Step 1: Import All Files into the Dataset First, we need to import necessary libraries and read the files into pandas DataFrames

```
import pandas as pd
import glob

# Define the file paths
file_paths = ['/content/Delhi 1 Flights.xlsx', '/content/Delhi 2 Flights_.xlsx', '/content/Hyderabad 1 flights.csv',
              '/content/Kolkata Flight.csv', '/content/Bangalore Flight.csv']

# Reading Excel files for Delhi
delhi_df1 = pd.read_excel('/content/Delhi 1 Flights.xlsx')
delhi_df2 = pd.read_excel('/content/Delhi 2 Flights_.xlsx')

# Reading CSV files
hyderabad_df = pd.read_csv('/content/Hyderabad 1 flights.csv')
kolkata_df = pd.read_csv('/content/Kolkata Flight.csv')
bangalore_df = pd.read_csv('/content/Bangalore Flight.csv')

# Concatenating Delhi data
delhi_df = pd.concat([delhi_df1, delhi_df2], ignore_index=True)
```

Step 2: Concatenate Datasets Next, we concatenate all the datasets into a single DataFrame

---

```
# Concatenate all city dataframes into one
all_flights_df = pd.concat([delhi_df, hyderabad_df, kolkata_df, bangalore_df], ignore_index=True)
```

Step 3: Perform Data Cleaning

(a) Check for Null Values and Imputation

```
# Checking for null values
null_values = all_flights_df.isnull().sum()

# Print null values to determine the extent
print("Null values in each column:\n", null_values)

# Choosing imputation method - let's assume mean for numeric and mode for categorical
for column in all_flights_df.columns:
    if all_flights_df[column].isnull().sum() > 0:
        if all_flights_df[column].dtype == 'object':
            all_flights_df[column].fillna(all_flights_df[column].mode()[0], inplace=True)
        else:
            all_flights_df[column].fillna(all_flights_df[column].mean(), inplace=True)
```

```
Null values in each column:
 Airline                  23189
Flight No.                4986
Source                    4986
Departure                 4986
No. of stops              4986
Arrival                   4986
Destination               4986
Ticket Class              4986
Flight Duration (hrs)     4986
Days left                 4986
Price                     4986
ï»¿Airline              169482
dtype: int64
```

(b) Identify and Handle Outliers To identify and handle outliers, we can use the IQR method.

```
# Using IQR to handle outliers for numerical columns
for column in all_flights_df.select_dtypes(include=['float64', 'int64']).columns:
    Q1 = all_flights_df[column].quantile(0.25)
    Q3 = all_flights_df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filtering out outliers
    all_flights_df = all_flights_df[(all_flights_df[column] >= lower_bound) & (all_flights_df[column] <= upper_bound)]
```

(c) Sanity Check We will check the range of key parameters to ensure data integrity.

```
# Sanity check: Print min and max values of key columns
print("Price: min =", all_flights_df['Price'].min(), ", max =", all_flights_df['Price'].max())
print("Days left: min =", all_flights_df['Days left'].min(), ", max =", all_flights_df['Days left'].max())
```

```
Price: min = 1603.0 , max = 62270.0
Days left: min = 1.0 , max = 49.0
```

(d) Format and Transform Data If necessary, format and transform the data, such as converting time columns to appropriate datetime formats.

```
early_morning_rows = all_flights_df[all_flights_df['Departure'] == 'Early_Morning']
print(early_morning_rows)
```

```
           Airline Flight No.     Source      Departure No. of stops  \
0         AirAsia     I5-548      Delhi  Early_Morning            1
2         AirAsia     I5-787      Delhi  Early_Morning            1
6         AirAsia     I5-764      Delhi  Early_Morning            1
7         AirAsia     I5-791      Delhi  Early_Morning            1
9         AirAsia     I5-548      Delhi  Early_Morning            1
...           ...        ...        ...            ...          ...
187663    Vistara     UK-808  Bangalore  Early_Morning            1
187666    Vistara     UK-810  Bangalore  Early_Morning            1
187668    Vistara     UK-808  Bangalore  Early_Morning            1
187670    Vistara     UK-810  Bangalore  Early_Morning            1
187679  Air_India     AI-804  Bangalore  Early_Morning            1

           Arrival Destination Ticket Class  Flight Duration (hrs)  Days left  \
0       Afternoon   Bangalore      Economy                   7.58       41.0
2       Afternoon   Bangalore      Economy                   8.17       41.0
6       Afternoon   Bangalore      Economy                  10.33       41.0
7       Afternoon   Bangalore      Economy                  10.92       41.0
9         Evening   Bangalore      Economy                  11.50       41.0
...           ...         ...          ...                    ...        ...
187663      Night     Chennai     Business                  12.25       49.0
187666      Night     Chennai     Business                  13.25       49.0
187668      Night     Chennai     Business                  14.75       49.0
187670      Night     Chennai     Business                  15.75       49.0
187679      Night     Chennai     Business                  17.42       49.0

           Price ï»¿Airline
0        3090.0    Vistara
2        3090.0    Vistara
6        3090.0    Vistara
7        3090.0    Vistara
9        3090.0    Vistara
...         ...        ...
187663  60396.0    Vistara
187666  60396.0    Vistara
187668  60396.0    Vistara
187670  60396.0    Vistara
187679  60396.0    Vistara

[36797 rows x 12 columns]
```

```
# Function to clean time data
def clean_time_column(time_series):
    # If the value is not a valid time, replace it with NaT
    cleaned_times = pd.to_datetime(time_series, errors='coerce', format='%H:%M')
    # Replace NaT with a default time, here using '00:00'
    return cleaned_times.fillna(pd.to_datetime('00:00').time())
```

```
# Apply the function to the 'Departure' and 'Arrival' columns
all_flights_df['Departure'] = clean_time_column(all_flights_df['Departure'])
all_flights_df['Arrival'] = clean_time_column(all_flights_df['Arrival'])

# Further transformation if needed
# For example, ensuring 'Ticket Class' is categorized
all_flights_df['Ticket Class'] = all_flights_df['Ticket Class'].astype('category')
```

Step 4: Save the Cleaned Dataset

```
# Save the cleaned DataFrame to a CSV file
all_flights_df.to_csv('cleaned_flights_data.csv', index=False)
```

```
Start coding or generate with AI.
```