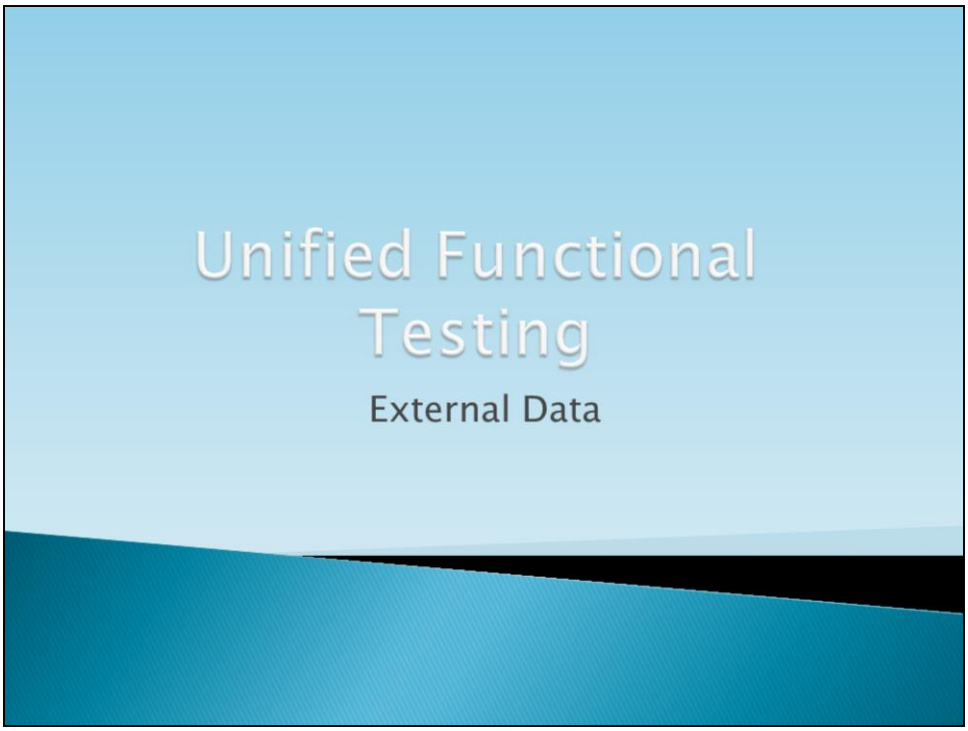

Unified Functional Testing
External Data

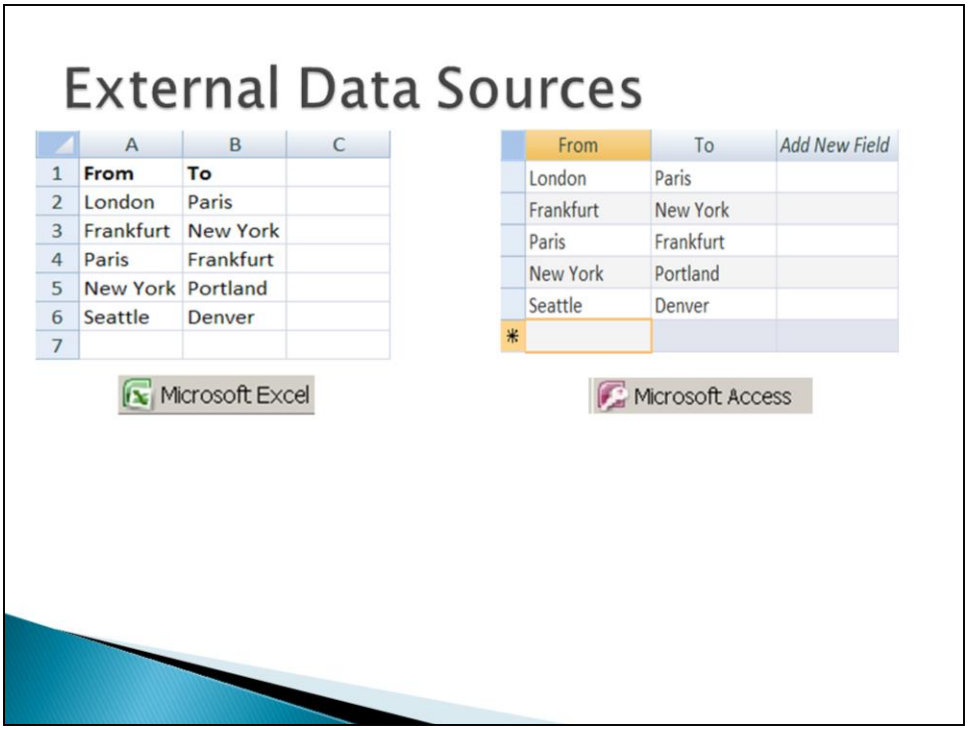
Lesson Objectives

By the end of this Lesson you will be able to:

- Import data from and export data to a Microsoft Excel worksheet.
- Use the Connection and RecordSet objects to query a database.
- Import and export data to text files.

Topics

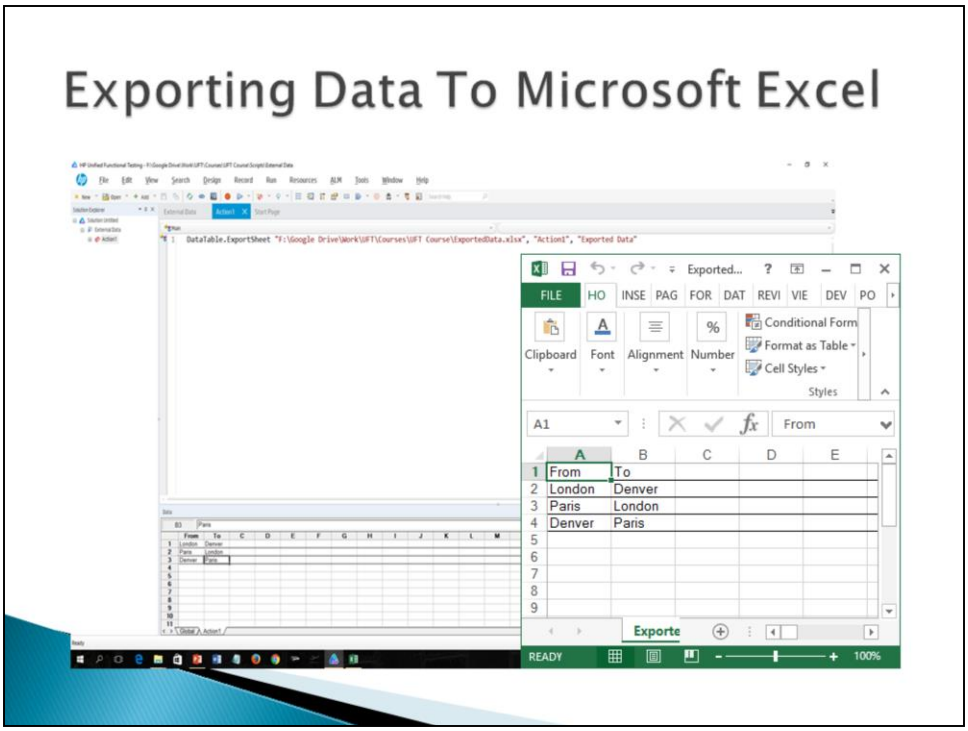
1. External resources types
2. Working with excel
3. Working with text files



You may want to test an application using different data sets.

For Example , The From and To column could contain in external data sources, such as Microsoft Excel ,Microsoft Access or TOAD.

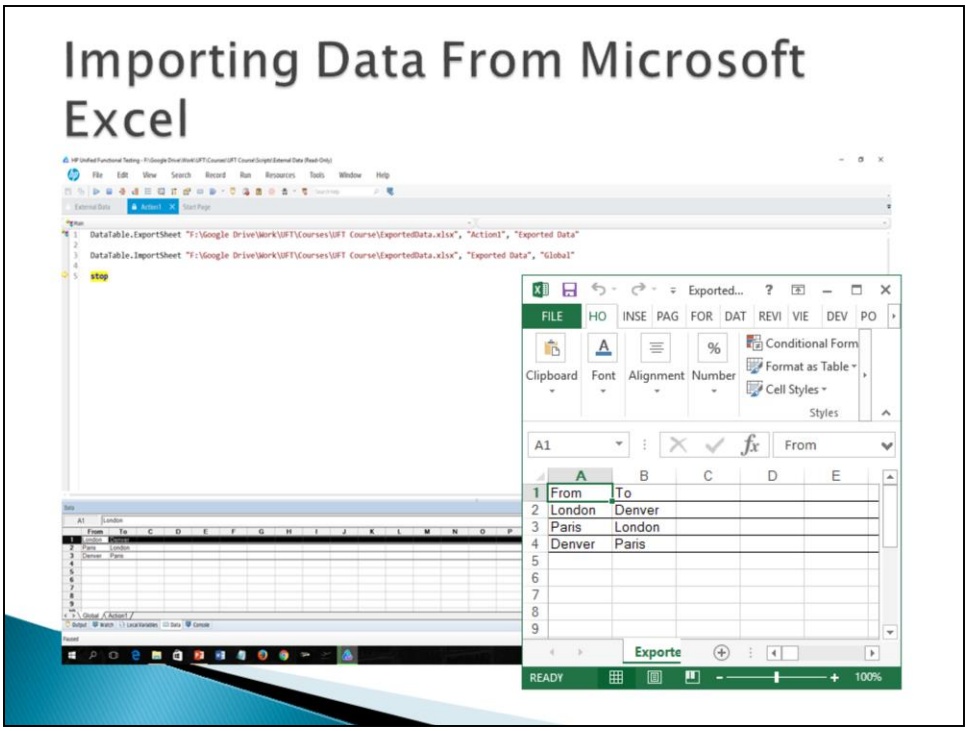
If you store the test data for a test in an external data source, you must retrieve the data from the data sources to run the test. You can also retrieve data at run-time and store it in an external data source for future use.



Use the **DataTable.ExportSheet** method to export a sheet of the run-time data table to a Microsoft Excel file. The data can be stored in an existing Microsoft Excel file or used to create a new Microsoft Excel file.

Export the data saved in the run-time data table to Microsoft Excel if you want to access the data in another UFT test or component.

Use the **DataTable.Export** method to export run-time data table sheets to a Microsoft Excel file.



Use the **DataTable.ImportSheet** method to import a worksheet of a Microsoft Excel file to a sheet in the run-time data table.

After the data is imported from a Microsoft Excel file into DATA TABLE, the DATA TABLE can drive the iterations of the test or action.

Use the data in the first row of the Microsoft Excel worksheet as column names for the UFT DATA TABLE.

Use the **DataTable.Import** method to import all sheets that exist in Microsoft Excel file to sheets that exist in the run-time Data Table.

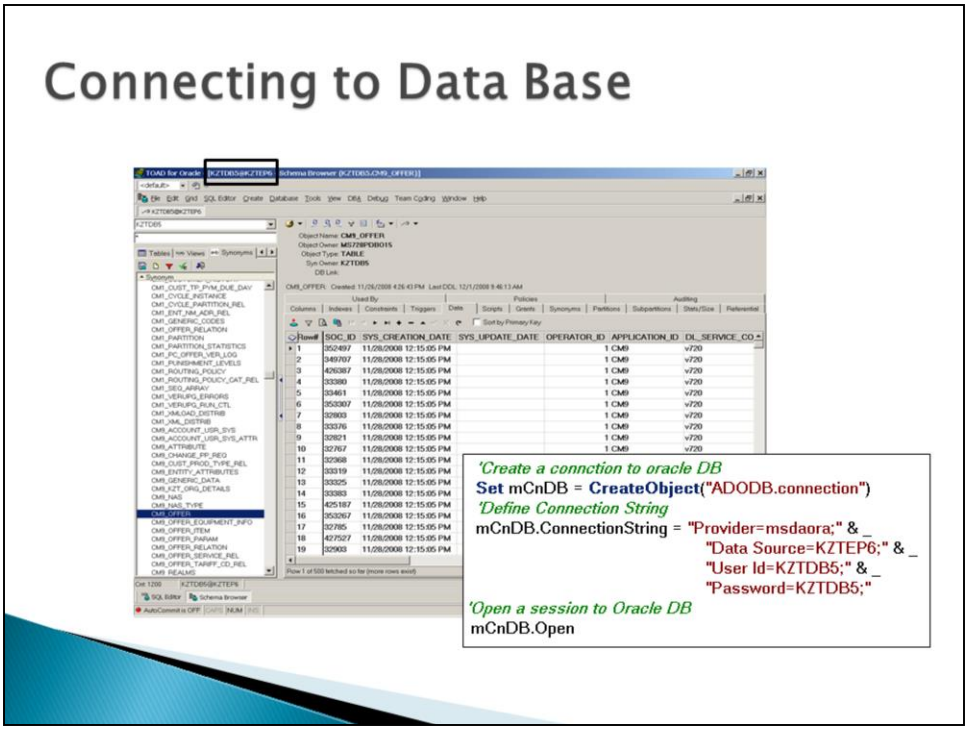
Retrieve Data From Data Base

Some steps to retrieve data from a database:

- Connect to the database.
- Execute a Structured Query Language (SQL) query.
- Examine the query results.
- Close the database session.

ActiveX Data Objects – ADO

- Microsoft ADO (ActiveX Data Objects) is a Component object model object for accessing data sources.
- There are two commonly used ADO Objects:
  - Connection – Opens a session with a database and executes a query.
    - The Connection object can be used to access any database with an Open Database Connectivity (ODBC) driver.
  - Recordset – Used to hold a set of records from a database table.
    - A Recordset object consist of records and columns (fields).

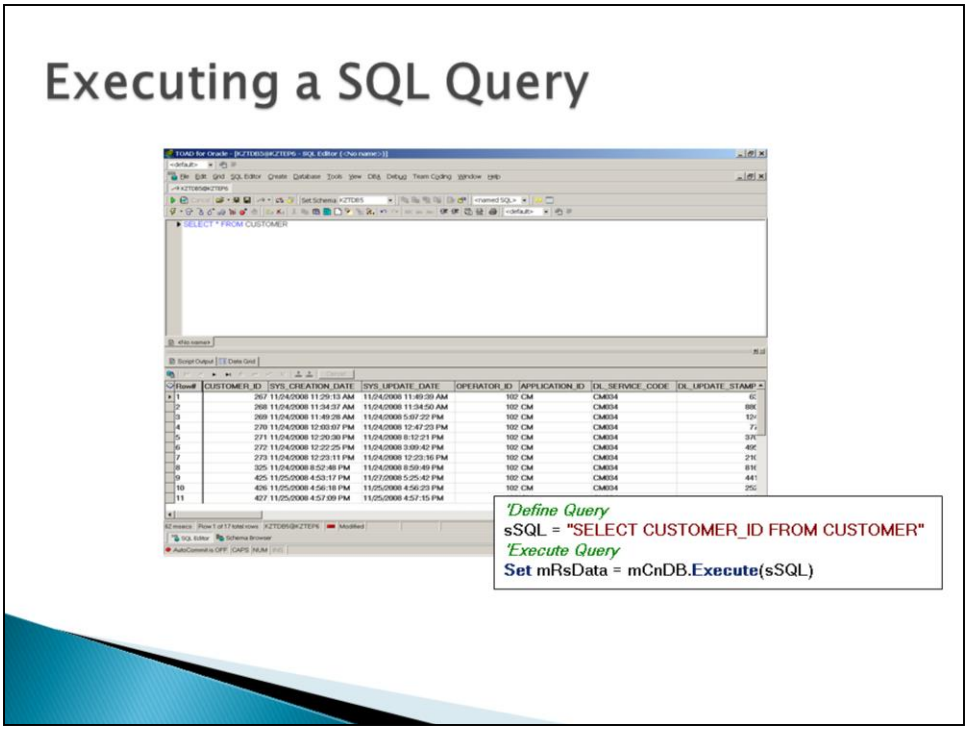


To connect to a database from a UFT script:

1. Create a Connection object.
2. Call the Open method of the Connection object.

Use the **ConnectionString** property of a Connection object to provide information about a database.

The **Open** method of the ConnectionString property, connects to a database.



After the database connection is established, run an SQL query against the database. Use the Execute method of the Connection object to retrieve data from a database.

The Execute method accepts an SQL statement as an input and returns a RecordSet object when the run completes.

Examining the Query Results

Properties/Methods	Description
BOF EOF	These properties determine if you are at the boundaries of the RecordSet object. BOF- Beginning Of File EOF- End Of File
MoveNext	Moves one record forward in the RecordSet object.
MovePrevious	Moves one record backward in the RecordSet object.
Move	moves multiple records forward or backward at a time.
Fields.Count	Indicates the number of columns of data returned by the SQL query.
Fields.Item("MyColumn").Value Fields("MyColumn")	Return the value saved in the specified column of the current record in the RecordSet object.

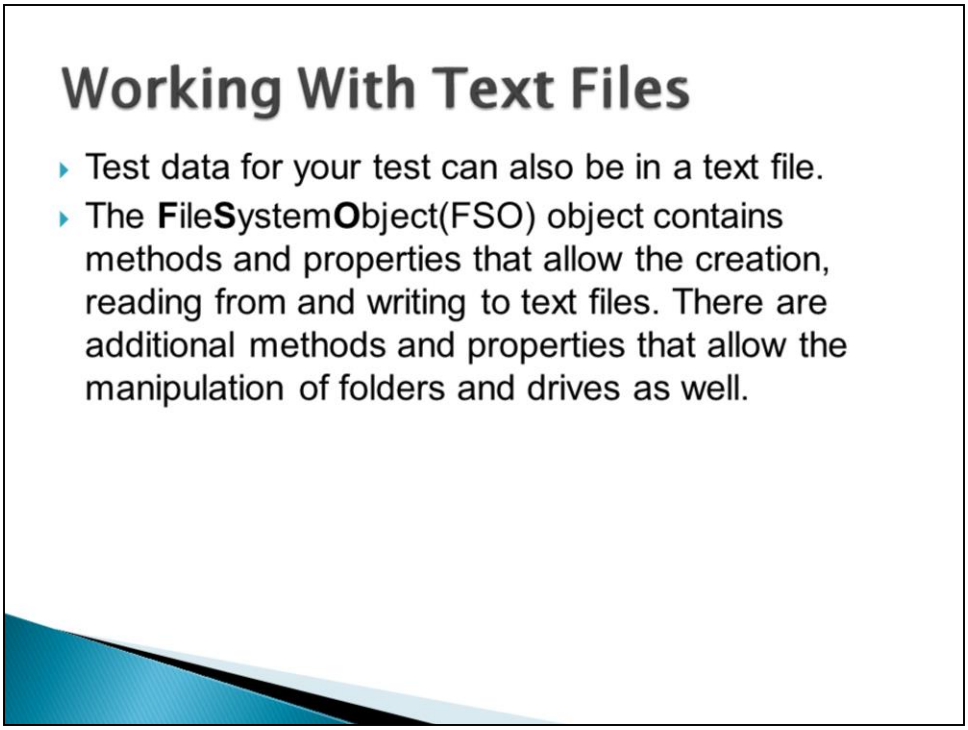
Closing The Data Base Session

After examining the output of an SQL query, close the database session by using the Close method.

Close methods are provided for the RecordSet and Connection objects.

After the RecordSet or Connection object is closed, you can set the variable to Nothing.

```
mCnDB.Close
mRsData.Close
Set mCnDB=Nothing
Set mRsData=Nothing
```



Test data for your test can also be in a text file.

The FileSystemObject object contains methods and properties that allow the creation, reading from and writing to text files. There are additional methods and properties that allow the manipulation of folders and drives as well.

Reading From A File

- The OpenTextFile method of the FileSystemObject object opens a text file and returns a TextStream object. The OpenTextFile method accepts a constant as an argument. This argument specifies the purpose for the file as reading, writing, or appending.
- A TextStream object facilitates sequential access to a text file.
- The TextStream object has the following methods for reading text from a file:
  - **Read**: Reads a specified number of characters from the TextStream object and returns a string.
  - **ReadLine**: Reads from a TextStream object until a newline character is encountered and returns a string.
  - **ReadAll**: Reads an entire TextStream object and returns a string. This method is not recommended for large files.

Writing To A File

- Use the OpenTextFile or the CreateTextFile method of the FileSystemObject object to create a new file for writing text to the file. Both the methods return a TextStream object.
- The TextStream object has the following methods for writing text to a file:
  - **Write**: Writes a string to an open file.
  - **WriteLine**: Writes a string to an open file, and adds a newline.
  - **WriteBlankLines**: Writes a given number of empty lines to an open file.

Commonly FSO Methods

Methods	Description	Example
CreateTextFile	Creates a specified file name and returns a TextStream object that can be used to read from or write to the file.	Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
DeleteFile	Deletes a specified file.	fso.DeleteFile("c:\testfile.txt")
FileExists	Returns True if a specified file exists; False if it does not.	nRc = fso.FileExists("c:\testfile.txt")
MoveFile	Moves one or more files from one location to another.	fso.MoveFile "c:\testfile.txt", "c:\Test"
MoveFolder	Moves one or more folders from one location to another.	fso.MoveFolder "c:\Training", "c:\Test"
GetParentFolderName	Returns a string containing the name of the parent folder of the last component in a specified path.	GetTheParent = fso.GetParentFolderName(Drivespec)

What's Next?

- Review Questions
- Next Lesson
  - The next lesson in the course is:
    Automation Testing Methodology Data

End of Lesson