

# Querying with Transact-SQL

## Lab 5 – Using Functions and Aggregating Data

### OVERVIEW

In this lab, you will write queries that use functions to retrieve, aggregate, and group data from the **AdventureWorksLT** database.

Before starting this lab, you should view **Module 5 – Using Functions and Aggregating Data** in the course *Querying with Transact-SQL*. Then, if you have not already done so, follow the instructions in the **Getting Started** document for this course to set up the lab environment.

If you find some of the challenges difficult, don't worry – you can find suggested solutions for all of the challenges in the **Lab Solution** folder for this module.

### WHAT YOU'LL NEED

- An Azure SQL Database instance with the **AdventureWorksLT** sample database. Review the **Getting Started** document for information about how to provision this.

### CHALLENGE 1: RETRIEVE PRODUCT INFORMATION

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

**Tip:** Review the documentation for [Built-In Functions](#) in the Transact-SQL Reference.

#### 1. Retrieve the name and approximate weight of each product

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.

#### 2. Retrieve the year and month in which products were first sold

Extend your query to include columns named **SellStartYear** and **SellStartMonth** containing the year and month in which Adventure Works started selling each product. The month should be displayed as the month name (for example, 'January').

#### 3. Extract product types from product numbers

Extend your query to include a column named **ProductType** that contains the leftmost two characters from the product number.

#### 4. Retrieve only products with a numeric size

Extend your query to filter the product returned so that only products with a numeric size are included.

## CHALLENGE 2: RANK CUSTOMERS BY REVENUE

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

**Tip:** Review the documentation for [Ranking Functions](#) in the Transact-SQL Reference.

### 1. Retrieve companies ranked by sales totals

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.

## CHALLENGE 3: AGGREGATE PRODUCT SALES

The product manager would like aggregated information about product sales.

**Tip:** Review the documentation for the [GROUP BY](#) clause in the Transact-SQL Reference.

### 1. Retrieve total sales by product

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.

### 2. Filter the product sales list to include only products that cost over \$1,000

Modify the previous query to include sales totals for products that have a list price of more than \$1000.

### 3. Filter the product sales groups to include only total sales over \$20,000

Modify the previous query to only include only product groups with a total sales value greater than \$20,000.

## NEXT STEPS

Well done! You've completed the lab, and you're ready to learn how to nest queries within one another in **Module 6 – Using Subqueries and APPLY** in the Course *Querying with Transact-SQL*.