Module 01 Notes

Overview of Advanced SQL & Business Intelligence

# ****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****

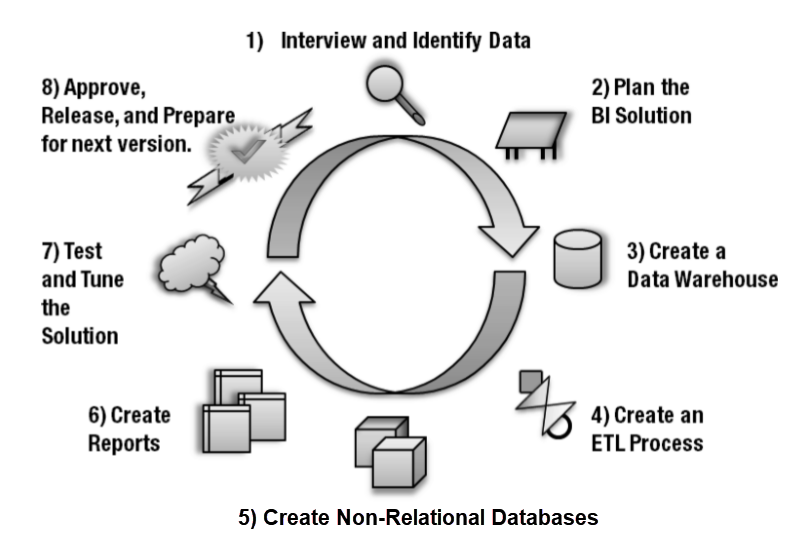
Install SQL Developer Edition (With Database Engine, **Cube** Server, **Tabular** Server, and **Integration** Server), **Visual Studio** 2019 Community edition, **PowerBI** Desktop, and **Report Builder**.

## Demonstration – Installing Software

In this demonstration, your instructor will show tips on how to get and **install the software** required for this course.

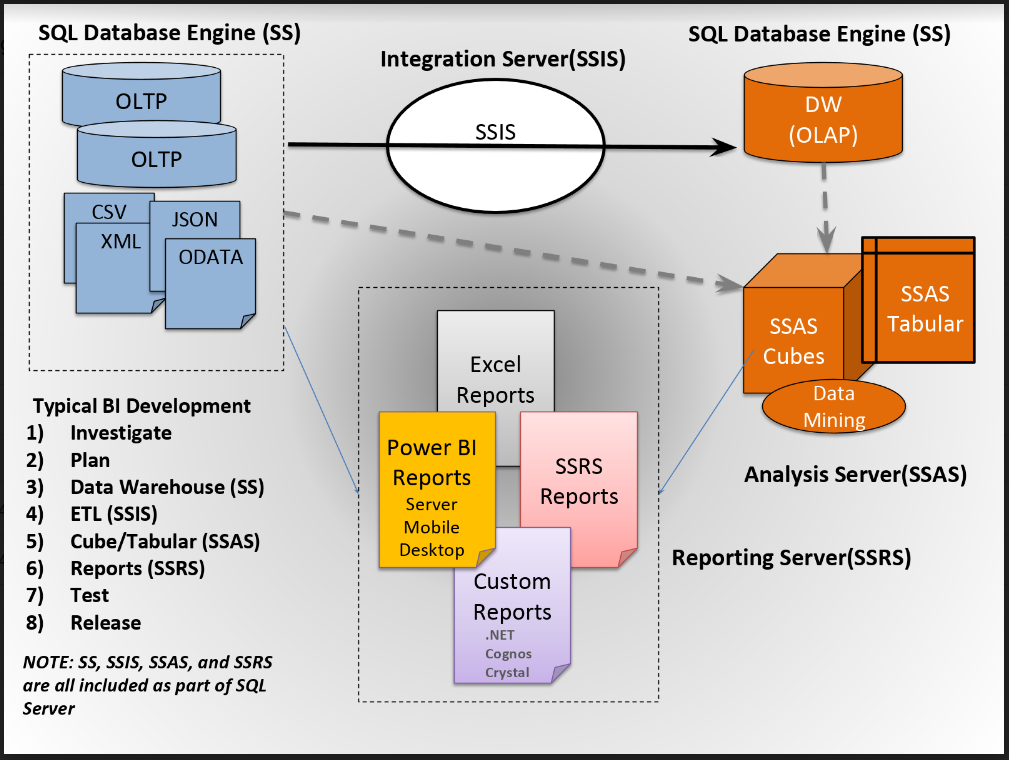
# What is a BI Solution?

* A BI solution is a **collection of objects** that **turns data into useful information**
* Solution **objects are designed, created, tested,** and ultimately **approved** to create a working BI solution
* **Eight steps** to use as a **guideline**



BI Solutions with Microsoft Tools

* **Microsoft** includes many **free tools** with the purchase of **SQL Server**

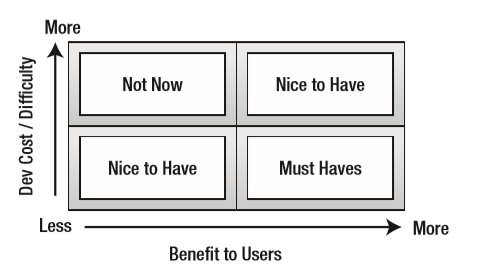


# Planning Solutions

* Planning is **fundamentally important**
* **No plan is perfect**
* A **protracted planning** process is the bane of many projects
* Even simple BI solutions can **beneﬁt from some planning**.

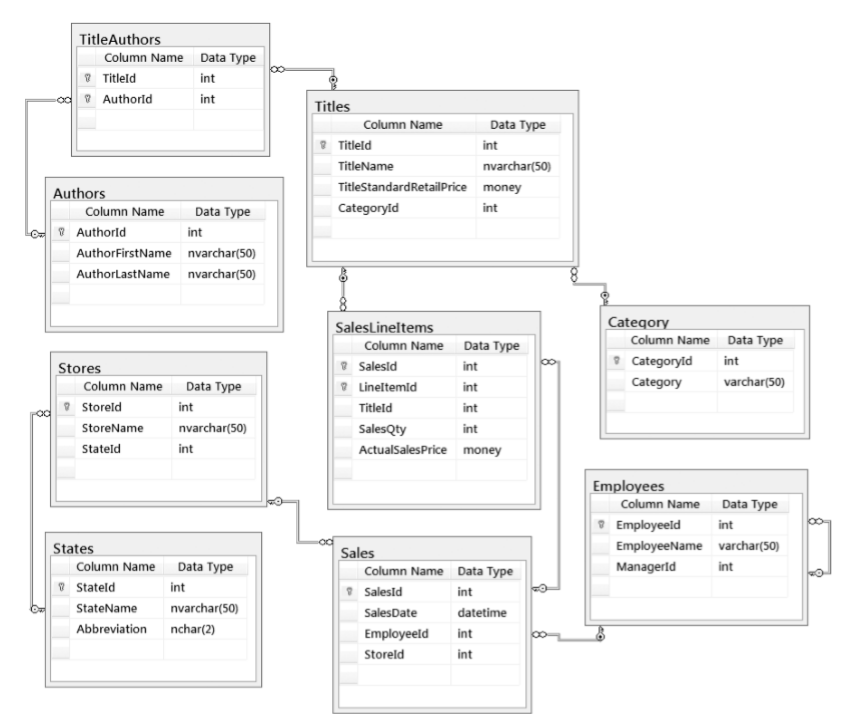
## Prioritizing

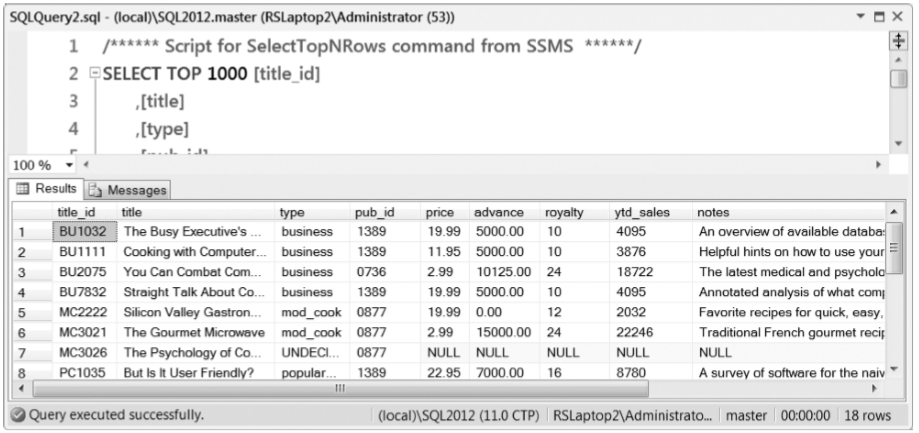
* **Create a list** of all the features that your client requested to be included in the solution
* **Examine existing** documentation and reports for inspiration.
* **Interviews** with the people may not **identify** all the **requirements** of the solution
* **Prioritize** what must be in this version of the solution.
* One method of doing this is to use a technique known as **four-quadrant prioritizing**



## Documenting the Requirements

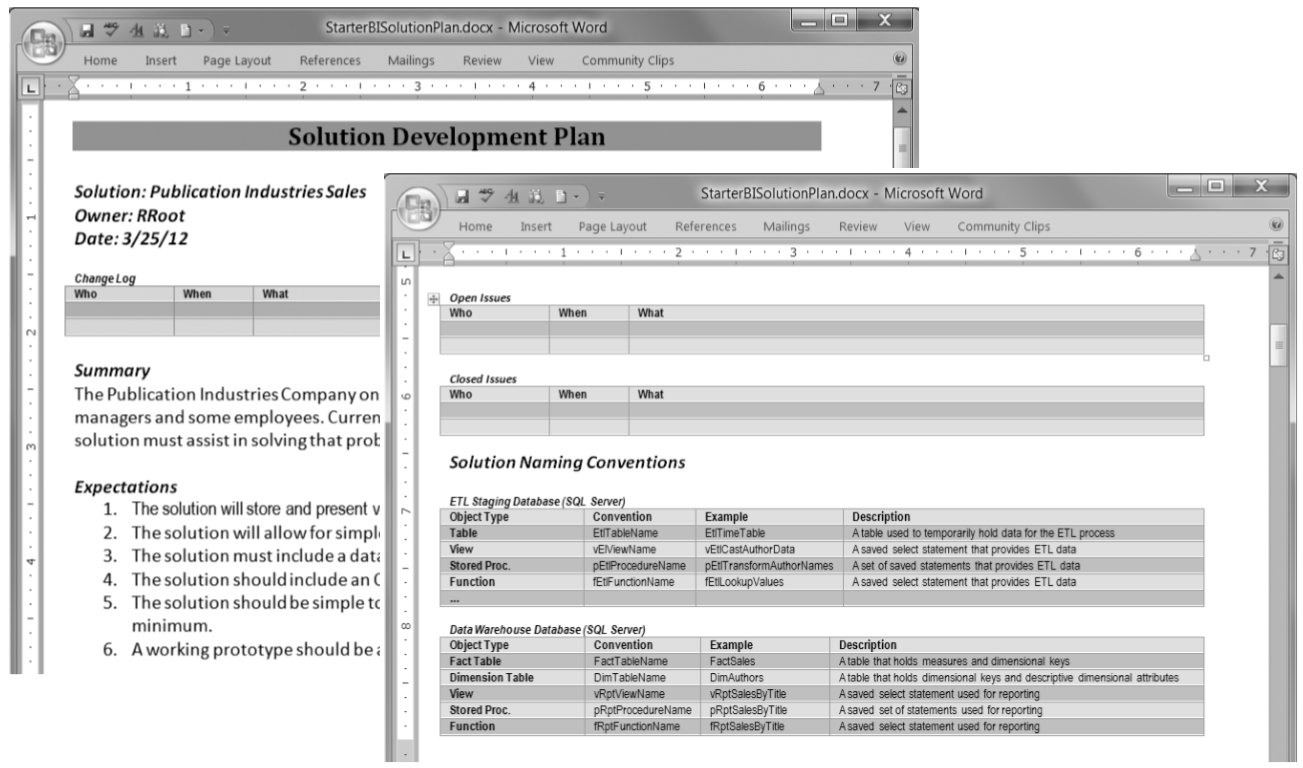
* **Locate the Source data**
* **Review** **the data** in each source table
* **Decide** **what to** **include**





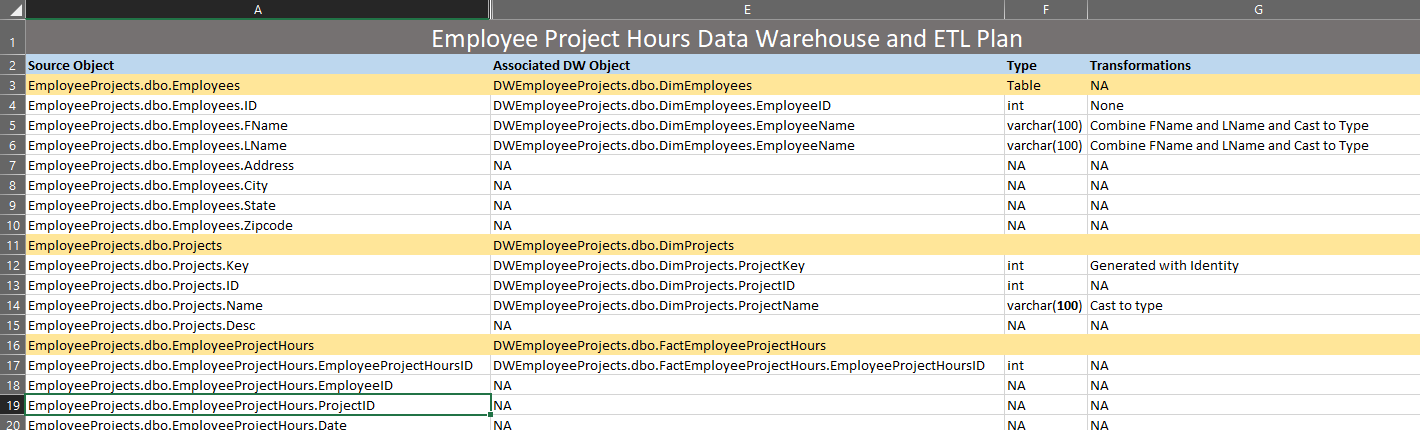
Solution document

* **Word** documents can **record designs**
* **Word** works well as a **formal document**



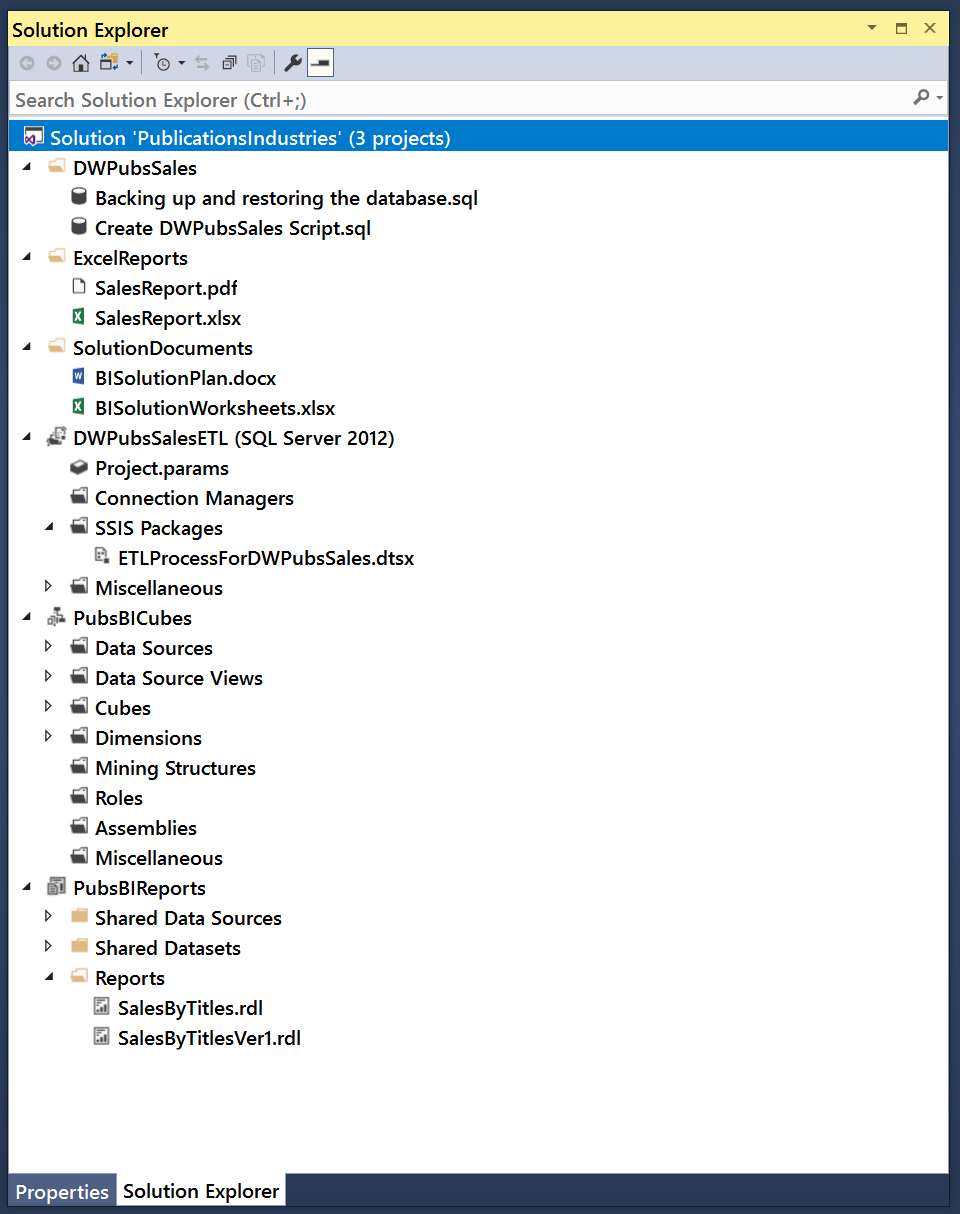
Metadata Worksheets

* **Excel** is an excellent choice for an **informal document**



Organizing your Solution

* A **BI solution** usually includes a **data warehouse along with several other components**
* **Visual Studio Projects** are **often a component** when using Microsoft servers
* One **Visual Studio Solution** can **hold many projects, scripts, and solution documents**



## Demonstration - Organizing with Visual Studio

In this demonstration, we look at an **example of organizing files** using Visual Studio.

1. Open the Visual Studio.
2. Create a "blank" Visual Studio Solution.
3. Create a folder and add documents to it.

# Designing a Data Warehouse

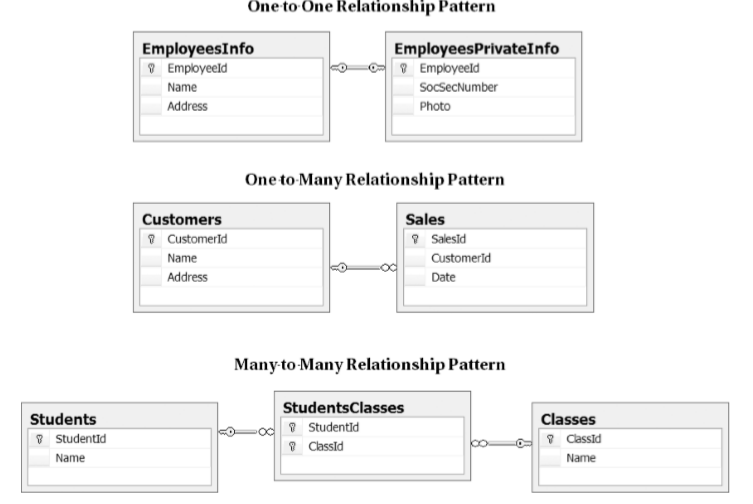
* A data warehouse is **a collection of data designed for the easy extraction** of information
* It can be in **any form**, including a series of **text ﬁles**, but most often it is **a relational database**
* Think of a data warehouse as a **reporting database**

## Data Mart

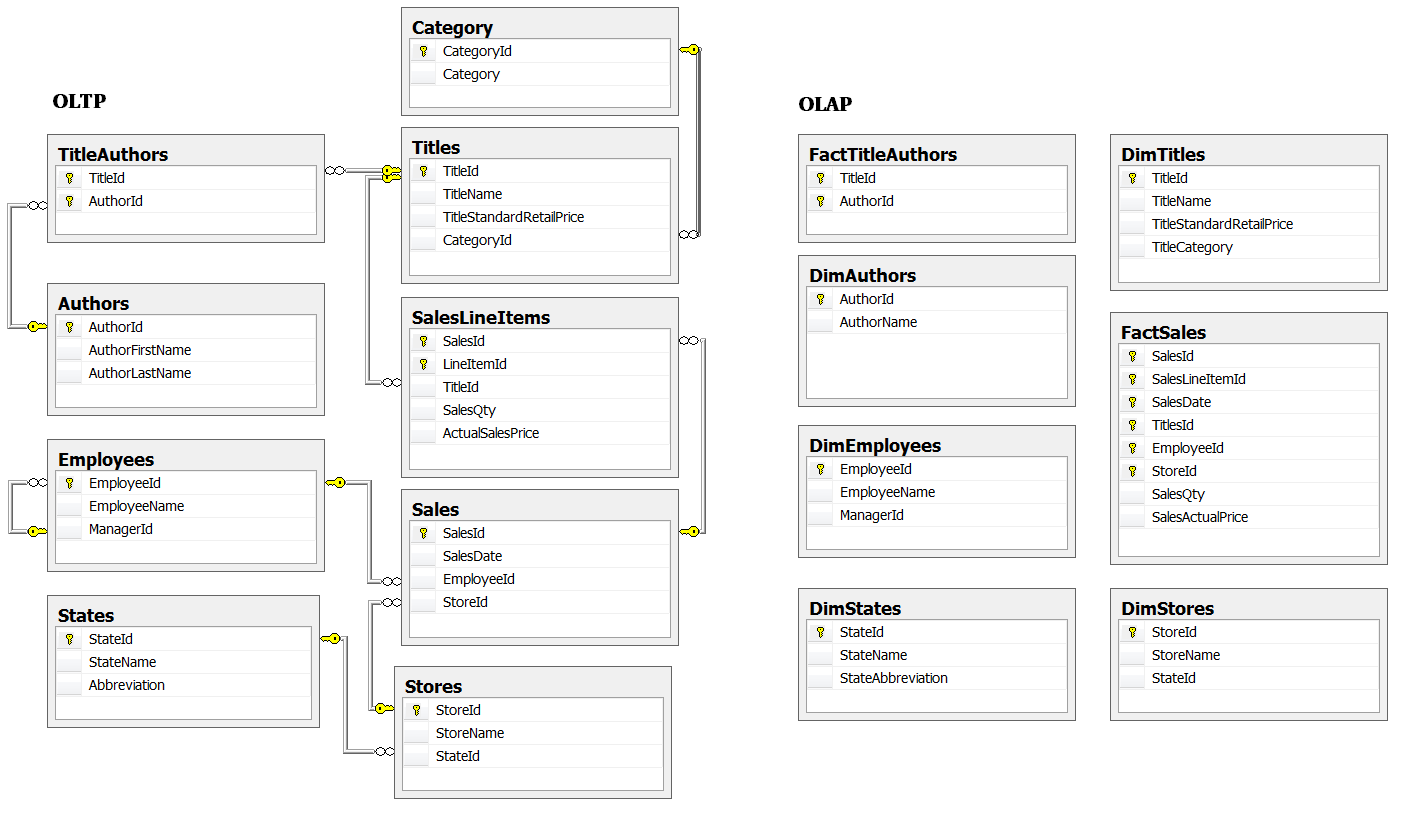
* A data mart is also a **collection of data for a given topic**.
* **More speciﬁc** than that of a data warehouse.
* **Created for a particular process**, such as a **sales event or taking inventory**
* **Designed around departments** within the company
* Typically, **better** off deﬁning the data mart based on a **process, not a department**

## Design Patterns of Data Warehouse Tables

* The three basic table patterns in a relational database are; **one-to-one, one-to-many, and many-to-many**



* Data Warehouse Design **(OLAP) design is similar to the OLTP**, but its focus is different
* The focus is on **report performance and simplicity**
* It should be **easy to understand** and consist of a **minimal set of tables** wherever possible
* Using a **bottom-up approach** is associated with Ralph Kimball’s design methods and **focuses on a particular process**, such as sales, and start building from there.
* **Make things consistent** so that **additional processes can be added later**, in what Kimball refers to as a bus architecture.



# Create an ETL Process

**E**xtracting the **data** from its **original** location, **t**ransforming **the data to be consistent with your new data** warehouse design, and **l**oading **the data into** the new **data warehouse** location, is known as **ETL processing**. You create an ETL process using a **combination** **of** programming, such as **SQL, Java, Python, C#**, etc., and/or **specialized tools,** such as SQL Server's Integration Server (**SSIS**).

Often, you **use SQL code to do the bulk of your ETL work**, since it is specifically designed to manipulate large amounts of data. Here is an Example of some simplistic ETL code (In real life this code would be much more complex).

-- Step 1) Code used to Clear tables (Will be used with SSIS Execute SQL Tasks)

Use DWPubsSales;

--1b) Clear all tables data warehouse tables

Delete From dbo.FactSales;

Delete From dbo.DimTitles;

Delete From dbo.DimStores;

Go

-- Step 2) Code used to fill tables (Will be used with SSIS Data Flow Tasks)

Insert into DimStores

Select

[StoreId] = Cast(stor\_id as nChar(4))

, [StoreName] = Cast(stor\_name as nVarchar(100))

, [StoreState] = Cast(state as nChar(2))

From pubs.dbo.stores;

Go

Insert into DimTitles

Select

[TitleId] = t.title\_id

, [TitleName] = Cast(t.title as nvarchar(100))

, [TitleType] = Cast(IIF([type] = 'UNDECIDED', 'undecided', [type] ) as nvarchar(100))

, [TitlePrice] = [price]

, [PublisherID] = p.pub\_id

, [PublisherName] = p.pub\_name

, [PublishedDate] = T.pubdate

From [Pubs].[dbo].[Titles] as T

Join [Pubs].[dbo].[Publishers] as P

On T.[pub\_id] = P.[pub\_Id]

WHERE T.price is NOT NULL;

Go

Insert into FactSales

Select

[OrderNumber] = Cast(ord\_num as nVarchar(50))

, [OrderDateKey] = Cast(ord\_date as Date)

, [TitleID] = title\_id

, [StoreID] = stor\_id

, [SalesQuantity] = qty

From Pubs.dbo.Sales as s;

--Verify the tables are filled

Go

Select \* From DimStores;

Select \* From DimTitles;

Select \* From FactSales;

Go

## Creating the ETL Stored Procedures

You would next create stored procedures that to hold your ETL code like this:

Create Procedure pETLDimStores

/\* Author: RRoot

\*\* Desc: Processes DimStores ETL

\*\* Change Log: When,Who,What

\*\* 2017-01-01,RRoot,Created Sproc.

\*/

As

Begin

Declare @RC int = 0;

Begin Try

Begin Transaction

-- ETL Transaction Code --

-- Step 1) Clear Table

Delete From DimStores;

-- Step 2) Fill Table

Insert into DimStores

Select

[StoreId] = Cast(stor\_id as nChar(4))

, [StoreName] = Cast(stor\_name as nVarchar(100))

, [StoreState] = Cast(state as nChar(2))

From pubs.dbo.stores;

Commit Transaction

Set @RC = +1

End Try

Begin Catch

Rollback Transaction

Print Error\_Message();

Set @RC = -1;

End Catch

Return @RC;

End

go

----Testing Code:

Declare @Status int;

Exec @Status = pETLDimStores;

Select @Status;

Select \* From DimStores;

go

## ETL Applications

If I were **using** a ETL application like **Microsoft's SSIS**, I would **use similar code** to configure an SSIS Package file, but using SQL code **inside** of **Execute SQL Tasks** and **Data Flow Tasks**.

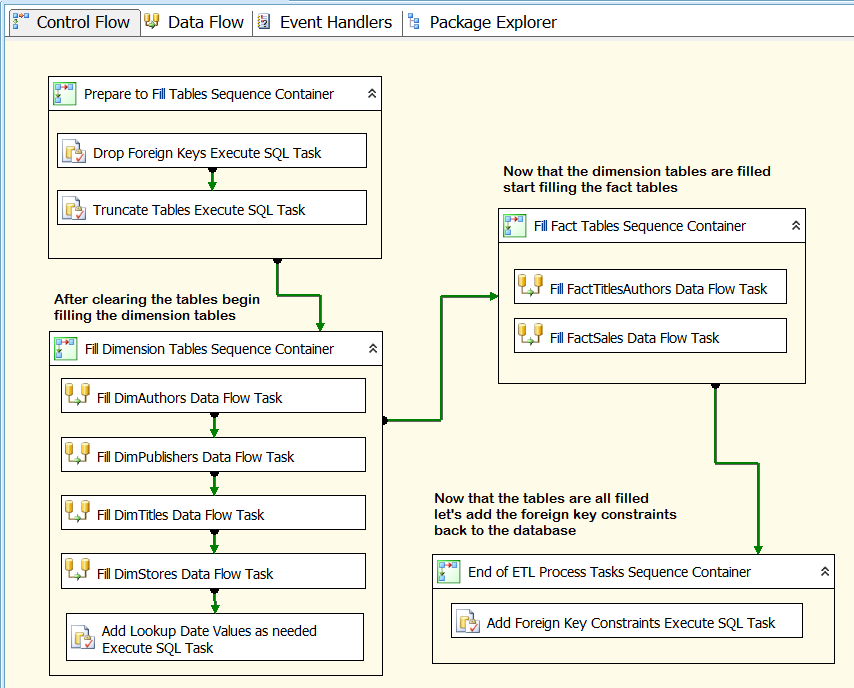


Figure: An ETL SSIS package

"Integration Services includes a rich **set** **of built-in tasks and transformations**; **tools for constructing packages**; and the Integration Services service for running and managing packages. You **can use** the graphical Integration Services tools to create solutions **without writing** a single line of **code**; **or** you can program the **extensive** Integration Services object model to create packages programmatically and **code custom tasks** and other package objects." (<https://docs.microsoft.com/en-us/sql/integration-services/sql-server-integration-services>, 2017)

## Demonstration – Creating a BI Solution

In this demonstration, your instructor will show how to create a very simple Business Intelligence (BI) solution.

# Creating Cubes, Tabular Structures, or Non-SQL databases

Microsoft SQL Server includes **two additional** **database options**; both are a part of SQL Server Analysis Services (SSAS). Both the SSAS cube databases and the SSAS tabular databases provide **increased reporting performance**. You can think of these **cubes and tabular structures as a set of many report tables combined into a single reporting object**. **Cube** databases store data on a hard drive, like SQL Servers relational databases, but in a **file structure optimized** for faster data reads. **Tabular** structures, store their data **in memory** and in a **highly compressed** format. Reporting applications pulling data from a in memory structure will be faster than off a hard drive, even with file optimization, but are **limited by the amount of RAM** the Server has access to.

**Non-SQL** databases, store their data on **several hard drives** at once which **spreads out the workload** involved with locating and retrieving report data. This can greatly increase **performance when working with BIG DATA sets** and has become a **popular** option. Like, cubes and tabular, these databases are not designed using the relational or dimension model, providing the benefit of a **less constrained, and perhaps less organized**, storage of data.

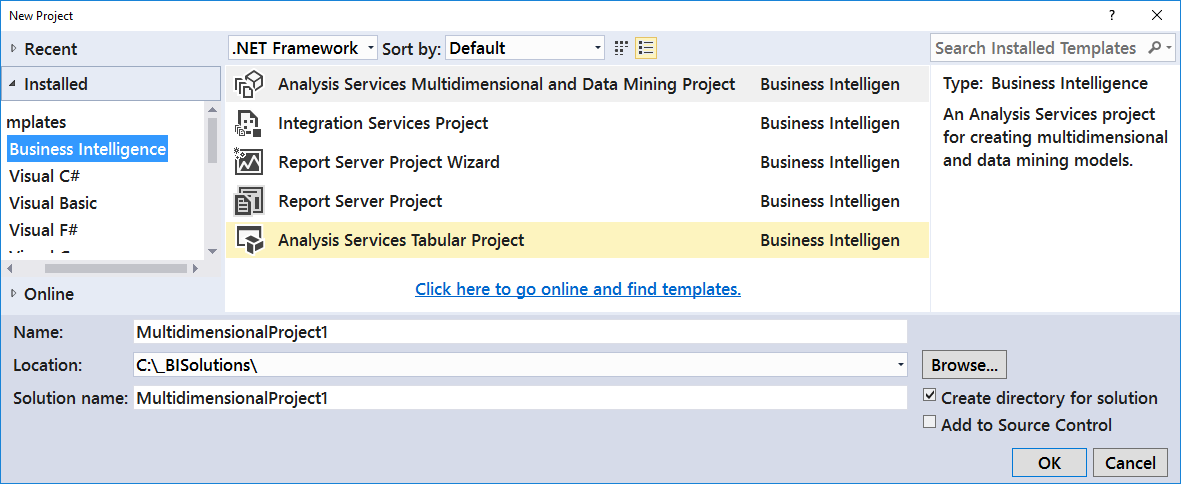


Figure: Creating a SSAS projects in Visual Studio

# Creating Reports

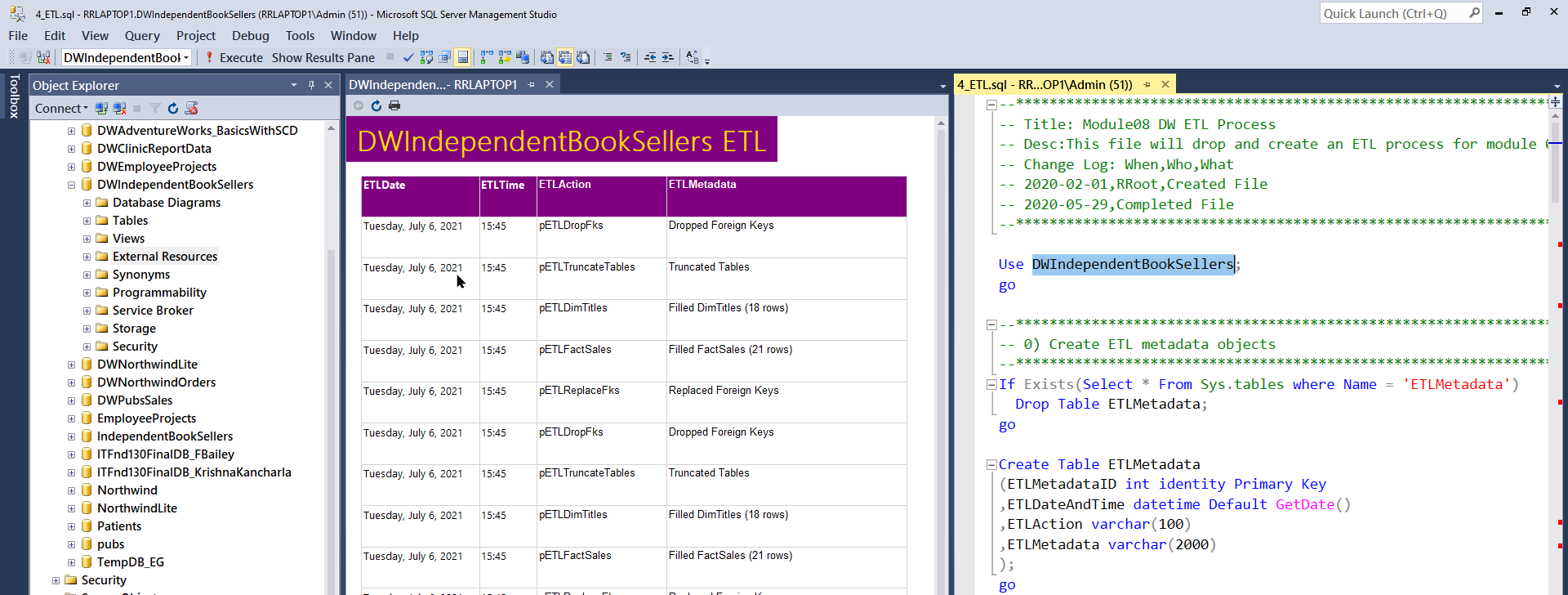
Once you have your data loaded into a data warehouse, cube, tabular structure, or non-SQL database, you need to create **preliminary reports**. Later, users of your solution, will create their own reports, but before you let them do so, these ﬁrst reports are **used for testing and evaluating the accuracy and performance** of your BI reporting.

Microsoft technologies to create your BI reports, including Excel, SQL Server Reporting Services (SSRS), and Power BI, but there are **many** other companies that make **reporting software**, such as Tableau.

## Demonstration – Creating a Management Report

In this demonstration, we look at an **creating a simple management report using Report Builder and SSMS**

1. Open the Report Builder.
2. Create a Blank Report.
3. Create a Data Source.
4. Create an Embedded DataSet.
5. Insert and configure a Table control.
6. Test the report.
7. View the report from Management Studio.



# Test and Tune the Solution

You need to **test** reports for **accuracy, visual consistency, and performance**. The **most important** of the three **is accuracy**. If the reports are slow or do not look professional, it is indeed cause for concern, but if your reports are inaccurate, your entire BI reporting solution will fail!

In **MS SQL Server**, people have been using **SQL Profiler** to test and turn SQL code being executed.

<https://docs.microsoft.com/en-us/sql/tools/sql-server-profiler/sql-server-profiler>

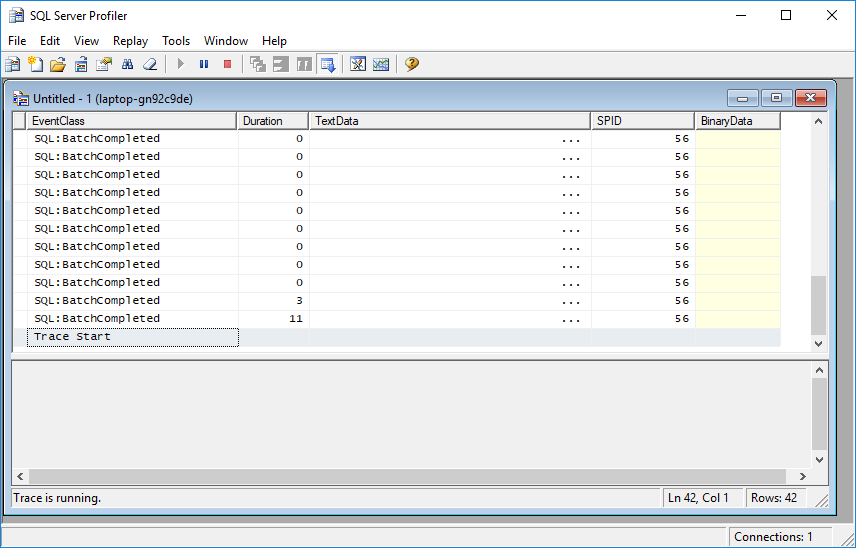


Figure: SQL Profiler user interface

While Profiler will continue for a while, MS is **replacing Profiler with Extended Events**.

“XE will replace the SQL Profiler in the future versions. By the moment, SQL Server includes Profiler and XE.

The XEs is a feature included in SQL Server 2008. It is a lighter option that consumes less resources than the Profiler. It also can monitor more events than the Profiler. For example, you can monitor Azure, Column Store Events, InMemory OLTP, AlwaysOn Events. In fact, Profiler is not adding new events since the SQL 2008. All the new features are available to be monitored only in XE and not in Profiler.” (<https://www.sqlshack.com/is-this-the-end-of-sql-profiler/> , 2017)

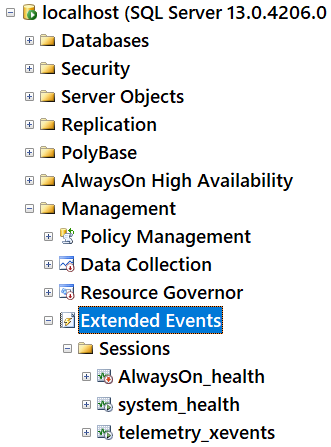


Figure: The Extend Events node in SSMS Object Explorer

Tip: For more information on Extended Events see: <https://docs.microsoft.com/en-us/sql/relational-databases/extended-events/quick-start-extended-events-in-sql-server>

# Approve, Release and Prepare

At the end of the solution development cycle, you need **to package and deploy your documents, scripts, databases, and reports**. You also need to create user documentation, as well as train your users to use your newly developed BI solution.

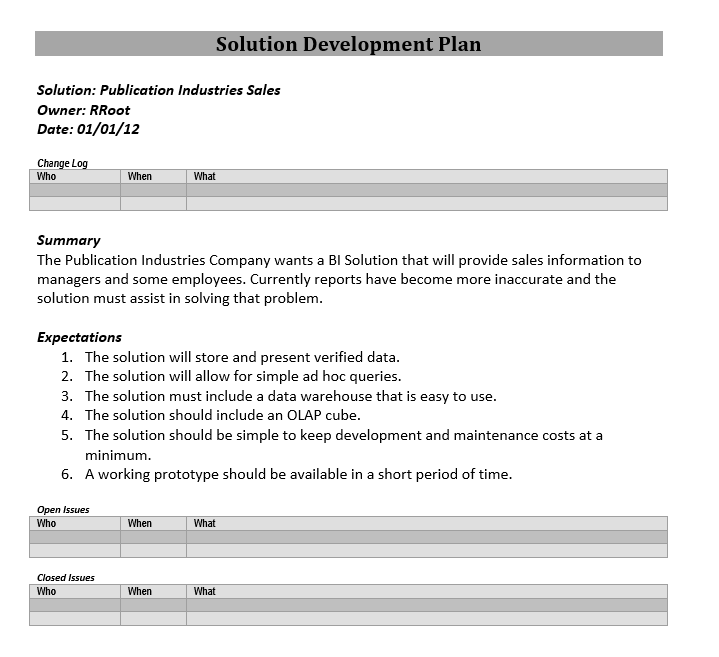


Figure: A formal solution development document