```
ubuntu@ubuntu-OptiPlex-3090:~$ sudo service mongodb start
[sudo] password for ubuntu:
ubuntu@ubuntu-OptiPlex-3090:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("25635e6a-4336-48e3-be23-a8bb665e00c8") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-09-08T15:44:54.646+0530 I STORAGE  [initandlisten]
2022-09-08T15:44:54.646+0530 I STORAGE  [initandlisten] ** WARNING: Using the XFS
filesystem is strongly recommended with the WiredTiger storage engine
2022-09-08T15:44:54.646+0530 I STORAGE  [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten]
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not
enabled for the database.
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten] **          Read and write access to data
and configuration is unrestricted.
2022-09-08T15:44:55.315+0530 I CONTROL  [initandlisten]
> use samarth
switched to db samarth
> db.createCollection("emp_info")
{ "ok" : 1 }
> show collections
emp_info
> db.emp_info.insert({id:"E101",ename:"abc",age:25,dept:"tester",sal:25000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E102",ename:"pqr",age:50,dept:"R&D",sal:50000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E103",ename:"def",age:56,dept:"R&D",sal:75000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E104",ename:"xyz",age:28,dept:"dev",sal:50000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.insert({id:"E105",ename:"mno",age:30,dept:"tester",sal:55000})
WriteResult({ "nInserted" : 1 })
> db.emp_info.find()
{ "_id" : ObjectId("6319c913eb1ee6a2f9b3d141"), "id" : "E101", "ename" : "abc", "age" : 25, "dept" :
"tester", "sal" : 25000 }
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "id" : "E102", "ename" : "pqr", "age" : 50, "dept" :
"R&D", "sal" : 50000 }
{ "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"), "id" : "E103", "ename" : "def", "age" : 56, "dept" :
"R&D", "sal" : 75000 }
{ "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"), "id" : "E104", "ename" : "xyz", "age" : 28, "dept" :
"dev", "sal" : 50000 }
{ "_id" : ObjectId("6319c993eb1ee6a2f9b3d145"), "id" : "E105", "ename" : "mno", "age" : 30, "dept" :
"tester", "sal" : 55000 }
> db.emp_info.find().pretty()
{
        "_id" : ObjectId("6319c913eb1ee6a2f9b3d141"),
        "id" : "E101",
        "ename" : "abc",
        "age" : 25,
        "dept" : "tester",
        "sal" : 25000
}
{
        "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"),
        "id" : "E102",
        "ename" : "pqr",
        "age" : 50,
        "dept" : "R&D",
        "sal" : 50000
}
{
        "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"),
        "id" : "E103",
        "ename" : "def",
        "age" : 56,
        "dept" : "R&D",
        "sal" : 75000
}
{
        "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"),
        "id" : "E104",
        "ename" : "xyz",
        "age" : 28,
        "dept" : "dev",
        "sal" : 50000
}
{
        "_id" : ObjectId("6319c993eb1ee6a2f9b3d145"),
        "id" : "E105",
        "ename" : "mno",
        "age" : 30,
        "dept" : "tester",
        "sal" : 55000
}
> db.emp_info.find({sal:{$gte:50000}})
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "id" : "E102", "ename" : "pqr", "age" : 50, "dept" :
"R&D", "sal" : 50000 }
{ "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"), "id" : "E103", "ename" : "def", "age" : 56, "dept" :
"R&D", "sal" : 75000 }
{ "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"), "id" : "E104", "ename" : "xyz", "age" : 28, "dept" :
"dev", "sal" : 50000 }
{ "_id" : ObjectId("6319c993eb1ee6a2f9b3d145"), "id" : "E105", "ename" : "mno", "age" : 30, "dept" :
"tester", "sal" : 55000 }
> db.emp_info.find({age:50},{dept:"R&D"})
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "dept" : "R&D" }
```

```
> db.emp_info.findOne({age:50},{dept:"R&D"})
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "dept" : "R&D" }
> db.emp_info.remove({dept:"tester"})
WriteResult({ "nRemoved" : 2 })
> db.emp_info.update({dept="dev"},{$set:{dept:"R&D"}},{multi:true})
2022-09-08T16:29:10.180+0530 E QUERY    [thread1] SyntaxError: missing : after property id
@(shell):1:24
> db.emp_info.update({dept:"dev"},{$set:{dept:"R&D"}},{multi:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.emp_info.update({dept:"tester"},{dept:"R&D"})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.emp_info.find({dept:"tester"},{dept:"R&D"})
> db.emp_info.find({dept:"tester","R&D"})
2022-09-08T16:31:55.732+0530 E QUERY    [thread1] SyntaxError: missing : after property id
@(shell):1:37
> db.emp_info.find({$or[{dept:"tester"},{dept:"R&D"}]})
2022-09-08T16:33:14.580+0530 E QUERY    [thread1] SyntaxError: missing : after property id
@(shell):1:21
> db.emp_info.find()
{ "_id" : ObjectId("6319c935eb1ee6a2f9b3d142"), "id" : "E102", "ename" : "pqr", "age" : 50, "dept" :
"R&D", "sal" : 50000 }
{ "_id" : ObjectId("6319c952eb1ee6a2f9b3d143"), "id" : "E103", "ename" : "def", "age" : 56, "dept" :
"R&D", "sal" : 75000 }
{ "_id" : ObjectId("6319c976eb1ee6a2f9b3d144"), "id" : "E104", "ename" : "xyz", "age" : 28, "dept" :
"R&D", "sal" : 50000 }
```

Q. Develop a MapReduce program to calculate the frequency of a given word in a given file.

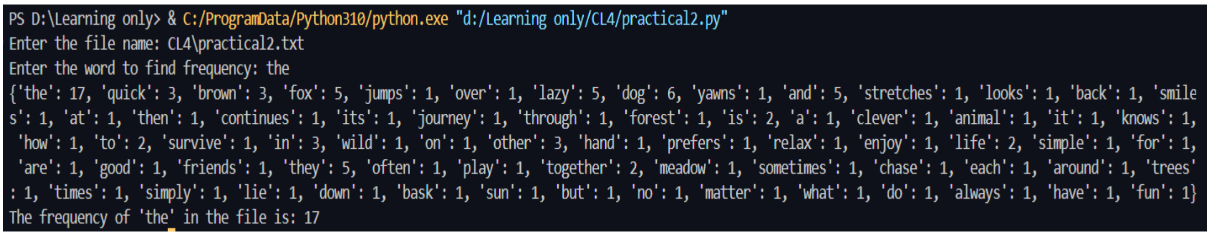
Wordcount.py

```
import re
from multiprocessing import Pool

WORD_RE = re.compile(r"[\w']+")

def read_file(filename):
    with open(filename, 'r') as file:
        return file.readlines()

def mapper(line):
    word_count = {}
    for word in WORD_RE.findall(line):
        word_count[word.lower()] = word_count.get(word.lower(), 0) + 1
    return word_count

def reducer(mapped_counts):
    reduced_counts = {}
    for word_count in mapped_counts:
        for word, count in word_count.items():
            reduced_counts[word] = reduced_counts.get(word, 0) + count
    print(reduced_counts)
    return reduced_counts

def main(filename, target_word):
    lines = read_file(filename)
    with Pool() as pool:
        mapped_counts = pool.map(mapper, lines)
    reduced_counts = reducer(mapped_counts)

    # Get the frequency of the target word
    target_frequency = reduced_counts.get(target_word.lower(), 0)
    print(f"The frequency of '{target_word}' in the file is: {target_frequency}")


if __name__ == "__main__":
    filename = input("Enter the file name: ")
    target_word = input("Enter the word to find frequency: ")
    main(filename, target_word)
```

samplefile.txt

The quick brown fox jumps over the lazy dog. The lazy dog yawns and stretches. The fox looks back and smiles at the dog. Then, the fox continues its journey through the forest. The quick brown fox is a clever animal. It knows how to survive in the wild. The lazy dog, on the other hand, prefers to relax and enjoy life. Life is simple for the lazy dog. The quick brown fox and the lazy dog are good friends. They often play together in the meadow. Sometimes, they chase each other around the trees. Other times, they simply lie down and bask in the sun. But no matter what they do, they always have fun together.

Output:



```
PS D:\Learning only> & C:/ProgramData/Python310/python.exe "d:/Learning only/CL4/practical2.py"
Enter the file name: CL4\practical2.txt
Enter the word to find frequency: the
{'the': 17, 'quick': 3, 'brown': 3, 'fox': 5, 'jumps': 1, 'over': 1, 'lazy': 5, 'dog': 6, 'yawns': 1, 'and': 5, 'stretches': 1, 'looks': 1, 'back': 1, 'smile
s': 1, 'at': 1, 'then': 1, 'continues': 1, 'its': 1, 'journey': 1, 'through': 1, 'forest': 1, 'is': 2, 'a': 1, 'clever': 1, 'animal': 1, 'it': 1, 'knows': 1,
 'how': 1, 'to': 2, 'survive': 1, 'in': 3, 'wild': 1, 'on': 1, 'other': 3, 'hand': 1, 'prefers': 1, 'relax': 1, 'enjoy': 1, 'life': 2, 'simple': 1, 'for': 1,
 'are': 1, 'good': 1, 'friends': 1, 'they': 5, 'often': 1, 'play': 1, 'together': 2, 'meadow': 1, 'sometimes': 1, 'chase': 1, 'each': 1, 'around': 1, 'trees'
: 1, 'times': 1, 'simply': 1, 'lie': 1, 'down': 1, 'bask': 1, 'sun': 1, 'but': 1, 'no': 1, 'matter': 1, 'what': 1, 'do': 1, 'always': 1, 'have': 1, 'fun': 1}
The frequency of 'the' in the file is: 17
```

```
import multiprocessing

def matrix_multiply_mapper(row, col):
    result = 0
    for i in range(len(row)):
        result += row[i] * col[i]
    return result

def matrix_multiply_worker(args):
    row_index, row, columns = args
    return [(row_index, col_index, matrix_multiply_mapper(row, col))
            for col_index, col in enumerate(columns)]

def matrix_multiply_reduce(results):
    final_result = {}
    for row_index, col_index, value in results:
        if row_index not in final_result:
            final_result[row_index] = {}
        final_result[row_index][col_index] = value

    return final_result

def map_reduce_matrix_multiply(matrix1, matrix2):
    num_workers = multiprocessing.cpu_count()
    pool = multiprocessing.Pool(processes=num_workers)

    args = [(i, matrix1[i], matrix2) for i in range(len(matrix1))]
    intermediate_results = pool.map(matrix_multiply_worker, args)

    pool.close()
    pool.join()

    final_result = matrix_multiply_reduce(
        [item for sublist in intermediate_results for item in sublist])

    return final_result

if __name__ == "__main__":
    matrix1 = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ]

    matrix2 = [
        [9, 8, 7],
        [6, 5, 4],
        [3, 2, 1]
    ]

    result = map_reduce_matrix_multiply(matrix1, matrix2)
    for row_index, row in result.items():
        print(row)
```

```
{0: 46, 1: 28, 2: 10}
{0: 118, 1: 73, 2: 28}
{0: 190, 1: 118, 2: 46}
```

Q. Develop a MapReduce program to find the grades of students

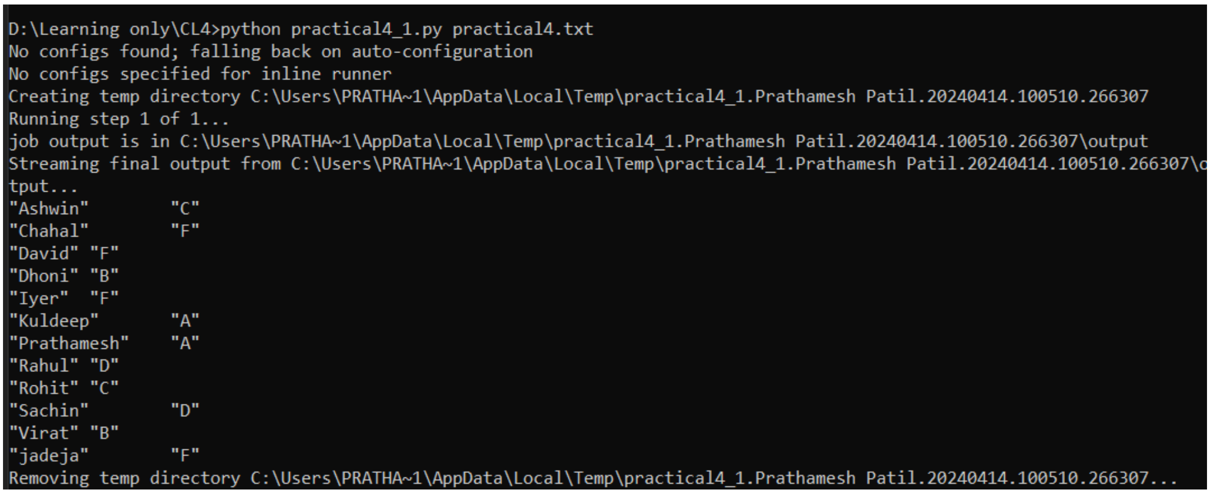
GradeCalculator.py

```
from mrjob.job import MRJob

class GradeCalculator(MRJob):
    def mapper(self, _, line):
        # Split the input line into name and score
        name, score = line.split(',')
        score = int(score)

        # Emit the name and score
        yield name, score

    def reducer(self, key, values):
        # Calculate the average score
        total_score = 0
        num_scores = 0
        for score in values:
            total_score += score
            num_scores += 1
        average_score = total_score / num_scores

        # Determine the grade based on the average score
        if average_score >= 90:
            grade = 'A'
        elif average_score >= 80:
            grade = 'B'
        elif average_score >= 70:
            grade = 'C'
        elif average_score >= 60:
            grade = 'D'
        else:
            grade = 'F'
        # Emit the name and grade
        yield key, grade


if __name__ == '__main__':
    GradeCalculator.run()
```

samplefile.txt

```
Prathamesh,95
Rohit,75
Virat,82
Sachin,68
Dhoni,88
Ashwin,73
Kuldeep,91
David,55
jadeja,50
Rahul,60
Iyer,45
Chahal,40
```

Output:

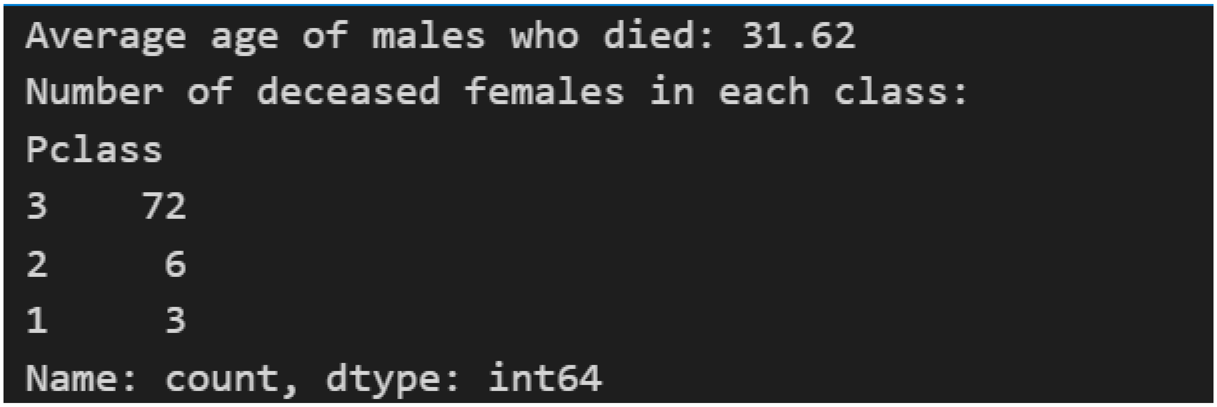


```
D:\Learning only\CL4>python practical4_1.py practical4.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory C:\Users\PRATHA~1\AppData\Local\Temp\practical4_1.Prathamesh Patil.20240414.100510.266307
Running step 1 of 1...
job output is in C:\Users\PRATHA~1\AppData\Local\Temp\practical4_1.Prathamesh Patil.20240414.100510.266307\output
Streaming final output from C:\Users\PRATHA~1\AppData\Local\Temp\practical4_1.Prathamesh Patil.20240414.100510.266307\o
tput...
"Ashwin"        "C"
"Chahal"        "F"
"David" "F"
"Dhoni" "B"
"Iyer"  "F"
"Kuldeep"       "A"
"Prathamesh"    "A"
"Rahul" "D"
"Rohit" "C"
"Sachin"        "D"
"Virat" "B"
"jadeja"        "F"
Removing temp directory C:\Users\PRATHA~1\AppData\Local\Temp\practical4_1.Prathamesh Patil.20240414.100510.266307...
```

Code:

```
import pandas as pd
def map_reduce_with_pandas(input_file):
 # Load the dataset
 df = pd.read_csv(input_file)

 # Map: Filter deceased males and transform data for average age
calculation
 deceased_males = df[(df['Survived'] == 0) & (df['Sex'] == 'male')]

 # Reduce: Calculate average age of deceased males
 average_age_deceased_males = deceased_males['Age'].mean()

 # Map: Filter deceased females and transform data for count by class
 deceased_females_by_class = df[(df['Survived'] == 0) & (df['Sex'] ==
'female')]
 # Reduce: Count deceased females by class
 count_deceased_females_by_class =
deceased_females_by_class['Pclass'].value_counts()
 return average_age_deceased_males, count_deceased_females_by_class

# Example usage
input_file = r'D:\BE SEM VIII\CL_IV_Code\titanic.csv' # Update this
to the path of your Titanic dataset CSV file
average_age, female_class_count = map_reduce_with_pandas(input_file)
print(f"Average age of males who died: {average_age:.2f}")
print("Number of deceased females in each class:")
print(female_class_count)
```

Output:



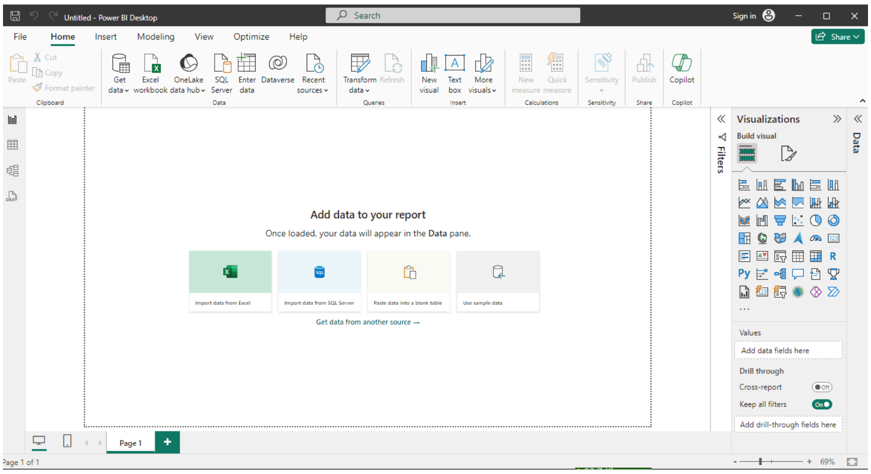
```
Average age of males who died: 31.62
Number of deceased females in each class:
Pclass
3    72
2     6
1     3
Name: count, dtype: int64
```

Assignment 6

Excel Data

Output:

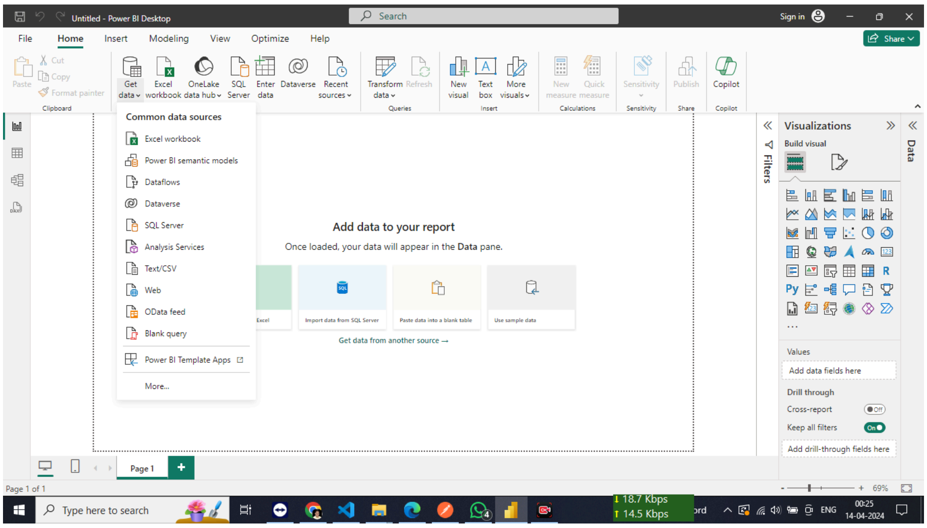
Step1:



Add data to your report

Once loaded, your data will appear in the **Data** pane.

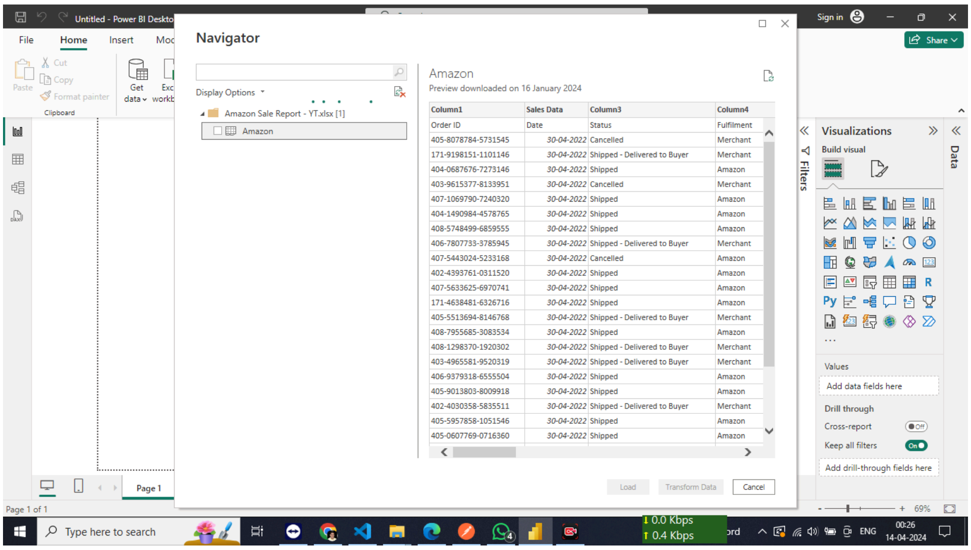
Step2:



Add data to your report

Once loaded, your data will appear in the **Data** pane.

Step3:



Navigator

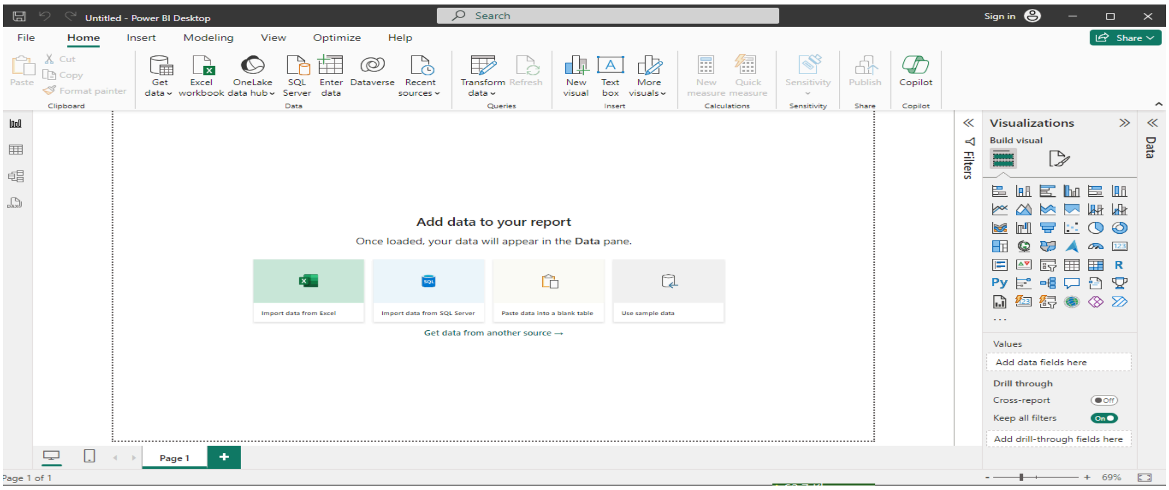
Amazon

Preview downloaded on 16 January 2024

Column1	Sales Data	Column3	Column4
Order ID	Date	Status	Fulfilment
405-8078784-5731545	30-04-2022	Cancelled	Merchant
171-9198151-1101146	30-04-2022	Shipped - Delivered to Buyer	Merchant
404-0687676-7273146	30-04-2022	Shipped	Amazon
403-9615377-8133951	30-04-2022	Cancelled	Merchant
407-1069790-7240320	30-04-2022	Shipped	Amazon
404-1490984-4578765	30-04-2022	Shipped	Amazon
408-5748499-6859555	30-04-2022	Shipped	Amazon
406-7807733-3785945	30-04-2022	Shipped - Delivered to Buyer	Merchant
407-5443024-5233168	30-04-2022	Cancelled	Amazon
402-4393761-0311520	30-04-2022	Shipped	Amazon
407-5633625-6970741	30-04-2022	Shipped	Amazon
171-4638481-6326716	30-04-2022	Shipped	Amazon
405-5513694-8146768	30-04-2022	Shipped - Delivered to Buyer	Merchant
408-7955685-3083534	30-04-2022	Shipped	Amazon
408-1298370-1920302	30-04-2022	Shipped - Delivered to Buyer	Merchant
403-4965581-9520319	30-04-2022	Shipped - Delivered to Buyer	Merchant
406-9379318-6555504	30-04-2022	Shipped	Amazon
405-9013803-8009918	30-04-2022	Shipped	Amazon
402-4030358-5835511	30-04-2022	Shipped - Delivered to Buyer	Merchant
405-5957858-1051546	30-04-2022	Shipped	Amazon
405-0607769-0716360	30-04-2022	Shipped	Amazon

Odata

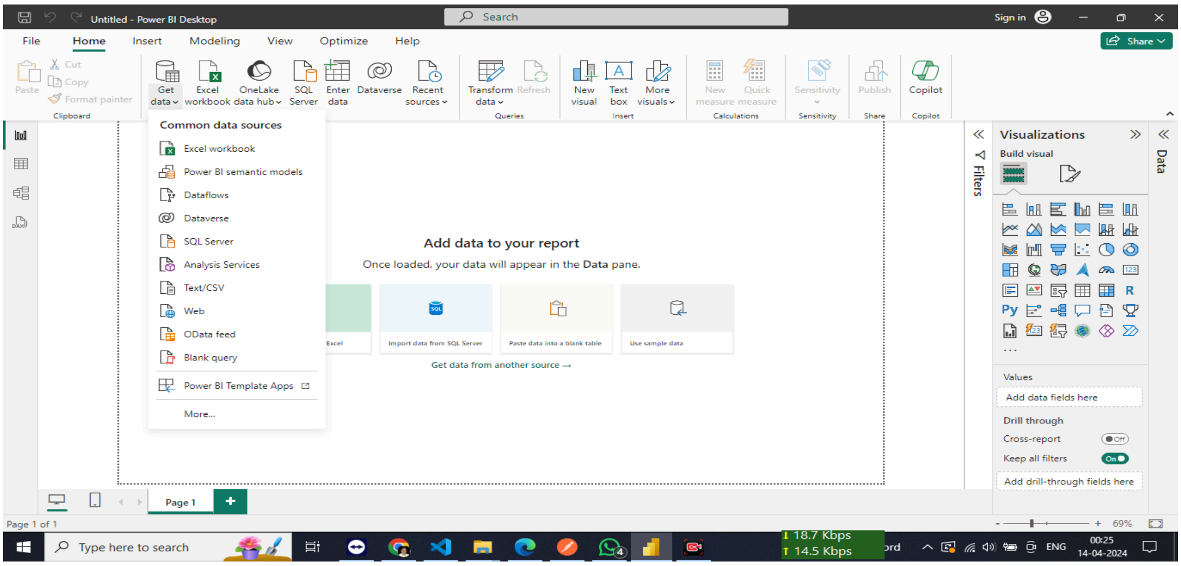
Step1:



Add data to your report

Once loaded, your data will appear in the **Data** pane.

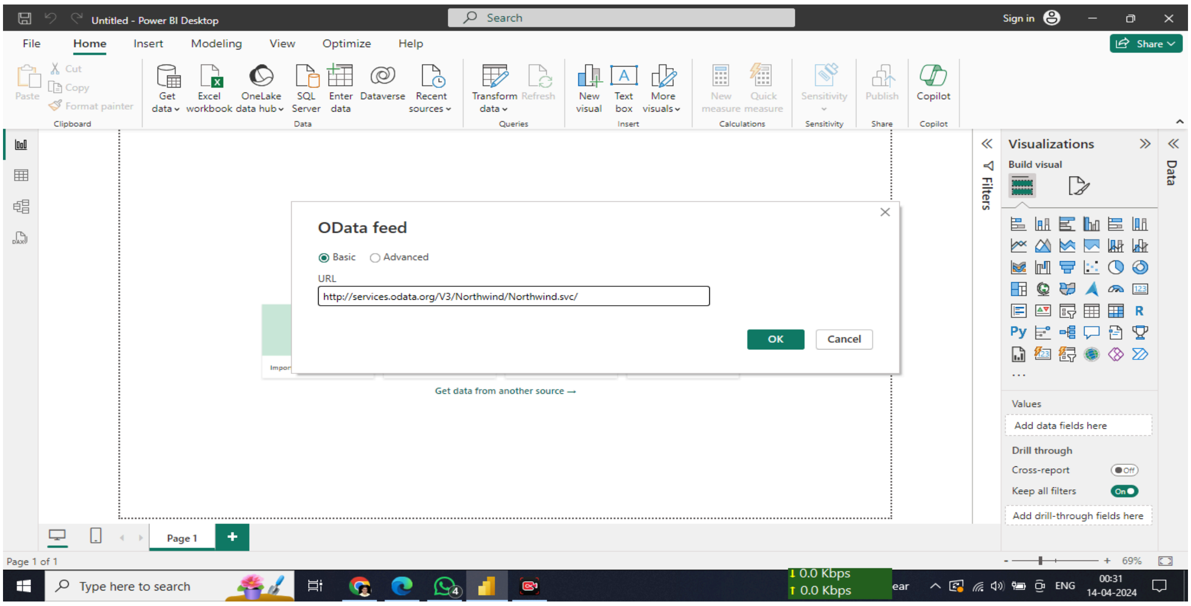
Step2:



Add data to your report

Once loaded, your data will appear in the **Data** pane.

Step3:



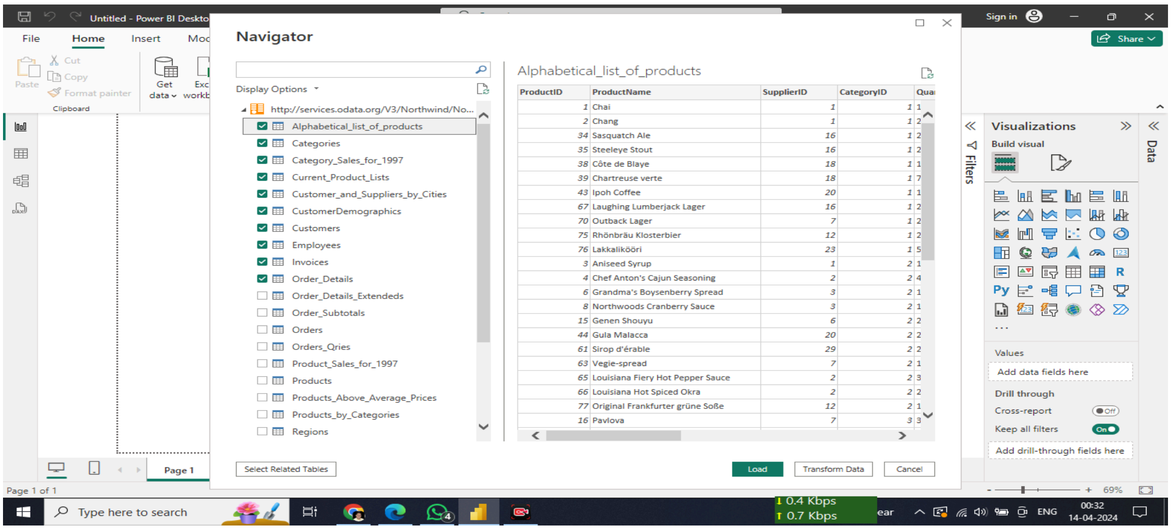
OData feed

● Basic ○ Advanced

URL

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

Step4:



Navigator

Display Options

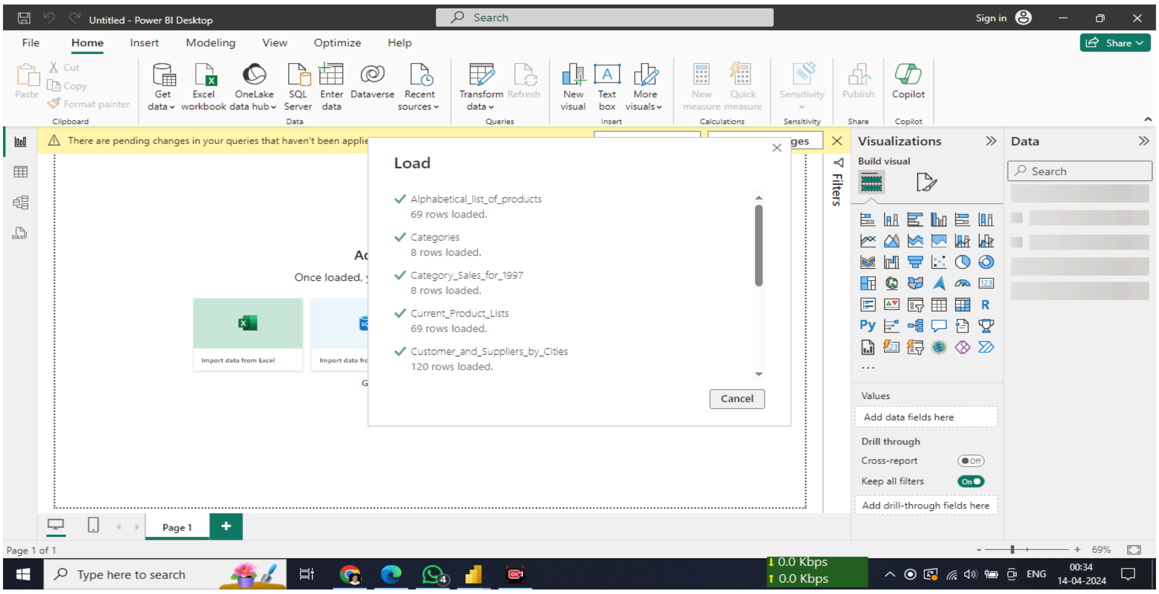
http://services.odata.org/V3/Northwind/No...

- ☑ 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_Extendeds
- □ Order_Subtotals
- □ Orders
- □ Orders_Qries
- □ Product_Sales_for_1997
- □ Products
- □ Products_Above_Average_Prices
- □ Products_by_Categories
- □ Regions

Alphabetical_list_of_products

ProductID	ProductName	SupplierID	CategoryID	Qua
1	Chai	1	1	1
2	Chang	1	1	2
34	Sasquatch Ale	16	1	2
35	Steeleye Stout	16	1	2
38	Côte de Blaye	18	1	1
39	Chartreuse verte	18	1	7
43	Ipoh Coffee	20	1	1
67	Laughing Lumberjack Lager	16	1	2
70	Outback Lager	7	1	2
75	Rhönbräu Klosterbier	12	1	2
76	Lakkalikööri	23	1	5
3	Aniseed Syrup	1	2	1
4	Chef Anton's Cajun Seasoning	2	2	4
6	Grandma's Boysenberry Spread	3	2	1
8	Northwoods Cranberry Sauce	3	2	1
15	Genen Shouyu	6	2	2
44	Gula Malacca	20	2	2
61	Sirop d'érable	29	2	2
63	Vegie-spread	7	2	1
65	Louisiana Fiery Hot Pepper Sauce	2	2	3
66	Louisiana Hot Spiced Okra	2	2	2
77	Original Frankfurter grüne Soße	12	2	1
16	Pavlova	7	3	3

Step5:



There are pending changes in your queries that haven't been applie

Load

✓ Alphabetical_list_of_products
69 rows loaded.

✓ Categories
8 rows loaded.

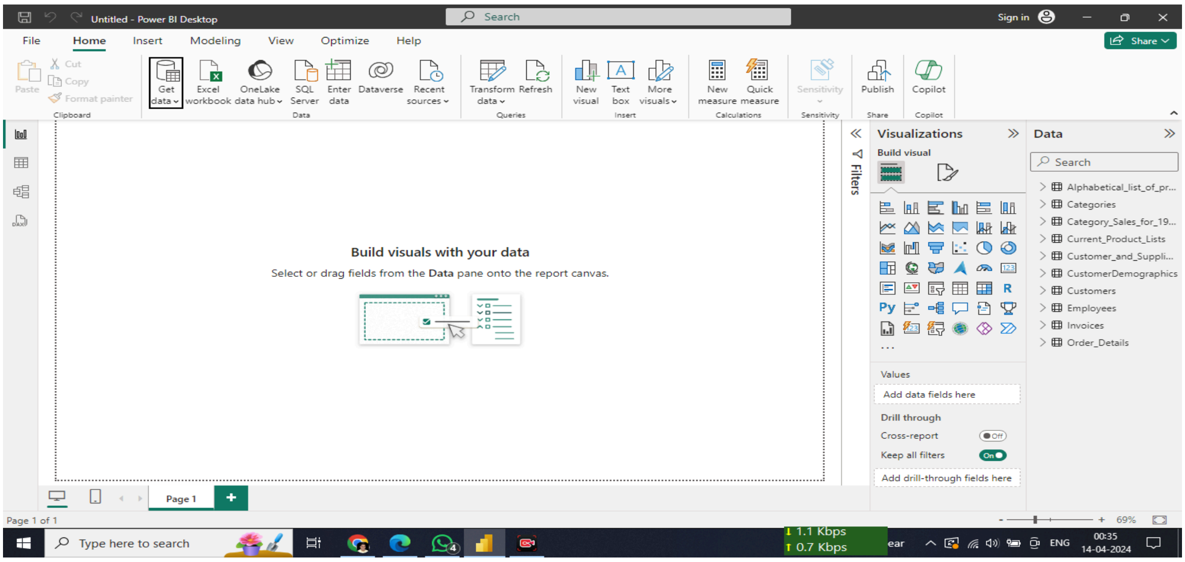
✓ Category_Sales_for_1997
8 rows loaded.

✓ Current_Product_Lists
69 rows loaded.

✓ Customer_and_Suppliers_by_Cities
120 rows loaded.

Cancel

Step6:



Build visuals with your data

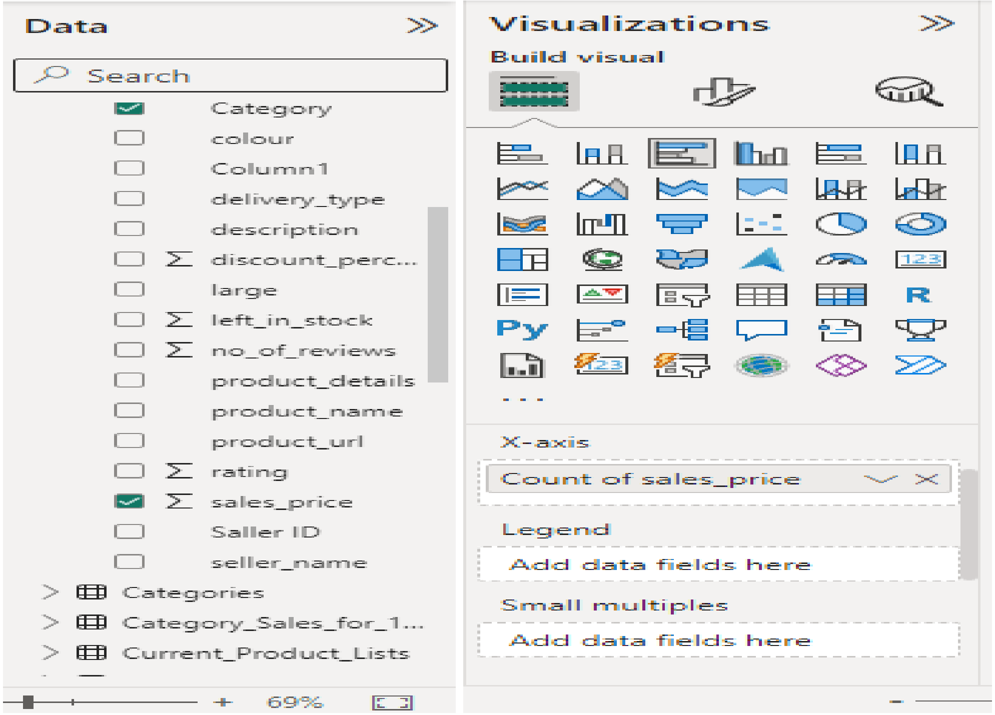
Select or drag fields from the **Data** pane onto the report canvas.

Data

- Alphabetical_list_of_pr...
- Categories
- Category_Sales_for_19...
- Current_Product_Lists
- Customer_and_Suppli...
- CustomerDemographics
- Customers
- Employees
- Invoices
- Order_Details

Assignment 7

Output:



Data

Search

- ☑ Category
- □ colour
- □ Column1
- □ delivery_type
- □ description
- □ Σ discount_perc...
- □ large
- □ Σ left_in_stock
- □ Σ no_of_reviews
- □ product_details
- □ product_name
- □ product_url
- □ Σ rating
- ☑ Σ sales_price
- □ Saller ID
- □ seller_name
- > Categories
- > Category_Sales_for_1...
- > Current_Product_Lists

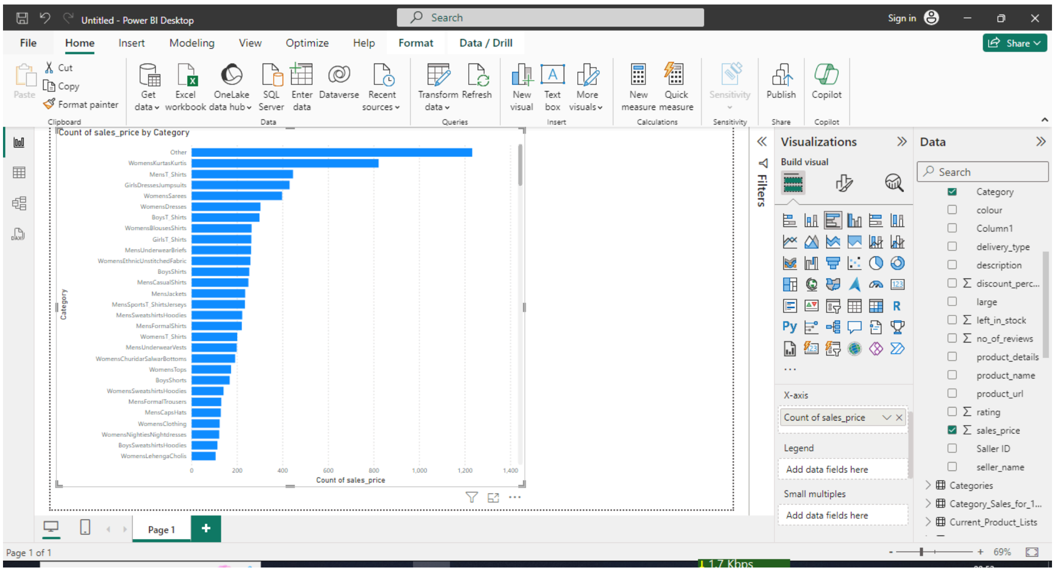
Visualizations

Build visual

X-axis: Count of sales_price

Legend: Add data fields here

Small multiples: Add data fields here



Untitled - Power BI Desktop

Visualizations

Build visual

X-axis

Count of sales_price

Legend

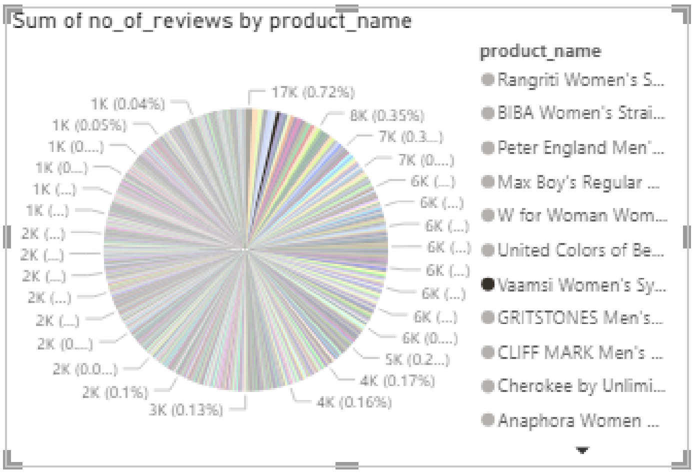
Add data fields here

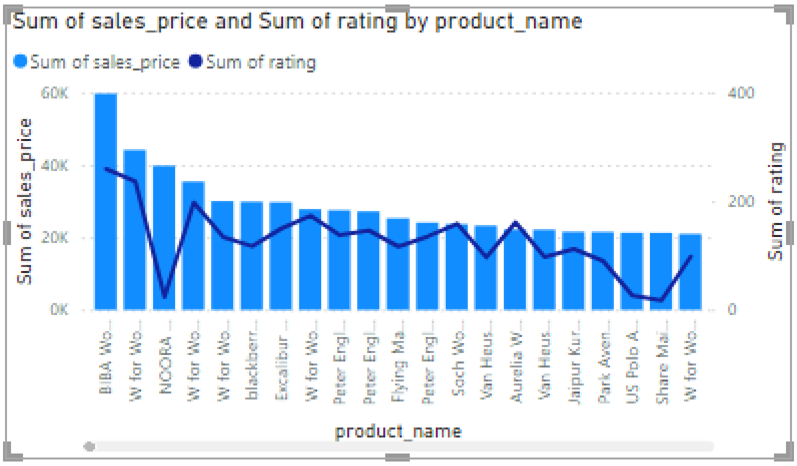
Small multiples

Add data fields here

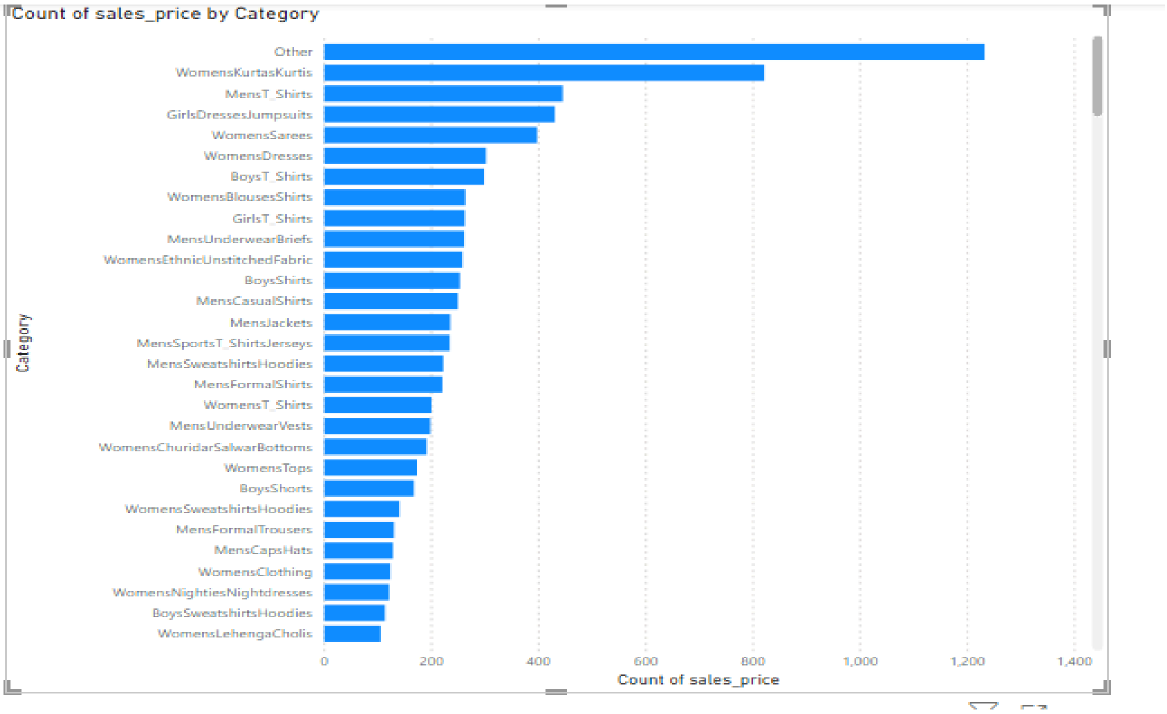
Data

- ☑ Category
- □ colour
- □ Column1
- □ delivery_type
- □ description
- □ Σ discount_perc...
- □ large
- □ Σ left_in_stock
- □ Σ no_of_reviews
- □ product_details
- □ product_name
- □ product_url
- □ Σ rating
- ☑ Σ sales_price
- □ Saller ID
- □ seller_name
- > Categories
- > Category_Sales_for_1...
- > Current_Product_Lists

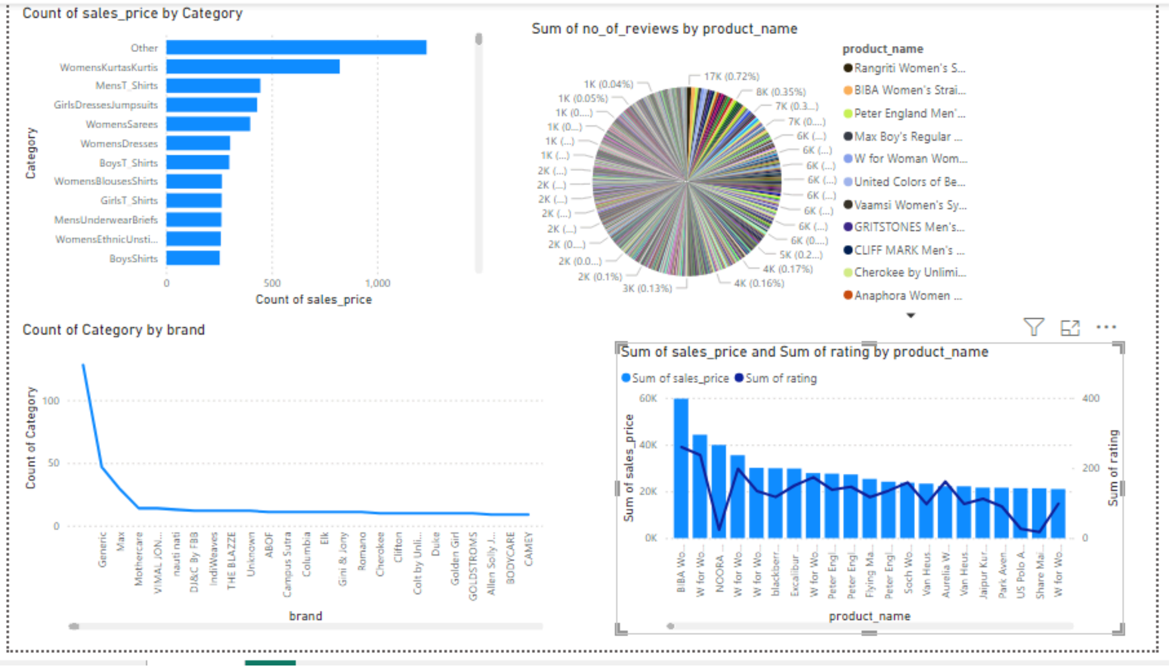




Sum of sales_price and Sum of rating by product_name
● Sum of sales_price ● Sum of rating
60K
40K
20K
0K
Sum of sales_price
400
200
0
Sum of rating
BIBA Wo...
W for Wo...
NOORA
W for Wo...
W for Wo...
blackberr...
Excalibur ...
W for Wo...
Peter Engl...
Peter Engl...
Flying Ma...
Peter Engl...
Soch Wo...
Van Heus...
Aurelia W...
Van Heus...
Jaipur Kur...
Park Aven...
US Polo A...
Share Mai...
W for Wo...
product_name



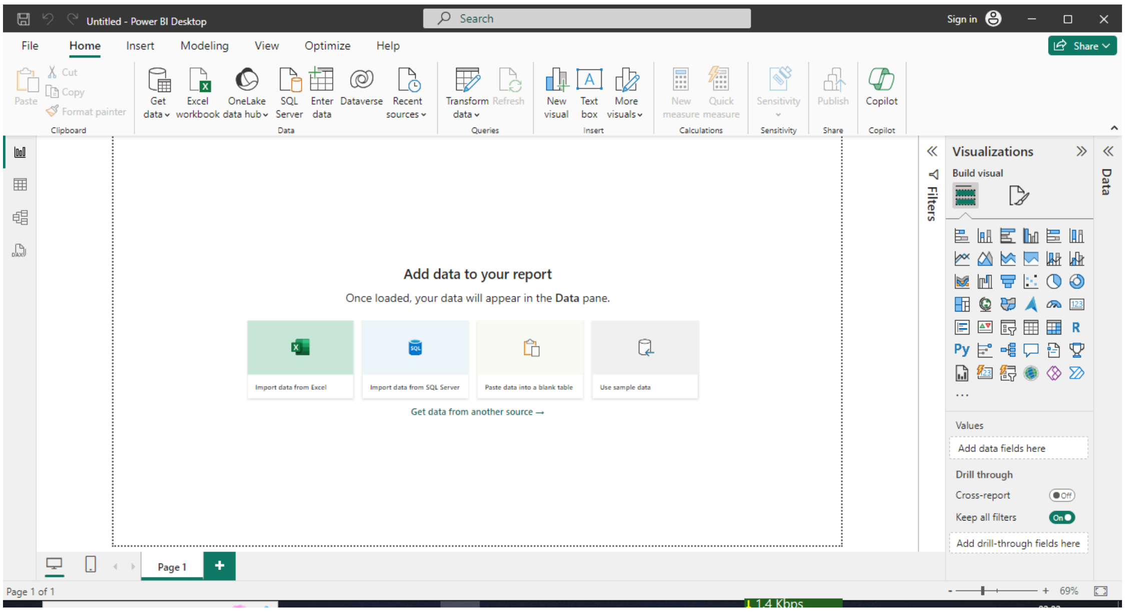
Count of sales_price by Category
Other
WomensKurtasKurtis
MensT_Shirts
GirlsDressesJumpsuits
WomensSarees
WomensDresses
BoysT_Shirts
WomensBlousesShirts
GirlsT_Shirts
MensUnderwearBriefs
WomensEthnicUnstitchedFabric
BoysShirts
MensCasualShirts
MensJackets
MensSportsT_ShirtsJerseys
MensSweatshirtsHoodies
MensFormalShirts
WomensT_Shirts
MensUnderwearVests
WomensChuridarSalwarBottoms
WomensTops
BoysShorts
WomensSweatshirtsHoodies
MensFormalTrousers
MensCapsHats
WomensClothing
WomensNightiesNightdresses
BoysSweatshirtsHoodies
WomensLehengaCholis
Category
0
200
400
600
800
1,000
1,200
1,400
Count of sales_price



Count of sales_price by Category
Other
WomensKurtasKurtis
MensT_Shirts
GirlsDressesJumpsuits
WomensSarees
WomensDresses
BoysT_Shirts
WomensBlousesShirts
GirlsT_Shirts
MensUnderwearBriefs
WomensEthnicUnsti...
BoysShirts
Category
0
500
1,000
Count of sales_price
Sum of no_of_reviews by product_name
17K (0.72%)
8K (0.35%)
7K (0.3...)
7K (0....)
6K (...)
6K (...)
6K (...)
6K (...)
6K (...)
6K (...)
6K (...)
6K (0....)
5K (0.2...)
4K (0.17%)
4K (0.16%)
3K (0.13%)
2K (0.1%)
2K (0.0...)
2K (0....)
2K (...)
2K (...)
2K (...)
2K (...)
2K (...)
1K (...)
1K (...)
1K (0...)
1K (0....)
1K (0.05%)
1K (0.04%)
product_name
● Rangriti Women's S...
● BIBA Women's Strai...
● Peter England Men'...
● Max Boy's Regular ...
● W for Woman Wom...
● United Colors of Be...
● Vaamsi Women's Sy...
● GRITSTONES Men's...
● CLIFF MARK Men's ...
● Cherokee by Unlimi...
● Anaphora Women ...
Count of Category by brand
100
50
0
Count of Category
Generic
Max
Mothercare
VIMAL JON...
nauti nati
DJ&C By FBB
IndiWeaves
THE BLAZZE
Unknown
ABOF
Campus Sutra
Columbia
Elk
Gini & Jony
Romano
Cherokee
Clifton
Colt by Unli...
Duke
Golden Girl
GOLDSTROMS
Allen Solly J...
BODYCARE
CAMEY
brand
Sum of sales_price and Sum of rating by product_name
● Sum of sales_price ● Sum of rating
60K
40K
20K
0K
Sum of sales_price
400
200
0
Sum of rating
BIBA Wo...
W for Wo...
NOORA
W for Wo...
W for Wo...
blackberr...
Excalibur ...
W for Wo...
Peter Engl...
Peter Engl...
Flying Ma...
Peter Engl...
Soch Wo...
Van Heus...
Aurelia W...
Van Heus...
Jaipur Kur...
Park Aven...
US Polo A...
Share Mai...
W for Wo...
product_name

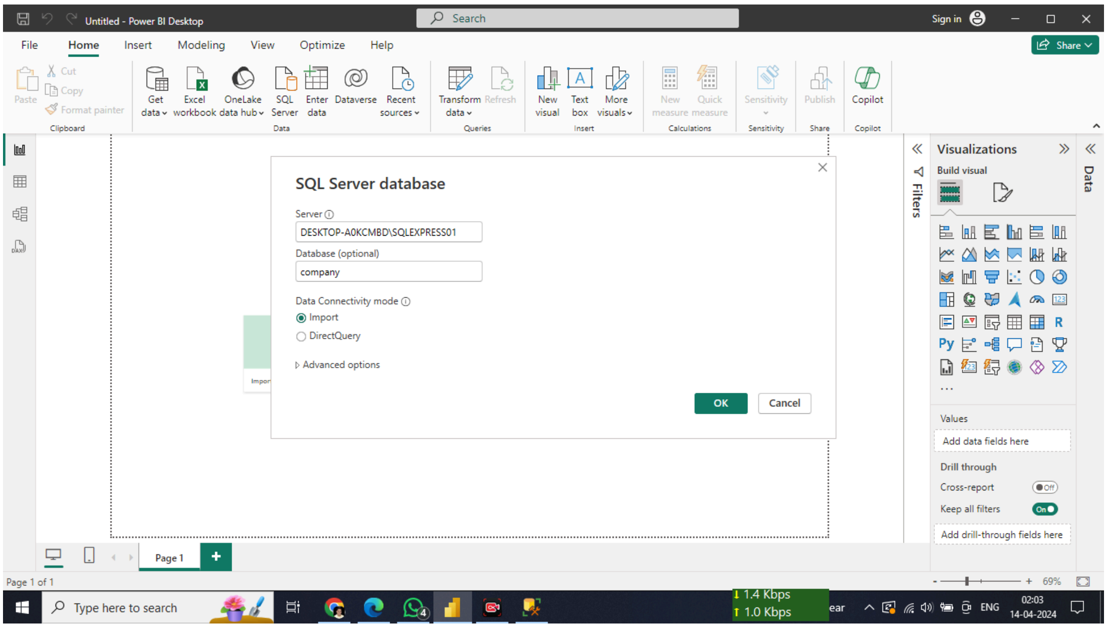
Assignment 8

Step1: Extraction



Add data to your report

Once loaded, your data will appear in the **Data** pane.



SQL Server database

Server: DESKTOP-A0KCMBD\SQLEXPRESS01

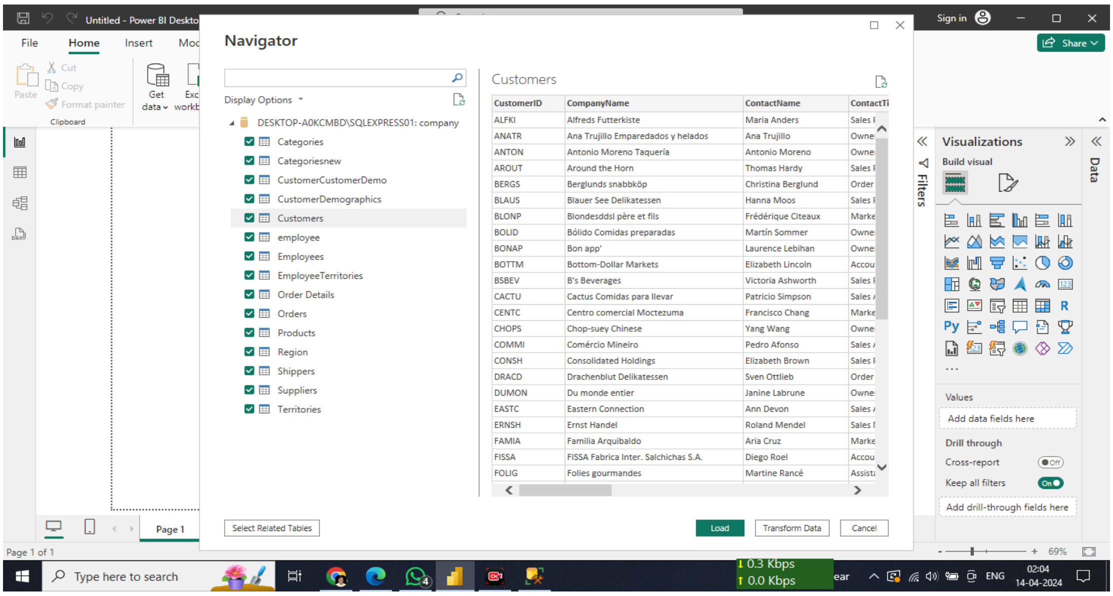
Database (optional): company

Data Connectivity mode

● Import

○ DirectQuery

▷ Advanced options



Navigator

Display Options

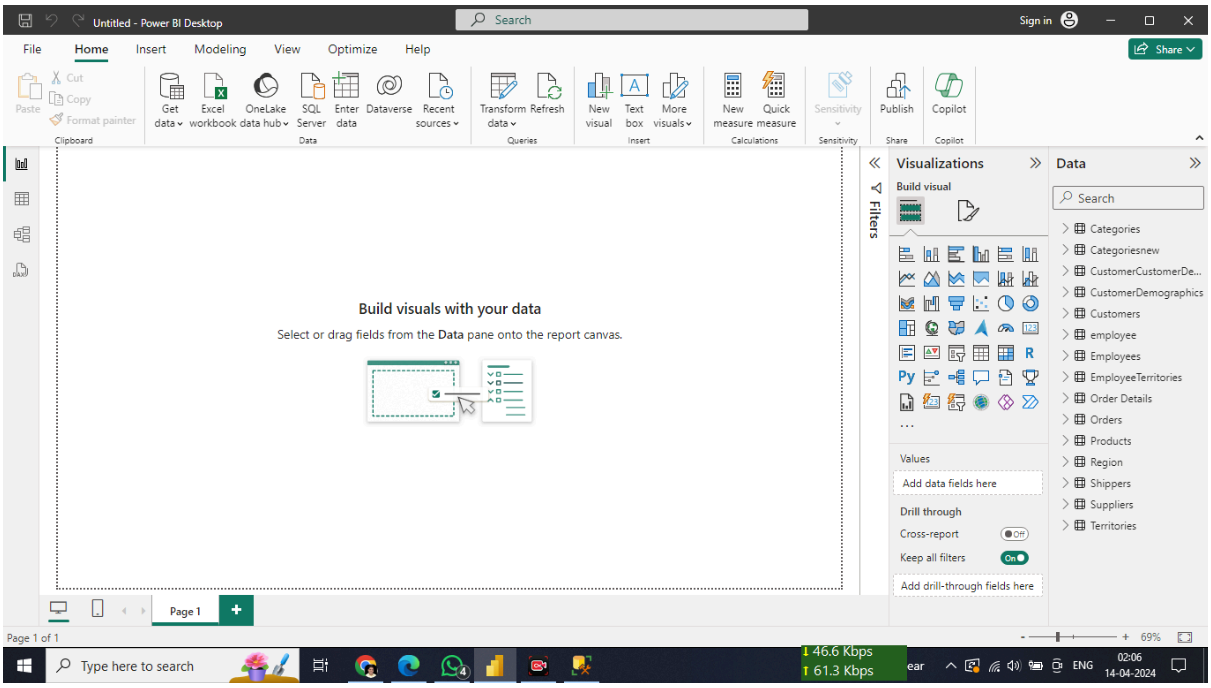
DESKTOP-A0KCMBD\SQLEXPRESS01: company

- ☑ Categories
- ☑ Categoriesnew
- ☑ CustomerCustomerDemo
- ☑ CustomerDemographics
- ☑ Customers
- ☑ employee
- ☑ Employees
- ☑ EmployeeTerritories
- ☑ Order Details
- ☑ Orders
- ☑ Products
- ☑ Region
- ☑ Shippers
- ☑ Suppliers
- ☑ Territories

Customers

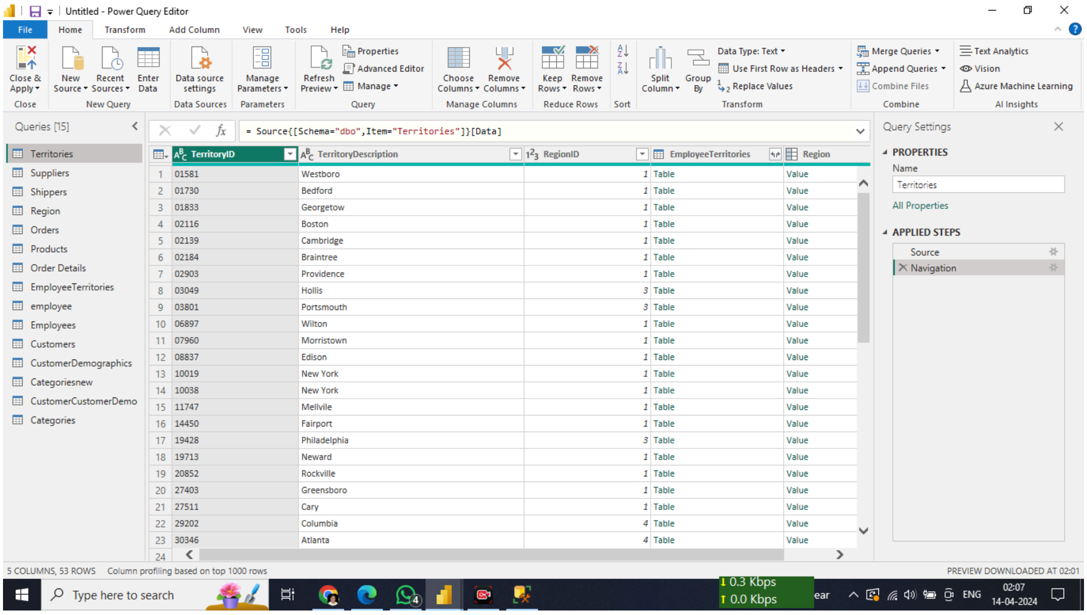
CustomerID	CompanyName	ContactName	ContactTi
ALFKI	Alfreds Futterkiste	Maria Anders	Sales
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owne
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owne
AROUT	Around the Horn	Thomas Hardy	Sales
BERGS	Berglunds snabbköp	Christina Berglund	Order
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales
BLONP	Blondesddsl père et fils	Frédérique Citeaux	Marke
BOLID	Bólido Comidas preparadas	Martín Sommer	Owne
BONAP	Bon app'	Laurence Lebihan	Owne
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accou
BSBEV	B's Beverages	Victoria Ashworth	Sales
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales
CENTC	Centro comercial Moctezuma	Francisco Chang	Marke
CHOPS	Chop-suey Chinese	Yang Wang	Owne
COMMI	Comércio Mineiro	Pedro Afonso	Sales
CONSH	Consolidated Holdings	Elizabeth Brown	Sales
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Order
DUMON	Du monde entier	Janine Labrune	Owne
EASTC	Eastern Connection	Ann Devon	Sales
ERNSH	Ernst Handel	Roland Mendel	Sales
FAMIA	Familia Arquibaldo	Aria Cruz	Marke
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Accou
FOLIG	Folies gourmandes	Martine Rancé	Assist

Step 2: Transform Data



Build visuals with your data

Select or drag fields from the **Data** pane onto the report canvas.



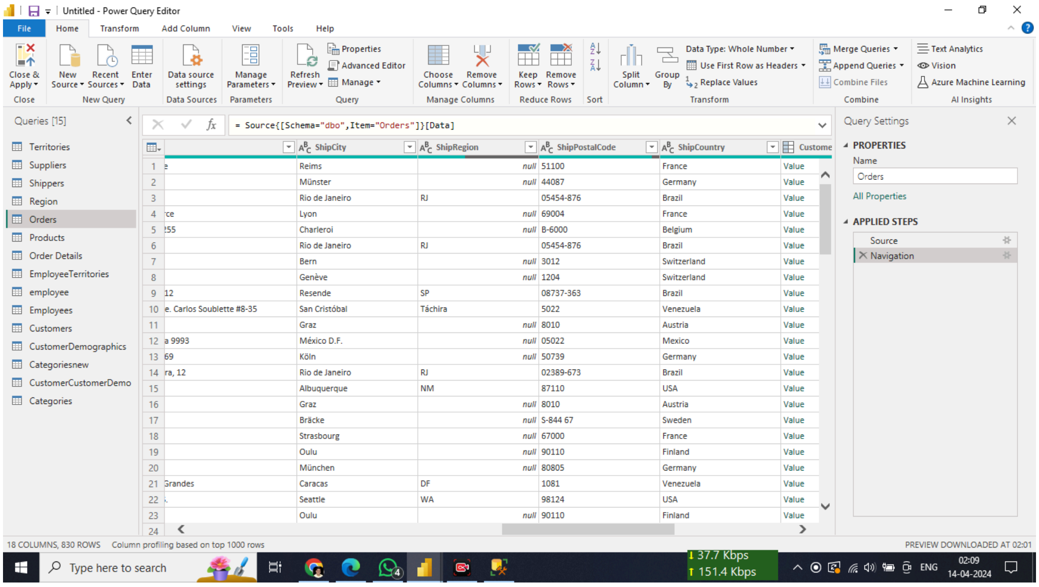
```
= Source{[Schema="dbo",Item="Territories"]}[Data]
```

	TerritoryID	TerritoryDescription	RegionID	EmployeeTerritories	Region
1	01581	Westboro	1	Table	Value
2	01730	Bedford	1	Table	Value
3	01833	Georgetow	1	Table	Value
4	02116	Boston	1	Table	Value
5	02139	Cambridge	1	Table	Value
6	02184	Braintree	1	Table	Value
7	02903	Providence	1	Table	Value
8	03049	Hollis	3	Table	Value
9	03801	Portsmouth	3	Table	Value
10	06897	Wilton	1	Table	Value
11	07960	Morristown	1	Table	Value
12	08837	Edison	1	Table	Value
13	10019	New York	1	Table	Value
14	10038	New York	1	Table	Value
15	11747	Mellvile	1	Table	Value
16	14450	Fairport	1	Table	Value
17	19428	Philadelphia	3	Table	Value
18	19713	Neward	1	Table	Value
19	20852	Rockville	1	Table	Value
20	27403	Greensboro	1	Table	Value
21	27511	Cary	1	Table	Value
22	29202	Columbia	4	Table	Value
23	30346	Atlanta	4	Table	Value
24					

5 COLUMNS, 53 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 02:01

Removing Null values:

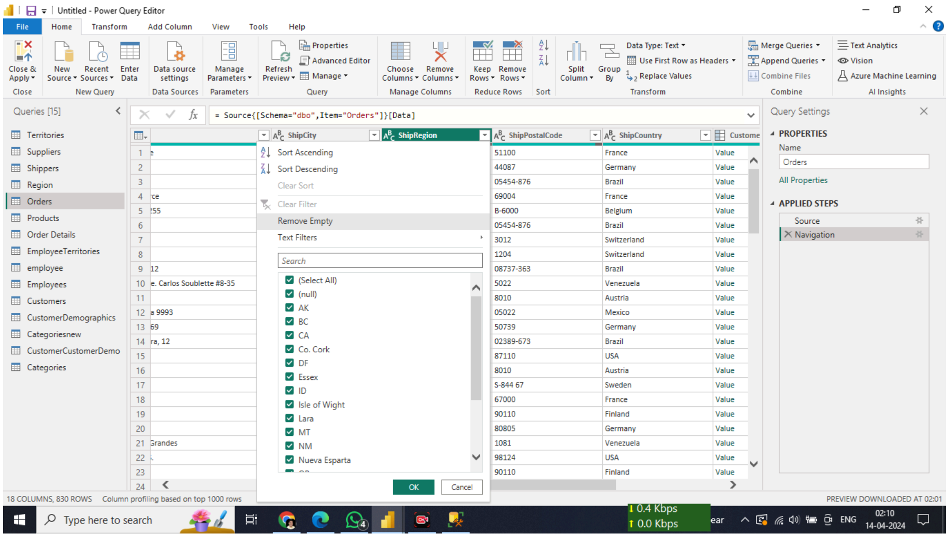


```
= Source{[Schema="dbo",Item="Orders"]}[Data]
```

		ShipCity	ShipRegion	ShipPostalCode	ShipCountry	Custome
1	e	Reims	null	51100	France	Value
2		Münster	null	44087	Germany	Value
3		Rio de Janeiro	RJ	05454-876	Brazil	Value
4	ce	Lyon	null	69004	France	Value
5	255	Charleroi	null	B-6000	Belgium	Value
6		Rio de Janeiro	RJ	05454-876	Brazil	Value
7		Bern	null	3012	Switzerland	Value
8		Genève	null	1204	Switzerland	Value
9	12	Resende	SP	08737-363	Brazil	Value
10	e. Carlos Soublette #8-35	San Cristóbal	Táchira	5022	Venezuela	Value
11		Graz	null	8010	Austria	Value
12	a 9993	México D.F.	null	05022	Mexico	Value
13	69	Köln	null	50739	Germany	Value
14	ra, 12	Rio de Janeiro	RJ	02389-673	Brazil	Value
15		Albuquerque	NM	87110	USA	Value
16		Graz	null	8010	Austria	Value
17		Bräcke	null	S-844 67	Sweden	Value
18		Strasbourg	null	67000	France	Value
19		Oulu	null	90110	Finland	Value
20		München	null	80805	Germany	Value
21	Grandes	Caracas	DF	1081	Venezuela	Value
22	.	Seattle	WA	98124	USA	Value
23		Oulu	null	90110	Finland	Value
24						

18 COLUMNS, 830 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 02:01



```
= Source{[Schema="dbo",Item="Orders"]}[Data]
```

ShipRegion filter list:

- ☑ (Select All)
- ☑ (null)
- ☑ AK
- ☑ BC
- ☑ CA
- ☑ Co. Cork
- ☑ DF
- ☑ Essex
- ☑ ID
- ☑ Isle of Wight
- ☑ Lara
- ☑ MT
- ☑ NM
- ☑ Nueva Esparta

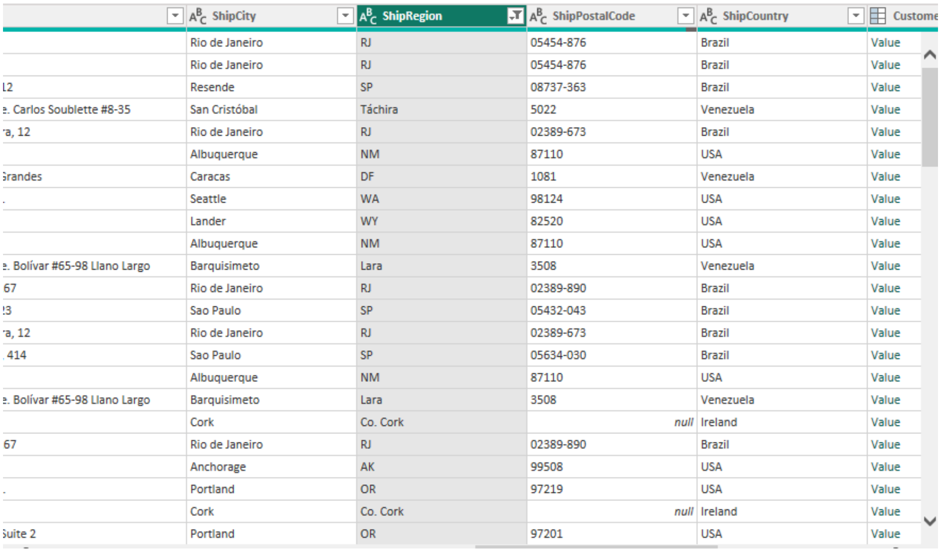
		ShipPostalCode	ShipCountry	Custome
1	e	51100	France	Value
2		44087	Germany	Value
3		05454-876	Brazil	Value
4	ce	69004	France	Value
5	255	B-6000	Belgium	Value
6		05454-876	Brazil	Value
7		3012	Switzerland	Value
8		1204	Switzerland	Value
9	12	08737-363	Brazil	Value
10	e. Carlos Soublette #8-35	5022	Venezuela	Value
11		8010	Austria	Value
12	a 9993	05022	Mexico	Value
13	69	50739	Germany	Value
14	ra, 12	02389-673	Brazil	Value
15		87110	USA	Value
16		8010	Austria	Value
17		S-844 67	Sweden	Value
18		67000	France	Value
19		90110	Finland	Value
20		80805	Germany	Value
21	Grandes	1081	Venezuela	Value
22	.	98124	USA	Value
23		90110	Finland	Value
24				

Query Settings — Name: Orders

APPLIED STEPS: Source, Navigation

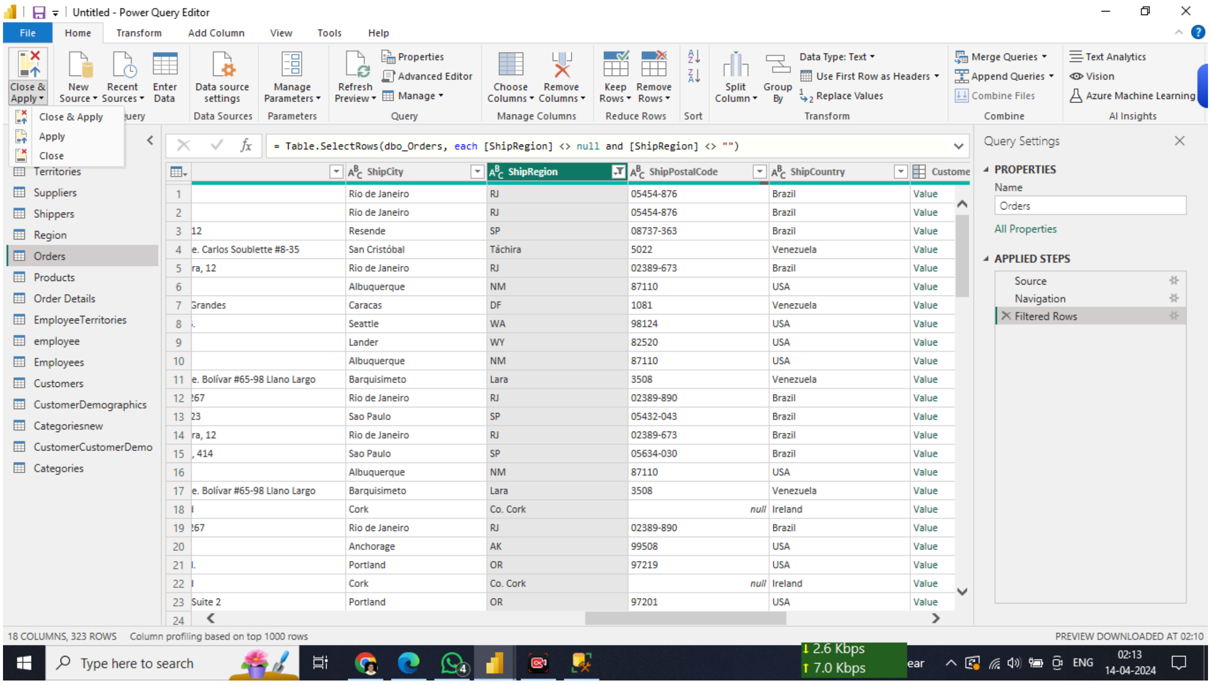
18 COLUMNS, 830 ROWS   Column profiling based on top 1000 rows   PREVIEW DOWNLOADED AT 02:01

Null values removed



	ShipCity	ShipRegion	ShipPostalCode	ShipCountry	Custome
	Rio de Janeiro	RJ	05454-876	Brazil	Value
	Rio de Janeiro	RJ	05454-876	Brazil	Value
l2	Resende	SP	08737-363	Brazil	Value
e. Carlos Soublette #8-35	San Cristóbal	Táchira	5022	Venezuela	Value
ra, 12	Rio de Janeiro	RJ	02389-673	Brazil	Value
	Albuquerque	NM	87110	USA	Value
Grandes	Caracas	DF	1081	Venezuela	Value
.	Seattle	WA	98124	USA	Value
	Lander	WY	82520	USA	Value
	Albuquerque	NM	87110	USA	Value
e. Bolívar #65-98 Llano Largo	Barquisimeto	Lara	3508	Venezuela	Value
67	Rio de Janeiro	RJ	02389-890	Brazil	Value
3	Sao Paulo	SP	05432-043	Brazil	Value
ra, 12	Rio de Janeiro	RJ	02389-673	Brazil	Value
414	Sao Paulo	SP	05634-030	Brazil	Value
	Albuquerque	NM	87110	USA	Value
e. Bolívar #65-98 Llano Largo	Barquisimeto	Lara	3508	Venezuela	Value
	Cork	Co. Cork	null	Ireland	Value
67	Rio de Janeiro	RJ	02389-890	Brazil	Value
	Anchorage	AK	99508	USA	Value
.	Portland	OR	97219	USA	Value
	Cork	Co. Cork	null	Ireland	Value
Suite 2	Portland	OR	97201	USA	Value

Close and apply



```
= Table.SelectRows(dbo_Orders, each [ShipRegion] <> null and [ShipRegion] <> "")
```

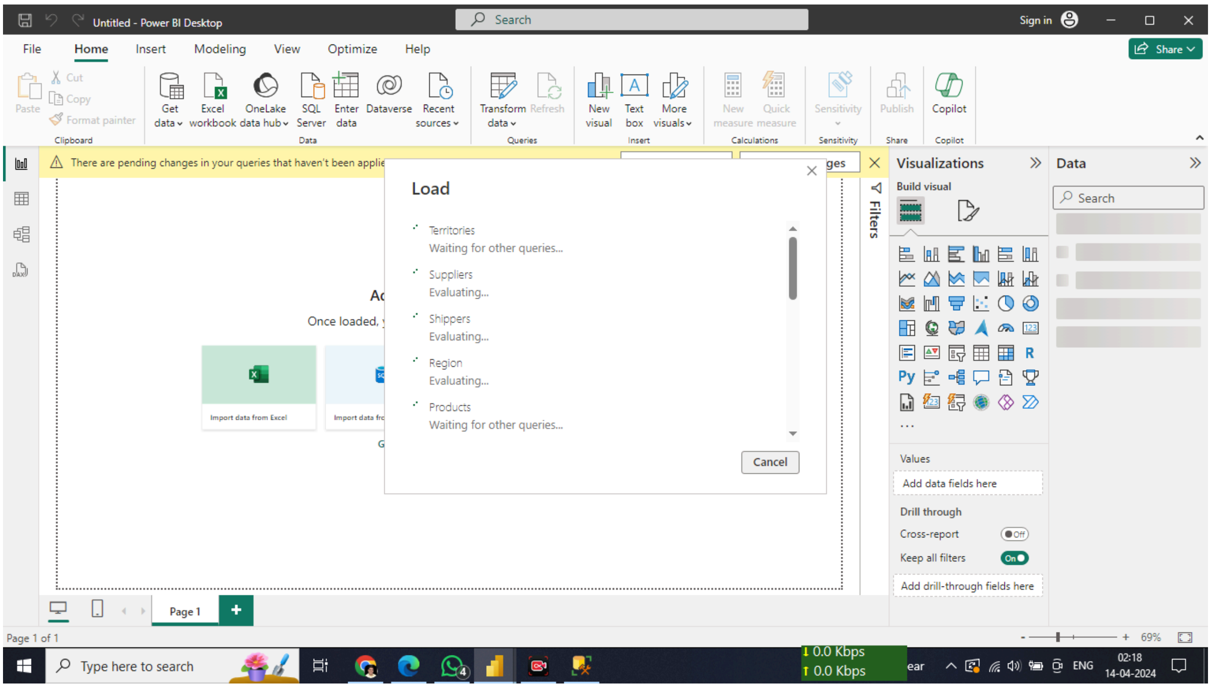
		ShipCity	ShipRegion	ShipPostalCode	ShipCountry	Custome
1		Rio de Janeiro	RJ	05454-876	Brazil	Value
2		Rio de Janeiro	RJ	05454-876	Brazil	Value
3	12	Resende	SP	08737-363	Brazil	Value
4	e. Carlos Soublette #8-35	San Cristóbal	Táchira	5022	Venezuela	Value
5	ra, 12	Rio de Janeiro	RJ	02389-673	Brazil	Value
6		Albuquerque	NM	87110	USA	Value
7	Grandes	Caracas	DF	1081	Venezuela	Value
8	.	Seattle	WA	98124	USA	Value
9		Lander	WY	82520	USA	Value
10		Albuquerque	NM	87110	USA	Value
11	e. Bolívar #65-98 Llano Largo	Barquisimeto	Lara	3508	Venezuela	Value
12	267	Rio de Janeiro	RJ	02389-890	Brazil	Value
13	23	Sao Paulo	SP	05432-043	Brazil	Value
14	ra, 12	Rio de Janeiro	RJ	02389-673	Brazil	Value
15	, 414	Sao Paulo	SP	05634-030	Brazil	Value
16		Albuquerque	NM	87110	USA	Value
17	e. Bolívar #65-98 Llano Largo	Barquisimeto	Lara	3508	Venezuela	Value
18	l	Cork	Co. Cork	null	Ireland	Value
19	267	Rio de Janeiro	RJ	02389-890	Brazil	Value
20		Anchorage	AK	99508	USA	Value
21	l.	Portland	OR	97219	USA	Value
22	l	Cork	Co. Cork	null	Ireland	Value
23	Suite 2	Portland	OR	97201	USA	Value
24						

Query Settings — Name: Orders

APPLIED STEPS: Source, Navigation, Filtered Rows

18 COLUMNS, 323 ROWS   Column profiling based on top 1000 rows   PREVIEW DOWNLOADED AT 02:10

Step 3:Load Data



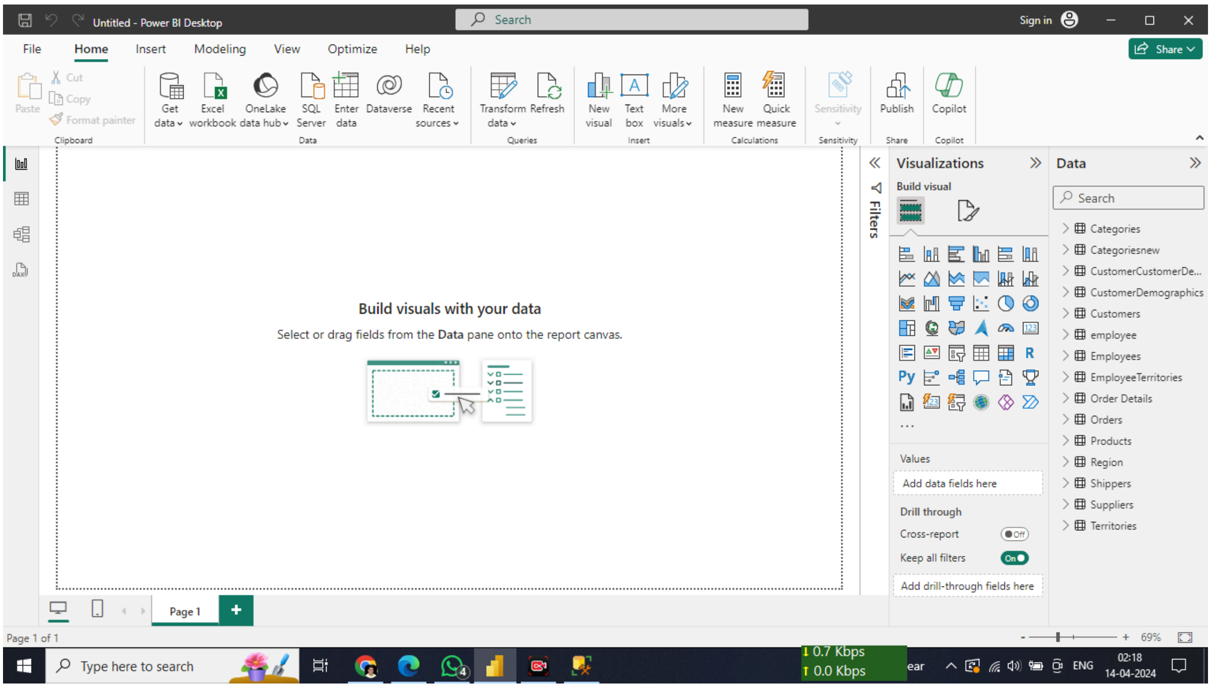
There are pending changes in your queries that haven't been applie

Load

- Territories
  Waiting for other queries...
- Suppliers
  Evaluating...
- Shippers
  Evaluating...
- Region
  Evaluating...
- Products
  Waiting for other queries...

Cancel

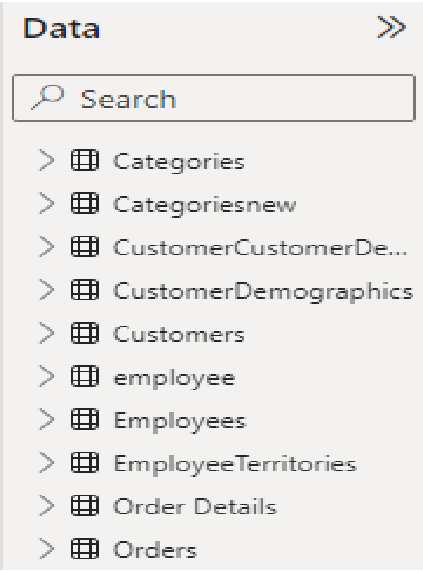
Data Loaded Successfully



Build visuals with your data

Select or drag fields from the **Data** pane onto the report canvas.

- Categories
- Categoriesnew
- CustomerCustomerDe...
- CustomerDemographics
- Customers
- employee
- Employees
- EmployeeTerritories
- Order Details
- Orders
- Products
- Region
- Shippers
- Suppliers
- Territories



Data

Search

- Categories
- Categoriesnew
- CustomerCustomerDe...
- CustomerDemographics
- Customers
- employee
- Employees
- EmployeeTerritories
- Order Details
- Orders

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import category_encoders as ce
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('car_evaluation.csv')
```

```
data.head()
```

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

data.columns=col_names
col_names
```

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1727 non-null   object
 1   maint     1727 non-null   object
 2   doors     1727 non-null   object
 3   persons   1727 non-null   object
 4   lug_boot  1727 non-null   object
 5   safety    1727 non-null   object
 6   class     1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```
data.isnull().sum()
```

```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

```
x=data.drop(['class'],axis=1)
y=data['class']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

```
x_train.shape,x_test.shape
```

```
((1208, 6), (519, 6))
```

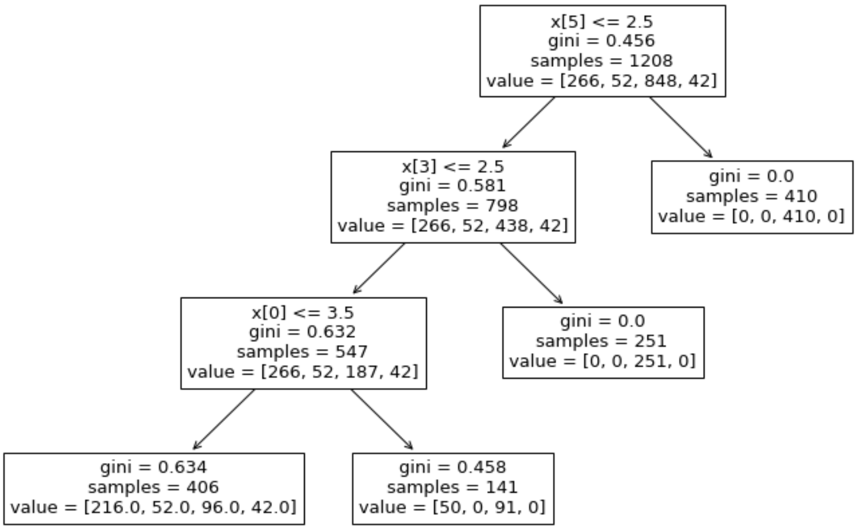
```
encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'])
x_train = encoder.fit_transform(x_train)
x_test = encoder.transform(x_test)
```

```
clf = DecisionTreeClassifier(criterion='gini',max_depth=3, random_state=0)

clf.fit(x_train,y_train)
```

```
▾            DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=0)
```

```
plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(clf.fit(x_train, y_train))
```

```
[Text(0.6666666666666666, 0.875, 'x[5] <= 2.5\ngini = 0.456\nsamples = 1208\nvalue = [266, 52, 848, 42]'),
 Text(0.5, 0.625, 'x[3] <= 2.5\ngini = 0.581\nsamples = 798\nvalue = [266, 52, 438, 42]'),
 Text(0.3333333333333333, 0.375, 'x[0] <= 3.5\ngini = 0.632\nsamples = 547\nvalue = [266, 52, 187, 42]'),
 Text(0.16666666666666666, 0.125, 'gini = 0.634\nsamples = 406\nvalue = [216.0, 52.0, 96.0, 42.0]'),
 Text(0.5, 0.125, 'gini = 0.458\nsamples = 141\nvalue = [50, 0, 91, 0]'),
 Text(0.6666666666666666, 0.375, 'gini = 0.0\nsamples = 251\nvalue = [0, 0, 251, 0]'),
 Text(0.8333333333333334, 0.625, 'gini = 0.0\nsamples = 410\nvalue = [0, 0, 410, 0]')]
```



```
y_pred = clf.predict(x_test)
```

```
print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with criterion gini index: 0.8150
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
iris_data = pd.read_csv("/content/Iris.csv")
X = iris_data.iloc[:, :-1]  # Features
y = iris_data.iloc[:, -1]
```

```
X.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2

Next steps: Generate code with X View recommended plots

```
y.head()
```

```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: Species, dtype: object
```

```
X.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
y.describe()
```

```
count             150
unique              3
top       Iris-setosa
freq               50
Name: Species, dtype: object
```

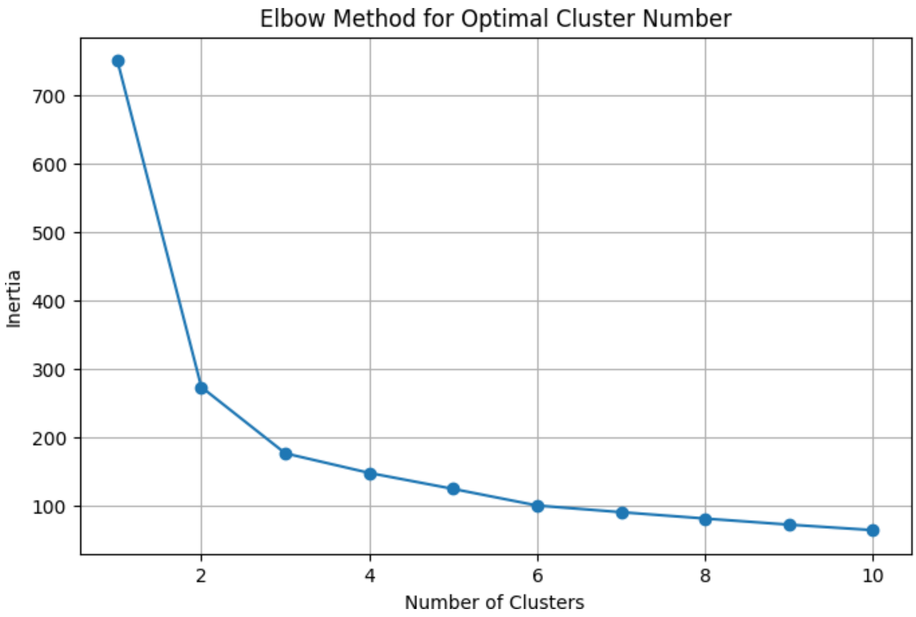
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
```

```
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
```

```
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal Cluster Number')
plt.grid(True)
plt.show()
```

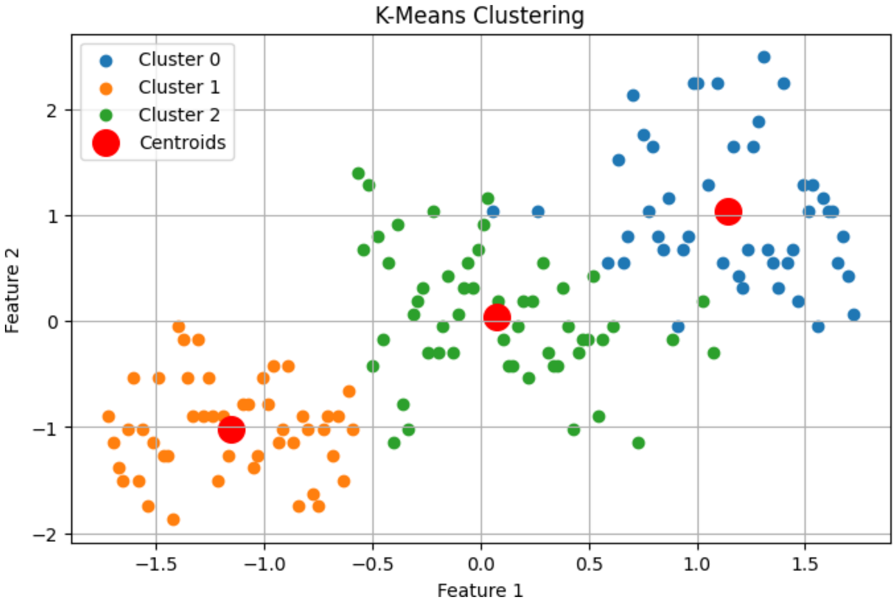


```
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(X_scaled)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
▾               KMeans
KMeans(n_clusters=3, random_state=42)
```

```
iris_data['Cluster'] = kmeans.labels_
centroids = kmeans.cluster_centers_
```

```
plt.figure(figsize=(8, 5))
for i in range(optimal_k):
    plt.scatter(X_scaled[iris_data['Cluster'] == i, 0], X_scaled[iris_data['Cluster'] == i, 1], label=f'Cluster {i}')
plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c='red', label='Centroids')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('K-Means Clustering')
plt.legend()
plt.grid(True)
plt.show()
```


K-Means Clustering
Cluster 0
Cluster 1
Cluster 2
Centroids
Feature 2
Feature 1
2
1
0
−1
−2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

ASSIGNMENT 1

PROBLEM STATEMENT

Mongo DB: Installation and Creation of database and Collection CRUD Document: Insert, Query, Update and Delete Document

OBJECTIVE

1. Understand the installation process of MongoDB and its basic configuration settings.
2. Learn how to create databases and collections in MongoDB to organize data efficiently.
3. Master the CRUD operations (Create, Read, Update, Delete) for managing documents within MongoDB collections.

THEORY

MongoDB Overview: Introduction to MongoDB as a NoSQL document-oriented database, highlighting its advantages and use cases.

Installation Process: Explanation of the steps required to download, install, and configure MongoDB on various operating systems.

Database and Collection Creation: Description of how to create databases and collections in MongoDB using the MongoDB Shell or graphical user interfaces (GUIs) like MongoDB Compass.

CRUD Operations: Overview of the four fundamental CRUD operations in MongoDB:

- Insert: Adding new documents to a collection.

- Query: Retrieving documents from a collection based on specified criteria.

- Update: Modifying existing documents in a collection.

- Delete: Removing documents from a collection.

Step 1: MongoDB Installation on Windows: Download the MongoDB Community Server from the MongoDB Download Center. Run the installer and follow the setup wizard. Add MongoDB's bin folder to the PATH environment variable for easy commandline access.

https://www.mongodb.com/try/download/community

Step 2: Create a Database and Collection:

Switch to Your New Database:

- use myNewDatabase Create a Collection by Inserting a Document:

- 48 db.myNewCollection.insertOne({name: "John Doe", age: 30}) MongoDB creates the database and collection upon inserting the first document.

Step 3: CRUD Operations

Create (Insert Document): Insert a single document:

- db.myNewCollection.insertOne({name: "Jane Doe", age: 25})

- Read (Query Document): Find one document: db.myNewCollection.findOne({name: "John Doe"})

- Update Document: Update a single document: db.myNewCollection.update One ({name: "John Doe"}, {$set: {age: 31}})

- Delete Document: Delete a single document:

db.myNewCollection.deleteOne({name: "Bob"})

CONCLUSION

The guide provides a step-by-step approach to installing MongoDB, creating databases and collections, and performing CRUD operations on documents. By mastering these fundamental operations, users can harness the power and flexibility of MongoDB for storing and managing data efficiently. This serves as a foundation for further exploration of MongoDB's advanced features and capabilities in application development and data management.

ORAL QUESTION

1. How do you create a new database in MongoDB?
2. What are the common data types supported in MongoDB documents?
3. What does CRUD stand for in the context of databases?