1. Insert a new customer into the customer table.

INSET INTO Customers (FirstName, LastName, Email, Address)

VALUES (‘John’, ‘Doe’, [‘john.doe@gmail.com’](mailto:‘john.doe@gmail.com’), ‘1234 This St’);

2. Show the queries to show the number of records in each table.

SELECT COUNT(\*) AS “Number of Customers” FROM Customers;

SELECT COUNT(\*) AS “Number of Orders” FROM Orders;

SELECT COUNT(\*) AS “Number of Products” FROM Products;

3. Shows the highest priced product, the lowest priced product, and the average priced product.

SELECT

MAX(Price) AS “Highest Price”,

MIN(Price) AS “Lowest Price”,

AVG(Price) AS “Average Price”

FROM Products;

4. Shows the total quantity on hand for each product

SELECT

ProductName,

SUM(QuantityOnHand) AS “Total Quantity On Hand”

FROM Products

GROUP BY ProductName;

5. Shows each separate product name (no repeats), along with the associated category description. Sort it by category description and then by product name, both in ascending order.

SELECT DISTINCT

p.ProductName AS “Product Name”,

c.Description AS “Category Description”

FROM Products p

JOIN Categories c ON p.CategoryID = c.CategoryID

ORDER BY c.Description, p.ProductName;

6. Write the same exact query from #4 above, but group the results by category ID.

SELECT

CategoryID,

SUM(QuantityOnHand) AS “Total Quantity On Hand”

FROM Products

GROUP BY CategoryID

7. Show the vendor name, product name, quantity ordered, and quantity on hand, where the quantity ordered is greater than the quantity on hand.

SELECT

V.VendorName AS “Vendor Name”,

p.ProductName AS “Product Name”,

o.QuantityOrdered AS “Quantity Ordered”,

p.QuantityOnHand AS “Quantity On Hand”

FROM Orders o

JOIN Products p ON o.ProductID = p.ProductID

JOIN Vendors v ON p.VendorID = v.VendorID

WHERE o.QuantityOrdered > p.QuantityOnHand;

8. Write the same query, but this time do not have repeats for the vendor name.

SELECT DISTINCT

v.VendorName AS “Vendor Name”,

p.ProductName AS “Product Name”,

o.QuantityOrdered AS “Quantity Ordered”,

p.QuantityOnHand AS “Quantity On Hand”

FROM Orders o

JOIN Products p ON o.ProductID = p.ProductID

JOIN Vendors v ON p.VendorID = v.VendorID

WHERE o.QuantityOrdered > p.QuantityOnHand;

9. Gives the customer first name, customer last name, order date, ship date, retail price,

and quoted price for those records where the quoted price is not equal to the retail price. Sort by orderID in ascending order.

SELECT

c.FirstName AS "First Name",

c.LastName AS "Last Name",

o.OrderDate,

o.ShipDate,

p.RetailPrice,

o.QuotedPrice

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN Products p ON o.ProductID = p.ProductID

WHERE o.QuotedPrice <> p.RetailPrice

ORDER BY o.OrderID;

10. Write the same query as #9 above except add a calculated field on the end called Savings, which represents the difference between the retail price and the quoted price. Sort on the Savings field first in descending order, followed by the customer number in ascending order. Only show the first 100 records in the query.

SELECT

c.FirstName AS "First Name",

c.LastName AS "Last Name",

o.OrderDate,

o.ShipDate,

p.RetailPrice,

o.QuotedPrice,

(p.RetailPrice - o.QuotedPrice) AS "Savings"

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN Products p ON o.ProductID = p.ProductID

WHERE o.QuotedPrice <> p.RetailPrice

ORDER BY "Savings" DESC, c.CustomerID ASC

LIMIT 100;

11. Shows the customer first name, customer last name, order number, ship date (cast it as, for example (date) o.shipdate), order date (cast it as, for example (date) o.orderdate), product name, and retail price, but only for those orders that shipped exactly one day after they were ordered. Use the date\_add() function in your answer.

SELECT

c.FirstName AS "First Name",

c.LastName AS "Last Name",

o.OrderNumber,

CAST(o.ShipDate AS DATE) AS "Ship Date",

CAST(o.OrderDate AS DATE) AS "Order Date",

p.ProductName,

p.RetailPrice

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN Products p ON o.ProductID = p.ProductID

WHERE o.ShipDate = DATE\_ADD(o.OrderDate, INTERVAL 1 DAY);

12. Show two different queries to show all of the customers in the customers table except

the customer inputted in #1 above. Note: One of the queries must use the IN operator.

-- Using NOT IN

SELECT \* FROM Customers

WHERE CustomerID NOT IN (SELECT CustomerID FROM Customers WHERE FirstName = 'John' AND LastName = 'Doe');

-- Using NOT EXISTS

SELECT \* FROM Customers c

WHERE NOT EXISTS (SELECT 1 FROM Customers WHERE FirstName = 'John' AND LastName = 'Doe' AND CustomerID = c.CustomerID);

13. Select all customers whose first name begins with a vowel (a, e, i, o or u) AND whose

last name ends with a vowel (a, e, i, o or u). Sort results by last name, then first name, both in ascending order.

SELECT FirstName, LastName FROM Customers

WHERE FirstName REGEXP '^[aeiou]' AND LastName REGEXP '[aeiou]$'

ORDER BY LastName, FirstName;

14. Select all customers whose first name begins with a vowel (a, e, i, o or u) OR whose

last name ends with a vowel (a, e, i, o or u). Sort results by last name, then first name, both in ascending order.

SELECT FirstName, LastName FROM Customers

WHERE FirstName REGEXP '^[aeiou]' OR LastName REGEXP '[aeiou]$'

ORDER BY LastName, FirstName;

15. Show the query to select all category descriptions, product names, and retail prices.

Sort results by category description, ascending order, then retail price descending order.

SELECT

c.Description AS "Category Description",

p.ProductName,

p.RetailPrice

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

ORDER BY c.Description ASC, p.RetailPrice DESC;

16. Show the query to select all category descriptions, product names, and retail prices,

but only for a category of "Bikes" or "Wheels". Sort results by category description

in ascending order, then retail price in descending order.

SELECT

c.Description AS "Category Description",

p.ProductName,

p.RetailPrice

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

WHERE c.Description IN ('Bikes', 'Wheels')

ORDER BY c.Description ASC, p.RetailPrice DESC;

17. Show the query to select all category descriptions, product names, and retail prices,

but only for a category not "Bikes" or "Wheels". Sort results by category description

in ascending order, then retail price in descending order.

SELECT

c.Description AS "Category Description",

p.ProductName,

p.RetailPrice

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

WHERE c.Description NOT IN ('Bikes', 'Wheels')

ORDER BY c.Description ASC, p.RetailPrice DESC;

18. Show the query to select all category descriptions, product names, retail price,

wholesale price, and vendor name. Sort results by category description in ascending

order, then vendor name in descending order, then retail price in descending order,

then wholesale price in descending order.

SELECT

c.Description AS "Category Description",

p.ProductName,

p.RetailPrice,

p.WholesalePrice,

v.VendorName

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

JOIN Vendors v ON p.VendorID = v.VendorID

ORDER BY c.Description ASC, v.VendorName DESC, p.RetailPrice DESC, p.WholesalePrice DESC;

19. Show the query to select all category descriptions, product names, retail price,

wholesale price, and vendor name. Sort results by category description in ascending order, then vendor name in descending order, then retail price in descending order, then wholesale price in descending order, but only for vendor "Lone Star Bike Supply" OR category "Bikes".

SELECT

c.Description AS "Category Description",

p.ProductName,

p.RetailPrice,

p.WholesalePrice,

v.VendorName

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

JOIN Vendors v ON p.VendorID = v.VendorID

WHERE v.VendorName = 'Lone Star Bike Supply' OR c.Description = 'Bikes'

ORDER BY c.Description ASC, v.VendorName DESC, p.RetailPrice DESC, p.WholesalePrice DESC;

20. Show the query to select all category descriptions, product names, retail price,

wholesale price, and vendor name. Sort results by category description in ascending order, then vendor name in descending order, then retail price in descending order, then wholesale price in descending order, but only for vendor "Lone Star Bike Supply" AND category "Bikes".

SELECT

c.Description AS "Category Description",

p.ProductName,

p.RetailPrice,

p.WholesalePrice,

v.VendorName

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

JOIN Vendors v ON p.VendorID = v.VendorID

WHERE v.VendorName = 'Lone Star Bike Supply' AND c.Description = 'Bikes'

ORDER BY c.Description ASC, v.VendorName DESC, p.RetailPrice DESC, p.WholesalePrice DESC;

21. Show the query to select all employee first name, employee last name, order ID,

product ID, and quantity ordered. Sort results by quantity ordered descending.

SELECT

e.FirstName AS "First Name",

e.LastName AS "Last Name",

o.OrderID,

o.ProductID,

o.QuantityOrdered

FROM Employees e

JOIN Orders o ON e.EmployeeID = o.EmployeeID

ORDER BY o.QuantityOrdered DESC;

22. Show the query to select all employee first name, employee last name, order ID,

quoted price, and quantity ordered where the quantity ordered is greater than 4.

Sort results by quantity ordered in descending order and then by quoted price in

descending order.

SELECT

e.FirstName AS "First Name",

e.LastName AS "Last Name",

o.OrderID,

o.QuotedPrice,

o.QuantityOrdered

FROM Employees e

JOIN Orders o ON e.EmployeeID = o.EmployeeID

WHERE o.QuantityOrdered > 4

ORDER BY o.QuantityOrdered DESC, o.QuotedPrice DESC;

23. Show the query to select all employee first name, employee last name, order ID,

quoted price, and quantity ordered where the quantity ordered not equal to 4.

Sort results by quantity ordered in descending order and then by quoted price in

descending order.

SELECT

e.FirstName AS “First Name”,

e.LastName AS “Last Name”,

o.OrderID,

o.QuotedPrice,

o.QuantityOrdered

FROM Employees e

JOIN Orders o ON e.EmployeeID = o.EmployeeID

WHERE o.QuantityOrdered <> 4

ORDER BY o.QuantityOrdered DESC, o.QuotedPrice DESC;

24. Do a Union query to combine all customer first and last names with all employee

first and last names.

SELECT FirstName, LastName FROM Customers

UNION

SELECT FirstName, LastName FROM Employees;

25. Show the query to select each customer's first name, last name, order number,

product number, and quoted price.

SELECT

c.FirstName AS “First Name”,

c.LastName AS “Last Name”,

o.OrderNumber,

o.ProductID,

o.QuotedPrice

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID