In [1]:
```
import platform
import socket
import os
socket.gethostname()
```

Out[1]:
```
'DESKTOP-R0861CH'
```

In [2]:
```
import pandas as pd
df = pd.read_csv('diabetes.csv')
```

In [3]:
```
df.head()
```

Out[3]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | O |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | |

In [4]:
```
import pandas_profiling as pp
pp.ProfileReport(df)
```

# Overview

## Dataset statistics

| | |
|---|---|
| **Number of variables** | 9 |
| **Number of observations** | 768 |
| **Missing cells** | 0 |
| **Missing cells (%)** | 0.0% |
| **Duplicate rows** | 0 |
| **Duplicate rows (%)** | 0.0% |
| **Total size in memory** | 54.1 KiB |
| **Average record size in memory** | 72.2 B |

## Variable types

| | |
|---|---|
| **Numeric** | 8 |
| **Categorical** | 1 |

## Warnings

| | |
|---|---|
| `Pregnancies` is highly correlated with `Age` | **High correlation** |
| `Age` is highly correlated with `Pregnancies` | **High correlation** |
| `Pregnancies` is highly correlated with `Age` | **High correlation** |
| `SkinThickness` is highly correlated with `Insulin` | **High correlation** |

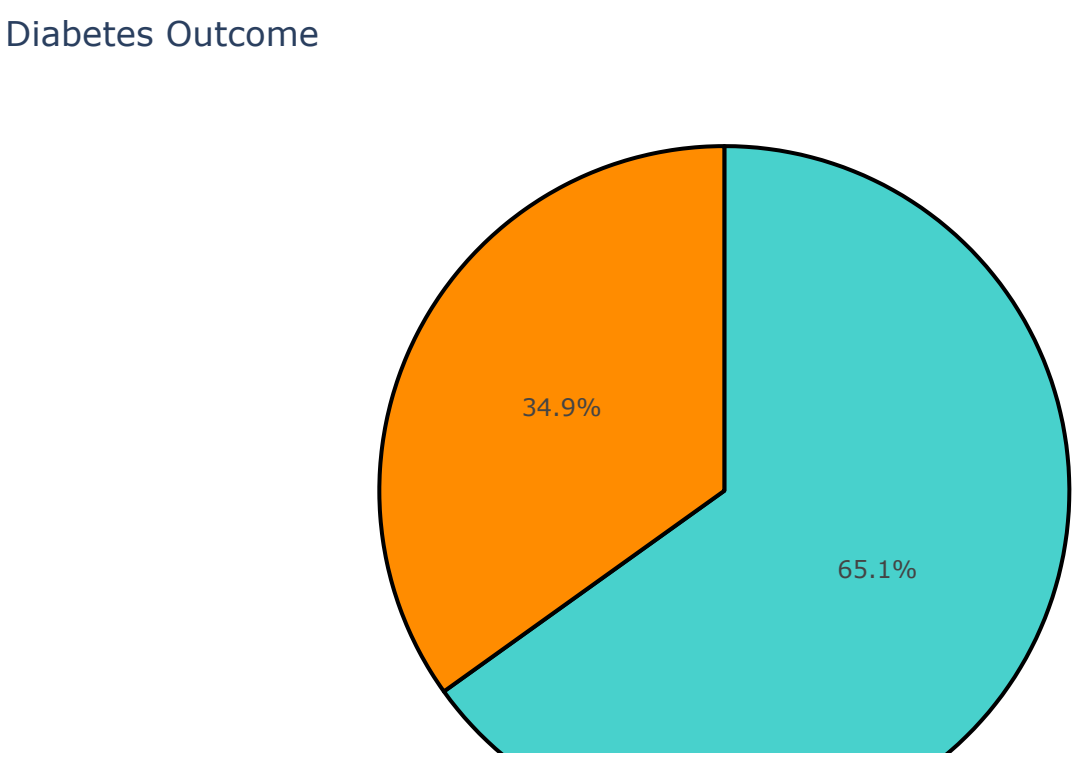
Out[4]:

In [5]:
```
# Visualization Imports
import matplotlib.pyplot as plt
import seaborn as sns
color = sns.color_palette()
get_ipython().run_line_magic('matplotlib', 'inline')
ffline as py
py.init_notebook_mode(connected=True)
```

```
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.express as px
import numpy as np
```

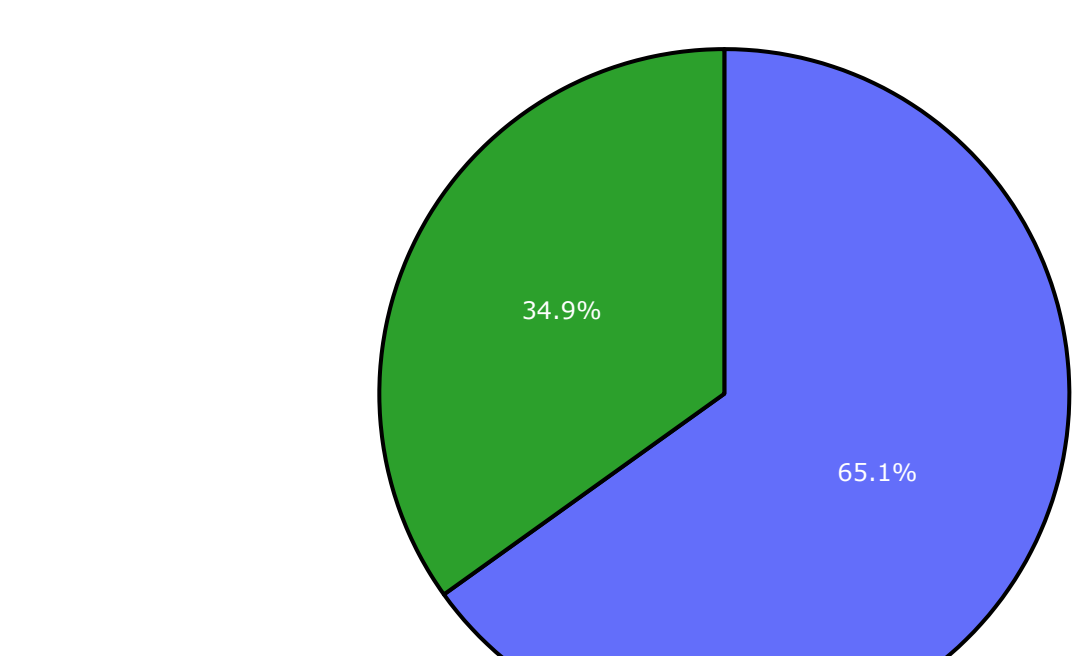
In [6]:

```
dist = df['Outcome'].value_counts()
colors = ['mediumturquoise', 'darkorange']
trace = go.Pie(values=(np.array(dist)),labels=dist.index)
layout = go.Layout(title='Diabetes Outcome')
data = [trace]
fig = go.Figure(trace,layout)
fig.update_traces(marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.show()
```



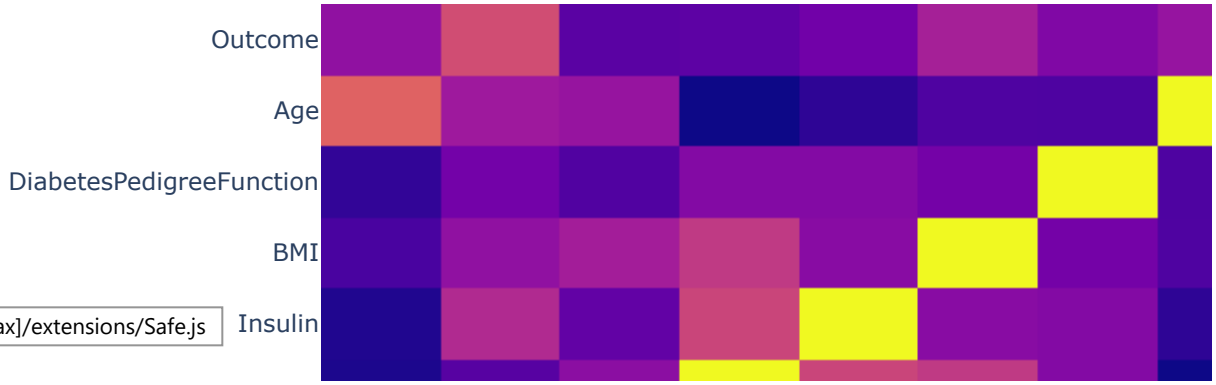

In [10]:

```
dist = df['Outcome'].value_counts()
colors = ['mediumlavender', '#2ca02c']
trace = go.Pie(values=(np.array(dist)),labels=dist.index)
layout = go.Layout(title='Diabetes Outcome')
data = [trace]
fig = go.Figure(trace,layout)
fig.update_traces(marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.show()
```



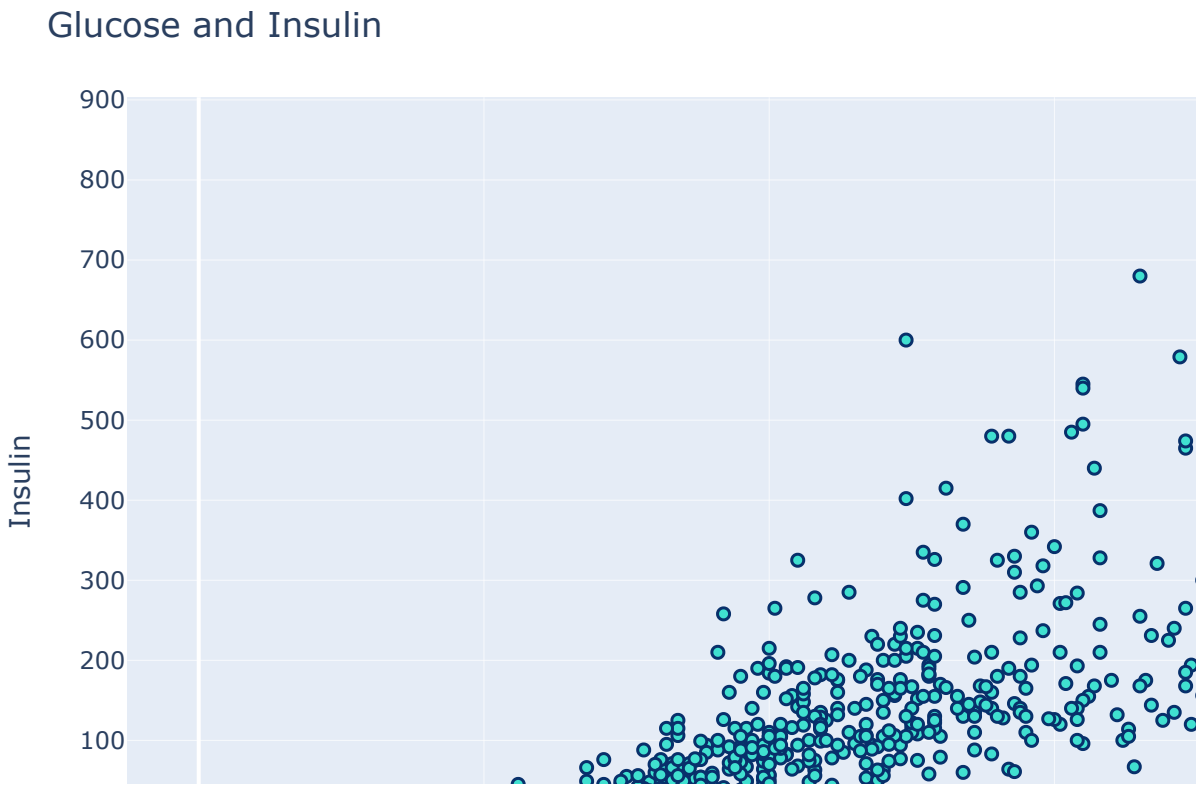

In [7]:

```
def df_to_plotly(df):
    return {'z': df.values.tolist(),
            'x': df.columns.tolist(),
            'y': df.index.tolist() }
import plotly.graph_objects as go
dfNew = df.corr()
fig = go.Figure(data=go.Heatmap(df_to_plotly(dfNew)))
fig.show()
```




In [11]:

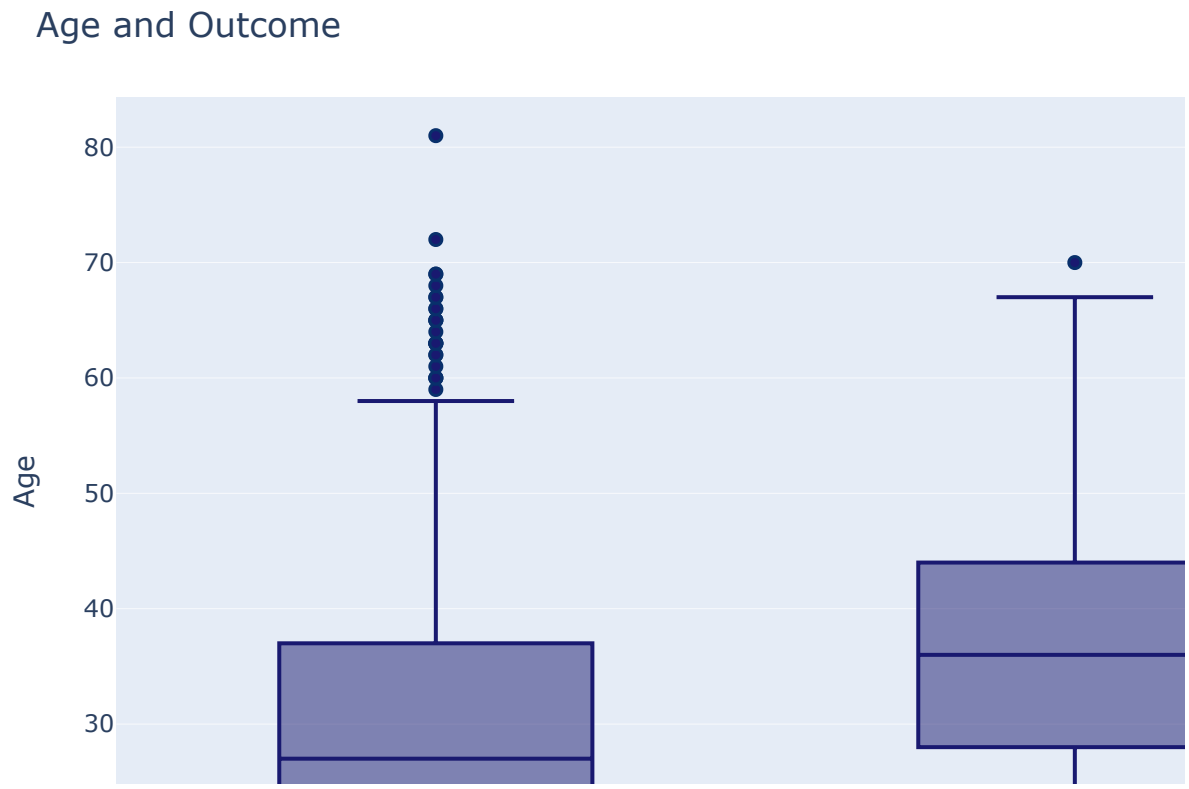
```
fig = px.scatter(df, x='Glucose', y='Insulin')
fig.update_traces(marker_color="turquoise",marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5)
fig.update_layout(title_text='Glucose and Insulin')
fig.show()
```




In [12]:

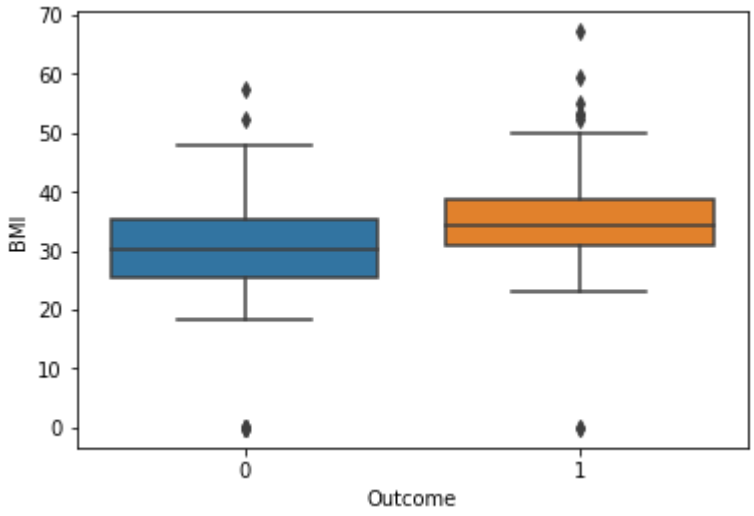
```
fig = px.box(df, x='Outcome', y='Age')
es(marker_color="midnightblue",marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5)
```

```
fig.update_layout(title_text='Age and Outcome')
fig.show()
```




In [15]:

```
plot = sns.boxplot(x='Outcome',y="BMI",data=df)
```




In [16]:

```
import pandas as pd
import numpy as np
```

```
from matplotlib import pyplot as plt

# Generate 'random' data
np.random.seed(0)
X = 2.5 * np.random.randn(100) + 1.5   # Array of 100 values with mean = 1.5, stddev =
res = 0.5 * np.random.randn(100)       # Generate 100 residual terms
y = 2 + 0.3 * X + res                  # Actual values of Y

# Create pandas dataframe to store our X and y values
df = pd.DataFrame(
    {'X': X,
     'y': y}
)

# Show the first five rows of our dataframe
df.head()
```

Out[16]:

| | X | y |
|---|---|---|
| **0** | 5.910131 | 4.714615 |
| **1** | 2.500393 | 2.076238 |
| **2** | 3.946845 | 2.548811 |
| **3** | 7.102233 | 4.615368 |
| **4** | 6.168895 | 3.264107 |

In [17]:

```
# Calculate the mean of X and y
xmean = np.mean(X)
ymean = np.mean(y)

# Calculate the terms needed for the numator and denominator of beta
df['xycov'] = (df['X'] - xmean) * (df['y'] - ymean)
df['xvar'] = (df['X'] - xmean)**2

# Calculate beta and alpha
beta = df['xycov'].sum() / df['xvar'].sum()
alpha = ymean - (beta * xmean)
print(f'alpha = {alpha}')
print(f'beta = {beta}')
```

```
alpha = 2.0031670124623426
beta = 0.3229396867092763
```

In [20]:

```
ypred = alpha + beta * X
ypred
```

Out[20]:

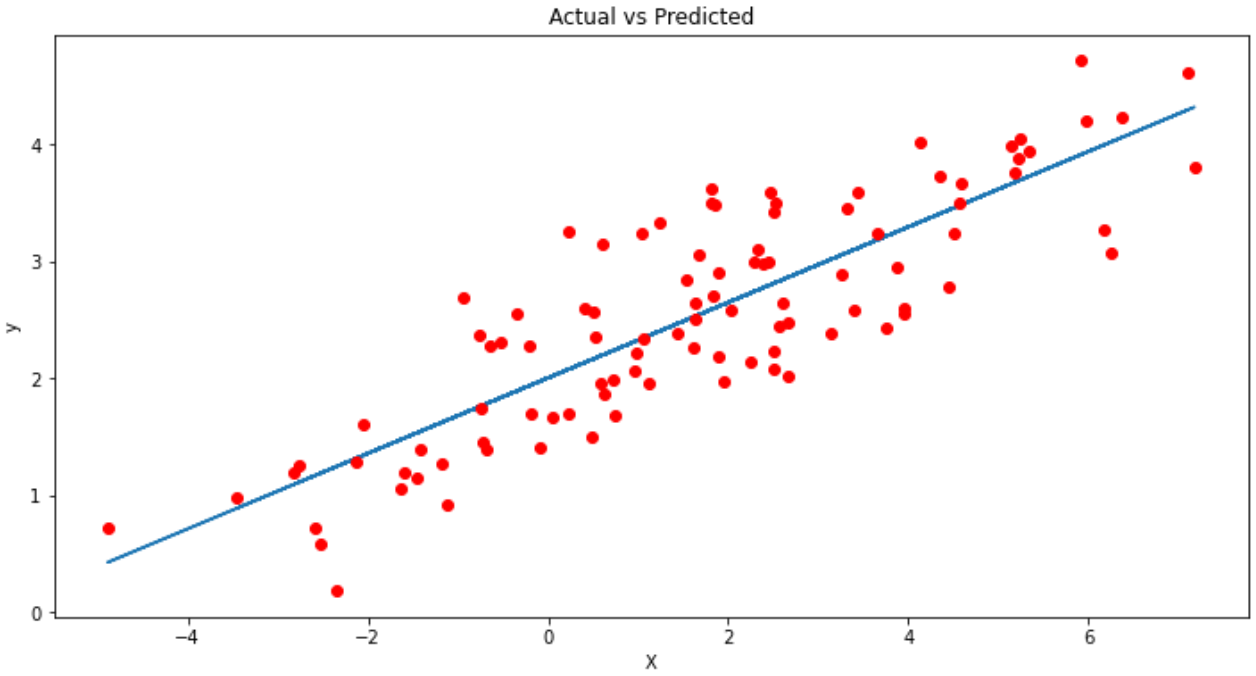
```
array([3.91178282, 2.81064315, 3.27775989, 4.29675991, 3.99534802,
       1.69857201, 3.25462968, 2.36537842, 2.40424288, 2.81907292,
       2.60387001, 3.66168312, 3.10199975, 2.58581077, 2.84592918,
       2.75696825, 3.69382011, 2.32194218, 2.74033151, 1.79802302,
       0.42642221, 3.015275  , 3.18547843, 1.88839019, 4.32006116,
       1.31339555, 2.52451965, 2.33645381, 3.72506464, 3.67386219,
       2.61267323, 2.79288576, 1.77082341, 0.88838207, 2.20668994,
       2.61380476, 3.48085076, 3.45831697, 2.17486854, 2.24351265,
       2.1338976 , 1.47613319, 3.11528277, 1.18459738, 2.31582084,
```

```
1.76462232, 2.79994197, 2.07517841, 1.53439407, 2.46482364,
2.83338994, 2.54127917, 2.73177699, 1.9754571 , 2.19471775,
1.94466613, 2.19729158, 1.83108353, 1.09386364, 2.6308214 ,
2.16319902, 1.17143718, 2.86120343, 1.75506992, 2.52951462,
3.07620724, 2.59171079, 3.40747079, 1.49064088, 2.81240675,
1.93469565, 1.78453915, 2.02024272, 2.23604485, 2.53292159,
1.54689373, 3.2148581 , 2.86352875, 1.24729141, 3.68911579,
4.01822118, 3.43926331, 2.34231437, 1.62310525, 3.33888732,
2.16207195, 3.47451661, 2.65572718, 3.2760653 , 2.77528867,
3.05802784, 2.49605373, 3.92939769, 2.59003892, 2.81212234])
```

In [21]:

```
# Plot regression against actual data
plt.figure(figsize=(12, 6))
plt.plot(X, ypred)     # regression line
plt.plot(X, y, 'ro')   # scatter plot showing actual data
plt.title('Actual vs Predicted')
plt.xlabel('X')
plt.ylabel('y')

plt.show()
```




In [23]:

```
# Import and display first five rows of advertising dataset
advert = pd.read_csv('advertising.csv')
advert.head()
```

Out[23]:

| | TV | Radio | Newspaper | Sales |
|---|---|---|---|---|
| **0** | 230.1 | 37.8 | 69.2 | 22.1 |
| **1** | 44.5 | 39.3 | 45.1 | 10.4 |
| **2** | 17.2 | 45.9 | 69.3 | 12.0 |
| **3** | 151.5 | 41.3 | 58.5 | 16.5 |
| **4** | 180.8 | 10.8 | 58.4 | 17.9 |

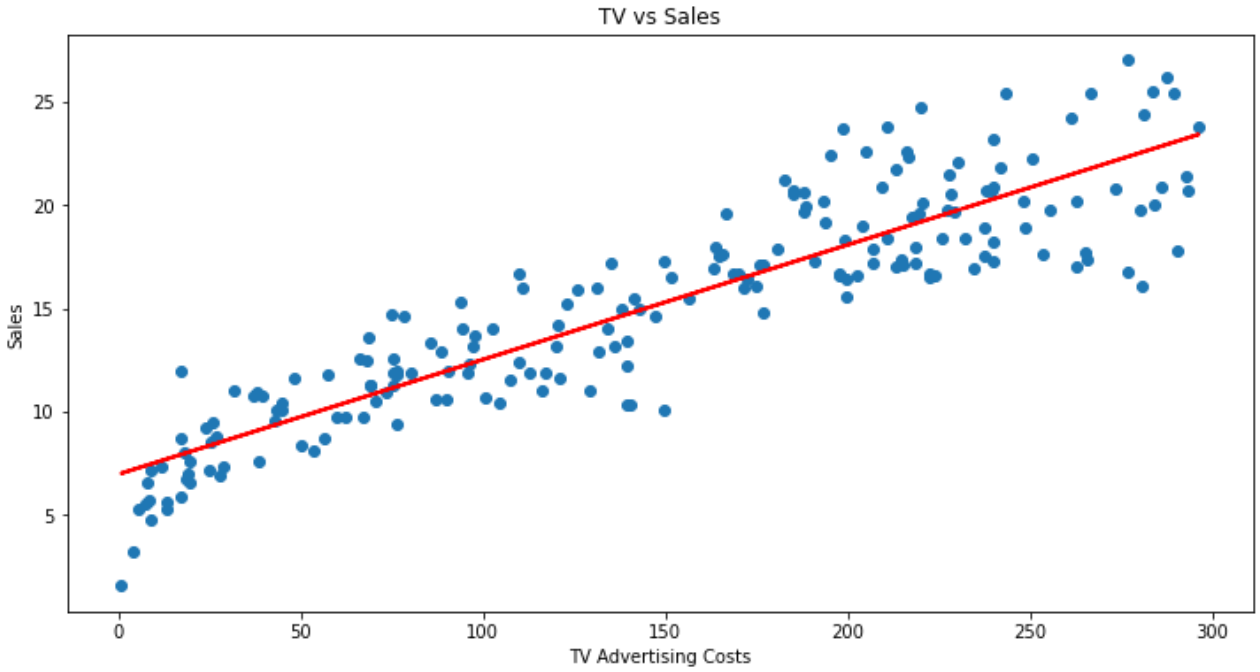
In [24]:
```
import statsmodels.formula.api as smf

# Initialise and fit linear regression model using `statsmodels`
model = smf.ols('Sales ~ TV', data=advert)
model = model.fit()
```

In [27]:
```
# Predict values
sales_pred = model.predict()

# Plot regression against actual data
plt.figure(figsize=(12, 6))
plt.plot(advert['TV'], advert['Sales'], 'o')           # scatter plot showing actual da
plt.plot(advert['TV'], sales_pred, 'r', linewidth=2)   # regression line
plt.xlabel('TV Advertising Costs')
plt.ylabel('Sales')
plt.title('TV vs Sales')

plt.show()
```




In [28]:
```
new_X = 400
model.predict({"TV": new_X})
```

Out[28]:
```
0    29.16073
dtype: float64
```

In [29]:
```
from sklearn.linear_model import LinearRegression

# Build linear regression model using TV and Radio as predictors
# Split data into predictors X and output Y
predictors = ['TV', 'Radio']
X = advert[predictors]
y = advert['Sales']

# Initialise and fit model
```

```
lm = LinearRegression()
model = lm.fit(X, y)
```

In [30]:
```
print(f'alpha = {model.intercept_}')
print(f'betas = {model.coef_}')
```

```
alpha = 4.630879464097768
betas = [0.05444896 0.10717457]
```

In [31]:
```
model.predict(X)
```

Out[31]:
```
array([21.21078412, 11.26581887, 10.48671441, 17.30620681, 15.63273693,
       10.34542196, 11.27702065, 13.27626614,  5.32420713, 15.7884357 ,
        8.85156828, 18.89326105,  9.68859218, 10.75417988, 19.26995575,
       20.38243344, 12.24510831, 24.19693004, 10.59582626, 15.21268363,
       19.49126811, 18.10365306,  7.05368143, 18.87282745,  9.37344932,
       19.32062224, 15.55185089, 19.49389028, 21.08221178, 10.18976923,
       23.61202043, 12.64300467, 10.0840803 , 21.23601486,  9.99168941,
       20.89860809, 23.85755331, 13.99264065,  9.83919073, 21.08572385,
       18.04678695, 17.8479762 , 23.58582996, 16.79663584,  8.75193486,
       16.57632034, 10.5759795 , 22.14092985, 18.69504157,  9.52745742,
       15.84202299, 11.12643101, 20.88281419, 19.52472489, 22.0212491 ,
       20.75520158,  8.03996233, 14.10457969, 21.42457912, 19.26492534,
        7.75824801, 23.4347471 , 19.32172162, 13.39515504, 16.35620987,
        9.38458127,  8.98251618, 13.76965098, 20.50436345, 21.14037783,
       18.75120943, 12.14197172,  9.62687247, 12.28747004, 18.88678214,
       10.23459567,  6.2997052 , 14.24645452,  8.12942354, 11.77220311,
       11.65234112, 18.12715599, 10.90653001, 13.1244568 , 20.86423915,
       17.12243079, 11.73263588, 15.00966701, 12.17167427, 15.73231986,
       12.46853029,  6.3488816 , 20.07476636, 22.20399557, 11.97914185,
       16.90911121, 15.7651051 , 16.9491583 , 24.93822776, 16.46155858,
       17.20117899, 24.65998836, 20.96994143, 16.70524181, 21.27670971,
       17.11229126,  7.17102377,  9.58521789,  5.38703068, 21.42014   ,
       17.80428628, 21.86382698, 15.84805026, 18.2511778 , 13.90455813,
       12.47110641, 13.74277117,  8.87651972, 15.42985551,  7.40198244,
       15.19679613,  7.98020812, 17.08466564, 15.0417867 , 20.58865461,
       10.64348878,  9.22467218,  8.99768611, 21.87753951,  9.16213238,
        8.91310676, 19.38155011,  8.00339907, 20.18910917, 10.77698457,
       12.29796912, 10.20458114, 22.63090513,  9.74800617, 19.40345598,
       10.44940089, 18.97162298, 20.19507107, 10.93713581, 11.45505314,
       12.47370034, 18.48644931, 23.12442071, 11.0190752 ,  9.82985195,
       21.40442928, 12.11947011, 17.88716162, 18.21281692, 17.11777774,
        6.09734523, 14.40573073, 12.92666072,  9.22267399, 13.7738197 ,
       15.96318493, 13.13400505, 16.82892341, 17.47730877, 12.58776386,
       17.7635543 ,  9.63527974, 16.44823231, 18.88850549, 21.24676946,
        8.59655253, 15.82768205,  7.85228798, 14.56102391, 17.10472187,
       24.94863323, 21.39267336, 14.73405424, 19.94340841, 14.71937307,
       13.4362406 , 17.10672007,  8.30180611, 24.89890714, 20.73284407,
       20.62648958, 12.45157609, 18.11198606, 21.69300877,  6.94588734,
       11.18648829,  9.89926137,  6.00681733, 18.21429811, 16.59730363,
        7.10737568, 10.28512696, 15.26506903, 24.57393672, 18.19018456])
```

In [32]:
```
new_X = [[300, 200]]
print(model.predict(new_X))
```

```
[42.40048195]
```



In [ ]: