In [54]:
```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
import ipywidgets as widgets
import io
import matplotlib.pyplot as plt
import seaborn as sns
import itertools

upload = widgets.FileUpload(accept='.csv,.xlsx,.json', multiple=False)
display(upload)
def process_file(upload):

    uploaded_file = next(iter(upload.value))
    file_name = uploaded_file['name']
    content=uploaded_file['content']
    file_data = io.BytesIO(content)
    try:

            if file_name.endswith('.csv'):
                df = pd.read_csv(file_data)
            elif file_name.endswith('.json'):
                df = pd.read_json(file_data)
            elif file_name.endswith('.xlsx'):
                df = pd.read_excel(file_data, engine='openpyxl')
            else:
                print('unsupported file format')


    except Exception as e:
            print('error loading file')
    return df


analyze_button = widgets.Button(description="Process File")
output = widgets.Output()

def on_button_click(b):
    with output:
        output.clear_output()
        data=process_file(upload)
        print('data analysis')
        print('data info:')
        numeric= data.select_dtypes(include=[np.number]).columns
        categoric=data.select_dtypes(exclude=[np.number]).columns
        print('number of numeric columns:',len(numeric))
        print(' number of categorical columns:',len(categoric))
        print(data.info())
        print(' '.join(['_' for i in range(100)]))
        print('number of missing values in each column:')
        print(data.isnull().sum())
        print(' '.join(['_' for i in range(100)]))
        print('treating the missing values:')
        k='true'
        if k=='true':
            columns_to_drop=[]
            for i in data.columns:
                if data[i].isnull().sum() >= (1/4) * len(data):
                   columns_to_drop.append(i)
                else:
                    if pd.api.types.is_numeric_dtype(data[i]):
                       imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
                    else:
                       imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
                data[i] = imputer.fit_transform(data[[i]]).ravel()
            data=data.drop(columns=columns_to_drop)
            print('updated values:')
            print(data.isnull().sum())
            print(' '.join(['_' for i in range(100)]))
        print('statestical information:')
        print(data.describe())
        print(' '.join(['_' for i in range(100)]))
        print('data visualization:')
        numeric_cols=data.select_dtypes(include=[np.number]).columns
        num_cols=len(numeric_cols)
        if num_cols < 2:
           print("Not enough numeric columns for correlation subplots.")
        plt.figure(figsize=(14,10))
        sns.heatmap(data[numeric_cols].corr(), annot=True, cmap='coolwarm', linewidths=0.5)
        plt.title('Correlation Heatmap')
        plt.show()
        print(' '.join(['_' for i in range(100)]))
        print('pairwise correlation:')
        if num_cols < 2:
            print("Not enough numeric columns for pairwise plots.")

        fig, axes = plt.subplots(nrows=num_cols, ncols=num_cols, figsize=(6*num_cols, 6*num_cols))
        fig.suptitle('pairwise correlation scatter plots', fontsize=16)
        for (i, j) in itertools.product(range(num_cols), range(num_cols)):
            if i == j:
                axes[i, j].hist(data[numeric_cols[i]], bins=20, color='skyblue', edgecolor='black')
                axes[i, j].set_title(f'{numeric_cols[i]} (Histogram)')
            else:
                axes[i, j].scatter(data[numeric_cols[j]], data[numeric_cols[i]], alpha=0.5, c='green')
                axes[i, j].set_xlabel(numeric_cols[j])
                axes[i, j].set_ylabel(numeric_cols[i])
        plt.tight_layout(rect=[0, 0, 1, 0.96])
        plt.show()
p=analyze_button.on_click(on_button_click)
display(analyze_button, output)
```

```
FileUpload(value=(), accept='.csv,.xlsx,.json', description='Upload')
Button(description='Process File', style=ButtonStyle())
Output()
```

In [ ]: