|  |  |  |
| --- | --- | --- |
| S.No. | Objective | Query |
| 1. | Select all records from the Customers table: | Select\* from Customers |
| 2. | select data from a database. | SELECT CustomerName, City FROM Customers; |
| 3. | return only distinct (different) values. | SELECT DISTINCT Country FROM Customers; |
| 4. | return the number of different countries | SELECT COUNT(DISTINCT Country) FROM Customers; ***or***  SELECT Count(\*) AS DistinctCountries FROM (SELECT DISTINCT Country FROM Customers); |
| 5. | records that fulfill a specified condition | SELECT \* FROM Customers WHERE Country='Mexico'; |
| 6. | Select all customers with a CustomerID greater than 80: | SELECT \* FROM Customers WHERE CustomerID > 80;  Other Operators: =,<,>,>=,<=,BETWEEN, LIKE, IN |
|  | OPERATORS: |  |
| 7. | Select all Products where price is between 50 and 60: | SELECT \* FROM Products WHERE Price BETWEEN 50 AND 60; |
| 8. | Like |  |
| 9. | In |  |
| 10. | AND(Both conditions must follow) | SELECT \* FROM Customers WHERE Country= 'Spain' AND CustomerName LIKE 'G%'; |
| 11. | OR(One must follow) | SELECT \* FROM Customers WHERE Country= 'Spain' OR CustomerName LIKE 'G%'; |
| 12. | Combine AND and OR | SELECT \* FROM Customers WHERE Country = 'Spain' AND (CustomerName LIKE 'G%' OR CustomerName LIKE 'R%'); |
| 13. | NOT | SELECT \* FROM Customers WHERE NOT Country = 'Spain' |
| 14. | Select customers that does not start with the letter 'A': | SELECT \* FROM Customers WHERE CustomerName NOT LIKE 'A%'; |
| 15. | Select customers with a customerID not between 10 and 60 | SELECT \* FROM Customers WHERE CustomerID NOT BETWEEN 10 AND 60; |
| 16. | Select customers that are not from Paris or London: | SELECT \* FROM Customers WHERE City NOT IN ('Paris', 'London'); |
| 17. | Select customers with a CustomerId not greater than 50: | SELECT \* FROM Customers WHERE NOT CustomerID > 50; or use !> instead of Not < |
|  |  |  |
| 18. | Insert Into | INSERT INTO table\_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...); |
|  | Inserts a new record in the "Customers | INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'); |
| 19. | Lists all customers with a NULL value in the "Address" field: (Same Not NULL) | SELECT CustomerName, ContactName, Address FROM Customers WHERE Address IS (NOT (=if required)) NULL; |
| 20. | updates the first customer  (Warning: must use WHERE else all records will be updated) | UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1; |
| 21. | Delete existing records in a table. | DELETE FROM table\_name WHERE condition; |
| 22. | Delete the customer "Alfreds Futterkiste" from the "Customers" table | DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste'; |
| 23. | Delete All Records | DELETE FROM Customers; |
| 24. | Delete a Table | DROP TABLE Customers; |
| 25. | Select only the first 3 records of the Customers table: | SELECT TOP 3 \* FROM Customers; ***or*** SELECT \* FROM Customers LIMIT 3; |
| 26. | selects the first 50% of the records from the "Customers" | SELECT TOP 50 PERCENT \* FROM Customers; ***or (for Oracle)*** SELECT \* FROM Customers FETCH FIRST 50 PERCENT ROWS ONLY; |
| 27. | Sort the result reverse alphabetically by CustomerName, and return the first 3 records: | SELECT TOP 3 \* FROM Customers ORDER BY CustomerName DESC; |
| 28. | Find the lowest price: | SELECT MIN(Price) FROM Products; |
| 29. | Find the highest price: | SELECT MAX(Price) FROM Products; |
| 30. | To give the column a new name, use the AS keyword: | SELECT MIN(Price) AS SmallestPrice FROM Products; |
| 31. | Find the total number of products in the Products table: | SELECT COUNT(\*) FROM Products; |
| 32. | Find the number of products where Price is higher than 20: | SELECT COUNT(ProductID) FROM Products WHERE Price > 20; |
| 33. | How many different prices are there in the Products table: | SELECT COUNT(DISTINCT Price) FROM Products; |
| 34. | Give the counted column a name by using the AS keyword(Use Alias) | SELECT COUNT(\*) AS [number of records] FROM Products; |
| 35. | Return the sum of all Quantity fields in the OrderDetails table: | SELECT SUM(Quantity) FROM OrderDetails; |
| 36. | Give the summarized column a name by using the AS keyword. | SELECT SUM(Quantity) AS total FROM OrderDetails; |
| 38. | Use an expression inside the SUM() function: | SELECT SUM(Quantity \* 10) FROM OrderDetails; |
| 39. | Find the average price of all products: | SELECT AVG(Price) FROM Products; |
| 40. | Give the AVG column a name by using the AS keyword. | SELECT AVG(Price) AS [average price] FROM Products; |
| 41. | Return all products with a higher price than the average price: | SELECT \* FROM Products WHERE price > (SELECT AVG(price) FROM Products); |
| 42. | Select all customers that starts with the letter "a": | SELECT \* FROM Customers WHERE CustomerName LIKE 'a%'; |
| 43. | Return all customers that ends with 'a': | SELECT \* FROM Customers WHERE CustomerName LIKE '%a'; |
| 44. | Return all customers that starts with 'a' or starts with 'b' | SELECT \* FROM Customers WHERE CustomerName LIKE 'a%' OR CustomerName LIKE 'b%'; |
| 45. | Return all customers that starts with "b" and ends with "s": | SELECT \* FROM Customers WHERE CustomerName LIKE 'b%s'; |
| 46. | Return all customers that contains the phrase 'or' | SELECT \* FROM Customers WHERE CustomerName LIKE '%or%'; |
| 47. | Return all customers that starts with "a" and are at least 3 characters in length: | SELECT \* FROM Customers  WHERE CustomerName LIKE 'a\_\_%'; |
| 48. | Return all customers that have "r" in the second position: | SELECT \* FROM Customers WHERE CustomerName LIKE '\_r%'; |
| 49. | Return all customers from Spain: | SELECT \* FROM Customers WHERE Country LIKE 'Spain'; |
| 50. | Return all customers with a City starting with any character, followed by "ondon": | SELECT \* FROM Customers WHERE City LIKE '\_ondon'; |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |