UNIT 3

DATABASE MANAGEMENT

SQL Commands

DML Commands

Consider following table as a base to perform different display operations

Student

ROLLNO	NAME	GENDER	MARKS	DOB	MOBILE_NO	STREAM
1	Raj Kumar	M	93	2000-09-17	9586774748	Science
2	Rahul singh	M	90	2000-09-11	9586767987	Commerce
3	Raman Verma	M	76	2000-02-22	NULL	Science
4	Suman	F	78	1999-12-03	9818675444	Humanities
5	Sugandha	F	82	1998-04-21	9845639990	vocational
6	Payal Goyal	F	80	1999-12-17	9897666650	Science
7	Samita Sachdeva	F	NULL	NULL	NULL	Commerce

7 rows in set (0.00 sec)

▶TO DISPLAY RECORDS OF TABLE

Select Command is used to display data of the table.

a) To display all the content of the table or To display all rows and all columns.

Example :

Select * from student;

b) To display all rows and specific columns.

Select name, marks from student;

c) To display all columns and specific rows (Use of WHERE clause)

Select * from student where marks<80;

d) To eliminate Redundant Data (Use of DISTINCT)

By default, data is select from all the rows of the table, even if the data appearing in the result gets duplicated. The DISTINCT keywords eliminates duplicate rows from the results of a SELECT statement

Select distinct stream from student;

Output-

STREAM
Science
Commerce
Humanities
Vocational

e) To display data based on a List (Use of IN / NOT IN)

To specify a list of values, IN operator is used.The IN operator selects values that match any value in a given list of values. It can replace use of logical expressions.

NOT IN operator finds rows that do not match in the list.

- For example, to display a list of students from different streams

Select name, gender from student where stream in ('Commerce', 'Humanities') ;

or

Select name, gender from student where stream = 'Commerce' or stream = 'Humanities';

Output -

NAME	GENDER
Rahul singh	M
Suman	F
Samita Sachdeva	F

f) To display data based on a Range (Use of BETWEEN Clause)

BETWEEN allows to specify range of values to search in any column. It is used with AND clause and it will include the specified values during the searching

- For example, to display a list of students with their marks who scored marks in the range of 80 to 90

Select name, marks from student where marks between 80 and 90;

Output -

NAME	MARKS
Rahul singh	90
Sugandha	82
Payal Goyal	80

g) To display data based on Pattern Matching(Use of Like Clause)

SQL also includes a string- matching operator, LIKE, allows to search based on pattern.

- ✓ It is used in case we don't want to search an exact value
- ✓ In case we don't know that exact value
- ✓ In case we know only the pattern of value like name starting from any particular letter, or ending with and containing any particular letter or word.

Patterns are described using two special wildcard characters:

Percent (%) - The % character matches any substring.

Underscore (_) - The _ character matches any one character.

For Example

- ➢ Search for student whose name begins from 's'

Select * from student where name like 's%';

- ➢ Search for student whose name ends with 'r'

Select * from student where name like '%r';

- ➢ Search for student whose name contains 'a' anywhere

Select * from student where name like '%a%'

- ➢ Search for student whose dob is in December

Select * from student where dob like '%_12_%'

- ➢ Search student whose name is of 5 letters begins from 's'

Select * from student where name like 's_ _ _ _ ';

h)To display data in arranged order(Use of Order By)

By default records will come in the output in the same order in which it was entered. To see the output rows in sorted or arranged in ascending or descending order.SQL provides ORDER **BY** clause.

- ✓ By default output will be in ascending order(ASC)
- ✓ To see output in descending order we use DESC clause with ORDER BY.

Example

- Display information of students in ascending order of name

  Select * from student order by name;

- Display information of students in descending order of marks

  Select * from student order by marks desc;

i) To search NULL Values (Use of IS NULL)

The NULL value in a column can be searched using IS NULL in where clause. Relational operators can't be used with NULL values.

Example

Display rollno and name of students whose mobile no is not available

Select rollno, name from students where mobile_no is null;

Quick Revision

TABLE: GRADUATE

SNO	NAME	STIPEND	SUBJECT	AVERAGE	DIV
1	KARAN	400	PHYSICS	68	I
2	DIWAKAR	450	COMP. Sc.	68	I
3	DIVYA	175	CHEMISTRY	40	NULL
4	REKHA	350	PHYSICS	63	I
5	ARJUN	500	MATHS	70	I
6	SABINA	400	CEHMISTRY	55	II
7	JOHN	150	PHYSICS	34	NULL
8	ROBERT	450	MATHS	68	I
9	RUBINA	500	COMP. Sc.	62	I
10	VIKAS	400	MATHS	57	II

Write SQL commands for the following statements:

i. List the names of those students who have obtained DIV I sorted by NAME.

ii. Display a report, listing NAME, STIPEND, SUBJECT and amount of stipend received in a year assuming that the STIPEND is paid every month.

iii. To display NAME and STIPEND of students who are either PHYSICS or COMPUTER SC graduates.

iv. To insert a new row in the GRADUATE table: 11,"KAJOL", 300, "computer sc", 75, 1

v. Display details of those students whose NAME starts with 'R'.

vi. Display NAME and SUBJECT of those students who have AVERAGE in the range of 65 and 75.

vii. List SUBJECT's available for graduation.

viii. List the names of student who scored 'I' division in 'Maths';

ix. List names of students whose division is NULL.

x. Increase the value of stipend of students of computer science by 200.

SQL Commands

DML Commands

Consider following table as a base to perform different display operations

Emp

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	Nitin	HR	45000

USE OF AGGREGATE FUNCTIONS

Aggregate function is used to perform calculation on group of rows and return the calculated summary. They can be applied to all rows in a table or to a subset of the table specified by WHERE clause.

Available aggregate functions are:

1. **avg( )** – to compute average value
2. **min( )** – to find minimum value
3. **count( )** – to count non- null values in a column
4. **max( )** – to find maximum value

5. **count(*)** – to count total number of rows in a table
6. **sum( )** – to find the total value

Example :

```
Select SUM(salary) from emp;
```

Output

SUM(salary)
161000

```
Select SUM(salary) from emp where dept='sales';
```

Output

SUM(salary)
59000

```
Select AVG(salary) from emp where dept='sales';
```

Output

AVG(salary)
29500

```
Select COUNT(name) from emp;
```

Output

COUNT(name)
5

```
Select COUNT(salary) from emp where dept='HR';
```

Output

COUNT(salary)

1

Select COUNT(DISTINCT dept) from emp;

Output

COUNT(DISTINCT dept)

Sales

IT

HR

Select MAX(Salary) from emp;

Output

MAX(Salary)

45000

Select MIN(salary) from emp where dept='Sales';

Output

MIN(salary)

24000

NOTE - Count (*) function is used to count the number of rows in query output whereas count() is used to count values present in any column excluding NULL values.

All aggregate function ignores the NULL values.

USE OF GROUP BY CLAUSE

GROUP BY clause is used to divide the table into logical groups and we can perform aggregate functions in those groups.

In this case aggregate function will return output for each group.

Aggregate functions by default takes the entire table as a single group that's why we are getting the sum (), avg( ), etc output for the entire table.

Consider a situation, any organization wants the sum () of all the jobs separately, or wants to find the average salary of every job. In this case we have to logically divide our table into groups based on job, so that every group will be passed to aggregate function for calculation and aggregate function will return the result for every group.

Example:

- **Display department wise total salary**

  Select dept, SUM(salary) from emp group by dept;

Output –

Dept	Sum(Salary)
Sales	59000
IT	57000
HR	45000

- **Display frequency of each department with its minimum**

& maximum salary

```
Select dept, COUNT(*),MAX(salary),MIN(Salary) from
emp;
```

Output –

Dept	Count(*)	MAX(salary)	MIN(Salary)
Sales	2	35000	24000
IT	2	30000	27000
HR	1	45000	45000

USE OF HAVING WITH GROUP BY CLAUSE

- If we want to filter or restrict some rows from the output produced by GROUP BY then we use HAVING clause. It is used to put condition on group of rows.
- With having clause we can use aggregate functions also.
- With WHERE we cannot use aggregate function.

Example:

℘ **Display department wise total salary whose count is more than one**

```
Select dept, SUM(salary) from emp group by dept having
count(*)>1;
```

Output –

Dept	Sum(Salary)
Sales	59000
IT	57000

- Display minimum & maximum salary of IT & HR department

```
Select dept, MAX(salary), MIN(salary) from emp group by
dept having dept in('IT','HR');
```

Output –

Dept	MAX(salary)	MIN(Salary)
IT	30000	27000
HR	45000	45000

USE OF EQUI JOIN

- Join is a query which combines rows of two or more tables.
- Equijoin is a type of join with a join condition containing an equality operator.
- An equi-join returns only the rows that have equivalent values for the specified columns.
- The joins are formed based on equal '=' condition in 'WHERE' clause ...

Consider following tables as a base for equi -join **statements**

Emp

Empno	Name	Dno	Salary
1	Ravi	103	24000
2	Sunny	102	35000
3	Shobit	101	30000
4	Vikram	102	27000
5	Nitin	101	45000

Department

Dno	DName	Location
101	Sales	A-Block
102	Accounts	B-Block
103	Marketing	A-Block
104	IT	C-Block
105	HR	D- Block

Example:

- **Display name of employees with their departments name**

```
Select Name, Dname from emp e, department d where
e.Dno=d.Dno;

                OR

Select Name, Dname from emp ,department where
emp.Dno=department.Dno;
```

Output –

Name	DName
Ravi	Marketing
Sunny	Accounts
Shobit	Sales
Vikram	Accounts
Nitin	Sales

℘ Display name and location of department who are working in IT department

Select Name, Location from emp e, department d where e.Dno=d.Dno and DNome= 'IT';

OR

Select Name, Location from emp ,department where emp.Dno=department.Dno and DNome= 'IT';

Output –

Name	Location
Vikram	C Block

Quick Revision

Q1

Table: SAP

SAPID	ItemCode	ItemName	ItemStorageLocation
S1001	1001	Receiver	W12-B3-R24
S1002	1002	Transponder	W13-B7-R87
S1003	1003	Battery Bank	W21-B1-R87
S1004	1004	Inverter	W21-B11-R2
S1005	1005	Genset	W22-B15-R16

Table : Store

StoreID	Item Code	StoreLocation	ReceivedDate
1201	1001	Hauz Khas	2016/05/20
1202	1002	Rajouri Garden	2016/06/14
1203	1003	Rohini	2016/05/06
1204	1004	Hauz Khaas	2016/07/15
1205	1005	Rajendra Place	2016/05/27

Write SQL Commands for the following:

1) To display the ItemCode, ItemName and ReceivedDate of all the items .

2) To display details of items in descending order of their names.

3) To display SAPID, ItemName, ItemStorageLocation of all the items whose received date is after 2nd May 2016.

4) To display SAPID, ItemName, StoreID of all those items whose store location is "HauzKhas"

Q2

Table: CABHUB

Vcode	VehicleName	Make	Color	Capacity	Charges	
100	Innova	Toyota	WHITE	7	15	
102	SX4	Suzuki	BLUE	4	14	
104	C Class	Mercedes	RED	4	35	
105	A-Star	Suzuki	WHITE	3	14	
108	Indigo	Tata	SILVER	3	12	

Table: CUSTOMER

CCode	CName	VCode	
1	Hemant Sahu	101	
2	Raj Lal	108	
3	Feroza Shah	105	
4	Ketan Dhal	104	

(a) Write SQL commands for the following statements:

1) To display the names of all white colored vehicles
2) To display name of vehicle, make and capacity of vehicles in ascending order of their sitting capacity
3) To display the highest charges at which a vehicle can be hired from CABHUB.
4) To display the customer name and the corresponding name of the vehicle hired by them.

(b) Give the output of the following SQL queries:

1) SELECT COUNT(DISTINCT Make) FROM CABHUB;
2) SELECT MAX(Charges), MIN(Charges) FROM CABHUB;
3) SELECT COUNT(*), Make FROM CABHUB GROUP BY Make;
4) SELECT VehicleName FROM CABHUB WHERE Capacity = 4;