Structured Query Language (SQL) Checkpoint

This checkpoint is designed to test your understanding of writing queries in the Structured Querying Language (SQL).

Specifically, this checkpoint will cover:

- Reading an Entity Relationship Diagram
- Writing a query to return specific columns from a SQL database
- Writing a query to filter the rows from a SQL database
- Sorting observations using SQL
- Joining tables using SQL

Data Understanding

In this repository under the file path `Northwind.sqlite` there is a SQLite database file containing information about the fictional trading company "Northwind Traders".

The tables of interest for this checkpoint will be:

`Product`: A table containing information about products sold by Northwind Traders.

`Order`: A table containing high level information about an order submitted to Northwind Traders.

`Shipper`: A table containing information about the shipping companies Northwind Traders employ to handle the shipping of their products.

Requirements

1. Select an entire table.

2. Select all columns. Filter the rows.

3. Select a single column. Filter the rows using two conditions.

4. Sort in descending order. Return the first five rows.

5. Join two tables. Filter the rows.

Setup

This checkpoint will test the resulting data each of your SQL queries generate. For each requirement, your query should be written as a string, and assigned to the requested variable name. The tests do not inspect the casing or formatting of your SQL query.

In the cell below we import relevant libraries.

```
# Run this cell without changes
import sqlite3
import pandas as pd
```

In the cell below we...

- Open up a connection to the SQLite database and store the connection in a variable called `conn`
- Initialize a SQLite cursor object with the variable name `cursor`.

```
# Run this cell without changes
northwind_path = 'Northwind.sqlite'

# Open up a connection
conn = sqlite3.connect(northwind_path)
# Initialize a cursor
cursor = conn.cursor()
```

Below is an Entity Relationship Diagram for the Northwind Database

> The text is quite small in the below image. Here is a link to the raw image file, where the text is slightly larger.

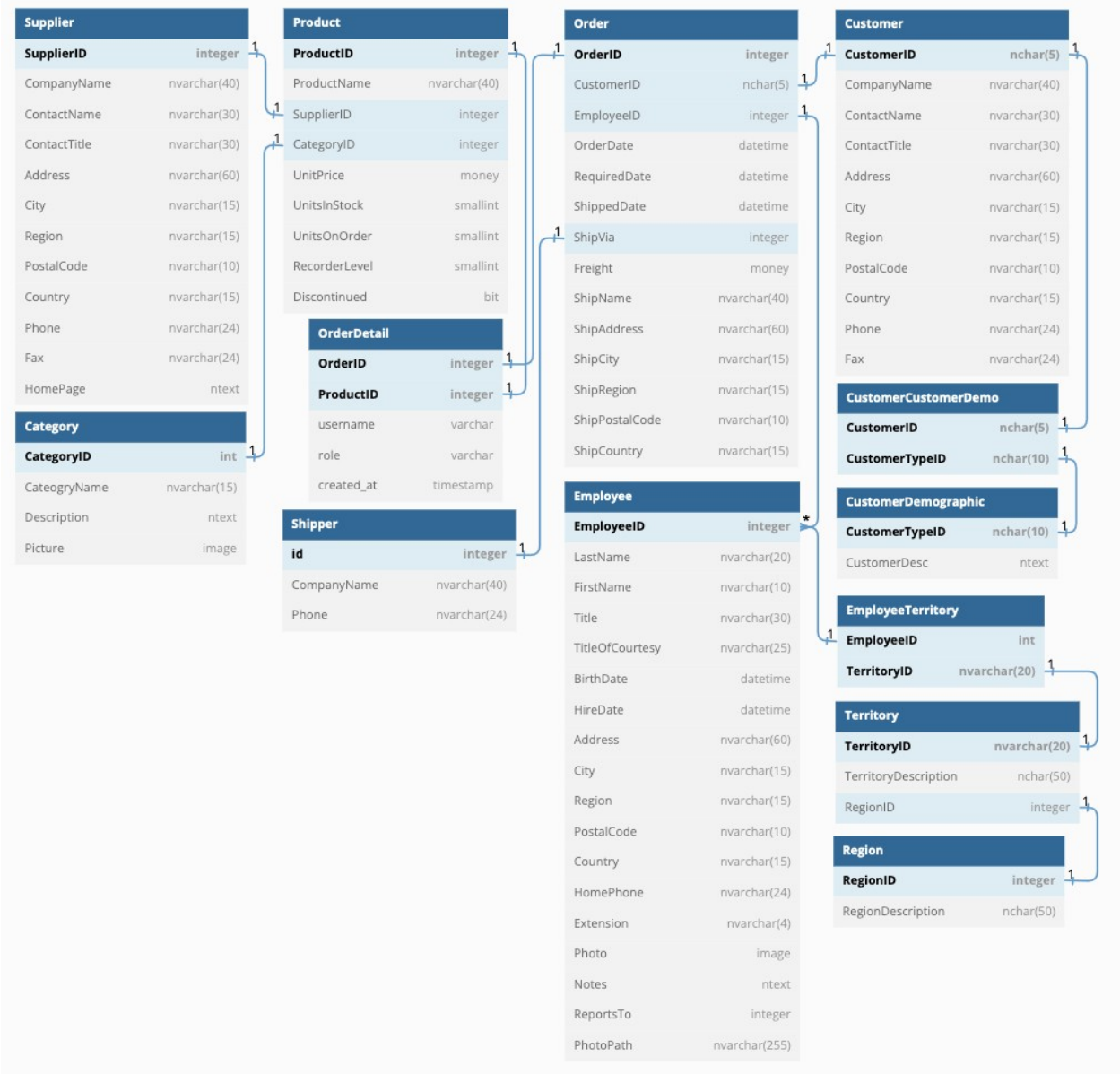

Table Names

Below, we use `pd.read_sql` to output the table names in the SQLite database. When writing your queries, you should use the table names as listed below.

```
# Run this cell without changes
table_name_query = """SELECT name
                      AS 'Table Names'
                      FROM sqlite_master
                      WHERE type='table';"""

pd.read_sql(table_name_query, conn)
```

```
              Table Names
0                Employee
1                Category
2                Customer
3                 Shipper
4                Supplier
5                   Order
6                 Product
7             OrderDetail
8   CustomerCustomerDemo
9    CustomerDemographic
10                Region
11             Territory
12     EmployeeTerritory
```

1. Select an entire table.

In a string variable named `first_query`, write a SQL query that returns all rows and columns from the `product` table.

When passed into `pd.read_sql` this query should return a dataframe with a head that looks like this:

	Id	ProductName	SupplierId	CategoryId	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
0	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0
1	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0
2	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0
3	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0
4	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1

```
# CodeGrade step1
# Replace None with appropriate code

first_query = """SELECT * FROM Product;"""

# Use the line below to check your query's output
pd.read_sql(first_query, conn).head()
```

```
   Id                   ProductName  SupplierId  CategoryId  \
0   1                          Chai           1           1
1   2                         Chang           1           1
2   3                Aniseed Syrup           1           2
3   4  Chef Anton's Cajun Seasoning           2           2
4   5        Chef Anton's Gumbo Mix           2           2

       QuantityPerUnit  UnitPrice  UnitsInStock  UnitsOnOrder
ReorderLevel  \
0   10 boxes x 20 bags      18.00            39             0
10
1   24 - 12 oz bottles      19.00            17            40
25
2  12 - 550 ml bottles      10.00            13            70
25
3       48 - 6 oz jars      22.00            53             0
0
4             36 boxes      21.35             0             0
0

   Discontinued
0             0
1             0
2             0
3             0
4             1
```

```
# first_query should be a string
assert type(first_query) == str

# first_query should be a SQL query
first_query_df = pd.read_sql(first_query, conn)
```

2. Select all columns. Filter the rows.

In a string variable named `second_query`, write a SQL query that returns all columns and rows from the `Product` table where `discontinued` has a value of `1`.

When passed into `pd.read_sql` the query's resulting dataframe should look like this:

	Id	ProductName	SupplierId	CategoryId	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
0	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1
1	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	1
2	17	Alice Mutton	7	6	20 - 1 kg tins	39.00	0	0	0	1

	Id	ProductName	SupplierId	CategoryId	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
3	24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.50	20	0	0	1
4	28	Rössle Sauerkraut	12	7	25 - 825 g cans	45.60	26	0	0	1
5	29	Thüringer Rostbratwurst	12	6	50 bags x 30 sausgs.	123.79	0	0	0	1
6	42	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14.00	26	0	0	1
7	53	Perth Pasties	24	6	48 pieces	32.80	0	0	0	1

```
# CodeGrade step2
# Replace None with appropriate code

second_query = """
    SELECT * FROM Product
    WHERE
        Discontinued = 1;
"""

# Use the line below to check your query's output
pd.read_sql(second_query, conn)
```

```
   Id                     ProductName  SupplierId  CategoryId  \
0   5         Chef Anton's Gumbo Mix           2           2
1   9                Mishi Kobe Niku           4           6
2  17                   Alice Mutton           7           6
3  24             Guaraná Fantástica          10           1
4  28              Rössle Sauerkraut          12           7
5  29        Thüringer Rostbratwurst          12           6
6  42  Singaporean Hokkien Fried Mee          20           5
7  53                  Perth Pasties          24           6

        QuantityPerUnit  UnitPrice  UnitsInStock  UnitsOnOrder
ReorderLevel  \
0              36 boxes      21.35             0             0
0
1      18 - 500 g pkgs.      97.00            29             0
0
2        20 - 1 kg tins      39.00             0             0
0
3      12 - 355 ml cans       4.50            20             0
0
4       25 - 825 g cans      45.60            26             0
```

```
0
5  50 bags x 30 sausgs.     123.79              0              0
0
6       32 - 1 kg pkgs.      14.00             26              0
0
7             48 pieces      32.80              0              0
0

   Discontinued
0             1
1             1
2             1
3             1
4             1
5             1
6             1
7             1
```

```
# second_query should be a string
assert type(second_query) == str

# second_query should be a SQL query
second_query_df = pd.read_sql(second_query, conn)
```

3. Select a single column. Filter the rows using two conditions.

In a string variable named `third_query`, write a SQL query that returns the name of a product if the product is not in stock and is not discontinued.

When passed into `pd.read_sql` the query's resulting dataframe should look like this:

	ProductName
0	Gorgonzola Telino

```
q = 'select Discontinued from Product;'
pd.read_sql(q, conn).Discontinued.unique()
```

```
array([0, 1], dtype=int64)
```

```
# CodeGrade step3
# Replace None with appropriate code

third_query = """
    SELECT ProductName FROM Product
    WHERE
        UnitsInStock = 0
        AND
        Discontinued = 0;
```

```
"""

# Use the line below to check your query's output
pd.read_sql(third_query, conn)
```

```
         ProductName
0  Gorgonzola Telino
```

```
# third_query should be a string
assert type(third_query) == str

# third_query should be a SQL query
third_query_df = pd.read_sql(third_query, conn)
```

4. Sort in descending order. Return the first five rows.

In a string variable named `fourth_query`, write a SQL query that returns the product name and unit price.

- Order by unit price in descending order.
- Use a SQL command so only the first five rows are queried.

When passed into `pd.read_sql`, this query's resulting dataframe should look like this:

	ProductName	UnitPrice
0	Côte de Blaye	263.50
1	Thüringer Rostbratwurst	123.79
2	Mishi Kobe Niku	97.00
3	Sir Rodney's Marmalade	81.00
4	Carnarvon Tigers	62.50

```
# CodeGrade step4
# Replace None with appropriate code

fourth_query = """
    SELECT
        ProductName, UnitPrice
    FROM Product
    ORDER BY
        UnitPrice DESC
    LIMIT 5;
"""

# Use the line below to check your query's output
pd.read_sql(fourth_query, conn)
```

```
               ProductName  UnitPrice
0            Côte de Blaye     263.50
1  Thüringer Rostbratwurst     123.79
```

```
2          Mishi Kobe Niku      97.00
3   Sir Rodney's Marmalade      81.00
4         Carnarvon Tigers      62.50

# fourth_query should be a string
assert type(fourth_query) == str

# fourth_query should be a SQL query
fourth_query_df = pd.read_sql(fourth_query, conn)
```

5. Join two tables. Filter the rows.

In a string variable named `fifth_query`, write a SQL query that returns the name and phone number for shippers who have charged more than $1000 for shipping cost.

Hint:

- "Freight", in the Order table, represents shipping cost.
- For this question, the word "order" is both a table name AND a SQL command. To help SQL understand that you are referencing the table `order` in the database, you will need to wrap the word `order` in quotation marks.

When passed into `pd.read_sql`, the query's resulting dataframe should look like this:

	CompanyName	Phone
0	Federal Shipping	(503) 555-9931

```
# delete
q = """
SELECT * FROM 'Order'
LIMIT 1;
"""
pd.read_sql(q, conn)

      Id CustomerId  EmployeeId   OrderDate RequiredDate ShippedDate
ShipVia  \
0  10248      VINET           5  2012-07-04   2012-08-01  2012-07-16
3

   Freight                   ShipName         ShipAddress ShipCity  \
0    32.38  Vins et alcools Chevalier  59 rue de l'Abbaye    Reims

       ShipRegion ShipPostalCode ShipCountry
0  Western Europe          51100      France

# CodeGrade step5
# Replace None with appropriate code

fifth_query = """
    SELECT
```

```
        Shipper.CompanyName, Shipper.Phone
    FROM Shipper
    JOIN 'Order'
        ON Shipper.Id = 'Order'.ShipVia
    WHERE
        'Order'.Freight > 1000;
"""

# Use the line below to check your query's output
pd.read_sql(fifth_query, conn)
```

```
        CompanyName            Phone
0  Federal Shipping  (503) 555-9931
```

```
# fifth_query should be a string
assert type(fifth_query) == str

# fifth_query should be a SQL query
fifth_query_df = pd.read_sql(fifth_query, conn)
```