ADO.NET Summary

1. Data Set object
   1.1. Tables - DataTable objects
   Collections
   - Columns - A DataColumn collectiom
   - Rows - A DataRow collection
   - Constraints – A collection of UniqueConstraints, and ForeignKeyConstraints.

```
**** UniqueConstraint *****
myDataColumn.Unique = true;  //Creation of a constraint

UniqueConstraint myConstraint = new UniqueConstraint( myDataSet.Table[0].Columns ["ID"])
UniqueConstraint myConstraint = new UniqueConstraint(myDataColumn);
myDataTable.Constraints.Add(myConstraint);

DataColumn[] myColumns = new DataColumn[2];
myColumns[0] = EmployeesTable.Columns["FirstName"];
myColumns[1] = EmployeesTable.Columns["LastName"];
UniqueConstraint myConstraint = new UniqueConstraint(myColumns);
EmployeesTable.Constraints.Add(myConstraint);
**** ForeignKeyConstraint *****
ForeignKeyConstraint myConstraint = new
           ForeignKeyConstraint(CustomersTbl.Columns["CustomerID"],
                                OrdersTbl.Columns["CustomerID"]);
CustomersTbl.Constraints.Add(myConstraint);
 - UpdateRule
 - DelecteRule
 - AcceptRule
 (Cascade, None, SetDefault, SetNull)
```

   1.2. Relations - A DataRelation collection

```
DataRelation CustomersOrders = new DataRelation("Data Relation 1", column1,
                                                                   column2);
myDataSet.Relations.Add(CustomersOrders);
DataRow[] ChildRows;
DataRow ParentRow;
ChildRows = myDataSet.Tables["Customers"].Rows[1].GetChildRows(CustomersOrders);
ParentRow = myDataSet.Tables["Orders"].Rows[5].GetParentRow(CustomersOrders);
```

   1.3. ExtendedProperties –A PropertyCollection collection

```
myDataSet.ExtendedProperties.Add("password", "1#HgeC");
strVar = myDataSet.ExtendedProperties["password"].ToString();
```

2. Data Provider (System.Data.OleDB, System.Data.SqlClient)

   2.1. Connection object – (OleDbConnection, SqlConnection, ConnectionString, BeginTransaction)

   2.2. Command object
      2.2.1. Basic properties
      - Connection
      - CommandType (Uses ExecuteReader)
         - CommandType .Text
         - CommandType .StoredProcedure
         - CommandType .TableDirect
      - CommandText

      2.2.2. Basic Methods
      - ExecuteNonQuery
      - ExecuteScalar (return object)
      - ExecuteReader
      - ExecuteXmlReader

      2.2.3. Parameters collection (Uses OleDbParameter class, and SqlParameterClass)

```
SqlParameter myParameter = new SqlParameter();
MyParameter.ParameterName = "@name";
MyParameter.Value = "Daniel";
myCommand.Parameters.Add(myParameter)

myCommand.Parameters[0].Value = "Hello World";
myCommand.Parameters["@name"].Value = "Hello World";";
```

2.2.4. Transaction property

```
System.Data.OleDb.OleDbTransaction myTransaction = null;
try
{
   myConnection.Open();
   myTransaction = myConnection.BeginTransaction();
   Update1.Transaction = myTransaction;
   Update2.Transaction = myTransaction;
   Update1.ExecuteNonQuery();
   Update2.ExecuteNonQuery();
   myTransaction.Commit();
}
catch (Exception ex)
{
   myTransaction.Rollback();
}
finally
{
   myConnection.Close();
}
//Other way
ContactMgmt.Open();
transDelete = ContactMgmt.BeginTransaction (IsolationLevel.ReadCommitted);
SqlCommand cmdDelete = new SqlCommand("DELETE FROM Contacts WHERE ContactID= " +
                                + intContactID.ToString(), ContactMgmt, transDelete);
```

2.3. DataReader object

```
myConnection.Open();
System.Data.OleDb.OleDbDataReader myReader = myOleDbCommand.ExecuteReader();
//System.Data.SqlClient.SqlDataReader mySqlReader;
while (myReader.Read())
{
  strCustomer = myReader["Customers"].ToString();
  object myObject = myReader[3];
  myString = myReader.GetString(3);
  IndexValue = myReader.GetOrdinal("CustomerID");
  strCustomer = myReader.GetString(IndexValue);
}
myReader.Close();
myConnection.Close;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Do
{
   while (myReader.Read())
   {
      // Add code here to loop through the records of the current result set
   }
} while (myReader.NextResult());
```

2.4. DataAdapter object

2.4.1. Fill method

2.4.2. SelectCommand, UpdateCommand, InsertCommand, DeleteCommand properties

2.4.3. RowUpdated event

Provides the SqlRowUpdateEventArgs or the OleDbRowUpdateEventargs

- Command, Errors, RecordsAffected, Row,
- Status
    - UpdateStatus.Continue
    - UpdateStatus.ErrorOcurred
    - UpdateStatus.SkipAllRemainderRows
    - UpdateStatus.SkipCurrentRow

```
DaCustomers.RowUpdated += new SqlRowUpdatedEventHandler(myDataAdapter_RowUpdated);

private void myDataAdapter_RowUpdated(object sender,
                                System.Data.SqlClient.SqlRowUpdatedEventArgs e)
{
   if (e.Status == UpdateStatus.ErrorsOccurred)
   {
      MessageBox.Show(e.Errors.Message);
      e.Status = UpdateStatus.SkipCurrentRow;
   }
}
```

3. DataView

3.1. DataView

```
DataView myDataView = new DataView(myDataTable);

DataView myDataView = new DataView();
myDataView.Table = myDataTable;

myDataView.Sort = "State DESC, City";

myDataView.RowFilter = "City = 'Seattle' AND State = 'WA'";
myDataView.RowFilter = "City LIKE 'Se*t%e'";

myDataView.RowStateFilter = DataViewRowState.Unchanged | DataViewRowwState.Added;

myDataView [0] ["ID"] = "Test";

DataRowView myDataRowView = myDataView.AddNew();
MyDataRowView ["item'] = "test";

// Unchanged, Added, Deleted, OriginalRows, CurrentRows, ModifiedCurrent, ModifiedOriginal
```

3.2. DataViewManager

```
  //Creation with the dataset as a parameter
DataViewManager myDataViewManager = new DataViewManager(myDataSet);
  //Creation and assigning the dataset property
DataViewManager myOtherDataViewManager = new DataViewManager();
myOtherDataViewManager.DataSet = myOtherDataSet;

  //Set the RowFilter Property of the DataView Associated with the Customers table
myDataViewManager.DataViewSettings["Customers"].RowFilter = "State = 'WA'";
  //To retrieve a DataView from the DataViewManager use:
DataView myDataView;
myDataView = myDataViewManager.CreateDataView(DataSet1.Tables[0]);
```

4.