renfrewshire_hygiene_analysis

June 2, 2025

1 Renfrewshire Food Hygiene Data Analysis

1.1 Setup

Import the SQLite database for analysis. The database contains business names, post codes, coordinates and hygiene rating. To update the database, use the `data/fetch_data.py` file to update the database from the XML source on the food ratings website: https://ratings.food.gov.uk/open-data

In this case we will be using Renfrewshire data for local business insights. The food hygiene ratings have the following scheme in Scotland:

- **Pass**: means they meet the legal requirements for food hygiene.
- **Improvement Requried**: means the business didn't meet the legal requirements and needs to make improvements.
- **Exempt Premises** means the business has been inspected by a local authority food safety officer, met the pass criteria, but don't meet the criteria to be part of the scheme. These businesses are low-risk to people's health in terms of food safety and you perhaps wouldn't normally think of them as a food business – for example, newsagents, chemist shops or visitor centres selling tins of biscuits.
- **Awaiting Inspection**: means a new business or new business owner is waiting for an inspection.

Further information can be found at this link: https://www.foodstandards.gov.scot/consumers/food-safety/buying-food-eating-out/food-hygiene-information-scheme/about-the-food-hygiene-information-scheme

[51]:
```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
import seaborn as sns
from os import getcwd
import folium
from folium.plugins import HeatMap


# Configure plots
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (15, 9)
```

```
# Connect to the SQLite database
#Run fetch_data.py in the data directory to update the database from the FHIS␣
 ↪website
home_path = getcwd()
print(home_path)
conn = sqlite3.connect("/mnt/d/renfrewshire_business_insights/data/
 ↪renfrewshire_hygiene.db") #adjust path accordingly
```

```
/mnt/d/renfrewshire_business_insights/reports
```

1.2 Overview of Data

Have a quick look at the data to understand the column types and structure.

[52]:
```
#Initial scoping of the SQL database to confirm all is working well
df = pd.read_sql_query("SELECT * FROM establishments;", conn)
#print(df.head(10))
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   FHRSID                     1465 non-null   object
 1   LocalAuthorityBusinessID   1465 non-null   object
 2   BusinessName               1465 non-null   object
 3   BusinessType               1465 non-null   object
 4   BusinessTypeID             1465 non-null   object
 5   AddressLine1               677 non-null    object
 6   AddressLine2               1398 non-null   object
 7   AddressLine3               1397 non-null   object
 8   AddressLine4               1342 non-null   object
 9   PostCode                   1386 non-null   object
 10  RatingValue                1465 non-null   object
 11  RatingKey                  1465 non-null   object
 12  RatingDate                 1441 non-null   object
 13  LocalAuthorityCode         1465 non-null   object
 14  LocalAuthorityName         1465 non-null   object
 15  LocalAuthorityWebSite      1465 non-null   object
 16  LocalAuthorityEmailAddress 1465 non-null   object
 17  Scores                     0 non-null      object
 18  SchemeType                 1465 non-null   object
 19  NewRatingPending           1465 non-null   object
 20  Longitude                  1352 non-null   object
 21  Latitude                   1352 non-null   object
 22  Geocode                    0 non-null      object
dtypes: object(23)
```

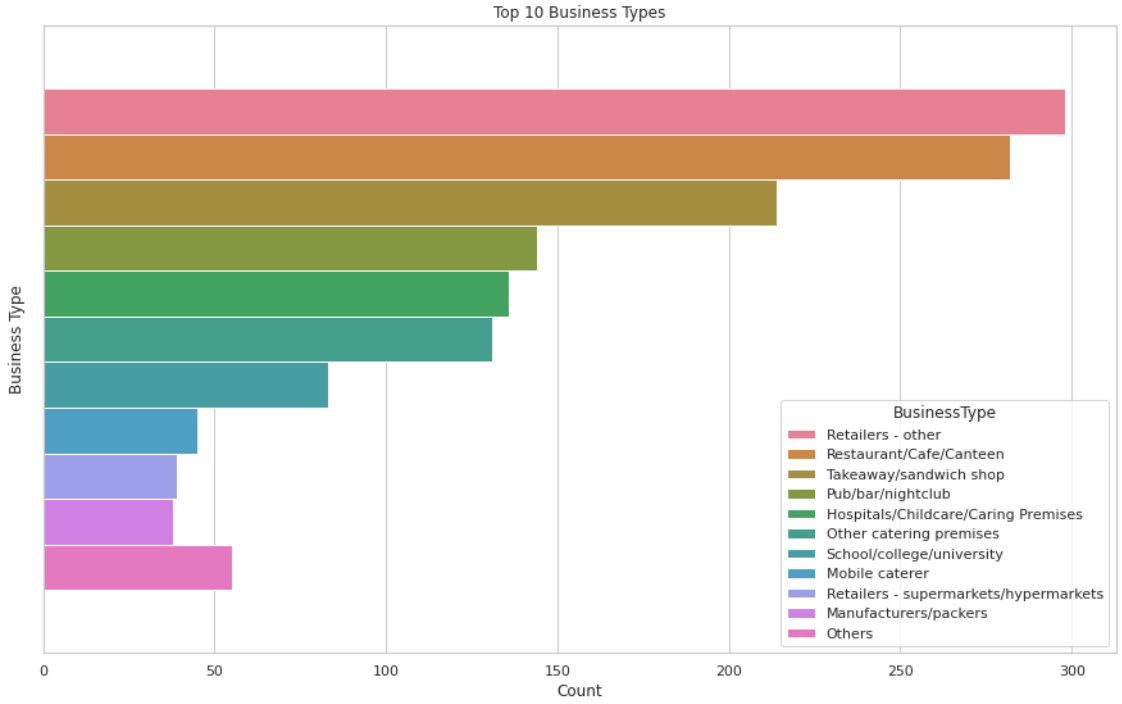
```
memory usage: 263.4+ KB
```

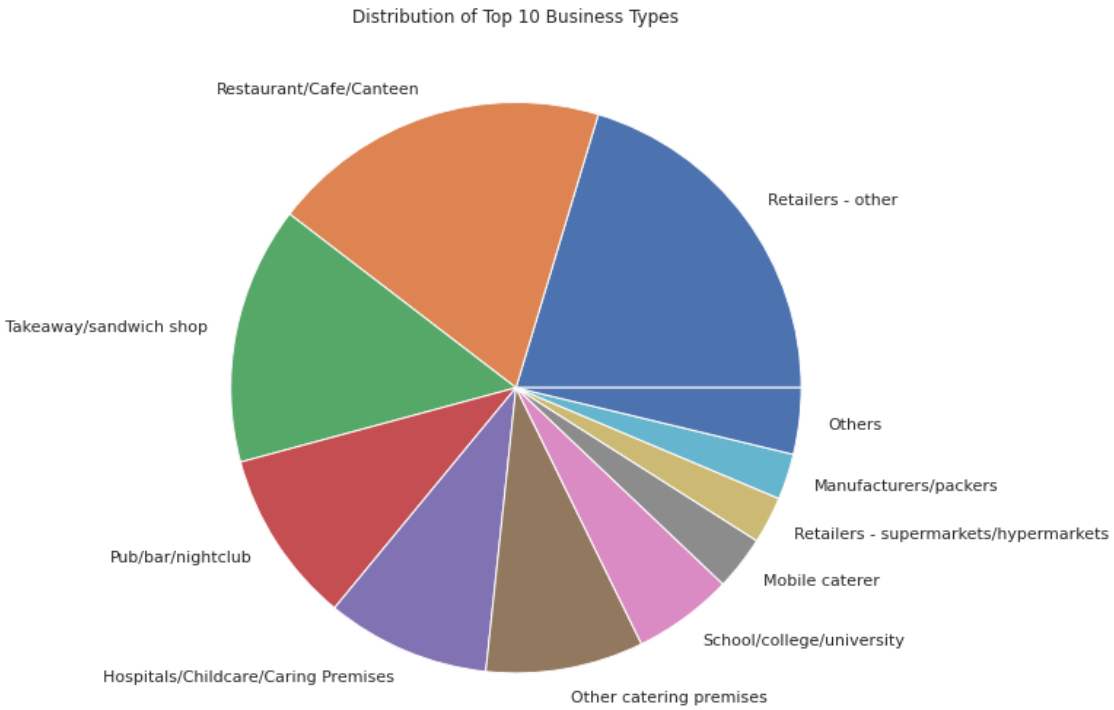
1.3 Top 10 Business Types by Count

Look at the top 10 business categories by number registered in the Renfrewshire area. We can generate a bar plot with the count of businesses and a pie chart to show their distribution.

[53]:
```
#Use SQL to read the database and write to a Pandas DataFrame
business_counts = pd.read_sql_query("""
SELECT BusinessType, COUNT(*) as Count
FROM establishments
GROUP BY BusinessType
ORDER BY Count DESC;
""", conn)

#Manipulate the dataframe to produce an others category for below top 10
#the top 10
df2 = business_counts[:10].copy()

#others - sum the number of businesses
new_row = pd.DataFrame(data = {
    'BusinessType' : ['Others'],
    'Count' : [business_counts['Count'][10:].sum()]
})

#combining top 10 with Others
business_counts = pd.concat([df2, new_row])
print(business_counts)

#Plotting
#print(business_counts)
sns.barplot(data=business_counts, x="Count", hue="BusinessType", legend = True)
plt.title("Top 10 Business Types")
plt.xlabel("Count")
plt.ylabel("Business Type")
plt.show()

#Pie chart
# define Seaborn color palette to use
business_counts.plot.pie(y = "Count", labels = business_counts["BusinessType"],␣
 ↪legend = False)
plt.title("Distribution of Top 10 Business Types")
#plt.xlabel("Rating")
plt.ylabel("") #leave the ylabel empty
#plt.ylabel("Number of Establishments")
plt.show()
```

	BusinessType	Count
0	Retailers - other	298
1	Restaurant/Cafe/Canteen	282
2	Takeaway/sandwich shop	214
3	Pub/bar/nightclub	144
4	Hospitals/Childcare/Caring Premises	136
5	Other catering premises	131
6	School/college/university	83
7	Mobile caterer	45
8	Retailers - supermarkets/hypermarkets	39
9	Manufacturers/packers	38
0	Others	55





Almost half of the businesses are eateries which is to be expected for a food hygiene survey. Retailers - other covers a lot of ground with pharmacists, supermarkets and convenience stores hence the large proportion of the dataset.

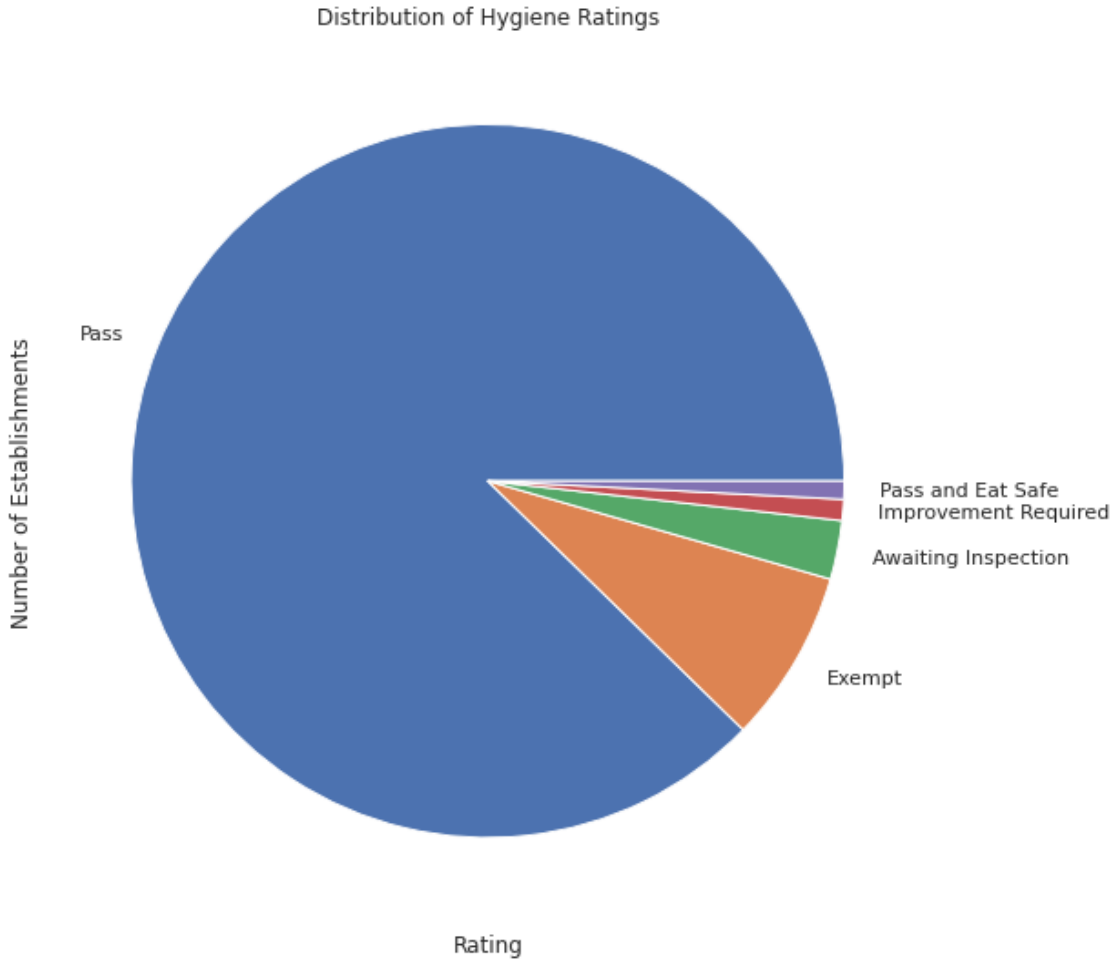
1.4 Hygiene Score Distribution

We can group the businesses by hygiene rating score to get an idea of the proportion who have passed, who needs improvement and other circumstances.

[54]:
```
# Get rating value counts
rating_counts = pd.read_sql_query("""
SELECT RatingValue, COUNT(*) as Count
FROM establishments
GROUP BY RatingValue
ORDER BY Count DESC
""", conn)

# Plot
print(rating_counts) # print the data frame
total = rating_counts["Count"].sum()
print("Total counts is: ", total)

rating_counts.plot.pie(y = "Count", labels = rating_counts["RatingValue"],␣
↪legend = False)
```

```
plt.title("Distribution of Hygiene Ratings")
plt.xlabel("Rating")
plt.ylabel("Number of Establishments")
plt.show()
```

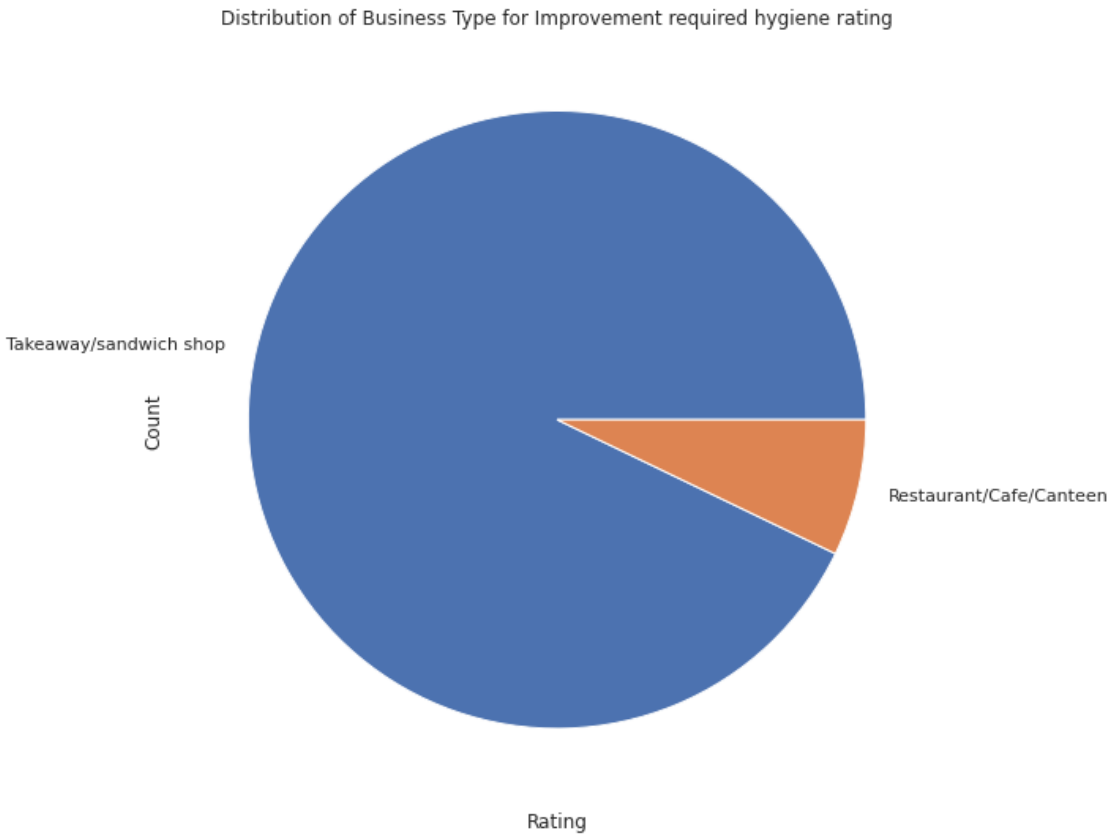
```
            RatingValue  Count
0                  Pass   1285
1                Exempt    115
2   Awaiting Inspection     39
3  Improvement Required     14
4     Pass and Eat Safe     12
Total counts is:  1465
```

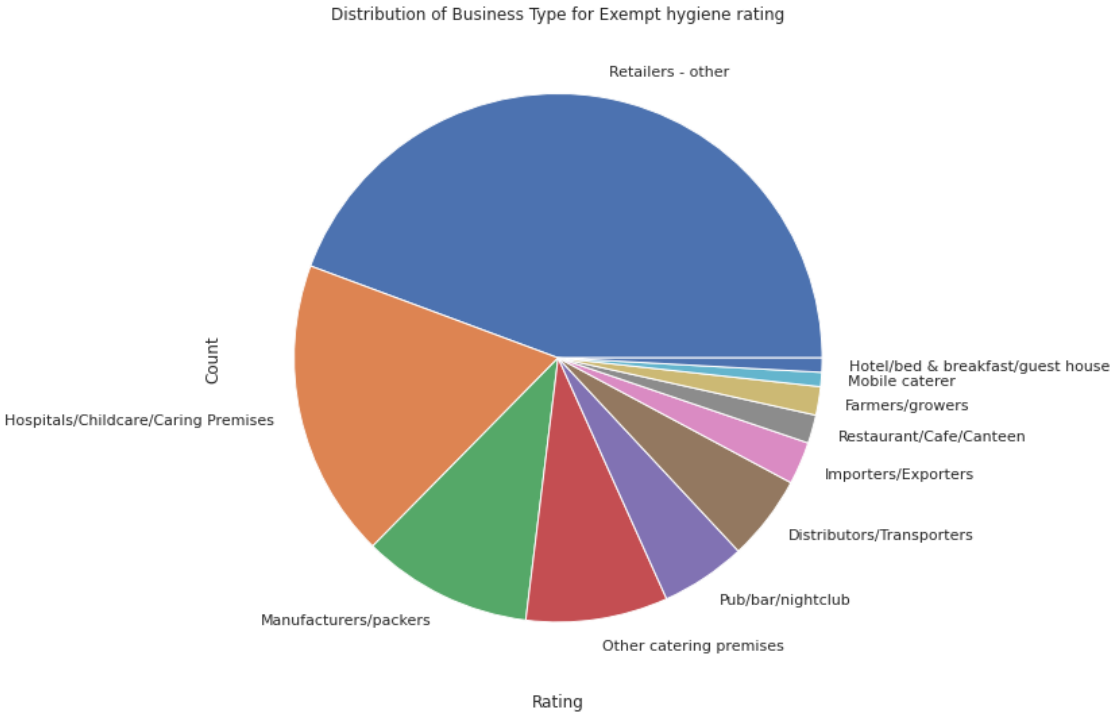


1.5 Deep dive into hygiene ratings

We can breakdown the individual ratings to find any correlations between business type and hygiene rating.

[55]:
```
#Get the improvement required ratings along with various parameters
improvement_required = pd.read_sql_query("""
SELECT RatingValue, BusinessType, COUNT(*) as Count
FROM establishments
WHERE RatingValue = 'Improvement Required'
GROUP BY BusinessType
ORDER BY Count DESC;
""", conn)

#Print
#print(improvement_required) # print the data frame

#Plotting
improvement_required.plot.pie(y = "Count", labels =␣
 ↪improvement_required["BusinessType"], legend = False)
plt.title("Distribution of Business Type for Improvement required hygiene␣
 ↪rating")
plt.xlabel("Rating")
#plt.ylabel("Number of Establishments")
plt.show()

#Get the Exempt ratings along with various parameters
exempt = pd.read_sql_query("""
SELECT RatingValue, BusinessType, COUNT(*) as Count
FROM establishments
WHERE RatingValue = 'Exempt'
GROUP BY BusinessType
ORDER BY Count DESC;
""", conn)

#Print
#print(exempt) # print the data frame

#Plotting
exempt.plot.pie(y = "Count", labels = exempt["BusinessType"], legend = False)
plt.title("Distribution of Business Type for Exempt hygiene rating")
plt.xlabel("Rating")
#plt.ylabel("Number of Establishments")
plt.show()

#Calculations using the global data frame
takeaway = df[df["BusinessType"] == "Takeaway/sandwich shop"]
#takeaway.head(10)
```


Distribution of Business Type for Improvement required hygiene rating
Takeaway/sandwich shop
Restaurant/Cafe/Canteen
Count
Rating


Distribution of Business Type for Exempt hygiene rating
Retailers - other
Hotel/bed & breakfast/guest house
Mobile caterer
Farmers/growers
Restaurant/Cafe/Canteen
Importers/Exporters
Distributors/Transporters
Pub/bar/nightclub
Other catering premises
Manufacturers/packers
Hospitals/Childcare/Caring Premises
Count
Rating

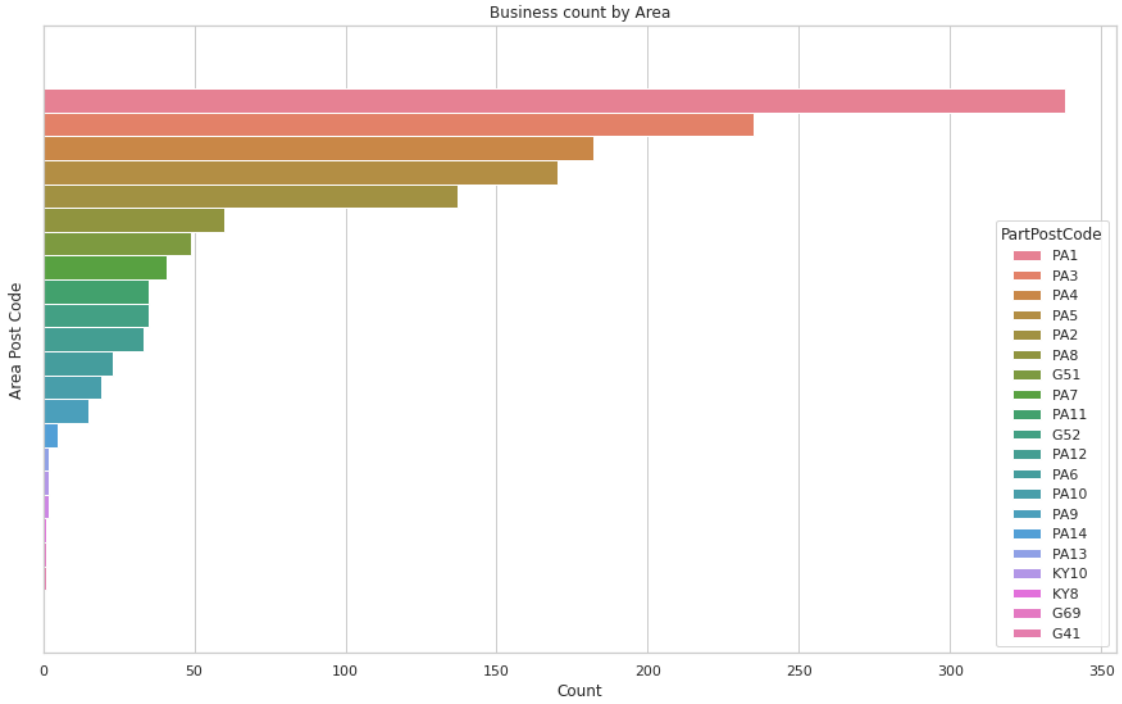
We can see that the majority of businesses that received an 'Improvement Required' score are classified as Takeawy/sandwich shop. It should be noted that Takeaway/sandwich shops take up a large proportion of the dataset but when comapred to the Restaurant/Cafe/Canteen category they are behind in food hygiene rating.

For the Exempt status we see a wide variety of business types. Exempt status is granted for businesses that don't produce their own food but do sell pre-packaged products and medicines which is corroborated here: https://essentialfoodhygiene.co.uk/what-are-the-three-food-hygiene-ratings-for-scotland/. This would explain large section being retailers - other.

1.6 Businesses by Post Code

Using the intial part of a UK postcode, an indication of geographical area can be found. Lets find how many business fit in these areas.

[56]:
```
business_by_area = pd.read_sql_query("""
SELECT COUNT(*) as "Number of businesses", SUBSTR(PostCode, 1, instr(PostCode,␣
 ↪' ')) as PartPostCode
FROM establishments
GROUP BY PartPostCode
ORDER BY "Number of businesses" DESC;
""", conn)

#print(business_by_area)
sns.barplot(data=business_by_area, x="Number of businesses",␣
 ↪hue="PartPostCode", legend = True)
plt.title("Business count by Area")
plt.xlabel("Count")
plt.ylabel("Area Post Code")
plt.show()
```



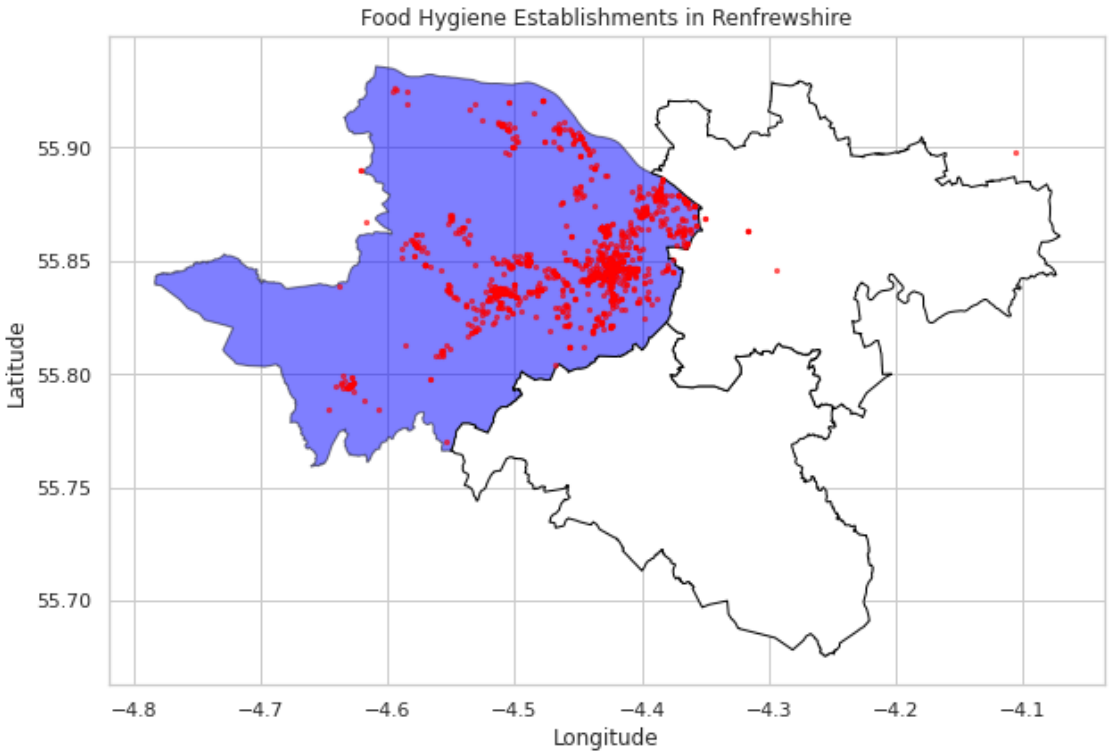
1.7 Map business location data using Geopandas

Using a shapefile for the local authority boundaries from the Improvement Service (license below), the business location data can be placed on a map.

"The dataset is provided under Open Government Licence (OGL) for download and use. You are free to copy, publish, distribute and transmit the information as long as you acknowledge the source as coming from Improvement Service under OGL."

[57]:
```
import geopandas as gpd

#Get dataframe with outlier postcodes removed
df_geo = pd.read_sql_query("""
SELECT BusinessName, BusinessType, RatingValue, PostCode, SUBSTR(PostCode, 1,␣
 ↪instr(PostCode, ' ')) as PartPostCode, Longitude, Latitude
FROM establishments
WHERE PartPostCode LIKE 'PA%' OR PartPostCode LIKE 'G%';
""", conn)

#Load Scottish local authority boundaries
#Please look at the README file to find instructions on how to download the␣
 ↪boundary shapefiles
la_gdf = gpd.read_file("/mnt/d/renfrewshire_business_insights/data/pub_las.
 ↪shp") #local authority GeoDataFrame
```

```
#print(la_gdf.columns.tolist()) #print all available columns in the GeoDataFrame
#print("All available authority names: ") #Check all available authority names
#print(la_gdf["local_auth"].unique())  # Optional: inspect names

#Filter for Renfrewshire, East Renfrewshire and Glasgow City
ren_gdf = la_gdf[la_gdf["local_auth"] == "Renfrewshire"].copy()
east_ren_gdf = la_gdf[la_gdf["local_auth"] == "East Renfrewshire"].copy()
glasgow_gdf = la_gdf[la_gdf["local_auth"] == "Glasgow City"].copy()

#Filter out null coordinates from the hygiene dataframe
df_geo = df_geo[df_geo['Latitude'].notnull() & df_geo['Longitude'].notnull()]

#Convert DataFrame to GeoDataFrame
points_gdf = gpd.GeoDataFrame(
    df_geo,
    geometry=gpd.points_from_xy(df_geo.Longitude.astype(float), df_geo.Latitude.
 ↪astype(float)),
    crs="EPSG:4326"
)

#Ensure CRS matches
ren_gdf = ren_gdf.to_crs(epsg=4326)
east_ren_gdf = east_ren_gdf.to_crs(epsg=4326)
glasgow_gdf = glasgow_gdf.to_crs(epsg=4326)

#Plotting
fig, ax = plt.subplots(figsize=(10, 10))
ren_gdf.plot(ax=ax, color='blue', edgecolor='black', alpha = 0.5)
east_ren_gdf.plot(ax=ax, color='white', edgecolor='black')
glasgow_gdf.plot(ax=ax, color='white', edgecolor='black')
points_gdf.plot(ax=ax, markersize=5, alpha=0.6, color='red')
plt.title("Food Hygiene Establishments in Renfrewshire")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```

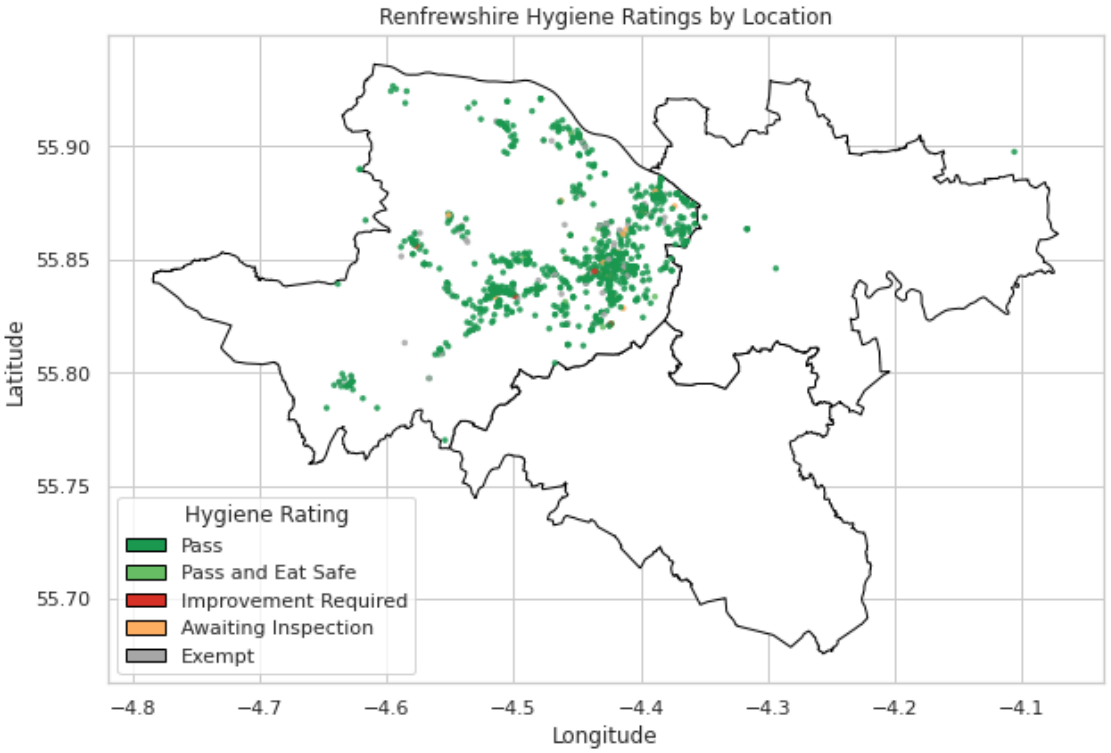


The highlighted area corresponds to the Renfrewshire council area, while the top right area is Glasgow City and the bottom area is East Renfrewshire. We can see most of the businesses are located in the eastern side of Renfrewshire which corresponds to Paisley, Johnstone and Renfrew. We can corroborate this with an interactive map later in the notebook.

Before that, we can plot hygiene ratings onto this map as follows.

[58]:
```
#Generate the local authority boundaries to a single GeoDataFrame
included_areas = ["Renfrewshire", "East Renfrewshire", "Glasgow City"]
boundary_gdf = la_gdf[la_gdf["local_auth"].isin(included_areas)].copy()␣
 ↪#boundary dataframe
boundary_gdf = boundary_gdf.to_crs(epsg=4326) #covnert to consitent CRS

# Map each string rating to a colour
rating_colors = {
    "Pass": "#1a9850",                # green
    "Pass and Eat Safe": "#66bd63",   # light green
    "Improvement Required": "#d73027",# red
    "Awaiting Inspection": "#fdae61", # orange
    "Exempt": "#a6a6a6"               # grey
}

#Create a legend for the plot
```

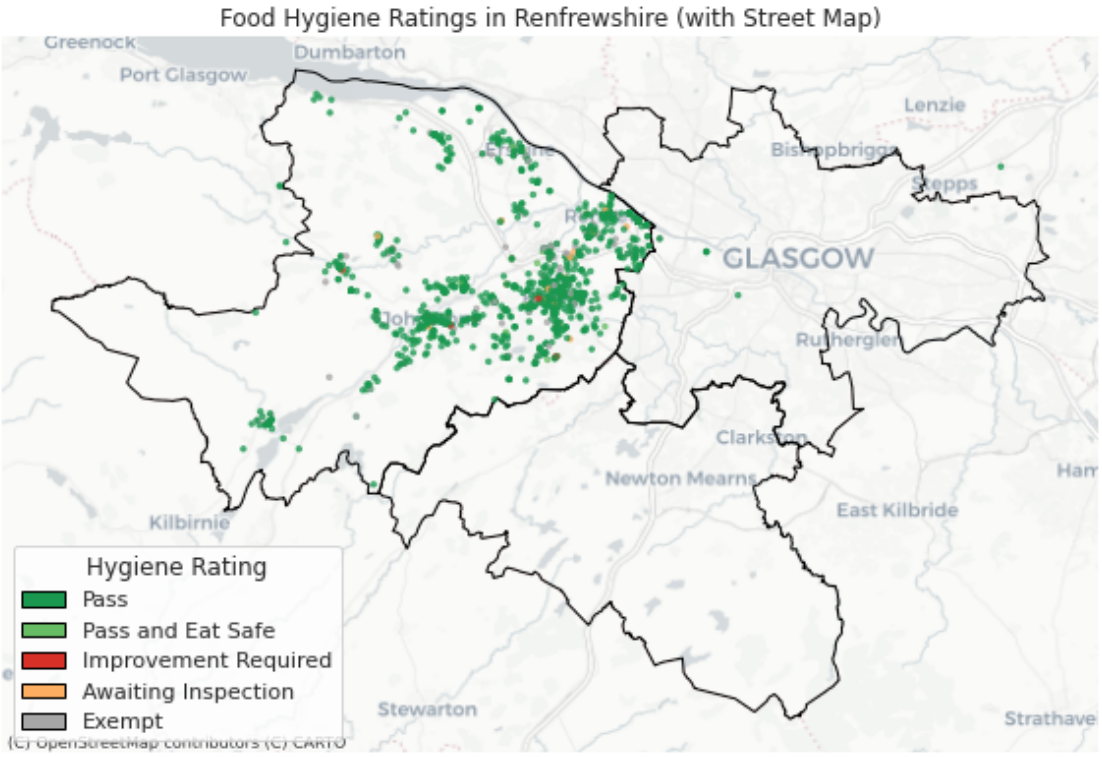
```
legend_elements = [
    Patch(facecolor=color, edgecolor='black', label=label)
    for label, color in rating_colors.items()
]

# Map rating to colours by adding a colour column to the dataframe
points_gdf["color"] = points_gdf["RatingValue"].map(rating_colors)

# #Diagnostics
# print(points_gdf[["Longitude", "Latitude", "geometry"]].head())
# print(points_gdf.geom_type.unique())
# print(points_gdf.crs)

#Plot
fig, ax = plt.subplots(figsize=(10, 10))
boundary_gdf.plot(ax=ax, color="white", edgecolor="black")
points_gdf.plot(ax=ax, markersize=6, color=points_gdf["color"], alpha=0.8)

plt.title("Renfrewshire Hygiene Ratings by Location")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.grid(True)
plt.legend(handles=legend_elements, title="Hygiene Rating", loc='lower left')
plt.show()
```



As the vast majority of hygiene ratings were pass, it is expected to see most of the data points be the same colour.

Using `contextily` we can create static map with geographic data to layer under the above plot.

[59]:
```
#Use contextily to plot street map underneath plots
import contextily as ctx

#Reproject both GeoDataFrames to EPSG:3857 (Web Mercator)
points_web = points_gdf.to_crs(epsg=3857)
boundary_web = boundary_gdf.to_crs(epsg=3857)

fig, ax = plt.subplots(figsize=(10, 10))

#Plot boundary outline
boundary_web.plot(ax=ax, color='none', edgecolor='black')
points_web.plot(ax=ax, markersize=6, color=points_web['color'], alpha=0.7)␣
 ↪#Plot hygiene points (coloured by rating, as before)

#Add basemap tiles
#ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik) #OpenMap full␣
 ↪colour
ctx.add_basemap(ax, source=ctx.providers.CartoDB.Positron) #grayscale overlay

plt.title("Food Hygiene Ratings in Renfrewshire (with Street Map)")
plt.axis("off")
plt.legend(handles=legend_elements, title="Hygiene Rating", loc='lower left')
plt.show()
```



1.8 Interactive map using Folium and GeoPandas

[60]:
```
print(points_web["color"])

#Generate the base map
#map_center = [points_web["Latitude"].astype(float).mean(),␣
 ↪points_web["Longitude"].astype(float).mean()]
#m = folium.Map(location=map_center, zoom_start=12, tiles="CartoDB Positron")

m = points_web.explore(
     column = "RatingValue",
     tiles = None,
     tooltip = ["BusinessName", "PostCode", "RatingValue"],
     popup = False,
     cmap = "jet_r",
     legend_kwds = {"caption": "Markers Hygiene Rating"},
     name = "Markers" #name of the layer
)

#Add tile layer with customer layer name
folium.TileLayer(
    tiles="https://{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png",
```

```
    attr="© OpenStreetMap contributors & CartoDB",
    name="Light map",     #This name appears in the layer control
    control=True,
    show = True
).add_to(m)

#Dark Mode (CartoDB Dark Matter)
folium.TileLayer(
    tiles="https://{s}.basemaps.cartocdn.com/dark_all/{z}/{x}/{y}{r}.png",
    name="Dark Map",
    attr="© OpenStreetMap & CartoDB",
    show = False,
).add_to(m)

#Satellite (Esri World Imagery)
folium.TileLayer(
    tiles="https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/
 ↪MapServer/tile/{z}/{y}/{x}",
    name="Satellite",
    attr="Tiles © Esri",
    show = False
).add_to(m)

#add a business density heatmap
heat_data = [[row["Latitude"], row["Longitude"]] for _, row in points_web.
 ↪iterrows()]
heat = folium.FeatureGroup(name="Business denisty heatmap", show = True)
HeatMap(heat_data, min_opacity=0.4, radius=15).add_to(heat)
heat.add_to(m)

#Add a layer highlighting Improvement required
improve_df = points_web[points_web["RatingValue"] == "Improvement Required"]␣
 ↪#find the points
improve_layer = folium.FeatureGroup(name = "Improvement Required (Red␣
 ↪Markers)", show = False)
for _, row in improve_df.iterrows():
    tooltip_text = (
        f"<b>{row['BusinessName']}</b><br>"
        f"Hygiene Rating: {row['RatingValue']}<br>"
        f"Postcode: {row['PostCode']}"
    )

    folium.CircleMarker(
        location=[row["Latitude"], row["Longitude"]],
        radius=5,
        color="black",
        weight = 2,
```

```
        fill=True,
        fill_color="red",
        fill_opacity=0.9,
        tooltip=folium.Tooltip(tooltip_text)
    ).add_to(improve_layer)

improve_layer.add_to(m)

#Add improvement required heatmap
improve_heat_layer = folium.FeatureGroup(name="Improvement Required (Heatmap)",␣
 ↪show = False)
heat_data = [[row["Latitude"], row["Longitude"]] for _, row in improve_df.
 ↪iterrows()]
HeatMap(heat_data, min_opacity=0.4, radius=15, blur=10).
 ↪add_to(improve_heat_layer)
improve_heat_layer.add_to(m)

#Add layer control toggle
folium.LayerControl(collapsed = False).add_to(m)

#add sources
m.get_root().html.add_child(folium.Element("""
    <div style="position: fixed; bottom: 5px; left: 5px; font-size: 11px;␣
 ↪background-color: white; padding: 4px; border: 1px solid #ccc;">
        Source: <a href='https://ratings.food.gov.uk/open-data/en-GB'␣
 ↪target='_blank'>FSA Hygiene Ratings</a>
    </div>
"""))


#Save
m.save("/mnt/d/renfrewshire_business_insights/docs/renfrewshire_hygiene_ratings.
 ↪html")
```

```
0       #1a9850
1       #1a9850
2       #1a9850
3       #1a9850
4       #1a9850
        ...
1376    #1a9850
1377    #1a9850
1378    #1a9850
1379    #d73027
1380    #1a9850
Name: color, Length: 1338, dtype: object
```

1.8.1 Outliers

There are three postcodes that start in KY, which is not a post code in the vicinity of Renfrewshire. We can look at these results more closely to determine why this could be.

[61]:
```
outliers = pd.read_sql_query("""
SELECT BusinessName, BusinessType, PostCode, AddressLine1,
AddressLine2, AddressLine3, AddressLine4
FROM establishments
WHERE PostCode LIKE "KY%";
""", conn)

outliers.head()
```

[61]:

	BusinessName	BusinessType	PostCode	AddressLine1	AddressLine2	AddressLine3	AddressLine4
0	DM Fish Merchants Ltd	Mobile caterer	KY8 1HQ	None	55 Mavis Bank	Buckhaven	None
1	DNM Fish Mechant	Retailers - other	KY10 3YP	None	23 Lindsay Berwick Place	Anstruther	None
2	Sandra Hodge	Retailers - other	KY103HE	None	1 St Ayles Crescent	Anstruther	None
3	Yvonne Dehn	Mobile caterer	KY10 3YN	None	13 North Marches	Anstruther	Fife

Two of the results are mobile caterers which could explain the lack of a business address within the confines of Renfrewshire.

1.9 Yelp review integration

Note: Yelp review data in this analysis is simulated using a mock dataset to demonstrate integration and enrichment logic. This is to maintain a low cost to the project. In a production setting, this would be populated via the Yelp Fusion API or a licensed source.

[62]:
```
#Generate mock Yelp review dataframe

yelp_mock = pd.DataFrame({
    "BusinessName": [
        "Domino's Pizza",
        "Greggs",
        "Costa Coffee",
        "KFC",
        "Subway",
        "The Craig Dhu",
        "La Mesa",
        "Zambretto Italian",
        "Cardosi's",
        "Cafe Su",
```

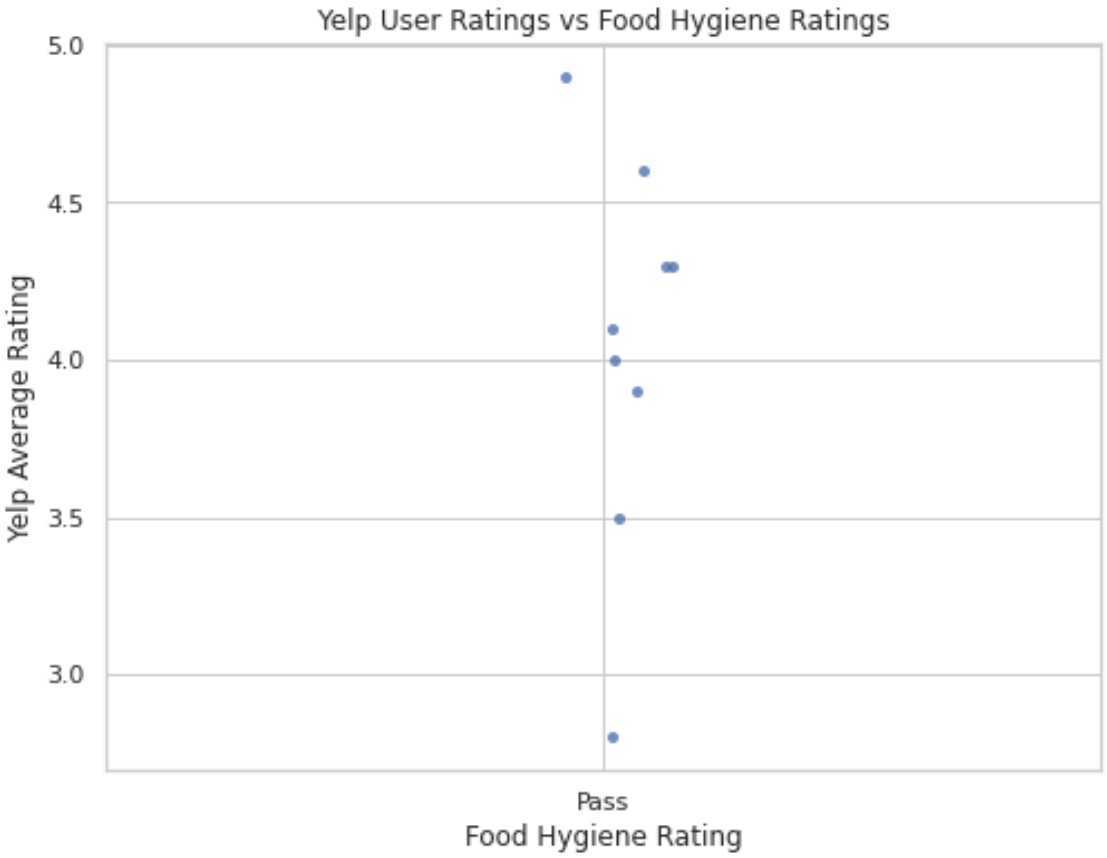
```
    ],
    "YelpRating": [3.4, 4.1, 4.0, 2.8, 3.5, 4.6, 4.3, 4.3, 4.9, 3.9],
    "ReviewCount": [122, 80, 195, 76, 110, 56, 33, 72, 98, 41],
    "PostCode": [
        "PA1 1EX", "PA1 2UZ", "PA3 4EP", "PA1 2AB", "PA1 2BS",
        "PA2 6DA", "PA1 1XU", "PA3 2AP", "PA1 2AR", "PA4 8QE"
    ]
})
```

[63]:
```
#Standardise column format
points_web["BusinessName_clean"] = points_web["BusinessName"].str.lower().str.
 ↪strip()
yelp_mock["BusinessName_clean"] = yelp_mock["BusinessName"].str.lower().str.
 ↪strip()

#print(points_web[["BusinessName", "PostCode"]].head(10))
#print(yelp_mock[["BusinessName", "PostCode"]].head(10))

#print(points_web.head(10))

#Merge on cleaned name and postcode
enriched_df = points_web.merge(
    yelp_mock,
    how="left",
    on=["BusinessName_clean", "PostCode"],
    suffixes=("", "_Yelp")
)

#Check how many matches succeeded
matched = enriched_df["YelpRating"].notnull().sum()
print(f"Yelp data matched for {matched} establishments.")
```

```
Yelp data matched for 9 establishments.
```

[64]:
```
compare_df = enriched_df.dropna(subset=["YelpRating"]) #only keep rows with a␣
 ↪Mock Yelp score

plt.figure(figsize=(8, 6))
sns.stripplot(
    data=compare_df,
    x="RatingValue",
    y="YelpRating",
    jitter=True,
    alpha=0.8
)
plt.title("Yelp User Ratings vs Food Hygiene Ratings")
plt.xlabel("Food Hygiene Rating")
```

```
plt.ylabel("Yelp Average Rating")
plt.grid(True)
plt.show()


#Plot heatmap
heat_df = compare_df[["Latitude", "Longitude", "YelpRating"]].copy()
heat_df["Latitude"] = heat_df["Latitude"].astype(float)
heat_df["Longitude"] = heat_df["Longitude"].astype(float)

#Create base map centered on Renfrewshire
map_center = [heat_df["Latitude"].mean(), heat_df["Longitude"].mean()]
m = folium.Map(location=map_center, zoom_start=12, tiles="CartoDB Positron")

#Create weighted heatmap points
heat_data = [[row["Latitude"], row["Longitude"], row["YelpRating"]] for _, row␣
 ↪in heat_df.iterrows()]

#Add heatmap layer
HeatMap(heat_data, min_opacity=0.5, radius=15, blur=12, max_zoom=1).add_to(m)

#Save or show map
m.save("/mnt/d/renfrewshire_business_insights/docs/yelp_heatmap.html") #saved␣
 ↪in reports directory
```



1.10 Tableau conversion

[65]:
```
#Convert points GeoDataFrame to csv
#points_web.drop(columns="geometry").to_csv("/mnt/d/
↪renfrewshire_business_insights/data/establishments_clean.csv", index = False)
```

1.11 Conclusions

"This analysis explored food hygiene trends in Renfrewshire using publicly available inspection data, geospatial mapping, and simulated Yelp enrichment. It demonstrates core data skills including SQL, Python, spatial joins, and visual communication."

1.12 Save and Close

[66]:
```
conn.close()
```

[ ]:
```

```