Malla Reddy College Engineering (Autonomous)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad, Telangana-500100 www.mrec.ac.in

Department of Information Technology

II B. TECH II SEM (A.Y.2018-19)

80515 Database Management Systems Lab

2018-19 Onwards (MR-18)	MALLA REDDY ENGINEERING COLLEGE (Autonomous)	B.Tech. IV Semester		
Code: 80515	**DATABASE MANAGEMENT SYSTEMS LAB (Common for CSE and IT)**	**L**	**T**	**P**
Credits: 2		**-**	**1**	**2**

Prerequisite: NIL

Course Objectives:

This course enable the students to learn and understand the fundamentals of data models and conceptualize and depict a database system using ER diagram, learn about SQL and relational database design, build the databases using normalization techniques, study the basic issues of transaction processing and concurrency control and to explore the concepts of file organization techniques

Software Requirements: Mysql 5.6.10

List of Programs:

Roadway Travels: "Roadway Travels" is in business since 1997 with several buses connecting different places in India. Its main office is located in Hyderabad.

The company wants to computerize its operations in the following area

- Reservations and Ticketing
- Cancellations

Reservations & Cancellation:

Reservations are directly handled by booking office. Reservations can be made 30 days in advance and tickets issued to passenger. One passenger/ person can book many tickets (to his/her family). Cancellations are also directly handed at the booking office.

In the process of Computerization of Roadway Travels you have to design and develop a Database which consists the data of Buses, Passengers, Tickets and Reservation and cancellation details. You should also develop query‘s using SQL to retrieve the data from the database.

The above process involves many steps like 1. Analyzing the problem and identifying the

Entities and Relationships 2. E-R Model 3. Relational Model 4. Normalization 5. Creating the database 6. Querying. Students are supposed to work on these steps week wise and finally create a complete ―Database system to Roadway Travels. Examples are given at every experiment for guidance to students.

1: E-R Model

Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any.

Example: **Entities:** 1. BUS 2. Ticket 3. Passenger

Relationships: 1. Reservation 2. Cancellation

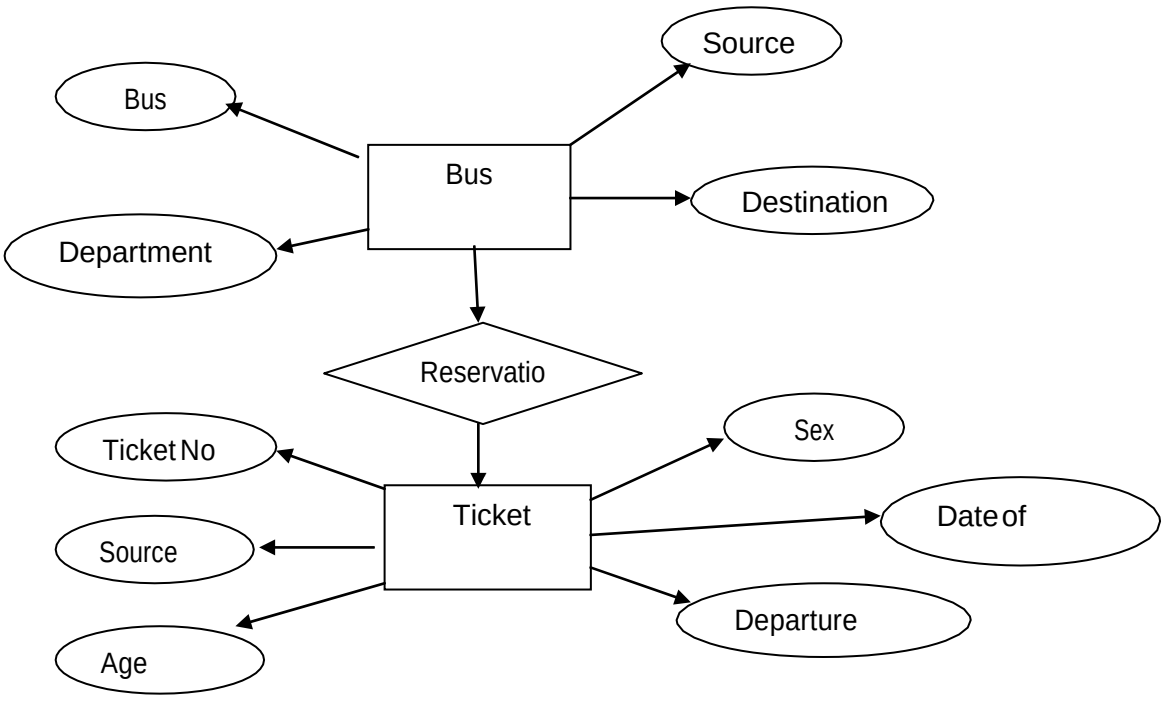
PRIMARY KEY ATTRIBUTES: Ticket ID (Ticket Entity) ;Passport ID (Passenger Entity) ; Bus_NO (Bus Entity)

Apart from the above mentioned entities you can identify more. The above mentioned are few.

Note: The student is required to submit a document by writing the Entities andKeys to the lab teacher.

2: Concept design with E-R Model

Relate the entities appropriately. Apply cardinalities for each relationship. Identify strong entities and weak entities (if any). Indicate the type of relationships (total / partial). Try to incorporate generalization, aggregation, specialization etc wherever required.



Note: The student is required to submit a document by drawing the E-RDiagram to the lab teacher.

3: Relational Model

Represent all the entities (Strong, Weak) in tabular fashion. Represent relationships in a tabular fashion. There are different ways of representing relationships as tables based on the cardinality. Represent attributes as columns in tables or as tables based on the requirement. Different types of attributes (Composite, Multi-valued, and Derived) have different way of representation.

Example: E-R diagram for bus

Example: The passenger tables look as below. This is an example. You can add more attributes based on your E-R model. This is not a normalized table.

Passenger

Name	Age	Sex	Address	Passport Id

Note: The student is required to submit a document by Represent relationshipsin a tabular fashion to the lab teacher.

4: Normalization

Database normalization is a technique for designing relational database tables to minimize duplication of information and, in so doing, to safeguard the database against certain types of logical or structural problems, namely data anomalies.

For example, when multiple instances of a given piece of information occur in a table, the possibility exists that these instances will not be kept consistent when the data within the table is updated, leading to a loss of data integrity.

Passportid	Ticketid

A table that is sufficiently normalized is less vulnerable to problems of this kind, because its structure reflects the basic assumptions for when multiple instances of the same information should be represented by a single instance only.

For the above table in the First normalization we can remove the multi valued attribute. Ticket_id and place it in another table along with the primary key of passenger.

First Normal Form: The above table can be divided into two tables as shown below.

Passenger

Name	Age	Sex	Address	Passport ID

You can do the second and third normal forms if required. Anyhow Normalized tables are given at the end.

5: Installation of Mysql and practicing DDL commands

Installation of MySql. In this week you will learn Creating databases, How to create

tables, altering the database, dropping tables and databases If not required. You will also try truncate, rename commands etc.

Example for creation of a normalized "Passenger" table.

CREATE TABLE Passenger (Passport_id INTEGER PRIMARY KEY, Name VARCHAR (50) Not NULL, Age Integer Not NULL, Sex Char, Address VARCHAR (50) Not NULL);

Similarly create all other tables.

***Note: Detailed creation of tables is given at the end*.**

6: Practicing DML commands

DML commands are used to for managing data within schema objects. Some examples:

SELECT - retrieve data from the a database INSERT - insert
data into a table

UPDATE - updates existing data within a table

DELETE - deletes all records from a table, the space for the records remain

Inserting values into Bus table:

Insert into Bus values (1234, 'hyderabad', _tirupathi');
Insert into Bus values (2345, 'hyderabd', 'Banglore');
Insert into Bus values (23, _hyderabad', _Kolkata');
Insert into Bus values (45, _Tirupathi', _Bangalore');
Insert into Bus values (34, _hyderabad', _Chennai');

Inserting values into Bus table:

Insert into Passenger values (1, 45,'ramesh', 45,'M','abc123'); Insert into Passenger values (2, 78,'geetha', 36,'F','abc124'); Insert into Passenger values (45, 90, _ram', 30, _M', _abc12'); Insert into Passenger values (67, 89, _ravi', 50, _M', _abc14'); Insert into Passenger values (56, 22, _seetha', 32, _F', _abc55');

Few more Examples of DML commands:

Select * from Bus; (selects all the attributes and display)
UPDATE BUS SET Bus No = 1 WHERE BUS NO=2;

7: Querying

In this week you are going to practice queries (along with sub queries) using ANY, ALL, IN, Exists, NOT EXISTS, UNION, INTERSECT, Constraints etc.

Practice the following Queries:

1. Display unique PNR_no of all passengers.
2. Display all the names of male passengers.
3. Display the ticket numbers and names of all the passengers.
4. Display the source and destination having journey time more than 10 hours.
5. Find the ticket numbers of the passengers whose name start with 'A'and ends with 'H'.
6. Find the names of passengers whose age is between 30 and 45.
7. Display all the passengers names beginning with 'A'
8. Display the sorted list of passengers names

8 and 9: Querying (continued…)

You are going to practice queries using Aggregate functions (COUNT, SUM, AVG, and

MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views.

Write a Query to display the Information present in the Passenger and cancellation tables.
Hint: Use UNION Operator.

Write a Query to display different travelling options available in British Airways.
Displaythe number of days in a week on which the 9W01 bus is available.

Find number of tickets booked for each PNR_no using GROUP BY CLAUSE. **Hint:** Use GROUP BY on PNR_No.

Find the distinct PNR numbers that are present.

Find the number of tickets booked in each class where the number of seats is greater than 1. **Hint:** Use GROUP BY, WHERE and HAVING CLAUSES.

Find the total number of cancelled seats.

10: Triggers
In this week you are going to work on Triggers. Creation of insert trigger, delete trigger, update trigger. Practice triggers using the above database.
Eg: CREATE TRIGGER updcheck BEFORE UPDATE ON passenger FOR
EACH ROW
BEGIN
IF NEW.TickentNO > 60 THEN SET New.Tickent no
= Ticket no;
ELSE
SET New.Ticketno = 0; END IF;
END;

11: Procedures
In this session you are going to learn Creation of stored procedure, Execution of procedure and modification of procedure. Practice procedures using the above database.
Eg:CREATE PROCEDURE myProc ()
BEGINage>=40; End;

12: Cursors
In this week you need to do the following: Declare a cursor that defines a result set. Open the cursor to establish the result set. Fetch the data into local variables as needed from the cursor, one row at a time. Close the cursor when done
CREATE PROCEDURE myProc (in_customer_id INT) BEGIN
DECLARE v_id INT;
DECLARE c1 CURSOR FOR SELECT stdId, stdFirstname FROM students WHERE stdId=in_customer_id;
OPEN c1;
FETCH c1 into v_id, v_name; Close c1;

END;

Tables
BUS
Bus No: Varchar: PK (Public key) Source: Varchar Destination: Varchar

Passenger
PPNO: Varchar(15) : PK Name: Varchar(15) Age : int (4) Sex:Char(10) : Male / Female Address: VarChar(20)

Passenger_Tickets
PPNO: Varchar(15): PK Ticker_No: Numeric(9)

Reservation
PNR_No: Numeric(9) : FK Journey_date : datetime(8) No_of_seats : int (8) Address: Varchar (50) Contact_No: Numeric (9) -->should not be less than 9 and should not accept any other character other than Integer Status: Char (2): Yes / No

Cancellation
PNR_No: Numeric(9) : FK Journey_date : datetime(8) No_of_seats : int (8) Address : Varchar (50) Contact_No: Numeric (9) --> Should not be less than 9 and Should not accept any other character other than Integer Status: Char (2) : Yes / No

Ticket
Ticket_No: Numeric (9): PK Journey_date: datetime(8) Age : int (4) Sex:Char(10) : Male / Female Source : Varchar Destination : Varchar Dep_time : Varchar

TEXT BOOKS

1. Rick F.Vander Lans, "**Introduction to SQL**", Pearson education.
2. B.Rosenzweig and E.Silvestrova,"**Oracle PL/SQL**", Pearson education.

REFERENCES

1. M.Mc Laughlin,"**Oracle Database 11g PL/SQL Programming**", TMH.
2. J.J.Patrick,"**SQL Fundamentals**", Pearson Education
3. Steven Feuerstein,"**Oracle PL/SQL Programming**", SPD.
4. Dr.P.S.Deshpande, "**SQL & PL/SQL for Oracle 10g**", Black Book, Dream Tech.

Course Outcomes:
At the end of the course, students will be able to
1. **Design** and implement a database schema for a given problem.
2. **Generate** queries on a database using SQL commands.
3. **Declare** and enforce integrity constraints on a database using a state-of-the-art RDBMS.
4. **Make** use of procedures for data accessing and manipulations.

CO- PO Mapping **(3/2/1 indicates strength of correlation) 3-Strong, 2-Medium, 1-Weak**															
COs	**Programme Outcomes(POs)**												**PSOs**		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3			2				3		2	3	3	
CO2	3	3	3			3				3		3	3	3	
CO3	3	3	3			2				3		2	3	3	
CO4	3	3	3			3				3		3	3	3	

Objectives:

Students will have the ability to:

- Keep abreast of current developments to continue their own professional development.
- To engage themselves in lifelong learning of Database management systems theories and technologies this enables them to purse higher studies.
- To interact professionally with colleagues or clients located abroad and the ability to overcome challenges that arises from geographic distance, cultural differences, and multiple languages in the context of computing.
- Develop team spirit, effective work habits, and professional attitude in written and oral forms, towards the development of database applications

Outcomes:

Students will be able to demonstrate their skills

- In drawing the ER, EER, and UML Diagrams.
- In analyzing the business requirements and producing a viable model for the implementation of the database.
- In converting the entity-relationship diagrams into relational tables.
- To develop appropriate Databases to a given problem that integrates ethical, social, legal, and economic concerns.

INDEX

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 S ystem. 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

Roll_no	Sname	S_Address
1	Rahul	Satelite
2	Sachin	Ambawadi
3	Saurav	Naranpura

Student Marksheet Table

Rollno	Sub1	Sub2	Sub3
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 is 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.

ROADWAY TRAVELS

"Roadway Travels" is in business since 1977 with several buses connecting different places in India. Its main office is located in Hyderabad.

The company wants to computerize its operations in the following areas:

Reservations
Ticketing

Cancellations

Reservations :

Reservations are directly handeled by booking office.reservations can be made 60 days in advance in either cash or credit. In case the ticket is not available,a wait listed ticket is issued to the customer. This ticket is confirmed against the cancellation.

Cancellation and modification:

Cancellations are also directly handed at the booking office. Cancellation charges will be charged.

Wait listed tickets that do not get confirmed are fully refunded.

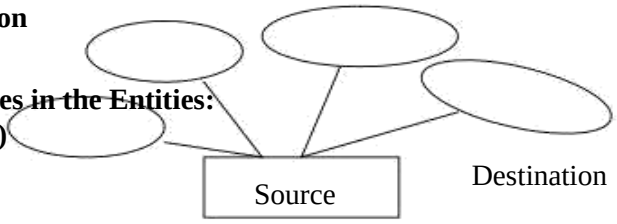
AIM: Analyze the problem and come with the entities in it. Identify what Data has to be persisted in the databases.

The Following are the entities:

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

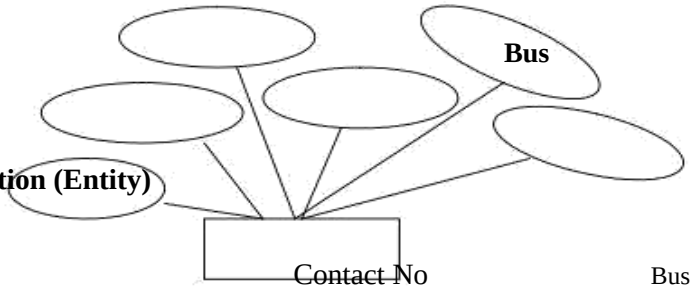
The attributes in the Entities:

Bus:(Entity)



Couch Type

Bus No



Reservation (Entity)

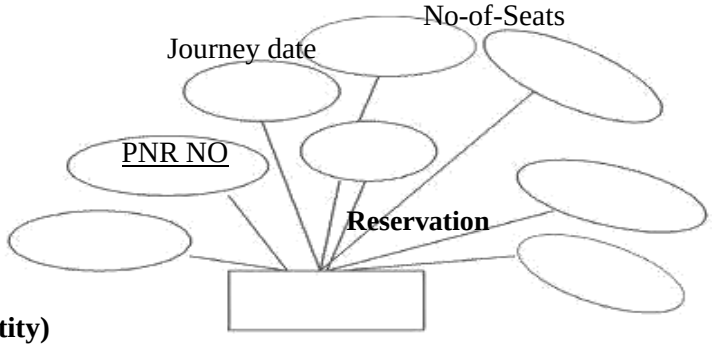
Contact No

Bus No

No-of-Seats

Journey date

Address



Ticket :(Entity)

Dep- Time

Source

Age

Sex

Journey date

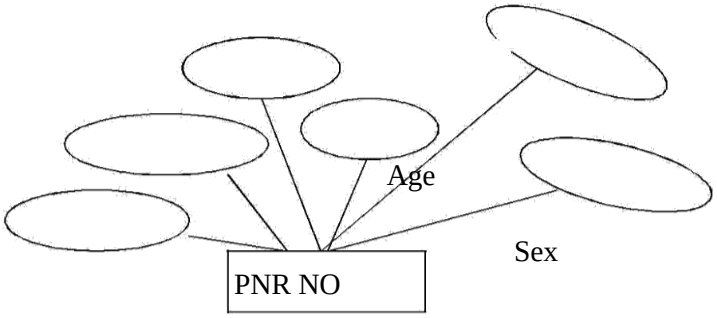
Destination

Ticket No

Bus No

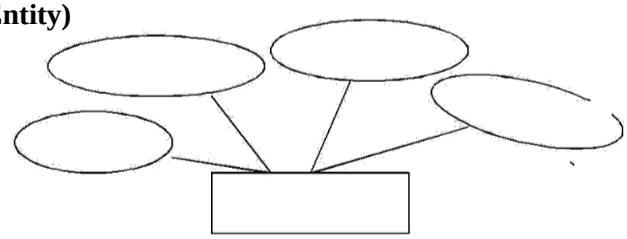
Ticket

Passenger:



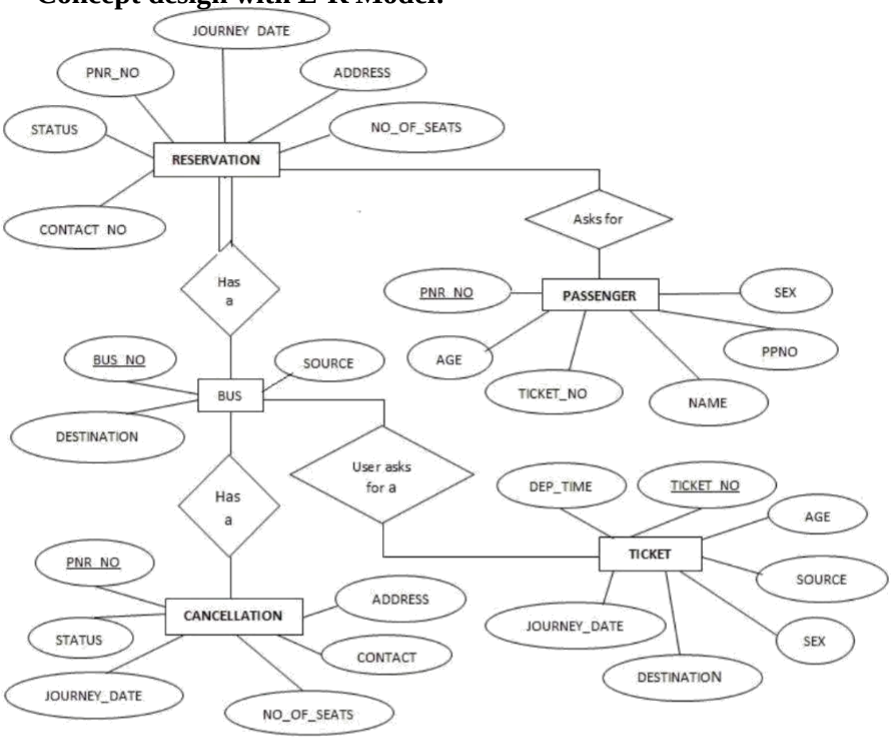
Passenger

Cancellation (Entity)



Cancellation

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

Before we take a look on above-mentioned commands we will see the data types available in Oracle.

Oracle Internal Data types

When you create a table in Oracle, a few items should be important, not only do you have to give each table a name(e.g. employee, customer), you must also list all the columns or fields (e.g. First_name, Mname, Last_name) associated with the table. You also have to specify what type of information thattable will hold to the database. For example, the column Empno holds numeric information. An Oracle database can hold many different types of data.

Data type Description

Char(Size) Stores fixed-length character data to store alphanumeric values, with a maximum size of 2000 bytes. Default and minimum size is 1 byte.

Varchar2(Size) Stores variable-length character data to store alphanumeric values, with maximum size of 4000 bytes.

char(Size) Stores fixed-length character data of length size characters or bytes, depending on the choice of national character set. Maximum size if determined by the number of bytes required storing each character with an upper limit of 2000 bytes. Default and minimum size is 1 character or 1 byte, depending on the character set.

Nvarchar2(Size) Stores variable-length character string having maximum length size characters or bytes, depending on the choice of national character set. Maximum size is determined by the number of bytes required to store each character, with an upper limit of 4000 bytes.

Long Stores variable-length character data up to 2GB(Gigabytes). Its lenth would be restricted based on memory space available in the computer.

Number [p,s] Number having precision p and scale s. The precision p indicates total number of digit varies from 1 to 38. The scale s indicates number of digit in fraction part varies from -84 to 127.

Date Stores dates from January 1, 4712 B.C. to December 31, 4712 A.D. Oracle predefine format of Date data type is DD-MON-YYYY.

Raw (Size) Stores binary data of length size. Maximum size is 2000 bytes. One must have to specify size with RAW type data, because by default it does not specify any size.

Long Raw Store binary data of variable length up to 2GB(Gigabytes). LOBS - LARGE OBJECTS

LOB is use to store unstructured information such as sound and video clips, pictures upto 4 GB size.

CLOB A Character Large Object containing fixed-width multi-byte characters. Varying-

width character sets are not supported. Maximum size is 4GB.

NCLOB A National Character Large Object containing fixed-width multi-byte characters.

Varying-width character sets are not supported. Maximum size is 4GB. Stores national character set data.

BLOB To store a Binary Large Object such a graphics, video clips and sound files. Maximum size is 4GB.

BFILE Contains a locator to a large Binary File stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4GB.Apart from oracle internal data types, user can create their own data type, which is used in database and other database object. We will discuss it in the later part.

The following are tabular representation of the above entities and relationships

BUS:

COLOUMN NAME	**DATA TYPE**	**CONSTRAINT**
Bus No	varchar2(10)	**Primary Key**
Source	varchar2(20)	
Destination	varchar2(20)	
Couch Type	varchar2(20)	

Reservation:

COLOUMN NAME	**DATA TYPE**	**CONSTRAINT**
PNRNo	number(9)	**Primary Key**
Journey date	Date	
No-of-seats	integer(8)	
Address	varchar2(50)	
Contact No	Number(9)	Should be equal to 10 numbers and not allow other than numeric
BusNo	varchar2(10)	**Foreign key**
Seat no	Number	

Ticket:

COLOUMN NAME	**DATA TYPE**	**CONSTRAINT**
Ticket_No	number(9)	**Primary Key**
Journey date	Date	
Age	int(4)	
Sex	Char(10)	
Source	varchar2(10)	
Destination	varchar2(10)	
Dep-time	varchar2(10)	
Bus No	Number2(10)	

Passenger:

COLOUMN NAME	**DATA TYPE**	**CONSTRAINT**
PNR No	Number(9)	**Primary Key**
Ticket No	Number(9)	Foreign key
Name	varchar2(15)	
Age	integer(4)	
Sex	char(10)	(Male/Female)
Contact no	Number(9)	Should be equal to 10 numbers and not allow other than numeric

Cancellation:

COLOUMN NAME	**DATA TYPE**	**CONSTRAINT**
PNR No	Number(9)	Foriegn-key
Journey-date	Date	
Seat no	Integer(9)	
Contact_No	Number(9)	Should be equal to 10 numbers and not allow other than numeric

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

1. Steps for installing MySQL

Step1 **1**

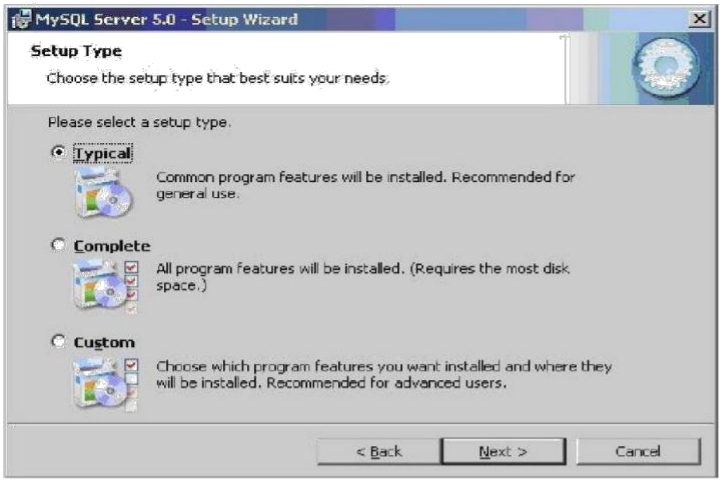
Make sure you already downloaded the **MySQL essential 5.0.45 win32.msi file**. Double click on the .msi file.

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.**

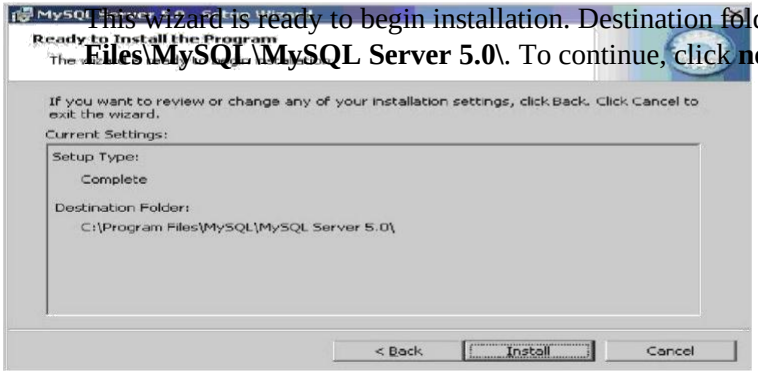
Step3 **3**

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**.



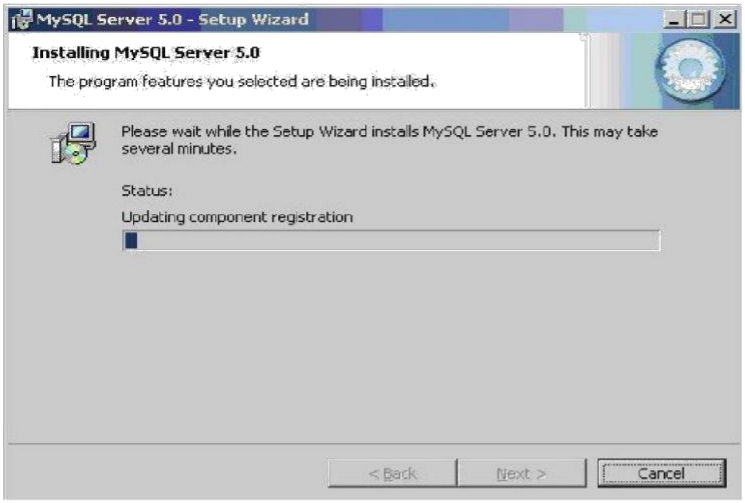
Step4 **4**

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.

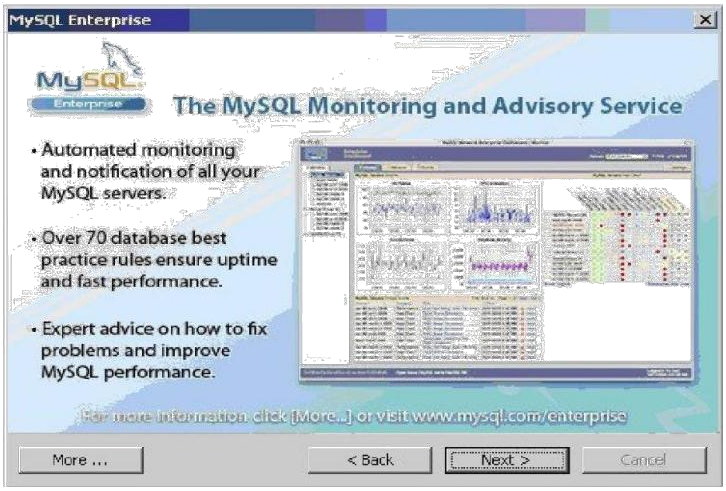


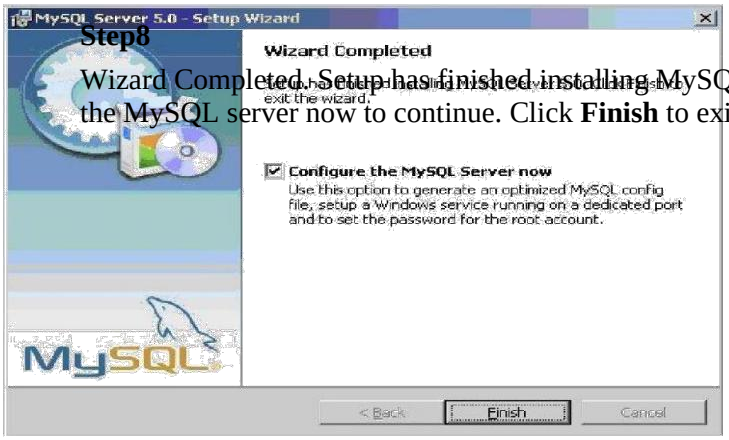
Step6

To continue, click **next.**

Step7 **7**

To continue, click **next.**





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

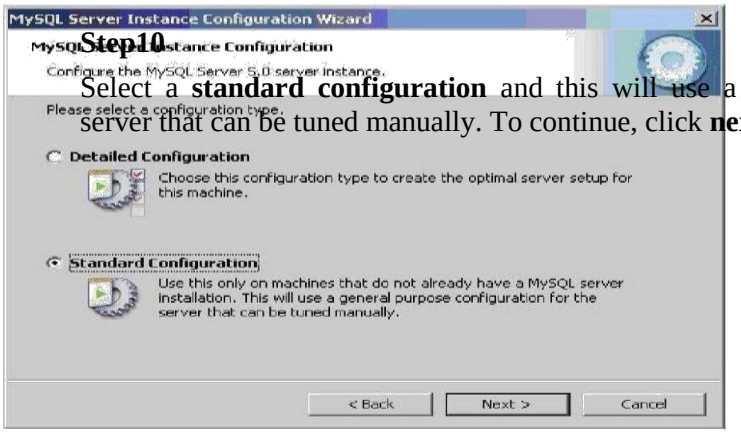
d.

Step9 **9**

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

To continue, click **next**.

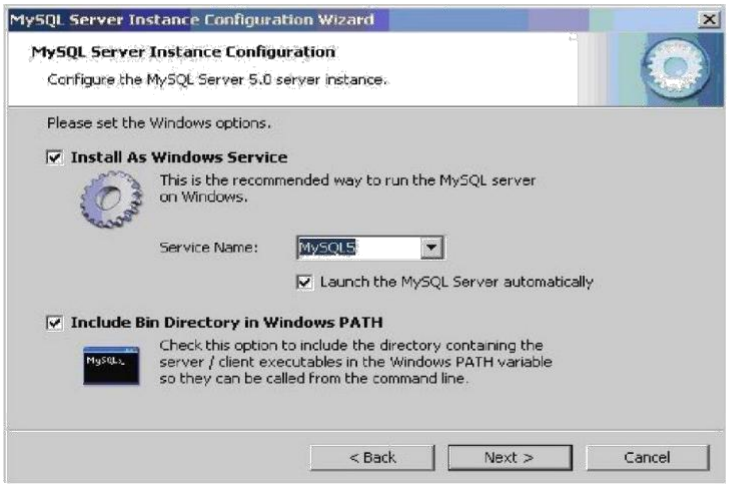
Step10 **10**



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 **11**

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

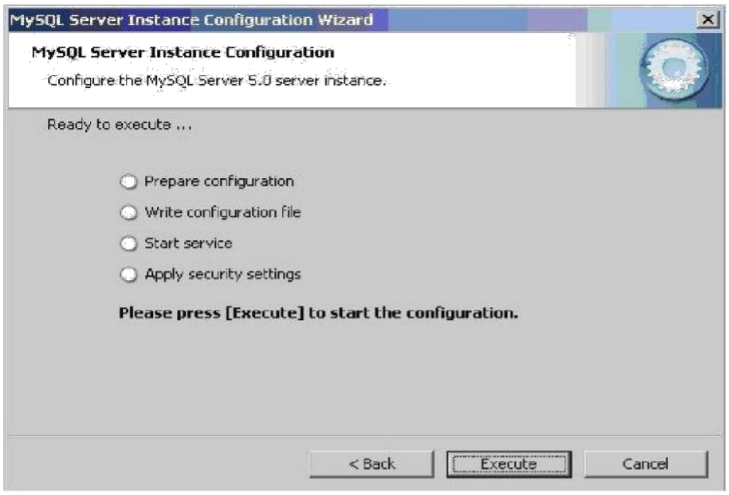


Step12 **12**

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

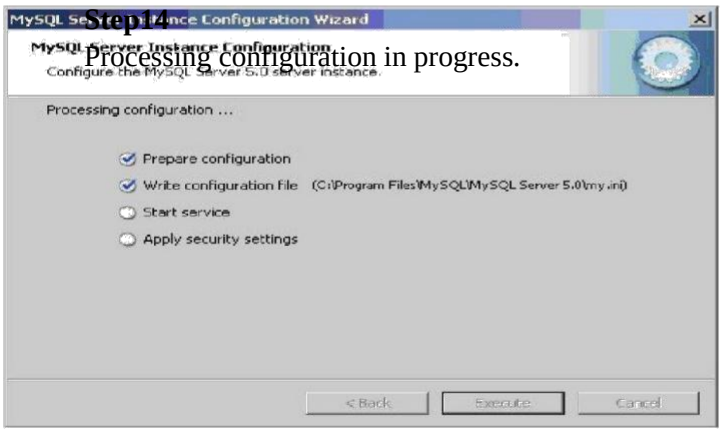
Step13 **13**

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



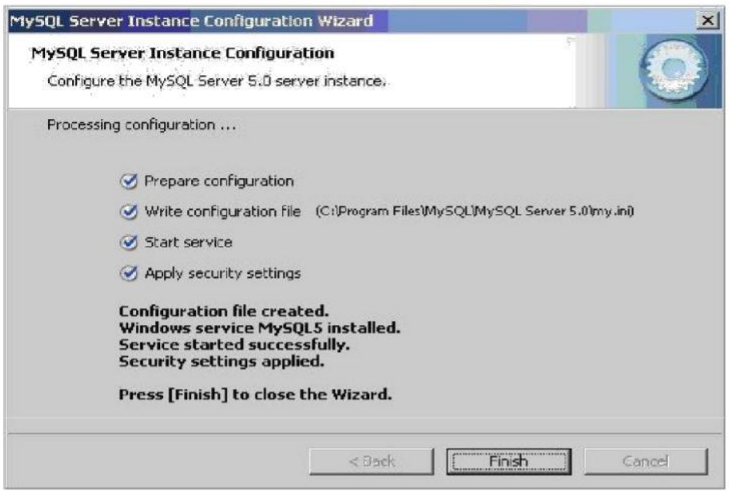
Step14 **14**

Processing configuration in progress.



Step15 **15**

Configuration file created. Windows service MySQL 5 installed. Press **finish** to close the wizard.



2. Practicing DDL & DML Commands

Data Definition Language

The data definition language is used to create an object, alter the structure of an object and also drop already created object. The Data Definition Languages used for table definition can be classified into following:

- Create table command
- Alter table command
- Truncate table command
- Drop table command

Creating of Tables on ROAD WAY TRAVELS:

Table is a primary object of database, used to store data in form of rows and columns.

It is created using following command:

Create Table <table_name> (column1 datatype(size), column2 datatype(size),column(n) datatype(size));

Example:

SQL> create table Bus(Bus_No varchar(5), source varchar(20), destination varchar(20),CouchType varchar2(10),fair number);

Table Created.

create table for the object-relation feature we will discuss it afterwards.

Desc command

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

Desc <table name>

SQL> desc bus;

Name	Null?	Type
-	NOT NULL	INTEGER2(5)
BUS_NO		VARCHAR2(20)
SOURCE		VARCHAR2(20)
DESTINATION		VARCHAR2(10)
COUCH TYPE		
FAIR	NUMBER	

SQL> Describe the university database

Test Outputs:

Signature of the lab In charge

Date:

Extra:1.Create a table Universities DB

Reservation Table:

SQL> create table Reservation(PNR_NO Numeric(9), No_of_seats Number(8), Address varchar(50), Contact_No Numeric(9), Status char(3)); Table created.

SQL> desc Reservation

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

Test Output:

Signature of the lab incharge
Date:

Exercise: Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budge

Cancellation Table:

```
SQL> create table Cancellation(PNR_NO Numeric(9), No_of_seats Number(8),
Address varchar(50), Contact_No Numeric(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:

Signature of the lab incharge
Date:

Exercise: Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.)

Assignment Evaluation

Signature

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Ticket Table:

```
SQL> create table Ticket(Ticket_No Numeric(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:

Signature of the lab incharge

Date:

Exercise: Each project is managed as the project's principal investigator).

\

Alteration of Table

Addition of Column(s)

Addition of column in table is done using:

Alter table <table_name> add(column1 datatype, column2 datatype _);
SQL> ALTER TABLE Passenger ADD FOREIGN KEY (PNR_NO)
REFERENCES Reservation(PNR_NO);
Table altered.

SQL> ALTER TABLE Cancellation ADD FOREIGN KEY (PNR_NO) REFERENCES Reservation(PNR_NO);
Table altered.
Test output:

Signature of lab incharge
Date:

SQL> alter table Ticket modify tiketnonumber(10);
Table altered.
Test ouput:

Signature of lab incharge
Date:

Deletion of Column

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

SQL>Alter Table Emp_master drop column comm;

Test output:

Signature of the lab incharge
Date:

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

For Example,

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

Signature of the lab incharge
Date:

Alter table <table_name> drop unused columns;

Test output:

Signature of the lab incharge
Date:

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

Test output:

Signature of the lab incharge
Date:

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:

Assignment Evaluation **Signature**

0: Not Done [ ] 1: Incomplete [ ] 2: Late complete [ ]

3: Needs improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Signature of the instructor [ ] Date: [\ \ ]

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:

Signature of lab incharge
Date:

AIM: Applying Constraints on Road Way Travels Tables.

Constraints

Domain Integrit y constraints
Entity Integrity constraints
Referential Integrity constraint

Oracle allows programmers to define constraints

Column Level

Table Level

Example

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

Table created.

Test Output:

Signature of the Incharge
Date:

Exercise: Create table professor by using not null constaint

Check Constraint

SQL> create table Reservation(PNR_NO Numeric(9), No_of_seats Number(8), Address varchar(50), Contact_No Numeric(10) constraint ck check(length(contact_no)=10), Status char(3));

Table created.
Test Output:

Signature of the incharge
Date:

Exercise: Create table manage by using check constraints

Check constraint with alter command

SQL> alter table Ticket add constraint check_age check(age>18);

Table altered.

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Entity Integrity Constraints

This type of constraints are further classified into

Unique Constraint

Primary Key Constraint

Unique Constraint

Example:

SQL> create table Ticket(Ticket_No Numeric(9) unique, age number(4), sex char(4) l, source varchar(2), destination varchar(20), dep_time varchar(4));

Test Output:

Signature of the incharge
Date:

Exercise: Create table Professor by using unique

constraint **Unique constraint with alter command**

Example:

SQL> Alter table ticket add constraint uni1 Unique (ticket_no);
Table Altered.
Test Output:

Signature of the incharge
Date:

Exercise: Alter table professor constraint

Primary key constraint at the column level

Example:

SQL> create table Ticket(Ticket_No Numeric(9) constraint pk primary key, age number(4), sex char(4) l, source varchar(2), destination varchar(20), dep_time varchar(4));

Table created.
Test Output:

Signature of the incharge
Date:

SQL> insert into ticket values (1001,26, 'M', 'KPHB','MTM','20:00');

1 row created.

Test Outpu:

Signature of the incharge

Date:

Exercise: Apply primary key constraint on professor table SSN column.

Example:

SQL> create table vendor_master (ven_code varchar(5), ven_name varchar(20), venadd1 varchar(15), venadd2 varchar(15),vencity varchar(15), constraint pr_com primary key (ven_code,ven_name));

Table created.
Test Output:

Signature of the incharge
Date:

Primary key with alter command:

SQL> alter table bus add constraint pr primary key (busno);

Table altered.
C

Exercise: Apply primary key with alter command on project number of University Database

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Referential Integrity Constraint
References constraint defined at column level

Example:

SQL> create table Passenger(PNR_NO Numeric(9) references reservation , Ticket_NO Numeric(9) references ticket, Name varchar(20), Age Number(4), Sex char(10), PPNO varchar(15));

Table created.
Test Output:

Signature of the incharge
Date:

Exercise: Apply References constraint University Database Department table.

Foreign Key Constraint with alter command

SQL> alter table reservation add constraint fk_icode foreign key (busno) references

bus(bus_no);

Table altered.
Test Output:

Signature of the incharge
Date:

Exercise: Apply Foregin Key Constriant with alter on professor SSN with Department No

Remember that when we add constraint at table level foreign key keyword is must.

SQL> delete from bus where bus_no = 2011;
Test output:

Signature of the lab incharge
Date:

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> /

SQL> /

SQL> /

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

☐ \ \ ☐

Signature of the instructor Date:

b) Simple Select Command

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

SQL> select * from emp_master;
Test Output:

Signature of the incharge
Date:

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

SQL> select empno, ename, salary from emp_master;
Test Output:

Signature of the incharge
Date:

SQL> select * from Passenger;
Test Output:

Signature of the incharge
Date:

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

SQL> select distinct deptno from emp_master;
Test Output:

Signature of the incharge
Date:

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:

Signature of the incharge
Date:

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

Signature of the incharge
Date:

SQL>Select*from passenger;
Test Output:

Assignment Evaluation **Signature**

0: Not Done [ ] 1: Incomplete [ ] 2: Late complete [ ]

3: Needs improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Signature of the instructor Date:

DROP Table

SQL> drop table Cancellation;
Table dropped.
Test Output:

Signature of the incharge
Date:

Select command with DDL and DML command.

Table Creation with select statement

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

Example

Insert data using Select statement

Syntax:

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

Example

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

Test Output:

Signature of the incharge
Date:

Example

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

Signature of the incharge
Date:

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:

Assignment Evaluation **Signature**

0: Not Done | 1: Incomplete | 2: Late complete

3: Needs improvement | 4: Complete | 5: Well Done

\ \

Signature of the instructor Date:

Aim: **Practice queries using ANY, ALL, IN, EXISTS, UNION, INTERSECT**
Union: The union operator returns all distinct rows selected by two or more queries.

SQL> select order_no from order_master;

Test Output:

Signature of the incharge
Date:

SQL> select order_no from order_detail;
Test Output:

Signature of the incharge
Date:

Example:

SQL>select order_no from order_master union select order_no from

order_detail;

Test Output:

Signature of the incharge
Date:

Union All :

Example:

SQL> select order_no from order_master union all select order_no from

order_detail.

Test Output:

Signature of the incharge
Date:

Intersect :

Example:

SQL> select order_no from order_master intersect select

order_no from order_detail;

Test Output:

Signature of the incharge
Date:

Minus :
Example:

SQL> select order_no from order_master minus select order_no from order_detail;
Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

EXAMPLE QUERIES:

1. Display Unique PNR_NO of all Passengers

SQL> select PNR_NO from
passenger;

Test Output:

Signature of the incharge
Date:

2. Display all the names of male Passengers

SQL> select Name from Passenger where Sex='m';

Test Output:

Signature of the incharge
Date:

3. Display Ticket numbers and names of all Passengers

SQL> select Ticket_NO,Name from Passenger;

Test Output:

Signature of the incharge
Date:

4. Display the source and destination having journey time more than 10 hours.

SQL> select source, destination from Ticket where Journey_Dur>10;

Test Output:

Signature of the incharge
Date:

5. Find the ticket number of passenger whose name starts with ‘S’ and ends with ‘H’

SQL> select Ticket_NO from Passenger where Name like'S%'and name like'%N'; Test Output:

Signature of the incharge
Date:

6. Find the names of the passenger whose age is between 20 and 40

SQL> select Name from Passenger where age between 20 and 40;

Test Output:

Signature of the incharge
Date:

7. Display all the name of the passengers beginning with ‘r’

SQL> select Name from Passenger where Name like 'r%';
Test Output:

Signature of the incharge
Date:

8. Display the sorted list of Passenger Names

SQL> select Name from Passenger ORDER BY Name;

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

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:

Signature of the incharge
Date:

2) Min (Minimum):
Example:

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

Signature of the incharge
Date:

3) Max (Maximum):

Example:

SQL>select max(salary) from emp_master;

Test Output:

Signature of the incharge
Date:

4) Sum:

Example:

SQL>Select sum(comm) from emp_master;
Test Output:

Signature of the incharge
Date:

5) Count: Syntax:

Count(*) Count(column

name) Count(distinct

column name

Example:

SQL>Select count(*) from emp_master;
Test Output:

Signature of the incharge
Date:

Example:
SQL> select count(comm) from emp_master;
Test Output:

Signature of the incharge
Date:

.

Example:
SQL>Select count(distinct deptno) from emp_master;

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: ☐ \ ☐ \

Group By Clause
Example:

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

Test Output:

Signature of the incharge
Date:

Having Clause
Example

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

Test Output:

Signature of the incharge
Date:

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:

Signature of the incharge
Date:

.

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

Test Output:

Signature of the incharge
Date:

SQL *Plus having following operators.
Example

SQL> select salary+comm from emp_master;
Salary+comm
Test Output:

Signature of the incharge
Date:

Example:

SQL> select salary+comm net_sal from emp_master;

Test Output:

Signature of the incharge
Date:

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

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Comparison Operators:

Example:

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

Test Output:

Signature of the incharge
Date:

IN Operator:

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

Test Output:

Signature of the incharge
Date:

LIKE Operator:

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

Test Output:

Signature of the incharge
Date:

Logical operator:

SQL>select*From emp_master where job like „_lerk‟;
Test Output:

Signature of the incharge
Date:

AND Operator:
SQL> select * from emp_master where salary > 5000 and comm < 750 ;
Test Output:

Signature of the incharge
Date:

OR Operator:

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

Test Output:

Signature of the incharge
Date:

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

Signature of the incharge
Date:

The Oracle Table Dual"
SQL> select 2*2 from dual;
Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Single Row Functions (Scalar Functions):

String Functions:

1) Initcap (Initial Capital): This String function is used to capitalize first character of the input string.

Syntax: initcap(string)

Example:

SQL> select initcap('azure') from dual;

Test Output:

Signature of the incharge
Date:

2) Lower: This String function will convert input string in to lower case.

Syntax: Lower(string)

Example:

SQL> select lower('AZURE') from dual;

Test Output:

Signature of the incharge
Date:

3) Upper: This string function will convert input string in to upper case.

Syntax: Upper(string)

Example:

SQL> select upper(‘azure’) from dual;

Test Output:

Signature of the incharge
Date:

4) Ltrim (Left Trim):

Syntax: Ltrim(string,set)

Example:

SQL>select ltrim(‘azuretech’,’azure’) from dual;
Test Output:

Signature of the incharge
Date:

5) **Rtrim (Right Trim):**

Syntax: Rtrim(string,set)

Example:

SQL>select rtrim(‘azuretrim’,’trim’) from dual;

Test Output:

Signature of the incharge
Date:

6) Translate:

Syntax: Translate(string1, string2, string3)

Example:

SQL>select translate(‘abcde’,’xaybzcxdye’,’tanzmulrye’) from dual;

Test Output:

Signature of the incharge
Date:

7) Replace:

Syntax: Replace(string, searchstring, replacestring)

Example:

SQL> select replace(‘jack and jue’,’j’,’bl’) from dual;
Test Output:

Signature of the incharge
Date:

8) Substr:

Syntax: Substr (string, starts [, count])

Example:

SQL>select substr (‘azuretechnology’,4,6) from dual;

Test Output:

Signature of the incharge
Date:

9) Chr:

Syntax: Chr(number)

Test Output:

Signature of the incharge
Date:

Example:

SQL>select chr(65) from
dual;
Test Output:

Signature of the incharge
Date:

10) Lpad (Left Pad):

Syntax: Lpad(String,length,pattern)

Example:

Sql > select lpad('Welcome',15,'*') from dual;

Test Output:

Signature of the incharge
Date:

11) Rpad (Right Pad):
Syntax: Lpad(String,length,pattern)

Example:

SQL> select rpad('Welcome',15,'*') from dual;
Test Output:

Signature of the incharge
Date:

12) Length:

Syntax:Length(string)

Example:

SQL>select length('auzre') from dual;

Test Output:

Signature of the incharge
Date:

13) Decode:

Syntax: Select decode(column name,if,then,if,then_ ..) from <tablename>;
Example:
SQL> select deptno,decode(deptno,10, ‘Sales’, 20, ‘Purchase’, ‘Account’)
DNAME from emp_master;
Test Output:

Signature of the incharge
Date:

14) Concatenation (||) Operator:

Syntax: Concat(string1,string2)

SQL> select concat(‘Azure’,’ Technology’) from dual;

Test Output:

Signature of the incharge
Date:

SQL> select ‘ename is ‘||ename from emp_master;
Test Output:

Signature of the incharge
Date:

Numeric Functions:
1) Abs (Absolute):

Syntax: Abs(Negetive Number)

Example:

SQL> select Abs(-10) from dual;

Test Output:

Signature of the incharge
Date:

2) Ceil

Syntax: Ceil(Number)

Example:

SQL>select Ceil (23.77) from dual;

Test Output:

Signature of the incharge
Date:

3) Floor:
Syntax: Floor(Number)

Example:

SQL>select Floor(45.3) from dual;

Test Output:

Signature of the incharge
Date:

4) Power:

Syntax: Power(Number, Raise)

Example:

SQL>Select power (5,2) from dual;

Test Output:

Signature of the incharge
Date:

5) Mod:

Syntax: Mod(Number, DivisionValue)

Example:
SQL>select Mod(10,3) from dual;

Test Output:

Signature of the incharge
Date:

6) Sign:

. SQL>select sign(-45) from dual;

Test Output:

Signature of the incharge
Date:

SQL>Select sign(45) from dual;
Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Date Function:

1) Add_Months:

Syntax: Add_Months(Date,no.of Months)

Example:
SQL> select Add_Months(sysdate,2) from dual;
Test Output:

Signature of the incharge
Date:

2) Last_day:

Syntax: Last_day(Date)
Example:
SQL> select sysdate, last_day(sysdate) from dual;
Test Output:

Signature of the incharge
Date:

3) Months_Between:

Syntax: Months_Between(Date1,Date2)

Example:

SQL>select months_between(sysdate,'02-AUG-01') onths_ from dual;

Test Output:

Signature of the incharge
Date:

4) Next_Day:.

Syntax: Next_Day(Date,Day)

Example:

SQL>select next_day(sydate, 'sunday') ext_ from dual;

Test Output:

Signature of the incharge
Date:

5) Round:
Syntax: Round (Date, [fmt])
Example:
SQL>Select round('4-sep-01','day') ounded_ from dual;
Test Output:

Signature of the incharge
Date:

6) Trunc (Truncate):

Syntax: Trunc(Date,[fmt])
Example:
SQL>Select Trunc('4-sep-01','day') runcated_ from dual;
Test Output:

Signature of the incharge
Date:

Conversion Functions:

To_Number(

) To_Char()

To_Date()

To_Number:
Example:
SQL>Select to_number('50') from dual;

Test Output:

Signature of the incharge
Date:

2) To_Char:

Syntax: To_char(no,[fmt])
Example:
SQL> select to_char(17145,'$099,999') har_ from dual;
Test Output:

Signature of the incharge
Date:

Syntax: To_char(Date,[fmt])
Example:

SQL>select to_char(hiredate, 'month dd yyyy') ireDate_ from emp_master

where salary = 10000;
Test Output:

Signature of the incharge
Date:

3) To_Date:

Syntax: To_date(char,[fmt])
Example:
SQL>select to_date('27 January 2000','dd/mon/yy') ate_ from dual;

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

\ \

Signature of the instructor Date:

SQL>select*From Reservation UNION select*from Cancellation;
Test Output:

Signature of the incharge
Date:

SQL>select pnr_no,count(*) as no occurrences from passenger group by pnr_no having count(*)>0;
Test Output:

Signature of the incharge
Date:

SQL> select PNR_NO,sum(No_of_seats) from Reservation group by PNR_NO;
Test Output:

Signature of the incharge
Date:

4. Find the number of seats booked in each class where the number of seats is greater than 1.
SQL> select class, sum(No_of_seats) from Reservation where class='a 'or class='b' or class= 'c' group by class having sum(No_of_seats)>1;
Test Output:

Signature of the incharge
Date:

5. Find the total number of cancelled seats.
SQL> select sum(No_of_seats) from Cancellation;
Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

6. Creating and dropping views

AIM : Implement Joins

Syntax for joining tables

select columns from table1, table2, ... where logical expression;

Simple Join :

Example:

SQL> select * from order_master , order_detail where Order_master.order_no = order_detail.order_no;

Test Output:

Signature of the incharge
Date:

Example:

SQL> select a.*, b.* from itemfile a, order_detail b where a.max_level< b.qty_ord

and a.itemcode = b.itemcode;

Test Output:

Signature of the incharge
Date:

Self Join :

Example:

SQL> select a.ename, a.salary, b.ename, b.salary from emp a, emp b where a.mgr = b.empno;

Test Output:

Signature of the incharge
Date:

Outer Join :

Example:

SQL> select * from order_master a, order_detail b where a.order_no = b.order_no(+);
Test Output:

Signature of the incharge
Date:

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:

Signature of the incharge
Date:

Example:

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

Test Output:

Signature of the incharge
Date:

Example:

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

Test Output:

Signature of the incharge
Date:

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

Test Output:

Signature of the incharge
Date:

Example:

SQL> select * from order_detail where qty_ord =all(select qty_hand from itemfile where

itemrate =250);

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

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:

Signature of the incharge
Date:

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:

Signature of the incharge
Date:

For delete:

Test Output:

Signature of the incharge
Date:

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:

Signature of the incharge
Date:

SQL>Insert into EmpDept_Vw values(…);

Test Output:

Signature of the incharge
Date:

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

Test Output:

Signature of the incharge
Date:

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

Signature of the incharge
Date:

SQL>Create View EmpRO As select * from Employee with Read Only; Test Output:

Assignment Evaluation **Signature**

0: Not Done [ ] 1: Incomplete [ ] 2: Late complete [ ]

3: Needs improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Signature of the instructor [ ] Date: [\ \ ]

To Create View With Check option:

SQL>Create View EmpCk As Select * from Employee Where Deptno=10 WithCheck Option;

Test Output:

Signature of the incharge

Date:

Destroying a view:

Syntax: Drop View <View_Name>;

Example:

SQL>Drop View Emp_Vw;

Test Output:

Signature of the incharge
Date:

SQL> create view v1 as select * from Passenger full natural join Reservation; View created.

a) INSERT

SQL> insert into male_pass values(&PNR_NO,&age);

Test Output:

Signature of the incharge
Date:

b) DROP VIEW

SQL> drop view male_pass;

Test Output:

Assignment Evaluation **Signature**

0: Not Done | 1: Incomplete | 2: Late complete

3: Needs improvement | 4: Complete | 5: Well Done

Signature of the instructor Date: \ \

AIM : Implement Indexes:
An index is an ordered list of the contents of a column, (or a group of columns) of a table.

Test Output:

Signature of the incharge
Date:

Example:

Select order_no,order_date,client_no From Sales_order Where client_no=’C00001’;

Client_no ROWID

Test Output:

Signature of the incharge
Date:

Index:

Syntax: Create Index <Index Name> On <Table Name>(ColumnName); **Example:**
SQL>Create Index idx_client_no On Client_master (Client_no) ;

Test Output:

Signature of the incharge
Date:

Creating Composite Index:
Syntax: Create Index <Index Name> On <Table Name>(ColumnName, ColumnName); **Example:**
SQL>Create Index idx_sales_order On Sales_order (Order_no,product_no) ;

Test Output:

Signature of the incharge
Date:

Creation of Unique Index:
Syntax: Create Unique Index <Index Name> On <Table Name> (Column Name);
Syntax: Create Unique Index <Index Name> On <Table Name> (ColumnName,ColumnName);
Example:
SQL>Create Unique Index idx_client_no On Client_master (Client_no);
Test Output:

Signature of the incharge
Date:

Dropping Indexes:
Syntax: Drop Index <Index Name>;
Example:
SQL>Drop Index idx_client_no;

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: ☐

Aim : Implementing Operations on relations using PL / SQL.

PL/SQL Block

declare

<declaration of variables, constants, function, procedure,

cursor etc.>;

begin

<executable statement(s)>;

exception

<exception handling>;

end;

/

Example

Begin

Insert into emp(empno,ename) values(100,'Shruti');

Insert into emp(empno,ename) values(101,'Yesha');

End;

/

Test Output:

Signature of the incharge
Date:

SQL>Set Serveroutput On

Example

Write a pl/sql program welcome

Test Output:

Signature of the incharge
Date:

Example

Insert value into dept table using pl/sql

Test Output:

Assignment Evaluation **Signature**

0: Not Done | 1: Incomplete | 2: Late complete

3: Needs improvement | 4: Complete | 5: Well Done

Signature of the instructor Date: \ \

Example

Write a pl/sql program To get the area of the circle provided the radius is given.

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Example

Write a pl/sql program To get the name and salary of specified employee.
Test Output:

Signature of the incharge
Date:

Example

Write a pl/sql program To get the name and salary of specified employee using %type attribute.
Test Output:

Signature of the incharge
Date:

Example

Write a pl/sql program To get the name and salary of specified employee

using %type attribute

Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

PL/SQL Control Structures

Example

Write a pl/sql program Accept Number from a User and display Hello message if the entered number is Positive.
Test Output:

Signature of the incharge
Date:

Example

Write a pl/sql program to Display Salary of a specified employee increasing by 500 if its salary is more than 3000.

Test Output:

Signature of the incharge
Date:

Example

Write a pl/sql program to Accept number from a user and find out whether it is Odd or Even.

Test Output:

Signature of the incharge
Date:

Example

Write a pl/sql program to Accept employee number from a user and increase its salary depends on the current salary as follows.

Salary Increment

>= 5000 12.5%; <5000 11%

Test Output:

Signature of the incharge
Date:

Write a pl/ sql program by using Iterative Control / Loops

Test Output:

Assignment Evaluation **Signature**

0: Not Done [ ] 1: Incomplete [ ] 2: Late complete [ ]

3: Needs improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Signature of the instructor [ ] Date: [\ \ ]

Write a pl/sql program by using while loop.

Test Output:

Signature of the incharge
Date:

Write a pl/sql program by using FOR Loop

Test Output:

Signature of the incharge
Date:

.

Exception Handling

Example

Write a PL/SQL Block to accept employee name from a user if it is exist display its salar y otherwise display appropriate message using exception handling.

Test Output:

Signature of the incharge
Date:

Example

Write a PL/SQL Block to display the salary of that employee whose age is 45 year otherwise display appropriate message using exception handling. Test Output:

Assignment Evaluation **Signature**

0: Not Done [ ] 1: Incomplete [ ] 2: Late complete [ ]

3: Needs improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Signature of the instructor [ ] Date: [\ \ ]

Example

Write a PL/SQL Block to insert add one row in employee table with employee number and name.Display appropriate message using exception handling on duplication entry of employee number.

Test Output:

Signature of the incharge
Date:

Aim : Writing triggers

Database Triggers:
Example
Test Output:

Signature of the incharge
Date:

Example
Create or replace trigger upperdname before insert or update

on dept for each row

Test Output:

Signature of the incharge
Date:

Example

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

Emp.

Test Output:

Signature of the incharge
Date:

Example

Create or replace trigger find_tran before insert or

update or delete on dept for each row Test Output:

Signature of the incharge
Date:

Examples:

Create of insert trigger, delete trigger and update trigger.
Test Output:

Signature of the incharge
Date:

SQL> @trigger

Trigger created.

SQL> select * from Bus;
Test Output:

Assignment Evaluation **Signature**

0: Not Done ____ 1: Incomplete ____ 2: Late complete ____

3: Needs improvement ____ 4: Complete ____ 5: Well Done ____

Signature of the instructor ____ Date: \ \

b) Create Trigger updchek before update on Ticket For Each

Row Test Output:

Signature of the incharge
Date:

b) CREATE OR RELPLACE TRIGGER trig1 before insert on Passenger

for each row

Test Output:

Assignment Evaluation **Signature**

0: Not Done | 1: Incomplete | 2: Late complete

3: Needs improvement | 4: Complete | 5: Well Done

Signature of the instructor Date: \ \

AIM : Implement Cursors:

Example
Test Output:

Signature of the incharge
Date:

Aim; Implement the %notfound Attribute
Write a cursor by using The %notfound Attribute
Test Output:

Signature of the incharge
Date:

Aim; Implement the %found Attribute
Write a cursor program by using The % found Attribute
Test Output:

Signature of the incharge
Date:

Aim: Implement The %rowCount Attribute:

Write a cursor program by using the %rowCount Attribute:
Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Aim:

a) To write a Cursor to display the list of Male and Female Passengers.

Test Output:

Signature of the incharge
Date:

b)To write a Cursor to display List of Passengers from Passenger Table.

Test Output:

Signature of the incharge
Date:

SQL>@Passenger
Test Output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

AIM : Implement SubPrograms in PL/SQL.

Create a procedure, which receives a number and display whether it is odd or

even.

Test Output:

Signature of the incharge
Date

SQL > execute/exec procedure_name(parameter list)

SQL> Exec example1(7)

Test Output:

.

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Example

Make a procedure, which will accept a number and return it's Square.

Test Output:

Signature of the incharge
Date:

Example

Pass employee no and name and store into employee table.
Test Output:

Signature of the incharge
Date:

SQL> exec emp_add(1013,'DAXESH');
Test Output:

Signature of the incharge
Date:

Example

Create a Procedure, which receives employee number and display emplo yee name, Designation and salary.

Signature of the incharge
Date:

SQL>exec empdata(1051)
Test output:

Signature of the incharge
Date:

Example

Write a PL/SQL block, which will use a user-defined procedure, which accept employee number and return employee name and department name in a out parameter.

Test output:

Signature of the incharge
Date:

Example

Create a procedure, which receives department number and get total Salary of that
Test output:

Signature of the incharge
Date:

Example

Write procedure to accept Department number and display Name, Designation and Age of each employee belonging to such Department.

Test output:

Signature of the incharge

Date:

SQL>exec dept_list(20);

Test output:

Signature of the incharge
Date:

Example

Create a procedure, which will accept Deptno and Display no of employee under different grade.

Test output:

AIM : Implement Functions of PL/SQL.

Signature of the incharge
Date:

SQL>exec empcount(30);

Test output:

Assignment Evaluation **Signature**

0: Not Done [ ] 1: Incomplete [ ] 2: Late complete [ ]

3: Needs improvement [ ] 4: Complete [ ] 5: Well Done [ ]

Signature of the instructor [ ] Date: [\ \ ]

Example

Create a function to get cube of passed number

Test output:

Signature of the incharge
Date:

SQL> select cube(5) from dual;

Test output:

Signature of the incharge
Date:

Example
Write a Function to find out maximum salary for the passed designation.
Test output:

Signature of the incharge
Date:

.

SQL> SELECT MAXJOB(‘ANALYST’) FROM

DUAL; Test output:

Signature of the incharge
Date:

Example

Create a Function to find out existence of employee whose name is passed as a parameter

Test output:

Signature of the incharge
Date:

Example

Write a Function to find out Total salary for the passed department Name.

Test output:

Signature of the incharge
Date:

Example

Write a function to check whether passed number is Odd or Even.

Test output:

Signature of the incharge
Date:

Example

Write a Function to find out total annual income for the employee, who's number we passed.

Test output:

Signature of the incharge
Date:

SQL> select ann_income(1010) from dual;

Test output:

Signature of the incharge
Date:

Example

Create a function, which receives three arguments, first two as a number and third as a arithmetic.

Test output:

Assignment Evaluation **Signature**

0: Not Done ☐ 1: Incomplete ☐ 2: Late complete ☐

3: Needs improvement ☐ 4: Complete ☐ 5: Well Done ☐

Signature of the instructor ☐ Date: \ \

Additional Experiments:

1. Create relations for the following schemas and write queries for retrieving data. Student(sid,sname,address)

   Course(cid,cname,fee)

   Enrolled(cid,sid,doj)

2. Apply key constraints & participation constraints for the following relations. emp(eid,ename,sal)

   dept(did,dname,location)

   manages(did,eid,day)

3. Create relations for the following schemas and write quaries for retrieving. Professor (SSN,Nmae,Age,Rank)

   Projects (project no,Sponsor Name,starting date,ending date,budget) Graduate (SSN,Name,Age,Degree)

4. Create relations for the following schemas and write queries for retrieving data. and apply key constraints

5. Create relations for the following schemas and write queries for retrieving data. and apply key constraints

   Sailors

   (sid,sname,rating,age)

   Boats(bid,bname,color)

   Reserves(sid,bid,day)

6. Create relations for the following schemas and write queries for retrieving data. and apply key constraints

   Flights(flno,from,to,distance,departs,arrives)

   Aircraft(aid,anme,cruisingrange)

   Certified(eid,aid)

   Employees(eid,enmae,salary)

JAVA PROGRAMMING

LABORATORY MANUAL

B.TECH
(II YEAR – II SEM)
(2016-17)

Department of Computer Science and Engineering

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

Objectives:

- To prepare students to become familiar with the Standard Java technologies of J2SE
- To prepare students to excel in Object Oriented programming and to succeed as a Java Developer through global rigorous education.
- To provide Students with a solid foundation in OOP fundamentals required to solve programming problems and also to learn Advanced Java topics like J2ME, J2EE, JSP, JavaScript
- To train Students with good OOP programming breadth so as to comprehend, analyze, design and create novel products and solutions for the real life problems.
- To inculcate in students professional and ethical attitude, multidisciplinary approach and an ability to relate java programming issues to broader application context.
- To provide student with an academic environment aware of excellence, written ethical codes and guidelines and lifelong learning needed for a successful professional career.

Outcomes:

Upon successful completion of this course, the students will be able to:

- Able to analyze the necessity for Object Oriented Programming paradigm and over structured programming and become familiar with the fundamental concepts in OOP.
- Demonstrate an ability to design and develop java programs, analyze, and interpret object oriented data and report results.
- Demonstrate an ability to design an object oriented system, AWT components or multithreaded process as per needs and specifications.
- Demonstrate an ability to visualize and work on laboratory and multidisciplinary tasks like console and windows applications both for standalone and Applets programs

INDEX

PROGRAM -1 Date:

Aim: Write a java program to find the Fibonacci series using recursive and non recursive functions

Program:

```
//Class to write the recursive and non recursive functions.
class fib
{
int a,b,c;
//        Non recursive function to find the
Fibonacci series. void nonrecursive(int n)
{
a=0;
b=1;
c=a+b;
System.out.print(b);
while(c<=n)
{
System.out.print(c);
a=b;
b=c;
c=a+b;
}
}
// Recursive function to find the Fibonacci
series. int recursive(int n)
{
if(n==0)
        return (0);
         if(n==1)
                return (1);
        else
                return(recursive(n-1)+recursive(n-2));
}
}
// Class that calls recursive and non recursive functions
 . class fib1
{
public static void main(String args[])
{
int n;
// Accepting the value of n at run
 time. n=Integer.parseInt(args[0]);
System.out.println("the recursion using non recursive is"); // Creating object for
the fib class.fib f=new fib();
// Calling non recursive function of
 fib class. f.nonrecursive(n);
System.out.println("the recursion using recursive is"); ffor(int i=0;i<=n;i++)
{
// Calling recursive function of fib class. int
F1=f.recursive(i); System.out.print(F1);
}
}
}
```

Three Test Outputs:

Signature of the faculty

EXERCISE:

1. Write a java program to print the multiplication table .
2. Write a java program to find the Factorial of a given integer using recursive and non recursive functions

PROGRAM -2 Date:

Aim: Write a java program to multiply two given matrices.

```
// Class to find multiplication
of matrices. class matri
{
public static void main(String args[])
{
// Accept the number of rows and columns at
run time. int m=Integer.parseInt(args[0]);
int n=Integer.parseInt(args[1]);
// Initialize the arrays.
int a[][]=new int[m][n]; int b[][]=new int[m][n]; int c[][]=new int[m][n]; int i=2;
// Loop to accept the values into a
matrix. for(int j=0;j<m;j++)
{for(int k=0;k<n;k++)
{
a[j][k]=Integer.parseInt(args[i]);
i++;
}
}
// Loop to accept the values into
b matrix. for(int j=0;j<m;j++)
{
for(int k=0;k<n;k++)
{
        b[j][k]=Integer.parseInt(args[i]);
        i++;
}
}
// Loop to multiply two matrices .
for(int j=0;j<m;j++)
{
for(int k=0;k<n;k++)
  {
c[j][k]=0;
for(int l=0;l<m;l++)
{
        c[j][k]=c[j][k]+(a[j][l]*b[l][k]);
}
}
}
// Loop to display the result
. for(int j=0;j<m;j++)
{
for(int k=0;k<n;k++)
{
System.out.print(c[j][k]);
}
System.out.println();
}
}
}
```

Three test outputs:

Signature of the faculty

PROGRAM -3 Date:

Aim: Write a java program that reads a line of integers and displays each integers and the sum of all integers use String Tokenizer. Program:

```
// Import the packages to access methods of Scaneer and //StringTokenizer.
import java.util.Scanner;
import java.util.StringTokenizer;

// Class to accept integers and find the sum using StringTokenizer //class.
public class TokenTest1
{
public static void main( String args[] )
{
// Accept the values at run time.
Scanner scanner = new Scanner( System.in );
System.out.println( "Enter sequence of integers (with space between them) and
press Enter" );
// Getting the count of integers that were
entered. String digit = scanner.nextLine();
// Creating object of StringTokenizer class.
StringTokenizer tokens = new StringTokenizer( digit); int i=0,dig=0,sum=0,x;
// Loop to determine the tokens and find the
sum. while ( tokens.hasMoreTokens() )
{
String s=tokens.nextToken(); dig=Integer.parseInt(s);
System.out.print(dig+""); sum=sum+dig;
}

// Display the output. System.out.println();
System.out.println( "sum is "+sum );
}
}
```

Three test outputs:

Signature of the faculty

EXERCISE:

1. Write a java program to find all even and odd integers up to a given integer.
2. Write a java program to add and subtract two given matrices.
3. Write a java program that reads a line of integers and displays each integers and the product of all integers use String Tokenizer.

PROGRAM -4 Date: Aim: Write a java program that checks whether a given string is palindrome or not Program:

```
// Class to find whether string is palindrome or
not. class palindrome
{
public static void main(String args[])
{
// Accepting the string at run time.


// Finding the length of the
string. l=s.length();
// Loop to find the reverse of
the string. for(j=l-1;j>=0;j--)
{
s1=s1+s.charAt(j);
}
// Condition to find whether two strings are equal // and display the
message. if(s.equals(s1))
System.out.println("String "+s+" is
palindrome"); else
System.out.println("String "+s+" is not palindrome");
}

}
```

Three test outputs:

Signature of the faculty

EXERCISE:

1. Write a java program to sort the given integers in ascending/descending order.
2. Write a java program to display characters in a string in sorted order.
3. write a program that uses a sequence inputstream to output the contents of two files.
4. Write a java program that reads a file and displays the file on the screen, with an asterisk mark before each line.
5. Write a java program that displays the number of characters, lines, words, white spaces in a text file.

PROGRAM -5 A) Date:

Aim: Write an applet program that displays a simple message

Program:

Applet1.java:

```
// Import the packages to access the classes and methods in awt and
applet classes. import java.awt.*; import java.applet.*;
public class Applet1 extends Applet
{
// Paint method to display the message.
public void paint(Graphics g)
{
g.drawString("HELLO          WORLD",20,20);
}
}
```

Applet1.html:

```
/* <applet code="Applet1" width=200 height=300> </applet>*/
```

Three test Outputs:

Signature of the faculty

EXERCISE:1.Write an applet program that accepts an integer and display the factorial of a given integer. 2Write an applet program that accepts an integer and display the prime numbers up to that given integer.

PROGRAM -5 B Date:

Aim: Write a Java program compute factorial value using Applet

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class FactorialApplet extends Applet implements ActionListener
{
	/*<applet code="FactorialApplet" height=300 width=300>
	</applet>*/
	Label l1,l2;
	TextField t1,t2;
	Button b1;
	public void init()
	{
		setLayout(new FlowLayout(FlowLayout.LEFT));
		l1=new Label("Enter the value:");
		add(l1);
		t1=new TextField(10);
		add(t1);
		l2=new Label("Factorial value is:");
		add(l2);
		t2=new TextField(10);
		add(t2);
		b1=new Button("Compute");
		add(b1);
		b1.addActionListener(this);
	}
	public void actionPerformed(ActionEvent e)
	{
		if((e.getSource())==b1)
		{
			int value=Integer.parseInt(t1.getText());
			int fact=factorial(value);
			t2.setText(String.valueOf(fact));
		}
	}
	int factorial(int n)
	{
		if(n==0)
			return 1;
		else
			return n*factorial(n-1);
	}
}
```

Three Test Outputs:

Signature of the faculty

Exercise: write an applet program for displaying the circle in green color.

PROGRAM -6 Date:

Aim: Write a java program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result.

Program:

```
import javax.swing.*;
import javax.swing.JOptionPane; import
java.awt.*; import java.awt.event.*;

// Class that initialize the applet and create
calculator. public class Calculator extends JApplet
{
public void init()
{
CalculatorPanel calc=new CalculatorPanel(); getContentPane().add(calc);
}
}
// Class that creates the calculator panel .
class CalculatorPanel extends JPanel implements ActionListener
{
// Creation of JButton.
JButton n1,n2,n3,n4,n5,n6,n7,n8,n9,n0,plus,minus,mul,div,dot,equal;
static JTextField result=new JTextField("0",45); static String
lastCommand=null; // Create the JObjectPane.
JOptionPane p=new JOptionPane(); double
preRes=0,secVal=0,res; private static void assign(String
no) {
if((result.getText()).equals("0"))
result.setText(no); else if(lastCommand=="=")
{
result.setText(no); lastCommand=null; }
else
result.setText(result.getText()+no);
}

 // Creation of control panel of calculator and adding buttons using
 GridLayout. public CalculatorPanel()
{
setLayout(new GridLayout());
 result.setEditable(false);
 result.setSize(300,200);
 add(result);
JPanel panel=new JPanel();
panel.setLayout(new GridLayout(5,5));
n7=new JButton("7"); panel.add(n7);

n7.addActionListener(this);
n8=new JButton("8");
panel.add(n8);
 n8.addActionListener(this);
n9=new JButton("9");
```

```
panel.add(n9);
 n9.addActionListener(this);
 div=new JButton("/");
panel.add(div);
div.addActionListener(this);
 n4=new JButton("4");
panel.add(n4);
 n4.addActionListener(this);
 n5=new JButton("5");
 panel.add(n5);
n5.addActionListener(this);
n6=new JButton("6");
 panel.add(n6);
n6.addActionListener(this);
 mul=new JButton("*");
 panel.add(mul);
 mul.addActionListener(this);
 n1=new JButton("1");
panel.add(n1);
n1.addActionListener(this);
 n2=new JButton("2");
 panel.add(n2);
n2.addActionListener(this);
 n3=new JButton("3");
 panel.add(n3);
 n3.addActionListener(this);
minus=new JButton("-");
 panel.add(minus);
 minus.addActionListener(this);
dot=new JButton(".");
 panel.add(dot);
 dot.addActionListener(this);
n0=new JButton("0");
 panel.add(n0); n0.addActionListener(this);
 equal=new JButton("=");
 panel.add(equal);
 equal.addActionListener(this);
 plus=new JButton("+");
panel.add(plus);
 plus.addActionListener(this);
 add(panel);

}
// Implementing method in ActionListener.
public void actionPerformed(ActionEvent ae)
{
if(ae.getSource()==n1)
          assign("1");
    else if(ae.getSource()==n2)
                  assign("2");
     else if(ae.getSource()==n3)
                   assign("3");
else if(ae.getSource()==n4)
```

```
            assign("4");
 else if(ae.getSource()==n5)
            assign("5");
 else if(ae.getSource()==n6)
            assign("6");
 else if(ae.getSource()==n7)
            assign("7");
 else if(ae.getSource()==n8)
             assign("8");
 else if(ae.getSource()==n9)
            assign("9");
 else if(ae.getSource()==n0)
            assign("0");
else     if(ae.getSource()==dot)
{
if(((result.getText()).indexOf("."))==-1) result.setText(result.getText()+"."); }
else     if(ae.getSource()==minus)
{
preRes=Double.parseDouble(result.getText()); lastCommand="-";
result.setText("0");
}
else     if(ae.getSource()==div)
{
preRes=Double.parseDouble(result.getText());
lastCommand="/";
result.setText("0");
 }
else if(ae.getSource()==equal)
{
 secVal=Double.parseDouble(result.getText());
 if(lastCommand.equals("/"))
            res=preRes/secVal;
else if(lastCommand.equals("*"))
            res=preRes*secVal;
else if(lastCommand.equals("-"))
            res=preRes-secVal;
else if(lastCommand.equals("+"))
            res=preRes+secVal;
result.setText(" "+res); lastCommand="=";
}
else     if(ae.getSource()==mul)
{
preRes=Double.parseDouble(result.getText());
lastCommand="*";
result.setText("0");
}
else     if(ae.getSource()==plus)
{
preRes=Double.parseDouble(result.getText());
lastCommand="+";
result.setText("0");
}
}
```

}

Calculator.html:

```
<applet code="Calculator" width=200 height=300> </applet>
```

Three Test Outputs:

Signature of the faculty

EXERCISE:

Write a java program that use a Grid Layout to arrange Buttons for alphabets. Add a text field to display the words.

PROGRAM -7 Date:

Aim: Write a Java program for display the exception in a message dialogbox import java.awt.*;

```
import javax.swing.*;
import java.awt.event.*;
public class NumOperations extends JApplet implements
ActionListener {
	/*<applet code="NumOperations"
width=300 height=300> </applet>*/
	JLabel l1,l2,l3;
	JTextField t1,t2,t3;
	JButton b1;
	public void init()
	{
		Container contentPane=getContentPane();
		contentPane.setLayout(new FlowLayout());
		l1=new JLabel("Enter num1:");
		contentPane.add(l1);
		t1=new JTextField(15);
		contentPane.add(t1);
		l2=new JLabel("Enter num2:");
		contentPane.add(l2);
		t2=new JTextField(15);
		contentPane.add(t2);
		l3=new JLabel("The Result");
		contentPane.add(l3);
		t3=new JTextField(15);
		contentPane.add(t3);
		b1=new JButton("Divide");
		contentPane.add(b1);
		b1.addActionListener(this);
	}
	public void actionPerformed(ActionEvent e)
	{
		if(e.getSource()==b1)
		{
			try
			{
				int a=Integer.parseInt(t1.getText());
				int b=Integer.parseInt(t1.getText());
				Float c=Float.valueOf(a/b);
				t3.setText(String.valueOf(c));
			}
			catch(NumberFormatException e1)
			{
				JOptionPane.showMessageDialog(this,"Not a valid number");
			}
			catch(ArithmeticException e2)
			{
				JOptionPane.showMessageDialog(this,e2.getMessage());
			}
		}
```

```
        }
}
```

Three test outputs:

Signature of the faculty

Exercise:write a java program that illustrate the use of GridBaglayout.

PROGRAM -8 Date:

Aim: Write a Java program that implements a multi-thread application that has three threads

Program:

```
// Class that create the thread.
class NewThread implements Runnable
{ String name; Thread t;
// NewThread constructor that takes the thread name as
 parameter. NewThread(String threadname)
{
 name=threadname; t=new Thread(this,name);
System.out.println("new thread"+t); t.start();
}

//        Method to run the
thread. public void run()
{
//        The code that may generate the exception. try
{
//        Loop to display the thread name and the
value. for(int i=0;i<5;i++)
{
 System.out.println(name+""+i); Thread.sleep(1000);
}
}
// The block that catches the
 exception. catch(Exception e)
{System.out.println("child interrupted");

}
System.out.println(name+""+"exiting");
}
}

// Class that takes the thread name and run the main
thread. class multithread
{
 public static void main(String args[
 ]) { // Creating child threads.
new NewThread("one"); new NewThread("two");
new NewThread("three");
// Block that may generate the
exception. try
{
for(int i=5;i>0;i--)

{
 System.out.println("main thread"+i);
Thread.sleep(10000);
}
}
```

```
// Block that catch the
 exception. catch(Exception e)
{
System.out.println("main thread interrupted");
}
System.out.println("main thread exiting");
}
}
```

Three test outputs:

Signature of the faculty

Exercise: Write a java program that correctly implements producer consumer problem using the concept of inter thread communication.

PROGRAM -9 A) Date:

Aim: Write a java program that connects to a database using JDBC

Program:

```
import java.sql.Connection;
import java.sql.DriverManager;
public class PostgreSQLJDBC
 {
   public static void main(String args[])
{
     Connection c = null;
     try {
        Class.forName("org.postgresql.Driver");
        c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/testdb",
           "postgres", "123");
     } catch (Exception e) { e.printStackTrace();
        System.err.println(e.getClass().getName()+":

        "+e.getMessage()); System.exit(0);
     }
     System.out.println("Opened database successfully");
   }
}
```

Three test outputs:

Signature of the faculty

Program

B): Write a java program to connect to a database using JDBC and insert values into it

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class PostgreSQLJDBC
{
  public static void main(String args[])
{
    Connection c = null;
    Statement stmt = null;
    try {
      Class.forName("org.postgresql.Driver");
      c = DriverManager
        .getConnection("jdbc:postgresql://localhost:5432/testdb",
        "manisha", "123");
      c.setAutoCommit(false);
      System.out.println("Opened database successfully");
      stmt = c.createStatement();
      String sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
          + "VALUES (1, 'Paul', 32, 'California',
     20000.00 );"; stmt.executeUpdate(sql);

      sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
          + "VALUES (2, 'Allen', 25, 'Texas', 15000.00
      );"; stmt.executeUpdate(sql);

      sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
          + "VALUES (3, 'Teddy', 23, 'Norway', 20000.00
      );"; stmt.executeUpdate(sql);

      sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
          + "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00
      );"; stmt.executeUpdate(sql);

      stmt.close();
      c.commit();
      c.close();
    } catch (Exception e) {
      System.err.println( e.getClass().getName()+": "+ e.getMessage()
      ); System.exit(0);
    }
    System.out.println("Records created successfully");
  }
}
```

Three test outputs:

Signature of the faculty

Program

C): Write a java program to connect to a database using JDBC and delete values from it import java.sql.Connection;

```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;


public class PostgreSQLJDBC6 {
  public static void main( String args[] )
    {
     Connection c = null;
     Statement stmt = null;
     try {
     Class.forName("org.postgresql.Driver");
       c = DriverManager
          .getConnection("jdbc:postgresql://localhost:5432/testdb",
          "manisha", "123");
       c.setAutoCommit(false);
       System.out.println("Opened database successfully");

       stmt = c.createStatement();
       String sql = "DELETE from COMPANY where
       ID=2;"; stmt.executeUpdate(sql);
       c.commit();
       ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;"
       ); while ( rs.next() ) {
```

```
            int id = rs.getInt("id");
            String name = rs.getString("name");
            int age = rs.getInt("age");
            String address = rs.getString("address");
            float salary = rs.getFloat("salary");
            System.out.println( "ID = " + id );
            System.out.println( "NAME = " + name );
            System.out.println( "AGE = " + age );
            System.out.println( "ADDRESS = " + address );
            System.out.println( "SALARY = " + salary );
            System.out.println();
          }
          rs.close();
          stmt.close();
          c.close();
        } catch ( Exception e ) {
          System.err.println( e.getClass().getName()+": "+ e.getMessage()
          ); System.exit(0);
        }
        System.out.println("Operation done successfully");
      }
}
```

Three test outputs:

Signature of the faculty

PROGRAM -10 Date:

Aim: Write a java program to simulate a traffic light

Program:

```
import javax.swing.*;
import java.awt.*;
 import java.awt.event.*;
// Class that allows user to select the traffic lights.
public class Trafficlight extends JFrame implements ItemListener
{
JRadioButton redbut,yellowbut,greenbut;
 public Trafficlight()
{
Container c = getContentPane();
c.setLayout(new FlowLayout());
// Create the button group.
ButtonGroup group= new ButtonGroup();
 redbut = new JRadioButton("Red");
 yellowbut = new JRadioButton("Yellow");
greenbut = new JRadioButton("Green");
 group.add(redbut);
 group.add(yellowbut);
 group.add(greenbut);
// Add the buttons to the
container.   c.add(redbut);
c.add(yellowbut);
c.add(greenbut);
// Add listeners to perform action
 redbut.addItemListener(this);
 yellowbut.addItemListener(this);
 greenbut.addItemListener(this);
 addWindowListener(new WindowAdapter()
{
// Implement methods in Window Event class.
public void windowClosing(WindowEvent e)
{
System.exit(0);
}
} );
setTitle("Traffic Light
"); setSize(250,200);
setVisible(true);
}
// Implement methods in Item Event class.
public void itemStateChanged(ItemEvent e)
{
String  name=  "  ",color="
"; if(redbut.isSelected() )
          name = "Red";
 else if(yellowbut.isSelected() )
           name = "Yellow";
else if(greenbut.isSelected() )
          name = "Green";
```

```
JOptionPane.showMessageDialog(null,"The "+name+" light is simulated,
"MessgeBox", JOptionPane.INFORMATION_MESSAGE); }

public static void main(String args[] )
{
new    trafficlight();
}
}
```

Three Test Outputs:

Signature of the faculty

EXERCISE:

Write a java program that lets the user select one the three options: IT, CSE or ECE. When a radio button is selected, the radio button is turned on and only one option can be on at a time no option is on when program starts.

PROGRAM -11 Date:

Aim: Write a java program to create an abstract class named shape that contains an empty method named number of sides (). Provide three classes named trapezoid, triangle and Hexagon such that each one of the classes extends the class shape. Each one of the class contains only the method number of sides () that shows the number of sides in the given geometrical figures.

Program:

```
// Abstract class that contains
abstract method. abstract class Shape
{
abstract void numberOfSides();
}
// Classes that illustrates the
abstract method. class Trapezoid
{
void  numberOfSides()
{
System.out.println("The no. of side's in trapezoidal are6");
}
}
class Triangle
{
void  numberOfSides()
{
System.out.println("The no. of side's in triangle are:3 ");
}
}
class Hexogon
{
void numberOfSides()
{System.out.println("The no. of side's in hexogon are:6 ");

}
}
// Class that create objects and call
the method. class ShapeDemo
{
public static void main(String args[])
{
Trapezoid obj1 = new Trapezoid();
Triangle obj2 = new Triangle();
Hexogon obj3 = new Hexogon();
obj1.numberOfSides();
obj2.numberOfSides();
obj3.numberOfSides(); }
}
```

Three test outputs:

Signature of the faculty

Exercise:write a program to compute area of different shapes using abstract class.

PROGRAM -12 Date: Aim:Write a java program to display the table using labels in Grid layout import java.awt.*;

```
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
public class TableDemo extends JFrame
{
	int i=0;
	int j=0;
	Object TabData[][]=new Object[5][2];
	JTable mytable;
	FileInputStream fr;
	DataInputStream in;
	public TableDemo()
	{
		String str=" ";
		Container contentpane=getContentPane();
		contentpane.setLayout(new BorderLayout());
		final String[] Column={","};
		try
		{
			FileInputStream fr=new FileInputStream("table.txt");
			DataInputStream in=new DataInputStream(fr);
			if((str=in.readLine())!=null)
			{
				StringTokenizer s=new StringTokenizer(str,",");
				while(s.hasMoreTokens())
				{
					for(int k=0;k<2;k++)
					{
						Column[k]=s.nextToken();
					}
				}
			}
			while((str=in.readLine())!=null)
			{
				StringTokenizer s=new StringTokenizer(str,",");
				while(s.hasMoreTokens())
				{
					for(j=0;j<2;j++)
					{
						TabData[i][j]=s.nextToken();
					}
					i++;
				}
			}
		}catch(Exception e)
		 {
			System.out.println(e.getMessage());
		}
```

```
            mytable=new JTable(TabData,Column);
            int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED; int
            h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
            JScrollPane scroll=new JScrollPane(mytable,v,h);
            contentpane.add(scroll,BorderLayout.CENTER);
        }
        public static void main(String args[])
        {
            TableDemo t=new TableDemo();
            t.setSize(300,300);
            t.setVisible(true);
            t.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```

Three test outputs:

Signature of the faculty

PROGRAM -13 Date:

Aim:Write a java program for handling mouse events

Program: mouseevent.java import java.awt.*;
import java.awt.event.*; import java.applet.*;

```
// Class that handles mouse events.
public class mouseevent extends Applet implements MouseListener,MouseMotionListener
{
 String msg="";
int mousex=0,mousey=0;
//        Method to initialize
the applet. public void init()
{
        addMouseListener(this);
        addMouseMotionListener(this);
}

//        Method to handle mouse clicked event .

public void mouseClicked(MouseEvent me)
{
mousex=0;
mousey=10; msg="mouse clicked"; repaint();
}

// Method to handle mouse entered event . public void mouseEntered(MouseEvent me)
{
mousex=0;
mousey=10; msg="mouse Entered"; repaint();
}

// Method to handle mouse entered event .
public void mouseExited(MouseEvent me)
{
 mousex=0; mousey=10; msg="mouse exited";
repaint();
}
  // Method to handle mouse pressed event .
 public void mousePressed(MouseEvent me)
{
 mousex=me.getX();
mousey=me.getY(); msg="down";
repaint();
}

//        Method to handle mouse relesed event .
 public void mouseReleased(MouseEvent me)
{
mousex=me.getX();
mousey=me.getY();
msg="Up";
```

```
repaint();
}


//          Method to handle mouse dragged event .
public void mouseDragged(MouseEvent me)
{
mousex=me.getX();
mousey=me.getY();
msg="";
showStatus("Dragged mouse at"+mousex+""+mousey); repaint();
}

// Method to handle mouse moved event .
public void mouseMoved(MouseEvent me)
{
showStatus("Moving mouseat"+me.getX()+""+me.getY());
}

// Method to display the message .
public void paint(Graphics g)
{
g.drawString(msg,mousex,mousey);
}
}
```

mouseevent.html:

```
/* <applet code=”mouseevent” width=200
height=200> </applet> */
```

Three Test Outputs:

Signature of the faculty

EXERCISE:

1.Write a java program for handling KEY BOARD events.

PROGRAM -14 Date:

Aim:Write a Java program loads phone no,name from a text file using hash table

Program:

```
// Demonstrate a Hashtable
import java.util.*;
class HTDemo {
public static void main(String args[]) {
Hashtable balance = new Hashtable();
Enumeration names;
String str;
double bal;
balance.put("John Doe", new Double(3434.34));
balance.put("Tom Smith", new Double(123.22));
balance.put("Jane Baker", new Double(1378.00));
balance.put("Todd Hall", new Double(99.22));
balance.put("Ralph Smith", new Double(-19.08));
// Show all balances in hash table.
names = balance.keys();
while(names.hasMoreElements()) {
str = (String) names.nextElement();
System.out.println(str + ": " +
balance.get(str));
}
System.out.println();
// Deposit 1,000 into John Doe's account
bal = ((Double)balance.get("John Doe")).doubleValue();
balance.put("John Doe", new Double(bal+1000));
System.out.println("John Doe's new balance: " +
balance.get("John Doe"));
}
}
```

Three test outputs:

Signature of the faculty

Exercise:

Write a Java program loads list of student names and roll numbers from a text file

PROGRAM -15 Date:

Aim: Implement the above program to load phone no, name from database instead of text file

```
import java.sql.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class PostgreSQLJDBC {
  public static void main( String args[] )
    {
     Connection c = null;
     Statement stmt = null;
     try {
       Class.forName("org.postgresql.Driver");
       c = DriverManager
          .getConnection("jdbc:postgresql://localhost:5432/testdb",
          "manisha", "123");
       System.out.println("Opened database successfully");

       stmt = c.createStatement();
       String sql = "CREATE TABLE COMPANY " +
                    "(ID INT PRIMARY KEY    NOT NULL," +
                    " NAME          TEXT  NOT NULL, " +
                    " AGE           INT   NOT NULL, " +
                    " ADDRESS       CHAR(50), " +
                    " SALARY        REAL)";
       stmt.executeUpdate(sql);
       stmt.close();
       c.close();
     } catch ( Exception e ) {
       System.err.println( e.getClass().getName()+": "+ e.getMessage()
       ); System.exit(0);
     }
     System.out.println("Table created successfully");
    }
}
```

Three test outputs:

Signature of the faculty

Exercise: Implement the above program to load emp details name,salary,address, from database .

PROGRAM -16 Date:

Aim:Write a Java program that takes tab separated data from a text file and inserts them into a database.

Program:

```
import java.io.BufferedReader;
import java.io.FileReader;

public class TabSeparatedFileReader {

   public static void main(String args[]) throws Exception
      { /**
       * Source file to read
       data from. */
      String dataFileName = "C:/temp/myTabSeparatedFile.txt";

      /**
       * Creating a buffered reader to read the
       file */
      BufferedReader bReader = new BufferedReader( new
            FileReader(dataFileName));
       String
       line; /**
       * Looping the read block until all lines in the file
       are read. */
      while ((line = bReader.readLine()) != null) {

         /**
          * Splitting the content of tabbed separated
          line */
         String datavalue[] = line.split("\t");
         String value1 = datavalue[0];
         String value2 = datavalue[1];
         int value3 = Integer.parseInt(datavalue[2]);
         double value4 = Double.parseDouble(datavalue[3]);

         /**
          * Printing the value read from file to the
          console */
         System.out.println(value1 + "\t" + value2 + "\t" + value3 + "\t"
               + value4);
      }
      bReader.close();
   }
```

Three test outputs:

Signature of the faculty

Exercise:

Write a program to reverse the specified n number of characters from the given text file and insert the data into database.

PROGRAM -17 Date:

Aim: Write a Java program that prints the meta-data of a given table

Program:

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.SQLException;
public class JDBCDriverInformation {
        static String userid="scott", password = "tiger";
        static String url = "jdbc:odbc:bob";
        static Connection con = null;
        public static void main(String[] args) throws Exception
          { Connection con = getOracleJDBCConnection();
          if(con!= null){
            System.out.println("Got Connection."); DatabaseMetaData meta
            = con.getMetaData(); System.out.println("Driver Name :
            "+meta.getDriverName()); System.out.println("Driver Version
            : "+meta.getDriverVersion());

          }else{
                  System.out.println("Could not Get Connection");
          }
        }

        public static Connection getOracleJDBCConnection(){

                try {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                } catch(java.lang.ClassNotFoundException e) {
                        System.err.print("ClassNotFoundException: ");
                        System.err.println(e.getMessage());
                }

                try {
                  con = DriverManager.getConnection(url, userid,
                password); } catch(SQLException ex) {
                        System.err.println("SQLException: " + ex.getMessage());
                }

                return con;
        }


}
```

Three test outputs:

Signature of the faculty

Exercise: Write a Java program that prints the meta-data of a given hash table.