In [ ]:

In [ ]:

Variable selection justification starts on page 9

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
import statsmodels.api as sm
```

Reading and viewing the data

In [32]:
```
bike_data = pd.read_csv('bike.csv')

#bike_data.info()
# bike_data.head()
```

Clubbing different types of crimes under broader categories | drawing inspiration for this from the previous assignment's suggestion for CBD.

In [4]:
```
category_2_crime_dict = dict()

category_2_crime_dict["violent_crimes"] = [
    "HOMICIDE",
    "ASSAULT",
    "BATTERY",
    "KIDNAPPING"
]


category_2_crime_dict["harrassment_crimes"] = [

    "INTIMIDATION",
    "STALKING"
]


category_2_crime_dict["property_crimes"] = [
    "BURGLARY",
    "MOTOR_VEHICLE_THEFT",
    "THEFT",
    "ARSON",
    "CRIMINAL_DAMAGE",
    "CRIMINAL_TRESPASS",
    "ROBBERY"
```

```
]


category_2_crime_dict["sex_crimes"] = [
    "CRIM_SEXUAL_ASSAULT",
    "SEX_OFFENSE",
    "PROSTITUTION",
    "HUMAN_TRAFFICKING",
    "PUBLIC_INDECENCY"
]


category_2_crime_dict["drug_related_crimes"] = [
    "NARCOTICS",
    "OTHER_NARCOTIC_VIOLATION"
]


category_2_crime_dict["white_collar_financial_crimes"] = [

    "DECEPTIVE_PRACTICE"

]


category_2_crime_dict["weapons_offenses"] = [

    "WEAPONS_VIOLATION",
    "CONCEALED_CARRY_LICENSE_VIOLATIO"

]


category_2_crime_dict["public_order_crimes"] = [

    "PUBLIC_PEACE_VIOLATION",
    "GAMBLING",
    "LIQUOR_LAW_VIOLATION",
    "INTERFERENCE_WITH_PUBLIC_OFFICER"

]


category_2_crime_dict["crimes_against_children"] = [
    "OFFENSE_INVOLVING_CHILDREN"
]

category_2_crime_dict["miscellaneous_crimes"] = [

    "OBSCENITY",
    "OTHER_OFFENSE"

]
```

In [ ]:

Deriving the values for the broader categories | drawing inspiration for this from the previous assignment's suggestion for CBD.

In [25]:
```
list_of_all_crimes = list()

for broader_category, crimes in category_2_crime_dict.items():
    list_of_all_crimes.extend( crimes )
```

In [ ]:

Repeating this process for variables related to location as well.

In [26]:
```
category_2_area_dict = dict()

category_2_area_dict["central_business_district_index"] = [

    "CBD",
    "Limited_Business_License",
    "Retail_Food_Establishment",
    "CAPACITY"
]


category_2_area_dict["transportation_accessibility_index"] = [

    "CTA_BUS_STATIONS",
    "CTA_TRAIN_STATIONS"

]

category_2_area_dict["urban_suburban_spectrum_indicator"] = [

    "PARK_AREA_ACRES",
    "POPULATION_SQ_MILE",
    "BIKE_ROUTES"

]


category_2_area_dict["socioeconomic_status_indicator"] = [

    "PER_CAPITA_INCOME",
    "MINORITY"

]
```

In [ ]:

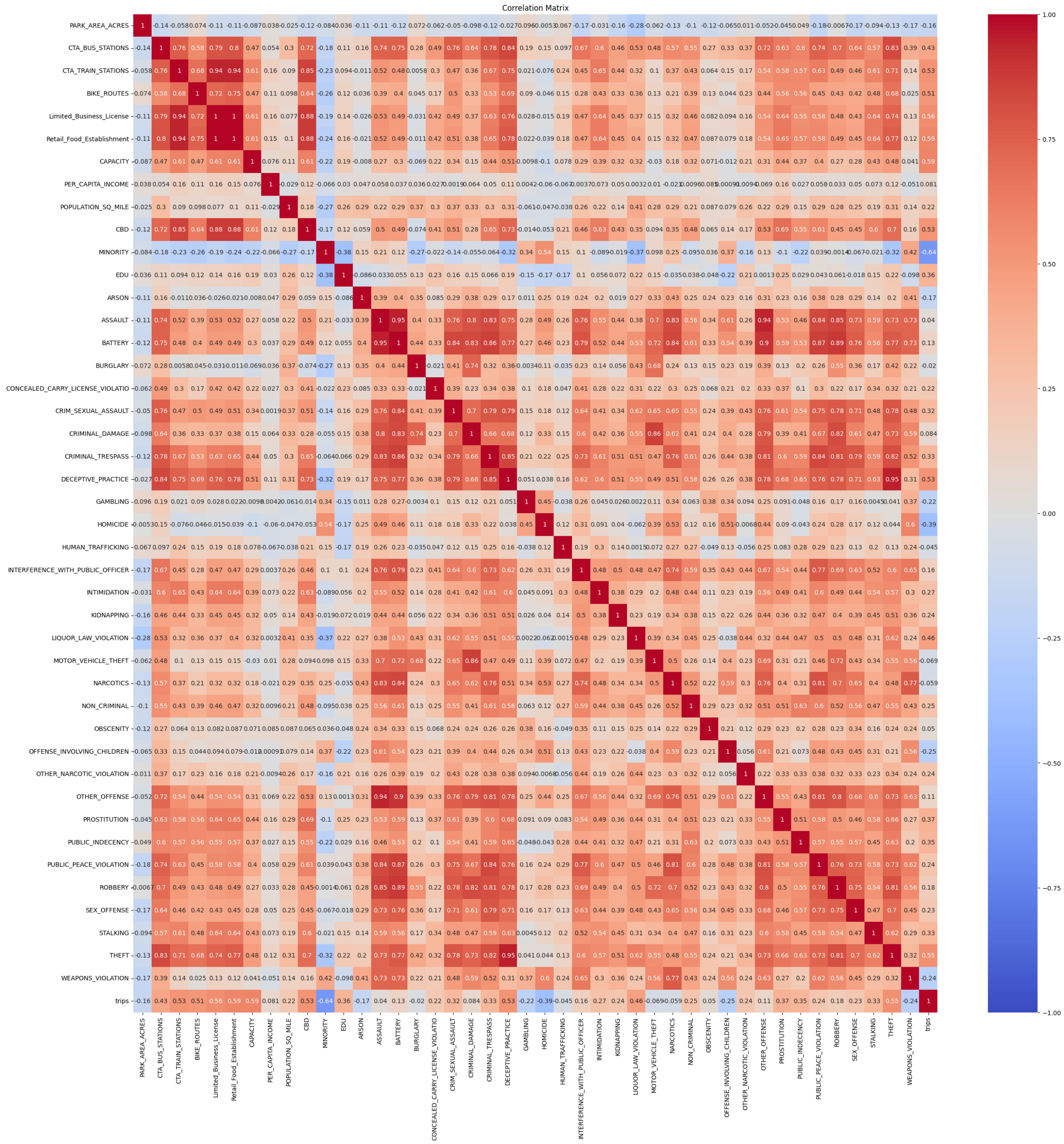
In [30]:
```
list_of_all_area_indicators = list()

for broader_category, indicators in category_2_area_dict.items():
```

```
    list_of_all_area_indicators.extend( indicators )
```

In [29]:
```
bike_data_selected = bike_data.drop(['ID', 'i'], axis=1)
bike_data_selected.columns
```

Out[29]:
```
Index(['PARK_AREA_ACRES', 'CTA_BUS_STATIONS', 'CTA_TRAIN_STATIONS',
       'BIKE_ROUTES', 'Limited_Business_License', 'Retail_Food_Establishment',
       'CAPACITY', 'PER_CAPITA_INCOME', 'POPULATION_SQ_MILE', 'CBD',
       'MINORITY', 'EDU', 'ARSON', 'ASSAULT', 'BATTERY', 'BURGLARY',
       'CONCEALED_CARRY_LICENSE_VIOLATIO', 'CRIM_SEXUAL_ASSAULT',
       'CRIMINAL_DAMAGE', 'CRIMINAL_TRESPASS', 'DECEPTIVE_PRACTICE',
       'GAMBLING', 'HOMICIDE', 'HUMAN_TRAFFICKING',
       'INTERFERENCE_WITH_PUBLIC_OFFICER', 'INTIMIDATION', 'KIDNAPPING',
       'LIQUOR_LAW_VIOLATION', 'MOTOR_VEHICLE_THEFT', 'NARCOTICS',
       'NON_CRIMINAL', 'OBSCENITY', 'OFFENSE_INVOLVING_CHILDREN',
       'OTHER_NARCOTIC_VIOLATION', 'OTHER_OFFENSE', 'PROSTITUTION',
       'PUBLIC_INDECENCY', 'PUBLIC_PEACE_VIOLATION', 'ROBBERY', 'SEX_OFFENSE',
       'STALKING', 'THEFT', 'WEAPONS_VIOLATION', 'trips'],
      dtype='object')
```

In [12]:
```
correlation_matrix = bike_data_selected.corr()

# Visualize the correlation matrix
plt.figure(figsize=(29, 29))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, cente
plt.title('Correlation Matrix')
plt.show()
```



In [ ]:

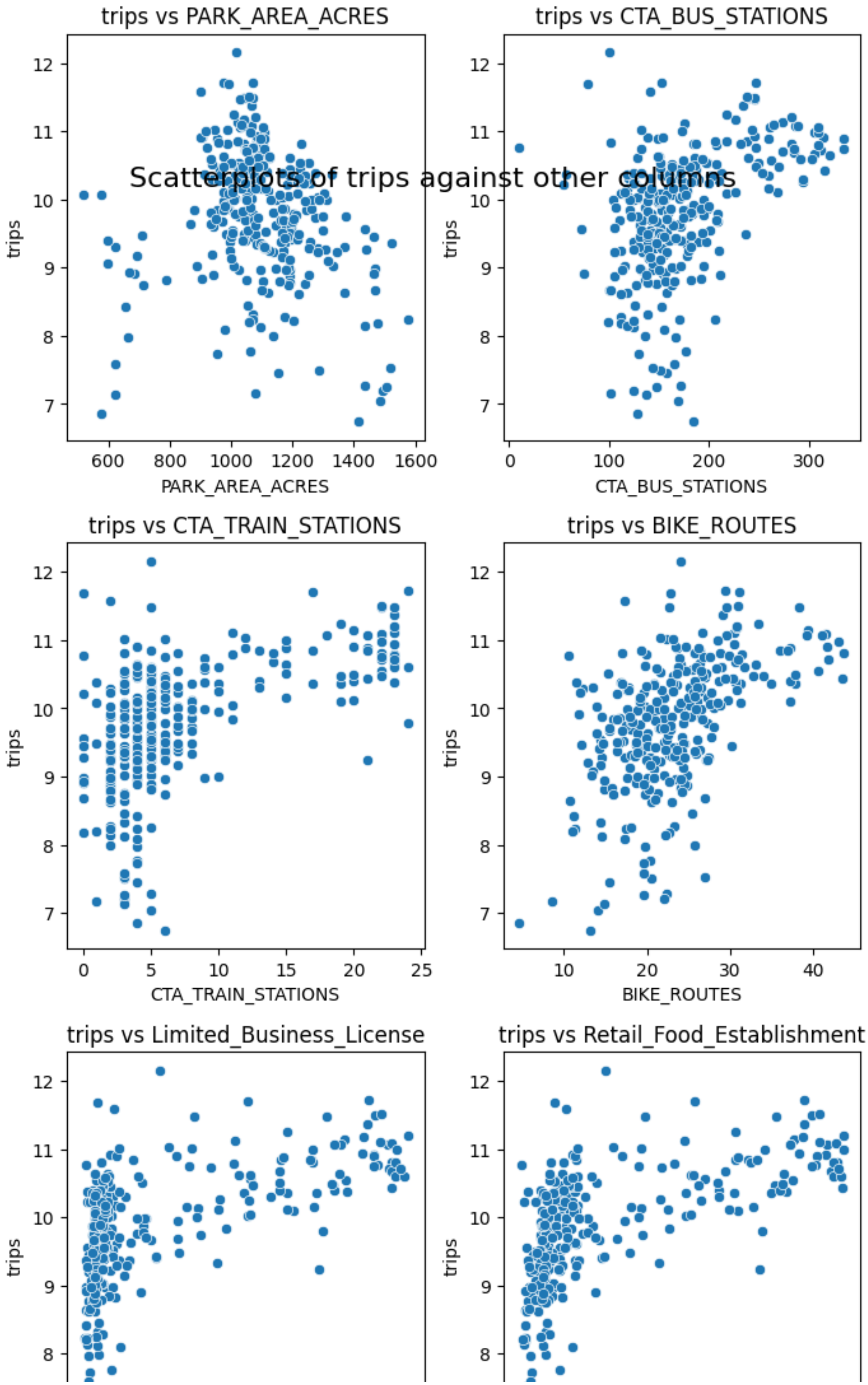
In [15]:
```python
corr_with_trips = correlation_matrix['trips'].sort_values(ascending=False)
print(f"Correlations for {corr_with_trips[:7]}:")
```

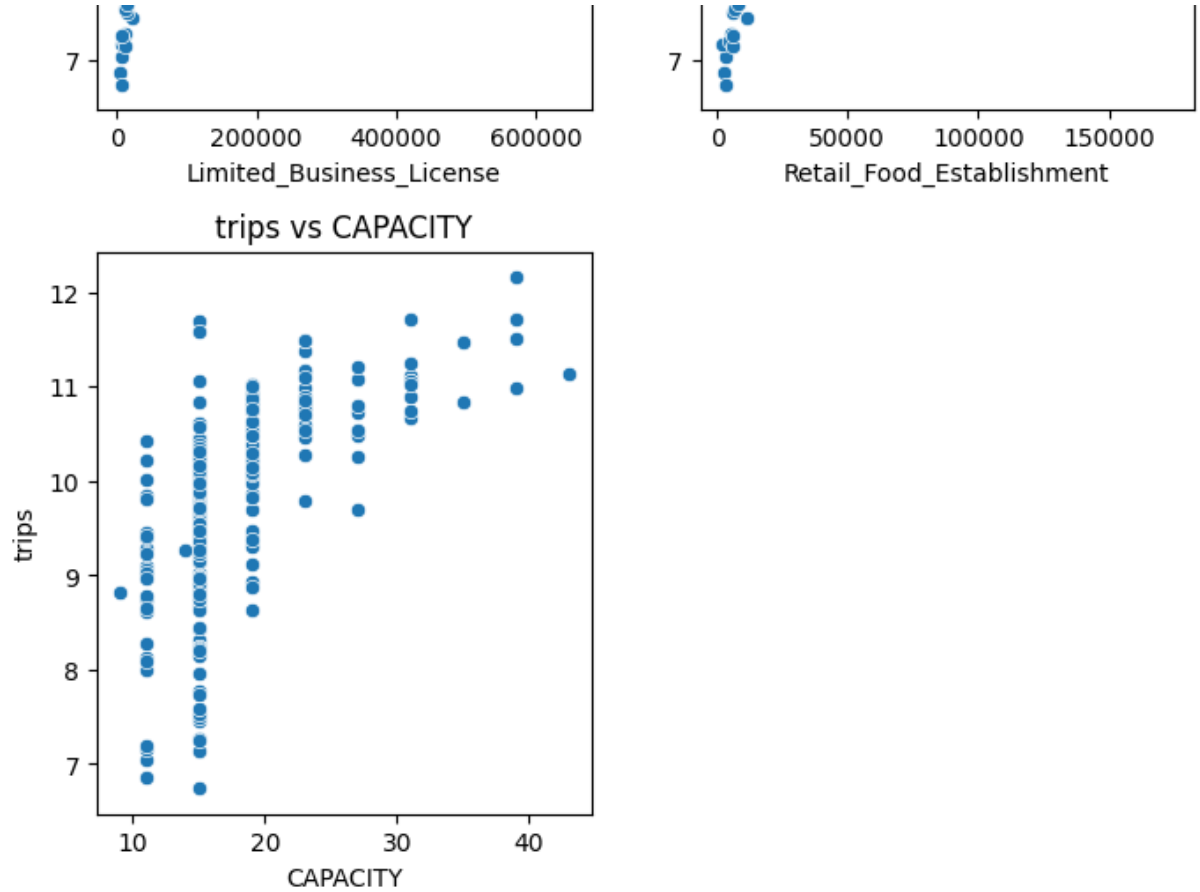
```
Correlations for trips                   1.000000
CAPACITY                    0.594428
Retail_Food_Establishment   0.593184
Limited_Business_License    0.560613
THEFT                       0.551326
DECEPTIVE_PRACTICE          0.528705
CBD                         0.526904
Name: trips, dtype: float64:
```

In [17]:
```python
# corr_with_trips[:7].keys()
```

Create scatterplots

In [19]:
```
# Specify the column to plot against others
target_column = 'trips'  # Replace with your desired column name

# Get the list of other columns
other_columns = [col for col in bike_data_selected.columns if col != target_column]

# Calculate the number of rows needed for subplots
n_rows = (len(other_columns) + 1) // 2  # Round up to the nearest integer

# Create subplots
fig, axes = plt.subplots(n_rows, 2, figsize=(7, 4 * n_rows))
fig.suptitle(f'Scatterplots of {target_column} against other columns', fontsize=16)

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Create scatterplots
for i, column in enumerate(other_columns[:7]):
    sns.scatterplot(data = bike_data_selected, x = column, y = target_column, ax =
    axes[i].set_title(f'{target_column} vs {column}')

# Remove any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```


Scatterplots of trips against other columns
trips vs PARK_AREA_ACRES
trips
PARK_AREA_ACRES
trips vs CTA_BUS_STATIONS
CTA_BUS_STATIONS
trips vs CTA_TRAIN_STATIONS
CTA_TRAIN_STATIONS
trips vs BIKE_ROUTES
BIKE_ROUTES
trips vs Limited_Business_License
trips vs Retail_Food_Establishment



In [ ]:

Create a dataframe where the columns are composites of other predictors/variables

In [20]:
```
bike_composite = pd.DataFrame()
```

In [21]:
```
for key, cols in category_2_crime_dict.items():
    bike_composite[key] = bike_data[ cols  ].mean( axis = 1 )

bike_composite.head()
```

Out[21]:

	violent_crimes	harrassment_crimes	property_crimes	sex_crimes	drug_related_crimes	wl
0	3.782505	1.497866	5.253022	2.402420	2.654134	
1	2.743382	1.098612	4.649653	1.315850	1.791759	
2	3.142549	0.693147	4.303136	1.221850	2.152033	
3	3.258730	0.693147	4.507747	1.416342	2.402011	
4	3.305614	0.000000	5.043972	1.229694	1.985146	

In [22]:
```
for key, cols in category_2_area_dict.items():
    bike_composite[key] = bike_data[ cols  ].mean( axis = 1 )

bike_composite.head()
```

Out[22]:

	violent_crimes	harrassment_crimes	property_crimes	sex_crimes	drug_related_crimes	wl
0	3.782505	1.497866	5.253022	2.402420	2.654134	
1	2.743382	1.098612	4.649653	1.315850	1.791759	
2	3.142549	0.693147	4.303136	1.221850	2.152033	
3	3.258730	0.693147	4.507747	1.416342	2.402011	
4	3.305614	0.000000	5.043972	1.229694	1.985146	

In [ ]:

Given the widely different scales for the variables, let's standardize them.

In [31]:

```
import numpy as np
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
# bike_composite['trips'] = bike_data['trips']
bike_composite_normalized = pd.DataFrame(scaler.fit_transform(bike_composite), colu
bike_composite_normalized.head()
```

Out[31]:

	violent_crimes	harrassment_crimes	property_crimes	sex_crimes	drug_related_crimes	wl
0	1.239313	1.028713	1.175994	2.128938	1.034912	
1	-0.582588	0.369963	0.074633	0.024254	-0.533186	
2	0.117273	-0.299036	-0.557883	-0.157825	0.121916	
3	0.320976	-0.299036	-0.184395	0.218906	0.576464	
4	0.403177	-1.442696	0.794405	-0.142631	-0.181542	

In [ ]:

Variable selection process and justifications

In [ ]:

In [24]:

```
import numpy as np
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

X = bike_composite_normalized.select_dtypes(include=[np.number])  # Select numeric
# X = bike_composite.select_dtypes(include=[np.number])  # Select numeric columns
```

```
# Add a constant term to the predictors
X = add_constant(X)

# Calculate VIF for each predictor
vif_data = pd.DataFrame()
vif_data["Variable"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1]

# Sort the dataframe by VIF in descending order
vif_data = vif_data.sort_values("VIF", ascending=False)

# Display the VIF results
print(vif_data[:-1])
```

```
                            Variable       VIF
6         white_collar_financial_crimes  8.753918
1                        violent_crimes  8.324619
12  transportation_accessibility_index  7.916262
11      central_business_district_index  7.771529
3                       property_crimes  6.422962
8                   public_order_crimes  5.840298
4                            sex_crimes  5.241250
5                   drug_related_crimes  4.836818
10                 miscellaneous_crimes  4.186070
7                      weapons_offenses  3.786115
2                   harrassment_crimes  3.019494
9               crimes_against_children  2.041083
13    urban_suburban_spectrum_indicator  1.439726
14      socioeconomic_status_indicator  1.076220
```

The following have a higher than 5 VIF:

- white_collar_financial_crimes,
- violent_crimes,
- transportation_accessibility_index,
- central_business_district_index,
- property_crimes,
- public_order_crimes,

indicating that these have relatively higher correlation with the other predictors/variables.

And the following variables were found to have lower/moderate VIF:

- sex_crimes
- drug_related_crimes
- miscellaneous_crimes
- weapons_offenses
- harrassment_crimes
- crimes_against_children
- urban_suburban_spectrum_indicator
- socioeconomic_status_indicator

In [ ]:

On encountering predictors with reasonably high VIFs, one of the next steps is to analyse the context-specific relationship between the predictors to try to decide whether to include them in the model, or drop them altogether.

However, before doing so, let's try to build a simple OLS and view the significance of the different predictors involved.

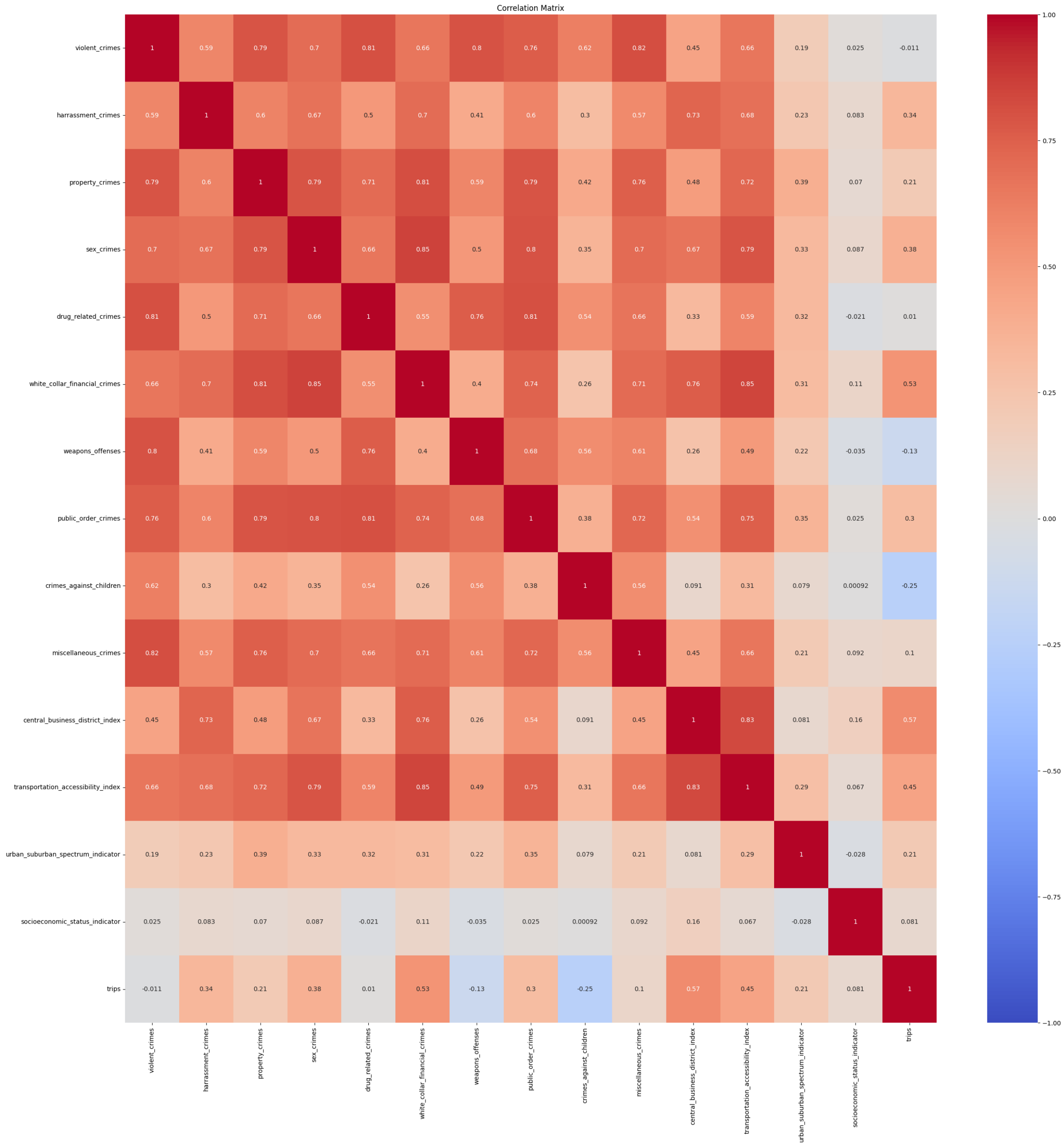
In [26]:

```
scaler = StandardScaler()
bike_composite_normalized['trips'] = pd.DataFrame(scaler.fit_transform(bike_data[ [
```

We can see this through a visualisation of the correlation matrix as well.

In [27]:

```
correlation_matrix = bike_composite_normalized.corr()

# Visualize the correlation matrix
plt.figure(figsize=(29, 29))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, cente
plt.title('Correlation Matrix')
plt.show()
```



In [ ]:

Evaluating correlation between the newly formed composite variables and trips

In [28]:
```
corr_with_trips = correlation_matrix['trips'].sort_values(ascending=False)
print(f"Correlations for {corr_with_trips}:")
```

```
Correlations for trips                                     1.000000
central_business_district_index       0.567670
white_collar_financial_crimes         0.528705
transportation_accessibility_index    0.450481
sex_crimes                            0.377540
harrassment_crimes                    0.339513
public_order_crimes                   0.297616
urban_suburban_spectrum_indicator     0.214696
property_crimes                       0.209129
miscellaneous_crimes                  0.103838
socioeconomic_status_indicator        0.081194
drug_related_crimes                   0.010194
violent_crimes                       -0.011469
weapons_offenses                     -0.132591
crimes_against_children              -0.246415
Name: trips, dtype: float64:
```

In [ ]:

None of the variables have too high a correlation with trips. However,

- central_business_district_index
- white_collar_financial_crimes
- transportation_accessibility_index

have relatively higher correlations.

In [ ]:

Let's also run a drop1 test and try to interpret the values in conjunction with VIF and correlation values.

Deciding on which variables to include

In [34]:

```
# Single term deletions

# Model:
# trips ~ violent_crimes + harrassment_crimes + property_crimes +
#     sex_crimes + drug_related_crimes + white_collar_financial_crimes +
#     weapons_offenses + public_order_crimes + crimes_against_children +
#     miscellaneous_crimes + central_business_district_index +
#     transportation_accessibility_index + urban_suburban_spectrum_indicator +
#     socioeconomic_status_indicator
#                                   Df Sum of Sq    RSS     AIC F value    Pr(>F)
# <none>                                         118.12 -249.63
# violent_crimes                     1    3.7182 121.83 -242.33  8.9715  0.002983
# harrassment_crimes                 1    0.0873 118.20 -251.41  0.2106  0.646669
# property_crimes                    1    3.0519 121.17 -243.98  7.3639  0.007060
# sex_crimes                         1    0.4097 118.53 -250.59  0.9885  0.320963
# drug_related_crimes                1    0.5702 118.69 -250.19  1.3758  0.241794
# white_collar_financial_crimes      1   17.9517 136.07 -209.19 43.3149 2.231e-10
# weapons_offenses                   1    1.3850 119.50 -248.13  3.3419  0.068584
# public_order_crimes                1    7.8885 126.01 -232.24 19.0338 1.797e-05
```

```
# crimes_against_children           1    0.8989 119.02 -249.36  2.1690  0.141926
# miscellaneous_crimes              1    0.7709 118.89 -249.68  1.8600  0.173697
# central_business_district_index   1    3.1478 121.27 -243.74  7.5952  0.006230
# transportation_accessibility_index 1   0.0356 118.15 -251.54  0.0860  0.769587
# urban_suburban_spectrum_indicator  1   3.0912 121.21 -243.88  7.4586  0.006707
# socioeconomic_status_indicator    1    0.0223 118.14 -251.57  0.0538  0.816725
# ---
# Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Among the significant predictors:

- violent_crimes
- property_crimes
- white_collar_financial_crimes
- public_order_crimes
- central_business_district_index
- urban_suburban_spectrum_indicator

the following had displayed high VIF.

- white_collar_financial_crimes
- violent_crimes
- central_business_district_index
- property_crimes

REASONING FOR INCLUSION OF ALL SIGNIFICANT VARIABLEs

Thinking through which variables could be dropped for further analyses.

white_collar_financial_crimes:- central_business_district_index can be said to be highly correlated with incidences of white_collar_financial_crimes. There could be a fork relationship here, as both have moderate correlation with trips. Hence, we will retain white_collar_financial_crimes.

violent_crimes and property_crimes - These 2 can be said to be correlated given that the types of crimes in both of these require almost equal amounts of disregard for other humans. Both these have correlation with transportation_accessibility_indicator too, with both crimes to be slightly more prevalent in areas that rank higher on this index. transportation_accessibility_indicator itself has moderate correlation with trips.

There might be a fork relationship here - violent_crimes and property_crimes being more prevalent due to some other factos in better connected areas, and all of them having an impact on trips. So, we would include both of these in the model.

A similar remark can be made about central_business_district_index (w) in conjunction with transportation_accessibility_indicator.

Hence, let's proceed with all the variables in the model despite the multi-collinearity.

In [37]:
```
y = bike_composite_normalized['trips']
```

In [38]:
```
np.random.seed(0)
```

In [39]:
```
from sklearn.linear_model import Ridge, RidgeCV
from sklearn.model_selection import RepeatedKFold, cross_val_score
import numpy as np
```

In [40]:
```
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
```

In [41]:
```
alphas = np.logspace(-3, 3, 100)  # Example range
```

In [42]:
```
model = RidgeCV(alphas=alphas, cv=cv, scoring='neg_mean_squared_error')
model.fit(
    X[
    ["violent_crimes", "property_crimes", "white_collar_financial_crimes",\
     "public_order_crimes", "central_business_district_index", "urban_suburban_spec
    y)
```

Out[42]:

▾ RidgeCV

```
RidgeCV(alphas=array([1.00000000e-03, 1.14975700e-03, 1.32194115e-03, 1.51
991108e-03,
       1.74752840e-03, 2.00923300e-03, 2.31012970e-03, 2.65608778e-03,
       3.05385551e-03, 3.51119173e-03, 4.03701726e-03, 4.64158883e-03,
       5.33669923e-03, 6.13590727e-03, 7.05480231e-03, 8.11130831e-03,
       9.32603347e-03, 1.07226722e-02, 1.23284674e-02, 1.41747416e-02,
       1.62975083e-02, 1.87381742e-0...
       7.05480231e+01, 8.11130831e+01, 9.32603347e+01, 1.07226722e+02,
       1.23284674e+02, 1.41747416e+02, 1.62975083e+02, 1.87381742e+02,
```

In [43]:
```
best_alpha = model.alpha_
best_alpha
```

Out[43]:
```
2.848035868435802
```

In [44]:
```
final_model = Ridge(alpha=best_alpha)
final_model.fit(X[["violent_crimes", "property_crimes", "white_collar_financial_cri
                   "public_order_crimes", "central_business_district_index", "urban
```

Out[44]:

```
▾          Ridge                    i ?

Ridge(alpha=2.848035868435802)
```

In [45]:

```
scores = cross_val_score(final_model, X[["violent_crimes", "property_crimes", "whit
                                        "public_order_crimes", "central_business_d
```

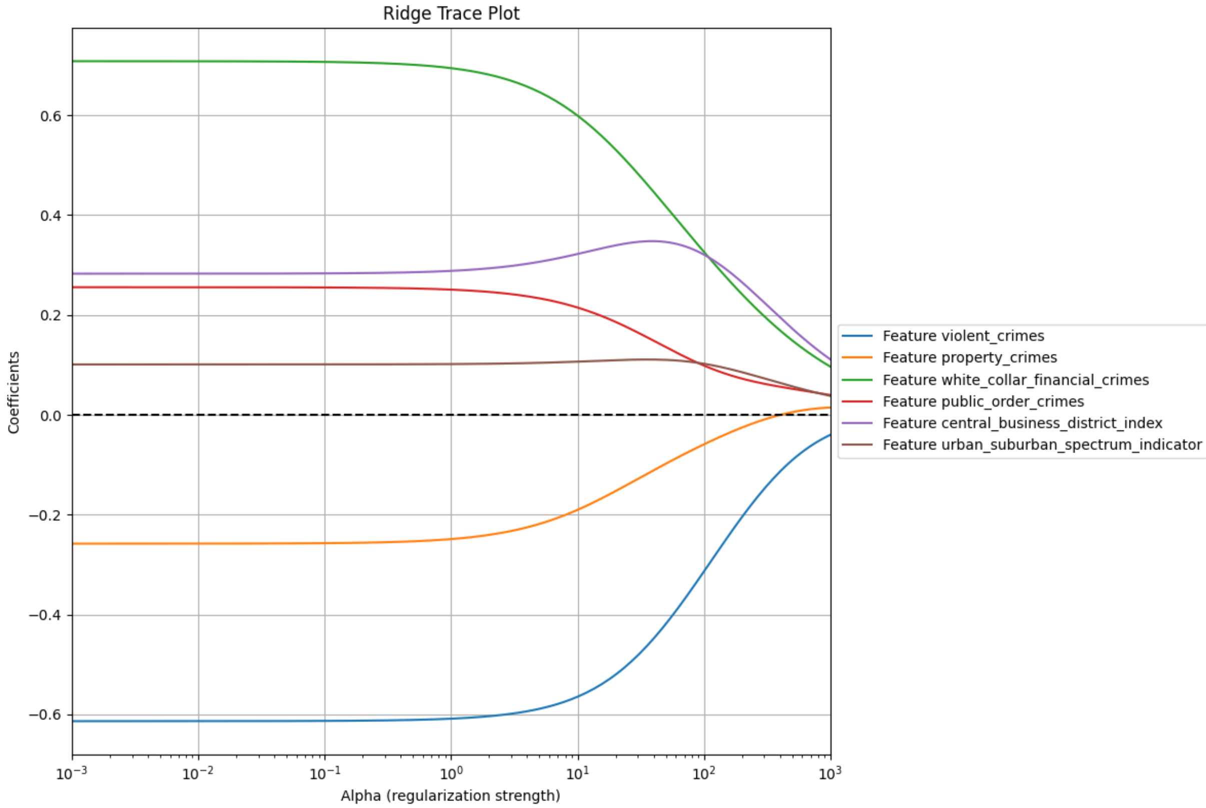
In [ ]:

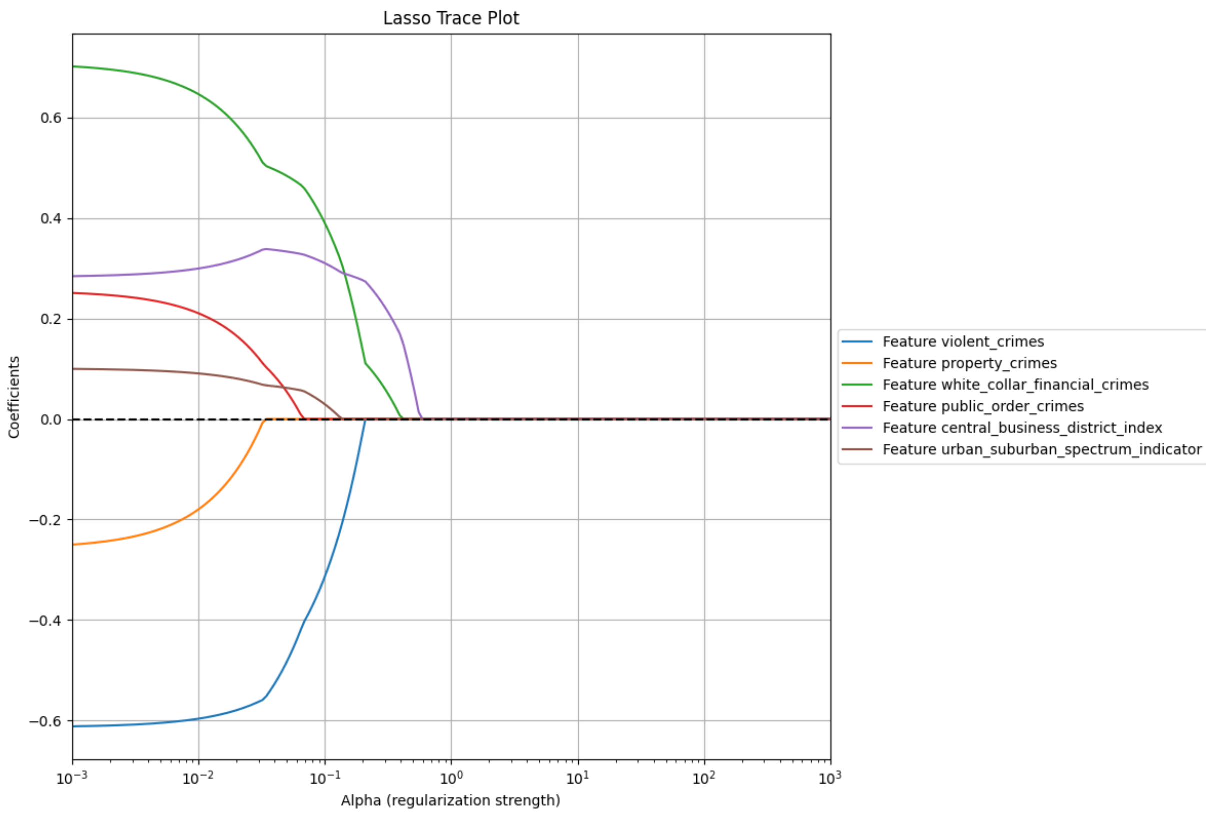
Building the ridge and lasso trace plots

In [47]:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_regression

# # Generate sample data
# X, y = make_regression(n_samples=100, n_features=10, noise=0.1, random_state=42)
cols = X[["violent_crimes", "property_crimes", "white_collar_financial_crimes", "pu
          "central_business_district_index", "urban_suburban_spectrum_indicator"]

# # Standardize features
# scaler = StandardScaler()
# X_scaled = scaler.fit_transform(X)

# Define range of alpha values
alphas = np.logspace(-3, 3, 200)

# Function to fit model and get coefficients
def get_coefs(model_class, alphas, X, y):
    coefs = []
    for alpha in alphas:
        model = model_class(alpha=alpha)
        model.fit(X, y)
        coefs.append(model.coef_)
    return np.array(coefs)

# Get coefficients for Ridge and Lasso
ridge_coefs = get_coefs(Ridge, alphas, X[["violent_crimes", "property_crimes", "whi
                                         "public_order_crimes", "central_business_
lasso_coefs = get_coefs(Lasso, alphas, X[["violent_crimes", "property_crimes", "whi
                                         "public_order_crimes", "central_business_

# Function to plot trace
def plot_trace(ax, alphas, coefs, title):
    for i in range(X[["violent_crimes", "property_crimes", "white_collar_financial_
                      "public_order_crimes", "central_business_district_index", "ur
        ax.semilogx(alphas, coefs[:, i], label=f'Feature {cols[i]}')
    ax.set_xlabel('Alpha (regularization strength)')
    ax.set_ylabel('Coefficients')
    ax.set_title(title)
    ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

```
    ax.grid(True)
    ax.axhline(y=0, color='k', linestyle='--')
    ax.set_xlim(alphas.min(), alphas.max())

# Create subplots
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 16))

# Plot Ridge trace
plot_trace(ax1, alphas, ridge_coefs, 'Ridge Trace Plot')

# Plot Lasso trace
plot_trace(ax2, alphas, lasso_coefs, 'Lasso Trace Plot')

plt.tight_layout()
plt.show()
```





In [ ]:

Based on observations from all the analyses:

Our conclusion about property_crimes affecting trips is unstable.

Our conclusions for the following being important predictors can be said to be reasonable -

- Violent crimes,
- white collar financial crimes,
- public order crimes,
- property crimes,
- central business district index
- urban suburban spectrum indicator

However, here are some rather unintuitive observations:

- 'white collar financial crimes' affecting trips instead of a variable like 'sex crimes'
- 'harrassment crimes' and 'transportation_accessibility_index' not being significant

In [ ]:

In [ ]:

In [ ]:

In [ ]: