SQL Keywords

1. ADD KEYWORD

The ADD command is used to add a column in an existing table.

The following SQL adds an "Email" column to the "Customers" table:

Example

```
ALTER TABLE Customers
ADD Email varchar(255);
```

2. ALTER Keyword

ALTER TABLE

- The ALTER TABLE command adds, deletes, or modifies columns in a table.

- The ALTER TABLE command also adds and deletes various constraints in a table.

- The following SQL adds an "Email" column to the "Customers" table:

Example

ALTER TABLE Customers

ADD Email varchar(255);

3.ALL KEYWORD

The ALL command returns true if all of the subquery values meet the condition.

The following SQL statement returns TRUE and lists the productnames if ALL the records in the OrderDetails table has quantity = 10:

Example

```
SELECT ProductName

FROM Products

WHERE ProductID = ALL (SELECT

ProductID FROM OrderDetails WHERE

Quantity = 10);
```

4.AND KEYWORD

The AND command is used with WHERE to only include rows where both conditions is true.

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

Example

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

5.ANY KEYWORD

The ANY command returns true if any of the subquery values meet the condition.

The following SQL statement returns TRUE and lists the productnames if it finds ANY records in the OrderDetails table where quantity = 10:

Example

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);
```

6.BETWEEN KEYWORD

The BETWEEN command is used to select values within a given range. The values can be numbers, text, or dates.

The BETWEEN command is inclusive: begin and end values are included.

The following SQL statement selects all products with a price BETWEEN 10 and 20:

Example

```
SELECT * FROM Products

WHERE Price BETWEEN 10 AND 20;
```

7. CASE KEYWORD

The CASE command is used is to create different output based on conditions.

The following SQL goes through several conditions and returns a value when the specified condition is met:

Example

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN "The quantity is greater than 30"
    WHEN Quantity = 30 THEN "The quantity is 30"
    ELSE "The quantity is under 30"
END
FROM OrderDetails;
```

8.WHERE KEYWORD

SELECT

The WHERE command filters a result set to include only records that fulfill a specified condition.

The following SQL statement selects all the customers from "Mexico" in the "Customers" table:

Example

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

9.ADD CONSTRAINT

The ADD CONSTRAINT command is used to create a constraint after a table is already created.

The following SQL adds a constraint named "PK_Person" that is a PRIMARY KEY constraint on multiple columns (ID and LastName):

Example

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person
PRIMARY KEY (ID,LastName);
```

10.AS KEYWORD

The AS command is used to rename a column or table with an alias.

An alias only exists for the duration of the query.

Alias for Columns

The following SQL statement creates two aliases, one for the CustomerID column and one for the CustomerName column:

Example

```
SELECT CustomerID AS ID,
CustomerName AS Customer
FROM Customers;
```

11.ASC KEYWORD

The ASC command is used to sort the data returned in ascending order.

The following SQL statement selects all the columns from the "Customers" table, sorted by the "CustomerName" column:

Example

```
SELECT * FROM Customers
ORDER BY CustomerName ASC;
```

12.CREATE DATABASE

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

The following SQL creates a database called "testDB":

Example

```
CREATE DATABASE testDB;
```

13.CREATE OR REPLACE VIEW KEYWORD

The CREATE OR REPLACE VIEW command updates a view.

The following SQL adds the "City" column to the "Brazil Customers" view:

Example

```
CREATE OR REPLACE VIEW [Brazil Customers] AS
```

```
SELECT CustomerName, ContactName,
City
FROM Customers
WHERE Country = "Brazil";
```

14.DROP COLUMN

The DROP COLUMN command is used to delete a column in an existing table.

The following SQL deletes the "ContactName" column from the "Customers" table:

Example

```
ALTER TABLE Customers
DROP COLUMN ContactName;
```

15.FULL OUTER JOIN KEYWORD

The FULL OUTER JOIN command returns all rows when there is a match in either left table or right table.

The following SQL statement selects all customers, and all orders:

```
SELECT Customers.CustomerName,
Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON
Customers.CustomerID=Orders.Customer
ID
```

```
ORDER BY Customers.CustomerName;
```

16.GROUP BY KEYWORD

The GROUP BY command is used to group the result set (used with aggregate functions: COUNT, MAX, MIN, SUM, AVG).

The following SQL lists the number of customers in each country:

Example

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

17.IS NULL

The IS NULL command is used to test for empty values (NULL values).

The following SQL lists all customers with a NULL value in the "Address" field:

Example

```
SELECT CustomerName, ContactName,
Address
FROM Customers
WHERE Address IS NULL;
```

18.INNER JOIN

The INNER JOIN command returns rows that have matching values in both tables.

The following SQL selects all orders with customer information:

Example

```
SELECT Orders.OrderID,
Customers.CustomerName
FROM Orders
INNER JOIN Customers ON
Orders.CustomerID =
Customers.CustomerID;
```

19.LEFT JOIN

The LEFT JOIN command returns all rows from the left table, and the matching rows

from the right table. The result is NULL from the right side, if there is no match.

The following SQL will select all customers, and any orders they might have:

Example

```
SELECT Customers.CustomerName,
Orders.OrderID
FROM Customers
```

```
LEFT JOIN Orders ON
Customers.CustomerID =
Orders.CustomerID
ORDER BY Customers.CustomerName;
```

20.ORDER BY

The ORDER BY command is used to sort the result set in ascending or descending order.

The ORDER BY command sorts the result set in ascending order by default. To sort

the records in descending order, use the DESC keyword.

The following SQL statement selects all the columns from the "Customers" table, sorted by the "CustomerName" column:

Example

```
SELECT * FROM Customers
ORDER BY CustomerName;
```

21.RIGHT JOIN

The RIGHT JOIN command returns all rows from the right table, and the matching records from the left table. The result is NULL from the left side, when there is no match.

The following SQL will return all employees, and any orders they might have placed:

Example

```
SELECT Orders.OrderID,
Employees.LastName,
Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON
Orders.EmployeeID =
Employees.EmployeeID
ORDER BY Orders.OrderID;
```