



**Rayat Shikshan Sanstha's
KARMAVEER BHauraO PATIL COLLEGE,VASHI
(Empowered Autonomous)**



Reaccredited NAAC with Grade 'A++' (CGPA3.51)|ISO 9001:2015 Certified Institute 'Best College' Award by University of Mumbai

[DEPARTMENT OF INFORMATION TECHNOLOGY]

CERTIFICATE

This is to certify that Mr./Miss: SHUBHAM VIKAS NAVALE
student of Class M S C I T from **Karmaveer Bhaurao Patil College, Vashi, Navi Mumbai** has satisfactorily completed the practical course in subject **Data Science**. As per the syllabus laid by the college during the Academic Year **2024-25**.

ROLL NO.:240609

EXAM NO.:240609

Date: __/__/2025

Namrata Bagal

Course Coordinator

MADHURI GABHANE

Head, Department of IT

External Examiner

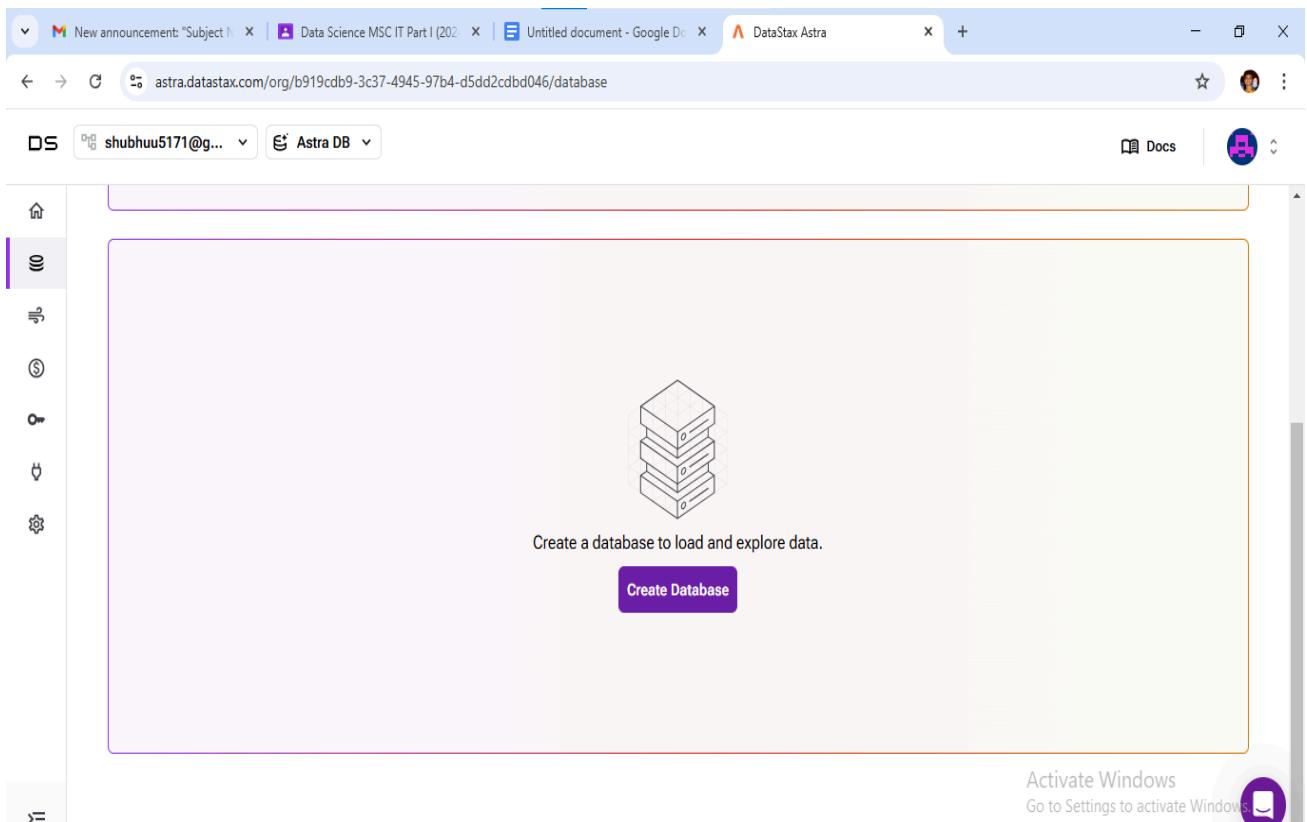
INDEX

SR.NO.		PRACTICAL NAME	DATE	SIGN
01		Creating data models using Cassandra [datastax Astra].		
02	A	Conversion from different formats to HORUS format.CSV to HORUS		
	B	Conversion from different formats CSV to AUDIO		
03		Bad Data: Removing leading or lagging spaces from string		
04		Assessing the Data		
05		Transforming the image		
06		Perform to deal with Missing Values in Pandas.		
07		Import order data from an OData feed.		

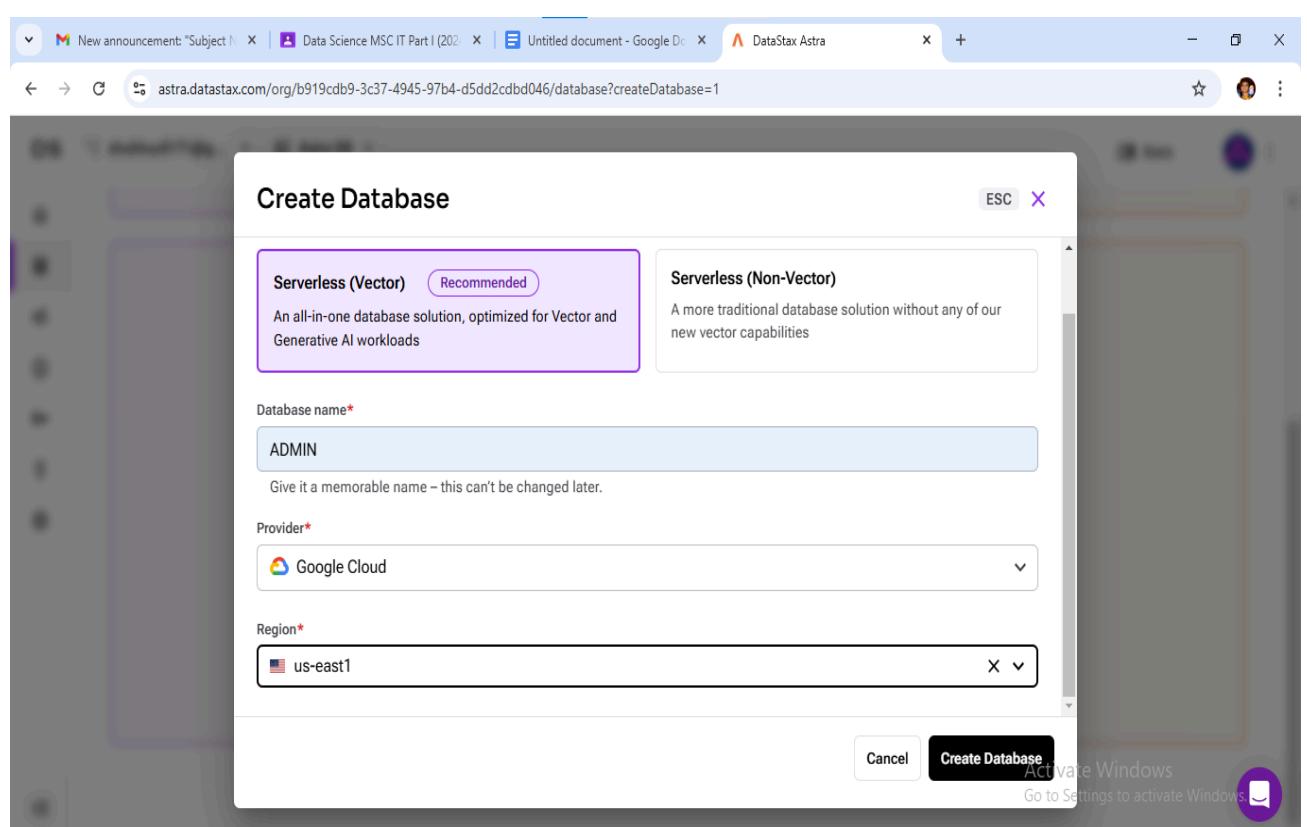
Practical No. 1

Aim: - Creating data models using Cassandra [datastax Astra].

- Create Database .



- Database name-> Provider Google Cloud -> Region-> us-east1.



- From Dashboard Select -> ADMIN.

The screenshot shows the DataStax Astra Admin interface. The left sidebar has a tree view with 'ADMIN' selected. The main area shows 'Database Details' with an API Endpoint (us-east1 https://39de5869-15a1-479b-bc73-f4a2c2b603a2), Application Tokens (Generate Token), and Region (Google Cloud us-east1). A 'Get started' section with a 'Load Data' button is also present.

- From ADMIN -> Data Explorer -> Keyspace.

The screenshot shows the DataStax Astra Admin interface with the 'Data Explorer' tab selected in the sidebar. It displays a 'Keyspace' dropdown set to 'default_keyspace' and a 'Create collection' section with 'Create Sample Collection' and 'Create Empty Collection' buttons. A 'Create collection' section with instructions is also visible.

- Select Keyspace -> ds

The screenshot shows the DataStax Astra Data Explorer interface. The top navigation bar includes tabs for YouTube, Home - Google Drive, Data Science MSC IT, Untitled document, Data Science - Google, DataStax Astra, and another DataStax Astra tab. The URL in the address bar is astra.datastax.com/org/b919cdb9-3c37-4945-97b4-d5dd2cd046/database/39de5869-15a1-479b-bc73-f4a2c2b603a2/data-explorer.

The left sidebar has a purple header labeled "ADMIN". It contains links for Home, Databases (with "ds" highlighted), Streaming, Billing, Tokens, Integrations, and Settings. A "Command Palette" button and a "Free Plan" button are also present.

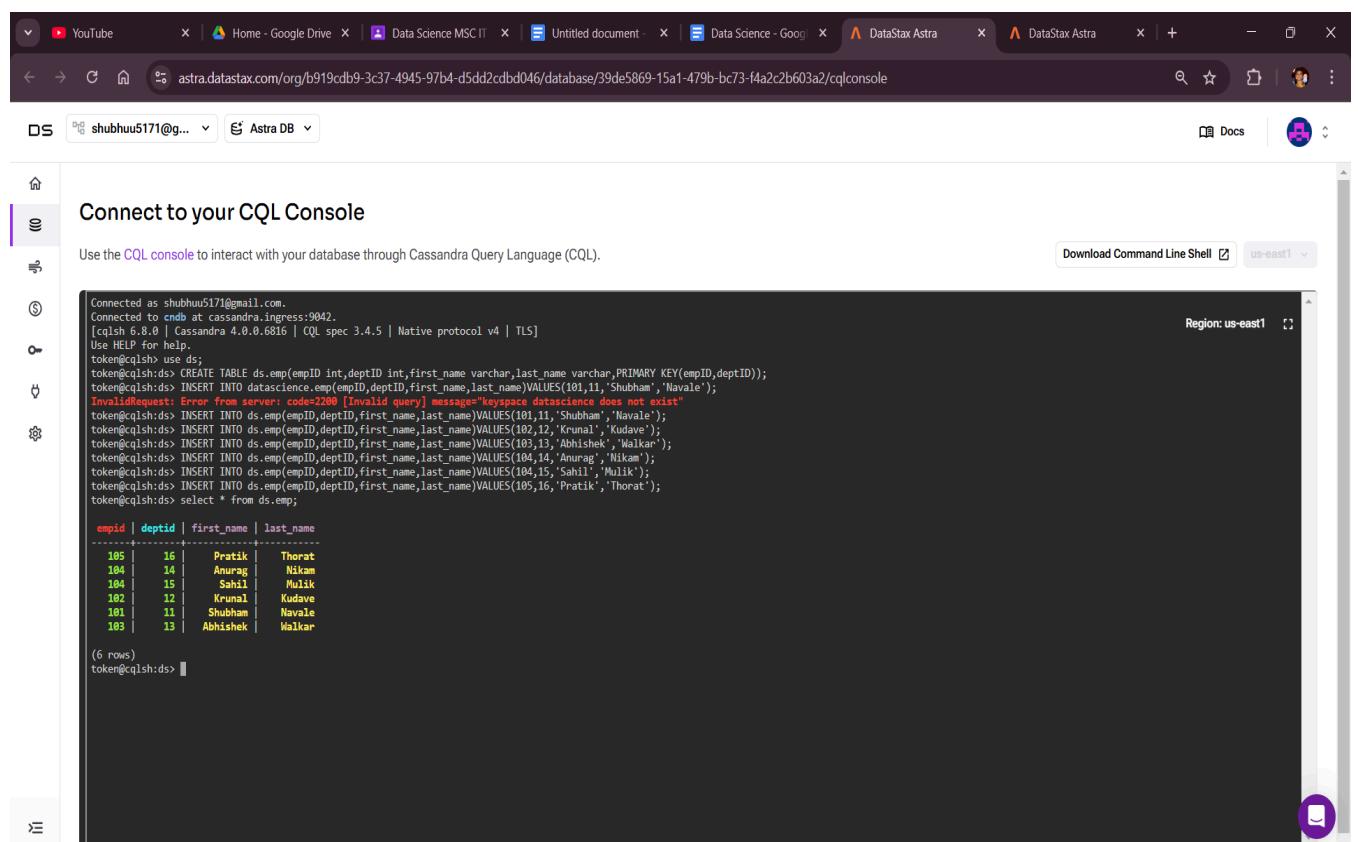
The main dashboard shows the "ADMIN" database with the "ds" keyspace selected. The "Data Explorer" tab is active. The "Collections" section shows a "Create Collection" button. A large orange callout box highlights the "Create collection" section, which includes a stack of folder icons, a "Create collection" button, and a sub-instruction: "Create a sample collection pre-loaded with data, create an empty collection, or [use our clients](#)". Below this are "Create Sample Collection" and "Create Empty Collection" buttons.

At the bottom right of the interface is a purple circular icon with a white message icon.

- Select -> CQL (Cassandra Query Language) after creating a keyspace.

1A) Create a TABLE EMPLOYEE and perform operations on it.

- CREATE TABLE dataservice.emp(empID int,deptID int,first_name varchar,last_name varchar,PRIMARY KEY(empID,deptID));
- INSERT INTO ds.emp(empID,deptID,first_name,last_name) VALUES(101,11,'Shubham','Navale');
- INSERT INTO ds.emp(empID,deptID,first_name,last_name) VALUES(102,12,'Krunal','Kudave');
- INSERT INTO ds.emp(empID,deptID,first_name,last_name) VALUES(103,13,'Abhishek','Walkar');
- INSERT INTO ds.emp(empID,deptID,first_name,last_name) VALUES(104,14,'Anurag','Nikam');
- INSERT INTO ds.emp(empID,deptID,first_name,last_name) VALUES(104,15,'Sahil','Mulik');
- INSERT INTO ds.emp(empID,deptID,first_name,last_name) VALUES(105,16,'Pratik','Thorat');
- select * from ds.emp;



The screenshot shows the DataStax Astra CQL console interface. The browser tab is 'astradatastax.com/org/b919cdb9-3c37-4945-97b4-d5dd2cdbd046/database/39de5869-15a1-479b-bc73-f4a2c2b603a2/cqlconsole'. The interface includes a navigation bar with 'Docs' and a user icon, and a sidebar with icons for home, connect, and regions. The main area displays a CQL console with the following content:

```

Connected as shubhuu5171@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4 | TLS]
Use HELP for help.
token@cqlsh:ds> use ds;
token@cqlsh:ds> CREATE TABLE ds.emp(empID int,deptID int,first_name varchar,last_name varchar,PRIMARY KEY(empID,deptID));
token@cqlsh:ds> INSERT INTO dataservice.emp(empID,deptID,first_name,last_name)VALUES(101,11,'Shubham','Navale');
InvalidRequest: Error from server: code=200 [Invalid query] message="Keyspace dataservice does not exist"
token@cqlsh:ds> INSERT INTO ds.emp(empID,deptID,first_name,last_name)VALUES(101,11,'Shubham','Navale');
token@cqlsh:ds> INSERT INTO ds.emp(empID,deptID,first_name,last_name)VALUES(102,12,'Krunal','Kudave');
token@cqlsh:ds> INSERT INTO ds.emp(empID,deptID,first_name,last_name)VALUES(103,13,'Abhishek','Walkar');
token@cqlsh:ds> INSERT INTO ds.emp(empID,deptID,first_name,last_name)VALUES(104,14,'Anurag','Nikam');
token@cqlsh:ds> INSERT INTO ds.emp(empID,deptID,first_name,last_name)VALUES(104,15,'Sahil','Mulik');
token@cqlsh:ds> INSERT INTO ds.emp(empID,deptID,first_name,last_name)VALUES(105,16,'Pratik','Thorat');
token@cqlsh:ds> select * from ds.emp;

```

empid	deptid	first_name	last_name
105	16	Pratik	Thorat
104	14	Anurag	Nikam
104	15	Sahil	Mulik
102	12	Krunal	Kudave
101	11	Shubham	Navale
103	13	Abhishek	Walkar

(6 rows)

- **Query For DELETING one ROW: -**
- DELETE FROM ds.emp where empID=105;
- Select * From ds.emp;

The screenshot shows a browser window with multiple tabs open, including YouTube, Google Drive, Data Science MSC IT, Untitled document, Data Science - Google, DataStax Astra, and another DataStax Astra tab. The main content area is the DataStax Astra CQL console.

Connect to your CQL Console

Use the [CQL console](#) to interact with your database through Cassandra Query Language (CQL).

Connected as shubhuhu5171@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | Cassandra 4.0.0.6016 | CQL spec 3.4.5 | Native protocol v4 | TLS]
Use HELP for help.
token@cqlsh> select * from ds.emp;

empid	deptid	first_name	last_name
105	16	Pratik	Thorat
104	14	Anurag	Nikam
104	15	Sahil	Mulik
102	12	Krunal	Kudave
101	11	Shubham	Navale
103	13	Abhishek	Walkar

(6 rows)

token@cqlsh> DELETE FROM ds.emp where empID=105;
token@cqlsh> select * from ds.emp;

empid	deptid	first_name	last_name
104	14	Anurag	Nikam
104	15	Sahil	Mulik
102	12	Krunal	Kudave
101	11	Shubham	Navale
103	13	Abhishek	Walkar

(5 rows)

token@cqlsh>

1B) Create a TABLE STUDENT and perform operations on it.

- **Query for CREATING TABLE student: -**

```
CREATE TABLE ds.stud(studID int, deptID int, first_name varchar,  
last_name varchar, year_of_joining int, PRIMARY KEY(studID, deptID));
```

- **Query for INSERTING VALUES into student table: -**

```
INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)  
VALUES (101,1,'Abhishek','Walkar', 2025);
```

```
INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)  
VALUES (101,2,'Shubham','Navale', 2025);
```

```
INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)  
VALUES (103,3,'AK','koli', 2025);
```

```
INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)  
VALUES (104,4,'Pratik','Kadam', 2025);
```

```
INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)  
VALUES (105,5,'sumit','patil', 2025);
```

- **Query For displaying the TABLE STUDENT: -**

```
SELECT * FROM ds1.stud;
```

The screenshot shows a browser window with multiple tabs open, including YouTube, Google Drive, Data Science MSC IT, Untitled document, Data Science - Google, DataStax Astra, and DataStax Astra. The main content area is a CQL console for DataStax Astra. It displays the following session output:

```
Connected as shubhuu5171@gmail.com.  
Connected to cndb at cassandra.ingress:9042.  
[cqlsh 6.8.0 | Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4 | TLS]  
Use HELP for help.  
token@cqlsh> use ds1;  
token@cqlsh:ds1> CREATE TABLE ds1.stud(studID int, deptID int, first_name varchar, last_name varchar, year_of_joining int, PRIMARY KEY(studID, deptID));  
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (101,1,'Abhishek','Walkar', 2025);  
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (101,2,'Shubham','Navale', 2025);  
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (103,3,'AK','koli', 2025);  
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (104,4,'Pratik','Kadam', 2025);  
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (105,5,'sumit','patil', 2025);  
token@cqlsh:ds1> select * from ds1.stud;
```

Below the command history, a table is displayed with the following data:

studID	deptid	first_name	last_name	year_of_joining
105	5	sumit	patil	2025
104	4	Pratik	Kadam	2025
101	1	Abhishek	Walkar	2025
101	2	Shubham	Navale	2025
103	3	AK	koli	2025

(5 rows)

- **Query For displaying the TABLE STUDENT: -**

```
SELECT * FROM ds1.stud;
```

The screenshot shows a browser window with multiple tabs open, including YouTube, Google Drive, Data Science MSC II, Untitled document, Data Science - Google, DataStax Astra, and another DataStax Astra tab. The main content area is a CQL console titled "Astra DB".

The console displays the following text:

```
Connected as shubhuu5171@gmail.com.
Connected to cndo at cassandra.ingress:9042.
[cqlsh 6.8.0] Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4 | TLS
Use HELP for help.
token@cqlsh> use ds1;
token@cqlsh:ds1> CREATE TABLE ds1.stud(studID int, deptID int, first_name varchar, last_name varchar, year_of_joining int, PRIMARY KEY(studID, deptID));
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (101,1,'Abhishek','Walkar', 2025);
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (101,2,'Shubham','Navale', 2025);
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (103,3,'AK','koli', 2025);
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (104,4,'Pratik','Kadam', 2025);
token@cqlsh:ds1> INSERT INTO ds1.stud(studID, deptID, first_name, last_name, year_of_joining)VALUES (105,5,'sumit','patil', 2025);
token@cqlsh:ds1> select * from ds1.stud;

studid | deptid | first_name | last_name | year_of_joining
-----+-----+-----+-----+-----+
  105 |     5 |    sumit |    patil |      2025
  104 |     4 |   Pratik |    Kadam |      2025
  101 |     1 | Abhishek |   Walkar |      2025
  101 |     2 |   Shubham |   Navale |      2025
  103 |     3 |        AK |      koli |      2025

(5 rows)
token@cqlsh:ds1> DELETE FROM ds1.stud where studID=105;
token@cqlsh:ds1> select * from ds1.stud;

studid | deptid | first_name | last_name | year_of_joining
-----+-----+-----+-----+-----+
  104 |     4 |   Pratik |    Kadam |      2025
  101 |     1 | Abhishek |   Walkar |      2025
  101 |     2 |   Shubham |   Navale |      2025
  103 |     3 |        AK |      koli |      2025

(4 rows)
```

Region: us-east1

Practical No. 2

Aim: Conversion from different formats to HORUS format.CSV

✓ A. CSV to HORUS

```
import pandas as pd

from datetime import datetime

df=pd.read_csv("Country_code.csv", encoding="UTF-8")

print(df)



|     | Country                   | ISO-2-CODE | ISO-3-Code | ISO-M49 |
|-----|---------------------------|------------|------------|---------|
| 0   | Afghanistan               | AF         | AFG        | 4       |
| 1   | Aland Islands             | AX         | ALA        | 248     |
| 2   | Albania                   | AL         | ALB        | 8       |
| 3   | Algeria                   | DZ         | DZA        | 12      |
| 4   | American Samoa            | AS         | ASM        | 16      |
| ..  | ...                       | ...        | ...        | ...     |
| 242 | Wallis and Futuna Islands | WF         | WLF        | 876     |
| 243 | Western Sahara            | EH         | ESH        | 732     |
| 244 | Yemen                     | YE         | YEM        | 887     |
| 245 | Zambia                    | ZM         | ZMB        | 894     |
| 246 | Zimbabwe                  | ZW         | ZWE        | 716     |



[247 rows x 4 columns]


```

```
processdata=df
```

```
processdata.drop("ISO-2-CODE", axis=1)
```

```


|     | Country                   | ISO-3-Code | ISO-M49 |
|-----|---------------------------|------------|---------|
| 0   | Afghanistan               | AFG        | 4       |
| 1   | Aland Islands             | ALA        | 248     |
| 2   | Albania                   | ALB        | 8       |
| 3   | Algeria                   | DZA        | 12      |
| 4   | American Samoa            | ASM        | 16      |
| ..  | ...                       | ...        | ...     |
| 242 | Wallis and Futuna Islands | WLF        | 876     |
| 243 | Western Sahara            | ESH        | 732     |
| 244 | Yemen                     | YEM        | 887     |
| 245 | Zambia                    | ZMB        | 894     |
| 246 | Zimbabwe                  | ZWE        | 716     |



247 rows x 3 columns


```

```
processdata.drop("ISO-2-CODE", axis=1, inplace=True)
```

```
processdata.drop("ISO-3-Code", axis=1, inplace=True)
```

```
processdata
```

	Country	ISO-M49
0	Afghanistan	4
1	Aland Islands	248
2	Albania	8
3	Algeria	12
4	American Samoa	16
...
242	Wallis and Futuna Islands	876
243	Western Sahara	732
244	Yemen	887
245	Zambia	894
246	Zimbabwe	716

247 rows × 2 columns

```
new=datetime.now()
```

```
new
```

```
datetime.datetime(2025, 3, 4, 16, 21, 2, 589485)
```

```
dt_string=new.strftime("%d-%m-%y %H:%M:%S")
print("Date and Time: ", dt_string)
```

```
Date and Time: 04-03-25 16:21:02
```

```
f=open("CountryCode.txt", "a")
f.write("Deleting column activity recorded ")
f.write(dt_string)
f.close()
```

```
processdata.rename(columns={"Country": "CountryName"}, inplace=True)
```

```
processdata
```

	CountryName	ISO-M49
0	Afghanistan	4
1	Aland Islands	248
2	Albania	8
3	Algeria	12
4	American Samoa	16
...
242	Wallis and Futuna Islands	876
243	Western Sahara	732
244	Yemen	887
245	Zambia	894
246	Zimbabwe	716

247 rows × 2 columns

```
processdata.rename(columns={"ISO-M49": "Code"}, inplace=True)
processdata
```

	CountryName	Code
0	Afghanistan	4
1	Aland Islands	248
2	Albania	8
3	Algeria	12
4	American Samoa	16
...
242	Wallis and Futuna Islands	876
243	Western Sahara	732
244	Yemen	887
245	Zambia	894
246	Zimbabwe	716

247 rows × 2 columns

```
processdata.set_index("Code", inplace=True)
```

processdata

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

247 rows × 1 columns

```
processdata.sort_values("CountryName", axis=0, ascending=False, inplace=True)
```

processdata

	CountryName
Code	
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 × 1 columns

Start coding or [generate](#) with AI.

```
OutputFileName="Hours_Country_Code.csv"  
processdata.to_csv(OutputFileName, index=False)
```

Practical No. 2

Aim: Conversion from different formats to HORUS format.CSV

✓ B. CSV to AUDIO

```
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

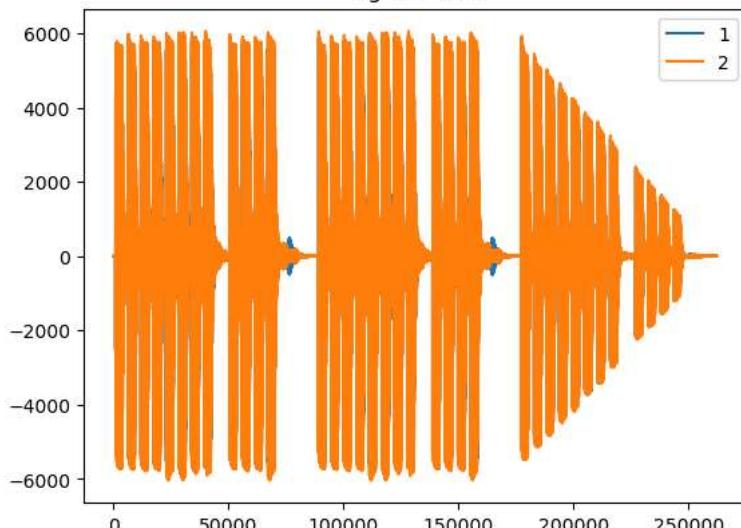
def show_info(aname, a, r):
    print("audio", aname)
    print("rate:", r)
    print("shape:", a.shape)
    print("dtype:", a.dtype)
    print("min,max:", a.min(), a.max())

def plot_info(aname, a, r):
    stitle = 'Signal Wave-' + aname + "at" + str(r) + "hz"
    plt.title("Signal Wave")
    slegend = []
    # Iterate through the channels of the audio data
    for i in range(a.shape[1]):
        slabel = "CH" + str(i + 1)
        slegend = slegend + [str(i + 1)]
        plt.plot(a[:, i], label=slabel)
    plt.legend(slegend)
    plt.show()

wavfile1 = "C:/Users/KBP/Desktop/ds/sample.wav"
print("Processing:", wavfile1)
inputRate, inputData = wavfile.read(wavfile1)
show_info('2 Channel', inputData, inputRate)
plot_info('2 Channel', inputData, inputRate) # Call plot_info with inputData
processdata = pd.DataFrame(inputData)
scolumns = ["Ch1", "Ch2"]

→ Processing: C:/Users/KBP/Desktop/ds/sample.wav
audio 2 Channel
rate: 44100
shape: (262094, 2)
dtype: int16
min,max: -6038 6056
C:/Users/KBP/AppData/Local/Temp/ipykernel_6764/3627020737.py:27: WavFileWarning: Chunk (non-data) not understood, skipping it.
inputRate, inputData = wavfile.read(wavfile1)
```

Signal Wave



```
processdata.columns=scolumns  
outputData=processdata  
outputData
```

	Ch1	Ch2
0	0	0
1	0	0
2	0	0
3	0	0
4	1	-1
...
262089	-3	-1
262090	3	2
262091	-2	-2
262092	0	2
262093	2	-2

262094 rows × 2 columns

```
outputfiledata="Horus-Audio.csv"  
outputData.to_csv(outputfiledata,index=False)  
print("Audio to Horus done")
```

→ Audio to Horus done

Practical No. 3

✓ Aim: Bad Data: Removing leading or lagging spaces from string

```
import string
import datetime as dt
print('#1 Removing leading or lagging spaces from a data entry');
baddata = " Data Science with too many spaces is bad!!! "
baddata

→ #1 Removing leading or lagging spaces from a data entry
' Data Science with too many spaces is bad!!! '

baddata.lstrip()

→ 'Data Science with too many spaces is bad!!! ' 

baddata.rstrip()

→ ' Data Science with too many spaces is bad!!! ' 

baddata

→ ' Data Science with too many spaces is bad!!! ' 

baddata.strip()

→ 'Data Science with too many spaces is bad!!! '
```

2 Removing nonprintable characters from a data entry

```
print('#2 Removing nonprintable characters from a data entry')
printable = set(string.printable)
baddata = "Data\x00Science with\x02 funny"
baddata
#2 Removing nonprintable characters from a data entry 'Data\x00Science with\x02 funny'
cleandata=".join(filter(lambda x:x in string.printable,baddata))"
print('Bad Data:',baddata);

→ #2 Removing nonprintable characters from a data entry
Bad Data: Data Science with funny
```

3 Reformatting data entry to match specific formatting criteria.

```
# Convert YYYY/MM/DD to DD Month YYYY
print('# 3 Reformatting data entry to match specific formatting criteria.')
baddate= dt.date(2019, 10, 31)
baddata= format(baddate,'%Y-%m-%d')
baddata

→ # 3 Reformatting data entry to match specific formatting criteria.
'2019 10 31'

gooddate= dt.datetime.strptime(baddata, '%Y-%m-%d')
gooddata= format(gooddate,'%d %B %Y')
print('Bad Data:',baddata)
print('Good Data:',gooddata)

→ Bad Data: 2019-10-31
Good Data: 31 October 2019
```

✓ Practical No. 4

✓ Aim: Assessing the Data.

```
import pandas as pd
```

✓ A) Create a Data frame using Python.

```
data = {"Name": ["SHUBHAM", "SAHIL", "KRUNAL"], "Age": [21, 21, 20], "std": ("MSc", "MSc", "TY")}  
df = pd.DataFrame(data)
```

	Name	Age	std	
0	SHUBHAM	21	MSc	
1	SAHIL	21	MSc	
2	KRUNAL	20	TY	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ B) INSERT COLUMN NAME, AGE, ROLL NO USING PANDAS

```
df.insert(3, "Roll no", ["Ghansoli", "Vashi", "Sanpada"])#insert row and columns  
df
```

	Name	Age	std	Roll no	
0	SHUBHAM	21	MSc	Ghansoli	
1	SAHIL	21	MSc	Vashi	
2	KRUNAL	20	TY	Sanpada	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ C) Rename the column name

```
df = df.rename(columns={"Roll no": "Address"})#rename the column name
```

```
df
```

	Name	Age	std	
0	SHUBHAM	21	MSc	
1	SAHIL	21	MSc	
2	KRUNAL	20	TY	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ D) Delete the column

```
df = df.drop(columns=["std"])#To delete the column  
df
```

	Name	Age	Address	
0	SHUBHAM	21	Ghansoli	
1	SAHIL	21	Vashi	
2	KRUNAL	20	Sanpada	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ E) Update any value using index.

```
df.loc[1,"Address"]="UP">#update any value using index  
df
```

	Name	Age	Address	
0	SHUBHAM	21	Ghansoli	
1	SAHIL	21	UP	
2	KRUNAL	20	Sandada	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
df.loc[2,"Age"]=19#update any value using index  
df
```

	Name	Age	Address	
0	SHUBHAM	21	Ghansoli	
1	SAHIL	21	UP	
2	KRUNAL	19	Sandada	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ F) To delete the row

```
df = df.drop(index=2)#To delete the row  
df
```

	Name	Age	Address	
0	SHUBHAM	21	Ghansoli	
1	SAHIL	21	UP	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ G) To get particular values of any columns

```
df.get("Name")#To get particular values of any columns
```

	Name
0	SHUBHAM
1	SAHIL

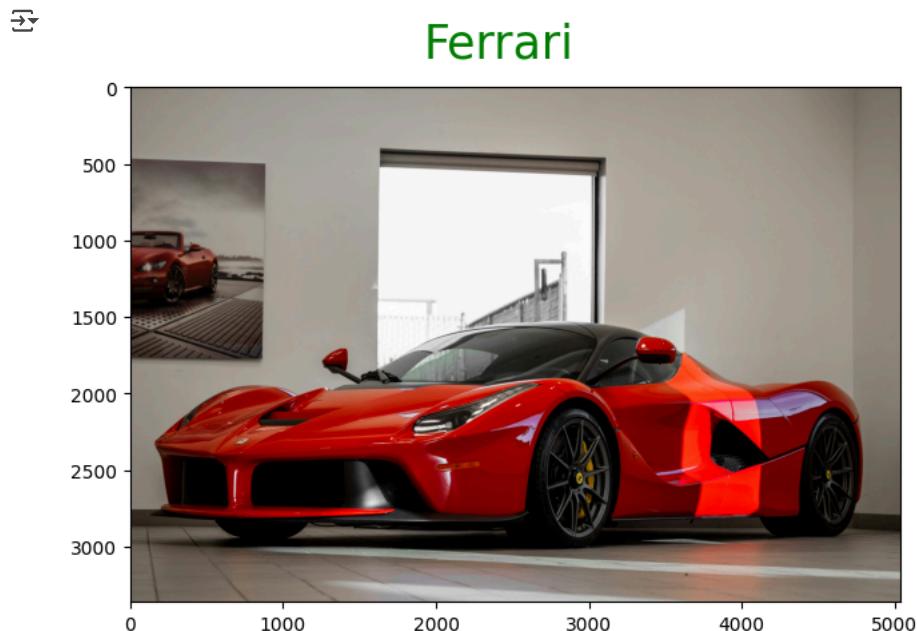
Practical No. 5

✓ Aim: Conversion from different formats to HORUS format.CSV

```
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

img='/content/FERRARI.jpg'
imageIn=Image.open(img)

fig1=plt.figure(figsize=(10,5))
fig1.suptitle('Ferrari', fontsize=25, color="green")
imgplot=plt.imshow(imageIn)
plt.show()
```



```
imageWidth,imageHeight = imageIn.size
imageMatrix=np.asarray(imageIn)
pixelsCnt=(imageWidth * imageHeight)
print('Pixel:', pixelsCnt)
print('Size:', imageWidth, 'X', imageHeight)
print(imageMatrix)
```

→ Pixel: 16942801
Size: 5041 X 3361
[[[126 123 118]
 [130 127 122]
 [136 133 128]
 ...
 [116 108 97]
 [117 109 98]
 [120 112 101]]

 [[[122 119 114]
 [138 135 130]
 [140 137 132]
 ...
 [122 114 103]
 [120 112 101]
 [118 110 99]]]

 [[[138 135 130]
 [134 131 126]
 [134 131 126]
 ...
 [117 109 98]
 [124 116 105]
 [123 115 104]]]
 ...

```
[[106 98 87]
 [106 98 87]
 [110 102 91]
```

...

```
[127 119 108]
[127 119 108]
[131 123 112]]
```

```
[[103 95 84]
 [111 103 92]
 [119 111 100]
```

...

```
[128 120 109]
[130 122 111]
[136 128 117]]
```

```
[[110 102 91]
 [112 104 93]
 [120 112 101]
```

...

```
[142 132 122]
[135 125 115]
[129 121 110]]]
```

Practical No:6

✓ Aim: Perform to deal with Missing Values in Pandas

```
import pandas as pd
df=pd.read_csv('/content/Book1.csv')
df
```

	Name	AI	DS	MA	Total
0	Sahil	56.0	NaN	76.0	44.000000
1	Krunal	69.0	53.0	NaN	40.666667
2	Vivek	56.0	55.0	56.0	55.666667
3	NaN	45.0	NaN	NaN	15.000000
4	Sandesh	63.0	55.0	23.0	47.000000
5	NaN	61.0	45.0	NaN	35.333333
6	Megha	56.0	NaN	53.0	36.333333
7	Pallavi	NaN	45.0	47.0	30.666667

```
df.info()
```

```
→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   Name     6 non-null      object  
 1   AI       7 non-null      float64 
 2   DS       5 non-null      float64 
 3   MA       5 non-null      float64 
 4   Total    8 non-null      float64 
dtypes: float64(4), object(1)
memory usage: 448.0+ bytes
```

1. Drop the column where any of the element is missing values.

```
df.dropna(how='any',axis=0)
```

	Name	AI	DS	MA	Total
2	Vivek	56.0	55.0	56.0	55.666667
4	Sandesh	63.0	55.0	23.0	47.000000

```
df.dropna(how='any',axis=1)
```

	Total
0	44.000000
1	40.666667
2	55.666667
3	15.000000
4	47.000000
5	35.333333
6	36.333333
7	30.666667

2. Drop the column where all element are missing values

```
df.dropna(how='all',axis=0)
```

⤵

	Name	AI	DS	MA	Total	
0	Sahil	56.0	NaN	76.0	44.000000	
1	Krunal	69.0	53.0	NaN	40.666667	
2	Vivek	56.0	55.0	56.0	55.666667	
3	NaN	45.0	NaN	NaN	15.000000	
4	Sandesh	63.0	55.0	23.0	47.000000	
5	NaN	61.0	45.0	NaN	35.333333	
6	Megha	56.0	NaN	53.0	36.333333	
7	Pallavi	NaN	45.0	47.0	30.666667	

⤶ ⤷

Practical No: 7

Aim: Import order data from an OData feed.

You import data into power BI Desktop from the sample northwind odata feed at the following

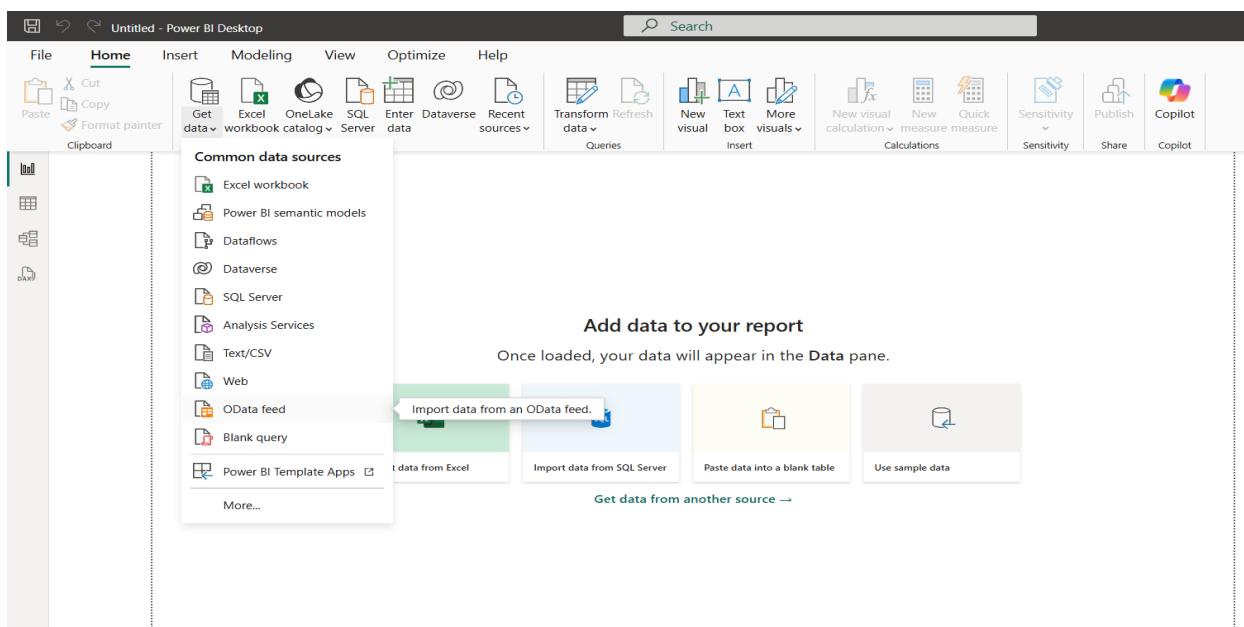
URL which you can copy (and then paste) in the steps below:

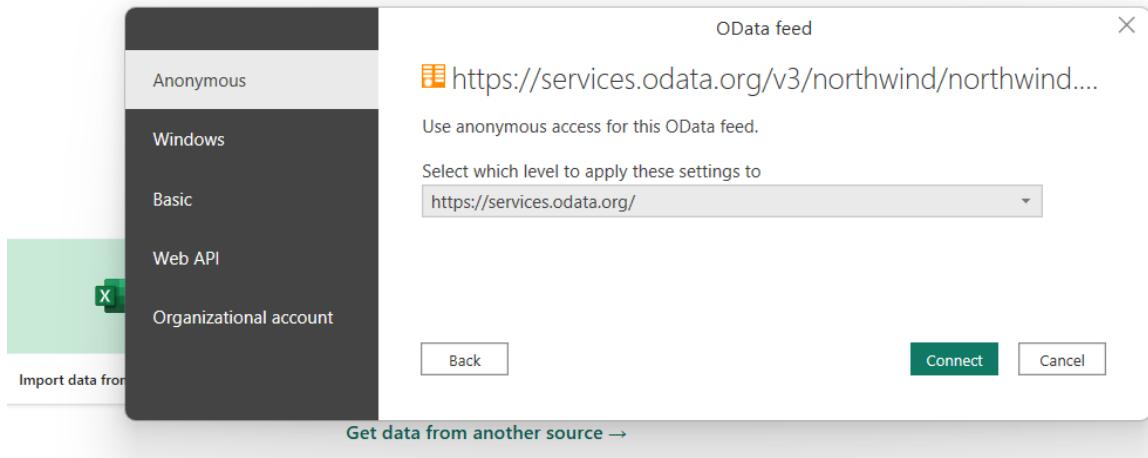
<https://services.odata.org/v3/northwind/northwind.svc/>

STEP1:

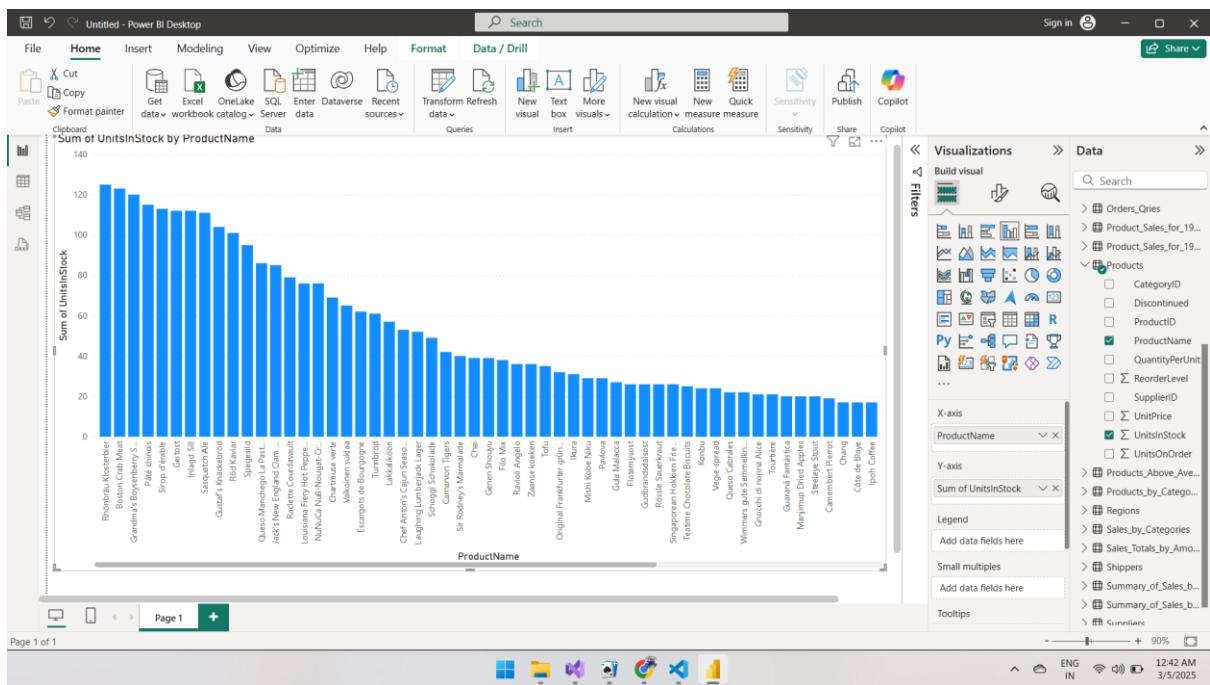
Connect to an Odata feed:

1. From the Home ribbon tab in query editor, select get data.
2. Browse the odata feed data source
3. In the odata feed dialog box paste the URL for the Northwind OData feed.
4. Select Ok.





1: Create charts showing Units in Stock by Product and Total Sales by Year



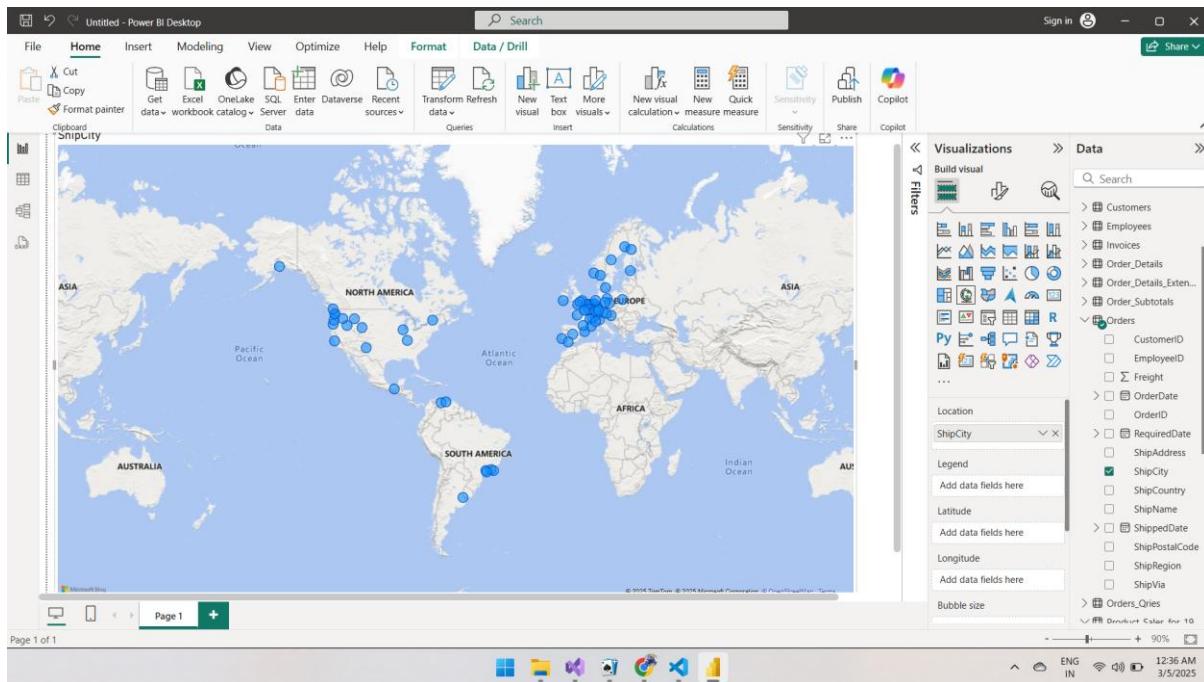
2. Drag OrderDate to the canvas beneath in the first chart, then drag line total (again, from the fields, pane) on to the visual, then select line chart. The following visualization created.

The screenshot shows two windows of Power BI Desktop. The top window displays a line chart titled "Sum of Quantity and Sum of UnitPrice by ProductID". The chart has "ProductID" on the X-axis (ranging from 0 to 80) and "Sum of Quantity and Sum of UnitPrice" on the Y-axis (ranging from 0K to 6K). The chart contains two data series: "Sum of Quantity" (blue line) and "Sum of UnitPrice" (red line). The bottom window shows a table view of product data with columns including ProductID, ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued, and CategoryName. A "Column tools" ribbon is visible above the table, and a "Data" pane on the right side lists various data models and tables.

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	CategoryName
1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	False	Beverages
2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	False	Beverages
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14.00	111	0	15	False	Beverages
35	Steelye Stout	16	1	24 - 12 oz bottles	18.00	20	0	15	False	Beverages
38	Côte de Blaye	18	1	12 - 75 cl bottle	263.50	17	0	15	False	Beverages
39	Chartreuse verte	18	1	1750 cc per bottle	18.00	69	0	5	False	Beverages
43	Ipoh Coffee	20	1	16 - 500 g tins	46.00	17	10	25	False	Beverages
67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14.00	52	0	10	False	Beverages
70	Outback Lager	7	1	24 - 355 ml bottles	15.00	15	10	30	False	Beverages
75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7.75	125	0	25	False	Beverages
76	Lakalikakori	23	1	500 ml	18.00	57	0	20	False	Beverages
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	False	Condiments
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	False	Condiments
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	False	Condiments
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	False	Condiments
15	Genen Shouyu	6	2	24 - 250 ml bottles	15.50	39	0	5	False	Condiments
44	Gula Malacca	20	2	20 - 2 kg bags	19.45	27	0	15	False	Condiments
61	Siroop d'érable	29	2	24 - 500 ml bottles	28.50	113	0	25	False	Condiments
63	Vegie-spread	7	2	15 - 625 g jars	43.90	24	0	5	False	Condiments
65	Louisiana Fiery Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05	76	0	0	False	Condiments
66	Louisiana Hot Spiced Okra	2	2	24 - 8 oz jars	17.00	4	100	20	False	Condiments
77	Original Frankfurter grüne Soße	12	2	12 boxes	13.00	32	0	15	False	Condiments
16	Pavlova	7	3	32 - 500 g boxes	17.45	29	0	10	False	Confections
19	Teatime Chocolate Biscuits	8	3	10 boxes x 12 pieces	9.20	25	0	5	False	Confections
20	Sir Rodney's Marmalade	8	3	30 gift boxes	81.00	40	0	0	False	Confections
21	Sir Rodney's Scones	8	3	24 pkgs x 4 pieces	10.00	3	40	5	False	Confections
25	NutNuCo Nougat-Creme	11	3	20 - 450 g glasses	14.00	76	0	30	False	Confections
36	Crunchy Granola	11	3	100 - 750 g boxes	21.95	112	0	0	False	Confections

3. Next, drag Ship Country to a space on the canvas in the top right. Because you selected a geographic field, a map was created automatically. Now drag Line Total to the Values field; the circles on the map for each country are now relative in size to the Line Total for orders shipped to that country.

STEP 2: Interact with your report visuals to analyze further



Step 3: Creating Relationships with tables.

