

Show the associated query(ies) for each question below. Write each query in the "best" way possible, i.e., think before you do please. I realize that some of the queries may look a little weird, random, or whatever. Please do them anyway.

1. You have been asked to insert the following 3 records into the customers table:

CustomerFirstName: Evan

CustomerLastName: Gudmestad

CustomerStreetAddress: 4431 Finney Avenue

CustomerCity: St. Louis

CustomerState: MO

CustomerZipCode 63113

CustomerAreaCode: 314

CustomerPhoneNumber: 286-3691

CustomerFirstName: Charles

CustomerLastName: Corrigan

Customer Street Address: 751 Parr Rd.

CustomerCity: Wentzville

CustomerState: MO

CustomerZipCode 63385

CustomerAreaCode: 314

CustomerPhoneNumber: 286-4848

CustomerFirstName: Jeff

CustomerLastName: Scott

CustomerStreetAddress: 751 Parr Rd.

CustomerCity: Wentzville

CustomerState: MO

CustomerZipCode 63385

CustomerAreaCode: 314

CustomerPhoneNumber: 286-3675

Show how to add all 3 of these records to the Customers table using a single insert query.

**INSERT INTO customers (CustomerFirstName, CustomerLastName,**

**CustomerStreetAddress, CustomerCity, CustomerState, CustomerZipCode, CustomerAreaCode, CustomerPhoneNumber)**

**VALUES**

**('Evan', 'Gudmestad', '4431 Finney Avenue', 'St. Louis', 'MO', '63113', 314, '286-3691');**

**INSERT INTO customers (CustomerFirstName, CustomerLastName,**

**CustomerStreetAddress, CustomerCity, CustomerState, CustomerZipCode, CustomerAreaCode, CustomerPhoneNumber)**

**VALUES**

**('Charles', 'Corrigan', '751 Parr Rd.', 'Wentzville', 'MO', '63385', 314, '286-4848');**

**INSERT INTO customers (CustomerFirstName, CustomerLastName,**

**CustomerStreetAddress, CustomerCity, CustomerState, CustomerZipCode, CustomerAreaCode, CustomerPhoneNumber)**

**VALUES**

**('Jeff', 'Scott', '751 Parr Rd.', 'Wentzville', 'MO', '63385', 314, '286-3675' );**

2. You have been asked to do further updates to the Customers table. First, show the query to list each customer's current city, state, and zip code, but only for those customers in any of the following cities only:

Austin, Palm Springs, San Diego, Seattle

Then, using the results of this query, make the following changes to the Customers table using/showing four different UPDATE queries. Also tell how many total records were affected from all 4 queries.

Change all Austin TX zipcodes to 73301.

Change all Palm Springs CA zipcodes to 92292.

Change all San Diego CA zipcodes to 91911.

Change all Seattle WA zipcodes to 98117.

**SELECT CustomerCity AS 'Customer City',**

**CustomerState AS 'Customer State',**

**CustomerZipCode AS 'Customer Zip Code'**

**FROM customers**

**WHERE CustomerCity IN ('Austin', 'Palm Springs', 'San Diego', 'Seattle');**

**UPDATE customers**

**SET CustomerZipCode = '73301'**

**WHERE CustomerCity = 'Austin'**

**AND CustomerState = 'TX';**

**UPDATE customers**

**SET CustomerZipCode = '92292'**

**WHERE CustomerCity = 'Palm Springs'**

**AND CustomerState = 'CA';**

**UPDATE customers**

**SET CustomerZipCode = 91911**

**WHERE CustomerCity = 'San Diego'**

**AND CustomerState = 'CA';**

**UPDATE customers**

**SET CustomerZipCode = 98117**

**WHERE CustomerCity = 'Seattle'**

**AND CustomerState = 'WA';**

3. Show the query to use a right join to search for any/all customers who have orders. This should be all customers except Evan G., Charles C., and Jeff S. Next, show the query to delete Jeff Scott from the table.

**SELECT DISTINCT(CONCAT(c.Customerfirstname, ' ' ,**

**c.Customerlastname)) as "Customer Name",**

**o.OrderID as "Order #"**

**FROM customers c**

**RIGHT JOIN orders o**

**ON c.Customerid = o.Customerid;**

**DELETE**

**FROM customers**

**WHERE customerid = '1032 ';**

4. Show the Alter Table query to add the following new field to the end of the Employees table.

NAME TYPE

-----------------------------------------------------

employeehiredate date

**ALTER TABLE  Employees  
ADD  COLUMN EmployeeHireDate date;**

Then, write the 8 update query commands to update the Employees table records as follows:

FULL NAME HIRE DATE

--------------------------------------------------------------

Ann Patterson 2010-09-21

Mary Thompson 2011-11-11

Matt Berg 2007-12-13

Carol Viescas 2000-09-19

Kirk DeGrasse 2000-10-10

David Viescas 2000-09-19

Kathryn Patterson 2015-02-02

Susan McLain 2017-09-18

**UPDATE employees**

**SET employeeHireDate = '2010-09-21'**

**WHERE employeeID = 701;**

**UPDATE employees**

**SET employeeHireDate = '2011-11-11'**

**WHERE employeeID = 702;**

**UPDATE employees**

**SET employeeHireDate = '2007-12-13'**

**WHERE employeeID = 703;**

**UPDATE employees**

**SET employeeHireDate = '2000-09-19'**

**WHERE employeeID = 704;**

**UPDATE employees**

**SET employeeHireDate = '2000-10-10'**

**WHERE employeeID = 705;**

**UPDATE employees**

**SET employeeHireDate = '2000-09-19'**

**WHERE employeeID = 706;**

**UPDATE employees**

**SET employeeHireDate = '2015-02-02'**

**WHERE employeeID = 707;**

**UPDATE employees**

**SET employeeHireDate = '2017-09-18'**

**WHERE employeeID = 708;**

5. Write a query that shows each employee first name, employee last name, hire date, and the number of days they have been employed. Use the MySQL curdate() function for the current date.

**SELECT employeeFirstName as "Employee First Name",**

**employeeLastName as "Employee Last Name",**

**employeeHireDate as "Employee Hire Date",**

**DATEDIFF(curdate(), employeeHireDate)**

**as "Days Employed"**

**FROM employees;**

6. Write a query that shows the total number of days between the oldest hire date employee and the newest hire date employee. Also, write an aggregate query that shows the earliest hire date, the latest hire date, the average hire date, and the of unique hire dates.

**SELECT MAX(employeehiredate) as "Newest Hire",**

**MIN(employeehiredate) as " Oldest Hire",**

**DATEDIFF(MAX(employeehiredate),**

**MIN(employeeHireDate))**

**as "Diff Days"**

**FROM employees;**

7. Show the Alter Table query to add the following new field to the end of the Product\_Vendors table.

NAME TYPE

-----------------------------------------------------

expectedDeliverDate date

**ALTER TABLE  ProductVendors  
ADD  COLUMN productVendorsdeliveryDate date;**

Show the query to Fill the expected delivery date by adding the deliveryDateamount to the current date. Again, use the MySQL curdate() function for the current date. Show all fields, including the new calculated field and sort by the actual delivery date descending .

**SELECT ProductID AS "Product #",**

**VendorID AS "Vendor ID",**

**ProductVendorWholesalePrice AS "Wholesale Price",**

**ProductVendorDaysToDeliver AS "Days to Delivery",**

**DATE\_ADD(curdate(), INTERVAL ProductVendorDaysToDeliver DAY)**

**AS "Actual Delivery Date"**

**FROM ProductVendors;**

8. Show a query that provides the following info: product name, quantity ordered, retail price and wholesale price. Sort the results based on quantity ordered in descending order, retail price descending, wholesale price descending.

**SELECT p.ProductName AS "Product Name",**

**od. OrderDetailQuantityOrdered AS "Qty Ordered",**

**p.ProductPrice AS "Retail Price",**

**pv.ProductVendorWholesalePrice AS "Wholesale Price"**

**FROM products p,**

**orderdetails od,**

**productvendors pv**

**WHERE p.ProductID = pv.ProductID**

**AND od.ProductID = p.ProductID**

**ORDER BY p.ProductQty desc,**

**p.ProductPrice desc,**

**pv.ProductVendorWholesalePrice desc;**

9. Rewrite the query above. Use an IF statement to check on the quantity ordered. If said quantity is >= 5, add a new calculated column called “Updated Wholesale Price”, which is the wholesale price \* .75 rounded to 2 decimal places.

**SELECT p.ProductName AS "Product Name",**

**od. OrderDetailQuantityOrdered AS "Qty Ordered",**

**p.ProductPrice AS "Retail Price",**

**pv.ProductVendorWholesalePrice AS "Wholesale Price",**

**IF (p.ProductQty >= 5,**

**ROUND(pv.ProductVendorWholesalePrice \* .75, 2),**

**pv.ProductVendorWholesalePrice) AS "Updated Price"**

**FROM products p,**

**orderdetails od,**

**productvendors pv**

**WHERE p.ProductID = pv.ProductID**

**AND od.ProductID = p.ProductID**

**ORDER BY p.ProductQty desc,**

**p.ProductPrice desc,**

**pv.ProductVendorWholesalePrice desc;**

10. From the Customers table, write a query that provides the customer first name, customer last name, customer street address, custcity and a “calculated column” called “StateName” which uses a MySQL case statement to show the state name based on the state abbreviation (CA, OR, TX, WA, or Unknown).

**SELECT customerfirstname AS "Cust First Name",**

**customerlastname AS "Cust Last Name",**

**customerstreetaddress AS "Cust Street Address",**

**CASE WHEN customerstate = 'CA' THEN 'California'**

**WHEN customerstate = 'OR' THEN 'Oregon'**

**WHEN customerstate = 'TX' THEN 'Texas'**

**WHEN customerstate = 'WA' THEN 'Washington'**

**ELSE 'Unknown'**

**END AS StateName**

**FROM customers**

**ORDER BY StateName;**

11. Show a query that provides the customer first name, customer last name, and sum of quantity ordered. Group the results by customer last name, followed by quantity ordered in descending order.

**SELECT CONCAT (c.CustomerFirstName, ' ',**

**c.CustomerLastName) AS 'Customer Name',**

**od.OrderDetailQuantityOrdered**

**FROM customers c, orders o, orderdetails od**

**WHERE c.CustomerID = o.CustomerID**

**AND o.OrderID = od.OrderID**

**GROUP BY customerlastname**

**ORDER BY od.OrderDetailQuantityOrdered Desc;**

12. Repeat the query in #11 above, but only show the results for those customers with a sum of quantity ordered less than or equal to 4.

**SELECT CONCAT (c.CustomerFirstName, ' ',**

**c.CustomerLastName) AS 'Customer Name',**

**od.OrderDetailQuantityOrdered**

**FROM customers c, orders o, orderdetails od**

**WHERE c.CustomerID = o.CustomerID**

**AND o.OrderID = od.OrderID**

**GROUP BY customerlastname**

**HAVING od.OrderDetailQuantityOrdered <= 4**

**ORDER BY od.OrderDetailQuantityOrdered Desc;**