EDA of Student Performance Data - Fazal Rehman

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv("C:/Users/fazal/Downloads/stud.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course
0	female	group B	bachelor's degree	standard	none
1	female	group C	some college	standard	completed
2	female	group B	master's degree	standard	none
3	male	group A	associate's degree	free/reduced	none
4	male	group C	some college	standard	none

In [5]:
```
df.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
```

In [6]:
```
df.describe()
```

Out[6]:

	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
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

In [8]:
```
df.shape
```

Out[8]:
```
(1000, 8)
```

Data Checks to Perform.

1. Check Missing Values,
2. Check Duplicates,
3. Check data type,
4. check the number of unique values of each column,
5. check statistics of data set,
6. check various categories present in the different categorical column

In [9]:
```
df.isnull().sum() #missing values
```

Out[9]:
```
gender                         0
race_ethnicity                 0
parental_level_of_education    0
lunch                          0
test_preparation_course        0
math_score                     0
reading_score                  0
writing_score                  0
dtype: int64
```

In [11]:
```
df.duplicated().sum() #check duplicates
```

Out[11]:
```
0
```

In [12]:
```
df.duplicated()
```

```
Out[12]: 0      False
         1      False
         2      False
         3      False
         4      False
                ...
         995    False
         996    False
         997    False
         998    False
         999    False
         Length: 1000, dtype: bool
```

```
In [13]: df.nunique()
```

```
Out[13]: gender                         2
         race_ethnicity                 5
         parental_level_of_education    6
         lunch                          2
         test_preparation_course        2
         math_score                    81
         reading_score                 72
         writing_score                 77
         dtype: int64
```

```
In [24]: [feature for feature in df.columns if df[feature].dtype=='O']
         [feature for feature in df.columns if df[feature].dtype!='O']
```

```
Out[24]: ['math_score', 'reading_score', 'writing_score']
```

```
In [25]: #Segrregate numerical and categorical features
         num_features=[feature for feature in df.columns if df[feature].dtype!='O']
         cat_features=[feature for feature in df.columns if df[feature].dtype=='O']
```

```
In [26]: df['gender'].value_counts()
```

```
Out[26]: gender
         female    518
         male      482
         Name: count, dtype: int64
```

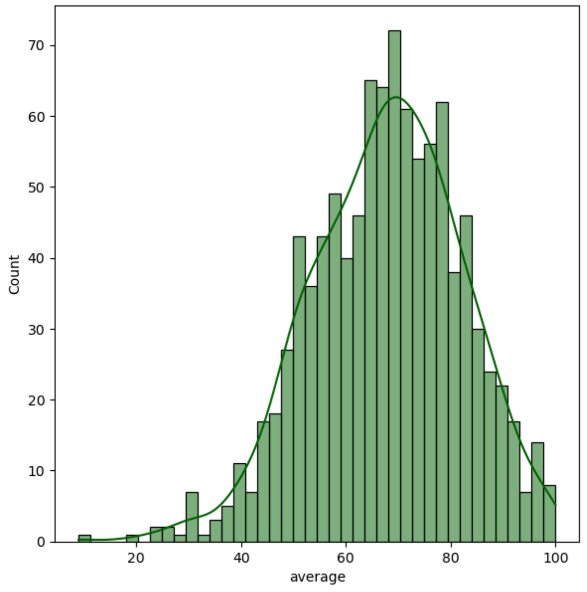
```
In [56]: df["total_score"]=(df["math_score"]+df["reading_score"]+df["writing_score"])
         df["average"]=df["total_score"]/3
         df.head()
```

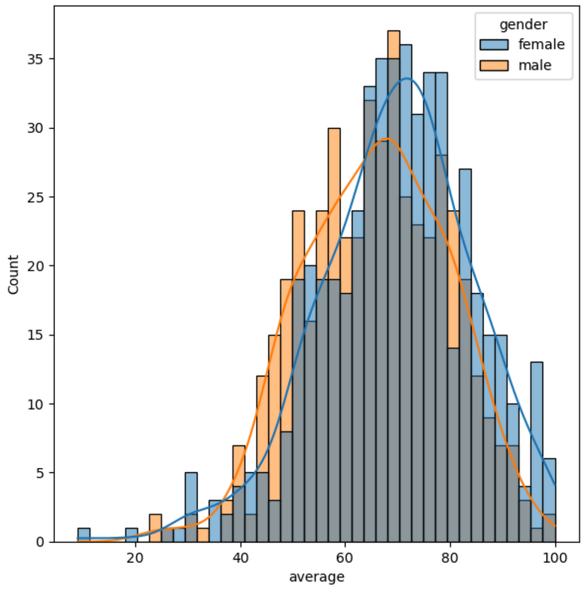
Out[56]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course
0	female	group B	bachelor's degree	standard	none
1	female	group C	some college	standard	completed
2	female	group B	master's degree	standard	none
3	male	group A	associate's degree	free/reduced	none
4	male	group C	some college	standard	none

In [32]:
```python
#Exploring more Visualisation
fig,axis=plt.subplots(1,2,figsize=(15,7))
plt.subplot(121)
sns.histplot(data=df,x="average",bins=40,kde=True,color="darkgreen")
plt.subplot(122)
sns.histplot(data=df,x="average",bins=40,kde=True,hue="gender")
```

Out[32]:
```
<Axes: xlabel='average', ylabel='Count'>
```



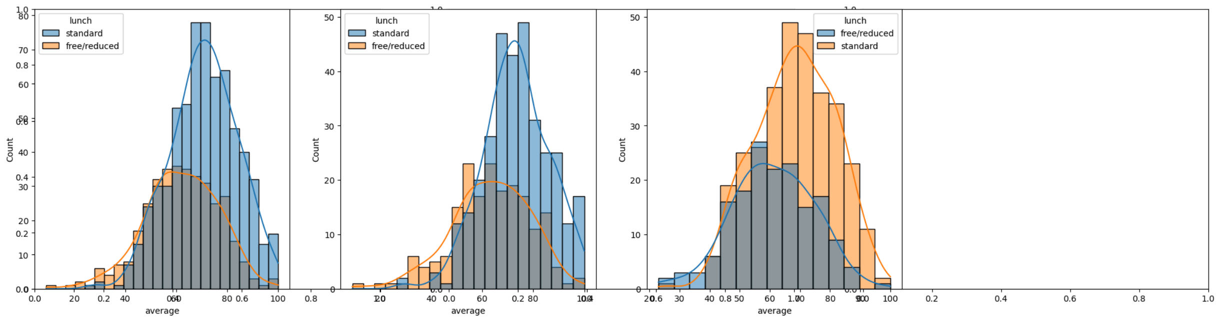


Interpretation -

Female Students tend to perform well than male students

In [40]:
```python
plt.subplots(1,3,figsize=(25,6))
plt.subplot(141)
sns.histplot(data=df,x="average",kde=True,hue="lunch")
plt.subplot(142)
sns.histplot(data=df[df.gender=="female"],x="average",hue="lunch",kde=True)
plt.subplot(143)
sns.histplot(data=df[df.gender=="male"],x="average",hue="lunch",kde=True)
```

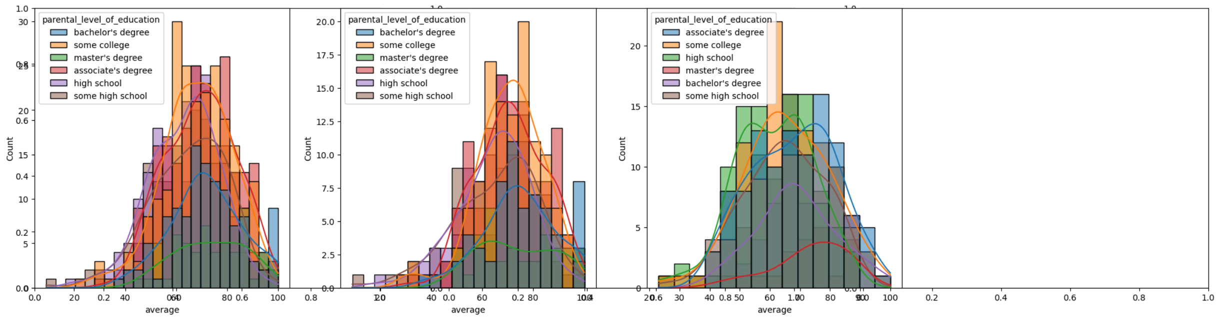
Out[40]:
```
<Axes: xlabel='average', ylabel='Count'>
```



Interpretation -

1. Standard Lunch help students perform well in the exams
2. Standard Lunch help students perform well in the exams be it male or female

In [42]:
```
plt.subplots(1,3,figsize=(25,6))
plt.subplot(141)
sns.histplot(data=df,x="average",kde=True,hue="parental_level_of_education")
plt.subplot(142)
sns.histplot(data=df[df.gender=="female"],x="average",hue="parental_level_of_educat
plt.subplot(143)
sns.histplot(data=df[df.gender=="male"],x="average",hue="parental_level_of_educatio
plt.show()
```



Interpretation -

1. In general parent's education doesn't help student perform well in exam.
2. 3rd plot shows that parent's whose education is of associate's degrees or masters degree their male child tend to perform well in the exam
3. 2nd plot shows there is no effect of parent's education on female students

In [47]:
```
sns.heatmap(df.corr()
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[47], line 1
----> 1 sns.heatmap(df.corr())

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:10704, in DataFrame.corr(sel
f, method, min_periods, numeric_only)
  10702 cols = data.columns
  10703 idx = cols.copy()
> 10704 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10706 if method == "pearson":
  10707     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:1889, in DataFrame.to_numpy
(self, dtype, copy, na_value)
   1887 if dtype is not None:
   1888     dtype = np.dtype(dtype)
-> 1889 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1890 if result.dtype is not dtype:
   1891     result = np.array(result, dtype=dtype, copy=False)

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1656, in BlockM
anager.as_array(self, dtype, copy, na_value)
   1654         arr.flags.writeable = False
   1655 else:
-> 1656     arr = self._interleave(dtype=dtype, na_value=na_value)
   1657     # The underlying data was copied within _interleave, so no need
   1658     # to further copy if copy=True or setting na_value
   1660 if na_value is lib.no_default:

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1715, in BlockM
anager._interleave(self, dtype, na_value)
   1713     else:
   1714         arr = blk.get_values(dtype)
-> 1715     result[rl.indexer] = arr
   1716     itemmask[rl.indexer] = 1
   1718 if not itemmask.all():

ValueError: could not convert string to float: 'female'
```

In [58]:
```
df1=df[["math_score","reading_score","writing_score","total_score","average"]]
```

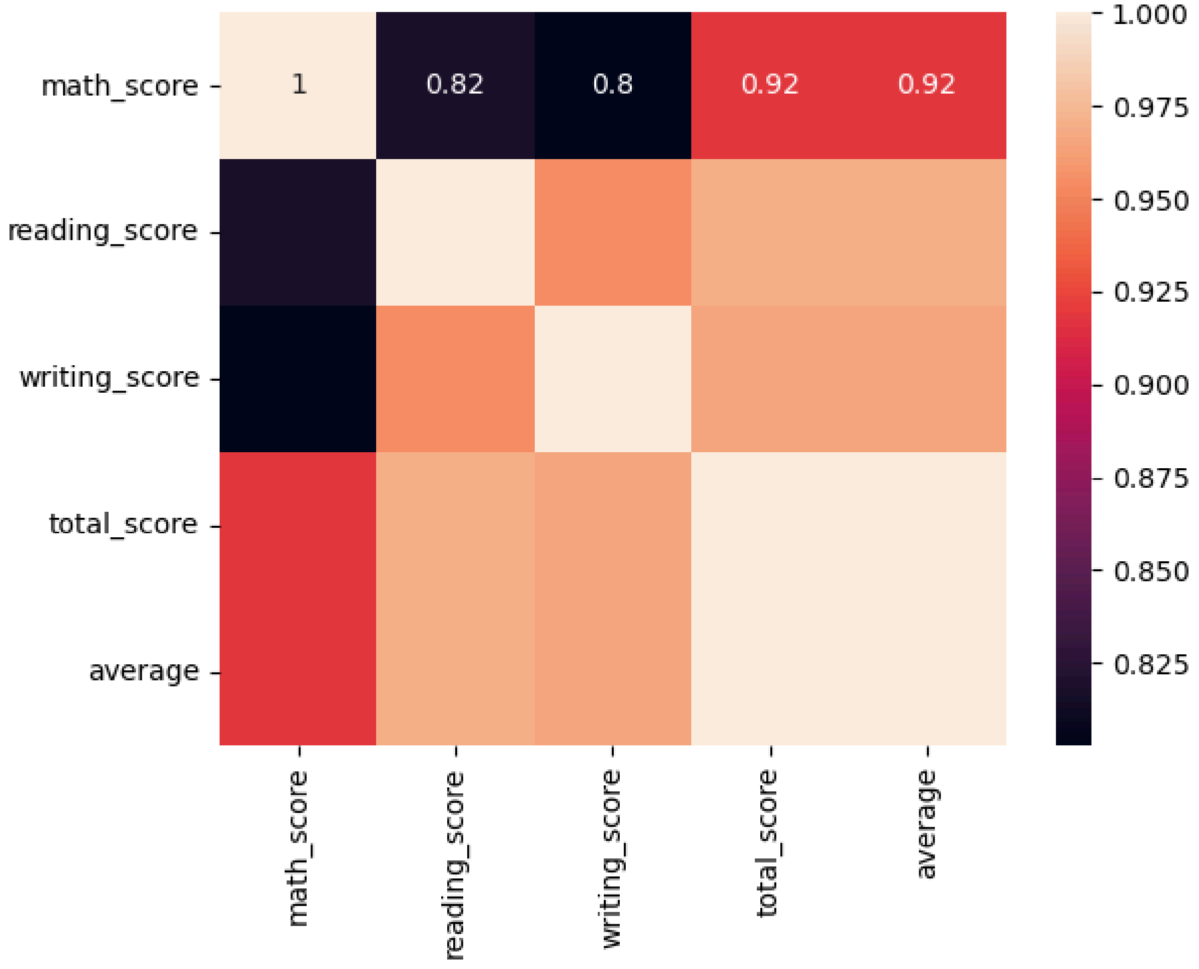
In [59]:
```
df1.corr()
```

Out[59]:

	math_score	reading_score	writing_score	total_score	average
math_score	1.000000	0.817580	0.802642	0.918746	0.918746
reading_score	0.817580	1.000000	0.954598	0.970331	0.970331
writing_score	0.802642	0.954598	1.000000	0.965667	0.965667
total_score	0.918746	0.970331	0.965667	1.000000	1.000000
average	0.918746	0.970331	0.965667	1.000000	1.000000

```
In [60]: sns.heatmap(df1.corr(),annot=True)
```

```
Out[60]: <Axes: >
```


math_score
reading_score
writing_score
total_score
average