Customer : ID, Customer Name, Customer Age, Customer Address, Customer Salary
Order: ID, Customer ID, Date, Amount

1) most frequently used and important is Inner joins ->Equijoins

```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.common_field = table2.common_field;
```

2) **LEFT JOIN** returns all rows from the left table, even if there are no matches in the right table.

```
SELECT table1.column1, table2.column2...
FROM table1
LEFT JOIN table2
ON table1.common_field = table2.common_field;
```

3) **RIGHT JOIN** returns all rows from the right table, even if there are no matches in the left table.

```
SELECT table1.column1, table2.column2...
FROM table1
RIGHT JOIN table2
ON table1.common_field = table2.common_field;
```

4) **FULL JOIN** combines the results of both left and right outer joins.

```
SELECT table1.column1, table2.column2...
FROM table1
FULL JOIN table2
ON table1.common_field = table2.common_field;
```

If your Database does not support FULL JOIN like MySQL does not support FULL JOIN, then you can use **UNION ALL** clause to combine two JOINS as follows:

```
SQL> SELECT  ID, NAME, AMOUNT, DATE
     FROM CUSTOMERS
     LEFT JOIN ORDERS
```

```
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
UNION ALL
     SELECT  ID, NAME, AMOUNT, DATE
     FROM CUSTOMERS
     RIGHT JOIN ORDERS
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
```

5)**SELF JOIN** is used to join a table to itself as if the table were two tables

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;
```

6)**CARTESIAN JOIN** or **CROSS JOIN** returns the Cartesian product of the sets of records from the two or more joined tables.

```
SELECT table1.column1, table2.column2...
FROM  table1, table2 [, table3 ]
```

7) Left Excluding JOIN

```
SELECT <select_list>
FROM Table_A A
LEFT JOIN Table_B B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

8) Right Excluding JOIN

```
SELECT <select_list>
FROM Table_A A
RIGHT JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

9) Outer Excluding JOIN

```
SELECT <select_list>
FROM Table_A A
FULL OUTER JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL OR B.Key IS NULL
```

10) ORDER BY

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

11) **UNION** clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

To use UNION, each SELECT must have the same number of columns selected, the same number of column expressions, the same data type, and have them in the same order, but they do not have to be the same length.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

UNION

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

12) The **UNION ALL** operator is used to combine the results of two SELECT statements including duplicate rows.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

UNION ALL

SELECT column1 [, column2 ]
```

```
FROM table1 [, table2 ]
[WHERE condition]
```

13) INDEXES an index is a pointer to data in a table. For faster searches. When index is created, it is assigned a ROWID for each row before it sorts out the data.

```
CREATE INDEX index_name
on table_name (column1, column2);
```

14)
```
ALTER TABLE table_name ADD column_name datatype;

ALTER TABLE table_name DROP COLUMN column_name;

ALTER TABLE table_name MODIFY COLUMN column_name datatype;

ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
```

```
ALTER TABLE table_name
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

```
ALTER TABLE ORDERS
   ADD FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (ID);
```

15) Constraints This ensures the accuracy and reliability of the data in the database.

```
       ID   INT              NOT NULL,
       NAME VARCHAR (20)     NOT NULL,
       AGE  INT              NOT NULL,
       ADDRESS  CHAR (25) ,
       SALARY   DECIMAL (18, 2),
       PRIMARY KEY (ID)
);
```

```
ALTER TABLE CUSTOMERS
   MODIFY SALARY  DECIMAL (18, 2) NOT NULL;
```

```
ALTER TABLE CUSTOMERS
   ALTER COLUMN SALARY DROP DEFAULT;
```

```
ALTER TABLE CUSTOMER ADD PRIMARY KEY (ID);
```

16) Views, which are kind of virtual tables, allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.

- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.

- Summarize data from various tables which can be used to generate reports.

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS
WHERE age IS NOT NULL
WITH CHECK OPTION;
```

17)
```
SAVEPOINT SAVEPOINT_NAME;
ROLLBACK TO SAVEPOINT_NAME;
```

18) Nested Queries

```
SELECT *
     FROM CUSTOMERS
     WHERE ID IN (SELECT ID
                  FROM CUSTOMERS
                  WHERE SALARY > 4500) ;
```

Views
Subset of the database
    Contains **virtual data** derived from the database files but is not explicitly stored

Phases for designing a database:
    Requirements specification and analysis
    Conceptual design
    Logical design
    Physical design

Program-data independence
    Structure of data files is stored in DBMS catalog separately from access programs
Program-operation independence
    Operations specified in two parts:
        Interface includes operation name and data types of its arguments
        Implementation can be changed without affecting the interface

Database designers are responsible for:
    Identifying the data to be stored
    Choosing appropriate structures to represent and store this data

    Types
        Casual end users
        Naive or **parametric end users**
        Sophisticated end users
        Standalone users
Extending database capabilities for new applications
    Extensions to better support specialized requirements for applications
    Enterprise resource planning (ERP)
    Customer relationship management (CRM)
Databases versus information retrieval
    Information retrieval (IR)
        Deals with books, manuscripts, and various forms of library-based articles

- Database
  - Collection of related data (recorded facts)
- DBMS
  - Generalized software package for implementing and maintaining a computerized database
- Several categories of database users
- Database applications have evolved
  - Current trends: IR, Web