Laboratory Manual

Data Base Management System
(CS-502)

For

Third Year Students
Department: Computer Science & Engineering

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Department of Computer Science and Engineering

Vision of CSE Department:

The department envisions to nurture students to become technologically proficient, research competent and socially accountable for the welfare of the society.

Mission of the CSE Department:

I. To provide high quality education through effective teaching-learning process emphasizing active participation of students.

II. To build scientifically strong engineers to cater to the needs of industry, higher studies, research and startups.

III. To awaken young minds ingrained with ethical values and professional behaviors for the betterment of the society.

Program Educational Objectives:

Graduates will be able to

I. Our engineers will demonstrate application of comprehensive technical knowledge for innovation and entrepreneurship.

II. Our graduates will employ capabilities of solving complex engineering problems to succeed in research and/or higher studies.

III. Our graduates will exhibit team-work and leadership qualities to meet stakeholder business objectives in their careers.

IV. Our graduates will evolve in ethical and professional practices and enhance socioeconomic contributions to the society.

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Program Outcomes (POs):

Engineering Graduates will be able to:

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

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Course Outcomes
DBMS(CS-502)

CO1 :	Understand basic concepts and identify various data models (E-R modelling concepts) and apply these concepts for designing databases.
CO2 :	Apply SQL, relational database theory and describe relational algebra expression, tuple and domain relational expression for writing queries in relational algebra.
CO3 :	Identify and improve the database design by normalization, key constraints technique.
CO4 :	Analyze software and design the ER-diagram for it and apply the concept of PL/SQL, ANSI SQL.
CO5 :	Evaluate and optimize queries and transaction processes for solving real world problems.

Course	Course Outcomes	CO Attainment	PO1	PO2	PO3	PO4	PO5	PO 6	PO7	PO8	PO9	PO1	PO1	PO1	PSO	PSO	PSO
CO1	Understand basic concepts and identify various data models (E-R modelling concepts) and apply these concepts for designing databases		2	1	1	1	1										
CO2	Apply SQL, relational database theory and describe relational algebra expression, tuple and domain relational expression for writing queries in relational algebra.		1	1		1	1									2	
CO3	Identify and improve the database design by normalization, key constraints technique		2	2		1	1									1	
CO4	Analyze software and design the ER-diagram for it and apply the concept of PL/SQL , ANSI SQL.		2	1	1	1					1	2					
CO5	Evaluate and optimize queries and transaction processes for solving real world problems.		1	2		2	1				1	1	1				1

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

List of Program

S. No.	List	Course Outcome	Page No.
	SQL introduction	CO2	1-51
1	Delete duplicate row from the table.	CO2	52
2	Display the alternate row from table	CO2	53
3	Delete alternate row from table.	CO2	54
4	Update multiple rows in using single update statement.	CO2	55
5	Find the third highest paid and third lowest paid salary.	CO2	56
6	Display the 3rd, 4th, 9th rows from table.	CO2	57
7	Display the ename, which is start with j, k, l or m.	CO2	58
8	Show all employees who were hired the first half of the month.	CO2	59
9	Display the three records in the first row and two Records in the second row and one record in the Third row in a single sql statements.	CO2	60
10	Write a sql statements for rollback commit and save points.	CO2	61-65
11	Write a pl/sql for select, insert, update and delete statements.	CO4	66
12	Write a pl/sql block to delete a record. If delete operation is successful return 1 else return 0.	CO4	67-68
13	Display name, hire date of all employees using cursors.	CO4	69
14	Display details of first 5 highly paid employees using cursors.	CO4	70
15	Write a database trigger which fires if you try to insert, update, or delete after 7'o' clock.	CO4	71-73
16	Write a data base trigger, which acts just like primary key and does not allow duplicate values.	CO4	74
17	Create a data base trigger, which performs the action of the on delete cascade.	CO4	75
18	Write a data base trigger, which should not delete from emp table if the day is Sunday.	CO4	76
19	In this subject the students are supposed to prepare a small database application in complete semester like financial accounting system, Railway reservation system, institute timetable management system. Student record system, library management system, hospital management system etc. In RDBMS as follows: **Section A:** Solving the case studies using ER data model (Design of the database) **Section B:** Implement a mini project for the problem taken in Section A	CO1, CO2, CO3, CO4, CO5	

SQL Introduction

Pronounced as SEQUEL: Structured English QUERY Language

- Pure non-procedural query language
- Designed and developed by IBM, Implemented by Oracle
- 1978 System/R IBM- 1st Relational DBMS
- 1979 Oracle and Ingres
- 1982 SQL/DS and DB2 IBM
- Accepted by both ANSI + ISO as Standard Query Language for any RDBMS
- SQL86 (SQL1) : first by ANSI and ratified by ISO (SQL-87), minor revision on 89 (SQL-89)
- SQL92 (SQL2) : major revision
- SQL99 (SQL3) : add recursive query, trigger, some OO features, and non-scholar type
- SQL2003 : XML, Window functions, and sequences (Not free)
- Supports all the three sublanguages of DBMS: DDL, DML, DCL
- Supports Aggregate functions, String Manipulation functions, Set theory operations, Date Manipulation functions, rich set of operators (IN, BETWEEN, LIKE, IS NULL, EXISTS)
- Supports REPORT writing features and Forms for designing GUI based applications

DATA DEFINITION, CONSTRAINTS, AND SCHEMA CHANGES
Used to CREATE, ALTER, and DROP the descriptions of the database tables (relations)

Data Definition in SQL
CREATE, ALTER and DROP
table……relation
row……tuple
column……attribute

DATA TYPES

- Numeric: NUMBER, NUMBER(s,p), INTEGER, INT, FLOAT, DECIMAL
- Character: CHAR(n), VARCHAR(n), VARCHAR2(n), CHAR VARYING(n)
- Bit String: BLOB, CLOB
- Date and Time: DATE (YYYY-MM-DD) TIME(HH:MM:SS)
- Timestamp: DATE + TIME
- USER Defined types

LAB REQUIREMENTS

For DBMS Lab implementation:

- MySQL version 5.1-8.0
- 2 GB RAM
- 800 MB Minimum Disk Space

Learn the Data Definition Language (DDL) commands in RDBMS, Data Manipulation Language (DML) and Data Control Language (DCL).

DDL Commands:

1. The Create Table Command: - it defines each column of the table uniquely. Each
column has minimum of three attributes, a name, data type and size.
Syntax:-Create table <table name> (<col1> <datatype>(<size>), <col2> <datatype><size>));

Ex:-create table emp(empno number(4) primary key, ename char(10));

2. Modifying the structure of tables.

a) Add new columns
Syntax:-Alter table <tablename> add(<new col><datatype(size),<new col>datatype(size));

Ex:-alter table emp add(sal number(7,2));

3. **Dropping a column from a table.**
Syntax:-Alter table <tablename> drop column <col>;

Ex:-alter table emp drop column sal;

4. Modifying existing columns.
Syntax:-Alter table <tablename> modify(<col><newdatatype>(<newsize>));

Ex:-alter table emp modify(ename varchar2(15));

5. Renaming the tables
Syntax:-Rename <oldtable> to <new table>;

Ex:-rename emp to emp1;

6. Truncating the tables.
Syntax:-Truncate table <tablename>;

Ex:-trunc table emp1;

7. Destroying tables.
Syntax:-Drop table <tablename>;

Ex:-drop table emp;

DML commands:

1. Inserting Data into Tables: - once a table is created the most natural thing to do is load
this table with data to be manipulated later.
Syntax:-insert into <tablename> (<col1>,<col2>) values(<exp>,<exp>);

2. Delete operations.
a) Remove all rows
Syntax:-delete from <tablename>;

b) Removal of a specified row/s
Syntax:-delete from <tablename> where <condition>;

3. Updating the contents of a table.
a) Updating all rows
Syntax:-Update <tablename> set <col>=<exp>,<col>=<exp>;

b) Updating selected records.
Syntax:-Update <tablename> set <col>=<exp>,<col>=<exp> where <condition>;

4. Types of data constrains.
a) Not null constraint at column level.
Syntax: -<col><datatype>(size)not null

b) unique constraint
Syntax:-Unique constraint at column level.
<col><datatype>(size)unique;

c) unique constraint at table level:
Syntax:-Create table
tablename(col=format,col=format,unique(<col1>,<col2>);

d) Primary key constraint at column level
Syntax:-<col><datatype>(size)primary key;

e) Primary key constraint at table level.
Syntax:-Create table tablename(col=format,col=format Primary key(col1>,<col2>);

f) Foreign key constraint at column level.
Syntax:-<col><datatype>(size>) references <tablename>[<col>];

g) Foreign key constraint at table level
Syntax:-foreign key(<col>[,<col>])references <tablename>[(<col>,<col>)

h) Check constraint
check constraint constraint at column level.
Syntax:-<col><datatype>(size) check(<logical expression>)

i) Check constraint constraint at table level.
Syntax:-check(<logical expression>)

DCL commands:

Oracle provides extensive feature in order to safeguard information stored in its tables from unauthorized viewing and damage. The rights that allow the user of some or all oracle resources on the server are called privileges.
a) Grant privileges using the GRANT statement
The grant statement provides various types of access to database objects such as tables, views and sequences and so on.
Syntax:-GRANT <object privileges>
ON <objectname>
TO<username>
[WITH GRANT OPTION];

b) Revoke permissions using the REVOKE statement:
The REVOKE statement is used to deny the Grant given on an object.
Syntax:-REVOKE<object privilege> ON FROM<user name>;
CREATE DATABASE <DB_NAME>;
Example for creating Database
CREATE DATABASE Test;

The above command will create a database named Test, which will be an empty schema without any table.

To create tables in this newly created database, we can again use the create command.

Creating a Table

create command can also be used to create tables. Now when we create a table, we have to specify the details of the columns of the tables too. We can specify the names and datatypes of various columns in the create command itself.

Following is the syntax,

```
CREATE TABLE <TABLE_NAME>
(
   column_name1 datatype1,
   column_name2 datatype2,
   column_name3 datatype3,
   column_name4 datatype4
);
```

create table command will tell the database system to create a new table with the given table name and column information.

Example for creating Table

```
CREATE TABLE Student(
   student_id INT,
   name VARCHAR(100),
   age INT);
```

The above command will create a new table with name Student in the current database with 3 columns, namely student_id, name and age. Where the column student_id will only store integer, name will hold upto 100 characters and age will again store only integer value.

If you are currently not logged into your database in which you want to create the table then you can also add the database name along with table name, using a dot operator.

For example, if we have a database with name Test and we want to create a table Student in it, then we can do so using the following query:

```
CREATE TABLE Test.Student(
   student_id INT,
   name VARCHAR(100),
   age INT);
```

SQL: ALTER command

Alter command is used for altering the table structure, such as,

- To add a column to existing table
- To rename any existing column
- To change datatype of any column or to modify its size.

- To drop a column from the table.

ALTER Command: Add a new Column

Using ALTER command, we can add a column to any existing table.
Here is an Example for this,

```
ALTER TABLE student ADD(
   address VARCHAR(200)
);
```

The above command will add a new column address to the table student, which will hold data of type varchar which is nothing but string, of length 200.

ALTER Command: Add multiple new Columns

Using ALTER command we can even add multiple new columns to any existing table. Following is the syntax,

```
ALTER TABLE student ADD(
   father_name VARCHAR(60),
   mother_name VARCHAR(60),
   dob DATE);
```

The above command will add three new columns to the student table

ALTER Command: Add Column with default value

ALTER command can add a new column to an existing table with a default value too. The default value is used when no value is inserted in the column. Following is the syntax,

```
ALTER TABLE table_name ADD(
   column-name1 datatype1 DEFAULT some_value
);
```

Here is an Example for this,

```
ALTER TABLE student ADD(
   dob1 DATE DEFAULT '1999-08-01'
);
```

The above command will add a new column with a preset default value to the table student.

ALTER Command: Modify an existing Column

ALTER command can also be used to modify data type of any existing column. Following is the syntax,

```
ALTER TABLE table_name modify(
   column_name datatype
);
```

Here is an Example for this,

```
ALTER TABLE student MODIFY
   address varchar(300);
```

Remember we added a new column address in the beginning? The above command will modify the address column of the student table, to now hold upto 300 characters.

ALTER Command: Rename a Column

Using ALTER command you can rename an existing column. Following is the syntax,

```
alter table student change  column address loc char(20);
```

```
alter table tq1 modify id int auto_increment primary key;
```

ALTER Command: Drop a Column

ALTER command can also be used to drop or remove columns. Following is the syntax,

ALTER TABLE table_name DROP(
column_name);
Here is an example for this,

ALTER TABLE student DROP dob;
The above command will drop the address column from the table student.

Alter table to add primary key (email);

SQL Truncate, Drop or Rename a Table

TRUNCATE command
TRUNCATE command removes all the records from a table.
But this command will not destroy the table's structure.
When we use TRUNCATE command on a table its (auto-increment) primary key is also initialized. Following is its syntax,

TRUNCATE TABLE table_name
Here is an example explaining it,

TRUNCATE TABLE student;
The above query will delete all the records from the table student.

In DML commands, we will study about the DELETE command which is also more or less same as the TRUNCATE command.

DROP command
DROP command completely removes a table from the database.
This command will also destroy the table structure
and the data stored in it. Following is its syntax,

DROP TABLE table_name
Here is an example explaining it,

DROP TABLE student;

The above query will delete the student table completely. It can also be used on Databases, to delete the complete database.

For example, to drop a database

DROP DATABASE Test
The above query will drop the database with name Test from the system.

RENAME
RENAME command is used to set a new name for any
existing table. Following is the syntax,

RENAME TABLE old_table_name to new_table_name
Here is an example explaining it.

RENAME TABLE student to students_info;
The above query will rename the table student to students_info.

DML command

Using INSERT SQL command

Data Manipulation Language (DML) statements are used for managing data in database. DML commands are not auto-committed. It means changes made by DML command are not permanent to database, it can be rolled back.

Talking about the Insert command, whenever we post a Tweet on Twitter,
the text is stored in some table, and as we post a new tweet,
a new record gets inserted in that table.

INSERT command
Insert command is used to insert data into a table.
Following is its general syntax,

load data local infile 'E:/dbms/a.txt' into table tw;

INSERT INTO table_name VALUES(data1, data2, ...)
Lets see an example,

Consider a table student with the following fields.

s_id name age
INSERT INTO student VALUES(101, 'Adam', 15);
The above command will insert a new record into student table.

s_id name age
101 Adam 15

Insert value into only specific columns
We can use the INSERT command to insert values for only some specific columns of a row. We can specify the column names along with the values to be inserted like this

INSERT INTO student(id, name) values(102, 'Alex');
The above SQL query will only insert id and name values in the newly inserted record.

Insert NULL value to a column
Both the statements below will insert NULL value into age column of the student table.

INSERT INTO student(id, name) values(102, 'Alex');
Or,
INSERT INTO Student VALUES(102,'Alex', null);
The above command will insert only two column values and the other column is set to null.

S_id S_Name age
101 Adam 15
102 Alex

Insert Default value to a column
INSERT INTO Student VALUES(103,'Chris', default)
S_id S_Name age
101 Adam 15
102 Alex
103 chris 14
Suppose the column age in our tabel has a default value of 14.

Also, if you run the below query, it will insert default value into the age column, whatever the default value may be.

INSERT INTO Student VALUES(103,'Chris')
Using UPDATE SQL command

Auto_increment
create table ta1(id int primary key AUTO_INCREMENT);

Let's take an example of a real-world problem.
These days, Facebook provides an option for Editing
your status update, how do you think it works? Yes,
using the Update SQL command.

Let's learn about the syntax and usage of the UPDATE command.

UPDATE command
UPDATE command is used to update any record of data in a table.
Following is its general syntax,

UPDATE table_name SET column_name = new_value WHERE some_condition;
WHERE is used to add a condition to any SQL query, we will soon study about it in detail.
Lets take a sample table student,

student_id name age
101 Adam 15
102 Alex
103 chris 14

UPDATE student SET age=18 WHERE student_id=102;
S_id S_Name age
101 Adam 15
102 Alex 18
103 chris 14
In the above statement, if we do not use the WHERE clause, then our update query will update age for all the columns of the table to 18.

Updating Multiple Columns
We can also update values of multiple columns using a single UPDATE statement.

UPDATE student SET name='Abhi', age=17 where s_id=103;
The above command will update two columns of the record which has s_id 103.

s_id	name	age
101	Adam	15
102	Alex	18
103	Abhi	17

UPDATE Command: Incrementing Integer Value
When we have to update any integer value in a table, then we can fetch and update the value in the table in a single statement.

For example, if we have to update the age column of student table every year for every student, then we can simply run the following UPDATE statement to perform the following operation:

UPDATE student SET age = age+1;
As you can see, we have used age = age + 1 to increment the value of age by 1.

NOTE: This style only works for integer values.

DELETE command
DELETE command is used to delete data from a table.

Following is its general syntax,

DELETE FROM table_name;
Let's take a sample table student:

s_id	name	age
101	Adam	15
102	Alex	18
103	Abhi	17

Delete all Records from a Table
DELETE FROM student;
The above command will delete all the records from the table student.

Delete a particular Record from a Table
In our student table if we want to delete a single record, we can use the WHERE clause to provide a condition in our DELETE statement.

DELETE FROM student WHERE s_id=103;
The above command will delete the record where s_id is 103 from the table student.

S_id	S_Name	age
101	Adam	15
102	Alex	18

Commit, Rollback and Savepoint SQL commands

Transaction Control Language(TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table
by DML statements. It also allows statements to be grouped together into logical transactions.

COMMIT command
COMMIT command is used to permanently save any
transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Following is commit command's syntax,

COMMIT;

ROLLBACK command
This command restores the database to last committed state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

Following is rollback command's syntax,

ROLLBACK TO savepoint_name;

SAVEPOINT command
SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Following is savepoint command's syntax,

SAVEPOINT savepoint_name;
In short, using this command we can name the
different states of our data in any table and then
rollback to that state using the ROLLBACK command
whenever required.

Example :
CREATE TABLE customer (a INT, b CHAR (20), INDEX (a));

Query OK, 0 rows affected (0.00 sec)
mysql> -- Do a transaction with autocommit turned on.
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (10, 'Heikki');
Query OK, 1 row affected (0.00 sec)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
mysql> -- Do another transaction with autocommit turned off.
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (15, 'John');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO customer VALUES (20, 'Paul');
Query OK, 1 row affected (0.00 sec)

```
mysql> DELETE FROM customer WHERE b = 'Heikki';
Query OK, 1 row affected (0.00 sec)
mysql> -- Now we undo those last 2 inserts and the delete.
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT * FROM customer;

+------+--------+
| a    | b      |
+------+--------+
|   10 | Heikki |
+------+--------+
1 row in set (0.00 sec)
```

Using Savepoint and Rollback
Following is the table class,

id	name
1	Abhi
2	Adam
4	Alex

Lets use some SQL queries on the above table and see the results.

```
INSERT INTO class VALUES(5, 'Rahul');
COMMIT;
UPDATE class SET name = 'Abhijit' WHERE id = '5';
SAVEPOINT A;
INSERT INTO class VALUES(6, 'Chris');
SAVEPOINT B;
INSERT INTO class VALUES(7, 'Bravo');
SAVEPOINT C;
SELECT * FROM class;
```

NOTE: SELECT statement is used to show the data stored in the table.
The resultant table will look like,

id	name
1	Abhi
2	Adam
4	Alex

5	Abhijit
6	Chris
7	Bravo

use the ROLLBACK command to roll back the state of data to the savepoint B.

ROLLBACK TO B;

SELECT * FROM class;
Now our class table will look like

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris

Again, use the ROLLBACK command to roll back the state of data to the savepoint A

ROLLBACK TO A;

SELECT * FROM class;
Now the table will look like,

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit

So now you know how the commands COMMIT, ROLLBACK and SAVEPOINT works.

Using GRANT and REVOKE

Data Control Language(DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,

System: This includes permissions for creating session, table, etc and all types of other system privileges.

Object: This includes permissions for any command or query to perform any operation on the database tables.
In DCL we have two commands,

GRANT: Used to provide any user access privileges or other priviliges for the database.
REVOKE: Used to take back permissions from any user.

Allow a User to create session
When we create a user in SQL, it is not even allowed to login and create a session until and unless proper permissions/priviliges are granted to the user.

Following command can be used to grant the session creating priviliges.

GRANT CREATE SESSION TO username;

Allow a User to create table
To allow a user to create tables in the database, we can use the below command,

GRANT CREATE TABLE TO username;

Provide user with space on tablespace to store table
Allowing a user to create table is not enough to start storing data in that table. We also must provide the user with priviliges to use the available tablespace for their table and data.

ALTER USER username QUOTA UNLIMITED ON SYSTEM;
The above command will alter the user details and will provide it access to unlimited tablespace on system.

NOTE: Generally unlimited quota is provided to Admin users.

Grant all privilege to a User
sysdba is a set of priviliges which has all the permissions in it. So if we want to provide all the privileges to any user, we can simply grant them the sysdba permission.

GRANT sysdba TO username
Grant permission to create any table

Sometimes user is restricted from creating come tables with names which are reserved for system tables. But we can grant privileges to a user to create any table using the below command,

GRANT CREATE ANY TABLE TO username
Grant permission to drop any table
As the title suggests, if you want to allow user to drop any table from the database, then grant this privilege to the user,

GRANT DROP ANY TABLE TO username

To take back Permissions
And, if you want to take back the privileges from any user, use the REVOKE command.

REVOKE CREATE TABLE FROM username

Using the WHERE SQL clause

WHERE clause is used to specify/apply any condition while retrieving, updating or deleting data from a table. This clause is used mostly with SELECT, UPDATE and DELETE query.

When we specify a condition using the WHERE clause then the query executes only for those records for which the condition specified by the WHERE clause is true.

Syntax for WHERE clause
Here is how you can use the WHERE clause with a DELETE statement, or any other statement,

DELETE FROM table_name WHERE [condition];
The WHERE clause is used at the end of any SQL query, to specify a condition for execution.

Time for an Example
Consider a table student,

s_id	name	age	address
101	Adam	15	Chennai

102	Alex	18	Delhi
103	Abhi	17	Banglore
104	Ankit	22	Mumbai

Now we will use the SELECT statement to display data of the table, based on a condition, which we will add to our SELECT query using WHERE clause.

Let's write a simple SQL query to display the record for student with s_id as 101.

```
SELECT s_id, name, age, address FROM student WHERE s_id = 101;
```

Following will be the result of the above query.

s_id	name	age	address
101	Adam	15	Noida

Applying condition on Text Fields

In the above example we have applied a condition to an integer value field, but what if we want to apply the condition on name field. In that case we must enclose the value in single quote ' '. Some databases even accept double quotes, but single quotes is accepted by all.

```
SELECT s_id, name, age, address FROM student WHERE name = 'Adam';
```

Following will be the result of the above query.

s_id	name	age	address
101	Adam	15	Noida

Operators for WHERE clause condition

Following is a list of operators that can be used while specifying the WHERE clause condition.

Operator	Description
=	Equal to
!=	Not Equal to
<	Less than

> Greater than
<= Less than or Equal to
>= Greate than or Equal to
BETWEEN: Between a specified range of values
LIKE: This is used to search for a pattern in value.
IN In a given set of values

SQL LIKE clause

LIKE clause is used in the condition in SQL query with the WHERE clause. LIKE clause compares data with an expression using wildcard operators to match pattern given in the condition.

Wildcard operators
There are two wildcard operators that are used in LIKE clause.

Percent sign %: represents zero, one or more than one character.
Underscore sign _: represents only a single character.

Example of LIKE clause
Consider the following Student table.

s_id	s_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

SELECT * FROM Student WHERE s_name LIKE 'A%';

The above query will return all records where s_name starts with character 'A'.

s_id	s_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

Using _ and %

SELECT * FROM Student WHERE s_name LIKE '_d%';
The above query will return all records from Student table where s_name

contain 'd' as second character.

s_id	s_Name	age
101	Adam	15

Using % only

SELECT * FROM Student WHERE s_name LIKE '%x';

The above query will return all records from Student table where s_name contain 'x' as last character.

s_id	s_Name	age
102	Alex	18

SQL ORDER BY Clause

Order by clause is used with SELECT statement for arranging retrieved data in sorted order. The Order by clause by default sorts the retrieved data in ascending order. To sort the data in descending order DESC keyword is used with Order by clause.

Syntax of Order By

SELECT column-list|* FROM table-name ORDER BY ASC | DESC;

Using default Order by

Consider the following Emp table,

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SELECT * FROM Emp ORDER BY salary;

The above query will return the resultant data in ascending order of the salary.

eid	name	age	salary
403	Rohan	34	6000
402	Shane	29	8000
405	Tiger	35	8000
401	Anu	22	9000

404 Scott 44 10000
Using Order by DESC
Consider the Emp table described above,

SELECT * FROM Emp ORDER BY salary DESC;
The above query will return the resultant data in descending order of the salary.

eid	name	age	salary
404	Scott	44	10000
401	Anu	22	9000
405	Tiger	35	8000
402	Shane	29	8000
403	Rohan	34	6000

SQL Group By Clause

Group by clause is used to group the results of a SELECT query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

Syntax for using Group by in a statement.

```
SELECT column_name, function(column_name)
FROM table_name
WHERE condition
GROUP BY column_name
```

Example of Group by in a Statement
Consider the following Emp table.

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	9000
405	Tiger	35	8000

Here we want to find name and age of employees grouped by their salaries or in other words,
we will be grouping employees based on their salaries,
hence, as a result, we will get a data set, with unique salaries listed, along side the first employee's name and age to

have that salary. Hope you are getting the point here!

group by is used to group different row of data together based on any one column.

SQL query for the above requirement will be,

```
SELECT name, age
FROM Emp GROUP BY salary
```

Result will be,

name	age
Rohan	34
Shane	29
Anu	22

Example of Group by in a Statement with WHERE clause
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	9000
405	Tiger	35	8000

SQL query will be,

```
SELECT name, salary
FROM Emp
WHERE age > 25
GROUP BY salary
```

Result will be.

name	salary
Rohan	6000
Shane	8000
Scott	9000

You must remember that Group By clause will always come at the end of the SQL query, just like the Order by clause.

SQL HAVING Clause

Having clause is used with SQL Queries to give more precise condition for a statement.
It is used to mention condition in Group by based SQL queries,
just like WHERE clause is used with SELECT query.

Syntax for HAVING clause is,

```
SELECT column_name, function(column_name)
FROM table_name
WHERE column_name condition
GROUP BY column_name
HAVING function(column_name) condition
```

Example of SQL Statement using HAVING
Consider the following Sale table.

oid	order_name	previous_balance	customer
11	ord1	2000	Alex
12	ord2	1000	Adam
13	ord3	2000	Abhi
14	ord4	1000	Adam
15	ord5	2000	Alex

Suppose we want to find the customer whose previous_balance sum is more than 3000.

We will use the below SQL query,

```
SELECT *
FROM sale GROUP BY customer
HAVING sum(previous_balance) > 3000
```

Result will be,

oid	order_name	previous_balance	customer
11	ord1	2000	Alex

The main objective of the above SQL query was to find out the name of the customer who has had a previous_balance more than 3000, based on all the previous sales made to the customer, hence we get the first row in the table for customer Alex.

DISTINCT keyword

The distinct keyword is used with SELECT statement to retrieve unique values from the table. Distinct removes all the duplicate records while retrieving records from any table in the database.

Syntax for DISTINCT Keyword
SELECT DISTINCT column-name FROM table-name;

Example using DISTINCT Keyword
Consider the following Emp table. As you can see in the table below, there is employee name, along with employee salary and age.

In the table below, multiple employees have the same salary, so we will be using DISTINCT keyword to list down distinct salary amount, that is currently being paid to the employees.

eid	name	age	salary
401	Anu	22	5000
402	Shane	29	8000
403	Rohan	34	10000
404	Scott	44	10000
405	Tiger	35	8000

SELECT DISTINCT salary FROM Emp;
The above query will return only the unique salary from Emp table.

salary
5000
8000
10000

SQL AND & OR operator

The AND and OR operators are used with the WHERE clause to make more precise conditions for fetching data from database by combining more than one condition together.

AND operator

AND operator is used to set multiple conditions with the WHERE clause, alongside, SELECT, UPDATE or DELETE SQL queries.

Example of AND operator
Consider the following Emp table

eid	name	age	salary
401	Anu	22	5000
402	Shane	29	8000
403	Rohan	34	12000
404	Scott	44	10000
405	Tiger	35	9000

SELECT * FROM Emp WHERE salary < 10000 AND age > 25

The above query will return records where salary is less than 10000 and age greater than 25. Hope you get the concept here. We have used the AND operator to specify two conditions with WHERE clause.

eid	name	age	salary
402	Shane	29	8000
405	Tiger	35	9000

OR operator
OR operator is also used to combine multiple conditions with WHERE clause. The only difference between AND and OR is their behaviour.

When we use AND to combine two or more than two conditions, records satisfying all the specified conditions will be there in the result.

But in case of OR operator, atleast one condition from the conditions specified must be satisfied by any record to be in the resultset.

Example of OR operator
Consider the following Emp table

eid	name	age	salary
401	Anu	22	5000
402	Shane	29	8000
403	Rohan	34	12000
404	Scott	44	10000

405 Tiger 35 9000
SELECT * FROM Emp WHERE salary > 10000 OR age > 25

The above query will return records where either salary is greater than 10000 or age is greater than 25.

402 Shane 29 8000
403 Rohan 34 12000
404 Scott 44 10000
405 Tiger 35 9000

Division Operator in SQL

The division operator is used when we have to evaluate queries which contain the keyword ALL.

Some instances where division operator is used are:

Which person has account in all the banks of a particular city?
Which students have taken all the courses required to graduate?
In above specified problem statements, the description after the keyword 'all' defines a set which contains some elements and the final result contains those units which satisfy these requirements.

Another way how you can identify the usage of division operator is by using the logical implication of if...then. In context of the above two examples, we can see that the queries mean that,

If there is a bank in that particular city, that person must have an account in that bank.If there is a course in the list of courses required to be graduated, that person must have taken that course.

We shall see the second example, mentioned above, in detail.

SQL Constraints

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraints can be divided into the following two types,

Column level constraints: Limits only column data.
Table level constraints: Limits whole table data.

Constraints are used to make sure that the integrity of data is maintained in the database.
Following are the most used constraints that can be applied to a table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NOT NULL Constraint

NOT NULL constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column.
It enforces a column to contain a proper value.

One important point to note about this constraint is that it cannot be defined at table level.
Example using NOT NULL constraint

```
CREATE TABLE Student(s_id int NOT NULL, Name varchar(60), Age int);
```

The above query will declare that the s_id field of Student table will not take NULL value.

UNIQUE Constraint

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data. This constraint can be applied at column level or table level.

Using UNIQUE constraint when creating a Table (Table Level)
Here we have a simple CREATE query to create a table, which will have a column s_id with unique values.

CREATE TABLE Student(s_id int NOT NULL UNIQUE, Name varchar(60), Age int);
The above query will declare that the s_id field of Student table will only have unique values and wont take NULL value.

Using UNIQUE constraint after Table is created (Column Level)
ALTER TABLE Student ADD UNIQUE(s_id);
The above query specifies that s_id field of Student table will only have unique value.

Primary Key Constraint

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

Using PRIMARY KEY constraint at Table Level
CREATE table Student (s_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int);
The above command will creates a PRIMARY KEY on the s_id.

Using PRIMARY KEY constraint at Column Level
ALTER table Student ADD PRIMARY KEY (s_id);
The above command will creates a PRIMARY KEY on the s_id.

Foreign Key Constraint

FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand

FOREIGN KEY, let's see its use, with help of the below tables:

Customer_Detail Table

c_id	Customer_Name	address
101	Adam	Noida
102	Alex	Delhi
103	Stuart	Rohtak

Order_Detail Table

Order_id	Order_Name	c_id
10	Order1	101
11	Order2	103
12	Order3	102

In Customer_Detail table, c_id is the primary key which is set as foreign key in Order_Detail table.
The value that is entered in c_id which is set as foreign key in Order_Detail table must be present in Customer_Detail table where it is set as primary key. This prevents invalid data to be inserted into c_id column of Order_Detail table.

If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data.

Using FOREIGN KEY constraint at Table Level

```
CREATE table Order_Detail(
   order_id int PRIMARY KEY,
   order_name varchar(60) NOT NULL,
   c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id)
);
```

In this query, c_id in table Order_Detail is made as foriegn key, which is a reference of c_id column in Customer_Detail table.

Using FOREIGN KEY constraint at Column Level

```
ALTER table Order_Detail ADD FOREIGN KEY (c_id) REFERENCES
Customer_Detail(c_id);
```

Behaviour of Foriegn Key Column on Delete

There are two ways to maintin the integrity of data in Child table, when a particular record is deleted in the main table. When two tables are connected

with Foriegn key, and certain data in the main table is deleted, for which a record exits in the child table, then we must have some mechanism to save the integrity of data in the child table.

foriegn key behaviour on delete - cascade and Null

On Delete Cascade : This will remove the record from child table, if that value of foriegn key is deleted from the main table.
On Delete Null : This will set all the values in that record of child table as NULL, for which the value of foriegn key is deleted from the main table.
If we don't use any of the above, then we cannot delete data from the main table for which data in child table exists. We will get an error if we try to do so.
ERROR : Record in child table exist

CHECK Constraint

CHECK constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

Using CHECK constraint at Table Level

```
CREATE table Student(
   s_id int NOT NULL CHECK(s_id > 0),
   Name varchar(60) NOT NULL,
   Age int
);
```

The above query will restrict the s_id value to be greater than zero.

Using CHECK constraint at Column Level

```
ALTER table Student ADD CHECK(s_id > 0);
```

SQL Functions

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two categories,

- Aggregate Functions
- Scalar Functions

Aggregate Functions

These functions return a single value after performing calculations on a group of values. Following are some of the frequently used Aggregrate functions.

AVG() Function

Average returns average value after calculating it from values in a numeric column.

Its general syntax is,

SELECT AVG(column_name) FROM table_name
Using AVG() function
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find average salary will be,

SELECT avg(salary) from Emp;
Result of the above query will be,

avg(salary)
8200

COUNT() Function

Count returns the number of rows present in the table either based on some condition or without condition.

Its general syntax is,

SELECT COUNT(column_name) FROM table-name

Using COUNT() function
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to count employees, satisfying specified condition is,

SELECT COUNT(name) FROM Emp WHERE salary = 8000;
Result of the above query will be,

count(name)
2
Example of COUNT(distinct)
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query is,

SELECT COUNT(DISTINCT salary) FROM emp;
Result of the above query will be,

count(distinct salary)
4

FIRST() Function
First function returns first value of a selected column

Syntax for FIRST function is,

SELECT FIRST(column_name) FROM table-name;
Using FIRST() function
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query will be,

SELECT FIRST(salary) FROM Emp;
and the result will be,

first(salary)
9000

LAST() Function
LAST function returns the return last value of the selected column.

Syntax of LAST function is,

SELECT LAST(column_name) FROM table-name;
Using LAST() function
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query will be,

SELECT LAST(salary) FROM emp;
Result of the above query will be,

last(salary)
8000

MAX() Function
MAX function returns maximum value from selected column of the table.

Syntax of MAX function is,

SELECT MAX(column_name) from table-name;
Using MAX() function
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find the Maximum salary will be,

SELECT MAX(salary) FROM emp;
Result of the above query will be,

MAX(salary)
10000

MIN() Function
MIN function returns minimum value from a selected column of the table.

Syntax for MIN function is,

SELECT MIN(column_name) from table-name;
Using MIN() function
Consider the following Emp table,

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find minimum salary is,

SELECT MIN(salary) FROM emp;
Result will be,

MIN(salary)
6000

SUM() Function
SUM function returns total sum of a selected columns numeric values.

Syntax for SUM is,

SELECT SUM(column_name) from table-name;
Using SUM() function
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	Scott	44	10000
405	Tiger	35	8000

SQL query to find sum of salaries will be,

SELECT SUM(salary) FROM emp;
Result of above query is,

SUM(salary)
41000

Scalar Functions
Scalar functions return a single value from an input value. Following are some frequently used Scalar Functions in SQL.

UCASE() Function
UCASE function is used to convert value of string column to Uppercase characters.

Syntax of UCASE,

SELECT UCASE(column_name) from table-name;
Using UCASE() function
Consider the following Emp table

eid	name	age	salary
401	anu	22	9000
402	shane	29	8000
403	rohan	34	6000
404	scott	44	10000
405	Tiger	35	8000

SQL query for using UCASE is,

SELECT UCASE(name) FROM emp;
Result is,

UCASE(name)
ANU
SHANE
ROHAN
SCOTT
TIGER

LCASE() Function
LCASE function is used to convert value of string columns to Lowecase characters.

Syntax for LCASE is,

SELECT LCASE(column_name) FROM table-name;
Using LCASE() function
Consider the following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000
403	Rohan	34	6000
404	SCOTT	44	10000
405	Tiger	35	8000

SQL query for converting string value to Lower case is,

SELECT LCASE(name) FROM emp;
Result will be,

LCASE(name)
anu
shane
rohan
scott
tiger

MID() Function
MID function is used to extract substrings from column values of string type in a table.

Syntax for MID function is,

SELECT MID(column_name, start, length) from table-name;
Using MID() function
Consider the following Emp table

eid	name	age	salary
401	anu	22	9000
402	shane	29	8000
403	rohan	34	6000
404	scott	44	10000
405	Tiger	35	8000

SQL query will be,

SELECT MID(name,2,2) FROM emp;
Result will come out to be,

MID(name,2,2)
nu
ha
oh
co
ig

ROUND() Function

ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values.

Syntax of Round function is,

SELECT ROUND(column_name, decimals) from table-name;
Using ROUND() function
Consider the following Emp table

eid	name	age	salary
401	anu	22	9000.67
402	shane	29	8000.98
403	rohan	34	6000.45
404	scott	44	10000
405	Tiger	35	8000.01

SQL query is,

SELECT ROUND(salary) from emp;
Result will be,

ROUND(salary)
9001
8001
6000
10000
8000

SQL JOIN

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. It is used for combining column from two or more tables by using values common to both tables.

JOIN Keyword is used in SQL queries for joining two or more tables.
Minimum required condition for joining table, is (n-1) where n, is number of tables.
A table can also join to itself, which is known as, Self Join.

Types of JOIN

Following are the types of JOIN that we can use in SQL:

- Inner
- Outer
- Left
- Right
- Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the
first table with each row of the second table.

Cross JOIN Syntax is,

SELECT column-name-list
FROM
table-name1 CROSS JOIN table-name2;
Example of Cross JOIN
Following is the class table,

ID	NAME
1	abhi
2	adam
4	alex

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Cross JOIN query will be,

SELECT * FROM
class CROSS JOIN class_info;
The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	1	DELHI

4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI
4	alex	2	MUMBAI
1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

As you can see, this join returns the cross product of all the records present in both the tables.

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Inner Join Syntax is,

SELECT column-name-list FROM table-name1 INNER JOIN table-name2 WHERE table-name1.column-name = table-name2.column-name;

Example of INNER JOIN

Consider a class table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Inner JOIN query will be,

SELECT * from class INNER JOIN class_info where class.id = class_info.id;

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI

2	adam	2	MUMBAI
3	alex	3	CHENNAI

Natural JOIN

Natural Join is a type of Inner join which is based on
column having same name and same datatype present in both the tables to be joined.

The syntax for Natural Join is,

SELECT * FROM
table-name1 NATURAL JOIN table-name2;

Example of Natural JOIN

Here is the class table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Natural join query will be,

SELECT * from class NATURAL JOIN class_info;

The resultset table will look like,

ID	NAME	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

In the above example, both the tables being joined have ID column (same name and same datatype), hence the records for which value of ID matches in both the tables will be the result of Natural Join of these two tables.

OUTER JOIN

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

- Left Outer Join
- Right Outer Join
- Full Outer Join

LEFT Outer Join

The left outer join returns a resultset table with the matched data from the two tables and then the remaining rows of the left table and null from the right table's columns.

Syntax for Left Outer Join is,

SELECT column-name-list FROM
table-name1 LEFT OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;
To specify a condition, we use the ON keyword with Outer Join.

Left outer Join Syntax for Oracle is,

SELECT column-name-list FROM
table-name1, table-name2 on table-name1.column-name = table-name2.column-name(+);
Example of Left Outer Join
Here is the class table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI

3 CHENNAI
7 NOIDA
8 PANIPAT

Left Outer Join query will be,

SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id = class_info.id);

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null

RIGHT Outer Join

The right outer join returns a resultset table with the matched data from the two tables being joined, then the remaining rows of the right table and null for the remaining left table's columns.

Syntax for Right Outer Join is,

SELECT column-name-list FROM
table-name1 RIGHT OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;

Right outer Join Syntax for Oracle is,

SELECT column-name-list FROM
table-name1, table-name2
ON table-name1.column-name(+) = table-name2.column-name;

Example of Right Outer Join

Once again the class table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

5 ashish

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Right Outer Join query will be,

SELECT * FROM class RIGHT OUTER JOIN class_info ON (class.id = class_info.id);

The resultant table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

Full Outer Join

The full outer join returns a resultset table with the matched data of two table then remaining rows of both left table and then the right table.

Syntax of Full Outer Join is,

SELECT column-name-list FROM
table-name1 FULL OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;

Example of Full outer join is,

The class table,

ID	NAME
1	abhi
2	adam

3 alex
4 anu
5 ashish

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Full Outer Join query will be like,

```
SELECT * FROM class FULL OUTER JOIN class_info ON (class.id =
class_info.id);
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

SQL Alias - AS Keyword

Alias is used to give an alias name to a table or a column, which can be a resultset table too. This is quite useful in case of large or complex queries. Alias is mainly used for giving a short alias name for a column or a table with complex names.

Syntax of Alias for table names,

```
SELECT column-name FROM table-name AS alias-name
```

Following is an SQL query using alias,

```
SELECT * FROM Employee_detail AS ed;
```

Syntax for defining alias for columns will be like,

SELECT column-name AS alias-name FROM table-name;
Example using alias for columns,

SELECT customer_id AS cid FROM Emp;
Example of Alias in SQL Query
Consider the following two tables,

The class table,

ID	Name
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Below is the Query to fetch data from both the tables using SQL Alias,

SELECT C.id, C.Name, Ci.Address from Class AS C, Class_info AS Ci where C.id = Ci.id;
and the resultset table will look like,

ID	Name	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

SQL Alias seems to be quite a simple feature of SQL, but it is highly useful when you are working with more than 3 tables and have to use JOIN on them.

SET Operations in SQL

SQL supports few Set operations which can be performed on the table data. These are used to get meaningful results from data stored in the table, under different special conditions.

In this tutorial, we will cover 4 different types of SET operations, along with example:

- UNION
- UNION ALL
- INTERSECT
- MINUS

UNION Operation

UNION is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

union set operation in sql

Example of UNION
The First table,

ID	Name
1	abhi
2	adam

The Second table,

ID	Name
2	adam
3	Chester

Union SQL query will be,

```
SELECT * FROM First
UNION
SELECT * FROM Second;
```

The resultset table will look like,

ID	NAME
1	abhi
2	adam
3	Chester

UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.

union all set operation in sql

Example of Union All

The First table,

ID	NAME
1	abhi
2	adam

The Second table,

ID	NAME
2	adam
3	Chester

Union All query will be like,

```
SELECT * FROM First
UNION ALL
SELECT * FROM Second;
```

The resultset table will look like,

ID	NAME
1	abhi
2	adam
2	adam
3	Chester

INTERSECT

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of Intersect the number of columns and datatype must be same.

NOTE: MySQL does not support INTERSECT operator.

intersect set operatoin in sql

Example of Intersect
The First table,

ID	NAME
1	abhi
2	adam

The Second table,

ID	NAME
2	adam
3	Chester

Intersect query will be,

```
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
```

The resultset table will look like

ID	NAME
2	adam

MINUS
The Minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result.

minus set operation in sql

Example of Minus
The First table,

ID	NAME
1	abhi
2	adam

The Second table,

ID	NAME
2	adam
3	Chester

Minus query will be,

```
SELECT * FROM First
MINUS
SELECT * FROM Second;
```

The resultset table will look like,

ID	NAME
1	abhi

Program-1

Delete duplicate row from the table.

DELETE FROM DEPT WHERE DEPTNO IN (SELECT DEPTNO FROM DEPT
GROUP BY DEPTNO HAVING COUNT(DEPTNO)>1);

DELETE FROM emp A WHERE ROWID NOT IN(SELECT MIN(ROWID) FROM emp WHERE A.DEPTNO=B.DEPTNO);

OR

DELETE FROM DEPT A WHERE ROWID NOT IN (SELECT MIN(ROWID) FROM DEPT B WHERE A.DEPTNO=B.DEPTNO);

Ques1:-Delete the row containing name Ram?

Ques2:-Delete all the rows having same name more then once?

Ques3:- Delete the row of employee whose name start with M?

Program-2

Display the alternate row from table.

```
SELECT * FROM EMP WHERE ROWID IN(SELECT
DECODE(MOD(ROWNUM,2),0,ROWID) FROM EMP);

OR
SELECT * FROM GDEPT WHERE ROWID IN(SELECT
DECODE(MOD(ROWNUM,2),0,ROWID) FROM GDEPT);
```

Ques1:-Show the name of those employees who earn commission?

Ques2:-Show all employees who has no commission but have a10% hike in their salary?

Ques3:-Show the last name of all employees together with the number of years & the number of complete months that they have been employed?

Program-3

Delete alternate row from table.

```
DELETE  FROM GDEPT WHERE ROWID IN(DELETE
DECODE(MOD(ROWNUM,2),0,ROWID) FROM GDEPT);
```

Ques1:-Delete the row of employee who works in location Bombay?

Ques2:- Delete the row of employee whose name end with N?

Ques3:- Delete the row of employee whose salary is more then 25000?

Program-4

Update multiple rows in using single update statement

DISPLAY ALL THE DETAILS WHERE DEPT IS EITHER SALES OR RESEARCH

Select * from emp where dname = any(select dname from emp where dname = sales or dname = research);

Select * from emp where dname = any(select dname from emp where Dname like(sales,research));

Ques1:-Find the name of those entire employee who work in Delhi and update there location to Bombay?

Ques2:-Find the name of those dept which are in same city?

Ques3:- Write a query to raise the salary by 50% of those employees who do not have a commission?

Program-5

Find the third highest paid and third lowest paid salary.

SOL: SELECT MAX(SAL) FROM EMP WHERE
SAL<(SELECT MAX(SAL) FROM EMP WHERE
SAL<(SELECT MAX(SAL) FROM EMP));
SOL: SELECT ENAME,SAL FROM EMP
MINUS
SELECT ENAME,SAL FROM EMP WHERE
SAL>(SELECT MIN(SAL) FROM EMP WHERE
SAL>(SELECT MIN(SAL) FROM EMP WHERE
SAL>(SELECT MIN(SAL) FROM EMP WHERE
SAL>(SELECT MIN(SAL) FROM EMP))));

Ques1:-Write a query to find all those employee who are in the dept which has the max salary of all dept?

Ques2:- Write a query to find those entire employees who earn maximum salary?

Ques3:- Write a query to find those employees who work in that dept in which the higher salary taker works?

Program-6

DISPLAY from NTH ROW

```
SELECT * FROM DEPT WHERE ROWID NOT IN(SELECT ROWID
FROM DEPT   WHERE ROWNUM<=(SELECT COUNT(*)-&N FROM
DEPT));
```

Display the 3rd, 4th, 9th rows from table.

```
SELECT * FROM DEPT WHERE ROWID NOT IN(SELECT ROWID
FROM DEPT   WHERE ROWNUM<=(SELECT COUNT(*)-&3 FROM
DEPT));

SELECT * FROM DEPT WHERE ROWID NOT IN(SELECT ROWID
FROM DEPT   WHERE ROWNUM<=(SELECT COUNT(*)-&4 FROM
DEPT));

SELECT * FROM DEPT WHERE ROWID NOT IN(SELECT ROWID
FROM DEPT   WHERE ROWNUM<=(SELECT COUNT(*)-&9 FROM
DEPT));
```

Ques1:-Show the dept name of the dept where no clerk works?

Ques2:-show the dept number and the lowest salary of the dept with the highest average salary?

Program-7

Display the ename, which is start with j, k, l or m.

```
select ename
from employees
where name like 'J%'
or name like 'K%'
or name like 'L%' or name like 'M%' ;
```

or

```
select ename
from
employees
where name like '[JKLM]%'
```

Ques1:-Write a query to find that how many employees are there whose name ends with N?

Ques2:- Write a query to find that how many employees are there whose name ends with M without using like operator?

Program-8

Show all employees who were hired the first half of the month.

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date < trunc(sysdate,'MM')+15;
```

Ques1:-Write a query to find the data of that entire employee whose name ends with t?

Ques2:- Find the DOB of that employee who was born on the same date on which the maximum salary earner was born?

Program-9

Display the three record in the first row and two records in the second row and one record in the third row in a single sql statements.

```
 INSERT INTO TEMP(EMPNO,ENAME,JOB)
SELECT TOP 1 *
FROM
(SELECT TOP 2<some columns>
FROM<table>ORDER BY<something> ASC)ORDER BY <something>
DESC;
```

Ques1:-Find the average salary of employee according to their dept?

Ques2:-Find the standard deviation according to employee salary?

Program-10

Write a sql statement for rollback commit and save points.

```
SQL> SELECT * FROM DEPT;

    DEPTNO DNAME          LOC
---------- -------------- -------------
10 ACCOUNTING       NEW YORK
20 RESEARCH   DALLAS
30 SALES          CHICAGO
40 OPERATIONS       BOSTON
   50 CS          MYSORE

SQL> SAVEPOINT A
2  ;

Savepoint created.

SQL> INSERT INTO DEPT VALUES(60,'IP','BANGALORE');

1 row created.
 SQL>  SAVEPOINT B;

Savepoint created.

SQL>  INSERT INTO DEPT VALUES(70,'IT','GOA');

1 row created.

SQL> SELECT * FROM DEPT;
  DEPTNO DNAME          LOC
---------- -------------- -------------
10 ACCOUNTING       NEW YORK
20 RESEARCH   DALLAS
30 SALES          CHICAGO
40 OPERATIONS       BOSTON
50 CS               MYSORE
60 IP         BANGALORE
70 IT         GOA
```

```
7 rows selected.

SQL> ROLLBACK TO SAVEPOINT B;

Rollback complete.

SQL> SELECT * FROM DEPT;

   DEPTNO DNAME              LOC
--------- -------------- -------------
10 ACCOUNTING       NEW YORK
20 RESEARCH   DALLAS
30 SALES          CHICAGO
40 OPERATIONS       BOSTON
50 CS               MYSORE
60 IP         BANGALORE

6 rows selected.

 temp
~~~~
prodname     comment    date1

create table temp( prodname varchar2(10), comm varchar2(16),
date1 date);

declare
qty NUMBER(5);
pname VARCHAR2(10);
begin
select quantity,prodname into qty,pname from inv where
prodname='&productname';
if qty>5 then
DBMS_OUTPUT.PUT_LINE('THANK U FOR THE PURCHASES MADE
VISIT AGAIN');
update inv set quantity=quantity-1 where prodname=pname;
commit;
else
```

```
DBMS_OUTPUT.PUT_LINE('STOCK LEVEL IS BELOW ORDER
LEVEL');
insert into temp values(pname,'out of stock',sysdate);
commit;
end if;
end;
```

Ques1:-Draw a sequence diagram for roll back and save point activity in ATM transaction?

Ques2:-Write syntax for rollback SQL query using suitable example?

PL/SQL

PL/SQL stands for Procedural Language/SQL.
PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL.
The basic unit in PL/SQL is a block, All PL/SQL programs are made up of blocks, which can be nested within each other. Typically, each block performs a logical action in the program.

Block has the following structure:

```
DECLARE
/* Declarative section: variables, types, and local subprograms.
*/

BEGIN
/* Executable section: procedural and SQL statements go
here. */
/* This is the only section of the block that is required. */
EXCEPTION
/* Exception handling section: error handling statements go
here. */
END;
```

Let us see an example of the above

```
DECLARE

TEMP_COST NUMBER(10, 2);
```

```
BEGIN
SELECT COST FROM JD11.BOOK INTO TEMP_COST
WHERE ISBN = 21;
IF TEMP_COST > 0 THEN
UPDATE JD11.BOOK SET COST =
(TEMP_COST*1.175) WHERE ISBN = 21;
ELSE
UPDATE JD11.BOOK SET COST = 21.32 WHERE
ISBN = 21;
END IF;
COMMIT;
EXCEPTION
WHEN NO_DATA_FOUND THEN
INSERT INTO JD11.ERRORS (CODE, MESSAGE)
VALUES (99, ISBN 21 NOT FOUND);
END;
```

Only the executable section is required. The other sections are optional.
The only SQL statements allowed in a PL/SQL program are SELECT, INSERT, UPDATE, DELETE and several other data manipulation statements plus some transaction control.
Data definition statements like CREATE, DROP, or ALTER are not allowed.
The executable section also contains constructs such as assignments, branches, loops, procedure calls, and triggers, which are all described below (except triggers). PL/SQL is not case sensitive. C style comments (/* ... */) may be used.

To execute a PL/SQL program, we must follow the program text itself by
A line with a single dot (.), and then A line with run;
As with Oracle SQL programs, we can invoke a PL/SQL program either by typing it in sql plus or by putting the code in a file and invoking the file in the various ways we learned in Getting Started With Oracle.

What are the Variables?
Information is transmitted between a PL/SQL program and the database through variables. Every variable has a specific type associated with it. That type can be

- One of the types used by SQL for database columns
- A generic type used in PL/SQL such as NUMBER
- Declared to be the same as the type of some database Column

The most commonly used generic type is NUMBER. Variables of type NUMBER can hold either an integer or a real number.

The most commonly used character string type is VARCHAR(n), where n is the maximum length of the string in bytes. This length is required, and there is no default. For example, we
might declare:
DECLARE
price NUMBER;
myBeer VARCHAR(20);
You know that PL/SQL allows BOOLEAN variables, even
though Oracle does not support BOOLEAN as a type for
database columns.

Types in PL/SQL

Types in PL/SQL can be tricky. In many cases, a PL/SQL variable will be used to manipulate data stored in a existing relation. In this case, it is essential that the variable have the same type as the relation column. If there is any type mismatch, variable assignments and comparisons may not work the way you expect. To be safe, instead of hard coding the type of a variable, you should use the %TYPE operator.

For example:
DECLARE
gives PL/SQL variable myBeer whatever type was declared for
the name column in relation Beers.

Program-11

Write a pl/sql for select, insert, update and delete statements.

```
CREATE TABLE TEMP
( ENAME VARCHAR2(10),
DESIG VARCHAR2(10),
SAL NUMBER(7,2));

 DECLARE
NAME VARCHAR2(10);
DESIG VARCHAR2(10);
SALARY NUMBER(7,2);
ENO NUMBER(4):=&EMPNO;
BEGIN
SELECT ENAME,JOB,SAL INTO NAME,DESIG,SALARY FROM EMP
WHERE EMPNO=ENO;
DBMS_OUTPUT.PUT_LINE(ENO||' '||NAME||' '||SALARY||' '||DESIG);
IF DESIG='CLERK' THEN
DELETE FROM EMP WHERE EMPNO=ENO;
INSERT INTO TEMP VALUES(NAME,DESIG,SALARY);
DBMS_OUTPUT.PUT_LINE('DELETED FROM EMP AND INSERTED TO
TEMP');
COMMIT;
ELSIF DESIG='MANAGER' THEN
UPDATE EMP SET SAL=SALARY+200 WHERE EMPNO=ENO;
DBMS_OUTPUT.PUT_LINE('INCREMENTED SALARY IS
'||TO_CHAR(SALARY+200));
END IF;
END;
```

Ques1:- Write a pl/sql for merge statement using suitable example?

Ques2:-Write a query to create a view for DEPT table(Full view,View of fragmented table) ?

Program-12

Write a pl/sql block to delete a record. If delete operation is successful return 1 else return 0.

```
create or replace function fun3(n emp.empno%type) return number is
a number;
begin
delete from emp where empno=n;
if sql%found then
return 1;
else
return 0;
end if;
--exception
--when no_data_found then
--return 100;
end;
declare
n number;
begin
n:=fun3(&empno);
dbms_output.put_Line(n);
if n=0 then
dbms_output.put_line('deletion unsuccessfull');
elsif n=1 then
dbms_output.put_line('deletion successfull');
end if;
end;
```

Cursors

What are Cursors?

A cursor is a variable that runs through the tuples of some relation. This relation can be a stored table, or it can be the answer to some query. By fetching into the cursor each tuple of the relation, we can write a program to read and process the value of each such tuple. If the relation is stored, we can also update or delete the tuple at the current cursor position. The example below illustrates a cursor loop. It uses our example relation T1(e,f) whose tuples are pairs of integers. The

program will delete every tuple whose first component is less than the second, and insert the reverse tuple into T1.

```
DECLARE
/* Output variables to hold the result of the query: */
a T1.e%TYPE;
b T1.f%TYPE;
/* Cursor declaration: */
CURSOR T1Cursor IS
SELECT e, f
FROM T1
WHERE e < f
FOR UPDATE;
BEGIN
OPEN T1Cursor;
LOOP
/* Retrieve each row of the result of the above query
into PL/SQL variables: */
FETCH T1Cursor INTO a, b;
/* If there are no more rows to fetch, exit the loop: */
EXIT WHEN T1Cursor%NOTFOUND;
/* Delete the current tuple: */
DELETE FROM T1 WHERE CURRENT OF T1Cursor;
/* Insert the reverse tuple: */

INSERT INTO T1 VALUES(b, a);
END LOOP;
/* Free cursor used by the query. */
CLOSE T1Cursor;
END;
```

Program-13

Display name, hire date of all employees using cursors.

```
DECLARE
cursor c1 is select ename,hiredate from emp;
name varchar(20);
hdate date;
begin
open c1;
loop
fetch c1 into name,hdate;
exit when c1%NOTFOUND;
dbms_output.put_line(name||' '||hdate);
end loop;
close c1;
end;
```

Ques1:-Display maximum salary using cursor?

Ques2:-Display salary of all employee in descending order using cursor?

Program-14

Display details of first 5 highly paid employees using cursors

```
DECLARE
cursor c1 is select * from emp order by sal desc;
a c1%rowtype;
begin
open c1;
loop
fetch c1 into a;
exit when c1%rowcount>6;
dbms_output.put_line(a.ename||' '||a.sal||' '||a.job||'
'||C1%ROWCOUNT);
end loop;
close c1;
end;
```

Ques1:-Write a query to find the details of those employee who have same job using cursor?

Ques2:-Write a query to show dept where no sales representative works using cursor?

Triggers

A trigger (essentially, a stored SQL statement associated with a table) is a database object that defines events that happen when some other event, called a triggering event, occurs. Create a trigger by using the CREATE TRIGGER statement. Triggers execute when an INSERT, UPDATE, or DELETE modifies a specified column or columns in the subject table. Typically, the stored SQL statements perform an UPDATE, INSERT, or DELETE on a table different from the subject table.
Sometimes a statement fires a trigger, which in turn, fires another trigger. Thus the outcome of one triggering event can itself become another trigger. The Teradata RDBMS processes and optimizes the triggered and triggering statements in parallel to maximize system performance.

Trigger Functions

Use triggers to perform various functions:

- Define a trigger on the parent table to ensure that UPDATEs and DELETEs performed to the parent table are propagated to the child table.
- Use triggers for auditing. For example, you can define a trigger which causes INSERTs in a log record when an employee receives a raise higher than 10%.
- Use a trigger to disallow massive UPDATEs, INSERTs, or DELETEs during business hours.

For example, you can use triggers to set thresholds for inventory of each item by store, to create a purchase order when the inventory drops below a threshold, or to change a price if the daily volume does not meet expectations.

Restrictions on Using Triggers

Teradata triggers do not support FastLoad and MultiLoad utilities and, and you must disable triggers before you run load utilities. In addition, a positioned (updatable cursor) UPDATE or DELETE is not allowed to fire a trigger and generates an error.
Note: You cannot define a join index on a table with a trigger.

CREATE TRIGGER <triggername> AFTER UPDATE/INSERT/DELETE
OF <COLUMN NAME> ON <TABLENAME> FOR EACH ROW
BEGIN

```
-----
executable statements;
-----
-----
END;
```

Program-15

Write a database trigger which fires if you try to insert, update, or delete after 7′o clock

```
CREATE OR REPLACE TRIGGER GEETIME BEFORE INSERT OR
UPDATE OR
 DELETE ON EMP for each row
DECLARE
A VARCHAR2(10);
BEGIN
SELECT TO_CHAR(SYSDATE,'HH:MI') INTO A FROM DUAL;
 IF A > '06:59' then
RAISE_APPLICATION_ERROR(-20500,'YOU CANT DO THIS
OPERATION
NOW');
END IF;
END;
```

Program-16

Write a data base trigger, which acts just like primary key and does not allow duplicate

```
CREATE OR REPLACE TRIGGER PRIKEY  BEFORE INSERT ON EMP
FOR EACH ROW
DECLARE
A NUMBER;
BEGIN
SELECT COUNT(*) INTO A FROM EMP WHERE
EMPNO=:NEW.EMPNO;
 IF A >=1 THEN
RAISE_APPLICATION_eRROR(-20500,'THE PRI KEY RULE IS
VOILATED');
ELSIF A=0 THEN
PRINT('RECORD IS INSERTED');
END IF;
END;
```

```
 SQL> INSERT INTO EMP(EMPNO,DEPTNO) VALUES(7788,20);
INSERT INTO EMP(EMPNO,DEPTNO) VALUES(7788,20)
ERROR at line 1:
*ORA-20500: THE PRI KEY RULE IS VOILATED
ORA-06512: at "GEETHA.PRIKEY", line 6
ORA-04088: error during execution of trigger 'GEETHA.PRIKEY'
SQL>  INSERT INTO EMP(EMPNO,DEPTNO) VALUES(77,20);


1 row created.
```

Program-17

Create a data base trigger, which performs the action of the on delete cascade

```
 CREATE OR REPLACE TRIGGER DELDEPT
AFTER DELETE ON DEPT FOR EACH ROW
BEGIN
 DELETE FROM EMP WHERE DEPTNO=:OLD.DEPTNO;
PRINT('RECORDS IN EMP ARE ALSO DELETED');
END;
```

Program-18

Write a data base trigger, which should not delete from emp table if the day is Sunday.

```
CREATE OR REPLACE TRIGGER EMPNO_CHECK
BEFORE DELETE   ON emp
BEGIN
 if  to_char(sysdate,'dAy')='SUNDAY'  then
raise_application_error(-20001,'TO DAY IS SUNDAY  ');
end if;
END;
```