- `SELECT DISTINCT Country FROM Customers;`
- `INSERT INTO Customers (CustomerName, City, Country) VALUES ('Cardinal', 'Stavanger', 'Norway');`
- `UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;`
- `DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';`
- `SELECT TOP 3 * FROM Customers;`
- `SELECT MIN(Price) FROM Products;`
- `SELECT COUNT(ProductID) FROM Products`
- `SELECT * FROM Customers WHERE CustomerName LIKE '_r%';`
- `SELECT * FROM Customers WHERE Country IN ('Germany', 'France', 'UK');`
- `SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;`
- `SELECT ProductID, ProductName, CategoryName FROM Products INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;`
- The `UNION` operator selects only distinct values by default. To allow duplicate values, use `UNION ALL`:
  `SELECT column_name(s) FROM table1 UNION ALL SELECT column_name(s) FROM table2;`
- The `GROUP BY` statement is often used with aggregate functions (`COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()`) to group the result-set by one or more columns
- The `HAVING` clause was added to SQL because the `WHERE` keyword cannot be used with aggregate functions.
  `SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country HAVING COUNT(CustomerID) > 5;`
- The `EXISTS` operator is used to test for the existence of any record in a subquery.The `EXISTS` operator returns TRUE if the subquery returns one or more records.
  `SELECT column_name(s) FROM table_name WHERE EXISTS (SELECT column_name FROM table_name WHERE condition);`
- ```
  CREATE TABLE Persons (
      PersonID int,
      LastName varchar(255),
      FirstName varchar(255),
      Address varchar(255),
      City varchar(255)
  );
  ```
- `DROP DATABASE databasename;`
- `ALTER TABLE Customers ADD/DROP COLUMN Email;`
- `ALTER TABLE table_name RENAME COLUMN old_name to new_name;`

The following constraints are commonly used in SQL:

- `NOT NULL` - Ensures that a column cannot have a NULL value
  ```

- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly
- ```
  CREATE TABLE Orders (
      OrderID int NOT NULL,
      OrderNumber int NOT NULL,
      PersonID int,
      PRIMARY KEY (OrderID),
      FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
  );
  ```
- ```
  CREATE INDEX idx_pname ON Persons (LastName, FirstName);
  ```
- Return all customers with a City starting with "L", followed by any 3 characters, ending with "on":
  ```
  SELECT * FROM Customers
  WHERE City LIKE 'L___on';
  ```
- Return all customers that ends with the pattern 'es':

  ```
  SELECT * FROM Customers
  WHERE CustomerName LIKE '%es';
  ```

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

  There are two wildcards often used in conjunction with the LIKE operator:

1. The percent sign % represents zero, one, or multiple characters
2. The underscore sign _ represents one, single character

Normalization forms, in the context of database design, are a set of rules to organize the data structure efficiently, minimize redundancy, and ensure data integrity.

1. First Normal Form (1NF): Ensures that each column in a table contains atomic values, meaning no multi-valued attributes are allowed. Each cell should hold a single value, and there should be no repeating groups or arrays.

2. Second Normal Form (2NF): Requires that a table is in 1NF and that all non-key attributes are fully functional dependent on the primary key. It

eliminates partial dependencies, meaning attributes should depend on the entire primary key, not just part of it.

3. Third Normal Form (3NF): Building upon 2NF, 3NF ensures that there are no transitive dependencies. In other words, no non-primary key attribute should depend on another non-primary key attribute.

4. Boyce-Codd Normal Form (BCNF): A more stringent form of 3NF, BCNF further eliminates anomalies by ensuring that every determinant in a table is a candidate key. It's a higher level of normalization and often requires more careful consideration in database design.

5. Fourth Normal Form (4NF): Focuses on multi-valued dependencies, ensuring that there are no independent multi-valued facts within the same table. It deals with situations where there are multiple independent multi-valued facts about the same entity.

6. Fifth Normal Form (5NF): Addresses cases of joining tables that have multiple overlapping candidate keys and ensures that every join dependency in the table is implied by the candidate keys.

---

1. ****Primary Key**:** A primary key uniquely identifies each record in a table. It ensures that each row in the table is unique and not null. There can be only one primary key in a table.

2. ****Foreign Key**:** A foreign key is a column or a set of columns in one table that refers to the primary key or a unique key in another table. It establishes a link between the two tables and enforces referential integrity.

3. ****Unique Key**:** A unique key constraint ensures that the values in a column (or a set of columns) are unique across the table. Unlike the primary key, a unique key can contain null values (but only one null value).

4. ****Composite Key**:** A composite key consists of multiple columns that together uniquely identify each row in a table. It's useful when no single column can uniquely identify a row, but the combination of columns does.

5. ****Index Key**:** An index key is a data structure that improves the speed of data retrieval operations on a table. It can be created on one or more columns to quickly locate rows based on the indexed columns.

6. ****Candidate Key**:** A candidate key is a column or a set of columns that can uniquely identify each row in a table. It's essentially a candidate for being chosen as the primary key.

7. ****Super Key**:** A super key is a set of one or more columns that uniquely identifies each row in a table. It can include more columns than necessary for uniquely identifying rows.