



Nashik Gramin Shikshan Prasarak Mandal's

**Brahma Valley College of Engineering
& Research Institute**

Recognized by AICTE, New Delhi • Govt. of Maharashtra • DTE & Affiliated to Pune University

Department of Computer Engineering

LABORATORY MANUAL

ACY-2024-25
Sem-II

LABORATORY PRACTICE-VI BE-COMPUTER ENGINEERING

SEMESTER-II

Subject Code: **410256**

INDEX

Sr. No.	Name of Assignment	Page No.	Date	Remark
410253(C): Business Intelligence				
Any 5 Assignments and 1 Mini Project are Mandatory				
Group 1				
1	Import the legacy data from different sources such as (Excel , Sql Server, Oracle etc.) and load in the target system. (You can download sample database such as Adventure works, Northwind, foodmart etc.)			
2	Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server.			
3	Create the cube with suitable dimension and fact tables based on ROLAP, MOLAP and HOLAP model.			
4	Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.			
5	Perform the data classification using classification algorithm. Or Perform the data clustering using clustering algorithm.			
Group 2				
6	Mini Project: Each group of 4 Students (max) assigned one case study for this; A BI report must be prepared outlining the following steps: a) Problem definition, identifying which data mining task is needed. b) Identify and use a standard data mining dataset available for the problem.			

410252(C) : Software Defined Networks

Any 3 Assignments and 1 Mini Project are Mandatory

Group 1

1.	Prepare setup for Mininet network emulation environment with the help of Virtual box and Mininet. Demonstrate the basic commands in Mininet and emulate different custom network topology (Simple, Linear, and Tree).View flow tables.			
2.	After studying open source POX and Floodlight controller, Install controller and run custom topology using remote controller like POX and floodlight controller. Recognize inserted flows by controllers.			
3.	Create a SDN environment on Mininet and configure a switch to provide a firewall functionality using POX controller. Ref: https://github.com/mininet/openflow-tutorial/wiki/Create-Firewall			
4.	Using Mininet as an Emulator and POX controller, build your own internet router. Write simple router with a static routing table. The router will receive raw Ethernet frames and process the packet forwarding them to correct outgoing interface. You must check the Ethernet frames are received and the forwarding logic is created so packets go to the correct interface. Ref: https://github.com/mininet/mininet/wiki/Simple Router			
5.	Emulate and manage a Data Center via a Cloud Network Controller: create a multi-rooted tree-like (Clos) topology in Mininet to emulate a data center. Implement specific SDN applications on top of the network controller in order to orchestrate multiple network tenants within a data center environment, in the context of network virtualization and management. Ref: https://opencourses.uoc.gr/courses/pluginfile.php/13576/mod_resource/content/2/exercise 5.pdf			
6.	Study Experiment: Study in details Cloud seeds automates IAAS using SDN and a high-performance network from Juniper SDN Framework.			

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work is based on overall performance and Laboratory assignments performance of student. Each Laboratory assignment assessment will assign grade/marks based on parameters with appropriate weightage. Suggested parameters for overall assessment as well as each Laboratory assignment assessment include- timely completion, performance, innovation, efficient codes, punctuality and neatness

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. The instructor may set multiple sets of assignments and distribute among batches of students. It is appreciated if the assignments are based on real world problems/applications. Use of open source software is encouraged. In addition to these, instructor may assign one real life application in the form of a mini-project based on the concepts learned. Instructor may also set one assignment or mini-project that is suitable to respective branch beyond the scope of syllabus.

Guidelines for Oral Examination

During oral assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement. The supplementary and relevant questions may be asked at the time of evaluation to test the student's for advanced learning, understanding of the fundamentals, effective and efficient implementation. So encouraging efforts, transparent evaluation and fair approach of the evaluator will not create any uncertainty or doubt in the minds of the students. So adhering to these principles will consummate our team efforts to the promising start of the student's academics.

GROUP 1: ASSIGNMENTS

410253: Business Intelligence

Group A

Assignment No: 1

Title of the Assignment:

Import the legacy data from different sources such as (Excel , Sql Server, Oracle etc.) and load in the target system. (You can download sample database such as Adventure works, Northwind, foodmart etc.).

Objective of the Assignment:

To introduce the concepts and components of Business Intelligence (BI)

Outcome:

Apply basic principles of elective subjects to problem solving and modeling.
Use tools and techniques in the area of software development to build mini projects

Pre-requisites:

1. Basics of dataset extensions.
 2. Concept of data import
-

Contents for Theory:

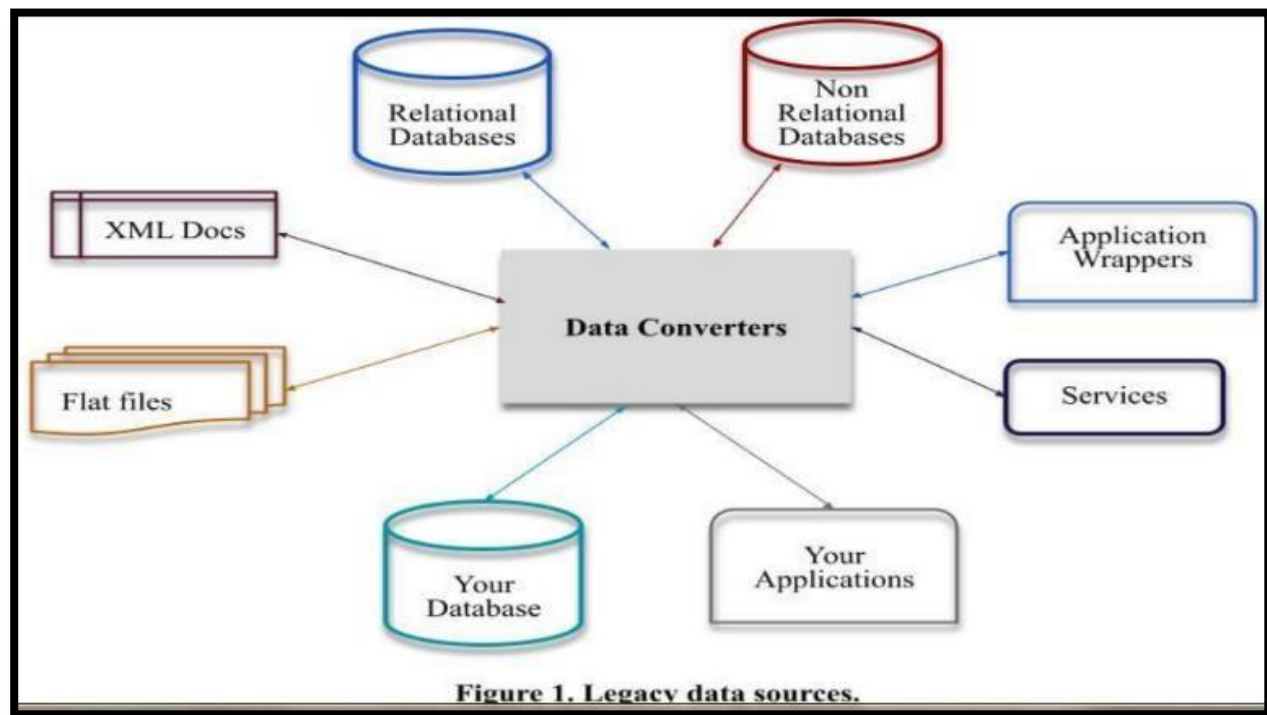
1. Legacy Data
2. Sources of Legacy Data
3. How to import legacy data step by step.

Theory:**1. What is Legacy Data?**

Legacy data, according to Business Dictionary, is "information maintained in an old or out-of-date formator computer system that is consequently challenging to access or handle."

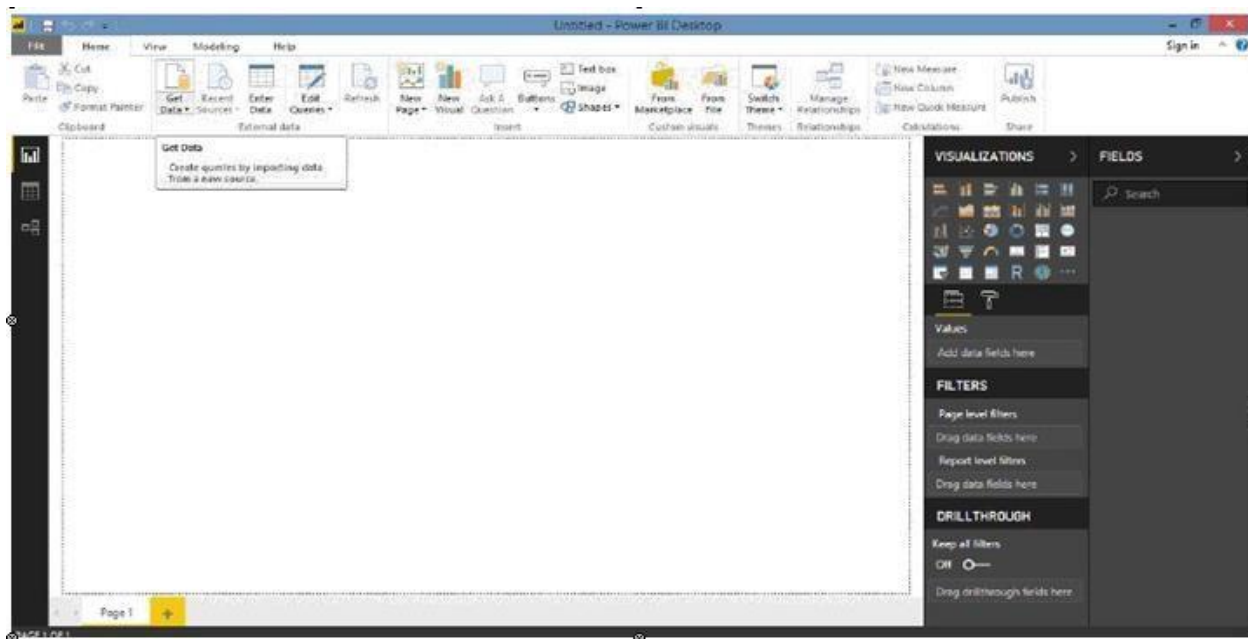
2. Sources of Legacy Data

Where does legacy data come from? Virtually everywhere. Figure 1 indicates that there are many sources from which you may obtain legacy data. This includes existing databases, often relational, although non-RDBs such as hierarchical, network, object, XML, object/relational databases, and NoSQL databases. Files, such as XML documents or "flat files" such as configuration files and comma-delimited text files, are also common sources of legacy data. Software, including legacy applications that have been wrapped (perhaps via CORBA) and legacy services such as web services or CICS transactions, can also provide access to existing information. The point to be made is that there is often far more to gaining access to legacy data than simply writing an SQL query against an existing relational database.

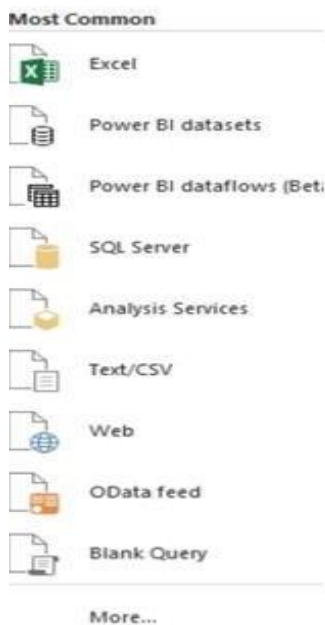


How to import legacy data step by step:

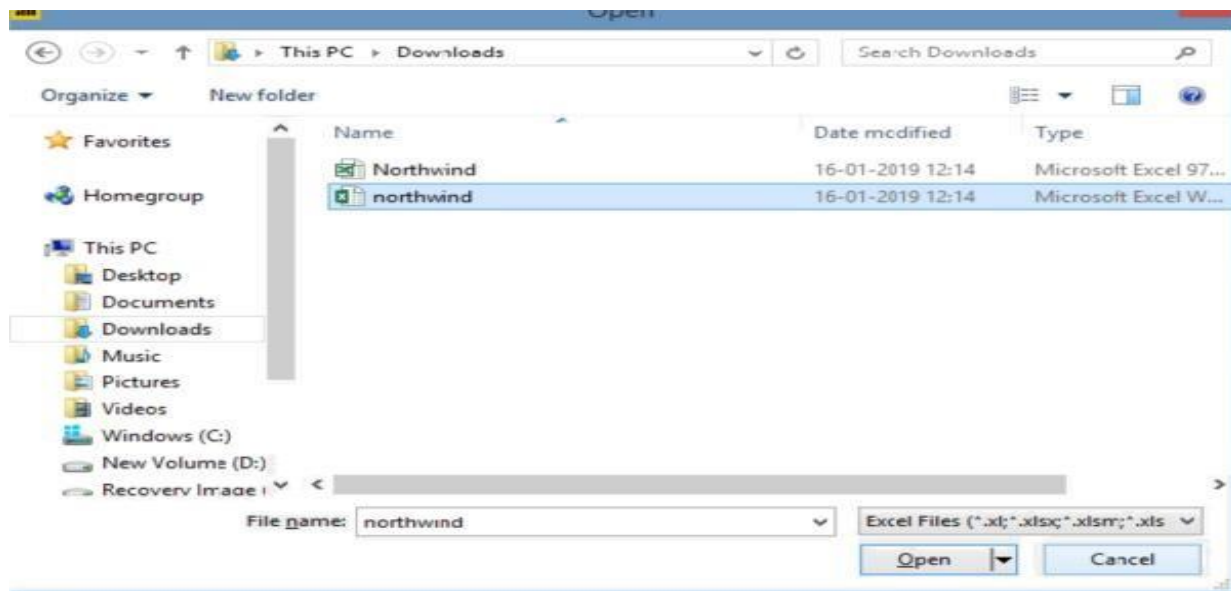
Step 1: Open Power BI



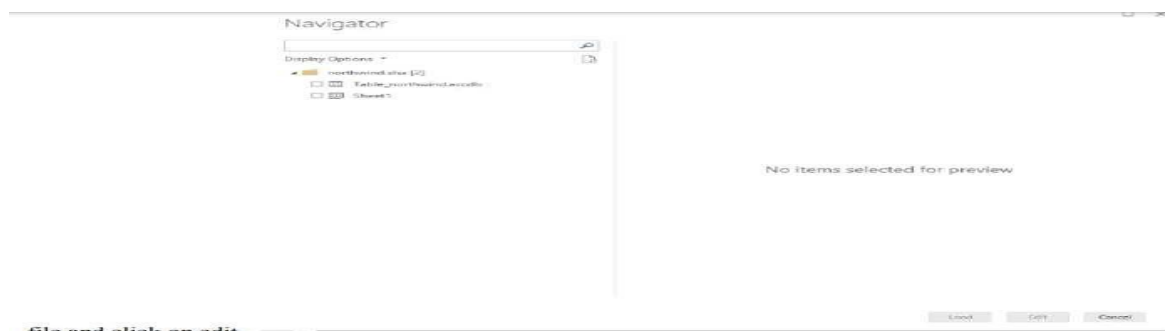
Step 2: Click on Get data following list will be displayed → select Excel



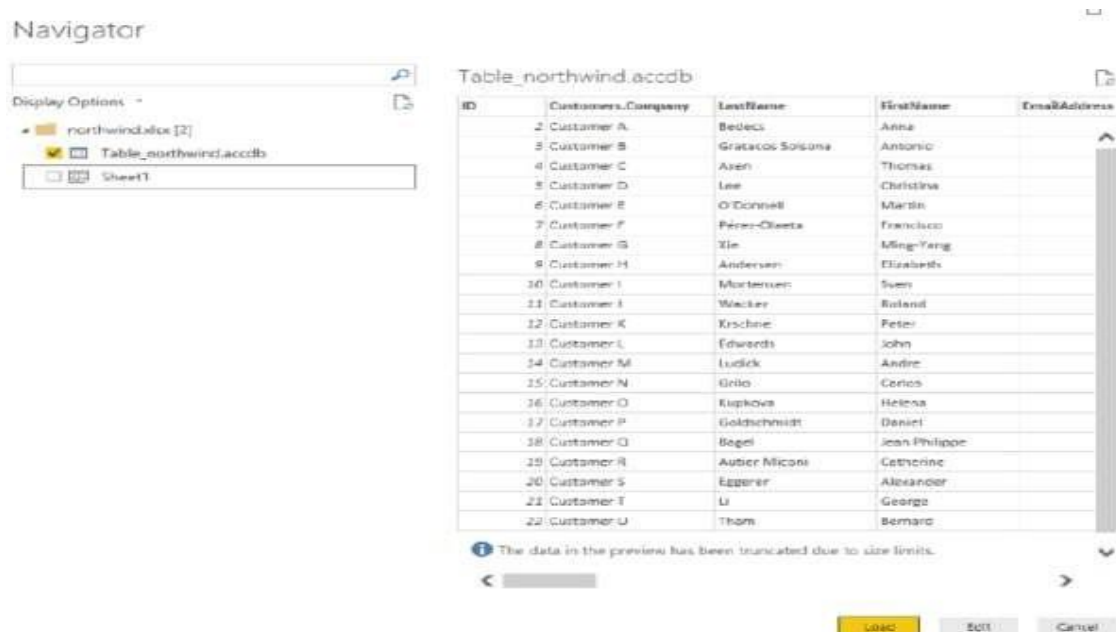
Step 3: Select required file and click on Open, Navigator screen appears

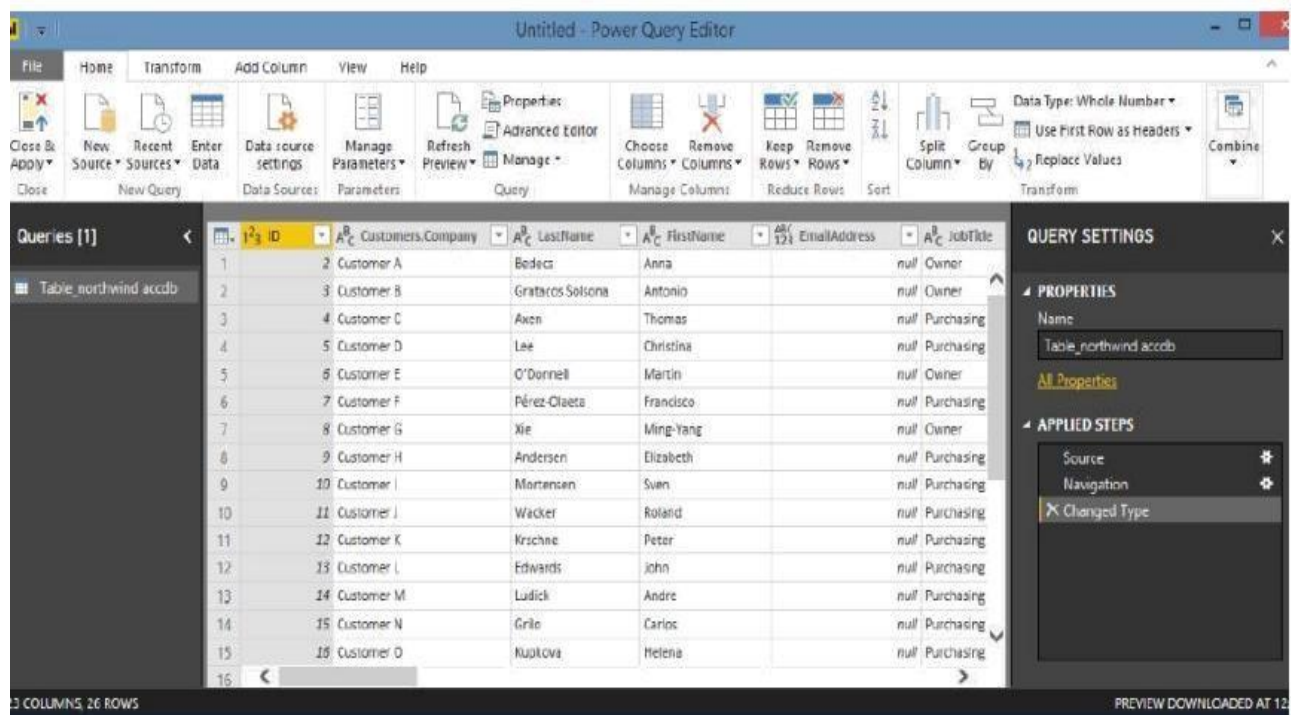
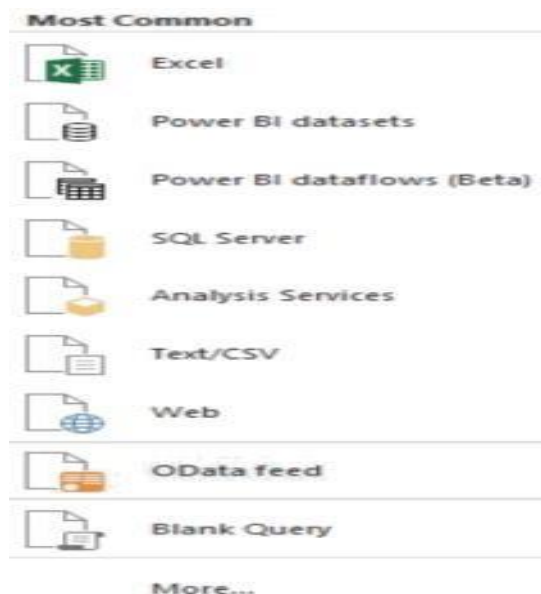


Step 4: Select file and click on edit



file and click on edit



Step 5: Power query editor appears**Step 6: Again, go to Get Data and select OData feed**

Step 7: Paste URL as <http://services.odata.org/V3/Northwind/Northwind.svc/> Click on ok

OData feed

☒ Basic
 ☐ Advanced

URL

<http://services.odata.org/V3/Northwind/Northwind.svc/>

OK Cancel

Step 8: Select orders table And click on edit

Note: If you just want to see preview you can just click on table name without clicking on checkbox
Click on edit to view table

Navigator

Display Options ▾

☒ <http://services.odata.org/V3/Northwind/No...>

- ☐ Alphabetical_list_of_products
- ☐ Categories
- ☐ Category_Sales_for_1997
- ☐ Current_Product_Lists
- ☐ Customer_and_Suppliers_by_Cities
- ☐ CustomerDemographics
- ☐ Customers
- ☐ Employees
- ☐ Invoices
- ☐ Order_Details
- ☐ Order_Details_Extended
- ☐ Order_Subtotals
- ☒ **Orders**
- ☐ Orders_Qries
- ☐ Product_Sales_for_1997
- ☐ Products

Orders

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
10248	VINET	5	04-07-1996 00:00:00	01-08-199
10249	TOMSP	6	05-07-1996 00:00:00	16-08-199
10250	HANAR	4	08-07-1996 00:00:00	05-08-199
10251	VICTE	3	08-07-1996 00:00:00	05-08-199
10252	SUPRD	4	09-07-1996 00:00:00	06-08-199
10253	HANAR	3	10-07-1996 00:00:00	24-07-199
10254	CHOPS	5	11-07-1996 00:00:00	08-08-199
10255	RICSU	9	12-07-1996 00:00:00	09-08-199
10256	WELLI	3	15-07-1996 00:00:00	12-08-199
10257	HILAA	4	16-07-1996 00:00:00	13-08-199
10258	ERNSH	1	17-07-1996 00:00:00	14-08-199
10259	CENTC	4	18-07-1996 00:00:00	15-08-199
10260	OTTIK	4	19-07-1996 00:00:00	16-08-199
10261	QUEDE	4	19-07-1996 00:00:00	16-08-199
10262	RATTC	8	22-07-1996 00:00:00	19-08-199
10263	ERNSH	9	23-07-1996 00:00:00	20-08-199
10264	FOLKO	6	24-07-1996 00:00:00	21-08-199
10265	BLONP	2	25-07-1996 00:00:00	22-08-199
10266	WARTH	3	26-07-1996 00:00:00	06-09-199
10267	FRANK	4	29-07-1996 00:00:00	26-08-199

The screenshot displays the Power Query Editor window titled "Untitled - Power Query Editor". The ribbon includes tabs for Transform, Add Column, View, and Help. The Transform tab is active, showing various data manipulation options. The main area displays a table with the following columns: OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, and ShippedDate. The table contains 15 rows of data. On the right side, the "QUERY SETTINGS" pane is open, showing the "PROPERTIES" section with the table name "Orders" and the "APPLIED STEPS" section with a single step "Source".

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate
1	10248	VINET	5	04-07-1996 00:00:00	01-08-1996 00:00:00	16-0
2	10249	TOMSP	6	05-07-1996 00:00:00	16-08-1996 00:00:00	10-0
3	10250	HANAR	4	08-07-1996 00:00:00	05-08-1996 00:00:00	12-0
4	10251	VICTE	3	08-07-1996 00:00:00	05-08-1996 00:00:00	15-0
5	10252	SUPRD	4	09-07-1996 00:00:00	06-08-1996 00:00:00	11-0
6	10253	HANAR	3	10-07-1996 00:00:00	24-07-1996 00:00:00	16-0
7	10254	CHOPS	5	11-07-1996 00:00:00	08-08-1996 00:00:00	23-0
8	10255	RICSU	9	12-07-1996 00:00:00	09-08-1996 00:00:00	15-0
9	10256	WELLI	3	15-07-1996 00:00:00	12-08-1996 00:00:00	17-0
10	10257	HILAA	4	16-07-1996 00:00:00	13-08-1996 00:00:00	22-0
11	10258	ERINSH	1	17-07-1996 00:00:00	14-08-1996 00:00:00	23-0
12	10259	CENTC	4	18-07-1996 00:00:00	15-08-1996 00:00:00	25-0
13	10260	OTTIK	4	19-07-1996 00:00:00	16-08-1996 00:00:00	29-0
14	10261	QUEDE	4	19-07-1996 00:00:00	16-08-1996 00:00:00	30-0
15	10262	RATTC	8	22-07-1996 00:00:00	19-08-1996 00:00:00	25-0

Conclusion: In this way we import the Legacy datasets using the Power BI Tool.

Group A

Assignment No: 2

Title of the Assignment:

Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sqlserver.

Objective of the Assignment:

To introduce the concepts and components of Business Intelligence(BI)

Outcome:

Apply basic principles of elective subjects to problem solving and modeling.

Use tools and techniques in the area of software development to build mini projects.

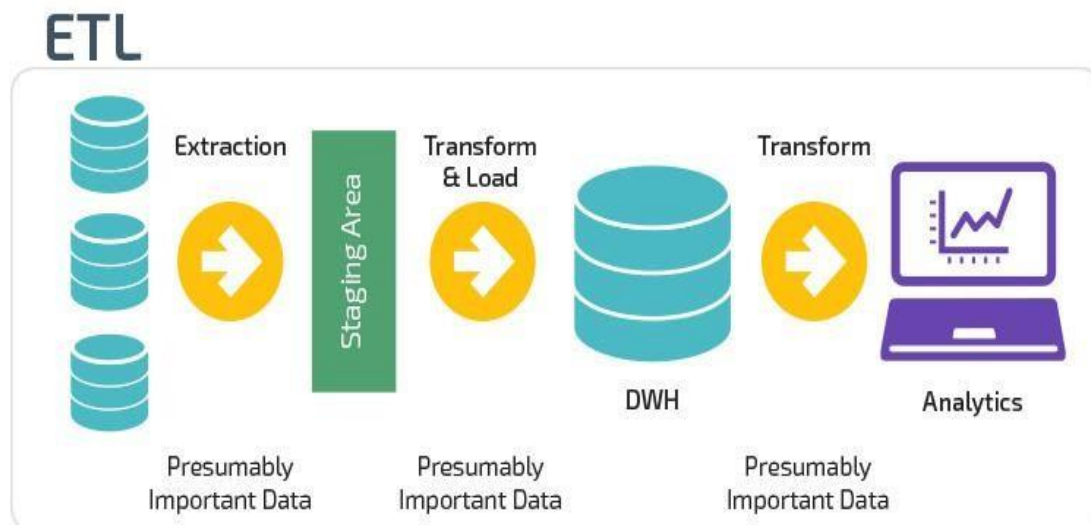
Pre-requisites:

1. Basics of ETL Tools.
2. Concept of Sql Server.

Theory:

ETL(Extract, Transform and Load)

ETL is a process in Data Warehousing and it stands for Extract, Transform and Load.



It is a process, in which an ETL tool extracts the data from various data source systems, transforms it in the staging area and then finally, loads it into the Data Warehouse system.

Extraction

1. **Identify the Data Sources:** The first step in the ETL process is to identify the data sources. This may include files, databases, or other data repositories.
2. **Extract the Data:** Once the data sources are identified, we need to extract the data from them. This may involve writing queries to extract the relevant data or using tools such as SSIS to extract data from files or databases.
3. **Validate the Data:** After extracting the data, it's important to validate it to ensure that it's accurate and complete. This may involve performing data profiling or data quality checks.

Transformation

1. **Clean and Transform the Data:** The next step in the ETL process is to clean and transform the data. This may involve removing duplicates, fixing invalid data, or converting data types. We can use tools such as SSIS or SQL scripts to perform these transformations.
2. **Map the Data:** Once the data is cleaned and transformed, we need to map the data to the appropriate tables and columns in the database. This may involve creating a data mapping document or using a tool such as SSIS to perform the mapping.

Loading

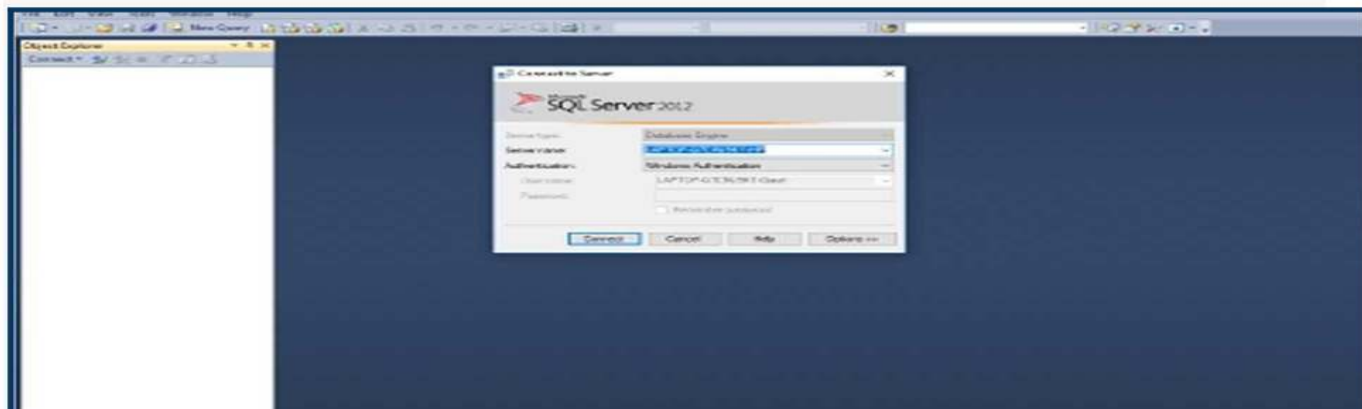
1. **Create the Database:** Before loading the data, we need to create the database and the appropriate tables. This can be done using SQL Server Management Studio or a SQL script.
2. **Load the Data:** Once the database and tables are created, we can load the data into the database. This may involve using tools such as SSIS or writing SQL scripts to insert the data into the appropriate tables.
3. **Validate the Data:** After loading the data, it's important to validate it to ensure that it was loaded correctly. This may involve performing data profiling or data quality checks to ensure that the data is accurate and complete.

Perform the Extraction Transformation and Loading (ETL) process to construct the database in the SQL server.

Software requirements: SQL SERVER 2012 FULL VERSION
(SQLServer2012SPI-FullSlipstream-ENU-x86)

Steps to install SQL SERVER 2012 FULL VERSION (SQLServer2012SPI-FullSlipstream-ENU-x86) are given in my previous post.

Step 1: Open SQL Server Management Studio to restore backup file



Step 2: Right click on Databases Restore Database

Step 3: Click on towards end of device box

Step 4: Click on Add Select path of backup files

Step 5: Select both files at a time

Step 6 : Click ok and in select backup devices window Addboth files of AdventureWorks

Step 7: Open SQL Server Data Tools

Select File New Project Business Intelligence Integration Services Project & give appropriate project name.

Step 8: Right click on Connection Managers in solution explorer and click on New Connection Manager.

Add the SSIS connection manager window.

Step 9: Select OLEDB Connection Manager and Click on Add

Step 10: Configure OLE DB Connection Manager window appears Click on New

Step 11: Select Server name(as per your machine) from drop down and databasename and click on Test connection.

If the test connection succeeded, click on OK.

Step 12: Click on OK

Connection is added to connection manager

Step 13: Drag and drop Data Flow Task in Control Flow tab

Step 14: Drag OLE DB Source from Other Sources and drop into Data Flow tab

Step 15: Double click on OLE DB source -> OLE DB Source Editor appears->click on New to add connection manager.

Select [Sales].[Store] table from drop down ok

Step 16: Drag ole db destination in data flow tab and connect both

Step 17: Double click on OLE DB destination

Click on New to run the query to get [OLE DB Destination] in Name of the table or the view.

Click on OK.

Step 18: Click on Start

Step 19: Go to SQL Server Management Studio

In database tab Adventureworks Right click on [dbo].[OLE DB Destination]Script Table as
SELECT To New Query Editor Window

Step 20: Execute the following query to get output.

```
USE [AdventureWorks2012]GO
```

```
SELECT [BusinessEntityID]
```

```
,[Name]
```

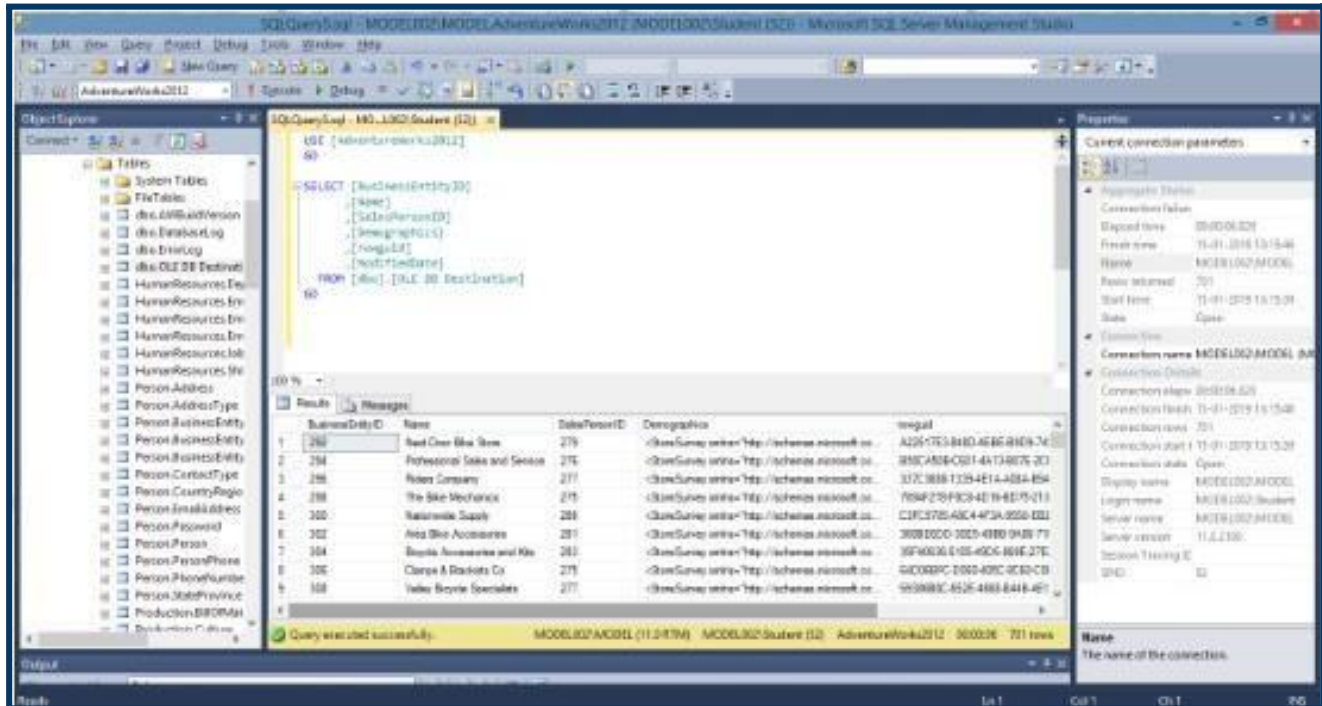
```
,[SalesPersonID]
```

```
,[Demographics]
```

```
,[rowguid]
```

```
,[ModifiedDate]
```


FROM [dbo].[OLE DB Destination]GO



Conclusion: In this way we can perform the ETL process to construct a database in SQL Server.

Group A

Assignment No: 3

Title of the Assignment:

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

Objective of the Assignment:

To introduce the concepts and components of Business Intelligence (BI).

Outcome:

Apply basic principles of elective subjects to problem solving and modeling.
Use tools and techniques in the area of software development to build mini projects

Pre-requisites:

1. Basics of OLAP.
 2. Concept of Multi-Dimensional Cube.
-

Theory:

1. What is a Fact Table ?

In Business Intelligence (BI), A Fact Table is a table that stores quantitative data or facts about a business process or activity. It is a central table in a data warehouse that provides a snapshot of a business at a specific point in time.

For example - A Fact Table in a retail business might contain sales data for each transaction, with dimensions such as date, product, store, and customer. Analysts can use the Fact Table to analyze trends and patterns in sales, such as which products are selling the most, which stores are performing well, and which customers are buying the most.

2. What is a ROLAP, MOLAP and HOLAP model

ROLAP, MOLAP, and HOLAP are three types of models used in Business Intelligence (BI) for organizing and analyzing data:

1. ROLAP (Relational Online Analytical Processing):

In this model, data is stored in a relational database, and the analysis is performed by joining multiple tables. ROLAP allows for complex queries and is good for handling large amounts of data, but it may be slower due to the need for frequent joins.

2. MOLAP (Multidimensional Online Analytical Processing):

In this model, data is stored in a multidimensional database, which is optimized for fast query performance. MOLAP is good for analyzing data in multiple dimensions, such as time, geography, and product, but may be limited in its ability to handle large amounts of data.

1. HOLAP (Hybrid Online Analytical Processing):

This model combines elements of both ROLAP and MOLAP. It stores data in both a relational and multidimensional database, allowing for efficient analysis of both large amounts of data and complex queries. HOLAP is a good compromise between the other two models, offering both speed and flexibility.

2. Create the cube with a suitable dimension and fact tables based on OLAP ?

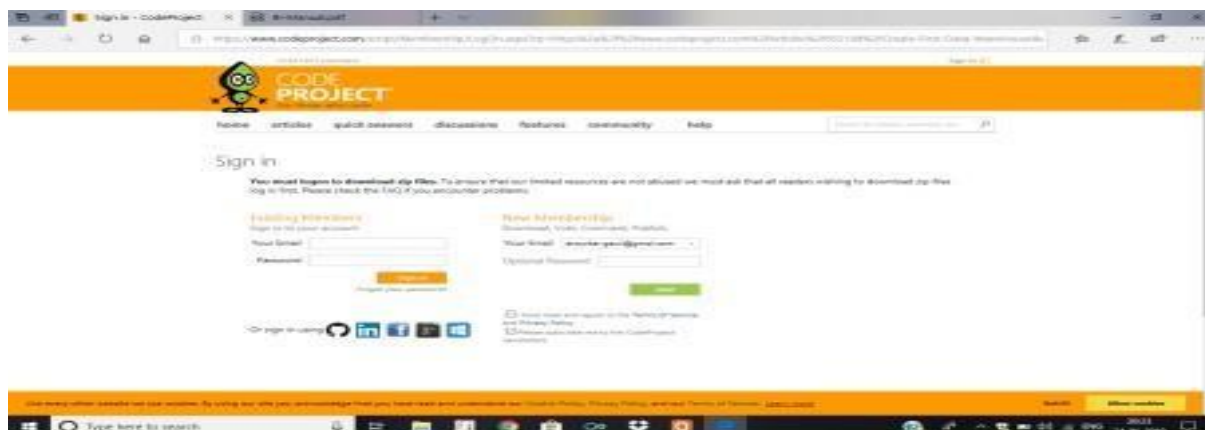
Step 1: Creating Data Warehouse

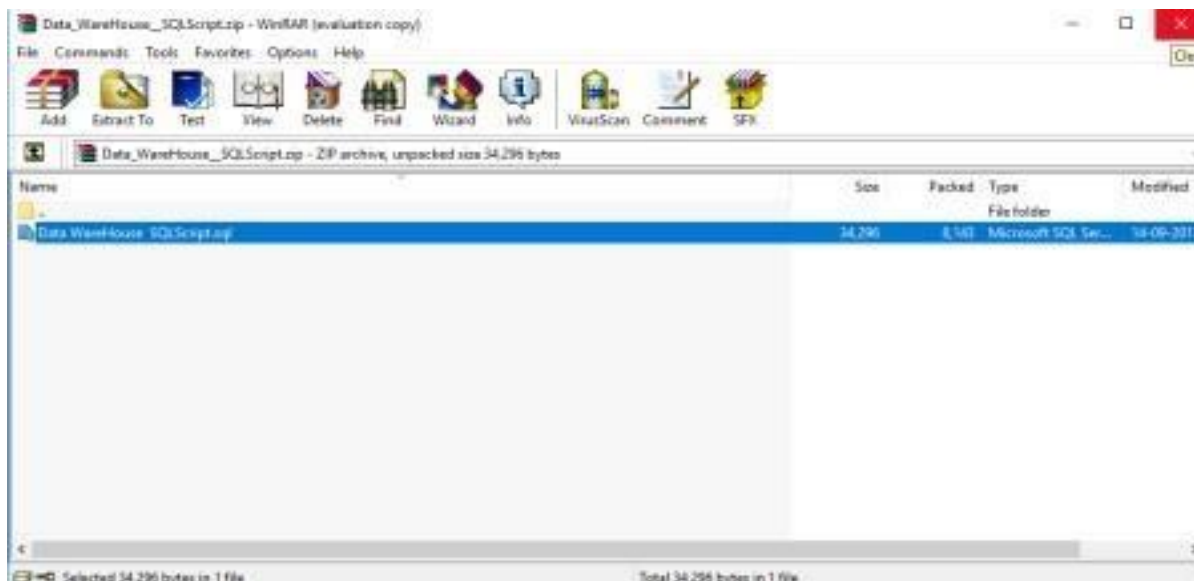
Let us execute our T-SQL Script to create a data warehouse with fact tables, dimensions and populate them with appropriate test values.

Download the 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.

Downloading "Data_WareHouse_SQLScript.zip" from the article

<https://www.codeproject.com/Articles/652108/Create-First-Data-Warehouse>





After downloading the extract file in the folder.

Follow the given steps to run the query in SSMS (SQL Server Management Studio).

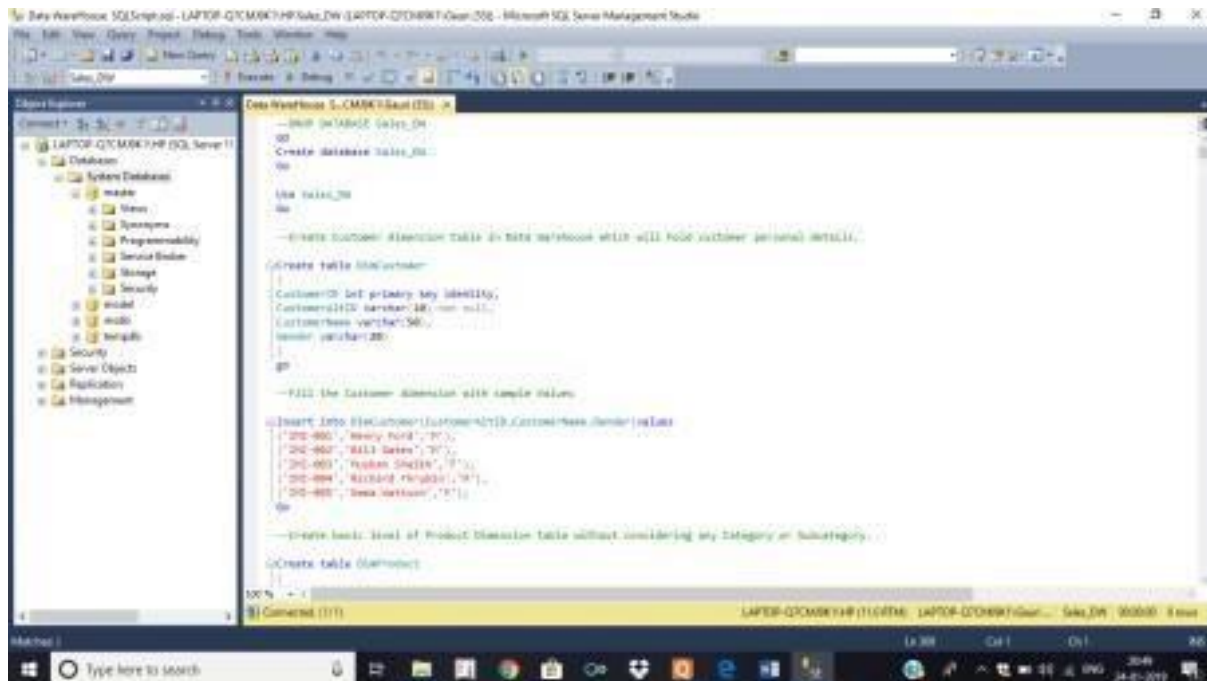
1. Open SQL Server Management Studio 2012
2. Connect Database Engine



Password for sa : admin123 (as given during installation) Click Connect.

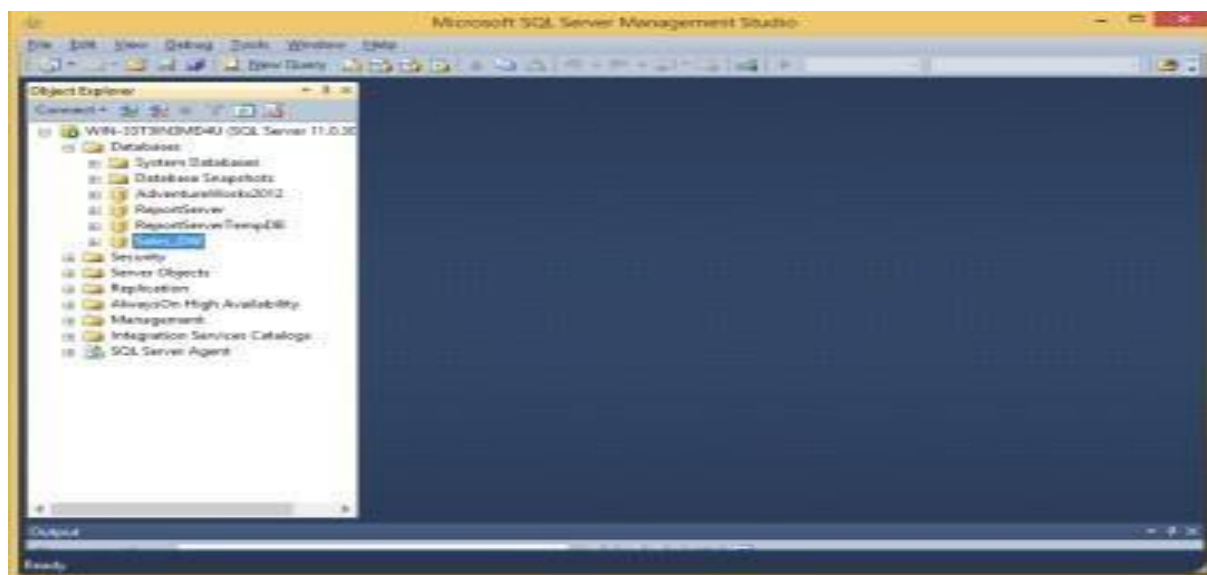
3. Open New Query editor
4. Copy paste Scripts given below in various steps in new query editor window one by one
5. To run the given SQL Script, press F5
6. It will create and populate “Sales_DW” database on your SQL Server OR

7. Go to the extracted sql file and double click on it.
8. New Sql Query Editor will be opened containing the Sales_DW Database.



9. Click on execute or press F5 by selecting the query one by one or directly click on Execute.

After completing execution save and close SQL Server Management studio & Reopen to see Sales_DW in Databases Tab.



Step 2: Start SSDT environment and create New Data Source Go to Sql Server DataTools --> Right click and run as administrator



Click on File → New → Project

In Business Intelligence → Analysis Services Multidimensional and Data Mining models → appropriate project name → click OK

Right click on Data Sources in solution explorer → New Data Source
Data Source Wizard appears

Click on New

Select Server Name → select Use SQL Server Authentication → Select or enter a database name (Sales_DW)

Note : Password for sa : admin123 (as given during installation of SQL 2012 fullversion)

Click Next

Select Inherit → Next

Click Finish

Sales_DW.ds gets created under Data Sources in Solution Explorer

Step 3: Creating New Data Source View

In Solution explorer right click on Data Source View → Select New Data Source View

Click Next

click Next

select FactProductSales(dbo) from Available objects and put in Includes Objects by clicking

Click Next

Click Finish

Sales DW.dsv appears in Data Source Views in Solution Explorer.

Step 4: Creating new cube

Right click on Cubes → New Cube

Select Use existing tables in Select Creation Method → Next

In Select Measure Group Tables → Select FactProductSales → Click Next

In Select Measures → check all measures → Next

In Select New Dimensions → Check all Dimensions → Next

Click on Finish

Sales_DW.cube is created

Step 5: Dimension Modification

In dimension tab → Double Click Dim Product.dim

Drag and Drop Product Name from Table in Data Source View and Add in AttributePane at left side

Step 6: 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 7: Deploy Cube

Right click on Project name → Properties

This window appears

Do following changes and click on Apply & ok

Right click on project name → Deploy

Deployment successful

To process cube right click on Sales_DW.cube → Process

Click run

Browse the cube for analysis in solution explorer

Conclusion : In this way we successfully implement cube with suitable dimension and fact tables based on ROLAP, MOLAP and HOLAP model

Group A

Assignment No: 4

Title of the Assignment:

Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.

Objective of the Assignment:

To introduce the concepts and components of Business Intelligence (BI)

Outcome:

Apply basic principles of elective subjects to problem solving and modeling.

Use tools and techniques in the area of software development to build mini projects

Pre-requisites:

1. Basics of Google Sheets.
2. Concept of Table, Chart.

Contents for Theory:

1. What is a Data Warehouse?
2. What is Pivot Table and Pivot Chart?
3. Steps for Creating a Pivot Table in Google Sheets.
4. Steps for Creating a Pivot Chart in Google Sheets.

Theory:**1. *What is a Data Warehouse?***

A data warehouse is a centralized repository of integrated and transformed data from multiple sources within an organization. It is designed to support business

intelligence (BI) activities, such as data analysis, reporting, and decision-making.

2. What is Pivot Table and Pivot Chart?

A pivot table is a powerful tool in spreadsheet software (such as Google Sheets or Microsoft Excel) that allows you to summarize and analyze large datasets by grouping and summarizing data in different ways. Pivot tables allow you to quickly create tables that show a summary of data based on specific criteria or dimensions. For example, you can use a pivot table to summarize sales data by region or by product category. A pivot chart is a graphical representation of the data in a pivot table. Pivot charts allow you to visualize the summarized data in a way that is easy to understand and interpret. They can be created based on the data in a pivot table, and can be customized in a variety of ways to better represent the data being analyzed. Pivot charts are especially useful when dealing with large amounts of data, as they can help identify patterns and trends that might not be immediately obvious from the raw data.

3. Steps for Creating a Pivot Table in Google Sheets.

1. Open a Google Sheets document with the data you want to use for the pivot table.
2. Select the range of data you want to use for the pivot table.
3. Click on the "Data" tab in the top menu, then click on "Pivot table."
4. In the "Create Pivot Table" dialog box, select the range of data you want to use for the pivot table and choose where you want to place the pivot table (in a new sheet or in the same sheet).
5. Click on "Create."
6. In the pivot table editor, drag and drop the columns you want to use for the pivot table into the "Rows," "Columns," and "Values" sections.
7. To add a filter to the pivot table, drag a column into the "Filter" section.
8. To customize the values in the pivot table, click on the drop-down menu in the "Values" section and choose the type of calculation you want to use (such as sum, count, or average).
9. Customize any additional options in the pivot table editor (such as sorting and formatting).
10. Click on "Update" to apply the changes and create the pivot table

4. Steps for Creating a Pivot Chart in Google Sheets.

1. Open a Google Sheets document with the data you want to use for the pivot chart.
2. Select the range of data you want to use for the pivot chart.
3. Click on the "Data" tab in the top menu, then click on "Pivot table."
4. In the "Create Pivot Table" dialog box, select the range of data you want to use for the pivot table and choose where you want to place the pivot table (in a new sheet or in the same sheet).
5. Click on "Create."
6. In the pivot table editor, drag and drop the columns you want to use for the

8. pivot chart into the "Rows" and "Values" sections.
7. Click on the "Chart" tab in the pivot table editor.
8. Choose the type of chart you want to use for the pivot chart from the drop-down menu.
9. Customize the chart options (such as chart title, axis labels, and colors) to your liking.
10. Click on "Update" to apply the changes and create the pivot chart.

Conclusion: In this way we pivot table and pivot chart using Google spreadsheets | Excel.

Group A

Assignment No: 5

Title of the Assignment:

Perform the data classification using classification algorithm. Or perform the data clustering using a clustering algorithm.

Objective of the Assignment:

To introduce the concepts and components of Business Intelligence (BI)

Outcome:

Apply basic principles of elective subjects to problem solving and modeling.

Use tools and techniques in the area of software development to build mini projects

Pre-requisites:

1. Basics of Tableau.
-

Contents for Theory:

1. What is Clustering and classification?
2. Clustering in Tableau:
3. Classification in Tableau:

Theory:

1. What is Clustering and classification?

Clustering and classification are two important techniques used in bioinformatics to analyze biological data. Clustering is the process of grouping similar objects or data points together based on their similarity or distance from each other. In bioinformatics, clustering is often used to group

Genes or proteins based on their expression patterns or sequences. Clustering can help identify patterns and relationships between different genes or proteins, which can

provide insights into their biological function and interactions. Classification, on the other hand, is the process of assigning a label or category to a new observation based on its features or characteristics. In bioinformatics, classification is often used to predict the function or activity of a new gene or protein based on its sequence or structure. Classification can help identify new drug targets or biomarkers for disease diagnosis and treatment. Both clustering and classification are important tools for analyzing large and complex biological datasets and can provide valuable insights into the underlying biological processes.

Clustering in Tableau:

1. Connect to the data: Connect to the data set that you want to cluster in Tableau.
2. Drag and drop the data fields: Drag and drop the data fields into the view, and select the data points that you want to cluster.
3. Choose a clustering algorithm: Select a clustering algorithm from the analytics pane in Tableau. Tableau provides several built-in clustering algorithms, such as K-Means and Hierarchical Clustering.
4. Define the number of clusters: Define the number of clusters that you want to create. You can do this manually or let Tableau automatically determine the optimal number of clusters.
5. Analyze the clusters: Visualize the clusters and analyze them using Tableau's built-in visualizations and tools.

Classification in Tableau:

1. Connect to the data: Connect to the data set that you want to classify in Tableau.
2. Drag and drop the data fields: Drag and drop the data fields into the view, and select the target variable that you want to predict.
3. Choose a classification algorithm: Select a classification algorithm from the analytics pane in Tableau. Tableau provides several built-in classification algorithms, such as Decision Trees and Random Forest.
4. Define the model parameters: Define the model parameters, such as the maximum tree depth or the number of trees to use in the forest.
5. Train the model: Train the model on a subset of the data using Tableau's built-in cross-validation functionality.
6. Evaluate the model: Evaluate the accuracy of the model using Tableau's built-in metrics, such as confusion matrix, precision, recall, and F1 score.
7. Predict the target variable: Use the trained model to predict the target variable for new data.

8. Visualize the results: Create visualizations to communicate the results of the classification analysis using Tableau's built-in visualization tools.

Conclusion: In this way we implement classification and clustering using Tableau.

GROUP B: ASSIGNMENTS

410252(C): Software Defined Networks

Group B

Assignment No: 1

Title of the Assignment:

Prepare setup for Mininet network emulation environment with the help of Virtualbox and Mininet. Demonstrate the basic commands in Mininet and emulate different custom network topology (Simple, Linear, and Tree).View flow tables.

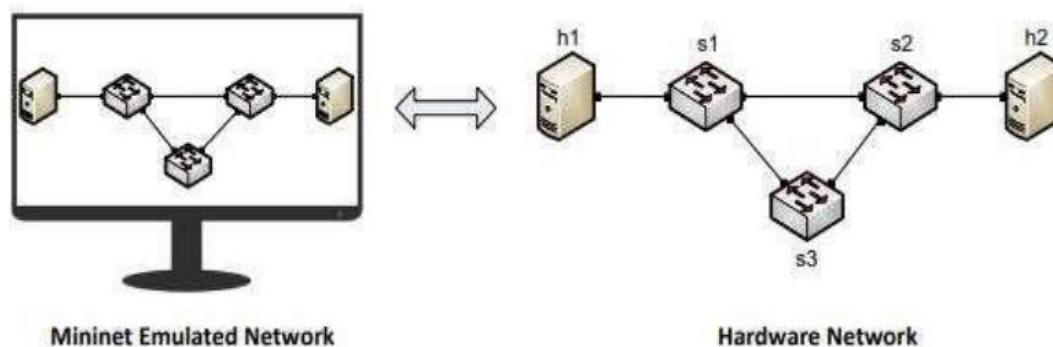
Objective of the Assignment:

1. Learn installation on Mininet emulator.
2. Learn basic commands in Mininet
3. Learn how to create different topologies in Mininet

Theory:

Mininet is a virtual testbed enabling the development and testing of network tools and protocols. With a single command, Mininet can create a realistic virtual network on any type of machine (Virtual Machine (VM), cloud-hosted, or native). Therefore, it provides an inexpensive solution and streamlined development running in line with production networks¹. Mininet offers the following features:

- Fast prototyping for new networking protocols.
- Simplified testing for complex topologies without the need of buying expensive hardware.
- Realistic execution as it runs real code on the Unix and Linux kernels.
- Open source environment backed by a large community contributing extensive documentation.



Mininet is useful for development, teaching, and research as it is easy to customize and interact with it through the CLI or the GUI. Mininet was originally designed to experiment with OpenFlow² and Software-Defined Networking (SDN)³. This lab, however, only focuses on emulating a simple network environment without SDN-based devices. Mininet's logical nodes can be connected into networks. These nodes are sometimes called containers, or more accurately, network namespaces. Containers consume sufficiently fewer resources than networks of over a thousand nodes have created, running on a single laptop. A Mininet container is a process (or group of processes) that no longer has access to all the host system's native network interfaces. Containers are then assigned virtual Ethernet interfaces, which are connected to other containers through a virtual switch⁴. Mininet connects a host and a switch using a virtual Ethernet (veth) link. The veth link is analogous to a wire connecting two virtual interfaces, as illustrated below.

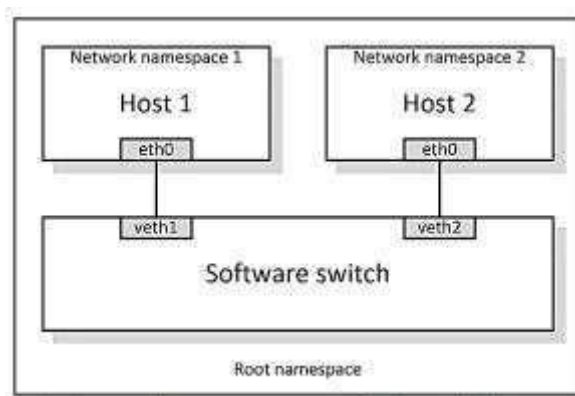


Figure 2. Network namespaces and virtual Ethernet links.

Each container is an independent network namespace, a lightweight virtualization feature that provides individual processes with separate network interfaces, routing tables, and Address Resolution Protocol (ARP) tables. Mininet provides network emulation opposed to simulation, allowing all network software at any layer to be simply run as is; i.e. nodes run the native network software of the physical machine. On the other hand, in a simulated environment applications and protocol implementations need to be ported to run within the simulator before they can be used.

Invoke Mininet using the default topology

Step 1. Launch a Linux terminal by holding the Ctrl+Alt+T keys or by clicking on the Linux

terminal icon.

Step 2. To start a minimal topology, enter the command shown below. When prompted for a password, type password and hit enter. Note that the password will not be visible as you type it.

```
$sudo mn
```

```
sdn@admin:~$ sudo mn
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet>
```

The above command starts Mininet with a minimal topology, which consists of a switch connected to two hosts as shown below.

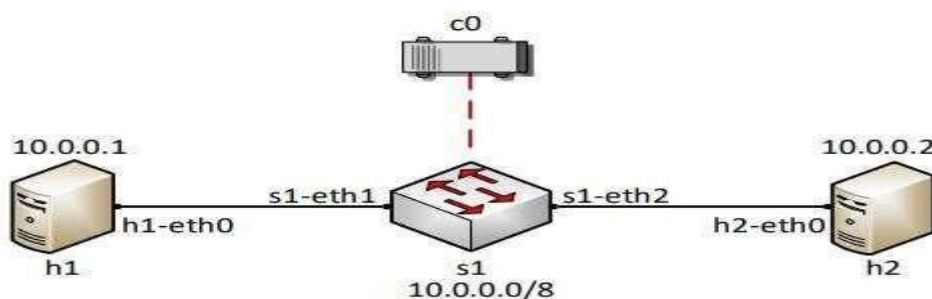


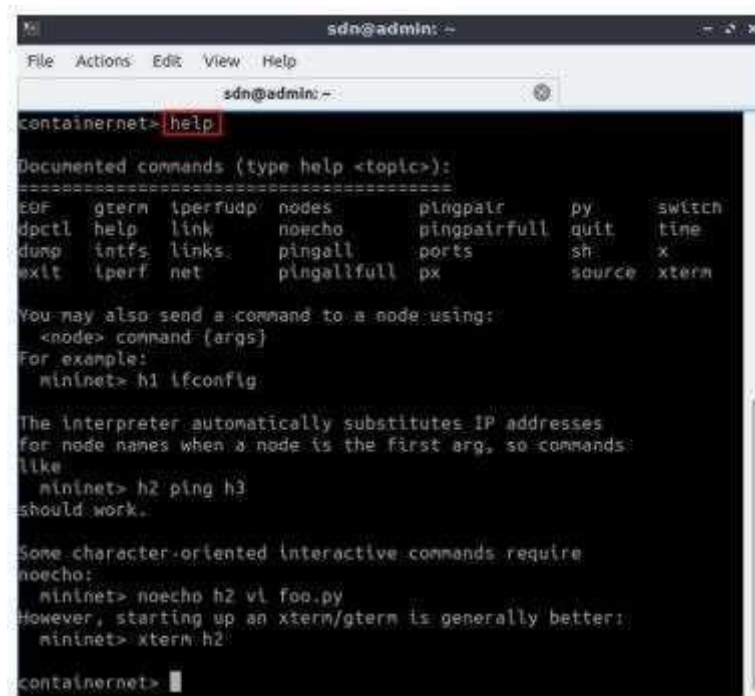
Figure 4. Mininet's default minimal topology

When issuing the `sudo mn` command, Mininet initializes the topology and launches its command line interface which looks like this:

```
mininet>
```

Step 3. To display the list of Mininet CLI commands and examples on their usage, type the following command:

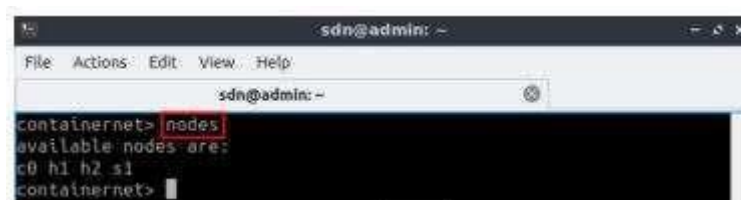
```
help
```



```
sdn@admin: ~  
File Actions Edit View Help  
sdn@admin: ~  
containernet> help  
Documented commands (type help <topic>):  
=====  
EOF      gterm  iperfudp  nodes      pingpair  py        switch  
dpctl    help   link      noecho     pingpairfull  quit      time  
dump     intfs  links     pingall    ports     sh        x  
exit     lperf  net       pingallfull  px        source    xterm  
  
You may also send a command to a node using:  
<node> command [args]  
For example:  
mininet> h1 ifconfig  
  
The interpreter automatically substitutes IP addresses  
for node names when a node is the first arg, so commands  
like  
mininet> h2 ping h3  
should work.  
  
Some character-oriented interactive commands require  
noecho:  
mininet> noecho h2 vi foo.py  
However, starting up an xterm/gterm is generally better:  
mininet> xterm h2  
containernet>
```

Step 4. To display the available nodes, type the following command:

nodes



```
sdn@admin: ~  
File Actions Edit View Help  
sdn@admin: ~  
containernet> nodes  
available nodes are:  
c0 h1 h2 s1  
containernet>
```

Step 5. It is useful sometimes to display the links between the devices in Mininet to understand the topology. Issue the command shown below to see the available links.



```
sdn@admin: ~  
File Actions Edit View Help  
sdn@admin: ~  
containernet> net  
h1 h1-eth0:s1-eth1  
h2 h2-eth0:s1-eth2  
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0  
c0  
containernet>
```

Test connectivity**h1 ping h2**

Mininet>dump: This command shows the dump information about all nodes available in the currentMininet network.

Creation of Topologies in mininet;

Linear topology

In mininet we have various topologies like minimal, single, reversed, linear, tree topologyetc.

- **Minimal/ Simple:** It is the most basic topology with two hosts and one switch. To runminimal topology we simply run the following command in the terminal window i.e.

Sudo mn --topo minimal

- **Single Topology:** It is the simple topology with one switch and N hosts. To run thistopology we run following command in terminal window i.e.

Sudo mn – topo single,3

- **Reversed Topology:** It is similar to the single connection but order of connection betweenhosts and switch is reversed. To run reversed topology we use the command in terminal window i.e.

Sudo mn –topo reversed,3

Conclusion: We learn how to install Mininet and basic commands, and how to create topology.

FAQ:-

1] What is difference between Emulator and simulator?

2] What is difference between Software defined network and traditional

network?3] What is the use of SDN or Application of SDN

4] What are the challenges in Traditional network.

Group B

Assignment No: 2

Title of the Assignment:

After studying open source POX and Floodlight controller, Install controller and run custom topology using remote controller like POX and floodlight controller.

Recognize inserted flows by controllers.

Objective of the Assignment:

1. Learn installation of controller
 2. Run custom topology using remote controller like POX and floodlight controller.
-

Theory:

Open source POX:

POX provides a framework for communicating with SDN switches using either the OpenFlow or OVSDB protocol. Developers can use POX to create an SDN controller using the Python programming language. It is a popular tool for teaching about and researching software defined networks and network applications programming.

POX can be immediately used as a basic SDN controller by using the stock components that come bundled with it.

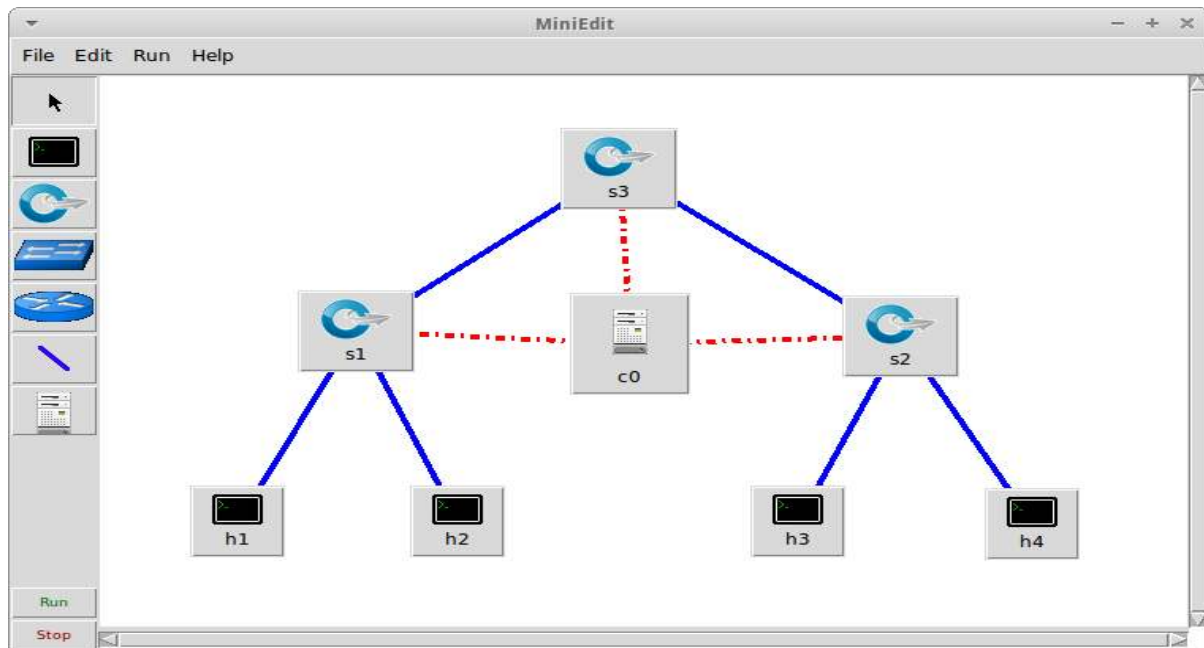
Developers may create a more complex SDN controller by creating new POX components. Or, developers may write network applications that address POX's API

Floodlight controller :

Floodlight Controller is an SDN Controller developed by an open community of developers, many of which from Big Switch Networks, that uses with the OpenFlow protocol to orchestrate traffic flows in a software-defined networking (SDN) environment. OpenFlow is one of the first and most widely used SDN standards; it defines the open communications protocol in an SDN environment that allows the SDN Controller (brains of the network) to speak to the forwarding plane (switches, routers, etc.) to make changes to the network.

Build the network and use a remote controller

Build the network consisting of a tree of switches with a central core switch connected to two other switches that are connected to two hosts, each. Connect a controller to all the switches.



Start the POX controller

Before we start the POX SDN controller, we need to determine which components we want to run when we start the controller.

Select the POX components to run

To select the correct stock components, determine what behavior we want the network of switches to exhibit. Then we will select the stock POX component that provides that functionality.

In this practical, we will use components that make POX work like a Layer 2 learning switch, and that dump copies of packets received by the controller to the controller's log file (so we can see what packets the controller sees), and that list controller events to the console log screen in an easy-to-read format. According to the POX documentation, the stock components that do these tasks are: `forwarding.l2_pairs`, `info.packet_dump`, `samples.pretty_log`, and `log.level`.

Start POX

To start the POX controller with the selected stock components, enter the following command on a terminal session connected to the Mininet VM.

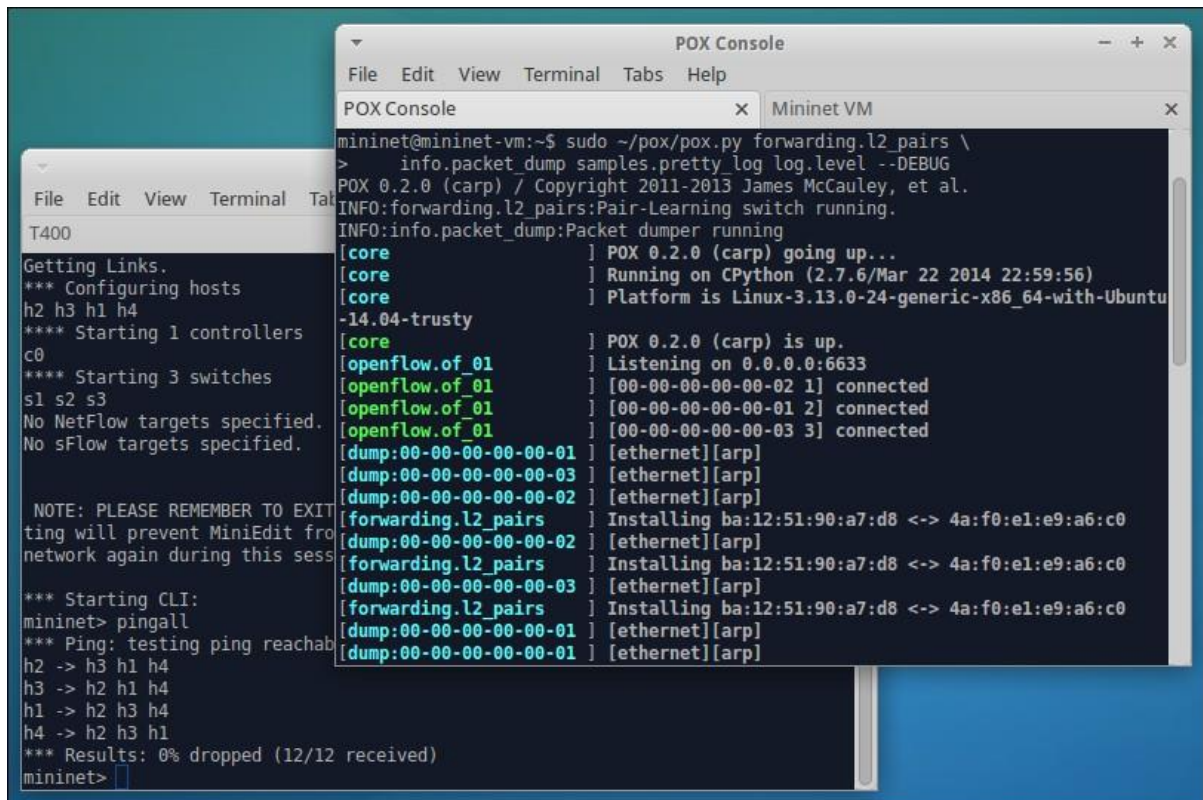
```
mininet-vm:~$ sudo ~/pox/pox.py forwarding.l2_pairs \ info.packet_dump
samples.pretty_log log.level --DEBUG
```

In the POX console, see that the log shows the controller starts and connects to the switches previously set up by the Mininet network simulator:

Test the controller

The forwarding.l2_pairs component is a very simple application that just matches MAC addresses so it creates a simple scenario to study.

Generate some test traffic between hosts to see how POX builds flows in the network. Run the Mininet `pingall` command, which runs ping tests between each host in the emulated network. This generates traffic to the controller every time a switch receives a packet that has a destination MAC address that is not already in its flow table.



```

mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.l2_pairs \
> info.packet_dump samples.pretty_log log.level --DEBUG
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:forwarding.l2_pairs:Pair-Learning switch running.
INFO:info.packet_dump:Packet dumper running
[core] POX 0.2.0 (carp) going up...
[core] Running on CPython (2.7.6/Mar 22 2014 22:59:56)
[core] Platform is Linux-3.13.0-24-generic-x86_64-with-Ubuntu
-14.04-trusty
[core] POX 0.2.0 (carp) is up.
[openflow.of_01] Listening on 0.0.0.0:6633
[openflow.of_01] [00-00-00-00-00-02 1] connected
[openflow.of_01] [00-00-00-00-00-01 2] connected
[openflow.of_01] [00-00-00-00-00-03 3] connected
[dump:00-00-00-00-00-01] [ethernet][arp]
[dump:00-00-00-00-00-02] [ethernet][arp]
[dump:00-00-00-00-00-03] [ethernet][arp]
[forwarding.l2_pairs] Installing ba:12:51:90:a7:d8 <-> 4a:f0:e1:e9:a6:c0
[dump:00-00-00-00-00-02] [ethernet][arp]
[forwarding.l2_pairs] Installing ba:12:51:90:a7:d8 <-> 4a:f0:e1:e9:a6:c0
[dump:00-00-00-00-00-03] [ethernet][arp]
[forwarding.l2_pairs] Installing ba:12:51:90:a7:d8 <-> 4a:f0:e1:e9:a6:c0
[dump:00-00-00-00-00-01] [ethernet][arp]
[dump:00-00-00-00-00-01] [ethernet][arp]

mininet> pingall
*** Ping: testing ping reachability
h2 -> h3 h1 h4
h3 -> h2 h1 h4
h1 -> h2 h3 h4
h4 -> h2 h3 h1
*** Results: 0% dropped (12/12 received)
mininet>
  
```

You can see in the POX console window the log messages showing what is happening. When the POX controller running the forwarding.l2_pairs component receives a packet from a switch, it tells the switch to flood the ARP packet out its other ports to other switches or hosts. One host eventually responds to the ARP request and then the forwarding.l2_pairs component sends OpenFlow messages to each switch to load the required flows into the switch flow tables.

Checking flow tables

To see the contents of the flow tables on all switches, execute the Mininet command:

```
mininet> dpctl dump-flows
```

To check ARP tables on each host, execute the Mininet arp command. For example, to show the ARP table for host h1, enter the following command:

```
mininet> h1 arp
```

To clear all flow tables on all switches, enter the Mininet command:

```
mininet> dpctl del-flows
```

Conclusion

We learn how to install controller and run custom topology using remote controller like POX and floodlight controller. Recognize inserted flowsby controller

Group B

Assignment No: 3

Title of the Assignment:

Create a SDN environment on mininet and configure a switch to provide a firewall functionality using pox controller.

Objective of the Assignment:

1. Learn installation of Mininet
2. Configured a switch to provide firewall functionality using POX controller.

Theory:

SDN environment

An SDN (Software Defined Networking) environment is a network architecture that separates the network control plane from the data plane, allowing for centralized management and programmability. In an SDN environment, the network control plane is managed by a software controller, which communicates with the network devices using the OpenFlow protocol.

To create an SDN environment, several components are required, including:

1. SDN controller: The SDN controller is the central management component that communicates with the network devices using the OpenFlow protocol.
2. Network devices: The network devices, such as switches and routers, are responsible for forwarding data packets based on the instructions from the SDN controller.
3. OpenFlow protocol: The OpenFlow protocol is used to communicate between the SDN controller and the network devices.
4. Network applications: Network applications can be developed to run on top of the SDN environment, providing additional network services and functions.
5. Network topology: The network topology is the physical or logical arrangement of the network devices and links.

SDN environment is a network architecture that separates the control plane from the data plane, allowing for centralized management and programmability. SDN environments offer greater flexibility, scalability, and automation in network management, and enable new network

services and applications.

Firewall functionality

Firewall functionality is a security feature that is used to protect computer networks from unauthorized access and malicious traffic. In the context of SDN, a firewall can be implemented using an SDN controller, which controls the network switches and enforces security policies.

The firewall functionality implemented by an SDN controller typically involves setting up flow rules on the network switches to block or allow traffic based on certain criteria, such as the source and destination IP addresses, the transport protocol used, and the ports being used. For example, an SDN controller can set up flow rules to block traffic from certain IP addresses, or to block traffic on certain ports that are known to be used by malware.

POX controller

POX is an open-source SDN controller that is written in Python and is designed to support the OpenFlow protocol. POX is designed to be lightweight and flexible, and it provides a number of useful features for building and managing SDN networks.

Some of the key features of POX include:

1. **Modular design:** POX is designed to be modular and extensible, which makes it easy to add new features and functionality.
2. **Python-based:** POX is written in Python, which makes it easy to read, understand, and modify the code. This also means that it is easy to integrate with other Python-based tools and libraries.
3. **OpenFlow support:** POX supports the OpenFlow protocol, which is used to communicate between the controller and the switches in an SDN network.
4. **Debugging tools:** POX includes a number of built-in debugging tools that make it easy to test and troubleshoot SDN applications.
5. **Flexible API:** POX provides a flexible API that can be used to build custom SDN applications and services.

Installation Process :

To create an SDN environment on Mininet and configure a switch to provide a firewall functionality using POX controller, you can follow the steps below:

1. Install Mininet and POX controller:

- Follow the installation guide at <http://mininet.org/download/> to install Mininet
- Follow the installation guide at <https://github.com/noxrepo/pox/wiki/Installation> to install POX controller

2. Create a simple topology:

- Open a terminal and run `sudo mn --topo single,3 --mac --switch ovsk --controller remote`
- This will create a topology with one switch and three hosts, and set the controller as remote.

3. Configure the switch to provide firewall functionality:

- Open a new terminal and navigate to the POX controller directory
- Create a new Python file called `firewall.py`
- Copy and paste the following code into the `firewall.py` file:

python

```
from pox.core import core
import pox.openflow.libopenflow_01 as oflog =

core.getLogger()

class Firewall(object):

    def __init__(self):
        core.openflow.addListeners(self)
        log.info("Firewall initialized.")

    def _handle_ConnectionUp(self, event):msg =
        of.ofp_flow_mod() msg.priority = 65535
        msg.match.dl_type = 0x0800
        msg.match.nw_proto = 6
        msg.match.tp_dst = 80
        msg.actions.append(of.ofp_action_output(port=of.OFPP_CONTROLLER)
        )
        event.connection.send(msg)
```

```
def launch():  
    core.registerNew(Firewall)
```

4. Run the POX controller:

- In the terminal where you created the `firewall.py` file, run `./pox.py log.level --DEBUG openflow.discovery openflow.spanning_tree --no-flood --hold-down firewall`
- This will run the POX controller with the `firewall.py` module.

5. Test the firewall functionality:

- In the Mininet terminal, run `h1 ping h2`
- The ping should be successful
- In the Mininet terminal, run `h1 wget h2`
- The wget should fail because port 80 is blocked by the firewall.

That's it! You have created an SDN environment on Mininet and configured a switch to provide firewall functionality using POX controller.

Conclusion :

We learn how to install Mininet and how to create an SDN environment on Mininet and configure a switch to provide firewall functionality using POX controller.

Group B

Assignment No: 4

Title of the Assignment:

Using Mininet as an Emulator and POX controller, build your own internet router. Write simple router with a static routing table. The router will receive raw Ethernet frames and process the packet forwarding them to correct outgoing interface. You must check the Ethernet frames are received and the forwarding logic is created so packets go to the correct interface.

Objective of the Assignment:

1. To understand the concept of Software Defined Networking (SDN).
 2. To use Mininet and POX controller to build an internet router.
-

Theory:

Router:

- A router is a device that connects different computer networks together. It helps to direct traffic on the internet by sending data packets between networks. The primary function of a router is to route the incoming data packets to the correct destination.
- When a data packet is sent on the internet, it contains information about the sender and the recipient, as well as the actual data being sent. The router receives the packet and reads the information about the recipient. It then checks its own routing table to see where the recipient is located.
- The routing table is a list of destinations and the corresponding paths that the router should use to send the data packets to the correct destination. Once the router has determined the correct path, it sends the packet to the next router or directly to the destination if it is located on the same network.
- Routers are used to connect different types of networks together, such as local area networks (LANs) or wide area networks (WANs). They are also used to connect devices within a network, such as computers, printers, and other devices.
- Routers can perform other functions besides routing data packets. For example, they can act as firewalls to protect the network from unauthorized access, or they can perform network address translation (NAT) to hide the internal network addresses from the outside world.

Ethernet Frames:

- Ethernet frames are the basic units of data in Ethernet networks. They contain important information about the data being transmitted and how it should be forwarded to its destination. Here are some key points to explain Ethernet frames:
- Ethernet frames are composed of several fields that contain information about the data being transmitted. Some of these fields include:
- Destination MAC address: The MAC address of the device that the frame is being sent to.
- Source MAC address: The MAC address of the device that sent the frame.
- Type/length: Indicates the type of data contained in the frame (e.g., IP packets).
- Payload: The data being transmitted.
- Ethernet frames are used to transmit data between devices on an Ethernet network, such as a local area network (LAN).
- The size of Ethernet frames is limited to a maximum of 1518 bytes, including the header and trailer.
- Ethernet frames are typically transmitted using a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol, which helps to avoid collisions when multiple devices are transmitting data simultaneously.
- When an Ethernet frame is transmitted, it is first sent to the switch or router closest to the source device. The switch or router then examines the destination MAC address in the frame to determine which interface the frame should be forwarded to.
- If the destination MAC address is not known, the switch or router may broadcast the frame to all devices on the network to try to find the device with the correct MAC address.

Routing Tables:

- Routing tables are used by routers to determine the next hop for packets based on their destination IP address.
- A static routing table is one that is manually configured and does not change dynamically.

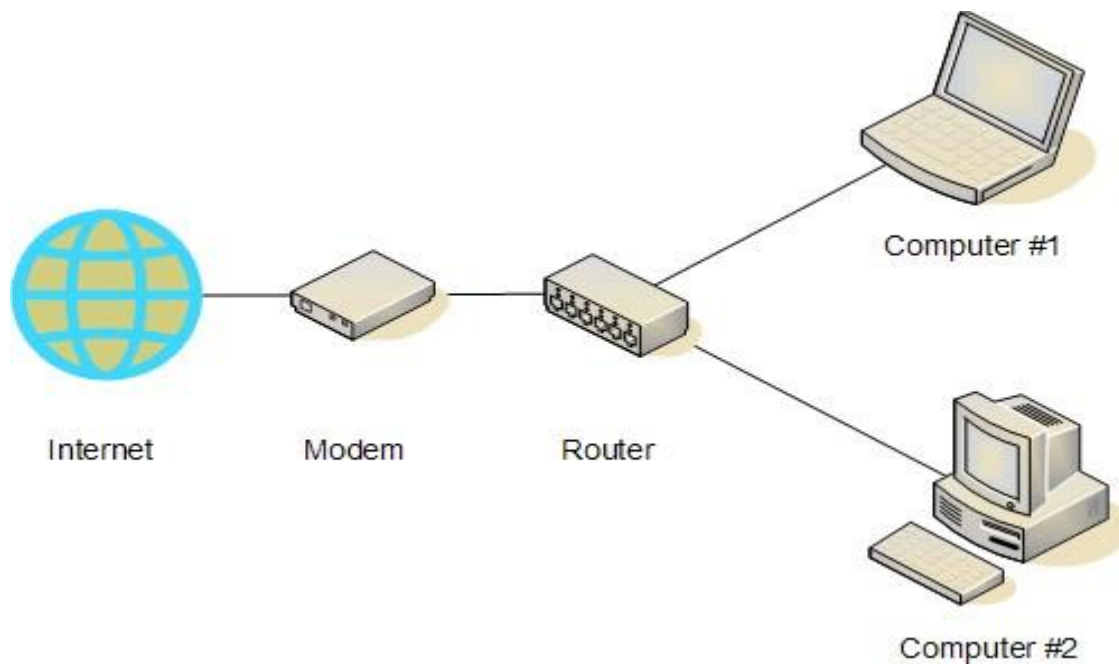


Fig. 1: Overview of Router's Role in a Network

Procedure:

Check Out Starter Code

```
cd ~  
git clone https://huangty@bitbucket.org/huangty/cs144_lab3.gitcd krish/  
git checkout --track remotes/origin/standalone
```

Install Simple Router POX module

```
cd ~/krish  
./config.sh
```

Configuration Files

There will be two configuration files:

- * ~/krish/IP_CONFIG: Listed out the IP addresses assigned to the emulated hosts.
- * ~/krish/router/rtable (also linked to ~/krish/rtable): The static routing table used for the simple router.

The default IP_CONFIG and rtable should look like the following:

```
> cat ~/krish/IP_CONFIG  
server1 192.168.2.2  
server2 172.64.3.10
```

```
client          10.0.1.100
sw0-eth1 192.168.2.1
sw0-eth2 172.64.3.1
sw0-eth3 10.0.1.1
> cat ~/krish/rtable
10.0.1.100 10.0.1.100 255.255.255.255 eth3

192.168.2.2 192.168.2.2 255.255.255.255 eth1
172.64.3.10 172.64.3.10 255.255.255.255 eth2
```

Test Connectivity of Your Emulated Topology:

Configure the environment by running the config.sh file

```
> cd ~/krish/
> ./config.sh
```

Start Mininet emulation by using the following command

```
> cd ~/krish/
> ./run_mininet.sh
```

You should be able to see some output like the following:

```
*** Shutting down stale SimpleHTTPServers
*** Shutting down stale webserversserver1
192.168.2.2
server2 172.64.3.10
client 10.0.1.100
sw0-eth1 192.168.2.1
sw0-eth2 172.64.3.1
sw0-eth3 10.0.1.1
*** Successfully loaded ip settings for hosts
{'server1': '192.168.2.2', 'sw0-eth3': '10.0.1.1', 'sw0-eth1':
'192.168.2.1', 'sw0-eth2': '172.64.3.1', 'client': '10.0.1.100', 'server2':
'172.64.3.10'}
*** Creating network
*** Creating network
*** Adding controller
*** Adding hosts:
client server1 server2
*** Adding switches:
sw0
*** Adding links:
(client, sw0) (server1, sw0) (server2, sw0)
*** Configuring hosts client
server1 server2
*** Starting controller
*** Starting 1 switchessw0
*** setting default gateway of host server1server1
192.168.2.1
*** setting default gateway of host server2server2
172.64.3.1
```



```
*** setting default gateway of host clientclient
10.0.1.1

*** Starting SimpleHTTPServer on host server1
*** Starting SimpleHTTPServer on host server2
*** Starting CLI:
mininet>
```

Keep this terminal open, as you will need the mininet command line for debugging. Now, use another terminal to continue the next step. (Do not press ctrl-z.)

Mininet requires a controller, which we implemented in POX (revision f95dd1a81584d716823bbf565fa68254416af603). To run the controller, use the following command:

```
> cd ~/krish/
> ln -s ../pox
> ./run_pox.sh
```

You should be able to see some output like the following:

```
POX 0.0.0 / Copyright 2011 James McCauley
DEBUG:.home.ubuntu.krish.pox_module.cs144.ofhandler:*** ofhandler:
Successfully loaded ip settings for hosts
{'server1': '192.168.2.2', 'sw0-eth3': '10.0.1.1', 'sw0-eth1':
'192.168.2.1', 'sw0-eth2': '172.64.3.1', 'client': '10.0.1.100', 'server2':
'172.64.3.10'}

INFO:.home.ubuntu.krish.pox_module.cs144.srhandler:created server
DEBUG:.home.ubuntu.krish.pox_module.cs144.srhandler:SRServerListener listening on 8888
DEBUG:core:POX 0.0.0 going up...
DEBUG:core:Running on CPython (2.7.3/Aug 1 2012 05:14:39)
INFO:core:POX 0.0.0 is up.
This program comes with ABSOLUTELY NO WARRANTY. This program is freeware,
and you are welcome to redistribute it under certain conditions.Type
'help(pox.license)' for details.
DEBUG:openflow.of_01:Listening for connections on 0.0.0.0:6633Ready.
POX>
```

Please note that you have to wait for Mininet to connect to the POX controller before you continue to the next step. Once Mininet has connected, you will see the following output:

```
INFO:openflow.of_01:[Con 1/249473472573510] Connected to e2-e5-11-b6-b0-46
DEBUG:.home.ubuntu.krish.pox_module.cs144.ofhandler:Connection [Con
1/249473472573510]
DEBUG:.home.ubuntu.krish.pox_module.cs144.srhandler:SRServerListener catchRouterInfo
even, info={'eth3': ('10.0.1.1', '86:05:70:7e:eb:56', '10Gbps',
3), 'eth2': ('172.64.3.1', 'b2:9e:54:d8:9d:cd', '10Gbps', 2), 'eth1':
('192.168.2.1', '36:61:7c:4f:b6:7b', '10Gbps', 1)}, rtable=[]
```

Keep POX running. Now, open yet another terminal to continue the next step. (Don't press ctrl-z.)

Now you are ready to test out the connectivity of the environment setup. To do so, run the binary file of the solution, `sr_solution`:

```
> cd ~/krish/
> ./sr_solution
```

You should be able to see some output like the following:

Loading routing table from server, clear local routing table>Loading routing table

Destination	Gateway	Mask	Iface
10.0.1.100	10.0.1.100		255.255.255.255 eth3
192.168.2.2	192.168.2.2		255.255.255.255 eth1
172.64.3.10	172.64.3.10		255.255.255.255 eth2

Client ubuntu connecting to Server localhost:8888Requesting topology 0

successfully authenticated as ubuntu

Loading routing table from server, clear local routing table>Loading routing table

Destination	Gateway	Mask	Iface
10.0.1.100	10.0.1.100		255.255.255.255 eth3
192.168.2.2	192.168.2.2		255.255.255.255 eth1
172.64.3.10	172.64.3.10		255.255.255.255 eth2

Router interfaces:

```
eth3    HWaddr86:05:70:7e:eb:56
        inet addr 10.0.1.1
eth2    HWaddrb2:9e:54:d8:9d:cd
        inet addr 172.64.3.1
eth1    HWaddr36:61:7c:4f:b6:7b
        inet addr 192.168.2.1
<-- Ready to process packets -->
```

In this particular setup, 192.168.2.2 is the IP for `server1`, and 172.64.3.10 is the IP for `server2`. You can find the IP addresses in your `IP_CONFIG` file.

Now, back to the terminal where Mininet is running. To issue an command on the emulated host, type the host name followed by the command in the Mininet console. For example, the following command issues 3 pings from the `client` to `server1`.

```
mininet> client ping -c 3 192.168.2.2
```

You should be able to see the following output.

```
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.  
64 bytes from 192.168.2.2: icmp_req=1 ttl=63 time=66.9 ms  
64 bytes from 192.168.2.2: icmp_req=2 ttl=63 time=49.9 ms  
64 bytes from 192.168.2.2: icmp_req=3 ttl=63 time=68.8 ms
```

You can also use traceroute to see the route between client to server1.

```
mininet> client traceroute -n 192.168.2.2
```

You should be able to see the following output.

```
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets  
 1  10.0.1.1  146.069 ms  143.739 ms  143.523 ms  
 2  192.168.2.2  226.260 ms  226.070 ms  225.868 ms
```

Finally, to test the web server is properly working at the server1 and server2, issue an HTTP request by using *wget* or *curl*.

```
mininet> client wget http://192.168.2.2
```

You should be able to see the following output.

```
--2012-12-17 06:52:23-- http://192.168.2.2/  
Connecting to 192.168.2.2:80... connected. HTTP request  
sent, awaiting response... 200 OKLength: 161 [text/html]  
Saving to: `index.html'
```

```
OK  
100% 17.2M=0s
```

```
2012-12-17 06:52:24 (17.2 MB/s) - `index.html' saved [161/161]
```

Conclusion: Thus, we learned how to build our own internet router using Mininet as an emulator and POX controller.

Group B

Assignment No: 5

Title of the Assignment:

Emulate and manage a Data Center via a Cloud Network Controller: create a multi-rooted tree-like (Clos) topology in Mininet to emulate a data center. Implement specific SDN applications on top of the network controller in order to orchestrate multiple network tenants within a data center environment, in the context of network virtualization and management.

Objective of the Assignment:

1. To understand the concept of Software Defined Networking (SDN).
 2. To emulate and manage a data center environment using a cloud network controller.
 3. To learn how to create a multi-rooted tree-like (Clos) topology in Mininet and implement specific SDN applications to orchestrate multiple network tenants within a data center environment.
-

Requirements:

1. Computer with any Linux OS installed
 2. Python Runtime Environment
 3. Mininet Emulator
 4. POX Controller
-

Theory:

Data Center:

- A data center is a large facility used to house computer systems and related equipment, such as storage devices and networking equipment.
 - It is designed to provide a secure and controlled environment for these systems to operate in, as they are critical to the functioning of many businesses and organizations.
 - Data centers are used to store, process, and manage large amounts of data and applications, such as websites, mobile apps, and databases.
-

- They are typically equipped with backup power supplies, cooling systems, and fire suppression systems to ensure the equipment is protected and can operate continuously.
- Data centers can range in size from small server rooms to large, multi-story facilities that can house thousands of servers.
- They can be operated by a single organization, or by multiple organizations who share the resources of the data center.
- Data centers can be located on-premises, meaning they are physically located at the same site as the organization they serve, or they can be located off-premises, in a separate location, such as a colocation facility or a cloud provider's data center.
- The location and design of a data center can impact its efficiency and environmental impact, so many data centers are designed to be energy-efficient and environmentally friendly.

Data Center (Clos) Topology:

- The Clos topology is a network topology that is commonly used in data centers to provide high levels of scalability and redundancy.
- It is based on a multi-rooted tree-like structure, with multiple layers of switches or routers, each of which is interconnected in a specific way to provide maximum performance and reliability.
- At the core of the Clos topology are three layers of switches or routers, known as the spine layer, the leaf layer, and the host layer.
- The spine layer consists of a set of high-capacity switches or routers that are interconnected in a full-mesh topology, providing high-speed interconnectivity between the leaf layer switches.
- The leaf layer consists of a set of switches or routers that are connected to each spine switch, forming a leaf-spine architecture. Each leaf switch is typically connected to a large number of host devices, such as servers or storage devices.
- The host layer consists of the individual devices themselves, which are connected to the leaf switches. In this way, the host layer can scale up or down as needed, while the spine and leaf layers remain relatively static.
- One of the key advantages of the Clos topology is its ability to provide high levels of redundancy and fault tolerance. By using a full-mesh spine layer, traffic can be easily rerouted around any failed switches or links, ensuring that the network remains operational even in the event of a failure.
- Another advantage of the Clos topology is its ability to scale horizontally, allowing additional leaf switches to be added as needed to support increasing numbers of devices.
- However, one potential disadvantage of the Clos topology is its complexity. Because of the multiple layers and interconnectivity between switches, it can be challenging to manage and troubleshoot in large-scale deployments.

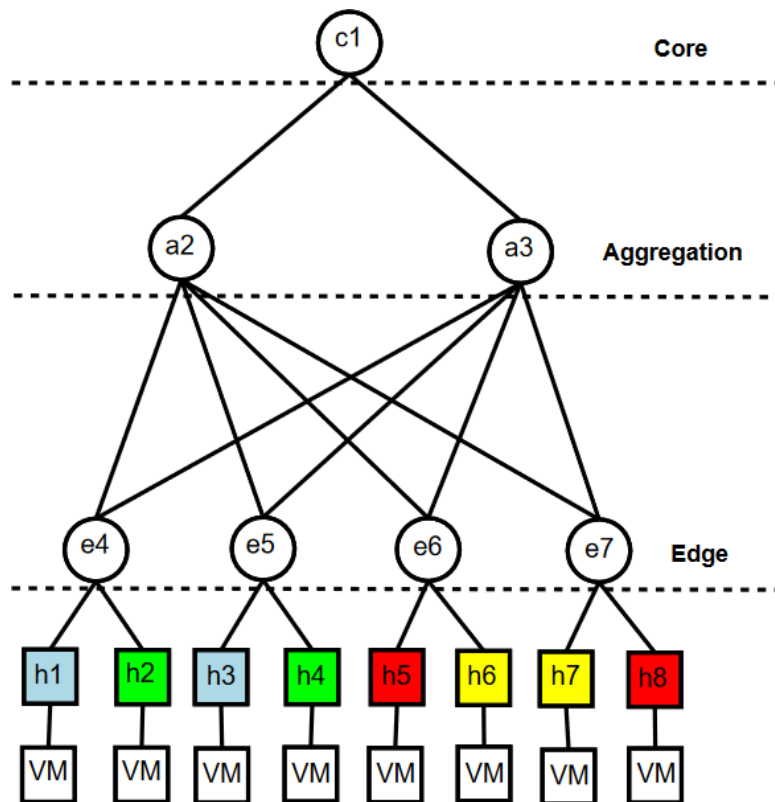


Figure 1: Sample layered data center topology

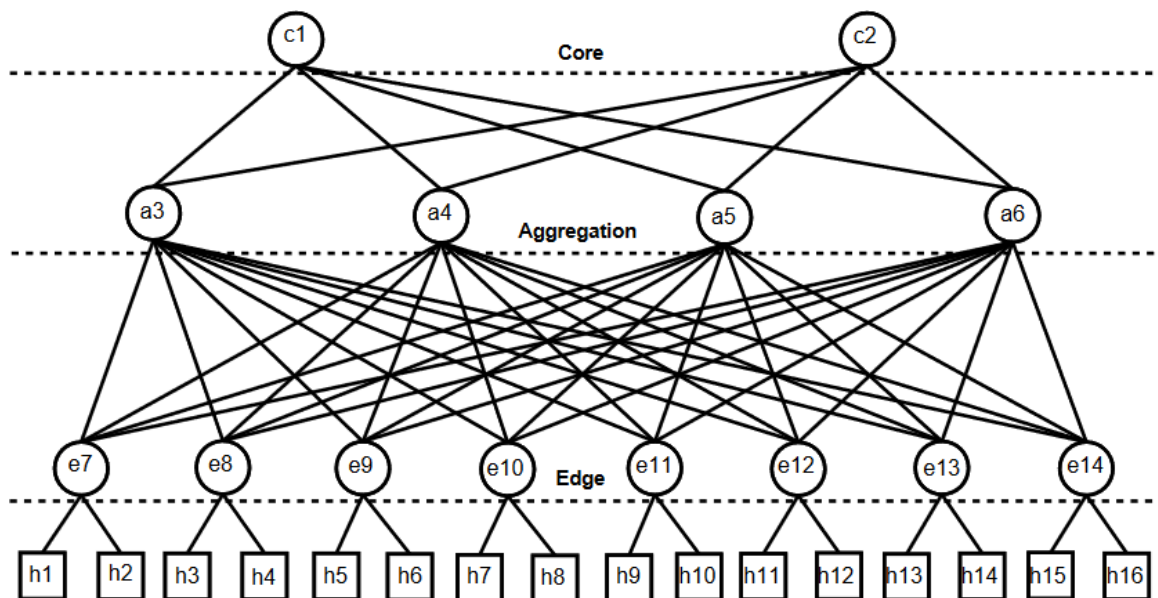
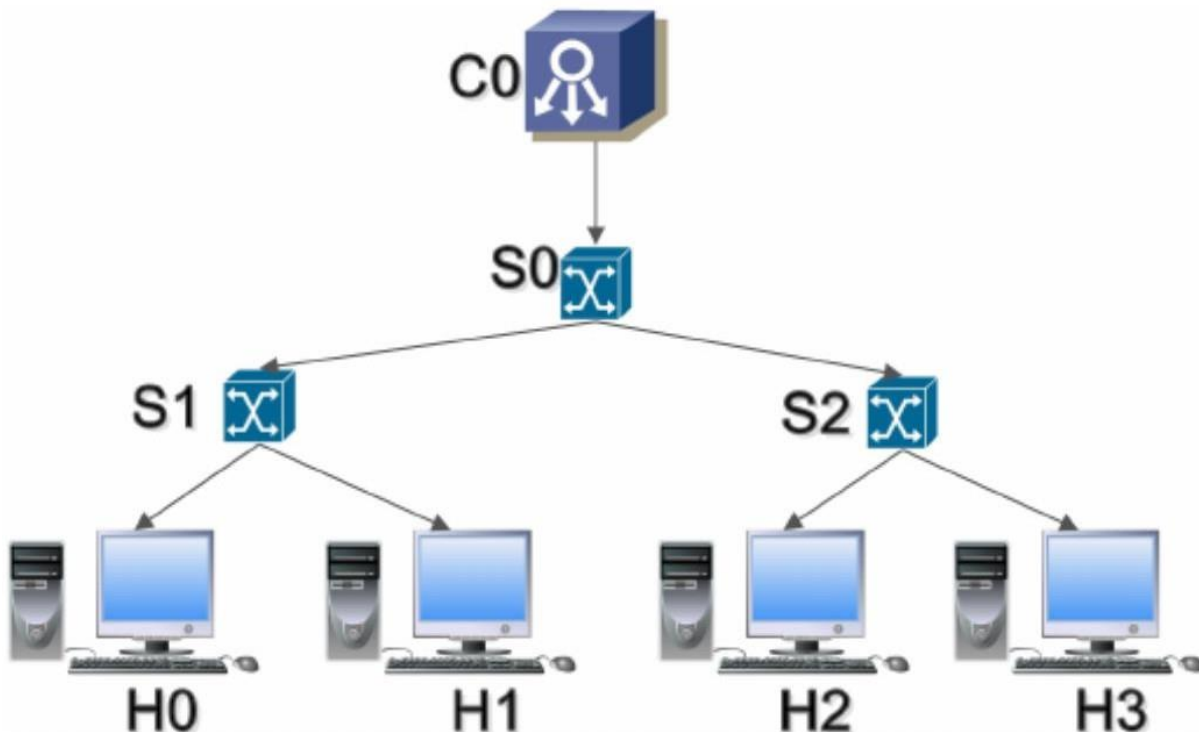


Figure 2: Clos topology with 2 core switches and fanout=2

**Procedure:**

- **Step 0: Install Mininet and POX**
 - Mininet is a network emulator that allows you to create and run virtual networks on a single machine. POX is a network controller that allows you to program and control the behavior of the network using Python scripts.
 - You can install these tools by following the instructions on their websites: <http://mininet.org/download/> and <https://github.com/noxrepo/pox>
 - Alternatively, you can use a virtual machine image provided by the course that already has these tools installed. You can download it from here: <https://drive.google.com/file/d/1y9v9Z0n1yQl7y8J6wYw6j0p6bY4D7fL2/view>
 - To run the virtual machine, you need to install VirtualBox: <https://www.virtualbox.org/wiki/Downloads>
 - To access the virtual machine, you need to use SSH: <https://www.ssh.com/ssh/putty/download>
- **Step 1: Run the script clos topo.py to create the data center network topology in Mininet**

- This script is provided by the course and it defines the structure and parameters of the network. The network consists of four layers: core, aggregation, edge, and host. Each layer has a number of switches and links that connect them. The switches are OpenFlow-enabled and can be controlled by POX.
- To run the script, you need to open a terminal and type: `sudo python clos_topo.py`
- This will create the network topology and start Mininet. You will see a prompt like this: `mininet>`
- You can use various commands to interact with the network, such as `pingall`, `dump`, `nodes`, etc. You can see a list of commands by typing `help`.
- **Step 2: Run the script `CloudNetController.py` to start the POX controller that will manage the network**
 - This script is provided by the course and it contains the code relating to the SDN applications that you will implement. The script uses POX's API to communicate with the switches and install flow rules on them. The script also defines some helper functions and classes that you will use in your tasks.
 - To run the script, you need to open another terminal and type:
`./pox.py log.level --DEBUG
CloudNetController`
 - This will start POX and load the script. You will see some messages like this: `INFO:core:POX 0.5.0 (carp) is up.`
 - The controller will connect to the switches and send them some initial flow rules. You will see some messages like

```
this: DEBUG:CloudNetController:Installing initial flow rules for  
switch s1
```

- **Step 3: Implement basic routing**

- To do this, we need to write a function called `task1_shortest_path_routing` that takes two parameters: `src` and `dst`, which are the source and destination host names, respectively.
- The function should return a list of switch names that form the shortest path between the source and destination hosts. For example, if the shortest path from `h1` to `h4` is `s1-s2-s4`, the function should return `['s1', 's2', 's4']`.
- To find the shortest path, you can use the `get_shortest_path` function that is provided by the script. This function takes two parameters: `src` and `dst`, which

are the source and destination switch names, respectively. It returns a list of switch names that form the shortest path between them.

For example, if the shortest path from s1 to s4 is s1-s2-s4, the function returns ['s1', 's2', 's4'].

- To use the `get_shortest_path` function, you need to map the host names to switch names. You can use

the `host_to_switch` dictionary that is provided by the script. This dictionary maps each host name to its connected switch name. For example, `host_to_switch['h1']` returns 's1'.

- To write the function, you can follow these steps:
 - Initialize an empty list called `path`.
 - Get the switch name of the source host by using the `host_to_switch` dictionary and store it in a variable called `src_switch`.
 - Get the switch name of the destination host by using the `host_to_switch` dictionary and store it in a variable called `dst_switch`.
 - Call the `get_shortest_path` function with `src_switch` and `dst_switch` as parameters and store the result in a variable called `switch_path`.
 - Append `src_switch` to `path`.
 - Loop through `switch_path` from index 1 to the end and append each element to `path`.
 - Append `dst_switch` to `path`.
 - Return `path`.

- **Step 4: Implement firewall policies to block traffic between certain hosts or applications**

- In this task, you need to write a function called `task2_firewall_policies` that takes three parameters: `src`, `dst`, and `app`, which are the source host name, destination host name, and application name, respectively.

- The function should return a boolean value that indicates whether the traffic should be blocked or not. For example, if there is a firewall policy that blocks traffic from h1 to h2 for app1, the function should return `True` when called with `src='h1'`, `dst='h2'`, and `app='app1'`.

- To implement the firewall policies, you can use the `firewall_policies` list that is provided by the script. This list contains tuples of three elements: source host name, destination host name, and application name. Each tuple represents a firewall policy that blocks traffic between those hosts for that application.

For example, ('h1', 'h2', 'app1') means that traffic from h1 to h2 for app1 is blocked.

- To write the function, you can follow these steps:
 - Initialize a variable called `blocked` with `False`.
 - Loop through the `firewall_policies` list and check if there is a tuple that matches the parameters.
 - If there is a match, set `blocked` to `True` and break out of the loop.
 - Return `blocked`.

- **Step 5: Implement virtual network slicing to isolate different tenants or applications**

- In this task, you need to write a function called `task3_virtual_network_slicing` that takes three parameters: `src`, `dst`, and `app`, which are the source host name, destination host name, and application name, respectively.
- The function should return a boolean value that indicates whether the traffic should be allowed or not. For example, if there is a virtual network slice that isolates tenant1 from tenant2, the function should return `False` when called with `src='h1'`, `dst='h3'`, and `app='app1'`, where h1 belongs to tenant1 and h3 belongs to tenant2.
- To implement the virtual network slicing, you can use the `virtual_network_slices` dictionary that is provided.

Python Implementation of the above:

```
def task1_shortest_path_routing(src, dst):
    # Initialize an empty list called path
    path = []
    # Get the switch name of the source host by using the host_to_switch
    dictionary
    src_switch = host_to_switch[src]
    dst_switch = host_to_switch[dst]
    switch_path = get_shortest_path(src_switch, dst_switch)
    # Append src_switch to path
    path.append(src_switch)
    # Loop through switch_path from index 1 to the end and append each
    element to path
    for i in range(1, len(switch_path)):
        path.append(switch_path[i])
    # Append dst_switch to path
    path.append(dst_switch)
    # Return path
    return path
```

```
def task2_firewall_policies(src, dst, app):
    # Initialize a variable called blocked with False
    blocked = False
    # Loop through the firewall_policies list and check if there is a tuple
    # that matches the parameters
    for policy in firewall_policies:
        # If there is a match, set blocked to True and break out of the loop
        if policy == (src, dst, app):
            blocked = True
            break
    # Return blocked
    return blocked
```

```
def task3_virtual_network_slicing(src, dst, app):
    # Get the tenant name of the source host by using the
    # host_to_tenant dictionary
    src_tenant = host_to_tenant[src]
    # Get the tenant name of the destination host by using the
    # host_to_tenant dictionary
    dst_tenant = host_to_tenant[dst]
    # Check if the source and destination hosts belong to the
    # same tenant
    if src_tenant == dst_tenant:
        # If they do, return True to allow the
        # traffic
        return True
    else:
        # If they don't, check if there is a virtual network slice that
        # allows traffic between them for that application
        # Use the virtual_network_slices dictionary to get the list of
        # applications that are allowed for each pair of tenants
        # Use the tuple (src_tenant, dst_tenant) as the key to access
        # the dictionary
        allowed_apps = virtual_network_slices[(src_tenant,
        dst_tenant)] # Check if the application is in the list
        # of allowed apps
        if app in allowed_apps:
            # If it is, return True to allow the
            # traffic
            return True
        else:
            # If it isn't, return False to block the
            # traffic
            return False
```

```
kris@Ubuntu-22:~$ sudo mn --controller remote --custom datacenter.py --topo datacenter
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
a1 a2 a3 a4 c1 c2 e1 e2 e3 e4 e5 e6 e7 e8
*** Adding links:
(a1, e1) (a1, e2) (a2, e3) (a2, e4) (a3, e5) (a3, e6) (a4, e7) (a4, e8) (c1, a1) (c1, a2) (c1, a3) (c1, a4) (c2
, a1) (c2, a2) (c2, a3) (c2, a4) (e1, h1) (e2, h2) (e3, h3) (e4, h4) (e5, h5) (e6, h6) (e7, h7) (e8, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 14 switches
a1 a2 a3 a4 c1 c2 e1 e2 e3 e4 e5 e6 e7 e8 ...
*** Starting CLI:
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=1 Destination Host Unreachable
From 10.0.0.2 icmp_seq=2 Destination Host Unreachable
From 10.0.0.2 icmp_seq=3 Destination Host Unreachable
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=2437 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=1436 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=428 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.035 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.034 ms
^C
--- 10.0.0.1 ping statistics ---
19 packets transmitted, 5 received, +3 errors, 73.6842% packet loss, time 18370ms
rtt min/avg/max/mdev = 0.034/860.299/2437.445/947.184 ms, pipe 4
```

Figure 4: Expected Output

Conclusion: Thus, we learned how to emulate and manage a data center via a cloud network controller using Mininet and SDN applications. We created a multi-rooted tree-like (Clos) topology in Mininet to emulate a data center with multiple hosts and switches. We implemented specific SDN applications on top of the network controller in order to orchestrate multiple network tenants within a data center environment, in the context of network virtualization and management. We also experimented with different traffic patterns and routing algorithms to optimize the network performance.