LABORATORY MANUAL

ON

DATABASE MANAGEMENT SYSTEM LAB.

OF

FOURTH SEMESTER

(Department of Information Technology)

GOVERNMENT POLYTECHNIC, BHUBANESWAR
Government of Odisha
ସରକାରୀ ବହୁବୃତ୍ତି ଅନୁଷ୍ଠାନ, ଭୁବନେଶ୍ୱର

VISION AND MISSION OF THE INSTITUTE

VISION

The Vision of the Government Polytechnic, Bhubaneswar is to be a premier Diploma Institution in the Eastern India by transforming the learners into competent technocrats by imparting technical education in multiple disciplines.

MISSION

The mission of the Government Polytechnic, Bhubaneswar is to:

- Focus on quality teaching and learning by providing better academic environment for both faculties and students.
- Develop professionally and technically skilled students who work for betterment of the society
- Nurture positive attitude and social values amongst the students and staff.

VISION AND MISSION OF THE DEPARTMENT

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

The Vision of the Department of Information Technology is to develop knowledge and skill of students by quality education to make responsible and competent technocrats who will work for the needs of industries and betterment of the society.

MISSION

The Mission of the Department of Information Technology is to:

M-1. Provide an excellent learning environment for the students to be technically efficient in solving problems.

M-2. Prepare the students with strong fundamental concepts, analytical capability and programming skills.

M-3. Create a nurturing environment for lifelong learning.

M-4. Generate social awareness and responsibility within the students to serve the mankind and protect the environment.

Program Educational Objectives (PEOs)

Program Educational Objective of Diploma in Information Technology:

The Program Educational Objectives (PEOs) of the Department of Information Technology are:

PEO-1: ***Strong Fundamentals & Technically skilled:*** *To equip the students with sound knowledge in mathematics, science and basic engineering fundamentals and develop skill to solve engineering problems.*

PEO-2: ***Higher study and Professional responsibilities:*** *To enable students to pursue higher studies for professional growth and/or work as an entrepreneur for successful professional career.*

PEO-3: ***Teamwork and lifelong learning:*** *To inculcate soft skill, professional and ethical attitude, teamwork, leadership skills and multidisciplinary approach within the students which will enable them to correlate societal issue with engineering solution and pursue lifelong learning*

Program Outcomes and Program Specific Outcomes (PSOs)

Program Outcomes for an Engineering diploma graduate:

i) **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems

ii) **Problem analysis:** Identify and analyze well-defined engineering problems using codified standard methods.

iii) **Design/ development of solutions:** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

iv) **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

v) **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.

vi) **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

vii) **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

Program Specific Outcome of Diploma in Information Technology:

PSO-1: Ability to design and develop methodologies in computer programming and apply those skills in the field of Information Technology.

PSO-2: Develop potentiality to apply the skills learnt for betterment of the society by updating it on the basis of the advanced technological changes and effectively communicate about well-defined engineering activities.

Pr.4-DATABASE MANAGEMENT SYSTEM LAB

Total Periods	60	Maximum Marks	100 Marks
Lab. Periods:	4 Periods /week	Term Works	50 Marks
Examination	3hours	End Semester Examination	50Marks

Experiment No-1

1.0 CONTEXT

1.1Subject: - Database management System Lab

1.2 Experiment: - 1

1.3 Topic: - SQL

1.4 Sub-Topic: -RDBMS

1.5 Title of the Experiment: -Introduction to SQL

1.6 Environment/Location: - Computer lab of IT Dept.

2.0 AIM OF THE EXPERIMENT: - INTRODUCTION TO SQL

3.0 OBJECTIVE OF THE EXPERIMENT:-

4.0 DESIGN OF EXPERIMENT:-

4.1 Apparatus required: - Computer System with specifications

- Window OS-10
- ORACLE-11g (software)
- 6GB RAM
- 64Bit ROM
- 4.2 Experiment used:- Computer system with peripherals

5.0 THEROY: -

SQL

SQL is a standard language for storing, manipulating and retrieving data in databases.

Our SQL tutorial will teach you how to use SQL in: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix and other database systems.

SQL is a standard language for accessing and manipulating databases.

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases

- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and view

SQL is a Standard:-

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language. However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

RDBMS:-

RDBMS stands for Relational Database Management System.
RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
The data in RDBMS is stored in database objects called **tables**. A table is a collection of related data entries and it consists of columns and rows.

```
SELECT * FROM Customers;
```

Every table is broken up into smaller entities called **fields.**

The fields in the Customers table consist of CustomerID, CustomerName, Contact Name, Address, City, PostalCode and Country.

A **field** is a column in a table that is designed to maintain specific information about every record in the table.

A **record**, also called a row, is each individual entry that exists in a table. For example, there are 91 records in the above Customers table. A record is a horizontal entity in a table.

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

SQL Syntax

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

In this tutorial we will use the well-known Northwind sample database (included in MS Access and MS SQL Server).

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	AlfredsFutterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglundssnabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

The table above contains five records (one for each customer) and seven columns (CustomerID, CustomerName, ContactName, Address, City, PostalCode, and Country).

SQL Statements

Most of the actions you need to perform on a database are done with SQL statements.

The following SQL statement selects all the records in the "Customers" table:

Example

```
SELECT * FROM Customers;
```

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

In this tutorial, we will use semicolon at the end of each SQL statement.

Some of the Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

The SQL SELECT Statement

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

The data returned is stored in a result table, called the result-set.

SELECT Syntax
SELECT *column1*, *column2, ...*
FROM *table name*;

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT * FROM *table name*;

Demo Database

Below is a selection from the "Customers" table in the North wind sample database:

Customer ID	Customer Name	Contact Name	Address	City	Postal Code	Country
1	AlfredsFutterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos	México	05023	Mexico

			2312	D.F.		
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglundssnabbköp	Christina Berglund	Berguvsvägen8	Luleå	S-958 22	Sweden

SELECT Column Example

The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

Example

```
SELECT Customer Name, City FROM Customers;
```

SELECT * Example

The following SQL statement selects all the columns from the "Customers" table:

Example

```
SELECT * FROM Customers;
```

SQL SELECT DISTINCT Statement

The SQL SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

```
SELECT DISTINCT column1, column2, ...
FROM table name;
```

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

Customer ID	Customer Name	Contact Name	Address	City	Postal Code	Country
1	AlfredsFutterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglundssnabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

SELECT Example Without DISTINCT

The following SQL statement selects ALL (including the duplicates) values from the "Country" column in the "Customers" table:

Example

```
SELECT Country FROM Customers;
```

SELECT DISTINCT Examples

The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

Example

```
SELECT DISTINCT Country FROM Customers;
```

The following SQL statement lists the number of different (distinct) customer countries:

Example

```
SELECT COUNT (DISTINCT Country) FROM Customers;
```

6.0 OBSERVATION: - Brief idea about SQL and Most Important SQL Commands.

8.0 SAFETY &PRECAUTIONS:-

1. Turn off the computer and lights of the lab while leaving.
2. Do not temper or corrupt any file in data base
3. Do not eat chewing gum in the lab
4. Handle the electrical item carefully

9.0 REVIEW QUESTIONS:-

1. What is SQL?
2. What is the use of SQL?

3. Define DBMS and RDBMS.
4. What is a Database table?
5. What is Field and Record?
6. What are the Most Important SQL Commands?
7. What is SQL SELECT Statement?

Experiment No-2

1.0 CONTEXT

1.1Subject: - Database management System Lab

1.2 Experiment: - 2

1.3 Topic: - SQL

1.4 Sub-Topic: -WHERE Clause, Use of SQL AND, OR and NOT Operators and The SQL ORDER BY Keyword

1.5 Title of the Experiment: -SQL WHERE Clause, **, Use of** SQL AND, OR and NOT Operators and The SQL ORDER BY Keyword

1.6 Environment/Location: - Computer lab of IT Dept.

2.0 AIM OF THE EXPERIMENT: - **Use SQL WHERE Clause, Use of** SQL AND, OR and NOT Operators and The SQL ORDER BY Keyword

3.0 OBJECTIVE OF THE EXPERIMENT: - To know how to use WHERE Clause to access information from a database.

4.0 DESIGN OF EXPERIMENT:-

4.1 Apparatus required: - Computer System with specifications

- **Window OS-10**
- **ORACLE-11g (software)**
- **6GB RAM**
- **64Bit ROM**

4.2 Experiment used: - Computer system with Peripherals

5.0 THEROTY:-

The SQL WHERE Clause

The WHERE clause is used to filter records.
The WHERE clause is used to extract only those records that fulfill a specified condition.

WHERE Syntax
SELECT *column1*, *column2, ...*
FROM *table name*
WHERE *condition*;

WHERE Clause Example

The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

```
Example
SELECT * FROM Customers
WHERE Country='Mexico';
```

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

```
Example
SELECT * FROM Customers
WHERE Customer ID=1;
```

Operators in the WHERE Clause

The following operators can be used in the WHERE clause:

Operator	**Description**
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern

IN	To specify multiple possible values for a column

The SQL AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators.
The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table name
WHERE NOT condition;
```

AND Example

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

Example

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

OR Example

The following SQL statement selects all fields from "Customers" where city is "Berlin" OR "München":

Example

```
SELECT * FROM Customers
WHERE City='Berlin' OR City='München';
```

NOT Example

The following SQL statement selects all fields from "Customers" where country is NOT "Germany":

Example

```
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

Combining AND, OR and NOT

You can also combine the AND, OR and NOT operators.

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city must be "Berlin" OR "München" (use parenthesis to form complex expressions):

Example

```
SELECT * FROM Customers
WHERE Country='Germany' AND (City='Berlin' OR City='München');
```

The following SQL statement selects all fields from "Customers" where country is NOT "Germany" and NOT "USA":

Example

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT Country='USA';
```

The SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

ORDER BY Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

ORDER BY Example

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

Example

```
SELECT * FROM Customers
ORDER BY Country;
```

ORDER BY DESC Example

The following SQL statement selects all customers from the "Customers" table, sorted DESCENDING by the "Country" column:

Example

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

6.0 OBSERVATION: - Brief idea about SQL WHERE Clause, Use of SQL AND, OR and NOT Operators and The SQL ORDER BY Keywords.

7.0 SAFETY &PRECAUTIONS:-

1. Turn off the computer and lights of the lab while leaving.
2. Do not temper or corrupt any file in data base
3. Do not eat chewing gum in the lab

4. Handle the electrical item carefully

8.0 REVIEW QUESTIONS:-

1. What is use of WHERE Clause in SQL?
2. What is the use of SQL AND, OR and NOT Operators**in** SQL?
3. What are the comparison operators?
4. What is the use of ORDER By clause?
5. What is ASC and DESC?

Experiment No-3

1.0 CONTEXT

1.1 Subject: - Database management System Lab

1.2 Experiment: - 3

1.3 Topic: - SQL

1.4 Sub-Topic: -insert into statement

1.5 Title of the Experiment: - SQL insert into statements

1.6 Environment/Location: - Computer lab of IT Dept.

2.0 AIM OF THE EXPERIMENT: - **Use SQL insert into statements**

3.0 OBJECTIVE OF THE EXPERIMENT: - To know how to use insert into statement to access information from a database.

4.0 DESIGN OF EXPERIMENT: -

4.1 Apparatus Required: - Computer System with specifications

- **Window OS-10**
- **ORACLE-11g (software)**
- **6GB RAM**
- **64Bit ROM**

4.2 Experiment used: - Computer system with Peripherals

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two ways.

The first way specifies both the column names and the values to be inserted:

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

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

```
INSERT INTO table name
VALUES (value1, value2, value3, ...);
```

INSERT INTO Example

The following SQL statement inserts a new record in the "Customers" table:

Example

```
INSERT INTO Customers (Customer Name, Contact Name, Address, City, Postal Code, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

Insert Data Only in Specified Columns

It is also possible to only insert data in specific columns.

The following SQL statement will insert a new record, but only insert data in the "Customer Name", "City", and "Country" columns (Customer ID will be updated automatically):

Example

```
INSERT INTO Customers (Customer Name, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

IS NULL Syntax

```
SELECT column names
FROM table name
WHERE column name IS NULL;
```

IS NOT NULL Syntax

```
SELECT column names
FROM table name
WHERE column name IS NOT NULL;
```

The IS NULL Operator

The IS NULL operator is used to test for empty values (NULL values).

The following SQL lists all customers with a NULL value in the "Address" field:

Example

```
SELECT Customer Name, Contact Name, Address
FROM Customers
WHERE Address IS NULL;
```

The IS NOT NULL Operator

The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

The following SQL lists all customers with a value in the "Address" field:

Example

```
SELECT Customer Name, Contact Name, Address
FROM Customers
WHERE Address IS NOT NULL;
```

The SQL UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax

```
UPDATE table name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

UPDATE Table

The following SQL statement updates the first customer (Customer ID = 1) with a new contact person *and* a new city.

Example

```
UPDATE Customers
SET Contact Name = 'Alfred Schmidt', City= 'Frankfurt'
WHERE Customer ID = 1;
```

UPDATE Multiple Records

It is the WHERE clause that determines how many records will be updated.

The following SQL statement will update the contactname to "Juan" for all records where country is "Mexico":

Example

```
UPDATE Customers
SET Contact Name='Juan'
WHERE Country='Mexico';
```

The SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

DELETE Syntax

```
DELETE FROM table name WHERE condition;
```

SQL DELETE Example

The following SQL statement deletes the customer "AlfredsFutterkiste" from the "Customers" table:

Example

```
DELETE FROM Customers WHERE Customer Name='AlfredsFutterkiste';
```

Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table name;
```

The following SQL statement deletes all rows in the "Customers" table, without deleting the table:

Example

```
DELETE FROM Customers;
```

6.0 OBSERVATION: - Brief idea about SQL insert into statement, update a table, delete record from table

7.0 SAFETY & PRECAUTIONS: -

1. turn off the computer and lights of the lab while leaving.
2. do not temper or corrupt any file in data base
3. do not eat chewing gum in the lab
4. handle the electrical item carefully

8.0 REVIEW QUESTIONS: -

1. What is use of insert into in SQL?
2. What is the use of null**in** SQL?
3. What are the ways to update table?
4. What is the use of delete statement?

EXPERIMENT-4

1.0 CONTEXT

1.1 Subject: - Database management System Lab

1.2 Experiment: - 4

1.3 Topic: - SQL

1.4 Sub-Topic: -sql max(),min(),count(),sum(),avg() functions

1.5 Title of the Experiment: - SQL max(),min(),count(),sum(),avg() functions.

1.6 Environment/Location: - Computer lab of IT Dept.

2.0 AIM OF THE EXPERIMENT: - **Use SQL max(), min(), count(), sum(), avg() functions**
3.0 OBJECTIVE OF THE EXPERIMENT: - To know how to use max(),min(),count(),sum(),avg() functions to access information from a database.

4.0 DESIGN OF EXPERIMENT: -

4.1 Apparatus Required: - Computer System with specifications

- **Window OS-10**
- **ORACLE-11g (software)**
- **6GB RAM**
- **64Bit ROM**

4.2 Experiment used: - Computer system with Peripherals

SQL MIN () and MAX () Functions

The SQL MIN () and MAX () Functions

The MIN () function returns the smallest value of the selected column.

The MAX () function returns the largest value of the selected column.

MIN () Syntax
SELECT MIN (*column name*)
FROM *table name*
WHERE *condition*;

MAX() Syntax

```
SELECT MAX(column name)
FROM table name
WHERE condition;
```

MIN() Example

The following SQL statement finds the price of the cheapest product:

Example

```
SELECT MIN(Price) AS Smallest Price
FROM Products;
```

MAX() Example

The following SQL statement finds the price of the most expensive product:

Example

```
SELECT MAX(Price) AS Largest Price
FROM Products;
```

The SQL COUNT(), AVG() and SUM() Functions

The COUNT() function returns the number of rows that matches a specified criteria.

The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

COUNT() Syntax

```
SELECT COUNT(column name)
FROM table name
WHERE condition;
```

AVG () Syntax

```
SELECT AVG (column_name)
FROM table_name
WHERE condition;
```

SUM () Syntax

```
SELECT SUM (column_name)
FROM table_name
WHERE condition;
```

COUNT () Example

The following SQL statement finds the number of products:

Example

```
SELECT COUNT(Product ID)
FROM Products;
```

AVG () Example

The following SQL statement finds the average price of all products:

Example

```
SELECT AVG(Price)
FROM Products;
```

SUM () Example

The following SQL statement finds the sum of the "Quantity" fields in the "Order Details" table:

Example

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character
- The percent sign and the underscore can also be used in combinations!
- LIKE Syntax
- SELECT *column1, column2, ...*
  FROM *table_name*
  WHERE *column* LIKE *pattern*;

SQL LIKE Examples

The following SQL statement selects all customers with a Customer Name starting with "a":

Example

```
SELECT * FROM Customers
WHERE Customer Name LIKE 'a%';
```

The following SQL statement selects all customers with a Customer Name ending with "a":

Example

```
SELECT * FROM Customers
WHERE Customer Name LIKE '%a';
```

The following SQL statement selects all customers with a Customer Name that have "or" in any position:

Example

```
SELECT * FROM Customers
WHERE Customer Name LIKE '%or%';
```

The following SQL statement selects all customers with a Customer Name that have "r" in the second position:

Example

```
SELECT * FROM Customers
WHERE Customer Name LIKE '_r%';
```

The following SQL statement selects all customers with a Customer Name that starts with "a" and are at least 3 characters in length:

Example

```
SELECT * FROM Customers
WHERE Customer Name LIKE 'a__%';
```

The following SQL statement selects all customers with a Contact Name that starts with "a" and ends with "o":

Example

```
SELECT * FROM Customers
WHERE Contact Name LIKE 'a%o';
```

The following SQL statement selects all customers with a Customer Name that does NOT start with "a":

Example

```
SELECT * FROM Customers
WHERE Customer Name NOT LIKE 'a%';
```

The SQL IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

IN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

or:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

IN Operator Examples

The following SQL statement selects all customers that are located in "Germany", "France" or "UK":

Example

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

he following SQL statement selects all customers that are NOT located in "Germany", "France" or "UK":

Example

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

The following SQL statement selects all customers that are from the same countries as the suppliers:

Example

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

The SQL BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

BETWEEN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

BETWEEN Example

The following SQL statement selects all products with a price BETWEEN 10 and 20:

Example

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

NOT BETWEEN Example

To display the products outside the range of the previous example, use NOT BETWEEN:

Example

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

SQL Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.
Aliases are often used to make column names more readable.
An alias only exists for the duration of the query.

Alias Column Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

Alias for Columns Examples

The following SQL statement creates two aliases, one for the Customer ID column and one for the Customer Name column:

Example

```
SELECT Customer ID AS ID, Customer Name AS Customer
FROM Customers;
```

The following SQL statement creates two aliases, one for the Customer Name column and one for the Contact Name column. **Note:** It requires double quotation marks or square brackets if the alias name contains spaces:

6.0 OBSERVATION: - Brief idea about SQL Use SQL max(), min(), count(), sum(), avg() functions, like operator, in operator, between operator.

7.0 SAFETY & PRECAUTIONS: -

1. Turn off the computer and lights of the lab while leaving.
2. Do not temper or corrupt any file in data base
3. Do not eat chewing gum in the lab
4. Handle the electrical item carefully

8.0 REVIEW QUESTIONS: -

1. What is use of min() and max() in SQL?
2. What is the use of count()**in** SQL?
3. What the use of sum() and avg() in sql?
4. What is the use of like and in operator?
5. What is the use of between operator?

EXPERIMENT-5

1.0 CONTEXT

1.1 Subject: - Database management System Lab

1.2 Experiment: - 5

1.3 Topic: - SQL

1.4 Sub-Topic: -sql join and group join and having clause.

1.5 Title of the Experiment: - SQL join and group join functions.

1.6 Environment/Location: - Computer lab of IT Dept.

2.0 AIM OF THE EXPERIMENT: - **Use join and group join functions and having clause**
3.0 OBJECTIVE OF THE EXPERIMENT: - To know how to use join and group join functions to access information from a database.

4.0 DESIGN OF EXPERIMENT: -

4.1 Apparatus required: - Computer System with specifications

- **Window OS-10**
- **ORACLE-11g (software)**
- **6GB RAM**
- **64Bit ROM**

4.2 Experiment used: - Computer system with Peripherals

SQL JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Let's look at a selection from the "Orders" table:

Order ID	Customer ID	Order Date
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Then, look at a selection from the "Customers" table:

Customer ID	Customer Name	Contact Name	Country
1	AlfredsFutterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Notice that the "Customer ID" column in the "Orders" table refers to the "Customer ID" in the "Customers" table. The relationship between the two tables above is the "Customer ID" column.

Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

Example

```
SELECT Orders.Order ID, Customers.Customer Name, Orders.Order Date
FROM Orders
INNER JOIN Customers ON Orders.Customer ID=Customers.Customer ID;
```

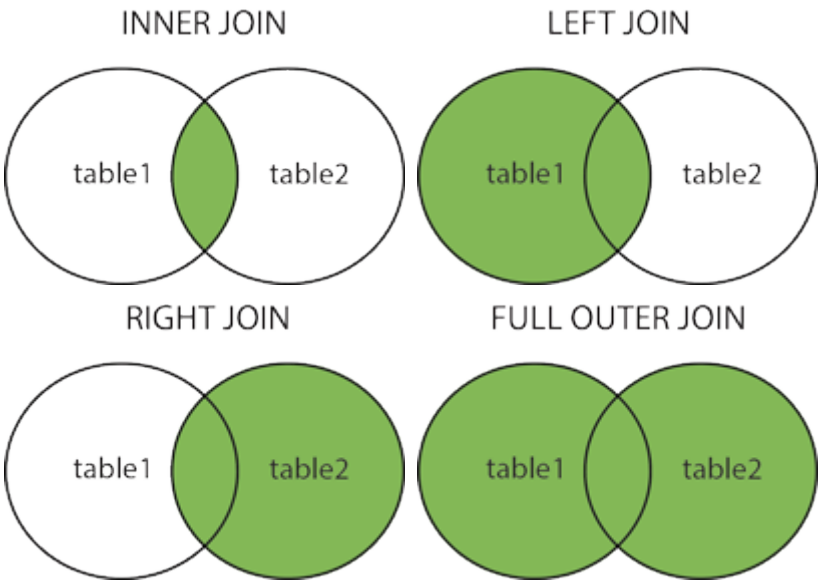
And it will produce something like this:

Order ID	Customer Name	Order Date
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglundssnabbköp	8/12/1996

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table



The SQL GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

Customer ID	Customer Name	Contact Name	Address	City	Postal Code	Country
1	AlfredsFutterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglundssnabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

SQL GROUP BY Examples

The following SQL statement lists the number of customers in each country:

Example

```
SELECT COUNT(Customer ID), Country
FROM Customers
GROUP BY Country;
```

The following SQL statement lists the number of customers in each country, sorted high to low:

Example

```
SELECT COUNT(Customer ID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(Customer ID) DESC;
```

The SQL HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

HAVING Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

SQL HAVING Examples

The following SQL statement lists the number of customers in each country. Only include countries with more than 5 customers:

Example

```
SELECT COUNT(Customer ID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(Customer ID) > 5;
```

The following SQL statement lists the number of customers in each country, sorted high to low (Only include countries with more than 5 customers):

Example

```
SELECT COUNT(Customer ID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(Customer ID) > 5
ORDER BY COUNT(Customer ID) DESC;
```

6.0 OBSERVATION: - Brief idea about SQL Use of join and group join functions, having clause

7.0 SAFETY & PRECAUTIONS: -

1. Turn off the computer and lights of the lab while leaving.
2. Do not temper or corrupt any file in data base
3. Do not eat chewing gum in the lab
4. Handle the electrical item carefully

8.0 REVIEW QUESTIONS: -

1. What is use of join in SQL?
2. What is the use of group join**in** SQL?
3. Name different type of join in sql?
4. What is the use of having clause?

EXPERIMENT-6

1.0 CONTEXT

1.1 Subject: - Database management System Lab

1.2 Experiment: - 6

1.3 Topic: - SQL

1.4 Sub-Topic: -sql create database statement.

1.5 Title of the Experiment: - SQL create database statement.

1.6 Environment/Location: - Computer lab of IT Dept.

2.0 AIM OF THE EXPERIMENT: - **Use create database statement.**
3.0 OBJECTIVE OF THE EXPERIMENT: - To know how to create database and create table sql drop

4.0 DESIGN OF EXPERIMENT: -

4.1 Apparatus Required: - Computer System with specifications

- **Window OS-10**
- **ORACLE-11g (software)**
- **6GB RAM**
- **64Bit ROM**

4.2 Experiment used: - Computer system with Peripherals

SQL CREATE DATABASE Statement

The SQL CREATE DATABASE Statement

The CREATE DATABASE statement is used to create a new SQL database.

Syntax
CREATE DATABASE *database name*;

CREATE DATABASE Example

The following SQL statement creates a database called "testDB":

Example

```
CREATE DATABASE testDB;
```

Tip: Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases with the following SQL command: SHOW DATABASES;

The SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax

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

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

Tip: For an overview of the available data types, go to our complete Data Types Reference.

SQL CREATE TABLE Example

The following example creates a table called "Persons" that contains five columns: Person ID, Last Name, First Name, Address, and City:

Example

```
CREATE TABLE Persons (
   Person ID int,
   Last Name varchar(25),
   First Name varchar(25),
   Address varchar(25),
   City varchar(20)
);
```

The SQL DROP TABLE Statement

The DROP TABLE statement is used to drop an existing table in a database.

Syntax

```
DROP TABLE table_name;
```

Note: Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!

SQL DROP TABLE Example

The following SQL statement drops the existing table "Shippers":

Example

```
DROP TABLE Shippers;
```

SQL TRUNCATE TABLE

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

Syntax

```
TRUNCATE TABLE table_name;
```

SQL ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype;
```

The following SQL adds an "Email" column to the "Customers" table:

Example

```
ALTER TABLE Customers
ADD Email varchar(255);
```

6.0 OBSERVATION: - Brief idea about SQL Use of join and group join functions, having clause

7.0 SAFETY & PRECAUTIONS: -

1. Turn off the computer and lights of the lab while leaving.
2. Do not temper or corrupt any file in data base
3. Do not eat chewing gum in the lab
4. Handle the electrical item carefully

8.0 REVIEW QUESTIONS: -

1. How to create database in SQL?
2. How to create tableinSQL?
3. How to drop table in sql?
4. What is the use of truncate clause?
5. What is the use of alter table?

EXPERIMENT-7

1.0 CONTEXT

1.1 Subject: - Database management System Lab

1.2 Experiment: - 7

1.3 Topic: - SQL

1.4 Sub-Topic: -sql arithmetic operator.

1.5 Title of the Experiment: - SQL arithmetic, bitwise, comparison operator.

1.6 Environment/Location: - Computer lab of IT Dept.

2.0 AIM OF THE EXPERIMENT: - **arithmetic, bitwise, comparison operator.**
3.0 OBJECTIVE OF THE EXPERIMENT: - To know how arithmetic, bitwise, comparison operator.

4.0 DESIGN OF EXPERIMENT: -

4.1 Apparatus Required: - Computer System with specifications

- **Window OS-10**
- **ORACLE-11g (software)**
- **6GB RAM**
- **64Bit ROM**

4.2 Experiment used: - Computer system with Peripherals

SQL Operators

SQL Arithmetic Operators

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo

SQL Bitwise Operators

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

SQL Comparison Operators

Operator	**Description**
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

SQL Compound Operators

Operator	**Description**
+=	Add equals
-=	Subtract equals
*=	Multiply equals

/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^-=	Bitwise exclusive equals
|*=	Bitwise OR equals

SQL Logical Operators

Operator	**Description**
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE

SOME	TRUE if any of the subquery values meet the condition

SQL Data Types

Each column in a database table is required to have a name and a data type.

An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

Note: Data types might have different names in different database. And even if the name is the same, the size and other details may be different! Always check the documentation!

MySQL Data Types (Version 8.0)

In MySQL there are three main data types: string, numeric, and date and time.

String data types:

Data type	**Description**
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The *size* parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes

BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

Numeric data types:

Data type	**Description**
BIT(*size*)	A bit-value type. The number of bits per value is specified in *size*. The *size* parameter can hold a value from 1 to 64. The default value for *size* is 1.
TINYINT(*size*)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(*size*)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255)

MEDIUMINT(*size*)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
INTEGER(*size*)	Equal to INT(size)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)
FLOAT(*size*, *d*)	A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(*p*)	A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter
DOUBLE PRECISION(*size*, *d*)	
DECIMAL(*size*, *d*)	An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0.
DEC(*size*, *d*)	Equal to DECIMAL(size,d)

Note: All the numeric data types may have an extra option: UNSIGNED or ZEROFILL. If you add the UNSIGNED option, MySQL disallows negative values for the column. If you add the ZEROFILL option, MySQL automatically also adds the UNSIGNED attribute to the column.

Date and Time data types:

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(*fsp*)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(*fsp*)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(*fsp*)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

6.0 OBSERVATION: - Brief idea about SQL Use of join and group join functions, having clause

7.0 SAFETY & PRECAUTIONS: -

1. Turn off the computer and lights of the lab while leaving.
2. Do not temper or corrupt any file in data base
3. Do not eat chewing gum in the lab
4. Handle the electrical item carefully

8.0 REVIEW QUESTIONS: -

1. What is the use of arithmetic in SQL?
2. What is the use of comparison inSQL?
3. What is the use of compound in SQL?
4. What is the use of logical clause?
5. What are the different types of data type in SQL?

ASSINGMENT FOR DBMS LAB-1

Q1 . Create a Employee table having attributes EMPNO NUMBER(4), ENAME VARCHAR2(10),JOB VARCHAR2(9),MGR NUMBER(4),HIREDATE DATE, SAL NUMBER(7, 2),COMM NUMBER(7, 2,DEPTNO NUMBER and inset data to the above Tables.

Q2. **Create a Department table having attributes** DEPTNO NUMBER (2), DNAME VARCHAR2 (14), LOC VARCHAR2 (13) AND inset data to the above Tables.

ASSINGMENT FOR DBMS LAB-II

1. Show the Structure of DEPT. Select all data from DEPT table. Create a query to display unique jobs from the EMP table.

2. Write a query to Name the column headings EMP#, Employee, Job and Hire date, respectively. Run the query.

3. Create a query to display the Name and salary of employees earning more than Rs.2850.Save the query and run it.

4. Create a query to display the employee name and department no. for employee no. 7566.

5. Display the employee name, job and start date of employees hire date between Feb.20.1981 and May 1, 1981. Order the query in ascending order of start date.

6. Display the name and title of all employees who don't have a Manager.

7. Display the name, salary and comm. For all employee who earn comm. Sort data in descending order of salary and comm.

8. Display the name job, salary for all employees whose job is Clerk or Analyst their salary is not equal to Rs.1000, Rs.3000, Rs.5000.

9. Write a query to display the date. Label the column DATE.

10. Create a unique listing of all jobs that are in department 30.

11. Write a query to display the name, department number and department name for all employees.

12. Write a query to display the employee name, department name, and location of all employee who earn a commission.

13. Write a query to display the name, job, department number and department name for all employees who works in DALLAS.

14. Write a query to display the number of people with the same job. Save the query @ run it.

15. Create a query to display the employee name and hire date for all employees in same department.

16. Display the employee name and salary of all employees who report to KING.

17. Display the mane, department name and salary of any employee whose salary and commission matches both the salary and commission of any employee located in DALLAS.

18. Create a student database table using create command using Regd. No as Primary Key , insert data of your class.

19. Delete the information of student having roll No -15 and City- Bhubaneswar. Rename the Student database table to STUDENT INFORMATION.

20. Practice of all Data Retrieval, DML, DDL, TCL and DCL commands used in Oracle by writing queries.