# Querying Data with Transact-SQL - Final Assessment

## Section 1

1. **Select the quantity per unit for all products in the Products table.**

SELECT QuantityPerUnit

FROM dbo.Products

1. **Select the unique category IDs from the Products table.**

SELECT DISTINCT CategoryID

FROM dbo.Products

1. **Select the names of products from the Products table which have more than 20 units left in stock.**

SELECT ProductName

FROM dbo.ProductS

WHERE UnitsInStock > 20

1. **Select the product ID, product name, and unit price of the 10 most expensive products from the Products table.**

SELECT TOP 10 ProductID, ProductName, UnitPrice

FROM dbo.Products

ORDER BY UnitPrice DESC

1. **Select the product ID, product name, and quantity per unit for all products in the Products table. Sort your results alphabetically by product name (where A comes first).**

SELECT ProductID, ProductName, QuantityPerUnit

FROM dbo.Products

ORDER BY ProductName

1. **Select the product ID, product name, and unit price of all products in the Products table. Sort your results by number of units in stock, from greatest to least.**

**Skip the first 10 results and get the next 5 after that.**

SELECT ProductID, ProductName, UnitPrice

FROM dbo.Products

ORDER BY UnitsInStock DESC

OFFSET 10 ROWS

FETCH NEXT 5 ROWS ONLY

1. **Use STR, CONVERT, and NVARCHAR(30) where appropriate to display the first name, employee ID and birthdate (as Unicode in ISO 8601 format) for each employee in the Employees table.**

**Each result should be a single string in the following format, where each <<value>> is replaced by the appropriately converted value:**

**<<FirstName>> has an EmployeeID of <<EmployeeID>> and was born <<BirthDate>>**

**Your submission should exactly match the format given above.**

SELECT FirstName + ' has an EmployeeID of ' + CAST(EmployeeID AS VarChar(1)) + ' and was born ' + CONVERT(NVARCHAR(30), BirthDate, 126)

FROM Employees

1. **Select from the Orders table.**

**The first column of your result should be a single string in exactly the following format:**

**<<ShipName>> is from <<ShipCity or ShipRegion or ShipCountry>>**

**If there is no ShipCity, then you should select ShipRegion, and if there is no ShipRegion you should select ShipCountry.**

SELECT ShipName + ' is from ' + COALESCE(ShipCity, ShipRegion, ShipCountry)

FROM Orders

1. **Select the ship name and ship postal code from the Orders table. If the postal code is missing, display 'unknown'.**

SELECT ShipName, ISNULL(ShipPostalCode, 'unknown')

FROM Orders

1. **Using the Suppliers table, select the company name, and use a simple CASE expression to display 'outdated' if the company has a fax number, or 'modern' if it doesn't. Do this by specifying IS NULL in your conditional statement.**

**Alias the result of the CASE expression to Status.**

SELECT CompanyName,

CASE

WHEN Fax IS NOT NULL THEN 'outdated'

ELSE 'modern'

END AS Status

FROM Suppliers

## Section 2

1. **Get the order ID and unit price for each order by joining the Orders table and the Order Details table.**

**Note that you need to use [Order Details] since the table name contains whitespace.**

SELECT o.OrderID, od.UnitPrice

FROM dbo.Orders AS o

JOIN dbo.[Order Details] AS od

ON o.OrderID = od.OrderID

1. **Get the order ID and first name of the associated employee by joining the Orders and Employees tables.**

SELECT o.OrderID, e.FirstName

FROM dbo.Orders AS o

JOIN dbo.Employees AS e

ON o.EmployeeID = e.EmployeeID

1. **Get the employee ID and related territory description for each territory an employee is in, by joining the Employees, EmployeeTerritories and Territories tables.**

SELECT e.EmployeeID, t.TerritoryDescription

FROM dbo.Employees AS e

JOIN dbo.EmployeeTerritories AS et

ON e.EmployeeID = et.EmployeeID

JOIN dbo.Territories AS t

ON et.TerritoryID = t.TerritoryID

1. **Select all the different countries from the Customers table and the Suppliers table using UNION.**

SELECT c.Country

FROM dbo.Customers AS c

UNION

SELECT s.Country

FROM dbo.Suppliers AS s

1. **Select all the countries, including duplicates, from the Customers table and the Suppliers table using UNION ALL.**

SELECT c.Country

FROM dbo.Customers AS c

UNION ALL

SELECT s.Country

FROM dbo.Suppliers AS s

1. **Using the Products table, get the unit price of each product, rounded to the nearest dollar.**

SELECT ROUND(p.UnitPrice, 0)

FROM dbo.Products AS p

1. **Using the Products table, get the total number of units in stock across all products.**

SELECT SUM(UnitsInStock)

FROM dbo.Products AS p

1. **Using the Orders table, get the order ID and year of the order by using YEAR(). Alias the year as OrderYear.**

SELECT OrderID, YEAR(OrderDate) AS OrderYear

FROM dbo.Orders AS o

1. **Using the Orders table, get the order ID and month of the order by using DATENAME(). Alias the month as OrderMonth.**

SELECT OrderID, DATENAME(M, OrderDate) AS OrderMonth

FROM dbo.Orders AS o

1. **Use LEFT() to get the first two letters of each region description from the Region table.**

SELECT LEFT(RegionDescription, 2)

FROM dbo.Region

1. **Using the Suppliers table, select the city and postal code for each supplier, using WHERE and ISNUMERIC() to select only those postal codes which have no letters in them.**

SELECT City, PostalCode

FROM dbo.Suppliers

WHERE ISNUMERIC(PostalCode) = 1

1. **Use LEFT() and UPPER() to get the first letter (capitalized) of each region description from the Region table.**

SELECT UPPER(LEFT(RegionDescription, 1))

FROM dbo.Region

## Section 3

1. **Use a subquery to get the product name and unit price of products from the Products table which have a unit price greater than the average unit price from the Order Details table.**

**Note that you need to use [Order Details] since the table name contains whitespace.**

SELECT ProductName, UnitPrice

FROM dbo.Products

WHERE UnitPrice >

(

SELECT AVG(od.UnitPrice)

FROM dbo.[Order Details] as od

)

1. **Select from the Employees and Orders tables. Use a subquery to get the first name and employee ID for employees who were associated with orders which shipped from the USA.**

SELECT FirstName, EmployeeID

FROM Employees

WHERE EmployeeID IN

(SELECT EmployeeID

FROM Orders

WHERE ShipCountry = 'USA'

)

1. **Use the # to create a new temporary table called ProductNames which has one field called ProductName (a VARCHAR of max length 40).**

**Insert into this table the names of every product from the Products table. Note that there are two syntaxes for the INSERT INTO statement. Use the syntax that does not specify the column names since the table only has one field.**

**Select all columns from the ProductNames table you created.**

**Note: you need to specify the Products table as Products, not dbo.Products.**

CREATE TABLE #ProductNames

(ProductName VARCHAR(40))

INSERT INTO #ProductNames

SELECT ProductName FROM Products

SELECT \*

FROM #ProductNames

## Section 4

1. **Use CHOOSE() and MONTH() to get the season in which each order was shipped from the Orders table. You should select the order ID, shipped date, and then the season aliased as ShippedSeason. You can copy and paste the below into your query.**

**'Winter', 'Winter', 'Spring', 'Spring', 'Spring', 'Summer', 'Summer', 'Summer', 'Autumn', 'Autumn', 'Autumn', 'Winter'**

**Be careful to filter out any NULL shipped dates.**

SELECT OrderID, ShippedDate,

CHOOSE(MONTH(ShippedDate),'Winter', 'Winter', 'Spring', 'Spring', 'Spring', 'Summer', 'Summer', 'Summer', 'Autumn', 'Autumn', 'Autumn', 'Winter') AS ShippedSeason

FROM dbo.Orders

WHERE ShippedDate IS NOT NULL

1. **Using the Suppliers table, select the company name and use a simple IIF expression to display 'outdated' if a company has a fax number, or 'modern' if it doesn't. Alias the result of the IIF expression to Status.**

SELECT CompanyName, IIF ( Fax IS NOT NULL, 'outdated', 'modern' ) AS Status

FROM dbo.Suppliers

1. **Select from the Customers, Orders, and Order Details tables. Note that you need to use [Order Details] since the table name contains whitespace.**

**Use GROUP BY and ROLLUP() to get the total quantity ordered by all countries, while maintaining the total per country in your result set.**

**Your first column should be the country, and the second column the total quantity ordered by that country, aliased as TotalQuantity.**

SELECT Country, sum(Quantity) as TotalQuantity

FROM dbo.Customers as c

JOIN dbo.Orders as o

ON c.CustomerID = o.CustomerID

JOIN dbo.[Order Details] as od

ON o.OrderID = od.OrderID

GROUP BY ROLLUP(Country)

1. **From the Customers table, use GROUP BY to select the country, contact title, and count of that contact title aliased as Count, grouped by country and contact title (in that order).**

**Then use CASE WHEN, GROUPING\_ID(), and ROLLUP() to add a column called Legend, which shows one of two things:**

* **When the GROUPING\_ID is 0, show '' (i.e., nothing)**
* **When the GROUPING\_ID is 1, show Subtotal for << Country >>'**

**Do not use ORDER BY to order your results.**

1. **Convert the following query to be pivoted, using PIVOT().**

**SELECT CategoryID, AVG(UnitPrice)**

**FROM Products**

**GROUP BY CategoryID;**

**Your result set should look like this:**

| **Per Category** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Average Unit Price** | **37.9791** | **23.0625** | **25.1600** | **28.7300** | **20.2500** | **54.0066** | **32.3700** | **20.6825** |

SELECT 'Average Unit Price' as 'Per Category',

[1], [2], [3], [4], [5], [6], [7], [8]

FROM

(

SELECT CategoryId, AVG(UnitPrice) as UnitPrice

FROM Products

GROUP BY CategoryID)

AS SourceTable

PIVOT

(

AVG(UnitPrice)

FOR CategoryID IN ([1], [2], [3], [4], [5], [6], [7], [8])

)

AS PivotTable;

1. **Insert into the Region table the region ID 5 and the description 'Space'.**

**Then, in a second query, select the newly inserted data from the table using a WHERE clause.**

**Note: When you execute a query and the result is fetched, the database will be rolled back to its initial state. This means that you can click "Run Code" repeatedly, starting with a clean slate every time.**

INSERT INTO dbo.Region (RegionID, RegionDescription)

VALUES ('5', 'Space')

SELECT RegionID, RegionDescription

FROM Region

WHERE RegionID = '5' AND RegionDescription = 'Space'

1. **Update the region descriptions in the Region table to be all uppercase, using SET and UPPER().**

**Next, select all data from the table to view your updates.**

**Note: When you execute a query and the result is fetched, the database will be rolled back to its initial state. This means that you can click "Run Code" repeatedly, starting with a clean slate every time.**

UPDATE Region

SET RegionDescription = UPPER(RegionDescription)

SELECT \*

FROM Region