Note :

- SFWGHO

TABLE queries

1. selects the "CustomerName" and "City" columns from the "Customers" table.
2. selects all the columns from the "Customers" table.
3. selects only the DISTINCT values from the "Country" column in the "Customers" table.
4. lists the number of different distinct customer countries.
5. selects all the customers from the country "Mexico", in the "Customers" table.
6. selects all fields from "Customers" where country is "Germany" AND city is "Berlin".
7. selects all fields from "Customers" where city is "Berlin" OR "München".
8. selects all fields from "Customers" where country is NOT "Germany".
9. selects all fields from "Customers" where country is "Germany" AND city must be "Berlin" OR "München" (use parenthesis to form complex expressions).
10. selects all fields from "Customers" where country is NOT "Germany" and NOT "USA".
11. selects all customers from the "Customers" table, sorted by the "Country" column.
12. selects all customers from the "Customers" table, sorted DESCENDING by the "Country" column.
13. selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. (This means that it orders by Country, but if some rows have the same Country, then it orders them by CustomerName).
14. selects all customers from the "Customers" table, sorted ascending by the "Country" and descending by the "CustomerName" column.
15. inserts a new record in the "Customers" table.
16. insert a new record, but only insert data in the "CustomerName", "City", and "Country" columns.
17. lists all customers with a NULL value in the "Address" field.
18. lists all customers with a value (Not null) in the "Address" field.
19. updates the first customer with a new contact person and a new city.
20. update the ContactName to "Juan" for all records where country is "Mexico",
21. update ContactName to "Anjan" for all the records in the table.
22. deletes the customer "Alfreds Futterkiste" from the "Customers" table.
23. deletes all rows in the "Customers" table, without deleting the table.

24. selects the first three records from the "Customers" table.
25. selects the first 50% of the records from the "Customers" table.
26. selects the first three records from the "Customers" table, where the country is "Germany".
27. finds the price of the cheapest product.
28. finds the price of the most expensive product.
29. finds the number of products.
30. finds the average price of all products.
31. finds the sum of the "Quantity" fields in the "OrderDetails" table.
32. selects all customers with a CustomerName starting with "a".
33. selects all customers with a CustomerName ending with "a".
34. selects all customers with a CustomerName that have "or" in any position.
35. selects all customers with a CustomerName that have "r" in second position.
36. selects all customers with a CustomerName that start with "a" and are at least 2 characters in length.
37. selects all customers with a CustomerName that start with "a" and are at least 3 characters in length.
38. selects all customers with a CustomerName that start with "a" and ends with "o".
39. selects all customers with a CustomerName that does NOT start with "a".
40. selects all customers with a City starting with "ber".
41. selects all customers with a City containing the pattern "es".
42. selects all customers with a City starting with any character, followed by "ondon".
43. selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on".
44. selects all customers with a CustomerName starting with "a" and “b”.
45. selects all customers with a City starting with "b", "s", or "p". hint : using brackets.
46. selects all customers with a City starting with "a", "b", or "c". hint : using hyphen.
47. select all customers with a City NOT starting with "b", "s", or "p". hint : using ! or else Not Like.
48. selects all customers that are located in "Germany", "France" or "UK".
49. selects all customers that are NOT located in "Germany", "France" or "UK".
50. selects all customers that are from the same countries as the suppliers.
51. selects all products with a price between 10 and 20.
52. selects all products with a price not between 10 and 20.

53. selects all products with a price between 10 and 20. In addition; do not show products with a CategoryID of 1,2, or 3.
54. selects all products with a ProductName between Carnarvon Tigers and Mozzarella di Giovanni.
55. selects all products with a ProductName between Carnarvon Tigers and Chef Anton's Cajun Seasoning.
56. selects all products with a ProductName not between Carnarvon Tigers and Mozzarella di Giovanni.
57. selects all products with a ProductName not between Carnarvon Tigers and Mozzarella di Giovanni.
58. creates two aliases, one for the CustomerID column and one for the CustomerName column.
59. creates an alias named "Address" that combine four columns (Address, PostalCode, City and Country.
60. selects all the orders from the customer with CustomerID=4 (Around the Horn). We use the "Customers" and "Orders" tables, and give them the table aliases of "c" and "o" respectively.
61. selects all the orders from the customer with CustomerID=4 (Around the Horn). We use the "Customers" and "Orders" tables, and without alias.
62. selects all orders with customer information.
63. selects all orders with customer and shipper information.
64. select all customers, and any orders they might have with ascending order.
65. return all employees, and any orders they might have placed.
66. selects all customers, and all orders.
67. matches customers that are from the same city.
68. returns the cities (only distinct values) from both the "Customers" and the "Suppliers" table.
69. returns the cities (duplicate values also) from both the "Customers" and the "Suppliers" table.
70. returns the German cities (only distinct values) from both the "Customers" and the "Suppliers" table.
71. returns the German cities (duplicate values also) from both the "Customers" and the "Suppliers" table.
72. lists all customers and suppliers (by alias type customer and supplier).
73. lists the number of customers in each country.
74. lists the number of orders sent by each shipper.
75. lists the number of customers in each country. Only include countries with more than 5 customers.

76. lists the number of customers in each country, sorted high to low (Only include countries with more than 5 customers.
77. lists the employees that have registered more than 10 orders.
78. lists if the employees "Davolio" or "Fuller" have registered more than 25 orders.
79. returns TRUE and lists the suppliers with a product price less than 20.
80. returns TRUE and lists the suppliers with a product price equal to 22.
81. lists the ProductName if it finds ANY records in the OrderDetails table has Quantity equal to 10.
82. lists the ProductName if it finds ANY records in the OrderDetails table has Quantity larger than 99.
83. lists the ProductName if it finds ANY records in the OrderDetails table has Quantity larger than 1000.
84. lists the ProductName if ALL the records in the OrderDetails table has Quantity equal to 10. This will of course return FALSE because the Quantity column has many different values (not only the value of 10).
85. creates a backup copy of Customers.
86. using the IN clause to copy the table into a new table in another database.
87. copy only a few columns into a new table.
88. copy only the German customers into a new table.
89. copy data from more than one table into a new table.
90. create a new, empty table using the schema of another. Just add a WHERE clause that causes the query to return no data.
91. copies "Suppliers" into "Customers".
92. copies only the German suppliers into "Customers".
93. goes through conditions and returns a value when the first condition is met.
94. order the customers by City. However, if City is NULL, then order by Country.
95. return an alternative value when an expression is NULL.
96. creates a stored procedure named "SelectAllCustomers" that selects all records from the "Customers" table.
97. creates a stored procedure that selects Customers from a particular City from the "Customers" table.
98. creates a stored procedure that selects Customers from a particular City with a particular PostalCode from the "Customers" table.

DATABASE queries

99. Create database.
100. Drop database.
101. creates a full back up of the existing database "testDB" to the D disk.
102. creates a differential back up of the database "testDB".
103. creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City.
104. creates a new table called "TestTables" (which is a copy of the "Customers" table).
105. drops the existing table "Shippers".
106. delete the data inside a table, but not the table itself. (hint: truncate)
107. adds an "Email" column to the "Customers" table.
108. deletes the "Email" column from the "Customers" table.
109. Rename the old column name to new name.
110. change the data type of a column in a table.
111. ensure that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created.
112. create a NOT NULL constraint on the "Age" column where the "Persons" table is already created.
113. creates a UNIQUE constraint on the "ID" column when the "Persons" table is created.
114. name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns.
115. create a UNIQUE constraint on the "ID" column when the table is already created.
116. name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns.
117. drop a UNIQUE constraint.
118. creates a PRIMARY KEY on the "ID" column when the "Persons" table is created.
119. allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns.
120. create a PRIMARY KEY constraint on the "ID" column when the table is already created.
121. allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns.
122. drop a PRIMARY KEY constraint.
123. creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created.
124. allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns.

125. create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created.
126. allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns.
127. drop a FOREIGN KEY constraint.
128. creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older.
129. allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns.
130. create a CHECK constraint on the "Age" column when the table is already created.
131. allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns.
132. drop a CHECK constraint.
133. sets a DEFAULT value for the "City" column when the "Persons" table is created.
134. insert system values, by using functions like GETDATE() by default to date column.
135. create a DEFAULT constraint on the "City" column when the table is already created.
136. drop a DEFAULT constraint.
137. Creates an index on a table. Duplicate values are allowed.
138. Creates a unique index on a table. Duplicate values are not allowed.
139. creates an index named "idx_lastname" on the "LastName" column in the "Persons" table.
140. create an index on a combination of columns, you can list the column names within the parentheses, separated by commas.
141. DROP INDEX statement.
142. defines the "Personid" column to be an auto-increment primary key field in the "Persons" table.
143. let the AUTO_INCREMENT sequence start with another value, use the following SQL statement.
144. creates a view that shows all customers from Brazil.
145. creates a view that selects every product in the "Products" table with a price higher than the average price.
146. adds the "City" column to the "Brazil Customers" view. (Updating view)
147. DROP VIEW statement.