizard

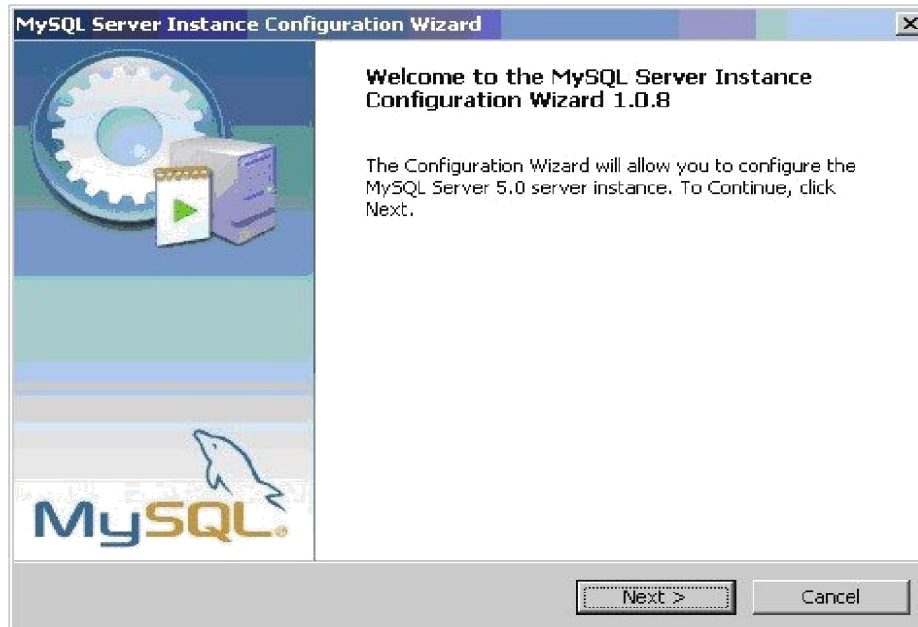


Step9

9

The configuration wizard will allow you to configure the MySQL Server 5.0 server instance.

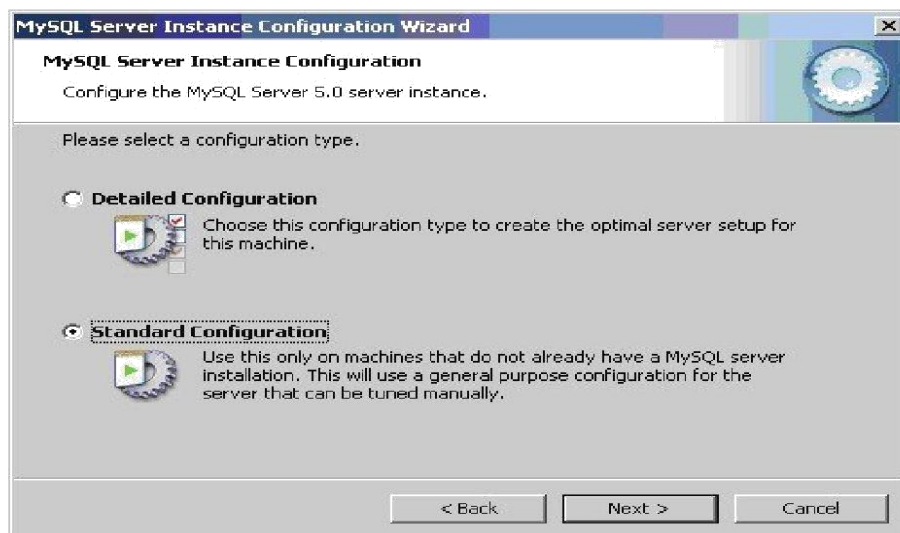
To continue, click **next**.



Step10

10

Select a **standard configuration** and this will use a general purpose configuration for the server that can be tuned manually. To continue, click **next**.



Step11

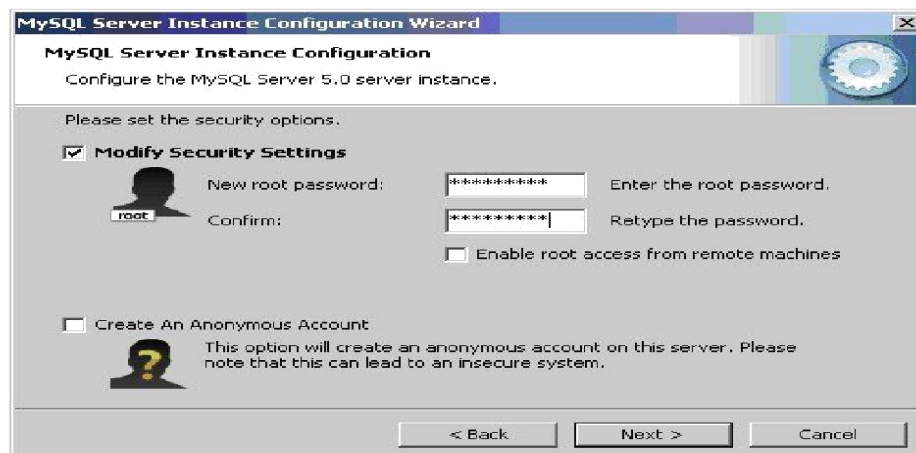
Check on the install as windows service and include bin directory in windows path. To continue, click next.



Step12

12

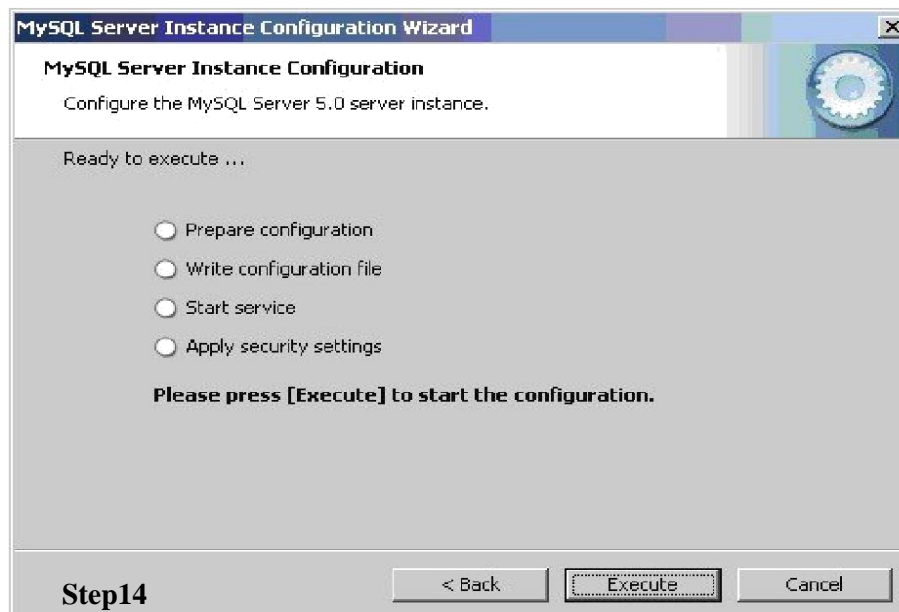
Please set the security options by entering the root password and confirm retype the password. continue, click next.



Step13

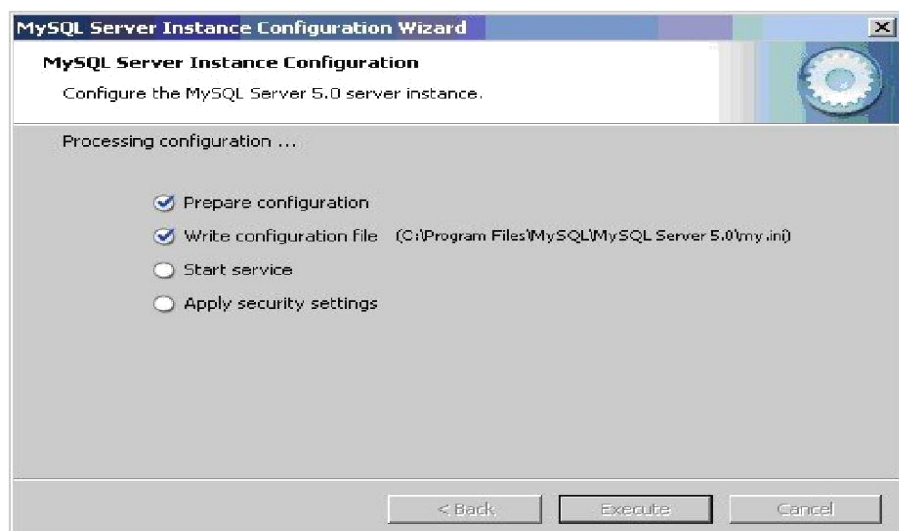
13

Ready to execute? Clicks **execute** to continue.



Processing configuration in progress.

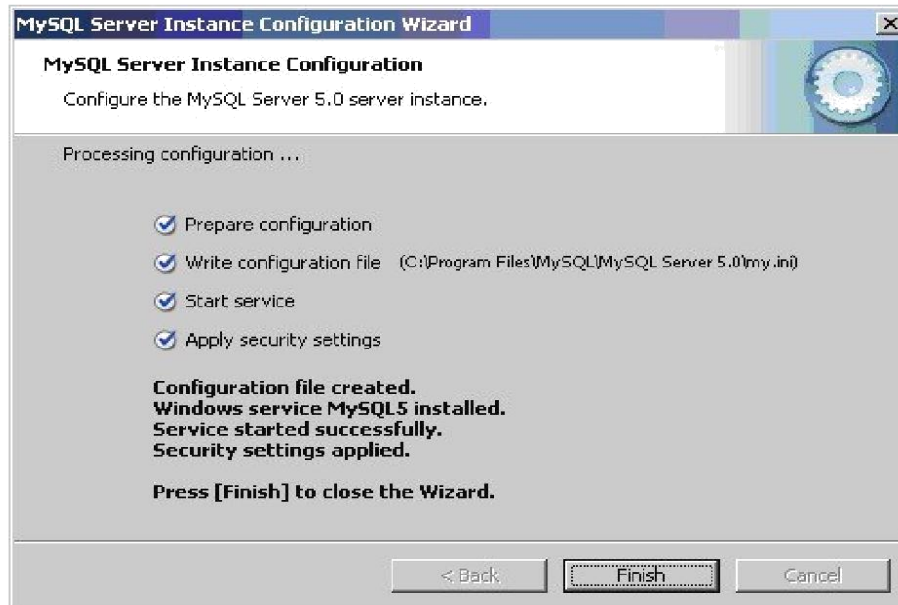
14



Step15

15

Configuration file created. Windows service MySQL5 installed. Press **finish** to close the wizard.



CREATION OF TABLES

(RELATIONAL MODEL)

1) Create a table called Employee with the following structure.

Name Type

Empno Number

Ename Varchar2(10)

Job Varchar2(10)

Mgr Number

Sal Number

- a. Add a column commission with domain to the Employee table.
- b. Insert any five records into the table.
- c. Update the column details of job
- d. Rename the column of Employ table using alter command.
- e. Delete the employee whose Emp no is 105.

SOLUTION:

```
SQL>create table employee (empnonumber,  
ename varchar2(10), job varchar2(10), mgr  
number, sal number);
```

Table created.

a. Add a column commission with domain to the Employee table.

```
SQL> alter table employee add  
(commission number);
```

Table altered.

```
SQL> desc employee;
```

EMPNO NUMBER

ENAME VARCHAR2(10)

JOB VARCHAR2(10)

MGR NUMBER

SAL NUMBER

COMMISSION NUMBER

b. Insert any five records into the table.

```
SQL> INSERT INTO employee (empno,'ename','job', mgr, sal,'commission') VALUES ('101', 'abhi',  
'manager',50000,10000);
```

Repeat above query 4 times.

c. Update the column details of table.

```
SQL> update employee set sal=90000 where empno=101;
```

d. Rename the column of Employee table using alter command.

```
SQL> ALTER TABLE employee RENAME  
COLUMN empno TO employ_id;
```

e. Delete the employee whose Emp no is 105.

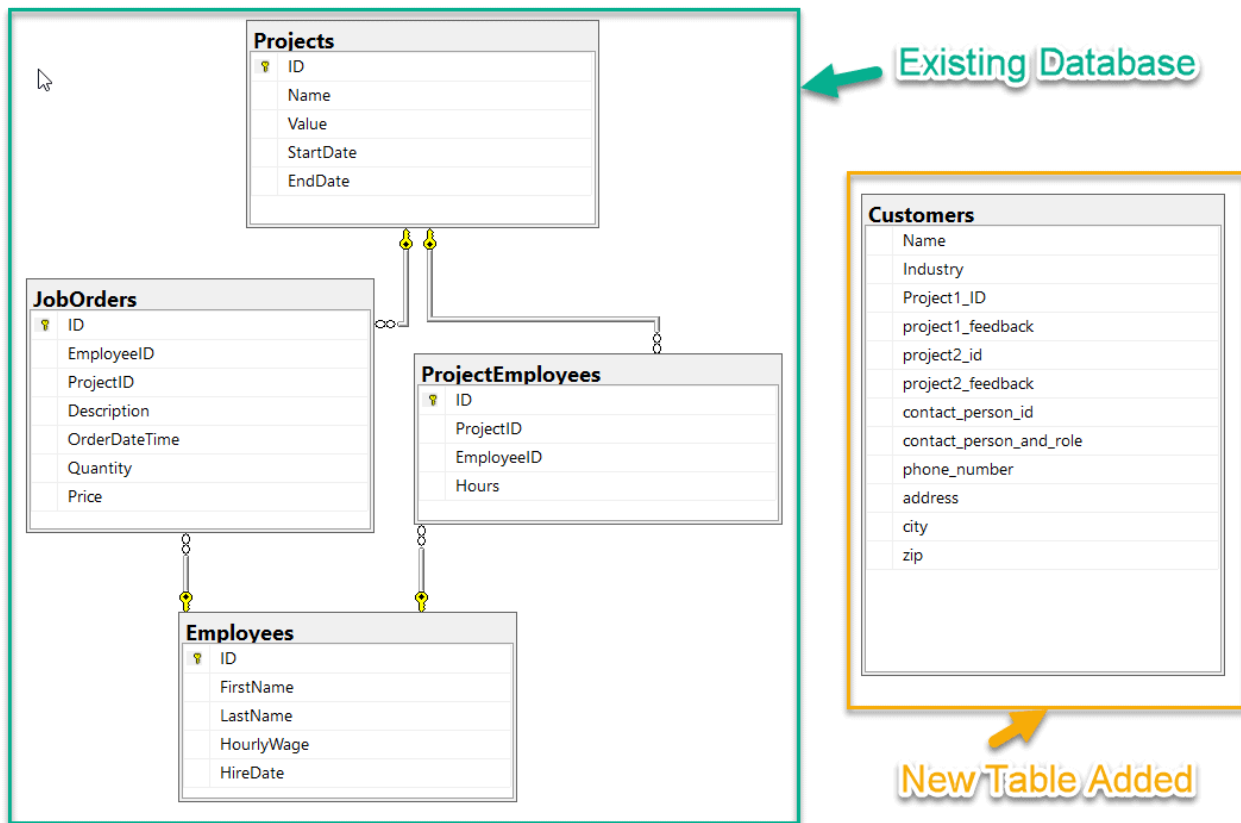
```
SQL> DELETE FROM employee WHERE employ_id=105;
```

PRACTICE QUIZ:

;

NORMALIZATION:

Let us consider the following database schema. As you can see in Fig 1, there are four tables (Existing Database) - Projects, Employees, ProjectEmployees, and JobOrders. Recently, the Customers table has also been added to the database to store the customers' information. As you can see in the diagram below, the Customers table has not been designed in a proper way to support the normal forms, let's go ahead and fix it.



```
CREATE TABLE Projects(  
[ID] INT PRIMARY KEY IDENTITY,  
[Name] VARCHAR(100),  
[Value] DECIMAL(5,2),  
StartDate DATE,  
EndDate DATE  
)  
GO  
CREATE TABLE Employees(  
[ID] INT PRIMARY KEY IDENTITY,
```

```

[FirstName] VARCHAR(50),
[LastName] VARCHAR(50),
[HourlyWage] DECIMAL(5,2),
[HireDate] DATE
)
GO

CREATE TABLE ProjectEmployees(
[ID] INT PRIMARY KEY IDENTITY,
[ProjectID] INT,
[EmployeeID] INT,
[Hours] DECIMAL(5,2),
CONSTRAINT FK_ProjectEmployees_Projects FOREIGN KEY ([ProjectID])
REFERENCES [Projects] ([ID]),
CONSTRAINT FK_ProjectEmployees_Employees FOREIGN KEY ([EmployeeID])
REFERENCES [Employees] ([ID])
)
GO

CREATE TABLE JobOrders(
[ID] INT PRIMARY KEY IDENTITY,
[EmployeeID] INT,
[ProjectID] INT,
[Description] TEXT,
[OrderDateTime] DATETIME,
[Quantity] INT,
[Price] DECIMAL(5,2),
CONSTRAINT FK_JobOrders_Projects FOREIGN KEY ([ProjectID]) REFERENCES
[Projects] ([ID]),
CONSTRAINT FK_JobOrders_Employees FOREIGN KEY ([EmployeeID]) REFERENCES
[Employees] ([ID])
)
GO

CREATE TABLE Customers (
[Name] VARCHAR(100),
[Industry] VARCHAR(100),
[Project1_ID] INT,
[Project1_Feedback] TEXT,
[Project2_ID] INT,

```

```
[Project2_Feedback] TEXT,  
[ContactPersonID] INT,  
[ContactPersonAndRole] VARCHAR(255),  
[PhoneNumber] VARCHAR(12),  
[Address] VARCHAR(255),  
[City] VARCHAR(255),  
[Zip] VARCHAR(5)  
)  
GO
```

OUTPUT:

AIM: PRACTICING DDL COMMANDS

Create a Table:

```
SQL> create table Cancellation (PNR_NO Number(9), No_of_seats Number(8), Address  
varchar(50), Contact_No Number(9), Status char(3));
```

Table created.

```
SQL> desc Cancellation
```

Name	Null?	Type
PNR_NO		NUMBER(9)
NO_OF_SEATS		NUMBER(8)
ADDRESS		VARCHAR2(50)
CONTACT_NO		NUMBER(9)
STATUS		CHAR(3)

Test Output:

Ticket Table:

```
SQL> create table Ticket(Ticket_No number(9) primary key, age number(4), sex char(4)  
Not null, source varchar(2), destination varchar(20), dep_time varchar(4));
```

Table created.

```
SQL> desc Ticket
```

Name	Null?	Type
TICKET_NO	NOT NULL	NUMBER(9)
AGE		NUMBER(4)
SEX	NOT NULL	CHAR(4)
SOURCE		VARCHAR2(2)
DESTINATION		VARCHAR2(20)
DEP_TIME		VARCHAR2(4)

Test Output:

Alteration of Table

Addition of Column(s)

Addition of column in table is done using:

```
SQL> alter table Ticket modify tiketnonumber(10);  
Table altered.
```

Test ouput:

Deletion of Column

Alter table <table_name> drop column <column name>;

```
SQL>Alter Table Emp_master drop column comm;
```

Test output:

Alter table <table_name> set unused column <column name>;

For Example,

```
SQL>Alter Table Emp_master set unused column comm;  
Test output:
```

Alter table <table_name> drop unused columns;

Test output:

Alter table <table_name> drop (Column1, Column2, _);

Test output:

Modification in Column

Modify option is used with Alter table_ when you want to modify any existing column.

Alter table <table name> modify (column1 datatype, _);

.

SQL> Alter table emp_master modify salary number(9,2);

Table altered.

Test output:

Truncate Table

Truncate table <table name> [Reuse Storage];

Example

SQL>Truncate Table Emp_master;

Or

SQL>Truncate Table Emp_master Reuse Storage;

Table truncated.

Test output:

AIM: PRACTICING DML COMMANDS

a) Insert command

Insert into <table name> values (a list of data values);

Insert into <table name>(column list) values(a list of data);

```
SQL> insert into emp_master (empno,ename,salary) values (1122,'Smith',8000);
```

1 row created.

Adding values in a table using Variable method.

```
SQL> insert into Passenger values (&PNR_NO, &TICKET_NO, '&Name', &Age, '&Sex', '&PPNO');
```

Enter value for pnr_no: 1

Enter value for ticket_no: 1

Enter value for name: SACHIN

Enter value for age: 12

Enter value for sex: m

Enter value for ppno: sd1234

```
old 1: insert into Passenger values(&PNR_NO,&TICKET_NO, '&Name', &Age, '&Sex', '&PPNO')
```

```
new 1: insert into Passenger values(1,1,'SACHIN',12,'m','sd1234')
```

1 row created.

```
SQL> /
```

```
SQL> /
```

SQL>/

SQL>/

SQL> insert into Bus values('&Bus_No','&source','&destination');

Enter value for bus_no: 1

Enter value for source: hyd

Enter value for destination: ban

old 1: insert into Bus values('&Bus_No','&source','&destination')

new 1: insert into Bus values('1','hyd','ban')

1 row created.

SQL> /

SQL> /

b) Simple Select Command

Select <column1>,<column2>,...,<column(n)> from <table name>;

SQL> select * from emp_master;

Test Output:

Exercise: Display the all column of University Database of Department.

SQL> select empno, ename, salary from emp_master;

Test Output:

SQL> select * from Passenger;

Test Output:

Exercise: Display the all column of University Database of project table

Distinct Clause

SQL> select distinct deptno from emp_master;

Test Output:

Exercise: Display the all column of University Database of project table by using distinct clause.

Select command with where clause:

Select <column(s)> from <table name> where [condition(s)];

Example

SQL> select empno, ename from emp_master where hiredate = '1-jan- 00';

Test Output:

SQL> update Passenger set age='43' where PNR_NO='2';

Test Output:

SQL>Select*from passenger;

Test Output:

SQL> drop table Cancellation;
Table dropped.

Test Output:

Select command with DDL and DML command.

Example 1:

Table Creation with select statement

create table <table name> as select <columnname(s)> from <existing table name>;

Example 2:

Insert data using Select statement

Syntax:

Inert into <tablename> (select <columns> from <tablename>);

Example 3:

SQL> insert into emp_copy (select * from emp_master);

Test Output:

Example 4:

SQL> insert into emp_copy(nm) (select name from emp_master);
Test Output:

Change Table Name

One can change the existing table name with a new name.

Syntax

Rename <OldName> To <NewName>;

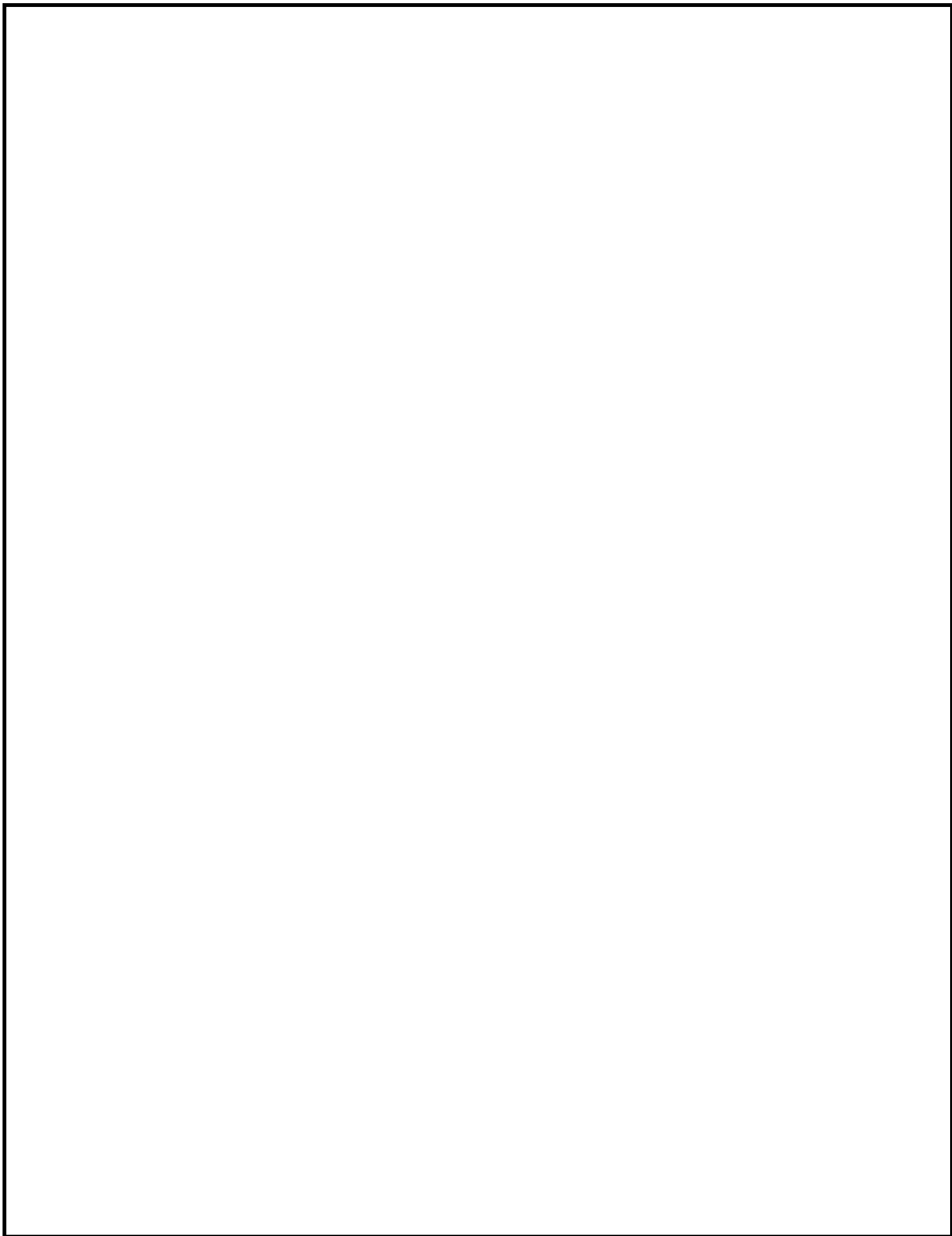
Example:

SQL> Rename emp_master_copy1 To emp_master1;

Table Renamed.

Test Output:

PRACTICE QUERIES:



Aim: Practice queries using ANY, ALL, IN, EXISTS, UNION, INTERSECT, JOIN

SQL> select order_no from order_master;

Test Output:

SQL> select order_no from order_detail;

Test Output:

Example:

```
SQL>select order_no from order_master union select order_no from  
order_detail;
```

Test Output:

Union All :**Example:**

```
SQL> select order_no from order_master union all select order_no from  
order_detail.
```

Test Output:

Intersect :**Example:**

```
SQL> select order_no from order_master intersect select order_no from  
order_detail;
```

Test Output:

Minus :

Example:

SQL> select order_no from order_master minus select order_no from order_detail;

Test Output:

PRACTICE QUERIES ON JOINS:

AIM: Implement Sub Queries:

Subquery

Example:

```
SQL> select * from order_master where order_no = (select order_no from order_detail where  
order_no = 'O001');
```

Test Output:

Example:

```
SQL> select * from order_master where order_no = (select order_no from order_detail);
```

Test Output:

Example:

SQL>Select * from order_master where order_no = any(select order_no from order_detail);

Test Output:

SQL> select * from order_master where order_no in(select order_no from order_detail);

Test Output:

AIM: Practice Queries using Aggregate functions, Group By, Having Clause and Order Clause.

1) Avg (Average): This function will return the average of values of the column specified in the argument of the column.

Example:

```
SQL> select avg(comm) from emp_master;  
Test Output:
```

2) Min (Minimum):

Example:

```
SQL>Select min(salary) from emp_master;  
Test Output:
```

3) Max (Maximum):

Example:

```
SQL>select max(salary) from emp_master;  
Test Output:
```

4) Sum:

Example:

SQL>Select sum(comm) from emp_master;

Test Output:

5) Count:

Syntax: Count(*)

Count(column name)

Count(distinct column name)

Example:

SQL>Select count(*) from emp_master;

Test Output:

Example:

SQL> select count(comm) from emp_master;

Test Output:

Group By Clause

Example:

SQL>select deptno,count(*) from emp_master group by deptno;

Test Output:

Having Clause

Example

SQL> select deptno,count(*) from emp_master group by deptno having Deptno is not null;

Test Output:

Order By Clause

Select<column(s)>from<Table Name>where[condition(s)][order by<column name>[asc /] desc];

Example:

SQL> select empno,ename,salary from emp_master order by salary;

Test Output:

SQL> select empno,ename,salary from emp_master order by salary desc;

Test Output:

SQL *Plus having following operators.

Example

SQL> select salary+comm from emp_master;

Salary+comm

Test Output:

Example:

SQL> select salary+comm net_sal from emp_master;

Test Output:

SQL> Select 12*(salary+comm) annual_netsal from emp_master;

Test Output

Comparison Operators:**Example:**

SQL> select * from emp_master where salary between 5000 and 8000;

Test Output:

IN Operator:

SQL>Select * from emp_master where deptno in(10,30);

Test Output:

LIKE Operator:

SQL>select*From emp_master where job like 'M%';

Test Output:

Logical operator:

SQL>select*From emp_master where job like „_lerk“;

Test Output:

AND Operator:

SQL> select * from emp_master where salary > 5000 and comm < 750 ;

Test Output:

OR Operator:

SQL>select * from emp_master where salary > 5000 or comm < 750;

Test Output:

NOT Operator:

SQL>select*from emp_master where not salary=10000;Test Output:

AIM : Implement Views:

Views

Syntax:Create View <View_Name> As Select statement;

Example:

SQL>Create View EmpView As Select * from Employee;

View created.

Syntax:Select columnname,columnname from <View_Name>;

Example:

SQL>Select Empno,Ename,Salary from EmpView where Deptno in(10,30);

Test Output:

Updatable Views:

Syntax for creating an Updatable View:

Create View Emp_vw As

Select Empno,Ename,Deptno from Employee;

View created.

SQL>Insert into Emp_vw values(1126,'Brijesh',20);

SQL>Update Emp_vw set Deptno=30 where Empno=1125;

1 row updated.

SQL>Delete from Emp_vw where Empno=1122;

View defined from Multiple tables (Which have no Referencing clause):

For insert/modify:

Test Output:

For delete:

Test Output:

View defined from Multiple tables (Which have been created with a Referencing clause):

Syntax for creating a Master/Detail View (Join View):

SQL>Create View EmpDept_Vw As

Select a.Empno,a.Ename,a.Salary,a.Deptno,b.Dname From Employee a,DeptDet b

Where a.Deptno=b.Deptno;

View created.

Test Output:

SQL>Update EmpDept_Vw set salary=4300 where Empno=1125;

Test Output:

SQL>Delete From EmpDept_Vw where Empno=1123;
Test Output:

PRACTICE QUERIES

Aim: Writing triggers

Example

Create or replace trigger upperdname before insert or update

on dept for each row

Test Output:

Example

Create or replace trigger emp_rest before insert or update or delete on

Emp.

Test Output:

Example

Create or replace trigger find_tran before insert or update or delete on dept for each row

Test Output:

Examples:

Create of insert trigger, delete trigger and update trigger.

Test Output:

b) Create Trigger updchek before update on Ticket For Each Row

Test Output:

b) CREATE OR RELPLACE TRIGGER trig1 before insert on Passenger for each
row

Test Output:

AIM : Implement Cursors:

Example

Aim: Implement the %notfound Attribute Write a cursor by using the %notfound Attribute

**Aim; Implement the %found Attribute
Write a cursor program by using The % found Attribute**

Aim: Implement The %rowCount Attribute:

Write a cursor program by using the %rowCount Attribute:

Test Output:

