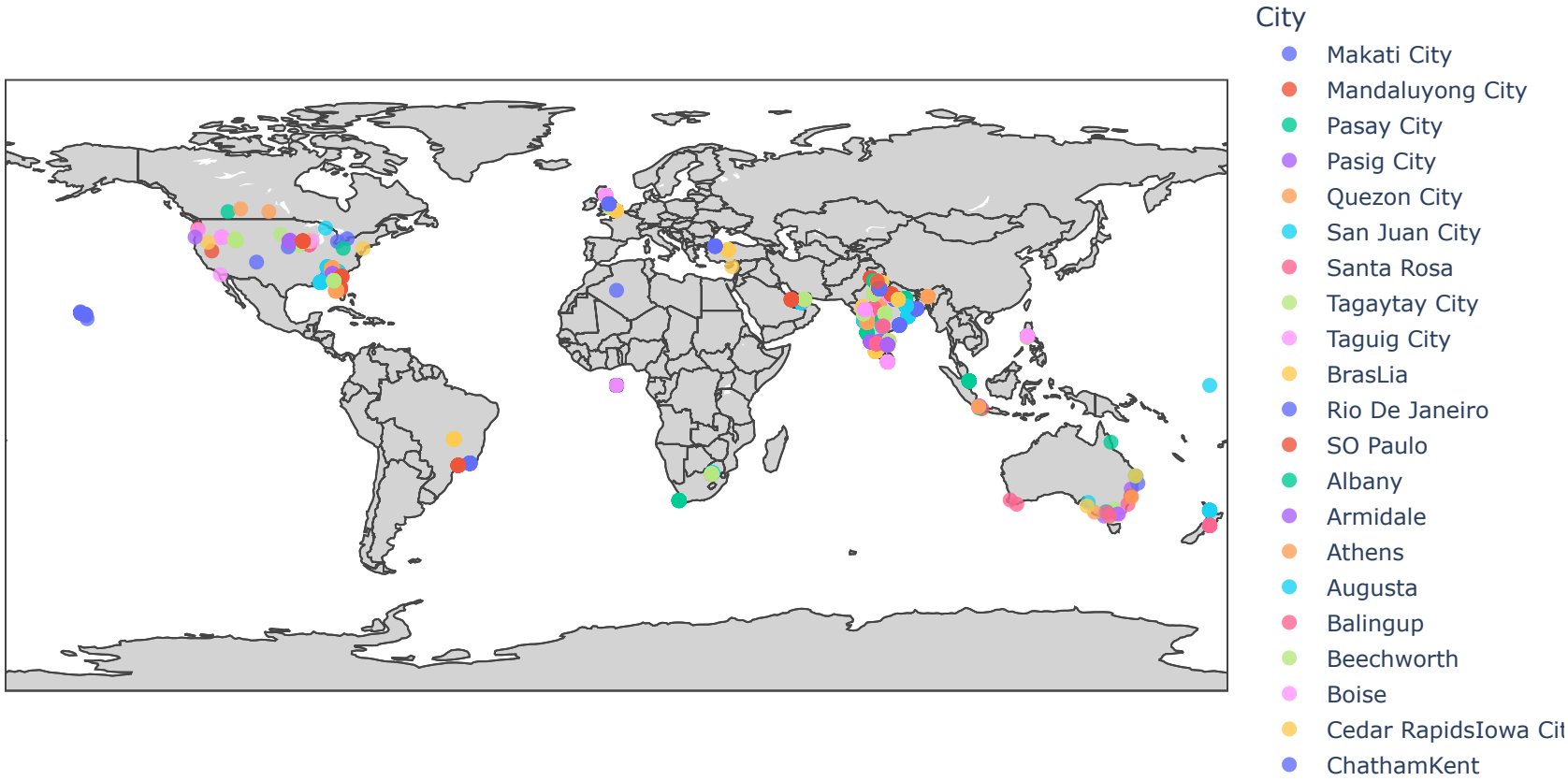
Double-click (or enter) to edit

```
import pandas as pd
import plotly.express as px

# Load the dataset
df = pd.read_csv("/content/Dataset 1.csv")

# Fix column name typo if needed
if 'Restaurnt_name' in df.columns:
    df.rename(columns={'Restaurnt_name': 'Restaurant_name'}, inplace=True)

# Remove duplicate columns if any
df = df.loc[:, ~df.columns.duplicated()]

# Convert Latitude and Longitude to numeric
df['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')
df['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce')

# Drop rows with missing data in required fields
df = df.dropna(subset=['Restaurant_name', 'Latitude', 'Longitude', 'City'])

# Plot the locations
fig = px.scatter_geo(
    df,
    lat='Latitude',
    lon='Longitude',
    hover_name='Restaurant_name',
    color='City',
    title='Restaurant Locations Based on Latitude and Longitude',
    projection='equirectangular',
    opacity=0.8
)

# Optional styling
fig.update_traces(marker=dict(size=8))
fig.update_layout(
```

```
        height=600,
        geo=dict(showland=True, landcolor='lightgray', showcountries=True)
    )

    # Save map as downloadable HTML
    map_path = "/content/restaurant_map.html"
    fig.write_html(map_path)

    # Show the map
    fig.show()
```




```
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans
import plotly.express as px
```

```
# Load dataset
df = pd.read_csv("/content/Dataset 1.csv")

# Clean column names (remove trailing spaces)
df.columns = df.columns.str.strip()

# Remove duplicate columns
df = df.loc[:, ~df.columns.duplicated()]

# Required columns
required_columns = ['Latitude', 'Longitude', 'Restaurant_name', 'City', 'Cuisines', 'Aggregate_rating', 'Price_range']

# Ensure required columns exist and are clean
for col in required_columns:
    if col not in df.columns:
        raise ValueError(f"Missing required column: {col}")
    if col in ['Latitude', 'Longitude', 'Aggregate_rating', 'Price_range']:
        df[col] = pd.to_numeric(df[col], errors='coerce')
    else:
        df[col] = df[col].astype(str)

# Drop rows with missing values in essential columns
df.dropna(subset=required_columns, inplace=True)

# Encode categorical features
df['Cuisines_Encoded'] = LabelEncoder().fit_transform(df['Cuisines'])
df['City_Encoded'] = LabelEncoder().fit_transform(df['City'])

# Prepare features and scale
features = df[['Latitude', 'Longitude', 'Aggregate_rating', 'Price_range', 'Cuisines_Encoded', 'City_Encoded']]
features_scaled = StandardScaler().fit_transform(features)

# Apply KMeans clustering
kmeans = KMeans(n_clusters=6, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(features_scaled)
df['Cluster_Label'] = df['Cluster'].apply(lambda x: f"Cluster {x}")

# Cluster description (for tooltip)
df['Clustering_Basis'] = "Latitude, Longitude, Cuisine, Rating, City, Price"
```

```
# Plot interactive map
fig = px.scatter_geo(
    df,
    lat='Latitude',
    lon='Longitude',
    color='Cluster_Label',
    hover_name='Restaurant_name',
    hover_data={
        'City': True,
        'Cuisines': True,
        'Aggregate_rating': True,
        'Price_range': True,
        'Clustering_Basis': True
    },
    title='Restaurant Clusters Based on Location, Cuisine, Rating, City, and Price Range',
    projection='equirectangular',
    opacity=0.8
)

fig.update_layout(
    height=650,
    geo=dict(showland=True, landcolor="lightgray", showcountries=True)
)

# Show map
fig.show()

# Save map and clustered data
fig.write_html("restaurant_clusters_map.html")
df.to_csv("clustered_restaurants.csv", index=False)
```

Restaurant Clusters Based on Location, Cuisine, Rating, City, and Price Range