C#

.NET Framework

- CLR
- FCL / BCL
- CTS
- CLS
- MSIL
- Automatic Memory Management
- Garbage Collection
- Managed and Unmanaged Code
- JIT Compiler

C# Basics

- Value Type
- Reference Type
- Pre-defined Types(size, Minimum & Maximum Values)
- Structure
- Enum
- Performing Bitwise Manipulation using Enum & Flags Attribute
- Enum.GetNames()
- Boxing and Unboxing
- Type Casting
- Implicit Conversion and Explicit Conversion
- Stack Memory vs. Heap Memory
- const, readonly, out , ref & params Keyword
- compiler options (/target /out & /r)
- Converting string to Other primitive datatypes
- Converting string to Enum using Enum.Parse()
- String.Format() method
- DateTime.Parse(),DateTime.ParseExact() & DateTime.TryParse()

OOP Concepts

- Class
- Object
- Access Specifiers (private, protected, public, protected internal)
- Constructor, Default Constructor, Static Constructor
- static keyword
- this keyword

- Method Overloading
- Operator Overloading (Operators which cannot be overloaded and how to overload Pair operators like < & >)
- Inheritance (C# does not support Multiple Inheritance)
- Sealed class
- base keyword
- Method overriding
- Compile time polymorphism & Run time Polymorphism
- virtual keyword
- override keyword
- Usage of is and as keyword
- new keyword(Allocation Memory and explicitly hides a member that is inherited from a base class)
- Abstract class and Abstract Method
- Usage of Interface
- Implicit and Explicit Interface implementation
- Usage of Properties & Indexers
- Partial Classes

Exception Handling

- try-catch block
- Multiple Catch Statements
- Nested try-catch
- Usage of finally block
- Usage of throws keyword
- Creating Custom Exception derived from ApplicationException

Collections & Generics

- Iterating collection using for loop, for each loop & IEnumerator
- Usage of ICollection, IList, IEnmerator, IEnumerable, IComparable & IComparer interfaces
- Limitation in Collections
- Usage of foreach Loop
- Usage of Interface in Collections
- TypeSafety feature in Generics
- Usage of Generics
- Usage of Dictionaries
- Usage of yield keyword

C# 3.0 features

- Implicitly Typed Variables & its limitations
- Usage of Property Initializers
- Usage of Collection Initializers
- Usage of Object Initializers
- Anonymous Types
- Extension Methods usage and Rules for Creating Extension Methods
- Anonymous Methods & Lambda Expressions

Delegate & Events

- Usage of Delegates
- Multicast Delegate
- Usage of EventHandler Delegate & EventArgs class
- Create Custom Delgates and Events
- Func<> Delegate, Action<> Delegate and Predicate<> Delegate examples

File Handling

- Working with Streams

Serialization

- Serializing and Deserializing an Object
- [Serializable], [Serialized] and [NonSerialized] attributes
- Usage of IDeserializationCallBack Interface

Attributes & Reflection

- Creating custom attributes
- AttributeUsage Attribute example
- Assembly Class
- Invoking Normal, static and private methods from an Assembly
- Usage of Binding Flags
- Accessing Instance Variables, Properties and Resource files from Assembly
- Creating PublickeyToken
- Creating and Assigning StrongName to Assembly (SN.exe)
- MSIL Disassembler (ILDASM.exe) & Assembler(ILASM.exe)
- Installing and uninstalling Assemblies in GAC using GACUTIL.exe
- Private Assembly, Shared Assemblies & Satellite Assemblies

Winforms

- Working with Form object & Form Properties(FormBorderStyle, Dock etc)
- Working with Windows controls (Button, Label, ToolTip, Checkbox, RadioButton, Menu, ErrorProvider ,pictureBox, ContextMenu etc)
- SDI and MDI Application (MDI Container)
- Working with Dialogs (OpenFileDialog, Color Dialog etc)
- Creating Custom Dialog
- Form Events
- Usage of Application class (Application.Run() & Application.Exit()) and ApplicationContext

WPF

- Usage of XAML
- WPF Property System w.r.t Dependency Property
- Milcore.dll (Media Integration Layer)
- Working with Layouts & Layout Properties (Grid Panel & Grid Splitter, Stack Panel , Dock Panel, Wrap Panel ,Canvas Panel)
- Event Handling in WPF(Routed events : Direct event, Bubbling Event & Tunneling event)
- Event Pairing (Bubbling Event & Tunneling event)
- RoutingStrategy
- Markup extensions {Binding }
- DataBinding (OneWay, Two Way, OneTime, OneWaytoSource)
- Usage of INotifyProperty Changed
- Usage of DataContext Property
- StaticResource and DynamicResource
- Using Styles in WPF

ADO.NET

Concepts

- Connected VS Disconnected Architecture
- Connection Pooling
- Connection String using Config file
- ADO.NET Data Providers
- Specifying the primary key of data table
- Creating a relation object
- Fetching the child rows on the basis of the selected parent row
- Search Results
- Microsoft Distributed Transaction Coordinator (MSDTC)
- AcceptRejectRule, DeleteRule, UpdateRule
- RowState & RowVersion
- Usage of DbProviderFactories
- Best way for Closing Connection Object
- Generating Commands using CommandBuilders for DataAdapters
- Usage of CommandBehavior Enum in Command's ExecuteReader()
- Usage of SQL Parameters

Connection Object

- BeginTransaction()
- Open()
- Close()
- ConnectionString Property
- ConnectionTimeout Property

Command Object

- CommandText Property
- CommandType Property
- Connection Property
- Transaction Property
- ExecuteScalar()
- ExecuteReader()
- ExecuteNonQuery()

DataReader Object

- FieldCount Property
- HasRows Property
- Close ()
- Dispose()
- IsDBNull()
- NextResult()
- Read()

DataAdapter object

- Fill()
- Update()
- DeleteCommand Property
- InsertCommand Property
- UpdateCommand Property
- SelectCommand Property

DataSet

- AcceptChanges()
- BeginInit()
- Clear()
- Clone()
- Copy()
- GetXml()
- GetXmlSchema()
- HasChanges()
- ReadXML()
- WriteXML()
- WriteXMLSchema()
- ReadXMLSchema()
- Tables Property

DataTable

- LoadDataRow()
- NewRow()
- Select()
- Columns Property
- Constraints Property
- DataSet Property

- DefaultView Property
- PrimaryKey Property
- Rows Property

DataRow

- GetChildRows Method

DataRelation

- ChildColumns Property
- ChildKeyConstraint Property
- ChildTable Property
- DataSet Property
- ParentColumns Property
- ParentKeyConstraint Property
- ParentTable Property
- RelationName Property

DataView

- ToTable()
- RowFilter Property
- RowStateFilter Property
- Sort Property

SqlException

- Class Property
- Errors Property
- LineNumber Property
- Number Property
- Procedure Property
- Server Property
- Source Property
- State Property

SqlTransaction

- Commit()
- Rollback()
- Save()
- Connection Property

ASP.NET

Basics

- Working with IIS
- Creating Virtual Directories and executing ASP.NET Programs
- ASP.NET Page Life cycle
- Page Class
- Page Class Methods and Order of Occurrence
- Usage of Server Tag (<% %>)
- Page Directive
- Import Directive
- Usage of Request.QuerString[] ,Request.Forms[], Request.Params[]
- Server side Controls
- Working with Server side Controls Events
- Page.IsPostBack
- AutoPostBack

Standard Controls

- Label
- Literal
- TextBox
- CheckBox
- Button
- LinkButton
- ImageButton
- Image
- ImageMap
- Panel
- Hyperlink
- PlaceHolder

Validation Controls

- Required Field Validator (Initial Value properties)
- Range Validator (MaximumValue, MinimumValue, Type)
- Compare Validator(Operator, ControlToCompare, ValueToCompare, Operator)
- Regular Expression Validator (ValidationExpression)
- Custom Validator(ClientValidationFunction, OnServerValidate)
- Validation Summary (ShowSummary, ShowMessageBox, DisplayMode, HeaderText)

- Common Properties (ControlToValidate, ErrorMessage, ValidationGroup, Display)
- Usage of CauseValidation Property for Button

MasterPages & Themes

- Creating Master Pages
- Accessing Controls from Master Pages in Content Page using Page.Master property
- Creating and Applying Themes
- Creating Skin File
- Apply theme to a page using Page Directive, Pre init method and web.config

Rich & Navigation Controls (go through important properties and how to use those controls)

- FileUpload
- Calendar
- AdRotator
- Wizard
- MultiView & View
- SiteMap Path
- Menu
- TreeView
- SiteMapDataSource

DataControls

- GridView
- DataList
- DetailsView
- FormView
- Repeator
- SqlDataSource
- ObjectDataSource
- Working with Template Columns
- Accessing Primary key value from GridView
    - GridView1.DataKeys[e.RowIndex].Value.ToString();
- Accessing Non template column value (2nd column) from GridView
    - (GridView1.Rows[e.RowIndex].Cells[1].Controls[0] as TextBox).Text;
- Accessing template column value from GridView
    - (GridView1.Rows[e.RowIndex]. FindControl ('txtFind') as TextBox).Text;
- Usage of FindControl()

State Management & Security

- Working with Cookies
    a. Expiring a Cookie
    b. Retrieve value from Cookie
    c. Adding Cookie
    d. Limitations of Cookie
- Working with Session
    a. SessionId
    b. Creating and Access Value from Session Variable & its scope
    c. Limitations of Session
    d. Session Methods
- Working with View State
    a. Limitations of ViewState
- Working with Application
    a. Usage of Application Variables (Page Hit counts)
- Usage of Global.ASAX (Application_Start, Application_End, Session_Start, Session_End, Application_Error(Server.GetLastError))
- Authentication & Authorization
- Membership Providers
- LoginControls

ASP.NET AJAX

- Updatepanel
    - PostbackTrigger
    - AsyncPostbackTrigger
- UpdateProgress
- ScriptManager
- ScriptReference
- XMLHTTPRequest Object

NUNIT, NCover, FxCop & Stylecop

- TestFixture
- TestFixtureSetUp
- SetUp
- TearDown
- TestFixtureTearDown
- Ignore
- Expected Exception
- Rules for creating Test Method
- NCover Usage
- FxCop Error Levels
- StyleCop Guidelines

- Usage of Log4net

SQL SERVER

Normalization

DDL statements

DML Statements

Aggregate Functions

JOINS

Subqueries

Important Queries

-- To Show all the databases

EXECUTE SP_Databases;

SELECT NAME FROM sys.sysdatabases

-- To Create a DATABASE

CREATE DATABASE training

-- To select a DATABASE:

USE training

--To see the tables in the current DATABASE

SELECT NAME FROM sys.tables

-- Create a TABLE

```
CREATE TABLE Employee(EmployeeID INT,EmployeeName VARCHAR(20),Age INT)

-- To see the structure of the TABLE

SP_HELP Employee

-- To delete a TABLE

drop TABLE Employee

-- To CREATE a TABLE with primary key and non-null value fields

CREATE TABLE Employee(EmployeeID INT PRIMARY KEY,EmployeeName VARCHAR(20) NOT NULL ,Age INT NOT NULL)

-- To insert a record :

INSERT INTO Employee(EmployeeID,EmployeeName,Age) VALUES(101,'Abishek',27);

-- To retreive the records (ALL Fields)

SELECT * FROM Employee;

-- To retreive the records (Particular Fields)

SELECT EmployeeName,Age FROM Employee

-- To retreive a particular record:

SELECT * FROM Employee WHERE EmployeeID = 101

-- Use of "as" and "arithmetic operators":

SELECT EmployeeId,EmployeeName,Age,Age-1 AS 'New Age' from Employee
```

```
-- To add a new column:

ALTER TABLE Employee ADD City VARCHAR(20);



--To update a record:

UPDATE Employee SET City = 'Chennai' WHERE EmployeeId = 101;


-- Inserting Few more records

INSERT INTO Employee(EmployeeID,EmployeeName,Age,City) VALUES(102,'Ganesh',39,'Mumbai');

INSERT INTO Employee(EmployeeID,EmployeeName,Age,City) VALUES(103,'Ajit',34,'Pune');

INSERT INTO Employee(EmployeeID,EmployeeName,Age,City) VALUES(104,'Karthik',28,'Bangalore');

INSERT INTO Employee(EmployeeID,EmployeeName,Age,City) VALUES(105,'Shilpa',24,'Mumbai');



-- Use of "and" : [Logical operator]

SELECT * FROM Employee WHERE City = 'Chennai' AND Age = 20;


-- NotEqual<> : [Relational Operator]

SELECT * FROM Employee WHERE Age<>20;


-- Between Operator

SELECT * FROM Employee WHERE Age BETWEEN 18 AND 27;


-- NOT Between Usage
```

```
SELECT * FROM Employee WHERE Age NOT BETWEEN 18 AND 27;

-- IN Operator

SELECT * FROM Employee WHERE City IN('Chennai','Mumbai')

-- NOT IN Operator

SELECT * FROM Employee WHERE City NOT IN('Bangalore','Pune')

-- Like operator :

SELECT * FROM Employee WHERE  EmployeeName LIKE 'A%'

-- Sorting in Ascending Order (default)

SELECT * FROM Employee ORDER BY Age ASC

-- Sorting in Descending Order

SELECT * FROM Employee ORDER BY Age DESC

-- To Eliminate Duplicate record

SELECT DISTINCT City FROM Employee

-- Aggregate Functions [max,min,sum,avg..]

SELECT MAX(age) FROM Employee

-- To count number of Records

SELECT COUNT(*) FROM Employee
```

```
-- Group BY

SELECT City, COUNT(*) AS 'No Of Employees' FROM Employee GROUP BY City


-- Having

SELECT City, MAX(Age) AS 'Maximum Age' FROM Employee GROUP BY City HAVING MAX(Age) > 30



-- SubQuery

SELECT MAX(age) FROM  Employee WHERE age < ( SELECT MAX(age) FROM Employee);


-- To Delete a record

DELETE FROM Employee WHERE EmployeeID = 105;


-- To delete contents of the table

TRUNCATE Employee;


-- To delete a Table

DROP TABLE Employee;


-- Creating a Table with auto_increment primary key value

CREATE TABLE Employee(EmployeeID INT IDENTITY(1001,1) PRIMARY KEY,EmployeeName VARCHAR(20) NOT NULL ,Age INT  NOT NULL)

INSERT INTO Employee(EmployeeName,Age) VALUES ('Ajit',34)


-- Create UserDefined Data Type
```

```
CREATE TYPE UDT_CITY FROM VARCHAR(25) NOT NULL

-- Making use of UserDefined Data Type
ALTER TABLE Employee ADD City UDT_CITY;

-- Deleting Table
DROP TABLE Employee;

-- Dropping UserDefined Data Type
DROP TYPE UDT_CITY

-- Entity Integrity (each row is unique)
CREATE TABLE Employee
(
        EmployeeID CHAR(9) CONSTRAINT pkEmpID PRIMARY KEY,
        EmailID VARCHAR(25) CONSTRAINT uqEmail UNIQUE
)

-- Domain Integrity (Set of Values)
ALTER TABLE Employee ADD CONSTRAINT ckEmpID CHECK(EmployeeID LIKE 'IGATE[0-9][0-9][0-9][0-9]'),
Gender CHAR(1) CONSTRAINT ckGender CHECK(Gender in ('M','F','T')),
City VARCHAR(25) CONSTRAINT dftCity DEFAULT 'Bangalore',
BasicPay INT CONSTRAINT ckBasicPay CHECK(BasicPay BETWEEN 2000 AND 5000)

SP_HELP Employee
```

```
INSERT INTO Employee(EmployeeID,EmailID,Gender,BasicPay)
VALUES('IGATE1234','karthik@igate.com','M',2500)

SELECT * FROM Employee


--Disabling Constraints

ALTER TABLE Employee NOCHECK CONSTRAINT ckBasicPay

INSERT INTO Employee(EmployeeID,EmailID,Gender,BasicPay)
VALUES('IGATE4321','sample@igate.com','M',1500)


--Enabling Constraints

ALTER TABLE Employee CHECK CONSTRAINT ckBasicPay

INSERT INTO Employee(EmployeeID,EmailID,Gender,BasicPay)
VALUES('IGATE4344','demo@igate.com','M',1500)



--Dropping Constraint

ALTER TABLE Employee DROP CONSTRAINT ckBasicPay


-- WITH CHECK(Adding Constrainst by checking against existing DATA)

ALTER TABLE Employee WITH CHECK ADD CONSTRAINT ckBasicPay CHECK(BasicPay BETWEEN 2000 AND
5000)


-- WITH NOCHECK(Adding Constrainst without checking against existing DATA)

ALTER TABLE Employee WITH NOCHECK ADD CONSTRAINT ckBasicPay CHECK(BasicPay BETWEEN 2000
AND 5000)


-- Creating Department Table
```

```
CREATE TABLE Department
(
        DeptID INT CONSTRAINT pkDeptID PRIMARY KEY,
   DeptName VARCHAR(15)
)

-- Referential Integrity (Enforce relationship between tables)
ALTER TABLE Employee ADD DeptID INT CONSTRAINT fkDeptID FOREIGN KEY REFERENCES
Department(DeptID)

-- Creating a Rule
CREATE RULE genderRule AS @gender IN ('M','F','T')

-- Binding a Rule
SP_BINDRULE genderRule,'Employee.Gender'

-- Unbinding a Rule
SP_UNBINDRULE 'Employee.Gender'

-- Dropping a Rule
DROP RULE genderRule

-- Unique Identifier
CREATE TABLE SampleTable(ID UNIQUEIDENTIFIER NOT NULL)
INSERT INTO SampleTable values(NEWID());
SELECT ID FROM SampleTable;
```

```
-- ALL Operator

CREATE TABLE T1(ID INT NOT NULL)

INSERT INTO T1 VALUES(1);

INSERT INTO T1 VALUES(3);

INSERT INTO T1 VALUES(5);

INSERT INTO T1 VALUES(7);

INSERT INTO T1 VALUES(9);


CREATE TABLE T2(ID INT NOT NULL)

INSERT INTO T2 VALUES(2);

INSERT INTO T2 VALUES(4);

INSERT INTO T2 VALUES(6);

INSERT INTO T2 VALUES(8);

INSERT INTO T2 VALUES(10);


-- ALL Operator

SELECT ID FROM T1 WHERE ID < ALL(SELECT ID FROM T2);

SELECT ID FROM T2 WHERE ID > ALL(SELECT ID FROM T1);


-- ANY Operator

SELECT ID FROM T1 WHERE ID < ANY(SELECT ID FROM T2);

SELECT ID FROM T2 WHERE ID > ANY(SELECT ID FROM T1 WHERE ID >=5);


INSERT INTO T1 VALUES(2);
```

```
INSERT INTO T2 VALUES(5);

INSERT INTO T1 VALUES(6);


-- SET Operations


-- UNION (Without Duplicate Values)

SELECT ID AS 'Union' FROM T1 UNION SELECT ID FROM T2


-- UNION ALL (With Duplicate Values)

SELECT ID AS 'Union ALL' FROM T1 UNION ALL SELECT ID FROM T2


-- INTERSECT (Common Values in both Tables)

SELECT ID AS 'Intersect' FROM T1 INTERSECT SELECT ID FROM T2


-- EXCEPT

SELECT ID AS 'EXCEPT' FROM T1 EXCEPT SELECT ID FROM T2


-- Subqueries

--Single Row SubQuery(Subquery Returns 1 value)

SELECT MAX(AGE) FROM Employee WHERE AGE < (SELECT MAX(AGE) FROM Employee WHERE AGE <
(SELECT MAX(AGE) FROM Employee))

)


--Multiple Row SubQuery(Subquery Returns more than one Row)

SELECT AGE FROM Employee WHERE AGE > ALL(SELECT Age FROM Employee WHERE AGE BETWEEN 24
AND 34)
```

```
-- Correlated SubQuery(subquery depends on the outer query for its values)

SELECT AGE FROM EMPLOYEE WHERE AGE < (SELECT MAX(AGE) FROM Employee)


-- Modify a Column

ALTER TABLE EMPLOYEE MODIFY COLUMN CITY VARCHAR(250)


-- DROP a Column

ALTER TABLE EMPLOYEE DROP COLUMN AGE


-- COMPUTE

SELECT EmployeeID,EmployeeName,Age,City FROM Employee COMPUTE AVG(AGE)


-- COMPUTE BY

SELECT EmployeeID,EmployeeName,Age,City FROM Employee ORDER BY CITY COMPUTE AVG(AGE) BY
CITY



DROP TABLE EMPLOYEE

DROP TABLE DEPARTMENT


--JOINS

CREATE TABLE Department(DeptID INT,DeptName VARCHAR(25))

INSERT INTO Department(DeptID,DeptName) VALUES (1,'Computer')

INSERT INTO Department(DeptID,DeptName) VALUES (2,'Accounts')

INSERT INTO Department(DeptID,DeptName) VALUES (3,'Maths')
```

```
INSERT INTO Department(DeptID,DeptName) VALUES (4,'Arts')

INSERT INTO Department(DeptID,DeptName) VALUES (5,'Sports')

INSERT INTO Department(DeptID,DeptName) VALUES (6,'NCC')



CREATE TABLE STUDENT(ID INT PRIMARY KEY,StudentName VARCHAR(25),DeptID INT)

INSERT INTO STUDENT(ID,StudentName,DeptID) VALUES (101,'Abishek',5)

INSERT INTO STUDENT(ID,StudentName,DeptID) VALUES (102,'Ganesh',4)

INSERT INTO STUDENT(ID,StudentName,DeptID) VALUES (103,'Shilpa',2)

INSERT INTO STUDENT(ID,StudentName,DeptID) VALUES (104,'Ajit',3)

INSERT INTO STUDENT(ID,StudentName,DeptID) VALUES (105,'Selva',2)

INSERT INTO STUDENT(ID,StudentName,DeptID) VALUES (106,'Karthik',1);

INSERT INTO STUDENT(ID,StudentName,DeptID) VALUES (107,'Mohan',8);


--INNER JOIN(EQUI Join)

SELECT ID,StudentName,DeptName FROM STUDENT INNER JOIN Department ON STUDENT.DeptID =
Department.DeptID


--INNER JOIN(NON EQUI Join)

SELECT ID,StudentName,DeptName FROM STUDENT INNER JOIN Department ON STUDENT.DeptID <>
Department.DeptID where ID =101


-- LEFT OUTER JOIN

SELECT ID,StudentName,DeptName FROM STUDENT LEFT OUTER JOIN Department ON STUDENT.DeptID
= Department.DeptID


-- RIGHT OUTER JOIN
```

```
SELECT ID,StudentName,DeptName FROM STUDENT RIGHT OUTER JOIN Department ON
STUDENT.DeptID = Department.DeptID


-- FULL OUTER JOIN

SELECT ID,StudentName,DeptName FROM STUDENT FULL OUTER JOIN Department ON STUDENT.DeptID
= Department.DeptID


--CROSS JOIN

SELECT Student.ID,Department.DeptID FROM Student CROSS JOIN Department


-- SELF JOIN

CREATE Table Employee(EmployeeID INT, EmployeeName VARCHAR(20),ManagerID INT)

INSERT INTO EMPLOYEE VALUES(101,'KARTHIK',102)

INSERT INTO EMPLOYEE VALUES(102,'LATHA',103)

INSERT INTO EMPLOYEE VALUES(103,'VEENA',103)

INSERT INTO EMPLOYEE VALUES(104,'ABISHEK',102)


SELECT T1.EmployeeName,T2.EmployeeName AS Manager FROM Employee T1 INNER JOIN Employee T2
ON T2.EmployeeID = T1.ManagerID


-- Index

-- Creating Default Clustered Index and Non Clustered Index using Primary Key and Unique Key

CREATE TABLE Person(PersonID INT CONSTRAINT pkPersonID PRIMARY KEY, PersonName
VARCHAR(20),EmailID VARCHAR(50),PanNumber CHAR(10) CONSTRAINT uqPanNumber
UNIQUE,PassportNumber VARCHAR(20));

SP_HELP Person
```

```
-- Creating Clustered Index(A table can have only 1 Clustered Index)

CREATE CLUSTERED INDEX pk1PersonID ON Person(PersonID)


-- Creating NONCLUSTERED Index on SecondaryKey

CREATE NONCLUSTERED INDEX nciEmailID ON Person(EmailID)

CREATE NONCLUSTERED INDEX nci1EmailID ON Person(EmailID)


-- information about Index on Tables

SP_HELPINDEX  Person


-- Droping NONCLUSTERED Index

DROP INDEX Person.nci1EmailID


-- Creating Composite Indexes

CREATE NONCLUSTERED INDEX nciEmailPass ON Person(EmailID,PassportNumber)

DROP INDEX Person.nciEmailPass


DROP TABLE Person


--Views

CREATE TABLE Category(CategoryID INT PRIMARY KEY,CategoryName VARCHAR(25))

INSERT INTO Category(CategoryID,CategoryName) VALUES (1,'Personal Computers')

INSERT INTO Category(CategoryID,CategoryName) VALUES (2,'Laptop')

INSERT INTO Category(CategoryID,CategoryName) VALUES (3,'Printers')
```

```
CREATE TABLE Product(ProductID INT PRIMARY KEY,ProductName VARCHAR(25),Price INT, CategoryID INT)

INSERT INTO Product(ProductID,ProductName,Price,CategoryID) VALUES (101,'PC-1223',30000,1)

INSERT INTO Product(ProductID,ProductName,Price,CategoryID) VALUES (102,'LP-1223',25000,2)

INSERT INTO Product(ProductID,ProductName,Price,CategoryID) VALUES (103,'PRI-1223',3000,3)

INSERT INTO Product(ProductID,ProductName,Price,CategoryID) VALUES (104,'PC-1263',28000,1)

INSERT INTO Product(ProductID,ProductName,Price,CategoryID) VALUES (105,'LP-1264',34000,2)

INSERT INTO Product(ProductID,ProductName,Price,CategoryID) VALUES (106,'PRI-1623',4000,3)


-- CREATING VIEW

CREATE VIEW ProductCategoryView  AS

SELECT ProductID, ProductName, Price, CategoryName

FROM  Product INNER JOIN

Category ON Product.CategoryID = Category.CategoryID


-- Working With View (SELECT)

SELECT ProductName,Price,CategoryName FROM ProductCategoryView


-- Working With View (Update)

UPDATE ProductCategoryView SET Price = Price + 100


-- Working With View (INSERT)

INSERT INTO ProductCategoryView(ProductID, ProductName, Price, CategoryName) VALUES (107,'PC-4556',3200,'Personal Computers')

INSERT INTO ProductCategoryView(ProductID, ProductName, Price) VALUES (107,'PC-4556',3200)
```

```
SELECT * FROM Product


-- Altering View

ALTER VIEW ProductCategoryView  AS

SELECT ProductID, ProductName, Price, CategoryName

FROM  Product LEFT OUTER JOIN

Category ON Product.CategoryID = Category.CategoryID


SELECT ProductName,Price,CategoryName FROM ProductCategoryView


-- Dropping View

DROP VIEW ProductCategoryView


SELECT TOP(2) ProductID FROM PRODUCT


DATE Formats

--------------

SELECT CONVERT(VARCHAR(20), GETDATE(), 100)

SELECT CONVERT(VARCHAR(8), GETDATE(), 1) AS [MM/DD/YY]

SELECT CONVERT(VARCHAR(10), GETDATE(), 101) AS [MM/DD/YYYY]

SELECT CONVERT(VARCHAR(8), GETDATE(), 2) AS [YY.MM.DD]

SELECT CONVERT(VARCHAR(8), GETDATE(), 3) AS [DD/MM/YY]

SELECT CONVERT(VARCHAR(10), GETDATE(), 103) AS [DD/MM/YYYY]

SELECT CONVERT(VARCHAR(8), GETDATE(), 4) AS [DD.MM.YY]
```

```
SELECT CONVERT(VARCHAR(10), GETDATE(), 104) AS [DD.MM.YYYY]

SELECT CONVERT(VARCHAR(8), GETDATE(), 5) AS [DD-MM-YY]

SELECT CONVERT(VARCHAR(10), GETDATE(), 105) AS [DD-MM-YYYY]

SELECT CONVERT(VARCHAR(9), GETDATE(), 6) AS [DD MON YY]

SELECT CONVERT(VARCHAR(11), GETDATE(), 106) AS [DD MON YYYY]

SELECT CONVERT(VARCHAR(10), GETDATE(), 7) AS [Mon DD, YY]

SELECT CONVERT(VARCHAR(12), GETDATE(), 107) AS [Mon DD, YYYY]

SELECT CONVERT(VARCHAR(8), GETDATE(), 108)

SELECT CONVERT(VARCHAR(26), GETDATE(), 109)

SELECT CONVERT(VARCHAR(8), GETDATE(), 10) AS [MM-DD-YY]

SELECT CONVERT(VARCHAR(10), GETDATE(), 110) AS [MM-DD-YYYY]

SELECT CONVERT(VARCHAR(8), GETDATE(), 11) AS [YY/MM/DD]

SELECT CONVERT(VARCHAR(10), GETDATE(), 111) AS [YYYY/MM/DD]

SELECT CONVERT(VARCHAR(6), GETDATE(), 12) AS [YYMMDD]

SELECT CONVERT(VARCHAR(8), GETDATE(), 112) AS [YYYYMMDD]

SELECT CONVERT(VARCHAR(24), GETDATE(), 113)

SELECT CONVERT(VARCHAR(12), GETDATE(), 114) AS [HH:MI:SS:MMM(24H)]

SELECT CONVERT(VARCHAR(19), GETDATE(), 120)

SELECT CONVERT(VARCHAR(23), GETDATE(), 121)

SELECT CONVERT(VARCHAR(23), GETDATE(), 126)

SELECT CONVERT(VARCHAR(26), GETDATE(), 130)

SELECT CONVERT(VARCHAR(25), GETDATE(), 131)
```