**PREDICT OUTPUT**

|  |  |  |
| --- | --- | --- |
| 1 | SELECT DISTINCT Country FROM Customers; |  |
| 2 | SELECT COUNT(DISTINCT Country) FROM Customers; |  |
| 3 | SELECT \* FROM Customers WHERE Country = 'Mexico'; |  |
| 4 | SELECT \* FROM Customers WHERE Country = 'Germany' AND City = 'Berlin'; |  |
| 5 | SELECT \* FROM Customers WHERE City = 'Berlin' OR City = 'Stuttgart'; |  |
| 6 | SELECT \* FROM Customers WHERE Country = 'Germany' OR Country = 'Spain'; |  |
| 7 | SELECT \* FROM Customers WHERE NOT Country = 'Germany'; |  |
| 8 | SELECT \* FROM Customers WHERE Country = 'Germany' AND (City = 'Berlin' OR City = 'Stuttgart'); |  |
| 9 | SELECT \* FROM Customers WHERE NOT Country = 'Germany' AND NOT Country = 'USA'; |  |
| 10 | SELECT \* FROM Customers ORDER BY Country; |  |
| 11 | SELECT \* FROM Customers ORDER BY Country DESC; |  |
| 12 | SELECT \* FROM Customers ORDER BY Country, CustomerName; |  |
| 13 | SELECT \* FROM Customers ORDER BY Country ASC, CustomerName DESC; |  |
| 14 | INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'); |  |
| 15 | SELECT CustomerName, ContactName, Address FROM Customers WHERE Address IS NULL; |  |
| 16 | SELECT CustomerName, ContactName, Address FROM Customers WHERE Address IS NOT NULL; |  |
| 17 | UPDATE Customers SET PostalCode = 00000 WHERE Country = 'Mexico'; |  |
| 18 | DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste'; |  |
| 19 | DELETE FROM Customers; |  |
| 20 | SELECT \* FROM Customers LIMIT 3; |  |
| 21 | SELECT \* FROM Customers WHERE Country='Germany' LIMIT 3; |  |
| 22 | SELECT MIN(Price) AS SmallestPrice FROM Products; |  |
| 23 | SELECT MAX(Price) AS LargestPrice FROM Products; |  |
| 24 | SELECT COUNT(ProductID) FROM Products; |  |
| 25 | SELECT AVG(Price) FROM Products; |  |
| 26 | SELECT SUM(Quantity) FROM Order\_details; |  |
| 27 | SELECT \* FROM Customers WHERE CustomerName LIKE 'a%'; |  |
| 28 | SELECT \* FROM Customers WHERE CustomerName LIKE '%a'; |  |
| 29 | SELECT \* FROM Customers WHERE CustomerName LIKE '%or%'; |  |
| 30 | SELECT \* FROM Customers WHERE CustomerName NOT LIKE 'a%'; |  |
| 31 | SELECT \* FROM Customers WHERE Country IN ('Germany', 'France', 'UK'); |  |
| 32 | SELECT \* FROM Customers WHERE Country IN (SELECT Country FROM Suppliers); |  |
| 33 | SELECT \* FROM Customers WHERE Country NOT IN ('Germany', 'France', 'UK'); |  |
| 34 | SELECT \* FROM Products WHERE Price BETWEEN 10 AND 20 AND CategoryID NOT IN (1,2,3); |  |
| 35 | SELECT \* FROM Products  WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'  ORDER BY ProductName; |  |
| 36 | SELECT CustomerName AS Customer, ContactName AS "Contact Person" FROM Customers; |  |
| 37 | SELECT CustomerName, CONCAT\_WS(', ', Address, PostalCode, City, Country) AS Address  FROM Customers; |  |
| 38 | SELECT o.OrderID, o.OrderDate, c.CustomerName  FROM Customers AS c, Orders AS o  WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID; |  |
| 39 | SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  FROM Orders  INNER JOIN Customers  ON Orders.CustomerID=Customers.CustomerID; |  |
| 40 | SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName FROM ((Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID) INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID); | Orders intersection  Customers intersection  Shippers |
| 41 | SELECT Customers.CustomerName, Orders.OrderID FROM Customers LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID ORDER BY Customers.CustomerName; |  |
| 42 | SELECT Orders.OrderID, Employees.LastName, Employees.FirstName FROM Orders RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID ORDER BY Orders.OrderID; |  |
| 43 | SELECT Customers.CustomerName, Orders.OrderID FROM Customers CROSS JOIN Orders; |  |
| 44 | SELECT Customers.CustomerName, Orders.OrderID  FROM Customers  CROSS JOIN Orders  WHERE Customers.CustomerID = Orders.CustomerID; |  |
| 45 | SELECT A.CustomerName AS CustomerName1, A.City  FROM Customers A, Customers B  WHERE A.CustomerID <> B.CustomerID  AND A.City = B.City  ORDER BY A.City; | Self Join  The table is joined with itself.  A and B are different table aliases for the same table. |
| 46 | SELECT City FROM Customers UNION SELECT City FROM Suppliers ORDER BY City; |  |
| 47 | SELECT City FROM Customers UNION ALL SELECT City FROM Suppliers ORDER BY City; |  |
| 48 | SELECT 'Customer' AS Type, ContactName, City, Country  FROM Customers  UNION  SELECT 'Supplier', ContactName, City, Country  FROM Suppliers |  |
| 49 | SELECT Country, COUNT(CustomerID)  FROM Customers  GROUP BY Country; |  |
| 50 | SELECT Country, COUNT(CustomerID)  FROM Customers  GROUP BY Country  ORDER BY COUNT(CustomerID) DESC; |  |
| 51 | SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID GROUP BY ShipperName; |  |
| 52 | SELECT Country, COUNT(CustomerID)  FROM Customers  GROUP BY Country  HAVING COUNT(CustomerID) > 5  ORDER BY COUNT(CustomerID) DESC; |  |
| 53 | SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders FROM (Orders INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID) GROUP BY LastName HAVING COUNT(Orders.OrderID) > 10; |  |
| 54 | SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID WHERE LastName = 'Davolio' OR LastName = 'Fuller' GROUP BY LastName HAVING COUNT(Orders.OrderID) > 25; |  |
| 55 | SELECT SupplierName FROM Suppliers WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20); |  |
| 56 | SELECT ProductName FROM Products WHERE ProductID = ANY   (SELECT ProductID   FROM Order\_details   WHERE Quantity = 10); |  |
| 57 | INSERT INTO table2 (column1, column2, column3, ...) SELECT column1, column2, column3, ... FROM table1 WHERE condition; |  |
| 58 | 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 AS QuantityText FROM Order\_details; |  |
| 59 | SELECT CustomerName, City, Country FROM Customers ORDER BY (CASE     WHEN City IS NULL THEN Country     ELSE City END); |  |
| 60 | SELECT ProductName, UnitPrice \* (UnitsInStock + UnitsOnOrder) FROM Products; (In this example, if any of the "UnitsOnOrder" values are NULL, the result will be NULL.) |  |
| 61 | SELECT ProductName, UnitPrice \* (UnitsInStock + IFNULL(UnitsOnOrder, 0)) FROM Products; |  |
| 62 | SELECT ProductName, UnitPrice \* (UnitsInStock + COALESCE(UnitsOnOrder, 0)) FROM Products; |  |

The following constraints are commonly used in SQL:

* [NOT NULL](https://www.w3schools.com/mySQl/mysql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/mySQl/mysql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/mySQl/mysql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/mySQl/mysql_foreignkey.asp) - Prevents actions that would destroy links between tables
* [CHECK](https://www.w3schools.com/mySQl/mysql_check.asp) - Ensures that the values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/mySQl/mysql_default.asp) - Sets a default value for a column if no value is specified
* [CREATE INDEX](https://www.w3schools.com/mySQl/mysql_create_index.asp) - Used to create and retrieve data from the database very quickly

|  |  |  |
| --- | --- | --- |
| ALTER TABLE - ADD Column | | ALTER TABLE Customers ADD Email varchar(255); |
| ALTER TABLE - DROP COLUMN | | ALTER TABLE Customers DROP COLUMN Email; |
| ALTER TABLE - MODIFY COLUMN | | ALTER TABLE Persons MODIFY COLUMN DateOfBirth year; |
| To create a UNIQUE constraint on the "ID" column when the table is already created | | ALTER TABLE Persons ADD UNIQUE (ID); |
| To drop a PRIMARY KEY constraint | | ALTER TABLE Persons DROP PRIMARY KEY; |
| FOREIGN KEY on CREATE TABLE | | CREATE TABLE Orders (     OrderID int NOT NULL,     OrderNumber int NOT NULL,     PersonID int,     PRIMARY KEY (OrderID),     FOREIGN KEY (PersonID) REFERENCES  Persons(PersonID) ); |
| FOREIGN KEY on ALTER TABLE | | ALTER TABLE Orders ADD FOREIGN KEY (PersonID)  REFERENCES Persons(PersonID); |
| MySQL CHECK Constraint The CHECK constraint is used to limit the value range that can be placed in a column.  If you define a CHECK constraint on a column it will allow only certain values for this column.  If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row. | | |
| CREATE TABLE Persons (ID int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Age int,  CHECK (Age>=18) ); | CHECK on ALTER TABLEALTER TABLE Persons ADD CHECK (Age>=18); | |
| The DEFAULT constraint is used to set a default value for a column.  The default value will be added to all new records, if no other value is specified. | CREATE TABLE Persons (     ID int NOT NULL,     LastName varchar(255) NOT NULL,     FirstName varchar(255),     Age int,     City varchar(255) DEFAULT ‘CityName’ ); | |
| The DEFAULT constraint can also be used to insert system values, by using functions like [CURRENT\_DATE()](https://www.w3schools.com/mySQl/func_mysql_current_date.asp): | CREATE TABLE Orders (     ID int NOT NULL,     OrderNumber int NOT NULL,     OrderDate date DEFAULT CURRENT\_DATE() ); | |
| **ALTER TABLE Persons ALTER City SET DEFAULT ‘CityName’;** | ALTER TABLE Persons ALTER City DROP DEFAULT; | |
| MySQL Date Data TypesSELECT \* FROM Orders WHERE OrderDate='2008-11-11' | * DATE - format YYYY-MM-DD * DATETIME - format: YYYY-MM-DD HH:MI:SS * TIMESTAMP - format: YYYY-MM-DD HH:MI:SS | |
| MySQL CREATE VIEW Statement | CREATE VIEW Brazil\_Customers AS SELECT CustomerName, ContactName FROM Customers WHERE Country = 'Brazil'; | |
| CREATE VIEWProducts\_Above\_Average\_Price AS SELECT ProductName, Price FROM Products WHERE Price > (SELECT AVG(Price) FROM Products); | **We can query this view as follows:**  **SELECT \* FROM**Products\_Above\_Average\_Price: | |