The data in "Weddings.xlsx" (see the attachment) show wedding costs and different variables that are likely to affect it (attendance, the couple's income, age of the bride, etc.). Build a regression model to predict wedding costs from all other variables. Note the "Payor" column contains a categorical variable, which takes two values - "Bride's family" and "Groom's family". To use the variable in the regression model, you need to first convert it to a numerical variable, with 0 representing one category and 1 representing the other. Analyze residuals to confirm that the assumptions of linear regression are valid for this model.

Interpret and comment on the regression results: (1) What is the quality of the model? (2) Which predictor variables are the most important?

If a couple is planning a wedding for 175 guests, the couple's income is $100,000, the bride's age is 33, the payor is the Bride's family, how much should they budget?

In [149]:
```
import pandas as pd #importing pandas package
import numpy as np ##importing numpy package for mathematical calculation
import matplotlib.pyplot as plt #importing pyplot module of matplotlib package
from sklearn import linear_model #Importing linar_model module from sklearn library which contains a lot of efficient tools for machine learning and statistical modeling
from statsmodels.api import OLS #import OLS from statsmodels.api
import statsmodels.api as sm #import Ostatsmodels.api
from statsmodels.stats.outliers_influence import variance_inflation_factor #import variance_inflation_factor from statsmodels.stats.outliers_influence
from seaborn import pairplot #importing pairplot module from seaborn library
from statsmodels.graphics.gofplots import qqplot #importing qqplot from statsmodels.graphics.gofplots

df = pd.read_excel("Weddings.xlsx") #reading data from excel file
df #output data to the screen
```

Out[149]:

	CoupleIncome	BrideAge	Payor	WeddingCost	Attendance
0	130000	22	Bride's Parents	60700	300
1	157000	23	Bride's Parents	52000	350
2	98000	27	Groom's Parents	47000	150
3	72000	29	Bride's Parents	42000	200
4	86000	25	Bride's Parents	34000	250
5	90000	28	Bride's Parents	30500	150
6	43000	19	Groom's Parents	30000	250
7	100000	30	Bride's Parents	30000	300
8	65000	24	Bride's Parents	28000	250
9	78000	35	Bride's Parents	26000	200
10	73000	25	Bride's Parents	25000	150
11	75000	27	Groom's Parents	24000	200
12	64000	25	Bride's Parents	24000	200
13	67000	27	Groom's Parents	22000	200
14	75000	25	Bride's Parents	20000	200
15	67000	30	Bride's Parents	20000	200
16	62000	21	Groom's Parents	20000	100
17	75000	19	Bride's Parents	19000	150
18	52000	23	Bride's Parents	19000	200
19	64000	22	Bride's Parents	18000	150
20	55000	28	Bride's Parents	16000	100
21	53000	31	Groom's Parents	14000	100
22	62000	24	Bride's Parents	13000	150
23	40000	26	Bride's Parents	7000	50
24	45000	32	Groom's Parents	5000	50

In [150]:
```
#Peforming data preparation as this is important step which needs to be performed before implementing linear regresssion model

#df.dtypes - to check the data types of the columns

df['Payor'] =df['Payor'].astype('category').cat.codes #first converts Payor column into categorical type and then 'cat.codes' converts categorical value to numerical in pandas dataframe
#Bride's Parents is assigned value 0 and Groom's Parents is assigned value 1

df  #data output to screen after the Payor column is changed to numerical value
```

Out[150]:

	CoupleIncome	BrideAge	Payor	WeddingCost	Attendance
0	130000	22	0	60700	300
1	157000	23	0	52000	350
2	98000	27	1	47000	150
3	72000	29	0	42000	200
4	86000	25	0	34000	250
5	90000	28	0	30500	150
6	43000	19	1	30000	250
7	100000	30	0	30000	300
8	65000	24	0	28000	250
9	78000	35	0	26000	200
10	73000	25	0	25000	150
11	75000	27	1	24000	200
12	64000	25	0	24000	200
13	67000	27	1	22000	200
14	75000	25	0	20000	200
15	67000	30	0	20000	200
16	62000	21	1	20000	100
17	75000	19	0	19000	150
18	52000	23	0	19000	200
19	64000	22	0	18000	150
20	55000	28	0	16000	100
21	53000	31	1	14000	100
22	62000	24	0	13000	150
23	40000	26	0	7000	50
24	45000	32	1	5000	50

In [151]:
```
model = sm.OLS.from_formula(
    'WeddingCost ~ CoupleIncome + BrideAge + Payor + Attendance', data=df).fit() #implementing multiple linear regression with Wedding cost as dependent variable and the other variables as predictors or independent variables
                                                                                 #fit is used to train the model


model.summary()  #summary used to get the summary of the model designed
```

Out[151]:

OLS Regression Results

Dep. Variable:	WeddingCost	**R-squared:**	0.754
Model:	OLS	**Adj. R-squared:**	0.705
Method:	Least Squares	**F-statistic:**	15.31
Date:	Sun, 08 Nov 2020	**Prob (F-statistic):**	6.97e-06
Time:	22:59:50	**Log-Likelihood:**	-254.71
No. Observations:	25	**AIC:**	519.4
Df Residuals:	20	**BIC:**	525.5
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3048.3092	1.16e+04	-0.263	0.795	-2.72e+04	2.11e+04
CoupleIncome	0.3238	0.079	4.112	0.001	0.160	0.488
BrideAge	-233.3111	380.541	-0.613	0.547	-1027.105	560.483
Payor	3712.8716	3359.726	1.105	0.282	-3295.394	1.07e+04
Attendance	54.1315	28.591	1.893	0.073	-5.507	113.770

Omnibus:	6.300	**Durbin-Watson:**	1.828
Prob(Omnibus):	0.043	**Jarque-Bera (JB):**	4.383
Skew:	0.965	**Prob(JB):**	0.112
Kurtosis:	3.697	**Cond. No.**	6.31e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.31e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Solution 1- Quality of the model

Quality of the model. The $R2$(0.754) and the adjusted $R2$(0.705) are near to 1 which indicates that some of the independent variables has helped to improve the model.

From the above result, I could clearly see that p-values of BrideAge is the highest(with pvalue 0.547) as compared to other predictors, so I would be removing the BrideAge variable and running the linear model again Also, In the table, I can see that the intercept is not significant (p=0.795, that is much greater than the usual significance level of 0.05), and so can be ignored when making predictions.

In [152]:
```
model = sm.OLS.from_formula(
    'WeddingCost ~ CoupleIncome + Payor + Attendance - BrideAge', data=df).fit() #running the linear model again excluding BrideAge variable
model.summary() #summary used to get the summary of the model designed
```

Out[152]:

OLS Regression Results

Dep. Variable:	WeddingCost	**R-squared:**	0.749
Model:	OLS	**Adj. R-squared:**	0.713
Method:	Least Squares	**F-statistic:**	20.92
Date:	Sun, 08 Nov 2020	**Prob (F-statistic):**	1.64e-06
Time:	22:59:52	**Log-Likelihood:**	-254.95
No. Observations:	25	**AIC:**	517.9
Df Residuals:	21	**BIC:**	522.8
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-9442.3236	4933.253	-1.914	0.069	-1.97e+04	816.938
CoupleIncome	0.3192	0.077	4.134	0.000	0.159	0.480
Payor	3693.2468	3309.275	1.116	0.277	-3188.768	1.06e+04
Attendance	57.9211	27.496	2.106	0.047	0.739	115.103

Omnibus:	4.784	**Durbin-Watson:**	1.903
Prob(Omnibus):	0.091	**Jarque-Bera (JB):**	3.252
Skew:	0.864	**Prob(JB):**	0.197
Kurtosis:	3.368	**Cond. No.**	2.90e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.9e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

From the above result, I could clearly see that p-values of Payor is the highest(with pvalue 0.277) as compared to other predictors, so I would be removing the Payor variable and and run the linear model for third time

In [153]:
```
model = sm.OLS.from_formula(
    'WeddingCost ~ CoupleIncome + Attendance - BrideAge -Payor', data=df).fit() ##running the linear model again excluding BrideAge and Payor variable
model.summary()
```

Out[153]:

OLS Regression Results

Dep. Variable:	WeddingCost	**R-squared:**	0.734
Model:	OLS	**Adj. R-squared:**	0.710
Method:	Least Squares	**F-statistic:**	30.41
Date:	Sun, 08 Nov 2020	**Prob (F-statistic):**	4.65e-07
Time:	22:59:52	**Log-Likelihood:**	-255.67
No. Observations:	25	**AIC:**	517.3
Df Residuals:	22	**BIC:**	521.0
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-7002.4471	4446.990	-1.575	0.130	-1.62e+04	2220.046
CoupleIncome	0.3124	0.077	4.036	0.001	0.152	0.473
Attendance	53.0309	27.296	1.943	0.065	-3.578	109.640

Omnibus:	5.864	**Durbin-Watson:**	1.926
Prob(Omnibus):	0.053	**Jarque-Bera (JB):**	4.228
Skew:	0.985	**Prob(JB):**	0.121
Kurtosis:	3.424	**Cond. No.**	2.44e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.44e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

solution 2 - Significant predictor variables

From the above result, I could clearly see that CoupleIncome is the highest significant variable followed by Attendance which is the second highest significant variable

In [154]:
```
dataset_independent_var= df[['CoupleIncome','BrideAge','Payor','Attendance']] #assigning  the predictor variable columns to dataset_independent_var

reg= linear_model.LinearRegression() #using LinearRegression class object
reg.fit(df[['CoupleIncome','BrideAge','Payor','Attendance']],df.WeddingCost) #fit used to train the model
#training set is columns 'CoupleIncome','BrideAge','Payor','Attendance' and the target variable is WeddingCost
```

Out[154]:
```
LinearRegression()
```

In [155]:
```
reg.intercept_  #value of intercept
```

Out[155]:
```
-3048.3091964161504
```

In [156]:
```
reg.coef_     #value of coefficient
```

Out[156]:
```
array([ 3.23790503e-01, -2.33311075e+02,  3.71287162e+03,  5.41315315e+01])
```

#solution 3

Wedding cost when couple's income is $100,000, the bride's age is 33, the payor is the Bride's family and guests are 175 in number is:

In [157]:
```
round(reg.predict([[100000,33,0,175]])[0],2)
```

Out[157]:
```
31104.49
```

Analyzing residuals to confirm that the assumptions of linear regression are valid for this model.

In [158]:
```
df1= df.corr() #using corr to find correlation between variables
print(df1)
```

```
              CoupleIncome  BrideAge     Payor  WeddingCost  Attendance
CoupleIncome      1.000000 -0.087019 -0.257266     0.829937    0.699811
BrideAge         -0.087019  1.000000  0.065011    -0.184026   -0.217648
Payor            -0.257266  0.065011  1.000000    -0.130047   -0.290040
WeddingCost       0.829937 -0.184026 -0.130047     1.000000    0.733293
Attendance        0.699811 -0.217648 -0.290040     0.733293    1.000000
```

Multicollinearity Test

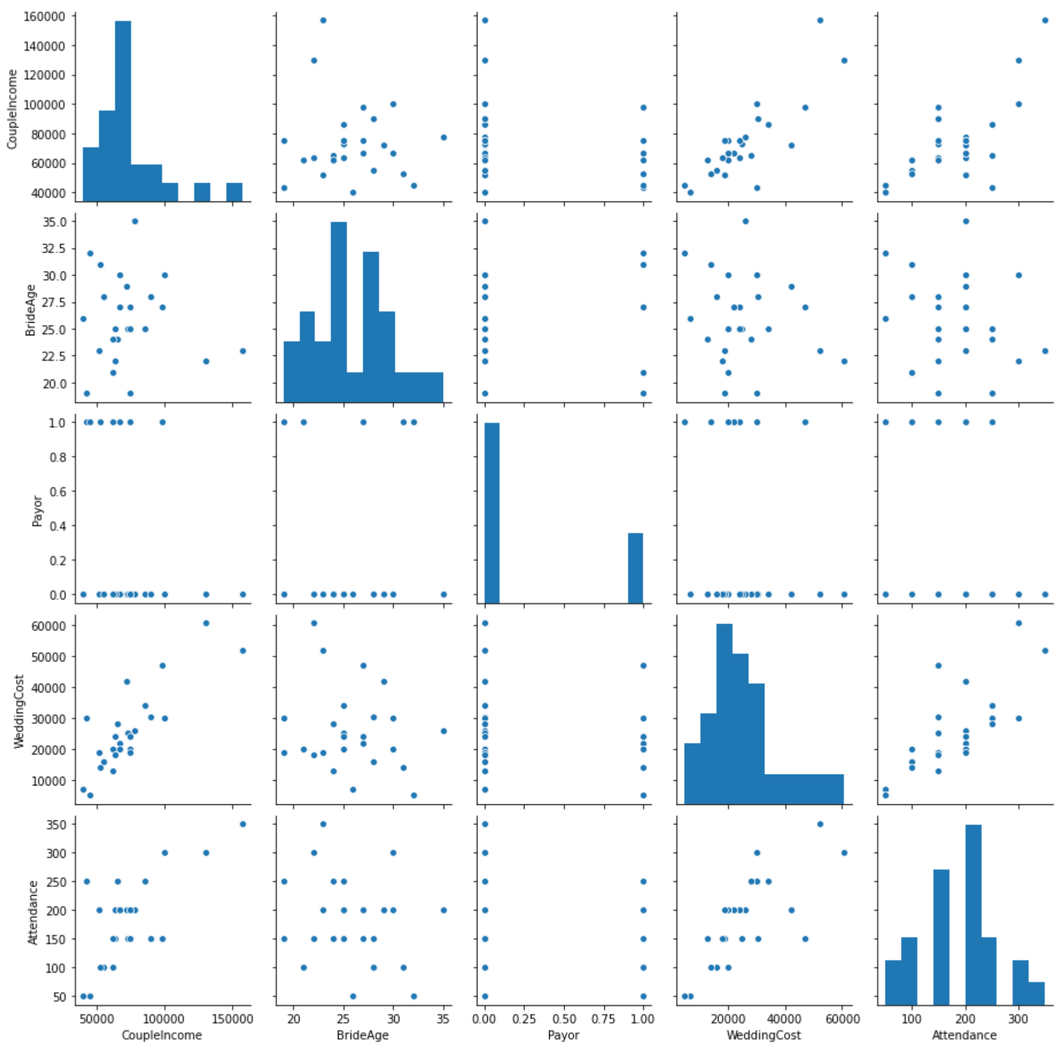
In [159]:
```
vif_data = pd.DataFrame() # VIF dataframe
vif_data["WeddingCost"] = dataset_independent_var.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(dataset_independent_var.values, i) #The Variance Inflation Factor (VIF) is a measure of colinearity among predictor variables within a multiple regression
for i in range(len(dataset_independent_var.columns))]

print(vif_data)
```

```
    WeddingCost        VIF
0  CoupleIncome  18.081861
1     BrideAge   8.975571
2         Payor   1.474974
3    Attendance  13.982807
```

As I can see, CoupleIncome and Attendance have very high values of VIF, indicating that these two variables are highly correlated. This is expected as the CoupleIncome in cost of wedding does influence the Attendance. Hence, considering these two features together leads to a model with high multicollinearity.

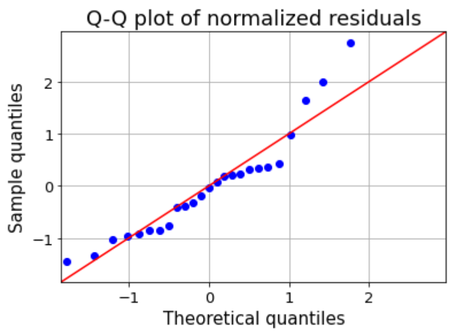
In [160]:
```
pairplot(df) #to see relationship among all the the variables
```

Out[160]: <seaborn.axisgrid.PairGrid at 0x2985c3c0af0>



In [164]:
```
model = sm.OLS.from_formula(
'WeddingCost ~ CoupleIncome + BrideAge + Payor + Attendance', data=df)
fitted = model.fit()


plt.figure(figsize=(8,5)) #creating the figure size
fig=qqplot(fitted.resid_pearson,line='45',fit='True') #resid_pearson signifies the residuals normalized to have unit variance.

plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.xlabel("Theoretical quantiles",fontsize=15) #The theoretical quantiles are placed along the x-axis. That is, the x-axis is not your data, it is simply an expectation of where your data should have been, if it were normal.
plt.ylabel("Sample quantiles",fontsize=15) # The actual data is plotted along the y-axis.
#The values are the standard deviations from the mean. So, 0 is the mean of the data, 1 is 1 standard deviation above, etc. This means, for instance, that 68.27% of all your data should be between -1 & 1, if you have a normal distribution

plt.title("Q-Q plot of normalized residuals",fontsize=18) #defining title

plt.grid(True)
plt.show()
```

```
<Figure size 576x360 with 0 Axes>
```



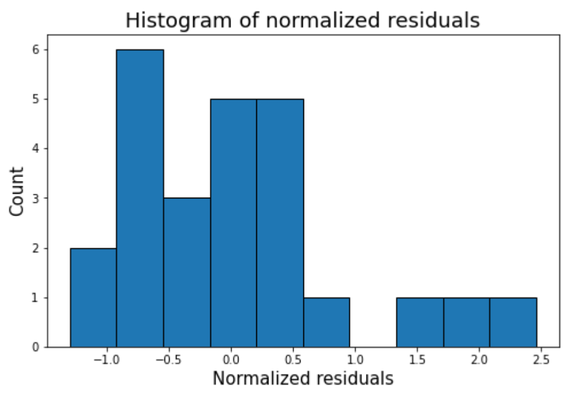
A Q-Q plot stands for a "quantile-quantile plot".

It is a plot where the axes are purposely transformed in order to make a normal (or Gaussian) distribution appear in a straight line. In other words, a perfectly normal distribution would exactly follow a line with slope = 1 and intercept = 0. Therefore, if the plot does not appear to be - roughly - a straight line, then the underlying distribution is not normal. If it bends up, then there are more "high flyer" values than expected.

In [165]:
```
mean_residuals= sum(fitted.resid)/len(fitted.resid) #check the mean of residuals is approx 0
mean_residuals
```

Out[165]:
```
4.8312358558177946e-11
```

In [167]:
```
plt.figure(figsize=(8,5))
plt.hist(fitted.resid_pearson,bins=10,edgecolor='k') #plotting histogram with bin size=10
plt.ylabel('Count',fontsize=15)
plt.xlabel('Normalized residuals',fontsize=15)
plt.title("Histogram of normalized residuals",fontsize=18)
plt.show()
```



The Q-Q plot (and the histogram above) shows that the normality assumption is not satisfied very well

The results of the Jarque-Bera test on the residuals also indicate that the p-value equals 0.112(>0.05) , therefore we cannot reject the null hypothesis of normal distribution.