# Author: Jeffrey Nicholls

# Title: Lab 4

# Date: February 9th, 2021

# Class: STY 1211

# Making sure the proper database is used

use my\_guitar\_shop;

# Question 1:

# Write a SELECT statement that joins the Categories table to the Products table

# and returns these columns: category\_name, product\_name, list\_price.

select category\_name, product\_name, list\_price

from products as p join categories as c on p.category\_id = c.category\_id

order by category\_name, product\_name;

# Question 2:

# 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.

select first\_name, last\_name, line1, city, state, zip\_code

from addresses as a join customers as c on a.customer\_id = c.customer\_id

where c.email\_address = "allan.sherwood@yahoo.com";

# Question 3:

# 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.

select first\_name, last\_name, line1, city, state, zip\_code

from addresses as a

join customers as c on a.customer\_id = c.customer\_id

where a.address\_id = c.shipping\_address\_id;

# Question 4:

# 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.

select last\_name, first\_name, order\_date, product\_name, item\_price, discount\_amount, quantity

from customers as c

join orders as o on c.customer\_id = o.customer\_id

join order\_items as o\_i on o.order\_id = o\_i.order\_id

join products as p on o\_i.product\_id = p.product\_id

order by last\_name, order\_date, product\_name;

# Question 5:

# 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.

select p1.product\_name, p1.list\_price

from products as p1

join products as p2 on p1.list\_price = p2.list\_price

where p1.product\_id != p2.product\_id;

# Question 6:

# 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 an outer join and only return rows where the product\_id column contains a null value.

select c.category\_name, p.product\_id

from categories as c left join products as p on c.category\_id = p.category\_id

where p.product\_id is null;

# Question 7:

# 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.

select "Shipped" as ship\_status, order\_id, order\_date

from orders

where ship\_date is not null

union

select "Not Shipped" as ship\_status, order\_id, order\_date

from orders

where ship\_date is null

order by order\_date;

EER Diagram Used:

