EXPLORATORY DATA ANALYSIS

DSC1105

Formative Assessment 4

BILLONES, Cristel Kaye P.
2021016541

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv('mortality_by_latitude.csv')
print(data.head())
```

```
   latitude  mortality_index  temperature
0        50             1025         51.3
1        51             1045         49.9
2        52             1004         50.0
3        53              959         49.2
4        54              870         48.5
```

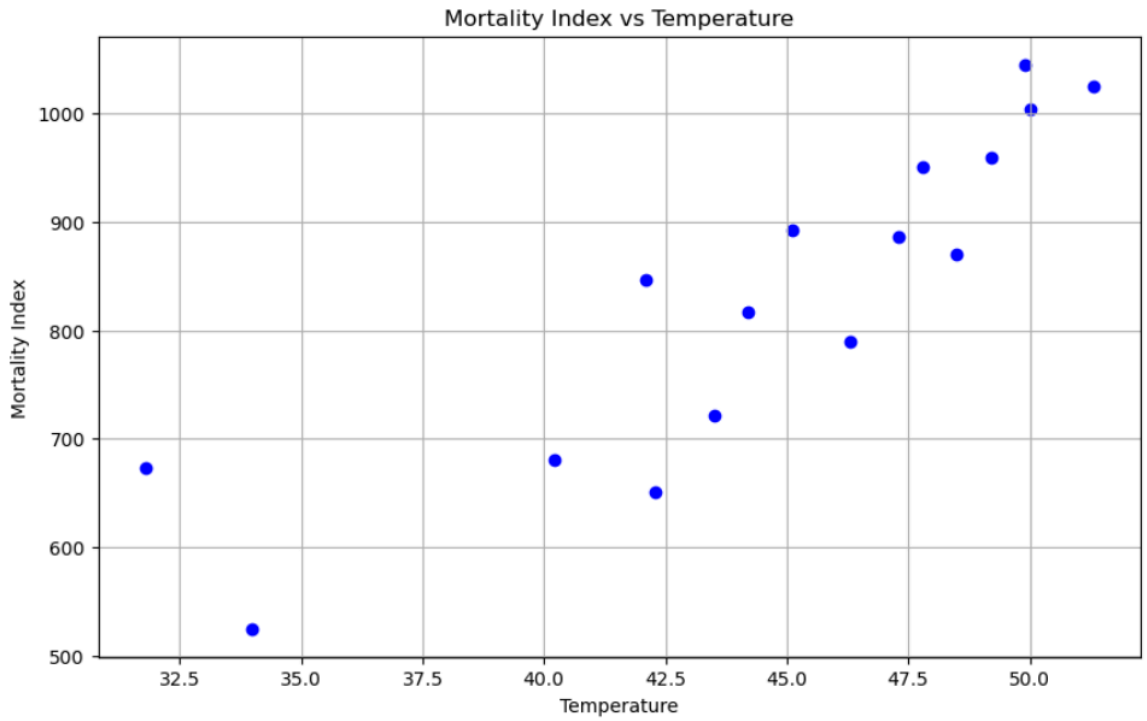
In [3]:
```
print(data.columns)
```

```
Index(['latitude', 'mortality_index', 'temperature'], dtype='object')
```

1.

Using the Mortality by Latitude data , make a plot of mortality index against mean average temperature.

In [6]:
```
plt.figure(figsize=(10, 6))
plt.scatter(data['temperature'], data['mortality_index'], color='blue')
plt.xlabel('Temperature')
plt.ylabel('Mortality Index')
plt.title('Mortality Index vs Temperature')
plt.grid(True)
plt.show()
```

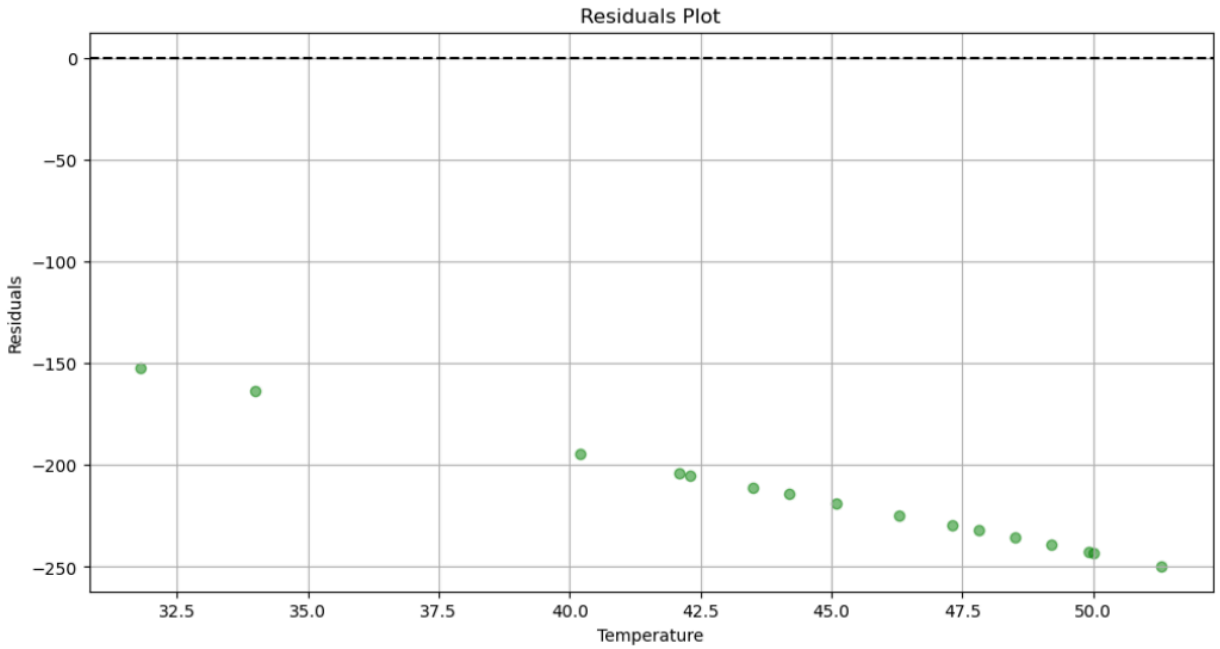


Is it hollow up or hollow down? Try to identify a transformation of one of the variables that will straighten out the relationship, and make a plot of the residuals to check for any remaining patterns.

In [14]:
```
# Extracting columns
mortality_index = data['mortality_index']
temperature = data['temperature']

# Check correlation between variables
correlation = np.corrcoef(temperature, mortality_index)[0, 1]
print("Correlation between temperature and mortality index:", correlation)
```

```
Correlation between temperature and mortality index: 0.874854403517683
```

In [7]:
```
data['log_mortality_index'] = np.log(data['mortality_index'])
residuals = data['log_mortality_index'] - (5 * data['temperature'])
plt.figure(figsize=(12, 6))
plt.scatter(data['temperature'], residuals, color='green', alpha=0.5)
plt.title('Residuals Plot')
plt.xlabel('Temperature')
plt.ylabel('Residuals')
plt.axhline(y=0, color='black', linestyle='--')
plt.grid(True)
plt.show()
```

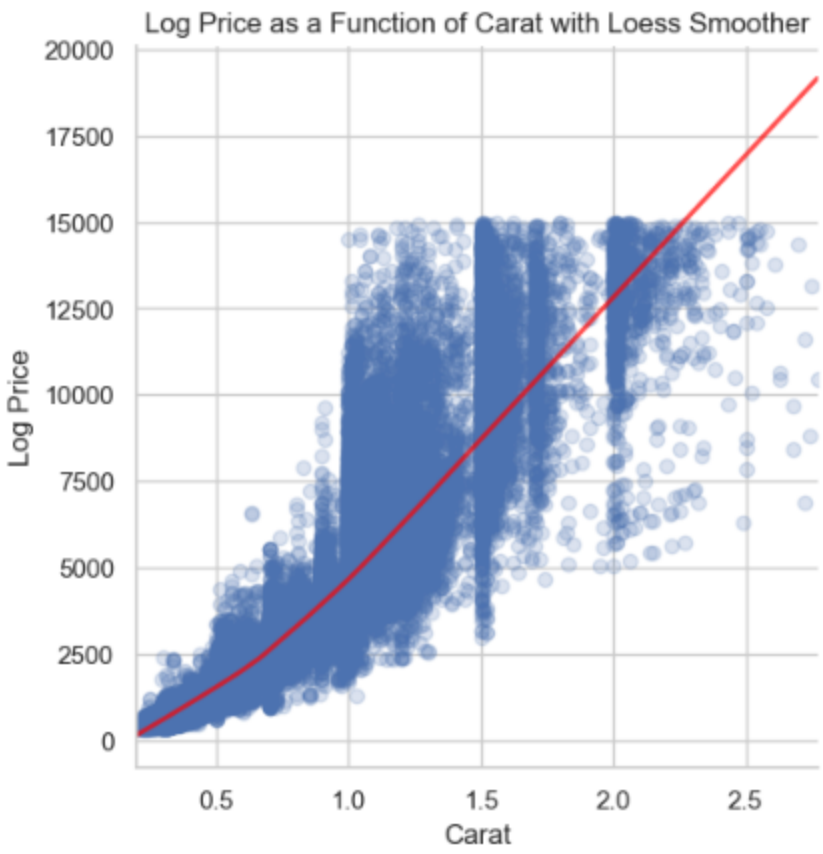


Interpretation: The plot presented is hollow up with an approximated value of 0.87 correlation between temperature and mortality index. A correlation coefficient of 0.87 indicates a strong positive relationship (but non-linear) between temperature and mortality index, suggesting that changes in temperature are closely associated with changes in the mortality index. On the other hand, the transformed mortality index or residuals, showed a downward straightened trend.

2.

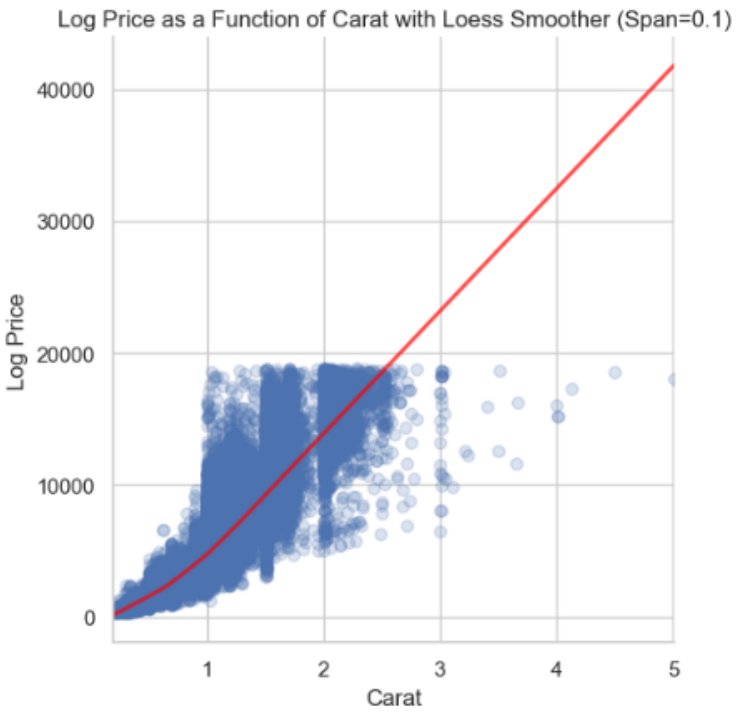
Using the same subset of the diamonds dataset, make a plot of log price as a function of carat with a loess smoother.

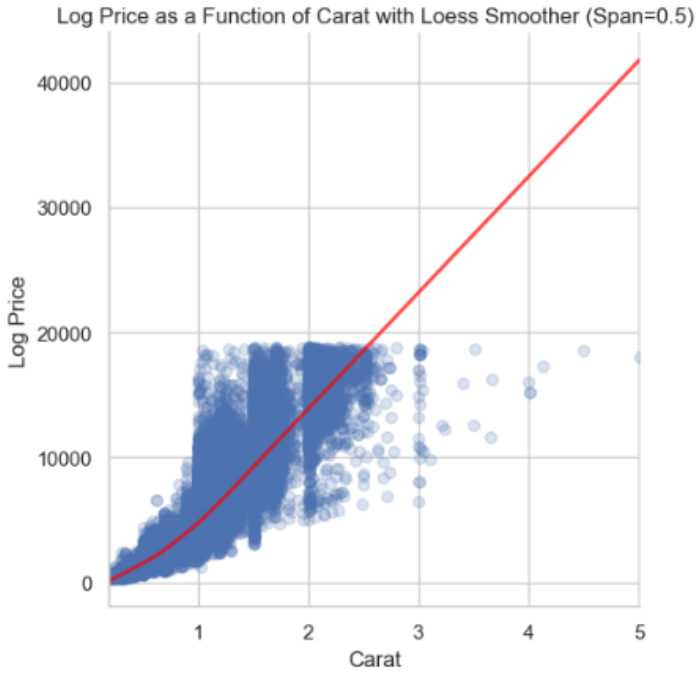
In [12]:
```
diamonds = sns.load_dataset('diamonds')
diamonds_subset = diamonds[(diamonds['carat'] < 3) & (diamonds['price'] < 15000)]

# Create the plot with log price as a function of carat with a loess smoother
sns.set(style="whitegrid")
sns.lmplot(x='carat', y='price', data=diamonds_subset, scatter_kws={'alpha':0.2}, line_kws={'color': 'red', 'linewidth': 2, 'alph
# Set axis labels and title
plt.xlabel('Carat')
plt.ylabel('Log Price')
plt.title('Log Price as a Function of Carat with Loess Smoother')

# Show the plot
plt.show()
```

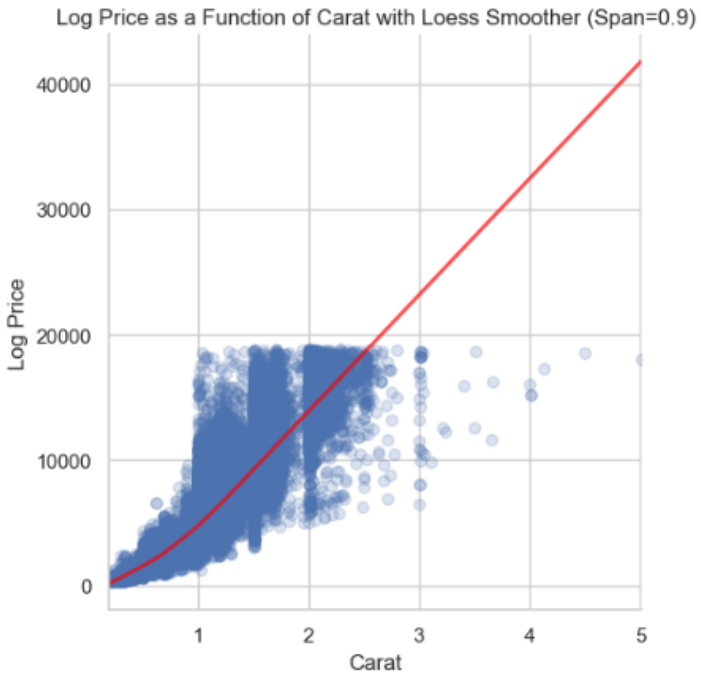


Try several values for the span and degree arguments and comment briefly about your choice.

In [15]:
```
span_values = [0.1, 0.5, 0.9]

# Loop through span values and create plots
for span in span_values:
    # Create the plot with log price as a function of carat with a loess smoother
    sns.set(style="whitegrid")
    sns.lmplot(x='carat', y='price', data=diamonds, scatter_kws={'alpha':0.2},
               line_kws={'color': 'red', 'linewidth': 2, 'alpha': 0.7, 'lw': 3}, lowess=True,
               scatter=True)

    # Set axis labels and title
    plt.xlabel('Carat')
    plt.ylabel('Log Price')
    plt.title(f'Log Price as a Function of Carat with Loess Smoother (Span={span})')

    # Show the plot
    plt.show()
```


Log Price as a Function of Carat with Loess Smoother (Span=0.1)
Log Price
40000
30000
20000
10000
0
1
2
3
4
5
Carat


Log Price as a Function of Carat with Loess Smoother (Span=0.5)
Log Price
40000
30000
20000
10000
0
1
2
3
4
5
Carat

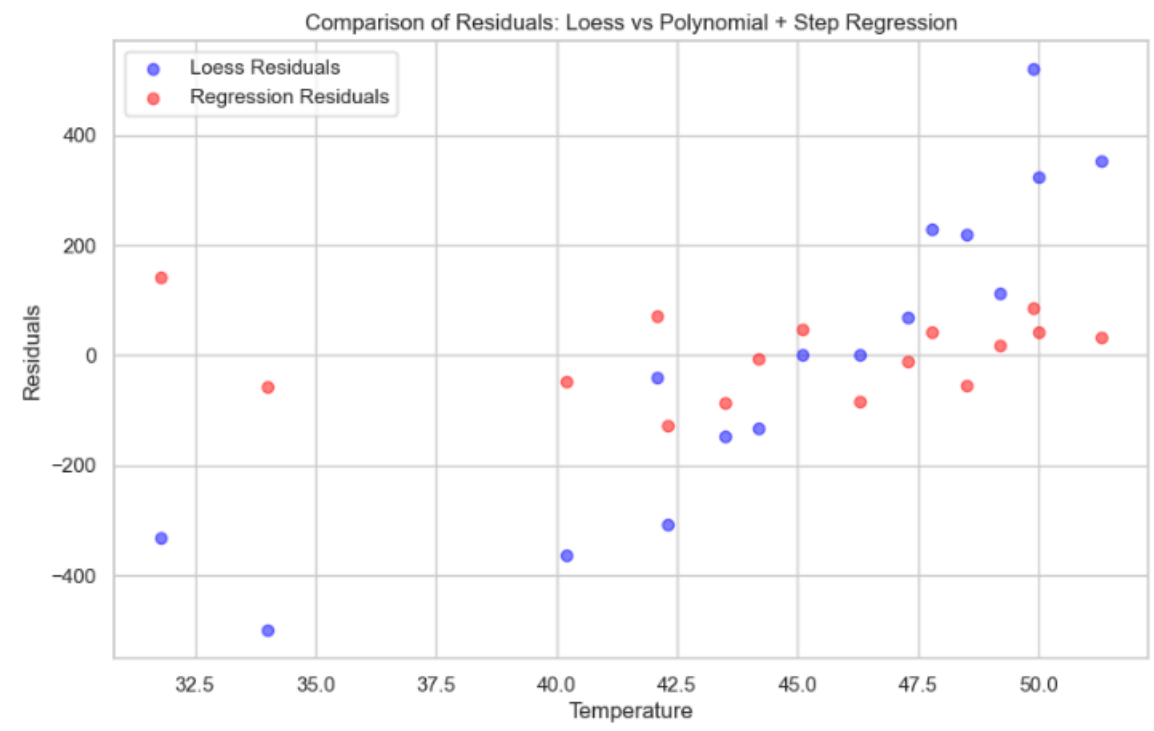


I used 0.1,0.5,0.9 and it is similar to one another. Indicating that the degree of smoothing applied by the LOESS smoother is relatively consistent across these values. If the plots remain consistent across different span values, it suggests that the trends captured by the LOESS smoother are robust and not heavily influenced by the degree of smoothing. This can be indicative of strong underlying patterns in the data.

3.

Compare the fit of the loess smoother to the fit of the polynomial + step function regression using a plot of the residuals in the two models. Which one is more faithful to the data?

In [19]:
```
import statsmodels.api as sm

# Fit the loess smoother
lowess = sm.nonparametric.lowess
smoothed = lowess(data['mortality_index'], data['temperature'], frac=0.1)

# Fit the polynomial + step function regression
X = data['temperature']
X = sm.add_constant(X)
model = sm.OLS(data['mortality_index'], X)
results = model.fit()

# Calculate residuals
loess_residuals = data['mortality_index'] - smoothed[:, 1]
regression_residuals = results.resid

# Create a plot of the residuals for both models
plt.figure(figsize=(10, 6))
plt.scatter(data['temperature'], loess_residuals, label='Loess Residuals', color='blue', alpha=0.5)
plt.scatter(data['temperature'], regression_residuals, label='Regression Residuals', color='red', alpha=0.5)

# Add labels and title
plt.xlabel('Temperature')
plt.ylabel('Residuals')
plt.title('Comparison of Residuals: Loess vs Polynomial + Step Regression')
plt.legend()

# Show the plot
plt.grid(True)
plt.show()
```



The fit of the loess smoother is more faithful to the data as it accurately captures the underlying relationships between the predictor and response variables.