Dr.Babasaheb Ambedkar Technological university,Lonere

Samarth Educational Trust's

Arvind Gavali College of Enginering, Satara

Lab Manual

Business Intelligence

Experiment No. 1

Title: Installation of SQL server and Power BI.

Aim: To study successfully installation of SQL server and Power BI.

Theory:

Steps for installing SQL server

To install SQL Server, you need to download it from the Microsoft.com website. Once the download completes, you double-click the file **SQLServer2017-SSEI-Dev.exe** to launch the installer.

1. The installer asks you to select the installation type, choose the Custom installation type allows you to step through the SQL Server installation wizard and select the features that you want to install.

2. Specify the folder for storing the installation files that the installer will download, then click the Install button.

3. The installer starts downloading the install package for a while.

4. Once the download completes, open the folder that stores the install package and double-click the SETUP.exe file.

5. The following window displays; select the installation option on the left.

6. Click the first link to launch a wizard to install SQL Server 2017.

7. Specify the edition that you want to install, select Developer edition and click the Next button.

8. Select the “I accept the license terms.” and click the Next button.

9. Check the “Use Microsoft Update to check for updates (recommended)” to get the security and other important updates for the SQL Server and click the Next button.

10. The installation checks for the prerequisites before installation. If no error found, click the Next button.

11. Select the features that you want to install. For now, you just need the Database Engine Services, just check the checkbox and click the Next button to continue.

12. Specify the name and install ID for the instance of the SQL Server and click the Next button.

13. Specify the service account and collation configuration. Just use the default configuration and click the Next button.

14. Specify the database engine security mode. First, choose Mixed Mode. Next, enter the password for the SQL Server system administrator (sa) account. Then, re-enter the same password to confirm it. After that, click the Add Current User button. Finally, click the Next button.

15. Verify the SQL Server 2017 features to be installed:

16. The installer starts the installation process.

17. Once it completes, the following window displays. Click the OK button.

18. Click the Close button to complete the installation.

Steps for installing Power BI

1. Download setup file for Microsoft Power metallic element mistreatment this URL – https://powerbi.microsoft.com/en-us/desktop/
2. To install it, open the setup file and click on “Next”.

3. Accept the license agreement.
4. Choose the destination folder.
5. Click “Next” and check the checkbox if you would like to envision a desktop route for Power metallic element desktop.
6. It will show the installation progress.
7. After prospering installation, it’ll raise to launch Power metallic element. Power metallic element Desktop is launched with success. We successfully install Power BI in your Windows.

Conclusion:

Experiment No. 2

Title: Import the legacy data from different sources such as (Excel, SqlServer, Oracle etc.) and load in the target system.

Aim: To study how to Import the legacy data from different sources such as (Excel, SqlServer, Oracle etc.) and load in the target system.

Theory:

Importing Excel Data

1) Launch Power BI Desktop.

2) From the Home ribbon, select Get Data. Excel is one of the Most Common data connections, so you can select it directly from the Get Data menu.

3) If you select the Get Data button directly, you can also select FIle > Excel and select Connect.

4) In the Open File dialog box, select the Products.xlsx file.

5) In the Navigator pane, select the Products table and then select Edit.

Importing Data from OData Feed

In this task, you'll bring in order data. This step represents connecting to a sales system. You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below:
http://services.odata.org/V3/Northwind/Northwind.svc/

Connect to an OData feed:

1) From the Home ribbon tab in Query Editor, select Get Data.

2) Browse to the OData Feed data source.

3) In the OData Feed dialog box, paste the URL for the Northwind OData feed.

4) Select OK.

5) In the Navigator pane, select the Orders table, and then select Edit.

Conclusion:

Experiment No. 3

Title: Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sqlserver / Power BI.

Aim: To study the Extraction Transformation and Loading (ETL) process to construct the database in the Sqlserver / Power BI.

Theory:
Step 1 : Data Extraction :
The data extraction is first step of ETL. There are 2 Types of Data Extraction
1. Full Extraction : All the data from source systems or operational systems gets extracted to staging area. (Initial Load)
2. Partial Extraction : Sometimes we get notification from the source system to update specific date. It is called as Delta load.
Source System Performance: The Extraction strategies should not affect source system performance.
Step 2 : Data Transformation :
The data transformation is second step.After extracting the data there is big need to do the transformation as per the target system.I would like to give you some bullet points of Data Transformation.
• Data Extracted from source system is in to Raw format. We need to transform it before loading in to target server.
• Data has to be cleaned, mapped and transformed
• There are following important steps of Data Transformation :
  1.Selection : Select data to load in target
  2.Matching : Match the data with target system
  3.Data Transforming : We need to change data as per target table structures
Real life examples of Data Transformation :
• Standardizing data : Data is fetched from multiple sources so it needs to be standardized as per the target system.
• Character set conversion : Need to transform the character sets as per the target systems. (Firstname and last name example)
• Calculated and derived values: In source system there is first val and second val and in target we need the calculation of first val and second val.
• Data Conversion in different formats : If in source system date in in DDMMYY format and in target the date is in DDMONYYYY format then this transformation needs to be done at transformation phase.
Step 3 : Data Loading
• Data loading phase loads the prepared data from staging tables to main tables.

ETL process in SQL Server:
Following are the steps to open BIDS\SSDT.
Step 1 − Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group. The following screen appears.

Step 2 − The above screen shows SSDT has opened. Go to file at the top left corner in the above image and click New. Select project and the following screen opens.
Step 3 − Select Integration Services under Business Intelligence on the top left corner in the above screen to get the following screen.
Step 4 − In the above screen, select either Integration Services Project or Integration Services Import Project Wizard based on your requirement to develop\create the package.
Modes
There are two modes − Native Mode (SQL Server Mode) and Share Point Mode.
Models
There are two models − Tabular Model (For Team and Personal Analysis) and Multi Dimensions Model (For Corporate Analysis).

The BIDS (Business Intelligence Studio till 2008 R2) and SSDT (SQL Server Data Tools from 2012) are environments to work with SSAS.
Step 1 − Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group. The following screen will appear.
Step 2 − The above screen shows SSDT has opened. Go to file on the top left corner in the above image and click New. Select project and the following screen opens.
Step 3 − Select Analysis Services in the above screen under Business Intelligence as seen on the top left corner. The following screen pops up.
Step 4 − In the above screen, select any one option from the listed five options based on your requirement to work with Analysis services.

ETL Process in Power BI

1) Remove other columns to only display columns of interest

In this step you remove all columns except **ProductID**, **ProductName**, **UnitsInStock**, and **QuantityPerUnit**
Power BI Desktop includes Query Editor, which is where you shape and transform your data connections. Query Editor opens automatically when you select **Edit** from Navigator. You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

1. In **Query Editor**, select the **ProductID, ProductName**, **QuantityPerUnit,** and **UnitsInStock** columns (use **Ctrl+Click** to select more than one column, or **Shift+Click** to select columns that are beside each other).

2. Select **Remove Columns > Remove** Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.

3. Change the data type of the UnitsInStock column
When Query Editor connects to data, it reviews each field and to determine the best data type. For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the **UnitsInStock** column’s datatype is Whole Number.
1. Select the **UnitsInStock** column.
2. Select the **Data Type drop-down button** in the **Home ribbon**.

3. If not already a Whole Number, select **Whole Number** for data type from the drop down (the Data Type: button also displays the data type for the current selection).

3. Expand the Order_Details table

The Orders table contains a reference to a Details table, which contains the individual products that were included in each Order. When you connect to data sources with multiples tables (such as a relational database) you can use these references to build up your query.

In this step, you expand the **Order_Details** table that is related to the Orders table, to combine the **ProductID**, **UnitPrice**, and **Quantity** columns from **Order_Details** into the **Orders table**. This is a representation of the data in these tables:

The Expand operation combines columns from a related table into a subject table. When the query runs, rows from the related table (**Order_Details**) are combined into rows from the subject table (**Orders**).

After you expand the Order_Details table, three new columns and additional rows are added to the Orders table, one for each row in the nested or related table.

1. In the Query View, scroll to the Order_Details column.
2. In the Order_Details column, select the expand icon ( ).
3. In the Expand drop-down:

a. Select (Select All Columns) to clear all columns.
b. Select ProductID, UnitPrice, and Quantity.
c. Click OK.

4. Calculate the line total for each Order_Details row

Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order_Details row.

Calculate the line total for each Order_Details row:

1. In the Add Column ribbon tab, click Add Custom Column.

2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order_Details.UnitPrice] * [Order_Details.Quantity].

3. In the New column name textbox, enter LineTotal.

4. Click OK.

5. Rename and reorder columns in the query

In this step you finish making the model easy to work with when creating reports, by renaming the final columns and changing their order.

1. In Query Editor, drag the LineTotal column to the left, after ShipCountry.

2.Remove the Order_Details. prefix from the Order_Details.ProductID, Order_Details.UnitPrice and Order_Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.

6. Combine the Products and Total Sales queries

Power BI Desktop does not require you to combine queries to report on them. Instead, you can create Relationships between datasets. These relationships can be created on any column that is common to your datasets
we have Orders and Products data that share a common 'ProductID' field, so we need to ensure there's a relationship between them in the model we're using with Power BI Desktop. Simply specify in Power BI Desktop that the columns from each table are related (i.e. columns that have the same values). Power BI Desktop works out the direction and cardinality of the relationship for you. In some cases, it will even detect the relationships automatically.
In this task, you confirm that a relationship is established in Power BI Desktop between the Products and Total Sales queries
Step 1: Confirm the relationship between Products and Total Sales

1. First, we need to load the model that we created in Query Editor into Power BI Desktop. From the Home ribbon of Query Editor, select Close & Load.

2. Power BI Desktop loads the data from the two queries.

3. Once the data is loaded, select the Manage Relationships button Home ribbon.

4. Select the New… button

5. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.

5. Select Cancel, and then select Relationship view in Power BI Desktop.

6. We see the following, which visualizes the relationship between the queries.

7. When you double-click the arrow on the line that connects the to queries, an Edit Relationship dialog appears.

8. No need to make any changes, so we'll just select Cancel to close the Edit Relationship dialog.

Conclusion:

Experiment No. 4

Title: Create the cube with suitable dimension and fact tables based on ROLAP, MOLAP and HOLAP model.

Aim: To study how to create the cube with suitable dimension and fact tables based on ROLAP, MOLAP and HOLAP model.

Theory:
Creating Data Warehouse
Let us execute our T-SQL Script to create data warehouse with fact tables, dimensions and populate them with appropriate test values.
Download T-SQL script attached with this article for creation of Sales Data Warehouse or download from this article "**Create First Data Warehouse**" and run it in your SQL Server.
Follow the given steps to run the query in **SSMS** (SQL Server Management Studio).

1. Open SQL Server Management Studio 2008

2. Connect Database Engine

3. Open **New Query** editor

4. Copy paste Scripts given below in various steps in new query editor window one by one

5. To run the given SQL Script, press F5

6. It will create and populate "Sales_DW" database on your SQL Server

Developing an OLAP Cube
For creation of OLAP Cube in Microsoft BIDS Environment, follow the 10 easy steps given below.
Step 1: Start BIDS Environment
Click on **Start Menu** -> Microsoft **SQL Server 2008 R2** -> Click **SQL Server Business Intelligence Development Studio.**
Step 2: Start Analysis Services Project
Click **File** -> **New** -> **Project** ->**Business Intelligence Projects** ->select **Analysis Services Project**-> Assign Project **Name** -> Click **OK**
Step 3: Creating New Data Source
3.1 In Solution Explorer, Right click on **Data Source** -> Click **New Data Source**
Step 3: Creating New Data Source
3.1 In Solution Explorer, Right click on **Data Source** -> Click **New Data Source**
3.2 Click on Next
3.3 Click on New **Button**
3.4 Creating **New connection**

1. Specify Your SQL **Server Name** where your Data Warehouse was created
2. Select Radio Button according to your **SQL Server Authentication** mode

3. Specify your **Credentials** using which you can connect to your SQL Server
4. Select database Sales_DW.
5. Click on **Test Connection** and verify for its success
6. Click OK.
3.5 Select Connection created in **Data Connections**-> Click **Next**
3.6 Select Option **Inherit**
3.7 Assign Data Source **Name** -> Click **Finish**
Step 4: Creating New Data Source View
4.1 In the Solution Explorer, Right Click on **Data Source View** -> Click on **New Data Source View**
4.2 Click **Next**
4.3 Select **Relational Data Source** we have created previously (Sales_DW)-> Click **Next**
4.4 First move your **Fact Table** to the right side to include in object list.
Select FactProductSales Table -> Click on Arrow Button to move the selected object to Right Pane.
4.5 Now to **add dimensions** which are **related** to your **Fact Table**, follow the given steps:
Select **Fact Table** in Right Pane (Fact product Sales) -> Click On **Add Related Tables**
4.6 It will add all associated dimensions to your Fact table as per relationship specified in your SQL DW (Sales_DW).
Click Next.
4.7 Assign **Name (SalesDW DSV)->** Click **Finish**
4.8 Now Data Source View is ready to use.
Step 5: Creating New Cube
5.1 In Solution Explorer -> Right Click on **Cube->** Click **New Cube**
5.2 Click **Next**
5.3 Select Option **Use existing Tables ->** Click **Next**
5.4 Select Fact Table Name from **Measure Group Tables (FactProductSales) ->** Click **Next**
5.5 Choose **Measures** from the List which you want to place in your Cube --> Click **Next**
5.6 Select All **Dimensions** here which are associated with your Fact Table-> Click **Next**
5.7 Assign **Cube Name** (SalesAnalyticalCube) -> Click **Finish**
5.8 Now your Cube is ready, you can see the newly created cube and dimensions added in your solution explorer.
Step 6: Dimension Modification
In Solution Explorer, double click on dimension **Dim Product ->** Drag and Drop Product Name from Table in Data Source View and Add in Attribute Pane at left side.
Step 7: Creating Attribute Hierarchy In Date Dimension
Double click On **Dim Date** dimension -> Drag and Drop Fields from Table shown in Data Source View to Attributes-> Drag and Drop attributes from leftmost pane of attributes to middle pane of Hierarchy.
Drag fields in sequence from Attributes to Hierarchy window (Year, Quarter Name, Month Name, Week of the Month, Full Date UK),
Step 8: Deploy the Cube
8.1 In Solution Explorer, right click on Project Name (SalesDataAnalysis) -- > Click **Properties**
8.2 Set **Deployment Properties** First
In Configuration Properties, Select Deployment-> Assign Your SQL Server Instance Name Where Analysis Services Is Installed (*mubin-pc\fairy*) (*Machine Name\Instance Name*) ->

Choose Deployment Mode **Deploy All** as of now ->Select Processing Option **Do Not Process** -> Click **OK**
8.3 In Solution Explorer, right click on **Project Name** (SalesDataAnalysis) -- > Click **Deploy**
8.4 Once Deployment will finish, you can see the message **Deployment Completed** in deployment Properties.
31
Step 9: Process the Cube
9.1 In Solution Explorer, right click on Project Name (SalesDataAnalysis) -- > Click **Process**
9.2 Click on **Run** button to process the Cube
9.3 Once processing is complete, you can see **Status** as **Process Succeeded** -->Click **Close** to close both the open windows for processing one after the other.
Step 10: Browse the Cube for Analysis
10.1 In Solution Explorer, right click on Cube Name (SalesDataAnalysisCube) -- > Click **Browse**
10.2 Drag and drop measures in to Detail fields, & Drag and Drop Dimension Attributes in Row Field or Column fields.
Now to **Browse Our Cube**

1. Product Name Drag & Drop into Column

2. Full Date UK Drag & Drop into Row Field

3. FactProductSalesCount Drop this measure in Detail area

Conclusion:

Experiment No. 5

Title: Execute the MDX queries to extract the data from the data warehouse.

Aim: To study the MDX queries to extract the data from the data warehouse.

Theory:

Step 1: Open SQL Server Management Studio and connect to Analysis Services.
Server type: Analysis Services
Server Name: (according to base machine)
Click on connect
Step 2: Click on New Query & type following query based on Sales_DW
select [Measures].[Sales Time Alt Key] on columns from [Sales DW]
Click on execute
select [Measures].[Quantity] on columns from [Sales DW]
select [Measures].[Sales Invoice Number] on columns from [Sales DW]
select [Measures].[Sales Total Cost] on columns from [Sales DW]
select [Measures].[Sales Total Cost] on columns , [Dim Date].[Year].[Year] on rows from [Sales DW]
select [Measures].[Sales Total Cost] on columns , NONEMPTY({[Dim Date].[Year].[Year]}) on rows from [Sales DW]
select [Measures].[Sales Total Cost] on columns from [Sales DW] Where [Dim Date].[Year].[Year].&[2013]

Conclusion:

Experiment No. 6

Title: Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data.

Aim: To study data visualization.

Theory:

Data virtualization is an approach to data management that allows an application to retrieve and manipulate data without requiring technical details about the data, such as how it is formatted at source, or where it is physically located,[1] and can provide a single customer view (or single view of any other entity) of the overall data.[2]

Unlike the traditional extract, transform, load ("ETL") process, the data remains in place, and real-time access is given to the source system for the data. This reduces the risk of data errors, of the workload moving data around that may never be used, and it does not attempt to impose a single data model on the data (an example of heterogeneous data is a federated database system). The technology also supports the writing of transaction data updates back to the source systems.[3] To resolve differences in source and consumer formats and semantics, various abstraction and transformation techniques are used. This concept and software is a subset of data integration and is commonly used within business intelligence, service-oriented architecture data services, cloud computing, enterprise search, and master data management.

A book store and have 100 books in storage. You sell a certain % for the highest price of \$50 and a certain % for the lower price of \$20.
If you sell 60% for the highest price, cell D10 calculates a total profit of 60 * \$50 + 40 * \$20 = \$3800.
Create Different Scenarios
But what if you sell 70% for the highest price? And what if you sell 80% for the highest price? Or 90%, or even 100%? Each different percentage is a different **scenario**. You can use the Scenario Manager to create these scenarios.
Note: You can simply type in a different percentage into cell C4 to see the corresponding result of a scenario in cell D10. However, what-if analysis enables you to easily compare the results of different scenarios. Read on.
1. On the Data tab, in the Forecast group, click What-If Analysis.
2. Click Scenario Manager.
The Scenario Manager dialog box appears.
3. Add a scenario by clicking on Add.
4. Type a name (60% highest), select cell C4 (% sold for the highest price) for the Changing cells and click on OK.
5. Enter the corresponding value 0.6 and click on OK again.
6. Next, add 4 other scenarios (70%, 80%, 90% and 100%).
Finally, your Scenario Manager should be consistent with the picture below:

Conclusion:

Experiment No. 7

Title: Implementation of Classification algorithm in R Programming.

Aim: To study implementation of Classification algorithm in R Programming.

Theory:
R is a very dynamic and versatile programming language for data science. This article deals with classification in R. Generally classifiers in R are used to predict specific category related information like reviews or ratings such as good, best or worst.
Various Classifiers are:

- Decision Trees
- Naive Bayes Classifiers
- K-NN Classifiers
- Support Vector Machines(SVM's)

Decision Tree Classifier

It is basically is a graph to represent choices. The nodes or vertices in the graph represent an event and the edges of the graph represent the decision conditions. Its common use is in Machine Learning and Data Mining applications.

Applications:
Spam/Non-spam classification of email, predicting of a tumor is cancerous or not. Usually, a model is constructed with noted data also called training dataset. Then a set of validation data is used to verify and improve the model. R has packages that are used to create and visualize decision trees.
The R package "party" is used to create decision trees.

Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.

```
# Get the data points in form of a R vector.
rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
# Convert it to a time series object.
rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)
# Print the timeseries data.
print(rainfall.timeseries)
# Give the chart file a name.
png(file = "rainfall.png")
# Plot a graph of the time series.
plot(rainfall.timeseries)
# Save the file.
dev.off()
```

Output:
When we execute the above code, it produces the following result and chart −

```
Jan      Feb      Mar      Apr     May      Jun     Jul     Aug     Sep
2012     799.0    1174.8   865.1  1334.6   635.4   918.5   685.5   998.6   784.2
         Oct     Nov     Dec
2012      985.0  882.8  1071.0
```

Conclusion:

Experiment No. 8

Title: Practical Implementation of Decision Tree using R Tool.

Aim: To study implementation of Decision Tree using R tool.

Theory:

```
install.packages("party")
```

The package "party" has the function **ctree()** which is used to create and analyze decison tree.

Syntax

The basic syntax for creating a decision tree in R is −

```
ctree(formula, data)
```

Input Data

We will use the R in-built data set named **readingSkills** to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker or not.

Here is the sample data.

```
# Load the party package. It will automatically load other
# dependent packages.
library(party)
# Print some records from data set readingSkills.
print(head(readingSkills))
```

When we execute the above code, it produces the following result and chart −

```
nativeSpeaker age shoeSize score
1 yes 5 24.83189 32.29385
2 yes 6 25.95238 36.63105
3 no 11 30.42170 49.60593
4 yes 7 28.66450 40.28456
5 yes 11 31.88207 55.46085
6 yes 10 30.07843 52.83124
Loading required package: methods
Loading required package: grid
..............................
..............................
```

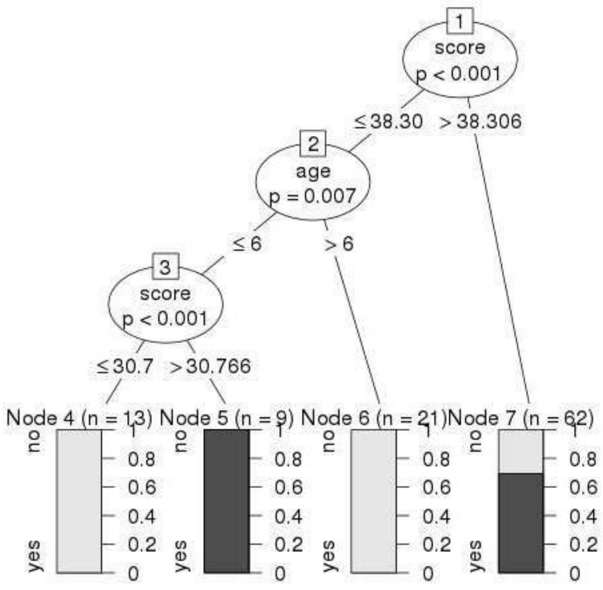
We will use the **ctree()** function to create the decision tree and see its graph.

```
# Load the party package. It will automatically load other
# dependent packages.
library(party)
# Create the input data frame.
input.dat <- readingSkills[c(1:105),]
# Give the chart file a name.
png(file = "decision_tree.png")
# Create the tree.
output.tree <- ctree(
nativeSpeaker ~ age + shoeSize + score,
data = input.dat)
```

```
# Plot the tree.
plot(output.tree)
# Save the file.
dev.off()
```

Output:-

```
null device
1
Loading required package: methods
Loading required package: grid
Loading required package: mvtnorm
Loading required package: modeltools
Loading required package: stats4
Loading required package: strucchange
Loading required package: zoo
Attaching package: ‘zoo’
The following objects are masked from ‘package:base’:
as.Date, as.Date.numeric
Loading required package: sandwich
```



Conclusion:

Experiment No. 9

Title: Prediction Using Linear Regression.

Aim: To study prediction using Linear Regression.

Theory:
In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.
y = ax + b is an equation for linear regression.
Where, **y** is the response variable, **x** is the predictor variable and ***a*** and ***b*** are constants which are called the coefficients.
A simple example of regression is predicting weight of a person when his height is known.
To do this we need to have the relationship between height and weight of a person.
The steps to create the relationship is −
• Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
• Create a relationship model using the **lm()** functions in R.
• Find the coefficients from the model created and create the mathematical equation using these
• Get a summary of the relationship model to know the average error in prediction. Also called **residuals**. • To predict the weight of new persons, use the **predict()** function in R.

Input Data
Below is the sample data representing the observations −
# Values of height
151, 174, 138, 186, 128, 136, 179, 163, 152, 131
# Values of weight.
63, 81, 56, 91, 47, 57, 76, 72, 62, 48
lm() Function
This function creates the relationship model between the predictor and the response variable.
Syntax
The basic syntax for **lm()** function in linear regression is −
lm(formula,data)
Following is the description of the parameters used −
• **formula** is a symbol presenting the relation between x and y.
• data is the vector on which the formula will be applied.

Create Relationship Model & get the Coefficients

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function.
relation <- lm(y~x)
print(relation)
```

When we execute the above code, it produces the following result –

```
Call:
lm(formula = y ~ x)
Coefficients:
(Intercept) x
-38.4551 0.6746
```

Get the Summary of the Relationship

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function.
relation <- lm(y~x)
print(summary(relation))
```

When we execute the above code, it produces the following result –

```
call:
lm(formula = y ~ x)
Residuals:
Min 1Q Median 3Q Max
-6.3002 -1.6629 0.0412 1.8944 3.9775
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -38.45509 8.04901 -4.778 0.00139 **
x 0.67461 0.05191 12.997 1.16e-06 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 3.253 on 8 degrees of freedom
Multiple R-squared: 0.9548, Adjusted R-squared: 0.9491
F-statistic: 168.9 on 1 and 8 DF, p-value: 1.164e-06
```

predict() Function

Syntax

The basic syntax for predict() in linear regression is −

```
predict(object, newdata)
```

Following is the description of the parameters used −

- **object** is the formula which is already created using the lm() function.
- newdata is the vector containing the new value for predictor variable.

Predict the weight of new persons

```
# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
# The resposne vector.
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function.
relation <- lm(y~x)
# Find weight of a person with height 170.
a <- data.frame(x = 170)
result <- predict(relation,a)
print(result)
```

Result:

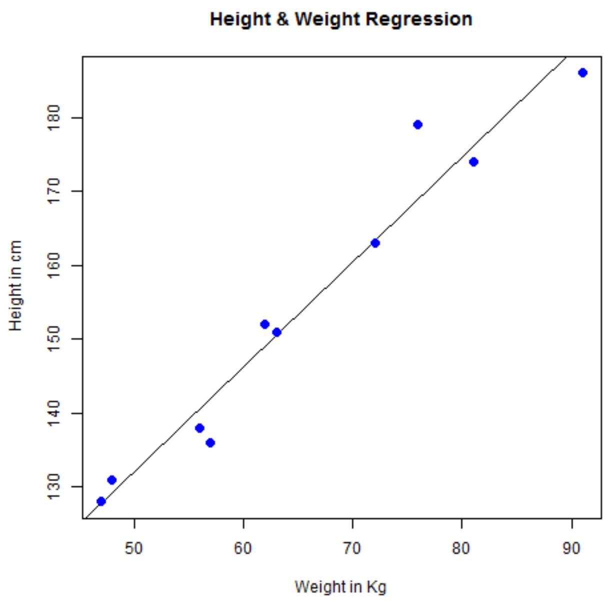
```
1
```

76.22869

Visualize the Regression Graphically

```
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)
# Give the chart file a name.
png(file = "linearregression.png")
# Plot the chart.
plot(y,x,col = "blue",main = "Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")
# Save the file.
dev.off()
```

output:



Conclusion:

Experiment No. 10

Title: Data Analysis using Time Series Analysis.

Aim: To study data analysis using Time Series Analysis.

Theory:

Time series is a series of data points in which each data point is associated with a timestamp. A simple example is the price of a stock in the stock market at different points of time on a given day. Another example is the amount of rainfall in a region at different months of the year. R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called **time-series object**. It is also a R data object like a vector or data frame.

The time series object is created by using the **ts()** function.

Syntax

The basic syntax for **ts()** function in time series analysis is −

```
timeseries.object.name <- ts(data, start, end, frequency)
```

Following is the description of the parameters used −

- data is a vector or matrix containing the values used in the time series.
- **start** specifies the start time for the first observation in time series.
- end specifies the end time for the last observation in time series.
- **frequency** specifies the number of observations per unit time.

Except the parameter "data" all other parameters are optional.

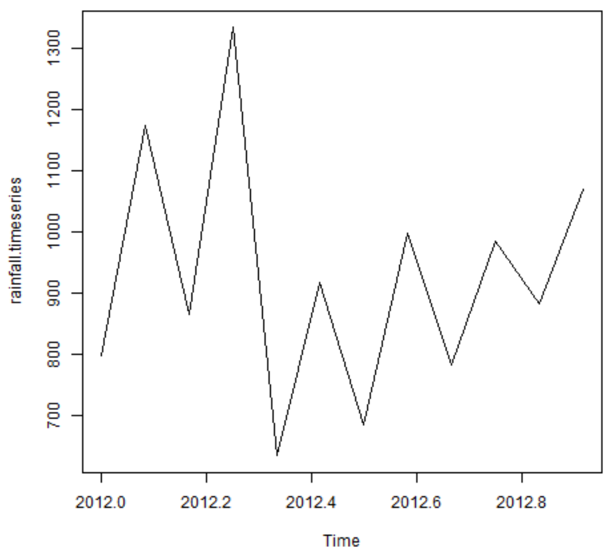
Example

Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.

```
# Get the data points in form of a R vector.
rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
# Convert it to a time series object.
rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)
# Print the timeseries data.
print(rainfall.timeseries)
# Give the chart file a name.
png(file = "rainfall.png")
# Plot a graph of the time series.
plot(rainfall.timeseries)
# Save the file.
dev.off()
```

Output:-

```
Jan Feb Mar Apr May Jun Jul Aug Sep
2012 799.0 1174.8 865.1 1334.6 635.4 918.5 685.5 998.6 784.2
Oct Nov Dec
2012 985.0 882.8 1071.0
```



Conclusion: