**Production Exam 2: Chapters 3, 4, 6   
106 points available – point values indicated**

Total Points per section as shown.

Please note sections D, and E, **you may omit one from each section.**

Section A = 8 points

Section B = 20 points

Section C = 12 points

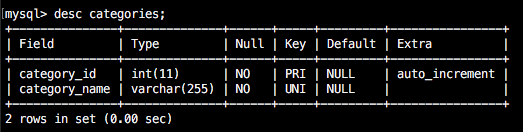
Section D = 36 points (choose 6)

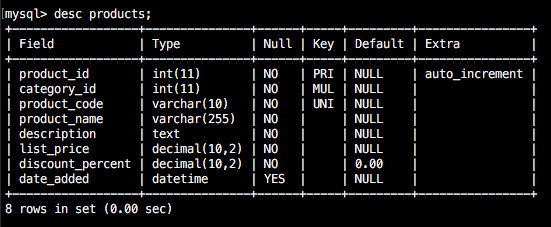
Section E = 30 points (choose 6)

Use cut and paste from your putty window to show SQL and results, including the number of rows returned from your query for full point value. All work to be done on dba120.abtech.edu using the MGS database.

**Section A: Retreive data from a single table** (2 x 4 ea = 8 pts)

1. a. Use the appropriate MySQL command to find the column definitions for the categories and the products tables …paste them here.



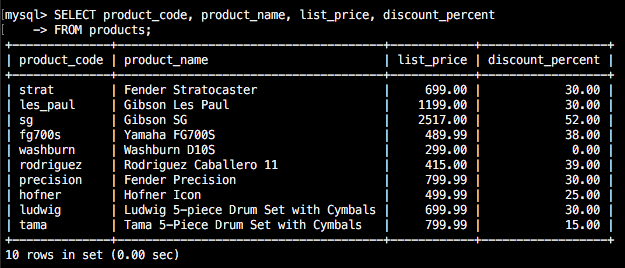


b. If you were to join these two tables, what field(s) would you use in your join condition?

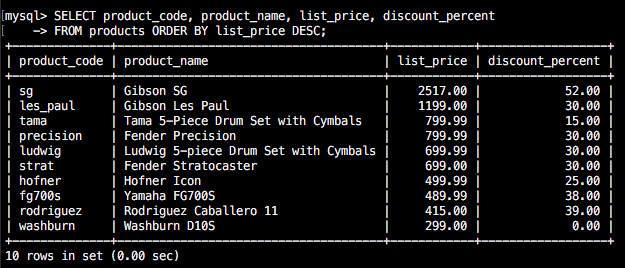
category\_id

**Section B: Retreive data from a single table** (5 x 4 ea = 20 pts)

1. Write a SELECT statement that returns four columns from the Products table: product\_code, product\_name, list\_price, and discount\_percent. Then, run this statement to make sure it works correctly.



Add an ORDER BY clause to this statement that sorts the result set by list price in descending sequence. Then, run this statement again to make sure it works correctly. This is a good way to build and test a statement, one clause at a time.

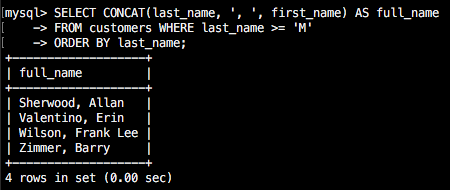


1. Write a SELECT statement that returns one column from the Customers table named full\_name that joins the last\_name and first\_name columns.

Format this column with the last name, a comma, a space, and the first name like this: Doe, John

Sort the result set by last name in ascending sequence.

Return only the customers whose last name begins with letters from M to Z.



1. Write a SELECT statement that returns these columns from the Products table:

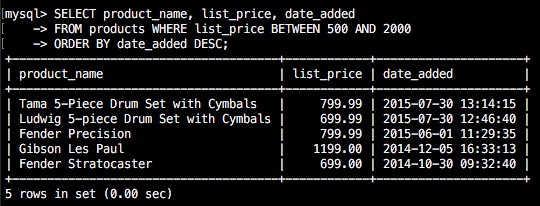
product\_name The product\_name column

list\_price The list\_price column

date\_added The date\_added column

Return only the rows with a list price that’s greater than 500 and less than 2000.

Sort the result set in descending sequence by the date\_added column.



1. Write a SELECT statement that returns these column names and data from the Products table:

product\_name The product\_name column

list\_price The list\_price column

discount\_percent The discount\_percent column

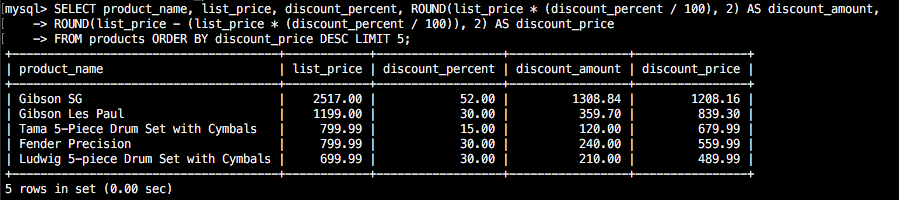
discount\_amount A column that’s calculated from the previous two columns

discount\_price A column that’s calculated from the previous three columns

Round the discount\_amount and discount\_price columns to 2 decimal places.

Sort the result set by discount price in descending sequence.

Use the LIMIT clause so the result set contains only the first 5 rows.



1. Write a SELECT statement that returns these column names and data from the Order\_Items table:

item\_id The item\_id column

item\_price The item\_price column

discount\_amount The discount\_amount column

quantity The quantity column

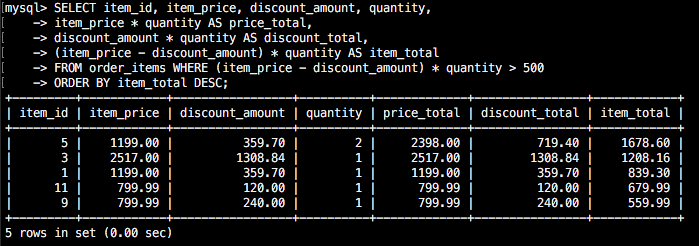
price\_total A column that’s calculated by multiplying the item price by the quantity

discount\_total A column that’s calculated by multiplying the discount amount by the quantity

item\_total A column that’s calculated by subtracting the discount amount from the item price and then multiplying by the quantity

Only return rows where the item\_total is greater than 500.

Sort the result set by item total in descending sequence.



Section C: Work with nulls and test expressions (3 x 4 ea = 12 pts)

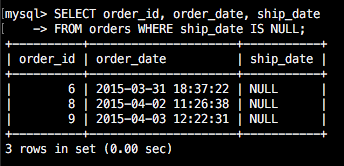
1. Write a SELECT statement that returns these columns from the Orders table:

order\_id The order\_id column

order\_date The order\_date column

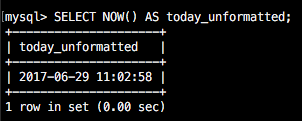
ship\_date The ship\_date column

Return only the rows where the ship\_date column contains a null value.



1. Write a SELECT statement without a FROM clause that uses the NOW function to create a row with these columns:

today\_unformatted The NOW function unformatted



9. Write a SELECT statement without a FROM clause that creates a row with these

columns:

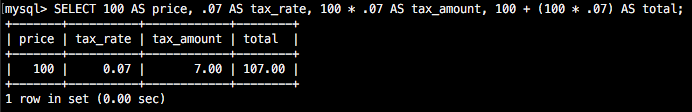
price 100 (dollars)

tax\_rate .07 (7 percent)

tax\_amount The price multiplied by the tax

total The price plus the tax

To calculate the fourth column, add the expressions you used for the first and third columns.

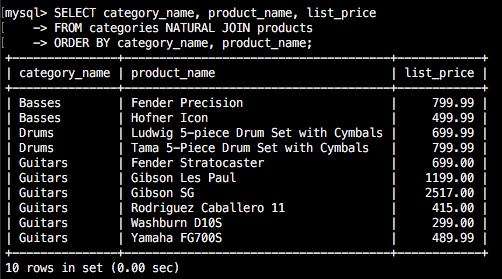


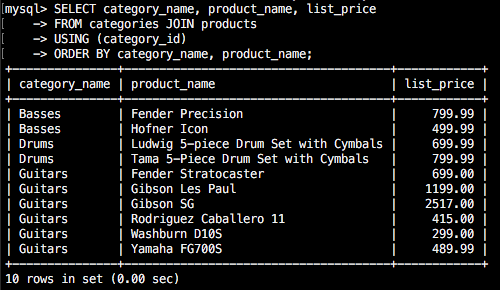
Section D: Retrieve data from two or more tables (*choose* 6 x 6 ea = 36 pts)

10. Write a SELECT statement that joins the Categories table to the Products table and returns these columns: category\_name, product\_name, list\_price.

Sort the result set by category\_name and then by product\_name in ascending sequence.

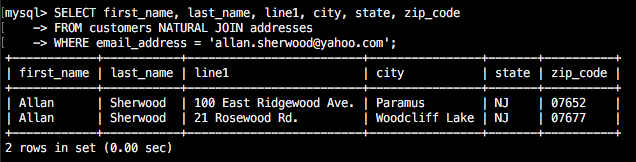
There are several ways to code this using different syntax ( 4 come to mind immediately), please show ***two*** different ways.





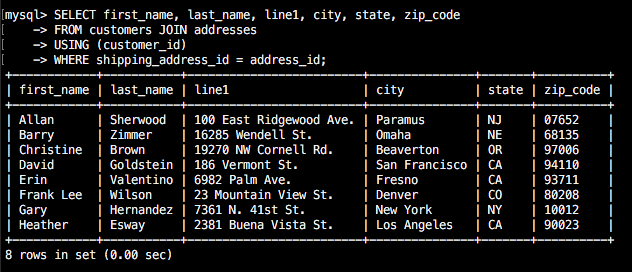
1. Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first\_name, last\_name, line1, city, state, zip\_code.

Return one row for each address for the customer with an email address of allan.sherwood@yahoo.com.



1. Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first\_name, last\_name, line1, city, state, zip\_code.

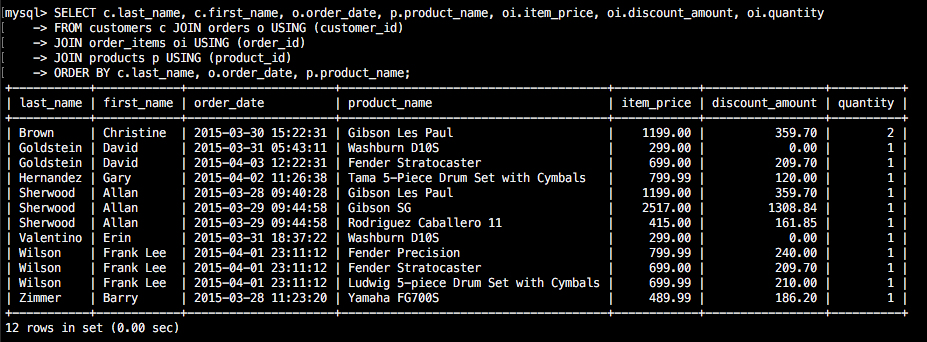
Return one row for each customer, but only return addresses that are the shipping address for a customer.



1. Write a SELECT statement that joins the Customers, Orders, Order\_Items, and Products tables. This statement should return these columns: last\_name, first\_name, order\_date, product\_name, item\_price, discount\_amount, and quantity.

Use aliases for the tables.

Sort the final result set by last\_name, order\_date, and product\_name.



1. Write a SELECT statement that returns the product\_name and list\_price columns from the Products table.

Return one row for each product that has the same list price as another product.   
*Hint: Use a self-join to check that the product\_id columns aren’t equal but the list\_price columns are equal.*

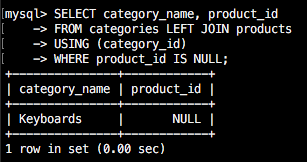
Sort the result set by product\_name.

1. Write a SELECT statement that returns these two columns:

category\_name The category\_name column from the Categories table

product\_id The product\_id column from the Products table

Return one row for each category that has never been used. *Hint: Use a left or right join and only return only the unmatched rows.*



1. Use the UNION operator to generate a result set consisting of three columns from the Orders table:

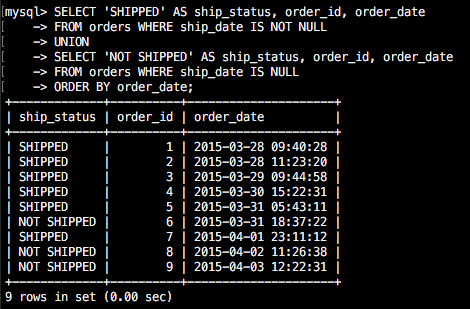
ship\_status A calculated column that contains a value of SHIPPED or NOT SHIPPED

order\_id The order\_id column

order\_date The order\_date column

If the order has a value in the ship\_date column, the ship\_status column should contain a value of SHIPPED. Otherwise, it should contain a value of NOT SHIPPED.

Sort the final result set by order\_date. *Hint: Write two separate statements that you can test and then union them together once they produce the desired result.*

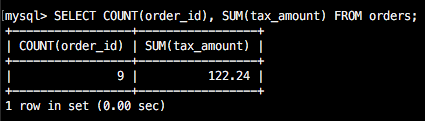


**Section E: Summary queries** (*choose* 6 x 5 ea = 30 pts)

1. Write a SELECT statement that returns these columns:

The count of the number of orders in the Orders table

The sum of the tax\_amount columns in the Orders table



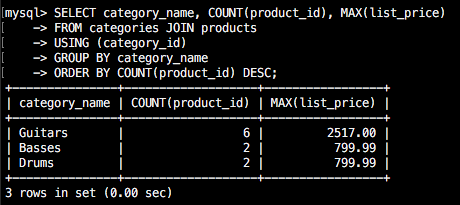
1. Write a SELECT statement that returns one row for each category that has products with these columns:

The category\_name column from the Categories table

The count of the products in the Products table

The list price of the most expensive product in the Products table

Sort the result set so the category with the most products appears first.



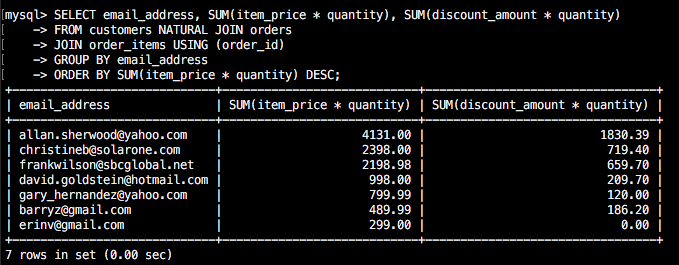
1. Write a SELECT statement that returns one row for each customer that has orders with these columns:

The email\_address column from the Customers table

The sum of the item price in the Order\_Items table multiplied by the quantity in the Order\_Items table

The sum of the discount amount column in the Order\_Items table multiplied by the quantity in the Order\_Items table

Sort the result set in descending sequence by the item price total for each customer.



1. Write a SELECT statement that returns one row for each customer that has orders with these columns:

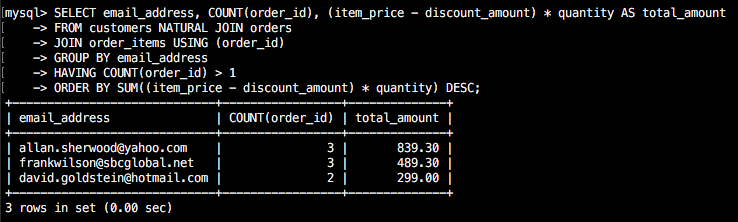
The email\_address from the Customers table

A count of the number of orders

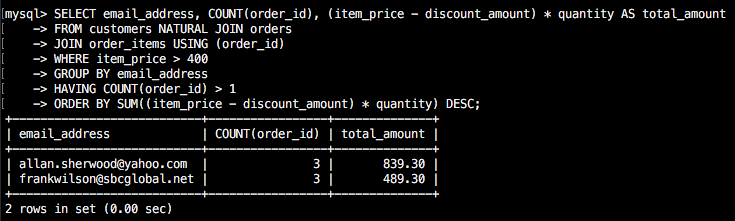
The total amount for each order (*Hint: First, subtract the discount amount from the price. Then, multiply by the quantity.*)

Return only those rows where the customer has more than 1 order.

Sort the result set in descending sequence by the sum of the line item amounts.



1. Modify the solution to exercise 20 so it only counts and totals line items that have an item\_price value that’s greater than 400.

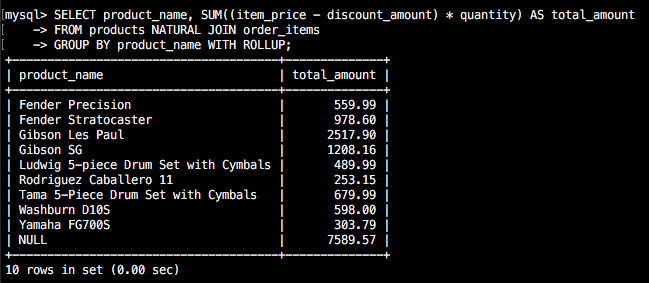


1. Write a SELECT statement that answers this question: What is the total amount ordered for each product? Return these columns:

The product name from the Products table

The total amount for each product in the Order\_Items (*Hint: You can calculate the total amount by subtracting the discount amount from the item price and then multiplying it by the quantity*)

Use the WITH ROLLUP operator to include a row that gives the grand total.



1. Write a SELECT statement that answers this question: Which customers have ordered more than one product? Return these columns:

The email address from the Customers table

The count of distinct products from the customer’s orders

