In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv("college11.csv")
data.head()
```

Out[2]:

| | StudentID | Gender | Parent_income | IQ | Encourage | Plan |
|---|---|---|---|---|---|---|
| **0** | 4558 | male | 53900.0 | 118 | encourage | plan |
| **1** | 4561 | female | 24900.0 | 87 | not encourage | not plan |
| **2** | 4563 | female | 65800.0 | 93 | not encourage | not plan |
| **3** | 4565 | male | 11440.0 | 117 | encourage | plan |
| **4** | 4567 | female | 16700.0 | 102 | not encourage | not plan |

In [3]:
```
data.isnull().any()
```

Out[3]:
```
StudentID        False
Gender           False
Parent_income     True
IQ               False
Encourage        False
Plan             False
dtype: bool
```

In [4]:
```
data.describe().T
```

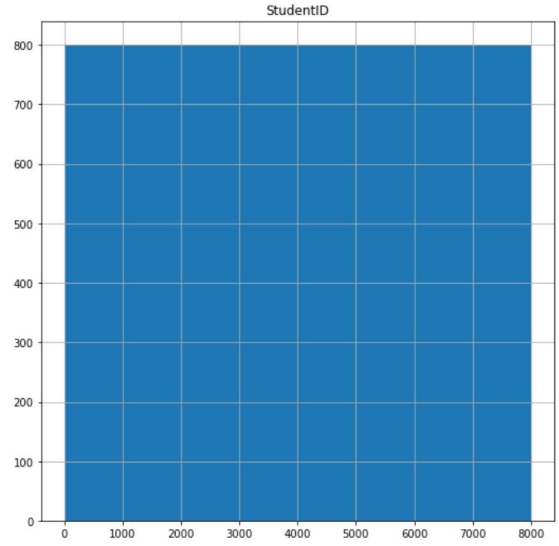
Out[4]:

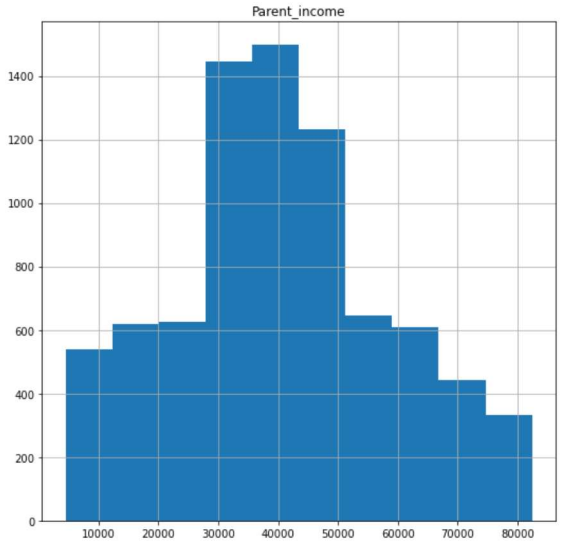
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **StudentID** | 8000.0 | 4000.500000 | 2309.545410 | 1.0 | 2000.75 | 4000.5 | 6000.25 | 8000.0 |
| **Parent_income** | 7995.0 | 40582.577861 | 18024.867192 | 4500.0 | 29400.00 | 39330.0 | 51590.00 | 82390.0 |
| **IQ** | 8000.0 | 99.577750 | 18.923655 | 60.0 | 90.00 | 100.0 | 110.00 | 140.0 |

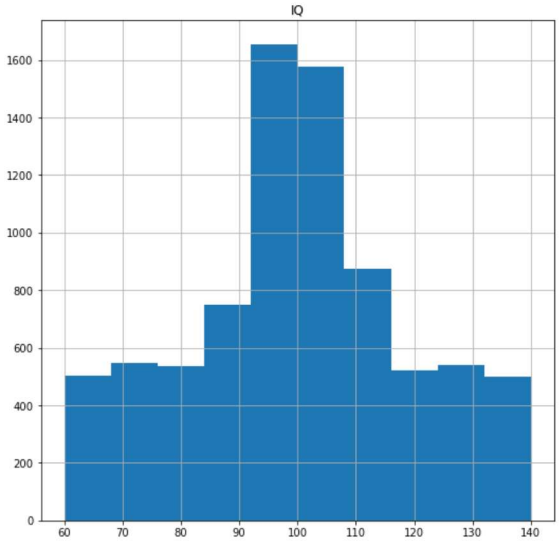
In [5]:
```
data_copy = data.copy(deep = True)
data_copy[['StudentID','Parent_income','IQ']] = data_copy[['StudentID','Parent_ir
data_copy.isnull().sum()
```

Out[5]:
```
StudentID        0
Gender           0
Parent_income    5
IQ               0
Encourage        0
Plan             0
dtype: int64
```

In [6]:
```
p = data.hist(figsize = (20,20))
```

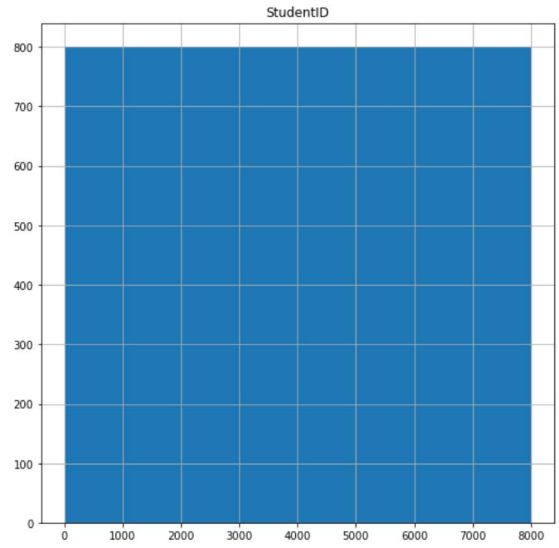


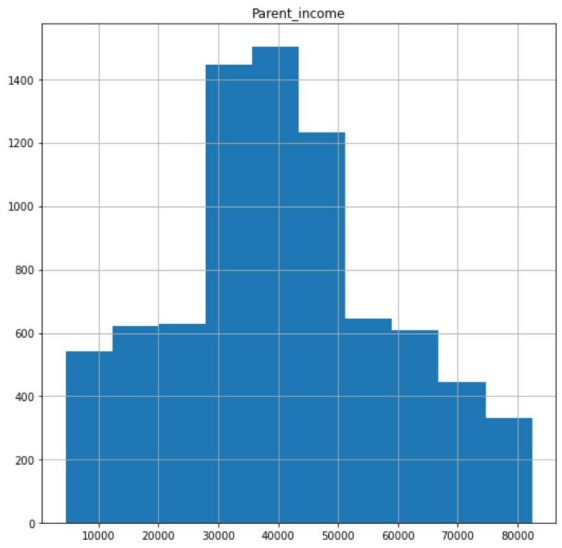



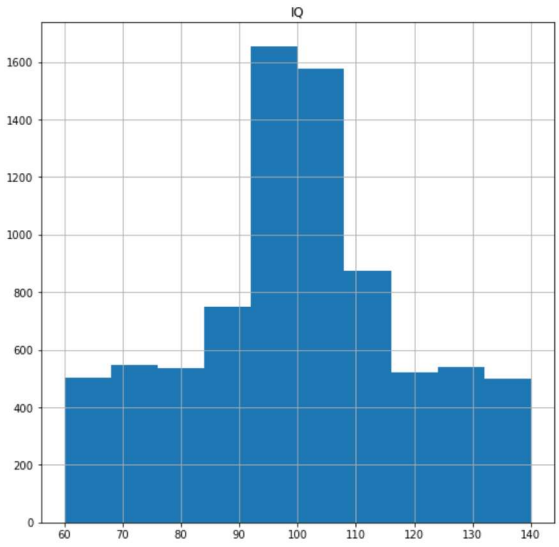




In [7]:
```
data_copy['StudentID'].fillna(data_copy['StudentID'].mean(), inplace = True)
data_copy['Parent_income'].fillna(data_copy['Parent_income'].mean(), inplace = Tr
data_copy['IQ'].fillna(data_copy['IQ'].median(), inplace = True)
```

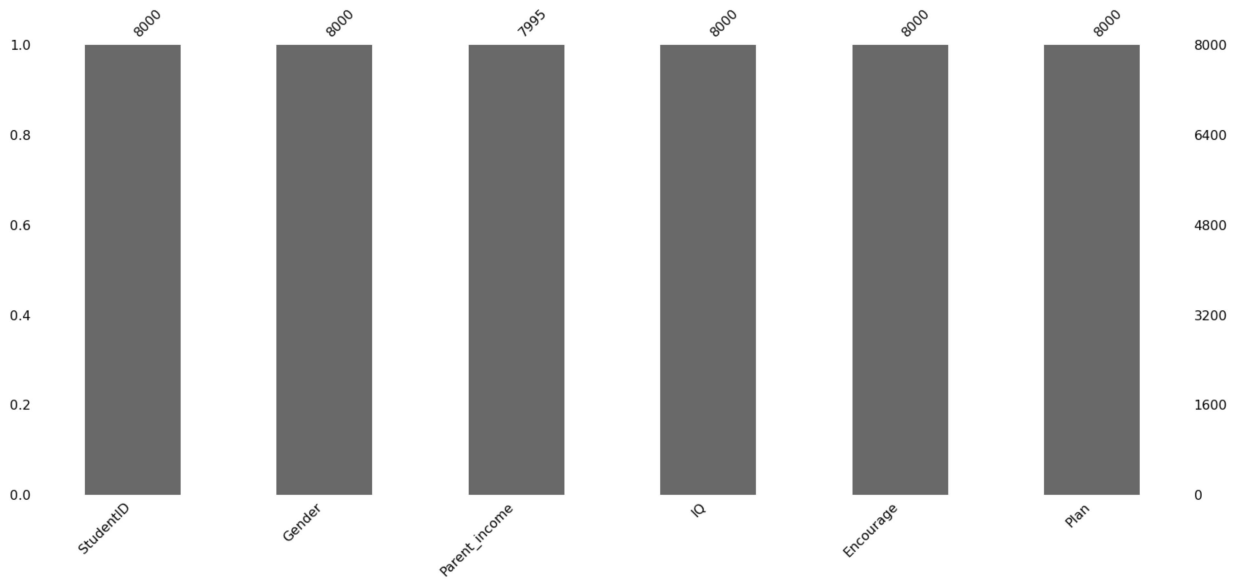
In [8]:
```
p = data_copy.hist(figsize = (20,20))
```



In [9]:
```
import missingno as msno
p = msno.bar(data)
```



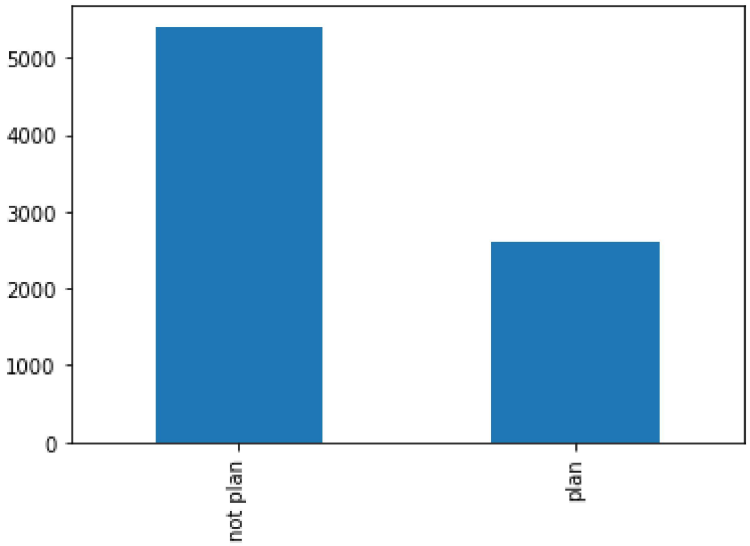

In [10]:
```
dummy = pd.get_dummies(data['Plan'])
```

In [11]:
```
dummy.head()
```

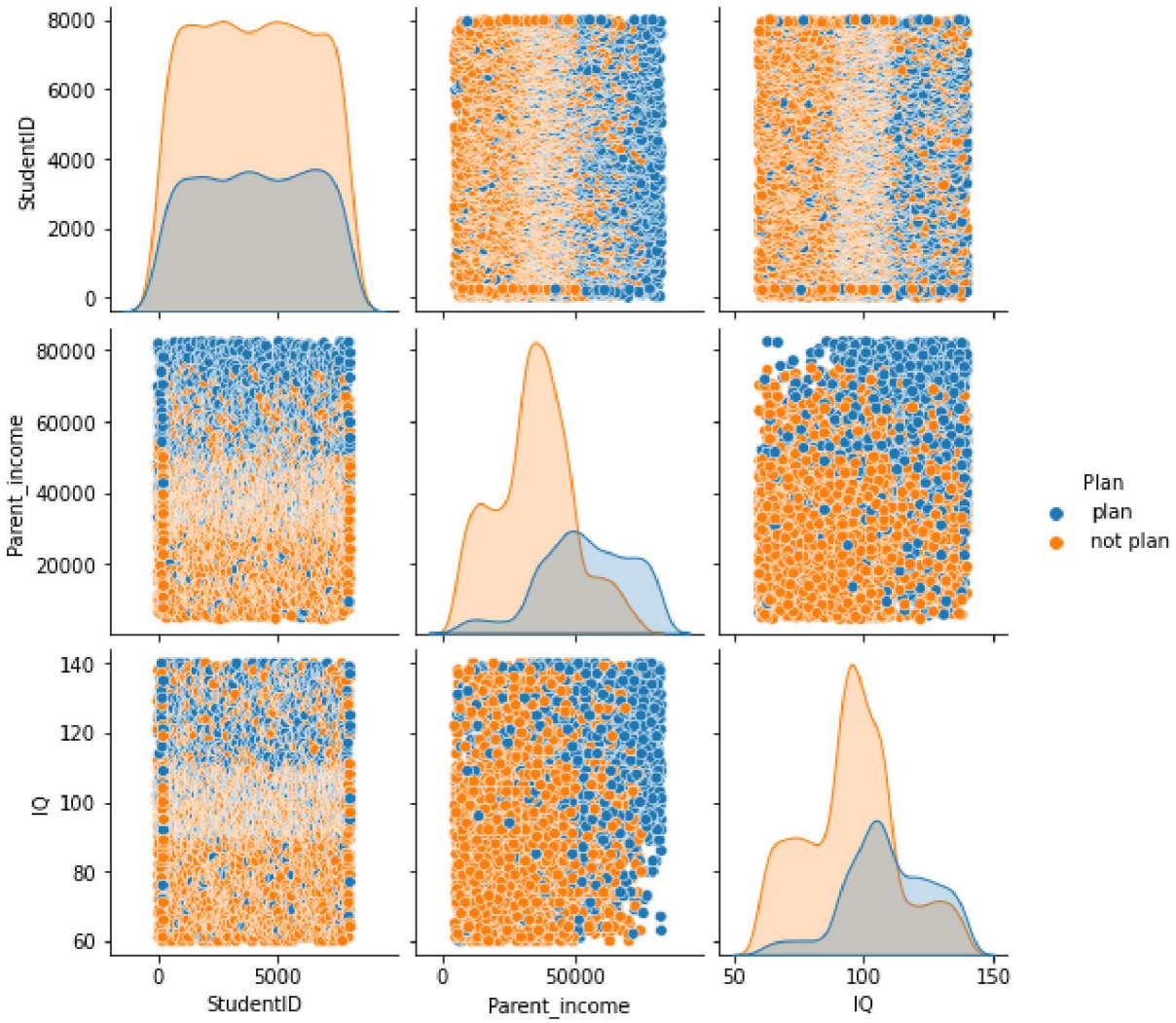
Out[11]:

| | not plan | plan |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 0 |
| **2** | 1 | 0 |
| **3** | 0 | 1 |
| **4** | 1 | 0 |

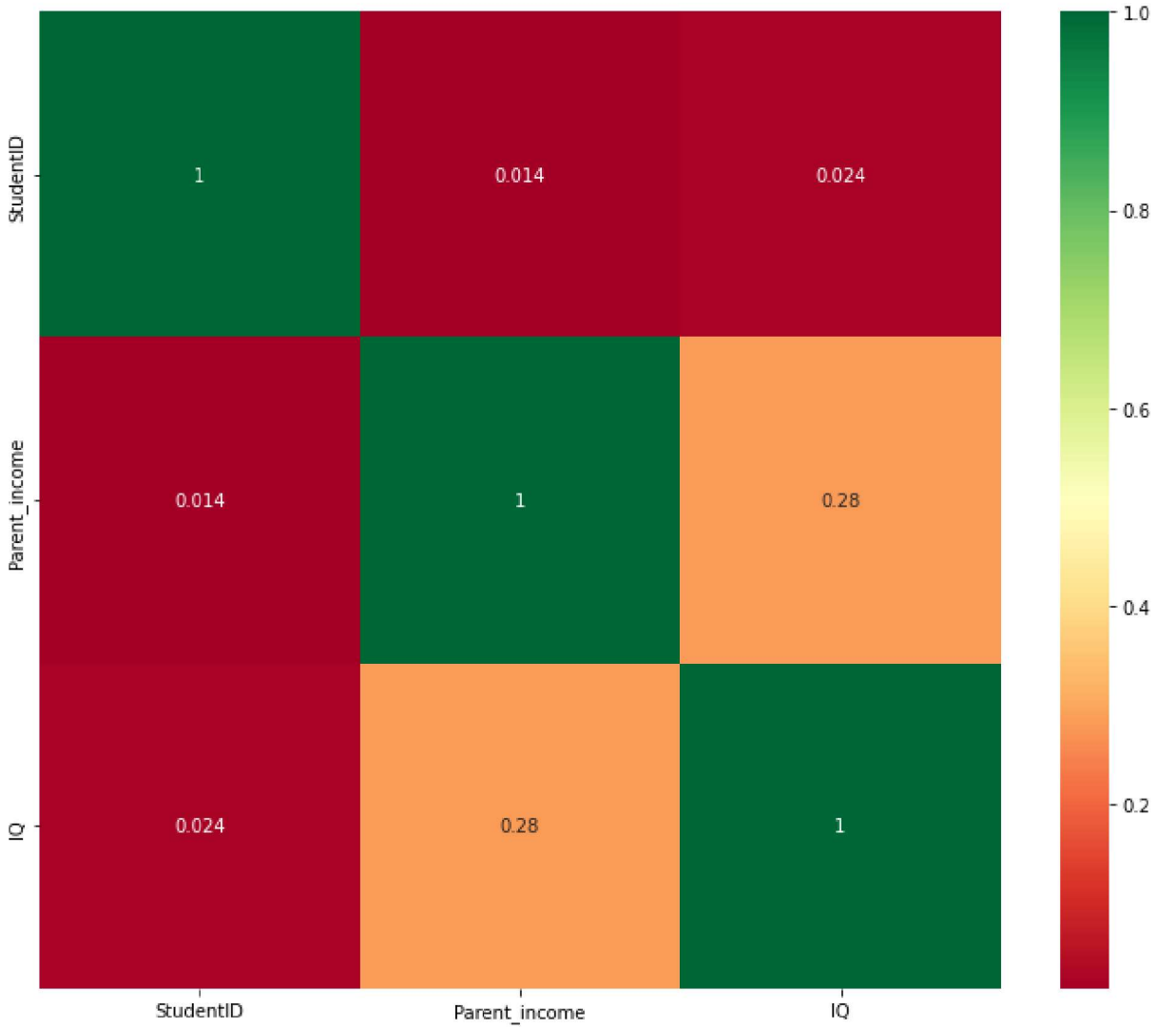
In [12]:
```
p=data.Plan.value_counts().plot(kind="bar")
```



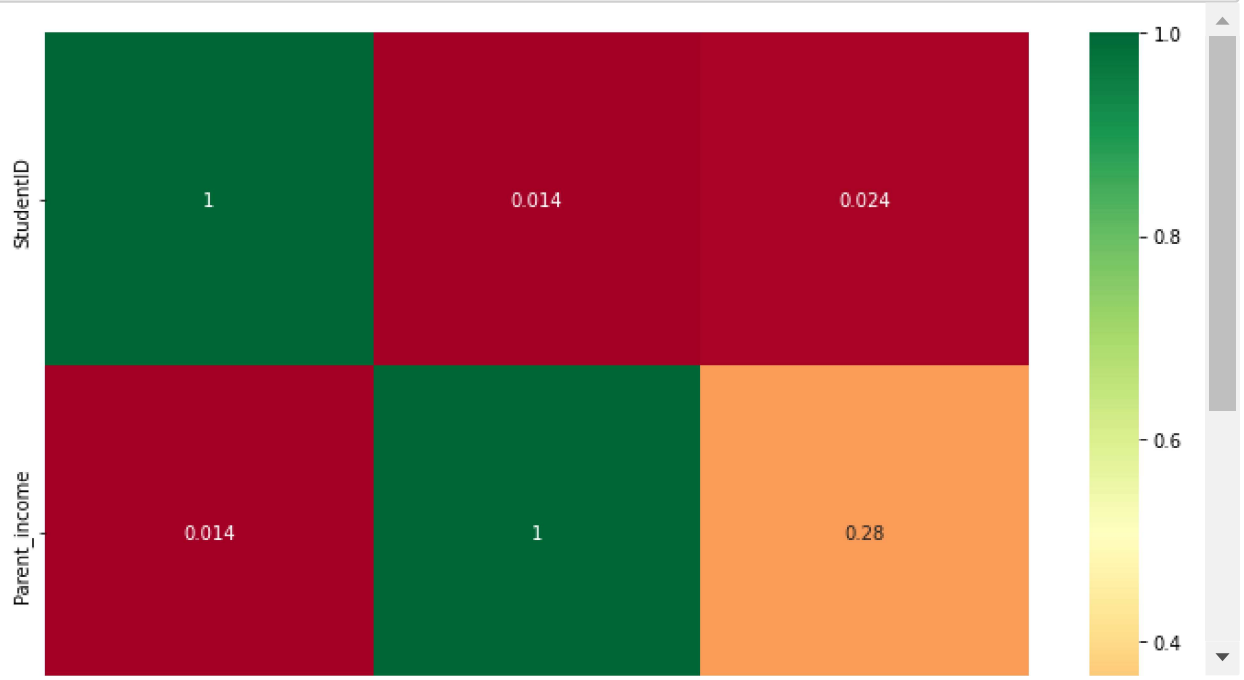
In [13]:
```
import seaborn as sns
p=sns.pairplot(data_copy, hue = 'Plan')
```



In [14]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,10))  # on this line I just set the size of figure to 12 b
p=sns.heatmap(data.corr(), annot=True,cmap ='RdYlGn')
```



In [15]:
```
plt.figure(figsize=(12,10))  # on this line I just set the size of figure to 12 b
p=sns.heatmap(data_copy.corr(), annot=True,cmap ='RdYlGn')
```




In [16]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Gender'] = le.fit_transform(data['Gender'])
data.head()
```

Out[16]:

| | StudentID | Gender | Parent_income | IQ | Encourage | Plan |
|---|---|---|---|---|---|---|
| **0** | 4558 | 1 | 53900.0 | 118 | encourage | plan |
| **1** | 4561 | 0 | 24900.0 | 87 | not encourage | not plan |
| **2** | 4563 | 0 | 65800.0 | 93 | not encourage | not plan |
| **3** | 4565 | 1 | 11440.0 | 117 | encourage | plan |
| **4** | 4567 | 0 | 16700.0 | 102 | not encourage | not plan |

In [17]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Plan'] = le.fit_transform(data['Plan'])
data.head()
```

Out[17]:

| | StudentID | Gender | Parent_income | IQ | Encourage | Plan |
|---|---|---|---|---|---|---|
| **0** | 4558 | 1 | 53900.0 | 118 | encourage | 1 |
| **1** | 4561 | 0 | 24900.0 | 87 | not encourage | 0 |
| **2** | 4563 | 0 | 65800.0 | 93 | not encourage | 0 |
| **3** | 4565 | 1 | 11440.0 | 117 | encourage | 1 |
| **4** | 4567 | 0 | 16700.0 | 102 | not encourage | 0 |

In [18]:
```
from sklearn.preprocessing import StandardScaler
```

In [19]:
```
sc_X = StandardScaler()
data.head()
```

Out[19]:

| | StudentID | Gender | Parent_income | IQ | Encourage | Plan |
|---|---|---|---|---|---|---|
| **0** | 4558 | 1 | 53900.0 | 118 | encourage | 1 |
| **1** | 4561 | 0 | 24900.0 | 87 | not encourage | 0 |
| **2** | 4563 | 0 | 65800.0 | 93 | not encourage | 0 |
| **3** | 4565 | 1 | 11440.0 | 117 | encourage | 1 |
| **4** | 4567 | 0 | 16700.0 | 102 | not encourage | 0 |

In [20]:
```
X = pd.DataFrame(sc_X.fit_transform(data_copy.drop(["Plan"], axis =1),),columns=[
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17200/3237691440.py in <module>
----> 1 X = pd.DataFrame(sc_X.fit_transform(data_copy.drop(["Plan"], axis =1),)
,columns=['StudentID', 'Gender', 'Parent_income', 'IQ', 'Encourage'])

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in fit_transform(sel
f, X, y, **fit_params)
    697         if y is None:
    698             # fit method of arity 1 (unsupervised transformation)
--> 699             return self.fit(X, **fit_params).transform(X)
    700         else:
    701             # fit method of arity 2 (supervised transformation)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\_data.py in fi
t(self, X, y, sample_weight)
    728         # Reset internal state before fitting
    729         self._reset()
--> 730         return self.partial_fit(X, y, sample_weight)
    731
    732     def partial_fit(self, X, y=None, sample_weight=None):

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\_data.py in pa
rtial_fit(self, X, y, sample_weight)
    764         """
    765         first_call = not hasattr(self, "n_samples_seen_")
--> 766         X = self._validate_data(X, accept_sparse=('csr', 'csc'),
    767                                 estimator=self, dtype=FLOAT_DTYPES,
    768                                 force_all_finite='allow-nan', reset=fir
st_call)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_data(se
lf, X, y, reset, validate_separately, **check_params)
    419             out = X
    420         elif isinstance(y, str) and y == 'no_validation':
--> 421             X = check_array(X, **check_params)
    422             out = X
    423         else:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner
_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check
_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all
_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimato
r)
    671                     array = array.astype(dtype, casting="unsafe", copy=
False)
    672                 else:
```

```
--> 673                     array = np.asarray(array, order=order, dtype=dtype)
    674             except ComplexWarning as complex_warning:
    675                 raise ValueError("Complex data not supported\n"

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray(a,
dtype, order, like)
    100         return _asarray_with_like(a, dtype=dtype, order=order, like=lik
e)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104


C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in __array__
(self, dtype)
   1991
   1992     def __array__(self, dtype: NpDtype | None = None) -> np.ndarray:
-> 1993         return np.asarray(self._values, dtype=dtype)
   1994
   1995     def __array_wrap__(

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray(a,
dtype, order, like)
    100         return _asarray_with_like(a, dtype=dtype, order=order, like=lik
e)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104


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

In [ ]:
```
y =data_copy.Plan
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X = pd.DataFrame(sc_X.fit_transform(data_copy.drop(["Outcome"], axis =1),),columr
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, randor
```

In [ ]:
```

```