Module 02 Notes

# Relational vs Non-Relational Data

# ****Module Overview****

In this module, you learn about the different **advanced features of SQL Server and its Non-relational database options**. The assignment will provide multiple ways to learn this material.

# ****Required Software****

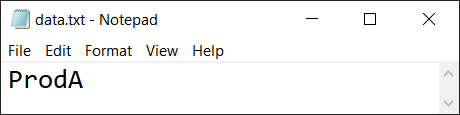
This module requires **SQL Developer Edition** (With Database Engine and **Cube** Servers), **Visual Studio** 2019 Community edition (SSAS **Cube** and SSAS **Tabular** **extension**), **Excel**, and **PowerBI** Desktop.

# Data in Files

**Data** is what you store, **and information** is what you get from stored data. Data is stored either as **binary** data **or** human-friendly **text.** Instead of **manual entry**, **applications** are used to create and manage data in files

## Text Files

Text can be **created by programming or** a text editor **application**, like Notepad.



## CSV

Separating multiple values has and is a **popular** way to store simple data

"Name","Price","Category"

"ProdA",9.99,"Cat1"

"ProdB",1.99,"Cat1"

"ProdC",":1.99,"Cat1,Cat2"

## JSON

Text files are being **updated** from a CSV to a JSON format

{"Products":[

{"Name":"ProdA","Price":9.99,"Category":"Cat1"}

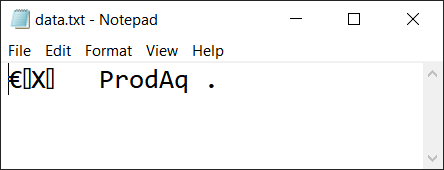
{"Name":"ProdB","Price":1.99,"Category":"Cat1"}

{"Name":"ProdC","Price":1.99,"Category":["Cat1","Cat2"]}

]}

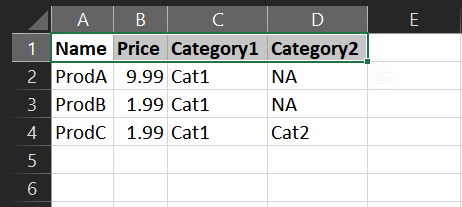
## Binary Files

* Binary files are created by **programming**, but **seldom** an **editor**
* Working with binary files directly is **tedious for humans**
* Binary file creation is more **complex**



## Spreadsheets

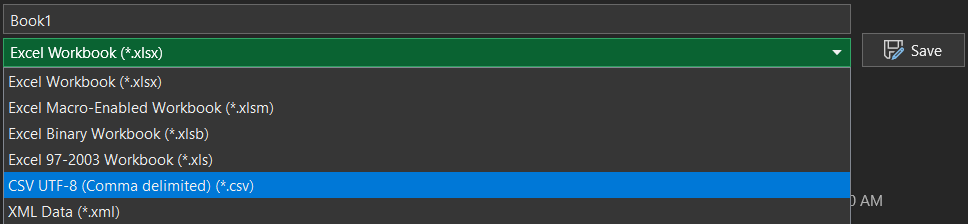
* Spreadsheets are pre-programmed **applications that manage data**
* Most **popular** due to **simplicity**



* Data in **usually saved in a binary** format by default.



* Most spreadsheet applications **allow data to be exported** to other file formats



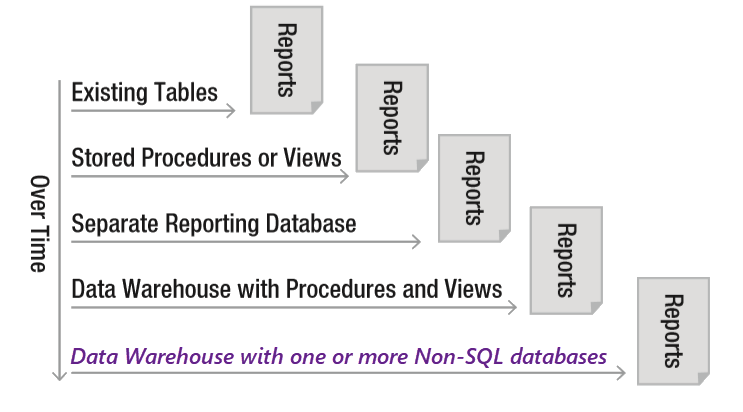
# Demo 01: File Data

In this demonstration we will review several different examples of file data.

* 0\_RelationalProductData.sql
* 1\_ProductData.csv
* 2\_ProductData.json
* 3\_ProductData.xml
* 4\_ProductData.xlsx

# Data in Relational Databases

* **Most** relational database applications **use SQL code** to create and manage data in files
* **Most** relational databases are used to track Transaction Processing (**OLTP**)
* **Some** relational databases are used for Analytical Processing (**OLAP**)



Create Database MyDataDB;

go

use MyDataDB;

go

Create Table Products(

"Name" varchar(100)

,"Price" money

,"Category" varchar(100)

);

go

Insert into Products

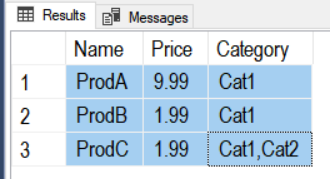
("Name","Price","Category")

Values

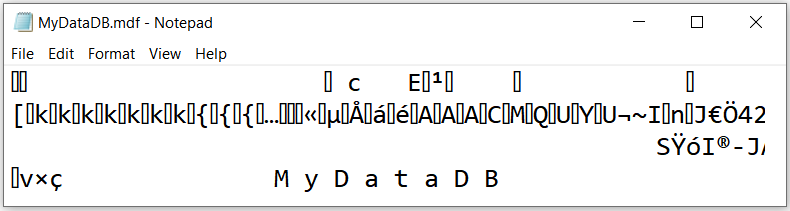
('ProdA',9.99,'Cat1')

,('ProdB',1.99,'Cat1')

,('ProdC',1.99,'Cat1,Cat2');

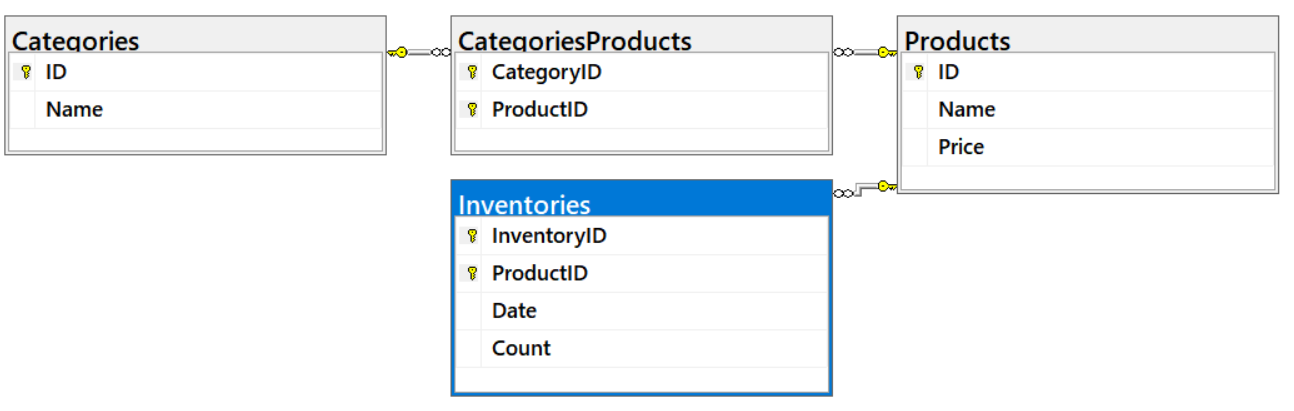


* **Most** relational database applications store data in **binary** files



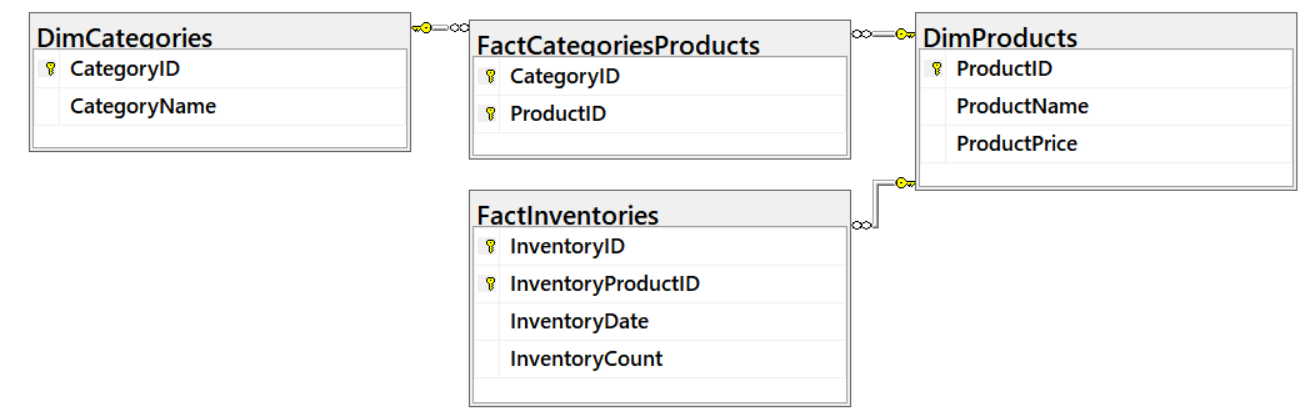
## Normalized Databases

* Normalized databases are **complex** in design
* Normalized databases require **complex code to extract data**
* The complex code is made **simpler using SQL** Code
* **OLTP** databases are often **more Normalized** than OLAP databases
* **Views combine** all **data into a single "logical" object**



## Data Warehouses (Stars and Snowflakes)

* Data warehouses allow you to **clean up the design** for better reporting
* Data warehouses consist of one or more tables **used for reporting**
* Data warehouses are designed to **simplify reporting queries** and **improve performance**
* Data warehouses may be **LESS normalized**
* Each **subject/event** in a data warehouse database is represented by a **set table (ex. Sales vs. Inventory data)**
* Data Marts are a set of tables for one subject/event (**Sales Data Mart vs. Inventory Data Mart**)
* **Events** are recorded as **Facts**, while **Subjects** are recorded as **Dimensions** of those facts
* **Event** **Facts** are usually recorded **with a date/time** component
* **Subjects** are described by **dimensions** (Product sold, or Product counted)
* Each **dimension** needs its **set of one or more tables**
* If **multiple tables** are part of the **same dimension**, you can choose either **star or snowflake**
* If there is a **many to many relationships**, you **must use a Bridge table**
* **Many to Many Bridges** are **Facts** **without a date/time** component



# Demo 02: Create a Data Warehouse Database

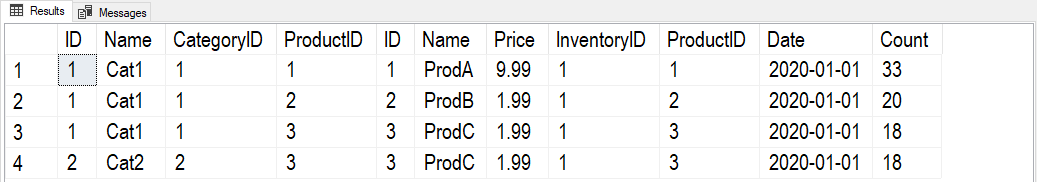
In this demo, we create a Transactional database (OLTP) and a Data Warehouse database (OLAP).

1. **Verify** that both SQL Server and Management Studio are installed

2. **Open, review, and execute** the SQL Script "0\_OLTPSourceDB.sql"

3. **Analyze** its data to answer these questions:

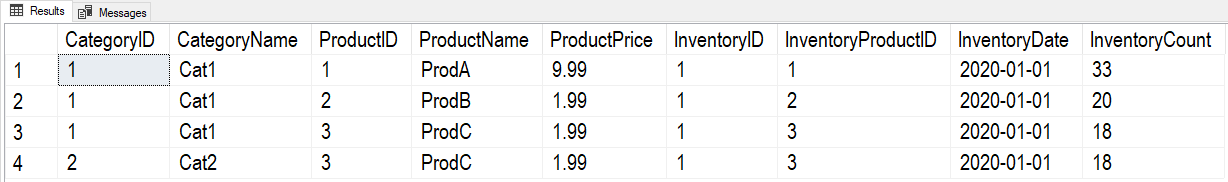
* How many "ProdC" item were counted in during the inventory?
* How many columns have the same name?



4. **Open, review, and execute** the SQL Script "1\_OLAPDataWarehouseDB.sql"

5. **Analyze** its data to answer these questions:

* How many "ProdC" items were counted?
* What are some differences between the OLTP and OLAP design?

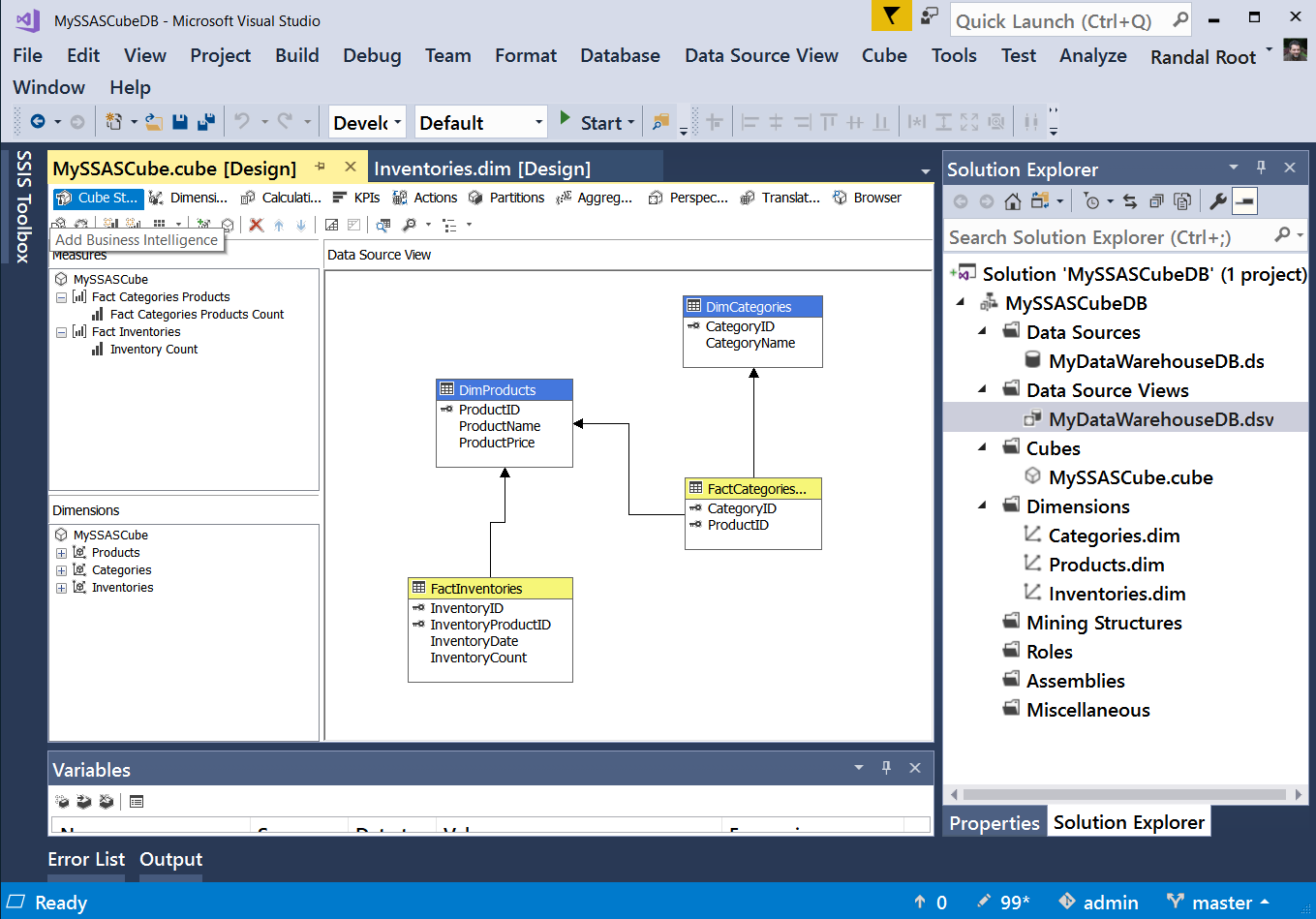


# Data in Non-Relational Databases

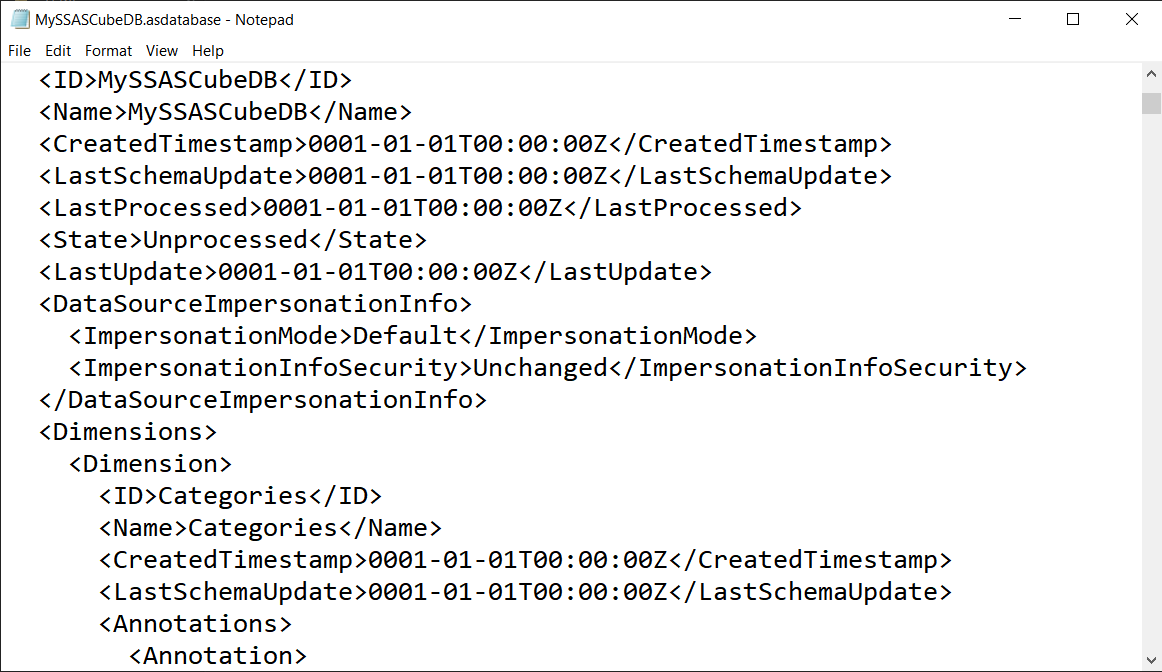
* Using non-relational databases **have become a popular choice**
* Non-relational databases are **designed to simplify reporting** queries and **improve performance**
* Non-normalized data **may take more drive space** but **may require less processing**

## Analysis Server "Multi-Dimensional" Cube structures

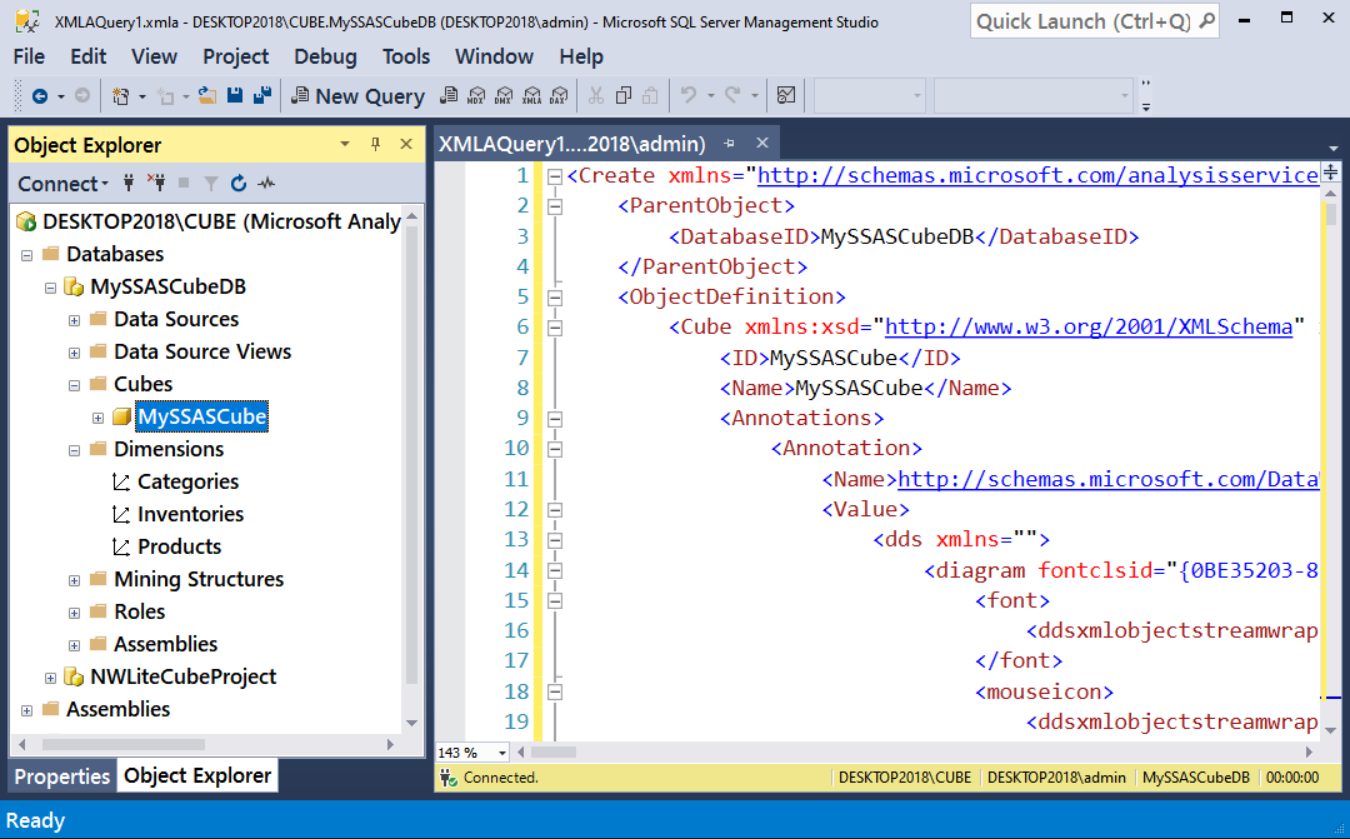
* Analysis Server Cube Databases allow you to **clean up the design** for better reporting
* Analysis Server Cubes Databases **combine** all the dimension and fact **data** **into a single "logical" object**
* You **use Visual Studio's SSAS Cube Extension** to create a SSAS Cube and dimensions



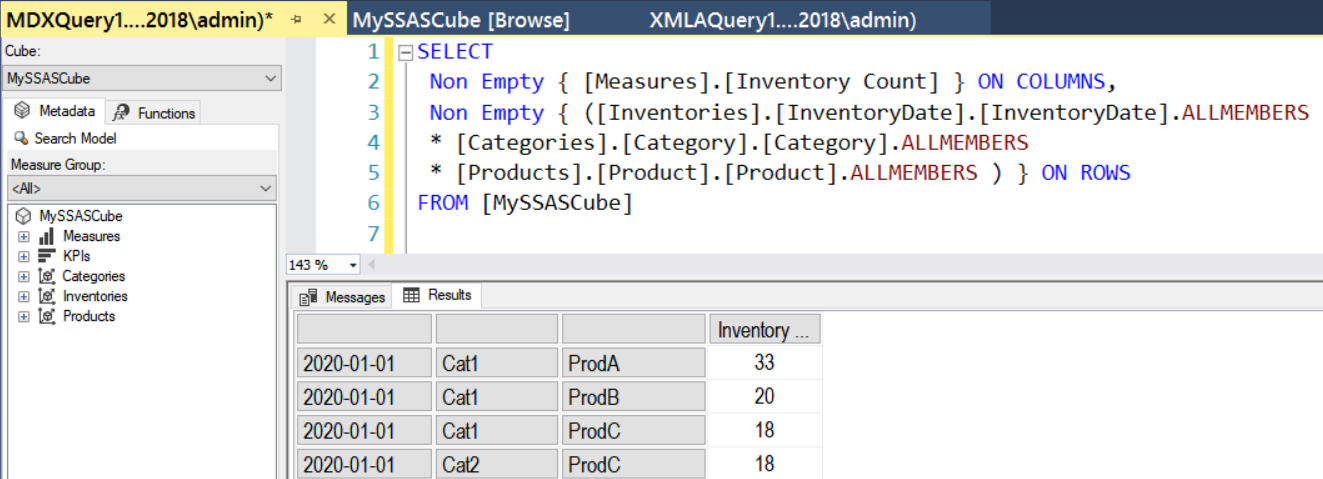
* **SSDT-BI** SSAS Cube projects **create** the cube and dimensions **using an XML language**



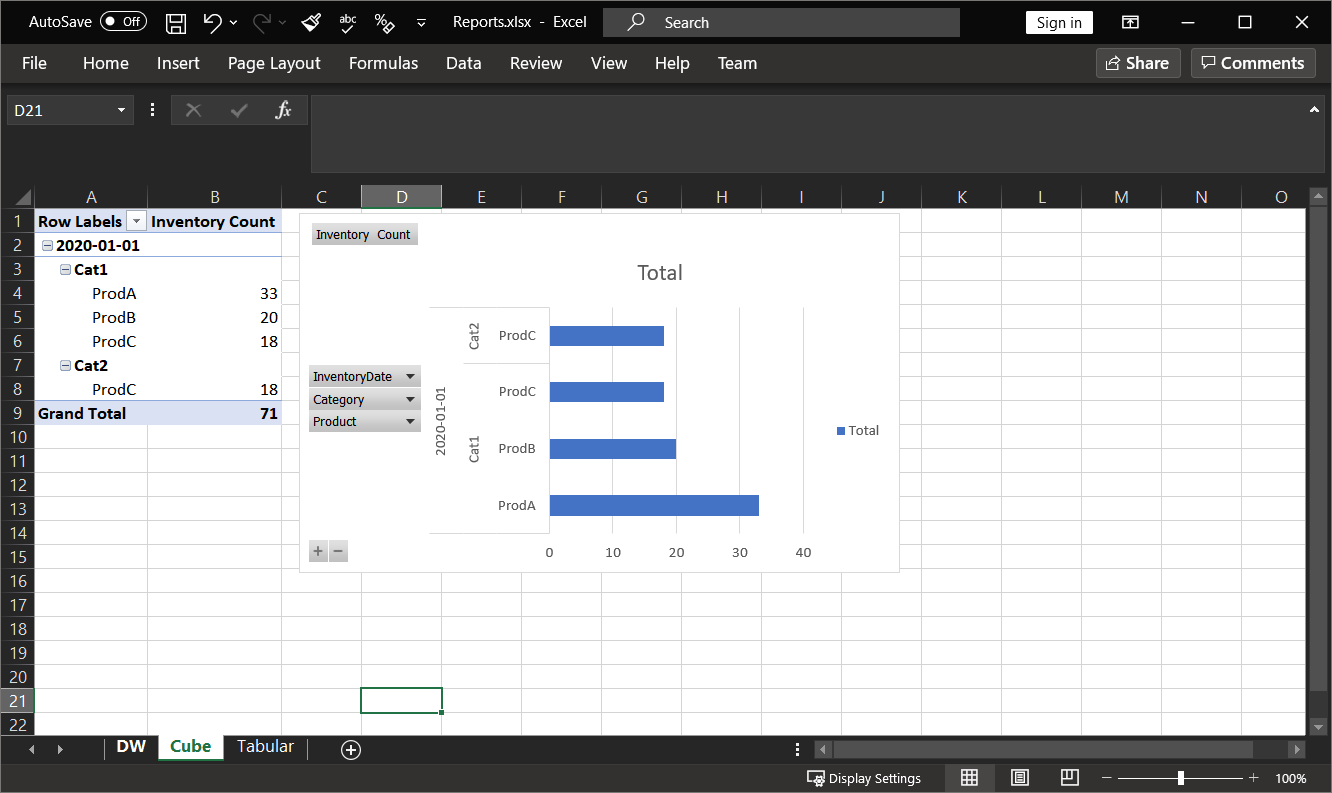
* **An XML language** called **XMLA, creates** a new Database, dimensions, and cube **on an SSAS server**



* You create **reports** using **MDX** code



* You can also use **applications** to make more **visual reports**
* SSAS Cube reports **may include "double counts"** in grand totals (unless configured correctly)

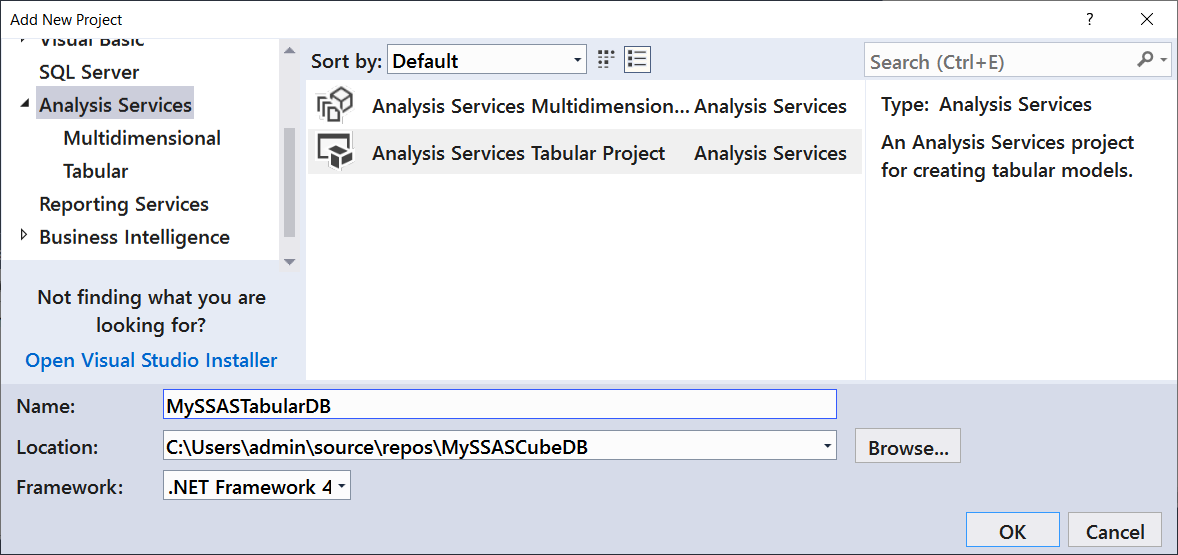


**Questions:**

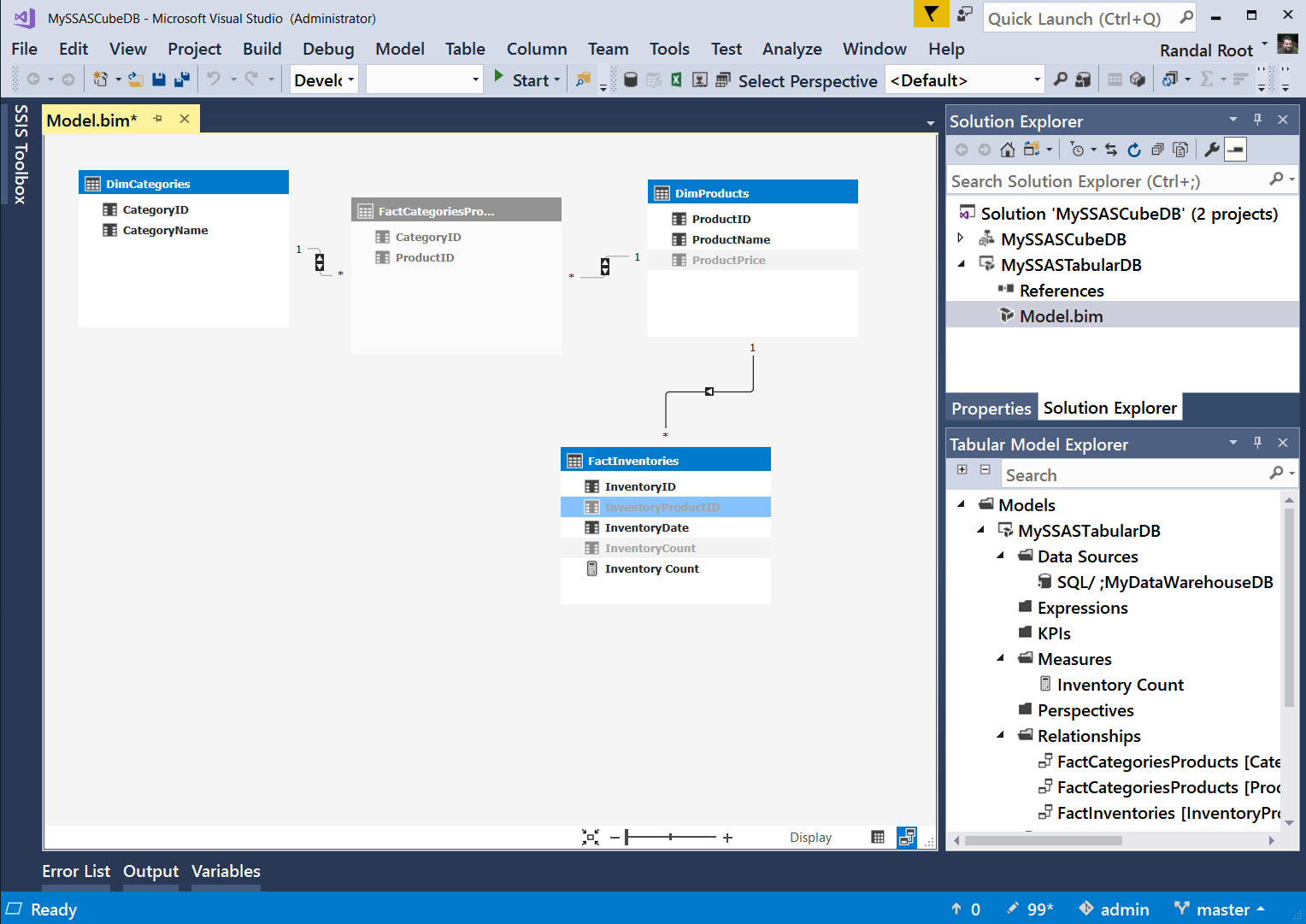
* How many "ProdC" items were counted?

## Analysis Server Tabular structures

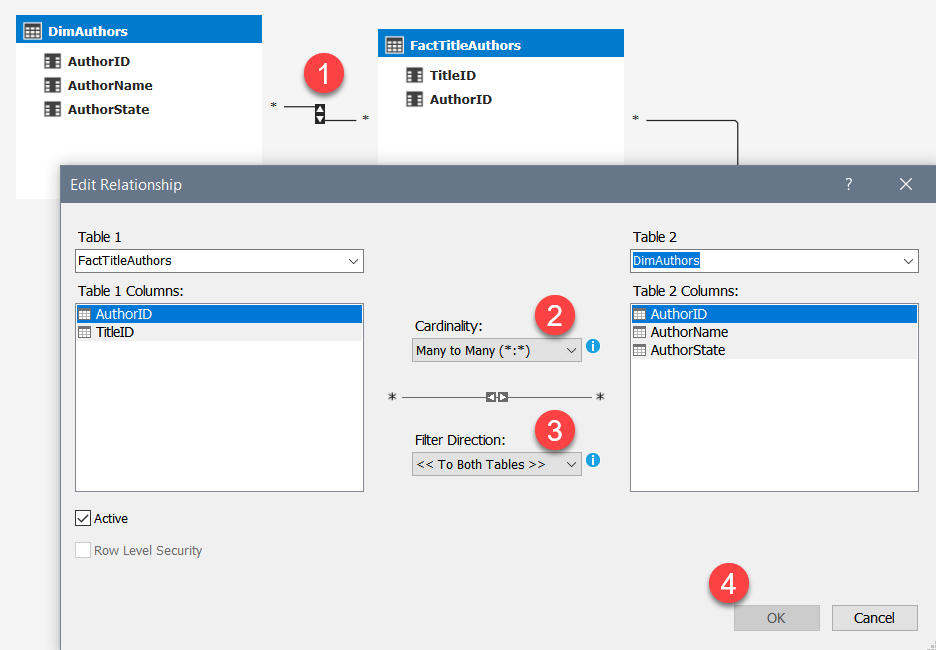
* Analysis Server Tabular databases allow you to **clean up the design** for better reporting
* Analysis Server Tabular databases **combine** all the dimension and fact **data into a single "logical" object**
* You use **SQL Server Data Tools for BI (SSDT-BI)** to create a SSAS Tabular databases



* Analysis Server Tabular **databases can have only one data model** (unlike Cubes)



* Analysis Server Tabular databases **support Many to Many relationships though bi-directional filters.**

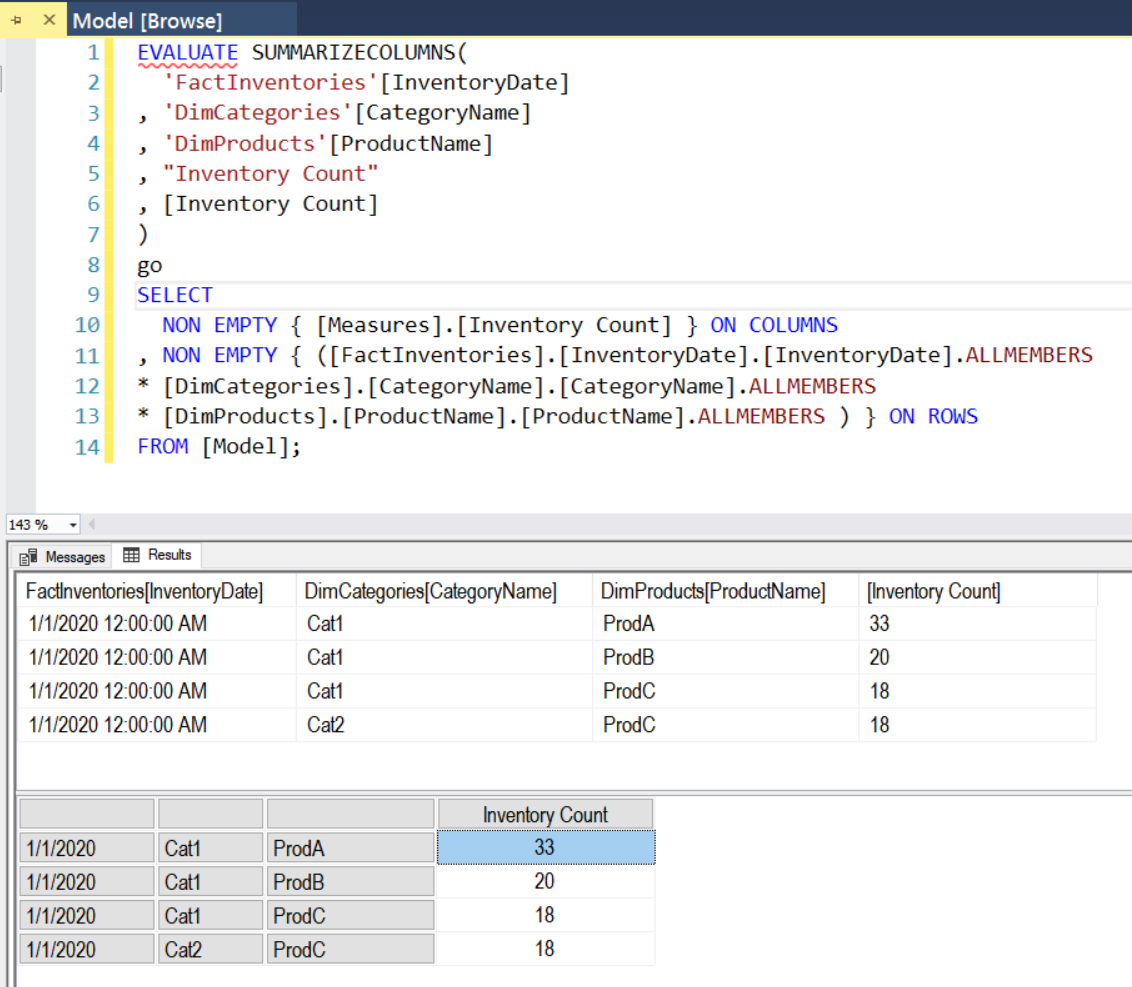


<https://docs.microsoft.com/en-us/analysis-services/tabular-models/bi-directional-cross-filters-tabular-models-analysis-services>

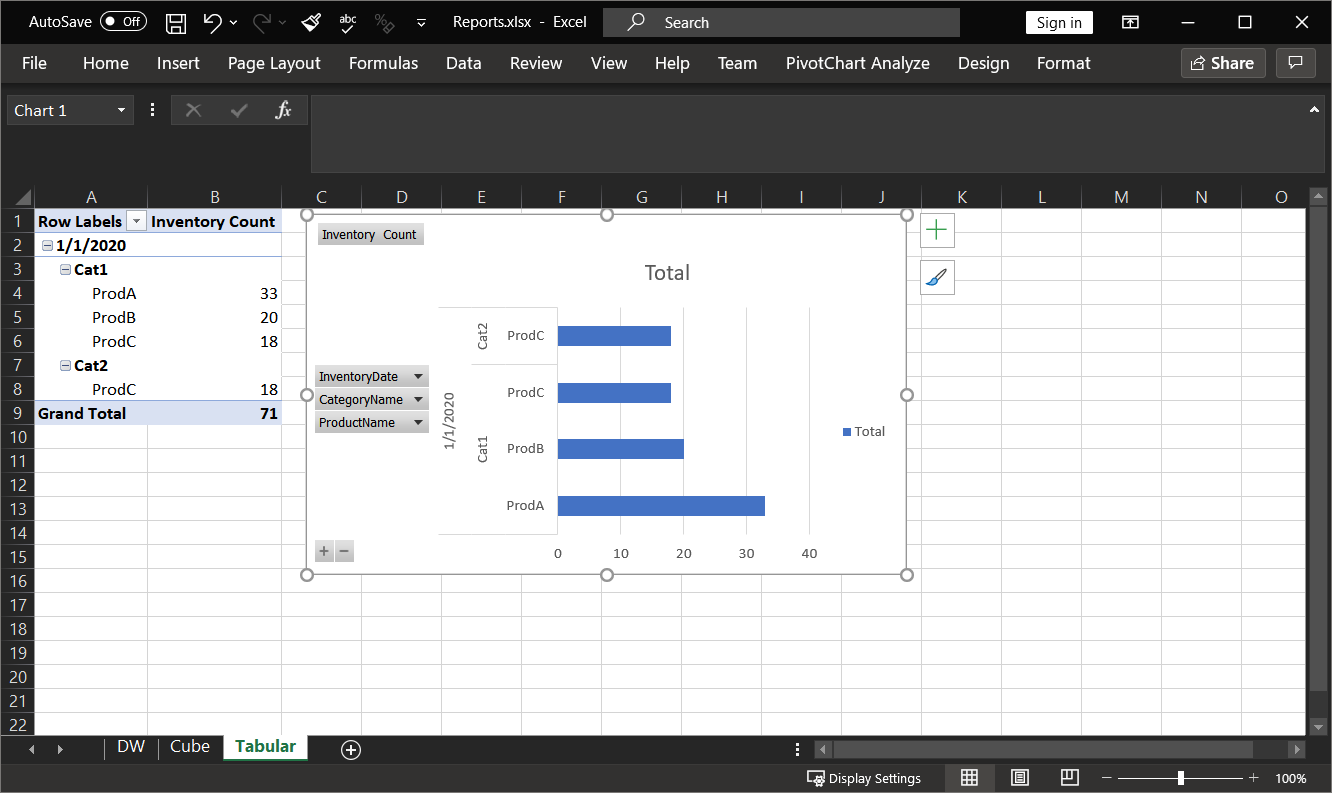
* SSDT-BI SSAS Tabular **projects uses JSON** for deployment and object scripting



* SSAS Tabular **databases use DAX or MDX** for queries



* You can also use **applications** to make more **visual reports**
* SSAS Tabular reports **may include "double counts"** in grand totals (unless configured correctly)



**Question:** How many "ProdC" items were counted?

# Demo 03: Create a Cubes and Tabular Databases

This demo will show how to create both Cube and Tabular databases, based on Demo 03's data warehouse database. Then, how to create a PowerBI report using the Tabular database's data.

# Extracting Non-Relational Data

MSSQL has allowed you to extract and import non-relational data from their relational database since SQL Server 2005. JSON data was added in SQL Server 2016.

* You can extract XML data from your data warehouse using the FOR XML clause in a SELECT statement
* You can extract JSON data from your data warehouse using the FOR JSON clause in a SELECT statement

Here is an example:

Begin Try Use Master; Drop Database MyDataDB; End Try Begin Catch End Catch

go

Create Database MyDataDB;

go

use MyDataDB;

go

Create Table Products(

"Name" varchar(100)

,"Price" Money

,"Category" varchar(100)

);

go

Insert into Products

("Name","Price","Category")

Values

('ProdA',9.99,'Cat1')

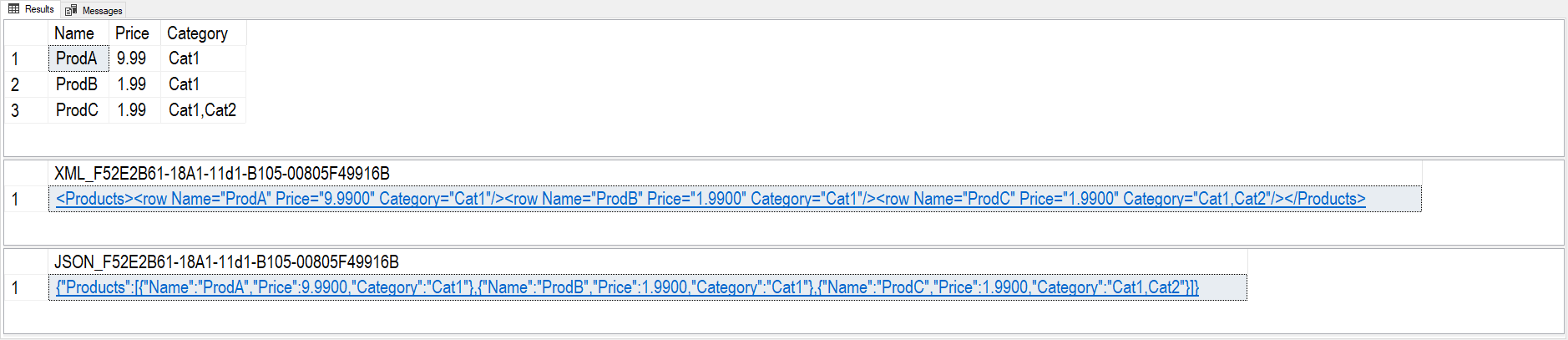
,('ProdB',1.99,'Cat1')

,('ProdC',1.99,'Cat1,Cat2');

Select Name, Price, Category From Products;

Select Name, Price, Category From Products for XML RAW, ROOT('Products');

Select Name, Price, Category From Products for JSON PATH, ROOT('Products');



# Demo04 Creating XML and JSON output with SQL Server

In this demonstration, we look at **ways to extra XML and JSON code from SQL Server**.