In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression,Lasso
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
import warnings
warnings.filterwarnings("ignore")
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [3]:
```
participants_data=pd.read_csv(r'C:\Users\18F17884\Desktop\participants_data\p
participants_data.head()
```

Out[3]:

	age	city	weight	height	BMI	gender	qualification	employment_status	marital_sta
0	31.0	Karachi	73.0	165.0	26.8	Male	Bachelors	Self Emplyed	Mar
1	21.0	Turkey	70.0	170.0	24.9	Male	Intermediate	Unemployed	Ot
2	41.0	Faisalabad	72.5	156.0	NaN	Male	Bachelors	Employed	Sir
3	22.0	Rawalpindi	75.0	155.0	32.0	Male	Bachelors	Self Emplyed	Sir
4	26.0	Harbin	75.0	179.0	23.4	Male	Masters	Unemployed	Sir

In [4]:
```
participants_data.shape
```

Out[4]:
```
(90, 12)
```

In [5]:
```
participants_data.isnull().sum()
```

Out[5]:
```
age                 0
city                0
weight              0
height              0
BMI                 4
gender              0
qualification       0
employment_status   0
marital_status      0
copilot_user        0
favourite_dish      0
salary              0
dtype: int64
```

In [6]:
```
participants_data=participants_data.dropna()
```

In [7]:
```
participants_data.isnull().sum()
```

Out[7]:
```
age                 0
city                0
weight              0
height              0
BMI                 0
gender              0
qualification       0
employment_status   0
marital_status      0
copilot_user        0
favourite_dish      0
salary              0
dtype: int64
```

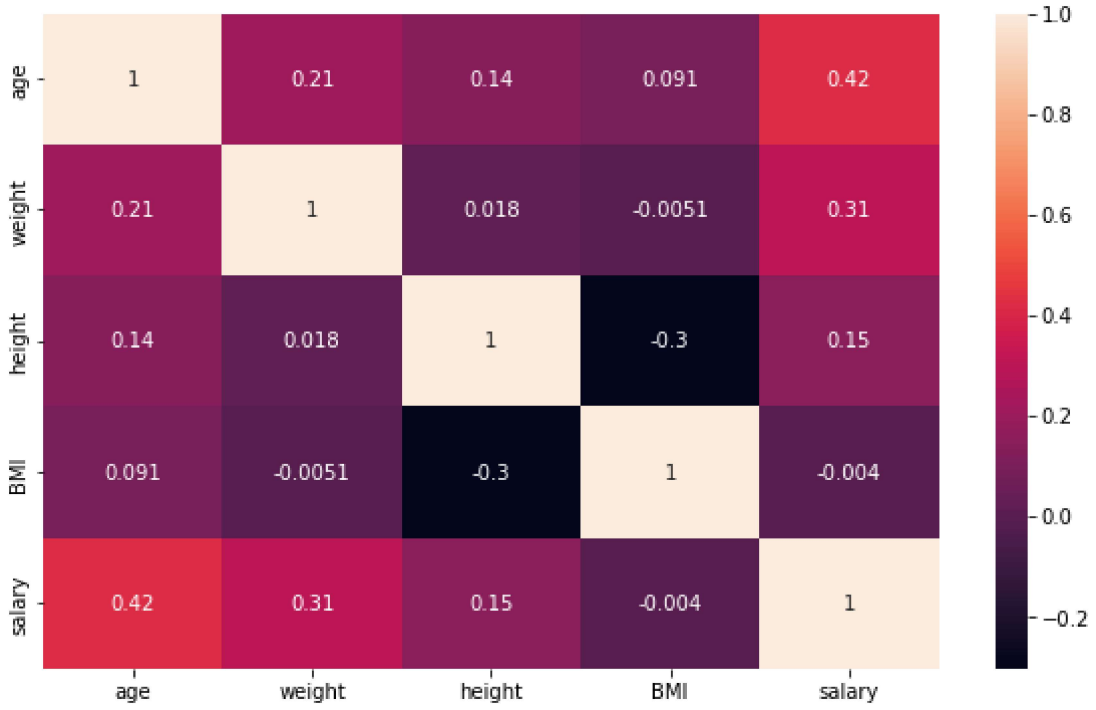
In [8]:
```
participants_data.shape
```

Out[8]:
```
(86, 12)
```

In [9]:
```
participants_data.dtypes
```

Out[9]:
```
age                 float64
city                 object
weight              float64
height              float64
BMI                 float64
gender               object
qualification        object
employment_status    object
marital_status       object
copilot_user         object
favourite_dish       object
salary              float64
dtype: object
```

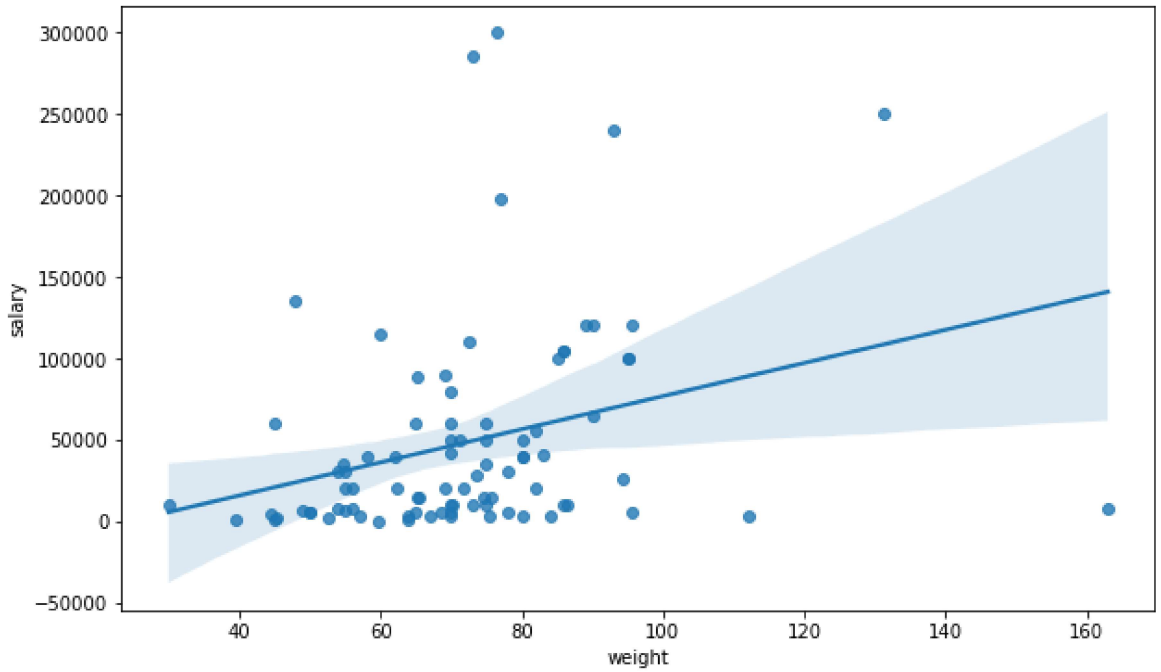
In [10]:
```
plt.figure(figsize=(10,6))
corr = participants_data.corr()
sns.heatmap(corr,annot=True)
plt.show()
```



In [11]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

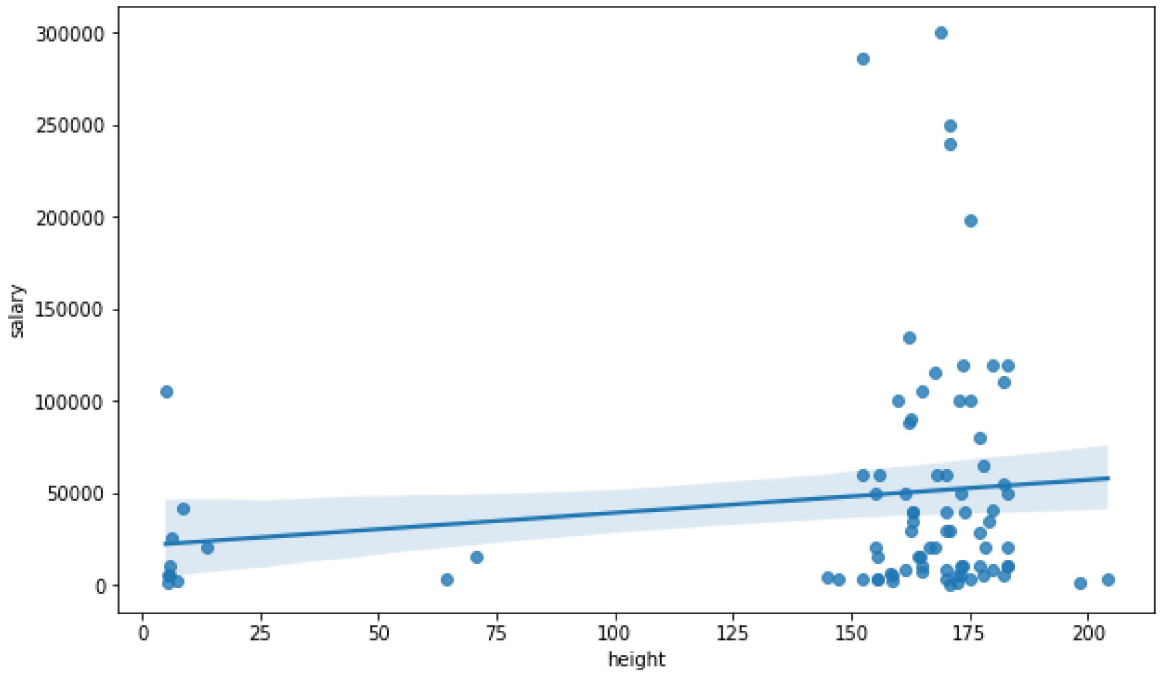
In [12]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="weight", y="salary", data=participants_data)
```

Out[12]: <AxesSubplot:xlabel='weight', ylabel='salary'>



In [13]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="height", y="salary", data=participants_data)
```

Out[13]: <AxesSubplot:xlabel='height', ylabel='salary'>

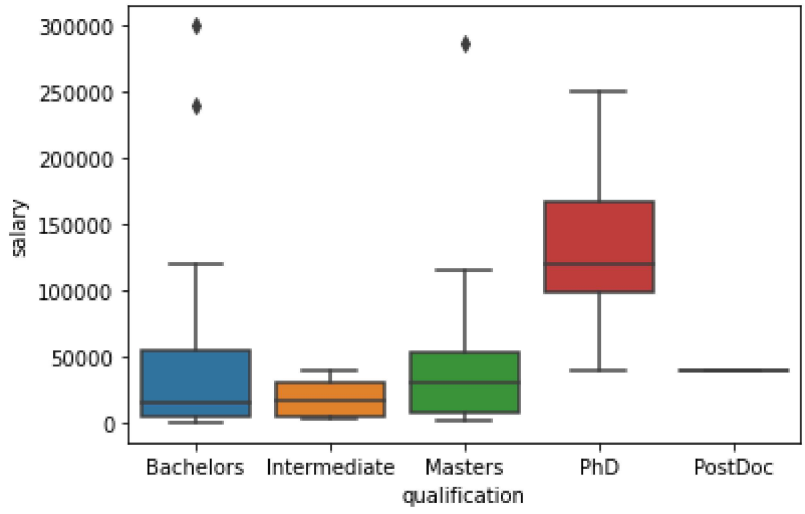


In [14]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(participants_data['weight'], participa
print("The Pearson who prefer indian or pakistani food is", pearson_coef, " w
```

```
The Pearson who prefer indian or pakistani food is 0.30568109669554944  wit
h a P-value of P = 0.004207964445406958
```

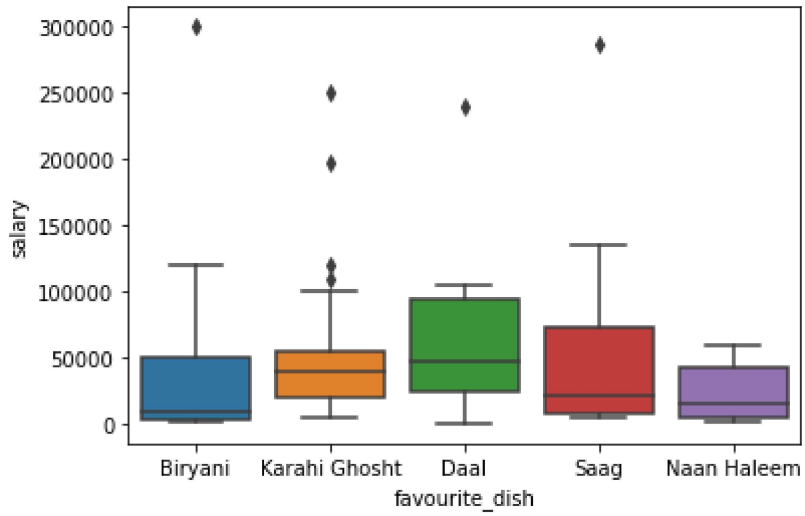
In [15]:
```
sns.boxplot(x="qualification", y="salary", data=participants_data)
```

Out[15]: <AxesSubplot:xlabel='qualification', ylabel='salary'>



In [16]:
```
sns.boxplot(x="favourite_dish", y="salary", data=participants_data)
```

Out[16]: <AxesSubplot:xlabel='favourite_dish', ylabel='salary'>



In [17]:
```
participants_data.drop(['age', 'weight', 'height','favourite_dish'], axis = 1
```

In [18]:
```
participants_data.shape
```

Out[18]: (86, 8)

In [19]:
```
participants_data.describe()
```

Out[19]:

	BMI	salary
count	86.000000	86.000000
mean	173.666315	48097.467674
std	1297.133846	63786.633342
min	0.800000	222.220000
25%	20.625000	5625.000000
50%	24.431500	20000.000000
75%	27.475000	60000.000000
max	12054.400000	300000.000000

In [20]:
```
participants_data.describe(include=['object'])
```

Out[20]:

	city	gender	qualification	employment_status	marital_status	copilot_user
count	86	86	86	86	86	86
unique	44	2	5	3	3	2
top	Karachi	Male	Bachelors	Unemployed	Single	Yes
freq	8	67	47	46	44	47

In [21]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import LabelEncoder
```

In [22]:
```
labelencoder = LabelEncoder()
participants_data.city = labelencoder.fit_transform(participants_data.city)
participants_data.gender = labelencoder.fit_transform(participants_data.gende
participants_data.qualification = labelencoder.fit_transform(participants_dat
participants_data.employment_status = labelencoder.fit_transform(participants
participants_data.marital_status = labelencoder.fit_transform(participants_da
participants_data.copilot_user = labelencoder.fit_transform(participants_data
```

In [23]:
```
participants_data.head(10)
```

Out[23]:

	city	BMI	gender	qualification	employment_status	marital_status	copilot_user	salary
0	20	26.8	1	0	1	0	1	10000.0
1	42	24.9	1	1	2	1	1	3000.0
3	36	32.0	1	0	1	2	1	50000.0
4	13	23.4	1	2	2	2	0	35000.0
6	25	22.7	1	0	2	2	1	20000.0
8	16	23.0	1	0	1	2	1	5000.0
9	20	26.0	1	0	2	0	1	10000.0
10	9	24.5	1	0	2	0	1	20000.0
11	2	27.1	1	0	2	2	1	15000.0
12	24	27.5	1	2	0	0	1	50000.0

In [24]:
```
# Calculate the z-score from with scipy
import scipy.stats as stats
participants_data = stats.zscore(participants_data)
participants_data = stats.zscore(participants_data)
```

In [25]:
```
participants_data
```

Out[25]:

	city	BMI	gender	qualification	employment_status	marital_status	copilot_us
0	-0.120425	-0.113888	0.532524	-0.849909	-0.231608	-1.118212	0.9109
1	1.826966	-0.115361	0.532524	0.030720	0.874965	-0.084166	0.9109
3	1.295859	-0.109855	0.532524	-0.849909	-0.231608	0.949879	0.9109
4	-0.740050	-0.116524	0.532524	0.911348	0.874965	0.949879	-1.0977
6	0.322164	-0.117067	0.532524	-0.849909	0.874965	0.949879	0.9109
...	...	...	...	...	...	...	
85	0.410681	-0.116524	0.532524	-0.849909	0.874965	0.949879	-1.0977
86	1.295859	-0.123581	0.532524	0.911348	-1.338181	-1.118212	-1.0977
87	-0.474497	-0.118696	-1.877849	-0.849909	0.874965	-1.118212	0.9109
88	-0.120425	-0.115778	-1.877849	-0.849909	0.874965	-1.118212	-1.0977
89	-0.297461	-0.108925	-1.877849	-0.849909	-0.231608	-1.118212	-1.0977

86 rows × 8 columns

In [26]:
```
x_train=participants_data.iloc[:,0:11]
y_train=participants_data.iloc[:,7]
```

In [27]:
```
x_train.head()
```

Out[27]:

	city	BMI	gender	qualification	employment_status	marital_status	copilot_use
0	-0.120425	-0.113888	0.532524	-0.849909	-0.231608	-1.118212	0.91092
1	1.826966	-0.115361	0.532524	0.030720	0.874965	-0.084166	0.91092
3	1.295859	-0.109855	0.532524	-0.849909	-0.231608	0.949879	0.91092
4	-0.740050	-0.116524	0.532524	0.911348	0.874965	0.949879	-1.09778
6	0.322164	-0.117067	0.532524	-0.849909	0.874965	0.949879	0.91092

In [28]:
```
y_train.head()
```

Out[28]:
```
0   -0.600767
1   -0.711152
3    0.030001
4   -0.206537
6   -0.443075
Name: salary, dtype: float64
```

In [29]:
```
from sklearn.model_selection import train_test_split
# splitting the data
x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_si
```

In [30]:
```
#print the shape of train and test data after splting
print (x_train.shape)
print (x_test.shape)
```

```
(68, 8)
(18, 8)
```

In [31]:
```
mlr = LinearRegression()
model_mlr = mlr.fit(x_train,y_train)
```

In [32]:
```
y_pred1 = model_mlr.predict(x_test)
```

In [47]:
```
MSE1 = mean_squared_error(y_test,y_pred1)
print('MSE is ', MSE1)
```

```
MSE is  1.483271664811708e-30
```

In [48]:
```
rf = RandomForestRegressor()
modelrf=rf.fit(x_train,y_train)
```

In [49]:
```
y_pred2 = modelrf.predict(x_test)
```

In [50]:
```
MSE2 = mean_squared_error(y_test,y_pred2)
print('MSE is ', MSE2)
```

```
MSE is  0.0010482534648488732
```

In [51]:
```
LassoModel1=Lasso()
lm=LassoModel1.fit(x_train,y_train)
```

In [52]:
```
y_pred3 = modelrf.predict(x_test)
```

In [53]:
```
MSE3 = mean_squared_error(y_test,y_pred2)
print('MSE is ', MSE3)
```

```
MSE is  0.0010482534648488732
```

In [54]:
```
scores = [('MLR', MSE1), ('Random Forest', MSE2), ('Lasso', MSE3)]
```

In [55]:
```
MSE = pd.DataFrame(data = scores, columns=['Model', 'MSE Score'])
MSE
```

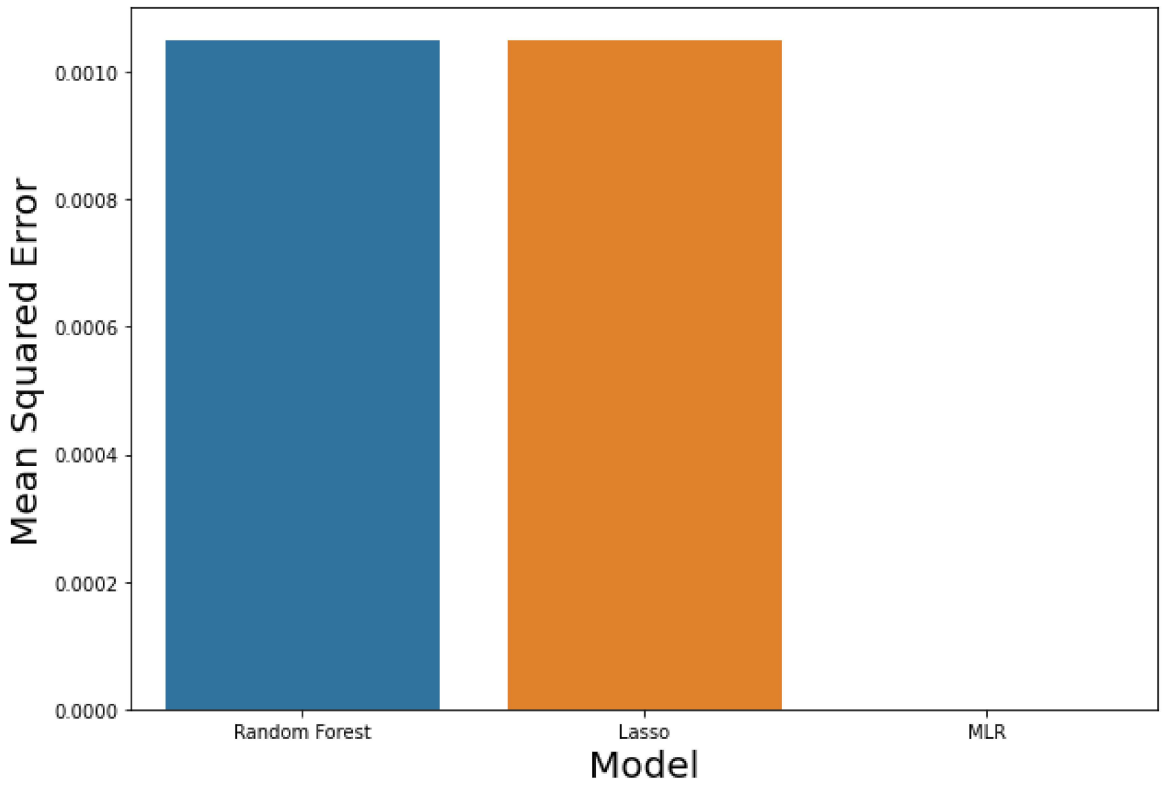
Out[55]:

	Model	MSE Score
0	MLR	1.483272e-30
1	Random Forest	1.048253e-03
2	Lasso	1.048253e-03

In [59]:
```
MSE.sort_values(by=(['MSE Score']), ascending=False, inplace=True)

f, axe = plt.subplots(1,1, figsize=(10,7))
sns.barplot(x = MSE['Model'], y=MSE['MSE Score'], ax = axe)
axe.set_xlabel('Model', size=20)
axe.set_ylabel('Mean Squared Error', size=20)

plt.show()
```



In [ ]:
```

```