SQL Practice (Part 1) - SOLUTIONS

Explore aggregations, grouping, and joins:

Example 1: Aggregate Sales per Product Line

Objective: Write a query that determines the aggregate number of products and the total sales revenue (derived from quantity and price) for each product line, with results ordered in descending fashion based on total sales.

```
Unset
SELECT
    productLine,
    COUNT(*) AS total_products,
    ROUND(SUM(orderdetails.quantityOrdered *
orderdetails.priceEach), 2) AS total_sales
FROM
    products
LEFT JOIN
    orderdetails ON products.productCode =
orderdetails.productCode
GROUP BY
    productLine
ORDER BY
    total_sales DESC;
```

Example 1 Results

	productLine	total_products	total_sales
▶	Classic Cars	1011	3853922.49
	Vintage Cars	657	1797559.63
	Motorcycles	359	1121426.12
	Trucks and Buses	308	1024113.57
	Planes	336	954637.54
	Ships	245	663998.34
	Trains	81	188532.92

Example 2: Employee Sales Performance Analysis

Objective: Write a query that calculates the aggregate number of orders and sales generated by each employee, with results sorted by total sales.

```
Unset
SELECT
    CONCAT(employees.firstName, ' ', employees.lastName) AS
employee_name,
    COUNT(orders.orderNumber) AS total_orders,
    ROUND(SUM(orderdetails.quantityOrdered *
orderdetails.priceEach), 2) AS total_sales
FROM
    employees
LEFT JOIN
    customers ON employees.employeeNumber =
customers.salesRepEmployeeNumber
LEFT JOIN
    orders ON customers.customerNumber = orders.customerNumber
LEFT JOIN
    orderdetails ON orders.orderNumber = orderdetails.orderNumber
GROUP BY
    employee_name
ORDER BY
    total_sales DESC;
```

Example 2 Results

employee_name	total_orders	total_sales
Gerard Hernandez	396	1258577.81
Leslie Jennings	331	1081530.54
Pamela Castillo	272	868220.55
Larry Bott	236	732096.79
Barry Jones	220	704853.91
George Vanauf	211	669377.05
Peter Marsh	185	584593.76
Loui Bondur	177	569485.75
Andy Fixter	185	562582.59
Steve Patterson	152	505875.42
Foon Yue Tseng	142	488212.67
Mami Nishi	137	457110.07
Martin Gerard	114	387477.47
Julie Firrelli	124	386663.20
Leslie Thompson	114	347533.03
Diane Murphy	0	NULL
Mary Patterson	0	NULL
Jeff Firrelli	0	NULL

Example 3: Customer Purchase Summary Report

Objective: Write a query that provides a detailed summary of customer purchasing activities, encompassing the count of orders, total expenditure, and temporal range of order placement, with results ordered by total expenditure.

```
Unset
SELECT
    customers.customerName,
    COUNT(orders.orderNumber) AS total_orders,
    ROUND(SUM(orderdetails.quantityOrdered *
orderdetails.priceEach), 2) AS total_spent,
    MIN(orders.orderDate) AS first_order_date,
    MAX(orders.orderDate) AS last_order_date
FROM
    customers
LEFT JOIN
    orders ON customers.customerNumber = orders.customerNumber
LEFT JOIN
    orderdetails ON orders.orderNumber = orderdetails.orderNumber
GROUP BY
    customers.customerName
ORDER BY
    total_spent DESC;
```

Example 3 Results

customerName	total_orders	total_spent	first_order_date	last_order_date
Euro+ Shopping Channel	259	820689.54	2003-01-31	2005-05-31
Mini Gifts Distributors Ltd.	180	591827.34	2003-03-26	2005-05-29
Australian Collectors, Co.	55	180585.07	2003-04-29	2004-11-29
Muscle Machine Inc	48	177913.95	2003-06-03	2004-12-01
La Rochelle Gifts	53	158573.12	2004-07-23	2005-05-31
Dragon Souveniers, Ltd.	43	156251.03	2003-04-16	2005-03-02
Down Under Souveniers, Inc	46	154622.08	2003-06-25	2005-04-08
Land of Toys Inc.	49	149085.15	2003-02-24	2004-11-15
AV Stores, Co.	51	148410.09	2003-03-18	2004-11-17
The Sharp Gifts Warehouse	40	143536.27	2004-05-11	2005-04-22
Salzburg Collectables	40	137480.07	2003-04-28	2005-05-17
Kelly's Gift Shop	48	137460.79	2003-07-07	2005-04-01
Anna's Decorations, Ltd	46	137034.22	2003-09-11	2005-03-09
Souveniers And Things Co.	46	133907.12	2003-07-16	2005-05-29
Corporate Gift Ideas Co.	41	132340.78	2003-10-10	2005-02-23
Saveley & Henriot, Co.	41	130305.35	2003-11-25	2004-03-02
Danish Wholesale Imports	36	129085.12	2003-02-11	2005-04-15
Rovelli Gifts	48	127529.69	2003-02-17	2004-11-12

Example 4: Order Magnitude by Status Category

Objective: Write a query that computes the count of orders, the average order value, and the maximum order value for each status category, with results sorted in descending order by average order value.

```
Unset
SELECT
    orders.status,
    COUNT(orders.orderNumber) AS total_orders,
    ROUND(AVG(orderdetails.quantityOrdered *
orderdetails.priceEach), 2) AS avg_order_value,
    MAX(orderdetails.quantityOrdered * orderdetails.priceEach) AS
max_order_value
FROM
    orders
LEFT JOIN
    orderdetails ON orders.orderNumber = orderdetails.orderNumber
GROUP BY
    orders.status
ORDER BY
    avg_order_value DESC;
```

Example 4 Results

	status	total_orders	avg_order_value	max_order_v
▶	Disputed	14	4368.48	9109.52
	On Hold	44	3853.99	10723.60
	In Process	41	3299.31	10072.00
	Shipped	2771	3199.24	11503.14
	Cancelled	79	3023.47	8316.34
	Resolved	47	2856.08	6043.94

Example 5: Order Magnitude by Status Category
Create a summary of sales for each product, including total sales,
minimum and maximum quantities sold,
the average quantity (rounded to two decimals),
and sort the results by total sales in descending order?

This SQL query aims to provide a summary of sales for each product. Here's a breakdown:

SQL Query Breakdown

```
Unset
SELECT
    products.productName AS 'Product Name',
    COUNT(orderdetails.quantityOrdered) AS 'Total Sales',
    MIN(orderdetails.quantityOrdered) AS 'Min Quantity Sold',
    ROUND(AVG(orderdetails.quantityOrdered), 2) AS 'Avg Quantity Sold',
    MAX(orderdetails.quantityOrdered) AS 'Max Quantity Sold'
FROM
    products
LEFT JOIN
    orderdetails ON products.productCode = orderdetails.productCode
GROUP BY
    products.productName
ORDER BY
    'Total Sales' DESC;
```

- **SELECT Clause:**
  - `products.productName AS 'Product Name'`: Selects the product name and aliases it as 'Product Name'.
  - `COUNT(orderdetails.quantityOrdered) AS 'Total Sales'`: Counts the number of orders for each product and aliases it as 'Total Sales'.
  - `MIN(orderdetails.quantityOrdered) AS 'Min Quantity Sold'`: Finds the minimum quantity sold for each product and aliases it as 'Min Quantity Sold'.
  - `ROUND(AVG(orderdetails.quantityOrdered), 2) AS 'Avg Quantity Sold'`: Calculates the average quantity sold for each product, rounds it to two decimal places, and aliases it as 'Avg Quantity Sold'.
  - `MAX(orderdetails.quantityOrdered) AS 'Max Quantity Sold'`: Finds the maximum quantity sold for each product and aliases it as 'Max Quantity Sold'.
- **FROM Clause:**
  - `products`: Specifies the 'products' table as the primary table.
- **LEFT JOIN Clause:**
  - `LEFT JOIN orderdetails ON products.productCode = orderdetails.productCode`: Joins the 'products' table with the 'orderdetails' table using the 'productCode' as

the joining key. A LEFT JOIN ensures that all products are included, even if they have no corresponding orders.

- **GROUP BY Clause:**
  - `GROUP BY products.productName`: Groups the results by product name, allowing for aggregate functions (COUNT, MIN, AVG, MAX) to be applied per product.
- **ORDER BY Clause:**
  - `ORDER BY 'Total Sales' DESC`: Orders the results in descending order based on the 'Total Sales', showing the products with the highest total sales first.

Expected Output

The query will produce a table with the following columns:

Product Name	**Total Sales**	**Min Quantity Sold**	**Avg Quantity Sold**	**Max Quantity Sold**
[Product Name 1]	[Count 1]	[Min 1]	[Avg 1]	[Max 1]
[Product Name 2]	[Count 2]	[Min 2]	[Avg 2]	[Max 2]
...	...	...	...	...