Econ 430 - Project 3

By:

1. Snehil Shandilya: 306077534
2. Steven Spear: 606085052
3. Dhriti Sahoo: 505928964

In [6]:
```
# Load Modules and Functions
import statsmodels.api as sm
import statsmodels as sms
import seaborn as sns
import statsmodels.formula.api as smf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import linearmodels as plm
import numpy as np
import linearmodels as plm
import scipy.stats as stats
```

Question 1: Panel Data Models

Part 1: Our analysis will attempt to predict a firm's level of investment given its market value, capital stock, and 20 annual observations on 11 firms.

Number of observations - 220 (20 years for 11 firms)

Number of variables - 5

Variables name definitions:

Invest - Gross investment in dollars in the year 1947 Value - Market value as of Dec. 31 in dollar in the year 1947 Capital - Stock of plant and equipment in dollars as of 1947 Firm - General Motors, US Steel, General Electric, Chrysler, Atlantic Refining, IBM, Union Oil, Westinghouse, Goodyear, Diamond Match, American Steel Year - 1935 - 1954

Part 2

Descriptive analysis of data

In [7]:
```
# Load the data:
df = pd.read_csv("/Users/dhritisahoo/Desktop/Grunfeld.csv")
df
# The data set is balanced.
#This means that we measure for the same units across the full period
```

Out[7]:

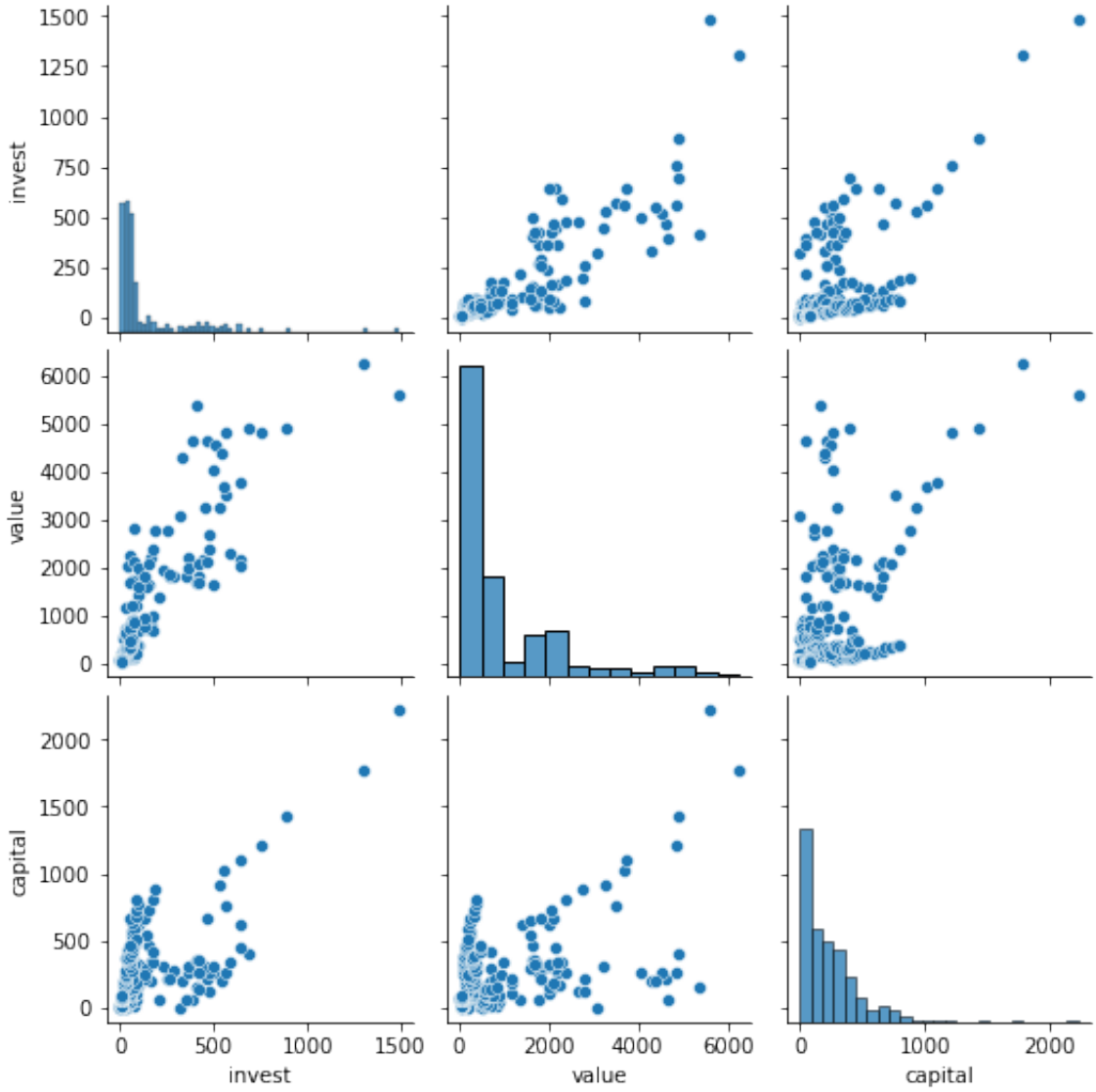
	Unnamed: 0	invest	value	capital	firm	year
0	1	317.600	3078.500	2.800	General Motors	1935
1	2	391.800	4661.700	52.600	General Motors	1936
2	3	410.600	5387.100	156.900	General Motors	1937
3	4	257.700	2792.200	209.200	General Motors	1938
4	5	330.800	4313.200	203.400	General Motors	1939
...	...	...	...	...	...	...
215	216	4.770	36.494	75.847	American Steel	1950
216	217	6.532	46.082	77.367	American Steel	1951
217	218	7.329	57.616	78.631	American Steel	1952
218	219	9.020	57.441	80.215	American Steel	1953
219	220	6.281	47.165	83.788	American Steel	1954

The pairplot compares the values between predictors to visually show trends.

The plot below suggests there is a positive correlation between investment, capital, and value.

In [8]:
```
import seaborn as sns
sns.pairplot(df,vars=['invest', 'value','capital'])
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x7fe3b8b7aa30>



The correlation plot below gives numerical evidence for what we saw in the pairplot. A higher value means a higher correlation where 1 is perfect correlation.

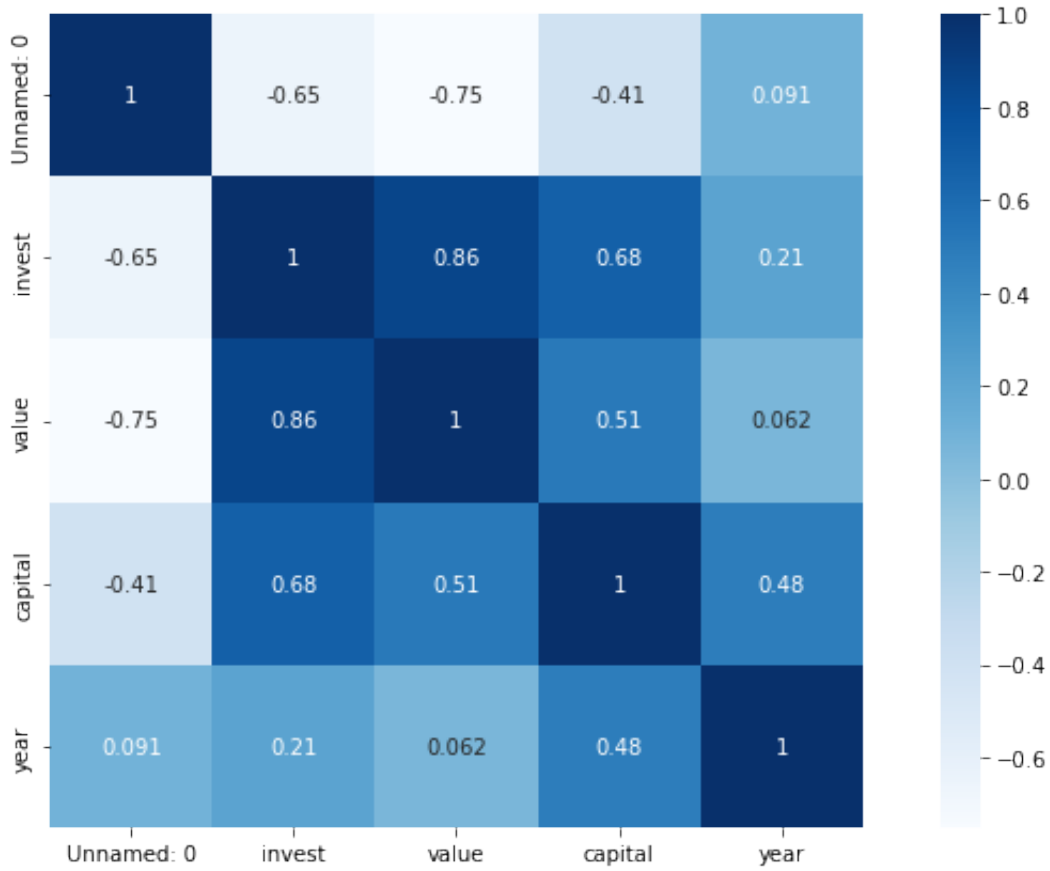
The correlation between invest and value is 0.86

The correlation between invest and capital is 0.68

The correlation between capital and value is 0.51

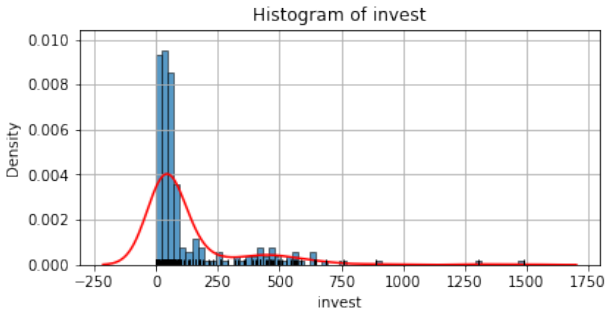
In [9]:
```
plt.figure(figsize=(13,7))
c= df.corr()
sns.heatmap(c,cmap="Blues",annot=True,square = True)
```

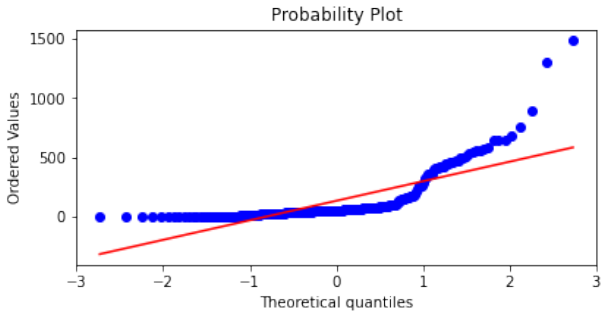
Out[9]: <AxesSubplot:>



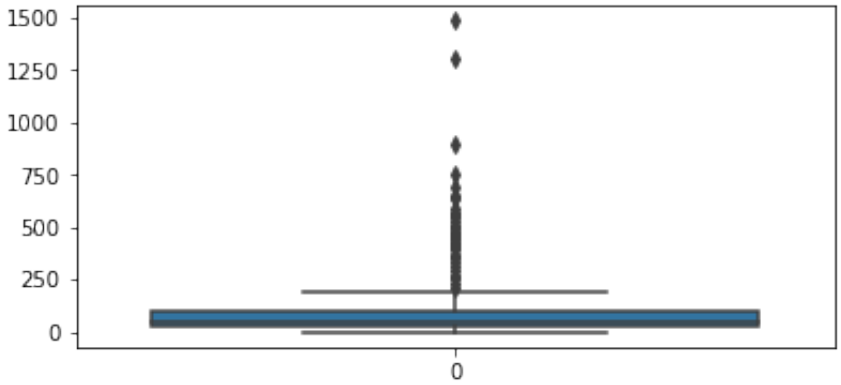
The histogram, probability plot, and boxplot all visually show the observations of invest to not be normally distributed.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [10]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of invest")
sns.histplot(df.invest, stat = "density")
sns.kdeplot(df.invest, color = "red")
sns.rugplot(df.invest, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df.invest, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df['invest'])
```



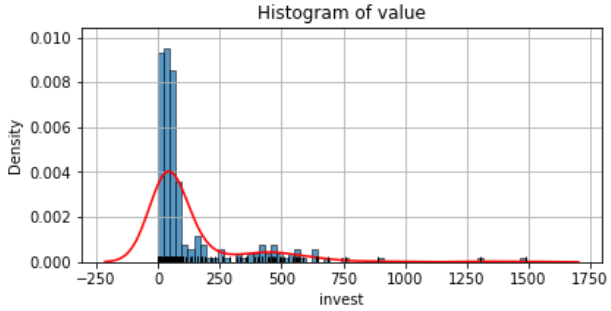


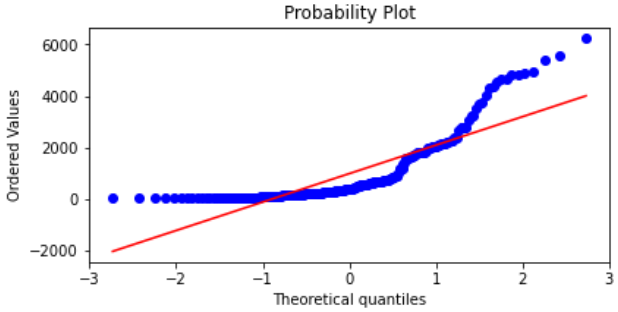
Out[10]: <AxesSubplot:>



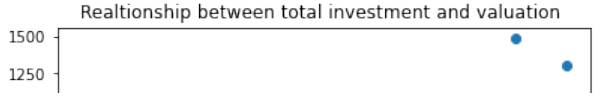
The histogram, probability plot, and boxplot all visually show the observations of value to not be normally distributed.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

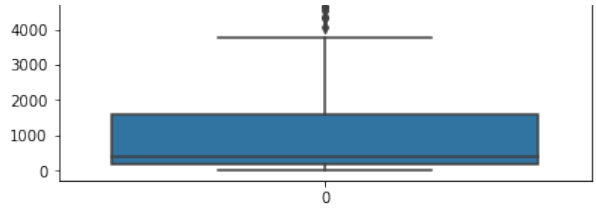
In [11]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of value")
sns.histplot(df.invest, stat = "density")
sns.kdeplot(df.invest, color = "red")
sns.rugplot(df.invest, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df.value, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df['value'])

plt.subplot(3,2,4)

plt.scatter(df["value"],df["invest"])
plt.xlabel("value")
plt.ylabel("invest")
plt.title("Realtionship between total investment and valuation ")
```

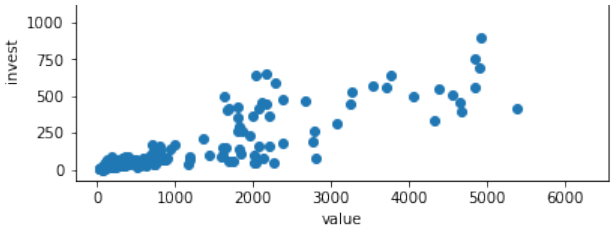




Out[11]:
```
Text(0.5, 1.0, 'Realtionship between total investment and valuation '
)
```

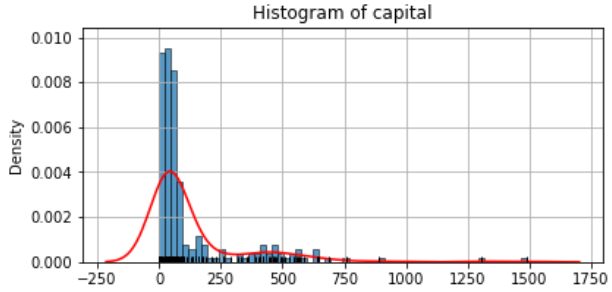


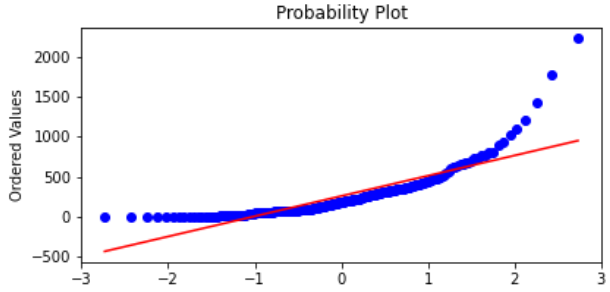




The histogram, probability plot, and boxplot all visually show the observations of capital to not be normally distributed.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

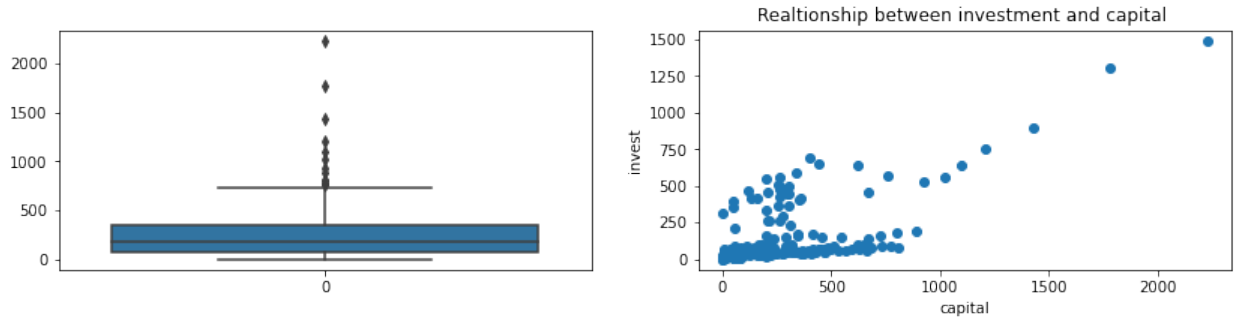
In [12]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of capital")
sns.histplot(df.invest, stat = "density")
sns.kdeplot(df.invest, color = "red")
sns.rugplot(df.invest, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df.capital, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df['capital'])

plt.subplot(3,2,4)

plt.scatter(df["capital"],df["invest"])
plt.xlabel("capital")
plt.ylabel("invest")
plt.title("Realtionship between investment and capital ")
```





invest Theoretical quantiles

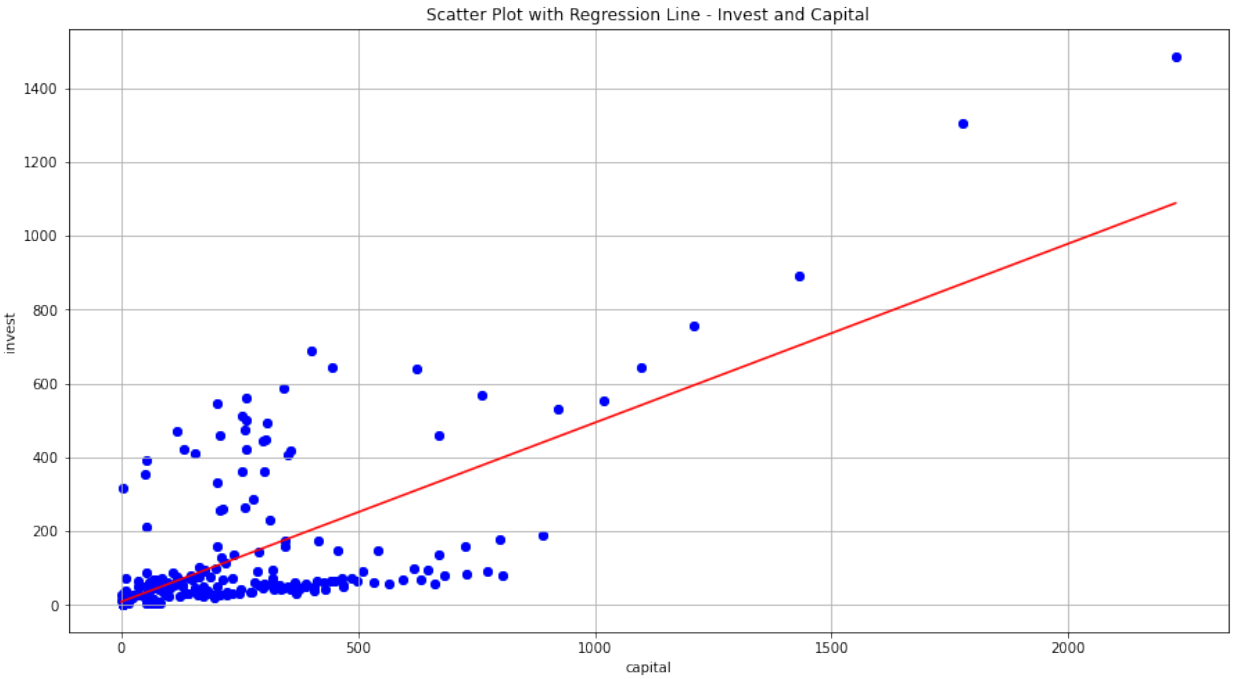
Out[12]: Text(0.5, 1.0, 'Realtionship between investment and capital ')



The scatterplot of invest and capital shows a few influential observations and a linear relationship.

In [13]:
```
plt.figure(figsize = (15, 8))


plt.scatter(df["capital"], df["invest"])

# Create regression line
m,b = np.polyfit(df["capital"], df["invest"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df.capital.max(), 1000)

# combining the two plots
plt.scatter(df["capital"], df["invest"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line - Invest and Capital")
plt.ylabel("invest")
plt.xlabel("capital")
plt.grid()
```

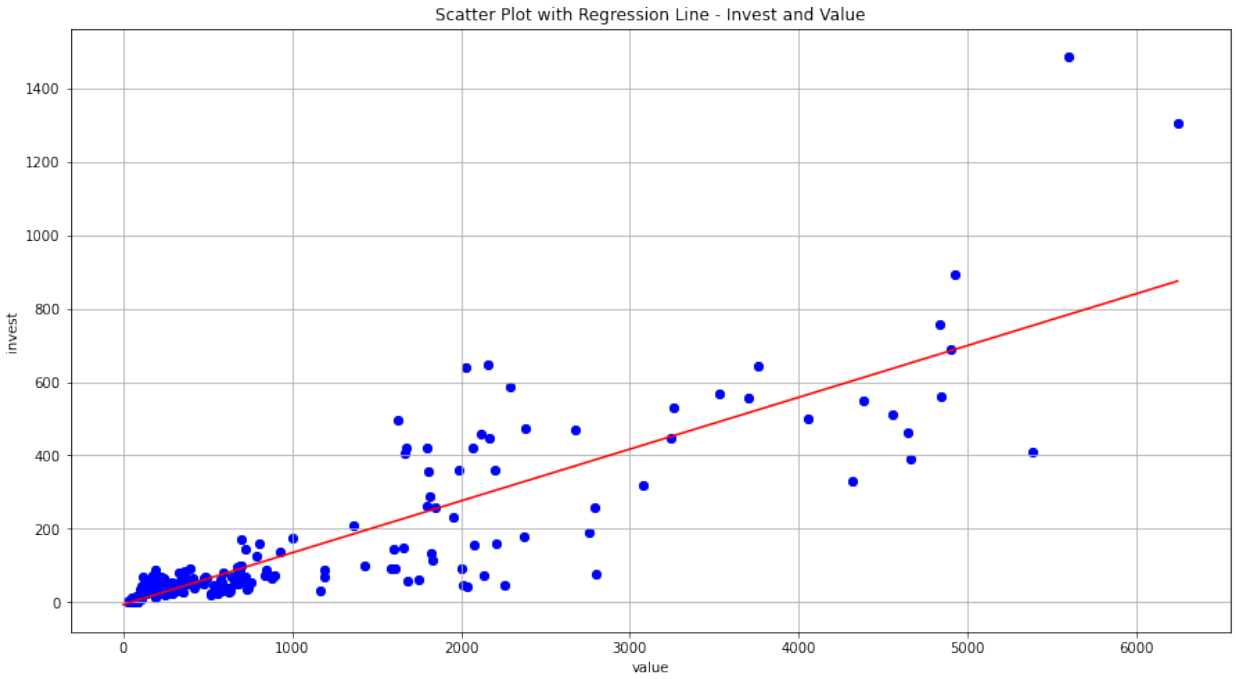
```
The slope of the regression line is: 0.48519136672262414 The Intercep
t is: 8.565055640258556
```



The scatterplot of invest and value shows two potentially influential observations and a linear relationship.

In [14]:
```
plt.figure(figsize = (15, 8))


plt.scatter(df["value"], df["invest"])

# Create regression line
m,b = np.polyfit(df["value"], df["invest"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df.value.max(), 100)

# combining the two plots
plt.scatter(df["value"], df["invest"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line - Invest and Value")
plt.ylabel("invest")
plt.xlabel("value")
plt.grid()
```

```
The slope of the regression line is: 0.141092580112949 The Intercept
is: -6.169093085712734
```



In the OLS results, we see that our primary predictors, capital and value, are significant, but we need to check for a multicollinearity problem that was suggested in the visual analysis in the correlation matrix.

In [15]:
```
# Specify the Model
model = smf.ols(formula='invest ~ capital + value', data=df)
```

```
result = model.fit()

print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  invest   R-squared:                       0.818
Model:                            OLS   Adj. R-squared:                  0.816
Method:                 Least Squares   F-statistic:                     487.3
Date:                Fri, 02 Dec 2022   Prob (F-statistic):           5.58e-81
Time:                        22:17:12   Log-Likelihood:                -1301.3
No. Observations:                 220   AIC:                             2609.
Df Residuals:                     217   BIC:                             2619.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -38.4101      8.413     -4.565      0.000     -54.992     -21.828
capital        0.2275      0.024      9.390      0.000       0.180       0.275
value          0.1145      0.006     20.753      0.000       0.104       0.125
==============================================================================
Omnibus:                       33.923   Durbin-Watson:                   0.357
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              139.212
Skew:                           0.491   Prob(JB):                     5.90e-31
Kurtosis:                       6.771   Cond. No.                     2.27e+03
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors i
s correctly specified.
[2] The condition number is large, 2.27e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

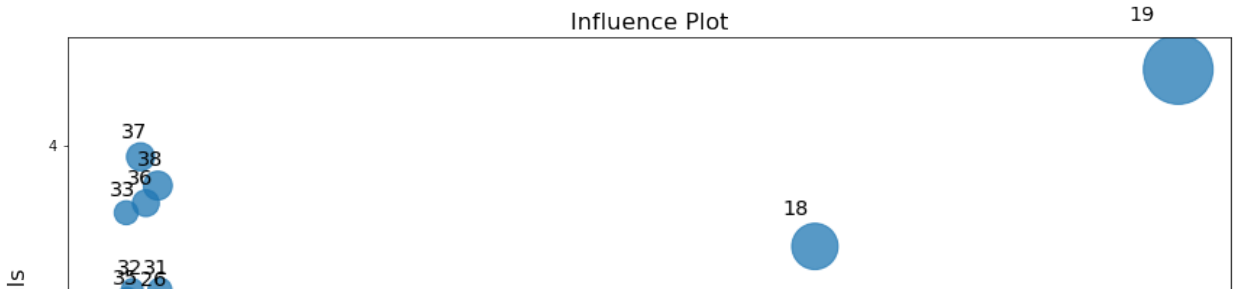
Because the VIF numbers are similar, there will be no vairables removed.Typically, a threshold a 4 would be used to eliminate variables with a higher score.

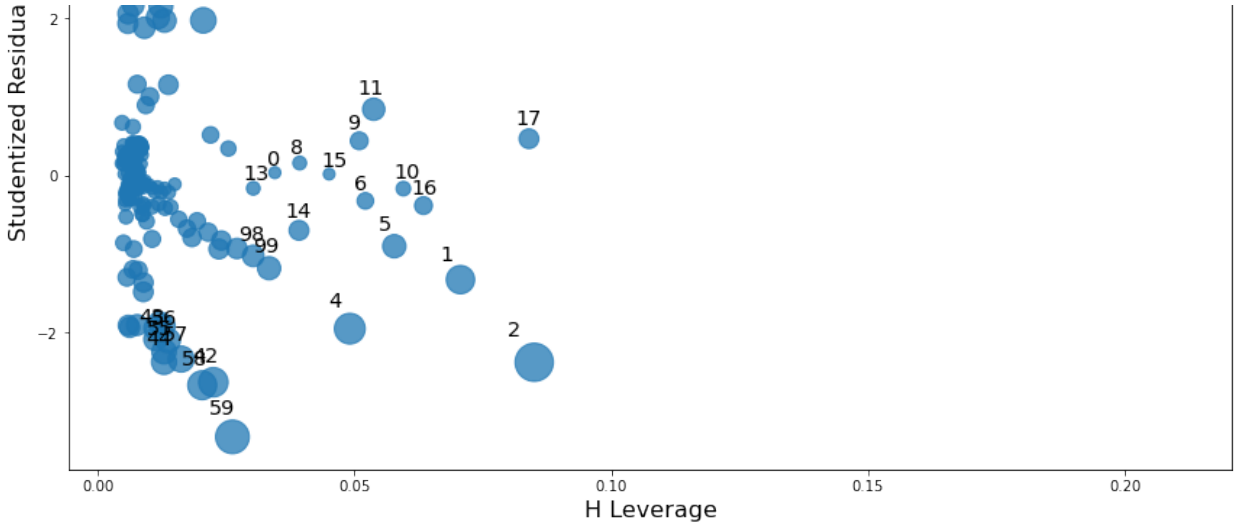
In [16]:
```
## variables are not multicollinear
import statsmodels.stats.outliers_influence as smo
import patsy as pt

# extract matrices using patsy:
y, X = pt.dmatrices('invest ~ capital + value', data=df, return_type='

# get VIF:
K = X.shape[1]
VIF = np.empty(K)
for i in range(K):
    VIF[i] = smo.variance_inflation_factor(X.values, i)
print(f'VIF: \n{VIF}\n')
```

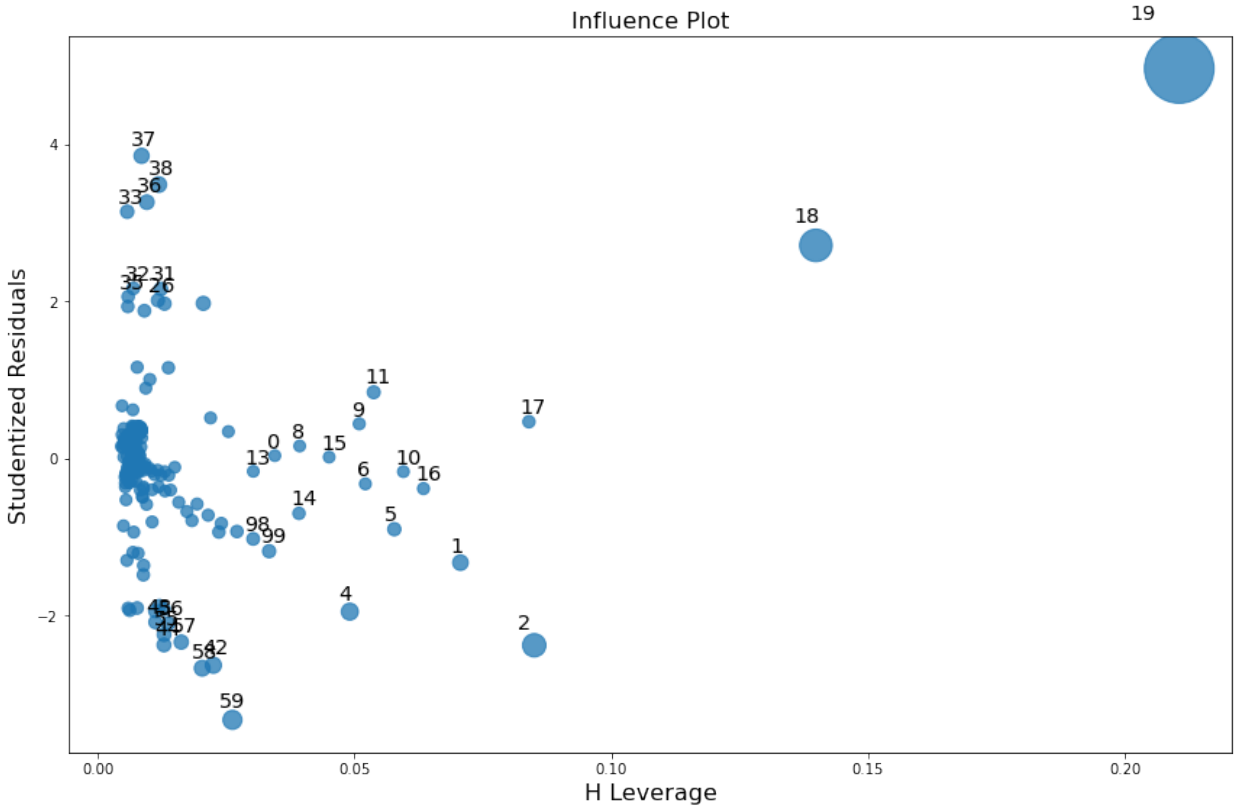
```
VIF:
[1.9106169  1.35615621 1.35615621]
```

The influence plots shows observations that have high leverage using DFFITS and Cook's Distance methods. Observations further to the right are potentially influential and problematic to the model.

In [17]:
```
# Outliers, high leverage, influential obs
figd, ax = plt.subplots(figsize=(12,8))
figd = sm.graphics.influence_plot(result, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sm.graphics.influence_plot(result, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```




Studentized Residua
H Leverage
0.00
0.05
0.10
0.15
0.20
2
0
−2
11
9
17
0
8
15
13
10
6
16
14
5
98
99
1
4
2
59


Influence Plot
Studentized Residuals
H Leverage
0.00
0.05
0.10
0.15
0.20
4
2
0
−2
19
18
37
38
36
33
11
9
17
0
8
15
13
10
6
16
14
5
98
99
1
4
2
59

The plots below show the relationship between predicted y values, residuals, and a specific predictor.

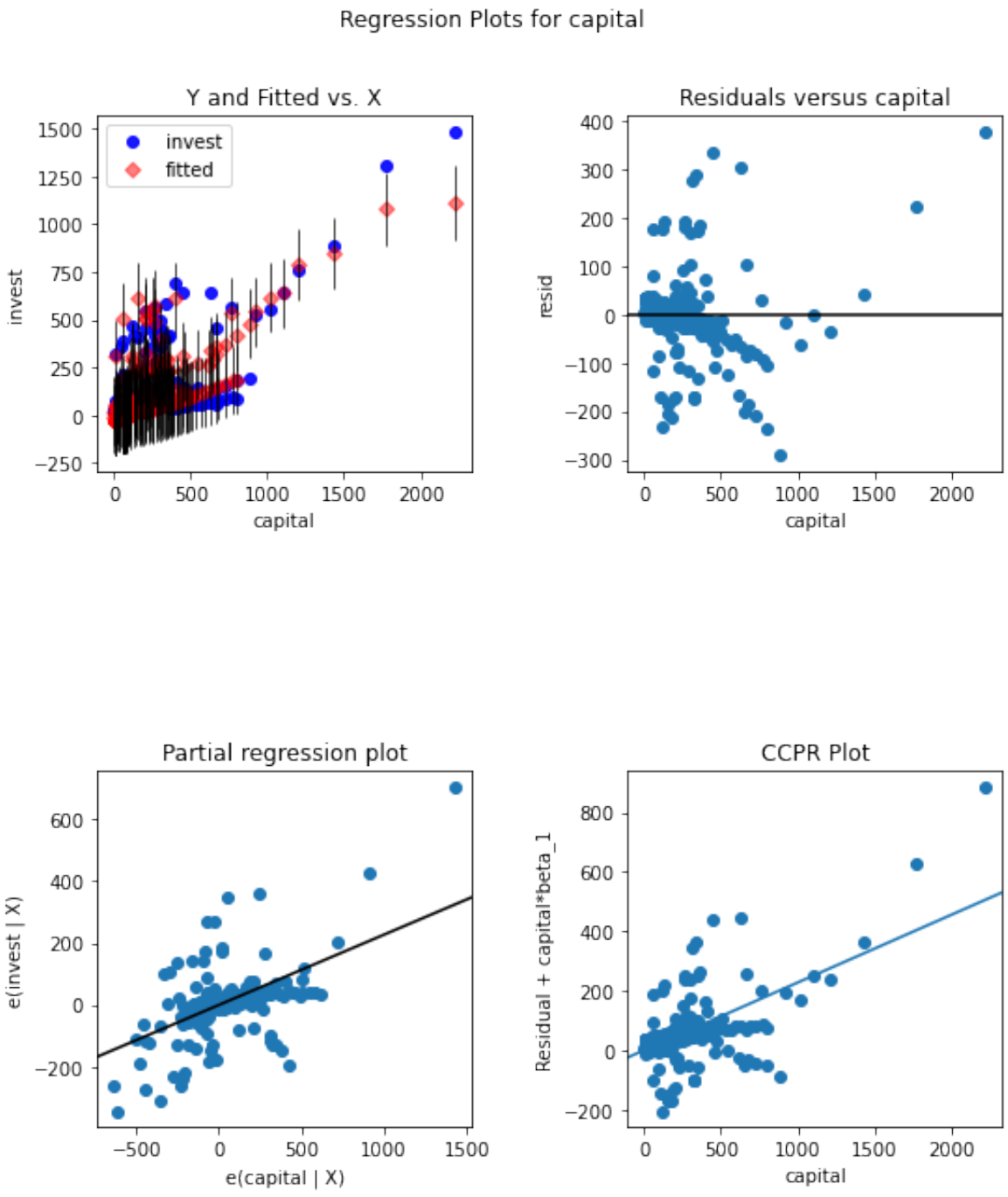
The residual plots exhibit a non-constant variance indicating heteroskedacity .

CCPR plot visiually indicated thatcapital is a linear response variable.

The partial regression plot shows the regression line if that predictor was the only predictor in the regression (showing a simple linear regression as opposed to the multiple regression with all the predictors).

In [18]:
```
fig = sm.graphics.plot_regress_exog(result, "capital")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



The plots below show the relationship between predicted y values, residuals, and a specific predictor.

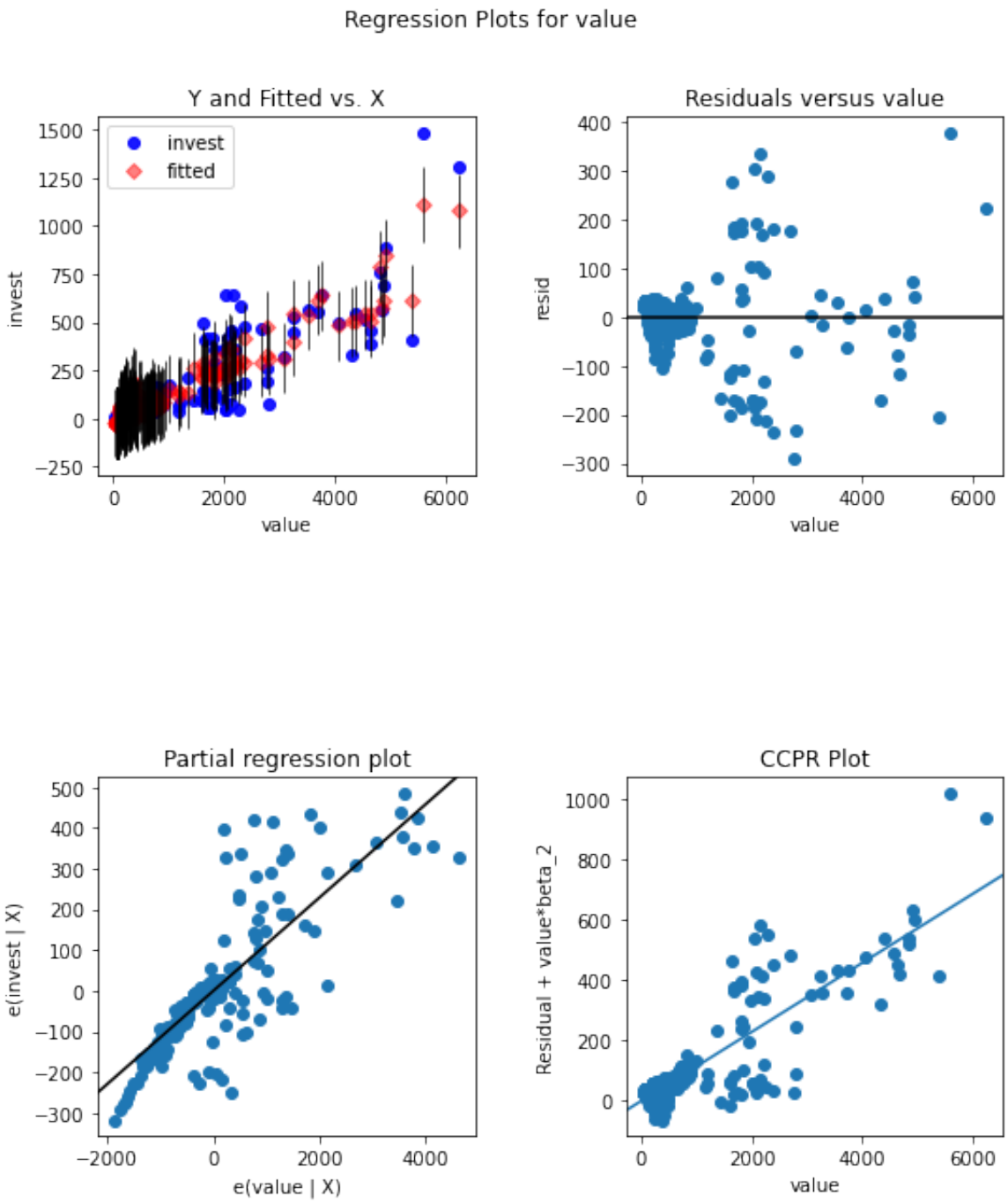
The residual plots indicate non constant variance.

The CCPR plot indicates somewhat of a linear relationship for the reponse variable value.

The partial regression plot shows the regression line if that predictor was the only predictor in the regression (showing a simple linear regression as opposed to the multiple regression with all the predictors).

In [19]:
```
fig = sm.graphics.plot_regress_exog(result, "value")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



Part 3

Model preference

In [20]:
```
df = df.set_index(['firm', 'year'], drop = False)
```

In [21]:
```
## Pooled Effect model
dd= plm.PooledOLS.from_formula(formula='invest ~ capital + value', dat
results_ols = dd.fit()
## Random Effect model
reg_re = plm.RandomEffects.from_formula(
    formula='invest~ capital + value + C(year)', data=df)
results_re = reg_re.fit()
# Fixed effect model
reg_fe = plm.PanelOLS.from_formula(
    formula='invest~ capital + value + C(year) + EntityEffects', data=
results_fe = reg_fe.fit()

# print results:
theta_hat = results_re.theta.iloc[0, 0]
print(f'theta_hat: {theta_hat}\n')

table_ols = pd.DataFrame({'b': round(results_ols.params, 4),
                          'se': round(results_ols.std_errors, 4),
                          't': round(results_ols.tstats, 4),
                          'pval': round(results_ols.pvalues, 4)})
print(f'table_ols: \n{table_ols}\n')

table_re = pd.DataFrame({'b': round(results_re.params, 4),
                         'se': round(results_re.std_errors, 4),
                         't': round(results_re.tstats, 4),
                         'pval': round(results_re.pvalues, 4)})
print(f'table_re: \n{table_re}\n')

table_fe = pd.DataFrame({'b': round(results_fe.params, 4),
                         'se': round(results_fe.std_errors, 4),
                         't': round(results_fe.tstats, 4),
                         'pval': round(results_fe.pvalues, 4)})
print(f'table_fe: \n{table_fe}\n')
```

```
theta_hat: 0.0

table_ols:
                b      se        t  pval
capital  0.1824  0.0231   7.8938   0.0
value    0.1078  0.0056  19.4085   0.0
```

```
table_re:
                          b       se        t    pval
C(year)[T.1935] -21.6815  28.3544  -0.7647  0.4454
C(year)[T.1936] -36.8679  28.6125  -1.2885  0.1991
C(year)[T.1937] -52.5230  28.8556  -1.8202  0.0702
C(year)[T.1938] -47.6455  28.4664  -1.6737  0.0958
C(year)[T.1939] -72.9290  28.6168  -2.5485  0.0116
C(year)[T.1940] -49.2023  28.6555  -1.7170  0.0875
C(year)[T.1941] -23.6826  28.6487  -0.8267  0.4094
C(year)[T.1942] -22.0378  28.6157  -0.7701  0.4421
C(year)[T.1943] -40.4773  28.6857  -1.4111  0.1598
C(year)[T.1944] -41.1787  28.6946  -1.4351  0.1528
C(year)[T.1945] -51.4238  28.7672  -1.7876  0.0754
C(year)[T.1946] -27.8022  28.8404  -0.9640  0.3362
C(year)[T.1947] -26.0464  29.0588  -0.8963  0.3712
C(year)[T.1948] -24.4840  29.3033  -0.8355  0.4044
C(year)[T.1949] -46.9765  29.5447  -1.5900  0.1134
C(year)[T.1950] -46.6205  29.6854  -1.5705  0.1179
C(year)[T.1951] -31.1508  29.8330  -1.0442  0.2977
C(year)[T.1952] -25.5088  30.3064  -0.8417  0.4010
C(year)[T.1953] -17.6278  31.0110  -0.5684  0.5704
C(year)[T.1954] -31.0731  31.8745  -0.9749  0.3308
capital           0.2166   0.0299   7.2436  0.0000
value             0.1158   0.0060  19.4340  0.0000

table_fe:
                           b       se        t    pval
C(year)[T.1935]  -30.5344  16.9435  -1.8021  0.0731
C(year)[T.1936]  -47.4937  19.3696  -2.4520  0.0151
C(year)[T.1937]  -66.9101  21.5220  -3.1089  0.0022
C(year)[T.1938]  -66.1582  17.7154  -3.7345  0.0002
C(year)[T.1939]  -93.6338  19.2782  -4.8570  0.0000
C(year)[T.1940]  -70.3592  19.6594  -3.5789  0.0004
C(year)[T.1941]  -47.0222  19.3042  -2.4359  0.0158
C(year)[T.1942]  -48.5338  17.9305  -2.7068  0.0074
C(year)[T.1943]  -68.3069  18.6788  -3.6569  0.0003
C(year)[T.1944]  -68.8545  18.8876  -3.6455  0.0003
C(year)[T.1945]  -80.0739  19.6781  -4.0692  0.0001
C(year)[T.1946]  -58.2888  20.1604  -2.8913  0.0043
C(year)[T.1947]  -65.4120  18.4042  -3.5542  0.0005
C(year)[T.1948]  -68.8652  18.3640  -3.7500  0.0002
C(year)[T.1949]  -95.7352  18.6200  -5.1415  0.0000
C(year)[T.1950]  -97.9222  19.0366  -5.1439  0.0000
C(year)[T.1951]  -85.3691  20.5665  -4.1509  0.0001
C(year)[T.1952]  -87.0235  21.1280  -4.1189  0.0001
C(year)[T.1953]  -89.0470  23.0464  -3.8638  0.0002
C(year)[T.1954] -112.3284  23.1984  -4.8421  0.0000
capital            0.3514   0.0210  16.6964  0.0000
value              0.1167   0.0129   9.0219  0.0000
```

Hausman Test

In [22]:
```
import numpy.linalg
## Fixed
b_fe = results_fe.params
b_fe_cov = results_fe.cov

## Random
results_re = reg_re.fit()
b_re = results_re.params
b_re_cov = results_re.cov

# Hausman test of FE vs. RE
# (I) find overlapping coefficients:
common_coef = set(results_fe.params.index).intersection(results_re.par

# (II) calculate differences between FE and RE:
b_diff = np.array(results_fe.params[common_coef] - results_re.params[c
df = len(b_diff)
b_diff.reshape((df, 1))
b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] - b_re_co
b_cov_diff.reshape((df, df))

# (III) calculate test statistic:
stat = abs(np.transpose(b_diff) @ np.linalg.inv(b_cov_diff) @ b_diff)
pval = 1 - stats.chi2.cdf(stat, df)

print(f'stat: {stat}\n')
print(f'pval: {pval}\n')
```

```
stat: 39.997997544327106

pval: 0.010817543732512758


/var/folders/33/z90mwnyd0ps6j1hfy_5yn41w0000gn/T/ipykernel_22926/1767
762092.py:16: FutureWarning: Passing a set as an indexer is deprecate
d and will raise in a future version. Use a list instead.
  b_diff = np.array(results_fe.params[common_coef] - results_re.param
s[common_coef])
/var/folders/33/z90mwnyd0ps6j1hfy_5yn41w0000gn/T/ipykernel_22926/1767
762092.py:16: FutureWarning: Passing a set as an indexer is deprecate
d and will raise in a future version. Use a list instead.
  b_diff = np.array(results_fe.params[common_coef] - results_re.param
s[common_coef])
/var/folders/33/z90mwnyd0ps6j1hfy_5yn41w0000gn/T/ipykernel_22926/1767
762092.py:19: FutureWarning: Passing a set as an indexer is deprecate
d and will raise in a future version. Use a list instead.
```

```
  b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] - b_re
_cov.loc[common_coef, common_coef])
/var/folders/33/z90mwnyd0ps6j1hfy_5yn41w0000gn/T/ipykernel_22926/1767
762092.py:19: FutureWarning: Passing a set as an indexer is deprecate
d and will raise in a future version. Use a list instead.
  b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] - b_re
_cov.loc[common_coef, common_coef])
```

The null hypothesis of the Hausman Test is that the Random Effects model is preferred. Because the test returns significant, we reject the null and conclude the Fixed Effects model is preferred.

Question 2

Qualitative Dependent Variable Models

Part 1

The Framingham dataset shows 15 predictors to determine the 10 year risk of coronary heart disease. The analysis will show which predictors are most likely to lead to an increased risk.

In [23]:
```
df1 = pd.read_csv("/Users/dhritisahoo/Desktop/framingham.csv")
df1
```

Out[23]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHy
0	1	39	4.0	0	0.0	0.0	0	
1	0	46	2.0	0	0.0	0.0	0	
2	1	48	1.0	1	20.0	0.0	0	
3	0	61	3.0	1	30.0	0.0	0	
4	0	46	3.0	1	23.0	0.0	0	
...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	
4234	1	51	3.0	1	43.0	0.0	0	
4235	0	48	2.0	1	20.0	NaN	0	
4236	0	44	1.0	1	15.0	0.0	0	
4237	0	52	2.0	0	0.0	0.0	0	

4238 rows × 16 columns

In [24]:
```
df1.isnull().any()
```

Out[24]:
```
male               False
age                False
education           True
currentSmoker      False
cigsPerDay          True
BPMeds              True
prevalentStroke    False
prevalentHyp       False
diabetes           False
totChol             True
sysBP              False
diaBP              False
BMI                 True
heartRate           True
glucose             True
TenYearCHD         False
dtype: bool
```

In [25]:
```
# filling NA values with median
df2 = df1.fillna(df1.median())
df2
```

Out[25]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentH
0	1	39	4.0	0	0.0	0.0	0	
1	0	46	2.0	0	0.0	0.0	0	
2	1	48	1.0	1	20.0	0.0	0	
3	0	61	3.0	1	30.0	0.0	0	
4	0	46	3.0	1	23.0	0.0	0	
...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	
4234	1	51	3.0	1	43.0	0.0	0	
4235	0	48	2.0	1	20.0	0.0	0	
4236	0	44	1.0	1	15.0	0.0	0	
4237	0	52	2.0	0	0.0	0.0	0	

4238 rows × 16 columns

We find NaN values in some of the columns. Therefore, we impute them by replacing the Nan values with the median. The reason we do not use Mean is because of the presence of extreme values in the model. Therefore, median seems the correct fit.

In [26]:
```
df2.isnull().any()
```

Out[26]:
```
male               False
age                False
education          False
currentSmoker      False
cigsPerDay         False
BPMeds             False
prevalentStroke    False
prevalentHyp       False
diabetes           False
totChol            False
sysBP              False
diaBP              False
BMI                False
heartRate          False
glucose            False
TenYearCHD         False
dtype: bool
```

Now, we see that the NaN values have been accounted for.

Part 2

Descriptive Analysis

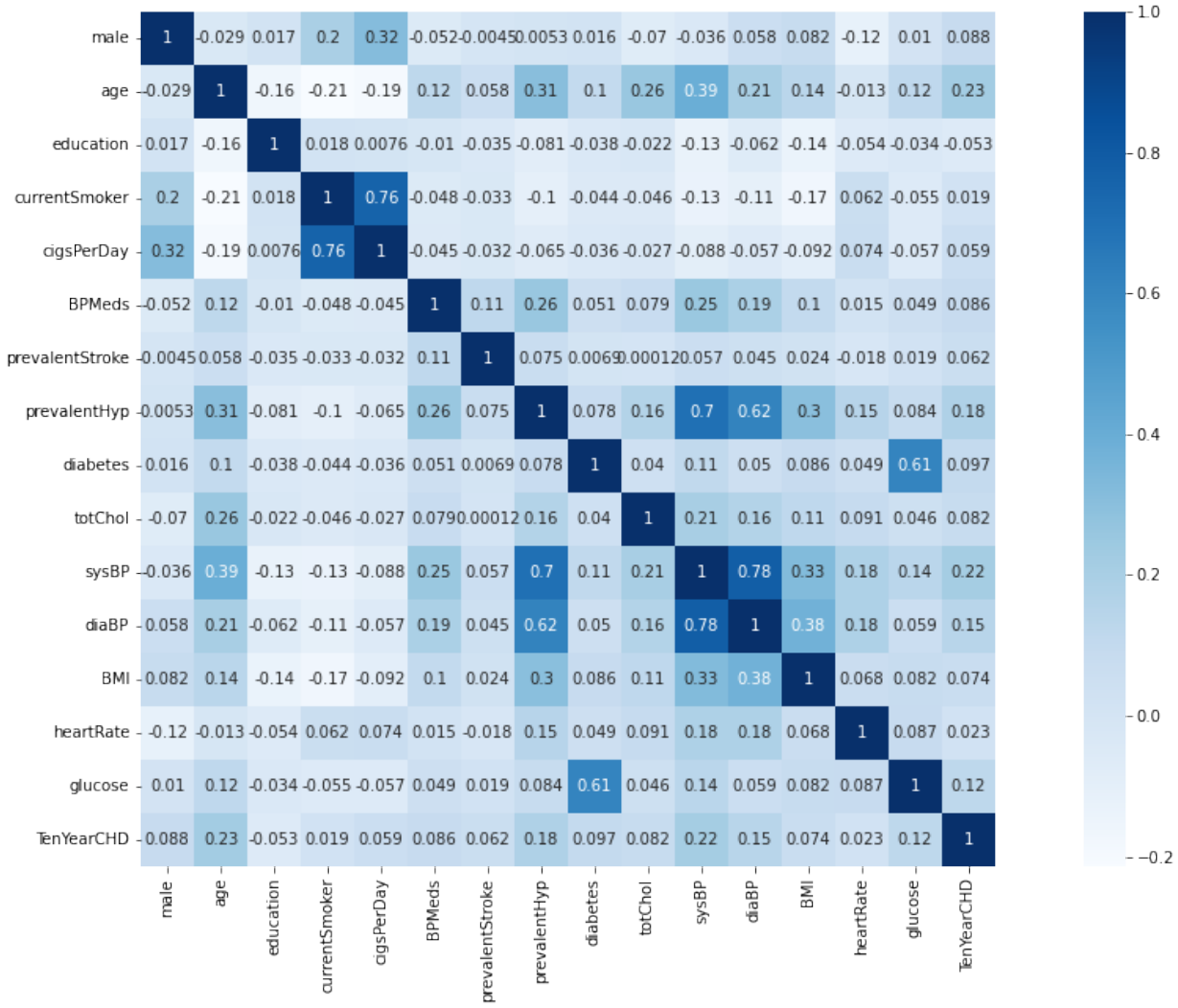
The correlation matrix and correlation plot show the relationship between non-binary predictors. We see that current smoker/cigsperday, SysBP and DiaBP with Prevalent hypertensive, and diabetes/glucose may have a strong correlation.

In [27]:
```
import seaborn as sns
sns.pairplot(df2,vars=['education', 'cigsPerDay', 'totChol','sysBP', '
```

Out[27]:
```
<seaborn.axisgrid.PairGrid at 0x7fe3baa5da30>
```

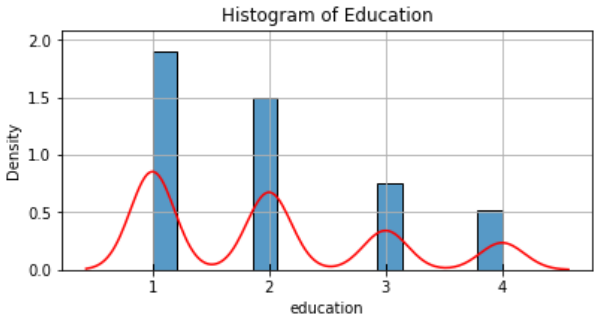
In [28]:
```
plt.figure(figsize=(20,10))
c= df2.corr()
sns.heatmap(c,cmap="Blues",annot=True,square = True)
```

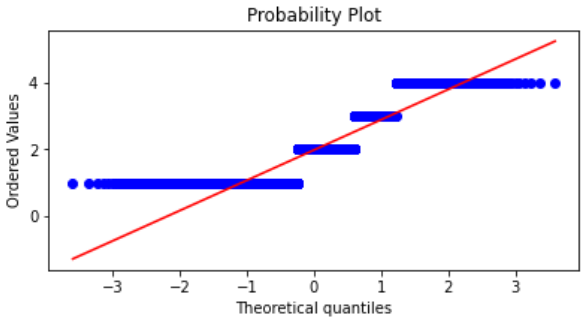
Out[28]: <AxesSubplot:>



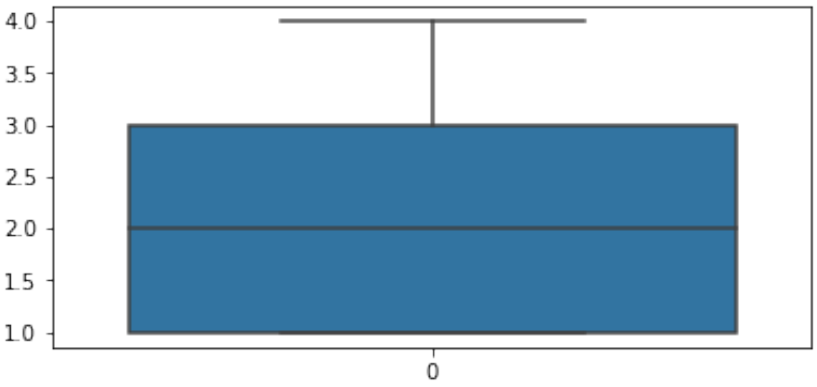
The histogram, probability plot, and boxplot all visually show the observations of education to not be normally distributed.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [29]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Education")
sns.histplot(df2.education, stat = "density")
sns.kdeplot(df2.education, color = "red")
sns.rugplot(df2.education, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.education, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['education'])
```



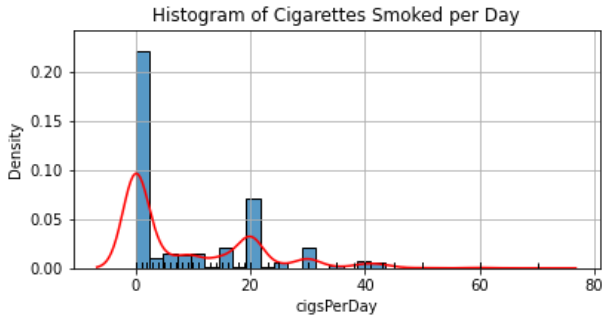


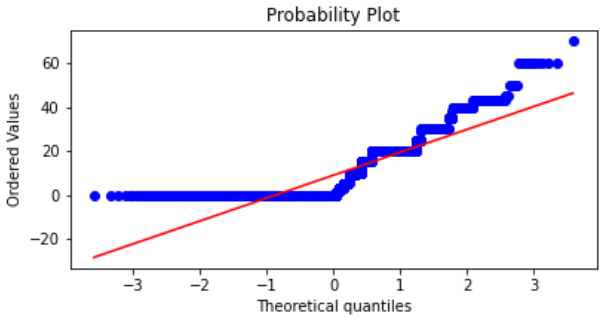
Out[29]: <AxesSubplot:>



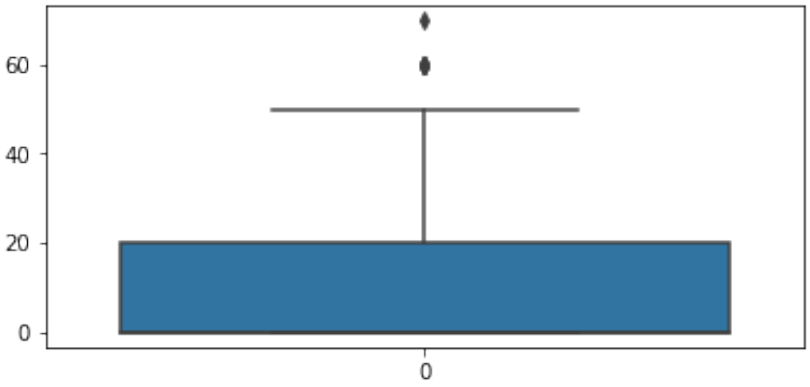
The histogram, probability plot, and boxplot all visually show the observations of cigsPerDay to not be normally distributed.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [30]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Cigarettes Smoked per Day")
sns.histplot(df2.cigsPerDay, stat = "density")
sns.kdeplot(df2.cigsPerDay, color = "red")
sns.rugplot(df2.cigsPerDay, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.cigsPerDay, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['cigsPerDay'])
```



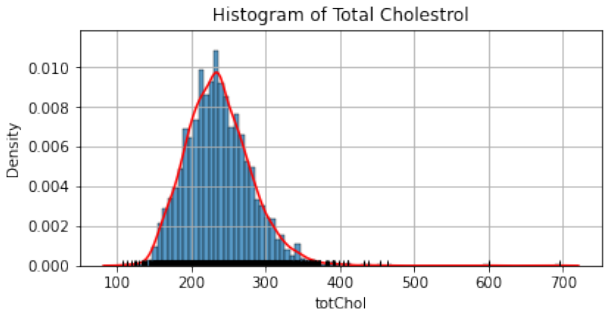


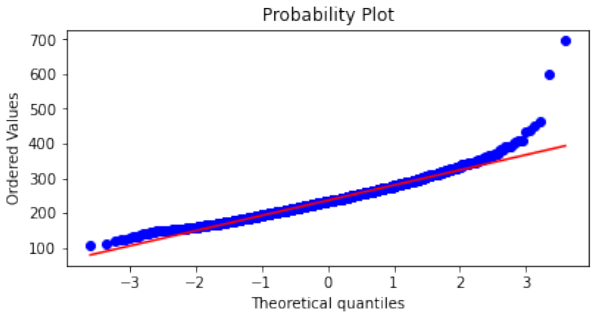
Out[30]: <AxesSubplot:>



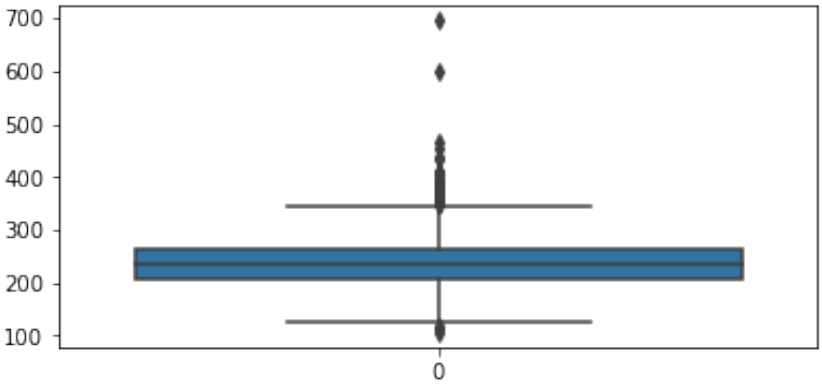
The histogram, probability plot, and boxplot all visually show the observations of total cholesterol to be mostly normally distributed, but skewed at the ends. The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [31]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Total Cholestrol")
sns.histplot(df2.totChol, stat = "density")
sns.kdeplot(df2.totChol, color = "red")
sns.rugplot(df2.totChol, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.totChol, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['totChol'])
```



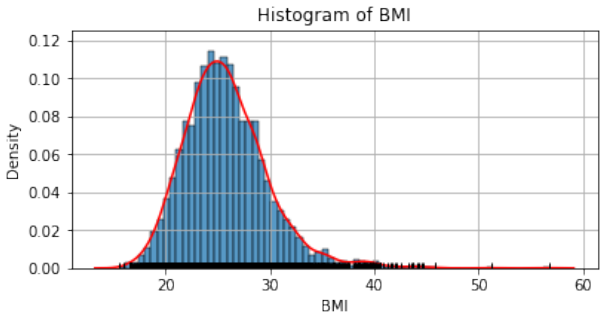


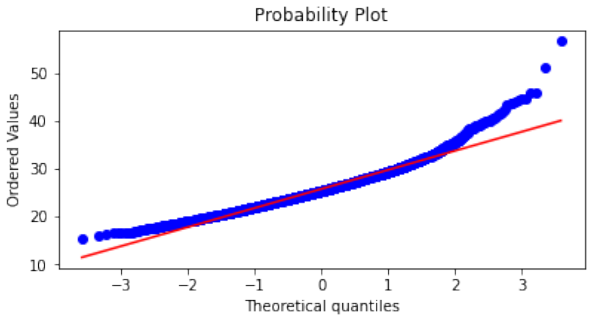
Out[31]: <AxesSubplot:>



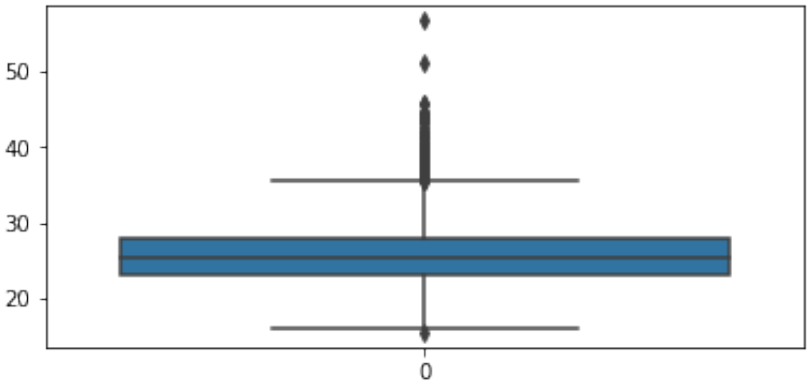
The histogram, probability plot, and boxplot all visually show the observations of BMI may be slightly skewed in distribution.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [32]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of BMI")
sns.histplot(df2.BMI, stat = "density")
sns.kdeplot(df2.BMI, color = "red")
sns.rugplot(df2.BMI, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.BMI, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['BMI'])
```



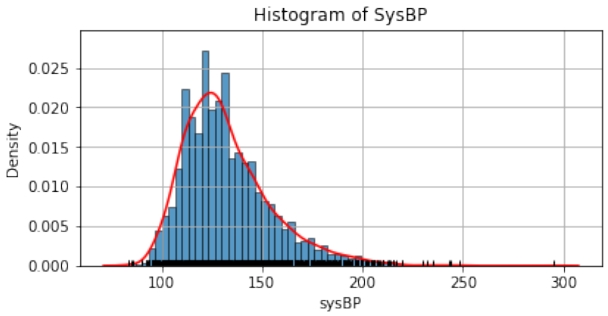


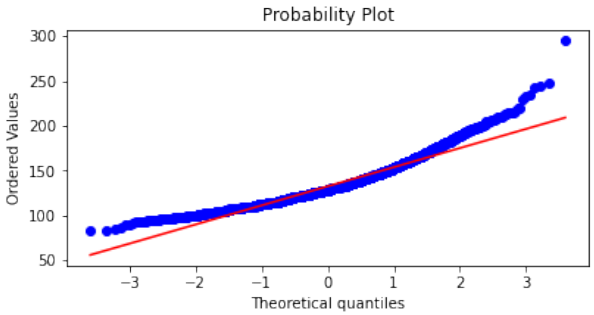
Out[32]: <AxesSubplot:>



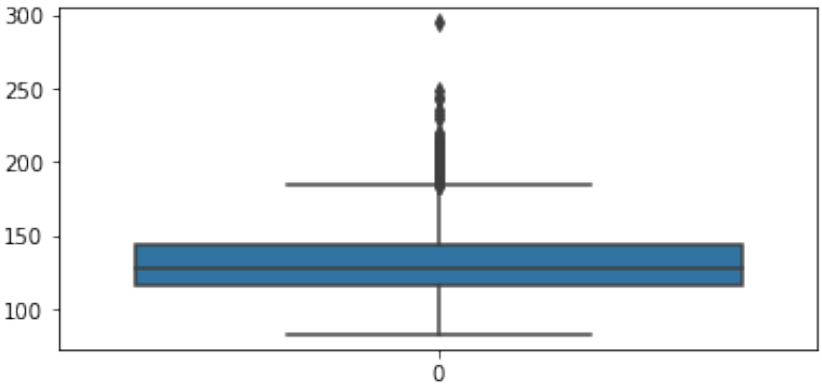
The histogram, probability plot, and boxplot all visually show the observations of systolic blood pressure to be slightly skewed in distribution.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [33]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of SysBP")
sns.histplot(df2.sysBP, stat = "density")
sns.kdeplot(df2.sysBP, color = "red")
sns.rugplot(df2.sysBP, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.sysBP, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['sysBP'])
```



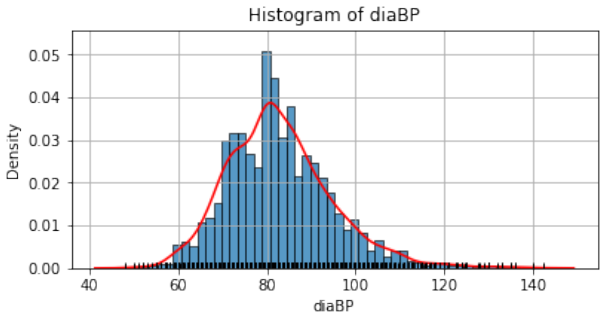


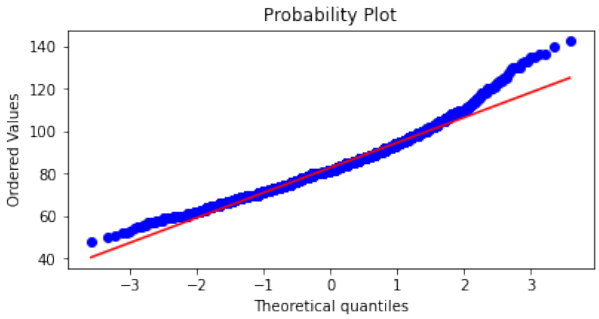
Out[33]: <AxesSubplot:>



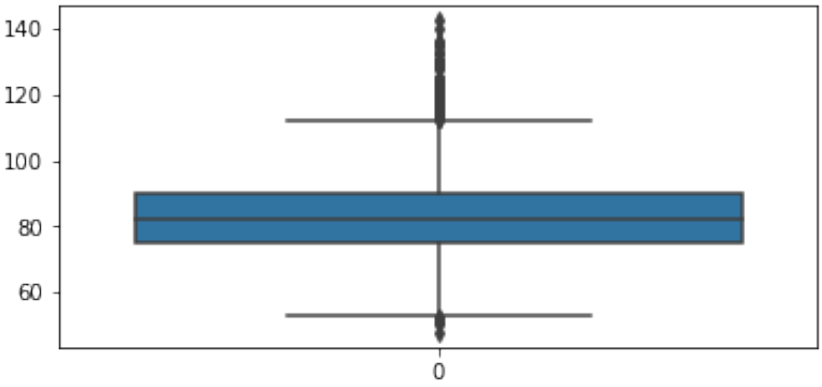
The histogram, probability plot, and boxplot all visually show the observations of diastolic blood pressure to be slightly skewed in distribution.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [34]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of diaBP")
sns.histplot(df2.diaBP, stat = "density")
sns.kdeplot(df2.diaBP, color = "red")
sns.rugplot(df2.diaBP, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.diaBP, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['diaBP'])
```



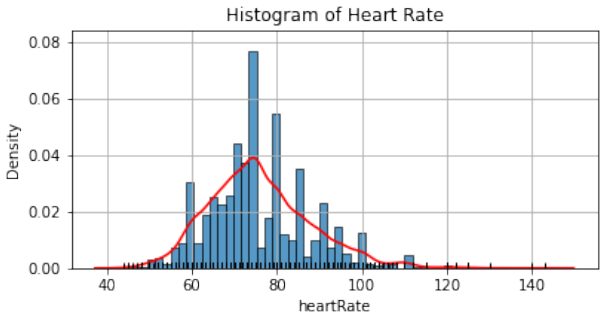


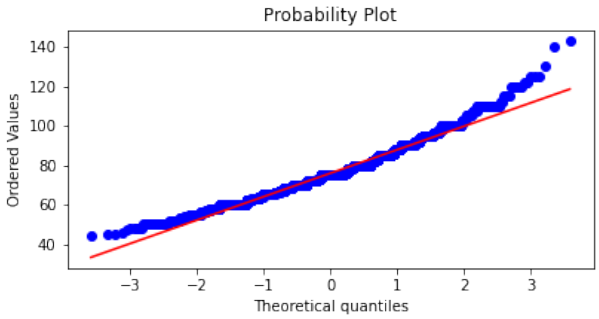
Out[34]: <AxesSubplot:>



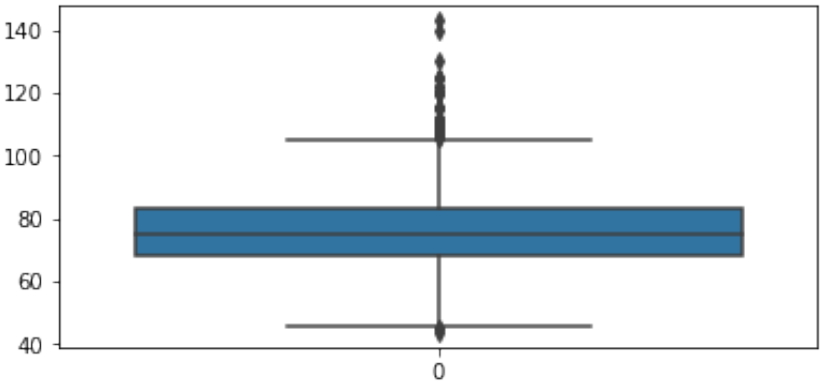
The histogram, probability plot, and boxplot all visually show the observations of heart rate to be sllightly skewed in distribution.The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [35]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Heart Rate")
sns.histplot(df2.heartRate, stat = "density")
sns.kdeplot(df2.heartRate, color = "red")
sns.rugplot(df2.heartRate, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.heartRate, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['heartRate'])
```



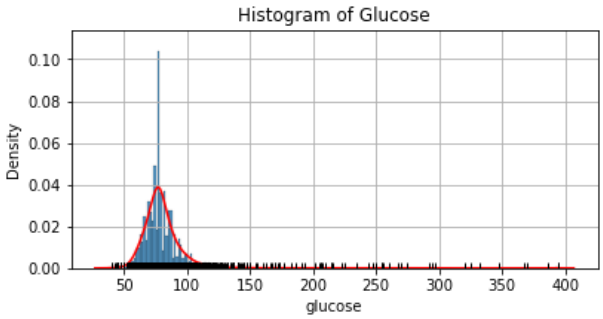


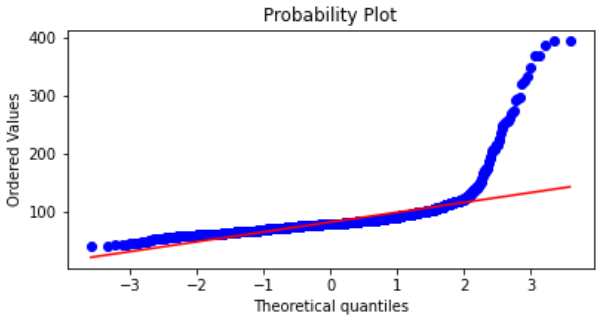
Out[35]: <AxesSubplot:>



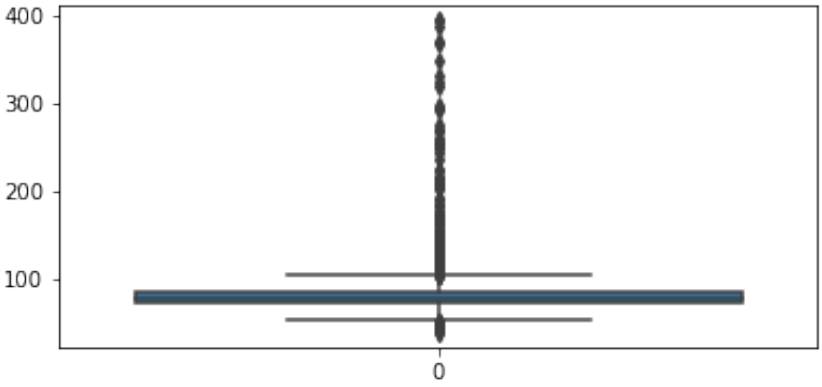
The histogram, probability plot, and boxplot all visually show the observations of glucose to not be normally distributed and is heavily skewed to the right. The red regression line in the probability plot shows what normally distributed observations would look like for comparison.

In [36]:
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Glucose")
sns.histplot(df2.glucose, stat = "density")
sns.kdeplot(df2.glucose, color = "red")
sns.rugplot(df2.glucose, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.glucose, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['glucose'])
```





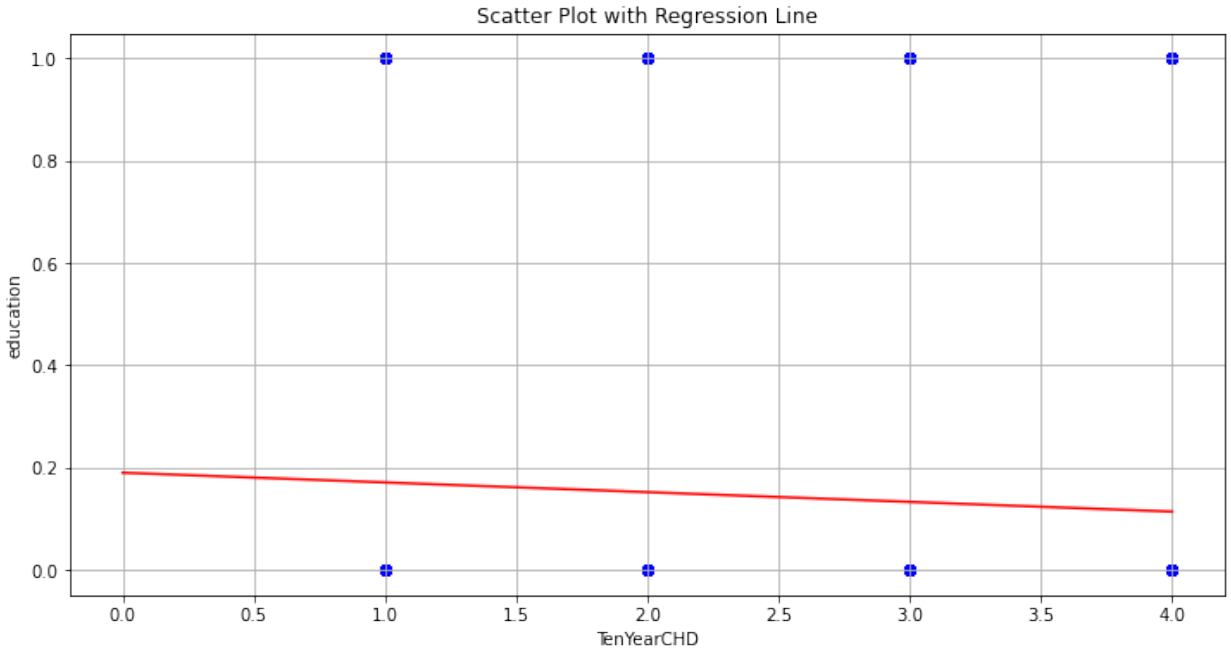
Out[36]: <AxesSubplot:>



Because the dependent variable (TenYearCHD) is binary, the observations in the scatter plots do not overlap. The red regression line shows the relationship between dependent variable and predictor and line of best fit.

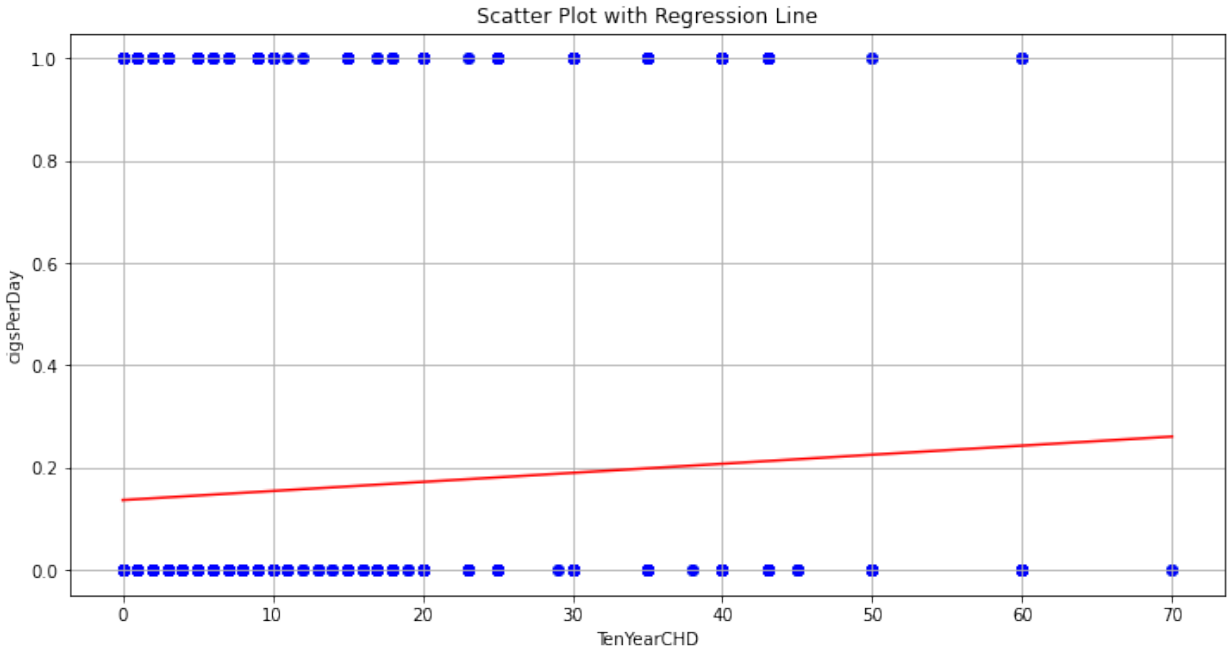
In [37]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["education"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["education"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equally spaced values
x_range = np.linspace(0, df2.education.max(), 1000)

# combining the two plots
plt.scatter(df2["education"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("education")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: -0.019030844803086838 The Interc
ept is: 0.1896294849110646
```



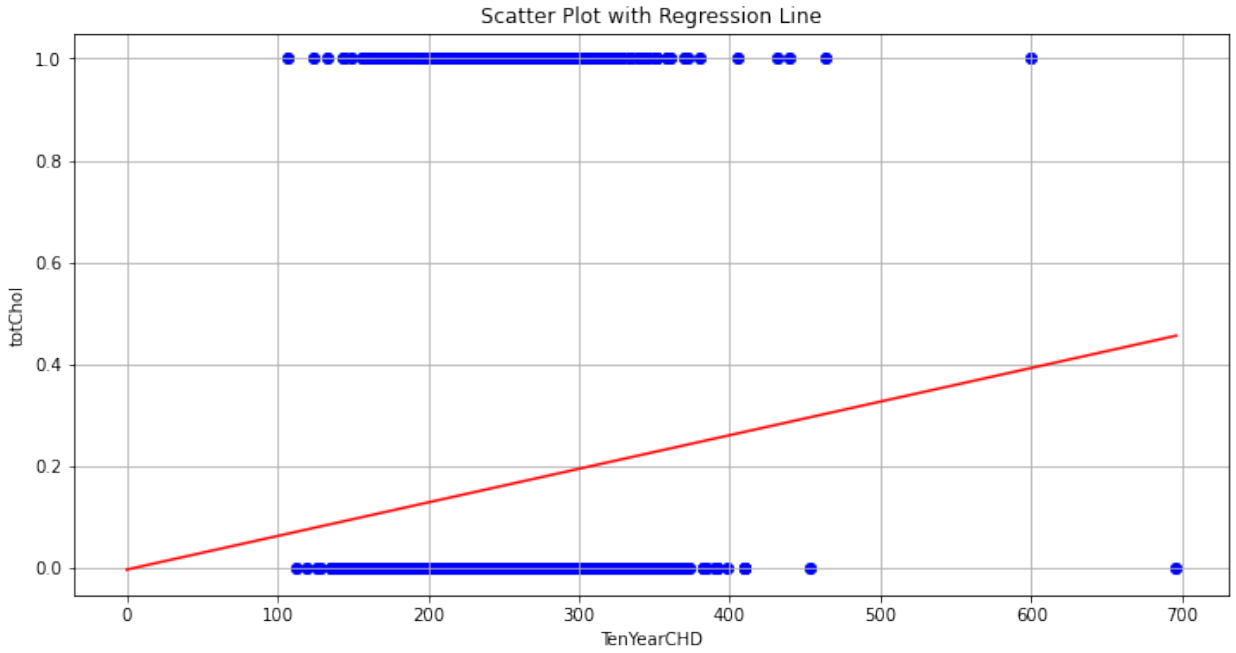
In [38]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["cigsPerDay"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["cigsPerDay"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.cigsPerDay.max(), 1000)

# combining the two plots
plt.scatter(df2["cigsPerDay"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("cigsPerDay")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.0017754223461071501 The Interc
ept is: 0.1360835643267134
```



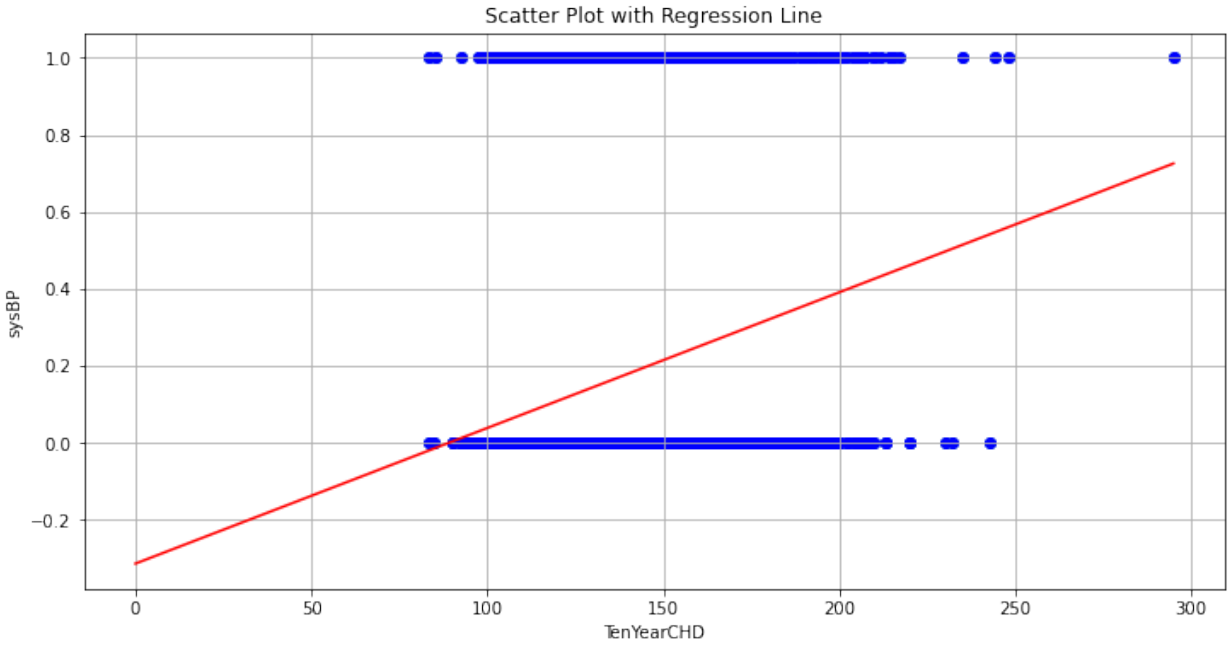
In [39]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["totChol"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["totChol"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.totChol.max(), 1000)

# combining the two plots
plt.scatter(df2["totChol"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("totChol")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.00066062862993396 The Intercep
t is: -0.004405373383779066
```



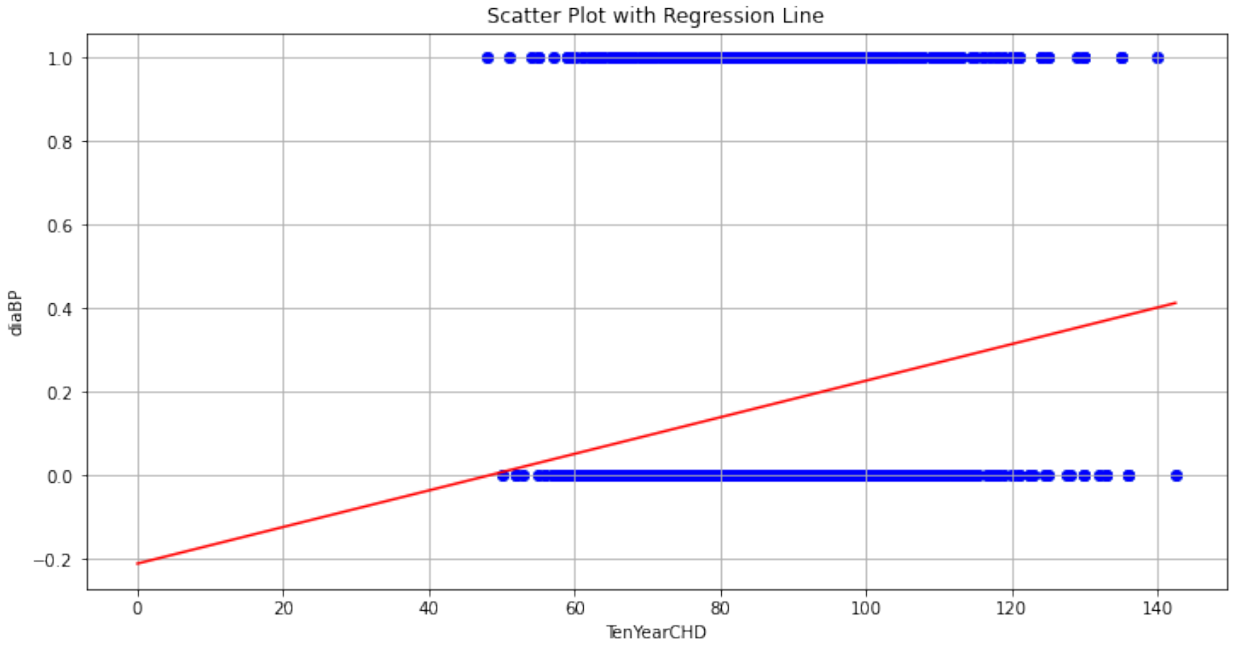
In [40]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["sysBP"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["sysBP"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.sysBP.max(), 1000)

# combining the two plots
plt.scatter(df2["sysBP"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("sysBP")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.0035258490097714794 The Interc
ept is: -0.3146961314644679
```



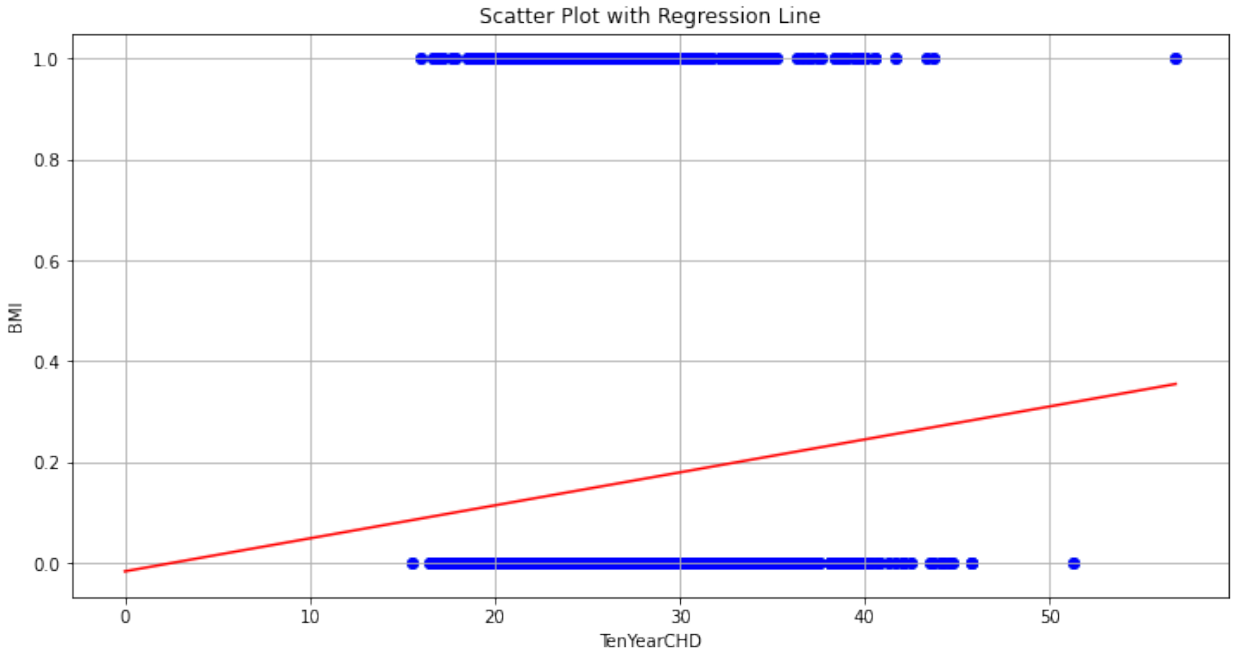
In [41]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["diaBP"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["diaBP"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.diaBP.max(), 1000)

# combining the two plots
plt.scatter(df2["diaBP"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("diaBP")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.004379680778598945 The Interce
pt is: -0.21108843952896542
```



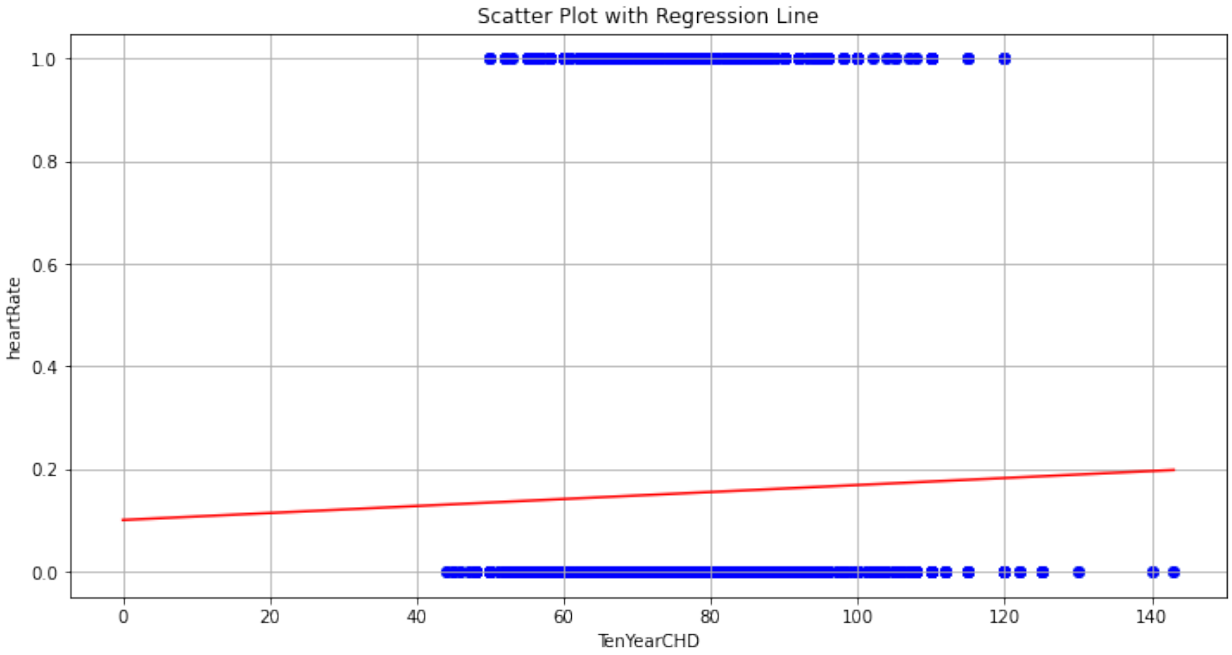
In [42]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["BMI"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["BMI"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.BMI.max(), 1000)

# combining the two plots
plt.scatter(df2["BMI"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("BMI")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.006545124857655006 The Interce
pt is: -0.016907093969931113
```



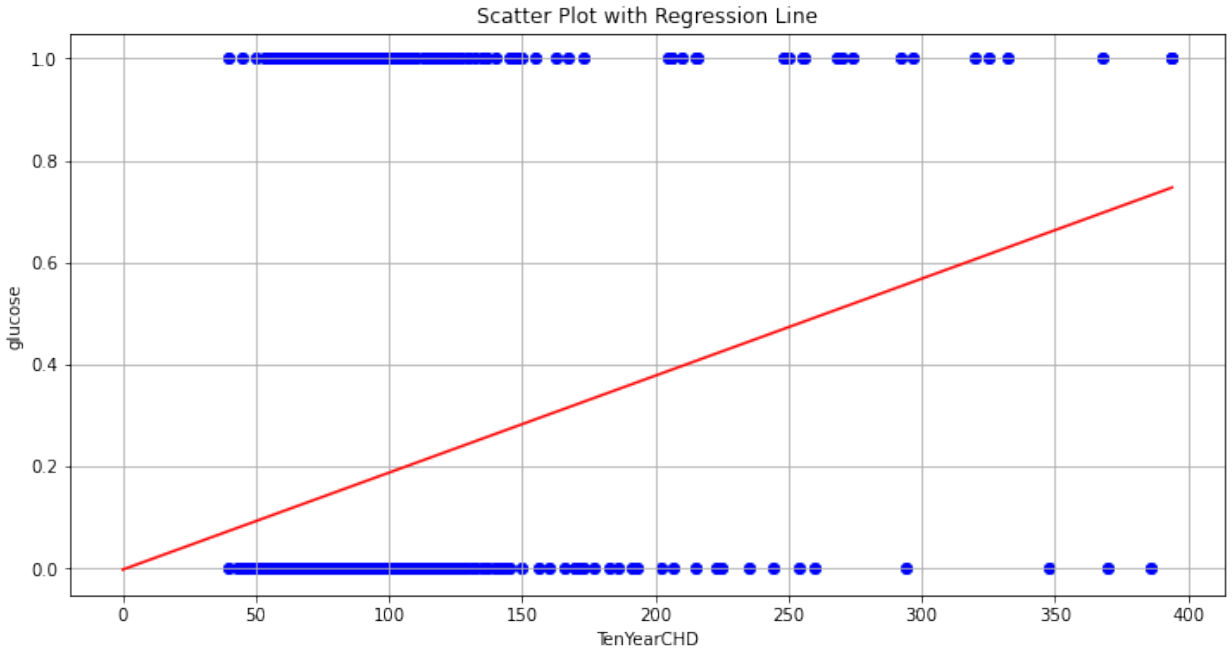
In [43]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["heartRate"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["heartRate"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.heartRate.max(), 1000)

# combining the two plots
plt.scatter(df2["heartRate"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("heartRate")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.0006824095727596032 The Interc
ept is: 0.10017810855342241
```



In [44]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["glucose"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["glucose"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:",

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.glucose.max(), 1000)

# combining the two plots
plt.scatter(df2["glucose"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("glucose")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.0019042600407882346 The Interc
ept is: -0.003435978165653731
```



In the initial model with all predictors, only male, age, cigsPerDay, prevelantStroke, sysBP, and glucose are statistically significant to predicting risk of heart disease. We need to check for multicollinearity which may bias the coefficients.

In [45]:
```
model = smf.ols(formula='TenYearCHD ~ male + age + education + current

result1 = model.fit()
```

```
result1 = model.fit()

print(result1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   R-squared:
0.097
Model:                            OLS   Adj. R-squared:
0.094
Method:                 Least Squares   F-statistic:
30.31
Date:                Fri, 02 Dec 2022   Prob (F-statistic):
9.97e-83
Time:                        22:17:23   Log-Likelihood:
-1454.9
No. Observations:                4238   AIC:
2942.
Df Residuals:                    4222   BIC:
3044.
Df Model:                          15
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.0
25      0.975]
------------------------------------------------------------------------------
Intercept         -0.5592      0.074     -7.557      0.000      -0.7
04      -0.414
male               0.0529      0.012      4.562      0.000       0.0
30       0.076
age                0.0069      0.001      9.642      0.000       0.0
06       0.008
education         -0.0018      0.005     -0.329      0.742      -0.0
12       0.009
currentSmoker     -0.0005      0.016     -0.028      0.977      -0.0
33       0.032
cigsPerDay         0.0027      0.001      3.783      0.000       0.0
01       0.004
BPMeds             0.0550      0.033      1.682      0.093      -0.0
09       0.119
prevalentStroke    0.1947      0.069      2.814      0.005       0.0
59       0.330
prevalentHyp       0.0288      0.016      1.770      0.077      -0.0
03       0.061
diabetes           0.0472      0.042      1.128      0.259      -0.0
35       0.129
totChol         8.908e-05      0.000      0.715      0.475      -0.0
00       0.000
```

```
sysBP              0.0023      0.000      4.926      0.000       0.0
01       0.003
diaBP             -0.0010      0.001     -1.270      0.204      -0.0
02       0.001
BMI               -0.0003      0.001     -0.194      0.846      -0.0
03       0.003
heartRate         -0.0002      0.000     -0.395      0.693      -0.0
01       0.001
glucose            0.0011      0.000      3.853      0.000       0.0
01       0.002
==============================================================================
Omnibus:                     1179.378   Durbin-Watson:
2.036
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
2451.322
Skew:                           1.690   Prob(JB):
0.00
Kurtosis:                       4.568   Cond. No.
4.47e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors i
s correctly specified.
[2] The condition number is large, 4.47e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

After running a VIF test and using a threshold of 4, there do not appear to be any variables that should be dropped.

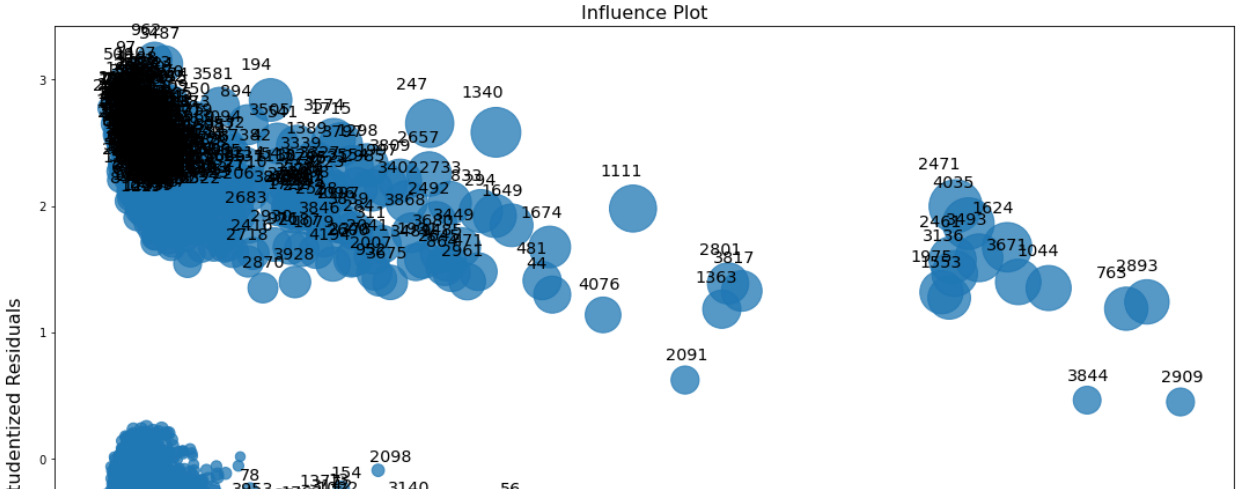
In [46]:
```
import statsmodels.stats.outliers_influence as smo
import patsy as pt

# extract matrices using patsy:
y, X = pt.dmatrices('TenYearCHD ~ male + age + cigsPerDay + prevalentS
                     data=df2, return_type='dataframe')

# get VIF:
K = X.shape[1]
VIF = np.empty(K)
for i in range(K):
    VIF[i] = smo.variance_inflation_factor(X.values, i)
print(f'VIF: \n{VIF}\n')
```

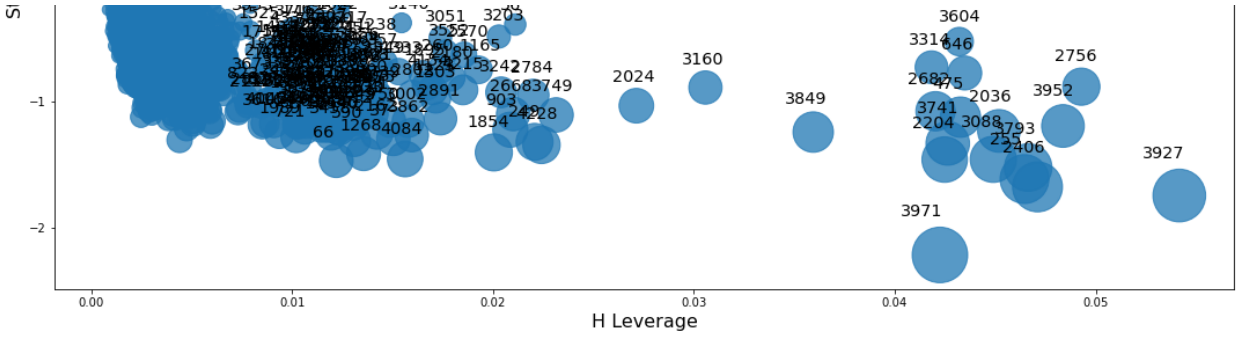
```
VIF:
[61.82979994  1.11346468  1.22819225  1.15596871  1.00526901  1.19791
733
  1.02581067]
```

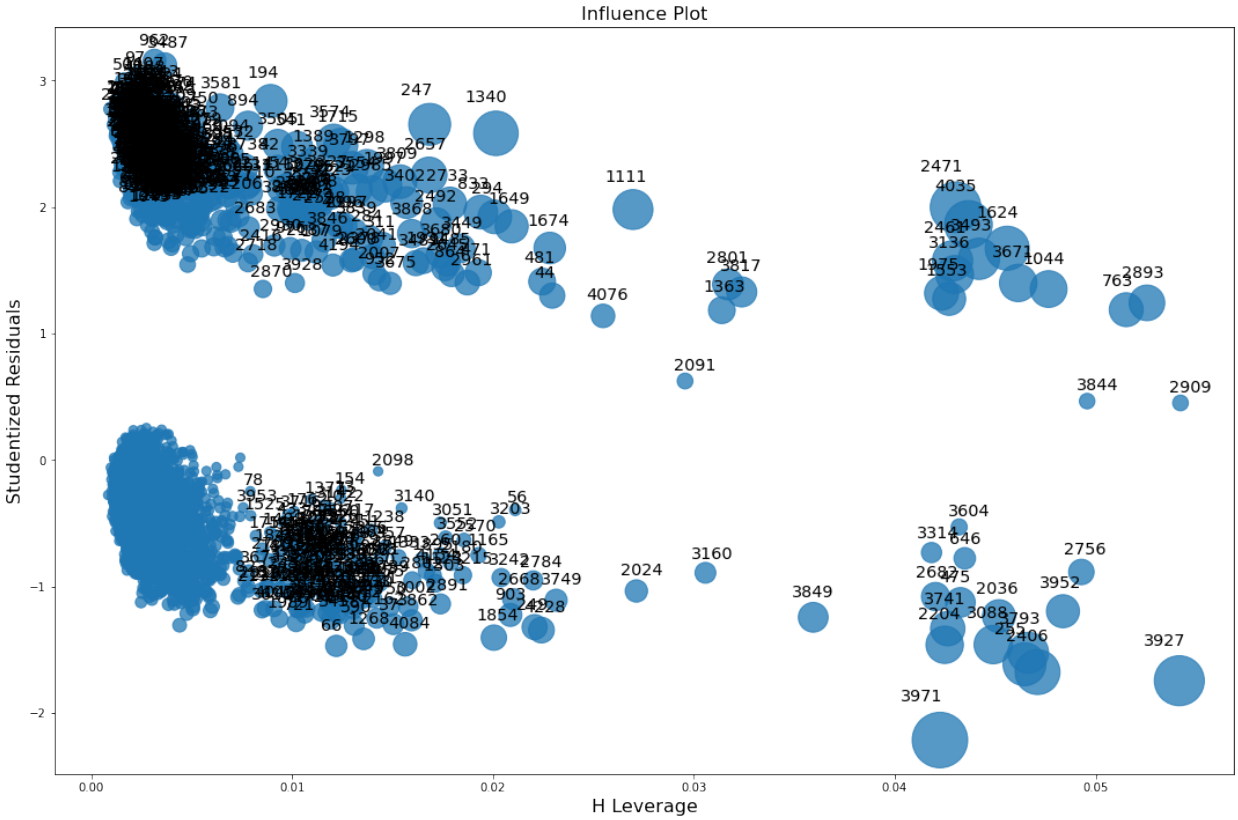
The influence plots shows observations that have high leverage using DFFITS and Cook's Distance methods. Observations further to the right are potentially influential and problematic to the model.

We can see that althought there are high leverage and outlier terms, we fail to find any influential observations which might be affecting our dataset.

In [47]:
```
figd, ax = plt.subplots(figsize=(15,10))
figd = sm.graphics.influence_plot(result1, ax = ax, criterion="DFFITS"
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(15,10))
fige = sm.graphics.influence_plot(result1, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```



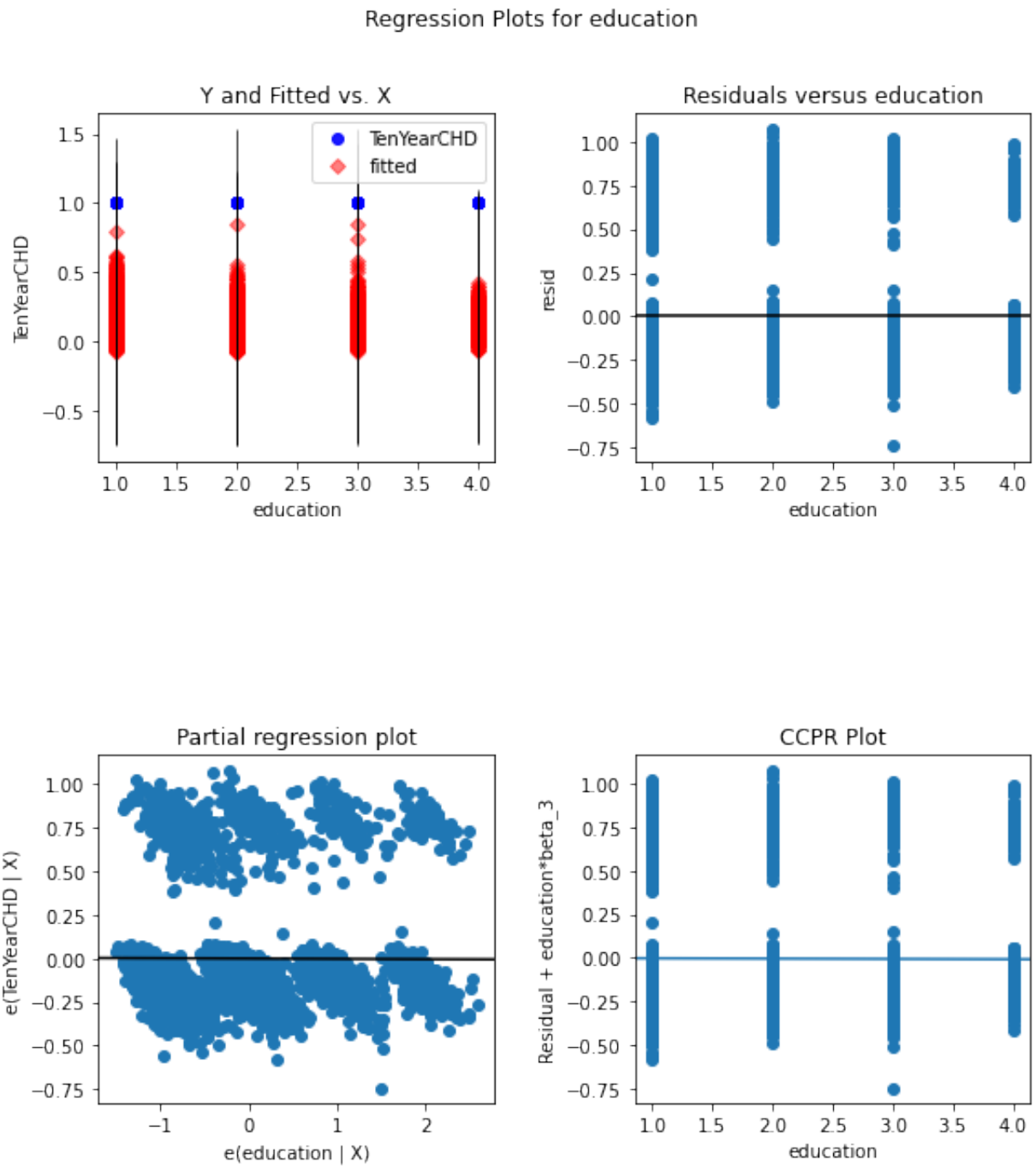




The plots below show the relationship between predicted y values, residuals, and a specific predictor. If the "residual versus predictor" line stays horizontal at zero, this visually supports not needing any additional variables in the model for better specificiation. The CCPR plot shows the relationship between the fitted predicted values of y against a specifc predictor to show their relationship. The partial regression plot shows the regression line if that predictor was the only predictor in the regression (showing a simple linear regression as opposed to the multiple regression with all the predictors).

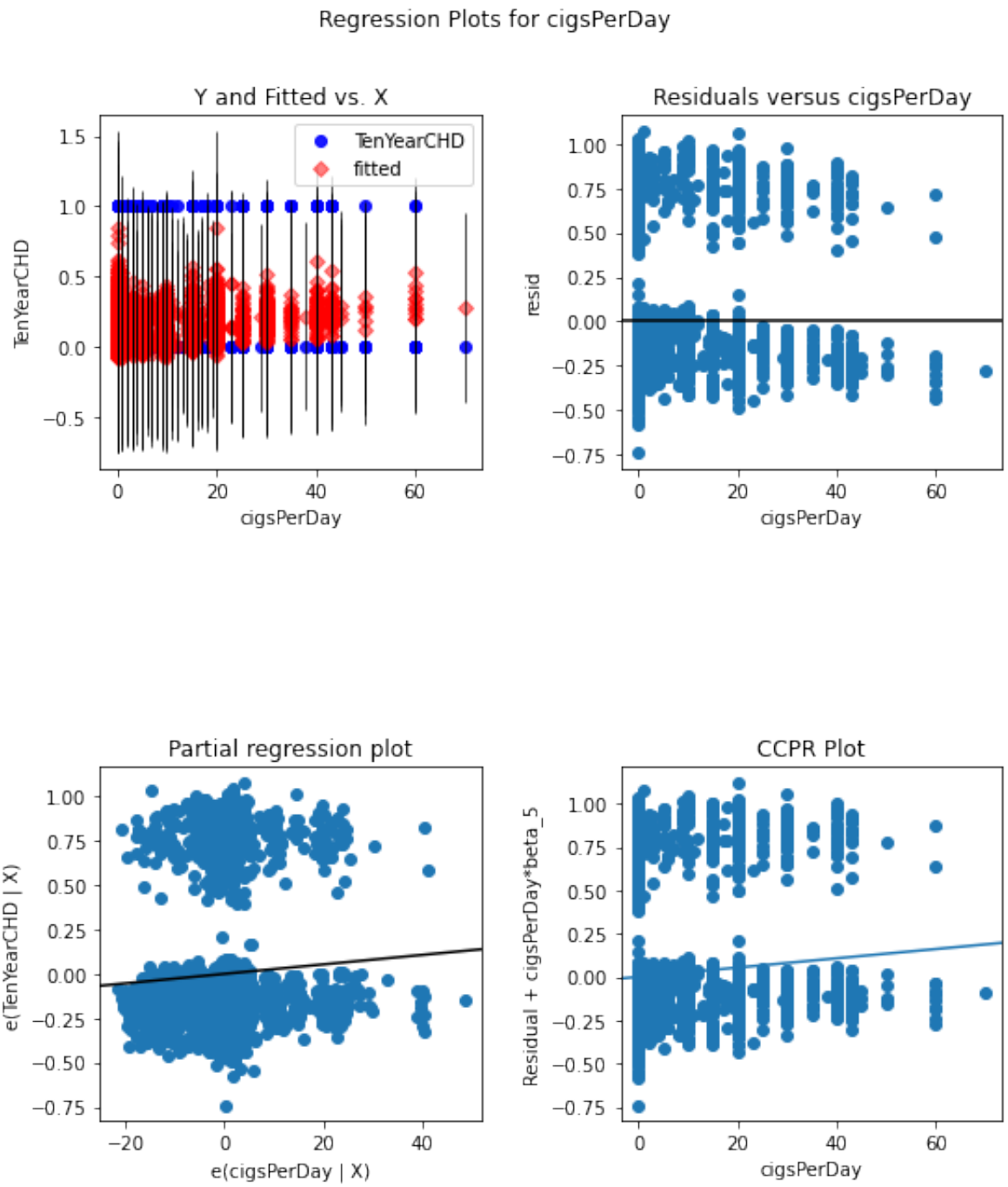
In [48]:
```
fig = sm.graphics.plot_regress_exog(result1, "education")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



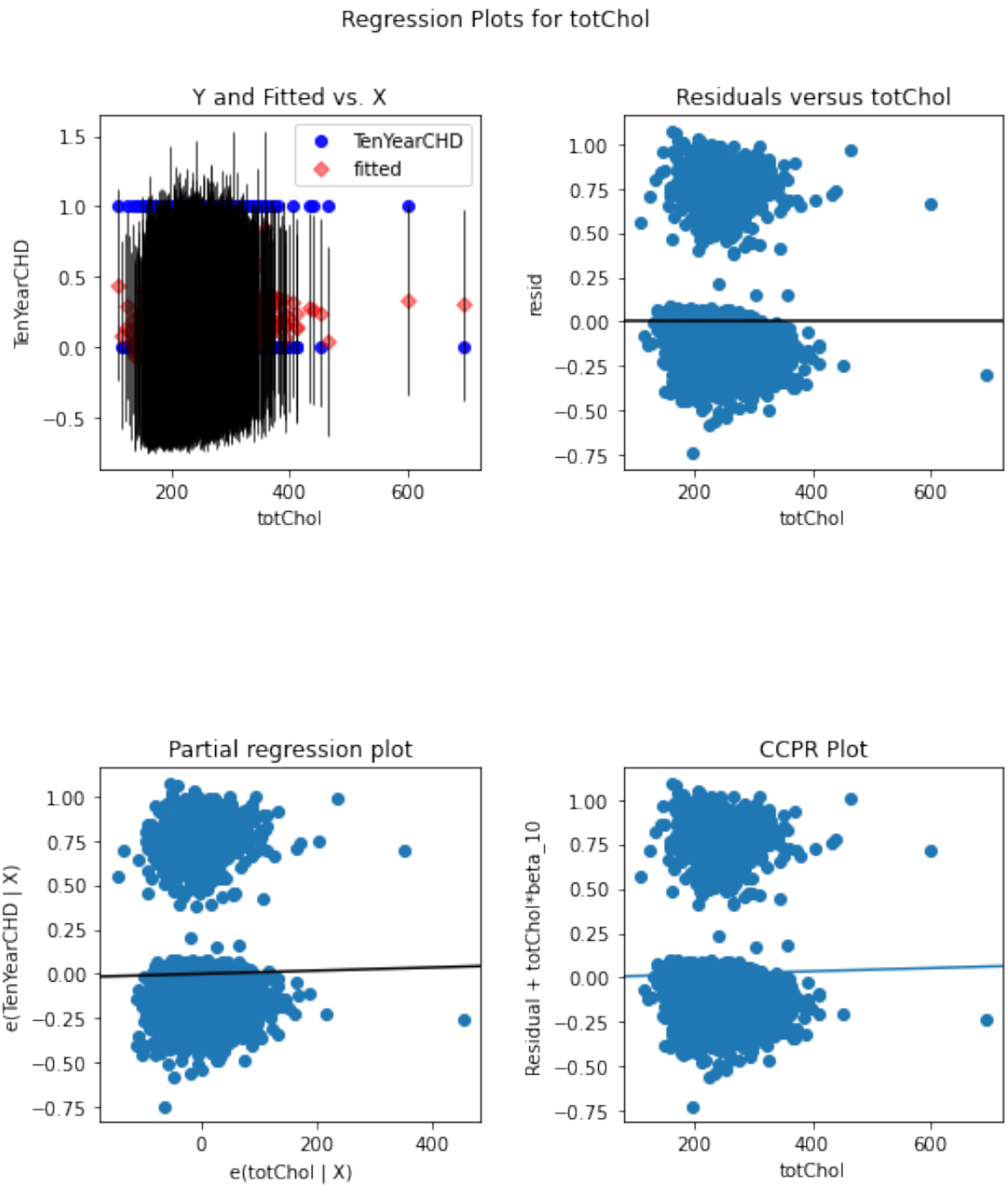
In [49]:
```
fig = sm.graphics.plot_regress_exog(result1, "cigsPerDay")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



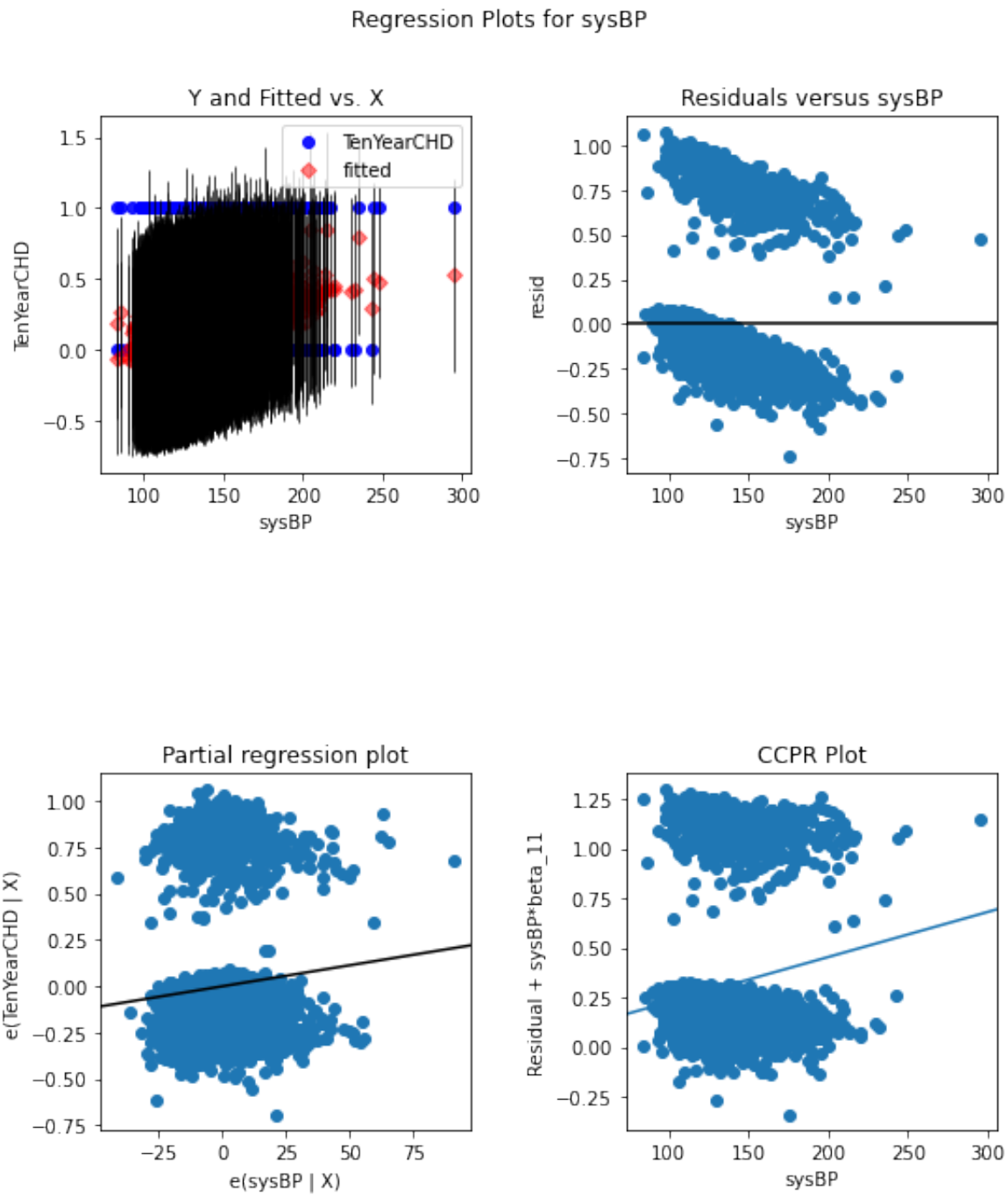
In [50]:
```
fig = sm.graphics.plot_regress_exog(result1, "totChol")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



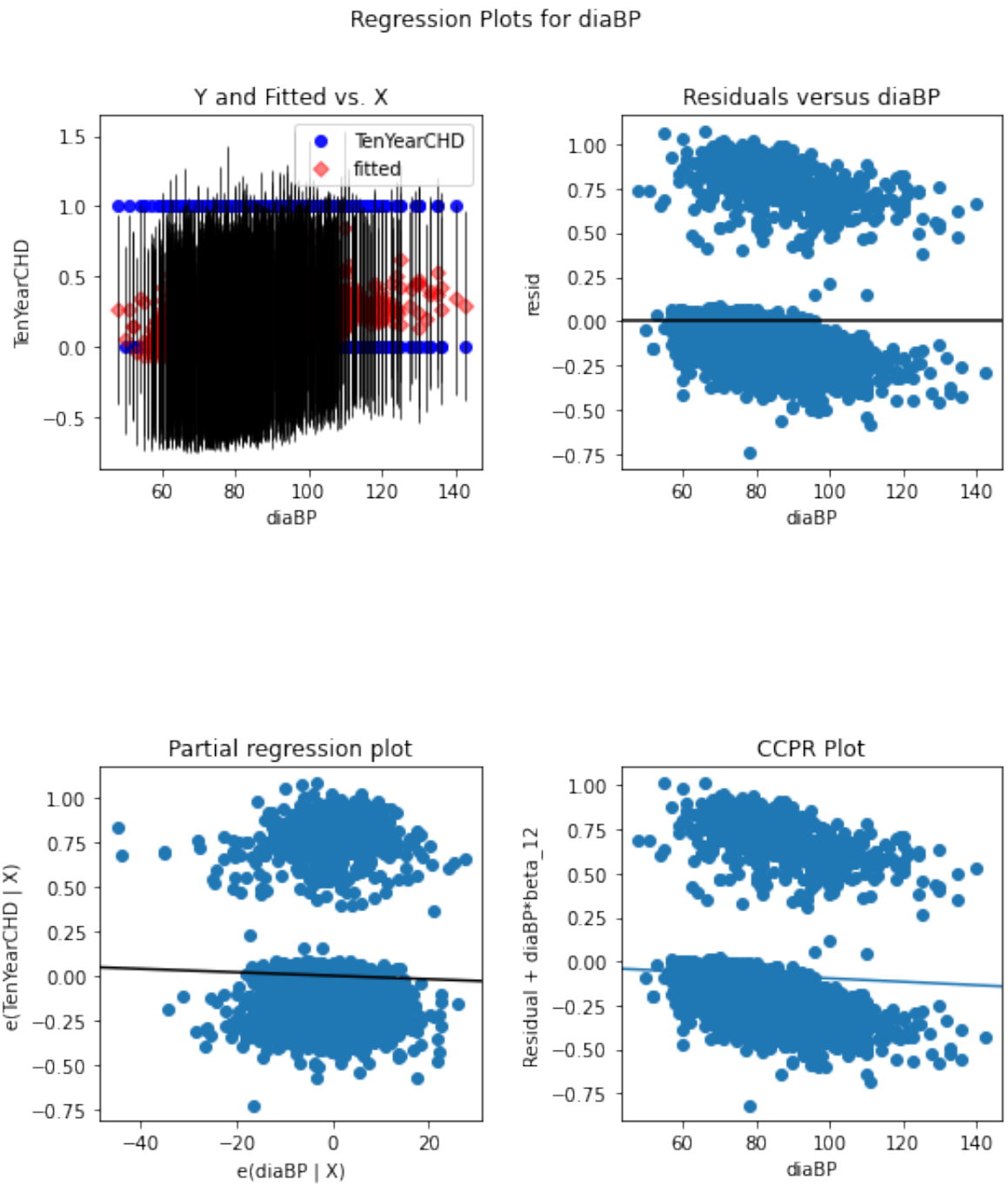
In [51]:
```
fig = sm.graphics.plot_regress_exog(result1, "sysBP")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



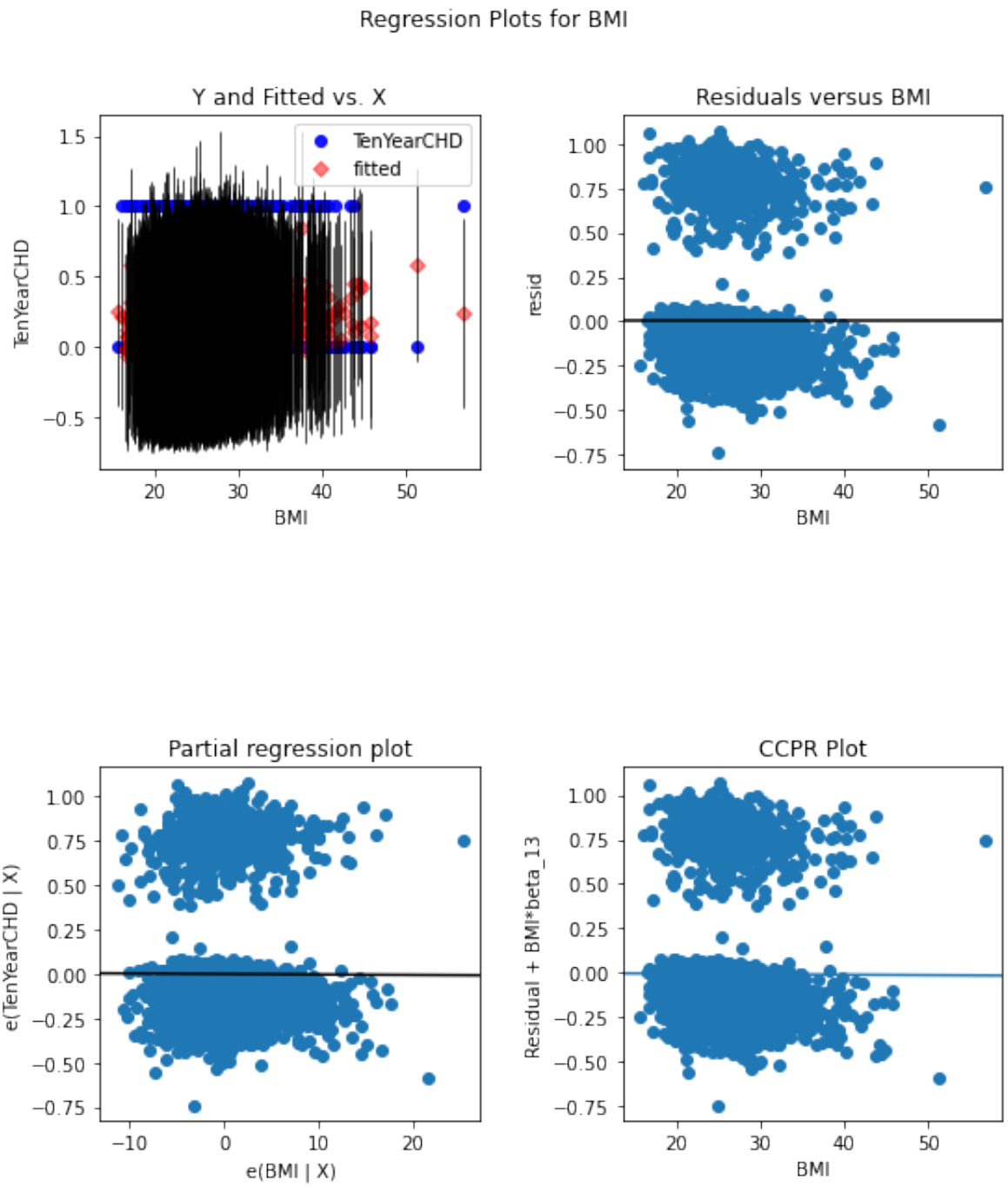
In [52]:
```
fig = sm.graphics.plot_regress_exog(result1, "diaBP")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



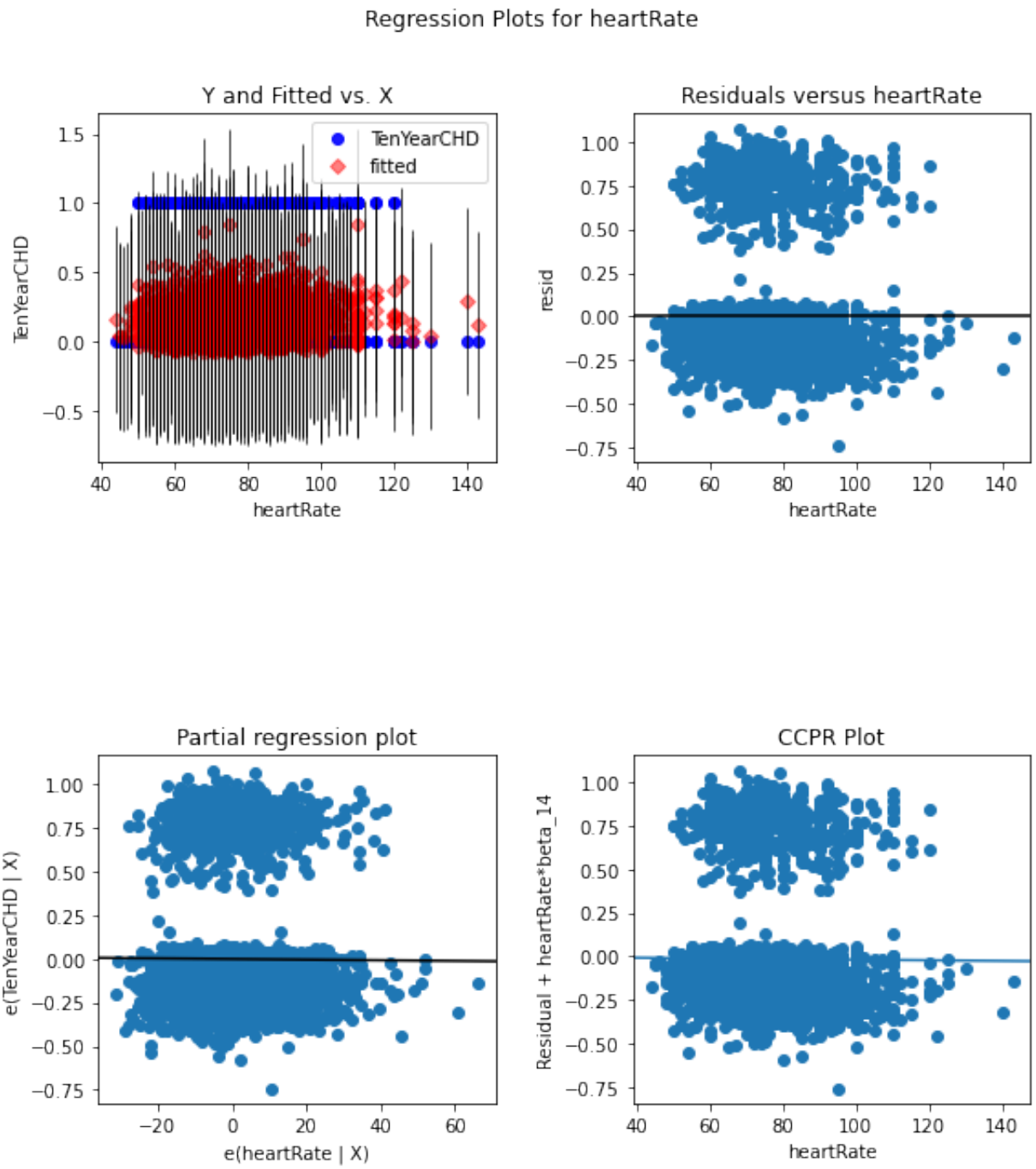
In [53]:
```
fig = sm.graphics.plot_regress_exog(result1, "BMI")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



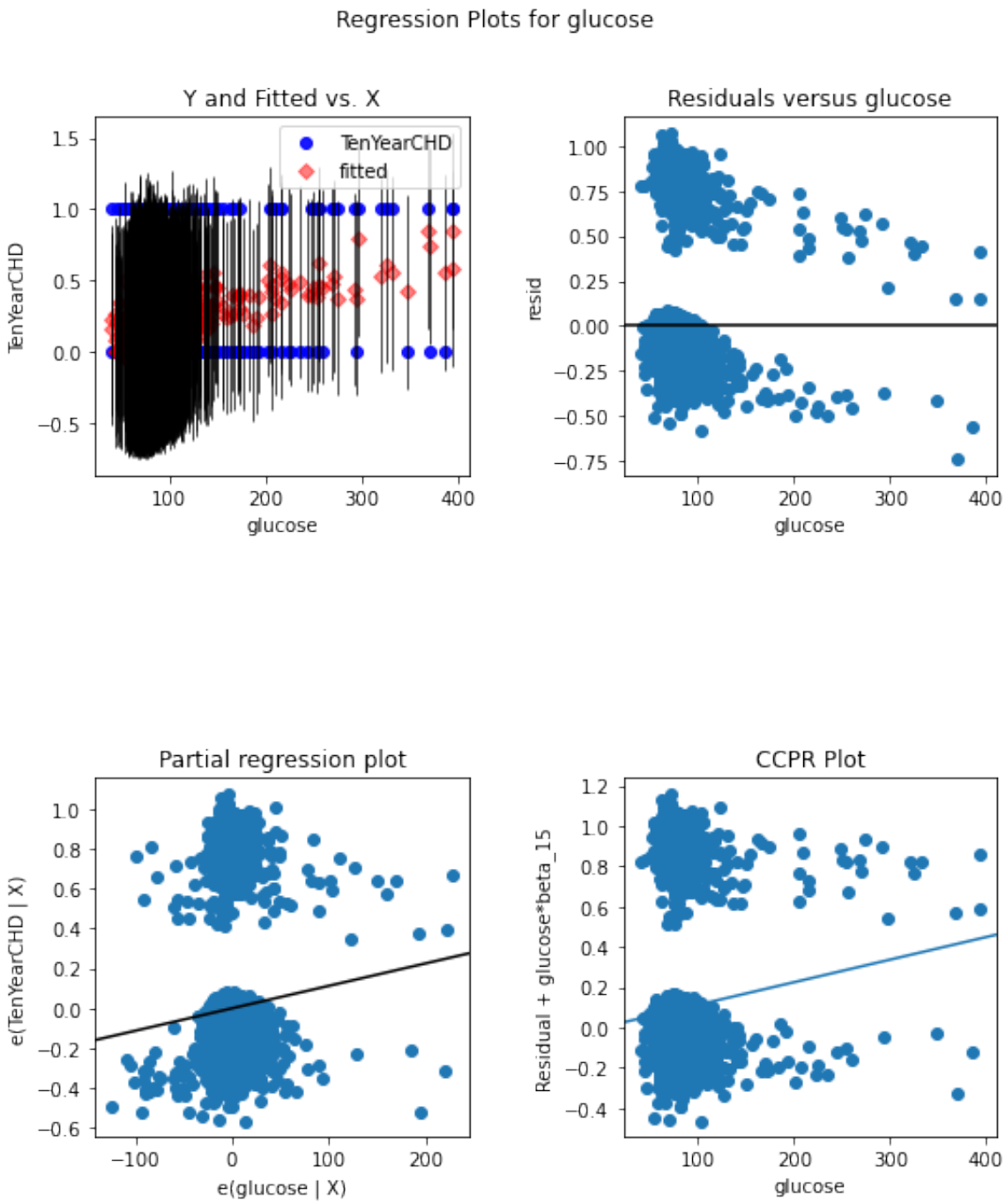
In [54]:
```
fig = sm.graphics.plot_regress_exog(result1, "heartRate")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



In [55]:
```
fig = sm.graphics.plot_regress_exog(result1, "glucose")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



Part 3 Identifying which model is your preferred one

Logit Model

Because the model has a binary dependent variable, logit may work better than probit because it is not based on a normal distribution of observations. The logit regression model returns a psuedo r squared of 11.15% and tests male, age, education, cigsPerDay, prevelantStroke, sysBP, and glucose to be significant.

In [56]:
```
import wooldridge as woo
import statsmodels.formula.api as smf


# Estimate a logit model:

reg_logit = smf.logit(formula='TenYearCHD ~ male + age + education + cu


results_logit = reg_logit.fit(disp=0)
print(f'results_logit.summary(): \n{results_logit.summary()}\n')


print(f'results_logit.llf: {results_logit.llf}\n')


print(f'results_logit.prsquared: {results_logit.prsquared}\n')
```

```
results_logit.summary():
                           Logit Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   No. Observations:
4238
Model:                          Logit   Df Residuals:
4222
Method:                           MLE   Df Model:
15
Date:                Fri, 02 Dec 2022   Pseudo R-squ.:
0.1115
Time:                        22:17:52   Log-Likelihood:
-1604.4
converged:                       True   LL-Null:
-1805.8
Covariance Type:            nonrobust   LLR p-value:
1.834e-76
==============================================================================
==============
                 coef    std err          z      P>|z|      [0.0
25      0.975]
```

```
------------------------------------------------------------------------
--------------
Intercept          -8.1149      0.665    -12.201      0.000      -9.4
18      -6.811
male                0.5030      0.100      5.011      0.000       0.3
06       0.700
age                 0.0621      0.006      9.992      0.000       0.0
50       0.074
education          -0.0131      0.046     -0.284      0.777      -0.1
04       0.078
currentSmoker       0.0133      0.143      0.093      0.926      -0.2
67       0.293
cigsPerDay          0.0214      0.006      3.793      0.000       0.0
10       0.032
BPMeds              0.2435      0.220      1.105      0.269      -0.1
88       0.675
prevalentStroke     0.9611      0.442      2.176      0.030       0.0
96       1.827
prevalentHyp        0.2307      0.128      1.796      0.073      -0.0
21       0.483
diabetes            0.1880      0.294      0.639      0.523      -0.3
89       0.765
totChol             0.0018      0.001      1.780      0.075      -0.0
00       0.004
sysBP               0.0141      0.004      3.983      0.000       0.0
07       0.021
diaBP              -0.0028      0.006     -0.474      0.636      -0.0
15       0.009
BMI                 0.0031      0.012      0.263      0.793      -0.0
20       0.026
heartRate          -0.0015      0.004     -0.384      0.701      -0.0
09       0.006
glucose             0.0067      0.002      3.134      0.002       0.0
03       0.011
========================================================================
==============

results_logit.llf: -1604.4031293469973

results_logit.prsquared: 0.1115153761286799
```

Linear Probability Model

The linear probability model returns an r squared of 9.72% and tests male, age, education, cigsPerDay, prevelantStroke, sysBP, and glucose to be significant as well.

In [57]:
```
import wooldridge as woo
import pandas as pd
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt


# Estimate a linear probability model:
reg_lin = smf.ols(formula='TenYearCHD ~ male + age + education + curre
                  data=df2)
results_lin = reg_lin.fit(cov_type='HC3')

# Print regression table:
table = pd.DataFrame({'b': round(results_lin.params, 4),
                      'se': round(results_lin.bse, 4),
                      't': round(results_lin.tvalues, 4),
                      'pval': round(results_lin.pvalues, 4)})
print(f'table: \n{table}\n')



X_new = pd.DataFrame(
    {'male': [1, 0], 'education': [1, 4],
     'currentSmoker': [0, 1], 'cigsPerDay': [0, 70],
     'BPMeds': [0, 1], 'prevalentStroke': [0, 1], 'prevalentHyp': [0,1
    'sysBP': [83.5,295], 'diaBP': [48,142.5], 'BMI': [15.54,56.8], 'he
predictions = results_lin.predict(X_new)

print(f'predictions: \n{predictions}\n')
print(results_lin.rsquared)
```

```
table:
                    b      se       t    pval
Intercept       -0.5592  0.0785 -7.1242  0.0000
male             0.0529  0.0118  4.4742  0.0000
age              0.0069  0.0007  9.3123  0.0000
education       -0.0018  0.0054 -0.3247  0.7454
currentSmoker   -0.0005  0.0161 -0.0289  0.9769
cigsPerDay       0.0027  0.0007  3.5956  0.0003
BPMeds           0.0550  0.0436  1.2606  0.2075
prevalentStroke  0.1947  0.0973  2.0016  0.0453
prevalentHyp     0.0288  0.0181  1.5890  0.1121
diabetes         0.0472  0.0518  0.9109  0.3623
totChol          0.0001  0.0001  0.6430  0.5202
sysBP            0.0023  0.0005  4.2675  0.0000
diaBP           -0.0010  0.0009 -1.1079  0.2679
BMI             -0.0003  0.0016 -0.1723  0.8632
heartRate       -0.0002  0.0005 -0.3828  0.7018
glucose          0.0011  0.0004  3.1355  0.0017

predictions:
```

```
0   -0.101301
1    1.425534
dtype: float64

0.09723007083551449
```

Probit Model

The probit regression model returns a psuedo r squared of 11.15% and tests male, age, education, cigsPerDay, prevelantStroke, sysBP, and glucose to be significant as well.

In [58]:
```
import wooldridge as woo
import statsmodels.formula.api as smf


# Estimate a probit model:
reg_probit = smf.probit(formula='TenYearCHD ~ male + age + education +
                        data=df2)
results_probit = reg_probit.fit(disp=0)
print(f'results_probit.summary(): \n{results_probit.summary()}\n')

# log likelihood value:
print(f'results_probit.llf: {results_probit.llf}\n')

# McFadden's pseudo R2:
print(f'results_probit.prsquared: {results_probit.prsquared}\n')
```

```
results_probit.summary():
                          Probit Regression Results
==============================================================================
==========
Dep. Variable:             TenYearCHD   No. Observations:
4238
Model:                         Probit   Df Residuals:
4222
Method:                           MLE   Df Model:
15
Date:                Fri, 02 Dec 2022   Pseudo R-squ.:
0.1115
Time:                        22:17:52   Log-Likelihood:
-1604.5
converged:                       True   LL-Null:
-1805.8
Covariance Type:            nonrobust   LLR p-value:
1.919e-76
==============================================================================
```

```
==============
                     coef    std err          z      P>|z|      [0.0
25      0.975]
------------------------------------------------------------------------
--------------
Intercept          -4.4907      0.359    -12.524      0.000      -5.1
93      -3.788
male                0.2617      0.055      4.786      0.000       0.1
55       0.369
age                 0.0342      0.003     10.020      0.000       0.0
28       0.041
education          -0.0114      0.025     -0.450      0.652      -0.0
61       0.038
currentSmoker       0.0201      0.079      0.255      0.798      -0.1
34       0.174
cigsPerDay          0.0118      0.003      3.683      0.000       0.0
06       0.018
BPMeds              0.1620      0.129      1.251      0.211      -0.0
92       0.416
prevalentStroke     0.5671      0.269      2.109      0.035       0.0
40       1.094
prevalentHyp        0.1266      0.072      1.761      0.078      -0.0
14       0.267
diabetes            0.1450      0.172      0.844      0.398      -0.1
92       0.482
totChol             0.0009      0.001      1.647      0.100      -0.0
00       0.002
sysBP               0.0078      0.002      3.874      0.000       0.0
04       0.012
diaBP              -0.0016      0.003     -0.464      0.643      -0.0
08       0.005
BMI                 0.0014      0.006      0.222      0.824      -0.0
11       0.014
heartRate          -0.0008      0.002     -0.395      0.692      -0.0
05       0.003
glucose             0.0036      0.001      2.951      0.003       0.0
01       0.006
==========================================================================
==============

results_probit.llf: -1604.4501143428351

results_probit.prsquared: 0.1114893568402544
```

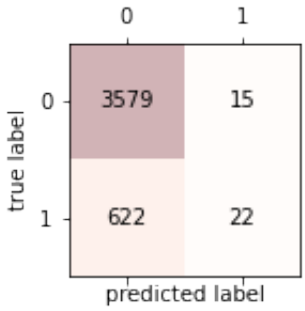
Estimating the Logit Model and confusion matrix

In [59]:
```
sklearn.linear_model import LogisticRegression
sklearn.metrics import accuracy_score
```

```
sklearn.metrics import accuracy_score
sklearn.metrics import confusion_matrix, classification_report, precis
t matplotlib.pyplot as plt
s = ['male', 'age' , 'education' , 'currentSmoker' , 'cigsPerDay' , 'Bl
LogisticRegression()
_mod = lr.fit(df2[X_cols], df2['TenYearCHD'])
mat = confusion_matrix(df2['TenYearCHD'], lr.predict(df2[X_cols]))
(conf_mat)
('Accuracy =', lr.score(df2[X_cols],df2['TenYearCHD']))


ax = plt.subplots(figsize=(2, 2))
tshow(conf_mat, cmap=plt.cm.Reds, alpha=0.3)
 in range(conf_mat.shape[0]):
or j in range(conf_mat.shape[1]):
   ax.text(x=j, y=i,
   s=conf_mat[i, j],
   va='center', ha='center')
label('predicted label')
label('true label')
how()

can print other metrics
(classification_report(df2['TenYearCHD'], lr.predict(df2[X_cols]), dig



 confusion_matrix(df2['TenYearCHD'], lr.predict(df2[X_cols]))
1=sum(sum(cm1))
acy = (cm1[0,0]+cm1[1,1])/total1
ficity = cm1[0,0]/(cm1[0,0]+cm1[0,1])
tivity = cm1[1,1]/(cm1[1,0]+cm1[1,1])
('Accuracy =', Accuracy)
('Specificity = ', Specificity)
('Sensitivity = ', Sensitivity)
```

```
[[3579   15]
 [ 622   22]]
Accuracy = 0.8496932515337423

/Users/dhritisahoo/opt/anaconda3/lib/python3.9/site-packages/sklearn/
linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to co
nverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sho
wn in:
    https://scikit-learn.org/stable/modules/preprocessing.html
(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#logisti
c-regression (https://scikit-learn.org/stable/modules/linear_model.ht
ml#logistic-regression)
  n_iter_i = _check_optimize_result(
```




```
              precision    recall  f1-score   support

           0      0.852     0.996     0.918      3594
           1      0.595     0.034     0.065       644

    accuracy                          0.850      4238
   macro avg      0.723     0.515     0.491      4238
weighted avg      0.813     0.850     0.789      4238

Accuracy = 0.8496932515337423
Specificity =  0.9958263772954925
Sensitivity =  0.034161490683229816
```

The logit model returns:

84.97% accuracy (ability of model to do binary classification correctly)

99.58% specificity (how accurately the model gives me true negatives)

3.42% sesitivity (how accurately the model gives me true positives)

From the confusion matrix we see that 3579 predictors were correctly predicted as 0 and 22 predictors were correctly predicted as 1. However, 15 predicted values were misclassified as 1 and 662 as 0.

In [60]:
```
def prs_result(model_name, model, df2):
    pred_values = model.predict(df2)
    pred_values = np.where(pred_values > 0.5, 1, 0)
    actual_values = df2['TenYearCHD']

    ACC = accuracy_score(actual_values, pred_values)
    ER = 1 - ACC
    SENS = recall_score(actual_values, pred_values)
    SPEC = recall_score(actual_values, pred_values, pos_label = 0)
    PPV = precision_score(actual_values, pred_values)
    NPV = precision_score(actual_values, pred_values, pos_label = 0)

    return pd.DataFrame([{'model': model_name, 'ACC': ACC, 'ER': ER, '
```

In [61]:
```
from sklearn.metrics import recall_score
```

Estimating the Probit Model and confusion matrix

In [62]:
```
def conf_matrix(model, df2):
    pred_values = model.predict(df2)
    pred_values = np.where(pred_values > 0.5, 1, 0)
    actual_values = df2['TenYearCHD']
    return pd.crosstab(actual_values, pred_values, rownames=['Actual']
```

In [63]:
```
print("Confusion matrix for probit model\n\n{0}".format(conf_matrix(re
```

```
Confusion matrix for probit model

Predicted     0   1
Actual
0          3577  17
1           595  49
```

In [64]:
```
precision_models_score_probit = prs_result('Probit Model', results_prc
precision_models_score_probit
```

Out[64]:

	model	ACC	ER	SENS	SPEC	PPV	NPV
0	Probit Model	0.85559	0.14441	0.07609	0.99527	0.74242	0.85738

The probit model returns:

85.56% accuracy (ability of model to do binary classification correctly)

99.53% specificity (how accurately the model gives me true negatives)

7.61% sesitivity (how accurately the model gives me true positives)

Estimating the Linear Probability Model and confusion matrix

In [65]:
```
print("Confusion matrix for linear probability model\n\n{0}".format(cc
```

```
Confusion matrix for linear probability model

Predicted     0   1
Actual
0          3589   5
1           621  23
```

In [66]:
```
precision_models_score_linear = prs_result('Linear Model', results_lir
precision_models_score_linear
```

Out[66]:

	model	ACC	ER	SENS	SPEC	PPV	NPV
0	Linear Model	0.85229	0.14771	0.03571	0.99861	0.82143	0.85249

The logit model returns:

85.23% accuracy (ability of model to do binary classification correctly)

99.86% specificity (how accurately the model gives me true negatives)

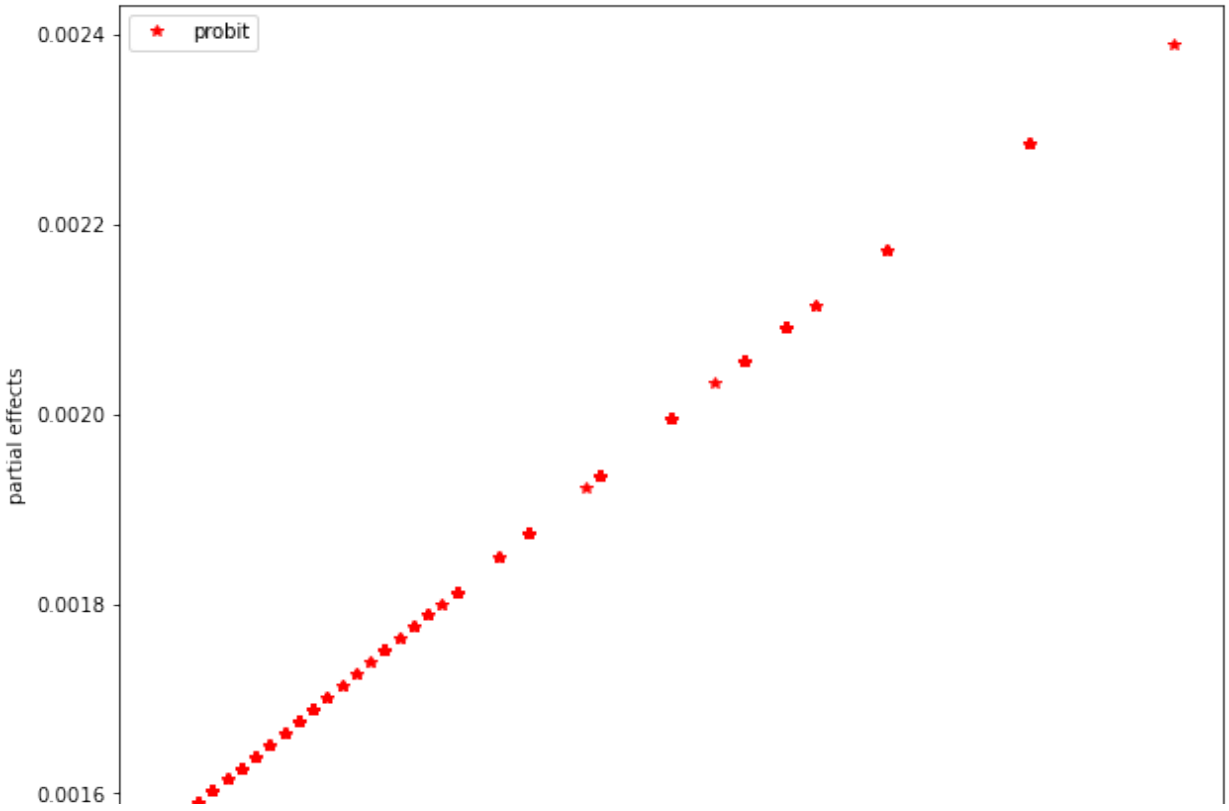
3.57% sesitivity (how accurately the model gives me true positives)

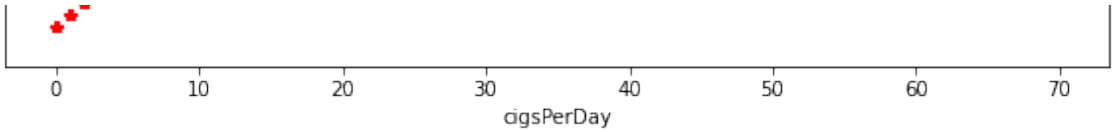
Looking at the estimation results above, we conclude that Probit model is preferres as it has the highest accuracy of 85.56%. In other words, Probit most correctly estimates our model and hence will be preferred.

Marginal Effects

In [67]:
```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
import scipy.stats as stats

#estimation
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ cigsPerDay', data = d
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['cigsPerDay'] * factor_probit


# plot APE's:
x = df2['cigsPerDay']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')
plt.ylabel('partial effects')
plt.xlabel('cigsPerDay')
plt.legend()
```

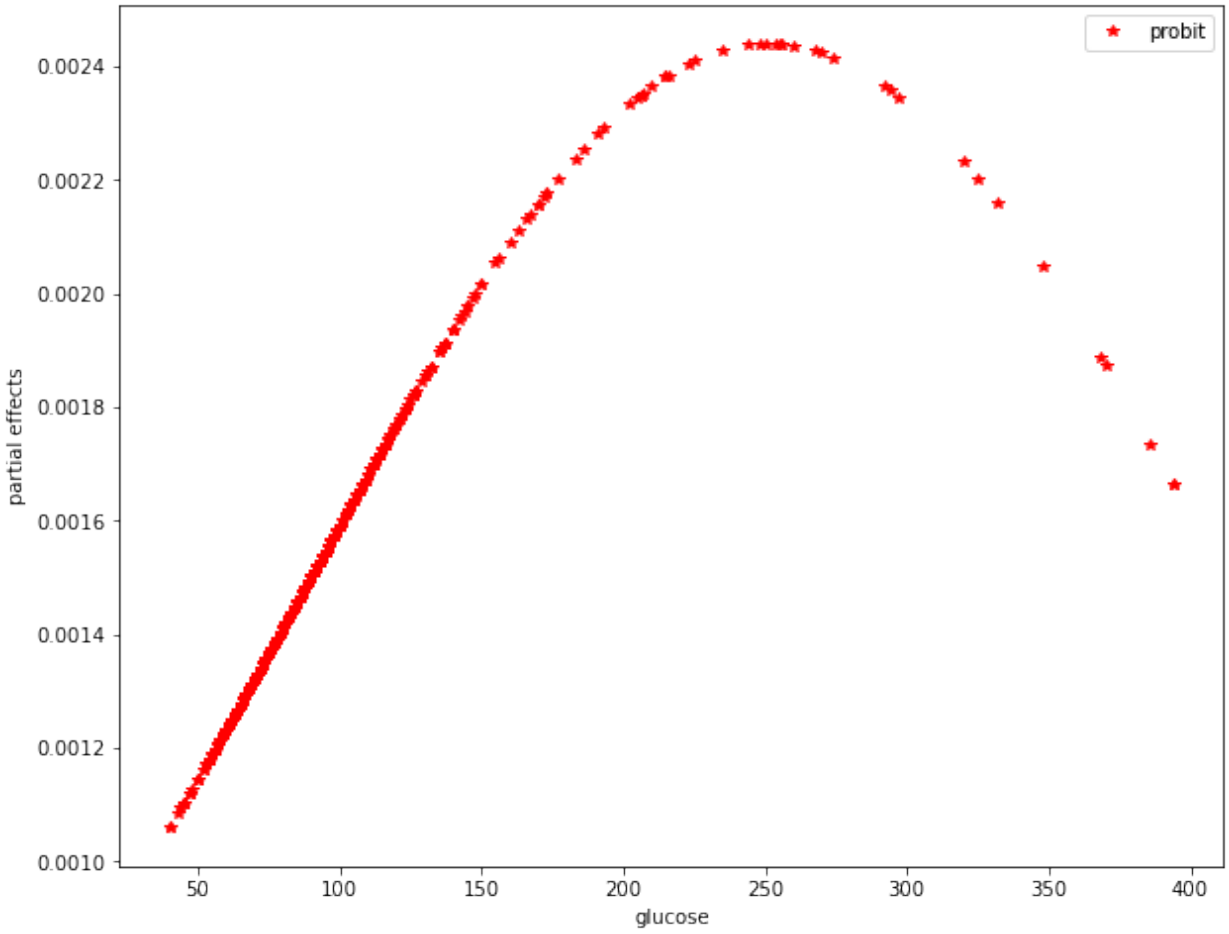
Out[67]: <matplotlib.legend.Legend at 0x7fe399db2d60>




0
10
20
30
40
50
60
70
cigsPerDay

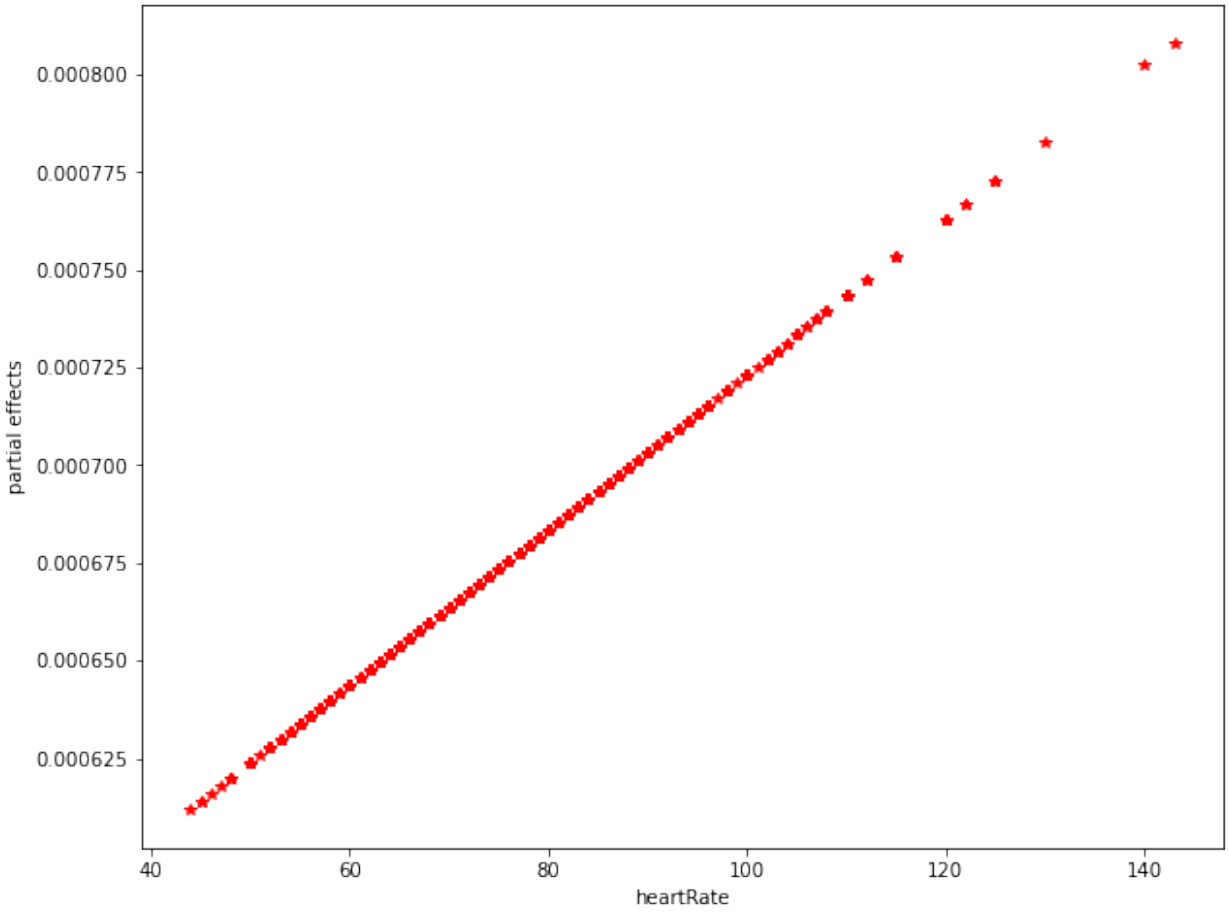
In [68]:
```
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ glucose', data = df2)
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['glucose'] * factor_probit


# plot APE's:
x = df2['glucose']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')
plt.ylabel('partial effects')
plt.xlabel('glucose')
plt.legend()
```

Out[68]: <matplotlib.legend.Legend at 0x7fe3aeb001f0>



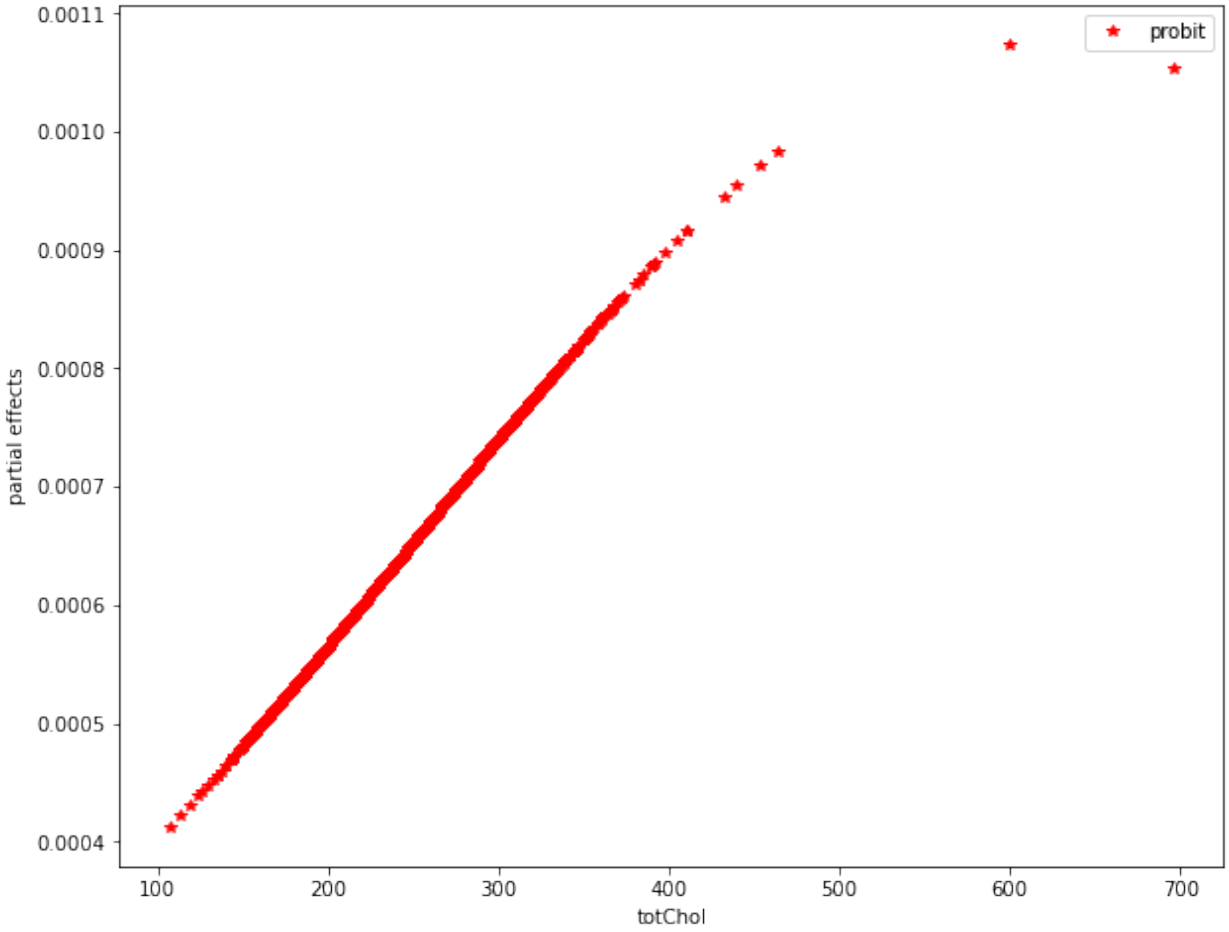
In [69]:
```
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ heartRate', data = df
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['heartRate'] * factor_probit

# plot APE's:
x = df2['heartRate']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')

plt.ylabel('partial effects')
plt.xlabel('heartRate')
```

Out[69]: Text(0.5, 0, 'heartRate')



In [70]:
```
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ totChol', data = df2)
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['totChol'] * factor_probit

# plot APE's:
x = df2['totChol']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')

plt.ylabel('partial effects')
plt.xlabel('totChol')
plt.legend()
```

Out[70]: <matplotlib.legend.Legend at 0x7fe3abc19580>



In [75]:
```
import wooldridge as woo
import pandas as pd
```

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import scipy.stats as stats


# estimate models:
reg_lin = smf.ols(formula='TenYearCHD ~ cigsPerDay + totChol + heartRa
results_lin = reg_lin.fit(cov_type='HC3')

reg_logit = smf.logit(formula='TenYearCHD ~ cigsPerDay + totChol + hea
results_logit = reg_logit.fit(disp=0)

reg_probit = smf.probit(formula=' TenYearCHD ~ cigsPerDay + totChol +
results_probit = reg_probit.fit(disp=0)

# manual average partial effects:
APE_lin = np.array(results_lin.params)

xb_logit = results_logit.fittedvalues
factor_logit = np.mean(stats.logistic.pdf(xb_logit))
APE_logit_manual = results_logit.params * factor_logit

xb_probit = results_probit.fittedvalues
factor_probit = np.mean(stats.norm.pdf(xb_probit))
APE_probit_manual = results_probit.params * factor_probit

table_manual = pd.DataFrame({'APE_lin': np.round(APE_lin, 4),
                             'APE_logit_manual': np.round(APE_logit_ma
                             'APE_probit_manual': np.round(APE_probit_
print(f'table_manual: \n{table_manual}\n')

# automatic average partial effects:
coef_names = np.array(results_lin.model.exog_names)
coef_names = np.delete(coef_names, 0)  # drop Intercept

APE_logit_autom = results_logit.get_margeff().margeff
APE_probit_autom = results_probit.get_margeff().margeff

table_auto = pd.DataFrame({'coef_names': coef_names,
                           'APE_logit_autom': np.round(APE_logit_autom
                           'APE_probit_autom': np.round(APE_probit_aut
print(f'table_auto: \n{table_auto}\n')
```

```
table_manual:
            APE_lin  APE_logit_manual  APE_probit_manual
Intercept   -0.1718           -0.4922            -0.5186
cigsPerDay   0.0020            0.0019             0.0019
totChol      0.0006            0.0006             0.0006
heartRate    0.0000            0.0001             0.0001
glucose      0.0019            0.0013             0.0014
```

```
table_auto:
   coef_names  APE_logit_autom  APE_probit_autom
0  cigsPerDay           0.0019            0.0019
1     totChol           0.0006            0.0006
2   heartRate           0.0001            0.0001
3     glucose           0.0013            0.0014
```

Marginal effects plots have been plotted for all the response variables which have high economic significance.

Part 4. Prediction based on prefered model.

From above, our prefered model is Probit model.

In [72]:
```
X_new = pd.DataFrame(
    {'male': [0, 0, 1, 1], 'age': [32, 41, 57, 70],
     'education': [1, 2, 3, 4], 'currentSmoker': [0, 0, 1, 1],
     'cigsPerDay': [0, 9, 20, 70], 'BPMeds': [0, 0, 1, 1],
     'prevalentStroke': [0, 0, 1, 1], 'prevalentHyp': [0, 0, 1, 1],
     'diabetes': [0, 0, 1, 1], 'totChol': [107, 190, 278, 696],
     'sysBP': [84, 106, 150, 295], 'diaBP': [48, 64, 86, 143],
     'BMI': [16, 21, 29, 57], 'heartRate': [44, 63, 84, 143],
     'glucose': [40, 55, 101, 394]})

predictions_probit = results_probit.predict(X_new)
print(f'predictions_probit: \n{predictions_probit}\n')
```

```
predictions_probit:
0    0.004711
1    0.027154
2    0.717347
3    0.999978
dtype: float64
```

Two of the predictions return as 0, two return as 1. The probit model has a specificity of 99.53% but a sensitivity of 7.61%. The negatives are likely correct, but we are less confident in the positives.

The reliability of our model is 84.97% to predict accurate dependendent binary classification.