**Production Exam 3: Chapters 7, 9 & 12   
100 points max – point value as indicated**

Use the DB indicated in each question. Show all SQL statements and result sets for each problem.

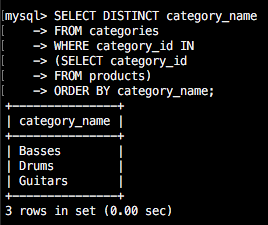
1. (**MGS** – 5 points) Write a SELECT statement that returns the same result set as this SELECT statement, but do not use a join. Instead, use a subquery in a WHERE clause that uses the IN keyword.

SELECT DISTINCT category\_name

FROM categories c JOIN products p

ON c.category\_id = p.category\_id

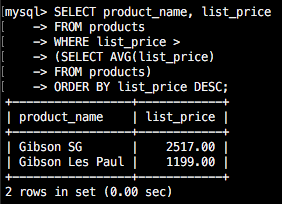
ORDER BY category\_name;

****

1. (**MGS** – 5 points) Write a SELECT statement that answers this question: Which products have a list price that’s greater than the average list price for all products?

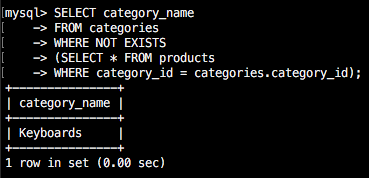
Return the product\_name and list\_price columns for each product.

Sort the results by the list\_price column in descending sequence.

****

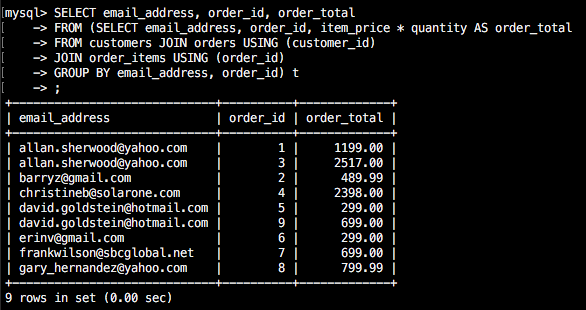
1. (**MGS** – 6 points) Write a SELECT statement that returns the category\_name column from the Categories table.

Return one row for each category that has never been assigned to any product in the Products table. Use a subquery introduced with the NOT EXISTS operator, not an outer join.



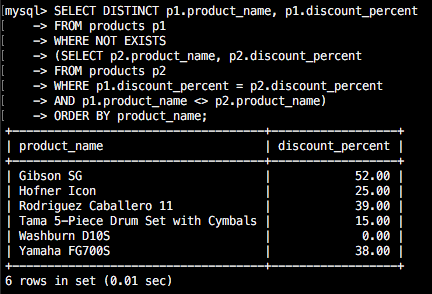
1. (**MGS** – 7 points) Write a SELECT statement that returns three columns: email\_address, order\_id, order total for each customer. HINT: You will most likely need to group by two columns. In addition, you must calculate the order total from the columns in the Order\_Items table.

Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer’s email address and the largest order for that customer.

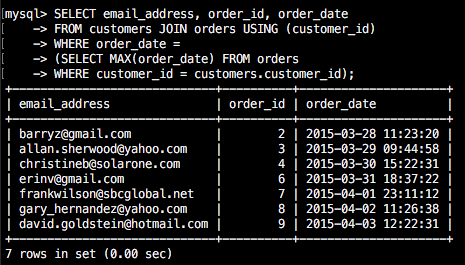
****

1. (**MGS** – 5 points) Write a SELECT statement that returns the name and discount percent of each product that has a unique discount percent. In other words, don’t include products that have the same discount percent as another product.

Sort the results by the product\_name column.

****

1. (**MGS** – 7 points) Use a correlated subquery to return one row per customer, representing the customer’s oldest order. Each row should include these three columns: email\_address, order\_id, and order\_date.

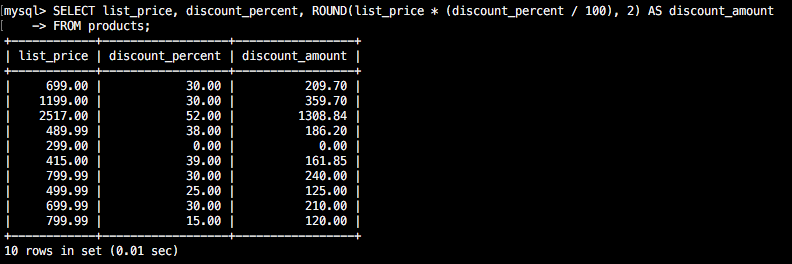
****

1. (**MGS** – 5 points) Write a SELECT statement that returns these columns from the Products table:

The list\_price column

The discount\_percent column

A column named discount\_amount that uses the previous two columns to calculate the discount amount and uses the ROUND function to round the result so it has 2 decimal digits.

****

1. (**MGS** – 6 points) Write a SELECT statement that returns these columns from the Orders table:

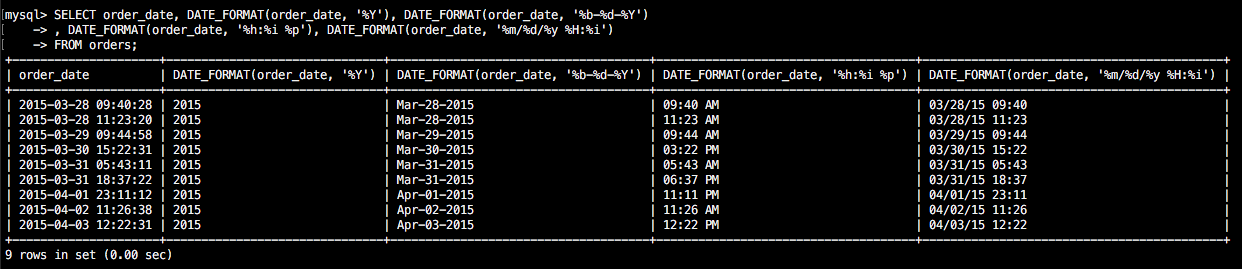
The order\_date column

A column that uses the DATE\_FORMAT function to return the four-digit year that’s stored in the order\_date column

A column that uses the DATE\_FORMAT function to return the order\_date column in this format: Mon-DD-YYYY. In other words, use abbreviated months and separate each date component with dashes.

A column that uses the DATE\_FORMAT function to return the order\_date column with only the hours and minutes on a 12-hour clock with an am/pm indicator

A column that uses the DATE\_FORMAT function to return the order\_date column in this format: MM/DD/YY HH:MM. In other words, use two-digit months, days, and years and separate them by slashes. Use 2-digit hours and minutes on a 24-hour clock. And use leading zeros for all date/time components.

****

1. (**MGS** – 8 points) Write a SELECT statement that returns these columns from the Orders table:

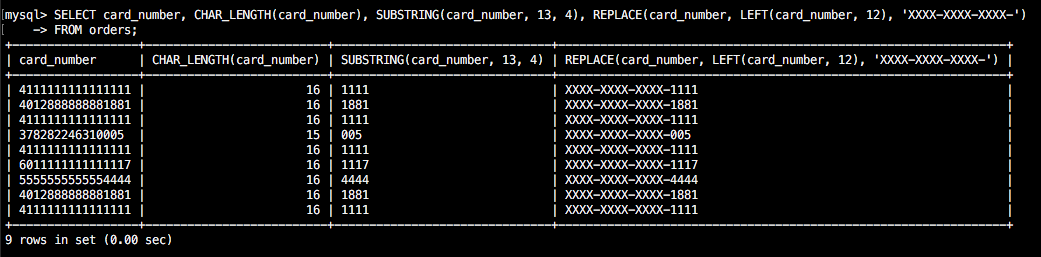
The card\_number column

The length of the card\_number column

The last four digits of the card\_number column

When you get that working right, add the columns that follow to the result set. This is more difficult because these columns require the use of functions within functions.

A column that displays the last four digits of the card\_number column in this format: XXXX-XXXX-XXXX-1234. In other words, use Xs for the first 12 digits of the card number and actual numbers for the last four digits of the number.

****

1. (**MGS** – 7 points) Write a SELECT statement that returns these columns from the Orders table:

The order\_id column

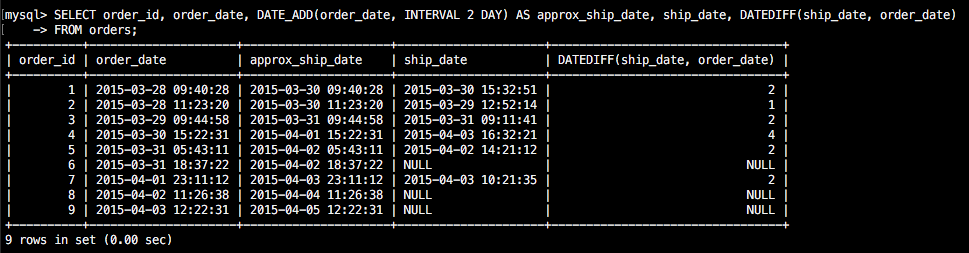
The order\_date column

A column named approx\_ship\_date that’s calculated by adding 2 days to the order\_date column

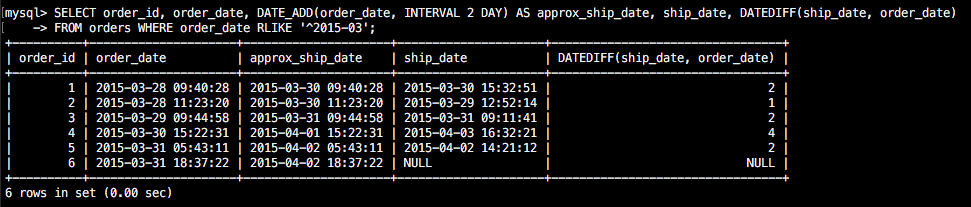
The ship\_date column

A column named days\_to\_ship that shows the number of days between the order date and the ship date

When you have this working, add a WHERE clause that retrieves just the orders for May 2015.

****

**There weren’t any orders from May 2015, so I assumed you meant March.**

****

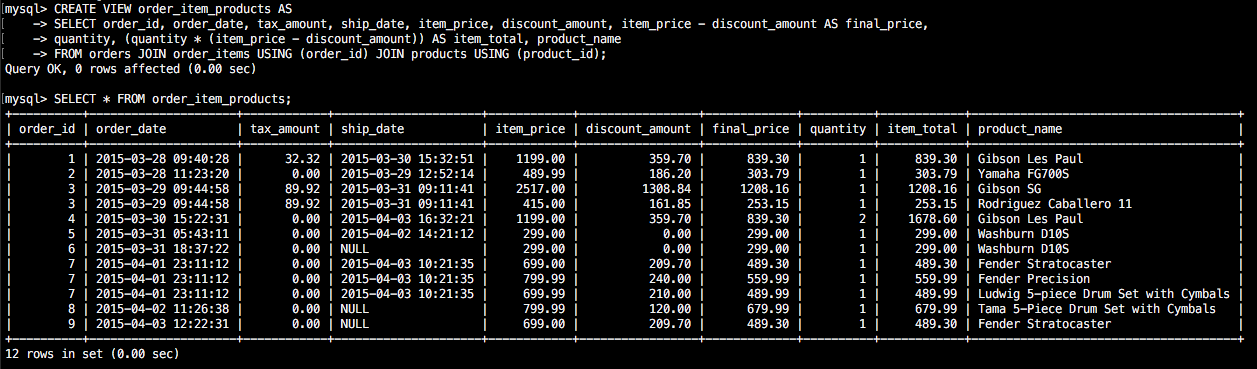
1. (**mgs\_*your userid*** *–* 8 points) Create a view named order\_item\_products that returns columns from the Orders, Order\_Items, and Products tables.

This view should return these columns from the Orders table: order\_id, order\_date, tax\_amount, and ship\_date.

This view should return these columns from the Order\_Items table: item\_price, discount\_amount, final\_price (the discount amount subtracted from the item price), quantity, and item\_total (the calculated total for the item).

This view should return the product\_name column from the Products table.

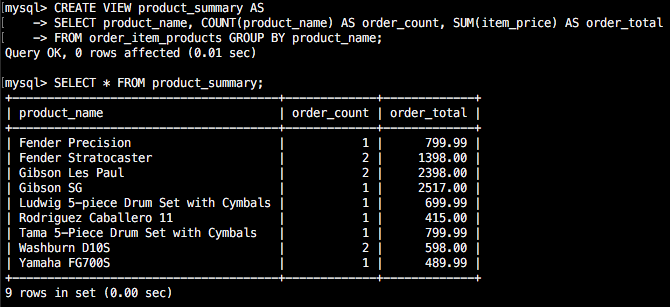
Show SQL for creating VIEW and then also a SELECT query and the output that lists all rows and all columns from the order\_item\_products view.



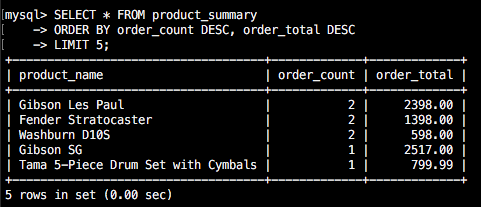
1. (**mgs\_*your userid*** *–* 7 points) Create a view named product\_summary that uses the view you created in exercise 9. This view should return summary information about each product.

Each row should include product\_name, order\_count (the number of times the product has been ordered) and order\_total (the total sales for the product).

Show SQL for creating VIEW and then also a SELECT query and the output that lists all rows and all columns from the product\_summary view.



1. (**mgs\_*your userid*** *–* 5 points) Write a SELECT statement that uses the view that you created in exercise 10 to get total sales for the five best selling products.

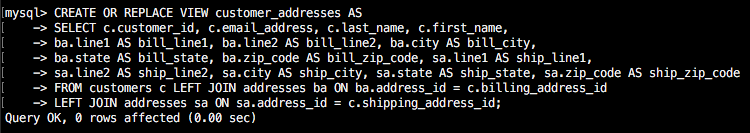


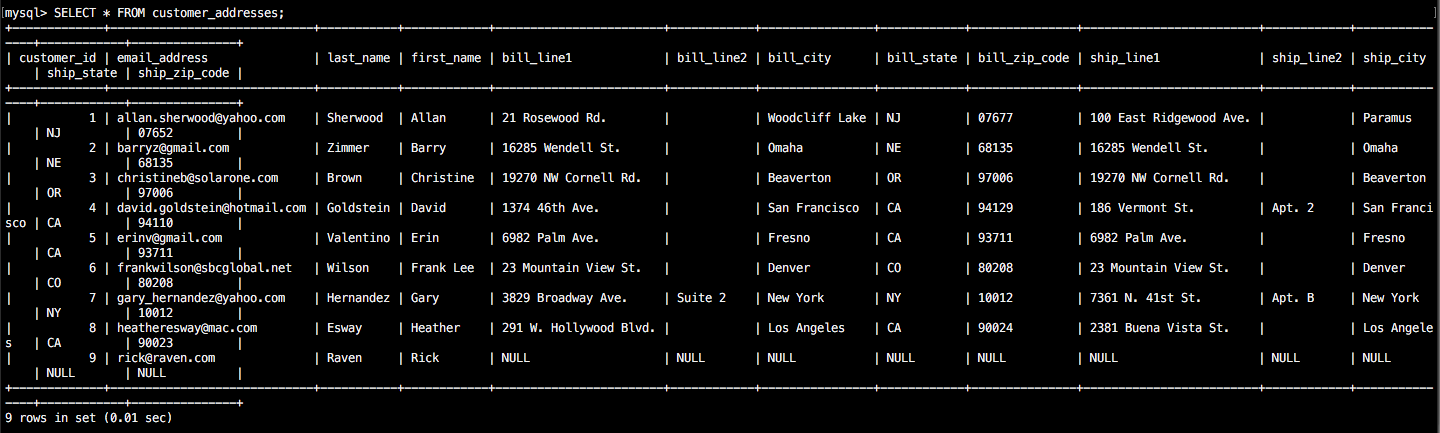
1. (**mgs\_*your userid*** *–* 8 points) Create a view named customer\_addresses that shows the shipping and billing addresses for each customer.

This view should return these columns from the Customers table: customer\_id, email\_address, last\_name and first\_name.

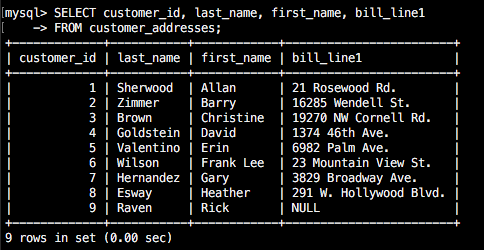
This view should return these columns from the Addresses table: bill\_line1, bill\_line2, bill\_city, bill\_state, bill\_zip, ship\_line1, ship\_line2, ship\_city, ship\_state, and ship\_zip.

The rows in this view should be sorted by the last\_name and then first\_name columns.





1. (**mgs\_*your userid*** *–* 5 points) Write a SELECT statement that returns these columns from the customer\_addresses view that you created in exercise 1: customer\_id, last\_name, first\_name, bill\_line1.

****

1. (**mgs\_*your userid*** *–* 6 points) Write an UPDATE statement that updates the Customers table using the customer\_addresses view you created in exercise 1. Set the first line of the shipping address to “1990 Westwood Blvd.” for the customer with an ID of 8. Then rerun SELECT statement from question 15 and show the results here.

