ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

The Motto of the University

(SEWA)

SKILL ENHANCEMENT **EMPLOYABILITY** **WISDOM**
ACCESSIBILITY

ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ, ਪਟਿਆਲਾ
JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA

DIPLOMA IN MOBILE APPLICATION DEVELOPMENT (DMAD)
SEMESTER-II
Course: DBMS Lab
Course Code: DBMS-2-01P

ADDRESS: C/28, THE LOWER MALL, PATIALA-147001
WEBSITE: www.psou.ac.in

SELF-INSTRUCTIONAL STUDY MATERIAL FOR JGND PSOU, ALL COPYRIGHTS WITH JGND PSOU, PATIALA

DBMS-2-01P: Data Base Management System (DBMS) Lab

Total Marks: 50
External Marks: 35
Internal Marks: 15
Credits: 2
Pass Percentage: 40%

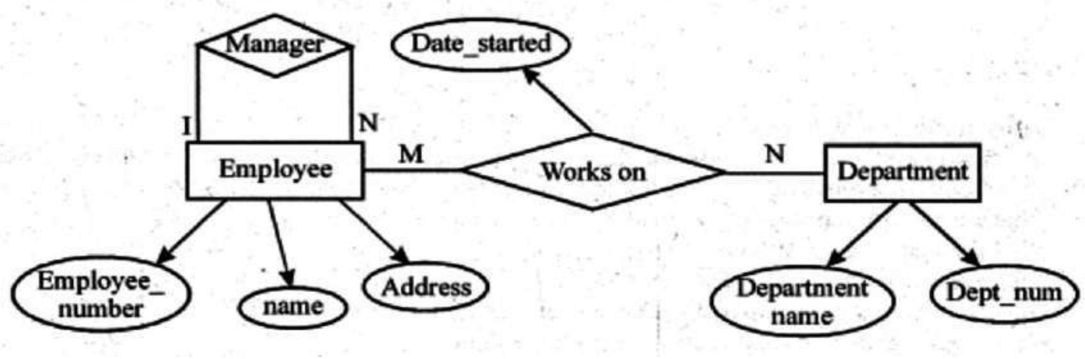
Course: Data Base Management System (DBMS) Lab	
Course Code: DBMS-2-01P	
Course Outcomes (COs) After the completion of this course, the students will be able to:	
CO1	Implement Basic DDL, DML and DCL commands.
CO2	Understand Data selection and operators used in queries and restrict data retrieval and control the display order.
CO3	Use Aggregate and group functions to summarize data.
CO4	Join multiple tables using different types of joins.
CO5	Implementation of different types of operators in SQL

Detailed List of Programs:

Programme No.	**Name of Program**
P1	Implementation of DDL commands of SQL with suitable examples • Create table • Alter table • Drop Table
P2	Implementation of DML commands of SQL with suitable examples • Insert • Update • Delete
P3	Implementation of different types of function with suitable examples • Number function • Aggregate Function • Character Function • Conversion Function • Date Function
P4	Implementation of different types of operators in SQL • Arithmetic Operators • Logical Operators • Comparison Operator • Special Operator • Set Operation

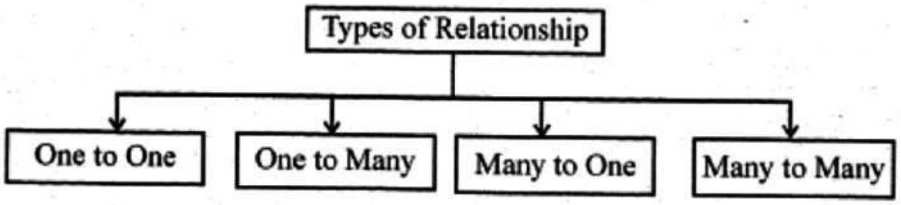
P5	Implementation of different types of Joins • Inner Join • Outer Join • Natural Join etc.
P6	Implementation of • Group by & having clause • Order by clause • Indexing
P7	Implementation of • Sub queries • Views
P8	Study & Implementation of different types of constraints.
P9	Study & Implementation of Database Backup & Recovery commands. Study & Implementation of Rollback, Commit, Savepoint.
P10	Creating Database /Table Space • Managing Users: Create User, Delete User • Managing roles:-Grant, Revoke

Q1. The E-R Diagram for an Employee Payroll System.



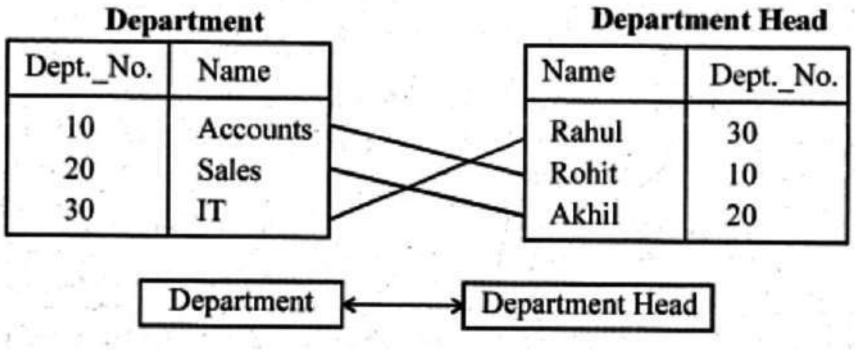
Q2. Explain with diagrammatical illustrations about the different types of relationships.

- It is used to connect the entities.
- The entities involved in given relationship are called participants.
- The no. of participants in a given relationship is called degree of *relationship*.
- ◇ sign is used to represent relationship among entities.
- Deposit is a relationship among entity customer and entity account.
- Relationship can be of four types which are as follows:



Types of Relationship

(a) One to One Relationship: In one to one relationship for one record in entity A, there is exactly one record in entity B. For example: we have two entities department and department head. There is one to one relationship because one department will be under one head and one head will be appointed for one department.



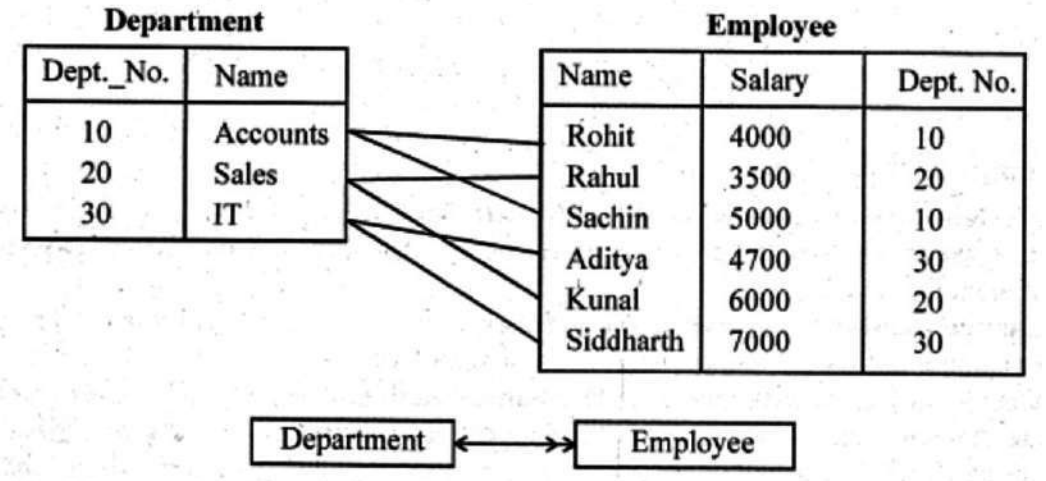
Department

Dept._No.	Name
10	Accounts
20	Sales
30	IT

Department Head

Name	Dept._No.
Rahul	30
Rohit	10
Akhil	20

(b) One to Many Relationship: In one to many relationships for one record in entity A, there is more than one record in entity B. For example: We have two entities department and employee. There is one to many relationships because there will be one department in a company and more than one employee will work in that particular department.



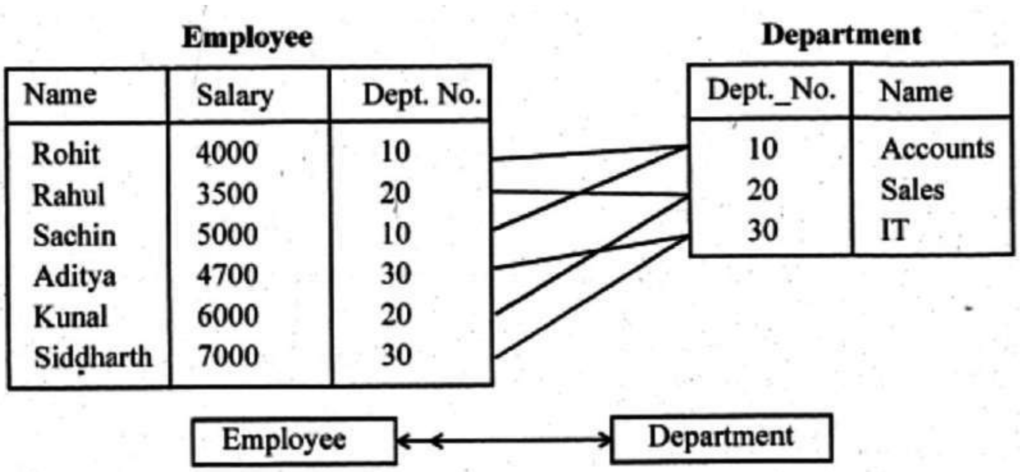
Department

Dept._No.	Name
10	Accounts
20	Sales
30	IT

Employee

Name	Salary	Dept. No.
Rohit	4000	10
Rahul	3500	20
Sachin	5000	10
Aditya	4700	30
Kunal	6000	20
Siddharth	7000	30

(c) Many to One Relationship: In many to one relationship, for many records in entity A, there is only one record in entity B. For example: We have two entities employee and department. There is many to one relationship because there will be many employees in a single



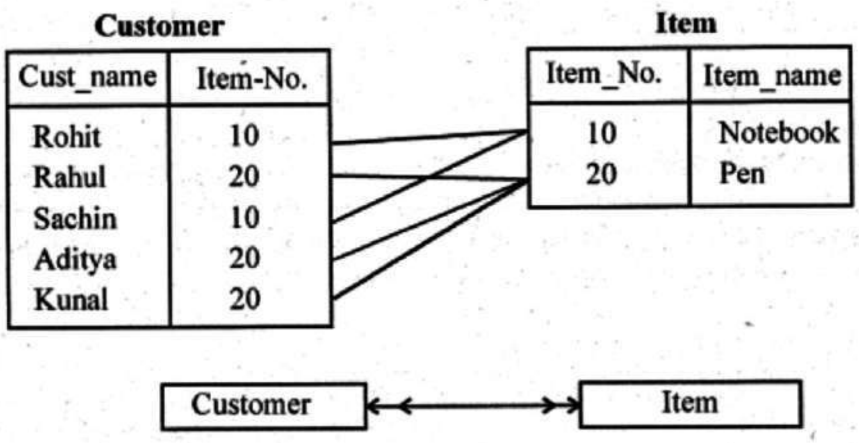
Employee

Name	Salary	Dept. No.
Rohit	4000	10
Rahul	3500	20
Sachin	5000	10
Aditya	4700	30
Kunal	6000	20
Siddharth	7000	30

Department

Dept._No.	Name
10	Accounts
20	Sales
30	IT

(d) Many to Many Relationship: In many to many relationships, for many record is an entity A, there will be many record in entity B. There is many to many relationship because there will be many customers for many items.



Customer

Cust_name	Item-No.
Rohit	10
Rahul	20
Sachin	10
Aditya	20
Kunal	20

Item

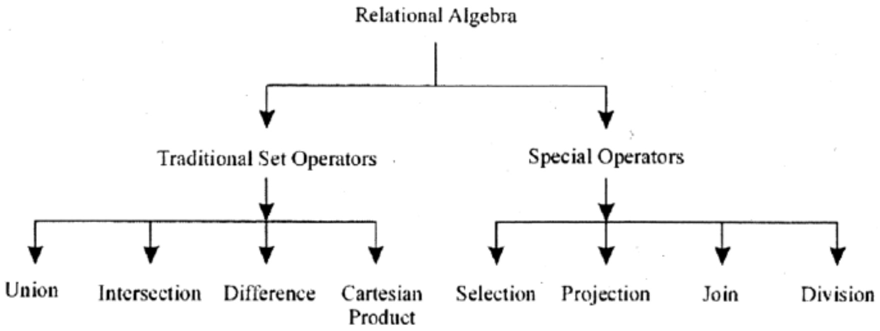
Item_No.	Item_name
10	Notebook
20	Pen

Q3. The various set operators available in relational algebra with suitable examples.

The relational model uses the concept of a mathematical relation in the form of table of values which acts as building block. The table is a logical representation of data in the form of rows and columns. The relational algebra is a formal query language applied on relational model. It is a procedural language which specifies the operations to be performed on relations. The operations are performed in form of sequence of algebra operations which results in a new relation/table. The relational algebra operations can be classified into two types.

Key Points:

1. Relational algebra is a procedural query language.
2. It consists of set of operators that take one or two relations as input and produce a new relation as output.
3. It uses relational operators.
4. It is of mainly two types which are as follows:



Classification of Relational Algebra

I. Traditional Set Operators

(a)Union Operator

(b)Intersection Operator

(c)Difference Operator

(d)Cartesian Product Operator

(a) Union Operator:

- Union of two relations is the set of all elements belonging to both relations.
- Result must not contain duplicate elements.
- It is denoted by U.
- For example: We want to list all the names and roll numbers which are present in both tables: 'A' and 'B'.

AB

Name	Roll Number
Akhil	211
Monika	129

Name	Roll Number
Aastha	112
Akhil	211

Formula:πName, Roll Number (A) U πName, Roll Number (B).

AUB

Name	**Roll Number**
Akhil	211
Monika	129
Aastha	112

(b) Intersection Operator:

- Intersection of two relations produces a relation which contains all elements that are common to both relations.
- It is denoted by ∩.
- For example: We want to list only those names and roll numbers which are common in both tables 'A' and 'B'.

A

Name	Roll Number
Akhil	211
Monika	129

B

Name	Roll Number
Aastha	112
Akhil	211

Formula:πName, Roll Number (A) ∩πName, Roll Number (B)

A∩B

Name	**Roll Number**
Akhil	211

(c) Difference Operator

- Difference operator is used to find those tuples which are present in one relation but not in another relation.
- It is denoted by (-) sign.
- For example: We want to list those names and roll numbers which are present in table 'A^{9} only, not in table'B'.

AB

Name	Roll Number
Akhil	211
Monika	129

Name	Roll Number
Aastha	112
Akhil	211

Formula:πName, Roll Number (A) –πName, Roll Number (B)

A-B

Name	Roll Number
Monika	129

B-A

Name	Roll Number
Aastha	112

(d) Cartesian Product

- Cartesian product operator is used to combine information from any two relations.
- It is denoted by (X) symbol.
- For example: We want to list the names of employees with all departments of tables 'A'and'B'.

AB

Name	Emp_No	Dept_Id
Akhil	101	11
Monika	102	12
Aastha	101	11

Dept_Name	Dept_Id
Production	11
Accounts	12

Formula: πName (A) X π Dept_Name (B)

AXB

Name	Dept_Name
Akhil	Production
Akhil	Accounts
Monika	Accounts
Monika	Production
Aastha	Production
Aastha	Accounts

II. Special Operators

(a) Selection Operator

(b) Projection Operator

(c) Join Operator

(d) Division Operator

(a) Selection Operator

- Selection operator selects tuples (rows) that satisfy a given condition.
- It is denoted by lower Greek letter sigma (a).
- We can also use folio wing symbols: = >,<>>=,<= #

- For example: We want to list the tuples (employees) who live in city 'chd'.

  Formula: σcity = "chd" (employee)

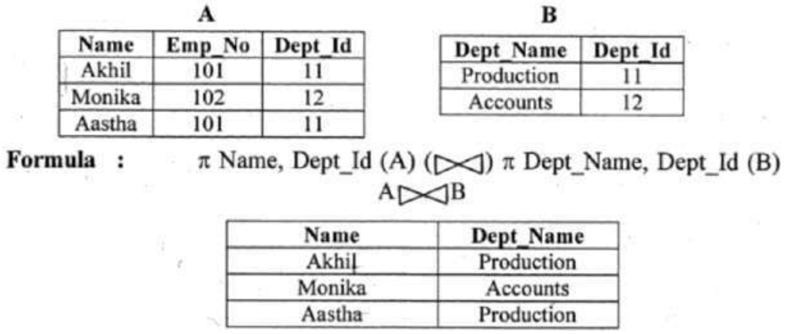
(b) Projection Operator

- Projection operator returns a new relation as output with certain attributes.
- It is denoted by Greek letter pie (π).
- For example: We want to list all the emp_no and name of employee.

Formula: πemp_no, name (employee)

(c) Join Operator

- Join operator is also known as natural join operator.
- It is denoted by the symbol ([><]).
- Cartesian product operator is used to combine two tables, but the output of Cartesianproduct is not correct
- Join operator is used to combine the two tables instead of Cartesian product operator.
- For example: We want to combine the two tables 'A'and 'B'.



A

Name	Emp_No	Dept_Id
Akhil	101	11
Monika	102	12
Aastha	101	11

B

Dept_Name	Dept_Id
Production	11
Accounts	12

Formula : π Name, Dept_Id (A) (⋈) π Dept_Name, Dept_Id (B)

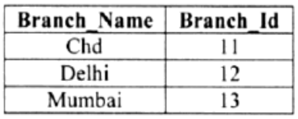
A⋈B

Name	Dept_Name
Akhil	Production
Monika	Accounts
Aastha	Production

(d) Division Operator

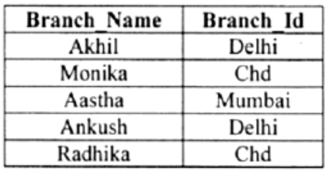
- Division operator will work on two relations (tables).
- It make another relation consisting of values of an attribute of one relation that match all the values in the another relation.
- It is denoted by the (÷) symbol.

A



Branch_Name	Branch_Id
Chd	11
Delhi	12
Mumbai	13

B



Branch_Name	Branch_Id
Akhil	Delhi
Monika	Chd
Aastha	Mumbai
Ankush	Delhi
Radhika	Chd

Formula: πName (A ÷B)

(A ÷B)

Name
Akhil

Q4. Explain relational calculus in detail.

1. It was first proposed by E.F.Codd.

2. It is a formal language used to symbolize logical arguments in mathematics.

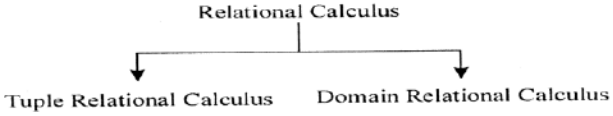
3. In relational calculus, query is expressed as formula containing number of variables and expression.

4. User will only tell the requirement without knowing the methods of retrieval.

5. User is not concerned with the procedure to obtain the results.

6. It is the responsibility of DBMS to transform these queries and give the result to the user.

7. Relational calculus is of mainly two types which are as follows:



Classification of Relation Calculus

I. Tuple Oriented Relational Calculus

- It is based on specifying a number of tuples variables.

- The query of tuple relational calculus is

  {t/COND(t)}

  t-> is tuple variable

  COND (t)->is conditional expression.

- The result of such query is a relation that contains all the types (rows) that satisfy condition COND (t).

  Query of relational calculus is:

  {t. title, t. author/Book(t) and t. PRICE > 100}

  It will give us title, author of all the books whose price is greater than 100.

  Expression of tuple relational calculus is:

  {t1. A1, t2.A2, t3.t3,…tn.An/COND (t1, t2, t3, …tn)}

  t1, t2 …. are tuple variables.

  A1, A2 … are the attributes of relations.

  COND is condition.

II. Domain oriented relational calculus

- Domain calculus is different from tuple calculus in the type of variables used in formula.
- In domain oriented relational calculus, variable range will be single value rather than multiple values.
- Expression of domain oriented relational calculus is:

  {X1, X2, …Xn | COND (X1, X2, … Xn)}

  X1, X2, …Xn are domain variables.

  COND is condition or formula of domain relation calculus.

  i.e. Get employee no. of for job clerk

  EX where EMP (emp no: EX, job = 'clerk')

  Get employee name that belongs to dept no. 10 and having salary > 2000.

  Ex where EMP (ename: EX, deptno = 10, sal> 2000)

Q5. What is INF? Give example to demonstrate how INF improves a table.

1. E.F. Cold introduced the first normal form (1NF) in 1970.
2. First normal form (1NF) eliminates the repeating columns from an un-normalized table.
3. In 1NF, there is no repeating column (group).
4. We convert un-normalized table into normalized for.
5. Primary key is required in each table to identify a record.
6. The purpose of primary key is to uniquely identify a record.
7. First normal from depends on the functional dependency.
8. Formula : f(x)=y

   For every value of x, there is only one value for y.
9. For example: The following table "Student" having columns (Name, Course, Roll Number) is an un-normalized table. We have to convert this un-normalized table into normalized table.

Student

Name	**Course**	**Roll Number**
Akhil	Science	211, 128
Monika	Computer	129
Aastha	Management	112

The above table "Student" is un-normalized because it contains more than one value for the column 'Roll Number'. 'Akhil' has two values (211, 128) for the column 'roll number' which is not possible. For normalization, there should be only one value in one column.

The following are two methods to convert un-normalized table into normalized table:

- **Method 1:** To convert the un-normalized table "Student" into normalized form, we decompose (divide) this un-normalized table into two tables.

Student 1

Name	**Course**
Akhil	Science
Monika	Computer
A'astha	Management

Student 2

Name	**Roll Number**
Akhil	211
Akhil	128
Monika	129
Aastha	112

- **Method 2:** To convert the un-normalized table "Student" into normalized form, we convert this this un-normalized table into flat table.

Student

Name	**Course**	**Roll Number**
Akhil	Science	211
Akhil	Science	128
Monika	Computer	129
Aastha	Management	112

Q6. Discuss 2NF. Discuss the problems that can be encountered in a table, which is

in INF, How 2NF solve them?

E.F. Codd introduced the second normal form (2NF) in 1971.

2. A relation is in 2NF if it fulfills the following conditions

- Relation should be in INF and
- Every non-key attribute (non-prime attribute) is fully functionally dependent on Primary key.

3. For example-.The following table "Products" having columns (Item, Price, Quantity, Order Number, and Order Date) is in INF.

Products

Item	Price	Quantity	Order Number	Order Date
Mobile	2000	20	11	1-7-2015
Sunglasses	1000	15	12	2-7-2015
Watch	800	18	13	3-7-2015
Wallet	600	12	14	4-7-2015

- The table "Products" has two primary key columns (Item and Order Number).
- Price (non-primary key column) is fully functionally dependent on Item (prime key column).
- Order Date (non-primary key column) is fully functionally dependent on Order Number (prime key column).
- The table "Products" can be converted into second normal form (2NF) by decomposing it into sub tables such as:

Item	Price
Mobile	2000
Sunglasses	1000
Watch	800
Wallet	600

Order Number	Order Date
11	1-7-2015
12	2-7-2015
13	3-7-2015
14	4-7-2015

Item	Quantity	Number
Mobile	20	11
Sunglasses	15	12
Watch	18	13
Wallet	12	14

Q7. What is a lock? Differentiate between exclusive and shared lock. Give suitable examples also.

1. A lock is a variable associated with the data item to describe its status.
2. Locks are used in concurrent transactions to ensure serializability.
3. It prevents undesired or inconsistent operations on shared resources by other current transactions.
4. They are used to make the isolation property of transaction in the concurrent environment.
5. They describe the status of the data item whether it has been modified or not.
6. A lock on any database object needs to be acquired by the transaction before accessing it.
7. If transaction ‘A’ acquires a lock on a database object and another transaction 'B'

needs to access that database object, then the existing type of lock is checked.

8. According to the locking scheme, if the existing type of lock (transaction 'A') is matched with another transaction's lock (transaction ^{C}B'), then transaction ^{4}B' can use that object.

9. But, if the existing type of lock (transaction 'A') is not matched with another transaction's lock (transaction 'B'), then transaction attempting access is aborted or blocked.

10. There are many types of locks but only one lock is used for each item in database.

Shared Locks

1. In a binary lock, only one transaction can get the lock on a particular data item. But in shared lock, more than one transaction can use shared fock at a particular time.

2. It is denoted by 'S'.

3. Shared lock is used only for reading purpose. It means, if a transaction want to read data then it will use shared lock on it.

4. Read lock is a shared lock. It means multiple transactions can have read lock on the same item in order to read it.

5. If a transaction 'A' has a shared lock on data item 'M', then other transaction 'B' can only read that data item 'M' not write.

6. For example:

   Lock_S (M): → It is used to request a shared lock on data item 'M'.

   Unlock (M): → It is used to unlock data item 'M'.

Exclusive Locks

1. In a binary lock, only one transaction can get the lock on a particular data item. But in exclusive look, more than one transaction can use exclusive lock at a

particular time.

2. It is denoted by 'X'

3. Exclusive lock is used only for writing purpose. It means, if a transaction want to write data then it will use exclusive lock on it.

4. Write tock is an exclusive lock. It means multiple transactions can have write lock on the same item in order to write it.

5. If a transaction 'T1' has obtains an exclusive lock on a data item then another transaction 'T2' cannot perform read but performs write operation.

6. If a transaction 'A' has a exclusive lock on data item 'M', then other transaction 'B' can only write that data item 'M' not read.

7. For example:

   Lock_X (M): → It is used to request an exclusive lock on data item 'M'.

   Unlock (M): →It is used to unlock data item 'M'.

Compatibility of Locks

Compatibility of Locks	**Shared**	**Exclusive**
Shared	True	False
Exclusive	False	False

1. **Shared lock is compatible with shared lock:** According to this, more than one transaction can read a data item. It means multiple transactions can have read lock on the same item in order to read it.

2. **Shared lock is not compatible with exclusive lock:** According to this, if a data item has exclusive lock, then no other transaction can make shared lock on that particular data item.

3. **Exclusive lock is not Compatible with exclusive** lock: According to this, if a data item has exclusive lock, then no other transaction can make exclusive lock on

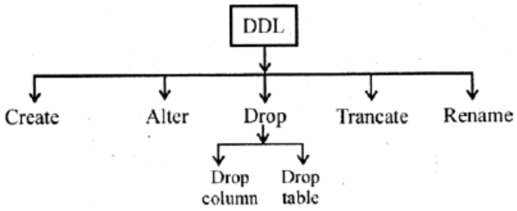
that particular data item. No two transactions can make exclusive lock simultaneously.

Q8. Explain DDC (Data Definition Language) commands in detail with suitable examples.

It is used for defining data structures. These SQL commands are used for creating, modifying and dropping the structure of database objects (relations).

These commands basically create, modify and drop the relations (tables) used in the database.

The following are the various DDL commands:



Parts of DDL

1. Create: The create table command is used to create a new table. It creates the relation (table) in a database. It includes its name, names and attributes of its columns. One can create any number of columns with this command. If we want to add or remove the columns after creating the table then we use alter table.

Syntax of Create New Table:->

```
SQL>CRE^VTE TABLE table_name
(
column_name1 data type,
column_name2 data type,
…….
column_nameN datatype
);
```

Note: We can also create a table from existing table by copying the existing table's column.

Syntax of Create Table from Existing Table:->

```
SQL> CREATE TABLE new_table
        As (SELECT * from old_table);
```

Examples of Create Command:-

1. We want to create a table 'STUD' in SQL.

   Then the query will be:

   ```
   SQL> CREATE TABLE STUD
   (
   NAME char (40),
   CLASS char (5),
   ROLL NUMBER (8)
   );
   Table created
   ```

2. We want to create a table 'BMP' in SQL. ***(Mostly queries of this book are based on this table 'EMP')***

```
SQL> CREATE TABLE EMP
(
ENAME char (15),
DEPTNO int,
JOB char (10),
EMPNO int,
SAL int,
```

```
HIREDATE int,
MGR int,
CITY char (10),
COMM int
);
Table created
```

2. **Alter:** It alters the structure of table from database. It alters the table along with the columns. One can add one more than one column in a particular table with alter command. With this command, filed type can be changed or a new field can be added. It is used to enable or disable the integrity constraint. It is used to modify the column values and constraints.

Syntax of Alter Command:

```
SQL> ALTER TABLE table_name
ADD/MODIFY/DROP column_name datatype;
```

Examples of Alter **Command:**

1. To add a column (DOB) in an existing table 'BMP'. Then the query will be:

```
SQL> ALTER TABLE EMP
ADD DOB date;
Table altered
```

2. To add multiples columns (DOB and MOBNO) to an existing table 'EMP'. Then the query will be:

```
SQL>ALTER TABLE EMP
ADD (DOB date, MOBNO (11));
Table altered
```

3. Drop: With the drop command, we can drop the columns from table or we can remove the table. It drops the column or constraints from the table. It deletes the string of a table. It cannot be recovered. It use with caution. Drop operation is used with the alter table command. It removes single column or multiple columns.

(a) Dropping Column: If we want to remove column, then we use drop operation with alter table command.

Syntax of Dropping the Column:

```
SQL>ALTER TABLE table_name
DROP COLUMN column_name;
```

Examples of Dropping the Column:

1. To drop a column 'City' in an existing table 'EMP'. Then the query will be:

```
SQL>ALTER TABLE EMP
DROP COLUMN CITY;
Table altered.
```

2. To drop multiple columns (Hiredate and City) in an existing table 'EMP'. Then the query will be:

```
SQL>ALTER TABLE EMP
DROP COLUMN (HIREDATE, CITY);
Table altered.
```

(b) Dropping Table: If we want to remove the table, then there is no need to use it with alter table command. We can directly remove one or more columns with drop table command.

- This command removes one or more table definitions and all data, indexes, triggers, constraints and permission specifications.
- If we drop a table with drop table command, it deletes all rows from that

particular table. The table structure is also removed from the database and it cannot get back.

Syntax of Dropping the Table

SQL> DROP TABLE table_name;

4. Truncate: It removes all the records from a table and memory. It releases the memory occupied by the records of the table. Data cannot be recovered after using the truncate command. Truncate command removes all the rows from a table.

Syntax of Truncate Command:

SQL> TRUNCATE TABLE table_name;

Example of Truncate Command:

We want to delete all rows from the table 'EMP'. Then the query will be:

SQL> TRUNCATE TABLE EMP;

5. Rename: It is used to rename the old table with a new name. The data will remain same, only name of table will be change with 'Rename Command'.

Syntax of Rename Command:

SQL> RENAME <Old Table_Name>to<New Table_Name>;

Example of Rename Command:

If we want to change the name of table 'EMP' to new name 'EMPLOYEE'. Then the query will be:

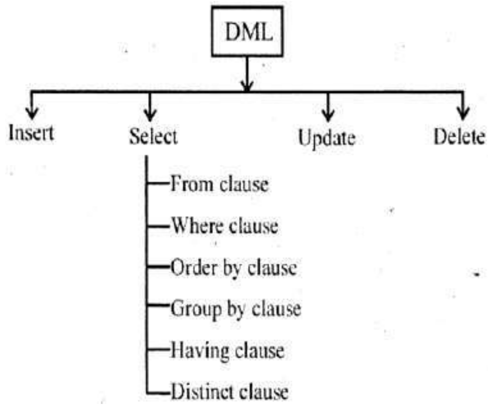
SQL>RENAME EMP TO EMPLOYEE;

Note: We use drop command for tables and delete command for records.

Q9. Explain DML (Data Manipulation Language) commands in detail with suitable examples.

- These commands are used for inserting, retrieving, deleting and modifying the data in a relation or a table.

- It includes the query language based on both relational algebra and tuple relation.
- These commands do not implicitly commit the current transaction.
- The folio wing are the various DML commands:



Parts of DML

1. Insert

- When a new table is created, there is no data in the table.
- Insert command is used to insert the records in the new table.
- Insert command is used to add records to an existing table.
- 'Values clause' is used with inset command. This command will insert value in all the columns of a table in sequence.

Syntax of Insert Command:

```
SQL> INSERT INTO table_name
         VALUES (value1, valueZ, valueS,.....);
```

OR

```
SQL> INSERT INTO table_name (column1, column2, column3,......)
         VALUES (value1, value2, value3,....);
```

Examples of Insert Command:

1. Insert record in different order. Then the query will be:

```
SQL> INSERT INTO EMP (name, city, salary, emp_no)
```

```
VALUES ('Mona','Nba', 4500, 4);
```

2. Insert the Null value in record. Then the query will be:

```
SQL> INSERT INTO EMP
VALUES (3,'Mona', Null, 4000);
```

3. Insert the records in selected columns. Then the query will be:

```
SQL> INSERT INTO EMP (name, city)
VALUES ('Mona', 5000);
```

4. Insert the values in the table 'EMP'. Then the query will be:

```
SQL> INSERT INTO EMP VALUES ('Nidhi',20,'Clerk',6258,900,9-5-83,
6801,'Chd');

SQL> INSERT INTO EMP VALUES ('Aastha',30,'SaIesman',6388,1500,1-
12- 89, 6587, 'Delhi', 300);

SQL> INSERT INTO EMP VALUES ('Sachin',30,'Salesman',6410,1350,25-1-
92,6587,'Pta',500);

SQL> INSERT INTO EMP VALUES ('Rohit',20,'Manager',6455,2875,27-12-
91,6728,'Nba');

SQL> INSERT INTO EMP VALUES ('Rahul',30,'Salesman',6543,1350,28-5-
87,6587,'Nba',1400);

SQL> INSERT INTO EMP VALUES ('Aditya',30,'Manager',6587,2750,17-8-
86,6728,'Pta');

SQL> INSERT INTO EMP VALUES ('Siddharth',10,'Manager',6671,
2550,29-9-  80,6728,'Chd',Null);

SQL> INSERT INTO EMP VALUES ('Kunar,20,'Analyst',6677,3000,8-12-82,
6455,'Delhi',Null);

SQL> INSERT INTO EMP VALUES ('AkhiP,10,'President',6728,5000,2-11-
85,Null,'DeIhi',NulI);

SQL> INSERT INTO EMP VALUES ('Prathiba',30,'Salesman',6733,1600,4-6- 85,6587,'Pta',0);

SQL> INSERT INTO EMP VALUES ('Manmeet',20,'Clerk',6765,1050,11-1-84,6677;'Ldh',Null);

SQL> INSERT INTO EMP VALUES ('Navreet',30,'Clerk',6800,950,25-3-84,6587,'Pta',Null);

SQL> INSERT INTO EMP VALUES ('Saira',20,'Analyst',6801,3000,15-4-80,6455,'Chd',Null);

SQL> INSERT INTO EMP VALUES ('Amit',10,'Clerk',6823,1400,25-8-85,6671,'Ldh',Null);
```

After inserting, values, the table 'EMP' will look like:

EMP

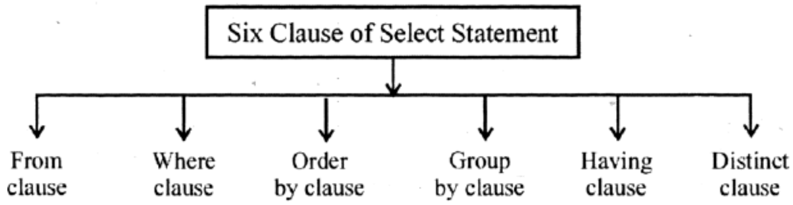
ENAME	DEPTNO	JOB	EMPNO	SAL	HIREDATE	MGR	CITY	COMM
Nidhi	20	Clerk	6258	900	9-5-83	6801	Chd	
Aastha	30	Salesman	6388	1500	1-12-89	6587	Delhi	300
Sachin	30	Salesman	6410	1350	25-1-92	6587	Pta	500
Rohit	20	Manager	6455	2875	27-12-91	6728	Nba	
Rahul	30	Salesman .	6543	1350	28-5-87	6587	Nba	1400
Aditya	30	Manager	6587	2750	17-8-86	6728	Pta	
Siddharth	10	Manager	6671	2550	29-9-80	6728	Chd	
Kunal	20	Analyst	6677	3000	8-12-82	6455	Delhi	
Akhil	10	President	6728	5000	2-11-85		Delhi,	
Prathiba	30 :	Salesman	6733	1600	4-6-85	6587	Pta	0
Manmeet	20	Clerk	6765	1050	11-1-84	6677	Ldh	
Navrget	30	Clerk	6800	950	25-3-84	6587	Pta	

Saira	20	Analyst	6801	3000	15-4-80	6455	Chd	
Amit	10	Clerk	6823	1400	25-8-85	6671	Ldh	

NOTE: ***(Mostly queries are based on this table*** *'EMP')*

2. **Select:** Once data in inserted into a table, the next step is to view the data contained in the table.

- In order to view the data contained in the table, the select statement is used.
- Select statement is a powerful tool and a most commonly used command.
- It is used to retrieve the data from a table in a database.
- We can also use arithmetic operators in select statement (see example 4, 5 and 6 of select statement).
- With the help of select command, one can retrieve information from one column or more than one column.
- The basic select statement has 6 clauses which are as follows:



Six Clauses of Select Statement

(a) **Select:** The select clause specifies the table columns that are retrieved. It always use with 'From Clause'.

Syntax of Select Command:

```
SQL> SELECT * FROM table_name;

                    OR

SQL> SELECT column_list FROM table_name
        [WHERE Clause]
        [GROUP BY Clause]
```

```
            [HAVING Clause]
            [ORDER BY Clause];
```

(b) **From:** From clause specifies the table accessed. It is mandatory. It always use with 'Select Command'.

Syntax of From Clause:

```
SQL> SELECT.* FROM table_name;
          OR
SQL> SELECT column_list FROM table_name
     [Where Clause]
     [Group By Clause]
     [Having Clause]
     [Order By Clause];
```

(c) **Where:** Where clause is used when we want to retrieve the specific information from a relation excluding other irrelevant data.

Syntax of Where Clause:

```
SQL> SELECT column_list FROM table_name
     [WHERE Clause];
```

Examples of'-Select Command', 'From Clause' and 'Where Clause':

1. Display all the information of all the employees from relation 'EMP'. Then the query will be:

   ```
   SQL> SELECT * FROM EMP;
   ```

Result:

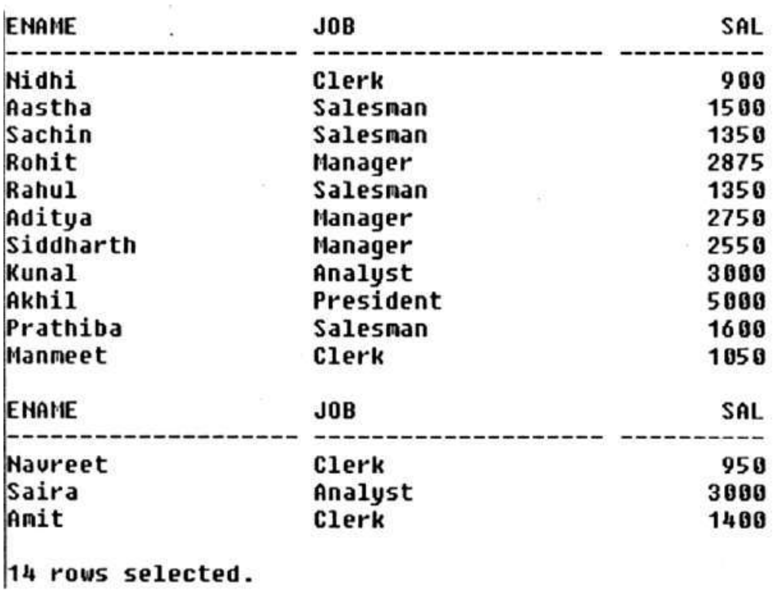
EMP

ENAME	DEPTNO	JOB	EMPNO	SAL	HIREDATE	MGR	CITY	COMM
Nidhi	20	Clerk	6258	100	9-5-83	6801	Chd	
Aastha	30	Salesman	6388	1500	1-12-89	6587	Delhi	300
Sachin	30	Salesman	6410	1350	25-1-92	6587	Pta	500
Rohit	20	Manager	6455	2875	27-12-91	6728	Nba	
Rahul	30	Salesman	6543	1350	28-5-87	6587	Nba	1400
Aditya	30	Manager	6587	2750	17-8-86	6728	Pta	
Siddharth	10	Manager	6671	2550	29-9-80	6728	Chd	
Kunal	20	Analyst	6677	3000	8-12-82	6455	Delhi	
Akhil	10	President	6728	5000	2-11-85		Delhi	
Prathiba	30	Salesman	6733	1600	4-6-85	6587	Pta	0
Manmeet	20	Clerk	6765	1050	11-1-84	6677	Ldh	
Navreet	30	Clerk	6800	950	25-3-84	6587	Pta	
Saira	20	Analyst	6801	3000	15-4-80	6455	Chd	
Amit	10	Clerk	6823	1400	25-8-85	6671	Ldh	

2. Display only the name, job and salary of all the employees from table "EMP”. Then the query will be;

```
SQL> Select ENAME, JOB, SAL
        From EMP;
```

Result:



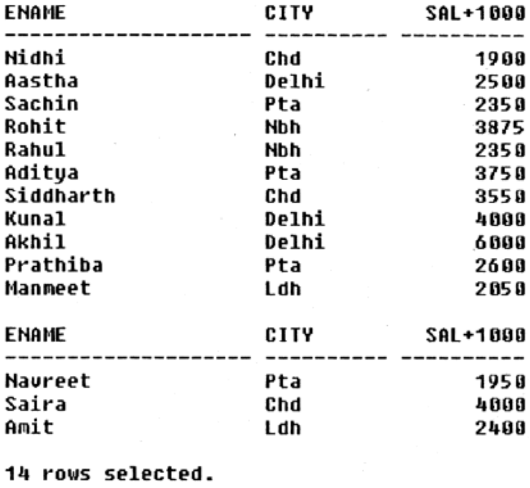
```
ENAME                JOB                         SAL
-------------------- -------------------- ----------
Nidhi                Clerk                       900
Aastha               Salesman                   1500
Sachin               Salesman                   1350
Rohit                Manager                    2875
Rahul                Salesman                   1350
Aditya               Manager                    2750
Siddharth            Manager                    2550
Kunal                Analyst                    3000
Akhil                President                  5000
Prathiba             Salesman                   1600
Manmeet              Clerk                      1050

ENAME                JOB                         SAL
-------------------- -------------------- ----------
Navreet              Clerk                       950
Saira                Analyst                    3000
Amit                 Clerk                      1400

14 rows selected.
```

3. Display name, city and salary of employees from relation 'EMP' where salary of each employee is increased by 1000. Then the query will be:

SQL> SELECT ENAME, CITY, SAL + 1000

FROM EMP;

Result:



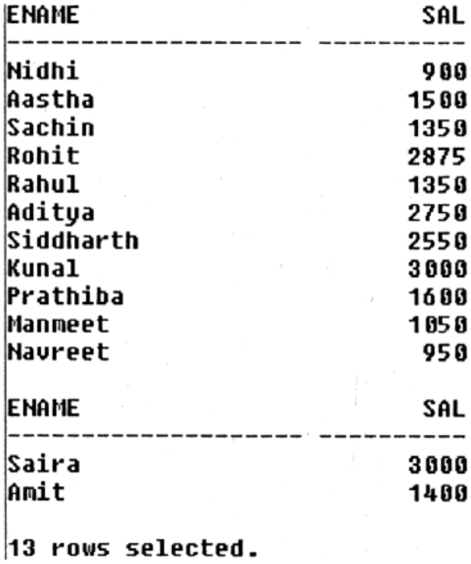
```
ENAME                CITY         SAL+1000
-------------------- ---------- ----------
Nidhi                Chd              1900
Aastha               Delhi            2500
Sachin               Pta              2350
Rohit                Nbh              3875
Rahul                Nbh              2350
Aditya               Pta              3750
Siddharth            Chd              3550
Kunal                Delhi            4000
Akhil                Delhi            6000
Prathiba             Pta              2600
Manmeet              Ldh              2050

ENAME                CITY         SAL+1000
-------------------- ---------- ----------
Navreet              Pta              1950
Saira                Chd              4000
Amit                 Ldh              2400

14 rows selected.
```

4. Display the name and salary of employees whose salary is less than 5000. Then the query will be:

SQL> SELECT ENAME, SAL from EMP

WHERE SAL <5000;

Result:



```
ENAME                       SAL
-------------------- ----------
Nidhi                       900
Aastha                     1500
Sachin                     1350
Rohit                      2875
Rahul                      1350
Aditya                     2750
Siddharth                  2550
Kunal                      3000
Prathiba                   1600
Manmeet                    1050
Navreet                     950

ENAME                       SAL
-------------------- ----------
Saira                      3000
Amit                       1400

13 rows selected.
```

5. Display the names of all the employees belonging to the department number 10 from the relation 'BMP'. Then the query will be:

SQL>SELECT ENAME FROM EMP

WHERE DEPTNO = 10;

Result:

```
ENAME
------------------
Siddharth
Akhil
Amit
```

(d) Order By: The 'Order By Clause' is used with 'Select Statement' to sort the results either in ascending or descending order. By default, it provides results in ascending order. We use column values to sort the table. We can use more than one column to sort the results.

Syntax of Order By Clause:

SQL> SELECT column_list FROM table_name

[ORDER BY Clause];

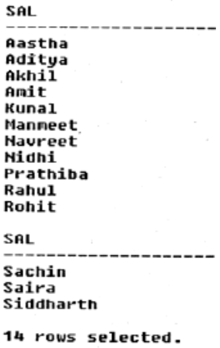
Examples of Order By Clause:

1. Sort the table 'EMP' by the salary of employees. Then the query will be:

SQL>SELECT ENAME SAL FROM EMP

ORDER BY SAL;

Result:



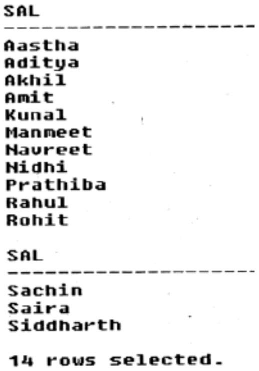
```
SAL
--------------------
Aastha
Aditya
Akhil
Amit
Kunal
Manmeet
Navreet
Nidhi
Prathiba
Rahul
Rohit

SAL
--------------------
Sachin
Saira
Siddharth

14 rows selected.
```

2. Sort the table 'BMP', by the name and salary of employees. Then the query will be:

SQL>SELECT ENAME SAL FROM EMP

ORDER BY ENAME, SAL;

Result:



```
SAL
--------------------
Aastha
Aditya
Akhil
Amit
Kunal
Manmeet
Navreet
Nidhi
Prathiba
Rahul
Rohit

SAL
--------------------
Sachin
Saira
Siddharth

14 rows selected.
```

(e) Group By: It is used to divide the rows into smaller groups. The 'Group By Clause' is used with 'Select Statement' to combine a group of rows based on the values of a particular column or expression. It groups the result after it retrieves the rows from a table. 'Group functions' can be used with 'Having Clause' and cannot be used with 'Where Clause'.

Syntax of Group By Clause:

```
SQL> SELECT column_list FROM table__name
     [GROUP BY Clause];
```

Example of Group By Clause:

To find the total amount of salary spent on each department from the table 'EMP'. Then the query will be:

```
SQL>SEI.ECT DEPTNO, SUM (SAL) AS TOTAL SALARY FROM EMP
     GROUP BY DEPTNO;
```

Group within Group: 'Group By Clause' can be used to provide results for 'Groups Within Groups'. Suppose we want to know the average amount of salary spent on job type 'Clerk' from department number '20'. We calculate the total amount of salary spent on each department. This is one group. Then we calculate the average amount of salary spent on each type of job from that particular department. This is group within group.

Example of Group within Group Clause:

To find the average monthly salary for each job type within department Then the query will be:

```
SQL>SELECT DEPTNO, JOB, AVG (SAL) AS AVERAGE SALARY
     FROM EMP GROUP BY DEPTNO, JOB;
```

(f) Having: It is similar to 'Where Clause', but it is used with group functions. It is used to filter the data. 'Having Clause' can be used with 'Group function' and cannot be used with 'Where Clause'. It restricts the groups that we return on the basis of group functions. It is used to specify which groups are to be displayed.

Syntax of Having Clause:

SQL> SELECT column_list FROM table_name

[HAVING Clause];

Example of 'Having Clause:

To find the department who has paid the total salary more than 8.00.6 to its employees. Then the query will be:

SQL>SELECT DEPTNO, SUM (SAL) AS TOTAL SALARY FROM EMP GROUP BY DEPTNO

HAVING SUM (SAL)>8000;

(g) **Distinct Clause:** The 'Distinct Clause' is used with 'Select Statement' to suppress the duplicate values if any in a column.

Example of 'Distinct Clause':

Display all the different jobs available in the table 'EMP'. Then the query will be:

SQL>SELECT DISTINCT JOB FROM EMP;

Result:

```
JOB
----------------------------
President
Clerk
Analyst
Salesman
Manager
```

3. Update

- Update command is used when there is a need to modify the data in a table.
- It is used to update existing records in a table.

- It updates single record or multiple records in a table.

Syntax of Update Command:

```
SQL> UPDATE table_name
        SET column1 = value, column2 = value2, ......
        WHERE some_column = some_value;
```

Examples of Update Command:

1. To give everybody a commission of Rs. 100 in the table 'EMP'. Then the query will be:

```
SQL>UPDATE EMP
        SET COMM = 100;
```

2. Update the Manager's salary to 8000 of department number 10 in the table 'EMP'. Then the query will be:

```
SQL>UPDATE EMP
        SET SAL = 8000
        WHERE JOB = 'Manager' AND DEPTNO = 10;
```

4. Delete

- It deletes one or more records from a table and sends it to recycle.
- It doesn't release the memory occupied by the records of the table. Data can be recovered.
- If any subset is defined with condition, then specific records or rows, are deleted, otherwise all records are deleted.
- Executing a delete command may cause triggers to rum which may cause deletion in other tables.
- Example: Sometimes two tables are linked by the foreign key. If we delete rows in one table, then we have to delete those rows from the second table to maintain the referential integrity.

Syntax of Delete Command:

```
SQL> DELETE FROM table_name [where condition];
                              OR
SQL> DELETE from table_name;
```

Examples of Delete Command:

1. Delete all the records of 'Manager' from the table 'EMP'. Then the query will be:

```
SQL>DELETE FROM EMP
WHERE JOB = 'Manager';
```

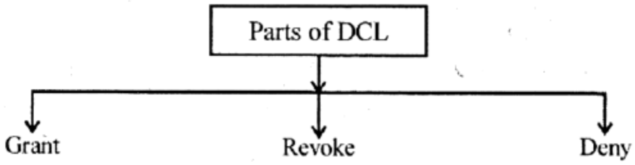
2. Delete all the records from the table 'EMP'. Then the query will be:

```
SQL>DELETE FROM EMP;
```

Q10. Explain DCL (Data Control Language) commands in detail with suitable examples.

- It is used to control access to data in a database. It also controls the security of the database.
- To control data in a database, privileges are given to user to access the data without any problem and with proper security.
- It basically provides security to database. Without privileges, no one can access the database.
- A user can access the database according to the privileges given to him.

The following are the various DCL commands:



Parts of DCL

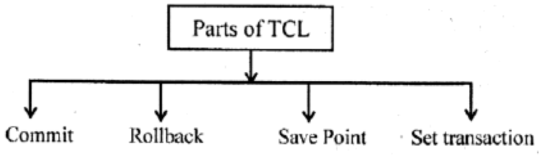
(a) **Grant:** It is used to give the permission to the user for restricted access to the database. It allows specified users to perform specified tasks.

(b) **Revoke:** It is used to cancel the previously granted or denied permissions to the users.

(c) **Deny:** It disallows the specified users from performing specified tasks.

Q11. Explain TCL (Transaction Control Language) commands in detail with suitable examples.

- TCL is used to manage the changes made by DML (data manipulation language) statements.
- These commands are used for revoking the transactions and to make the data commit to the database.
- Basically, it is used to manage the different transactions occurring within a database.
- Each transaction is completely isolated from other active transactions.
- User can make changes in the particular transaction in database with the transaction control language.
- At the end of the transaction, the database can make all the changes permanent in the database or undoes them all.
- If any problem fails in the middle of a transaction, then the database rolls back the transaction and restore the database into its former state.
- The following are the various TCL commands:



Parts of TCL

(a) **Commit**

- Commit command is used to save work done. The changes made in the database by the user are not visible to other users until they become permanent in the database.

- Commit command is used to permanent any changes made to the database during the current transaction by the user.
- Commit command is used to save all the changes made to the database since the last commit or rollback command.

Syntax of Commit Command:

```
SQL> COMMIT;
```

Example of Commit Command:

To delete the records of the employees permanently, belonging to the city 'Chd'.

```
SQL>DELETE FROM EMP
WHERE CITY = 'Chd';
SQL>COMMIT;
```

(b) Rollback

- It is used to restore the database to its original state since the last 'commit'.
- It is the inverse of the commit statement.
- It is used to undo the transactions that have not already been saved to the database.
- Oracle provides a facility to-roll back to the last committed state.

Example: We are performing the operations on the database and some problem occurs into the computer system. Yet we have not performed the commit statement, and then rollback command helps to come back to the last committed state.

Syntax of Rollback Command:

```
SQL> ROLLBACK;
```

(c) Savepoint

- Savepoint command is used to identify a point in a transaction from which we can later rollback.

- The Savepoint statement defines a Savepoint within a transaction.
- It is a special mark inside a transaction that allows all commands that are executed after it was established to be rolled back, restoring the transaction state to what it was at the time of Savepoint.
- Changes made after a Savepoint can be undone at any time prior to the end of the transaction.
- A transaction can have multiple savepoints.

Syntax of Savepoint Command:

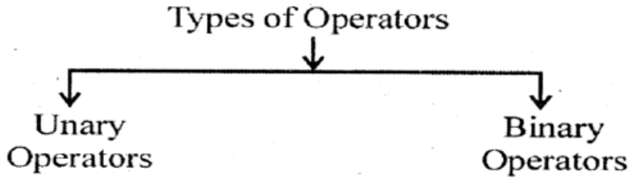
SQL> SAVEPOINT<savepoint name>;

(d) Set Transaction

- Set transaction command has no effect on any subsequent transactions.
- It is used to set the characteristics of the current transaction.
- This command is helpful to determine whether the transaction is read/write or read only.
- If a transaction is read only, then the insert, update, delete and copy commands are disallowed.

Q12. Discuss SQL Operators in detail with suitable examples.

- SQL supports a wide variety of operators. These operators are extensively used in SQL statements used by the user for the purpose of issuing a query to the database.
- The operators are mainly used in the Where clause, Having clause to filter the data to be selected.
- An operator is a symbol which is used to manipulate the data items (operands).



Types of Operators

- Operators are represented by keywords or by special characters.

On the basis-of operands, there are two types-of operators:

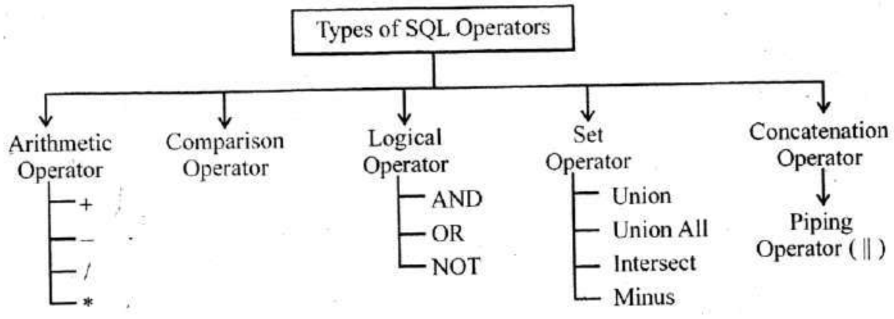
Unary Operator: *An* unary, operator operates on only one operand.

Format □ *operator* operand.

Binary Operator: *A* binary operator operates on two operands.

Format □ operand1 *operator* operand 2

The following are the various SQL operators:



Types of SQL Operators

Arithmetic Operator

- An arithmetic operator is used to add, subtract, multiply and divide the numeric values in an expression.
- It is used to perform the mathematical operations on one or more data items or operands of numeric data type.
- It also provides results in numeric values.

Sr. No.	Arithmetic Operator	Description
1	+	Used for addition in SQL
2	-	Used for subtraction in SQL
3	/	Used for division in SQL
4	*	Used for multiplication in SQL

Examples of Arithmetic Operator:

1. Add

Add Rs.500 in the employee's salary whose EMPNO is 6258 from the relation 'EMP'. Then the query will be:

```
SQL> SELECT SAL, SAL+500 FROM EMP
        WHERE EMPNO = 6258;
```

Result:

```
SAL       SAL + 500
------------------------------
900       1400
```

2. Subtract

Subtract the employee's commission from his salary whose EMPNO is 6388. Then the query will be:

```
SQL> SELECT SAL, SAL-COMM FROM EMP
        WHERE EMPNO = 6388;
```

Result:

```
SAL       SAL-COMM
------------------------------
1500      1200
```

3. Multiply

Multiply the salary of employee by 100 whose EMPNO is 6258 from the relation 'EMP'. Then the query will be:

```
SQL> SELECT SAL, SAL* 100 FROM EMP
        WHERE EMPNO = 6258;
```

Result:

```
SAL       SAL * 100
------------------------------
900       90000
```

Comparison Operator

- A comparison operator is used to compare the column data with specific values with the other column data values.
- It is also used along with the Select Statement to filter data based on specific conditions.

Sr. No.	Comparison Operator	Description
1	=	Equal to
2	!= OR o	Not equal to
3	<	Less than
4	>	Greater than
5	<=	Less than or equal to
6	>=	Greater than or equal to
7	LIKE	Performs pattern matching from columns. The LIKE operator is- used only with Char and match a pattern. % represents sequence of zero or more character.
8	IN	To check a value within a set. It is used to compare a column with more than one value.
9	BETWEEN	To check value within a range. It is used to compare data for a range of value.

10	ANY	To check whether one or more rows in the result set of a sub query meet the specified, condition
11	ALL	To check whether all rows in the result set of a sub query meet the specified condition.
12	EXISTS	To check whether a sub query returns any result.

Example of Equal to (=) Operator:

Display the records of the employees, who live in city 'Chd', from the relation 'EMP'. Then the query will be:

```
SQL> SELECT * FROM EMP
     WHERE CITY = 'Chd';
```

Result:

ENAME	DEPTNO	JOB	EMPNO	SAL	HIREDATE	MGR	CITY
Nidhi	20	Clerk	6258	900	9-5-83	6801	Chd
Siddharth	10	Manager	6671	2550	29-9-80	6728	Chd
Saira	20	Analyst	6801	3000	15-4-80	6455	Chd

Example of Not Equal to (!= OR <>) Operator:

Display the records of the employees, whose city is not equal to 'Chd', from the relation 'EMP'. Then the query will be:

```
SQL> SELECT * FROM EMP
     WHERE CITY! = 'Chd';
```

Result:

ENAME	DEPTNO	JOB	EMPNO	SAL	HIREDATE	MGR	CITY	COMM
Aastha	30	Salesman	6388	1500	1-12-89	6587	Delhi	300
Sacliin	30	Salesman	6410	1350	25-1-92	6587	Pta	500
Rohit	20	Manager	l3455	2875	27-12-91	6728	Nba	
Rahul	50	Salesman	6543	1350	28-5-87	6587	Nba_	1400
Aditya	30	Manager	6587	2750	17-8-86	6728	Pta	
Kunal	20	Analyst	6677	3000	842-82	6455	Delhi	
Akhil	10	President	6728	5000	2-11-85		Delhi	
Prathiba	30	Salesman	6733	1600	4-6-85	6587	Pta	0
Manmeet	20	Clerk	6765	1050	114-84	6677	Ldh	
Navreet	30	Clerk	6800	950	25-3-84	6587	Pta	
Amit	10	Clerk	6823 .	1400	25-8-85	6671	Ldh	

11 rows selected

Example of Less than (<) Operator:

Display the name of the employees, whose salary is less than '1400', from the table 'EMP'. Then the query will be:

```
SQL> SELECT ENAME FROM EMP
     WHERE SAL = 1400'
```

Result:

```
ENAME
---------------
Nidhi
Sachin
Rahul
Nanmeet
Naureet
```

Example of Greater than (>) Operator:

Display the name of the employees, whose salary is greater than '1400', from the table 'EMP'. Then the query will be:

```
SQL> SELECT ENAME FROM EMP
     WHERE SAI>1400;
```

Result:

```
ENAME
-------------
Nidhi
Sachin
Rahul
Manmeet
```

```
Navreet

Amit

6 rows selected.
```

Example of Less than or equal to (<=) Operator:

Display the name of the employees, whose salary is less than or equal to '1400', from the table 'EMP'. Then the query will be:

SQL> SELECT ENAME FROM EMP

WHERE SAL< =1400;

Result:

```
ENAME

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

Nidhi

Sachin

Rahul

Manmeet

Navreet

Amit

6 rows selected.
```

Example of Greater than (>=) Operator:

Display the name of the employees, whose salary is greater than or equal to '1400', the table 'EMP'. Then the query will be:

SQL> SELECT ENAME FROM EMP

WHERE SAL< =1400;

Result:

```
ENAME
---------------
Aastha
Rohit
Aditya
Siddharth
Kunal
Akhil
Prathiba
Saira
Amit
9 rows selected.
```

Examples of LIKE Operator:

1. Display the employees whose name start with 'S' from the table 'EMP'. Then the query will be:

```
SQL> SELECT ENAME FROM EMP
     WHERE ENAME LIKE 'S%';
```

Result:

```
ENAME
---------------
Sachin
Siddharth
Saira
```

2. Display the employees, whose name ends with 'S', from the table 'EMP'. Then the

query will be:

```
SQL> SELECT ENAME FROM EMP
        WHERE ENAME LIKE '%S';
```

Result:

NO ROW SELECTED.

- Display the employees, where 'S' is in the middle of the name, from the Table 'EMP'.

Then the query will be:

```
SQL> SELECT ENAME FROM EMP
    WHERE ENAME LIKE '%S%';
```

Result:

```
ENAME
--------
Aastha
```

Example of IN Operator:

Display the names of the employees, who are analyst and clerk, from the table 'EMP'. Then the query will be:

```
SQL>SELECT ENAME FROM EMP
        WHERE JOB IN ('Analyst', 'Clerk');
```

Result:

```
ENAME
Nidhi
Kunal
Manmeet
```

```
Navreet
Saira
Amit
6 rows selected.
```

Example of BETWEEN Operator:

Display the name and salary of all employees, whose salary is between 2000 and 3000, from the table 'EMP'. Then the query will be:

```
SQL>SELECT ENAME, SAL FROM EMP
    WHERE SAL BETWEEN 2000 AND 3000;
```

Result:

```
ENAME               SAL
----------------------
Rohit               2875
Aditya              2750
Siddharth           2550
Kunal               3000
Saira               3000
```

Logical Operator

- Logical operators compare two or more than two conditions at a time to determine whether a row can be selected for the output.
- When retrieving data using a Select Statement, we use logical operators in the Where Clause which allows us to combine more than one condition.

Sr. No.	Logical Operator	Description
1	AND	For the row to be selected all the specified conditions must be true.
2	OR	For the row to be selected at least one of the specified conditions must be true.
3	NOT	For the row to be selected, the specified conditions must be false.

- NOT is totally opposite of AND and OR operator. When we want to find those rows that do not satisfy a condition, then we use the NOT operator.

1. Examples of AND Operator:

- To find the names of the clerks from the table "EMP" who are working in the department number 20, then the query will be:

```
SQL> SELECT ENAME FROM EMP
     WHERE NOB = 'CLERK' AND DEPTNO = 20;
```

Result:

```
ENAHE
---------------
Nidhi
Manmeet
```

- To find the Ename, Sal, Job from the table "EMP" where salary is greater than 1500 and deptno is 30, then the query will be:

```
SQL> SELECT ENAME, SAL, JOB FROM EMP
     WHERE SAL>1500 AND DEPTNO = 30;
```

Result:

```
ENAME                         SAL JOB
---------------------------------------------
Rohit                         2175/Manager
Aditya                        2758 Manager
Prathiba                      1600 Salesman
```

- To find all the information of the employee's from the table "EMP" whose job is manager and deptno is 10, then the query will be:

SQL> SELECT * FROM EMP

WHERE JOB = 'Manager' AND DEPTNO = 10;

Result:

ENAME	DEPTON	JOB	EMPNO	SAL	HIREDATE	MGR	CITY
Siddharth	10	Manager	6671	2550	29-9-80	6728	Chd

2. Examples of OR Operator:

- To find the names of the employees from the table "EMP", who are analysts and clerk, then the query will be:

SQL> SELECT ENAME FROM EMP

WHERE JOB = -'Analyst' OR JOB = 'CIerk';

Result:

```
ENAME
---------------
Nidhi
Kunal
```

```
Navreet

Saira

Amit

6 rows selected.
```

- Display the Ename, Empno from the table "EMP", whose job is clerk or deptno is 10, then the query will be:

```
SQL> SELECT ENAME, EMPNO FROM EMP
     WHERE JOB = 'Clerk' .OR DEPTNO = 10;
```

Result:

```
ENAME                       EMPNO
------------------------------
Nidhi                       6258
Siddharth                   6671
Akhil                       6728
Manmeet                     6765
Navreet                     6888
Amit                        6823
6 rows selected.
```

3. NOT

- Display the names of the employees from the table "EMP", who are not clerks, then the query will be:

```
SQL> SELECT ENAME FROM EMP
     WHERE JOB <> 'Clerk';
```

OR

```
SQL> SELECT ENAME FROM EMP
        WHERE JOB! = 'Clerk';
```

Result:

```
ENAME
--------------
Aastha
Sachin
Rohit
Rahul
Aditya
Siddharth
Kunal
Akhil
Prathiba
Saira
10 rows selected.
```

- Display the name and deptno of employees from the table "EMP", who are not belonging to deptno 10 or 20, then the query will be:

```
SQL> SELECT ENAME, DEPTNO FROM EMP
    WHERE NOT (DEPTNO = 10 OR DEPTNO = 20);
```

Result:

```
ENAME                    DEPTNO
-----------------------------
```

```
Aastha                 30

Sachin                 30

Rohit                  30

Rahul                  30

Aditya                 30

Prathiba               30

Navreet                30

7 rows selected.
```

Set Operator

- Set operators are used to combine the results from two or more Select statements.
- The result of each Select Statement can be treated as a SET. Set operators are applied on these SETS to achieve the final result.
- Set operators follow some rules which are as follows:
- Number of columns should be in exact same order in all the queries.
- Number of columns should be same in all the queries.
- Data types of retrieved columns (selected statements) should be matched.

```
UNION ALL

SELECT Column List FROM Table2;
```

Example of Union All Operator:

Display all the jobs in department 10 and 20 from the table 'EMP'. Then the query will be:

```
SQL> SELECT JOB FROM EMP

     WHERE DEPTNO = 10

     UNION ALL

     SELECT JOB FROM EMP
```

WHERE DEPTNO = 20;

Result:

```
JOB
---------------
Manager
President
Clerk
Clerk
Analyst
Clerk
Analyst
7 rows selected.
```

NOTE: *Union operator provides results with automatically removal of duplicate values whereas Union All operator provides results without removal of any duplicate value.*

3. Intersect

Intersect operator combine the two table expressions into one and return a result set which consists of rows that appear in the results of both table expressions. It also removes all the duplicate rows from the result set.

Syntax of Intersect Operator:-

```
SQL> SELECT Column List FROM Table 1
     INTERSECT
     SELECT Column List FROM Table2;
```

Example of Intersect Operator:

Display all the jobs common in department 10 and 20 from the table 'EMP'. Then the query will be:

```
SQL> SELECT JOB FROM EMP
        WHERE DEPTNO = 10
        INTERSECT
        SELECT JOB FROM EMP
        WHERE DEPTNO = 20;
```

Syntax: ->

SQL><SELECT STATEMENT><SET OPERATOR>< SELECT STATEMENT > <ORDER BY Clause>;

Sr. No.	Set Operator	Description
1	Union	Returns all distinct rows selected by either query, excluding all duplicate rows.
2	Union All	Returns all rows selected by either query, including all duplicate rows.
3	Intersect	Returns all distinct rows selected by both queries.
4	Minus	Returns all distinct rows selected by the first query but not the second.

1. Union

It combines the results of two queries (same number of columns and compatible data types) into a single table of all matching rows. Union automatically removes all the duplicate values.

Syntax of Union Operator:

```
SQL> SELECT Column List FROM Table1
        UNION
```

```
                SELECT Column List FROM Table2;
```

Example of Union Operator:

- Display the different jobs in department 10 and 20 from the table 'EMP'. Then the query will be:

```
SQL> SELECT JOB FROM EMP
    WHERE DEPTNO = 10
    UNION
    SELECT JOB FROM EMP
    WHERE DEPTNO = 20;
```

Result:

```
JOB
------------
Analyst
Clerk
Manager
President
```

2. Union All

It combines the results of two queries (same number of columns and compatible data types) into a single table of all matching rows. It includes (shows) all the duplicate values.

Syntax of Union All Operator:

```
    SQL> SELECT Column List FROM Table1
```

Result:

```
JOB
-----------
Clerk
```

4. Minus

It compares each record in statement1 with a record in statement2. It returns the results with the records in statement1 that are not in statement2.

Rows retrieved by the second query are subtracted from the rows retrieved by the first query. Only those records are considered as a result which are present only in statement1 and not in statement2.

Syntax of Minus Operator:-

```
SQL> SELECT Column List FROM Table1
     MINUS
     SELECT Column List FROM Table2;
```

Example of Minus Operator:

Display all the unique jobs in the department 10 from the table 'EMP'. Then the query will be:

```
SQL> SELECT JOB FROM EMP
     WHERE DEPTNO = 10
     MINUS
     SELECT JOB FROM EMP
     WHERE DEPTNO = 20
     MINUS
     SELECT JOB FROM EMP
     WHERE DEPTNO = 30;
```

Result:

```
JOB
-----------------
President
```

Concatenation Operator

- Concatenation operator is used to combine the two or more data strings.
- The operands of the concatenation must be compatible strings.
- Character string cannot be concatenated with a binary string.
- Concat and vertical bars (..) both represent the concatenation operator.

Concatenation Operator	Description
Piping Operator (...)	It is used to combine two or more strings

Examples of Concatenation Operator:

- List the employee salary whose empno is 6728. Then the query will he:

  SQL> Select 'My Salary is =' .... Sal as Salary

  From EMP Where Empno = 6728.

Result: **My Salary is 5000.**

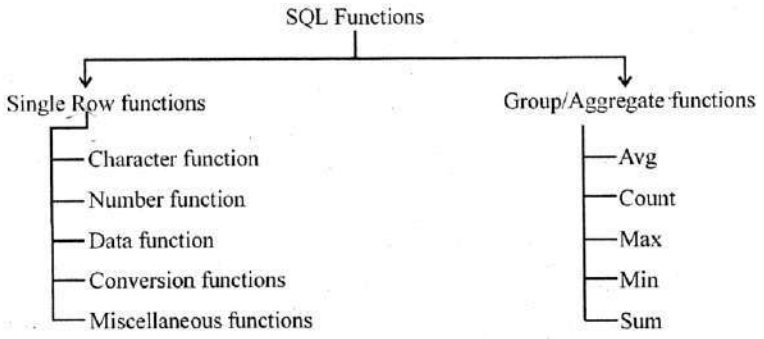
- List the employee name whose empno is 6728. Then the query will be:

  SQL> Select 'My Name is =' ... Ename as Name

  From EMP Where Empno = 6728.

Result: **My Name is Akhil.**

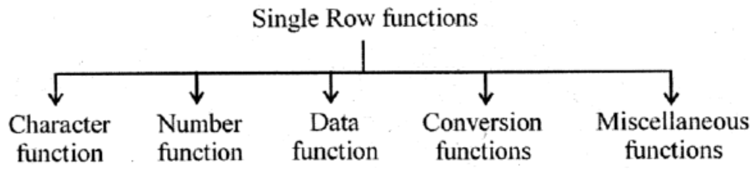
Q13. Explain SQL functions in detail with suitable examples.



SQL Functions

Single Row Functions

Single row functions operate on single rows only and returns one result per row. ***The types of single row functions are as follows:***



Single Row Functions

1. Character Functions

- It is also known as text functions.
- It is used to manipulate text strings.
- It accepts character input only and returns either character or numeric values.

The following are the types of character functions:

(a) LOWER **(string):** It converts uppercase or mixed case character strings into lowercase character strings.

Example: SQL>SELECT LOWER (JOB) FROM EMP;

Result:

```
LOWER(JOB)
----------------
clerk
salesman
salesman
manager
salesman
manager
manager
analyst
president
salesman
clerk

LOWER(JOB)
---------------
clerk
analyst
clerk
14 rows selected.
```

(b) UPPER (string): It converts lowercase or mixed case character strings into uppercase character strings.

Example: SQL>SELECT UPPER (JOB) FROM EMP;

Result:

```
UPPER(JOB)
--------------
CLERK
SALESMAN
SALESMAN
MANAGER
SALESMAN
MANAGER
MANAGER
ANALVST
PRESIDENT
SALESMAN
CLERK

UPPER(JOB)
-------------
CLERK
ANALVST
CLERK
14 rows selected.
```

(c) CONCAT (string1, string2): It is equivalent to the concatenation operator. It returns string1 concatenated with string2. It joins (combines) two string values together.

Example: SQL>SELECT CONCAT ('MONIKA', 'TATHAK') FROM DUAL;

Result: MONIKA PATHAK

(d) **LENGTH (string):** It is used to get the length of a string as a numeric value. **Example: SQL>SELECT LENGTH (Akhil) FROM DUAL;**

Result: 5

(e) **ASCII (string):** It is used to return the decimal representation of the first byte of string in the database character set.

Example: SQL> ASCII (Amit) FROM DUAL;

Result: 65

2. **Number Functions**

- It is used to perform operations on numbers.
- It accepts numeric input, only and returns numeric values.

The following are the types of numeric functions:

(a) **ABS (n):** It returns absolute value of numeric value.

Example: SQL>SELECT ABS (-29) FROM DUAL;

Result: 29

(b) **CEIL (n):** It returns the next smallest integer greater than or equal to parameter passed to n.

Example: SQL>SELECT CEIL (29.8) FROM DUAL;

Result: 30

(c) **FLOOR (n):** It returns the largest integer value less than or equal to parameter passed to n.

Example: SQL>SELECT FLOOR (29.8) FROM DUAL;

Result: 29

(d) **MOD (m,n):** It returns the remainder of m divided by n. It returns m if n is 0.

Example: SQL>SELECT MOD (16,3) FROM DUAL;

Result: 1

(e) SQRT (n): It returns the square root of n. The value of n cannot be negative.

Example: SQL>SELECT SQRT (25) ;FROM DUAL;

Result: 5

3. Date Functions

- Date functions operate on values of the Date datatype.
- It takes values of Date datatype as input and return values of Date datatype as output, except the Months_Between function, which returns a number.

The following are the types of date functions:

(a) SYSDATE: It returns the current system date and time on our local database.

Example: SQL>SELECT SYSDATE FROM DUAL;

Result:

```
SYSDATE
---------
18-JUN-15
```

(b) LAST_DAY: It returns the date of the last day of the month specified.

Example: SQL>SELECT SYSDATE LAST DAY (SYSDATE) FROM DUAL;

(c) CURRENT_DATE: It returns the current date in the Gregorian calendar for the session's time zone.

Example: SQL>SELECT SYSDATE CURRENT DAY (SYSDATE) FROM DUAL;

(d) NEXT_DAY: It returns the date of next specified day of the week after the 'date'.

Example: SQL>SELECT SYSDATE NEXT DAY (SYSDATE) FROM DUAL;

(e) ADD_MONTHS: It adds or subtracts the months to or from a date.

Example: SQL>SELECT SYSDATE, ADD_MONTHS (SYSDATE, 4) FROM DUAL;

Result:

```
SYSDATE      ADD_MONTH
----------------------
18-JUN-15    18-OCT-15
```

4. Conversion Functions

It converts the value from one form to another form.

The following are the types of conversion functions:

(a) Implicit Data Type Conversion: It occurs when the expression evaluator automatically converts the data from one data type to another.

(b) Explicit Data Type Conversion: It occurs when we explicitly converts the data from one data type to another.

5. Miscellaneous Functions

The following are the types of miscellaneous functions:

(a) GREATEST: It returns the greatest value in the list of expressions.

Example: SQL>SELECT GREATEST (2, 11, 25, 29) FROM DUAL;

Result: 29

(b) LEAST: It returns the smallest value in the list of expressions.

Example: SQL>SELECT LEAST (2, 11, 25, 29) FROM DUAL;

Result: 2

(c) USER: It returns the username of the current user logged on.

Example: SQL>SELECT USER FROM DUAL;

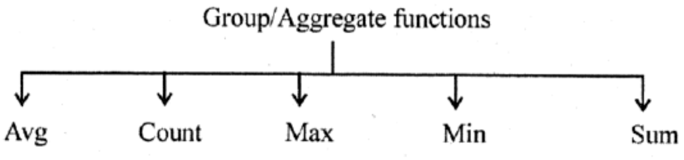
Result: SCOTT

Group/Aggregate Functions

- Aggregate functions are also known as Group functions or Summary functions.

- SQL supports the functions which can be used to select and compute the numeric, date columns and characters of the relation.
- These functions operate on multiple rows (group of rows) and return only one value for a group or table, therefore these functions are known as aggregate functions. By default, all rows are treated as one group in a table.

The types of aggregate functions are as follows:



Group/Aggregate Functions

STUD

Name	**Class**	**Roll Number**	**Marks**	**Age**
Akhil	C12	11	95	16
Monika	C12-	12	91	15
Aastha	M12	13	95	14
Rohit	E12	14	94	12
Rahul	E12	15	93	13
Ankush	C12	16	95	15
Radhika	M12	17	92	14

1. **Avg:** The Avg (average) function returns the arithmetic mean of the value of a column in a given relation. This function is applicable on numeric values.

Examples of Avg Function:->

- To find the average marks of the students from the table STUD, then the query will be:

SQL> SELECT AVG (Marks) FROM STUD;

Result: 93.51

- To find the average salary of the employees from the table EMR Then the query will be:

SQL> Select AVG (SAL) AS Average Salary FROM EMP;

Result:

```
Average Salary
------------------
2091.07143
```

2. Count: The Count function returns the number of rows in a relation (table). This function is used for numeric, character values and date. The Count function returns value only if it satisfies the condition stated in the Where Clause.

Examples of Count Function:

- To find the number of students from the table 'STUD'. Then the query will be:

SQL> Select COUNT (*) FROM STUD;

Result: 7

- To find the total number of employees from the table EMP, Then the query will be:

SQL> SELECT %COUNT (*) AS TOTAL EMPLOYEE FROM EMP;

Result:

```
TOTAL EMPLOYEE
----------------------------
        14
```

3. Max: The Max function returns the maximum of the values of a column from the given relation.

Examples of Max Function:

- To find the maximum marks from the table 'STUD'. Then the query will be:

SQL> MAX (Marks) FROM STUD;

Result: 95

- To find the maximum salary drawn by the employee from the table EMP. Then the query will be:

```
SQL> MAX (SAL) AS Maximum Salary FROM EMP;
```

Result:

```
Maximum Salary
-----------------------
5000
```

4. **Min:** The Min function returns the minimum of the values of a column from the given relation.

Examples of Min Function:

- To find the minimum marks from the table STUD. Then the query will be:

```
SQL> MIN (Marks) FROM STUD;
```

Result: 91

- To find the minimum salary drawn by the employee from the table EMP. Then the query will be:

```
SQL> MIN (SAL) AS Minimum Salary FROM EMP;
```

Result:

```
Minimum Salary
---------------------
900
```

5. **Sum:** The Sum function returns the sum of values (numeric type) of a column.

Example of Sum Function:

- To find the sum of marks from the table STUD. Then the query will be:

```
SQL> SELECT SUM (Marks) FROM STUD;
```

Result: **655**

- To find the total salary given to the employees from the table BMP. Then the query will be:

```
SQL> SELECT SUM (SAL) AS Total Salary FROM EMP;
```

Result:

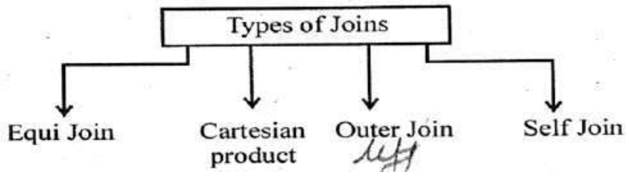
```
Total Salary
----------------
29275
```

Q14. Explain SQL joins in detail with suitable examples.

- Mostly we retrieve data from one table at a time. But what will we do if we need to retrieve data from multiple tables.
- Oracle provides the facility to retrieve the data from multiple tables with the help of joins.
- Joins are used to combine columns from different tables.
- Joins allow us to retrieve the data from multiple users in a single query.
- Joins permits us to select data from more than one table in one SQL statement (query).
- A join is used to combine rows from multiple tables.
- Joins are used to relate information in different tables.
- The connection between tables is established through the Where Clause.
- Where Clause is known as join condition.
- The rows retrieved after joining the two tables based on a condition in which one table act as a primary key and other act as a foreign key. Columns in both tables should be matched.

Syntax of Join:

```
SQL> SELECT tablel.column, table2.column, ......tableN.column
     FROM table1, table2, ........tableN.
     WHERE tablel.column1 = table2. column2;
```



Types of Join

Equi Join

- It is also known as Inner Join.
- When two tables are joined together using equality of values in one or more columns, they make an equi join.
- Equi join is used when we need to compare each record in two joined tables and comes with matching record.
- Table prefixes are utilized to prevent ambiguity.
- We use equi join (inner join) when we only want to return records where there is at least one row in both tables that match the join condition.
- Equi join uses the equal sign as the comparison operator.

Example of Equi Join:

First Table is BMP

Second Table is DEPT.

EMP

ENAME	DEPTNO	JOB	EMPNO	SAL	HIREDATE	MGR	CITY	COMM
Nidhi	20	Clerk	6258	900	9-5-83	6801	Chd	
Aastha	30	Salesman	6388	1500	1-12-89	6587	Delhi	300
Sachin	30	Salesman	6410	1350	25-1-92	6587	Pta	500
Rohit	20	Manager	6455	2875	27-12-91	6728	Nba	
Rahul	30	Salesman	6543	1350	28-5-87	6587	Nba	1400

Aditya	30	Manager	6587	2750	17-8-86	672S	Pta	
Siddharth	10	Manager	6671	2550	29-9-80	6728	Chd	
Kunal	20	Analyst	6677	3000	8-12-82	6455	Delhi	
Akhil	10	President	6728	5000	2-11-85		Delhi	
Prathiba	30	Salesman	6733	1600	4-6-85	6587	Pta	0
Manmeet	20	Clerk	6765	1050	11-1-84	6677	Ldh	
Navreet	30	Clerk	6800	950	25-3-84	6587	Pta	
Saira	20	Analyst	6801	3000	15-4-80	6455	Chd	
Amit	10	Clerk	6823	1400	25-8-85	6671	Ldh	

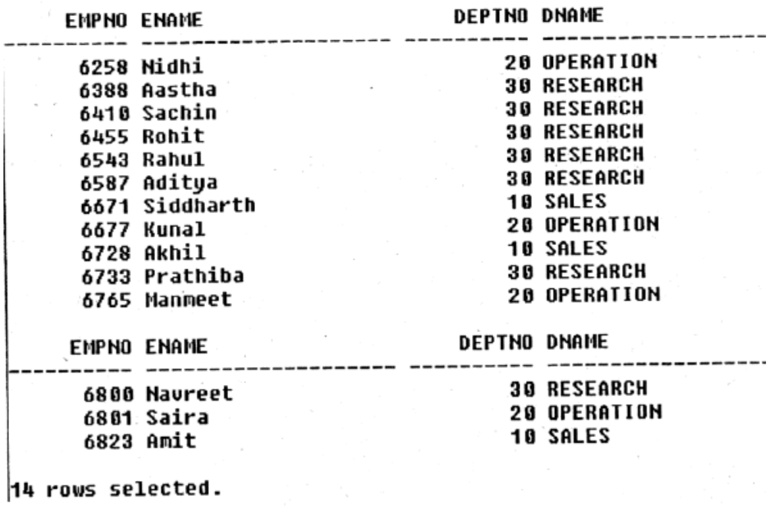
DEPT

DEPTNO	DNAME	LOG
10	Sales	London
20	Operation	Mumbai
30	Research	Paris
40	Accounting	New York

Then the query will be:

SQL> SELECT EMPNO, ENAME, EMP.DEPTNO, DNAME FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO;

Result:



```
     EMPNO ENAME                    DEPTNO DNAME
---------- -------------------- ---------- ------------------
      6258 Nidhi                        20 OPERATION
      6388 Aastha                       30 RESEARCH
      6410 Sachin                       30 RESEARCH
      6455 Rohit                        30 RESEARCH
      6543 Rahul                        30 RESEARCH
      6587 Aditya                       30 RESEARCH
      6671 Siddharth                    10 SALES
      6677 Kunal                        20 OPERATION
      6728 Akhil                        10 SALES
      6733 Prathiba                     30 RESEARCH
      6765 Manmeet                      20 OPERATION

     EMPNO ENAME                    DEPTNO DNAME
---------- -------------------- ---------- ------------------
      6800 Navreet                      30 RESEARCH
      6801 Saira                        20 OPERATION
      6823 Amit                         10 SALES

14 rows selected.
```

Explanation of Equi Join:

For Equi Join, both the table names should be mentioned.

Column name should be specified with the table name to avoid confusion.

Deptno of BMP table is joined with the deptno of DEPT table because Deptno exists in both the tables.

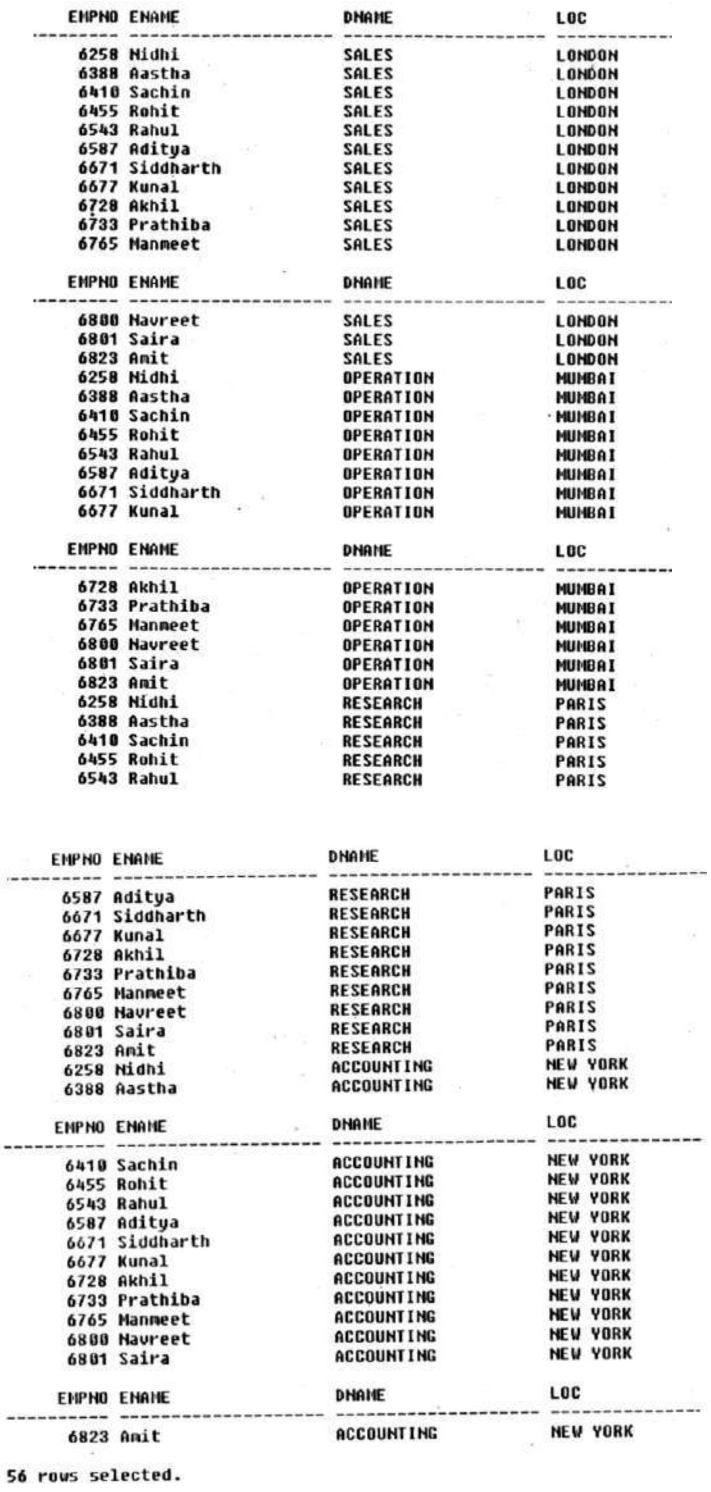
Cross Join

- It is also known as cartesian product or cartesian join.
- It returns the number of rows equal to the product of all rows in all rows in all the tables being joined.
- It provides results in mXn rows.
- It is used when we want to join every row of a table to every row of itself.

Example of Cross Join:

SQL>SELECT EMPNO, ENAME, DNAME, LOC FROM EMP, DEPT;

Result:



```
    EMPNO ENAME                DNAME                LOC
--------- -------------------- -------------------- -----------
     6258 Nidhi                SALES                LONDON
     6388 Aastha               SALES                LONDON
     6410 Sachin               SALES                LONDON
     6455 Rohit                SALES                LONDON
     6543 Rahul                SALES                LONDON
     6587 Aditya               SALES                LONDON
     6671 Siddharth            SALES                LONDON
     6677 Kunal                SALES                LONDON
     6728 Akhil                SALES                LONDON
     6733 Prathiba             SALES                LONDON
     6765 Manmeet              SALES                LONDON

    EMPNO ENAME                DNAME                LOC
--------- -------------------- -------------------- -----------
     6800 Navreet              SALES                LONDON
     6801 Saira                SALES                LONDON
     6823 Amit                 SALES                LONDON
     6258 Nidhi                OPERATION            MUMBAI
     6388 Aastha               OPERATION            MUMBAI
     6410 Sachin               OPERATION            MUMBAI
     6455 Rohit                OPERATION            MUMBAI
     6543 Rahul                OPERATION            MUMBAI
     6587 Aditya               OPERATION            MUMBAI
     6671 Siddharth            OPERATION            MUMBAI
     6677 Kunal                OPERATION            MUMBAI

    EMPNO ENAME                DNAME                LOC
--------- -------------------- -------------------- -----------
     6728 Akhil                OPERATION            MUMBAI
     6733 Prathiba             OPERATION            MUMBAI
     6765 Manmeet              OPERATION            MUMBAI
     6800 Navreet              OPERATION            MUMBAI
     6801 Saira                OPERATION            MUMBAI
     6823 Amit                 OPERATION            MUMBAI
     6258 Nidhi                RESEARCH             PARIS
     6388 Aastha               RESEARCH             PARIS
     6410 Sachin               RESEARCH             PARIS
     6455 Rohit                RESEARCH             PARIS
     6543 Rahul                RESEARCH             PARIS


    EMPNO ENAME                DNAME                LOC
---------- -------------------- -------------------- --------------
     6587 Aditya               RESEARCH             PARIS
     6671 Siddharth            RESEARCH             PARIS
     6677 Kunal                RESEARCH             PARIS
     6728 Akhil                RESEARCH             PARIS
     6733 Prathiba             RESEARCH             PARIS
     6765 Manmeet              RESEARCH             PARIS
     6800 Navreet              RESEARCH             PARIS
     6801 Saira                RESEARCH             PARIS
     6823 Amit                 RESEARCH             PARIS
     6258 Nidhi                ACCOUNTING           NEW YORK
     6388 Aastha               ACCOUNTING           NEW YORK

    EMPNO ENAME                DNAME                LOC
---------- -------------------- -------------------- --------------
     6410 Sachin               ACCOUNTING           NEW YORK
     6455 Rohit                ACCOUNTING           NEW YORK
     6543 Rahul                ACCOUNTING           NEW YORK
     6587 Aditya               ACCOUNTING           NEW YORK
     6671 Siddharth            ACCOUNTING           NEW YORK
     6677 Kunal                ACCOUNTING           NEW YORK
     6728 Akhil                ACCOUNTING           NEW YORK
     6733 Prathiba             ACCOUNTING           NEW YORK
     6765 Manmeet              ACCOUNTING           NEW YORK
     6800 Navreet              ACCOUNTING           NEW YORK
     6801 Saira                ACCOUNTING           NEW YORK

    EMPNO ENAME                DNAME                LOC
---------- -------------------- -------------------- --------------
     6823 Amit                 ACCOUNTING           NEW YORK

56 rows selected.
```

Explanation:

Table BMP has 14 rows.

Table DEPT has 4 rows.

Then, total number of rows = mXn

=14X4

=> Total number of rows =56 rows

Outer Join

- Outer join has symbol (+).
- It is used if there is any value in one table that do not have corresponding value in other table. Such rows are forcefully selected by it.
- It is used on one side of the join condition only and the corresponding columns for that row will have NULL value.

Example of Outer Join:

SQL>SELECT EMPNO, ENAME, E-MP.DEPTNO, DNAME, LOC FROM EMP,DEPT

WHERE EMP.DEPTNQ (+) = DEPT.DEPTNO;

Result:

EMPNO	ENAME	DEPTNO	DNAME	LOC
6258	Nidhi	20	Operation	Mumbai
6388	Aastha	30	Research	Paris
6410	Sachin	30	Research	Paris
6455	Rohit	30	Research	Paris
6543	Rahul	30	Research	Paris
6587	Aditya	30	Research	Paris
6671	Siddharth	10	Sale	Paris
6677	Kunal	20	Research	Paris

6728	Akhil	10	Sale	London
6733	Prathiba	30	Research	Mumbai
6765	Manmeet	20	Operation	London
6800	Navreet	30	Research	Paris
6801	Saira	20	Operation	Mumbai
6823	Amit	10	Sale	London

Self Join

- Self join is used when a table is joined/compared to itself.
- A table is joined to itself means each row of the table is combined with itself and with every row of the table.
- If we want to use self join, then we need to open the two copies of same table by using table aliases
- Table name aliases are defined in the From Clause of the query.
- Table alias is used to avoid confusion among two same tables.

Example of Self Join:

```
        SQL>SELECT WORKER.ENAME AS ENAME, MANAGER.ENAME AS
MANAGER
        FROM EMP WORKER, EMP MANGER
        WHERE WORKER.MGR = MANAGER.EMPNO;
```