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

   instrument_name, list_price, date_added

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

   Sort the result set by the date_added column in descending sequence.

   This should return 5 rows.

2. Write a SELECT statement that returns these column names and data from the order_instruments 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 the item_total column in descending sequence.

3. 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
today_formatted	The NOW function in this format: DD-Mon-YYYY

   This displays a number for the day, an abbreviation for the month, and a four-digit year.

   Try displaying today's date in different formats like : '01/2020' —- mm/yyyy

   30/01/2020 ———Dd /mm/ yy

   30th January 2020. — doth month yyyy

4. Write a SELECT statement that joins the Musicians 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 musician with an email address of david.goldstein@hotmail.com

5. Write a SELECT statement that joins the Musicians table to the Addresses table and returns these columns: first_name, last_name, line1, city, state, zip_code.

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

6. Write a SELECT statement that joins the Musicians, Orders, Order_instruments, and Instruments tables. This statement should return these columns: last_name, first_name, order_date, instrument_name, item_price, discount_amount, and quantity.

   Use aliases for the tables.

   Sort the final result set by the last_name, order_date, and instrument_name columns.

7. Write a SELECT statement that returns the instrument_name and list_price columns from the Instruments table.

   Return one row for each instrument that has the same list price as another instrument.
   Hint: Use a self-join to check that the instrument_id columns aren't equal but the list_price columns are equal.

   Sort the result set by the instrument_name column.

8. 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 the order_date column.