# Lab 6: Using Subqueries and APPLY

## Retrieving Product Price Information

AdventureWorks products each have a standard cost that indicates the cost of manufacturing the product, and a list price that indicates the recommended selling price for the product. This data is stored in the `SalesLT.Product` table.
Whenever a product is ordered, the actual unit price at which it was sold is also recorded in the `SalesLT.SalesOrderDetail` table.
Use subqueries to compare the cost and list prices for each product with the unit prices charged in each sale.

### Instructions

Retrieve the product ID, name, and list price for each product where the list price is higher than the average unit price for all products that have been sold.

```
SELECT p.ProductID,p.Name,p.ListPrice

FROM SalesLT.Product AS p

WHERE p.ListPrice >

(SELECT AVG(UnitPrice) FROM SalesLT.SalesOrderDetail)

ORDER BY ProductID;
```

## Retrieving Product Price Information (2)

AdventureWorks is interested in finding out which products are being sold at a loss.

### Instructions

Retrieve the product ID, name, and list price for each product where the list price is `100` or more, and the product has been sold for (strictly) less than `100`.
Remember, the `ProductID` in your subquery will be from the `SalesLT.SalesOrderDetail` table.

```
SELECT ProductID, Name, ListPrice

FROM SalesLT.Product

WHERE ProductID IN

 (SELECT ProductID from SalesLT.SalesOrderDetail WHERE UnitPrice < 100)

AND ListPrice >= 100

ORDER BY ProductID;
```

## Retrieving Product Price Information (3)

In order to get an idea of how many products are selling above or below list price, you want to gather some aggregate product data.

### Instructions

Retrieve the product ID, name, cost, and list price for each product along with the average unit price for which that product has been sold. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT ProductID, Name, StandardCost, ListPrice,

(SELECT AVG(sod.UnitPrice)

 FROM SalesLT.SalesOrderDetail AS sod

 WHERE P.ProductID = SOD.ProductID) AS AvgSellingPrice

FROM SalesLT.Product AS P

ORDER BY P.ProductID;
```

## Retrieving Product Price Information (4)

AdventureWorks is interested in finding out which products are costing more than they're being sold for, on average.

### Instructions

Filter the query for the previous exercise to include only products where the cost is higher than the average selling price. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT ProductID, Name, StandardCost, ListPrice,

(SELECT AVG(UnitPrice)

 FROM SalesLT.SalesOrderDetail AS SOD

 WHERE P.ProductID = SOD.ProductID) AS AvgSellingPrice

FROM SalesLT.Product AS P

WHERE StandardCost  >

(SELECT AVG(UnitPrice)

 FROM SalesLT.SalesOrderDetail AS SOD

 WHERE P.ProductID = SOD.ProductID)

ORDER BY P.ProductID;
```

# Retrieving Customer Information

The AdventureWorksLT database includes a table-valued user-defined function named `dbo.ufnGetCustomerInformation`. Use this function to retrieve details of customers based on customer ID values retrieved from tables in the database.

## Instructions

Retrieve the sales order ID, customer ID, first name, last name, and total due for all sales orders from the `SalesLT.SalesOrderHeader` table and the `dbo.ufnGetCustomerInformation` function. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT SalesOrderID,FirstName,LastName,CI.CustomerID,TotalDue

FROM SalesLT.SalesOrderHeader AS SOH

CROSS APPLY dbo.ufnGetCustomerInformation(CustomerID) AS CI

ORDER BY SOH.SalesOrderID;
```

# Retrieving Customer Information (2)

Use the table-valued user-defined function `dbo.ufnGetCustomerInformation` again to to retrieve details of customers based on customer ID values retrieved from tables in the database.

## Instructions

Retrieve the customer ID, first name, last name, address line 1 and city for all customers from the `SalesLT.Address` and `SalesLT.CustomerAddress`tables, and the `dbo.ufnGetCustomerInformation` function. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT CA.CustomerID, CI.FirstName, CI.LastName, A.AddressLine1, A.City

FROM SalesLT.Address AS A

JOIN SalesLT.CustomerAddress AS CA

ON A.AddressID = CA.AddressID

CROSS APPLY dbo.ufnGetCustomerInformation (CA.CustomerID) AS CI

ORDER BY CA.CustomerID;
```