**NAME – CHAITALI YUVRAJ WADGHULE**

**SUBJECT -DBMS QUERY**

**ROLL NO – EBEON0722618054**

* 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
* 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

SQL\*Plus: Release 11.2.0.2.0 Production on Tue Oct 11 10:37:35 2022

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect

Enter user-name: system

Enter password:

Connected.

**Create table employee2**

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

SQL> create table employee2(ID int,name varchar(20),address varchar(20),phone\_no int);

Table created.

**Insert values**

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

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

SQL> insert into employee2 values(111,'ramesh','pune',9820473820);

1 row created.

SQL> insert into employee2 values(112,'krishna','satara',9013658764);

1 row created.

SQL> insert into employee2 values(113,'priyansh','nashik',8733206517);

1 row created.

SQL> insert into employee2 values(114,'ganesh','nagpur',8292866534);

1 row created.

SQL> insert into employee2 values(115,'jay','khed',9045678320);

1 row created.

**Aggregate Functions**

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

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

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*;

SQL>SQL> select 'ramesh', 'pune' from employee2;

'RAMES 'PUN

------ ----

ramesh pune

ramesh pune

ramesh pune

ramesh pune

ramesh pune

SQL> select 'mr.patil', 'patil03@gmail.com' from teacher2;

'MR.PATI 'PATIL03@GMAIL.CO

-------- -----------------

mr.patil patil03@gmail.com

mr.patil patil03@gmail.com

mr.patil patil03@gmail.com

mr.patil patil03@gmail.com

mr.patil patil03@gmail.com

SQL>select 'nashik' from employee2;

'NASHI

------

nashik

nashik

nashik

nashik

nashik

**select distinct**

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

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

SQL> select distinct 'satara' from employee2;

'SATAR

------

satara

SQL> select count(distinct 'manchar') FROM employee2;

COUNT(DISTINCT'MANCHAR')

------------------------

1

SQL>select count(\*) as distinctaddresses from (select distinct address from employee2);

DISTINCTADDRESSES

-----------------

5

**Where**

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

SQL> select \* from employee2 where address='khed';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

115 jay khed 9045678320

SQL> select \* from employee2 where id=114;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

114 sita manchar 8292866534

**And, Or and Not**

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**

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

SQL> select \* from employee2 where name='priyansh' and address='nashik';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

113 priyansh nashik 8733206517

OR

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

SQL> select \* from employee2 where name='priyansh' or name='ramesh';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

113 priyansh nashik 8733206517

**NOT**

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

SQL> select \* from employee2 where not name='priyansh';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

112 krishna satara 9013658764

114 sita manchar 8292866534

115 jay khed 9045678320

SQL> select \* from employee2 where address='pune' and (name='ramesh' or name='sita');

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

SQL> select \* from employee2 where not address='pune' and not name='priyansh';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

112 krishna satara 9013658764

114 sita manchar 8292866534

115 jay khed 9045678320

**Order BY**

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.

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

SQL> select \* from employee2 order by address;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

115 jay khed 9045678320

114 sita manchar 8292866534

113 priyansh nashik 8733206517

111 ramesh pune 9820473820

112 krishna satara 9013658764

SQL> select \* from employee2 order by address desc;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

112 krishna satara 9013658764

111 ramesh pune 9820473820

113 priyansh nashik 8733206517

114 sita manchar 8292866534

115 jay khed 9045678320

SQL> select \* from employee2 order by address, name;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

115 jay khed 9045678320

114 sita manchar 8292866534

113 priyansh nashik 8733206517

111 ramesh pune 9820473820

112 krishna satara 9013658764

SQL> select \* from employee2 order by address asc, name desc;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

115 jay khed 9045678320

114 sita manchar 8292866534

113 priyansh nashik 8733206517

111 ramesh pune 9820473820

112 krishna satara 9013658764

SQL> select 'jay' from employee2 where 'khed' is not null;

'JA

---

jay

jay

jay

jay

jay

**UPDATE**

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

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

SQL> update employee2 set name = 'sita',address = 'manchar' WHERE ID = 114;

1 row updated.

SQL> select \* from employee2;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

112 krishna satara 9013658764

113 priyansh nashik 8733206517

114 sita manchar 8292866534

115 jay khed 9045678320

**Min and Max**

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

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

select min(phone\_no) as smallestphone\_no from employee2;

SMALLESTPHONE\_NO

----------------

8292866534

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

SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

SQL> select max(phone\_no) as largestphone\_no from employee2;

LARGESTPHONE\_NO

---------------

9820473820

**Count**

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

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

SQL> select count(id) from employee2;

COUNT(ID)

----------

5

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

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

SQL> select avg(id) from employee2;

AVG(ID)

----------

113

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

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

SQL> select sum(id) from employee2;

SUM(ID)

----------

565

**Like**

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 sign (\_) represents one, single character

The percent sign and the underscore can also be used in combinations!

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

SQL> select \* from employee2 where name like 'j%';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

115 jay khed 9045678320

SQL> select \* from employee2 where name like '%an%';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

113 priyansh nashik 8733206517

SQL> select \* from employee2 where name like '\_r%';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

112 krishna satara 9013658764

113 priyansh nashik 8733206517

SQL> select \* from employee2 where name not like 'a%';

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

112 krishna satara 9013658764

113 priyansh nashik 8733206517

114 sita manchar 8292866534

115 jay khed 9045678320

**IN**

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

The IN operator is a shorthand for multiple OR conditions.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

SQL> select \* from employee2 where address in ('khed', 'satara', 'nashik');

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

112 krishna satara 9013658764

113 priyansh nashik 8733206517

115 jay khed 9045678320

SQL> select \* from employee2 where address not in ('khed', 'satara', 'nashik');

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

114 sita manchar 8292866534

SQL> select \* from employee2 where address in (select address from employee2);

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

112 krishna satara 9013658764

113 priyansh nashik 8733206517

114 sita manchar 8292866534

115 jay khed 9045678320

**Between**

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.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

SQL> select \* from employee2 where id between 111 and 114;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

112 krishna satara 9013658764

113 priyansh nashik 8733206517

114 sita manchar 8292866534

SQL> select \* from employee2 where id not between 111 and 114;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

115 jay khed 9045678320

SQL> select \* from employee2 where id between 111 and 114 and phone\_no not in(1,2,3);

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

111 ramesh pune 9820473820

112 krishna satara 9013658764

113 priyansh nashik 8733206517

114 sita manchar 8292866534

SQL> select \* from employee2 where name between 'krishna' and 'sita' order by name;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

112 krishna satara 9013658764

113 priyansh nashik 8733206517

111 ramesh pune 9820473820

114 sita manchar 8292866534

SQL> select \* from employee2 where name not between 'krishna' and 'sita' order by name;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

115 jay khed 9045678320

**Join**

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

* (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

**1**



The INNER JOIN keyword selects records that have matching values in both tables.

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name = table2.column\_name;

**2**



The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

**3**



The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

### SELECT column\_name(s) FROM table1 RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

**4**



The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

**Tip:** FULL OUTER JOIN and FULL JOIN are the same.

SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2ON table1.column\_name = table2.column\_nameWHERE condition;

SQL> create table students(id int primary key,name varchar(20),grade int,gpa float,tardies int);

Table created.

SQL> create table teachers(id int primary key,name varchar(20),subject varchar(20),grade int);

Table created.

SQL> insert into students values(1,'chaitali',12,3.5,4);

1 row created.

SQL> insert into students values(2,'priti',11,3.7,4);

1 row created.

SQL> insert into students values(3,'prisha',10,3.3,5);

1 row created.

SQL> insert into students values(4,'prisha',11,3.3,3);

1 row created.

SQL> insert into students values(5,'rakhi',9,3.1,6);

1 row created.

SQL> insert into students values(6,'reema',12,3.7,5);

1 row created.

SQL> insert into teachers values(11,'mr.smith','math',5);

1 row created.

SQL> insert into teachers values(12,'mrs.swati','english',3);

1 row created.

SQL> insert into teachers values(13,'miss.swapna','hindi',7);

1 row created.

SQL> insert into teachers values(14,'mr.vivek','history',8);

1 row created.

SQL> insert into teachers values(15,'mrs.asha','marathi',8);

1 row created.

SQL> insert into teachers values(16,'mrs.abhita','science',4);

1 row created.

SQL> select \* from students;

ID NAME GRADE GPA TARDIES

---------- -------------------- ---------- ---------- ----------

1 chaitali 12 3.5 4

2 priti 11 3.7 4

3 prisha 10 3.3 5

4 prisha 11 3.3 3

5 rakhi 9 3.1 6

6 reema 12 3.7 5

6 rows selected.

SQL> select \* from teachers;

ID NAME SUBJECT GRADE

---------- -------------------- -------------------- ----------

11 mr.smith math 5

12 mrs.swati english 3

13 miss.swapna hindi 7

14 mr.vivek history 8

15 mrs.asha marathi 8

16 mrs.abhita science 4

6 rows selected.

SQL>SQL> select id as id,name as name from employee2;

ID NAME

---------- --------------------

111 ramesh

112 krishna

113 priyansh

114 sita

115 jay

**COMMIT**

***COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk. Whenever DDL commands such as INSERT, UPDATE and DELETE are used, the changes made by these commands are permanent only after closing the current session. So before closing the session, one can easily roll back the changes made by the DDL commands. Hence, if we want the changes to be saved permanently to the disk without closing the session, we will use the commit command***.

**Syntax:**

1. **COMMIT**;

SQL> CREATE TABLE t\_school(ID INT,School\_Name VARCHAR(40),Number\_Of\_Students INT,Number\_Of\_Teachers INT,Number\_Of\_Classrooms INT,EmailID VARCHAR(40));

Table created.

SQL> INSERT INTO t\_school VALUES(1, 'Boys Town Public School', 1000, 80, 12, 'btps15@gmail.com');

1 row created.

SQL> INSERT INTO t\_school VALUES (2, 'Guru Govind Singh Public School', 800, 35, 15, 'ggps25@gmail.com');

1 row created.

SQL> INSERT INTO t\_school VALUES (3, 'Delhi Public School', 1200, 30, 10, 'dps101@gmail.com');

1 row created.

SQL> INSERT INTO t\_school VALUES (4, 'Ashoks Universal School', 1110, 40, 40, 'aus17@gmail.com');

1 row created.

SQL> INSERT INTO t\_school VALUES (5, 'Calibers English Medium School', 9000, 31, 50, 'cems@gmail.com');

1 row created.

SQL> select \* from t\_school;

ID SCHOOL\_NAME NUMBER\_OF\_STUDENTS

---------- ---------------------------------------- ------------------

NUMBER\_OF\_TEACHERS NUMBER\_OF\_CLASSROOMS EMAILID

------------------ -------------------- ----------------------------------------

1 Boys Town Public School 1000

80 12 btps15@gmail.com

2 Guru Govind Singh Public School 800

35 15 ggps25@gmail.com

3 Delhi Public School 1200

30 10 dps101@gmail.com

ID SCHOOL\_NAME NUMBER\_OF\_STUDENTS

---------- ---------------------------------------- ------------------

NUMBER\_OF\_TEACHERS NUMBER\_OF\_CLASSROOMS EMAILID

------------------ -------------------- ----------------------------------------

4 Ashoks Universal School 1110

40 40 aus17@gmail.com

5 Calibers English Medium School 9000

31 50 cems@gmail.com

**SAVEPOINT**

***We can divide the database operations into parts. For example, we can consider all the insert related queries that we will execute consecutively as one part of the transaction and the delete command as the other part of the transaction. Using the SAVEPOINT command in SQL, we can save these different parts of the same transaction using different names.*For example*, we can save all the insert related queries with the savepoint named INS. To save all the insert related queries in one savepoint, we have to execute the SAVEPOINT query followed by the savepoint name after finishing the insert command execution.***

***Syntax:***

1. ***SAVEPOINT savepoint\_name;***

Savepoint created.

SQL> SAVEPOINT savepoint\_name;

Savepoint created.

NOT NULL Constraint

SQL> CREATE TABLE persons (ID int NOT NULL, LastName varchar(255) NOT NULL,FirstName varchar(255) NOT NULL, Age int);

**ROLLBACK**

We didn't need the updation carried on the record. Hence, we have rolled back to the savepoint named Insertion.

For confirming that we have got the same t\_school table that we had before carrying out the updation operation, we will again execute the SELECT query.

SQL>rollback to updation;

Rollback complete.

**ALTER**

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.

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

ALTER TABLE *table\_name*  
ADD *column\_name datatype*;

SQL> alter table employee2 add email varchar(20);

Table altered.

SQL> select \* from employee2;

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

EMAIL

--------------------

111 ramesh pune 9820473820

112 krishna satara 9013658764

113 priyansh nashik 8733206517

ID NAME ADDRESS PHONE\_NO

---------- -------------------- -------------------- ----------

EMAIL

--------------------

114 sita manchar 8292866534

115 jay khed 9045678320