WEB APPLICATIONS

Learning Objectives

After studying this lesson students will be able to:

- Define the term hypertext and state the purpose of HTML.
- Identify the main parts of an HTML document.
- Identify and state the purpose of different HTML elements.
- Differentiate between HTML elements, tags, and attributes.
- Create web pages using different basic and formatting tags.
- Differentiate between HTML and XML.

The World Wide Web (or simply the Web or WWW) is a system of sharing interlinked hypertext documents over the internet. These documents are stored on web-servers on the internet and contain text, images, videos and other multimedia. These documents also contain hyperlinks to navigate among them. HTML (Hyper Text Markup Language) is the basic language which is used to create Hypertext documents. In this lesson we are going to learn how to create hyper text documents using HTML.

Puzzle [7]

You have 5 jars of pills. Each pill weighs 10 gram, except for contaminated pills contained in one jar, where each pill weighs 9 gm. Given a scale, how could you tell which jar has the contaminated pills in just one measurement?

Well as we can infer from the puzzle given above that assimilation of information and then using the information to solve a problem are of utmost importance. Similarly, information retrieval from the World Wide Web is also of prime importance and this information retrieval is based on protocols.

Introduction

Internet is such a wonder box that whenever we need some information on any topic, we refer to internet. Information is available on almost all the topics - from Ayurveda to Advaced Medical Surgeries, from basic colors to advanced designs, from fundamentals of computers to latest developments in the field of supercomputers - you name it and internet has it. This information is actually stored on some computers on the net. These computers are called the servers. The information is stored in the form of some documents called hyper text documents. All the related documents on a server are linked together using hyperlinks. Therefore using hyperlinks we can move from one document to another. This is formally called navigation. There are a number of ways to create hyper text documents. There are many specialized software packages like Dreamweaver, CoffeCup etc. which are used to create web documents. The simplest way to create a web document is to use a text editor like notepad, notepad2 etc. and write code in HTML. A hyper text document on the web is also called a web page.

The information over the web is shared using a protocol called Hyper Text Transfer Protocol (HTTP)

History of World Wide Web

The concept of WWW was designed in 1989 by Tim Berners-Lee and scientists at CERN (Geneva), the European centre for High Energy Physics. Their purpose was to make sharing and retrieval of research material simpler. A year later they had developed a 'browser/editor' program and had named the program World Wide Web. The World Wide Web grew rapidly and attained its present form. Its further development is guided by the WWW Consortium (W3C) based at the Massachusetts Institute of Technology in Cambridge, Massachusetts

Fig.7.1 Tim Berners-Lee

Uniform Resource Locator

The uniform resource locator (URL) is the unique identifier of a web page. The address or URL of the current page you are on appears in the "Address Bar" of the web browser. You can go directly to a web page if you know its URL by simply typing the URL in the address bar. You can click in the address bar at any time and overwrite the current address with another URL to jump to a different web page.

The most general form of a URL syntax is as follows:

Protocol://domain name/<directory path>/<object name>

For example:
`http://www.openoffice.org/dev_docs/features/3.2/rc2.html`

The elements of this syntax are as follows:

Part	Description	Example
Protocol	represents name of the protocol which is used to transfer the data/web page	http, ftp
Domain Name	represents name of the web server where the web page resides	www.openoffice.org
Directory Path	Represents location of the web page on the web server	dev docs/features/3.2
Object Name	name of file	rc2.html

When the URL of a web page is given to the web browser, the browser sends a request for this page to the relevant web server. The web server, upon getting this request, sends the requested web-page to the browser and then the browser displays this page. A web browser knows how to display web pages.

Know More!

Semantic Web

The concept of Semantic Web occurred to Tim Berners-Lee, inventor of the WWW. The Semantic Web provides a common format that allows data to be shared and reused from diverse sources. Semantic Web is an effort to make reuse and republishing of data easier. Semantic web is based on concept based navigation of the web as opposed to fixed term navigation undertaken earlier.

Semantic Web provides technologies that can merge information from two sources. The component that helps in this is the Resource Description Framework (RDF). When information from two sources in RDF needs to be merged, we can concatenate the files into one big file - joining on those terms which are defined to correspond to the indicators in both the files.

More information on this is available on http://www.w3.org/RDF

Let us now start on our web page designing journey. We will use the tools that are already available on our system.

Following will be our first web page:

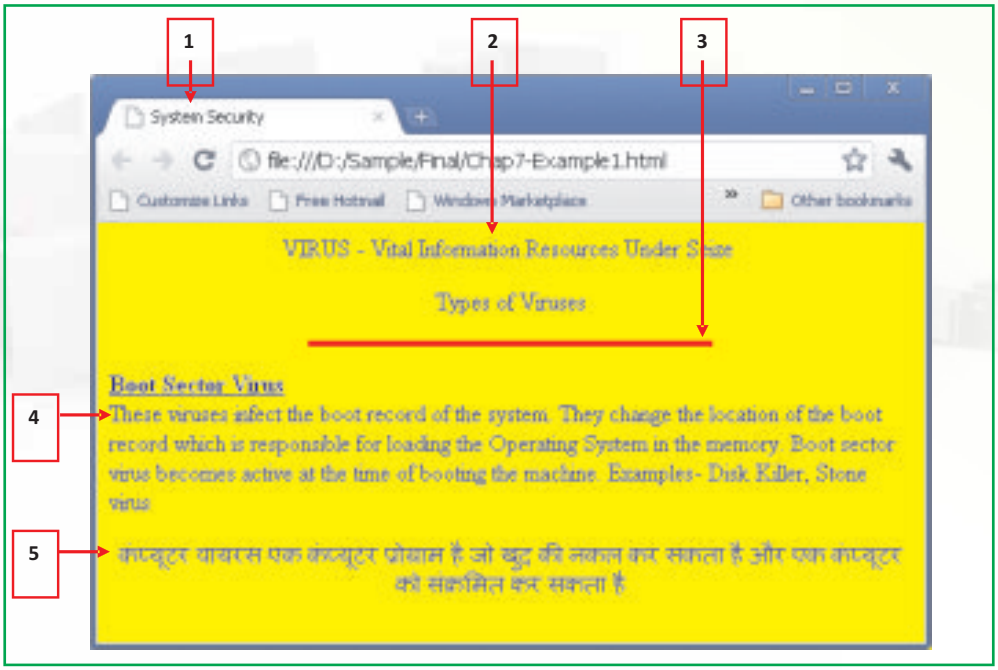


Figure 7.2: Web Page developed using HTML Tags

The highlighted features of this web page are:

1. Title - 'System Security'
2. Center Aligned Paragraph
3. Horizontal Rule
4. Left Aligned Paragraph
5. Contents in Hindi

Now let us look at the source code of the web page

```
<HTML>
   <HEAD>
     <TITLE> System Security</TITLE>
   </HEAD>
 <!--Beginning of Body tag-->
   <BODY bgcolor = "yellow" TEXT="blue"
       LINK="CYAN" ALINK="BLACK" VLINK="TEAL">
     <CENTER> VIRUS - Vital Information Resources Under Seize <BR>
     <P>Types of Viruses</P>
     <HR SIZE= 4 WIDTH= 50% COLOR="RED"> <CENTER>
     <P ALIGN="LEFT"><u><b>Boot Sector Virus</b></u> <BR>
     These viruses infect the boot  record of the system. They
 change the location of the boot record which is responsible
     for loading the Operating  System in the memory. Boot sector
virus becomes active at the time of booting the  machine.
Examples- Disk Killer, Stone virus  </P>
     <p lang="hi"> कंप्यूटर वायरस एक कंप्यूटर प्रोग्राम है जो खुद की नकल कर सकता है
और एक कंप्यूटर को संक्रमित कर सकता है </P>
   </BODY>
</HTML>
```

Figure 7.3: Code for Figure 7.2

The code is written using HTML.

HTML

HTML stands for Hyper Text Markup Language. It is a markup language used to create HTML documents. An HTML document defines a web page. For example, consider Figures 7.3 and 7.2 shown above. Figure 7.3 shows a text file which is actually an HTML document. Figure 7.2 shows the corresponding web page created by the browser (Google

Chrome in this example) using the HTML document given in Figure 7.3. Simply put, Figure 7.2 shows a web page defined by the HTML document shown in Figure 7.3. To define web pages HTML provides various markup elements. Using these elements we can specify various parts of a web page and also formatting of the contents of the web page. For example, some of the HTML elements used in Figure 7.3 are <HTML> . . . </HTML>, <P> . . . </P>,
 etc. HTML elements are used to specify headings, paragraphs, lists, tables, images and much more. When an HTML document is opened in a web browser, the browser interprets these elements and displays the HTML document as a web-page. Here we should understand the difference and relationship between an HTML document and a web page. An HTML document is a text file that contains HTML elements. An HTML document shown by a web browser is called a web page.

HTML is a subset of Standard Generalized Markup Language (SGML) and is specified by the World Wide Web Consortium (W3C).

Parts of an HTML document :

Any HTML document, in general, contains at least three elements - HTML, HEAD, and BODY. These elements are specified by the following respective tags:

1. <HTML> . . .</HTML>
2. <HEAD> . . .</HEAD>
3. <BODY> . . .</BODY>

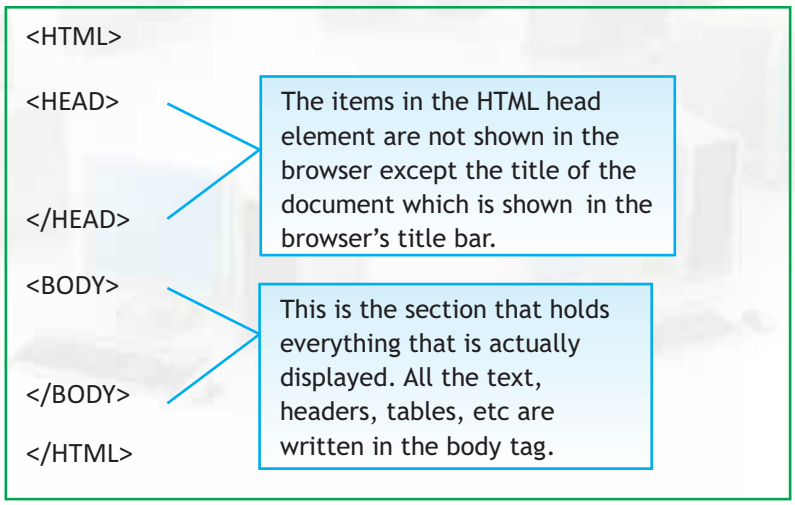


Figure 7.4: Parts of an HTML document

Each document consists of head and body text. The head contains the title and the body contains the actual text that is made up of paragraphs, lists, and other elements. Let us first learn to implement these examples on the system. We can do this easily by following the steps given below:

- Type the code in any text editor, e.g. gedit, Notepad2, Notepad etc.
- Save the file in a specified directory with an extension as .htm or .html
- Open the file in a web browser.

The corresponding web page is displayed in the browser window.

HTML: Elements, tags, and attributes

Any HTML document contains:

(a) information that is to be displayed by the web-browser.

(b) formatting information that tells the web-browser about lay-out of this information when it is displyed.

In an HTML document, the formatting information is given in the form of HTML elements. An HTML element is specified by the corresponding tags. For example BODY element is specified by the pair of tags <BODY> and </BODY>, B (Bold) element is specified by the pair of tags <B> and </B>. A tag may also contain attribute(s). An attribute defines a property of an element and is specified in the format **attribute = "value"** in the start tag of the element.

An example: BODY element is specified by the pair of tags <BODY> and </BODY>. Here <BODY> is the start tag and </BODY> is the end tag. By default, the background colour of a web page is white, but we can change it to yellow (or to any other colour) by specifying the BGCOLOR attribute in the <BODY> tag as <BODY BGCOLOR = "YELLOW">. The complete thing starting from the <BODY> (with or without attributes) tag till the </BODY> tag including the contents between these two tags is the BODY element.

HTML elements are of two types- Container elements and Empty elements. These are discussed below.

Container Elements :

A container element is specified by a pair of tags - Start tag and End tag. These tags also called ON tags and OFF tags. Start tag consists of the tag name enclosed in left and right angular brackets. The end tag is identical to the start tag, except for a slash (/) that precedes the text within angular brackets of the end tag. e.g.,

```
<BODY> . . . </Body>
```

Container elements contain parameters and the parameters of an element are given between the start tag and end tags.

```
<BODY> . . . </BODY>
```

↓

Parameters

Elements in HTML may also contain attributes that can be given along with the tag name in the angular brackets of the start tag. E.g.,

```
<BODY BGCOLOR="BLUE" TEXT="RED">
```

Attributes

Empty Elements

Empty elements have only a start tag and no end tag. Hence, an empty element has no parameters, but can take attributes, which are given within angular brackets of the start tag.

Example
 tag

> HTML is not case sensitive, i.e., tags and attributes can be written in small or capital alphabets. They can also be written using a mix of capital and small alphabets.

Let us now discuss HTML elements in detail:

HTML element

It is a container element started by <HTML> tag and ended by </HTML> tag. It identifies the document as an HTML document. It does not have any effect on the appearance of the document, but tells the browser that the current document is an HTML document.

Syntax: **<HTML> . . .</HTML>**

HTML element further contains HEAD and the BODY elements, which can further contain a number of other elements.

HEAD element - It is a container element started by <HEAD> tag and ended by </HEAD> tag. It defines the HTML document header and does not affect the appearance of the document in the browser window. The header contains information about the document.

Syntax: **<HEAD> . . .</HEAD>**

TITLE element - It is a container element started by <TITLE> tag and ended by </TITLE> tag. Every HTML document should contain the title to be displayed in the title bar of the browser window. If an HTML document does not contain a title, then the file name of the HTML document is displayed in the title bar. We can see in Figure 7.3 System Security is written in the TITLE element and thus it is displayed in the browser's title window as observed in figure 7.2. The TITLE element is placed in HEAD element.

Syntax: **<TITLE>. . .</TITLE>**

BODY element - It is a container element started by <BODY> tag and ended by </BODY> tag. It contains the main contents of the document as parameter.

Syntax: **<BODY> . . .</BODY>**

BODY tag contains many attributes. Some of the important attributes are discussed below.

1.	**Attribute:**	Background
	Use:	It is used to specify the path and filename of an image that has to be used as the background of the document. If the referenced image is smaller than the browser window, it will be tiled to fit and will scroll with the text on the page. The filename extension has to be specified along with the filename.
	Syntax:	`BACKGROUND= "path/filename.jpg"`
2.	**Attribute:**	BGCOLOR
	Use:	It sets the background colour of the web page. Most of the browsers recognize the popular colour names like RED, GREEN, YELLOW, GREY, AQUA ETC. If we want to specify a colour which does not have a specific name but we know its RRGGBB composition, then we can specify this RRGGBB composition in the BGCOLOR attribute to get the background color. E.g., in Figure 7.3, instead of writing BGCOLOR = "GREY", we could have written BGCOLOR = "#999999" and would have got the same effect. If a background image is also present, the BGCOLOR specified shines through regions where the background image is transparent.
	Syntax:	`BGCOLOR= "colorname"  OR BGCOLOR= "#rrggbb"`
3.	**Attribute:**	Text
	Use:	It sets the colour of the normal text in the document. Colour values can be given in the same way as that of the BGCOLOR attribute. The default is black (hexadecimal code #000000).
	Syntax:	`TEXT = "colorname"  OR TEXT = "#rrggbb"`

4.	**Attribute:**	Link
	Use:	This sets the colour of all of the non-visited links on the page. We can set this to any color of our choice. The default setting for a non-visited link is usually blue.
	Syntax:	`Link="colorname" OR LINK = "#rrggbb"`
5.	**Attribute:**	Alink
	Use:	This sets the colour of active links on the page. An active link is a link that has just been keypressed by the user. Here 'keypressed' does not mean 'clicked'. 'Keypressed' means that the mouse has been taken over the link and the left mouse button is pressed but not released. 'Clicked' means the button has been pressed and then released so that the browser opens the corresponding link. We can set active link colour to any colour. The default Alink colour is red.
	Syntax:	`Alink="colorname" OR ALINK = "#rrggbb"`
6.	**Attribute:**	Vlink
	Use:	This changes the colour of a visited link on the page. We can set this to any color. The default setting for a visited link is usually violet.
	Syntax:	`Vlink="colorname" OR VLINK = "#rrggbb"`

NOTE:

1. The BACKGROUND attribute when used, overrides the BGCOLOR attribute. Thus if an HTML document contains both of these tags, then the specified image will be displayed in the background of the page and the background color will be ignored. If however, the image is not found, the specified background colour will be used as background.

2. The World Wide Web Consortium (W3C) has listed 16 valid color names for HTML: **aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow.** Majority of browsers understand the names of many other colors also. Any color is composed of 3 basic colors: RED, GREEN, BLUE. If a color does not have a specific name (or if we don't know its name), its RGB (Red, Green, Blue) composition may be specified. This composition is specified in hexadecimal notation in the format "#rrggbb". The value of each of the composition colors is specified as a hexadecimal number and may vary from 00 to FF. For example, the color code "#00FF00" specifies pure green color and "#969696" specifies a color which contains a mix equal parts of Red, Green, and Blue. Just try and find out which color it displays.

3. Following this, whenever we say "colorname" in this chapter, it shall mean color name as well as "rrggbb" composition of the color.

To Write in Hindi (and other regional languages) on the Web Page

If lang attribute with a suitable value is added in the <HTML> tag, the whole web page is displayed in the specified language. E.g.

```
<html lang="hi">
```

Tells the web browser to display the web page in Hindi (Devnagari script).

Instead of **"hi"** we can use **"ks", "kok", "ne", "mr", "sa"** for Kashmiri, Konkani, Nepali, Marathi and Sanskrit respectively. If we want to use multiple languages on the same page, we can write this attribute with the <p> tag also.

```
<p lang="hi">
```

हम हिंदी लिख रहे हैं

We can download any of the devnagari script fonts like Susha, Mangal etc. and use them in the HTML document. These fonts use the unicode character encoding. Unicode makes the web pages automatically searchable. In case we use Unicode fonts in our document we have to select Encoding as Unicode while saving the file.

The steps to install and use Hindi font are given in the appendix at the end of this book.

Know More!

You can get more information about UTF-8 and Unicode from the net. There you can also find the information about how to use regional languages in the HTML documents.

P (Paragraph) element - It is a container element started by <P> tag and ended by </P> tag. This element is used to start a new paragraph. HTML does not recognize the return/enter key we enter in text editor; therefore we use <P> tag to start a new paragraph. <P> starts a new paragraph with extra space before the first line.

Syntax: **<P ALIGN="alignment" lang = "language"> . . .</P>**

The end tag </P> of Paragraph element is optional and may be skipped. But it is highly recommended to put this end tag. Skipping it may produce unexpected results or errors in some browsers. Future versions of HTML will not allow you to skip end tags.

ALIGN attribute specifies the horizontal alignment of paragraph. Any of the alignments - Center, Left, Right - can be specified.

LANG attribute specifies the language in which the paragraph is to be displayed as discussed above.

Horizontal Rule (HR) element - This is an empty element specified by <HR> tag. The <HR> tag draws a horizontal line across the document frame or window. We can use a horizontal line to visually divide the information into sections.

Syntax:

```
<HR ALIGN = "alignment" NOSHADE SIZE = "thickness"
                    WIDTH = "width" COLOR="colorname" >
```

The attributes used with <HR> are:

1.	**Attribute:**	ALIGN
	Use:	Specifies how the horizontal rule should be aligned. The alignment can be left, right, or center. Default is center alignment.
	Syntax:	`ALIGN = "Alignment"`
2.	**Attribute:**	NOSHADE
	Use:	produces a solid horizontal rule that has no shading.
	Syntax:	`NOSHADE`
3.	**Attribute:**	Size
	Use:	Size defines the thickness of the horizontal rule. This thickness is specified in the number of pixels. Default is 2 pixels.
	Syntax:	`SIZE = "n"` - where n is a natural number
4.	**Attribute:**	Width
	Use:	"Width" defines the horizontal width of the line. The default is the width of the page. The measurement value can be the number of pixels, e.g., "50" , or a percentage of the page width, e.g., "75%". Here 75% means that the horizontal rule will cover 75% of the page width.
	Syntax:	`WIDTH = "n/n%"` - where n is a natural number
5.	**Attribute:**	Color
	Use:	Sets the colour of the horizontal line.
	Syntax:	`COLOR= "colorname"`

Line Break (BR) Element - This is an empty element specified by
 tag.
 tag forces a line break which implies that the text/image following the tag will be moved to the next line when displayed in the browser.

Syntax: **
**

Well, now that we have designed our first html page we are familiar with the basic tags. The tags learnt so far will help us in designing very basic web pages, but if we want our web page to have different fonts, font sizes and headings we will have to delve further and learn the usage of formatting elements. Look at the web page shown below in which we have used formatting elements.

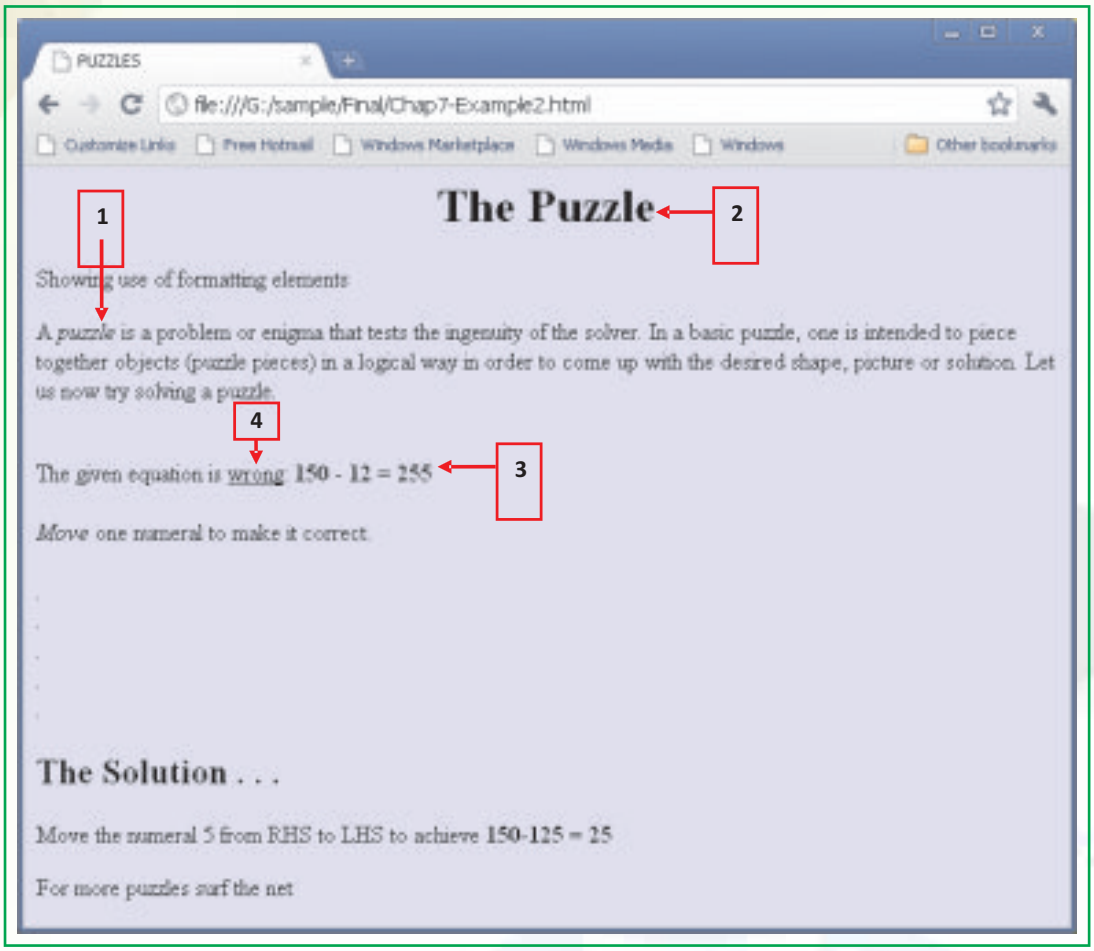


Figure 7.5 Web page created using formatting elements

The HTML code to generate this web page is given below:

```
<HTML>
   <HEAD>
     <TITLE>PUZZLES</TITLE>
   </HEAD>
   <BODY BGCOLOR="#E3E4FA">
     <H1 ALIGN="CENTER">The Puzzle</H1>
     <COMMENT>Showing use of formatting elements</COMMENT>
     <P>A <I>puzzle</I> is a problem or enigma that tests the
  ingenuity of the solver. In a basic puzzle, one is intended to
  piece together objects (puzzle pieces) in a logical way in order
  to come up with the desired shape, picture or solution. Let us
  now try solving a puzzle.<BR>
     <P>The given equation is <U>wrong</U>: <B>150 - 12 =
  255</B><BR><BR/>
     <I>Move</I> one numeral to make it correct.<BR >
     <BR>
      .
     <BR>
      .
     <BR>
      .
     <BR>
      .
     <BR>
      .
     <BR>

     <H2 >The Solution . . .</H2>
     <P>Move the numeral 5 from RHS to LHS to achieve <B>150-125 =
  25</B></P>
     <P>For more puzzles surf the net</P>
   </BODY>
  </HTML>
```

Figure 7.6 HTML code for figure 7.5

The text placed between <! and > or **<COMMENT>** and **</COMMENT>** is ignored by the browser. This text is called a comment.

Let us now understand the use of each of the elements that have been used to write this code.

Formatting Elements

Text Headings <Hn> Element - This is a container element specified by <Hn> tag, where n is a natural number from 1 to 6. This element is used to give section headings. H1 gives the most prominent heading, while H6 gives the least prominent heading. Heading element introduces a blank line above and below the header text.

Syntax: **<Hn>. . .<Hn>, where n is an integer in the range 1 to 6**

BoldFace <B> element - This is a container element started by <B> and ended by </B> tags. This element is used to display the text enclosed within the tags in bold form.

Syntax: **<B>. . .</B>**

Italics <I> element - This is a container element started by <I> and ended by </I> tags. This element is used to display the text enclosed within the tags in italics form.

Syntax : **<I>. . .</I>**

Underline <U> element - This is a container element started by <U> and ended by </U> tags. This element is used to underline the text enclosed within the tags.

Syntax: **<U>. . .</U>**

Until now, everything has been text based and we have designed some very nice text based web pages. We know, of course, that the World Wide Web (WWW) is much more than just plain text. It is the ability of the web to also provide images that makes it so popular. Images are of different types like pictures, graphics, icons, clip art, etc. Browse the web and you will see all kinds of images. Let us now look at another web page and see how we can set font, font size, font color and images in a web page. When we combine all the tags to design a web page, the web page can be made visually appealing.

Figure 7.7 Web Page with different fonts and alignment

Now we have designed a web page in which we have used different types of fonts, changed colour of the font and used images with different borders and alignment. The web page is looking nice, bright and colourful. Let us now look at the code of the above created web page and try to understand the use of the tags used to create the web page.

```
<HTML>
<HEAD>
<TITLE>Trees: A Renewable Resource</TITLE></HEAD>
<BODY>
<H1 ALIGN="CENTER">Trees A Renewable Resource</H1><BASEFONT
SIZE="4" >
<P><IMG SRC="E:\CBSE Book\HTML\tree2.jpg" BORDER=15 ALIGN="RIGHT"
WIDTH=100 HEIGHT=120>
<COMMENT>Using different fonts and font sizes</COMMENT>
<FONT SIZE="6" FACE="Georgia, Arial" COLOR="MAROON" >T</FONT>rees
are our breathing partners.We need trees in order to live. People
and animals depend on trees and plants for oxygen. As we breathe in,
our body takes in oxygen. As we breathe out, we give out carbon
dioxide. Trees do just the opposite. They take in carbon dioxide and
then release oxygen.</FONT><P>
<P><IMG SRC="E:\CBSE Book\HTML\t2.jpg" BORDER=8 ALIGN="LEFT"
WIDTH=80 HEIGHT=100>
```

```
<FONT SIZE="7" FACE="Forte, Arial" COLOR="GREEN" >T</font>rees are a
renewable natural resource.They can be renewed by replacing the
trees that have been cut  and planting more trees.We also depend on
forest products for things like the wood and paper. Actually,there
are more than 5,000 things that can be made from trees.</FONT></P>
<P><IMG SRC="E:\CBSE Book\HTML\tree2.jpg" BORDER=5 ALIGN="BOTTOM"
WIDTH=60 HEIGHT=80>
<font size="6" face="Century Gothic, Arial" color="RED"
>T</font>oday,  people are aware that our nation's forests and trees
are a valuable resource. It is in our best interest to conserve
them. Forests also provide a natural home to wildlife.Trees and
forests help us by cleaning our air, soil, and water.So let us
pledge to plant a sapling today!!
</FONT></P>
<P><B><FONT FACE="Curlz MT, Helvetica" SIZE="5" COLOR="#006600">
Plant trees to save our future generations.</FONT> </B> </P>
</BASEFONT>
</BODY>
</HTML>
```

Figure 7.8 Code for Web Page Figure 7.7

Image (IMG) element - This is an empty element specified by <IMG> tag. It is used to insert an image in a web page.

Syntax:

```
<IMG SRC="location" BORDER = "border"
    ALIGN = "alignment" HEIGHT = "height" WIDTH = "width">
```

Common attributes of IMG element are discussed below:

1.	**Attribute:**	SRC
	Use:	SRC tells the browser where the source for the image is, that is, the path and name of the image to be inserted. If no path is given, the source is assumed to be the current folder - the folder of the web page. Most browsers only support a few image formats or file types. They are GIF and JPEG.
	Syntax:	`SRC = "path/filename"`

2.	**Attribute:**	BORDER
	Use:	BORDER is used to place (or eliminate) a border of specified width around the image. Border widths are measured in pixels. The BORDER="0" means that we want no border.
	Syntax:	`BORDER = "n"  - where n is a natural number.`
3.	**Attribute:**	ALIGN
	Use:	ALIGN specifies the alignment of the image according to surrounding contents of the page. Alignment can be middle, top, bottom, left, right. Bottom is the default alignment!
	Syntax:	`ALIGN = "alignment"`
4.	**Attribute:**	WIDTH
	Use:	WIDTH specifies the horizontal width of the image. Width is specified in number of pixels or as the percentage of page width.
	Syntax:	`WIDTH = "n/n%"  - where n is a natural number`
5.	**Attribute:**	HEIGHT
	Use:	HEIGHT specifies the height of the image. Height is specified in number of pixels or as the percentage of page height.
	Syntax:	`HEIGHT = "n"  - where n is a natural number`

Know More!

Exactly how big is one pixel? Does it have a specific size? The answer is that a pixel does not have a specific size. Its size depends on a number of things. One of them is the resolution of the monitor being used. A high resolution monitor has a smaller size pixel than a low resolution monitor because a high resolution monitor has more pixels per inch (or centimetre). Therefore the size of your image will vary a little from computer to computer. However, your images (and text) will always be in the right proportion no matter which monitor they are viewed on - and so they will always look good in any browser.

FONT element - It is a container element started by <FONT> tag and ended by </FONT> tag. The FONT element is used to add style, size, and color to the text. We use the size, color, and face attributes to customize our fonts.

Syntax: `<FONT FACE="font name" SIZE = n COLOR = "color" > . . .</FONT>`

Attributes of FONT element are discussed below:

1.	**Attribute:**	FACE
	Use:	FACE is used to specify the name of the font. A list of font names separated by commas can be specified. If the first font is available on the system, it will be used. Otherwise the second will be tried, and so on. If none is available, the default font will be used.
	Syntax:	`FACE = "font name"`
2.	**Attribute:**	SIZE
	Use:	The SIZE attribute is used to specify the font size between 1 and 7 (7 is the largest).
	Syntax:	`SIZE = "n" (n is a natural number between 1 and 7)`
3.	**Attribute:**	COLOR
	Use:	Sets th color of the text. The color can be set by giving the name of the color or its hexadecimal value.
	Syntax:	`COLOR = "color"`

Given below is a list of common font faces which are supported by most browsers.

Arial	Comic Sans MS	Lucida Console
Arial Black	Courier New	Tahoma
Arial Narrow	Georgia	Times New Roman
Bookman Old Style	**Impact**	Trebuchet MS
Century Gothic	Mangal	Verdana

In the web pages designed so far the text was displayed in the form of a paragraph, but if we want to communicate some information in the form of bulleted/numbered lists then we have to use the LIST element. A page designed using list elements is shown below:

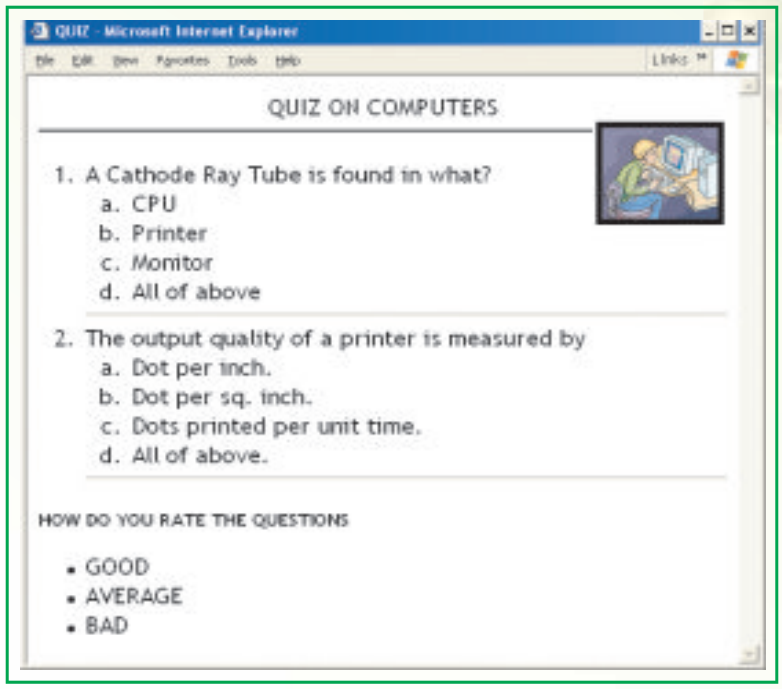


Figure 7.9 Web Page designed using Lists

Let us now look at the code for the web page shown in figure 7.9.

```
<HTML>
   <HEAD> <TITLE>QUIZ</TITLE> </HEAD>

  <BODY>
    <FONT FACE = "Times New Roman" size = "4">
    <CENTER>QUIZ ON COMPUTERS</CENTER>
    <IMG BORDER="5" SRC="QUIZ.jpg" ALIGN="RIGHT">
    <HR color="BLACK">
    <COMMENT>Beginning of first ordered list</COMMENT>
    <OL START=1>
      <LI>A Cathode Ray Tube is found in what?
      <COMMENT>Beginning of second ordered list</COMMENT>
        <OL START=1 TYPE=a>
          <LI>CPU
          <LI>Printer
          <LI>Monitor
          <LI>All of above
```

```
        </OL>
        <HR>
       <LI>The output quality of a printer is measured by
        <OL START=1 TYPE=a>
          <LI>Dot per inch.
          <LI>Dot per sq. inch.
          <LI>Dots printed per unit time.
          <LI>All of above.
        </OL>
        <HR>
    </OL>
    <H5>HOW DO YOU RATE THE QUESTIONS</H5>
    <UL TYPE=CIRCLE>
      <LI>GOOD
      <LI>AVERAGE
      <LI>BAD
      </UL>
     </FONT>
  </BODY>
  </HTML>
```

Figure 7.10 Code for Figure 7.9

Lists - Lists help us to organize the contents on the web page in an ordered and sequential manner. HTML supports three types of lists:

- Unordered lists (bulleted lists)
- Ordered lists (numbered)
- Definition Lists

In this chapter we shall study Unordered Lists and Ordered Lists only.

Ordered List (OL) element - It is a container element started by <OL> tag and ended by </OL> tag. It displays a numbered list. In a numbered list each item is preceded by a number or a letter. This element is used where the items are to be placed in a specific order.

Syntax: **<OL START = "n" TYPE = "A"/"a"/"I"/"i"/"1" > . . . </OL>**

Attributes of OL element are discussed below:

1.	**Attribute:**	START
	Use:	START indicates the starting number (or the serial number of the first element) of the list. This number must be a positive integer.
	Syntax:	START = "n" - where n is a natural number.
2.	**Attribute:**	TYPE
	Use:	TYPE defines the type of numbering sequence used for the list items. "A" specifies a sequence of uppercase letters. "a" specifies sequence of lowercase letters. "I" specifies a sequence of uppercase Roman numerals. "i" specifies a sequence of lowercase Roman numerals. "1" specifies a sequence of numbers.
	Syntax:	TYPE = "A"/"a"/"I"/"i"/"1"

Unordered List (UL) element - It is a container element started by <UL> tag and ended by </UL> tag. It displays a bulleted list. In a bulleted list each item is preceded by a small symbol called a bullet. The shape of the bullet can be a Circle, a Disk, or a Square.

Syntax: **`<UL Type="value">. . . </UL>`**

Attributes of UL element are discussed below:

1.	**Attribute:**	TYPE
	Use:	TYPE defines the type of bullet used for each list item. The value can be: o CIRCLE specifies a hollow bullet. " DISC specifies a solid round bullet. " SQUARE specifies a square bullet.
	Syntax:	**`START = "value"`**

List Item (LI) element - The LI element is an empty element specified by <LI> tag. It is used inside OL and UL elements to define list items. Each list item has to be preceded by <LI> tag.

Syntax: **<LI> Item Name**

By itself (if used without <OL> or <UL>), <LI> displays items in a bulleted list form. It means that we can use LI element without OL or UL element also. But it is advised not to do so.

Nested Lists - Nested lists occur when we put one list inside another list. The nested list would be a list item (marked by <LI>) of the parent list.

Can you write the code for the nested list as shown below:

```
1. Book Mathematics
    a. Chapter 1    b. Chapter 2    c. Chapter 3
2. Book English
    a. Chapter 1    b. Chapter 2    c. Chapter 3
3. Book Science
    a. Chapter 1    b. Chapter 2    c. Chapter 3
```

Tables - Let us now design a web page where we will display data in an organized manner with the use of tables. Tables are very useful in the communication of data in a meaningful way. Two examples of tables are shown below:

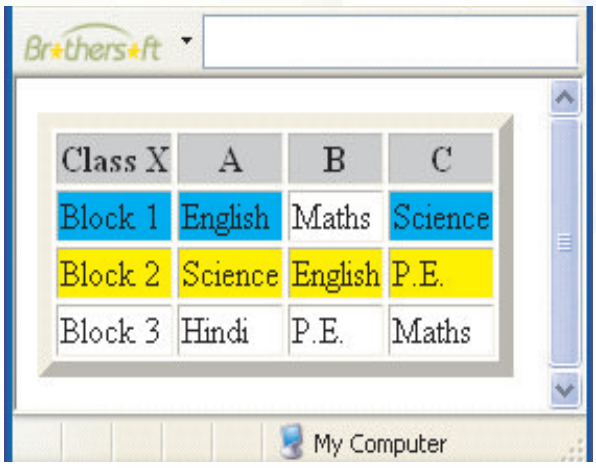


Figure 7.11 Time Table

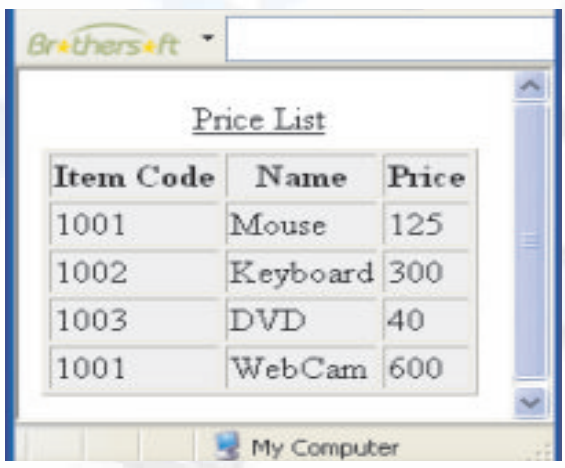


Figure 7.12 Price List made using table

Let us now look at the code for designing these web pages using tables:

```
<HTML>
 <BODY>
   <TABLE BORDER="7" CELLPADDING="7" CELLSPACING="10">
   <COMMENT>Beginning of table row</COMMENT>
     <TR BGCOLOR="#C0C0C0">
       <TD>Class X </TD>
       <TD> A</TD>
       <TD> B</TD>
       <TD> C</TD>
     </TR>
   <COMMENT>Beginning of second table row</COMMENT>
     <TR BGCOLOR="#00FFFF">
       <TD> Block 1</TD>
       <TD>English</TD>
       <TD BGCOLOR="#FFFFFF">Maths</TD>
       <TD>Science</TD>
     <TR BGCOLOR="#FFFF00">
       <TD> Block 2</TD>
       <TD>Science</TD>
       <TD>English</TD>
       <TD ROWSPAN="2">P.E.</TD>
     </TR>
     <TR BGCOLOR="#FFFFFF">
       <TD> Block 3</TD>
       <TD colspan=2>Hindi</TD>
      </TR>
   </TABLE>
 </BODY>
 </HTML>
```

Figure 7.13 Code for Time Table (Figure 7.11)

```
<HTML>
   <BODY>
     <TABLE BGCOLOR="#EEEEEE" BORDER="1">
     <CAPTION> <u>Price List</u> </CAPTION>
     <TR>
       <TH>Item Code</TH>
       <TH>Name</TH>
       <TH>Price</TH>
     </TR>
     <TR>
       <TD>1001</TD> <TD>Mouse</TD> <TD>125</TD>
     </TR>
     <TR>
       <TD>1002</TD> <TD>Keyboard</TD> <TD>300</TD>
     </TR>
     <TR>
       <TD>1003</TD> <TD>DVD</TD> <TD>40</TD>
     </TR>
     <TR>
       <TD>1001</TD> <TD>WebCam</TD> <TD>600</TD>
     </TR>
  </TABLE>
  </BODY>
  </HTML>
```

Figure 7.14 Code for Price List (Figure 7.12)

Let us now try and understand the code:

TABLE element - It is a container element started by <TABLE> tag and ended by </TABLE> tag. It is the basic element for creating a table.

Syntax:

```
<TABLE ALIGN="Alignment" BGCOLOR="colorname" BORDER="n"
                        >. . .</TABLE>
```

Attributes of TABLE how element are discussed below:

1.	**Attribute:**	ALIGN
	Use:	Specifies the horizontal placement of the table in relation to the window screen. LEFT aligns the table to the left (the default). RIGHT aligns the table to the right. CENTER aligns the table to the centre.
	Syntax:	`ALIGN = "Alignment"`
2.	**Attribute:**	BGCOLOR
	Use:	Sets the colour of the background for the table.
	Syntax:	`BGCOLOR = "colorname"`
3.	**Attribute:**	BORDER
	Use:	Sets the border size (width of the boundaries) of the table.
	Syntax:	`BORDER = "n"  - where n is a natural number.`

TABLE ROW (TR) element - It is a container element started by <TR> tag and ended by </TR> tag. It is used to define table rows. The TR element is used inside the TABLE element. The number of rows in a table corresponds to the instances of the TR element within the table element. The contents of a row are given as a parameter to the TR element. The contents of a table row may contain TH tags, which indicate table headings, and TD tags, which indicate table data.

Syntax:

```
<TR ALIGN ="Alignment" VALIGN="VAlignment"
                BGCOLOR="colorname"> . . . < / TR >
```

Attributes of TABLE ROW element are discussed below:

1.	**Attribute:**	ALIGN
	Use:	This controls whether the text in a row is aligned to the left, right, or centre of a cell. Default alignment is Left.
	Syntax:	`ALIGN = "Alignment"`
2.	**Attribute:**	VALIGN
	Use:	This controls whether the text in a row is aligned to the top, bottom, or middle of a cell. BOTTOM aligns the content with the cell's bottom. MIDDLE centres the content within the cell (the default). TOP aligns the content with the cell's top.
	Syntax:	`VALIGN = "VAlignment"`
3.	**Attribute:**	BGCOLOR
	Use:	It sets the background colour of the table row.
	Syntax:	`BGCOLOR = "colorname"`

TABLE DATA (TD) element - It is a container element started by <TD> tag and ended by </TD> tag. It is used to specify the text in a cell of the table. The TD tag is used inside the TR tag, which is inside the table tag. The number of columns in a table depends on the number of TD elements within the TR element.

Syntax:

```
<TD ALIGN="Alignment" VALIGN="VAlignment"
        BGCOLOR="colorname" > . . . </TD>
```

The attributes ALIGN, VALIGN, and BGCOLOR have their usual meaning as defined in <TR> element.

If used the attributes of the <TD> tag will replace the attributes of the <TR> tag.

TABLE HEADING (TH) element - It is a container element started by <TH> tag and ended by </TH> tag. It is used to create header values. The header values are displayed in a bold font and centre aligned. The TH element is used to create column or row headings.

Syntax:

```
<TH ALIGN ="Alignment" VALIGN = "VAlignment"
BGCOLOR = "colorname" NOWRAP >. . .</TH>
```

- The attributes ALIGN, VALIGN, and BGCOLOR have their usual meaning as defined in <TR> element.
- NOWRAP - Specifies that the contents of a cell cannot be broken, i.e., they do not wrap onto the next line.

So far we have studied about the usual HTML commands like <HTML>, <HEAD> and <BODY>. We have also learnt how to create lists and tables. But there is a limitation in all the web pages that we have created, they are not interactive. When we view our document in a web browser, we can merely view the information, we cannot interact with the web page.

FORMS in HTML - We can make a web page interactive by creating form(s) in it. A form allows the user to enter some data and this data can then be sent to a web server or to another web page to respond accordingly. Forms in HTML are used to handle operations like taking orders, conducting surveys, user registration etc. Let us now design a web page which is interactive wherein the user can enter some data. Look at the web page shown below:

Let us now write the code to make the web page shown above.

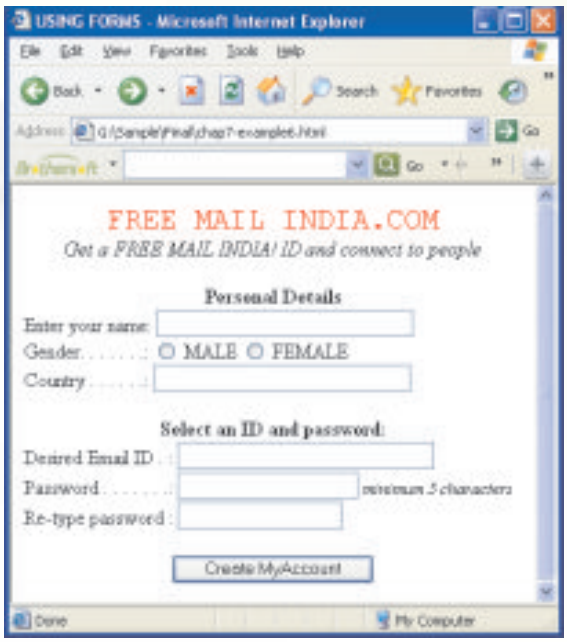


Figure 7.15 A Web Page containing a Form

```
<HTML>
   <HEAD> <TITLE> USING FORMS </TITLE> </HEAD>
   <BODY>
     <CENTER>
     <FONT face="Courier New" size=5 color=red>FREE MAIL
 INDIA.COM</FONT>
     <br>
     <FONT size=3>
       <I>Get a FREE MAIL INDIA! ID and connect to people</I>
     </FONT>
 <COMMENT>Beginning of form tag</COMMENT>
     <FORM NAME="FORM1" ACTION = "validate.html" METHOD = "GET">
       <B> Personal Details </B> </CENTER>
       Enter your name:
       <INPUT TYPE="TEXT" SIZE=30 NAME="t1"> <br>
       Gender. . . . . .:
       <INPUT TYPE="RADIO" NAME="r1">  MALE
       <INPUT TYPE="RADIO" NAME="r1">  FEMALE <br>
       Country . . . . . .:
       <INPUT TYPE="TEXT" SIZE=30 NAME="t2"> <br><br>
       <B><CENTER>Select an ID and password:</CENTER></B>
       Desired Email ID . :
       <INPUT TYPE="TEXT" SIZE=30 NAME="t3"> <BR>
       Password . . . . . .:
       <INPUT TYPE="PASSWORD" SIZE=20 NAME="t4">
       <FONT size=2><I> minimum 5 characters</I></FONT> <br>
       Re-type password :
       <INPUT TYPE="PASSWORD"  SIZE=18 NAME="t5"> <br><br>
       <CENTER><INPUT TYPE="SUBMIT" value="Create MyAccount"
 name="s1"></CENTER>
     </FORM>
   </BODY>
 </HTML>
```

Figure 7.16 Code for web Page containing Form tag

We have used a FORM element to create the form. We have used interface element INPUT with different values of TYPE attribute such as TEXT, RADIO, PASSWORD, and SUBMIT, to get the user input. All the elements in the form must be defined between the <FORM> and </FORM> tags. Other elements, such as heading, paragraph, and tables, etc., can also be used in the <FORM> and </FORM> tags.

Thus, when the form is displayed in a web browser, the user can make choices and text can be entered by using the interface elements defined with <INPUT> tag. The form made

can finally be submitted to a destination point whenever required. All these things will be discussed one by one as we progress.

FORM element - It is a container element started by <FORM> tag and ended by </FORM> tag. It is used to create a form on a web page.

Syntax: `<FORM NAME="FormName" ACTION = "URL" METHOD = "method">`

Attributes of FORM element are discussed below:

1.	**Attribute:**	NAME
	Use:	This specifies the name of the form. But this name will not be displayed on the form. As there can be more than one FORMs in an HTML document, a name is required to differentiate one form from another. The NAME attribute is optional if there is only one FORM on the web page.
	Syntax:	NAME = "FormName"
2.	**Attribute:**	ACTION
	Use:	This specifies the URL where the form-data is sent when the form is submitted. This URL is also called the destination of the form.
	Syntax:	ACTION = "URL"
3.	**Attribute:**	METHOD
	Use:	This specifies how the form-data is submitted. Form-data can be submitted using the methods get or post. With METHOD = "get", the form-data is submitted as URL variables, and with METHOD = "post", the form-data is submitted as HTTP post.
	Syntax:	METHOD = "method"

INPUT element - It is an empty element specified by <INPUT> tag. It is used to provide an input field in a form where the user can enter the data. An input field may be a textfield, a checkbox, a radio button, a button, and more. INPUT element is always used within the FORM element. Thus, INPUT element defines an object on the FORM which can receive user input.

Syntax: `<INPUT TYPE="FieldType" NAME="FieldName" VALUE="FieldText">`

Attributes of INPUT element are discussed below:

1.	**Attribute:**	TYPE
	Use:	The TYPE attribute determines the field type of input field to be provided in the form. Field type can be one of the following: TEXT, PASSWORD, RADIO, CHECKBOX, SUBMIT, RESET, BUTTON, IMAGE, HIDDEN, FILE.
	Syntax:	TYPE = "FieldType"
2.	**Attribute:**	NAME
	Use:	Specifies the name of the field. This name does not appear on the FORM. It is required for the identification/ differentiation as there can be more than one fields in a single FORM.
	Syntax:	NAME = "FieldName"
3.	**Attribute:**	VALUE
	Use:	Specifies the text to be displayed on the field.
	Syntax:	VALUE = "FieldText"

We will now discuss different field types (values of TYPE attribute) relevant to our syllabus.

1.	**FieldType: TEXT** Use: It provides a single line text input field where the user can enter text. **Additional Attributes:** SIZE = "n" - Sets the visible size of the text field to n characters. MAXLENGTH = "n" - Set the maximum number of characters that can be input in the text field to n.
2.	**FieldType: PASSWORD** **Use:** It provides a single line text input field where the user can enter password. A password field is different from a text field because a text field displays whatever characters are entered by the user whereas a password field shows one dot for each character input by the user. This is to prevent others from seeing the password. **Additional Attributes:** Same as those for TEXT field.
3.	**FieldType: RADIO** **Use:** It provides a radio button on the form. More than one radio buttons can have (and in general have) the same name. All the radio buttons that have the same name constitute a radio group. Only one radio button of a group can be selected at one time. That is, from a group of radio buttons, if the user selects a button, all the other buttons in the set are deselected. When a form is submitted, selected radio button's value (specified by the VALUE attribute) is submitted to the destination. **Additional Attributes:** **CHECKED -** Indicates that the radio button is selected, which can be deselected when another choice is made. At one time, only one radio button in a radio group can be specified as CHECKED.

4.	**FieldType: CHECKBOX** **Use:** It provides a check box on the form. With checkboxes, we can give the users a list of items to choose from. The user can choose more than one items from the list. We can make a group of checkboxes, by giving them the same name. When a form is submitted, selected cheboxes' values (specified by the VALUE attribute) are submitted to the destination. **Additional Attributes:** **CHECKED -** Indicates that the checkbox is to be displayed with a tick mark to show selection. This attribute is optional.
5.	**FieldType: SUBMIT** **Use:** This provides a button on the form. When this button is clicked, the form is submitted to the destination. **Additional Attributes:** None -
6.	**FieldType: RESET** **Use:** This provides a button on the form. When this button is clicked, the input fields on the form are reset to their default state. **Additional Attributes:** None -

XML

XML (eXtensible Markup Language) is also a markup language like HTML. But XML is different from HTML in the sense that HTML describes how to display and format the data, text and images in the browser whereas XML is used to describe the data. XML has nothing to do with presentation of data in the browser. The XML standard was created by W3C to provide an easy to use and standardized way to store self-describing data (Self-describing data is data that describes both its content and its structure).

The main benefit of XML is that it can be used to share data between two entirely different platforms. For example, we can take data from a database like MySQL, convert it into XML, and then share it with any other platform like MS Excel, HTML etc. Each of these receiving platforms can then convert the XML into a structure the platform uses normally. This way communication between two potentially different platforms is achieved using XML.

To store self-describing data, XML allows us to create our own tags. Once an XML file is created with the help of user-defined tags, it needs to be transformed into the target platform format. We shall take an example in which data in an XML file is transformed into HTML format. The file used for this purpose is an XSLT (eXtensible Stylesheet Language Transformations) file.

Example

Following is an XML code to store the data of a class - names of three students and one teacher. You can type this code in any text editor and save the file with any name and extension XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="class.xsl"?>
<class>
<student>Sumedha</student>
<student>Utkarsh</student>
<student>Tushar</student>
<teacher>Anil Kumar</teacher>
</class>
```

Figure 7.17 XML Code

Let us try to understand this code :

- **<?xml version="1.0" encoding = "UTF-8"?>**

  The first line in the code is the XML declaration which should always be

included. This is called the 'XML prolog'. The prolog in this example specifies that the XML code confirms to version 1.0 of XML standard, and the encoding scheme used is "UTF-8".

- **<?xml-stylesheet type="text/xsl" href="class.xsl"?>**

  This statement links this xml file to the file class.xsl. It means that the file class.xsl (shown later) will use this xml file and apply its rules to transform this XML document.

- **<class>...</class>**

  Each XML document contains one and only one root element (or parent element) which encapsulates the data described in the XML file. In this example <class> . . . </class> is the root element. The root element further contains sub-elements (or child elements) which describe the actual data values. In this example <student> . . . </student> and <teacher> . . . </teacher> are the sub-elements of <class> . . . </class>.

So, the above XML file stores the data of a class which has:

3 students: Sumedha, Utkarsh, and Tushar

1 Teacher: Anil Kumar

This XML file data will be transformed and represented in HTML format by the file class.xsl. This file name was specified in the XML code itself.The code of class.xsl is given below. You can type this code in a text editor and save the file as class.xsl. class.xsl becomes an XSLT (Extensible Stylesheet Language Transformations) file.

```
<?xml version="1.0" ?>

<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="teacher">

            <p><u><xsl:value-of select="."/></u></p>

    </xsl:template>
```

```
        <xsl:template match="student">
                <p><b><xsl:value-of select="."/></b></p>
        </xsl:template>
        <xsl:template match="/">
                <html>
                <body>
                <xsl:apply-templates/>
                </body>
                </html>
        </xsl:template>
</xsl:stylesheet>
```

Figure 7.18 Class.xsl code

Don't worry if this code is not clear to you. At this stage our aim is to understand the concept of data storage in an XML file and not to understand the syntax of XSLT files.

Now if you open the above XML file in a browser, you will see the data in the following format:

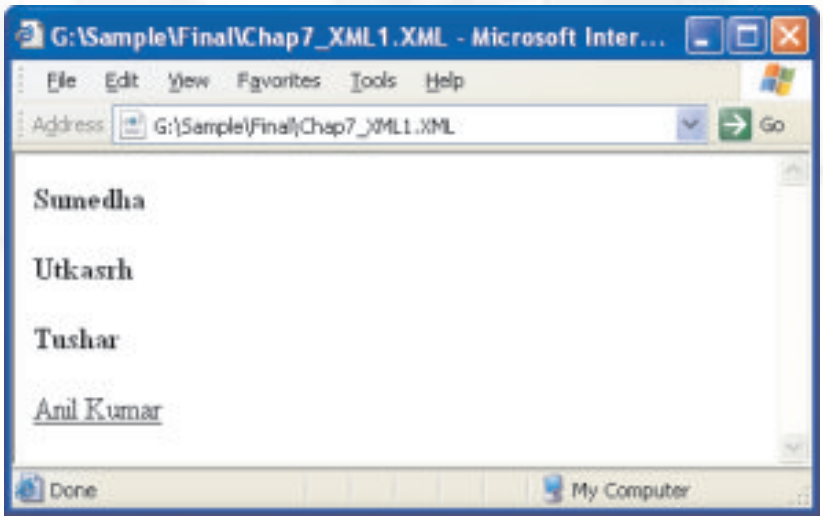


Figure 7.19 Output of XML file

Suppose Class.XSL has the following code:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="teacher">
        <p>Teacher:<u><xsl:value-of select="."/></u></p>
    </xsl:template>
    <xsl:template match="student">
        <p>Student: <b><xsl:value-of select="."/></b></p>
    </xsl:template>
    <xsl:template match="/">
        <html>
        <body>
        <xsl:apply-templates/>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Figure 7.20 Code for XSL sheet

Then the output would be:

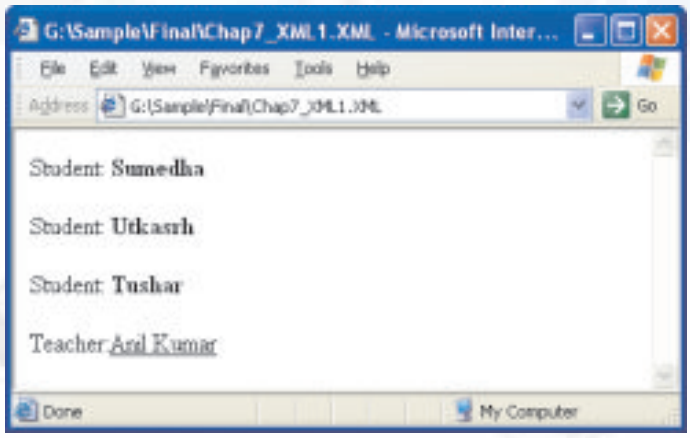


Figure 7.21 Output for XSL sheet

So the same data in an XML file can be represented in different ways. It all depends on how the data is transformed and presented.

The main difference between XML and HTML

- HTML is used to display data and to focus on formatting of data, whereas XML is used to describe data and focus on what data is.
- HTML tags are predefined, whereas XML tags are not predefined. We create our own tags.
- HTML tags are not case sensitive, whereas XML tags are case sensitive.

Features of XML

- XML is extensible. XML allows the user to create his own tags and document structure.
- XML can be used to store data. Applications can be written to store and retrieve information.
- XML can be used to exchange data. In the real world, databases contain data in different formats. It is difficult to exchange data between such systems over the Internet. Converting the data to XML can help in solving this problem and create data that can be read by different types of applications.
- XML is free. It can be written with a simple text editor or one of the many freely available XML authoring tools, such as XML Notepad.
- XML is a W3C recommendation.

Future Trends

In the last fifteen years internet has reinvented itself. Nowadays we shop, bank, work, meet people online and at times share what we are doing at any given moment of time. Technology is always evolving - and none quite as fast as the Internet. Let us try to visualize the future trends of internet.

Audio Web Surfing

In future we might see people traveling with headphones attached to their mobile devices while a Text-to-Speech application reads them the latest articles from their favorite sites. Browser will be able to comprehend the spoken command and open the particular website which the user has requested for. Audio surfing could be useful for commuters, children learning to read, tutors etc. For web developers, there may be new accessibility opportunities especially for multi-lingual sites.

Integration of data and applications

In future with the help of development in semantic web technologies we will click on the web page for the meeting, and our computer, will be able to comprehend it as a form of appointment, will pick up all the right information, and understand it and send it to all the right applications. Further, it will evoke those applications directly (using web services) needing little or no human intervention.

Web Accessibility for all

The W3C Web Accessibility Initiative (WAI) brings together people from industry, disability organizations, government, and research labs from around the world to develop guidelines and resources to help make the Web accessible to people with disabilities like auditory, cognitive, neurological, physical, speech, and visual disabilities.In future hardware might be adaptable for people with motoric disabilities. Software accessibility solutions for the future might include physical modelling. For instance in case of sight disabled there are no or little audio feedback from the games where concepts of physics are involved.In the future such games might be connected with a sound engine or realtime audio.

Summary

- Web server delivers (serves) content, such as web pages, using the Hypertext Transfer Protocol (HTTP), over the World Wide Web.
- A web browser is a client that initiates communication by making a request for a specific resource. The server then responds with the content of that resource, or an error message if unable to do provide the contents due to any reason.
- URL represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be a file or a web page.
- Various HTML tags are summarised in the following table:

Structure Elements			
Element	**Start Tag**	**Attributes**	**End Tag**
HTML File	<html>	none	</html>
File Header	<head>	none	</head>
File Title	<title>	none	</title>
Comment	<! . . .> <Comment>	Your comments go between <! and >.	</Comment>
Body	<body>	background, bgcolor, text, link, alink, vlink	</body>

Basic Text Elements			
Element	**Start Tag**	**Attributes**	**End Tag**
Line Break	 	none	
Paragraph		align	
Bold		none	
Italic		none	
Headline	<hn>	align	</hn>
Font		Face, size, color	
Horizontal Rule	<hr>	Size, width, noshade, align	

List Elements			
Element	**Start Tag**	**Attributes**	**End Tag**
Unordered List		type	
Ordered List		type, start	
List Item		none	

Image Element			
Element	**Start Tag**	**Attributes**	**End Tag**
Image	<img>	src, align, width, height	</img>

Table Elements			
Element	**Start Tag**	**Attributes**	**End Tag**
Table	<table>	Align, bgcolor, border	</table>
Table Row	<tr>	Align, valign, bgcolor	</tr>
Table Data	<td>	Align, valign, bgcolor	</td>
Table Header	<th>	Align, valign, bgcolor	</th>
Caption	<caption>	Align, valign, bgcolor	</caption>

Form Elements			
Element	**Start Tag**	**Attributes**	**End Tag**
Form	<form>	name, action, method	</form>
Input Field	<input>	Name, type	</input>

- XML stands for eXtensible Markup Language. It is a markup language like HTML but its is used to store the data and not to describe the presentation of data.
- Using XML data can be shared among various platforms.

EXERCISES

MULTIPLE CHOICE QUESTIONS

1. The address of a resource on the net is known as:

   (a) ISP (b) HTTP

   (c) URL (d) WWW

2. A program that serves requested HTML files and pages.

   (a) Web Address (b) Web Page

   (c) Web Server (d) None of these

3. HTML tags must be written within:

   (a) < > (b) { }

   (c) [ ] (d) ( )

4. Which of the following is the correct structure of HTML tags?

   (a) < HTML> </HTML> <HEAD> </HEAD> <BODY> </BODY>

   (b) <HTML> <HEAD> </HEAD> </HTML> <BODY> </BODY>

   (c) <HTML> <HEAD> <BODY> </BODY> </HEAD> </HTML>

   (d) <HTML> <HEAD> </HEAD> <BODY> </BODY> </HTML>

5. Which of the following tags is used to specify the items in a list.

(a) <LI> (b) <OL>

(c) <UL> (d) <DL>

6. The IMAGE tag uses the _____ attribute to specify the URL of the image to be displayed.

(a) SCR (b) SRC

(c) Source (d) None of these

7. The align attribute of <Table> tag refers to _____ placement of the table in relation to the window screen.

(a) Vertical (b) horizontal

(c) Both a and b (d) None of these

8. Choose the best suitable input type to input gender from the user:

(a) Text (b) Submit

(c) checkbox (d) Radio

9. Elements, such as heading, paragraph and tables etc can be contained in the FORM element.

(a) Holds True always (b) Holds True Sometimes

(c) Is Never True (d) None of these

10. Data entry on a web page can be done using:

(a) Tables (b) Formatting Tags

(c) Forms (d) Lists

11. Which tag is used to embed an image in an HTML document.

(a) <FIX> (b) <IMG>

(c) <IMAGE> (d) <FIX IMAGE>

12. To create a nested list we use the tag:

    (a) <OL> (b) <UL>

    (c) <NL> (d) combination of (a) and (b) as required.

13. Input type="_____" will send the form to its destination place.

    (a) Button (b) File

    (c) Submit (d) Reset

14. FACE is the attribute of which tag:

    (a) <BODY> (b) <FONT>

    (c) <P> (d) <IMG>

15. Radio buttons can be grouped together so that only one is selected at a time by using the attribute:

    (a) name (b) selected

    (c) checked (d) font

16. XML document is used to

    (a) Only interpret data

    (b) Store any kind of data

    (c) Store only highly structured data (like in databases)

    (d) Store only loosely structured data (like letters)

17. XML can be used to

    (a) Exchange data (b) Store data

    (c) Interpret data (d) All of these

18. Every XML document must begin with a

    (a) Root element (b) Child element

    (c) XML version details (d) Any of these

ANSWER THE FOLLOWING QUESTIONS

1. Explain the basic structure of an HTML document with the help of an example.
2. Explain with an example the difference between container and empty elements.
3. What is the importance of the BODY element? List the commonly used attributes within the <BODY> tag.
4. What is the purpose of the TITLE element in HTML?
5. Explain the src and align attribute of the IMG tag.
6. What is the difference between the <TD> and <TH> tags?
7. Explain align and bgcolor attributes of the <TR> tag.
8. How can we create an HTML page that allows the user to enter his details on that webpage? Explain with the help of suitable HTML code/example.
9. Mention the tags that are required to be used to create the given table.

Student Details

Name	Age	Gender
Nishi	14	Female
Rajat	16	Male

10. Differentiate between HTML and XML.
11. Write a few features of XML.

LAB EXERCISES

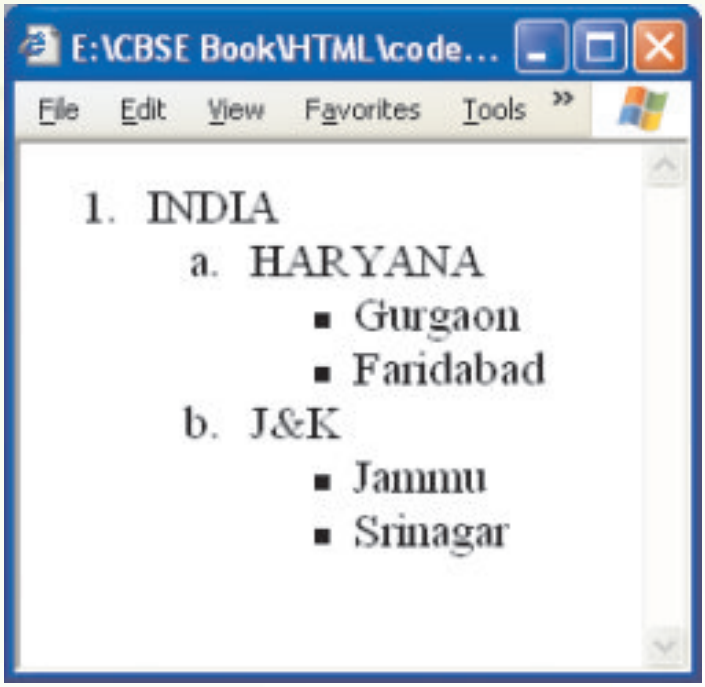
1. Write HTML code to display the following web pages:

a)

b)

Fresh Fruits	
Fresh Fruits help us stay healthy and fit. फल सेहत के लिए लाभदायक है	

d)



2. Write the HTML code to generate the following result:

a) ONE 1 2 3

b) TWO

ii. ONE

iii. TWO

iv. THREE

c) Things we learned in kindergarten:

1. share
2. play fair
3. don't hit people
4. put things back where we found them
5. say sorry when we hurt somebody

d) 1. Seek expert advice about the area

- Get the best maps. On the map select
  - landmarks
  - mountains
  - lakes
- Get a good compass and
  - check slope of land
  - check direction of flowing streams

2. If there is snow on the ground, stay close to:

- roads
- trails and
- waterways

3. Give the output for the following:

a)

```
< HTML >
< BODY >
<H1>The following is a list of a few chemicals: </H1)
<UL type="disc">
<LI> Sodium
<LI> Sulphur
<LI> Magnesium </UL>
</BODY>
</HTML>
```

b)
```
<HTML>
<HEAD><TITLE>SCHEDULE</TITLE></HEAD>
<BODY>
<TABLE BORDER="3">
<CAPTION>VOLUNTEER SCHEDULE</CAPTION>
<TR BGCOLOR="#AAFFCC">
    <TH>9 A.M.</TH> <TH>12 P.M.</TH> <TH>2 P.M.</TH>
</TR>
<TR>
    <TD>Anmol </TD> <TD>Abhinav </TD> <TD>Anika </TD>
</TR>
<TR>
    <TD>Riti </TD> <TD>Riya </TD> <TD>Sara </TD>
</TR>
<TR>
    <TD>Gul </TD> <TD>Reyana  </TD> <TD>Swayam </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

TEAM BASED TIME BOUND EXERCISE

(Team size recommended: 3 students each team)

Web Page Checklist

1. Students should perform search on a topic, such as: "nuclear armageddon" , "global warming" etc. and ask the team to give a presentation of about 3 minutes each.

2. The team should use this Checklist to try to evaluate systematically some of the search results.

Checklist Parameters	Website 1	Website 2
URL	_____	_____
Title	_____	_____
Personal / Oraganisational / Anonymous	☐ Personal ☐ Organisational ☐ Anonymous	☐ Personal ☐ Organisational ☐ Anonymous
What type of domain is it?	☐ com ☐ org/net ☐ edu ☐ gov/mil/us ☐ in ☐ other	☐ com ☐ org/net ☐ edu ☐ gov/mil/us ☐ in ☐ other
Does it correspond to the name of the site?	☐ Yes ☐ No	☐ Yes ☐ No
Appropriate for the content?	☐ Yes ☐ No	☐ Yes ☐ No
Navigation	☐ Simple ☐ Complex	☐ Simple ☐ Complex
All links on homepage work	☐ Yes ☐ No	☐ Yes ☐ No
Contact details	☐ Available ☐ Not Available	☐ Available ☐ Not Available

FIND OUT INFORMATION ON THE FOLLOWING TOPICS

- **Situation:** You have just acquired stocks of an IT firm. Find out how the stock is doing and find a copy of the annual report.
- **Situation:** Find information about Internet censorship? The class can be divided into 6 groups three groups can write in its favor and the rest can write against the topic.

MySQL - REVISION TOUR

Learning Objectives

After studying this lesson the students will be able to:

- Recall the terminology used in RDBMS concepts.
- Recall and effectively use the SQL commands and clauses learnt in class XI.
- Realize that much more is to be learnt for effective use of databases.

In the previous class, you have learnt some database concepts and SQL commands. You have also learnt how to access data from a table using various clauses of SELECT command. In this lesson we are going to revise the database concepts and SQL studied in class XI. That will make the base for further reading of the concepts and some more SQL commands which are really very useful in practical applications.

Database Concepts

You have already studied the following database concepts in class XI:

1. **Database:** A database is an organised collection of data.

   Software used to manage databases is called Data Base Management System (DBMS).

2. **Relational Database:** A database in which the data is stored in the form of relations (also called tables) is called a Relational Database. In other words a Relational Database is a collection of one or more tables.

3. **RDBMS:** A DBMS used to manage Relational Databases is called an RDBMS (Relational Data Base Management System). Some popular RDBMS software available are: Oracle, MySQL, Sybase, Ingress

4. **Benefits of using a DBMS are:**
   a. Redundancy can be controlled
   b. Inconsistence can be avoided
   c. Data can be shared
   d. Security restrictions can be applied.
5. **MySQL:** It is an Open Source RDBMS Software. It is available free of cost.
6. **Relation/Table:** A table refers to a two dimensional representation of data arranged in columns (also called fields or attributes) and rows (also called records or tuples). The tables in a database are generally related to each other to facilitate efficient management of the database. Interrelated tables also reduce the chances of errors in the database.
7. **Key:** A column or a combination of columns which can be used to identify one or more rows (tuples) in a table is called a key of the table.
8. **Primary Key:** The group of one or more columns used to uniquely identify each row of a relation is called its Primary Key.
9. **Candidate Key:** A column or a group of columns which can be used as the primary key of a relation is called a candidate key because it is one of the candidates available to be the primary key of the relation
10. **Alternate Key:** A candidate key of a table which is not made its primary key is called its Alternate Key.

Let us now revise MySQL concepts studied in class XI with the help of a Library database.

Note: *As this is a revision lesson, all variations of SQL commands are not covered here. It is suggested that they are revised through practical assignments.*

Think of the functioning of a computerised school library.

In a school library, there are books which are issued to the students and the staff. These books are also returned after a certain period of time. For proper functioning of this system the librarian maintains a database (an organised collection of data) which contains the relevant data of books, students, and staff. In a relational database, this data may be organised as a collection of the following tables:

BOOKS

AccNo	Title	Author	Publisher	Edition	Price
1001	Basic Economics	J.P. Mudgil	APH	2009	125
1002	MicroEconomics	Kavi Prakash	KPH	2009	175
1003	Macro Economics	Kavi Prakash	KPH	2010	200

AccNo is Accession Number of a book.

STUDENTS

AdmNo	LibCardNo	AccNo	IssueDate	ReturnDate
2010001	101	1002	2009-12-02	2009-12-08
2009012	102	1050	2010-02-16	NULL
2009013	102	1060	2010-02-16	NULL

STAFF

EmpNo	AccNo	IssueDate	ReturnDate
1	1001	2009-04-13	NULL
2	NULL	NULL	NULL
3	NULL	NULL	NULL

These three related tables collectively form the **Library** database.

Using the data stored in these tables the librarian can find out any type of relevant information regarding the books at any time. For example, she can find out:

- The list of books in the library which are published by any specific publisher or author.
- The list of books with price in a specified range.

- Admission numbers of the students who have not returned the books issued to them.
- The names of books issued to a particular staff member or student.

For the above tables the primary keys may be set as follows:

Table	Primary Key	Reason
BOOKS	AccNo	No two books have the same accession number.
STUDENTS	AdmNo	No two students have the same admission number.
STAFF	--	As there is no column for which each row has a unique value, there is no primary key of this table.

In the STUDENTS table LibCardNo can also be the primary key (Why?). It means that there are two columns in the table STUDENTS that are unique for each row and can be set as the primary key of the table. These two columns are *candidate keys* as they are the candidates for primary key. Out of these two candidate keys we have chosen AdmNo as the primary key. The other candidate key, i.e., LibCardNo, becomes the Alternate Key of the table. Had we chosen LibCardNo as the primary key, AdmNo would have been called the alternate key of the table.

In class XI we have studied that **degree** of a table is the number of columns in the table, and the number of rows in a table is called its **cardinality**.

Can you tell what are the degrees and cardinalities of tables of Library database?

Libraries function without computers also. Is there any advantage of setting up a computerised Library Database using an RDBMS? Discuss it with your teacher.

We shall be using MySQL for Library database.

Once MySQL is procured (downloaded from net, or copied from somewhere, or received free with some book), it can be installed and the process of creating Library database can be started.

Working with MySQL

You have already studied the following SQL concepts in class XI:

1. **SQL (Structured Query Language):** It is the language used to manipulate and manage databases and tables within them using an RDBMS.
2. **DDL (Data Definition Language):** This is a category of SQL commands. All the commands which are used to create, destroy, or restructure databases and tables come under this category. Examples of DDL commands are - CREATE, DROP, ALTER.
3. **DML (Data Manipulation Language):** This is a category of SQL commands. All the commands which are used to manipulate data within tables come under this category. Examples of DML commands are - INSERT, UPDATE, DELETE.
4. **DCL (Data Control Language):** This is a category of SQL commands. All the commands which are used to control the access to databases and tables fall under this category. Examples of DCL commands are - GRANT, REVOKE.

Different SQL commands studied in class XI are summarized below:

SNO	Command, Syntax, and Purpose	Category
1.	**Command:** CREATE DATABASE **Syntax:** CREATE DATABASE <databasename>; **Purpose:** Creates a database with the specified name.	DDL
2.	**Command:** CREATE TABLE **Syntax:** CREATE TABLE <tablename> (<column name1> <data type1> [,<column name2> <data type2>, . . <column nameN> <data typeN>]); **Purpose:** Creates a table with the specified name.	DDL

3.	**Command:** **Syntax:** **Purpose:**	ALTER TABLE ALTER TABLE <tablename> ADD <columnname> <datatype>; ALTER TABLE <tablename> DROP <columnname>; ALTER TABLE <tablename> MODIFY <column> <new_definition>; Modifies the structure of a table	DDL
4.	**Command:** **Syntax:** **Purpose:**	USE USE <databasename>; Opens the specified database for use.	DML
5.	**Command:** **Syntax:** **Purpose:**	SELECT DATABASE() SELECT DATABASE(); Shows the name of the current database	DML
6.	**Command:** **Syntax:** **Purpose:**	SHOW TABLES SHOW TABLES; Shows a list of tables present in the current database.	DML
7.	**Command:** **Syntax:** **Purpose:**	INSERT INSERT INTO <tablename> [<column1>, <column2>, ..., <columnn>] VALUES (<value1>, <value2>, ... <value n>); Inserts data into a table	DML

8.	**Command:** **Syntax:** **Purpose:**	SELECT SELECT <* / column name / expression> , [<column name/Expression list>] FROM <table name> [WHERE <condition>] [ORDER BY <column name / expression> [ASC/DESC]]; There are multiple ways to use SELECT. These will be explained with the help of examples in this lesson. Retrieves data from a table	DML
9.	**Command:** **Syntax:** **Purpose:**	DESCRIBE DESC[RIBE] <tablename>; Shows the structure of a table.	DML
10.	**Command:** **Syntax:** **Purpose:**	UPDATE UPDATE <tablename> SET <column name> = <value> [,<column name> = <value>, ...] [WHERE <condn>]; Updates/Modifies data in a table	DML
11.	**Command:** **Syntax:** **Purpose:**	DELETE DELETE FROM < tablename> [Where < condition>]; Deletes data from a table	DML

Following are the clauses which can be used with SELECT command:

a.	DISTINCT	Used to display distinct values from a column of a table.
b.	WHERE	Used to specify the condition based on which rows of a table are displayed.
c.	BETWEEN	Used to define the range of values within which the column values must fall to make a condition true. Range includes both the upper and the lower values.
d.	IN	Used to select values that match any value in a list of Specified values.
e.	LIKE	Used for pattern matching of string data using wildcard characters % and _ .
f.	IS NULL / NOT NULL	Used to select rows in which the specified column is NULL (or is NOT NULL)
g.	ORDER BY	Used to display the selected rows in ascending or in descending order of the specified column/expression.

Now, let us revise these commands with the help of Library database discussed above.

- After starting MySQL, our first job is to create a database in which all the tables will be stored. For this we give the statement:

```
CREATE DATABASE Library;
```

  This statement will create a database with the name Library using the command CREATE DATABASE.

- Now we have to open this database so that we can create tables and do other tasks in it. For this we give the following statement

```
USE Library;
```

After creating the database and opening it, we have to create tables. Before we create a table, we should very carefully plan columns that we want to have in the table and type of data to be stored in each column. The type of data decides what data types should be

specified for different columns of a table. There are many data types available in MySQL, but a few which are used most often are:

Class	Data Type
Text	CHAR(size)
	VARCHAR(size)
NUMERIC	DECIMAL(p,s)
	INT(size) or INTEGER(size)
date	DATE

After planning, we arrive at the following data types for columns of our tables in the Library database. Here the column sizes are based on our assumption. These may be different for different library databases.

BOOKS

Coulmn	Data Type	Reason
AccNo	INTEGER(5)	Accession number of a book is an integer in the range 1 to 99999.
Title	VARCHAR(30)	Title of a book may be maximum upto 30 characters long
Author	VARCHAR(30)	An author name may be maximum upto 30 characters long
Publisher	VARCHAR(10)	A publisher name may be maximum upto 10 characters long.
Edition	INTEGER(4)	Year of edition of the book
Price	DECIMAL(7, 2)	We assume that the price of a book will not exceed Rs. 99999.99

STUDENTS

Coulmn	Data Type	Reason
AdmNo	VARCHAR(7)	Here we assume that admission number consists of 4 digit year of admission followed by 3 digit admission serial number. Therefore, AdmNo 2009012 means that the students was admitted in the school in the year 2009 and he/she was the 12th student (in sequence) to take admission in that year.
LibCardNo	INTEGER(4)	Library card number is an integer in the range 1 to 9999.
AccNo	INTEGER(5)	Accession number of a book is an integer in the range 1 to 99999.
IssueDate	DATE	IssueDate is the date of issue of the book.
ReturnDate	DATE	ReturnDate is the date on which the book was returned.

STAFF

Coulmn	Data Type	Reason
EmpNo	INTEGER(3)	Employee number is an integer in the range 1 to 999.
AccNo	INTEGER(5)	Accession number of a book is an integer in the range 1 to 99999.
IssueDate	DATE	It is the date of issue of the book.
ReturnDate	DATE	It is the date on which the book was returned.

After deciding the columns to be kept in each table and their data types, we are ready to create the tables.

The statements to create these tables are as follows:

- To create the BOOKS table:

```
CREATE TABLE Books
    (AccNo INTEGER(5),
    Title VARCHAR(30),
    Author VARCHAR(30),
    Publisher VARCHAR(10),
    Edition INTEGER(4),
    Price DECIMAL(7,2));
```

- To create the STUDENTS table:

```
CREATE TABLE Students
    (AdmNo VARCHAR(7),
    LibCardNo INTEGER(4),
    AccNo INTEGER(5),
    IssueDate DATE,
    ReturnDate DATE);
```

- To create the STAFF table:

```
CREATE TABLE Staff
    (EmpNo INTEGER(3),
    AccNo INTEGER(5),
    IssueDate DATE,
    ReturnDate DATE);
```

- To verify that the tables are created, we give the statement:

```
SHOW TABLES;
```

- To further verify that the tables have been created as per the required specifications, we view the structure of each table by giving the statements:

```
DESC Books;

DESC Students;

DESC Staff;
```

- If the structure of a table is not as per the required specifications, we can modify it by using the ALTER TABLE command. Following are three examples of this:

(i) Suppose we decide to categorise the books into categories 'A', 'B' and 'C' based on some criteria. Then a new column Category of type CHAR(1) can be added to the table by the statement:

```
ALTER TABLE Books ADD Category CHAR(1);
```

(ii) Suppose we wish to change the size of column 'publisher' of Books table from 10 characters to 20 characters, then we can give the statement:

```
ALTER TABLE Books MODIFY Publisher VARCHAR(20);
```

(iii) Suppose we decide to remove the ReturnDate column from the table STAFF. We can do so by the statement:

```
ALTER TABLE Staff DROP ReturnDate;
```

- Once the tables are created as per the required specifications, we can populate these tables with the given sample data as follows:

→ For Books table:

```
INSERT INTO Books VALUES
    (1001, 'Basic Economics', 'J.P.Mudgil',
    'APH', 2009, 125);
INSERT INTO Books VALUES
    (1002, 'Micro Economics', 'Kavi Prakash',
    'KPH', 2009, 175);
```

→ For Students table:

```
INSERT INTO Students VALUES
    ('2010001', 101, 1002, '2009-12-02', '2009-12-08');
INSERT INTO Students VALUES
    ('2009012', 102, 1050,'2010-02-16', NULL);
```

→ For Staff table:

```
INSERT INTO Staff VALUES
        (1, 1001,'2009-04-13', NULL);
INSERT INTO Staff (EmpNO) VALUE (2);
```

- Now we can check whether the data has been entered into the tables correctly or not by entering the statements:

```
SELECT * FROM Books;
SELECT * FROM Students;
SELECT * FROM Staff;
```

We can use various clauses with SELECT command to display the data. Examples are shown below:

- To check the names (without repetition) of various publishers whose books are present in the library, we enter the statement:

```
SELECT DISTINCT Publisher FROM Books;
```

- To check the books of publisher 'APH' present in the library, we enter the statement:

```
SELECT * from Books WHERE Publisher = 'APH';
```

- To display the books list for which the price is between ₹ 250 and ₹ 500, we enter the statement:

```
SELECT * FROM Books WHERE Price > 250 and Price < 500;
```

- To display the books list for which the price is from ₹ 250 to ₹ 500, we enter the statement:

```
SELECT * FROM Books WHERE Price >= 250 and Price <= 500;
```

OR

```
SELECT * FROM Books WHERE Price BETWEEN 250 and 500;
```

- To display the details of books from the publishers 'APH' 'JPH', or 'ABD', we enter the statement:

```
SELECT * from Books
    WHERE Publisher = 'APH'
        OR Publisher = 'JPH'
        OR Publisher = 'ABD';
```

OR

```
SELECT * from Books
    WHERE Publisher IN ('APH', 'JPH', 'ABD');
```

- To list the AccNo, Title, and Price of all the books whose Title contains the word 'History', we can enter the statement:

```
SELECT AccNo, Title, Price from Books
    WHERE Title LIKE '%History%';
```

- To display the AccNo of all the books which have been issued to students but not returned by them, we enter the statement:

```
SELECT AccNo FROM Students
    WHERE IssueDate IS NOT NULL and ReturnDate IS NULL;
```

- To Display a List of all the Books in the alphabetical ascending order of Titles, we enter the statement:

```
SELECT * FROM Books ORDER BY Title;
```

- To get the same list in descending order of Titles, we enter the statement:

```
SELECT * FROM Books ORDER BY Title DESC;
```

- Now suppose a student returns a Book with AccNo 1245 on 12 August 2010, then this information can be recorded in the Students table as follows:

```
UPDATE Students
    SET ReturnDate = '2010-08-12'
        WHERE AccNo = 1245;
```

- If we want to delete the records of all the books by the Publisher 'PPP' with edition year earlier than 1990, we can enter the statement:

```
DELETE FROM Books
    WHERE Publisher = 'PPP' AND Edition < 1990;
```

Suppose for some analysis we decide to keep track of how many times each book is issued. This can be done if there is an extra column 'IssueFreq' in the table Books to keep this count. Every time a book is issued, the corresponding value in this column is incremented by 1. To add this extra column in the table, we enter the statement:

```
ALTER TABLE Books ADD IssueFreq INTEGER(3);
```

Functions in MySQL

Let us now revise the following single-row SQL functions which are available in MySQL:

Numeric Functions:

Sno	Name & Syntax	Description
1.	POWER(*x*,*y*) **OR** POW(*x*,*y*)	Returns the value of *x* raised to the power of *y*.
2.	ROUND(*x*)	Rounds the argument *x* to the nearest INTEGER.
3.	ROUND(*x*,*d*)	Rounds the argument *x* to d decimal places.
4.	TRUNCATE(*x*,*d*)	Truncates the argument *x* to d decimal places.

String Functions:

SNo	Name & Syntax	Description
1.	LENGTH(str)	Returns the length of a column or a string in bytes.
2.	CONCAT(str1,str2,...)	Returns the string that results from concatenating the arguments. May have one or more arguments.
3.	INSTR(str,substr)	Returns the position of the first occurrence of substring <substr> in the string <str>.
4.	LOWER(str) **or** LCASE(str)	Returns the argument <str> in lowercase. i.e., It changes all the characters of the passed string to lowercase.
5.	UPPER(str) **or** UCASE(str)	Returns the argument <str> in uppercase. i.e., It changes all the characters of the passed string to uppercase.
6.	LEFT(str,n)	Returns the first <n> characters from the string <str>
7.	RIGHT(str,n)	Returns the last <n> characters from the string <str>
8.	LTRIM(str)	Removes leading spaces, i.e., removes spaces from the left side of the string <str>.
9.	RTRIM(str)	Removes trailing spaces, i.e., removes spaces from the right side of the string <str>.
10.	TRIM(str)	Removes both leading and trailing spaces from the string <str>.
11.	SUBSTRING(str,m,n) **OR** SUBSTR(str, m, n) **OR** MID(str,m,n)	Returns <n> characters starting from the m^{th} character of the string <str>. If the third argument <n> is missing, then starting from m^{th} position, the rest of the string is returned. If <m> is negative, the beginning of the substring is the m^{th} character from the end of the string.
12.	ASCII(str)	Returns the ASCII value of the first character of the string <str>. Returns 0 if <str> is the empty string. Returns NULL if <str> is NULL.

Date and Time Functions:

SNo	Name & Syntax	Description
1	CURDATE()	Returns the current date in YYYY-MM-DD format or YYYYMMDD format, depending on whether the function is used in a string or numeric context.
2	NOW()	Returns the current date and time in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uuuuuu format, depending on whether the function is used in a string or numeric context.
3	SYSDATE()	Returns the current date and time in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uuuuuu format, depending on whether the function is used in a string or numeric context.
4	DATE(expr)	Extracts the date part of a date or datetime expression <expr>.
5	MONTH(date)	Returns the numeric month from the specified date, in the range 0 to 12. It returns 0 for dates such as '0000-00-00' or '2010-00-00' that have a zero month part.
6	YEAR(date)	Returns the year for specified date in the range 0 to 9999. It returns 0 for the "zero" date. Returns values like 1998, 2010,1996 etc.
7	DAYNAME(date)	It returns the name of the weekday for the specified date.
8	DAYOFMONTH(date)	Returns the day of the month in the range 0 to 31.
9	DAYOFWEEK(date)	Returns the day of week in number as 1 for Sunday, 2 for Monday and so on.
10	DAYOFYEAR(date)	Return the day of the year for the given date in numeric format in the range 1 to 366.

A few examples of these functions are given below:

1. Suppose a book with AccNo 1002 is lost by a student. The school charges a fine for the lost book. The fine is 10% of the book price. We wish to write a statement to calculate the fine rounded up to 2 DECIMAL places. The statement is

```
SELECT AccNo, ROUND(Price*10/100, 2) AS Fine
    FROM Books WHERE AccNo = 1002;
```

2. Suppose we want to create another table and one column in that table will be 'Author'. In order to optimally utilize the storage space, we wish to find out what is the maximum length of 'Author' in the table 'Books'. This value will guide us in deciding the field width of 'Author' in the new table. So we enter the statement:

```
SELECT Author, LENGTH(Author)
    FROM Books ORDER BY LENGTH(Author);
```

3. In order to produce a report we wish to display the book titles in upper case letters of the alphabet. So we enter the statement:

```
SELECT AccNo, UCASE(Title) FROM Books;
```

4. Suppose a book with accession number 1050 is returned and we wish to enter the current date as the ReturnDate in the table Students. We can give the statement:

```
UPDATE Students
    SET ReturnDate = CURDATE()
    WHERE AccNo = 1050;
```

5. Suppose we wish to find out which staff members got the books issued in the previous calender year, then we can give the statement:

```
SELECT * FROM Staff WHERE YEAR(IssueDate) =
    YEAR(CURDATE())-1;
```

What Next?

So far you have learnt how to work with single table at a time. None of the operations discussed in class XI or in this lesson required us to access data simultaneously from multiple tables. But in real life database implementations we have to access data from multiple tables simultaneously, as will be clear from the following examples.

Try to write queries for the following tasks in Library database:

(i) For all the books which have not been returned by the students, display the AccNO and Title of the book.

(ii) Display the details of all the books by the publisher 'APH' which are issued to the Staff.

(iii) Display the Price of each book which has been issued to the students but has not been returned.

If you observe carefully, you will find that these queries need to access data from two tables simultaneously. You have not yet studied how to do this.

In class XII, you will study SQL commands using which you will be able to do all such jobs and a lot more. For example:

(iv) Display the highest and the lowest values from Price column of the table Books.

(v) Display the Total number of books in the library whose edition is before 2001.

(vi) To cancel the operations done while working in MySQL.

(vii) To make the operations done permanent so that even in the case of system failure the work done stays.

(viii) To make sure that duplicate values are not accepted in the primary key column of a table.

(ix) To make sure that NULL is not accepted for any column in which we do not want to allow NULLs.

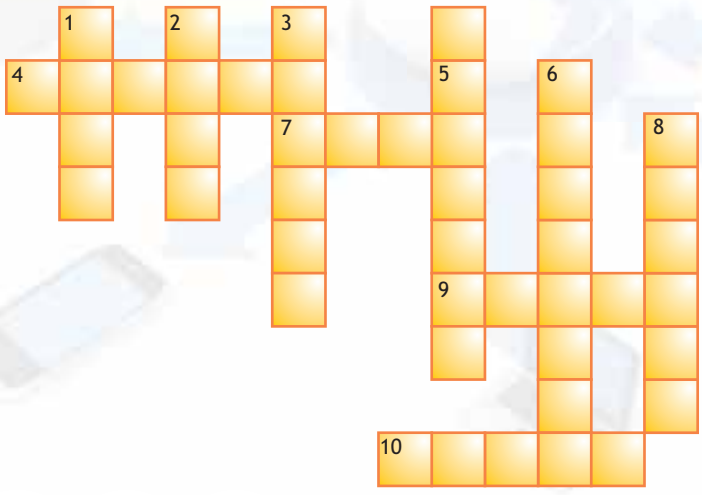
Summary:

- In an RDBMS data is stored in tables.
- A table refers to a two dimensional representation of data arranged in columns and rows.
- SQL (Structured Query Language) is used to manage databases in an RDBMS.
- SQL commands are divided into different categories: DDL, DML, and DCL.
- CREATE command is used to create databases and tables.

- The DESCRIBE command is used to view the structure of a table.
- ALTER TABLE statement is used to change the structure of a table ie. to add, remove or change its column(s).
- The SELECT command is used to fetch data from tables.
- The keyword DISTINCT is used to eliminate display of duplicate data.
- The WHERE clause is used to select specific rows.
- The BETWEEN operator defines the range of values that a column values must fall into to make the condition true.
- The IN operator selects values that match any value in the given list of values.
- % and _ are two wild card characters. The percent (%) symbol is used to represent any sequence of zero or more characters. The underscore (_) symbol is used to represent a single character.
- NULL means a value that is unavailable, unassigned, unknown or inapplicable.
- The results of the SELECT command can be displayed in the ascending or descending values of a column or columns using ORDER BY clause.
- UPDATE command is used to modify existing data in a table.
- DELETE command is used to delete rows from a table.

EXERCISES

Complete the following cross-word puzzle using SQL commands and clauses:



Clues:

Across	Down
4. Remains same until you change the no. of columns in the table.	1. Show the table structure
7. Show me the strings following a specific pattern.	2. Destroy the table
9. It contains data	3. Display the table data
10. To add another column to the table	5. I don't want data in this table
	6. It contains tables
	8. Populate the table

MULTIPLE CHOICE QUESTIONS

1. MySQL is a /an

   a. Relational Database b. Database

   c. RDBMS d. Table

2. USE <databasename> command

   a. is not used in MySQL b. is given before quitting MySQL

   c. opens a table d. opens a database

3. Number of columns in a table is called

   a. Power b. Degree

   c. Cardinality d. Design

4. A database

   a. Contains tables b. Is a part of a table

   c. Is same as a table d. Removes data from a table

5. Number of rows in a table is called

   a. Power
   b. Degree
   c. Cardinality
   d. Design

6. ORDER BY clause is used to sort data

   a. in ascending order
   b. in desceding order
   c. both (a) and (b)
   d. None of the above

7. A table can have

   a. Only one candidate key
   b. Only one primary key
   c. Only one alternate key
   d. No alternate key

8. Wild card characters are used

   a. In LIKE clause
   b. In ORDER BY clause
   c. In BETWEEN clause
   d. In IN clause

9. DDL is

   a. a part of SQL
   b. a part of DML
   c. a part of DCL
   d. None of the above

10. LIKE clause is used

    a. For pattern matching
    b. For table matching
    c. For inserting similar data in a table
    d. For deleting data from a table

LAB EXERCISES

Consider a database LOANS with the following table:

Table: Loan_Accounts

AccNo	Cust_Name	Loan_Amount	Instalments	Int_Rate	Start_Date	Interest
1	R.K. Gupta	300000	36	12.00	19-07-2009	
2	S.P. Sharma	500000	48	10.00	22-03-2008	
3	K.P. Jain	300000	36	NULL	08-03-2007	
4	M.P. Yadav	800000	60	10.00	06-12-2008	
5	S.P. Sinha	200000	36	12.50	03-01-2010	
6	P. Sharma	700000	60	12.50	05-06-2008	
7	K.S. Dhall	500000	48	NULL	05-03-2008	

Write SQL commands for the tasks 1 to 35 and write the output for the SQL commands 36 to 40:

Create Database and use it

1. Create the database LOANS.
2. Use the database LOANS.

Create Table / Insert Into

3. Create the table Loan_Accounts and insert tuples in it.

Simple Select

4. Display the details of all the loans.
5. Display the AccNo, Cust_Name, and Loan_Amount of all the loans.

Conditional Select using Where Clause

6. Display the details of all the loans with less than 40 instalments.
7. Display the AccNo and Loan_Amount of all the loans started before 01-04-2009.
8. Display the Int_Rate of all the loans started after 01-04-2009.

Using NULL

9. Display the details of all the loans whose rate of interest is NULL.
10. Display the details of all the loans whose rate of interest is not NULL.

Using DISTINCT Clause

11. Display the amounts of various loans from the table Loan_Accounts. A loan amount should appear only once.
12. Display the number of instalments of various loans from the table Loan_Accounts. An instalment should appear only once..

Using Logical Operators (NOT, AND, OR)

13. Display the details of all the loans started after 31-12-2008 for which the number of instalments are more than 36.
14. Display the Cust_Name and Loan_Amount for all the loans which do not have number of instalments 36.
15. Display the Cust_Name and Loan_Amount for all the loans for which the loan amount is less than 500000 or int_rate is more than 12.
16. Display the details of all the loans which started in the year 2009.
17. Display the details of all the loans whose Loan_Amount is in the range 400000 to 500000.
18. Display the details of all the loans whose rate of interest is in the range 11% to 12%.

Using IN Operator

19. Display the Cust_Name and Loan_Amount for all the loans for which the number of instalments are 24, 36, or 48. (Using IN operator)

Using BETWEEN Operator

20. Display the details of all the loans whose Loan_Amount is in the range 400000 to 500000. (Using BETWEEN operator)
21. Display the details of all the loans whose rate of interest is in the range 11% to 12%. (Using BETWEEN operator)

Using LIKE Operator

22. Display the AccNo, Cust_Name, and Loan_Amount for all the loans for which the Cust_Name ends with 'Sharma'.
23. Display the AccNo, Cust_Name, and Loan_Amount for all the loans for which the Cust_Name ends with 'a'.
24. Display the AccNo, Cust_Name, and Loan_Amount for all the loans for which the Cust_Name contains 'a'
25. Display the AccNo, Cust_Name, and Loan_Amount for all the loans for which the Cust_Name does not contain 'P'.
26. Display the AccNo, Cust_Name, and Loan_Amount for all the loans for which the Cust_Name contains 'a' as the second last character.

Using ORDER BY clause

27. Display the details of all the loans in the ascending order of their Loan_Amount.
28. Display the details of all the loans in the descending order of their Start_Date.
29. Display the details of all the loans in the ascending order of their Loan_Amount and within Loan_Amount in the descending order of their Start_Date.

Using UPDATE, DELETE, ALTER TABLE

30. Put the interest rate 11.50% for all the loans for which interest rate is NULL.
31. Increase the interest rate by 0.5% for all the loans for which the loan amount is more than 400000.
32. For each loan replace Interest with (Loan_Amount*Int_Rate*Instalments) 12*100.
33. Delete the records of all the loans whose start date is before 2007.
34. Delete the records of all the loans of 'K.P. Jain'
35. Add another column Category of type CHAR(1) in the Loan table.

Find the Output of the following queries

36. SELECT cust_name, LENGTH(Cust_Name), LCASE(Cust_Name), UCASE(Cust_Name) FROM Loan_Accounts WHERE Int_Rate < 11.00;
37. SELECT LEFT(Cust_Name, 3), Right(Cust_Name, 3), SUBSTR(Cust_Name, 1, 3) FROM Loan_Accounts WHERE Int_Rate > 10.00;
38. SELECT RIGHT(Cust_Name, 3), SUBSTR(Cust_Name, 5) FROM Loan_Accounts;
39. SELECT DAYNAME(Start_Date) FROM Loan_Accounts;
40. SELECT ROUND(Int_Rate*110/100, 2) FROM Loan_Account WHERE Int_Rate > 10;

Write the output produced by the following SQL commands:

41. SELECT POW(4,3), POW(3,4);
42. SELECT ROUND(543.5694,2), ROUND(543.5694), ROUND(543.5694,-1);
43. SELECT TRUNCATE(543.5694,2), TRUNCATE(543.5694,-1);
44. SELECT LENGTH("Prof. M. L. Sharma");
45. SELECT CONCAT("SHEIKH", " HAROON") "FULL NAME";
46. SELECT YEAR(CURDATE()), MONTH(CURDATE()), DAY(CURDATE());
47. SELECT DAYOFYEAR(CURDATE()), DAYOFMONTH(CURDATE()), DAYNAME(CURDATE());
48. SELECT LEFT("Unicode",3), RIGHT("Unicode",4);
49. SELECT INSTR("UNICODE","CO"), INSTR("UNICODE","CD");
50. SELECT MID("Informatics",3,4), SUBSTR("Practices",3);