Assignment 1

1) Based on the following table, design the three stages of reproducible workflow, includes the work you can do and the folder structure in each stage (reference study case in chapter 3). (5 points)

Height (Inches)	Weight (Pounds)	Age	Grip strength	Frailty
65.8	112	30	30	N
71.5	136	19	31	N
69.4	153	45	29	N
68.2	142	22	28	Y
67.8	144	29	24	Y
68.7	123	50	26	N
69.8	141	51	22	Y
70.1	136	23	20	Y
67.9	112	17	19	N
66.8	120	39	31	N

Solution:

Githib Repo Link: https://github.com/vn97d/Assignment1

Stage 1: Data Acquisition

Intensive data workflows involve raw data acquisition is the first stage. For example, imagine a study in which we collected field data on human measurements.

Table denotes measurements of humans who came for health checkup, in height per inches,In weight per pounds, Age, Grip Strength and Frailty of measurements in each of fields having no Frailty (N), with conventional management Frailty(Y).

Sample Human Measurements Data Set:

Height (Inches)	Weight (Pounds)	Age	Grip strength	Frailty
65.8	112	30	30	N
71.5	136	19	31	N
69.4	153	45	29	N
68.2	142	22	28	Y
67.8	144	29	24	Y
68.7	123	50	26	N
69.8	141	51	22	Y
70.1	136	23	20	Y
67.9	112	17	19	N
66.8	120	39	31	N

It is best to enter and save this data as a CSV file using a spreadsheet tool. For storing tabular data, CSV files are a popular plain text format where each row of a table is on a distinct line and the data for each column is separated by a comma

This file has to be named, then stored in a suitable location after being created. For instance, this table may be stored as raw_yield_data.csv.

A metadata file must be created and stored along with the data at the same time. Keeping track of the data's origin and any pertinent details is what the metadata file is for. A file of this nature, which we can store as README.txt along with the data file.

```
|-- Human_Measurements

|   |-- data_raw
|   |   |-- raw_yield_data.csv
|   |   |-- README.txt
|   |-- data_clean
|   |-- results
|   |-- src
```

Stage 2: Data Processing
Raw data almost always needs to be processed or cleaned before it can be utilized in an analysis after it has been gathered and stored in a project directory. This stage could entail eliminating useless data, subsetting the original data, eliminating outliers, and other related procedures. Therefore, the ideal method for handling raw data collection depends on the questions that a researcher wishes to address with this data and the specific sort of analysis intended for Stage 3.

In this case, a look at the table of raw data revealed measurements of people, which we might want to take out of the data before doing any more analysis. We also know that we can eliminate the no treatment from the table at this time given the ultimate purpose of performing a two-sample t-test comparing the conventional Frailty. Although such subsetting can increase the effectiveness of subsequent analysis of larger data sets, it is not technically required to remove these rows from a small table like this one.

We can easily create a quick script that reads the raw table, filters out the rows with NA yields and those with a field code of N, and saves the processed data as a result.

```
yield_data <- read.csv("yield_data.csv")
clean_yield_data <- na.omit(raw_yield_data[raw_yield_data$Field
!="N",])
write.csv(clean_yield_data, "clean_yield_data.csv")
```

These instructions should all be contained in a separate file that, when run, will read the raw data, process it, and then write the produced data file.

We may tweak the code to the following to make sure that a script in the src directory finds and saves the necessary files in the appropriate folders.

Scripts and other code are saved in the src subdirectory of the straightforward directory structure previously defined. We can change the code above to the code below, which changes the places where the files are read and written, to make sure that a script in the src directory will find and save the necessary files in the relevant folders. Furthermore, take note that we have added comments explaining what each line of code is supposed to accomplish.

```
### Read in the raw data, assuming we are working in the src directory
raw_yield_data <- read.csv("../data_raw/raw_yield_data.csv")

### Clean the data by removing rows with NA and where 'Field' == N
clean_yield_data <- na.omit(raw_yield_data[raw_yield_data$Field !=
"N",])

### Write the clean data to disk
write.csv(clean_yield_data, "../data_clean/clean_yield_data.csv")
```

The above instructions, when stored as a script called clean_data.R in the src subdirectory, will read the raw_yield_data.csv table from the data raw subfolder, clean it, and then save the cleaned table as clean_yield_data.csv in the data clean subfolder. To avoid ever confusing the original, raw data with any derived data products, the cleaned data is stored in a separate subdirectory from the raw data.

```
|-- Human_Measurements
|   |-- data_raw
|   |   |-- raw_yield_data.csv
|   |   |-- README.txt
|   |-- data_clean
|   |   |-- clean_yield_data.csv
|   |-- results
|   |-- src
|   |   |-- clean_data.R
```

Stage 3: Data Analysis

Data analysis is the third step in the fundamental reproducible workflow after data has been verified in and processed. Naturally, there are numerous various sorts of analyses that might be used in this situation, and numerous different types of outputs, including text-based results, tables, and figures, could be produced.

The script analysis.R in the src directory is where the following code should be saved. When run, it will read the cleaned data table, run the chosen t-test, and save the test's compiled results in the results subfolder as a plain text file called test results.txt. Tables and figures, among other findings, should also be kept in the results subdirectory, albeit they are not relevant here.

```
### Load clean data

clean_yield_data <- read.csv("../data_clean/clean_yield_data.csv")


### t-test of Weights by Field

typet_test_Weight_Field <- with(clean_yield_data, t.test(Weight ~ Field)


### Write test result to plain text file
capture.output(t_test_Weight_Field, file = "../results/test_results.txt")
```

At the conclusion of this stage, after the script analysis.R has been run in the same manner as the previous clean_data.R script, the project directory will appear as follows.

```
|-- Human_Measurements
|   |-- data_raw
|   |   |-- raw_yield_data.csv
|   |   |-- README.txt
|   |-- data_clean
|   |   |-- clean_yield_data.csv
|   |-- results
|   |   |-- test_results.txt
|   |-- src
|   |   |-- analysis.R
|   |   |-- clean_data.R
```

2) Perform 5 data visualization tasks on the student performance dataset given in the link below (create 5 different visualizations). Explain what kind analysis has become easier with each of the visualizations. Create the folder structure for this question similar to question 1. (15 points).

Solution:

Data link: https://app.box.com/s/ji910ez3ycw137rw07xnhielxey7ww41

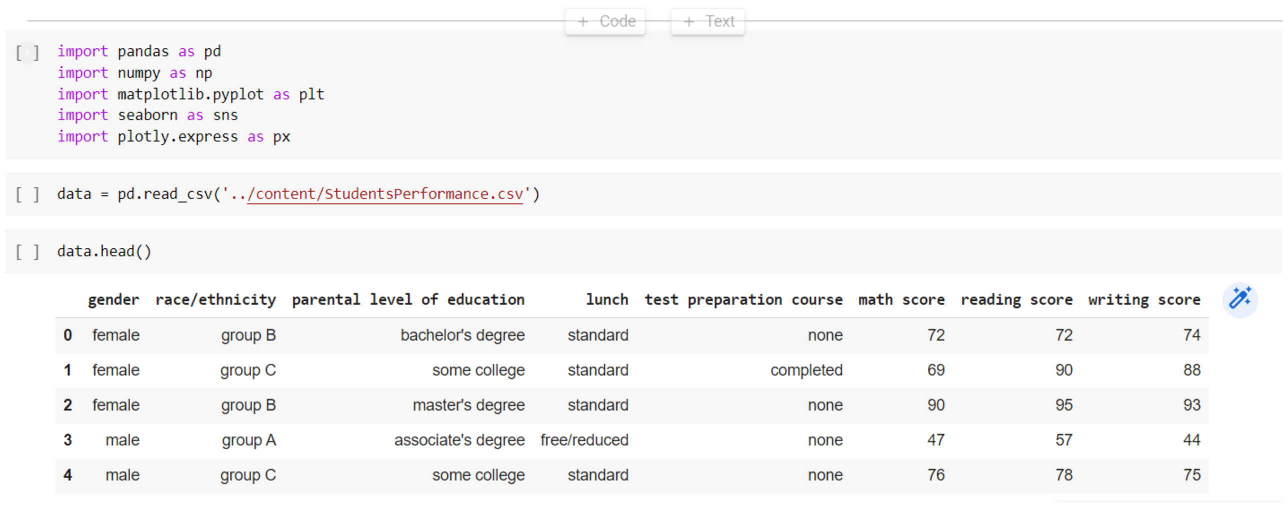
GitHub Repo Link: https://github.com/vn97d/Assignment1

The distribution of values for data points is one crucial application for visualizations.

Understanding the relationships between data features is another challenge that comes up during data exploration.

The type of variables you are looking at and what you hope to learn from them will determine the best chart for the job. It is preferable to utilize numerous plots to establish comparisons, show trends, and illustrate relationships between various variables than trying to make each plot as simple and unambiguous as possible.

5 data visualization tasks on the student performance dataset

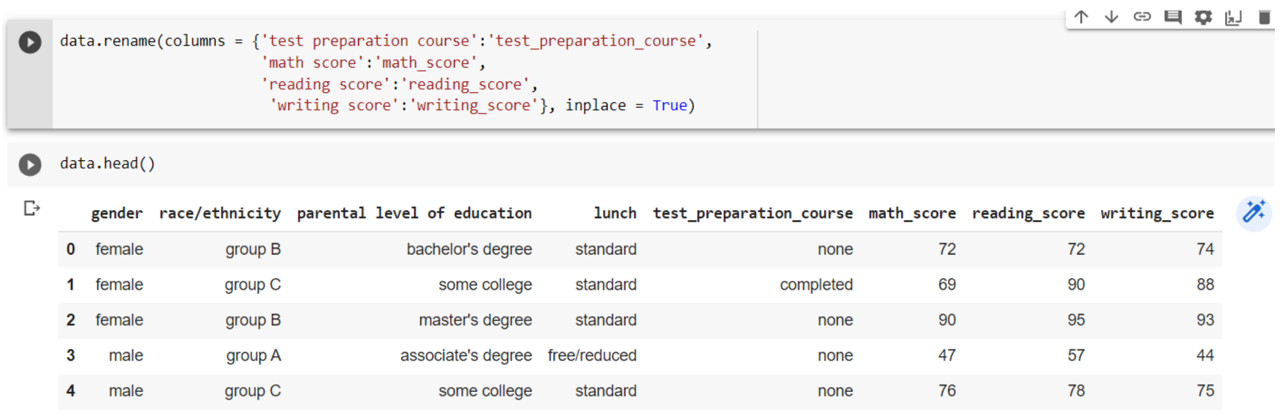


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
data = pd.read_csv('../content/StudentsPerformance.csv')
```

```
data.head()
```

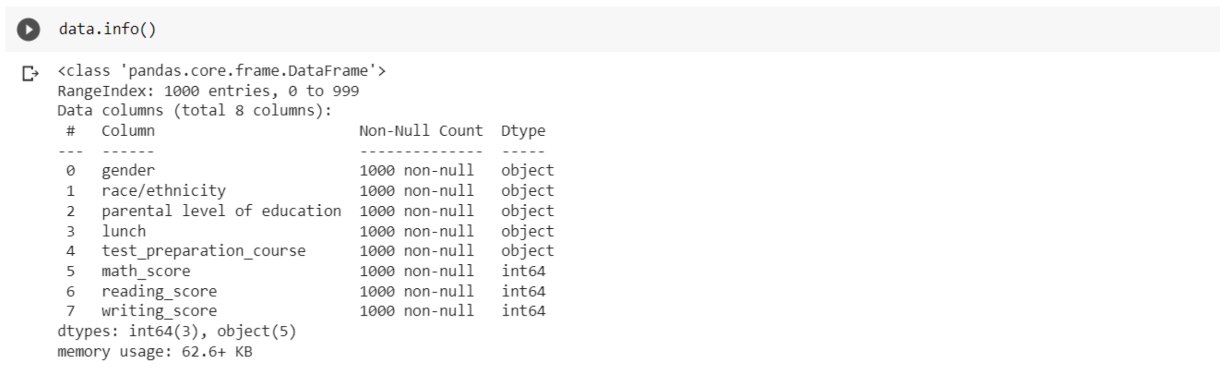
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75



```python
data.rename(columns = {'test preparation course':'test_preparation_course',
                       'math score':'math_score',
                       'reading score':'reading_score',
                        'writing score':'writing_score'}, inplace = True)
```

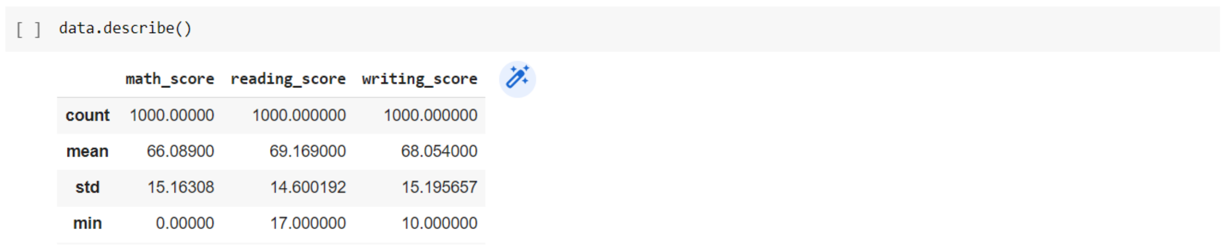
```python
data.head()
```

	gender	race/ethnicity	parental level of education	lunch	test_preparation_course	math_score	reading_score	writing_score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75



```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test_preparation_course      1000 non-null   object
 5   math_score                   1000 non-null   int64 
 6   reading_score                1000 non-null   int64 
 7   writing_score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

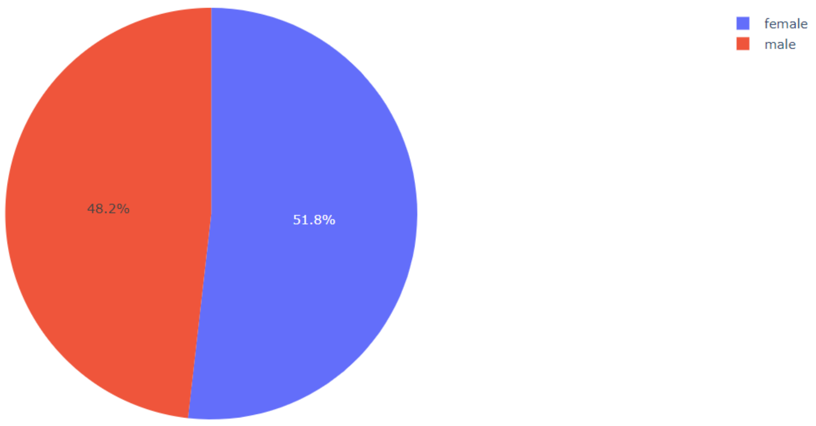


```python
data.describe()
```

	math_score	reading_score	writing_score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000

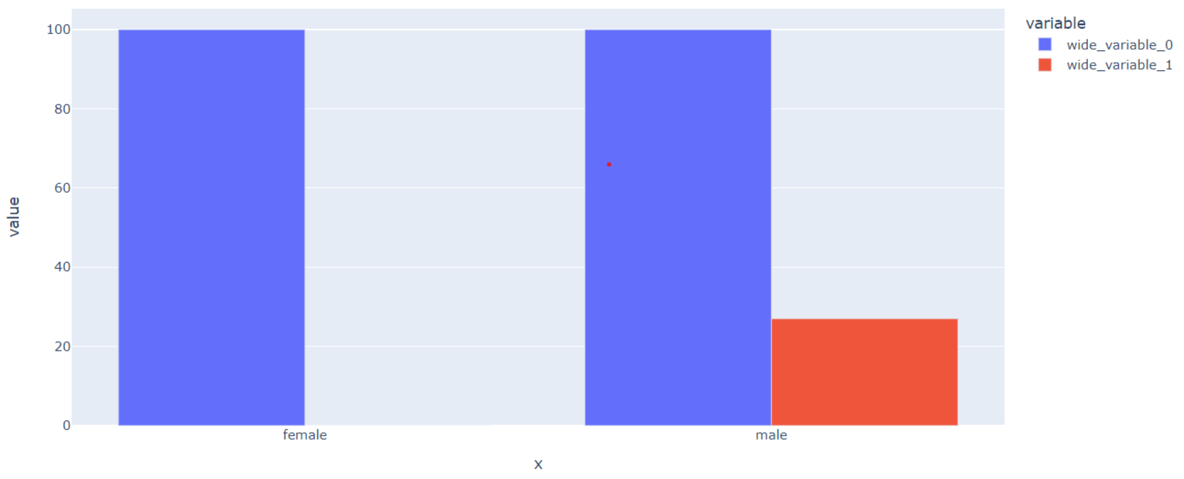
PIE:

```
fig1 = px.pie(data, values = data['gender'].value_counts().values, names = data['gender'].value_counts().index)
fig1.show()
```



BAR :

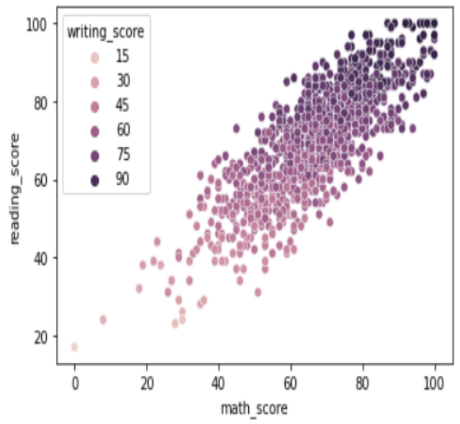
```
fig6 = px.bar(x = data.groupby('gender').math_score.max().index, y = [data.groupby('gender').math_score.max().values, data.groupby('gender').ma
fig6.show()
```



SCATTERPLOT:

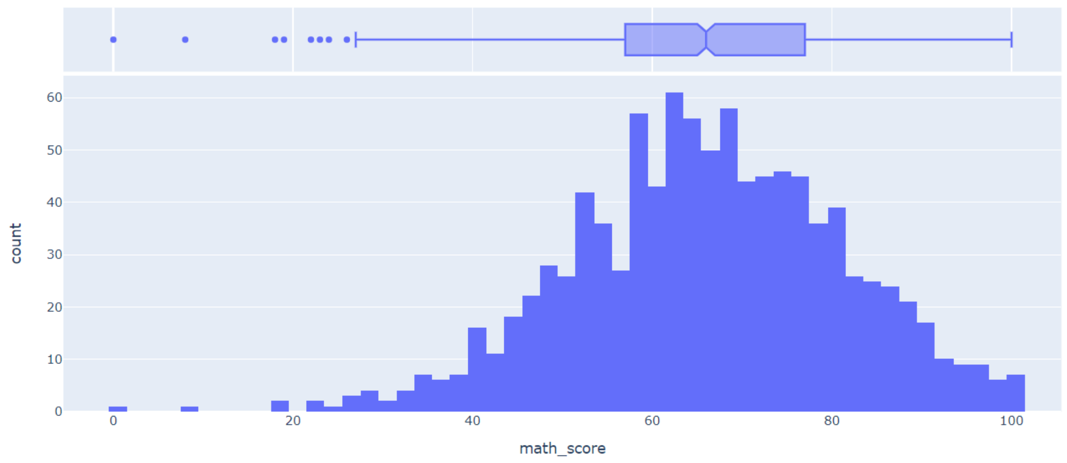
```
[ ] sns.scatterplot(x = data.math_score, y = data.reading_score, hue = data.writing_score)
```

```
<AxesSubplot:xlabel='math_score', ylabel='reading_score'>
```



HISTOGRAM:

```
fig10 = px.histogram(data, x="math_score", marginal = 'box')
fig10.show()
```

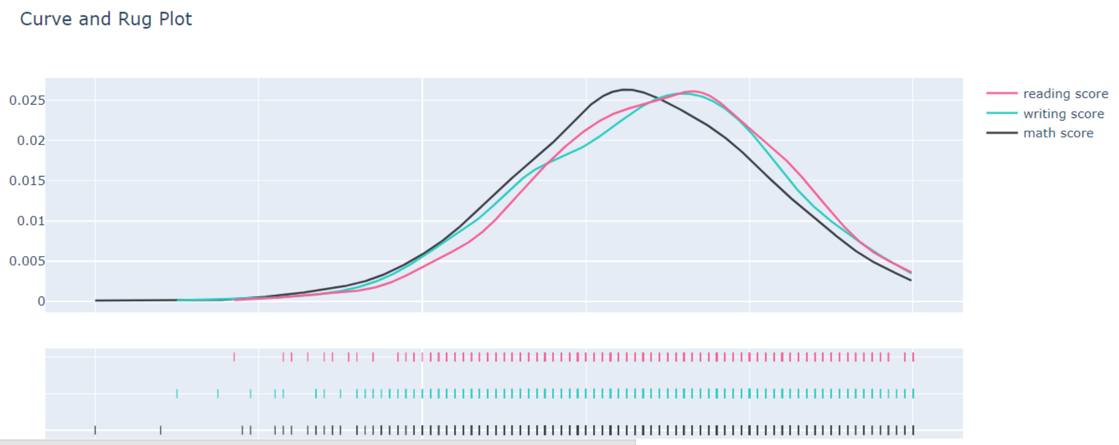


CURVE AND RUG PLOT:

```python

# Create distplot with curve_type set to 'normal'
fig = ff.create_distplot(hist_data, group_labels, show_hist=False, colors=colors)

# Add title
fig.update_layout(title_text='Curve and Rug Plot')
fig.show()
```


Curve and Rug Plot
reading score
writing score
math score