Predictive Modeling on Student Performance Data

Instrucions For project:

1. Download the "Student Performance" dataset from Kaggle: https://www.kaggle.com/spscientist/students-performance-in-exams
2. Build a predictive model to predict the ***math score*** of a student based on other variables such as gender, race/ethnicity, parental level of education, lunch, and test preparation course.
3. **Use** supervised learning algorithms such as **decision trees, logistic regression, and random forests** to build the model.
4. **Evaluate** the model **using appropriate metrics such as** accuracy, precision, recall, and F1 score.
5. **Use feature selection techniques** to identify the most important features for predicting the math score.
6. Submit the Jupyter Notebook and report as the deliverables.

1) Importing necessary modules

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
from sklearn.linear_model import LinearRegression
```

2) Loading Data into a dataframe

In [2]:
```
# some formatting
%matplotlib inline
sns.set(rc={'figure.figsize': [9, 9]}, font_scale=1.2)

# Load the dataset in csv file
df = pd.read_csv('StudentsPerformance.csv')
```

3) Data Preprocessing

3.1) Getting info about null values and columns

In [3]:
```
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
```

3.2) We need to rename the columns

In [4]:
```
df.rename(columns = {'gender':'Gender',
                     'race/ethnicity':'Race',
                     'parental level of education':'ParentEducation',
                     'lunch':'Lunch',
                     'test preparation course':'Course',
                     'math score':'Math',
                     'reading score':'Reading',
                     'writing score':'Writing'
                    }, inplace = True)
df.head()
```

Out[4]:

	Gender	Race	ParentEducation	Lunch	Course	Math	Reading	Writing
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

3.3) Lets see an overview of data

In [5]:
```
df.describe()
```

Out[5]:

	Math	Reading	Writing
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

3.3) Lets see the value counts of different values in columns

In [6]:
```
df['Race'].value_counts()
```

Out[6]:
```
group C    319
group D    262
group B    190
group E    140
group A     89
Name: Race, dtype: int64
```

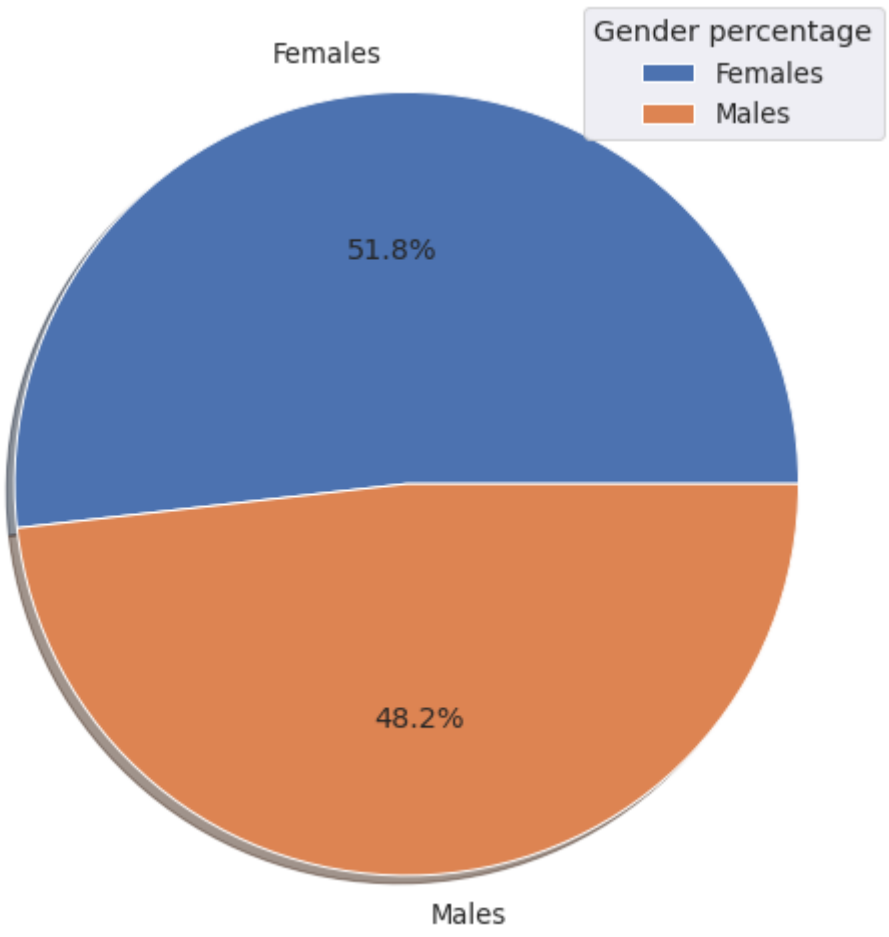
In [7]:
```
df['ParentEducation'].value_counts()
```

Out[7]:
```
some college          226
associate's degree    222
high school           196
some high school      179
bachelor's degree     118
master's degree        59
Name: ParentEducation, dtype: int64
```

4) Data Visualisation

4.1) Gender Percentage

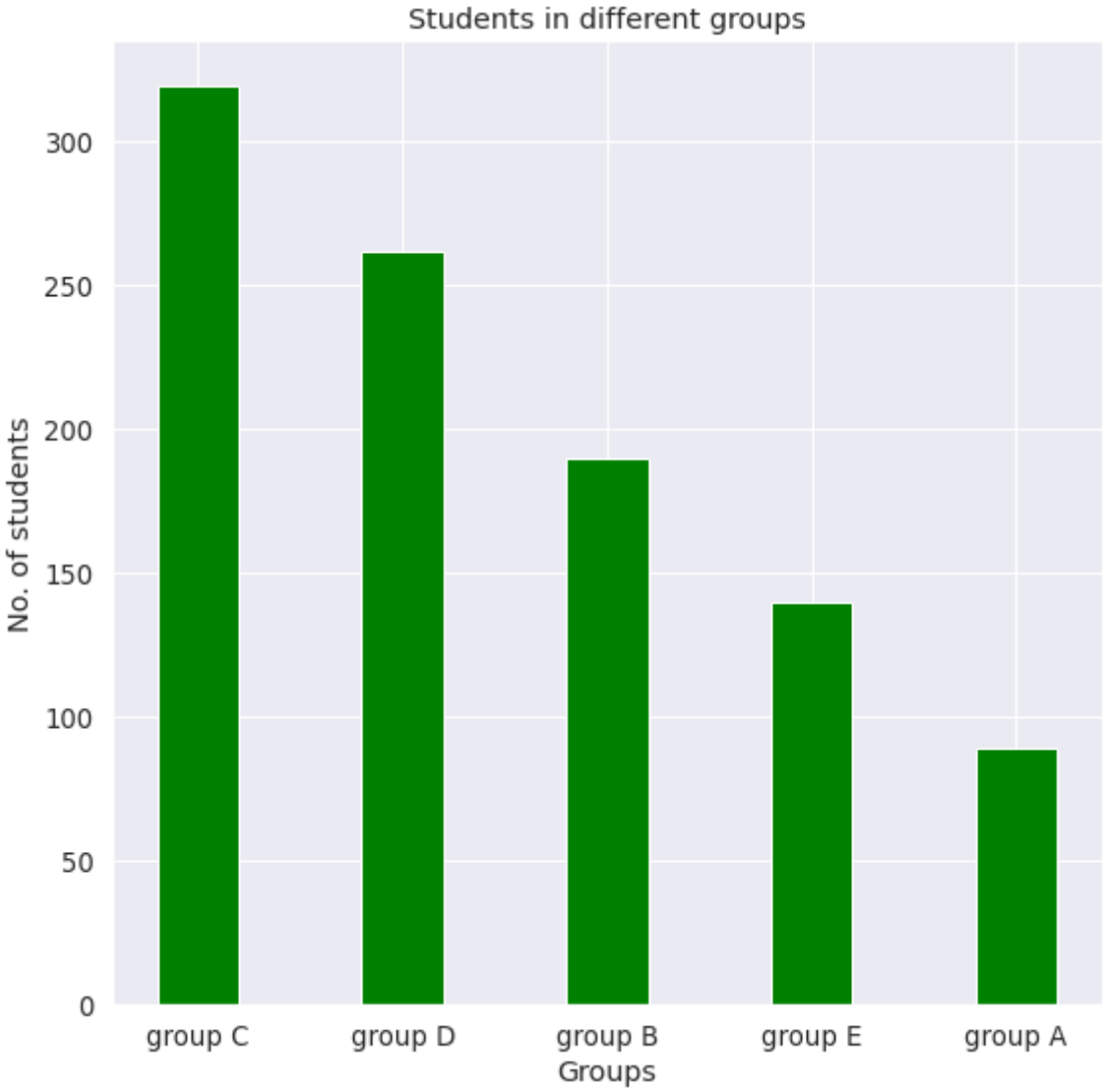
In [8]:
```
plt.pie(df['Gender'].value_counts(),labels=['Females','Males'],shadow = True,
plt.legend(title = "Gender percentage")
plt.show()
```



4.2) No. of students in each group

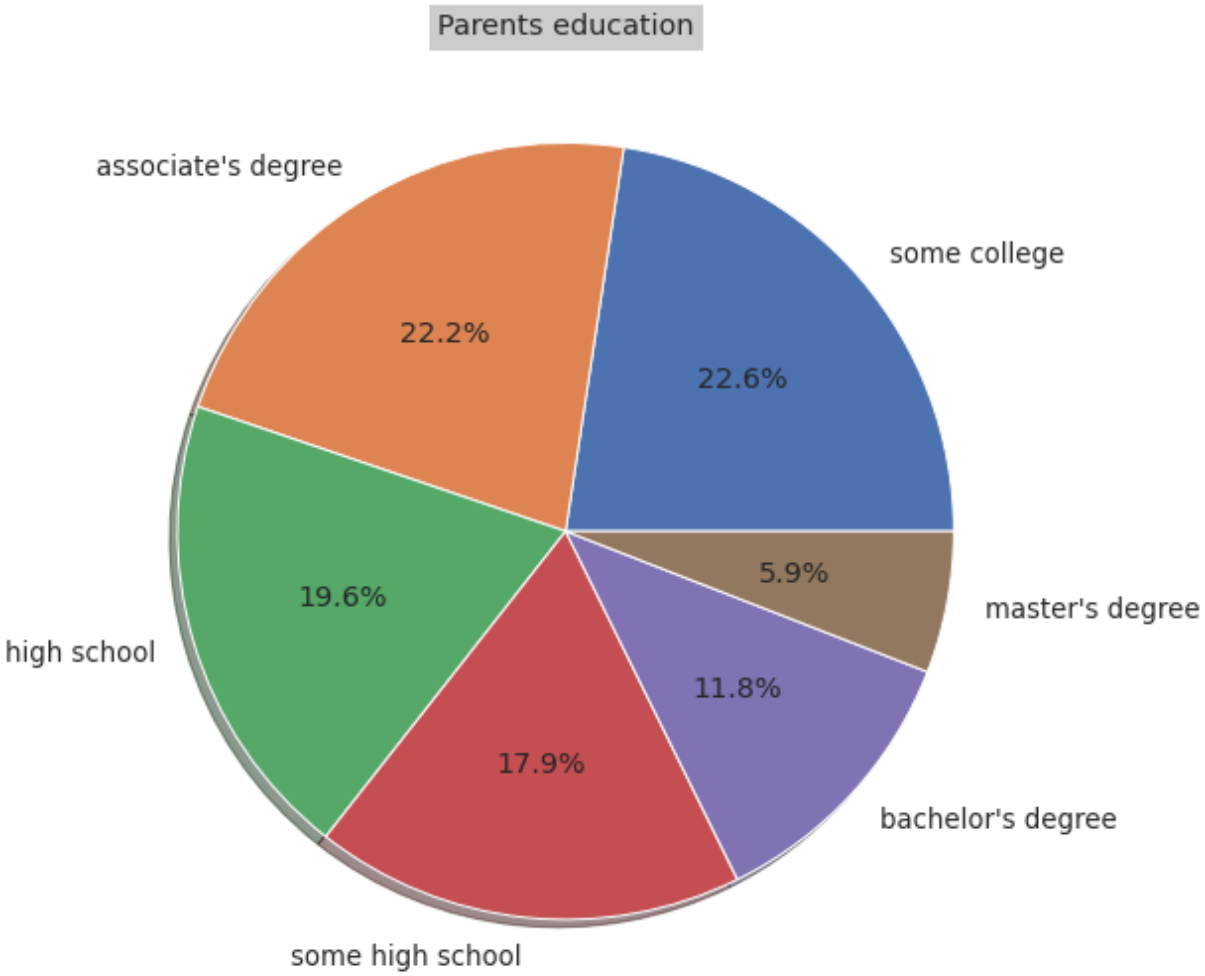
In [9]:
```
plt.bar(df['Race'].value_counts().index, df['Race'].value_counts(), color ='g
        width = 0.4)

plt.xlabel("Groups")
plt.ylabel("No. of students")
plt.title("Students in different groups")
plt.show()
```



4.3) Parental education

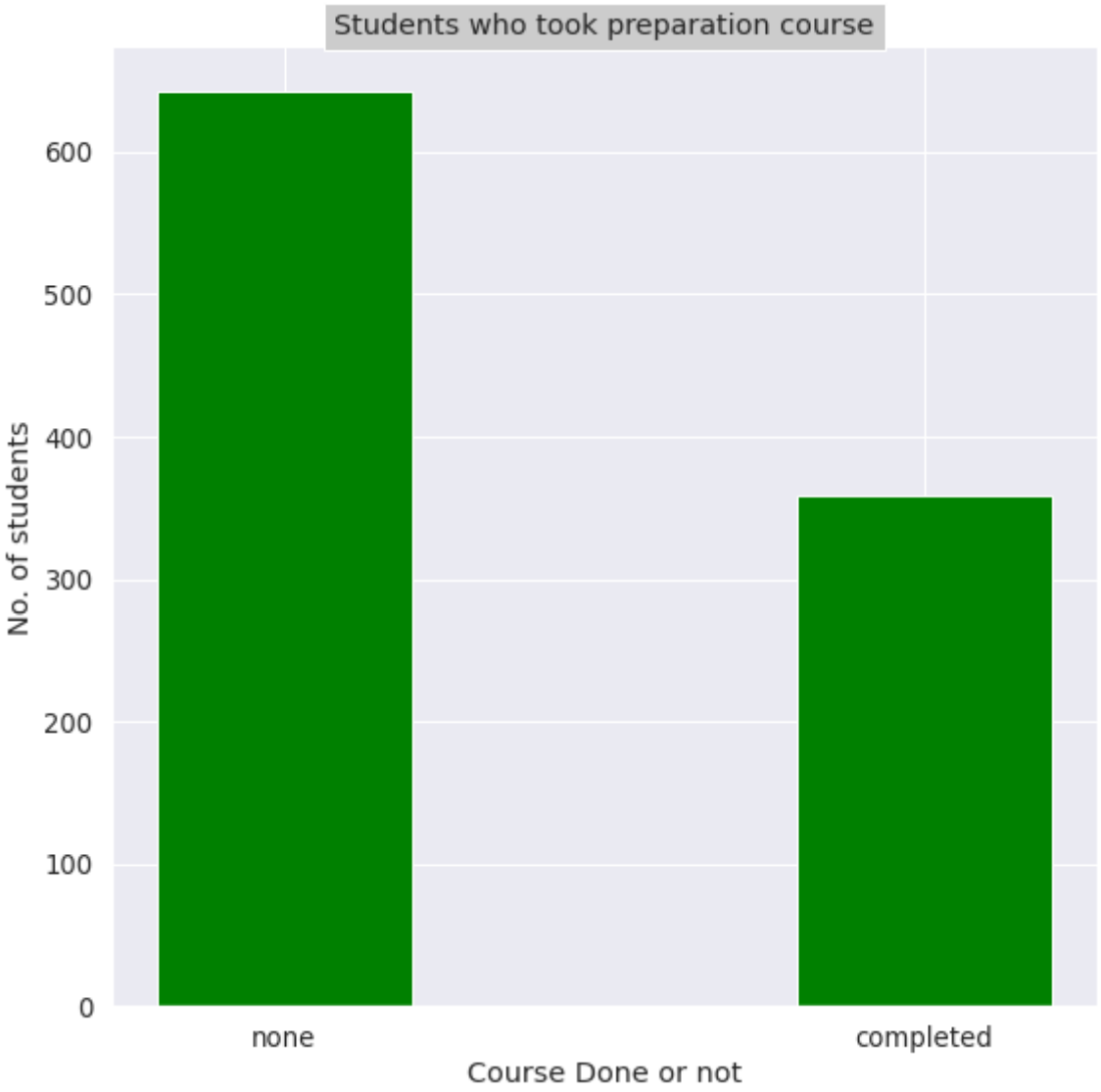
In [10]:
```
plt.pie(df['ParentEducation'].value_counts(),labels=df['ParentEducation'].val
plt.title("Parents education", bbox={'facecolor':'0.8', 'pad':5})
plt.show()
```



4.4) Students who took preparation Course

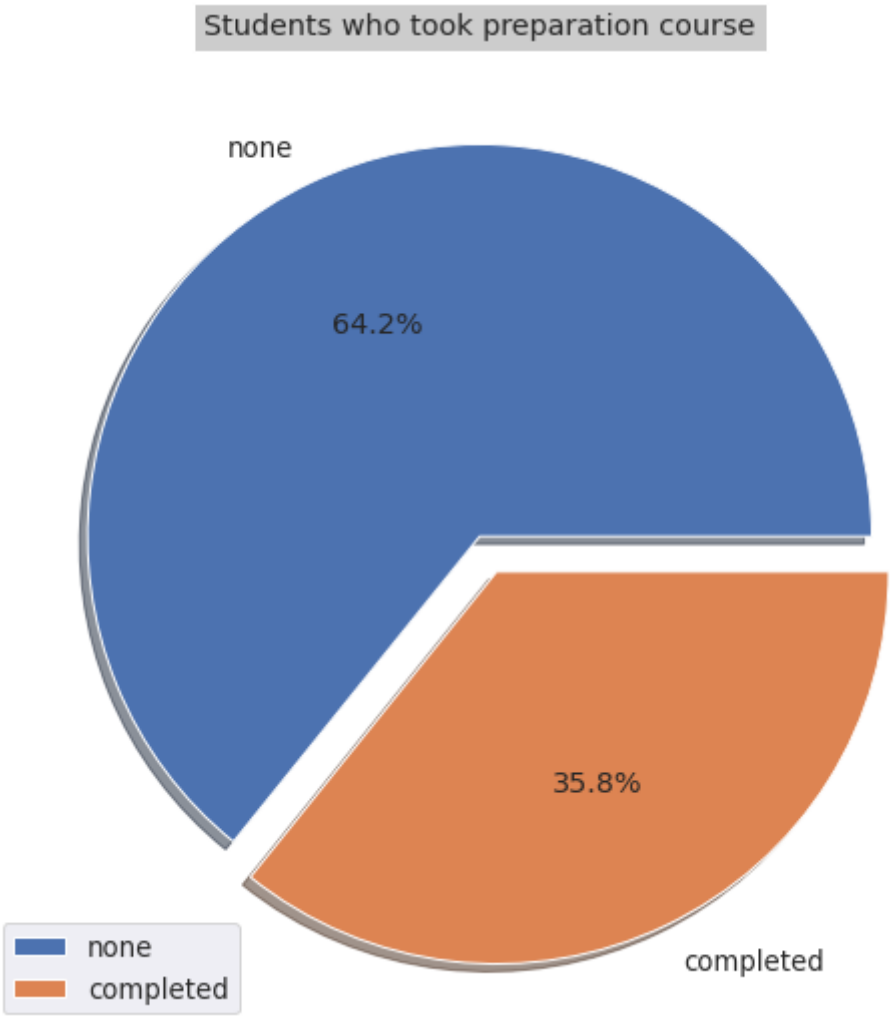
In [11]:

```
plt.bar(df['Course'].value_counts().index, df['Course'].value_counts(), color
        width = 0.4)
plt.xlabel("Course Done or not")
plt.ylabel("No. of students")
plt.title("Students who took preparation course",bbox={'facecolor':'0.8', 'pa
plt.show()
```



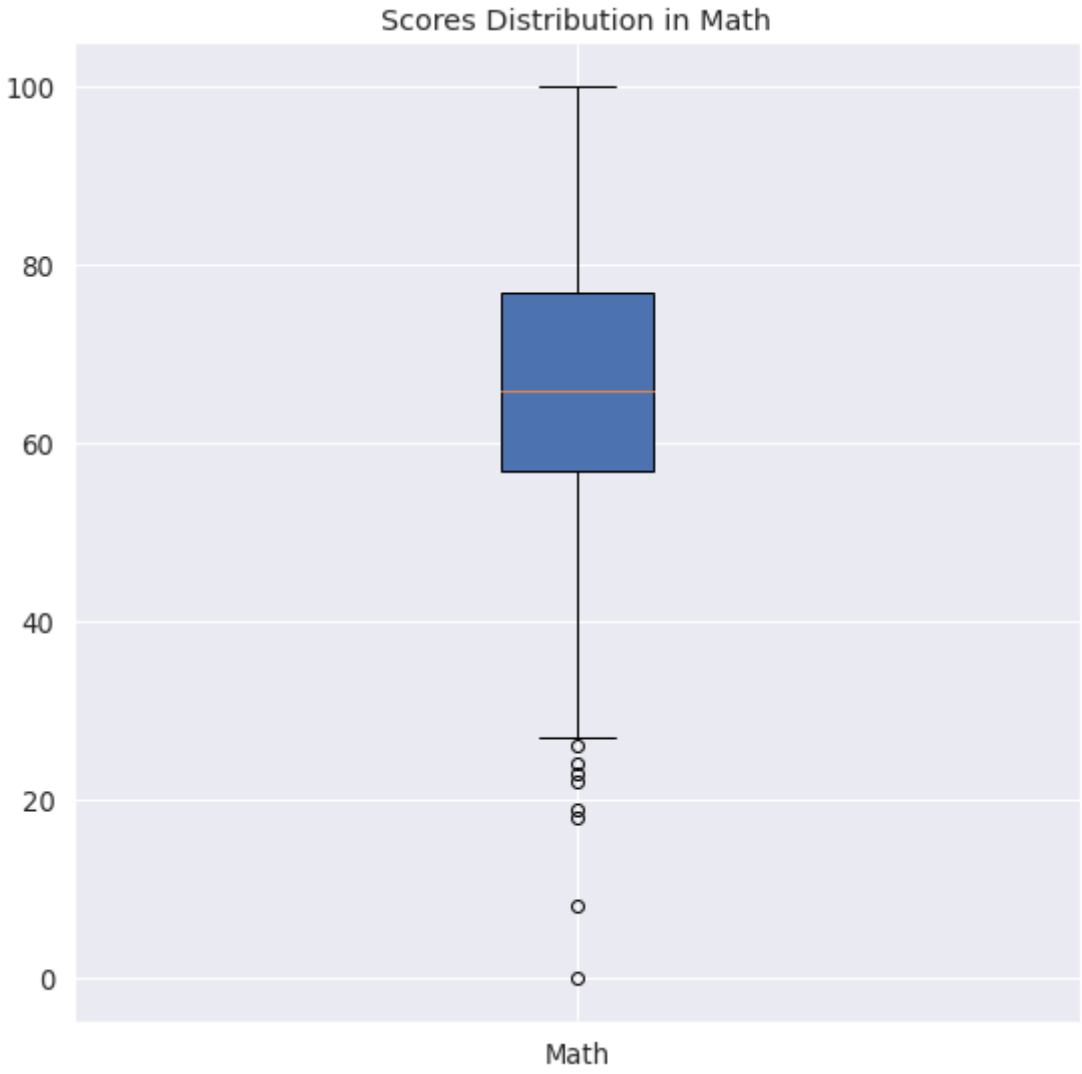
In [12]:

```
plt.pie(df['Course'].value_counts(),labels=df['Course'].value_counts().index,
plt.title("Students who took preparation course",bbox={'facecolor':'0.8', 'pa
plt.legend()
plt.show()
```



4.5) Scores Distribution in Math

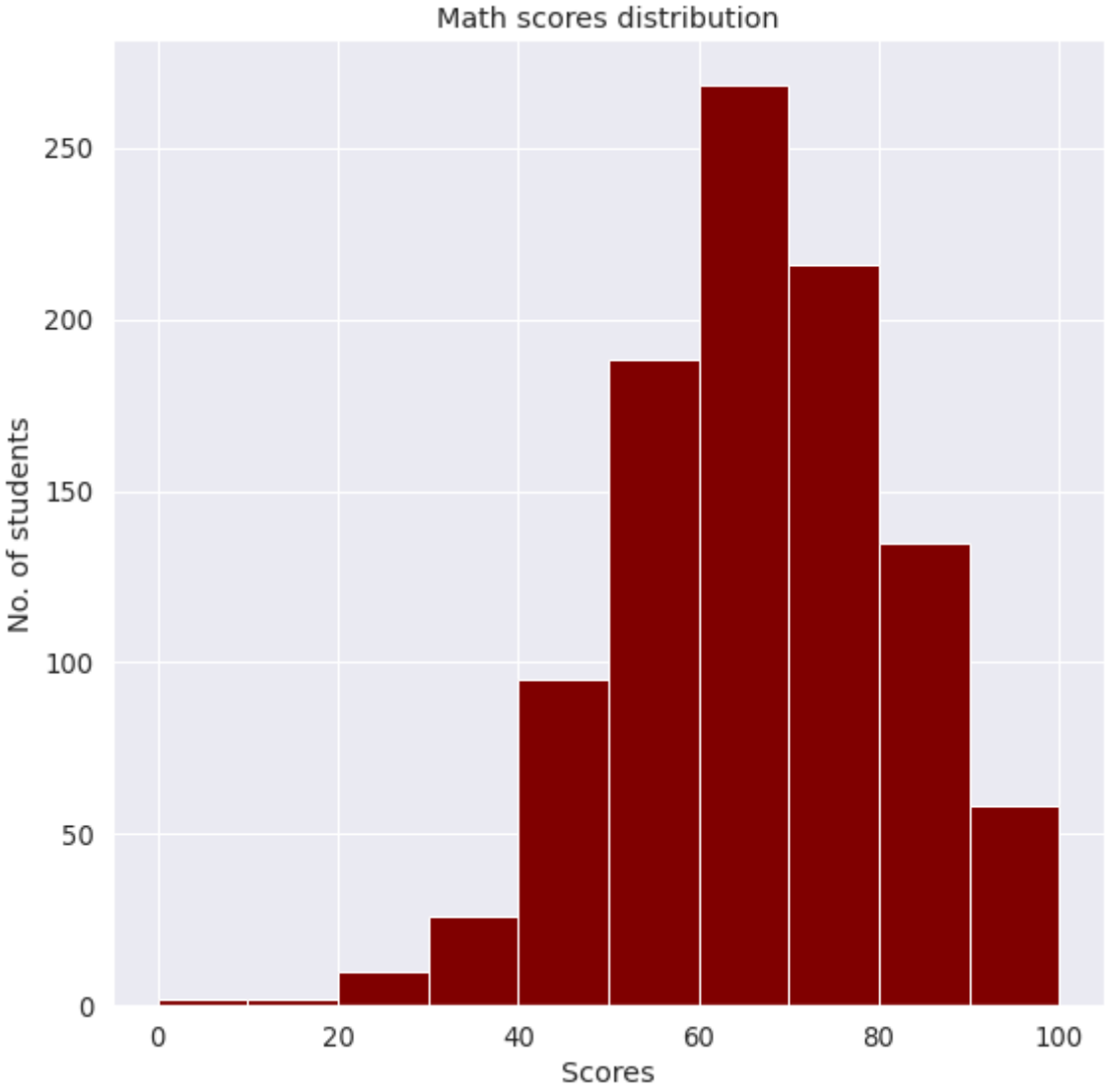
In [13]:
```
plt.title("Scores Distribution in Math")
plt.boxplot(df['Math'],patch_artist=True,labels=['Math'])
plt.show()
```



4.6) scores distribution visualization by histogram

In [14]:

```
plt.hist(df['Math'],color='maroon')
plt.xlabel('Scores')
plt.ylabel('No. of students')
plt.title('Math scores distribution')
plt.show()
```



4.8) Effect of parents education on Total score

In [15]:
```
df1= df.groupby('ParentEducation')['Math'].mean().reset_index()
df1
```

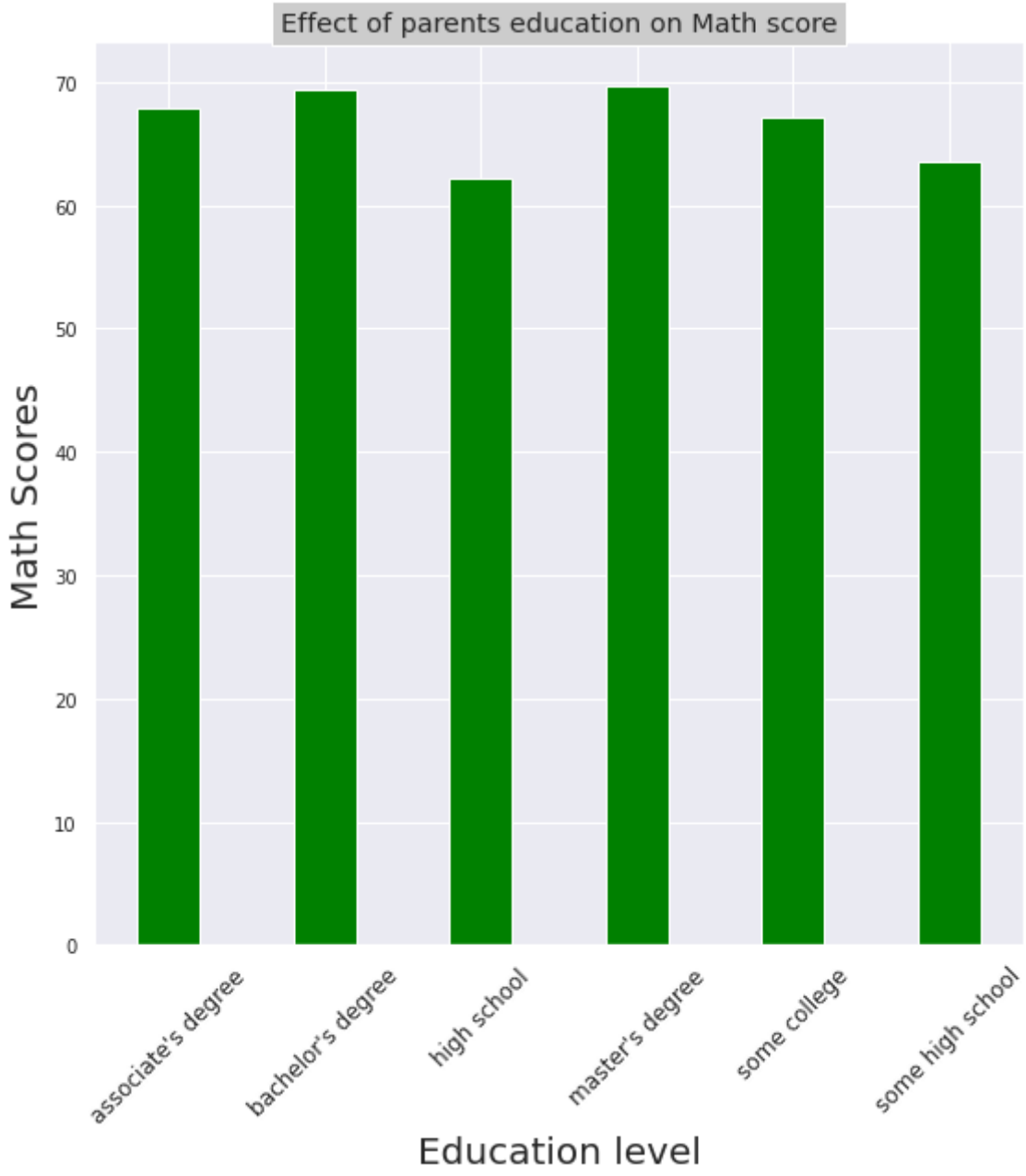
Out[15]:

	ParentEducation	Math
0	associate's degree	67.882883
1	bachelor's degree	69.389831
2	high school	62.137755
3	master's degree	69.745763
4	some college	67.128319
5	some high school	63.497207

Lets see this data by visualization

In [16]:
```
plt.bar(df1['ParentEducation'], df1['Math'], color ='green',
        width = 0.4)
plt.xlabel("Education level",size = 20)
plt.ylabel("Math Scores",size = 20)
plt.xticks(size = 12 ,rotation=45)
plt.yticks(size = 10)
```

```
plt.title("Effect of parents education on Math score",bbox={'facecolor':'0.8'
plt.show()
```



4.9) Did the students who took test preparation got higher grades?

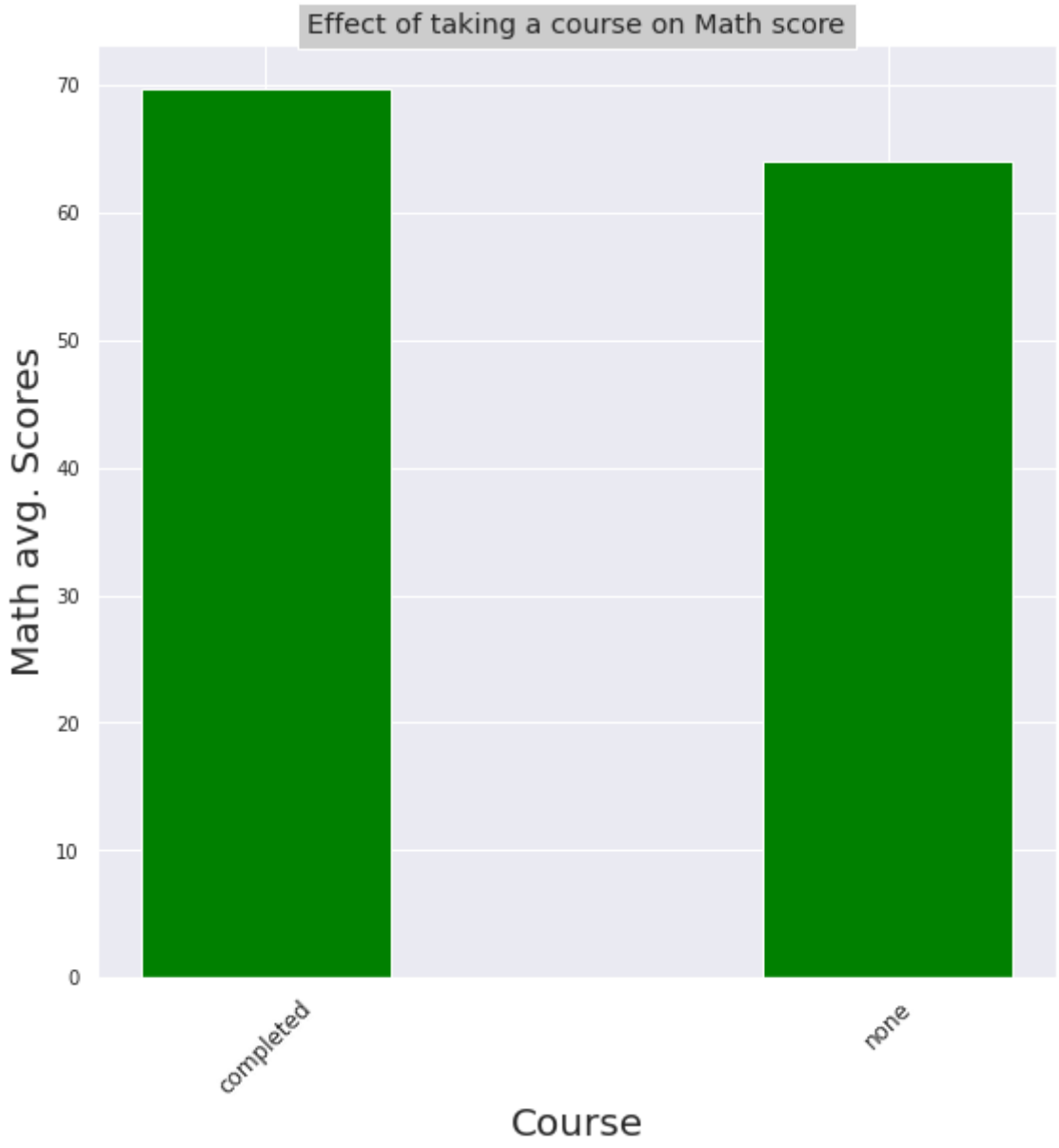
In [17]:
```
df2= df.groupby('Course')['Math'].mean().reset_index()
df2
```

Out[17]:

	Course	Math
0	completed	69.695531
1	none	64.077882

In [18]:
```
plt.bar(df2['Course'], df2['Math'], color ='green',
        width = 0.4)
plt.xlabel("Course",size = 20)
plt.ylabel("Math avg. Scores",size = 20)
plt.xticks(size = 12 ,rotation=45)
plt.yticks(size = 10)
```

```
plt.title("Effect of taking a course on Math score",bbox={'facecolor':'0.8',
plt.show()
```



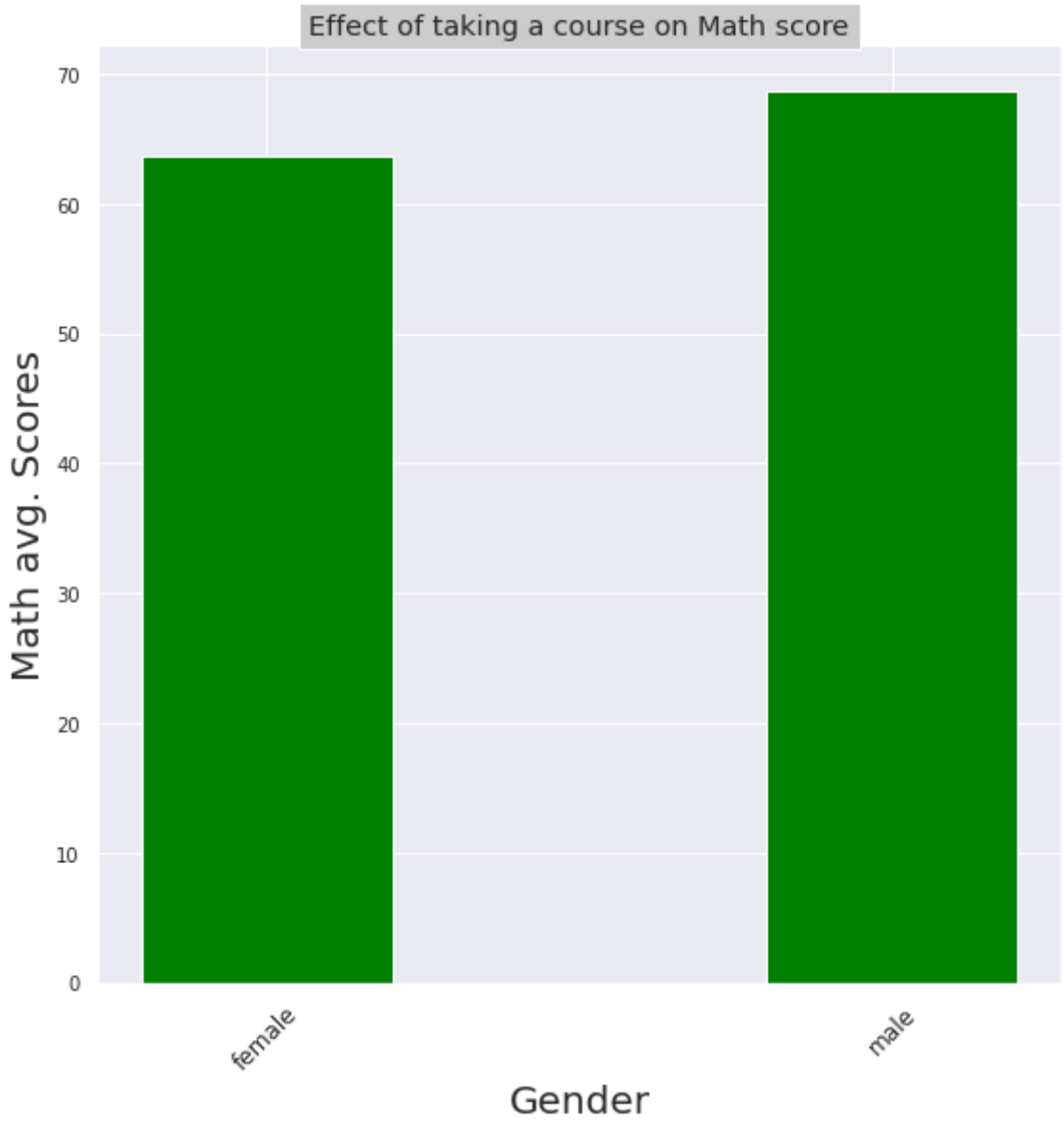
4.10) Who has highest score among both Genders?

In [19]:
```
df3= df.groupby('Gender')['Math'].mean().reset_index()
df3
```

Out[19]:

	Gender	Math
0	female	63.633205
1	male	68.728216

In [20]:
```
plt.bar(df3['Gender'], df3['Math'], color ='green',
        width = 0.4)
plt.xlabel("Gender",size = 20)
plt.ylabel("Math avg. Scores",size = 20)
plt.xticks(size = 12 ,rotation=45)
plt.yticks(size = 10)
plt.title("Effect of taking a course on Math score",bbox={'facecolor':'0.8',
plt.show()
```



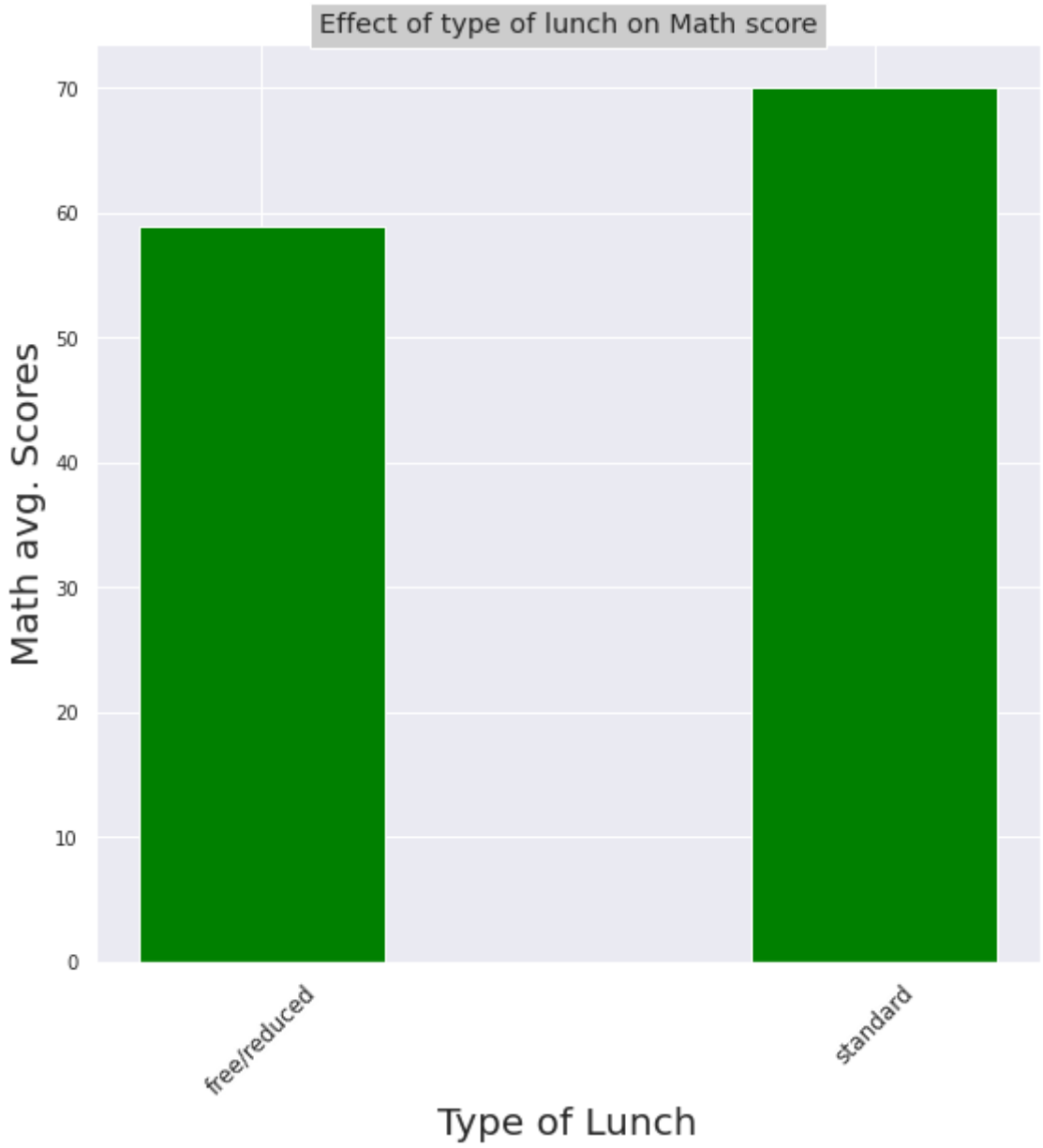
4.11) Lunch and Score

In [21]:
```
df3= df.groupby('Lunch')['Math'].mean().reset_index()
df3
```

Out[21]:

	Lunch	Math
0	free/reduced	58.921127
1	standard	70.034109

In [22]:
```
plt.bar(df3['Lunch'], df3['Math'], color ='green',
        width = 0.4)
plt.xlabel("Type of Lunch",size = 20)
plt.ylabel("Math avg. Scores",size = 20)
plt.xticks(size = 12 ,rotation=45)
plt.yticks(size = 10)
plt.title("Effect of type of lunch on Math score",bbox={'facecolor':'0.8', 'p
plt.show()
```



5) Lets get the data ready for machine learning

In [23]:

```
gender={
    'male':1,
    'female':0
}
df['Gender']=df['Gender'].map(gender)

race={
    'group A':0,
    'group B':1,
    'group C':2,
    'group D':3,
    'group E':4,
}
df['Race']=df['Race'].map(race)

Edu={
    "associate's degree":0,
    "bachelor's degree":1,
    "high school":2,
    "master's degree":3,
    "some college":4,
```

```
    "some high school":5
}
df['ParentEducation']=df['ParentEducation'].map(Edu)

lunch={
    'free/reduced':0,
    'standard':1
}
df['Lunch']=df['Lunch'].map(lunch)

course={
    'none':0,
    'completed':1
}
df['Course']=df['Course'].map(course)

df
```

Out[23]:

	Gender	Race	ParentEducation	Lunch	Course	Math	Reading	Writing
0	0	1	1	1	0	72	72	74
1	0	2	4	1	1	69	90	88
2	0	1	3	1	0	90	95	93
3	1	0	0	0	0	47	57	44
4	1	2	4	1	0	76	78	75
...	...	...	...	...	...	...	...	...
995	0	4	3	1	1	88	99	95
996	1	2	2	0	0	62	55	55
997	0	2	2	0	1	59	71	65
998	0	3	4	1	1	68	78	77
999	0	3	4	0	0	77	86	86

1000 rows × 8 columns

6) Lets split the training and testing data

In [24]:

```
from sklearn.model_selection import train_test_split

X=df.loc[:,['Gender' ,'Race' ,'ParentEducation' ,'Lunch' ,'Course','Reading',
y=df['Math']
X_train, X_test,y_train, y_test = train_test_split(X,y,random_state=42,test_s
print("X_train :\n",X_train)
print("Y_train :\n",y_train)
print("X_test :\n",X_test)
print("Y_test :\n",y_test)
```

```
X_train :
      Gender  Race  ParentEducation  Lunch  Course  Reading  Writing
29         0     3                3      1       0       70       75
535        0     2                1      0       1       83       83
695        0     3                4      0       0       89       86
557        1     2                3      0       0       67       66
836        1     4                2      1       0       64       57
..       ...   ...              ...    ...     ...      ...      ...
106        0     3                3      1       0      100      100
```

```
270       1     2                 1       1       0       63       61
860       0     2                 0       1       0       62       53
435       1     2                 4       0       1       48       53
102       0     3                 0       1       0       91       89

[800 rows x 7 columns]
Y_train :
 29     62
535    66
695    79
557    61
836    73
       ..
106    87
270    69
860    53
435    50
102    85
Name: Math, Length: 800, dtype: int64
X_test :
      Gender  Race  ParentEducation  Lunch  Course  Reading  Writing
521       0     2                0      1       0       86       84
737       0     1                4      0       1       66       73
740       1     3                1      1       0       73       72
660       1     2                4      0       0       77       73
411       1     4                4      1       1       83       78
..      ...   ...              ...    ...     ...      ...      ...
408       0     3                2      0       1       57       56
332       1     4                0      1       1       56       53
208       0     1                4      0       0       81       76
613       0     2                0      1       0       77       74
78        0     3                5      1       1       74       72

[200 rows x 7 columns]
Y_test :
 521    91
737    53
740    80
660    74
411    84
       ..
408    52
332    62
208    74
613    65
78     61
Name: Math, Length: 200, dtype: int64
```

7) Lets apply machine learning algorithms

7.1) Linear Regression

In [25]:

```
linearModel = LinearRegression()
linearModel = linearModel.fit(X_train,y_train)
print(f"Coefficient of determination(Accuracy): {linearModel.score(X_test,y_t
```

```
Coefficient of determination(Accuracy): 88.38026201112224%
```

7.2) Random Forest

In [26]:
```
model = RandomForestRegressor(n_estimators = 64, random_state = 0)
RandomForest = model.fit(X_train, y_train)
y_pred=RandomForest.predict(X_test)
Score = r2_score(y_test, y_pred)
print("\nAccuracy: ", Score*100,"%")
```

```
Accuracy:  84.89196832714583 %
```

7.3) LogisticRegression

In [27]:
```
Logistic = LogisticRegression(solver='lbfgs', max_iter=15000)
Logistic.fit(X_train,y_train)
y_pred=Logistic.predict(X_test)
print('Accuracy : ',r2_score(y_test,y_pred)*100,"%")
```

```
Accuracy :  67.83491631407126 %
/home/sahil/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  n_iter_i = _check_optimize_result(
```

Please note that is above issue is not resolved perfectly anywhere

In [ ]:
```
```