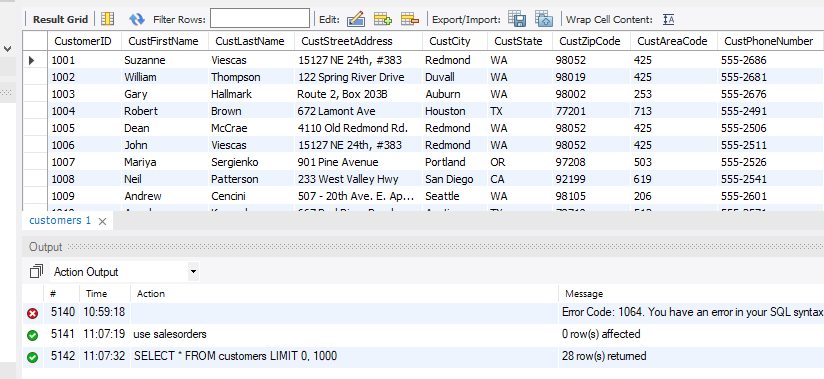
For this assignment, write queries using SQL to acquire data about customers, vendors, products, and employees in a fictitious sales database. These queries will cover many of the core aspects of writing SQL to produce data for reporting and analyzing information. There may be multiple ways to produce the same results, but ensure you are returning the requested fields.

Using the Sales Orders database, complete the queries below.

1. **Show all the information on our customers**.
2. Query:

SELECT \* FROM customers;

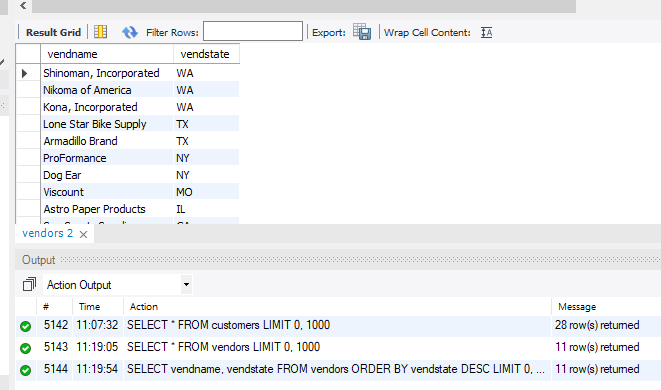
1. Columns: all in customers table
2. Expected Row Count: returned 28
3. Screenshot:



1. **Show a list of states, in reverse alphabetical order, where our vendors are located, and include the names of the vendor.**
2. Query:

SELECT vendname, vendstate FROM vendors  
ORDER BY vendstate DESC;

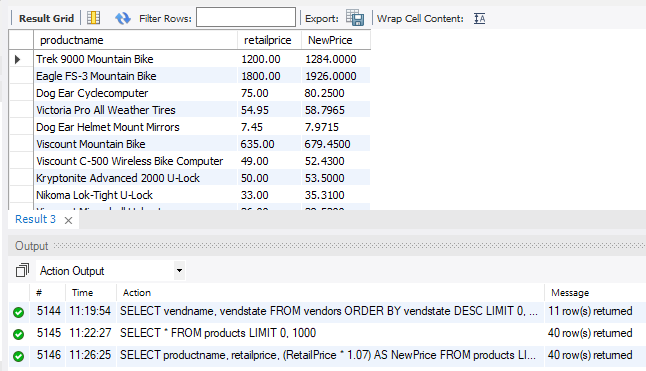
1. Columns: vendname, vendstate
2. Expected Row Count: returned 11
3. Screenshot:



1. **What if we adjusted the retail price of each product by increasing it 7 percent?**
2. Query:

SELECT productname, retailprice, (RetailPrice \* 1.07) AS NewPrice FROM products;

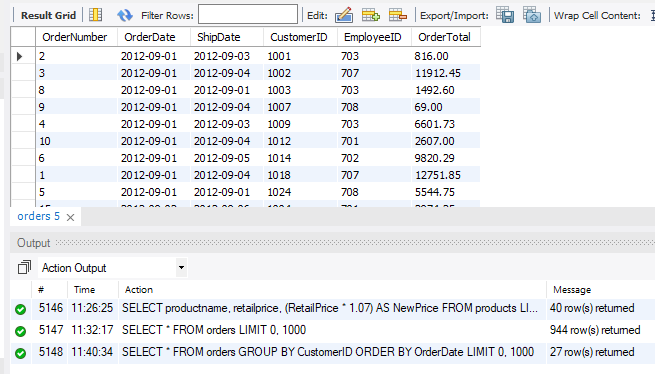
1. Columns: productname, retailprice, newprice
2. Expected Row Count: returned 40
3. Screenshot:



1. **Show a list of orders made by each customer in ascending date order.**
2. Query:

SELECT \* FROM orders  
GROUP BY CustomerID  
ORDER BY OrderDate;

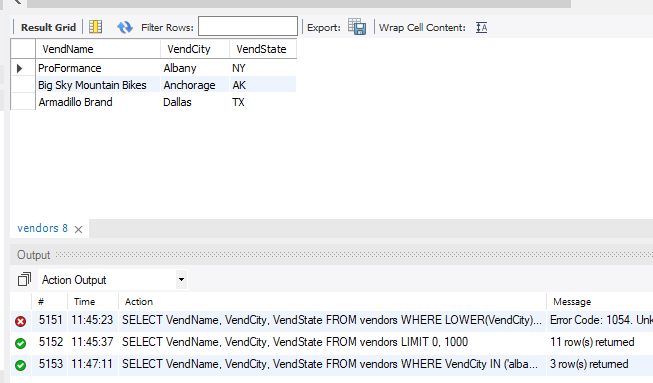
1. Columns: all from orders
2. Expected Row Count: returned 27
3. Screenshot:



1. **Give the names of all vendors based in Albany, Anchorage, and Dallas.**
2. Query:

SELECT VendName, VendCity, VendState FROM vendors  
WHERE VendCity IN ('albany', 'anchorage', 'dallas');

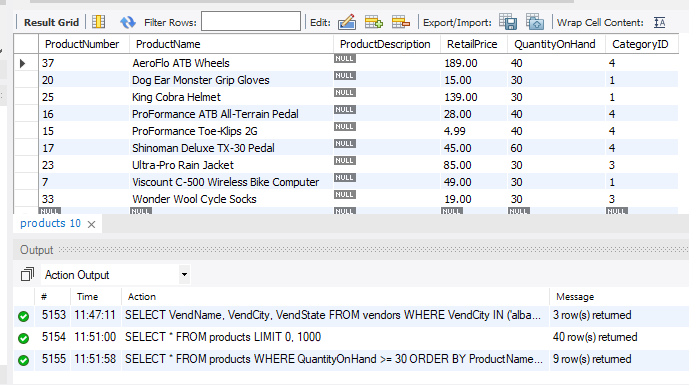
1. Columns: vendname, vendcity, vendstate
2. Expected Row Count: returned 3
3. Screenshot:



1. **Show an alphabetized list of products with a quantity on hand greater than or equal to 30.**
2. Query:

SELECT \* FROM products  
WHERE QuantityOnHand >= 30  
ORDER BY ProductName;

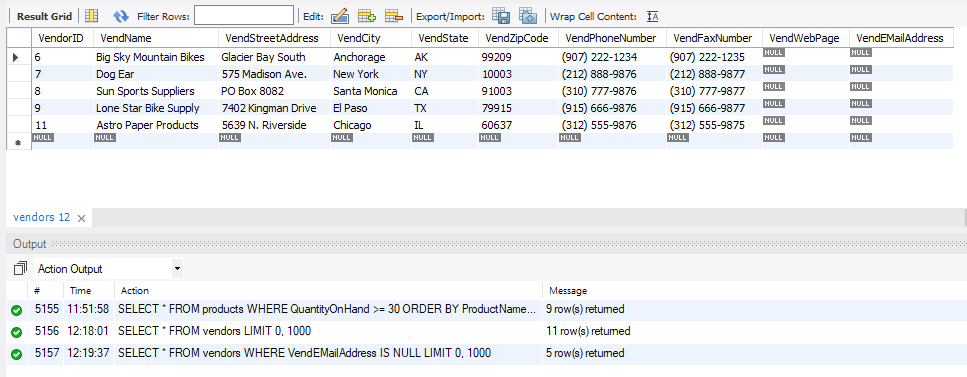
1. Columns: all from products
2. Expected Row Count: returned 9
3. Screenshot:



1. **What vendors do we work with that don’t have an email address?**
2. Query:

SELECT \* FROM vendors  
WHERE VendEMailAddress IS NULL;

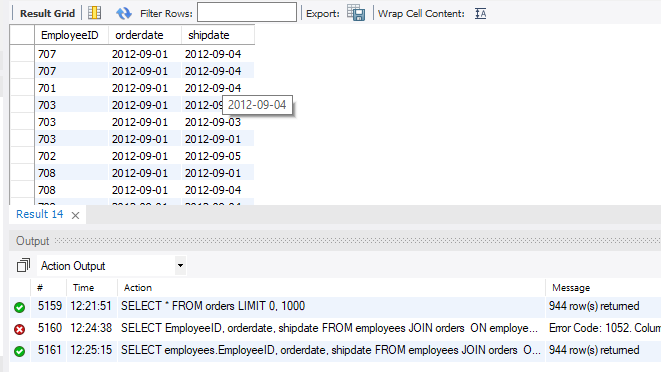
1. Columns: all from vendors
2. Expected Row Count: returned 5
3. Screenshot:



1. **List employees and the dates their orders shipped sorted by order date.**
2. Query:

SELECT employees.EmployeeID, orderdate, shipdate FROM employees  
JOIN orders   
ON employees.EmployeeID = orders.EmployeeID  
ORDER BY orders.OrderDate;

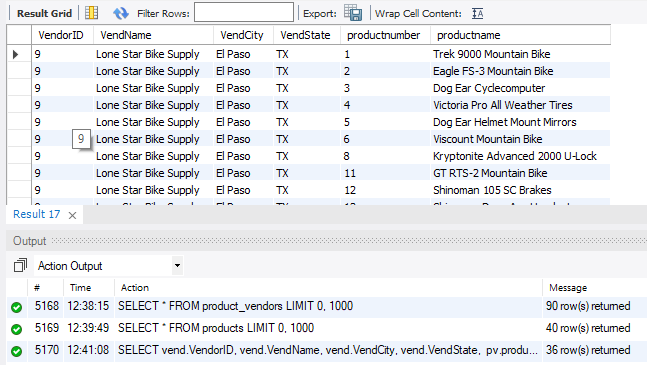
1. Columns: employeeid, orderdate, shipdate
2. Expected Row Count: returned 944
3. Screenshot:



1. **Show the vendors and products they supply to us for products over $75 for vendors in Texas.**
2. Query:

SELECT vend.VendorID, vend.VendName, vend.VendCity, vend.VendState,   
pv.productnumber, prod. productname  
FROM vendors vend  
JOIN product\_vendors pv ON vend.vendorid = pv.vendorid  
JOIN products prod ON pv.ProductNumber = prod.ProductNumber  
WHERE VendState LIKE 'TX';

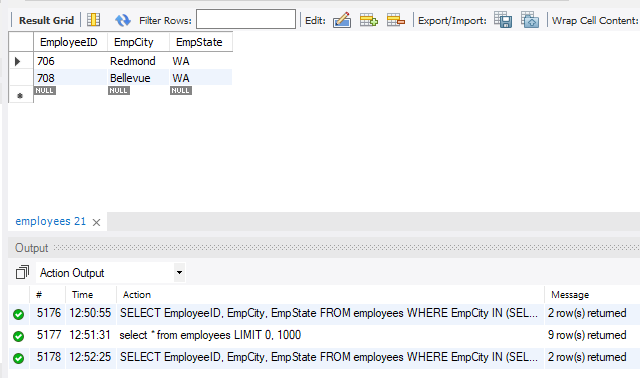
1. Columns: vendorid, vendname, vendcity, vendstate, productnumber, productname
2. Expected Row Count: returned 36
3. Screenshot:



1. **Show employees who live in the same city and state as our vendors.**
2. Query:

SELECT EmployeeID, EmpCity, EmpState  
FROM employees  
WHERE EmpCity IN (SELECT vendcity FROM vendors)  
AND EmpState IN (SELECT vendstate FROM vendors)  
ORDER BY EmpState;

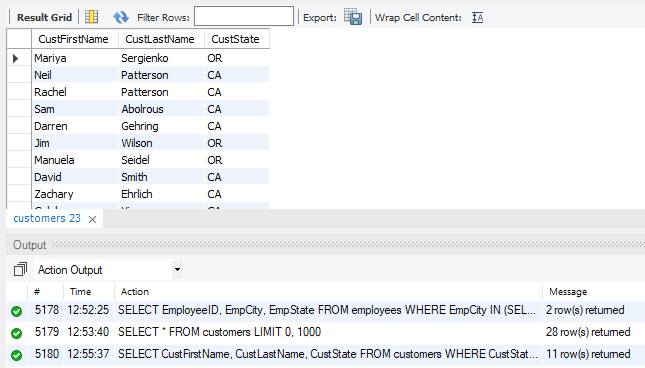
1. Columns: employeeid, empcity, empstate
2. Expected Row Count: returned 2
3. Screenshot:



1. **Display customers who have no sales rep (employees) in the same state.**
2. Query:

SELECT CustFirstName, CustLastName, CustState FROM customers  
WHERE CustState NOT IN (SELECT empstate FROM employees);

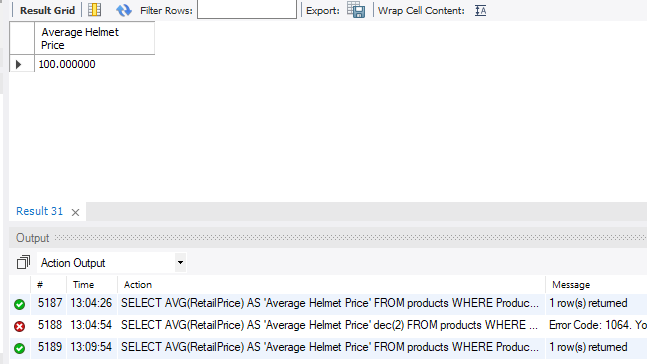
1. Columns: custfirstname, custlastname, custstate
2. Expected Row Count: returned 11
3. Screenshot:



1. **What is the average quoted price of a helmet?**
2. Query:

SELECT AVG(RetailPrice) AS 'Average Helmet Price' FROM products  
WHERE ProductName LIKE '%helmet';

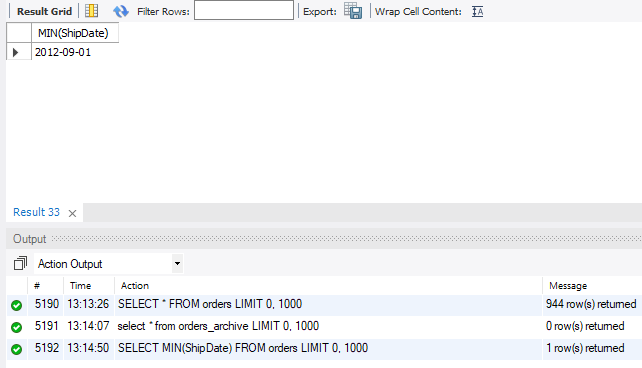
1. Columns: average helmet price
2. Expected Row Count: 1
3. Screenshot:



1. **What was the date of the earliest ship date?**
2. Query:

SELECT MIN(ShipDate) FROM orders;

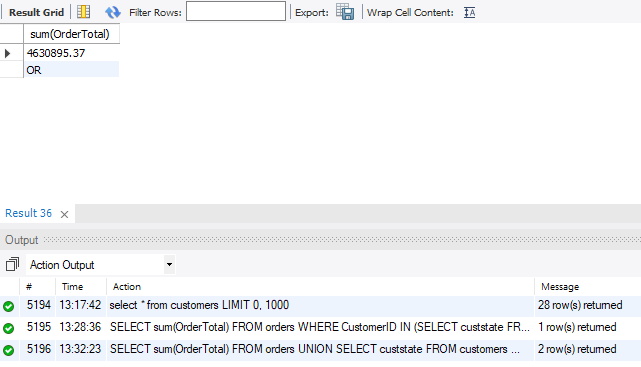
1. Columns: shipdate
2. Expected Row Count: 1
3. Screenshot:



1. **What is the total amount (in dollars) of orders from the state of Oregon?**
2. Query:

SELECT sum(OrderTotal) FROM orders  
UNION  
SELECT custstate FROM customers  
WHERE CustState LIKE 'OR';

1. Columns: sum(ordertotal), custstate
2. Expected Row Count: 2
3. Screenshot:



1. **Show each employee, the employee’s total sales (in dollars), the employee’s total sales item quantity, and the average item sales price ordered by the employee’s average item sales price highest to lowest.**
2. Query:

SELECT DISTINCT EmployeeID, sum(OrderTotal), count(OrderNumber), avg(OrderTotal) FROM orders  
GROUP BY EmployeeID  
ORDER BY avg(OrderTotal) DESC;

1. Columns: employeeid, sum(ordertotal), count(ordernumber), avg(ordertotal)
2. Expected Row Count: returned 8
3. Screenshot:

