STEP 1 - IMPORTING NECESSARY LIBRARIES

```
import os
import zipfile
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

STEP 2 - EXTRACTING FILES

```
# Define path to your ZIP file and extraction directory
zip_path = "/content/Florida Keys Coral Reef Evaluation Dataset.zip"
extract_dir = "/content/florida_keys_data"

# Extract files
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

# Check contents
extracted = os.listdir(extract_dir)
print("Extracted folders/files:", extracted)
```

```
Extracted folders/files: ['CREMP_CSV_files']
```

STEP 3 - IMPORT ALL CSV FILES

```
import glob

# Get all CSVs from CREMP_CSV_files folder
csv_folder = os.path.join(extract_dir, "CREMP_CSV_files")
csv_files = glob.glob(os.path.join(csv_folder, "*.csv"))

print(f"Found {len(csv_files)} CSV files.")

# Load all into a single DataFrame (optional: filter by shape/columns)
df_list = []
for f in csv_files:
    try:
        temp_df = pd.read_csv(f)
        df_list.append(temp_df)
    except Exception as e:
        print(f"Could not load {f}: {e}")

# Combine all loaded data
combined_df = pd.concat(df_list, ignore_index=True)
print("Combined shape:", combined_df.shape)
```

```
Found 12 CSV files.
Combined shape: (5382438, 130)
```

STEP 4 - CHECKING MISSSING VALUES

```
# Check the first few rows and columns
print(df.head())

# Check for missing values in each column
print(df.isnull().sum())
```

```
      Year Subregion  total_coral_cover  Subregion_encoded
0  2011.0        LK                0.4                  0
1  2011.0        LK                0.1                  0
2  2011.0        LK                0.5                  0
3  2011.0        LK                0.7                  0
4  2011.0        UK                1.5                  2
Year                 0
Subregion            0
total_coral_cover    0
Subregion_encoded    0
dtype: int64
```

STEP 5 - DROPPING MISSING VALUES IN YEAR AND TOTAL CORAL COVER COLUMNS

```
df.dropna(subset=['Year', 'total_coral_cover'], inplace=True)
```

STEP 6 - CONVERT COLUMNS TO APPROPRIATE DATA TYPES

```
# Ensure 'Year' is an integer
df['Year'] = df['Year'].astype(int)

# Ensure 'Subregion' is treated as categorical
df['Subregion'] = df['Subregion'].astype('category')
```

STEP 7 - LABEL ENCODING

```
# Label Encoding for 'Subregion' (if you need numerical encoding)
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Subregion'] = label_encoder.fit_transform(df['Subregion'])
```

STEP 8 - CHECK COLUMN NAMES

```
print("Columns in combined_df:\n", combined_df.columns.tolist())
```

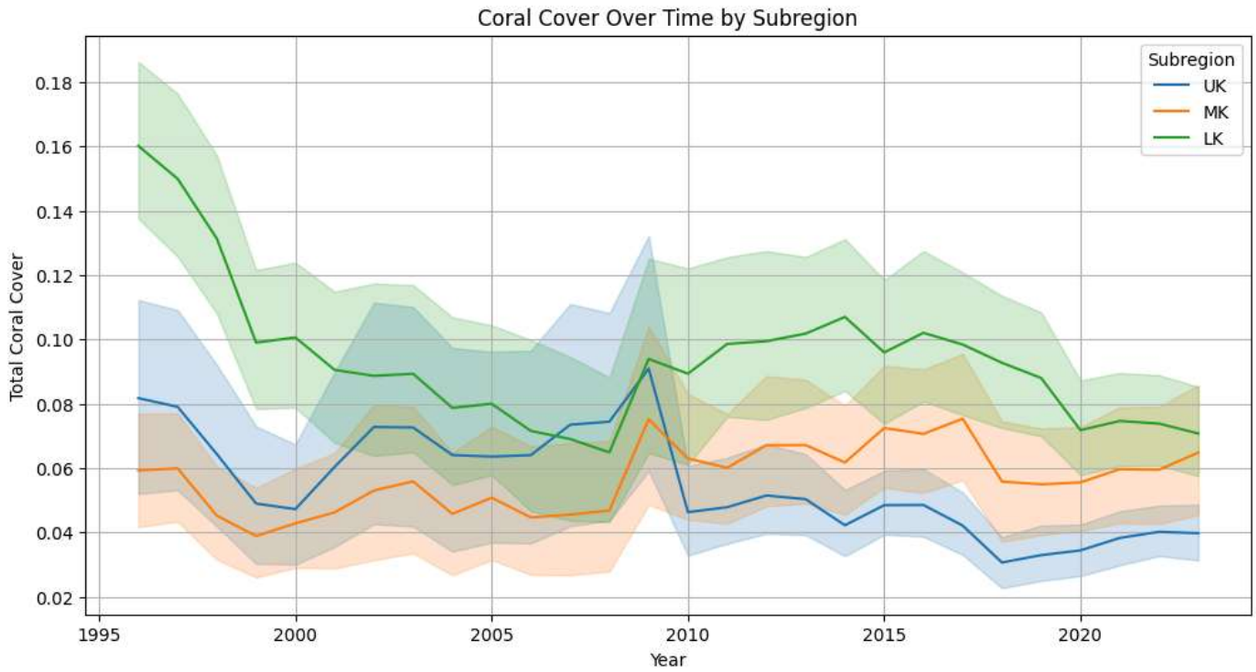
```
Columns in combined_df:
 ['Year', 'Date', 'Subregion', 'Habitat', 'SiteID', 'Site_name', 'StationID', 'Acropora_cervicornis', 'Acropora_palmat
```

STEP 9 - CHANGING COLUMN NAMES

```
# Replace 'total_coral_cover' with the actual column 'Stony_coral'
relevant_columns = [
    'Year', 'Subregion', 'Habitat', 'SiteID', 'StationID',
    'Stony_coral', 'species_richness', 'latDD', 'lonDD'
]

# Filter the actual existing columns
available_columns = [col for col in relevant_columns if col in combined_df.columns]
df = combined_df[available_columns].copy()

# Rename for consistency
df.rename(columns={'Stony_coral': 'total_coral_cover'}, inplace=True)

# Drop NA and convert Year to int
df.dropna(subset=['Year', 'total_coral_cover'], inplace=True)
df['Year'] = df['Year'].astype(int)
```

STEP 10 - Trends Over Time: Plotted coral cover and species richness over time (this was based on available columns)

```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='Year', y='total_coral_cover', hue='Subregion')
plt.title("Coral Cover Over Time by Subregion")
plt.ylabel("Total Coral Cover")
plt.xlabel("Year")
plt.grid(True)
plt.show()
```



STEP 11 - CHECKING LAT LONG MISSING VALUES

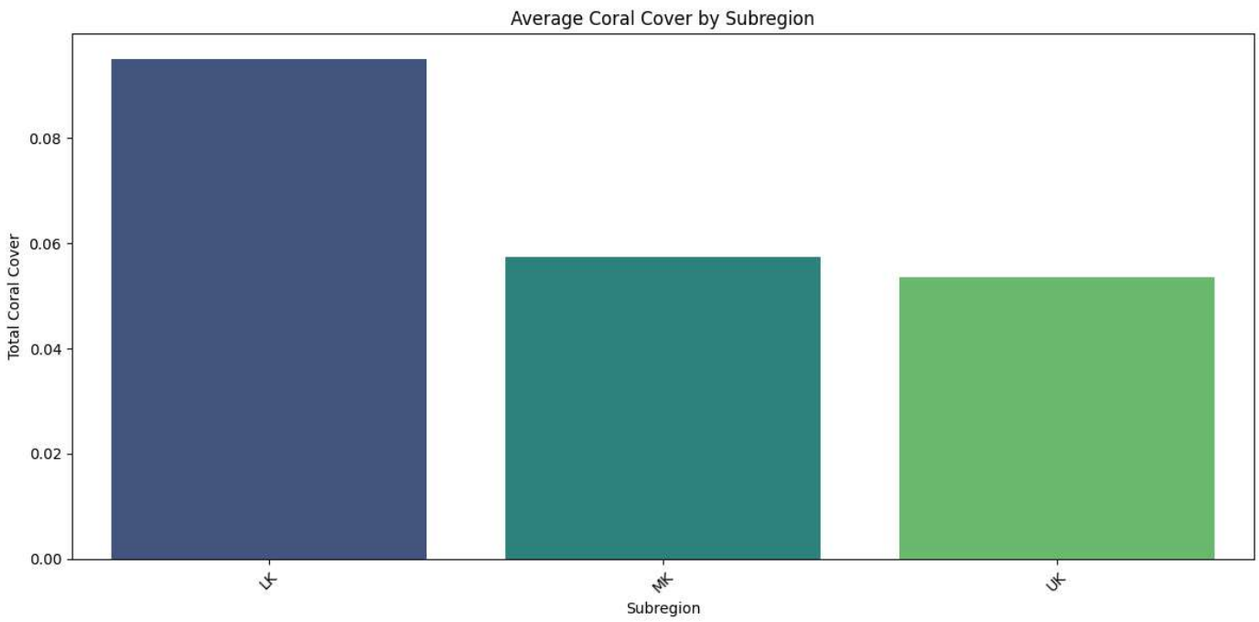
```
print(df[['latDD', 'lonDD', 'total_coral_cover']].dropna().head())
print("latDD null count:", df['latDD'].isna().sum())
print("lonDD null count:", df['lonDD'].isna().sum())
```

```
Empty DataFrame
Columns: [latDD, lonDD, total_coral_cover]
Index: []
latDD null count: 3918
lonDD null count: 3918
```

STEP 12 - Heatmap: Intended to visualize regional differences in coral cover, but we ran into issues due to missing lat/lon data

```
# Average coral cover by subregion
region_avg = df.dropna(subset=['total_coral_cover']).groupby('Subregion')['total_coral_cover'].mean().reset_index()

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(data=region_avg, x='Subregion', y='total_coral_cover', palette='viridis')
plt.xticks(rotation=45)
plt.title("Average Coral Cover by Subregion")
plt.ylabel("Total Coral Cover")
plt.xlabel("Subregion")
plt.tight_layout()
plt.show()
```

```
<ipython-input-8-8c6c38193550>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hu

  sns.barplot(data=region_avg, x='Subregion', y='total_coral_cover', palette='viridis')
```



STEP 13 - FEATURE SELECTION

```
# Select all stony coral columns (you can expand this list based on your dataset)
coral_species_cols = [
    'Acropora_cervicornis', 'Acropora_palmata', 'Acropora_prolifera',
    'Agaricia_fragilis', 'Agaricia_lamarcki', 'Colpophyllia_natans',
    'Diploria_labyrinthiformis', 'Montastraea_cavernosa',
    'Orbicella_faveolata', 'Porites_astreoides', 'Porites_porites',
    'Siderastrea_radians', 'Siderastrea_siderea'
]

# Create 'total_coral_cover' by summing selected columns (ignore NaNs)
combined_df['total_coral_cover'] = combined_df[coral_species_cols].sum(axis=1, skipna=True)
```

STEP 14 - MODEL FITTING AND EVALUATION

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
import numpy as np

# Now select the required columns
df = combined_df[['Year', 'Subregion', 'total_coral_cover']].copy()
df.dropna(inplace=True)

# Encode Subregion
le = LabelEncoder()
df['Subregion_encoded'] = le.fit_transform(df['Subregion'])

# Features and target
X = df[['Year', 'Subregion_encoded']]
y = df['total_coral_cover']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit model (with reduced complexity to avoid crash)
model = RandomForestRegressor(n_estimators=30, max_depth=10, random_state=42)
model.fit(X_train, y_train)

# Evaluate
preds = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE:", rmse)
```

```
RMSE: 5764.39585797676
```

STEP 15 - VISUALLIZING ACTUAL VS PREDICTED VALUES

```
import matplotlib.pyplot as plt

# Plot actual vs predicted
plt.figure(figsize=(10, 6))
plt.scatter(y_test, preds, alpha=0.7, color='b')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.xlabel('Actual Coral Cover')
plt.ylabel('Predicted Coral Cover')
plt.title('Actual vs Predicted Coral Cover')
plt.show()
```