Hands-on Lab 4.2:
Metrics for Continuous Delivery in TFS

Microsoft® patterns & practices proven practices for predictable results

Visual Studio ALM Rangers

Table of Contents

Objectives

This HOL demonstrates how to create custom reports in Microsoft Team Foundation Server (TFS) for some of the key metrics used with continuous delivery release pipelines. These metrics are

- Cycle time
- Mean Time To Recovery (MTTR)
- Mean Time Between Failures (MTBF)

The meaning and purpose of these metrics is covered in Chapter 5 of Building a Release Pipeline with Team Foundation Server 2012.

The lab is based on the **MSF for Agile Software Development 2013** TFS process template. The reports that are built in the lab may work with minor or no changes in earlier versions of the template, but they have not been tested. Two other default templates in TFS, Microsoft Visual Studio Scrum 2013 and MSF for CMMI Process Improvement 2013, have not been tested either, but you may find that you can adapt the techniques demonstrated in this lab to work with them.

Prerequisites

Here are the prerequisites for completing this lab.

- You need to create a TFS team project that is based on the **MSF for Agile Software Development 6.2**. Neither Team Foundation Service nor TFS Express will work because they do not have the required reporting capabilities.

  Note: An empty team project is a better starting point if you want the sample data to begin from a known starting point, but the reports will work with any project that is based on the MSF for Agile Process Template. You do not need to use the TreyResearch solution.

- You need **Microsoft Excel 2007** or later.
- Your user account should be a member of the **TfsWarehouseDataReader** role in the **Tfs_Warehouse** database. In order to add a user, connect as an administrator to the database. Read Grant Permissions to view or create SSRS reports in TFS for more information. Look for the section named **Add report authors to database roles** and follow steps 1 through 4 (steps 5 through 7 are not required for this lab).
- Optionally, if you want to run the SQL queries described in this lab, you will need **SQL Server Management Studio**.

Note: This lab uses Excel 2013. The Brian Keller VM uses Excel 2010. If you are using this environment, some of the views will be slightly different.

The first exercise in this lab is optional because it only adds some sample data that is used to test to see if the reports work properly. If you want to complete this exercise, you also need the following:

- You need **Visual Studio 2012** in order to use the **witAdmin.exe** command line tool and the Developer Command Prompt for Visual Studio 2012.
- Your user account must have the **Manage process template** permission for the TFS project collection that contains the TFS team project. For more information, see Collection-level permissions in Team Foundation Server Permissions on MSDN.

NOTE: If you are in the Project Collection Administrators security group in TFS (see Collection-level permissions), you already have all the required permissions for all the activities described in this lab.

The sample data used in the first exercise of this lab is in the **Lab04-Monitoring\Start-Lab** folder. You run this lab on your local computer.

Time

You should be able to complete all of the exercises in this lab in approximately 40 minutes.

Exercise 1: Preparing the Sample Data for the Reports (Optional)

In this exercise you create some work items in the target team project in order to have some meaningful data.

This exercise is optional, as it's only purpose is to provide data for the reports. If you already have meaningful data in a TFS project, in the form of **User Story** and **Bug** work items that are in the **Closed** state, you can skip this exercise and go to Exercise 2. Run the reports against your TFS project.

NOTE: Don't perform this exercise with a TFS team project that is handling actual user requests in a production environment. The changes you make to the process template during the exercise will make the team project stop working properly until you finish. Furthermore, you fill the team project with dummy data that is only useful for the purposes of this lab.

Task 1: Set Up the Process Template to Receive the Test Data

The reports you will build get their information from the following fields in the **User Story** and **Bug** work items.

- **Microsoft.VSTS.Common.ActivatedDate**
- **Microsoft.VSTS.Common.ClosedDate**

You will use the following fields to filter the data.

- **System.State**
- **System.Reason**

In order to prepare meaningful data to test the reports, you need to write information into these fields. Specifically, you need to create:

- **User Stories** in the **Closed** state with the reason **Acceptance tests pass**, and that have meaningful values for the activated and closed dates.
- **Bugs** in the **Closed** state with the reason **Verified**, and that have meaningful values for the activated and closed dates.

Normally, in the **MSF for Agile** process template, these fields and values are either marked as read only, modified by TFS as a result of specific state transitions, or are only available after going through several steps in the workflow that handles work items. To insert many work items with specific values in a single operation, you must configure the process template so that you can temporarily bypass these rules.

1. Log on to the machine where you have **Visual Studio** or **Team Explorer** installed, as the user who has the **Manage process template** permission over the TFS project collection that contains the TFS team project.

2. Create a new Agile Team project. In this lab, the name used is **Lab04-Metrics**.

3. Open a Visual Studio Developer command prompt. Use the witAdmin.exe command line tool to get a copy of the definition of the **User Story** and **Bug** work item types. You will modify them to bypass the rules. Run the following commands.

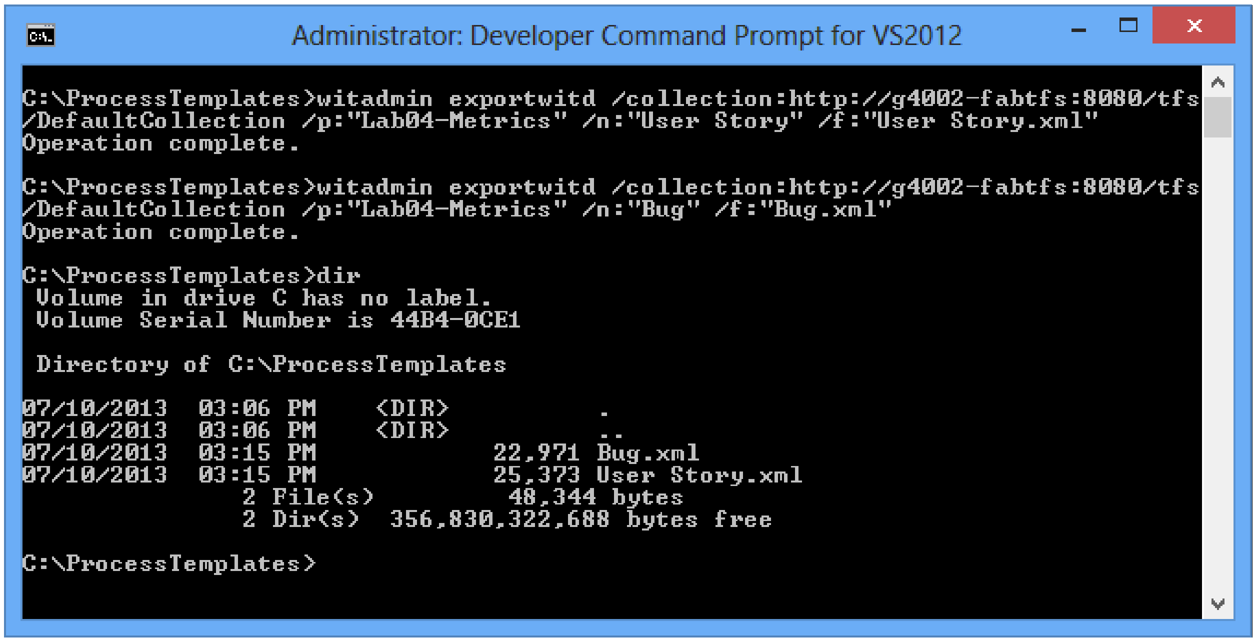
CMD

```
witadmin exportwitd
/collection:http://tfsServerURL:portNumber/virtualDirectory/CollectionName
/p:"Name of the Team Project" /n:"User Story" /f:"User Story.xml"
```

CMD

```
witadmin exportwitd
/collection:http://tfsServerURL:portNumber/virtualDirectory/CollectionName /p:
"Name of the Team Project" /n:"Bug" /f:"Bug.xml"
```

There should be two XML files, **User Story.xml** and **Bug.xml**, copied to your local folder.



```
C:\ProcessTemplates>witadmin exportwitd /collection:http://g4002-fabtfs:8080/tfs
/DefaultCollection /p:"Lab04-Metrics" /n:"User Story" /f:"User Story.xml"
Operation complete.

C:\ProcessTemplates>witadmin exportwitd /collection:http://g4002-fabtfs:8080/tfs
/DefaultCollection /p:"Lab04-Metrics" /n:"Bug" /f:"Bug.xml"
Operation complete.

C:\ProcessTemplates>dir
 Volume in drive C has no label.
 Volume Serial Number is 44B4-0CE1

 Directory of C:\ProcessTemplates

07/10/2013  03:06 PM    <DIR>          .
07/10/2013  03:06 PM    <DIR>          ..
07/10/2013  03:15 PM            22,971 Bug.xml
07/10/2013  03:15 PM            25,373 User Story.xml
               2 File(s)         48,344 bytes
               2 Dir(s)  356,830,322,688 bytes free

C:\ProcessTemplates>
```

4. Make a copy of these XML files by running the following commands. (You will need to revert the template later.)

CMD

```
copy "User Story.xml" "User Story - Bypass Rules.xml"
copy "Bug.xml" "Bug - Bypass Rules.xml"
```

5. Open the file **User Story - Bypass Rules.xml** for editing. Type the following command.

CMD

```
notepad "User Story - Bypass Rules.xml"
```

6. Locate the following section of the file.

XML

```xml
        <TRANSITION from="" to="New">
          <REASONS>
            <DEFAULTREASON value="New" />
          </REASONS>
        </TRANSITION>
```

Replace it with the following XML code.

XML

```xml
        <TRANSITION from="" to="Closed">
          <REASONS>
            <DEFAULTREASON value="Acceptance tests pass" />
          </REASONS>
        </TRANSITION>
```

You will now be able to directly create **User Story** work items that have the desired state and reason, and the activated and closed dates will be writable as well. Save the file and close the editor.

7. Open the file **Bug - Bypass Rules.xml** for editing. Type the following command.

CMD

```
notepad "Bug - Bypass Rules.xml"
```

8. Locate the following section in the file.

XML

```xml
        <TRANSITION from="" to="Active">
          <REASONS>
            <DEFAULTREASON value="New" />
            <REASON value="Build Failure" />
          </REASONS>
          <FIELDS>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedBy">
              <ALLOWEXISTINGVALUE />
              <COPY from="currentuser" />
              <VALIDUSER />
              <REQUIRED />
            </FIELD>
            <FIELD refname="Microsoft.VSTS.Common.ActivatedDate">
              <SERVERDEFAULT from="clock" />
            </FIELD>
          </FIELDS>
        </TRANSITION>
```

Replace it with the following XML code.

XML

```xml
        <TRANSITION from="" to="Closed">
```

```
          <REASONS>
            <DEFAULTREASON value="Verified" />
          </REASONS>
        </TRANSITION>
```

You will be now be able to directly create **Bug** work items that have the desired state and reason, and the activated and closed dates will be writable as well. Save the file and close the editor.

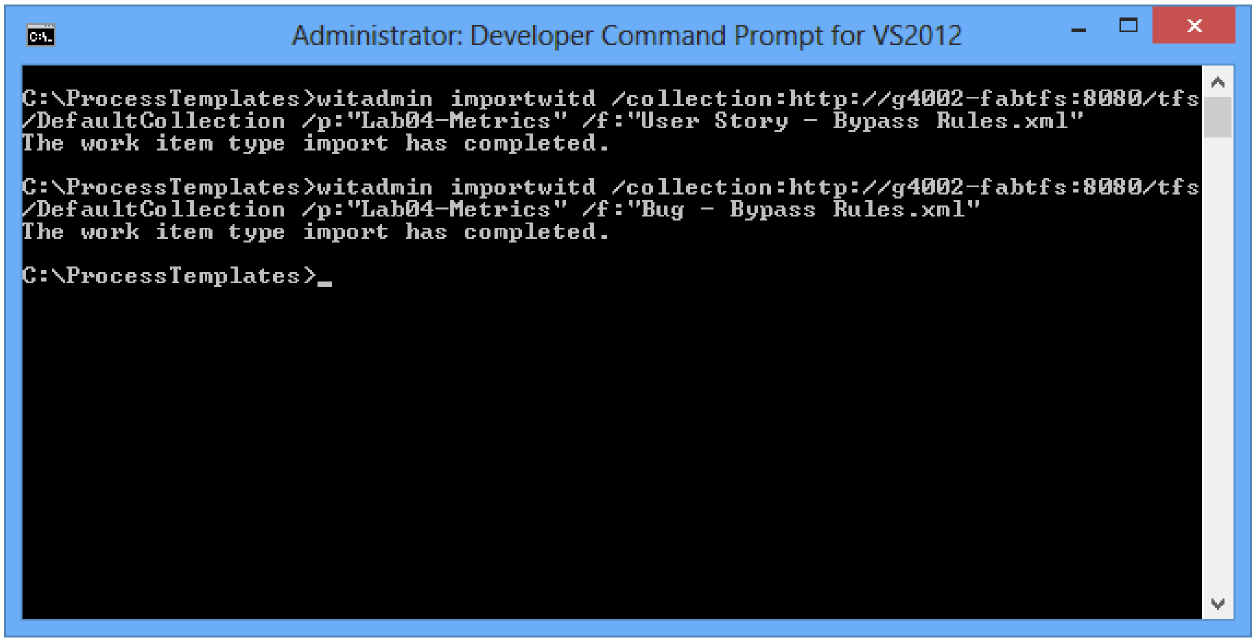
9. Upload the new work item type definitions with the bypassed rules to the TFS team project with the following commands.

CMD

```
witadmin importwitd
/collection:http://tfsServerURL:portNumber/virtualDirectory/CollectionName /p:
"Name of the Team Project" /f:"User Story - Bypass Rules.xml"
```

CMD

```
witadmin importwitd
/collection:http://tfsServerURL:portNumber/virtualDirectory/CollectionName /p:
"Name of the Team Project" /f:"Bug - Bypass Rules.xml"
```



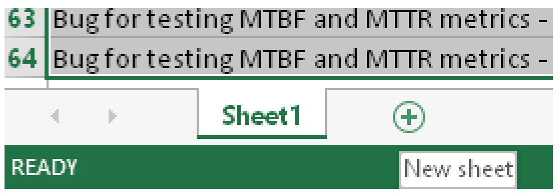
```
C:\ProcessTemplates>witadmin importwitd /collection:http://g4002-fabtfs:8080/tfs
/DefaultCollection /p:"Lab04-Metrics" /f:"User Story - Bypass Rules.xml"
The work item type import has completed.

C:\ProcessTemplates>witadmin importwitd /collection:http://g4002-fabtfs:8080/tfs
/DefaultCollection /p:"Lab04-Metrics" /f:"Bug - Bypass Rules.xml"
The work item type import has completed.

C:\ProcessTemplates>_
```

The team project can now receive new **User Stories** and **Bugs** that are in the **Closed** state, and that have the desired reasons and dates. You can leave the command prompt open because you will use it later.

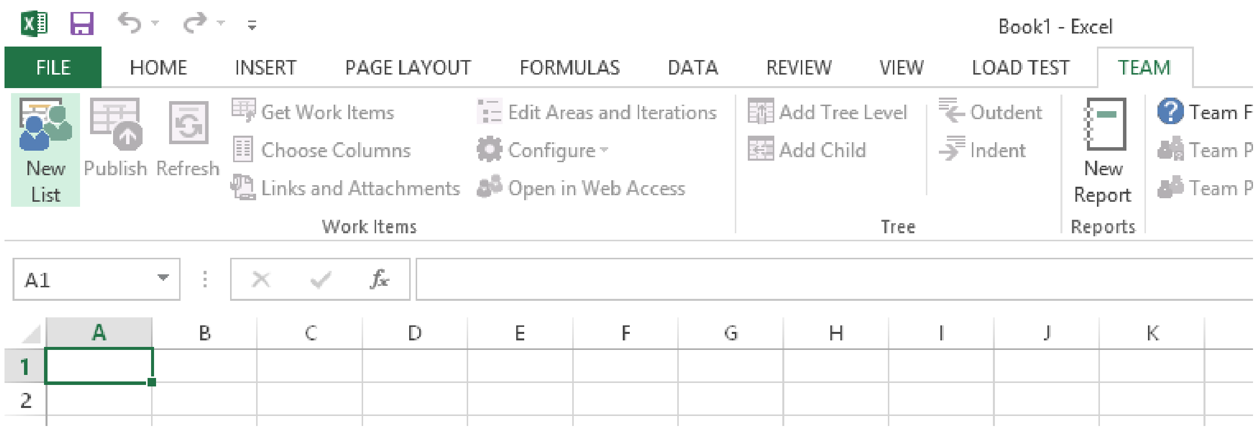
Task 2: Upload the Test Data to TFS

In this task you fill the team project with some **User Story** and **Bug** work items that will be used to test the custom reports.

1. Open the **Sample Work Items.xlsx** file that is located in the **Lab4-Monitoring\Start-Lab** folder. It contains 64 rows and 4 columns of data that represent **User Story** and **Bug** work items.
2. Click the **+** sign at the bottom of the current sheet to create a new sheet.

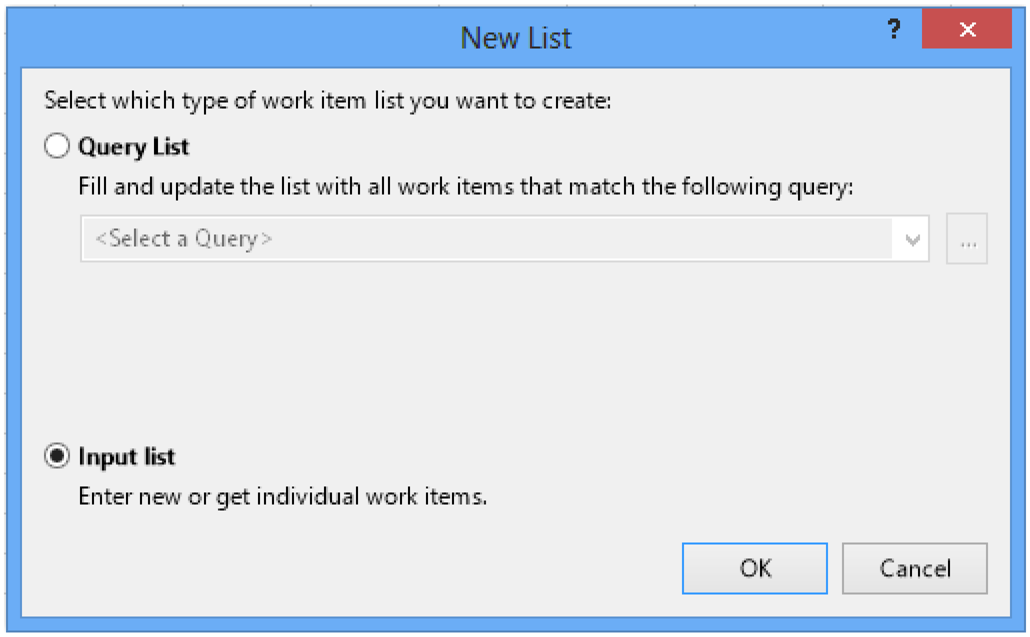

63 Bug for testing MTBF and MTTR metrics -
64 Bug for testing MTBF and MTTR metrics -
Sheet1
READY
New sheet

3. From the **TEAM** menu, click **New List**. This allows you to connect to TFS in order to import the data.



> **NOTE:** The TEAM menu isn't available unless you have installed Visual Studio Team Explorer or the TFS Client Object Model **after** Excel was installed on the machine.

4. Connect to your TFS server, project collection and team project.
5. The **New List** dialog box appears. Select **Input list**. Click **OK**.


New List

Select which type of work item list you want to create:

( ) Query List

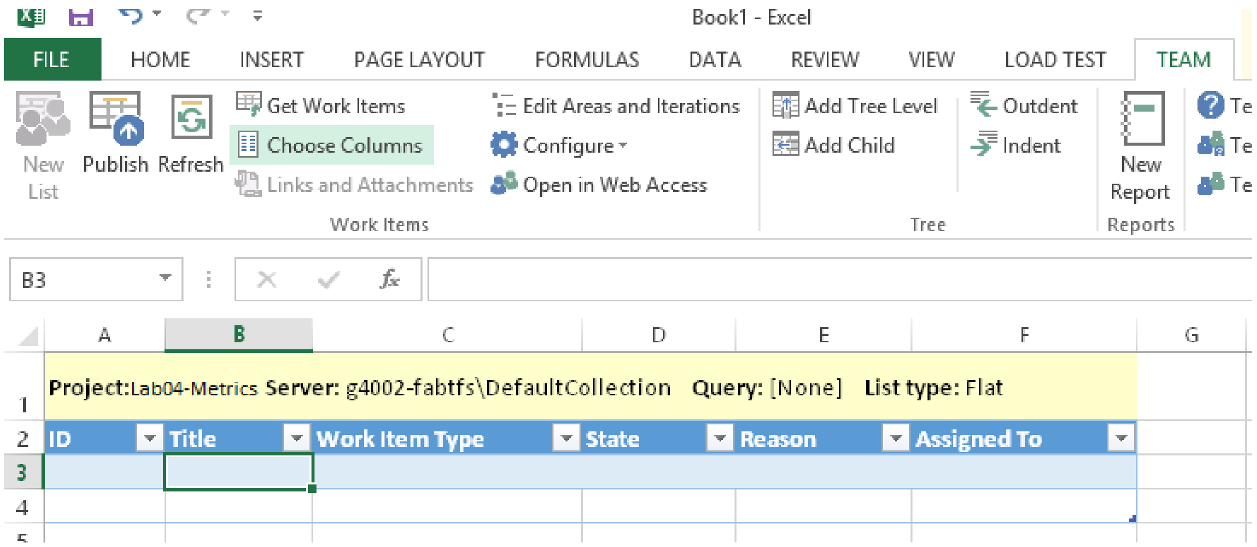
Fill and update the list with all work items that match the following query:

<Select a Query>

[✓ Input list]

Enter new or get individual work items.

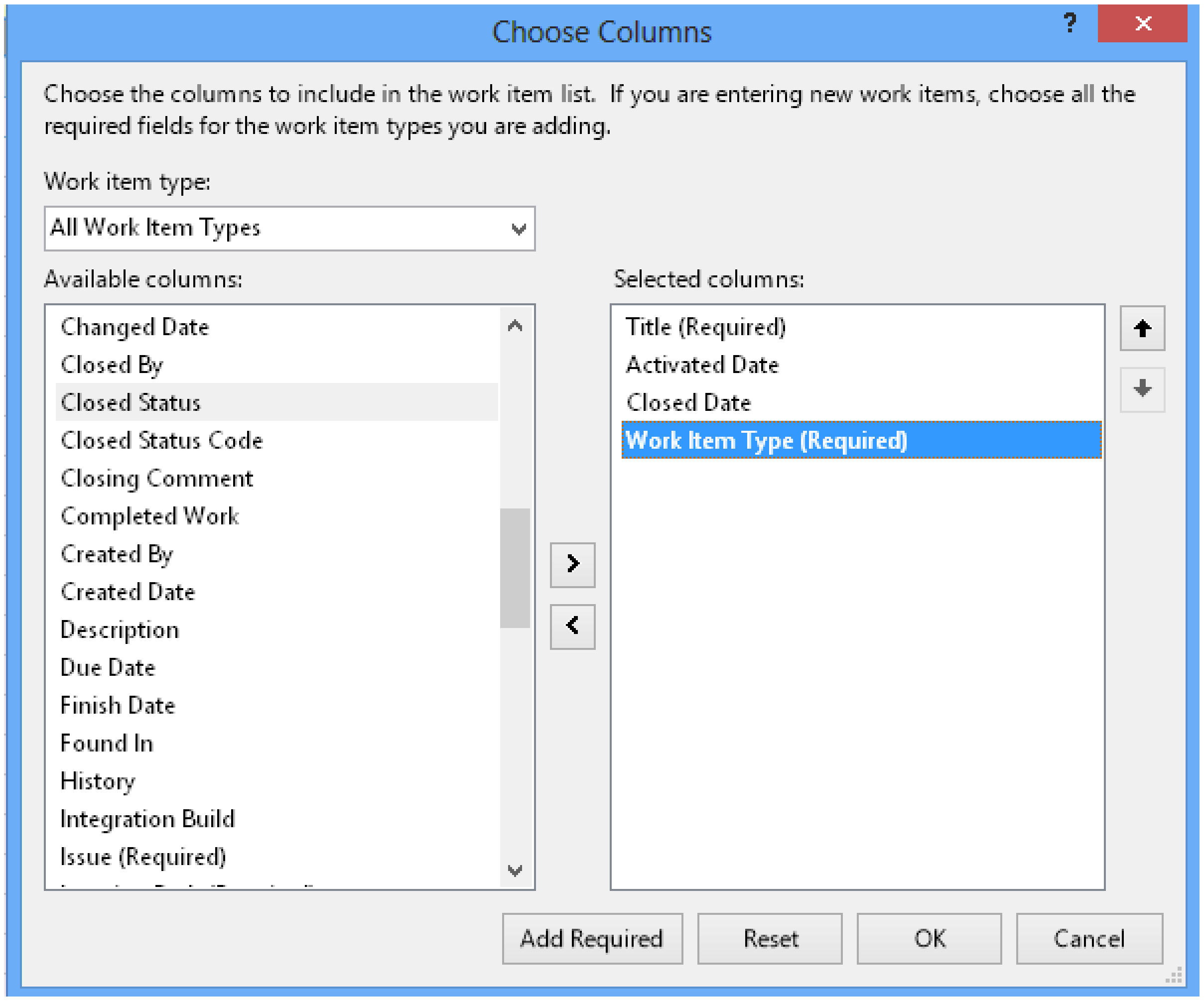
6. In the ribbon, click **Choose Columns**.



Project:Lab04-Metrics Server: g4002-fabtfs\DefaultCollection Query: [None] List type: Flat

ID	Title	Work Item Type	State	Reason	Assigned To

7. The **Choose Columns** dialog box appears. Use the up and down controls in the **Selected columns** section so that they look as they do in the screenshot. Click **OK**.


Choose Columns

Choose the columns to include in the work item list. If you are entering new work items, choose all the required fields for the work item types you are adding.

Work item type: All Work Item Types

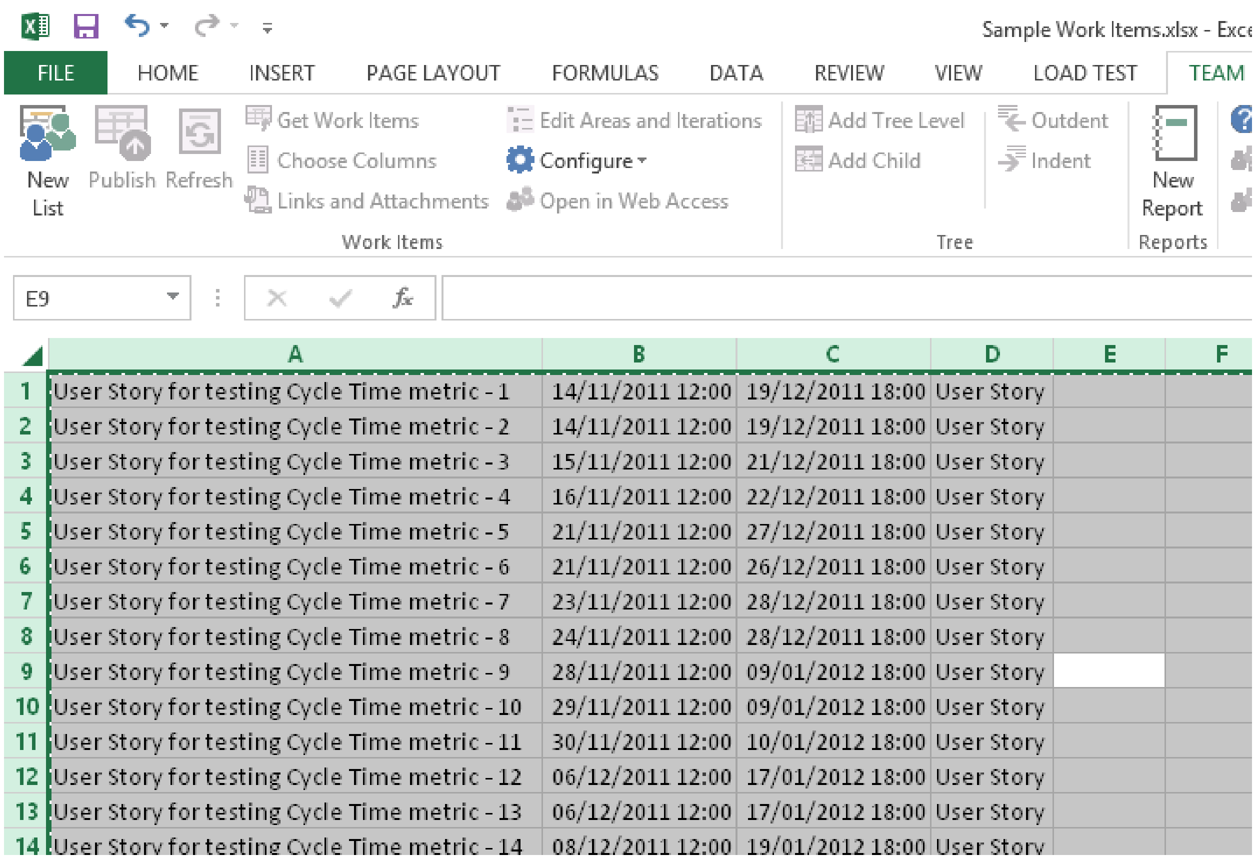
Available columns:

- Changed Date
- Closed By
- Closed Status
- Closed Status Code
- Closing Comment
- Completed Work
- Created By
- Created Date
- Description
- Due Date
- Finish Date
- Found In
- History
- Integration Build
- Issue (Required)

Selected columns:

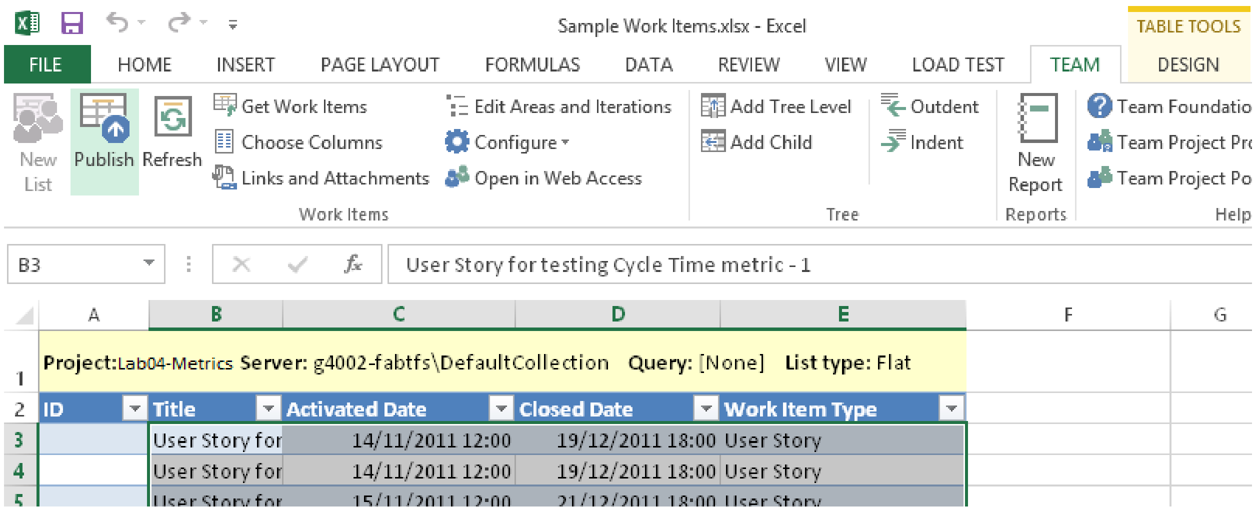
- Title (Required)
- Activated Date
- Closed Date
- Work Item Type (Required)

8. Go back to the first sheet. Select all the rows and columns that contain data, from cell **A1** to **D64**. Use **Ctrl+A** and then **Ctrl+C** so that the data is copied to the clipboard.



	A	B	C	D	E	F
1	User Story for testing Cycle Time metric - 1	14/11/2011 12:00	19/12/2011 18:00	User Story		
2	User Story for testing Cycle Time metric - 2	14/11/2011 12:00	19/12/2011 18:00	User Story		
3	User Story for testing Cycle Time metric - 3	15/11/2011 12:00	21/12/2011 18:00	User Story		
4	User Story for testing Cycle Time metric - 4	16/11/2011 12:00	22/12/2011 18:00	User Story		
5	User Story for testing Cycle Time metric - 5	21/11/2011 12:00	27/12/2011 18:00	User Story		
6	User Story for testing Cycle Time metric - 6	21/11/2011 12:00	26/12/2011 18:00	User Story		
7	User Story for testing Cycle Time metric - 7	23/11/2011 12:00	28/12/2011 18:00	User Story		
8	User Story for testing Cycle Time metric - 8	24/11/2011 12:00	28/12/2011 18:00	User Story		
9	User Story for testing Cycle Time metric - 9	28/11/2011 12:00	09/01/2012 18:00	User Story		
10	User Story for testing Cycle Time metric - 10	29/11/2011 12:00	09/01/2012 18:00	User Story		
11	User Story for testing Cycle Time metric - 11	30/11/2011 12:00	10/01/2012 18:00	User Story		
12	User Story for testing Cycle Time metric - 12	06/12/2011 12:00	17/01/2012 18:00	User Story		
13	User Story for testing Cycle Time metric - 13	06/12/2011 12:00	17/01/2012 18:00	User Story		
14	User Story for testing Cycle Time metric - 14	08/12/2011 12:00	19/01/2012 18:00	User Story		

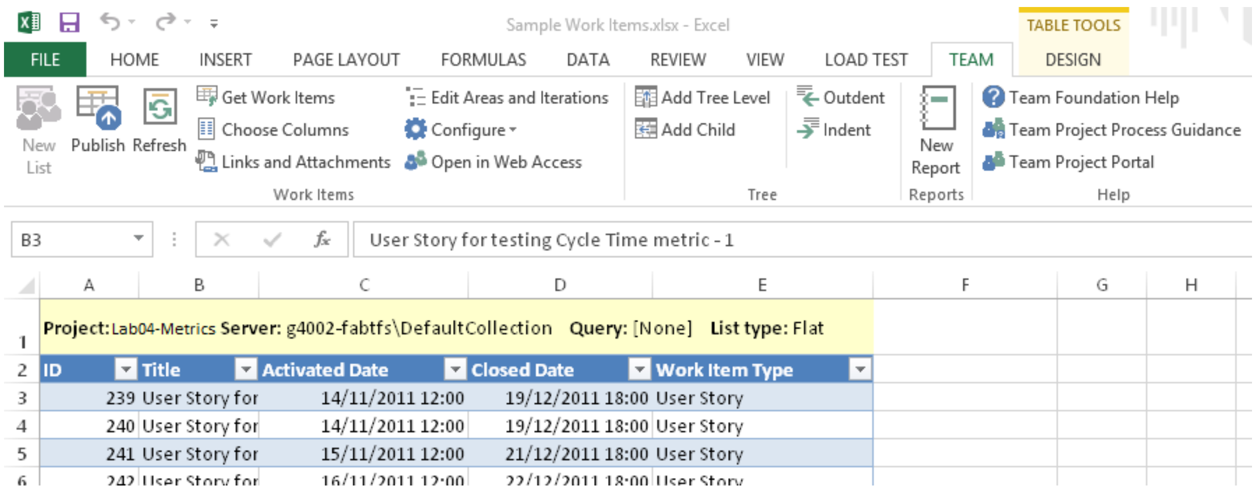
9. Go to the second sheet. Select the **B3** cell. Use **Ctrl-V** so that the data from the first sheet is pasted here. In the ribbon, click **Publish**.



Project: Lab04-Metrics **Server:** g4002-fabtfs\DefaultCollection **Query:** [None] **List type:** Flat

	ID	Title	Activated Date	Closed Date	Work Item Type
3		User Story for	14/11/2011 12:00	19/12/2011 18:00	User Story
4		User Story for	14/11/2011 12:00	19/12/2011 18:00	User Story
5		User Story for	15/11/2011 12:00	21/12/2011 18:00	User Story

10. After a few moments, the work items are created inside the TFS team project. Each one has an ID. You can close Excel without saving the changes when you are prompted. The following screenshot shows an example of the work items.



User Story for testing Cycle Time metric - 1

Project: Lab04-Metrics **Server:** g4002-fabtfs\DefaultCollection **Query:** [None] **List type:** Flat

ID	Title	Activated Date	Closed Date	Work Item Type
239	User Story for	14/11/2011 12:00	19/12/2011 18:00	User Story
240	User Story for	14/11/2011 12:00	19/12/2011 18:00	User Story
241	User Story for	15/11/2011 12:00	21/12/2011 18:00	User Story
242	User Story for	16/11/2011 12:00	22/12/2011 18:00	User Story

Task 3: Revert the Process Template to its Original State

In this task you return the process template to its original state.

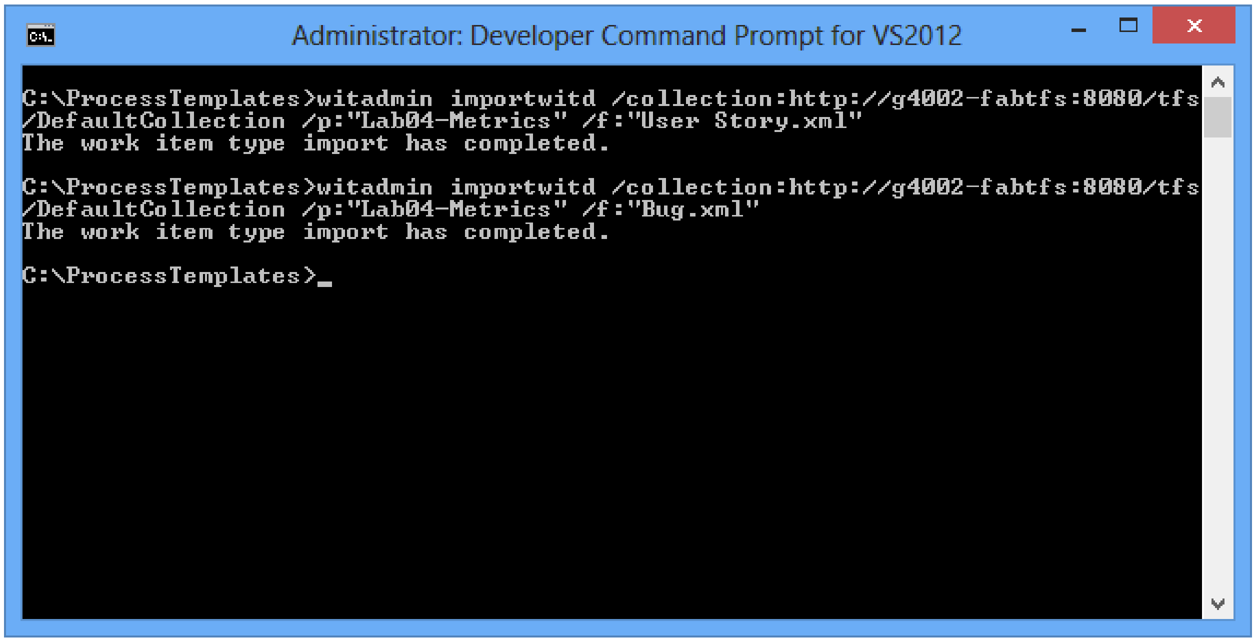
1. From the command prompt, upload the original work item type definitions to the TFS team project by typing the following commands.

CMD

```
witadmin importwitd
/collection:http://tfsServerURL:portNumber/virtualDirectory/CollectionName /p:
"Name of the Team Project" /f:"User Story.xml"
```

CMD

```
witadmin importwitd
/collection:http://tfsServerURL:portNumber/virtualDirectory/CollectionName /p:
"Name of the Team Project" /f:"Bug.xml"
```



```
C:\ProcessTemplates>witadmin importwitd /collection:http://g4002-fabtfs:8080/tfs
/DefaultCollection /p:"Lab04-Metrics" /f:"User Story.xml"
The work item type import has completed.

C:\ProcessTemplates>witadmin importwitd /collection:http://g4002-fabtfs:8080/tfs
/DefaultCollection /p:"Lab04-Metrics" /f:"Bug.xml"
The work item type import has completed.

C:\ProcessTemplates>_
```

You can delete the four XML files if you want. The process template is now back to its original state.

Exercise 2: Creating the Cycle Time Report

In this exercise you create the cycle time report.

There is an overall approach to follow when creating the reports in this lab.

Use Excel as the reporting tool. There are other options. The most relevant one is SQL Server Reporting Services (SSRS). SSRS reports have more features than Excel, such as formatting options, dynamic parameters and filters, subscriptions, and a better security model. However, this lab focuses on the reports themselves so it uses Excel, which is simpler. For more information see Create, Customize, and Manage Reports for Visual Studio ALM.

The data source is the relational store for TFS reporting, which is the **Tfs_Warehouse** database. It is the recommended source. You should not query the TFS operational databases directly. The TFS Analysis Services cube is also not needed for this lab. For more information about using the **Tfs_Warehouse** relational store, see Generate Reports Using the Relational Warehouse Database for Visual Studio ALM. You can also refer to Grant Holliday's blog post.

In general, the steps to create a report are:

1. Identify the information you need for the report.
2. Determine how to present the information.
3. Prepare the SQL query that retrieves the data from the **Tfs_Warehouse** database.

4. Create an Excel workbook with a connection to the **Tfs_Warehouse** database that runs the query and presents the results in the tabular and chart formats.

Task 1: Identify the Required information and Decide How to Present It

Cycle time is the time it takes from the moment you begin to implement a feature to the moment you release the feature.

In order to track cycle time, you need to determine the data you need and how to extract it from TFS. In the MSF for Agile process template, features are represented by the **User Story** work item so you will filter for that work item type.

Not all the user stories stored in the TFS team project are useful for tracking cycle time. Only those that have been completed and released to users are. For this lab, the criteria is to choose the user stories that are in the **Closed** state because of the **Acceptance tests pass** reason.

Note: This lab assumes that closed user stories are features that are released to users. If the criteria is different for your project, you can modify the process template to reflect this and filter on custom state and reason values. For example you could add a new **Released** state that is reached because of the reason **Deployed to production**.

You can see the workflow of states for the **User Story** work item.

You also need to filter for the team project you are using and you need two fields to calculate the cycle time. The first field is when the decision to implement the user story is made. In this lab, this is the date when the **User Story** work items become **Active**. In the MSF for Agile template, this value is automatically stored in the **Activated Date** field.

The second field is when the **User Story** work item's state becomes **Closed**. For the MSF for Agile template, this value is automatically stored in the **Closed Date** field.

You must also decide how to present the data in the final report. The calculation to be made is the time elapsed between the **Closed Date** value and the **Activated Date** value for each **User Story** work item. However, the report is more readable if the values are adjusted by units of time and by grouping.

This lab uses a day as the unit of time. The results are grouped so that they present the average cycle time per week. Showing the cycle time for every user story would produce too much information and the report would be confusing. Showing the average cycle time for the user stories completed each week makes a much more comprehensible report.

Note: There are no restrictions on the number of user stories being returned. If you have many work items in your team project and you want to keep the reports short, you can add a filter to take out the oldest user stories, or the ones that were closed before a given date.

Here is a summary of how to track the average cycle time per week**,** in days, since the beginning of the project. In order to filter the rows use the following items:

- The **User Story** work item.
- The **Closed** state.
- The **Acceptance tests pass** reason.
- The name of the team project collection
- The name of the team project

To calculate the cycle time for the rows, use the following items:

- The **Activated Date** field.
- The **Closed Date** field.

To group and present the information:

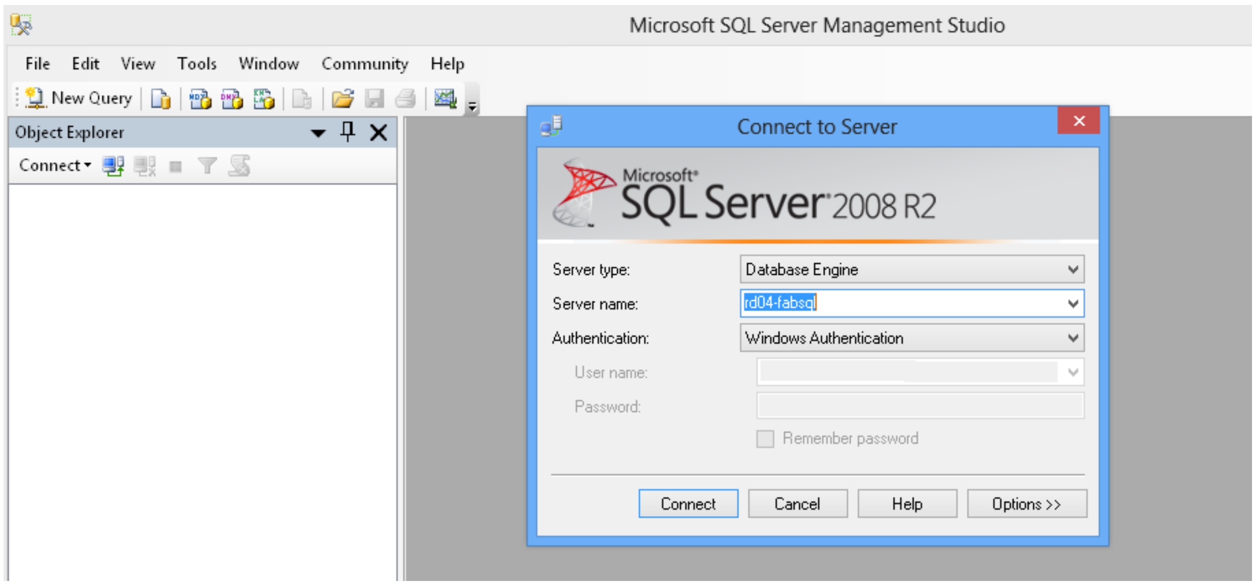
- Use **day** as the unit of time.
- Use **week** as the grouping scale.

Note: As with any other report in TFS, team members must make sure to keep the information in the work items up to date. In particular, for this report, they should write any feature to be incorporated into the product as a **User Story** work item, they should set its state to **Active** when they decide that it will be implemented, and they should set its state to **Closed** when it has been released.

Task 2: Prepare the SQL Query that Retrieves Data from the Tfs_Warehouse Database

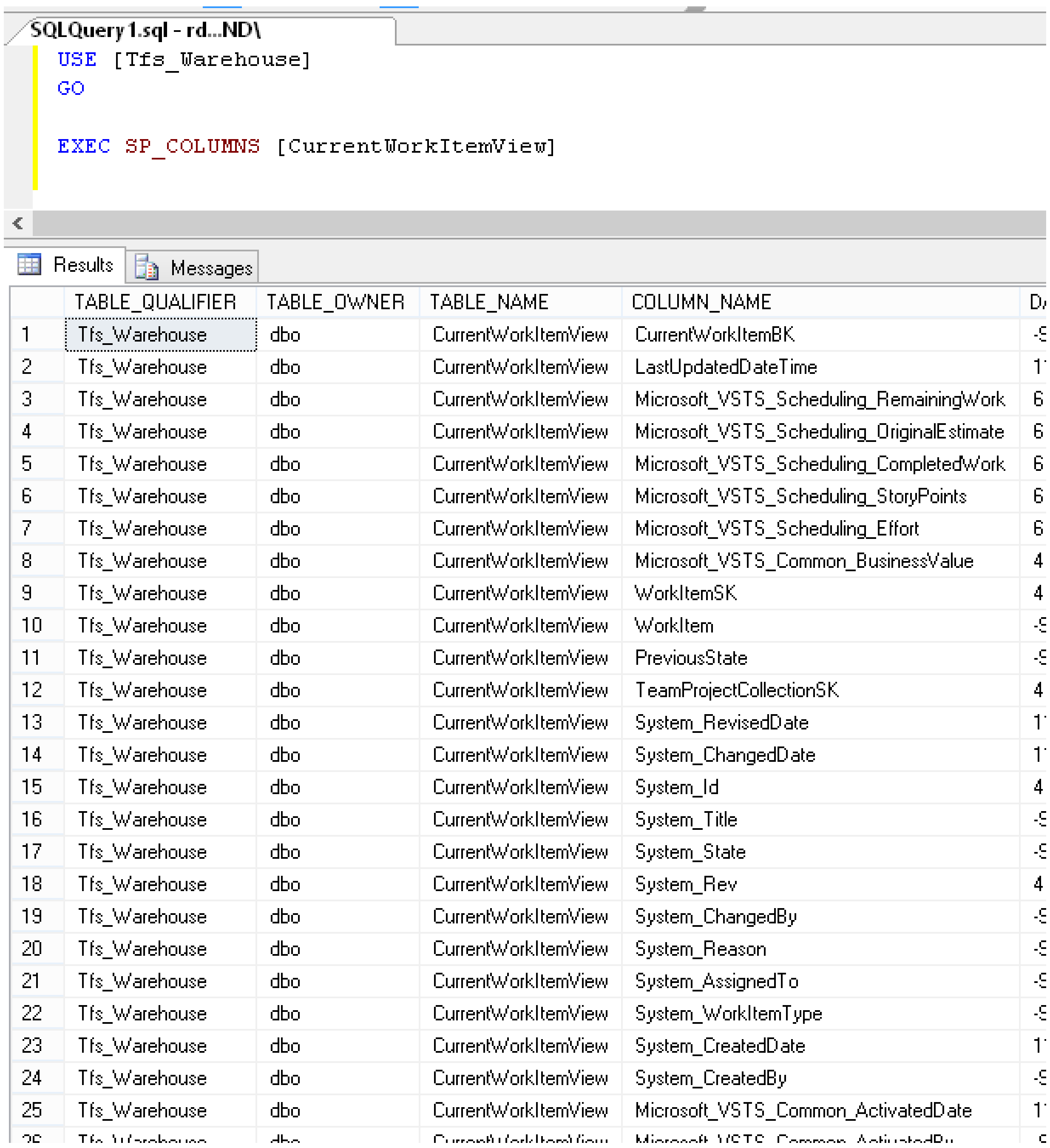
In this lab you use a SQL view named **CurrentWorkItemView** to retrieve information about **User Story** work items whose current state is **Closed**.

1. Although this step is optional, you can connect to the **Tfs_Warehouse** database from SQL Server Management Studio and run a simple query to get an idea of the database's organization. The following screenshot shows how to connect to the database.


Microsoft SQL Server Management Studio
File Edit View Tools Window Community Help
New Query
Object Explorer
Connect
Connect to Server
Microsoft SQL Server 2008 R2
Connect
Cancel
Help
Options >>

Server type: Database Engine
Server name: rd04-fabsql
Authentication: Windows Authentication
User name:
Password:
Remember password

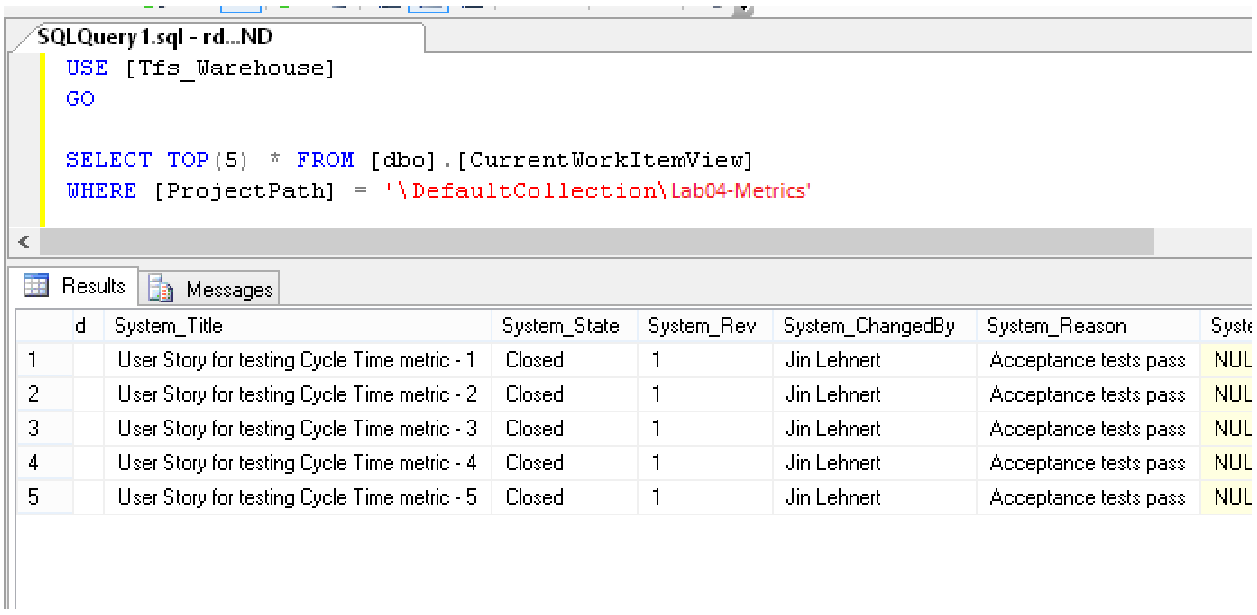
The following screenshot shows a simple query that displays the structure of the **Tfs_Warehouse** database's **CurrentWorkItemView** virtual table.



```
USE [Tfs_Warehouse]
GO

EXEC SP_COLUMNS [CurrentWorkItemView]
```

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	D
1	Tfs_Warehouse	dbo	CurrentWorkItemView	CurrentWorkItemBK	-9
2	Tfs_Warehouse	dbo	CurrentWorkItemView	LastUpdatedDateTime	1
3	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Scheduling_RemainingWork	6
4	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Scheduling_OriginalEstimate	6
5	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Scheduling_CompletedWork	6
6	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Scheduling_StoryPoints	6
7	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Scheduling_Effort	6
8	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Common_BusinessValue	4
9	Tfs_Warehouse	dbo	CurrentWorkItemView	WorkItemSK	4
10	Tfs_Warehouse	dbo	CurrentWorkItemView	WorkItem	-9
11	Tfs_Warehouse	dbo	CurrentWorkItemView	PreviousState	-9
12	Tfs_Warehouse	dbo	CurrentWorkItemView	TeamProjectCollectionSK	4
13	Tfs_Warehouse	dbo	CurrentWorkItemView	System_RevisedDate	1
14	Tfs_Warehouse	dbo	CurrentWorkItemView	System_ChangedDate	1
15	Tfs_Warehouse	dbo	CurrentWorkItemView	System_Id	4
16	Tfs_Warehouse	dbo	CurrentWorkItemView	System_Title	-9
17	Tfs_Warehouse	dbo	CurrentWorkItemView	System_State	-9
18	Tfs_Warehouse	dbo	CurrentWorkItemView	System_Rev	4
19	Tfs_Warehouse	dbo	CurrentWorkItemView	System_ChangedBy	-9
20	Tfs_Warehouse	dbo	CurrentWorkItemView	System_Reason	-9
21	Tfs_Warehouse	dbo	CurrentWorkItemView	System_AssignedTo	-9
22	Tfs_Warehouse	dbo	CurrentWorkItemView	System_WorkItemType	-9
23	Tfs_Warehouse	dbo	CurrentWorkItemView	System_CreatedDate	1
24	Tfs_Warehouse	dbo	CurrentWorkItemView	System_CreatedBy	-9
25	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Common_ActivatedDate	1
26	Tfs_Warehouse	dbo	CurrentWorkItemView	Microsoft_VSTS_Common_ActivatedBy	-9

The following screenshot shows a query that selects the top five records from the **CurrentWorkItemView**.



SQLQuery1.sql - rd...ND

```
USE [Tfs_Warehouse]
GO

SELECT TOP(5) * FROM [dbo].[CurrentWorkItemView]
WHERE [ProjectPath] = '\DefaultCollection\Lab04-Metrics'
```

Results | Messages

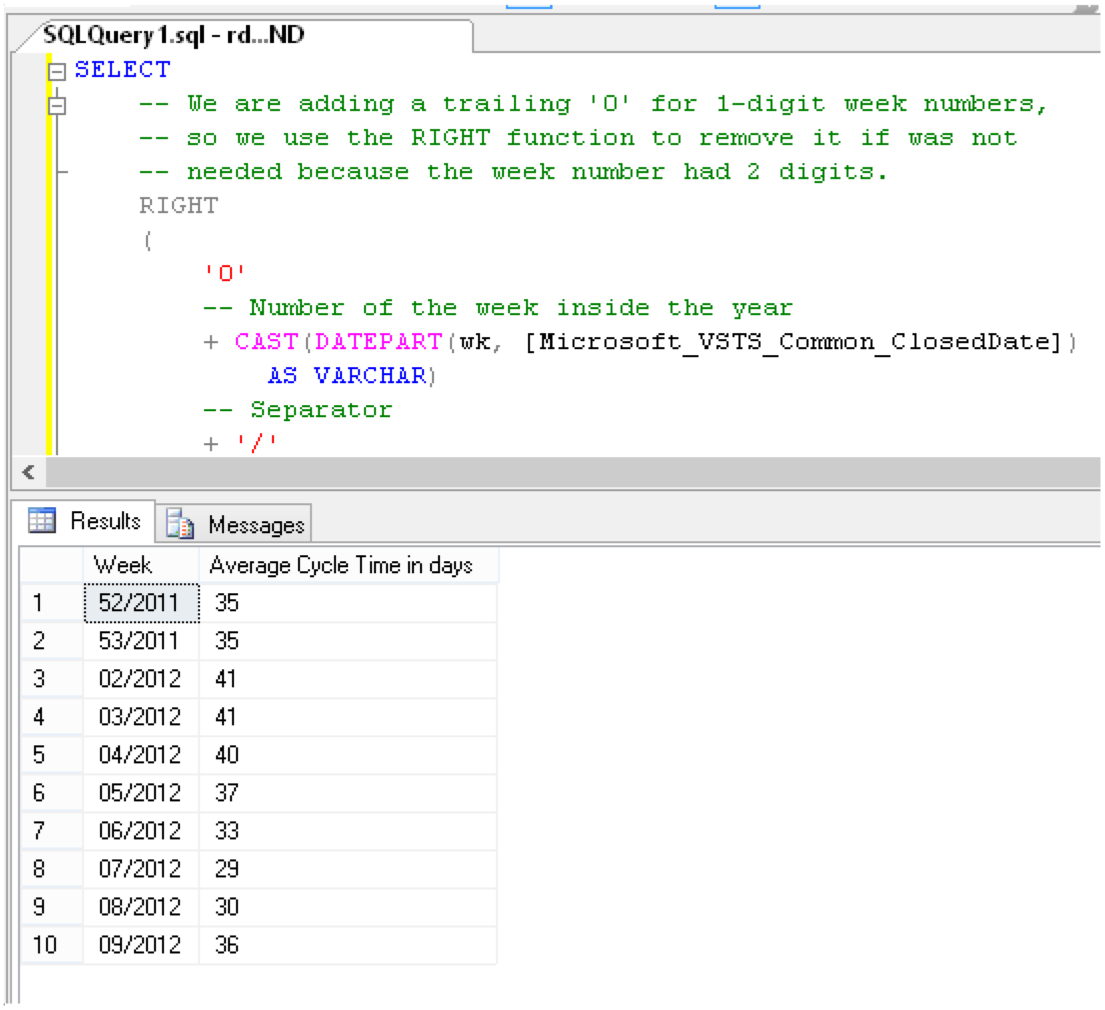
	d	System_Title	System_State	System_Rev	System_ChangedBy	System_Reason	Syste
1		User Story for testing Cycle Time metric - 1	Closed	1	Jin Lehnert	Acceptance tests pass	NUL
2		User Story for testing Cycle Time metric - 2	Closed	1	Jin Lehnert	Acceptance tests pass	NUL
3		User Story for testing Cycle Time metric - 3	Closed	1	Jin Lehnert	Acceptance tests pass	NUL
4		User Story for testing Cycle Time metric - 4	Closed	1	Jin Lehnert	Acceptance tests pass	NUL
5		User Story for testing Cycle Time metric - 5	Closed	1	Jin Lehnert	Acceptance tests pass	NUL

2. Identify the names of the columns in the **CurrentWorkItemView** that you need for the report:
   - To filter the rows you need:
     - The column name of the **User Story** work item, which is **System_WorkItemType.**
     - The column name of the **Closed** state, which is **System_State**.
     - The column name of the **Acceptance tests pass** reason, which is **System_Reason.**
     - The column name of the team project collection and the team project, which is **ProjectPath.**
   - Identify the names of the columns that you need to calculate the cycle time. They are:
     - The column name of the **Activated Date** field, which is **Microsoft_VSTS_Common_ActivatedDate**.
     - The column name of the **Closed Date** field, which is **Microsoft_VSTS_Common_ClosedDate**.
3. Write the query. Here is a commented, working version that you can use.

T-SQL

```
SELECT
        -- We are adding a trailing '0' for 1-digit week numbers,
        -- so we use the RIGHT function to remove it if was not
        -- needed because the week number had 2 digits.
        RIGHT
        (
              '0'
              -- Number of the week inside the year
              + CAST(DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate])
                     AS VARCHAR)
              -- Separator
              + '/'
              -- Year
              + CAST(DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate])
                     AS VARCHAR)
              -- Maximum length of the returned value (format: WW/YYYY)
              , 7
       ) AS [Week]
       -- Average Cycle Time in days, per each week
       ,AVG(
              DATEDIFF(dd, [Microsoft_VSTS_Common_ActivatedDate],
                     [Microsoft_VSTS_Common_ClosedDate])
              )
              AS [Average Cycle Time in days]
FROM
       [dbo].[CurrentWorkItemView]
WHERE
       [ProjectPath] = '\YOUR-PROJECTCOLLECTION NAME\TEAM PROJECT NAME'
       AND [System_WorkItemType] = 'User Story'
       AND [System_State] = 'Closed'
       AND [System_Reason] = 'Acceptance tests pass'
GROUP BY
       -- Grouping by week and year to calculate the average
       DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate]),
       DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate])
ORDER BY
       -- Ordering by year and week number
       DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate]),
DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate])
```

4. If you paste the query into SQL Management Studio and run it, you should get one row for each week when at least one user story was successfully released, along with the average cycle time in days for that week. The following screenshot shows the results if you run the query on the data that you prepared in the first exercise.



SQLQuery1.sql - rd...ND

```
SELECT
    -- We are adding a trailing '0' for 1-digit week numbers,
    -- so we use the RIGHT function to remove it if was not
    -- needed because the week number had 2 digits.
    RIGHT
    (
        '0'
        -- Number of the week inside the year
        + CAST(DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate])
            AS VARCHAR)
        -- Separator
        + '/'
```

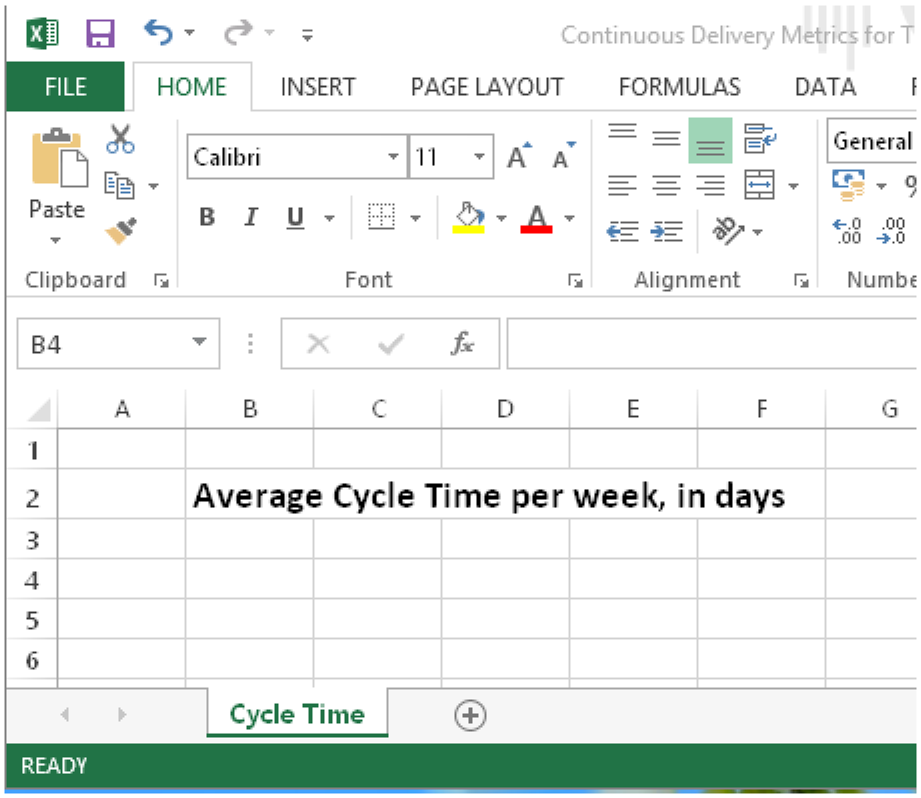
Results | Messages

	Week	Average Cycle Time in days
1	52/2011	35
2	53/2011	35
3	02/2012	41
4	03/2012	41
5	04/2012	40
6	05/2012	37
7	06/2012	33
8	07/2012	29
9	08/2012	30
10	09/2012	36

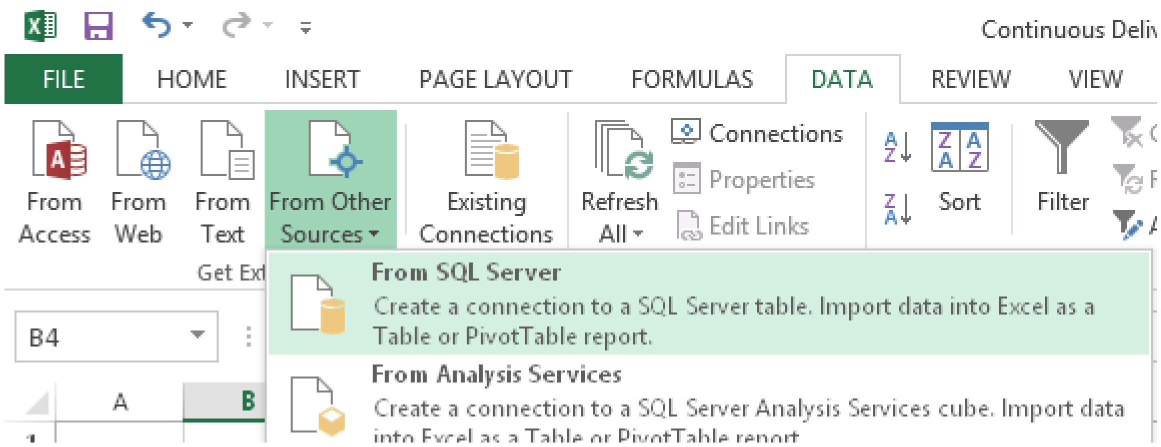
Task 3: Create an Excel Workbook to Present the Results

In this task you create an Excel workbook to present the data.

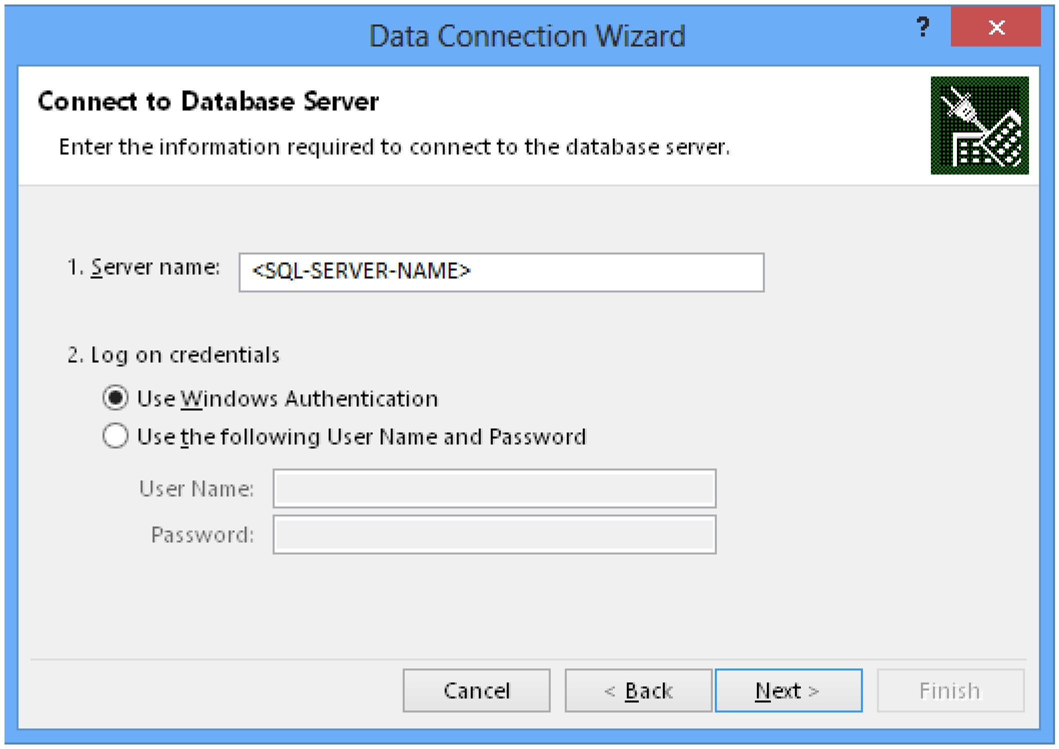
1. Create a new Excel workbook. Name it **Continuous Delivery Metrics for TreyResearch.xlsx.**
2. Change the name of the first sheet to **Cycle Time** and write a title for the report in the first cells. The following screenshot shows an example.



3. Select the cell where you want the data for the report to be inserted (for example, **B4**). From the **DATA** menu, select **From Other Sources.** Select **From SQL Server** from the drop-down list.



4. The **Data Connection Wizard** appears. In the **Server name** text box, enter the server name where the **Tfs_Warehouse** database is located. If necessary, enter the credentials. Click **Next**.


Data Connection Wizard

Connect to Database Server

Enter the information required to connect to the database server.

1. Server name: <SQL-SERVER-NAME>

2. Log on credentials

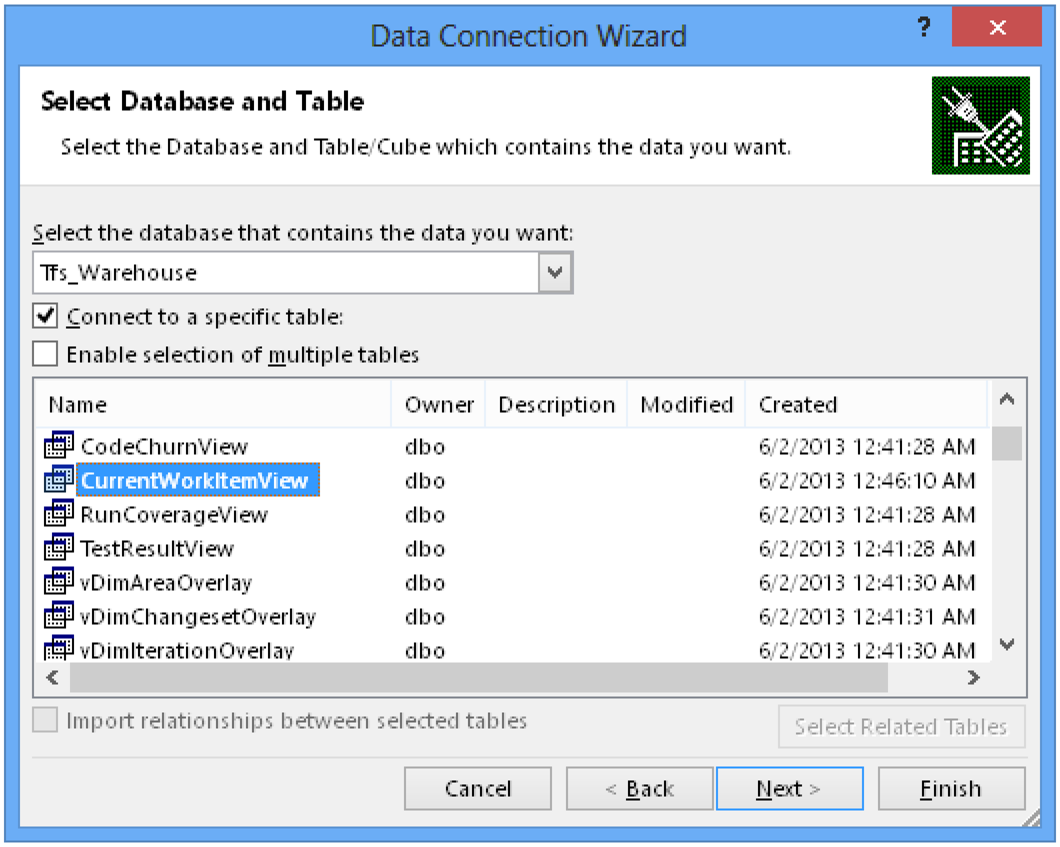
(●) Use Windows Authentication

( ) Use the following User Name and Password

User Name:

Password:

5. From the **Select the database that contains the data you want** drop-down list, select **Tfs_Warehouse.** Select **CurrentWorkItemView** from the list of tables. Click **Next**.



Select Database and Table

Select the Database and Table/Cube which contains the data you want.

Select the database that contains the data you want:

Tfs_Warehouse

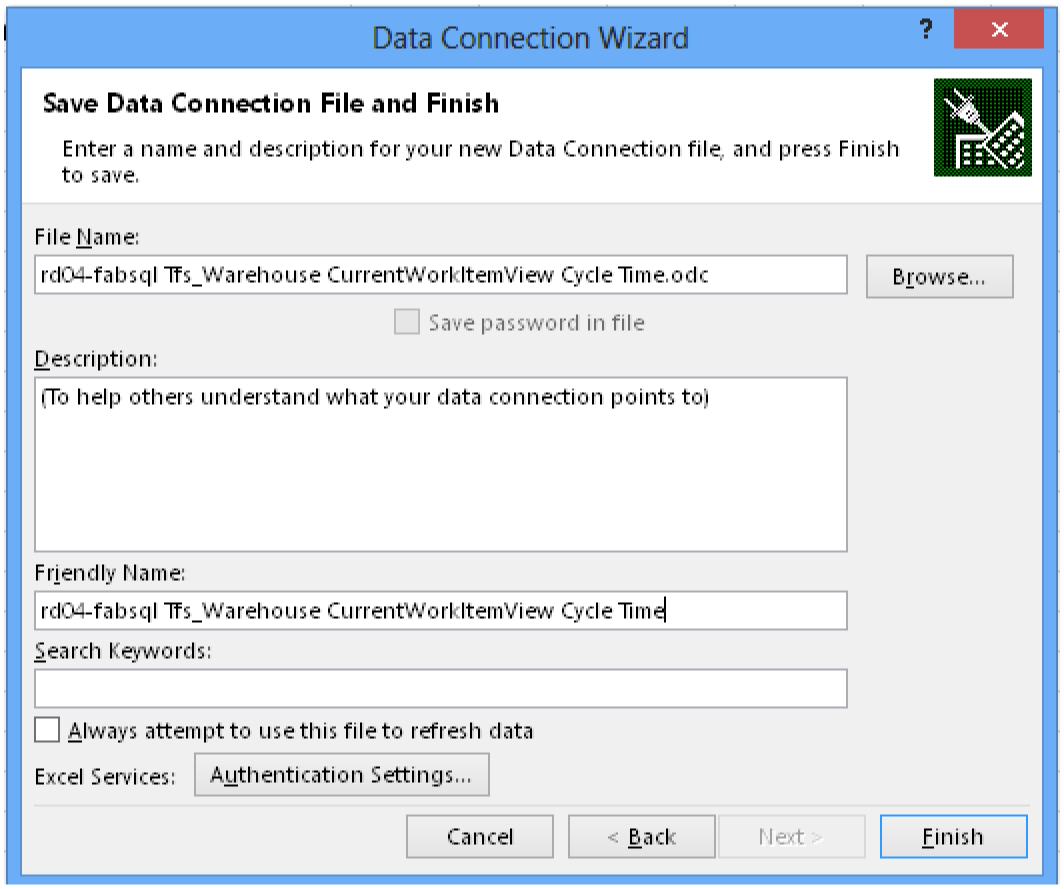
[✓] Connect to a specific table:

[ ] Enable selection of multiple tables

Name	Owner	Description	Modified	Created
CodeChurnView	dbo			6/2/2013 12:41:28 AM
CurrentWorkItemView	dbo			6/2/2013 12:46:10 AM
RunCoverageView	dbo			6/2/2013 12:41:28 AM
TestResultView	dbo			6/2/2013 12:41:28 AM
vDimAreaOverlay	dbo			6/2/2013 12:41:30 AM
vDimChangesetOverlay	dbo			6/2/2013 12:41:31 AM
vDimIterationOverlay	dbo			6/2/2013 12:41:30 AM

[ ] Import relationships between selected tables

6. Append the text **Cycle Time** to the **File Name** and **Friendly Name** fields so that there is no conflict with other reports in the same Excel workbook. Click **Finish**.



Save Data Connection File and Finish

Enter a name and description for your new Data Connection file, and press Finish to save.

File Name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle Time.odc

Save password in file

Description: (To help others understand what your data connection points to)

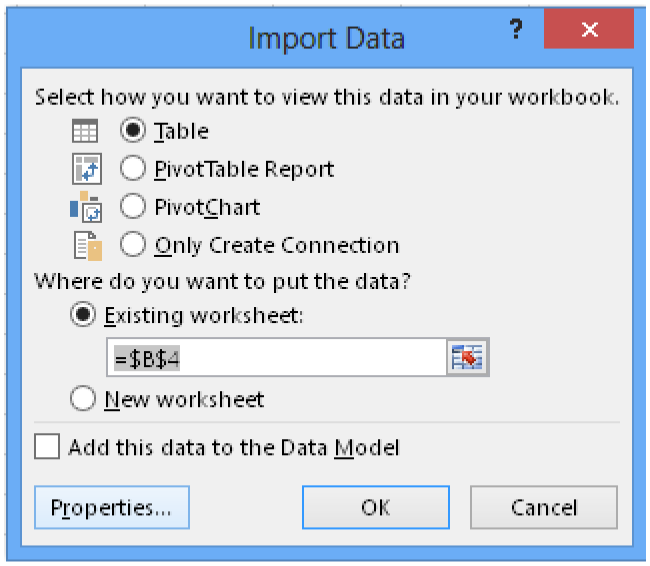
Friendly Name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle Time

Search Keywords:

Always attempt to use this file to refresh data

Excel Services: Authentication Settings...

7. The **Import Data** dialog box opens. Click **Properties**.



Select how you want to view this data in your workbook.

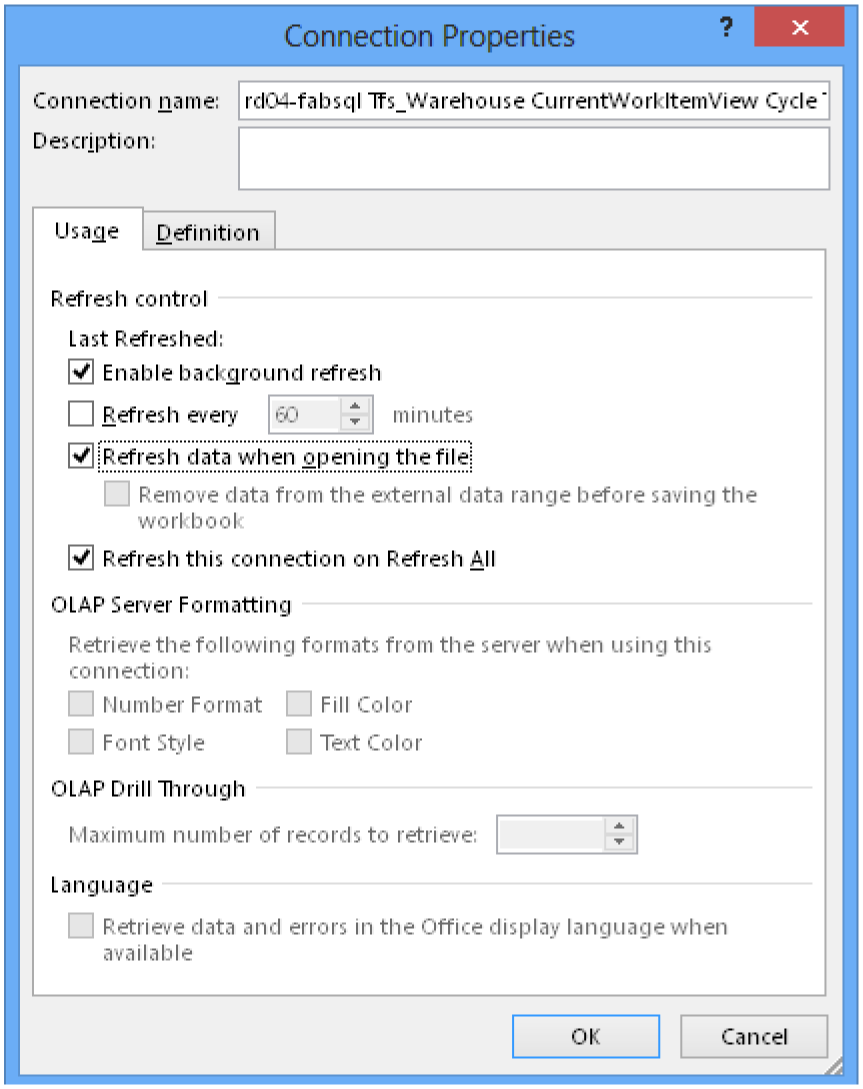
[✓ Table]
PivotTable Report
PivotChart
Only Create Connection

Where do you want to put the data?

[✓ Existing worksheet:] =$B$4
New worksheet

Add this data to the Data Model

8. The **Connection Properties** dialog box opens. Select the option **Refresh data when opening the file**, so that the report is updated each time the Excel workbook opens.



Connection name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle

Description:

Refresh control

Last Refreshed:

- [✓] Enable background refresh
- [ ] Refresh every 60 minutes
- [✓] Refresh data when opening the file
  - [ ] Remove data from the external data range before saving the workbook
- [✓] Refresh this connection on Refresh All

OLAP Server Formatting

Retrieve the following formats from the server when using this connection:

- [ ] Number Format
- [ ] Fill Color
- [ ] Font Style
- [ ] Text Color

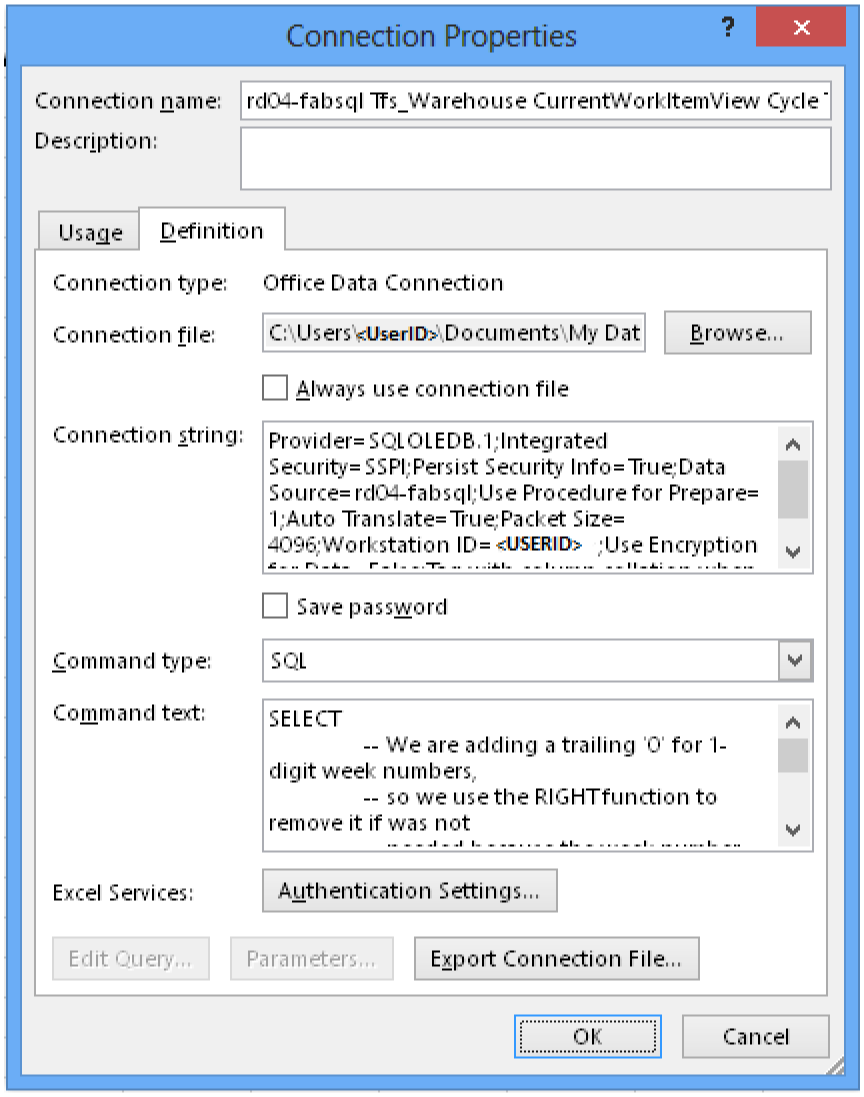
OLAP Drill Through

Maximum number of records to retrieve:

Language

- [ ] Retrieve data and errors in the Office display language when available

9. Select the **Definition** tab. From the **Command type** drop-down list, select **SQL**. In the **Command text** text box paste the entire query you prepared in Task 2, step 3. Click **OK**. You will get a warning. Ignore it and click **Yes**.



Connection name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle

Description:

Connection type: Office Data Connection

Connection file: C:\Users\<UserID>\Documents\My Dat

☐ Always use connection file

Connection string:

```
Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=True;Data
Source=rd04-fabsql;Use Procedure for Prepare=
1;Auto Translate=True;Packet Size=
4096;Workstation ID=<USERID> ;Use Encryption
```

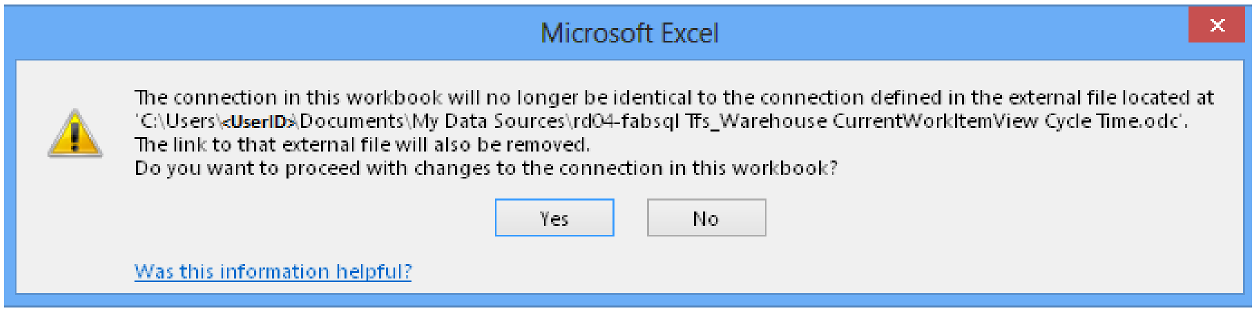
☐ Save password

Command type: SQL

Command text:

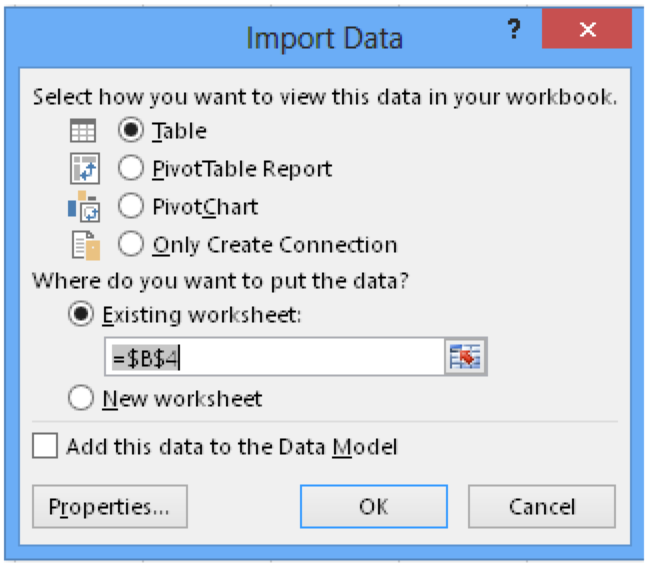
```
SELECT
            -- We are adding a trailing '0' for 1-
digit week numbers,
            -- so we use the RIGHT function to
remove it if was not
```

Excel Services:



The connection in this workbook will no longer be identical to the connection defined in the external file located at 'C:\Users\<UserID>\Documents\My Data Sources\rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle Time.odc'.
The link to that external file will also be removed.
Do you want to proceed with changes to the connection in this workbook?

10. The **Import Data** dialog box reappears. Click **OK.**


Import Data

Select how you want to view this data in your workbook.

(●) Table

( ) PivotTable Report

( ) PivotChart

( ) Only Create Connection

Where do you want to put the data?

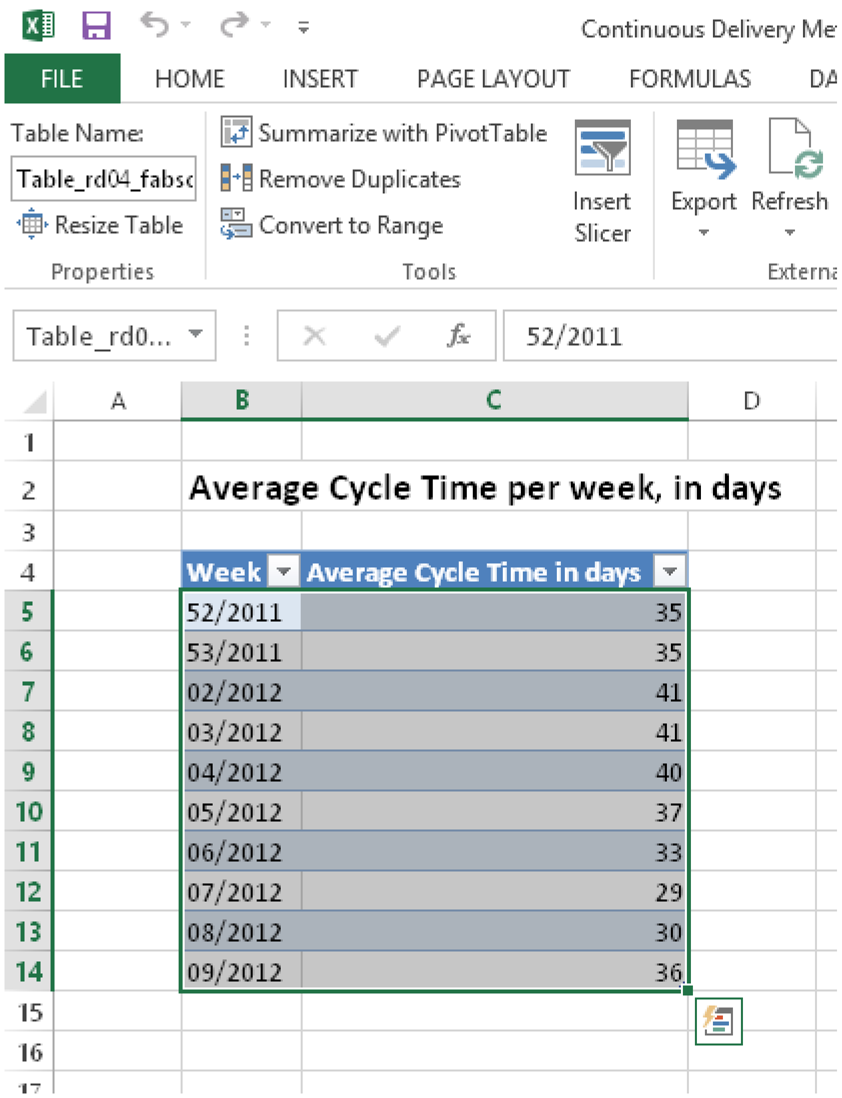
(●) Existing worksheet:

=$B$4

( ) New worksheet

☐ Add this data to the Data Model

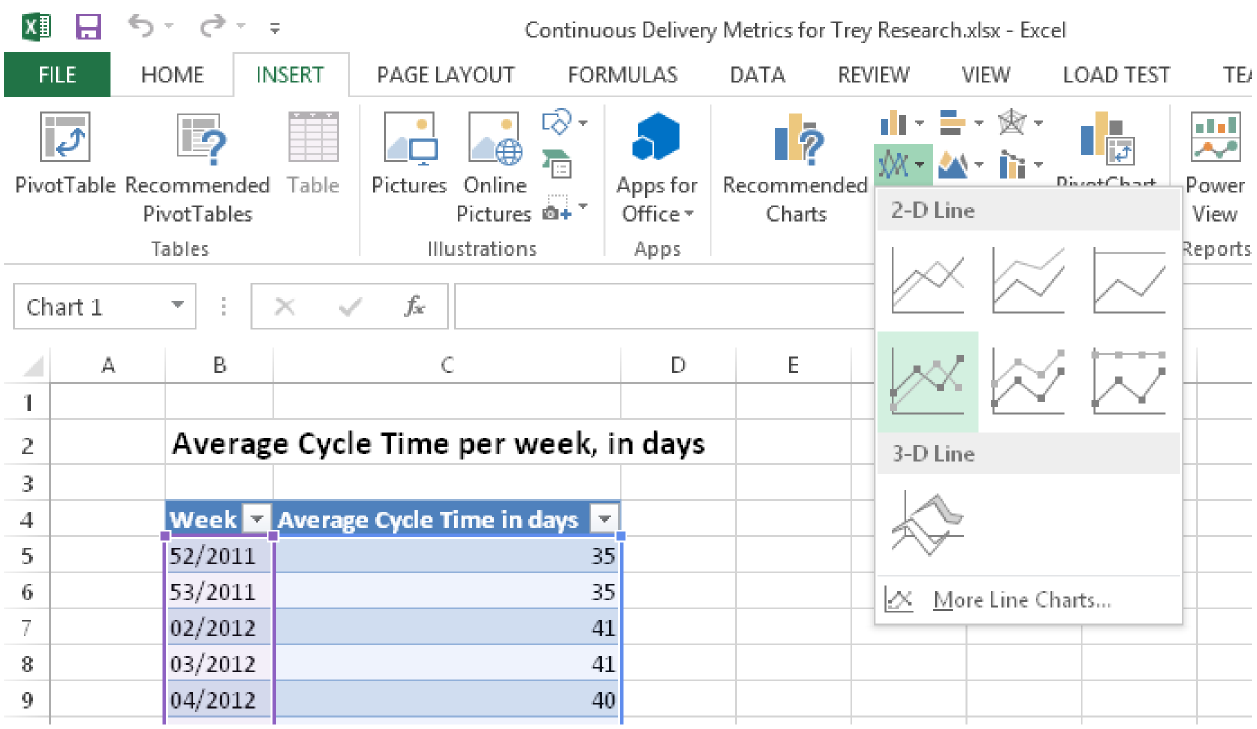
11. In your Excel sheet, you should see a table with the cycle time data in it. Select all the rows and columns that contain the data. In the following screenshot, this is from **B5** to **C14**.



Average Cycle Time per week, in days

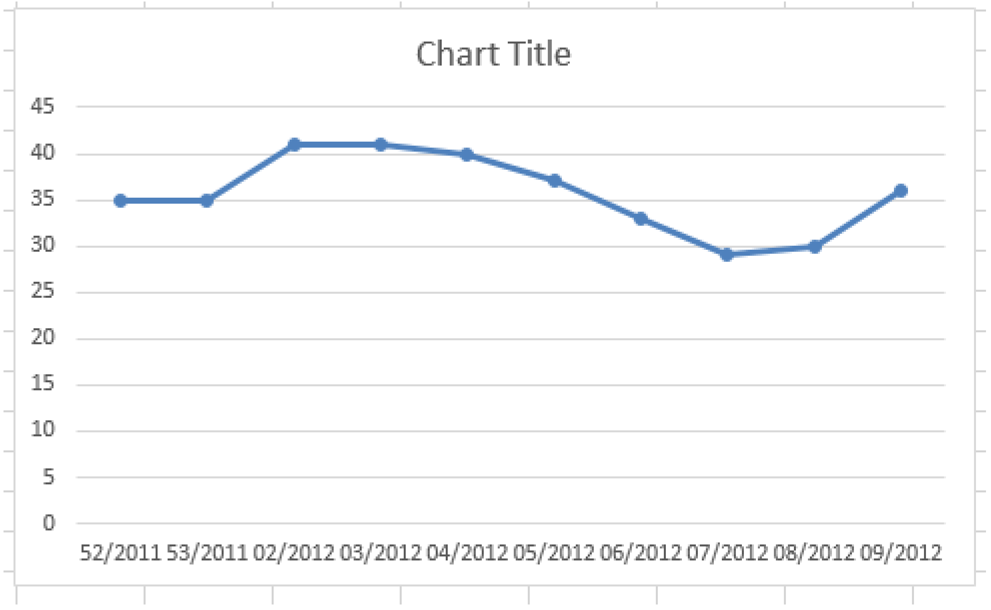
Week	Average Cycle Time in days
52/2011	35
53/2011	35
02/2012	41
03/2012	41
04/2012	40
05/2012	37
06/2012	33
07/2012	29
08/2012	30
09/2012	36

12. From the **INSERT** menu, click **Insert Line Chart.** Select **Line with Markers**.



	A	B	C	D	E
1					
2		Average Cycle Time per week, in days			
3					
4		Week	Average Cycle Time in days		
5		52/2011	35		
6		53/2011	35		
7		02/2012	41		
8		03/2012	41		
9		04/2012	40		

13. You should see a chart that resembles the following screenshot.



14. Format the resulting chart to suit your preferences. In the following screenshots, the **Chart Title** is removed and a linear trend line is added. (To do this, click the **+** sign that appears when you select the chart. A popout menu appears that gives you a variety of options.) The chart is also widened so that the weeks and years at the bottom are clearer, and the chart is positioned near the table.

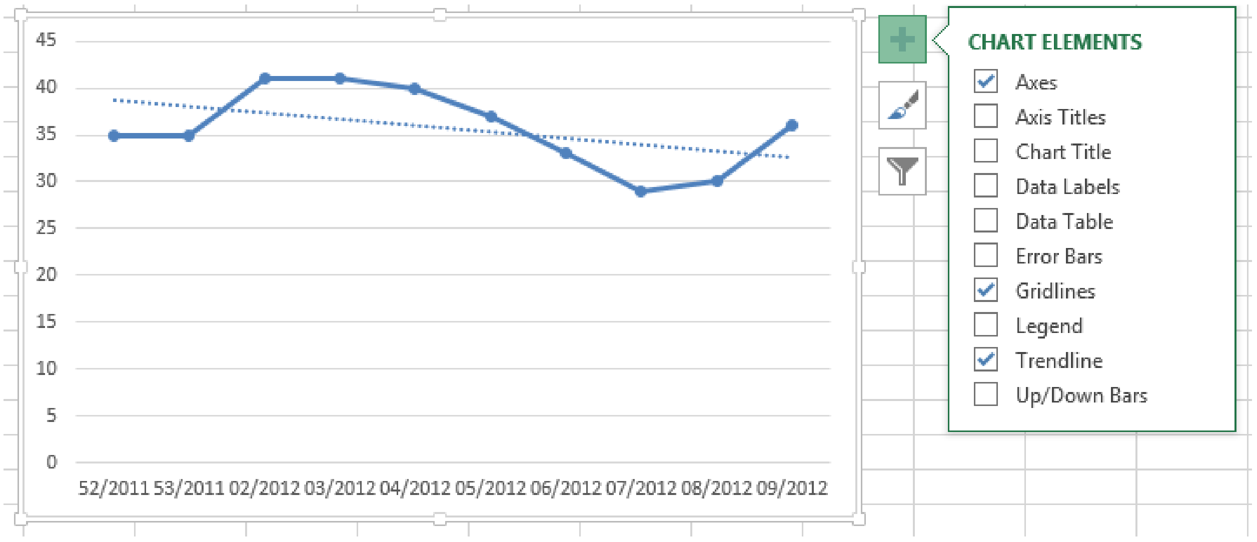
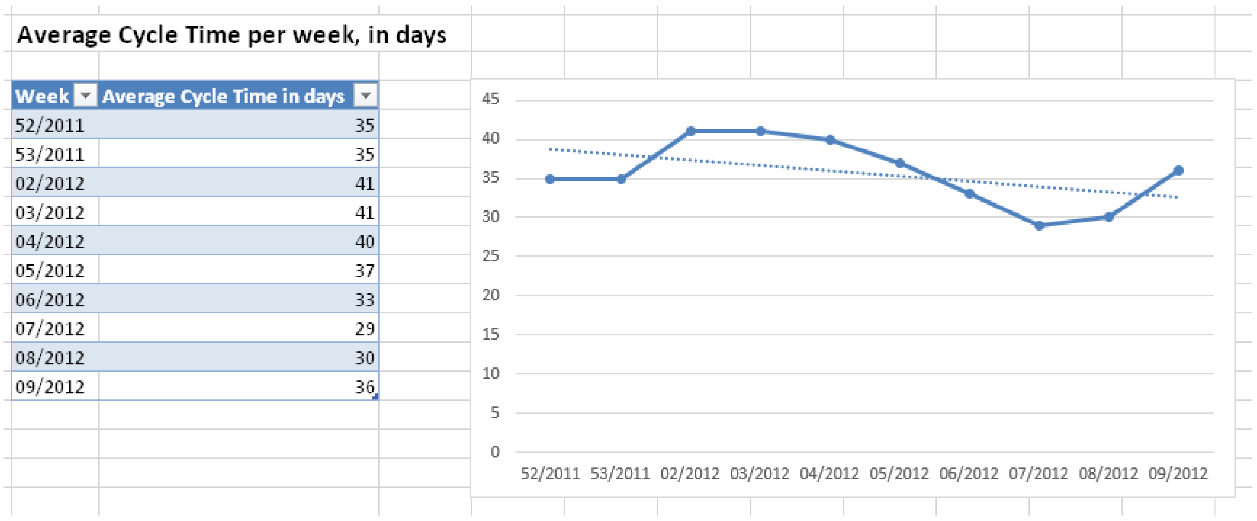


CHART ELEMENTS

- [✓] Axes
- [ ] Axis Titles
- [ ] Chart Title
- [ ] Data Labels
- [ ] Data Table
- [ ] Error Bars
- [✓] Gridlines
- [ ] Legend
- [✓] Trendline
- [ ] Up/Down Bars



Average Cycle Time per week, in days

Week	Average Cycle Time in days
52/2011	35
53/2011	35
02/2012	41
03/2012	41
04/2012	40
05/2012	37
06/2012	33
07/2012	29
08/2012	30
09/2012	36

15. Save and close the Excel workbook. The next time you open it, both the table and the chart will be updated to reflect the additional user stories that the team has released.

Note: The **Tfs_Warehouse** database is updated periodically by a job that uses data gathered from the operational database of the TFS collection. By default, the job runs every 2 minutes. You can change the time by using the **WarehouseControlWebService** methods, which are accessible from TFS. You can also force a manual update. For more information about the web service see Manually Process the Data Warehouse and Analysis Services Cube for Team Foundation Server. Because the **Tfs_Warehouse** is not updated in real time, you may find that the report doesn't reflect the last work item data you changed. You can wait a maximum of 2 minutes (if the default waiting time is used) and then refresh the data or you can reopen the workbook.

Task 4: Understanding the Results

In this task you learn how to interpret the data you've gathered. The data in the cycle time report allows you to infer valuable information about a project. The sample data used in this lab is from the Trey Research team's project. Examining the data gives you a good idea of how the cycle time changed in relation to the team's efforts to improve the pipeline.

Initial Cycle Time

At the beginning of their project, before they implemented any improvements, the team had an average cycle time of 35 days. This is close to the 6 weeks they estimated when they prepared their first value stream map. (See Chapter 2 of Building a Release Pipeline with Team Foundation Server 2012.) An average cycle time of 35 days means that during the last two weeks of 2011, they were releasing features they had decided to implement approximately 6 weeks earlier, around mid-November.

Changes in Cycle Time

You can use the report to determine if the team is releasing features faster than before. Of course, improving the cycle time shouldn't be because you've reduced the quality of your software, or because you've taken shortcuts. It should be because you've improved your release process and because you've removed bottlenecks. Cycle time can be combined with other metrics, such as MTTR and MTBF (covered later in this lab), to give a better understanding of the release process.

During the first weeks of 2012, the team saw their cycle time increase significantly. This is because they began to spend time improving their release process improvement, at the expense of implementing new features. Their efforts quickly yielded benefits because the cycle time was greatly reduced from the third to the seventh week of 2012.

After that, the cycle time increased again because the team spent time creating the TFS reports. These reports gave them a better understanding of their process, and let them know if the process was improving over time. The trend line shows that, even with its ups and downs, the cycle time has improved from where they were in the beginning.

Exercise 3: Creating the MTTR report

In this exercise you learn to create the MTTR report. The overall process is very similar to the cycle time report, so this exercise focuses on the differences between the two.

Task 1: Identify the Required information and Decide How to Present It

MTTR is the average time elapsed from the moment a problem is found in the production environment to the moment that the problem is fixed. For more information, see Chapter 5 of the guidance Building a Release Pipeline with Team Foundation Server 2012.

In order to track the MTTR, you determine the data you need and how to extract it from TFS. In the MSF for Agile process template, problems found in production are represented by the **Bug** work item, so you will filter on that work item type.

Not all the bugs stored in the TFS team project are useful for tracking the MTTR. Only those that have been fixed in production are. For this lab, the criteria is to choose the bugs that are in the **Closed** state, because of the **Verified** reason.

Note: The problem with this criteria is that the **Bug** work item is used not only for production bugs but for any bug that is found during the development process. To solve this issue, you can modify the **Bug** work item to include a new field that indicates whether the bug is a production bug one or not. Another option is to use a special value for the **Area** field. These modifications are out of the scope of this HOL, but if you decide to implement them, you can easily adapt the report query so that it filters by using the new field.

You can see the workflow of states for the **Bug** work item.

You also need to filter by the team project and you need two fields to calculate the MTTR. The first field is the moment when the bug is discovered in the production environment. In this lab, this is the date when the **Bug** work item's state became **Active**. In the MSF for Agile template, this value is automatically stored in the **Activated Date** field.

The second field is when the bug is fixed and its state becomes **Closed**. In the MSF for Agile template, this value is automatically stored in the **Closed Date** field.

You must also decide how to present the data in the final report. The calculation to be made is the time elapsed between the **Closed Date** value and the **Activated Date** value. However, the report would be more readable if the values were adjusted by units of time and by grouping.

This lab uses the hour as the unit of time. The results are grouped so that they present the MTTR per week. This gives a reasonable level of detail and makes it easier to see if the MTTR is getting smaller, which means it's improving over time.

There are no restrictions on the number of bugs being returned. If you have many work items in your team project and you want to keep the report short, you can add a filter to take out the oldest bugs, or the ones that were closed before a given date.

Here is a summary of how to track the MTTR per week, in hours, since the beginning of the project. In order to filter the rows, use the following items:

- The **Bug** work item.
- The **Closed** state.
- The **Verified** reason.
- The name of the team project collection
- The name of the team project.

To calculate the MTTR for these rows, use the following items.

- The **Activated Date** field.
- The **Closed Date** field.

To group and present the information:

- Use **hour** as the unit of time.
- Use **week** as the grouping scale.

Note: As with any other report in TFS, team members must make sure to keep the information in the work items up to date. For this report, they should register any bug found in production as a **Bug** work item and set its state to **Active**. They should set its state to **Closed** when it has been fixed in production.

Task 2: Prepare the SQL Query that Retrieves Data From the Tfs_Warehouse Database

The MTTR calculation uses the same approach as the cycle time calculation. All the required information is in the **CurrentWorkItemView**, in the **Tfs_Warehouse** database.

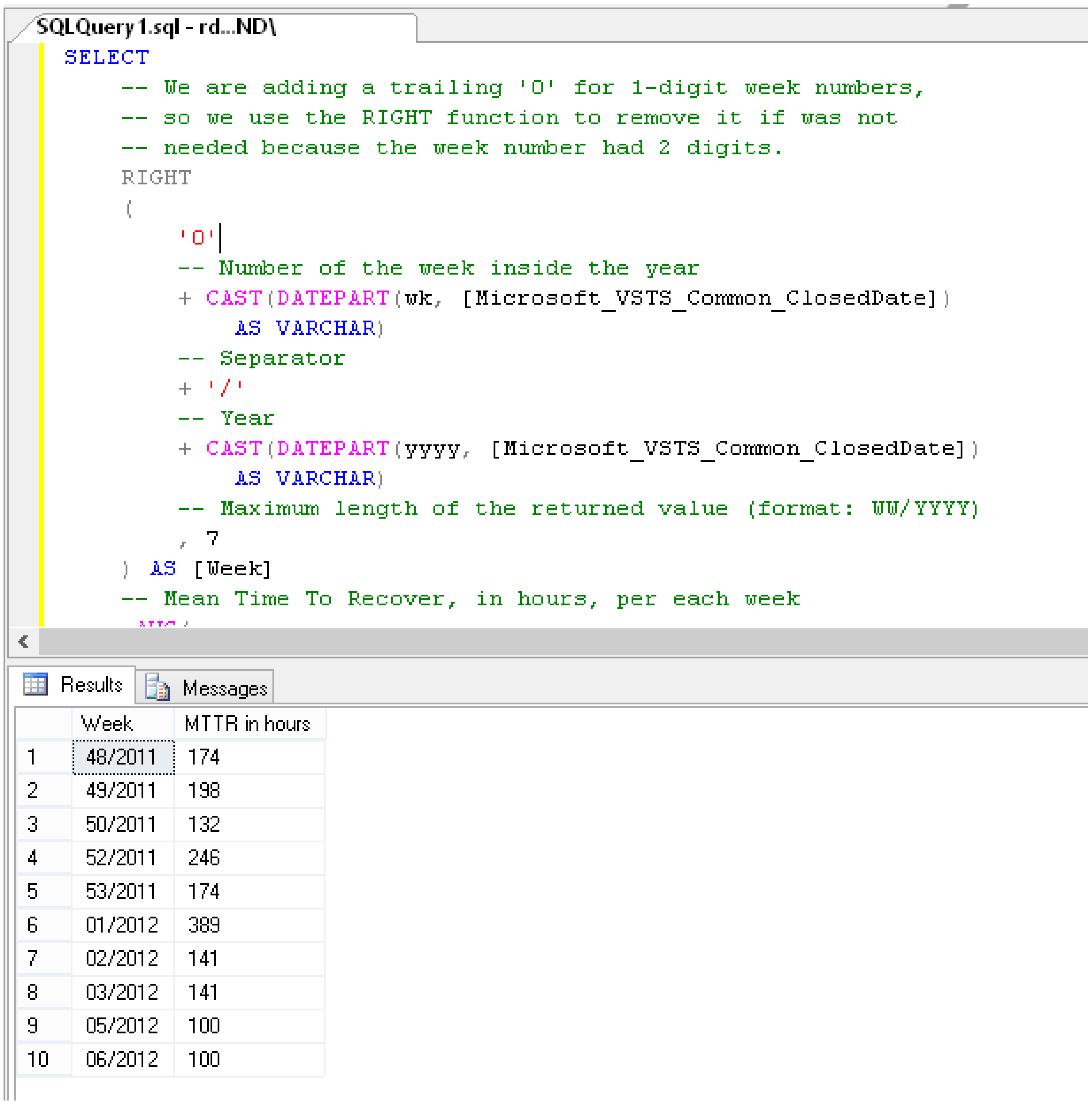
1. Identify the names of the columns in the **CurrentWorkItemView** that you need for your report.
    - To filter the rows, you need:
        - The column name of the **Bug** work item, which is **System_WorkItemType**.
        - The column name of the **Closed** state, which is **System_State**.
        - The column name of the team project collection and the team Project, which is **ProjectPath**.
    - To calculate the MTTR for these rows, you need:
        - The column name of the **Activated Date** field, which is **Microsoft_VSTS_Common_ActivatedDate**.
        - The column name of the **Closed Date** field, which is **Microsoft_VSTS_Common_ClosedDate**.
2. Write the query. Here is a commented, working version that you can use. It is almost the same as the cycle time query.

Note: Change '\YOUR-PROJECTCOLLECTION NAME\TEAM PROJECT NAME' to your project's path.

T-SQL

```
SELECT
       -- We are adding a trailing '0' for 1-digit week numbers,
       -- so we use the RIGHT function to remove it if was not
       -- needed because the week number had 2 digits.
       RIGHT
       (
              '0'
              -- Number of the week inside the year
              + CAST(DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate])
                     AS VARCHAR)
              -- Separator
              + '/'
              -- Year
              + CAST(DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate])
                     AS VARCHAR)
              -- Maximum length of the returned value (format: WW/YYYY)
              , 7
       ) AS [Week]
       -- Mean Time To Recover, in hours, per each week
       ,AVG(
              DATEDIFF(hh, [Microsoft_VSTS_Common_ActivatedDate],
                     [Microsoft_VSTS_Common_ClosedDate])
              )
              AS [MTTR in hours]
FROM
       [dbo].[CurrentWorkItemView]
WHERE
       [ProjectPath] = '\YOUR-PROJECTCOLLECTION NAME\TEAM PROJECT NAME'
       AND [System_WorkItemType] = 'Bug'
       AND [System_State] = 'Closed'
GROUP BY
       -- Grouping by week and year to calculate the average
       DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate]),
       DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate])
ORDER BY
       -- Ordering by year and week number
       DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate]),
       DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate])
```

3. If you paste the query into SQL Management Studio and run it, you should see one row for each week when at least one bug was closed, as well as the MTTR in hours for that week. The following screenshot show the query and its results if you use the data you prepared in the first exercise.


SQLQuery1.sql - rd...ND\

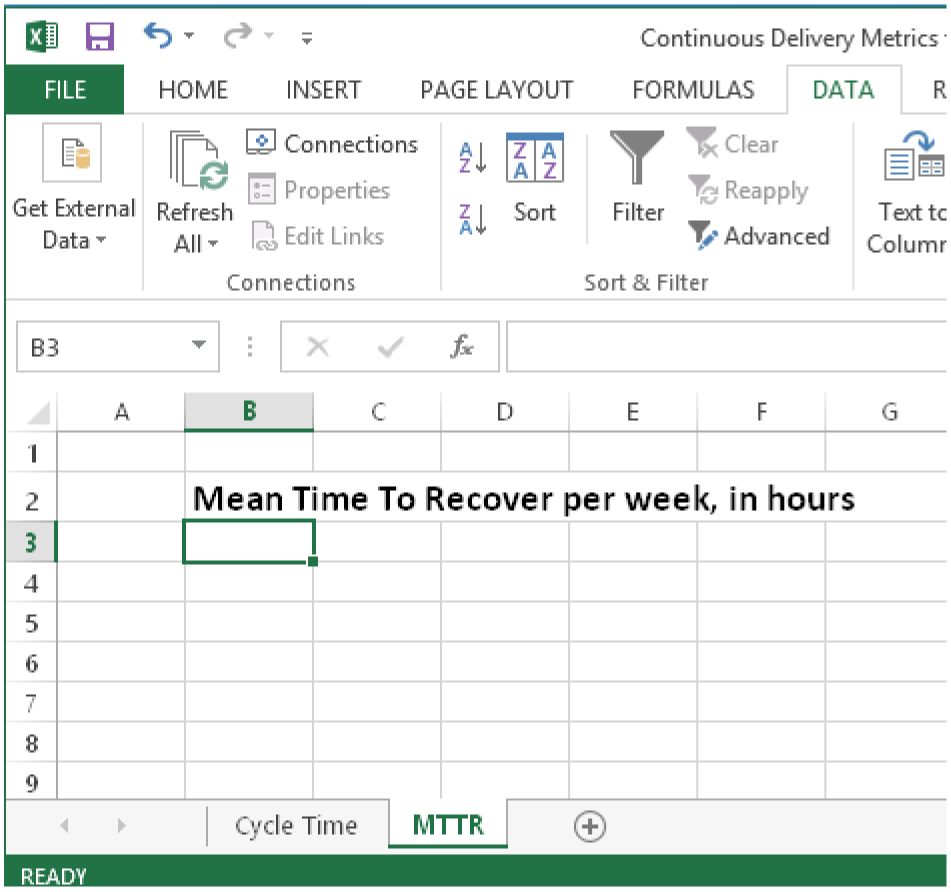
```
SELECT
    -- We are adding a trailing '0' for 1-digit week numbers,
    -- so we use the RIGHT function to remove it if was not
    -- needed because the week number had 2 digits.
    RIGHT
    (
        '0'
        -- Number of the week inside the year
        + CAST(DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate])
            AS VARCHAR)
        -- Separator
        + '/'
        -- Year
        + CAST(DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate])
            AS VARCHAR)
        -- Maximum length of the returned value (format: WW/YYYY)
        , 7
    ) AS [Week]
    -- Mean Time To Recover, in hours, per each week
```

	Week	MTTR in hours
1	48/2011	174
2	49/2011	198
3	50/2011	132
4	52/2011	246
5	53/2011	174
6	01/2012	389
7	02/2012	141
8	03/2012	141
9	05/2012	100
10	06/2012	100

Task 3: Create an Excel Sheet to Present the Results

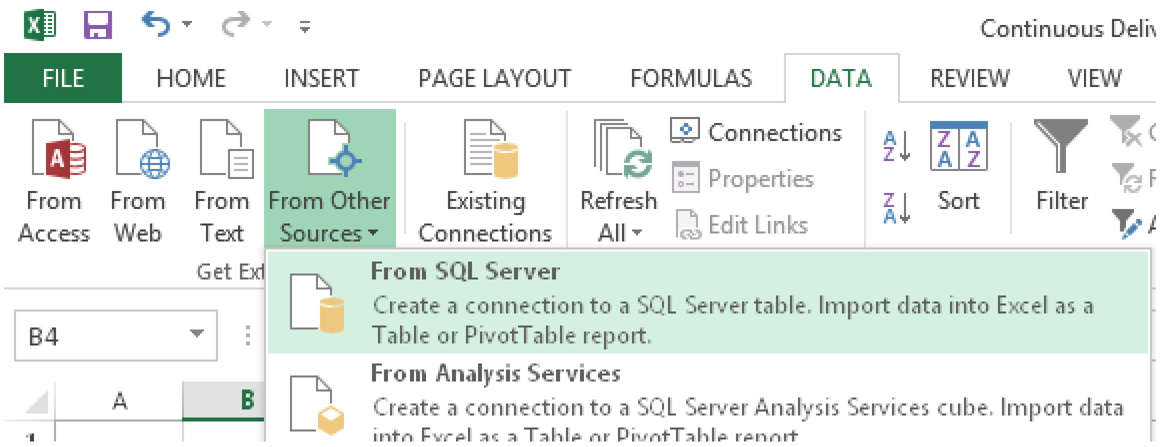
In this task you create a new Excel sheet to present the MTTR data.

1. Open the **Continuous Delivery Metrics for TreyResearch** Excel workbook**.**
2. Add a new sheet and name it **MTTR** and write a title for the report in the first cells. The first screenshot shows an example.



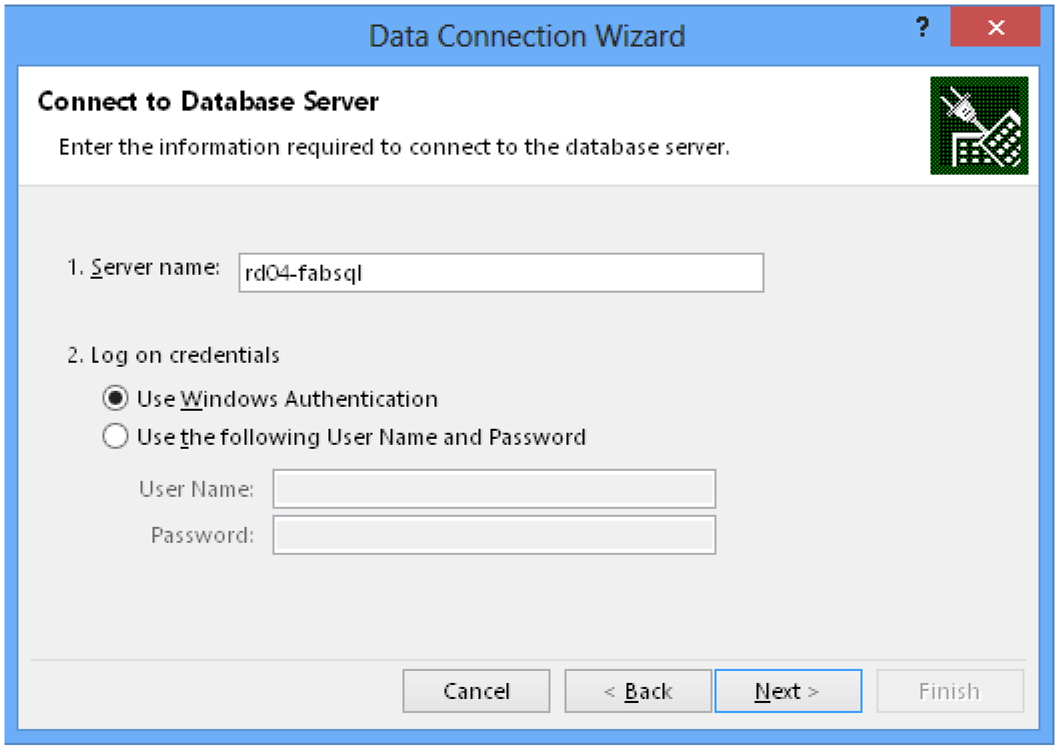
Mean Time To Recover per week, in hours

3. Select the cell where you want the data for the report to be inserted (for example, **B4**). From the **DATA** menu, select **From Other Sources.** Select **From SQL Server** from the drop-down list.



From SQL Server
Create a connection to a SQL Server table. Import data into Excel as a Table or PivotTable report.
From Analysis Services
Create a connection to a SQL Server Analysis Services cube. Import data into Excel as a Table or PivotTable report

4. The **Data Connection Wizard** appears. In the **Server name** text box, enter the server name where the **Tfs_Warehouse** database is located. If necessary, enter the credentials. Click **Next**.



Connect to Database Server

Enter the information required to connect to the database server.

1. Server name: rd04-fabsql

2. Log on credentials

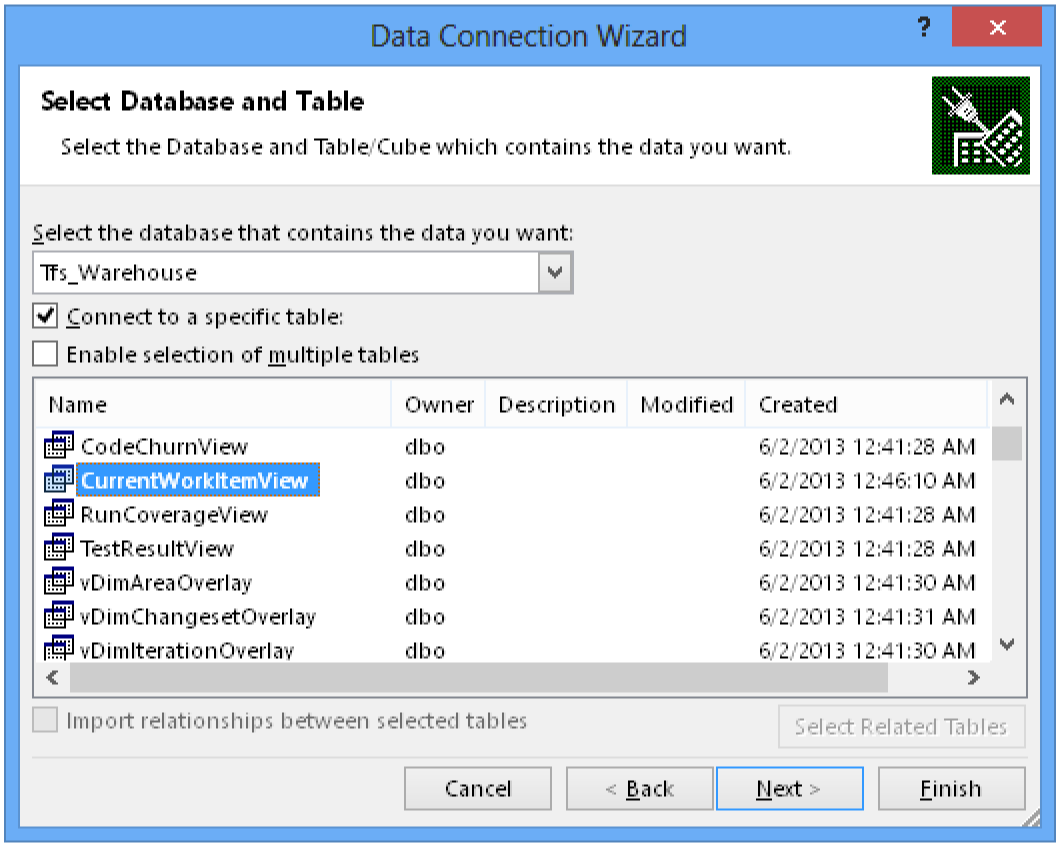
(●) Use Windows Authentication

( ) Use the following User Name and Password

User Name:

Password:

5. From the **Select the database that contains the data you want** drop-down list, select **Tfs_Warehouse**. Select **CurrentWorkItemView** from the list of tables. Click **Next**.



Data Connection Wizard

Select Database and Table

Select the Database and Table/Cube which contains the data you want.

Select the database that contains the data you want:

Tfs_Warehouse

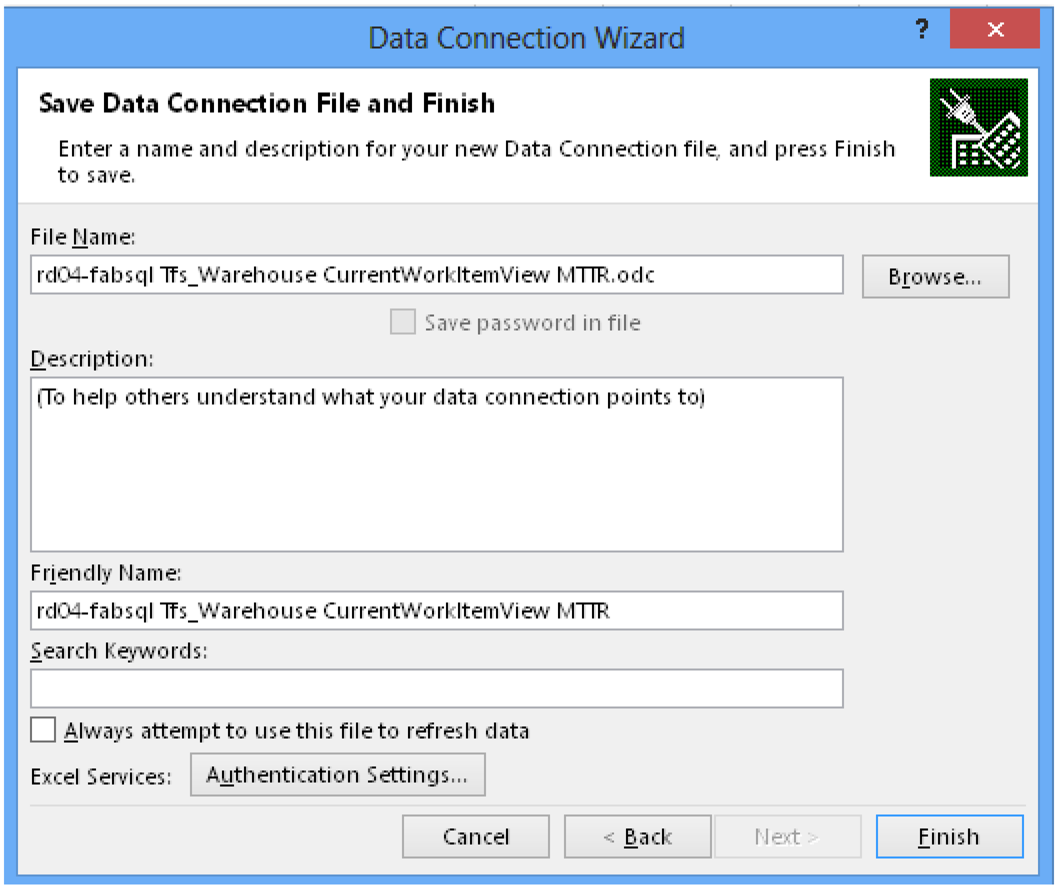
[✓] Connect to a specific table:

[ ] Enable selection of multiple tables

Name	Owner	Description	Modified	Created
CodeChurnView	dbo			6/2/2013 12:41:28 AM
CurrentWorkItemView	dbo			6/2/2013 12:46:10 AM
RunCoverageView	dbo			6/2/2013 12:41:28 AM
TestResultView	dbo			6/2/2013 12:41:28 AM
vDimAreaOverlay	dbo			6/2/2013 12:41:30 AM
vDimChangesetOverlay	dbo			6/2/2013 12:41:31 AM
vDimIterationOverlay	dbo			6/2/2013 12:41:30 AM

[ ] Import relationships between selected tables

6. Append the text **MTTR** to the proposed **File Name** and **Friendly Name** fields, so that there is no conflict with other reports in the same Excel workbook. Click **Finish**.


Data Connection Wizard

Save Data Connection File and Finish

Enter a name and description for your new Data Connection file, and press Finish to save.

File Name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView MTTR.odc

Save password in file

Description: (To help others understand what your data connection points to)

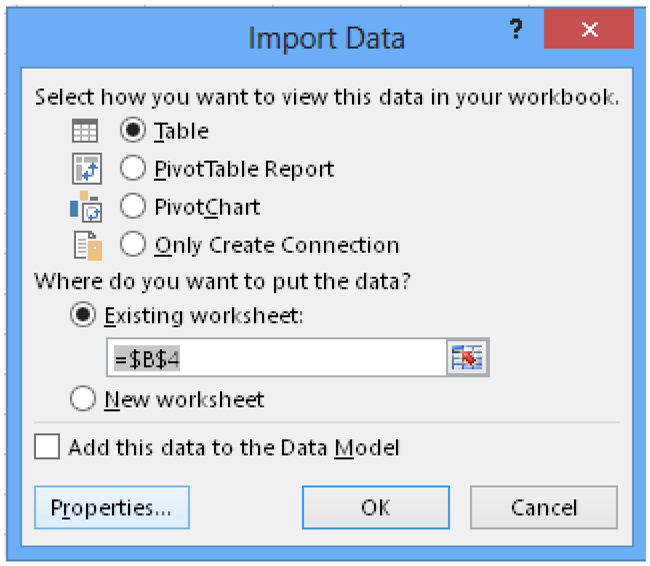
Friendly Name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView MTTR

Search Keywords:

Always attempt to use this file to refresh data

Excel Services: Authentication Settings...

7. The **Import Data** dialog box opens. Click **Properties**.


Import Data

Select how you want to view this data in your workbook.

[✓ Table] PivotTable Report PivotChart Only Create Connection

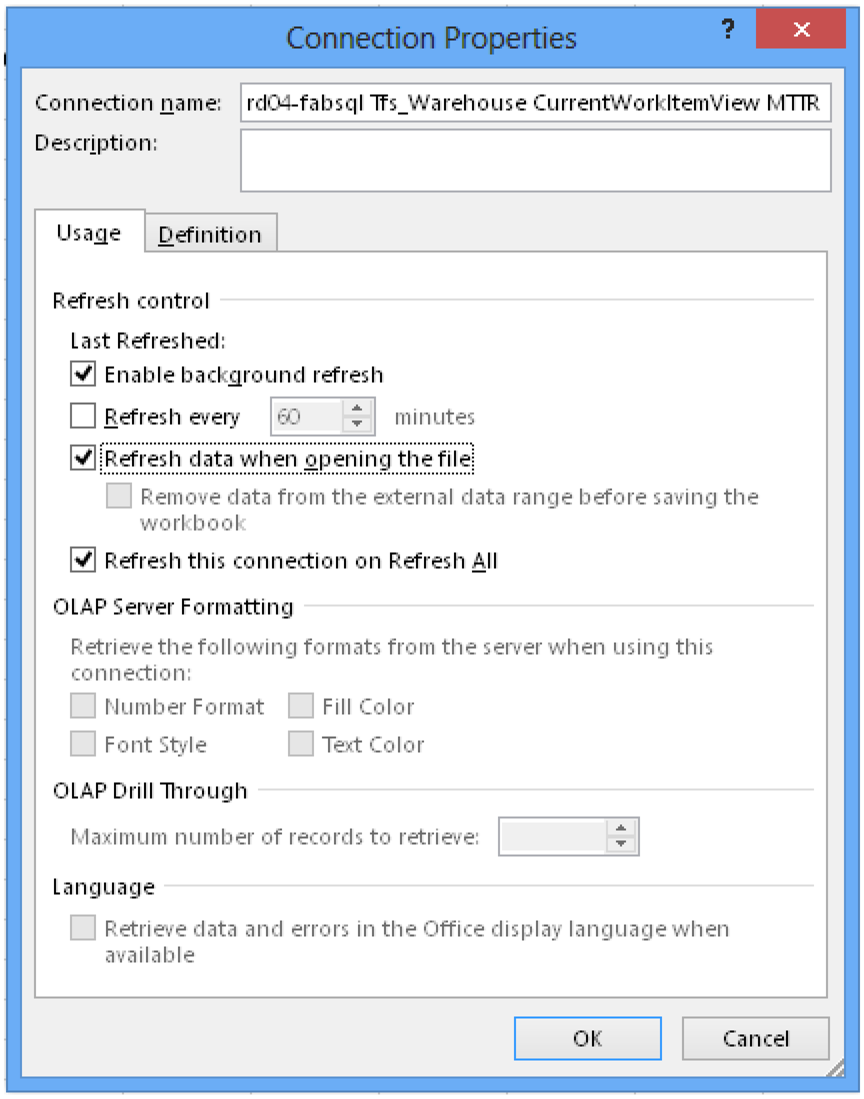
Where do you want to put the data?

[✓ Existing worksheet:] =$B$4

New worksheet

Add this data to the Data Model

8. The **Connection Properties** dialog box opens. Select the option **Refresh data when opening the file**, so the report gets updated each time the Excel workbook opens.


Connection Properties

Connection name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView MTTR

Description:

Refresh control

Last Refreshed:

- [x] Enable background refresh
- [ ] Refresh every 60 minutes
- [x] Refresh data when opening the file
  - [ ] Remove data from the external data range before saving the workbook
- [x] Refresh this connection on Refresh All

OLAP Server Formatting

Retrieve the following formats from the server when using this connection:

- [ ] Number Format
- [ ] Fill Color
- [ ] Font Style
- [ ] Text Color

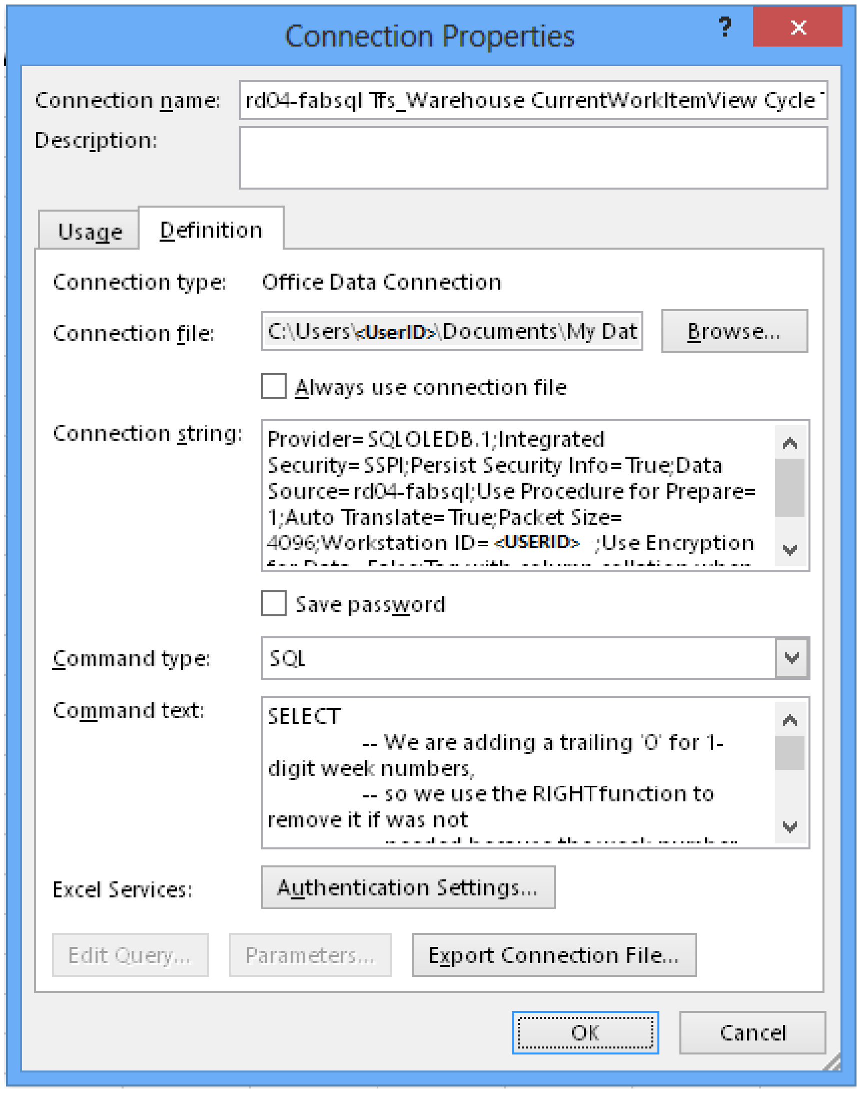
OLAP Drill Through

Maximum number of records to retrieve:

Language

- [ ] Retrieve data and errors in the Office display language when available

9. Select the **Definition** tab. From the **Command type** drop-down list, select **SQL**. In the **Command text** text box paste the entire query you prepared in Task 2, step 2. Click **OK**. You will get a warning. Ignore it and click **Yes**.


Connection Properties

Connection name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle T

Description:

Usage | Definition

Connection type: Office Data Connection

Connection file: C:\Users\<UserID>\Documents\My Dat

Browse...

☐ Always use connection file

Connection string:

```
Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=True;Data
Source=rd04-fabsql;Use Procedure for Prepare=
1;Auto Translate=True;Packet Size=
4096;Workstation ID=<USERID>  ;Use Encryption
```

☐ Save password

Command type: SQL

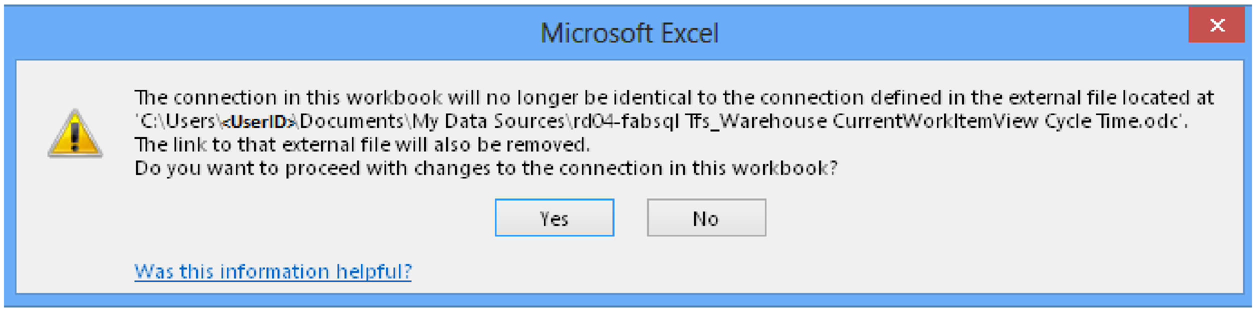
Command text:

```
SELECT
            -- We are adding a trailing '0' for 1-
digit week numbers,
            -- so we use the RIGHTfunction to
remove it if was not
```

Excel Services: Authentication Settings...

Edit Query... | Parameters... | Export Connection File...

OK | Cancel

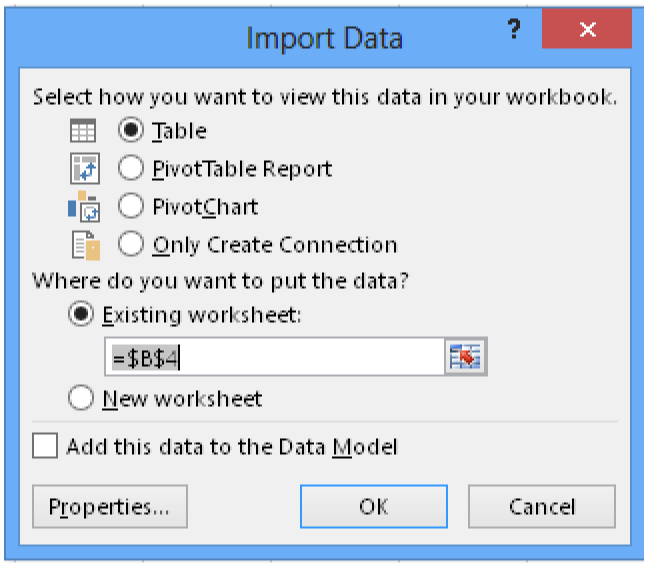

Microsoft Excel

The connection in this workbook will no longer be identical to the connection defined in the external file located at 'C:\Users\<UserID>\Documents\My Data Sources\rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle Time.odc'.
The link to that external file will also be removed.
Do you want to proceed with changes to the connection in this workbook?

Yes | No

Was this information helpful?

10. The **Import Data** dialog box reappears. Click **OK.**


Import Data

Select how you want to view this data in your workbook.

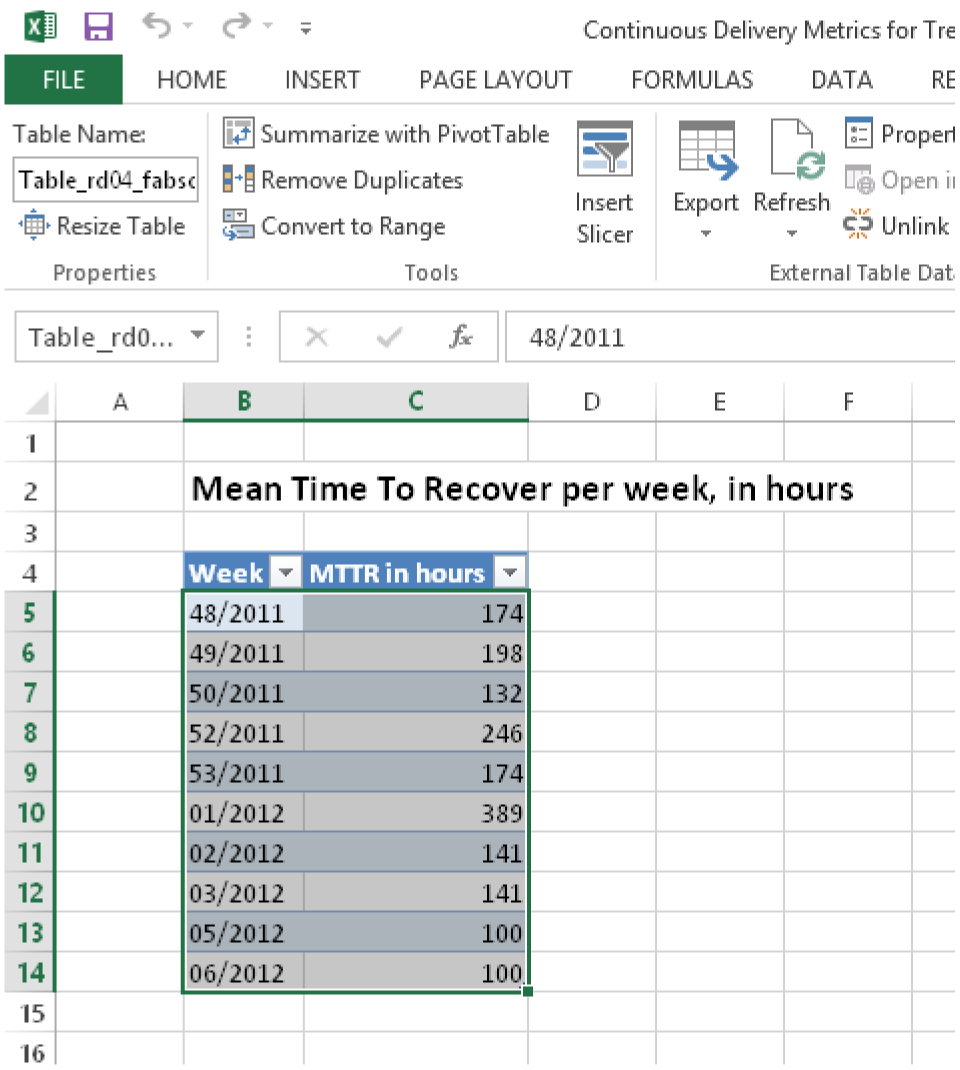
[✓ Table]
PivotTable Report
PivotChart
Only Create Connection

Where do you want to put the data?

[✓ Existing worksheet:]
=$B$4
New worksheet

Add this data to the Data Model

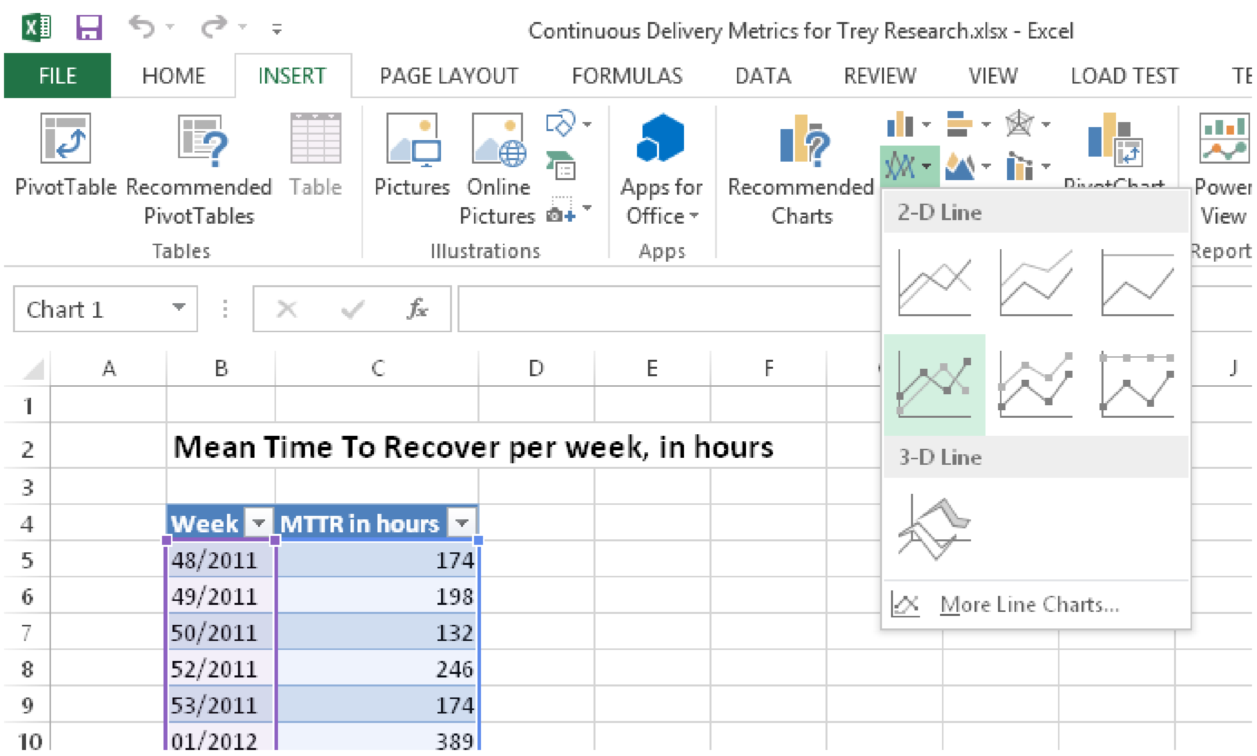
11. In your Excel sheet, you should see a table with the MTTR data in it. Select all the rows and columns containing the data. In the following screenshot, this is from **B5** to **C14**.



Mean Time To Recover per week, in hours

Week	MTTR in hours
48/2011	174
49/2011	198
50/2011	132
52/2011	246
53/2011	174
01/2012	389
02/2012	141
03/2012	141
05/2012	100
06/2012	100

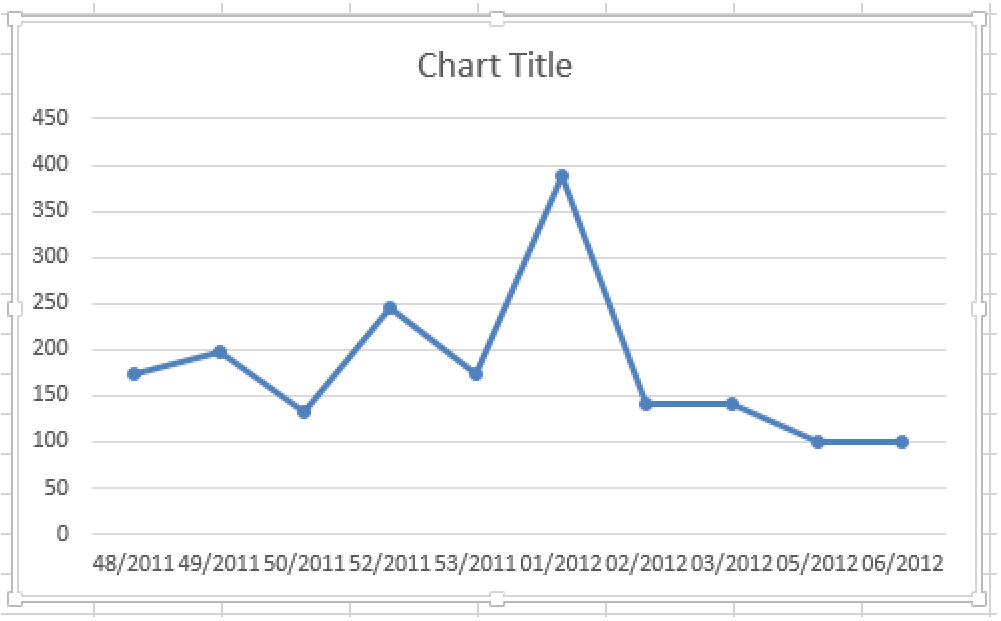
12. From the **INSERT** menu, click **Insert Line Chart**. Select **Line with Markers**.



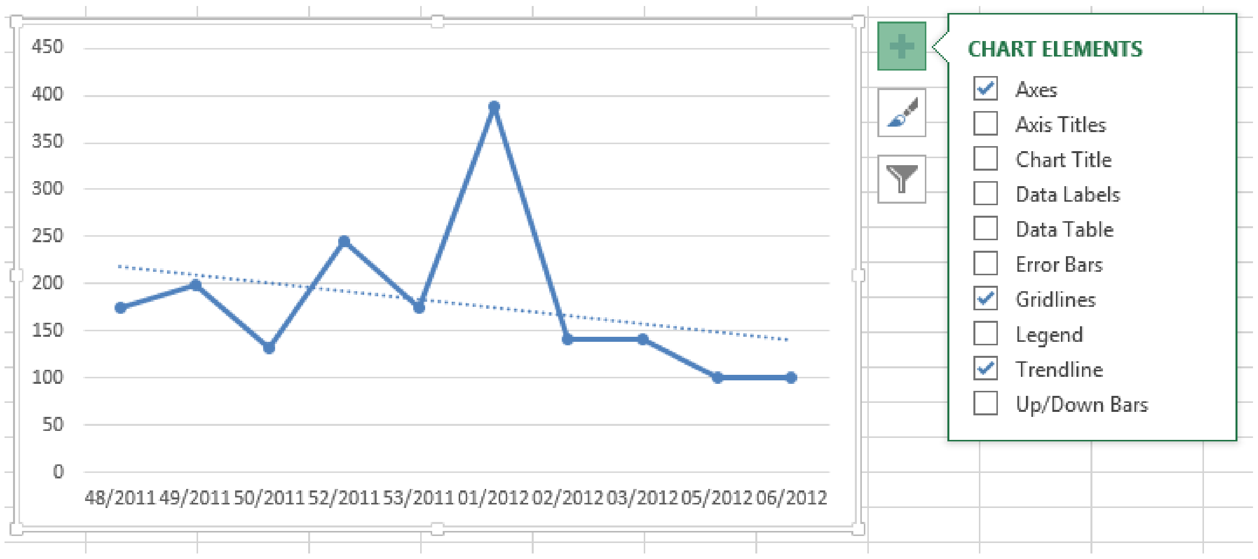
Mean Time To Recover per week, in hours

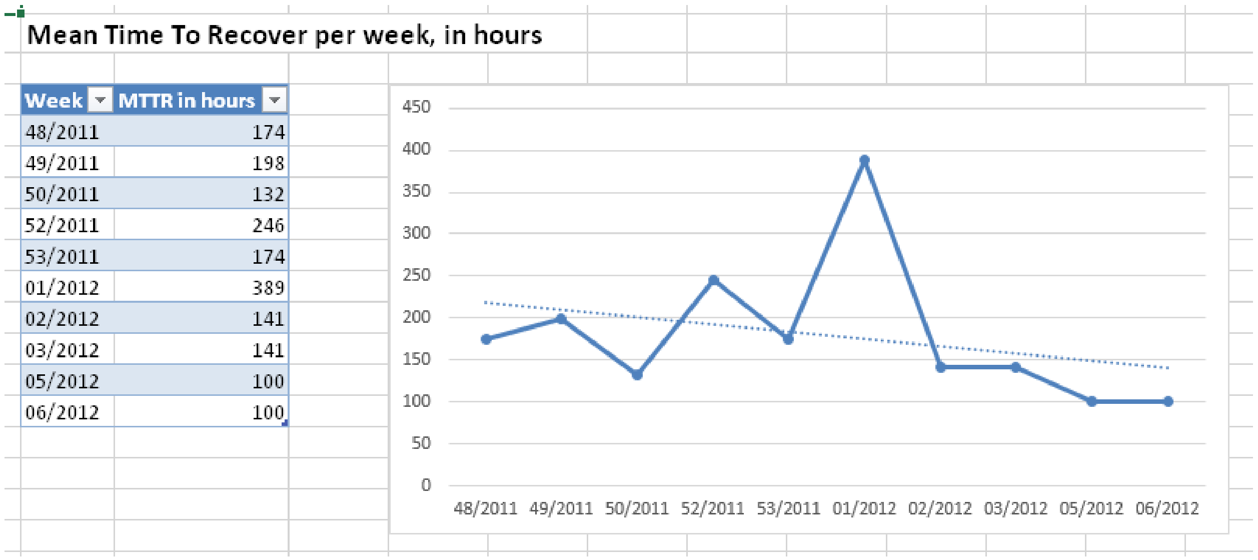
Week	MTTR in hours
48/2011	174
49/2011	198
50/2011	132
52/2011	246
53/2011	174
01/2012	389

13. You should see a chart that resembles the following screenshot.



14. Format the resulting chart to suit your preferences. In the following screenshots, the **Chart Title** is removed and a linear trend line is added. The chart is widened so that the weeks and years at the bottom are clearer, and the chart is positioned near to the table.





Mean Time To Recover per week, in hours

Week	MTTR in hours
48/2011	174
49/2011	198
50/2011	132
52/2011	246
53/2011	174
01/2012	389
02/2012	141
03/2012	141
05/2012	100
06/2012	100

15. Save and close the Excel workbook. The next time you open it, both the table and the chart will be updated to reflect the additional bus that the team has fixed.

Task 4: Understanding the Results

In this task you learn how to interpret the data you've gathered. The data in the MTTR report allows you to infer valuable information about a project. The sample data used in this lab is from the Trey Research team's project. Here are the conclusions.

Initial MTTR Results

At the beginning of the project, the Trey Research MTTR was erratic and difficult to predict. The process they used to fix bugs was not standardized and there were no good tools or best practices. An MTTR of 174 around mid-November of 2011 means that they were fixing production bugs that had been found approximately one week earlier.

Changes in MTTR

In the first weeks of 2012, the team saw a significantly larger (which means poorer) MTTR. This is because they began to spend time improving their release process, at the expense of fixing bugs. Their efforts quickly yielded benefits because the MTTR was greatly reduced from the second to the sixth week of 2012. It also became more predictable because of improvements to the pipeline, including automated deployments and tests.

If you compare MTTR with cycle time, you can see that the trend lines for both are similar. If you improve your process and reduce your cycle time, you must also be reducing your MTTR.

Exercise 4: Creating the MTBF report

In this exercise you learn to create the MTBF report. The overall process is very similar to the cycle time and MTTR reports, so this exercise focuses on the differences between them.

Task 1: Identify the Required Information and Decide How to Present It

The MTBF metric is the average time elapsed from the moment a problem is found in the production environment to the moment that the next problem is found in that environment. For more information about this metric, see Chapter 5 of Building a Release Pipeline with Team Foundation Server 2012.

In order to track the MTBF, you must determine the data you need and how to extract it from TFS. In the MSF for Agile process template, problems found in production are represented by the **Bug** work item, so you will filter on that work item type.

Not all the bugs stored in the TFS team project are useful for tracking the MTBF. Only those that have been fixed in production are. For this lab, the criteria is to choose the bugs that are in the **Closed** state, because of the **Verified** reason.

Note: This approach has the same issue with non-production bugs that was mentioned in the section on MTTR.

You also need to filter by the team project and you need one field to calculate the MTBF. It is the moment when the bug is discovered in the production environment, which is the **Activated Date** field. The calculation to determine the MTBF is the time elapsed between the **Activated Date** of one bug and the **Activated Date** of the next bug to be found.

There are no restrictions on the number of bugs being returned. If you have many work items in your team project and you want to keep the report short, you can add a filter to take out the oldest bugs, or the ones that were closed before a given date.

You must also decide how to present the data in the final report. This lab uses the hour as the unit of time. The results are grouped so that they present the MTBF per week. This gives a reasonable level of detail and makes it easier to see if the MTBF is growing larger, which means it's improving over time.

Here is a summary of how to track the MTBF per week, in hours, from the beginning of the project.

In order to filter the rows, use the following items:

- The **Bug** work item.
- The **Closed** state.
- The **Verified** reason. Note that the only valid reason for a bug to be in the **Closed** state in the MSF for Agile template is **Verified**, so you don't need to filter on it.
- The name of the team project collection.
- The name of the team project.

To calculate the MTBF for these rows, use the **Activated Date** field.

To group and present the information:

- Use **hour** as the unit of time.
- Use **week** as the grouping scale.

Note: As with any other report in TFS, team members must make sure to keep the information in the work items up to date. In particular, for this report, they should register any bug found in production as a **Bug** work item with a state of **Active**. They should set the state to **Closed** when it has been fixed in production.

Task 2: Prepare the SQL Query that Retrieves Data from the Tfs_Warehouse Database

You calculate the MTBF by using the same approach that you used for cycle time and MTTR. All the required information is in the **CurrentWorkItemView**, in the **Tfs_Warehouse** database. Nevertheless, the resulting query is quite different.

1. Identify the column names in the **CurrentWorkItemView** that you need for the report:
   - To filter the rows, you need:
     - ♦ The column name of the **Bug** work item, which is **System_WorkItemType**.
     - ♦ The column name of the **Closed** state, which is **System_State.**

        - The column name of the team project collection and the team project, which is **ProjectPath**.
    - To calculate the MTBF for these rows, you need the column name of the **Activated Date**, which is **Microsoft_VSTS_Common_ActivatedDate**.

2. Write the query. Here is a commented, working version that you can use.

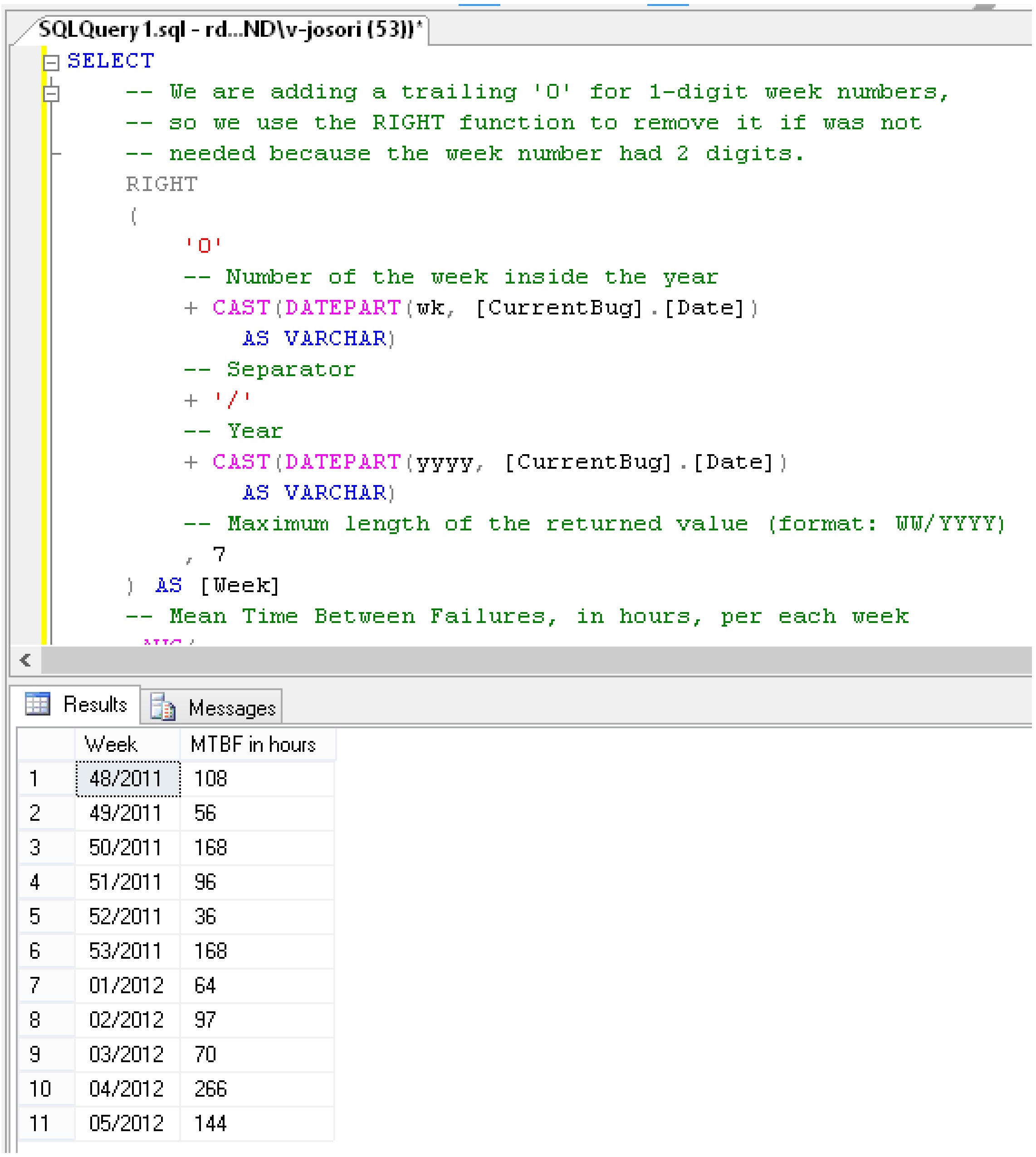
Note: Change '\YOUR-PROJECTCOLLECTION NAME\\TEAM PROJECT NAME' to your project's path name.

T-SQL

```
SELECT
        -- We are adding a trailing '0' for 1-digit week numbers,
        -- so we use the RIGHT function to remove it if was not
        -- needed because the week number had 2 digits.
        RIGHT
        (
                '0'
                -- Number of the week inside the year
                + CAST(DATEPART(wk, [CurrentBug].[Date])
                        AS VARCHAR)
                -- Separator
                + '/'
                -- Year
                + CAST(DATEPART(yyyy, [CurrentBug].[Date])
                        AS VARCHAR)
                -- Maximum length of the returned value (format: WW/YYYY)
                , 7
        ) AS [Week]
        -- Mean Time Between Failures, in hours, per each week
        ,AVG(
                DATEDIFF(hh, [FormerBug].[Date], [CurrentBug].[Date])
                ) AS [MTBF in hours]
FROM
(
        -- List of bugs, ordered by Activated date
        SELECT
                (
                        ROW_NUMBER()
                        OVER(ORDER BY [Microsoft_VSTS_Common_ActivatedDate] ASC)
                ) + 1 AS [ID]
                ,[Microsoft_VSTS_Common_ActivatedDate] AS [Date]
        FROM [dbo].[CurrentWorkItemView]
        WHERE
                [ProjectPath] = '\YOUR-PROJECTCOLLECTION NAME\TEAM PROJECT NAME'
                AND [System_WorkItemType] = 'Bug'
                AND [System_State] = 'Closed'
) AS [FormerBug]
```

```
INNER JOIN
(
       -- Just the same list of bugs as before, also ordered by Activated
       -- date, but with the rows shifted one position so we can
       -- calculate the time between one bug and the next one (notice
       -- that in this case the result of the ROW_NUMBER function is not
       -- increased by 1 to make the shift happen)
       SELECT
              (
                     ROW_NUMBER()
                     OVER(ORDER BY [Microsoft_VSTS_Common_ActivatedDate] ASC)
              ) AS [ID]
              ,[Microsoft_VSTS_Common_ActivatedDate] AS [Date]
       FROM [dbo].[CurrentWorkItemView]
       WHERE
              [ProjectPath] = '\YOUR-PROJECTCOLLECTION NAME\TEAM PROJECT NAME'
              AND [System_WorkItemType] = 'Bug'
              AND [System_State] = 'Closed'
) AS [CurrentBug]
ON [FormerBug].[ID] = [CurrentBug].[ID]
GROUP BY
       -- Grouping by week and year to calculate the average
       DATEPART(wk, [CurrentBug].[Date])
,DATEPART(yyyy, [CurrentBug].[Date])
```

3. If you paste the query into SQL Management Studio and run it, you should see one row for each week when at least one bug that was later fixed was found. You'll also see the MTBF in hours for that week. The following screenshot show the query and its results if you use the data you prepared in the first exercise.



SQLQuery1.sql - rd...ND\v-josori (53))*

```
SELECT
    -- We are adding a trailing '0' for 1-digit week numbers,
    -- so we use the RIGHT function to remove it if was not
    -- needed because the week number had 2 digits.
    RIGHT
    (
        '0'
        -- Number of the week inside the year
        + CAST(DATEPART(wk, [CurrentBug].[Date])
            AS VARCHAR)
        -- Separator
        + '/'
        -- Year
        + CAST(DATEPART(yyyy, [CurrentBug].[Date])
            AS VARCHAR)
        -- Maximum length of the returned value (format: WW/YYYY)
        , 7
    ) AS [Week]
    -- Mean Time Between Failures, in hours, per each week
```

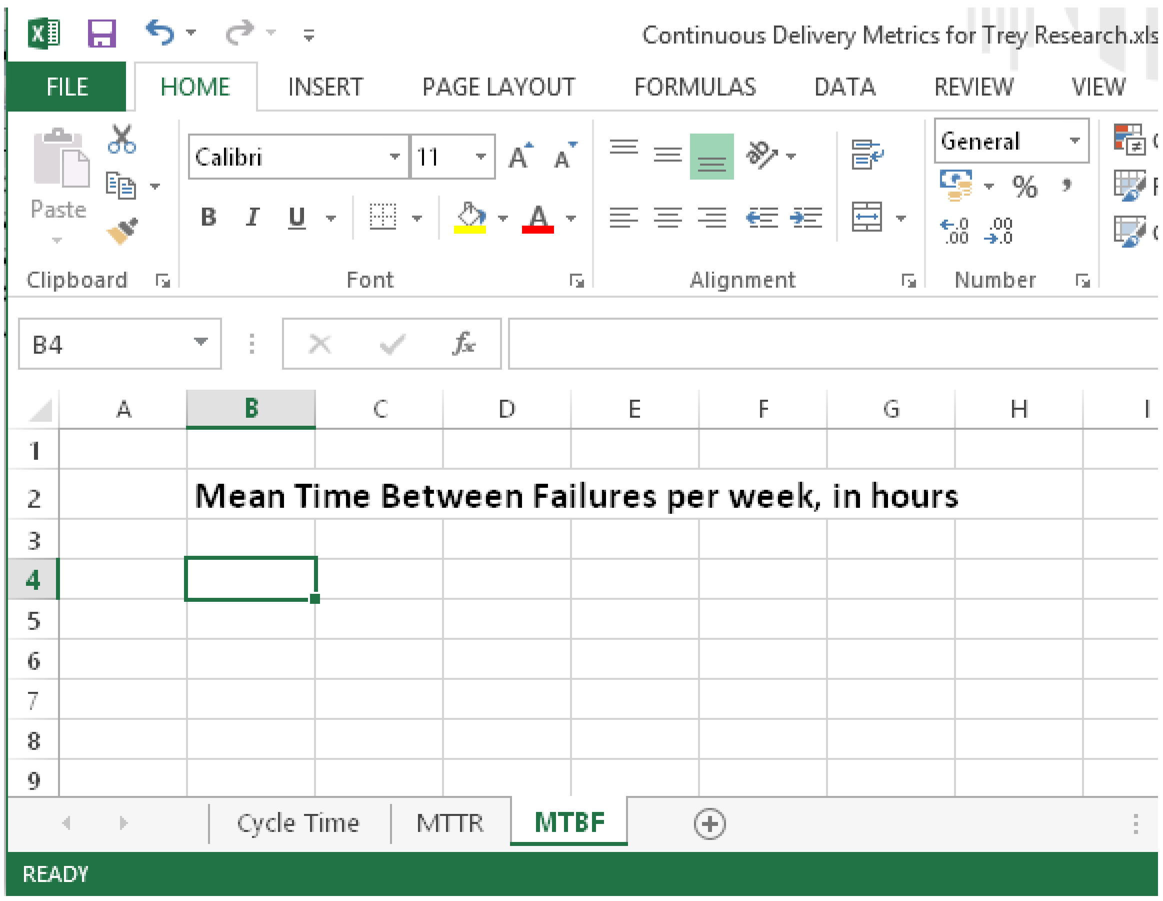
Results | Messages

	Week	MTBF in hours
1	48/2011	108
2	49/2011	56
3	50/2011	168
4	51/2011	96
5	52/2011	36
6	53/2011	168
7	01/2012	64
8	02/2012	97
9	03/2012	70
10	04/2012	266
11	05/2012	144

Task 3: Create an Excel Sheet to Present the Results

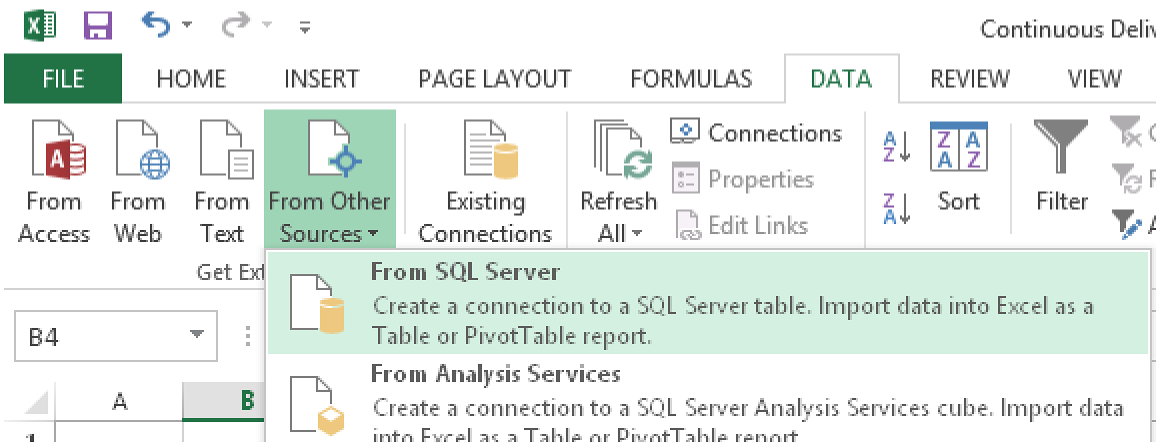
In this task you create a new Excel sheet to present the MTBF data.

1. Open the **Continuous Delivery Metrics for TreyResearch** Excel workbook.
2. Add a new sheet and name it **MTBF**. Write a title for the report in the first cells. The following screenshot shows an example.



Mean Time Between Failures per week, in hours

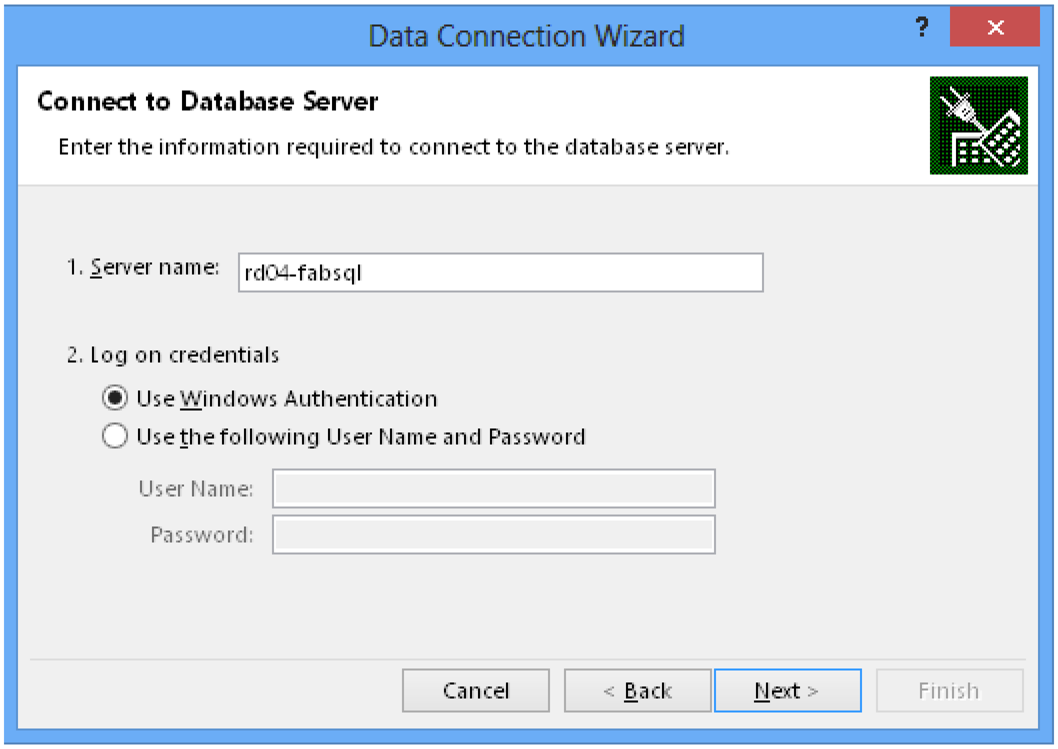
3. Select the cell where you want the data for the report to be inserted (for example, **B4**). From the **DATA** menu, select **From Other Sources**. Select **From SQL Server** from the drop-down list.



From SQL Server
Create a connection to a SQL Server table. Import data into Excel as a Table or PivotTable report.

From Analysis Services
Create a connection to a SQL Server Analysis Services cube. Import data into Excel as a Table or PivotTable report

4. The **Data Connection Wizard** appears. In the **Server name** text box, enter the server name where the **Tfs_Warehouse** database is located. If necessary, enter the credentials. Click **Next**.



Enter the information required to connect to the database server.

1. Server name: rd04-fabsql

2. Log on credentials

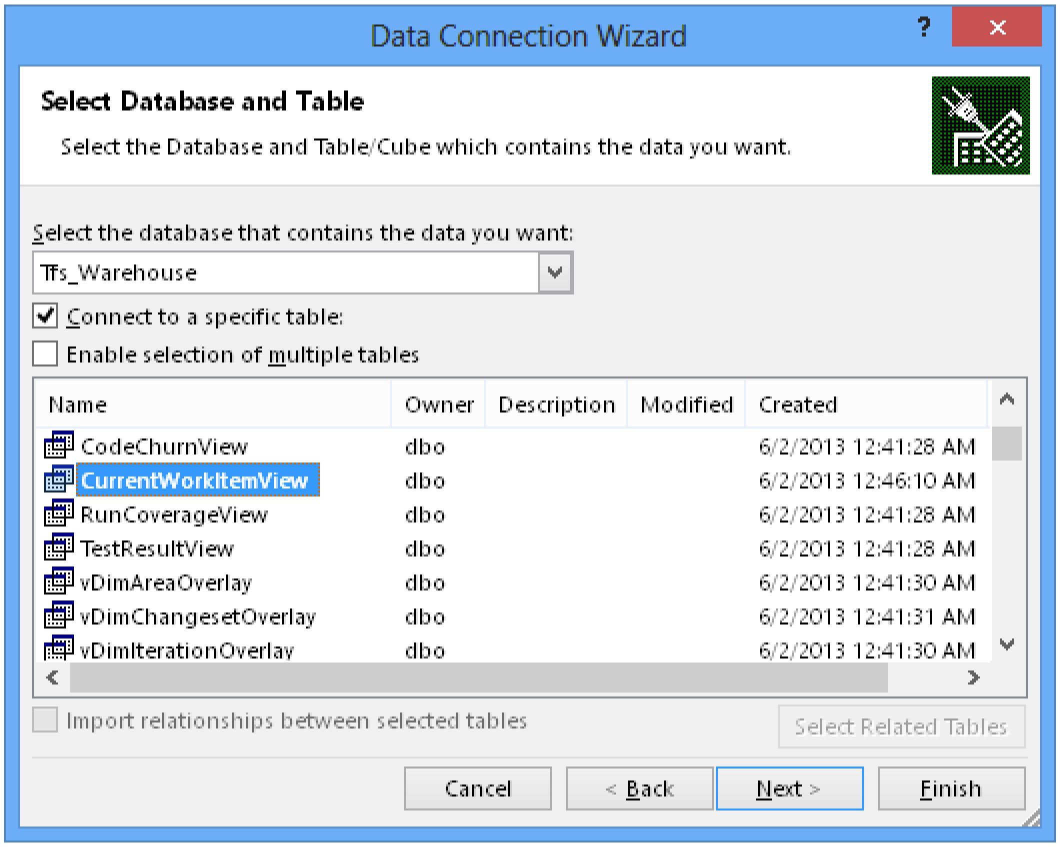
[✓ Use Windows Authentication]

Use the following User Name and Password

User Name:

Password:

5. From the **Select the database which contains the data you want** drop-down list, select **Tfs_Warehouse**. Select **CurrentWorkItemView** from the list of tables. Click **Next**.


Data Connection Wizard

Select Database and Table

Select the Database and Table/Cube which contains the data you want.

Select the database that contains the data you want:

Tfs_Warehouse

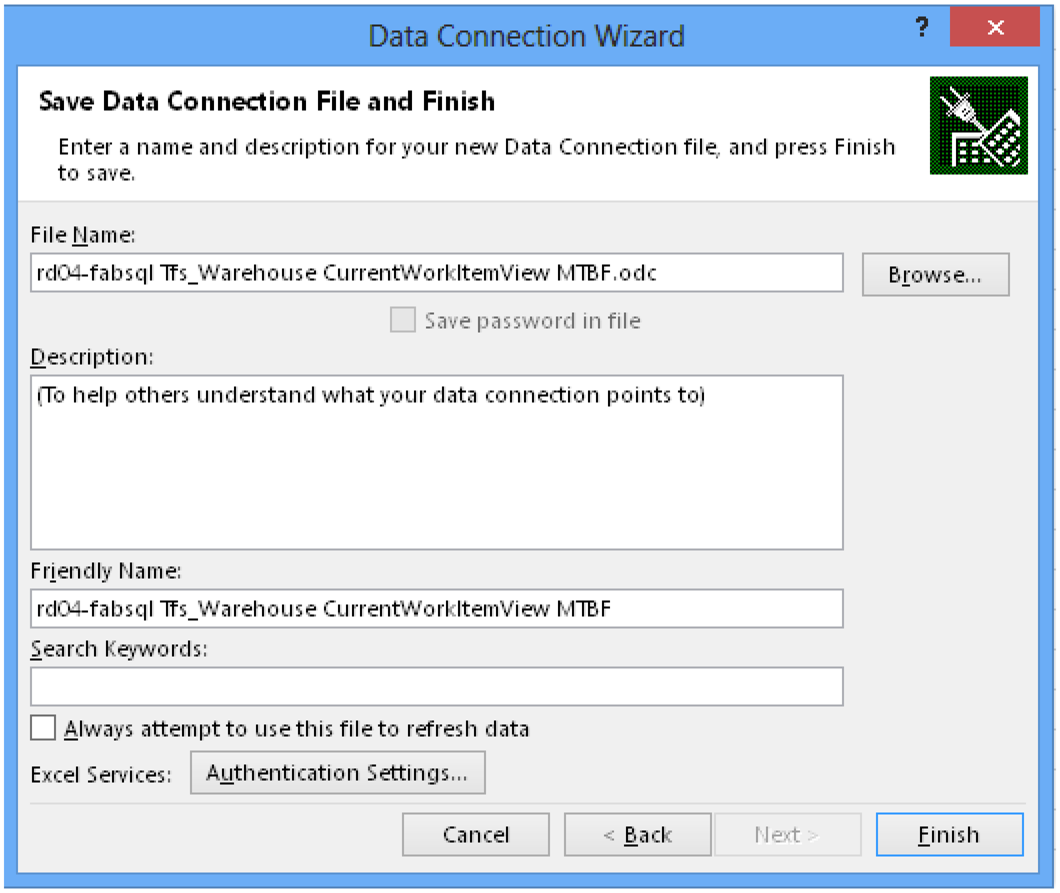
[✓] Connect to a specific table:

[ ] Enable selection of multiple tables

Name	Owner	Description	Modified	Created
CodeChurnView	dbo			6/2/2013 12:41:28 AM
CurrentWorkItemView	dbo			6/2/2013 12:46:10 AM
RunCoverageView	dbo			6/2/2013 12:41:28 AM
TestResultView	dbo			6/2/2013 12:41:28 AM
vDimAreaOverlay	dbo			6/2/2013 12:41:30 AM
vDimChangesetOverlay	dbo			6/2/2013 12:41:31 AM
vDimIterationOverlay	dbo			6/2/2013 12:41:30 AM

[ ] Import relationships between selected tables

6. Append the text **MTBF** to the proposed **File Name** and **Friendly Name** fields so that there is no conflict with other reports in the same Excel workbook. Click **Finish**.


Data Connection Wizard
Save Data Connection File and Finish
Enter a name and description for your new Data Connection file, and press Finish to save.
Browse...
Cancel
< Back
Next >
Finish

File Name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView MTBF.odc

Save password in file

Description: (To help others understand what your data connection points to)

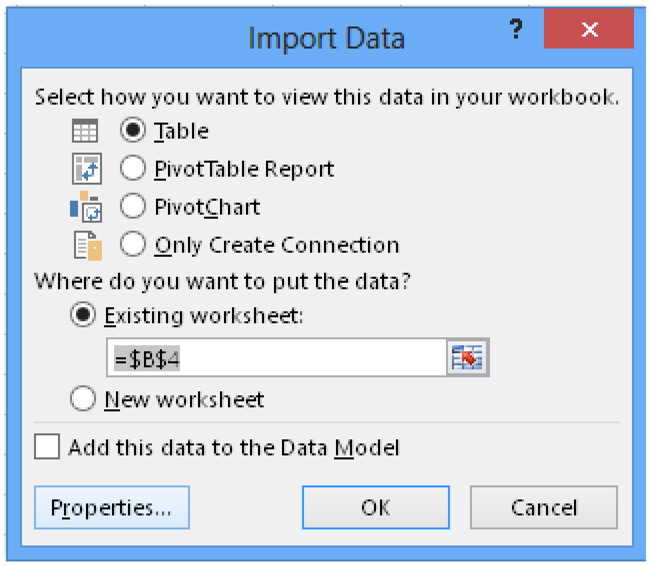
Friendly Name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView MTBF

Search Keywords:

Always attempt to use this file to refresh data

Excel Services: Authentication Settings...

7. The **Import Data** dialog box opens. Click **Properties**.



Select how you want to view this data in your workbook.

[✓ Table]

PivotTable Report

PivotChart

Only Create Connection

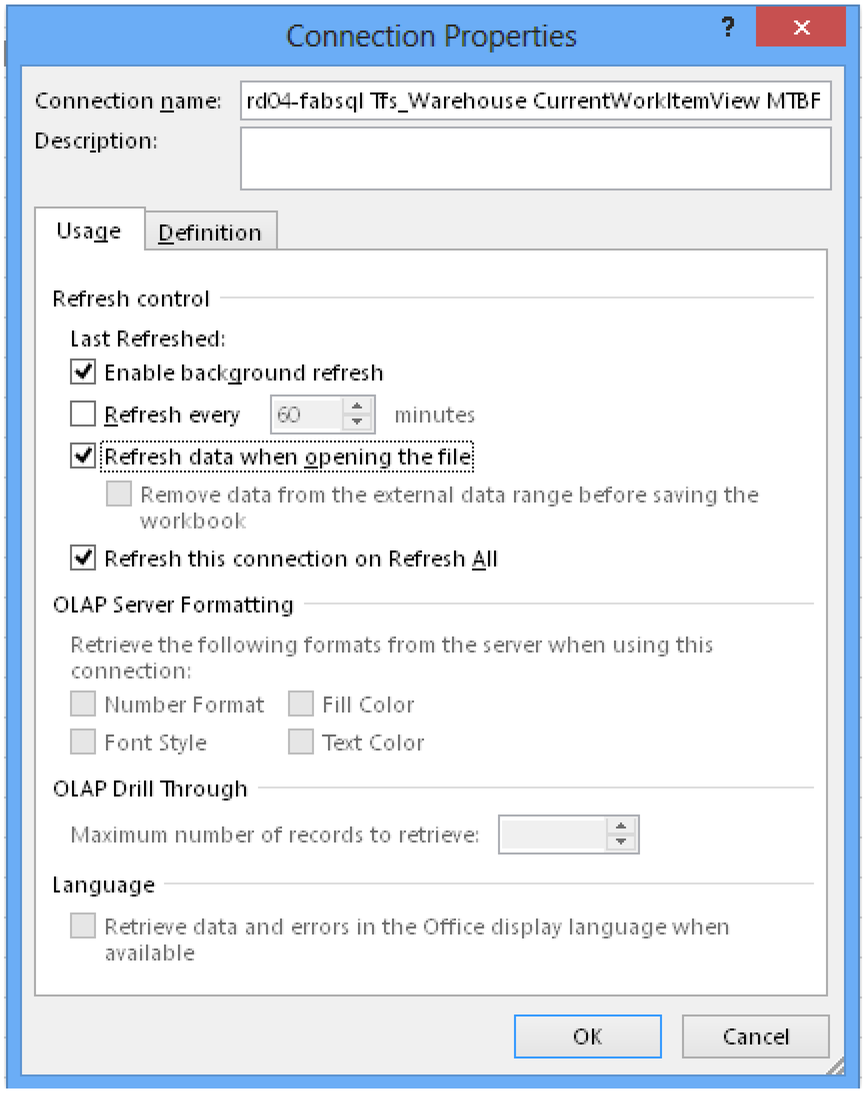
Where do you want to put the data?

[✓ Existing worksheet:] =$B$4

New worksheet

Add this data to the Data Model

8. The **Connection Properties** dialog box opens. Select the option **Refresh data when opening the file**, so that the report is updated each time the Excel workbook opens.


Connection Properties

Connection name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView MTBF

Description:

Usage | Definition

Refresh control

Last Refreshed:

- [x] Enable background refresh
- [ ] Refresh every 60 minutes
- [x] Refresh data when opening the file
  - [ ] Remove data from the external data range before saving the workbook
- [x] Refresh this connection on Refresh All

OLAP Server Formatting

Retrieve the following formats from the server when using this connection:

- [ ] Number Format
- [ ] Fill Color
- [ ] Font Style
- [ ] Text Color

OLAP Drill Through

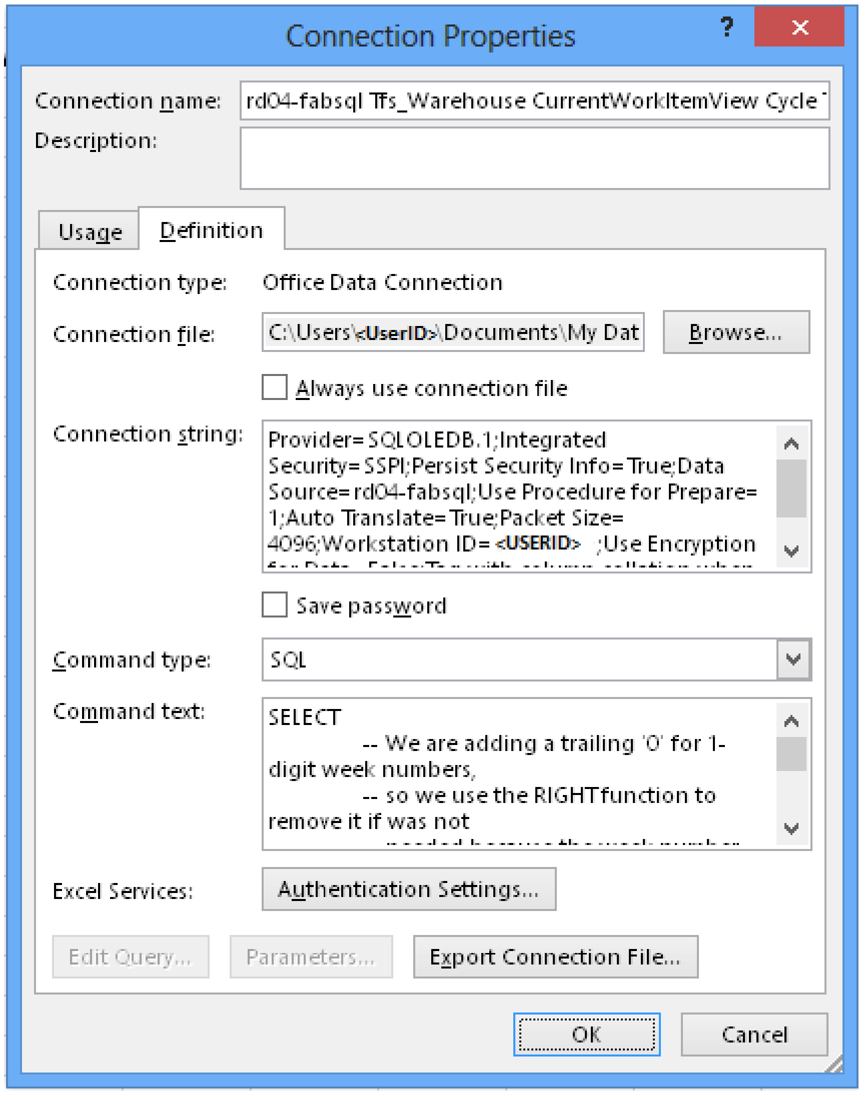
Maximum number of records to retrieve:

Language

- [ ] Retrieve data and errors in the Office display language when available

OK | Cancel

9. Select the **Definition** tab. Under the **Command type** drop-down list, select **SQL**. In the **Command text** text box paste the entire query you prepared in Task 2, step 2. Click **OK**. You will get a warning. Ignore it and click **Yes**.



Connection Properties

Connection name: rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle T

Description:

Connection type: Office Data Connection

Connection file: C:\Users\<UserID>\Documents\My Dat

☐ Always use connection file

Connection string:

```
Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=True;Data
Source=rd04-fabsql;Use Procedure for Prepare=
1;Auto Translate=True;Packet Size=
4096;Workstation ID=<USERID>  ;Use Encryption
```

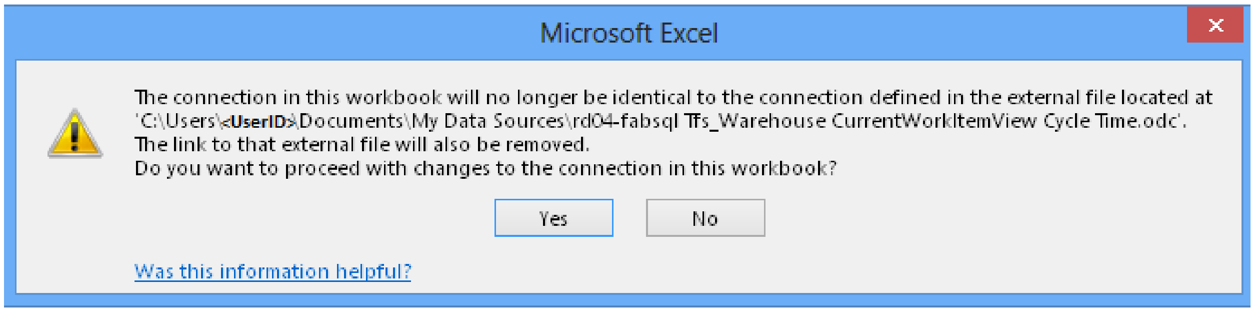
☐ Save password

Command type: SQL

Command text:

```
SELECT
            -- We are adding a trailing '0' for 1-
digit week numbers,
            -- so we use the RIGHT function to
remove it if was not
```

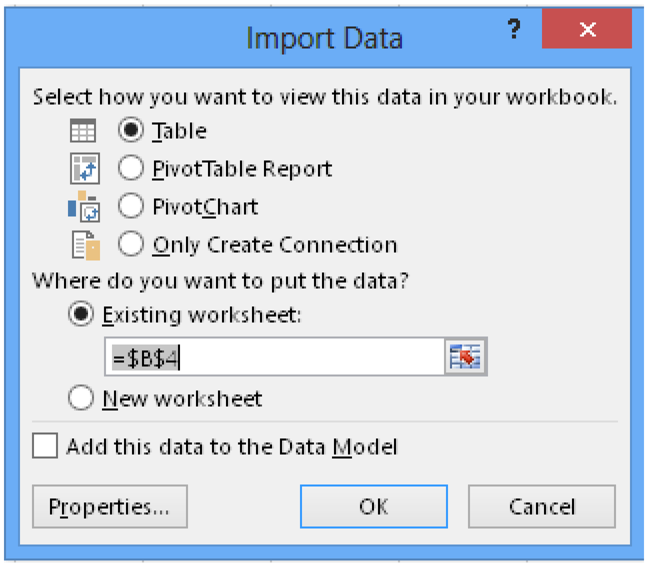
Excel Services:



Microsoft Excel

The connection in this workbook will no longer be identical to the connection defined in the external file located at 'C:\Users\<UserID>\Documents\My Data Sources\rd04-fabsql Tfs_Warehouse CurrentWorkItemView Cycle Time.odc'.
The link to that external file will also be removed.
Do you want to proceed with changes to the connection in this workbook?

10. The **Import Data** dialog reappears. Click **OK.**


Import Data

Select how you want to view this data in your workbook.

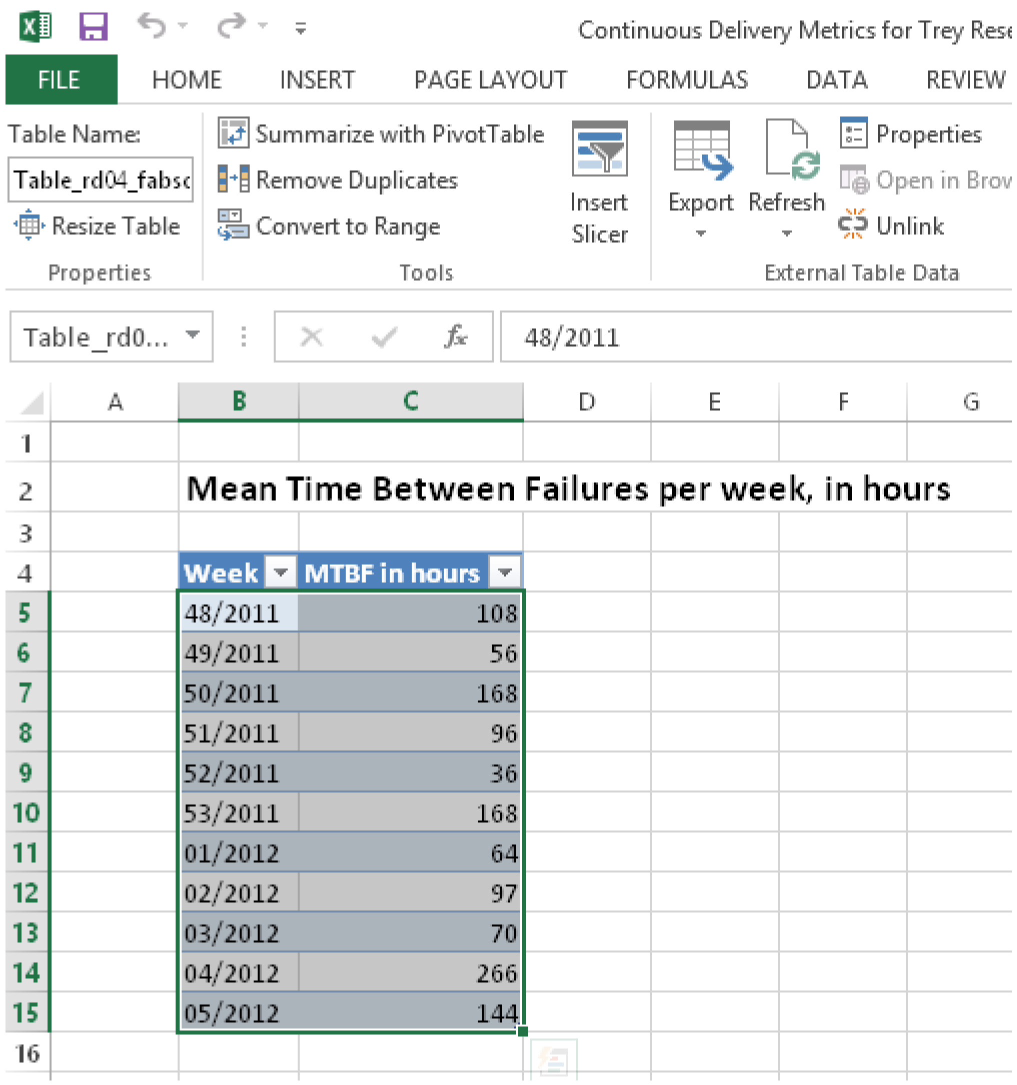
- [✓ Table]
- PivotTable Report
- PivotChart
- Only Create Connection

Where do you want to put the data?

- [✓ Existing worksheet:] =$B$4
- New worksheet

☐ Add this data to the Data Model

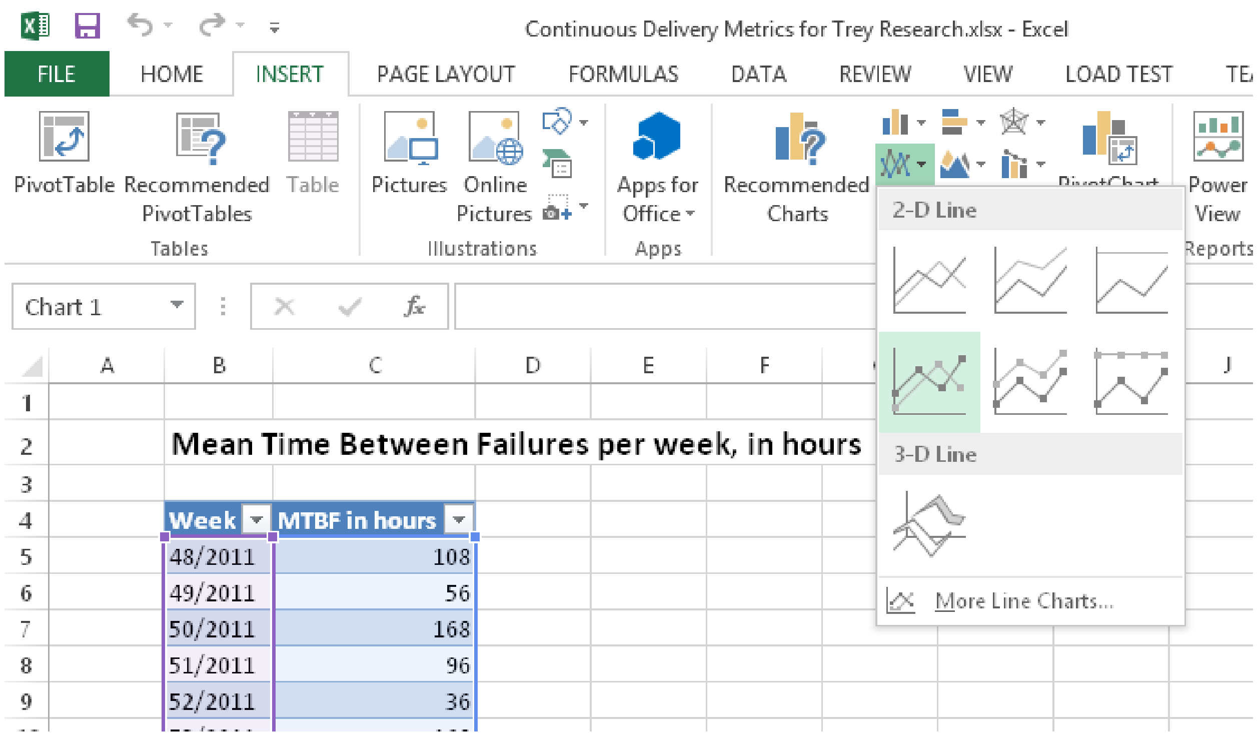
11. In your Excel sheet, you should see a table with the MTBF data in it. Select all the rows and columns containing the data. In the following screenshot, this is from **B5** to **C15**.



Mean Time Between Failures per week, in hours

Week	MTBF in hours
48/2011	108
49/2011	56
50/2011	168
51/2011	96
52/2011	36
53/2011	168
01/2012	64
02/2012	97
03/2012	70
04/2012	266
05/2012	144

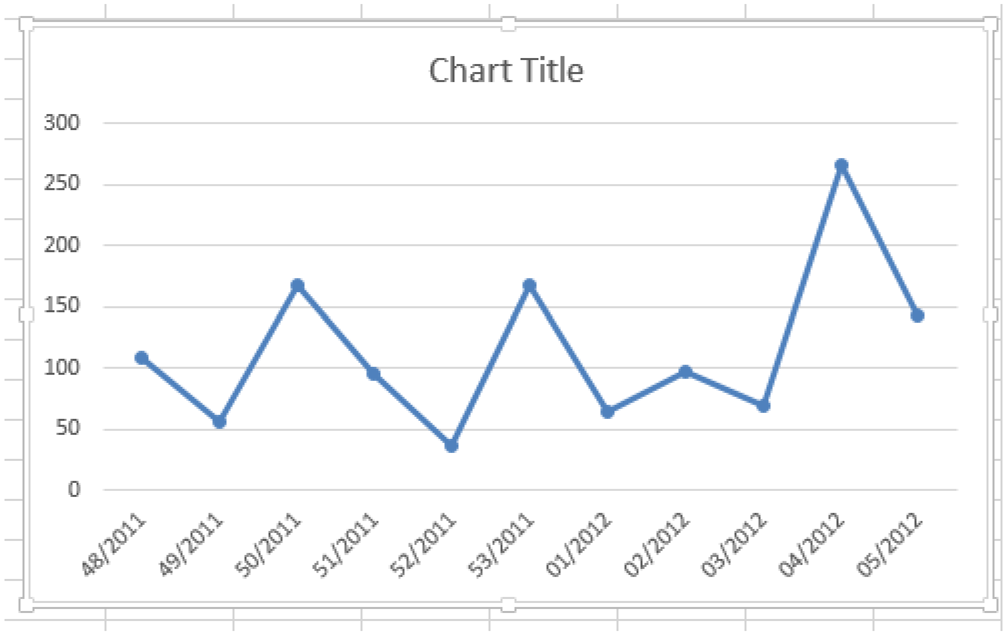
12. From the **INSERT** menu, click **Insert Line Chart**. Select **Line with Markers**.



Mean Time Between Failures per week, in hours

Week	MTBF in hours
48/2011	108
49/2011	56
50/2011	168
51/2011	96
52/2011	36

13. You should see a chart that resembles the following screenshot.



14. Format the resulting chart to suit your preferences. In the following screenshots, the **Chart Title** is removed and a linear trend line is added. The chart is widened so that the weeks and years at the bottom are clearer, and the chart is positioned near the table.

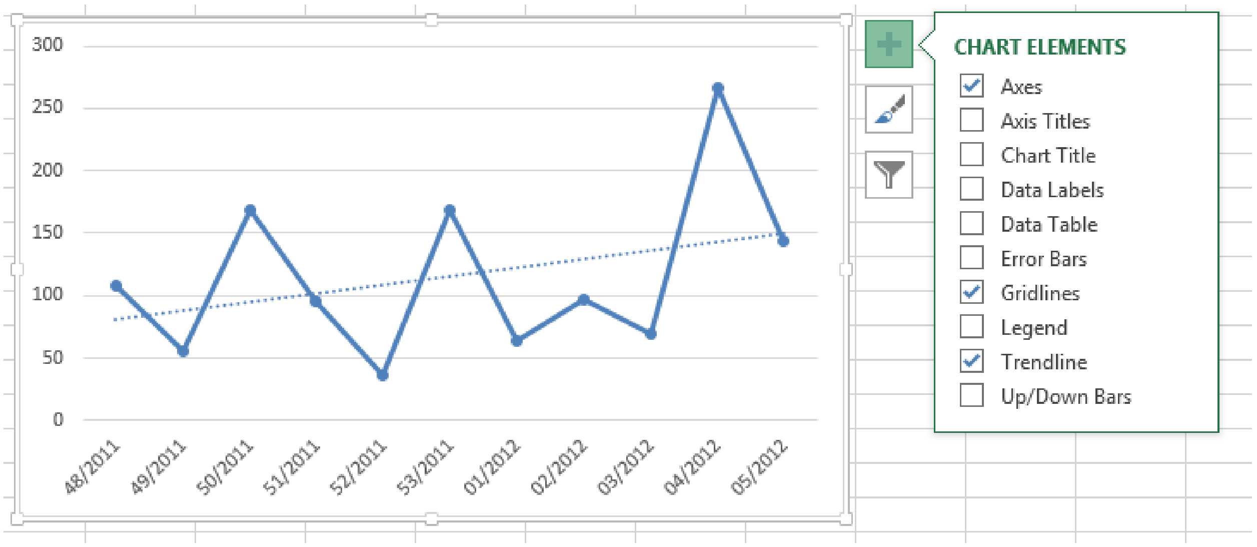
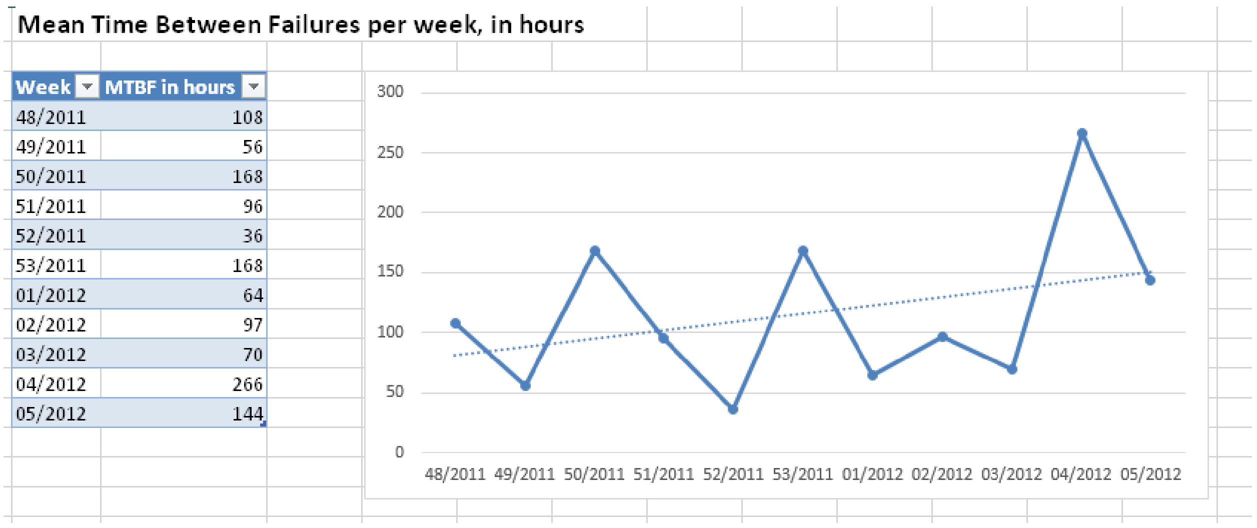


CHART ELEMENTS

- [✓] Axes
- [ ] Axis Titles
- [ ] Chart Title
- [ ] Data Labels
- [ ] Data Table
- [ ] Error Bars
- [✓] Gridlines
- [ ] Legend
- [✓] Trendline
- [ ] Up/Down Bars



Mean Time Between Failures per week, in hours

Week	MTBF in hours
48/2011	108
49/2011	56
50/2011	168
51/2011	96
52/2011	36
53/2011	168
01/2012	64
02/2012	97
03/2012	70
04/2012	266
05/2012	144

15. Save and close the Excel workbook. The next time you open it, both the table and the chart will be updated to reflect the additional bugs that the team has found and fixed.

Task 4: Understanding the Results

In this task you learn how to interpret the MTBF data you've gathered. Here are some conclusions you can make about the Trey Research project.

Initial MTBF Results

Around mid-November of 2011, the team initially had an MTBF of 108 hours. This means that they were finding production bugs about every four or five days.

Changes in MTBF

The MTBF is still unpredictable. Although the team has improved their release process, they still have to improve their quality. This means they need to examine where they should focus their testing efforts and the types of tests they need to perform. Nevertheless, the trend shows that they are steadily increasing (improving) their MTBF.

Summary

In this HOL, you learned how to create custom TFS reports that track three metrics that are key to evaluating the efficiency of a continuous delivery pipeline. The first metric is cycle time, which is the time that elapses from the moment you decide to implement a feature to the moment you release the feature.

The second metric is MTTR, which is the average time elapsed from the moment a problem is found in the production environment to the moment that the problem is fixed. The MTBF metric is the average time elapsed from the moment one problem is found in the production environment to the moment that the next problem is found in that environment.

In general, the steps to create a report are:

1. Identify the information you need for the report.
2. Determine how to present the information.
3. Prepare the SQL query that retrieves the data from the **Tfs_Warehouse** database.
4. Create an Excel workbook with a connection to the **Tfs_Warehouse** database that runs the query and presents the results in tabular and chart formats.

Helpful Tips

This section contains information on using the data with other metrics and other templates.

Cycle Time Reports

NOTE: An User Story could pass any amount of time in the **New** state before being activated, but we don't consider it part of the Cycle Time – it would be part of a different metric, the **Lead Time**. That metric is important for Lean and Agile process, but it is not a key indicator for Continuous Delivery and for improving the development process and the release pipeline.

TIP: If you want to prepare the report for the Visual Studio Scrum process template, and have a look at the corresponding equivalent fields and transitions, you will notice that the **Activated Date** field is not updated when a **Product Backlog Item** gets to the **Commited** state. But you can easily modify the template so it gets updated and cycle time can be measured.

TIP: For the MSF for CMMI template the field to use is also **Activated Date**, but in the **Requirement** work item type.

TIP: This is exactly the same for Product Backlog Items in the Visual Studio Scrum template; look for the **Closed Date** that gets updated when the PBI gets to the **Done** state.

TIP: For the MSF for CMMI template, you will need to use the **Resolved Date** field, that gets updated when the Requirement gets to the Resolved state.

TIP: It may be useful to have the units (days), the grouping (weeks) and the date for the oldest Story to return, as parameters for the report. That way we could easily change the report to track, for example, the average cycle time per day, in hours, in the last three months (or whichever other version of the report you care about). This is one of the features that you can have easily implemented if you build your report in SSRS.

Tips for MTTR reports

TIP: If you want to prepare the report for the Visual Studio Scrum process template, and have a look at the corresponding equivalent fields and transitions, you will notice that the workflow and values for the fields are a bit different. The **Activated Date** field is not updated when a **Bug** gets to the **Approved** state. But you can easily modify the template so it gets updated and MTTR can be measured.

TIP: For the MSF for CMMI template you can use also the **Bug** work item type, and look for the same **Activated Date** field.

TIP: This is exactly the same for Bugs in the Visual Studio Scrum template; look for the **Closed Date** that gets updated when the Bug gets to the **Done** state.

TIP: For the MSF for CMMI template, you will need to use the **Resolved Date** field, that gets updated when the Bug gets to the Resolved state.

TIP: It may be useful to have the units (days), the grouping (weeks) and the date for the oldest Bug to return, as parameters for the report. That way we could easily change the report to track, for example, the MTTR per day, in minutes, in the last three months (or whichever other version of the report you care about). This is one of the features that you can have easily implemented if you build your report in SSRS.

Copyright