Write SQL queries for each question below. For each example, use aliases for any table names (for any table joins) and for field names to rename them if/as necessary. For example, use “Product Number” and not ProductNumber.

1. Insert a new customer into the customer table.

INSERT INFO customers

(CustomerFirstName, CustomerLastName, CustomerStreetAddress

CustomerCity, CustomerState, CustomerZipCode, CustomerAreaCode,

CustomerPhoneNumber)

VALUES

(‘Danny’, ‘Gardiner’, ’16 Sassafras Parc Ct.’, ‘O’Fallon’, ’MO’, ‘63368’,

‘636’, ‘6364850142’)

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

SELECT COUNT(\*) AS ‘# Category Records’ FROM categories;

SELECT COUNT(\*) AS ‘# Customer Records’ FROM customers;

SELECT COUNT(\*) AS ‘# Employee Records’ FROM employees;

SELECT COUNT(\*) AS ‘# Vendor Records’ FROM vendors;

SELECT COUNT(\*) AS ‘# Product Records’ FROM products;

SELECT COUNT(\*) AS ‘# ProductVendor Records’ FROM productVendors;

SELECT COUNT(\*) AS ‘# Order Records’ FROM orders;

SELECT COUNT(\*) AS ‘# OrderDetail Records’ FROM orderDetails;

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

SELECT MAX(ProductPrice) AS ‘HIGHEST PRICE’,

MIN(ProductPrice) AS ‘LOWEST PRICE’,

AVG(ProductPrice) AS ‘AVERAGE PRICE’

FROM Products

1. Shows the total quantity on hand for each product

SELECT ProductID AS ‘Product ID’, ProductName AS ‘Product Name’,

SUM(ProductQty) AS ‘Total Qty’

FROM Products

GROUP BY ProductID, ProductName

ORDER BY ‘Total Qty’ DESC, ProductName

1. 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), c.CategoryDescription

FROM Products p

JOIN Categories c ON p.CategoryID = c.CategoryID

ORDER BY c.CategoryDescription ASC, p.ProductName ASC

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

SELECT p.ProductName, c.CategoryDescription

FROM Products p

JOIN Categories c ON p.CategoryID = c.CategoryID

GROUP BY p.ProductName, c.CategoryDescription

ORDER BY c.CategoryDescription ASC, p.ProductName ASC

1. 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, p.ProductName, od.OrderDetailQuantityOrdered,

p.ProductQty

FROM Vendors V

JOIN ProductVendors pv ON v.VendorID =pv.VendorID

JOIN Products p On p.ProductID = pv.ProductD

JOIN OrderDetails od ON p.ProductID = od.ProductID

WHERE od.OrderDetailQuantityOrdered > p.ProductQty;

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

SELECT DISTINCT v.VendorName, p.ProductName,

od.OrderDetailQuantityOrdered, p.ProductQty

FROM Vendors V

JOIN ProductVendors pv ON v.VendorID =pv.VendorID

JOIN Products p On p.ProductID = pv.ProductD

JOIN OrderDetails od ON p.ProductID = od.ProductID

WHERE od.OrderDetailQuantityOrdered > p.ProductQty;

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.CustomerFirstName, c.CustomerLastName, o.OrderShipDate,

p.ProductPrice,

od.OrderDetailQuotedPrice

FROM customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od on o.OrderID = od.OrderID

JOIN Products p On od.ProductID = p.ProductID

WHERE od.OrderDetailQuotedPrice != p.ProductPrice

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 TOP 100 c.CustomerFirstName, c.CustomerLastName, o.OrderShipDate,

p.ProductPrice, od.OrderDetailQuotedPrice,

(p.ProductPrice - od.OrderDetailQuotedPrice) AS SAVINGS

FROM customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od on o.OrderID = od.OrderID

JOIN Products p On od.ProductID = p.ProductID

WHERE od.OrderDetailQuotedPrice != p.ProductPrice

ORDER BY SAVINGS DESC, c.CustomerID ASC;

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.CustomerFirstName, c.CustomerLastName, o.OrderID,

CAST(o.OrderShipDate AS DATE)

CAST(o.OrderDate AS DATE)

p.ProductName, p.ProductPrice

FROM Customer c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON od.OrderID = o.OrderID

JOIN Products p ON od.ProductID = p.ProductID

WHERE DATEDIFF(day, o.orderDate, o.OrderShipDate) = 1;

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.

SELECT \* FROM Customers WHERE CustomerID != 1031

SELECT \* FROM Customers WHERE CustomerID <> 1031

SELECT \* FROM Customers WHERE CustomerID NOT IN (1031)

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 \*

FROM Customers

WHERE customerFirstName LIKE ‘[aeiou]%’

AND customerLastName LIKE ‘%[aeiou]’

ORDER BY customerlastname, customerfirstName

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 \*

FROM Customers

WHERE customerFirstName LIKE ‘[aeiou]%’

OR customerLastName LIKE ‘%[aeiou]’

ORDER BY customerlastname, customerfirstName

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.categorydescription, p.productName, p.productprice AS RETAILPRICE

FROM categories c

JOIN products p on c.categoryid = p.categoryid

ORDER BY c.categorydescription, p.productprice 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.categorydescription, p.productName, p.productprice AS RETAILPRICE

FROM categories c

JOIN products p on c.categoryid = p.categoryid

WHERE c.categorydescription IN (‘Bikes’, ‘Wheels’)

ORDER BY c.categorydescription, p.productprice 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.categorydescription, p.productName, p.productprice AS RETAILPRICE

FROM categories c

JOIN products p on c.categoryid = p.categoryid

WHERE c.categorydescription NOT IN (‘Bikes’, ‘Wheels’)

ORDER BY c.categorydescription, p.productprice 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.categorydescription, p.productname, p.productprice AS RetailPrice,

Pv.productvendorwholesaleprice, v.vendorname

FROM categories c

JOIN products p on c.categoryid

JOIN productvendors pv on p.productid = pv.productid

JOIN vendors v on pv.vendorid = v.vendorid

ORDER BY c.categorydescription, v.vendorname desc, p.productprice desc,

pv.productvendorwholesaleprice 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.categorydescription, p.productname, p.productprice AS RetailPrice,

Pv.productvendorwholesaleprice, v.vendorname

FROM categories c

JOIN products p on c.categoryid

JOIN productvendors pv on p.productid = pv.productid

JOIN vendors v on pv.vendorid = v.vendorid

WHERE v.vendorname = ‘Lone Star Bike Supply’

OR c.categorydescription = ‘bikes’

ORDER BY c.categorydescription, v.vendorname desc, p.productprice desc,

pv.productvendorwholesaleprice 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.categorydescription, p.productname, p.productprice AS RetailPrice,

Pv.productvendorwholesaleprice, v.vendorname

FROM categories c

JOIN products p on c.categoryid

JOIN productvendors pv on p.productid = pv.productid

JOIN vendors v on pv.vendorid = v.vendorid

WHERE v.vendorname = ‘Lone Star Bike Supply’

AND c.categorydescription = ‘bikes’

ORDER BY c.categorydescription, v.vendorname desc, p.productprice desc,

pv.productvendorwholesaleprice 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.employeefirstname, e.employeelastname,

od.orderid, od.productid,

FROM employees e

JOIN orders o on e.employeeid = o.employeeid

JOIN orderdetails od on o.orderid = od.orderid

ORDER BY od.orderdetailquantityordered 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.employeefirstname, e.employeelastname,

od.orderid, od.productid,

FROM employees e

JOIN orders o on e.employeeid = o.employeeid

JOIN orderdetails od on o.orderid = od.orderid

WHERE od.orderdetailquantityordered > 4

ORDER BY od.orderdetailquantityordered 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.employeefirstname, e.employeelastname,

od.orderid, od.productid,

FROM employees e

JOIN orders o on e.employeeid = o.employeeid

JOIN orderdetails od on o.orderid = od.orderid

WHERE od.orderdetailquantityordered != 4

ORDER BY od.orderdetailquantityordered desc;

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

first and last names.

SELECT customerfirstname, customerlastname

FROM customers

UNION

SELECT employeefirstname, employeelastname

FROM employees;

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

product number, and quoted price.

SELECT c.customerfirstname, c.customerlastname,

o.orderid, od.productid, od.orderdetailquotedprice

FROM customers c

JOIN orders o ON c.customerid = o.customerid

JOIN orderdetails od ON od.orderid = o.orderid;