Lab 5: Using Functions and Aggregating Data

Retrieving Product Information

Your reports are returning the correct records, but you would like to modify how these records are displayed.

Instructions

Write a query to return the product ID of each product, together with the product name formatted as upper case and a column named `ApproxWeight` with the weight of each product rounded to the nearest whole unit. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT p.ProductID, UPPER(p.Name) AS ProductName, ROUND(p.Weight, 0) AS ApproxWeight

FROM SalesLT.Product AS p;
```

Retrieving Product Information (2)

It would be useful to know when AdventureWorks started selling each product.

Instructions

Extend your query to include columns named `SellStartYear` and `SellStartMonth` containing the year and month in which AdventureWorks started selling each product. The month should be displayed as the month name (e.g. 'January'). Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT ProductID, UPPER(Name) AS ProductName, ROUND(Weight, 0) AS ApproxWeight,

    YEAR(p.SellStartDate) AS SellStartYear, DATENAME(m, p.SellStartDate) AS SellStartMonth

FROM SalesLT.Product AS p;
```

Retrieving Product Information (3)

It would also be useful to know the type of each product.

Instructions

Extend your query to include a column named `ProductType` that contains the leftmost two characters from the product number. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT ProductID, UPPER(Name) AS ProductName, ROUND(Weight, 0) AS ApproxWeight,

    YEAR(SellStartDate) AS SellStartYear, DATENAME(m, SellStartDate) AS SellStartMonth,
```

```
    LEFT(p.ProductNumber,2) AS ProductType
FROM SalesLT.Product AS p;
```

Retrieving Product Information (4)

Categorical data can be less useful in certain cases. Here, you only want to look at numeric product size data.

Instructions

Extend your query to filter the product returned so that only products with a numeric size are included. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT ProductID, UPPER(Name) AS ProductName, ROUND(Weight, 0) AS ApproxWeight,
    YEAR(SellStartDate) AS SellStartYear, DATENAME(m, SellStartDate) AS SellStartMonth,
    LEFT(ProductNumber, 2) AS ProductType
FROM SalesLT.Product AS p
WHERE ISNUMERIC (p.Size) = 1;
```

Ranking Customers By Revenue

The sales manager would like a list of customers ranked by sales.

Instructions

Write a query that returns a list of company names with a ranking of their place in a list of highest `TotalDue` values from the `SalesOrderHeader` table. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT C.CompanyName, SOH.TotalDue AS Revenue,
    RANK() OVER (ORDER BY SOH.TotalDue DESC) AS RankByRevenue
FROM SalesLT.SalesOrderHeader AS SOH
JOIN SalesLT.Customer AS C
ON SOH.CustomerID = C.CustomerID;
```

Aggregating Product Sales

The product manager would like aggregated information about product sales.

Instructions

Write a query to retrieve a list of the product names and the total revenue calculated as the sum of the `LineTotal` from the `SalesLT.SalesOrderDetail` table, with the results sorted in descending order of total revenue. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT p.Name, SUM(sod.LineTotal) AS TotalRevenue

FROM SalesLT.SalesOrderDetail AS sod

JOIN SalesLT.Product AS p

ON p.ProductID = sod.ProductID

GROUP BY p.Name

ORDER BY TotalRevenue DESC;
```

Aggregating Product Sales (2)

The product manager would like aggregated information about product sales.

Instructions

Modify the previous query to include sales totals for products that have a list price of more than `1000`. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT Name, SUM(LineTotal) AS TotalRevenue

FROM SalesLT.SalesOrderDetail AS SOD

JOIN SalesLT.Product AS P ON SOD.ProductID = P.ProductID

WHERE p.ListPrice > 1000

GROUP BY P.Name

ORDER BY TotalRevenue DESC;
```

Aggregating Product Sales (3)

The product manager would like aggregated information about the products which grossed a very large amount.

Instructions

Modify the previous query to only include products with total sales greater than `20000`. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT Name, SUM(LineTotal) AS TotalRevenue
```

```
FROM SalesLT.SalesOrderDetail AS SOD
JOIN SalesLT.Product AS P
ON SOD.ProductID = P.ProductID
WHERE p.ListPrice > 1000
GROUP BY P.Name
HAVING SUM(LineTotal) > 20000
ORDER BY TotalRevenue DESC
```