

PRACTICAL BOOK  
MSCIT (PART I) SEMESTER - I  
2024-25

Data Science  
SUBMITTED BY  
**Shivani Nagesh Shriyan**  
Seat No. **1310818**

Submitted in partial fulfillment of the requirement for  
Qualifying  
M.Sc. Part I Semester I Examination 2024-25

University of Mumbai  
Department of Information Technology  
Shailendra Degree College  
Shailendra Nagar, S.V. Road Dahisar(E) , Mumbai-68

**Shailendra Education Society's  
ARTS, COMMERCE & SCIENCE COLLEGE**

NAAC Re-accredited 'B++' grade with CGPA score 2.98  
Shailendra Nagar, Dahisar (East), Mumbai - 400 068.



**CERTIFICATE**

This is to certify that Miss **Shivani Nagesh Shriyan** of M.Sc. (I.T.) Part I Semester I has completed the practical work in the subject of "**Data Science**" during the academic year 2024- 25 under the guidance of Asst. Prof. **Hemchandra Kumbhar** being the partial requirement for the fulfillment of the curriculum of Master of Sofc iMenucmbe iani .Information Technology, University

Place: Mumbai

Date: / / 2025

-----  
Internal Examiner

-----  
External Examiner

-----  
M.Sc. (IT) Coordinator

College Seal

## Index

Sr.No.	Practical Title	Page No.	Signature
1	Creating and using database in Cassandra.	2	
2	Write the programs for Text Delimited CSV to HORUS format.	15	
3	Write the programs for XML to HORUS format.	16	
4	Write the programs for Video to HORUS format.	18	
5	Write the programs for Audio to HORUS format.	20	
6	Write the programs for CSV to JSON format.	26	
7	Write the programs for CSV to XML format.	27	
8	Write the R program to create the network routing diagram from the given data on routers.	29	
9	Write the R program to build acyclic graph.	30	
10	Write the R program to generate payroll from the given data.	32	
11	Import data from MS-Excel/ MS-Access in Power BI Desktop Show data Visualization	33	

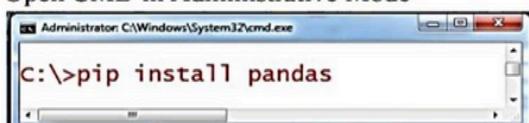
## Prerequisite for Data Science Practical

Vermeulen-Krennwallner-Hillman-Clark Group (VKHCG) is a hypothetical medium-size international company. It consists of four subcompanies: Vermeulen PLC, Krennwallner AG, Hillman Ltd, and Clark Ltd. VKHCG uses the R processing engine to perform data processing in 80% of the company business activities, and the other 20% is done by Python. Therefore, we will prepare an R and a Python environment to perform the examples. I will quickly advise you on how to obtain these additional environments, if you require them for your own specific business requirements.

- While installing Python check the option to Add Python to PATH Variable



- Open CMD in Administrative Mode



- Similarly install the following packages using pip

1. matplotlib	8. datetime	15. sqlalchemy
2. numpy	9. json	16. sql.connector
3. opencv-python	10. msgpack	17. geopandas
4. networkx	11. scipy	18. quandl
5. sys	12. geopy	19. mlxtend
6. uuid	13. pysqllite3	20. folium
7. pyspark	14. openpyxl	

## Sample Data

The following data is for the examples in

Type of file: Comma-separated values (CSV)

### 1. IP Addresses Data Sets

Location: VKHCG\01-Vermeulen\00-RawData

Data file: IP_DATA_C_VKHCG.csv	No. of Record: 255
Data file: IP_DATA_ALL.csv	No. of Records: 1,247,502
Data file: IP_DATA_CORE.csv	No. of Records: 3,562

### 2. Customer Data Sets

Location: VKHCG\02-Krennwallner\00-RawData

Data file: DE_Billboard_Locations.csv	No. of Records: 8,873
---------------------------------------	-----------------------

### 3. Logistics Data Sets

Location: VKHCG\03-Hillman\00-RawData

Data file: GB_Postcode_Full.csv	No. of Records: 1,714,591
Data file: GB_Postcode_Warehouse.csv	No. of Records: 3,005
Data file: GB_Postcodes_Shops.csv	No. of Records: 1,048,575

### 4. Exchange Rate Data Set

Location: VKHCG\04-Clark\00-RawData

Data file: Euro_ExchangeRates.csv	No. of Records: 4,697
-----------------------------------	-----------------------

### 5. Profit-and-Loss Statement Data Set

Location: VKHCG\04-Clark\00-RawData

Data file: Profit_And_Loss.csv	No. of Records: 2,442
--------------------------------	-----------------------

## Practical No. 1

### Creating and using database in Cassandra

Apache Cassandra requires Java 8 to run on a Windows system. Additionally, the Cassandra command-line shell (**cqlsh**) is dependent on Python 2.7 to work correctly.

To be able to install Cassandra on Windows, first you need to:

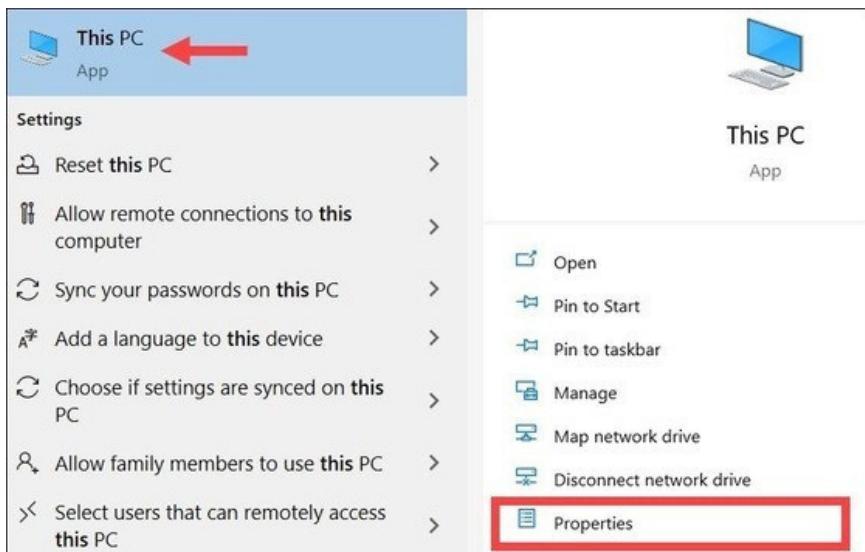
1. Download and Install **Java 8** and set environment variables.
2. Download and install **Python 2.7** and set environment variables.

**Step 1: Install Java 8 on Windows** The Java development kit contains all the tools and software you need to run applications written in Java. It is a prerequisite for software solutions such as Apache Cassandra.

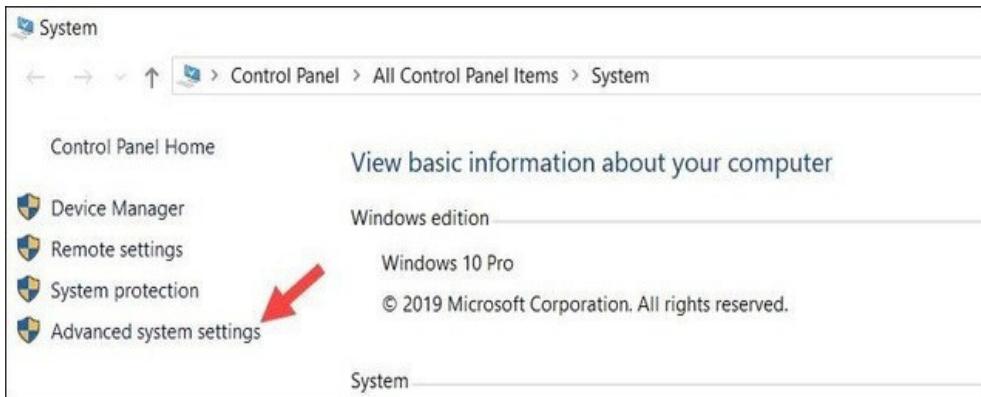
#### **Step 2: Configure Environment Variables for Java 8**

It is vital to configure the environment variables in Windows and define the correct path to the Java 8 installation folder.

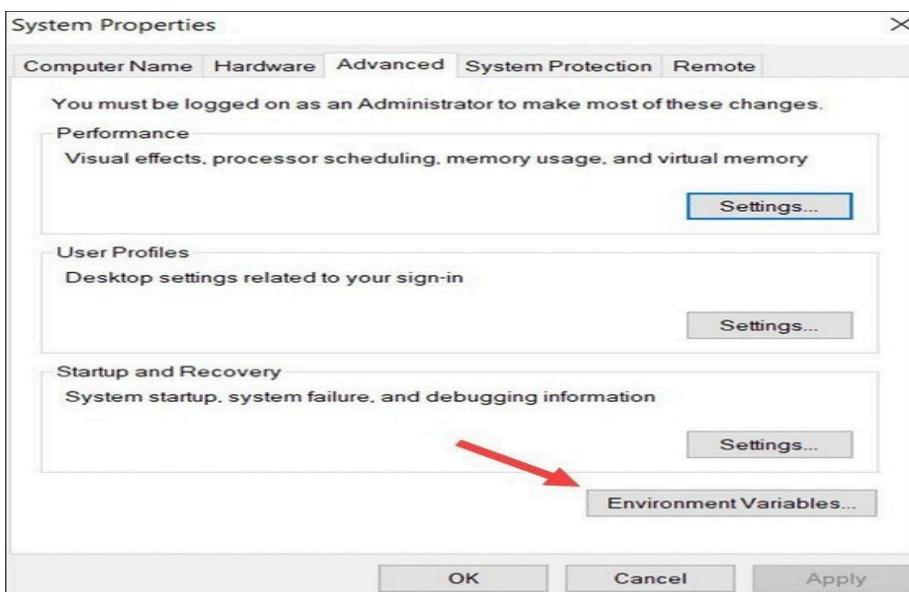
1. Navigate to **This PC > Properties**.



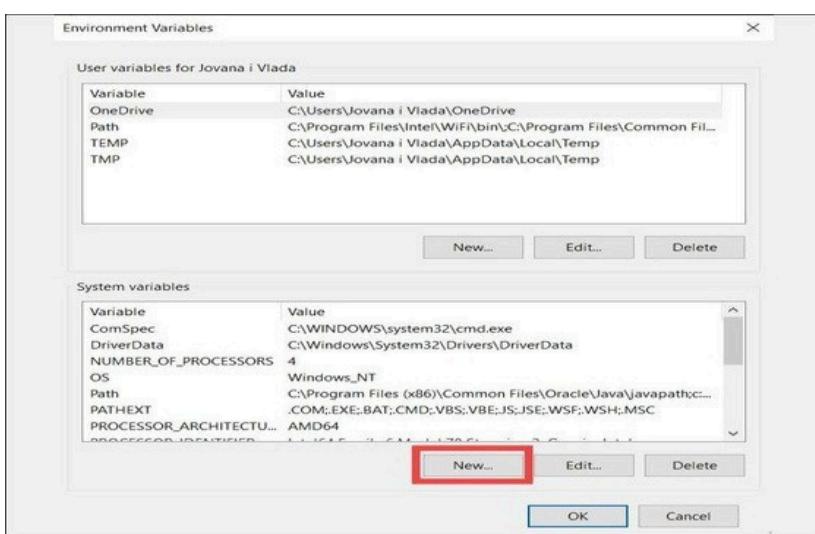
2. Select **Advanced system settings**.



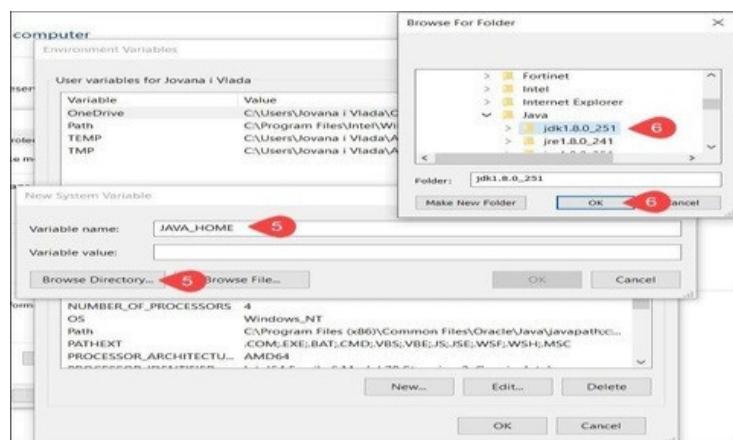
3. Click the Environment Variables... button.



4. Select New in the System Variable section.

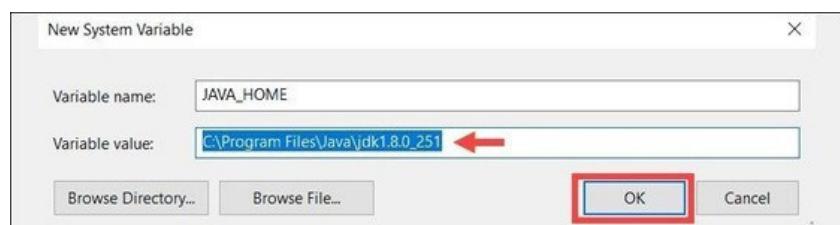


Enter **JAVA\_HOME** for the new variable name. Select the Variable value field and then the Browse Directory option.

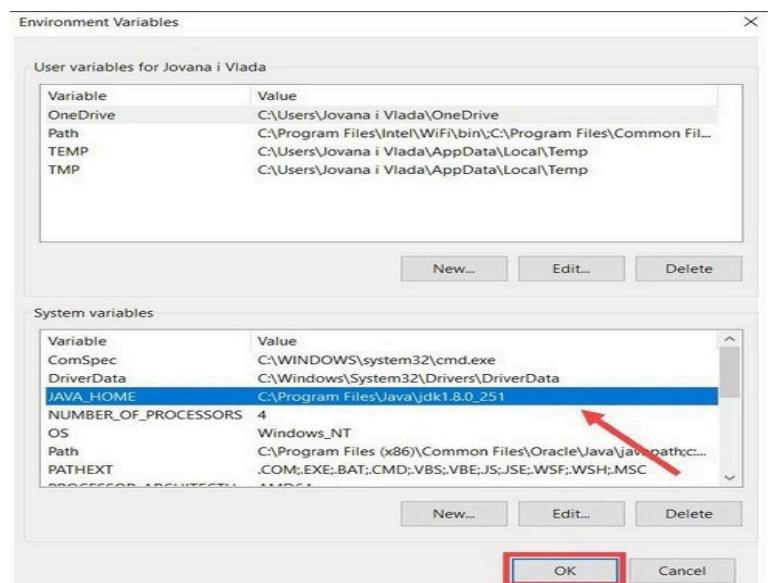


5. Navigate to **This PC > Local Disk C: > Program Files > Java > jdk1.8.0\_251** and select **OK**.

6. Once the correct path to the JDK 8 installation folder has been added to the **JAVA\_HOME** system variable, click **OK**.



7. You have successfully added the **JAVA\_HOME** system variable with the correct JDK 8 path to the variable list. Select **OK** in the main Environment Variables window to complete the process.



### Step 3: Install and Configure Python 2.7 on Windows

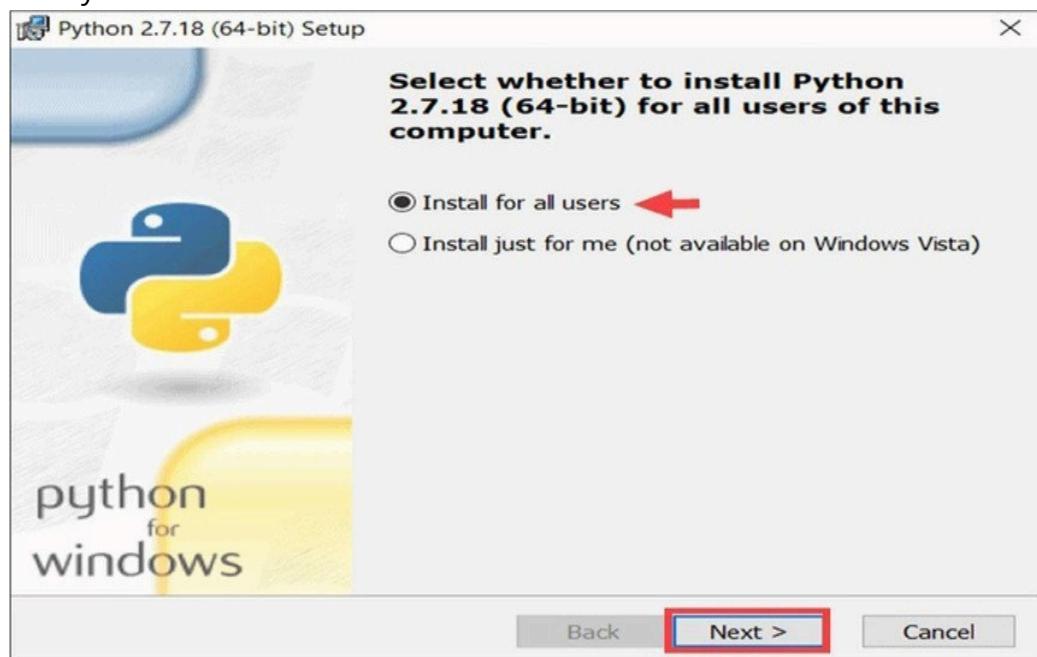
Users interact with the Cassandra database by utilizing the **cqlsh** bash shell. You need to install Python 2.7 for **cqlsh** to handle user requests properly.

Install Python 2.7 on Windows

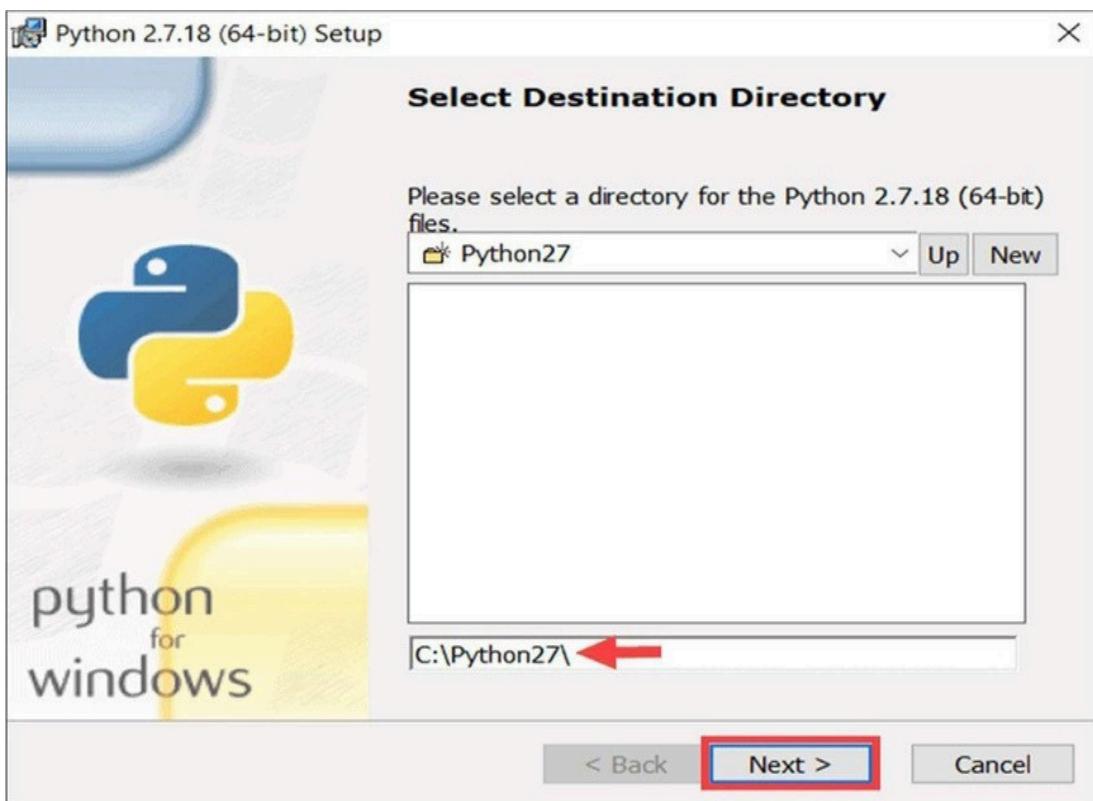
1. Visit the Python official download page and select the Windows x64 version link.

Python 2.7.18						
Release Date: April 20, 2020						
Python 2.7.18 is the last release of Python 2.						
Files						
Version	Operating System	Description	MD5 Sum	File Size	GPG	
Gzipped source tarball	Source release		38c84292658ed4456157195f1c9bcbe1	17539408	SIG	
XZ compressed source tarball	Source release		fd6cc8ec0a78c44036f825e739f36e5a	12854736	SIG	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	ce98eeb7bdff806685adc265ec1444463	24889285	SIG	
Windows debug information files	Windows		20b111ccfe8d06d2fe8c77679a86113d	25178278	SIG	
Windows debug information files for 64-bit binaries	Windows		bb0897ea20fda343e5179d413d4a4a7c	26005670	SIG	
Windows help file	Windows		b3b753dff1c7930243c1c40ec3a72b1	6322188	SIG	
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64	a425c758d38f8e28b56f4724b499239a	20598784	SIG	
Windows x86 MSI installer	Windows		db6ad9195b3086c6b4cef9493d738d2	19632128	SIG	

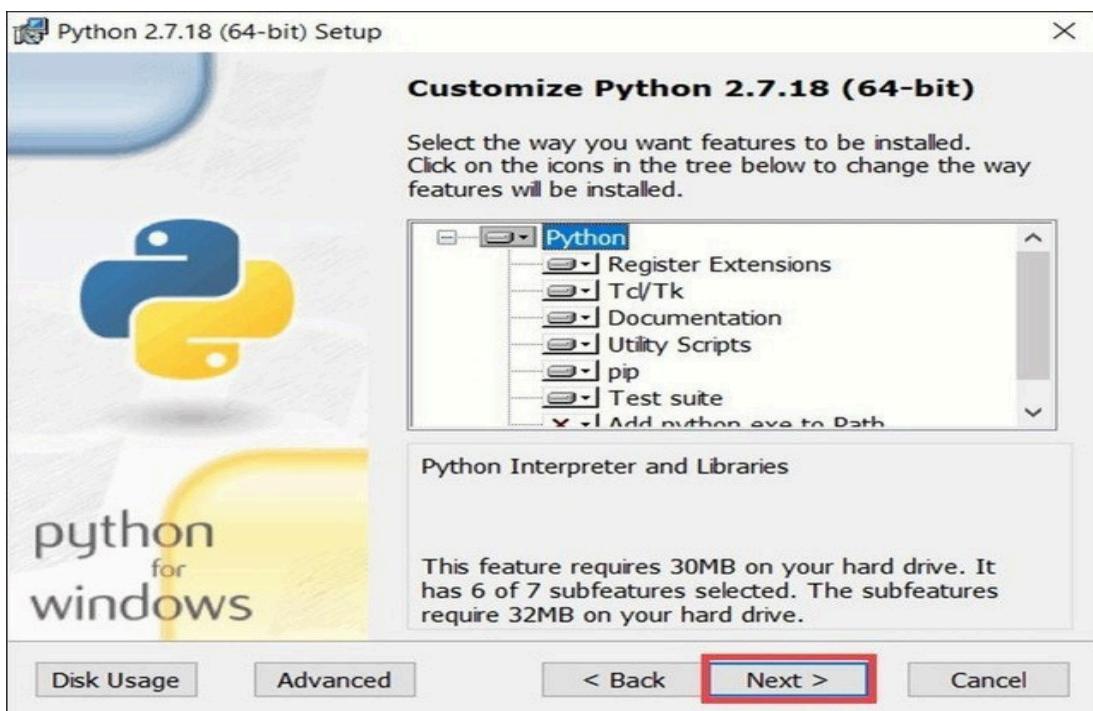
2. Define if you would like Python to be available to all users on this machine or just for your user account and select **Next**.



Specify and take note of the Python installation folder location. Feel free to leave the default location **C:Python27** by clicking **Next**.



The following step allows you to customize the Python installation package. Select Next to continue the installation using the default settings.

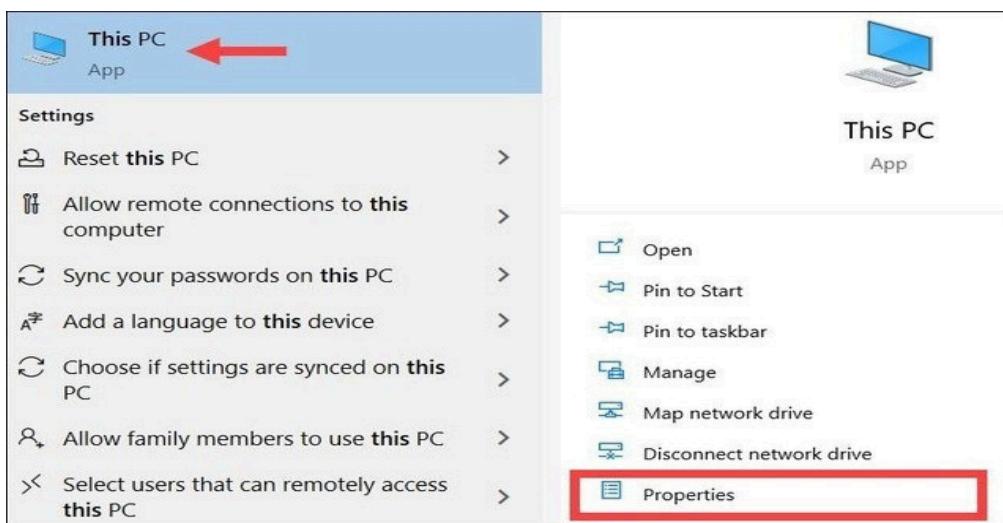


The installation process takes a few moments. Once it is complete, select **Finish** to conclude the installation process.



Edit Environment Variable for Python2.7

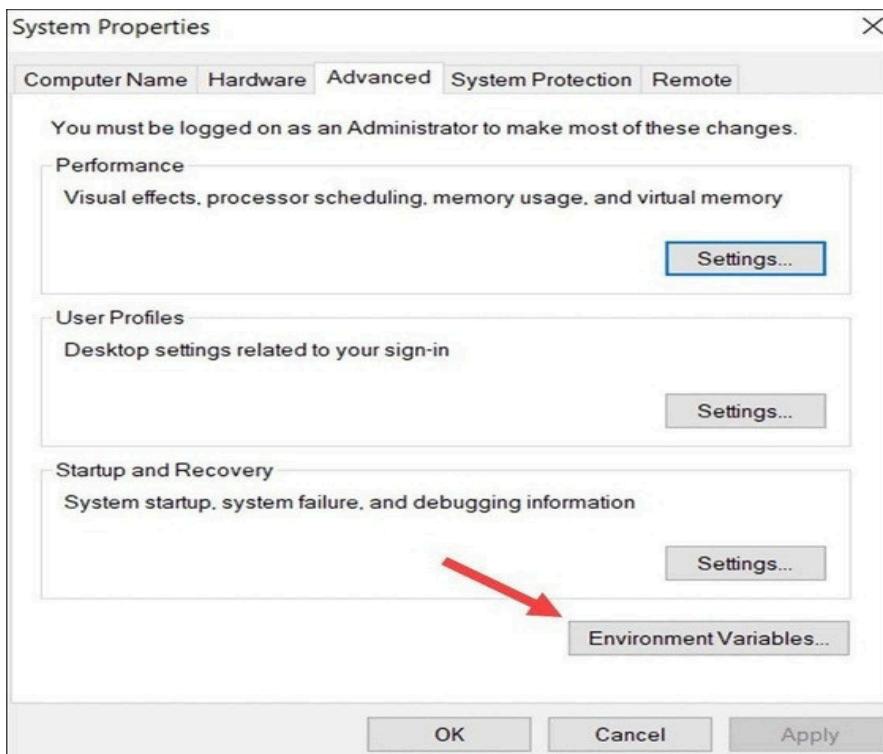
1. Navigate to **This PC>Properties**



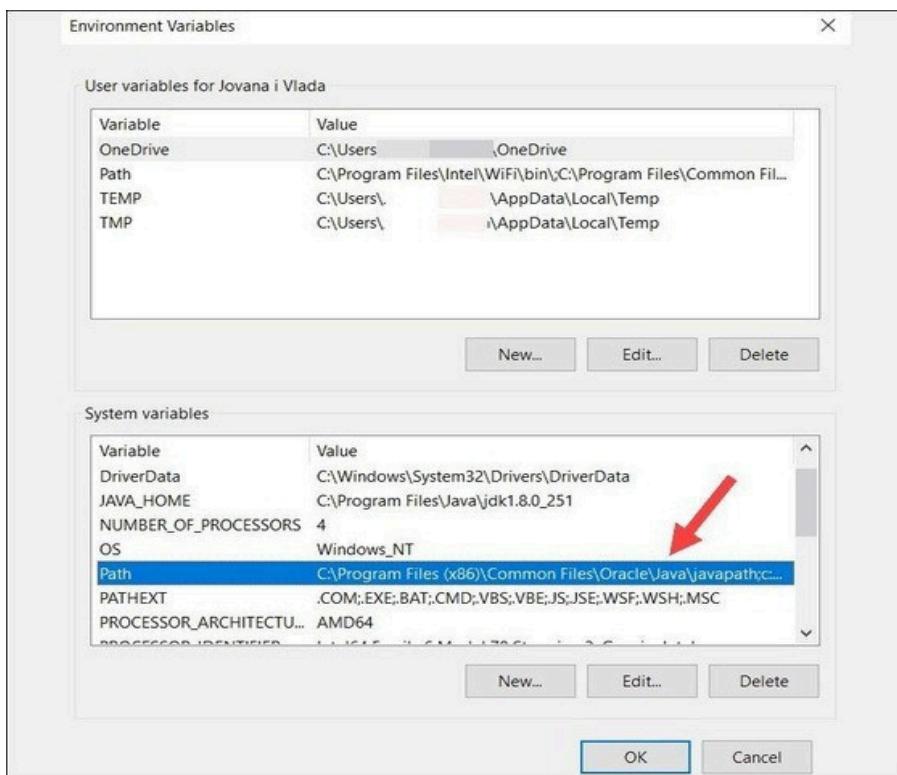
2. Select the **Advanced system settings** option.



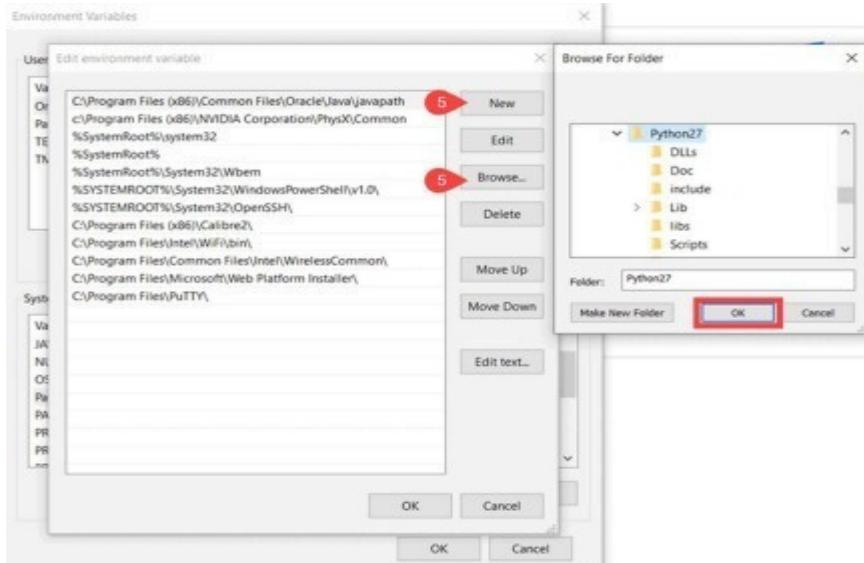
### 3. Click Environment Variables...



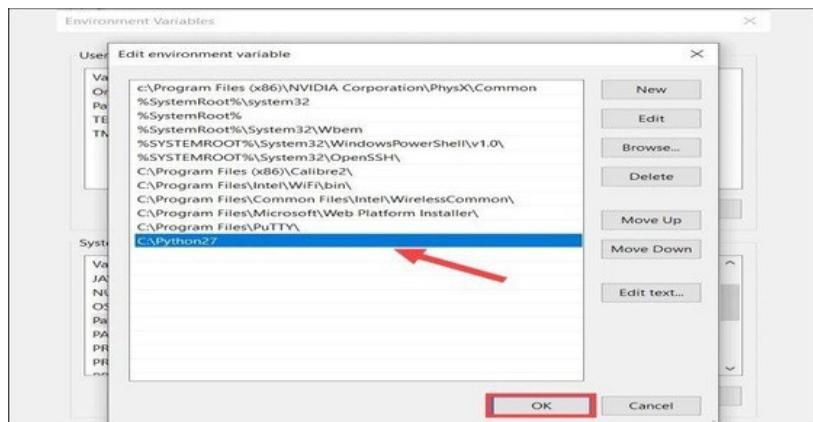
### 4. Double-click on the existing Path system variable.



5. Select **New** and then **Browse** to locate the Python installation folder quickly. Once you have confirmed that the path is correct, click **OK**.



6. Add the Python 2.7 path to the **Path** system variable by selecting **OK**.



#### Step 4: Download and Set Up Apache Cassandra

Download and Extract Cassandra tar.gz Folder

1. Visit the official Apache Cassandra Download page and select the version you would prefer to download. Currently, the latest available version 3.11.6

The screenshot shows the Apache Cassandra download page. The 'Latest version' section is highlighted with a red box, containing the text: 'Download the latest Apache Cassandra 3.11 release: 3.11.6 (pgp, sha256 and sha512), released on 2020-02-14.' Below this, the 'Older supported releases' section is also highlighted with a red box, listing previous versions: 'Apache Cassandra 3.0 is supported until 6 months after 4.0 release (date TBD). The latest release is 3.0.20 (pgp, sha256 and sha512), released on 2020-02-14.', 'Apache Cassandra 2.2 is supported until 4.0 release (date TBD). The latest release is 2.2.16 (pgp, sha256 and sha512), released on 2020-02-14.', and 'Apache Cassandra 2.1 is supported until 4.0 release (date TBD) with critical fixes only. The latest release is 2.1.21 (pgp, sha256 and sha512), released on 2019-02-11.'

2. Click the suggested **Mirror download link** to start the download process.

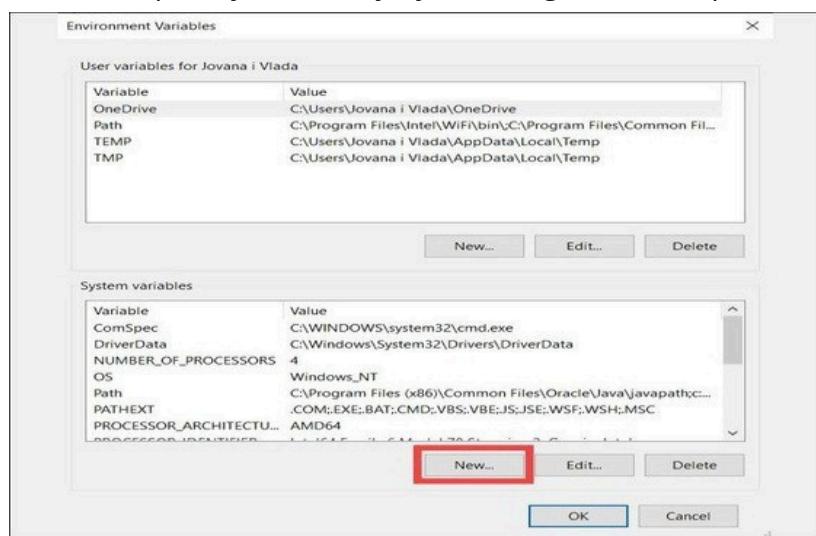
Unzip the compressed tar.gz folder using a compression tool such as 7-Zip or WinZip. In this example, the compressed folder was unzipped, and the content placed in the **C:Cassandraapache-cassandra-**

### 3.11.6 folder.

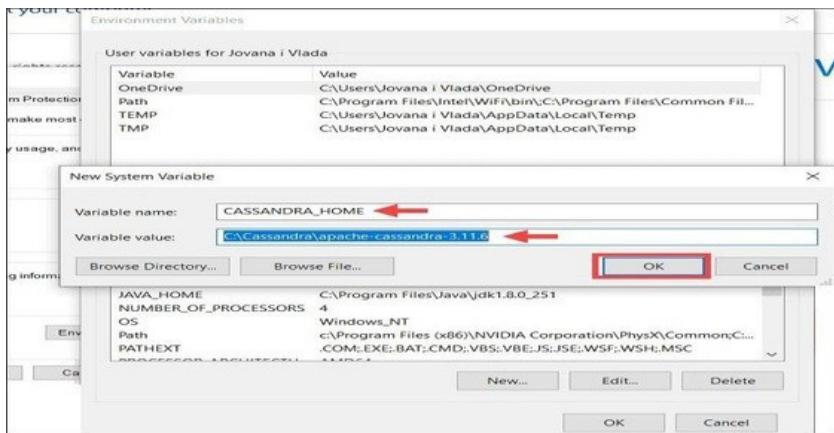
This PC > Local Disk (C:) > Cassandra > apache-cassandra-3.11.6			
	Name	Date modified	Type
Quick access	bin	5/15/2020 4:17 PM	File folder
Documents	conf	5/15/2020 4:17 PM	File folder
Downloads	data	5/15/2020 4:28 PM	File folder
Pictures	doc	5/15/2020 4:17 PM	File folder
Cassandra Slike	interface	5/15/2020 4:17 PM	File folder
Desktop	javadoc	5/15/2020 4:17 PM	File folder
Hadoop	lib	5/15/2020 4:18 PM	File folder
OneDrive	logs	5/15/2020 4:28 PM	File folder
This PC	pylib	5/15/2020 4:18 PM	File folder
3D Objects	tools	5/15/2020 4:18 PM	File folder
Desktop	.ToDelete	5/15/2020 4:28 PM	TODELETE File
Documents	CASSANDRA-14092	2/10/2020 11:57 PM	Text Document
Downloads	CHANGES	2/10/2020 11:57 PM	Text Document
Music	LICENSE	2/10/2020 11:57 PM	Text Document
Pictures	NEWS	2/10/2020 11:57 PM	Text Document
Videos	NOTICE	2/10/2020 11:57 PM	Text Document

### Configure Environment Variables for Cassandra

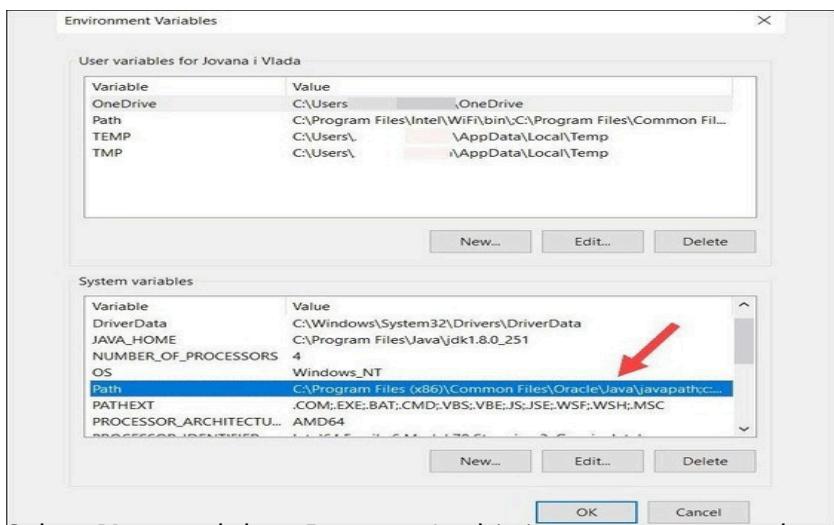
Add a completely new entry by selecting the **New** option.



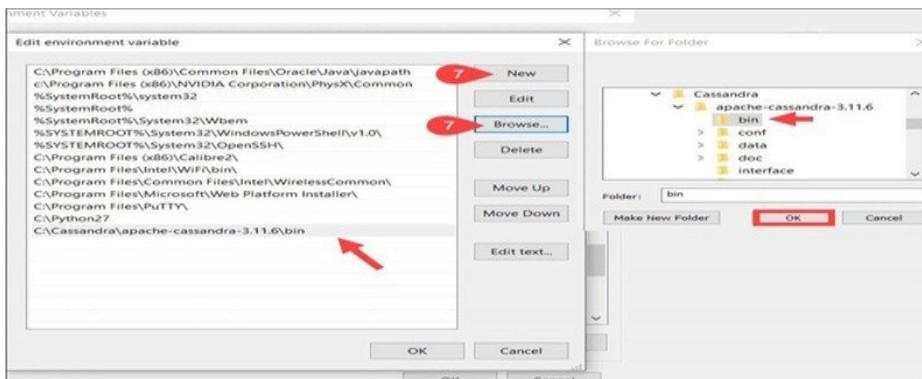
Type **CASSANDRA\_HOME** for Variable name, then for the Variable value column select the location of the unzipped **Apache Cassandra** folder. Based on the previous steps, the location is **C:Cassandraapache-cassandra-3.11.6**. Once you have confirmed that the location is correct, click **OK**.



Double click on the **Path** variable.



Select **New** and then **Browse**. In this instance, you need to add the full path to the **bin** folder located within the Apache Cassandra folder, **C:\Cassandra\apache-cassandra-3.11.6\bin**.



Hit the **OK** button and then again **OK** to save the edited variables.

## Step 5: Start Cassandra from Windows CMD

Navigate to the Cassandra bin folder. Start the Windows Command Prompt directly from within the bin folder by typing **cmd** in the address bar and pressing **Enter**.

Name	Date modified	Type	Size
cassandra	2/10/2020 11:57 PM	File	11 KB
cassandra	2/10/2020 11:57 PM	Windows Batch File	7 KB
cassandra.in.sh	2/10/2020 11:57 PM	Windows Batch File	4 KB
cassandra.in.sh	2/10/2020 11:57 PM	SH File	3 KB
cassandra	2/10/2020 11:57 PM	Windows PowerShell	13 KB
cqlsh	2/10/2020 11:57 PM	File	2 KB

Type the following command to start the Cassandra server: The system proceeds to start the Cassandra Server.

```
C:\Windows\System32\cmd.exe - cassandra
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Cassandra\apache-cassandra-3.11.6\bin>cassandra
WARNING! PowerShell script execution unavailable.
Please use 'powershell Set-ExecutionPolicy Unrestricted'
on this user-account to run cassandra with fully featured
functionality on this platform.
Starting with legacy startup options
Starting Cassandra Server
INFO [main] 2020-05-15 16:28:39,39 YamlConfigurationLoader.java:89 - Configuration location: file:/C:/Cassandra/apache-cassandra-3.11.6/conf/cassandra.yaml
INFO [main] 2020-05-15 16:28:42,652 Config.java:516 - Node configuration:[allocate_tokens_for_keyspace=null; authenticator=AllowAllAuthenticator; authorizer=AllowAllAuthorizer; auto_bootstrap=true; auto_snapshot=true; back_pressure_enabled=false; back_pressure_strategy=org.apache.cassandra.net.RateBasedBackPressure(high_ratio=0.9, factor=5, flow=FAST); batch_size_fail_threshold_in_kb=50; batch_size_warn_threshold_in_kb=5; batchlog_replay_throttle_in_kb=1024; broadcast_address=null; broadcast_rpc_address=null; buffer_pool_use_heap_if_exhausted=true; cas_contention_timeout_in_ms=1000; cdc_enab
```

Do not close the current cmd session.

## Step 6: Access Cassandra cqlsh from Windows CMD

While the initial command prompt is still running open a new command line prompt from the same bin folder. Enter the following command to access the Cassandra **cqlsh** bash shell:

You now have access to the Cassandra shell and can proceed to issue basic database commands to your Cassandra server.

```
C:\Windows\System32\cmd.exe - cqlsh
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Cassandra\apache-cassandra-3.11.6\bin>cqlsh
WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh>
```

You have successfully installed Cassandra on Windows.

## Selecting Keyspace for Cassandra Table

Before you start adding a table, you need to determine the keyspace where you want to create your table. There are two options to do this.

### Option 1: The USE Command

Run the USE command to select a keyspace to which all your commands will apply. To do that, in the cqlsh shell type:

USE keyspace\_name;

Then, you can start adding tables.

## **Option 2: Specify the Keyspace Name in the Query**

The second option is to specify the keyspace name in the query for table creation. The first part of the command, before column names and options, looks like this:

```
CREATE TABLE keyspace_name.table_name
```

This way, you immediately create a table in the keyspace you defined.

**Basic Syntax for Creating Cassandra Tables** Creating tables using SQL looks similar to SQL queries. In this section, we will show you the basic syntax for creating tables in Cassandra. The basic syntax for creating a table looks like this:

```
CREATE TABLE tableName (  
    columnName1 dataType,  
    columnName2 dataType,  
    columnName2 datatype  
    PRIMARY KEY (columnName)  
);
```

The following sections explain how to create tables with different types of primary keys. First, select a keyspace where you want to create a table. In our case:

```
USE businessinfo;
```

### **Create Table with Simple Primary Key**

The first example is a basic table with suppliers. The ID is unique for every supplier, and it will serve as the primary key. The SQL query looks like this:

```
CREATE TABLE  
suppliers ( supp_id int  
PRIMARY KEY,  
supp_city text,  
supp_email text,  
supp_fee int,  
supp_name text,  
supp_phone int  
);
```

To see if the table is in the keyspace, type in:

```
DESCRIBE TABLES;
```

## Practical No. 2

### Write the programs for Text Delimited CSV to HORUS format.

```
# Utility Start CSV to HORUS =====
import pandas as pd
# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1") print('Input
Data Values =====') print(InputData)
print('=====') # Processing Rules =====
ProcessData=InputData # Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True) ProcessData.drop('ISO-3-
CISoOd-eM',49axis=1,inplace=True)# Rename Country and
Pinrpolcaecses=DTarutae.)rename(columns={'CountryName'},
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True) #
Set new Index ProcessData.set_index('CountryNumber', inplace=True) # Sort
data by CurrencyNumber ProcessData.sort_values('CountryName', axis=0,
ascending=False,
inplace=True) print('Process Data Values
=====') print(ProcessData)
print('=====')
# Output Agreement =====
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False) print('CSV to HORUS -
Done') # Utility done =====
```

### Practical No. 3

#### Write the programs for XML to HORUS format

```
# Utility Start XML to HORUS ===== #
Standard                                         Tools
=====
import pandas as pd import xml.etree.ElementTree as ET header =
data.columns root = ET.Element('root') for row in range(data.shape[0]): entry =
ET.SubElement(root,'entry') for index in range(data.shape[1]):
schild=str(header[index]) child = ET.SubElement(entry, schild) if
str(data[schild][row]) != 'nan': child.text = str(data[schild][row]) else:
child.text = 'n/a'
entry.append(child) result =
ET.tostring(root) return
result def xml2df(xml:data):
root = ET.XML(xml:data)
all_records = [] for i, child in
enumerate(root): record = {} for
subchild in child:
record[subchild.tag] =
subchild.text
all_records.append(record) return
pd.DataFrame(all_records)
# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('=====')
print('Input Data Values =====')
print('=====')
print(InputData)
print('=====')
# Processing Rules =====
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE ProcessData.drop('ISO-2-
CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
```

```

# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False,
inplace=True)
print('=====')
print('Process Data Values =====')
print('=====')
print(ProcessData)
print('=====')

# Output Agreement =====
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====')
print('XML to HORUS - Done')
print('=====')

# Utility done =====

```

Output:

```

===== RESTART: C:\VKHCG\05-DS\9999-Data\XML2HORUS.py =====
=====
Input Data Values =====
=====

Squeezed text (707 lines). | =====

=====
Process Data Values =====
=====

CountryName
CountryNumber
716 Zimbabwe
894 Zambia
887 Yemen
732 Western Sahara
876 Wallis and Futuna Islands
...
16 American Samoa
12 Algeria
8 Albania
248 Aland Islands
4 Afghanistan

[247 rows x 1 columns]
=====

=====
XML to HORUS - Done
=====
```

## Practical No. 4

### Write the programs for Video to HORUS format.

```
# Utility Start Movie to HORUS (Part 1) =====
# Standard Tools
#=====
import os import shutil import cv2
#=====
sInputFileName='C:/VKHCG/05-DS/9999-Data/dog.mp4'
sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp' if
os.path.exists(sDataBaseDir):    shutil.rmtree(sDataBaseDir) if not
os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
print('=====')                         Movie          to          Frames')
print('Start')                         Movie          to          Frames')
print('=====')                         Movie          to          Frames')
vidcap = cv2.VideoCapture(sInputFileName) success,image = vidcap.read()
count = 0 while success:
    success,image = vidcap.read()    sFrame=sDataBaseDir + str('/dog-frame-' +
+ str(format(count, '04d')) + '.jpg')    print('Extracted: ', sFrame)
    cv2.imwrite(sFrame, image)      if os.path.getsize(sFrame) == 0:
        count += -1
    os.remove(sFrame)
    print('Removed: ', sFrame)

    if cv2.waitKey(10) == 27: # exit if Escape is
hit      break   if count > 15: # exit
break   count += 1

print('=====')                         Movie          to          Frames')
print('Generated : ', count, ' Frames')
print('=====')                         Movie          to          Frames')
print('Movie to Frames HORUS - Done')
print('=====')                         Movie          to          Frames')
# Utility done =====
```

```
>>>
= RESTART: C:\VKHCG\05-DS\9999-Data\MOVIE2HORUSFrame.py
=====
Start Movie to Frames
=====
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0000.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0001.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0002.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0003.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0004.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0005.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0006.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0007.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0008.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0009.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0010.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0011.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0012.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0013.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0014.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0015.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0016.jpg
=====
Generated : 16  Frames
=====
Movie to Frames HORUS - Done
=====
```

## Practical No. 5

### Write the programs for Audio to HORUS format.

```
# Utility Start Audio to HORUS ===== #
Standard Tools from scipy.io import wavfile import pandas as pd import
matplotlib.pyplot as plt import numpy as np def show_info(ename, a,r):
    print ('-----')
    print ("Audio:", ename)
    print ('-----') print ("Rate:", r)
    print ('-----') print ("shape:",
a.shape) print ("dtype:", a.dtype)
    print ("min, max:", a.min(), a.max())

    print ('-----')
    plot_info(ename, a,r) def
    plot_info(ename, a,r):
        sTitle= 'Signal Wave - '+ ename + ' at ' + str(r) + 'hz'
        plt.title(sTitle)

        sLegend=[] for c in
        range(a.shape[1]): sLabel =
        'Ch' + str(c+1)
        sLegend=sLegend+[str(c+1)
        ] plt.plot(a[:,c],
        label=sLabel)
        plt.legend(sLegend)
        plt.show()
sInputFileName='C:/VKHC
```

```

G/05-DS/9999-Data/2ch-
sound.wav'
print('=====
=====
=====')

print('Processing : ', sInputFileName)
print('=====')
InputRate, InputData = wavfile.read(sInputFileName) show_info("2
channel", InputData, InputRate) ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']

ProcessData.columns=sColumns OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('=====')

print('Processing : ', sInputFileName)
print('=====')
InputRate, InputData = wavfile.read(sInputFileName) show_info("4
channel", InputData, InputRate) ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']

ProcessData.columns=sColumns OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=====')

print('Processing : ', sInputFileName)
print('=====')
InputRate, InputData = wavfile.read(sInputFileName) show_info("6
channel", InputData, InputRate) ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']

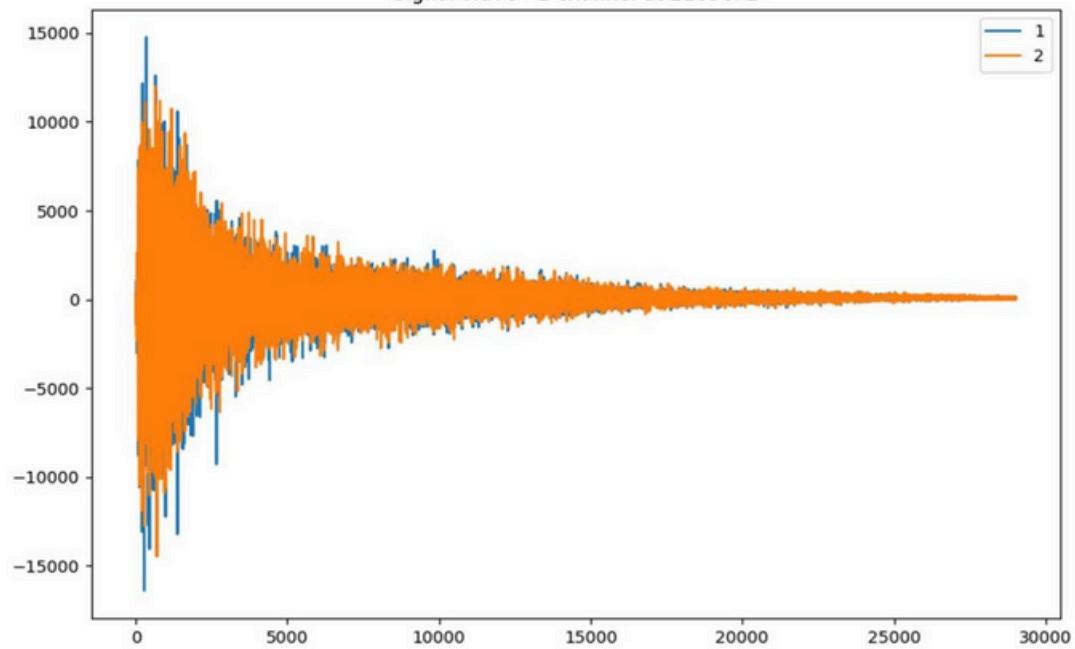
```

```
ProcessData.columns=sColumns OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='C:/VKHCG/05-DS/9999-Data/8ch-sound.wav'
print('=====')  
  
print('Processing : ', sInputFileName)
print('=====')  
InputRate, InputData = wavfile.read(sInputFileName) show_info("8
channel", InputData,InputRate) ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']  
  
ProcessData.columns=sColumns OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====')
print('Audio to HORUS - Done')
print('=====')  
  
# Utility done =====
```

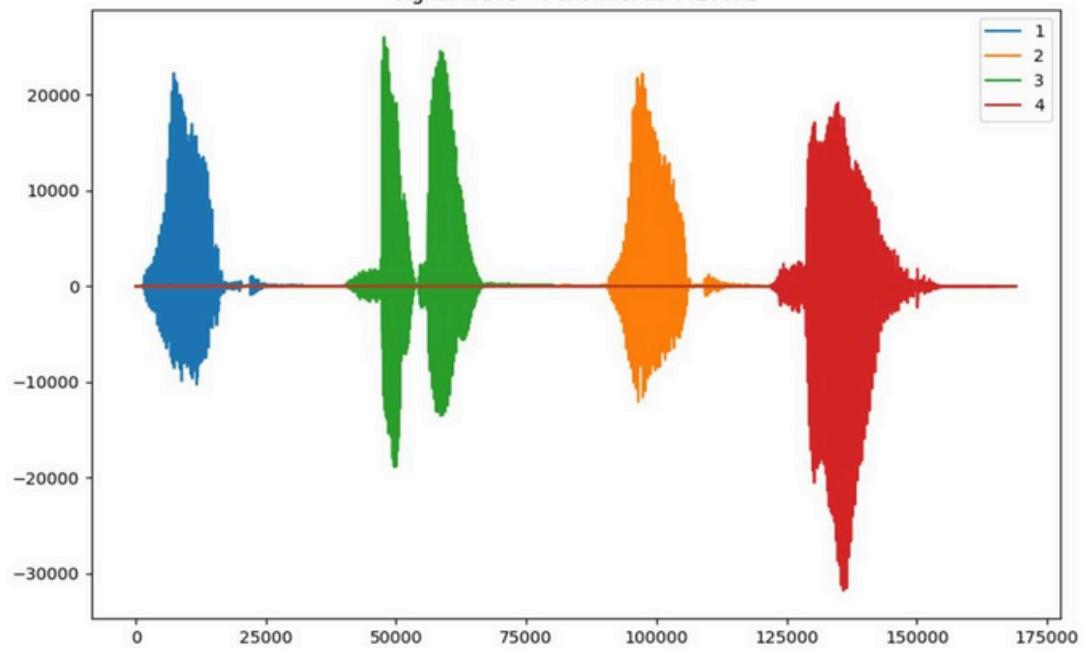
## Output:

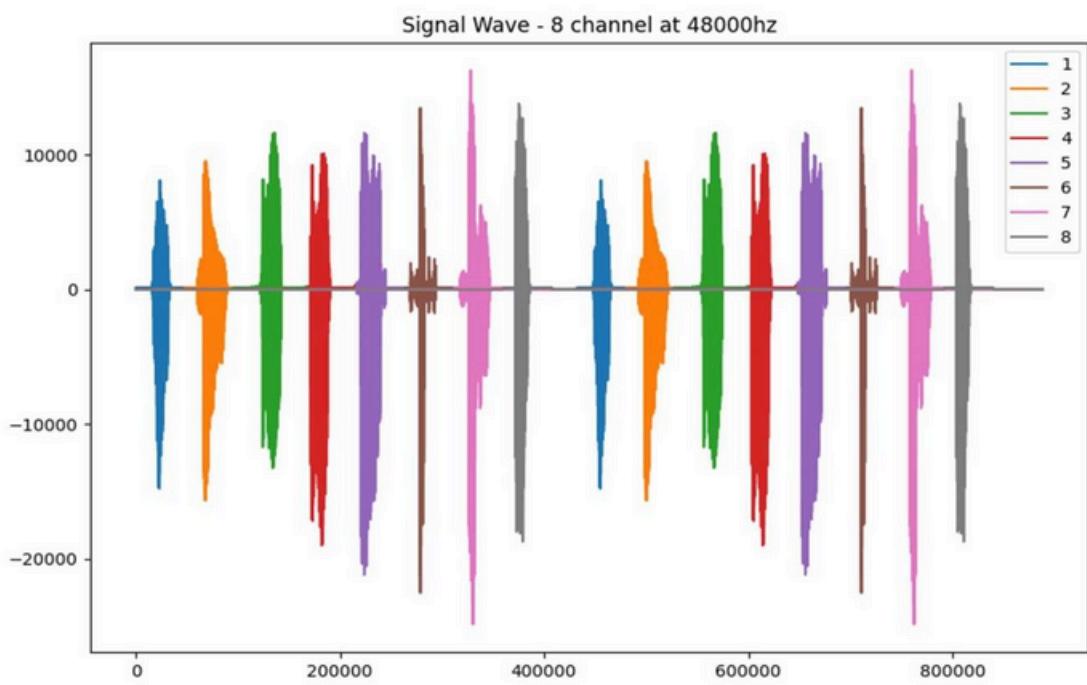
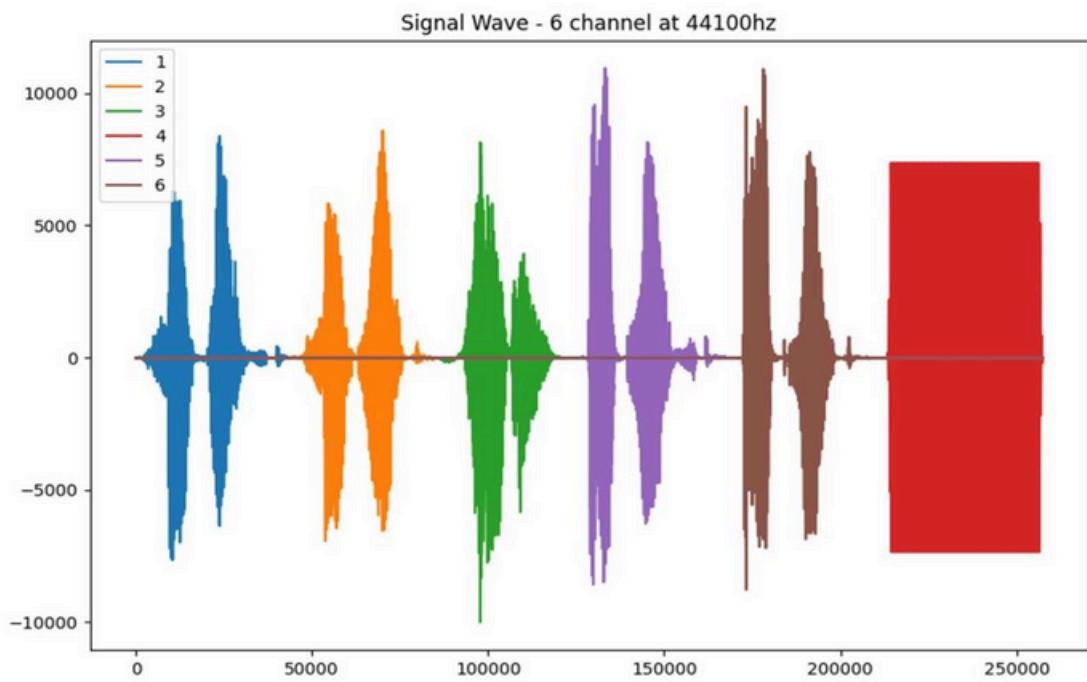
```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
=====
>>> ===== RESTART: Shell =====
>>> ===== RESTART: C:\V р KHCG\05-DS\9999-Data\AUDIO2HORUS.py =====
=====
Processing : C:/VKHCG/05-DS/9999-Data/2ch-sound.wav
=====
-----
Audio: 2 channel
-----
Rate: 22050
-----
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
-----
Processing : C:/VKHCG/05-DS/9999-Data/4ch-sound.wav
-----
-----
Audio: 4 channel
-----
Rate: 44100
-----
shape: (169031, 4)
dtype: int16
min, max: -31783 26018
-----
Processing : C:/VKHCG/05-DS/9999-Data/6ch-sound.wav
-----
-----
Audio: 6 channel
-----
Rate: 44100
-----
shape: (257411, 6)
dtype: int16
min, max: -10018 10957
-----
Processing : C:/VKHCG/05-DS/9999-Data/8ch-sound.wav
-----
-----
Audio: 8 channel
-----
Rate: 48000
-----
shape: (688000, 8)
dtype: int16
min, max: -24859 16303
-----
-----
Audio to HORUS - Done
=====
```

Signal Wave - 2 channel at 22050hz



Signal Wave - 4 channel at 44100hz





## Practical No. 6

### Write the programs for CSV to JSON format.

```
# Utility Start CSV to JSON =====
# Standard Tools
import pandas as pd
# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1") print('Input
Data Values =====') print(InputData)
print('=====')
# Processing Rules =====
ProcessData=InputData print('Process Data Values
=====') print(ProcessData)
print('=====')
# Output Agreement =====
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-
Data/Country_Code.json'
OutputData.to_json(sOutputFileName,orient='index') print('CSV to
JSON - Done')
# Utility done =====
```

Output:

```
===== RESTART: C:\VKHCG\05-DS\9999-Data\CSV2JSON.py =====
Input Data Values =====
    version https://git-lfs.github.com/spec/v1
0  oid sha256:b6bd794425c441ce48592e5931110771156...
1                               size 5901
=====
Process Data Values =====
    version https://git-lfs.github.com/spec/v1
0  oid sha256:b6bd794425c441ce48592e5931110771156...
1                               size 5901
=====
CSV to JSON - Done
```

## Practical No. 7

### Write the programs for CSV to XML format.

```
# Utility Start XML to HORUS =====
# Standard Tools import pandas
as pd import
xml.etree.ElementTree as ET def
df2xml(data):

    header = data.columns      root
= ET.Element('root')      for row in
range(data.shape[0]):

        entry = ET.SubElement(root,'entry')
for index in range(data.shape[1]):

            schild=str(header[index])
child = ET.SubElement(entry, schild)
if str(data[schild][row]) != 'nan':
child.text = str(data[schild][row])
else:
            child.text = 'n/a'
entry.append(child)
result = ET.tostring(root)
return result def
xml2df(xml_data):

    root = ET.XML(xml_data)
all_records = []      for i, child in
enumerate(root):

        record = {}      for
subchild in child:
            record[subchild.tag] =
subchild.text
all_records.append(record)      return
pd.DataFrame(all_records)

# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1") print('Input
Data Values =====')
print(InputData)
print('=====')

# Processing Rules =====
ProcessData=InputData
```

```
print('Process Data Values =====')
print(ProcessData)

print('=====') #
Output Agreement =====
sOutputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
sXML=df2xml(OutputData) file_out = open(sOutputFileName, 'wb')
file_out.write(sXML) file_out.close()

print('CSV to XML - Done')
# Utility done =====
```

Output:

```
===== RESTART: C:\VKHCG\05-DS\9999-Data\CSV2XML.py =====
Input Data Values =====
    version https://git-lfs.github.com/spec/v1
0  oid sha256:b6bd794425c441ce48592e5931110771156...
1                               size 5901
=====
Process Data Values =====
    version https://git-lfs.github.com/spec/v1
0  oid sha256:b6bd794425c441ce48592e5931110771156...
1                               size 5901
=====
CSV to XML - Done
```

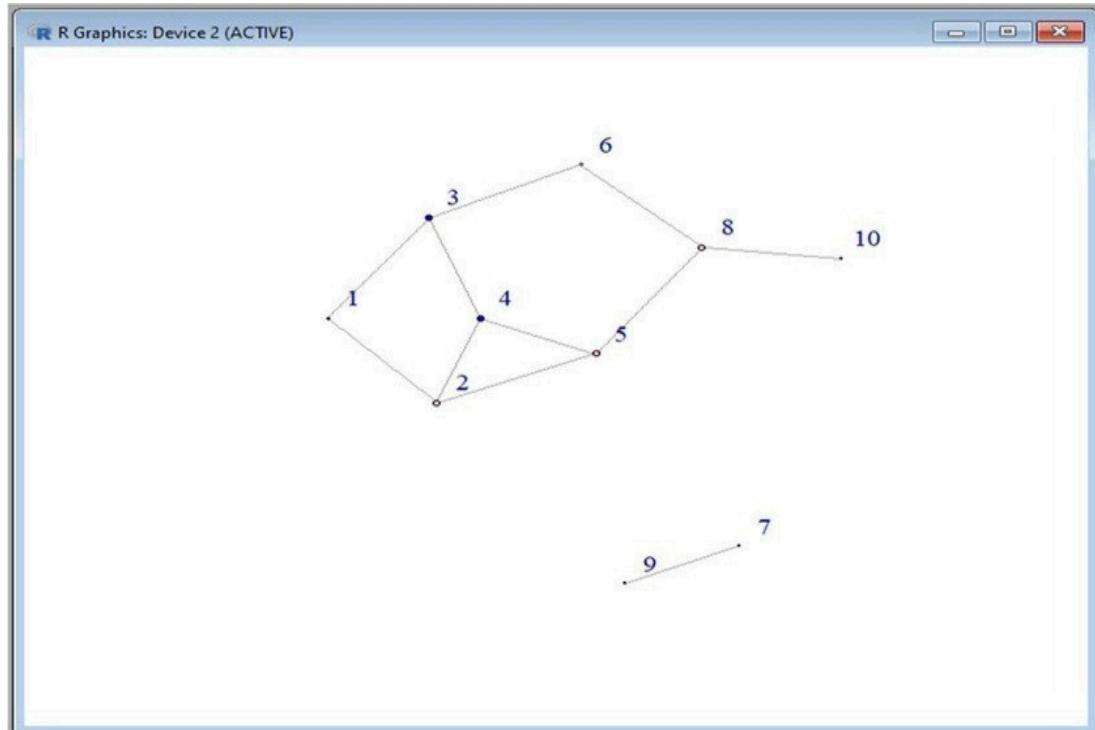
## Practical No. 8

**Write R program to create the network routing diagram from the given data on routers.**

### Solution:

**igraph package:** igraph is a library collection for creating and manipulating graphs and analyzing networks  
install.packages('igraph') library(igraph)

```
users <- data.frame(UserID = 1:10, Name = c("User1", "User2", "User3", "User4", "User5", "User6",  
"User7", "User8", "User9", "User10"), Gender = c("M", "F", "M", "M", "F", "F", "M", "F", "M", "M"))  
friendships <- data.frame(UserID1 = c(1, 1, 2, 2, 3, 3, 4, 5, 6, 7, 8), UserID2 = c(2, 3, 4, 5, 4, 6, 5,  
8, 8,  
9, 10)) graph <- graph.data.frame(friendships, directed = FALSE, vertices = users)  
V(graph)$color <- ifelse(V(graph)$Gender == "M", "blue", "pink") node_sizes <-  
degree(graph, mode = "all") layout <- layout_with_fr(graph) plot(graph, layout = layout,  
vertex.label.dist = 2, vertex.label.cex = 1.2, vertex.size = node_sizes)
```



## Practical No.9

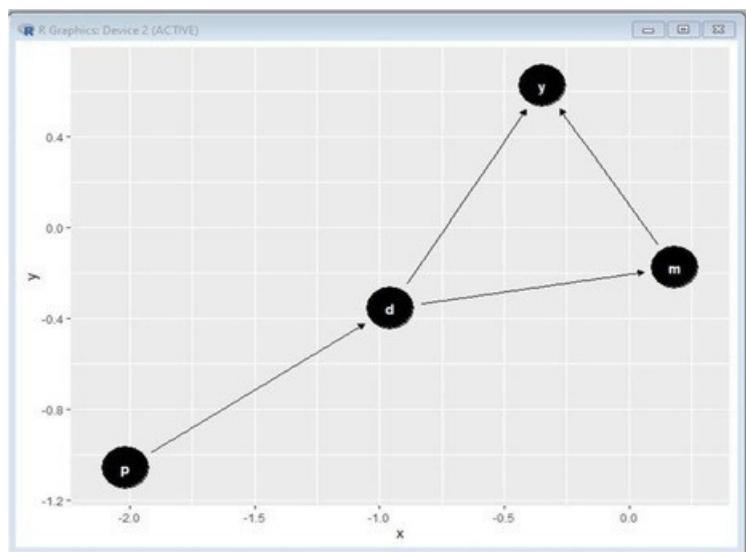
**Write R program to build acyclic graph.**

```
>install.packages("ggdag")
```

**Script 1:**

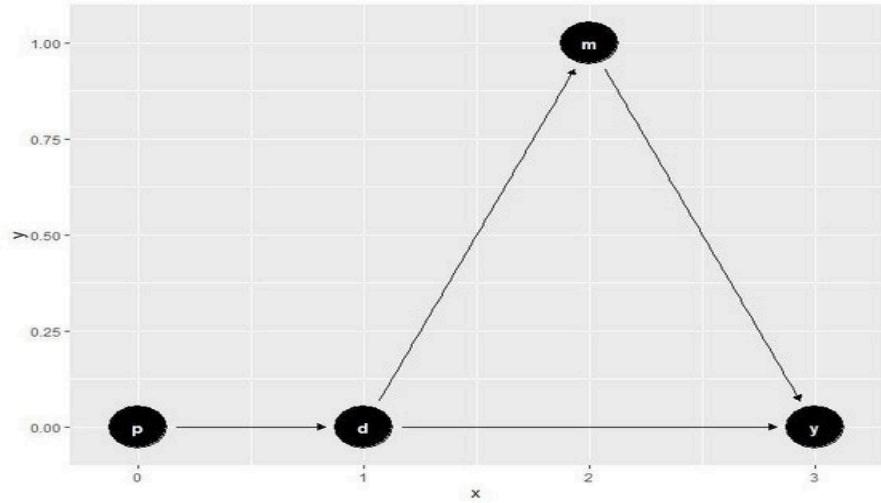
```
library(GGDAG) our_dag <- ggdag::dagify(d ~  
p,m ~ d,y ~ d,y ~ m) ggdag::ggdag(our_dag)
```

**Output:**



**Script 2:**

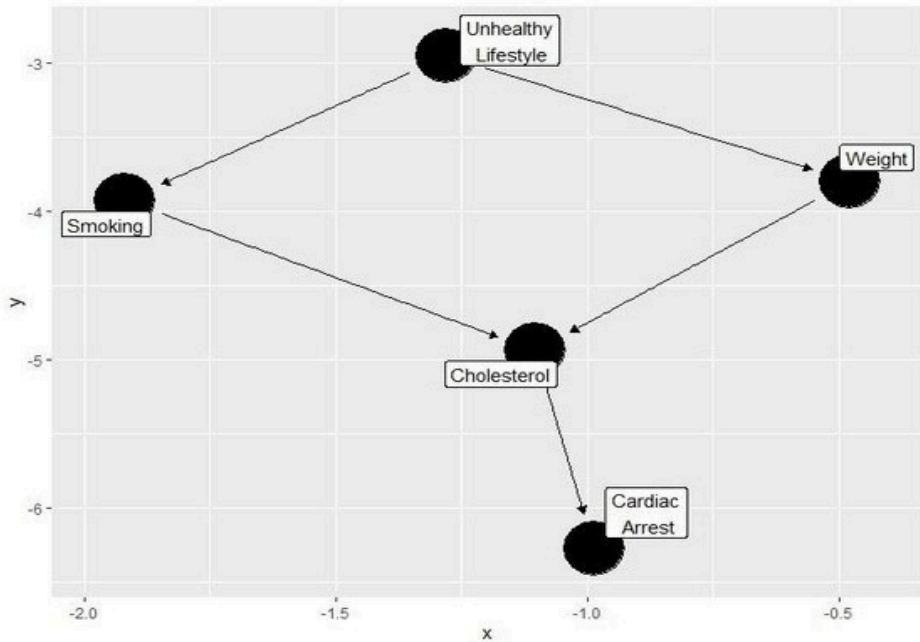
```
library(GGDAG)  
coord_dag <- list( x = c(p = 0, d = 1, m = 2, y = 3), y = c(p = 0, d = 0, m = 1, y  
= 0)) our_dag <- ggdag::dagify(d ~ p, m ~ d, y ~ d, y ~ m, coords =  
coord_dag) ggdag::ggdag(our_dag) theme_void()
```



### Script 3:

```

smoking_ca_dag <- ggdag::dagify(cardiacarrest ~ cholesterol, cholesterol ~ smoking + weight, smoking ~
unhealthy, weight ~ unhealthy, labels = c("cardiacarrest" = "Cardiac\n Arrest", "smoking" =
"Smoking", "cholesterol" = "Cholesterol", "unhealthy" = "Unhealthy\n Lifestyle", "weight" = "Weight"))
ggdag::ggdag(smoking_ca_dag, text = FALSE, use_labels = "label") theme_void()
  
```



## Practical No. 10

**Write R program to generate payroll from the given data.**

```
# Define Employee details
employees <- data.frame(
  Name = c("Ajay", "Vijay"),
  Basic_Salary = c(30000, 35000),
  HRA_Percentage = c(30, 40),
  PF_Percentage = c(12, 10)
)
# Calculate HRA for each employee employees$HRA <-
(employees$HRA_Percentage / 100) * employees$Basic_Salary
# Calculate PF for each employee employees$PF <-
(employees$PF_Percentage / 100) * employees$Basic_Salary
# Calculate Total Salary for each employee employees$Total_Salary <-
employees$Basic_Salary + employees$HRA - employees$PF
# Print Payroll print("Payroll
for Employees:")
print(employees)
```

```
> print(employees)
  Name Basic_Salary HRA_Percentage PF_Percentage    HRA    PF Total_Salary
1 John Doe      30000          30            12   9000  3600      35400
2 Jane Smith     35000          40            10  14000  3500      45500
```

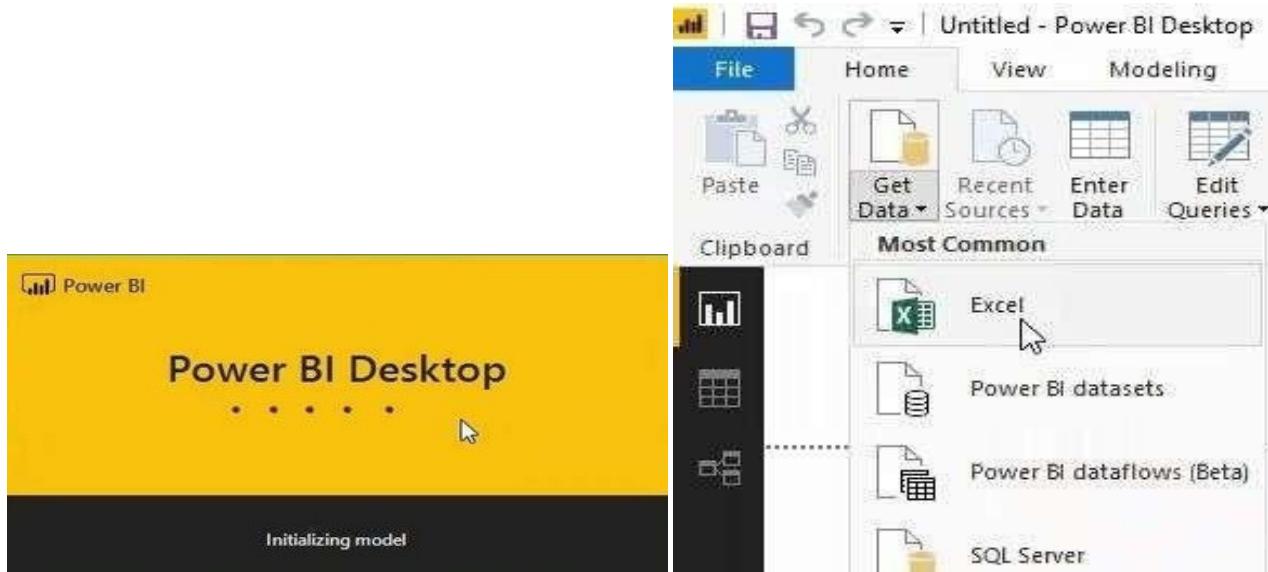
## Practical No 11

### Import data from MS-Excel/ MS-Access in Power BI Desktop Show data Visualization

Open MS-Excel and create sample data as given.

pid	pname	qty	price
101	HardDisk	2	3500
102	Mouse	3	650
103	KeyBoard	2	550
104	DDR3-RAM	1	1200
105	LED_TV	1	6500
106	WI-FI	2	3000

Now open Powe BI. Click on "Get Data" from "Home" Tab. Select "Excel" from menu.



Browse sample data file and click on open button. Connecting and loading procced. Available spreadsheet will appeared select datasheet which we want to load.

The screenshot shows the Power BI Desktop interface. On the left, the Navigator pane lists 'SQLServer\_to\_Excel.xlsx [2]' and 'Table\_DESKTOP\_DBCU8QH\_SQLEXPRESS...'. A checkbox next to 'SQL\_Data' is checked. To the right, a table titled 'SQL\_Data' is displayed with the following data:

pid	pname	qty	price
101	HardDisk	2	3500
102	Mouse	3	650
103	KeyBoard	2	550
104	DDR3-RAM	1	1200
105	LED_TV	1	6500
106	Wi-Fi	2	3000

Select Filels name requied to load or appeare in visualization.

The screenshot shows a bar chart with categories like 'HardDisk', 'Mouse', 'KeyBoard', 'DDR3-RAM', 'LED\_TV', and 'Wi-Fi'. The chart is based on the 'SQL\_Data' table. To the right, the 'FIELDS' pane is open, showing the selected fields: 'pid' and 'pname'. Below the pane, a preview of the 'SQL\_Data' table is shown with the same data as above.

pid	pname	qty	price
101	HardDisk	2	3500
102	Mouse	3	650
103	KeyBoard	2	550
104	DDR3-RAM	1	1200
105	LED_TV	1	6500
106	Wi-Fi	2	3000

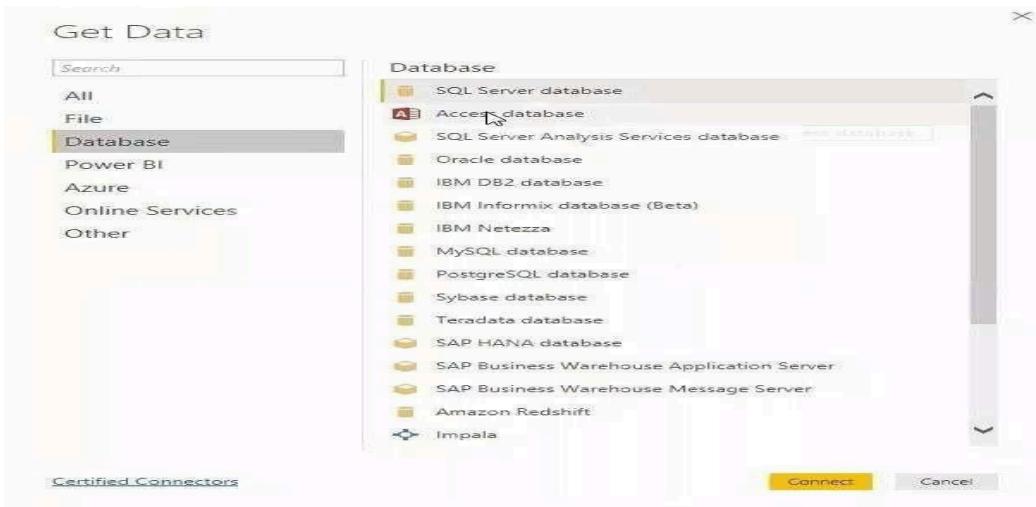
## Import data from MS-Access in Power BI Desktop Show data Visualization.

Open MS-Access, Create emp table in database with sample data as

The screenshot shows the 'emp' table in Microsoft Access. The table has columns: empid, ename, salary, and dept. The data is as follows:

empid	ename	salary	dept
101	Shravan	35000	1
102	Mokshada	32000	1
103	Komal	25000	3
104	Manoj	22000	2
105	Gopal	21500	3
*	0	0	

Now open Power BI, Click on Get Data and select Access Database Option from list



Browse MS-Access Database and open it. Choose table name and load it .

empid	ename	salary	dept
101	Shreavan	35000	1
102	Mokshada	32000	1
103	Komal	25000	3
104	Manoj	22000	2
105	Gopal	21500	3