Guided Lab - 304.6.1 - Joins and Clauses - Classicmodels Database

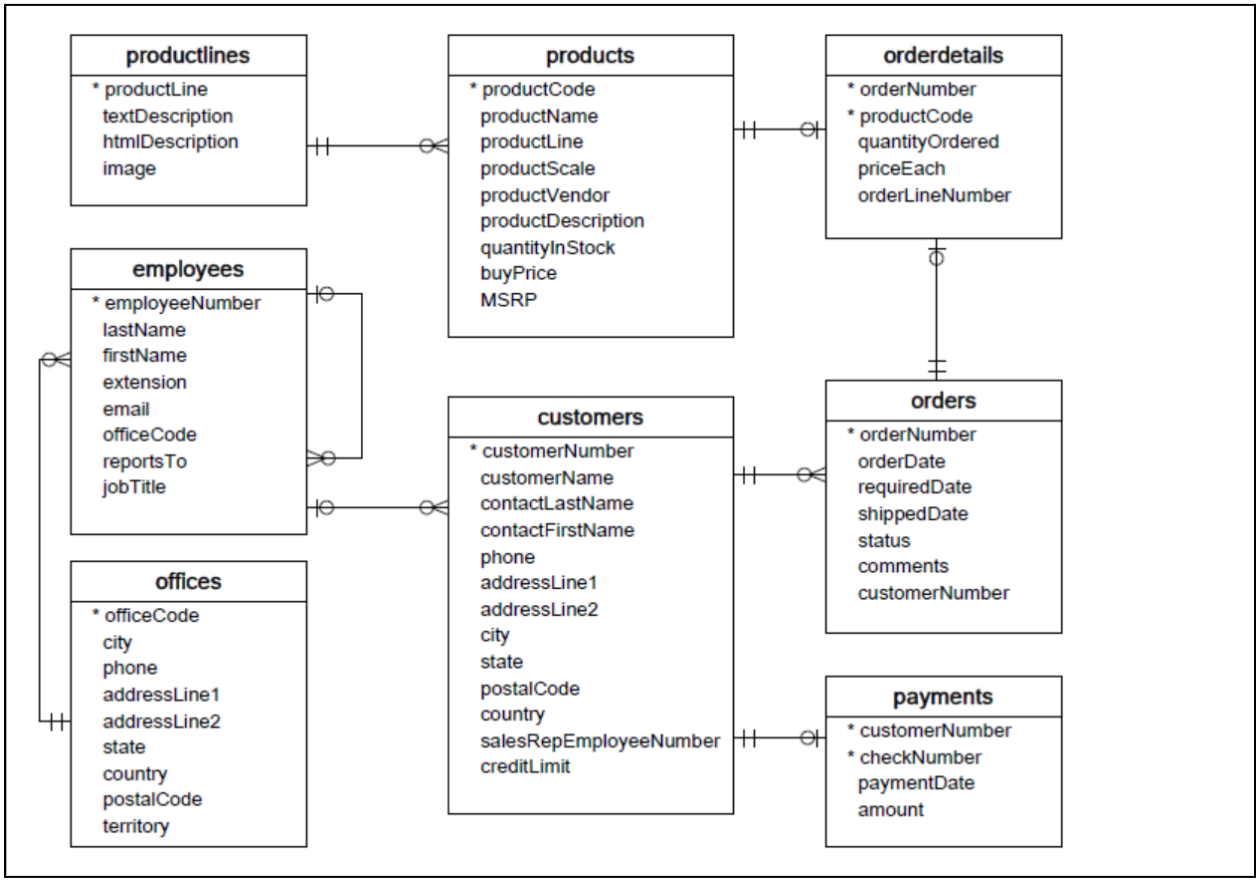
Lab Overview:

In this lab, we will demonstrate and utilize SQL join predicates, SQL clauses, and aggregate functions.

Prerequisites:

For this lab, you must have a ***"classicmodels"*** database. If you do not have a '**classicmodels**' database setup, **click here to download the database script file.**

The database schema is:



If you run into this error with MySQL:

Expression #1 of the SELECT list is not in the GROUP BY clause and contains nonaggregated column 'db.table.col,' which is not functionally dependent on columns in the GROUP BY clause; this is incompatible with sql_mode=only_full_group_by.

It is likely because you have the `ONLY_FULL_GROUP_BY` function enabled. To fix this, you have to disable it.

Run this command:

```
SET sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));
```

1. ### Problem Statement One:

Write a query to display each customer's name (as "Customer Name"), along with the name of the employee who is responsible for that customer's orders. The employee name should be in a single "Sales Rep" column, formatted as "lastName, firstName." The output should be sorted alphabetically by customer name.

Solution: **Run the below query on MySQL:**

```
select c.customerName as 'Customer Name',
concat(e.lastName, ', ', e.firstName) as 'Sales Rep'
from customers c JOIN employees e
on c.salesRepEmployeeNumber = e.employeeNumber
order by c.customerName asc;
```

2. **Problem Statement two:**

   To determine which products are the most popular with our customers. For each product, list the total quantity ordered, along with the total sale generated (total quantity ordered * priceEach) for that product. The column headers should be “Product Name,” “Total # Ordered,” and “Total Sale.” List the products by “Total Sale” descending.

Solution: Run the below query on MySQL:

```
select p.productName as 'Product Name', sum(od.quantityOrdered) as 'Total #
Ordered', sum(od.quantityOrdered * od.priceEach) as 'Total Sale'
from products p LEFT JOIN orderdetails od
on p.productCode=od.productCode
group by p.productName, p.buyPrice
order by 3 desc
```

3. **Problem Statement three:**

   Write a query that lists order status and the number of orders with that status. Column headers should be “Order Status” and “Total Orders.” Sort alphabetically by status.

Solution: Run below query on MySQL:

```
select status as 'Order Status', count(status) as 'Total Orders'
from orders
group by status
order by status;
```

4. **Problem Statement four:**

   Write a query to list, for each product line, the total number of products sold from that product line. The first column should be “Product Line” and the second should be “total Sold.” Order by the second column, descending.

Solution: **Run below query on MySQL:**

```
select
     pl.productLine as 'Product Line',
     count(od.productCode) as 'total Sold'
from productLines pl join products p
on pl.productLine=p.productLine
 JOIN orderdetails od on p.productCode=od.productCode
group by pl.productLine
order by 2 desc;
```

5. **Problem Statement five:**

Your product team is requesting data to help them create a bar chart of monthly sales made since the company's inception. Write a query to output the month (January, February, etc.), 4-digit year, and total sales for each month. The first column should be labeled 'Month,' the second column should be labeled 'Year,' and the third column should be labeled 'Payments Received.' Values in the third column should be formatted as numbers with two decimals (e.g., 694,292.68).

Solution: Run below query on MySQL:

```
select
   monthname(paymentDate) AS Month,
   year(paymentDate) AS Year,
   format(sum(amount), 2) AS 'Payments Received'
from payments
group by year(paymentDate), monthname(paymentDate)
order by paymentDate;
```

6. **Problem statement five:**

Write a query to display the Name, Product Line, Scale, and Vendor of all of the Car products — both classic and vintage. The output should display all vintage cars first (sorted alphabetically by name), and all classic cars last (also sorted alphabetically by name).

Solution: Run below query on MySQL

```
SELECT p.productName Name, p.productLine AS `Product Line`,
p.productScale AS `Scale`, p.productVendor AS `Vendor` FROM
productlines l NATURAL JOIN products p
WHERE l.productLine = "Classic Cars" OR l.productLine = "Vintage
Cars"
ORDER BY p.productLine DESC, p.productName ASC;
```

-- Sort order is vintage before classic, and then alphabetically.

Canvas submission Instructions: Please include the following deliverables in your submission -

- All queries should be written and submitted in a single SQL script file.
  - Example :**<your_name_labname>.sql**.
- Submit your SQL script file using the **Start Assignment** button in the top-right corner of the assignment page in Canvas.

CANVAS STAFF USE ONLY: Canvas Submission Guideline:

Instructions for Canvas Assignment Creation
Assignment Name: **GLAB - 304.6.1 - Joins and Clauses - classicmodels Database** **Points:** **100** **Assignment Group:** **Module 304 - Relational Databases and SQL - (Not Graded)** **Display Grade As:** **Complete/Incomplete** **Do not count this assignment towards the final grade:** **Checked** **Submission Types:** **Files Uploads** **Allowed Attempts:** **Unlimited** **Everything else is the default.**