Store Opening Toolkit

February 8, 2025

Jingyang (Madilyn) Xu, Xianglong (Jason) Wang, Yingxuan (Selina) Bian, Yuqing (Brandy) Huang

1 Documentation

1.1 Decision & Design

- A few topics were considered based on our interests such as travel itinerary planners, E-commerce product comparison tools, and currency conversion tools. Some directions were in our favor but some of the API keys required were not available publicly, for example, Amazon Product Advertising API requires a seller account, hence they are not actionable. After validating four accessible APIs: Safeway, Google Maps, Yelp Fusion, and OpenAI, we constructed a business use case that is most practical and interesting based on them.
- Our tool can provide valuable insights for potential business owners who plan to open physical stores in a certain location. The type of businesses includes but is not limited to restaurants, bars, and barber shops. In our case, we used ice cream shop as an example.
- Information about popular locations within the area in which one is interested in opening a store, the average operation hours of similar businesses in the area, as well as the number of competitors and their names and images, would be extracted from Google Maps and Yelp Fusion API. All the information will go through the Open AI API to produce recommendations to the potential business owners.

1.2 Challenges & Solutions

- Some APIs are not publicly available, which requires paid subscriptions or business emails: Use available free functions and 'borrow' a business email
- Google Maps API doesn't have information about: Foot traff ic & menu for more comprehen- sive and accurate recommendations
- OpenAI API cost more: Choose the alternative Google Gemini API which has $300 free trial
- Gradio is unable to handle CSV, JSON, and Image: Summarize information and input in text format

2 Fetch Info from GoogleMap

2.0.1 Extrat information for traff ic-heavy places

[9]:
```
import requests

## Set boundary for all SF
types = ["shopping_mall", "tourist_attraction", "transit_station", "restaurant"]
results = []

for place_type in types:
    response = requests.get("https://maps.googleapis.com/maps/api/place/
 nearbysearch/json", params={
        "location": "37.7529,-122.4474",
        "radius": 1000000,
        "type": place_type,
        "key": "###get your key"
    })
    results.extend(response.json().get("results", []))
```

[10]:
```
results = response.json()
```

[11]:
```
## Get traffic heavy places information within SF
def get_places(place_type):
    places = []
    if "results" in results:
        for place in results["results"]:
            name = place.get("name", "Unknown")
            lat = place["geometry"]["location"]["lat"]
            lng = place["geometry"]["location"]["lng"]
            place_id = place["place_id"]
            places.append({"name": name, "lat": lat, "lng": lng, "place_id":␣
 place_id})
    return places
```

[12]:
```
place_types = ["transit_station", "shopping_mall", "tourist_attraction",␣
 "park", "restaurant"] # Foot-traffic-heavy places

## Fetch places for all types
sf_places = []
import time
for place_type in place_types:
    sf_places.extend(get_places(place_type))
    time.sleep(1) ## Prevent API rate limits
```

[13]:
```
sf_places[:3] ## Check output (Only display the first 3 output as it's too long␣
 to print all)
```

[13]:
```
[{'name': 'Fairmont San Francisco',
  'lat': 37.7923897,
  'lng': -122.4104443,
  'place_id': 'ChIJN2S4EI2AhYAR9J4Qeh1U8Aw'},
 {'name': 'Hotel Shattuck Plaza',
  'lat': 37.86918319999999,
  'lng': -122.2684195,
  'place_id': 'ChIJ9SUZTJx-hYARuXH-EbqeQjU'},
 {'name': 'Graduate by Hilton Berkeley',
  'lat': 37.8679648,
  'lng': -122.2563185,
  'place_id': 'ChIJO1q-ry98hYARXdN27wXhtqg'}]
```

2.0.2 Get Popular Times Data from Google Places

[15]:
```
def get_popular_times(place_id):
    response = requests.get("https://maps.googleapis.com/maps/api/place/details/
  json",
    params = {
        "place_id": place_id,
        "fields": "name,current_opening_hours",
        "key":"###get your key"
    }
    results = response.json()

    if "result" in results and "current_opening_hours" in results["result"]:
        return results["result"]["current_opening_hours"]
    return None
```

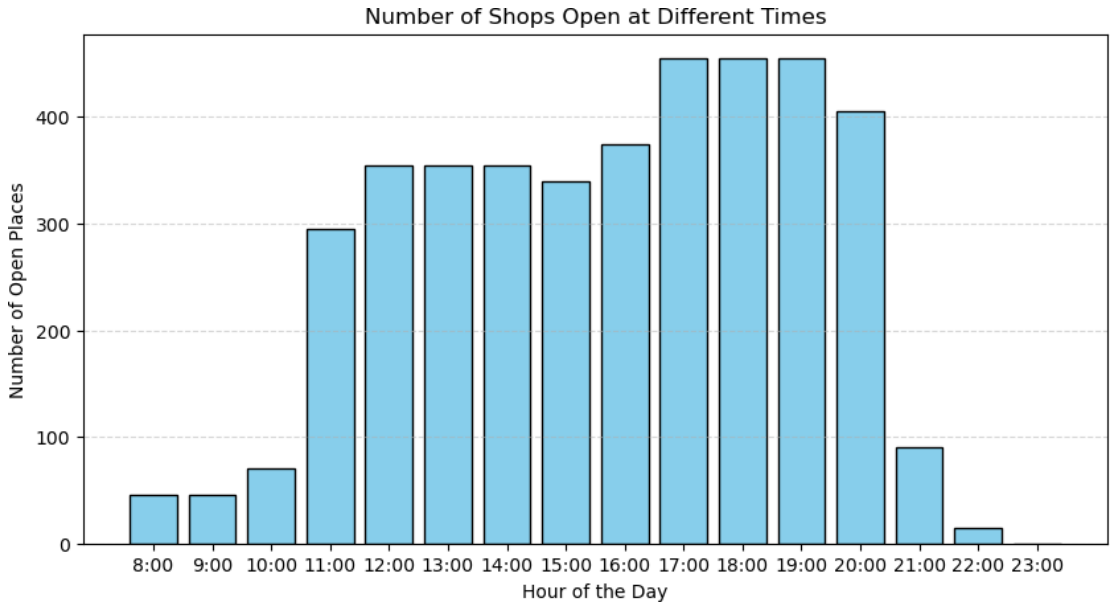
[16]:
```
## Fetch Popular Times for each place
for place in sf_places:
    place["current_opening_hours"] = get_popular_times(place["place_id"])
    time.sleep(1)  ## Prevent API rate limits
```

[17]:
```
import json
print(json.dumps(sf_places[:1], indent=2)) ## Print the first result
```


```
[
  {
    "name": "Fairmont San Francisco",
    "lat": 37.7923897,
    "lng": -122.4104443,
    "place_id": "ChIJN2S4EI2AhYAR9J4Qeh1U8Aw",
    "current_opening_hours": null
  }
]
```

[18]:
```
import matplotlib.pyplot as plt
import numpy as np

## Extract opening and closing times for all places
def opening_hours(sf_places):
    open= []

    for place in sf_places:
        # Check if 'current_opening_hours' and 'periods' keys exist
        if "current_opening_hours" in place and place["current_opening_hours"]
 is not None and "periods" in place["current_opening_hours"]:
            for period in place["current_opening_hours"]["periods"]:
                open_time = int(period["open"]["time"]) // 100
                close_time = int(period["close"]["time"]) // 100
                open.append((open_time, close_time))

    return open
```

[19]:
```
## Plot a histogram showing the number of places open at each hour from 8 AM to
  11 PM.
def opening_plot(sf_places):
    open_intervals = opening_hours(sf_places)
    hours = np.arange(8, 24)  # Time range from 8 AM (8) to 11 PM (23)
    open_counts = np.zeros_like(hours)       # Initialize counts to zero

    # Count how many places are open at each hour
    for start, end in open_intervals:
        for i, hour in enumerate(hours):
            if start <= hour < end:
                open_counts[i] += 1

    # Plot the histogram
    plt.figure(figsize=(10, 5))
    plt.bar(hours, open_counts, width=0.8, color='skyblue', edgecolor='black')

    plt.xlabel("Hour of the Day")
    plt.ylabel("Number of Open Places")
    plt.title("Number of Shops Open at Different Times")
    plt.xticks(hours, [f"{h}:00" for h in hours])
    plt.grid(axis="y", linestyle="--", alpha=0.5)

    plt.show()

# Call the function to plot
opening_plot(sf_places)
```



2.0.3 Create a City-Wide Heatmap

[21]:
```
# Extract locations & traffic intensity
results = response.json().get("results", [])
heatmap_data = [(place["geometry"]["location"]["lat"],␣
  ▫place["geometry"]["location"]["lng"]) for place in results]
```

[22]:
```
!pip install -q folium
import folium
from folium.plugins import HeatMap

# Create a map centered in San Francisco
m = folium.Map(location=[37.7749, -122.4194], zoom_start=13)

# Add heatmap
HeatMap(heatmap_data).add_to(m)

# Save map as an HTML file
m.save("heatmap.html")
```

2.0.4 Create a heatmap of the popular Ice Cream shops in SF

[24]:
```
types = ["restaurant", "cafe", "bakery"]
results = []

for place_type in types:
```

```
    response = requests.get("https://maps.googleapis.com/maps/api/place/
  nearbysearch/json", params={
        "location": "37.7529,-122.4474",
        "radius": 1000000,
        "type": place_type,
        "keyword": "cream",
        "key":  "##get your key"
    })
    results.extend(response.json().get("results", []))
```

[25]:
```
## Fetch places information
places = [
    (
        place.get("geometry", {}).get("location", {}).get("lat", None),
        place.get("geometry", {}).get("location", {}).get("lng", None),
        place.get("name", "Unknown"),
        place.get("rating", 0)
    )
    for place in results
    if place.get("geometry", {}).get("location")
]
```

[26]:
```
center = [37.7749, -122.4194]  # Center of SF
heatmap = folium.Map(location=center, zoom_start=13)
```

[27]:
```
# Convert places into heatmap format
heat_data = [(lat, lng, rating) for lat, lng, _, rating in places]
```

[28]:
```
# Add heat map layer
HeatMap(heat_data).add_to(heatmap)
```

[28]: <folium.plugins.heat_map.HeatMap at 0x7019a0929960>

[29]:
```
# Save map as HTML file
heatmap.save("heatmap_popularity.html")
print("Heatmap saved as heatmap_popularity.html")
```

Heatmap saved as heatmap_popularity.html

3 Fetch Info from Yelp

[31]:
```
!pip install -q requests
```

3.0.1 Connect with Yelp data by API

[33]:
```
# use yelp api key to connect with yelp data
import requests
import pandas as pd
from IPython.display import display, HTML


API_KEY =
  "##get your key"
headers = {"Authorization": f"Bearer {API_KEY}"}

# Function to fetch data with pagination
def get_yelp_data(offset):
    response = requests.get(
        "https://api.yelp.com/v3/businesses/search",
        headers=headers,
        params={
            "location": "San Francisco, CA",
            "categories": "icecream",
            "limit": 50,
            "offset": offset,
            "sort_by": "rating"
        }
    )
    return response.json()

# Fetch first 50 results
data_1 = get_yelp_data(offset=0)
businesses_1 = data_1['businesses']

# Fetch next 50 results
data_2 = get_yelp_data(offset=50)
businesses_2 = data_2['businesses']

# Combine both result sets
all_businesses = businesses_1 + businesses_2
df = pd.DataFrame(all_businesses)

# Display DataFrame
df
```

[33]:

	id	alias	\
0	wMsJI3LiRzTbdLSNWVlH3Q	cruisin-creams-san-jose	
1	cr8fQy_WP0yNe__YuHmJGg	gateaux-san-francisco	
2	P8I3dOY4Nxu4DUC6xSyEmg	golden-state-ice-cream-san-jose	
3	T_Fq2smXPE8y2jx23T2lHA	frogo-food-truck-san-francisco	

```
4    TE-xGT7CrOWGb0mqr-1VPw                    koolfi-creamery-san-francisco
..                      ...                                              ...
95   qBlWcG1IXmqJZsZLoJX47g                holloway-mikes-deli-san-francisco-2
96   u3npL1WfXFw5w36ciXvstw                      ghirardelli-san-francisco-3
97   gJsUMxHGXkEdN7hIMhhmKA                steep-creamery-and-tea-san-francisco
98   yHUapaoG0LNKVgUagsFXpQ              gelato-classico-italian-san-francisco-3
99   16e5laqIVZrId4DNoJeTaQ  ghirardelli-ice-cream-and-chocolate-shop-san-f...

                                        name  \
0                          Cruisin Creams
1                                 Gateaux
2                  Golden State Ice Cream
3                       FroGo Food Truck
4                        Koolfi Creamery
..                                   ...
95                   Holloway Mikes Deli
96                           Ghirardelli
97                   STEEP Creamery & Tea
98                 Gelato Classico Italian
99  Ghirardelli Ice Cream & Chocolate Shop

                                            image_url  is_closed  \
0   https://s3-media4.fl.yelpcdn.com/bphoto/3ufx_5...      False
1   https://s3-media3.fl.yelpcdn.com/bphoto/uvnjYQ...      False
2   https://s3-media3.fl.yelpcdn.com/bphoto/D8TaD7...      False
3   https://s3-media2.fl.yelpcdn.com/bphoto/h3ezMx...      False
4   https://s3-media3.fl.yelpcdn.com/bphoto/bs4-qf...      False
..                                                ...        ...
95                                                         False
96  https://s3-media3.fl.yelpcdn.com/bphoto/YdEhqc...      False
97  https://s3-media3.fl.yelpcdn.com/bphoto/wH7-mj...      False
98  https://s3-media3.fl.yelpcdn.com/bphoto/_YL-J-...      False
99  https://s3-media1.fl.yelpcdn.com/bphoto/4ydj9J...      False

                                                 url  review_count  \
0   https://www.yelp.com/biz/cruisin-creams-san-jo...           24
1   https://www.yelp.com/biz/gateaux-san-francisco...           46
2   https://www.yelp.com/biz/golden-state-ice-crea...           21
3   https://www.yelp.com/biz/frogo-food-truck-san-...           17
4   https://www.yelp.com/biz/koolfi-creamery-san-f...           25
..                                                ...          ...
95  https://www.yelp.com/biz/holloway-mikes-deli-s...            1
96  https://www.yelp.com/biz/ghirardelli-san-franc...            5
97  https://www.yelp.com/biz/steep-creamery-and-te...          298
98  https://www.yelp.com/biz/gelato-classico-itali...          275
99  https://www.yelp.com/biz/ghirardelli-ice-cream...          667
```

```
                                                categories  rating  \
0   [{'alias': 'icecream', 'title': 'Ice Cream & F...     5.0
1   [{'alias': 'cupcakes', 'title': 'Cupcakes'}, {...     5.0
2   [{'alias': 'icecream', 'title': 'Ice Cream & F...     5.0
3   [{'alias': 'foodtrucks', 'title': 'Food Trucks...     4.9
4   [{'alias': 'icecream', 'title': 'Ice Cream & F...     4.9
..                                                ...     ...
95  [{'alias': 'delis', 'title': 'Delis'}, {'alias...     5.0
96  [{'alias': 'chocolate', 'title': 'Chocolatiers...     4.0
97  [{'alias': 'icecream', 'title': 'Ice Cream & F...     4.0
98  [{'alias': 'icecream', 'title': 'Ice Cream & F...     4.0
99  [{'alias': 'icecream', 'title': 'Ice Cream & F...     4.0

                                          coordinates transactions price  \
0   {'latitude': 37.3589935302734, 'longitude': -1...           []    $$
1     {'latitude': 37.78204, 'longitude': -122.46014}           []   NaN
2   {'latitude': 37.3225513, 'longitude': -121.911...           []    $$
3   {'latitude': 37.82733805712616, 'longitude': -...           []   NaN
4   {'latitude': 37.78994615500449, 'longitude': -...           []   NaN
..                                                ...          ...   ...
95    {'latitude': 37.72167, 'longitude': -122.46199}           []   NaN
96  {'latitude': 37.80594915243758, 'longitude': -...           []   NaN
97  {'latitude': 37.782641, 'longitude': -122.391335}     [pickup]    $$
98  {'latitude': 37.8005473315716, 'longitude': -1...           []     $
99  {'latitude': 37.78840439586704, 'longitude': -...   [delivery]    $$

                                             location         phone  \
0   {'address1': '', 'address2': None, 'address3':...  +14087728719
1   {'address1': '', 'address2': None, 'address3':...  +14152909155
2   {'address1': '330 Race St', 'address2': '', 'a...  +14082794707
3   {'address1': '900 Avenue D', 'address2': None,...  +14159949265
4   {'address1': '50 Fremont St', 'address2': 'Ste...  +14153906210
..                                                ...          ...
95  {'address1': '845 Holloway Ave', 'address2': N...  +16504164595
96  {'address1': '900 N Point St', 'address2': '',...  +14154472846
97  {'address1': '270 Brannan St', 'address2': Non...  +14156069336
98  {'address1': '576 Union St', 'address2': '', '...  +14153542160
99  {'address1': '2 New Montgomery St', 'address2'...  +14155367830

     display_phone       distance  \
0   (408) 772-8719   68523.543144
1   (415) 290-9155    3376.115083
2   (408) 279-4707   67232.745884
3   (415) 994-9265    9201.475308
4   (415) 390-6210    4741.051463
..             ...            ...
95  (650) 416-4595    4906.801658
```

```
96  (415) 447-2846   5148.027099
97  (415) 606-9336   4643.999346
98  (415) 354-2160   5035.332210
99  (415) 536-7830   4296.194294

                                                business_hours  \
0    [{'open': [{'is_overnight': True,  'start': '00...
1                                                          []
2                                                          []
3                                                          []
4    [{'open': [{'is_overnight': False,  'start': '1...
..                                                       ...
95   [{'open': [{'is_overnight': False,  'start': '0...
96   [{'open': [{'is_overnight': False,  'start': '0...
97   [{'open': [{'is_overnight': False,  'start': '0...
98   [{'open': [{'is_overnight': False,  'start': '1...
99   [{'open': [{'is_overnight': False,  'start': '1...

                                              attributes
0    {'business_temp_closed': None,  'menu_url': 'ht...
1    {'business_temp_closed': None,  'menu_url': Non...
2    {'business_temp_closed': None,  'menu_url': 'ht...
3    {'business_temp_closed': None,  'menu_url': Non...
4    {'business_temp_closed': None,  'menu_url': 'ht...
..                                                 ...
95   {'business_temp_closed': None,  'menu_url': Non...
96   {'business_temp_closed': None,  'menu_url': 'ht...
97   {'business_temp_closed': 1798704000, 'menu_url...
98   {'business_temp_closed': None,  'menu_url': 'ht...
99   {'business_temp_closed': None,  'menu_url': 'ht...

[100 rows x 18 columns]
```

3.0.2 Extract Picture of Highest rating shops

[35]:
```
## view the image info of highest rating shops to get some insights
businesses_1 = data_1.get('businesses', [])
businesses_2 = data_2.get('businesses', [])
businesses = businesses_1 + businesses_2

# Create a DataFrame with selected fields
df = pd.DataFrame([
    {"Name": b["name"],
     "Category": ", ".join([c["title"] for c in b["categories"]]),
     "Rating": b["rating"],
     "Image_URL": b["image_url"]}
    for b in businesses
```

```
D

# Sort by rating (descending order)
df_shop_info = df.sort_values(by="Rating", ascending=False)
df_shop_info

# Convert Image URLs to HTML for direct display
def make_image_html(url):
    return f'<img src="{url}" width="100">'

df["Image"] = df["Image_URL"].apply(make_image_html)
df2 = df.drop(columns=["Image_URL"])  # Remove the URL column
df2[:3] # Display the first 3 rows
```

[35]:

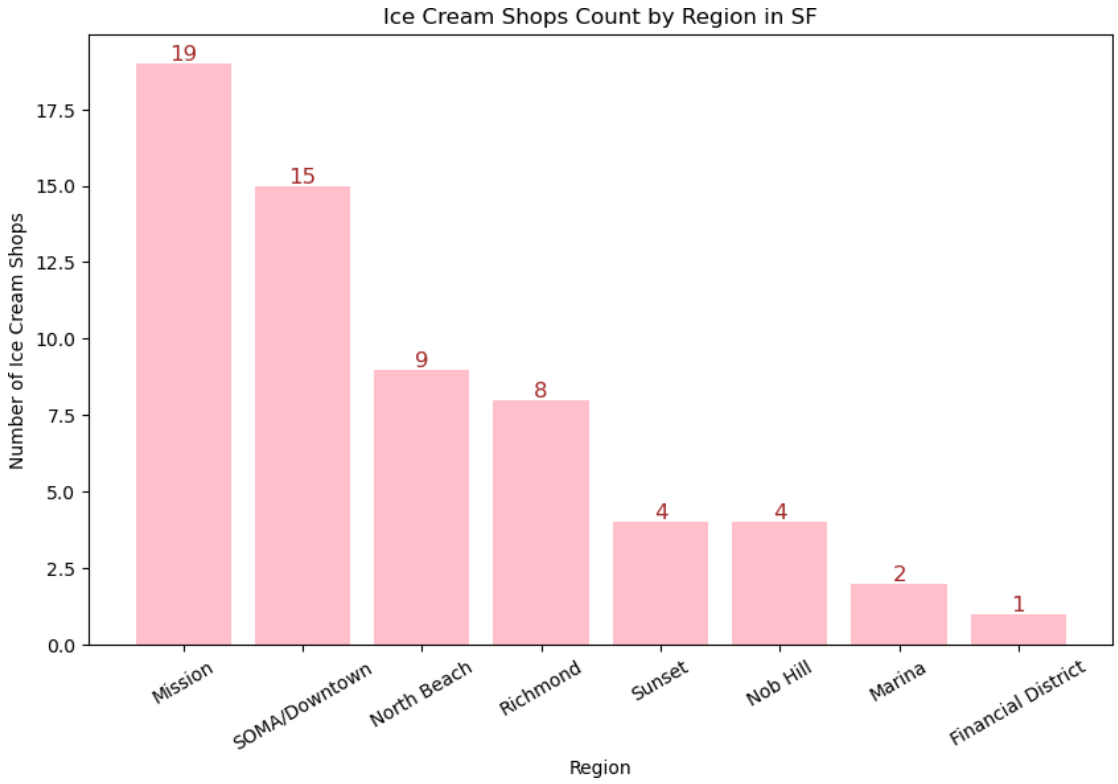
	Name	Category	Rating	Image
0	Cruisin Creams	Ice Cream & Frozen Yogurt, Food Trucks	5.0	<img src="https://s3-media4.fl.yelpcdn.com/bph...
1	Gateaux	Cupcakes, Bakeries, Ice Cream & Frozen Yogurt	5.0	<img src="https://s3-media3.fl.yelpcdn.com/bph...
2	Golden State Ice Cream	Ice Cream & Frozen Yogurt, Food Trucks	5.0	<img src="https://s3-media3.fl.yelpcdn.com/bph...

3.0.3 Extract Business Name & Zip Code

[37]:
```
## Compute count of ice cream shop in each region of San Francisco (limit 50 as an example to see results)
import matplotlib.pyplot as plt
shops = []
for data in [data_1, data_2]:
  if "businesses" in data:
      for biz in data["businesses"]:
          # Extract zip code (postal code)
          zip_code = biz["location"].get("zip_code", "Unknown")

          # Try converting to an integer (ignore invalid zip codes)
          try:
              zip_code = int(zip_code)
          except ValueError:
              zip_code = "Unknown"

          # Store results
          shops.append({
              "name": biz["name"],
              "address": ", ".join(biz["location"].get("display_address",["Unknown"])),
```

```
                "zip_code": zip_code
            })

# Convert to DataFrame
df = pd.DataFrame(shops)

# Count number of ice cream shops by zip code
zip_counts = df["zip_code"].value_counts().reset_index()
zip_counts.columns = ["zip_code", "shop_count"]

# Define Zip Code Clusters Based on Image
zip_groups = {
    "Mission": [94110, 94103, 94114],
    "SOMA/Downtown": [94103, 94105, 94107, 94108, 94111],
    "North Beach": [94133],
    "Sunset": [94116, 94122],
    "Richmond": [94118, 94121],
    "Marina": [94123],
    "Castro": [94114],
    "Financial District": [94104, 94105, 94108],
    "Nob Hill": [94109]
}

# Assign region based on zip code
df["region"] = df["zip_code"].apply(lambda x: next((region for region, zips in
  zip_groups.items() if x in zips), "Other"))

# Count number of ice cream shops by region
region_counts = df["region"].value_counts().reset_index()
region_counts.columns = ["region", "shop_count"]
region_counts = region_counts.drop(index=0)

# Create Bar Chart
plt.figure(figsize=(10, 6))
bars = plt.bar(region_counts['region'], region_counts['shop_count'],
  color='pink')

# Add count labels on top of each bar with increased font size
for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.1,   # Adjust
  height
             str(int(bar.get_height())), ha='center',
             fontsize=12, color='brown')# Increased font size

# Labels and Title
plt.xlabel("Region")
plt.ylabel("Number of Ice Cream Shops")
```

```
plt.title("Ice Cream Shops Count by Region in SF")
plt.xticks(rotation=30)  # Rotate x-axis labels for readability

# Show the plot
plt.show()

# Print grouped region data
print("\n Ice Cream Shops Count by Grouped Regions in SF:")
print(region_counts)
```



Ice Cream Shops Count by Grouped Regions in SF:

	region	shop_count
1	Mission	19
2	SOMA/Downtown	15
3	North Beach	9
4	Richmond	8
5	Sunset	4
6	Nob Hill	4
7	Marina	2
8	Financial District	1

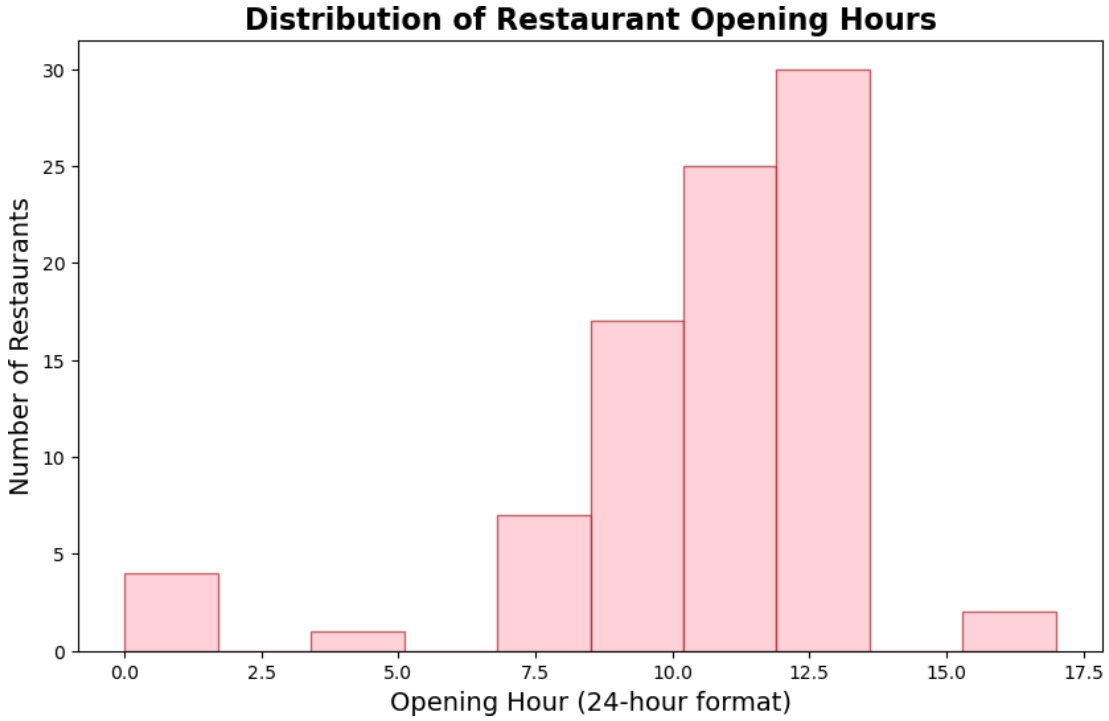
3.0.4 Extract Business IDs to Fetch Hours

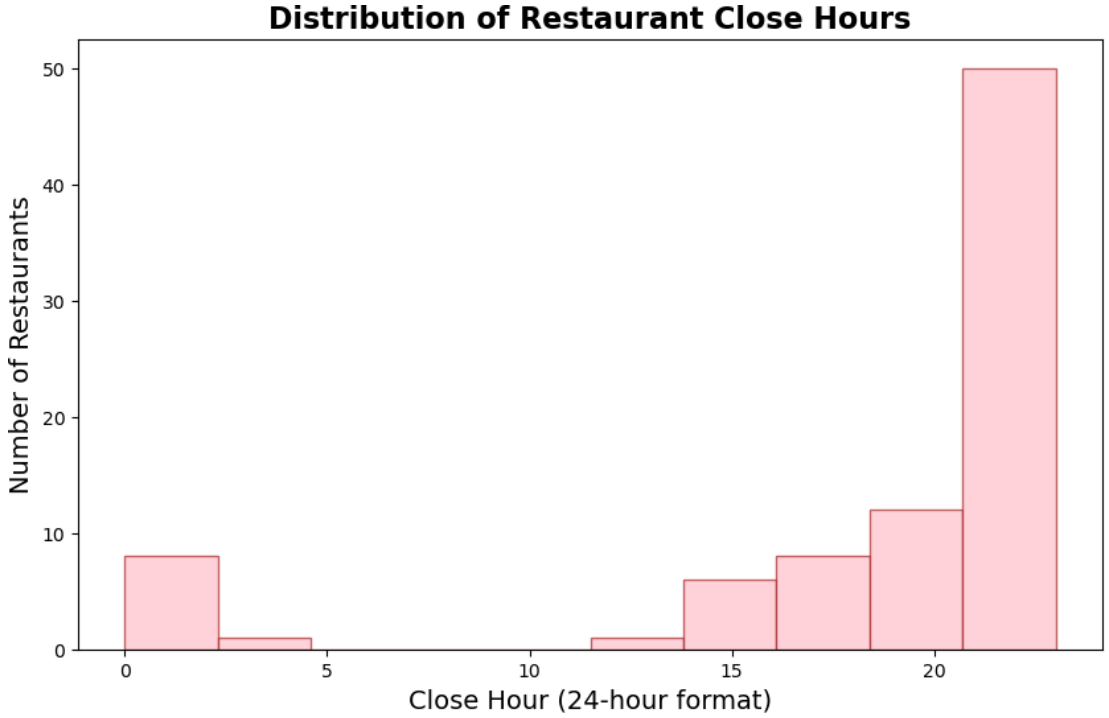
[39]:
```
import time
business_ids_1 = [biz["id"] for biz in data_1["businesses"]]
business_ids_2 = [biz["id"] for biz in data_2["businesses"]]
business_ids = business_ids_1 + business_ids_2

# Fetch Saturday's Hours for Each Business
def get_saturday_hours(business_id):
    business_url = f"https://api.yelp.com/v3/businesses/{business_id}"

    # Sleep for 0.5 seconds before making the request
    time.sleep(0.5)

    response = requests.get(business_url, headers=headers)
    business_data = response.json()

    # Extract Business Name
    name = business_data.get("name", "Unknown")

    # Extract Saturday's Open Hours
    saturday_open_time, saturday_close_time, daily_hours = "N/A", "N/A", None

    if "hours" in business_data:
        for entry in business_data["hours"][0]["open"]:
            if entry["day"] == 5:  # Day 5 = Saturday
                saturday_open_time = f"{entry['start'][:2]}:{entry['start'][2:
]}"  # Convert HHMM to HH:MM
                saturday_close_time = f"{entry['end'][:2]}:{entry['end'][2:]}" 
# Convert HHMM to HH:MM

                # Calculate daily operation hours
                start_hour = int(entry["start"][:2])
                end_hour = int(entry["end"][:2])
                daily_hours = end_hour - start_hour
                if daily_hours < 0:
                    daily_hours += 24

    return {
        "name": name,
        "saturday_open": saturday_open_time,
        "saturday_close": saturday_close_time,
        "saturday_hours": daily_hours
    }

# Fetch Saturday's hours for all businesses (with sleep delay)
ice_cream_hours = [get_saturday_hours(bid) for bid in business_ids]
```

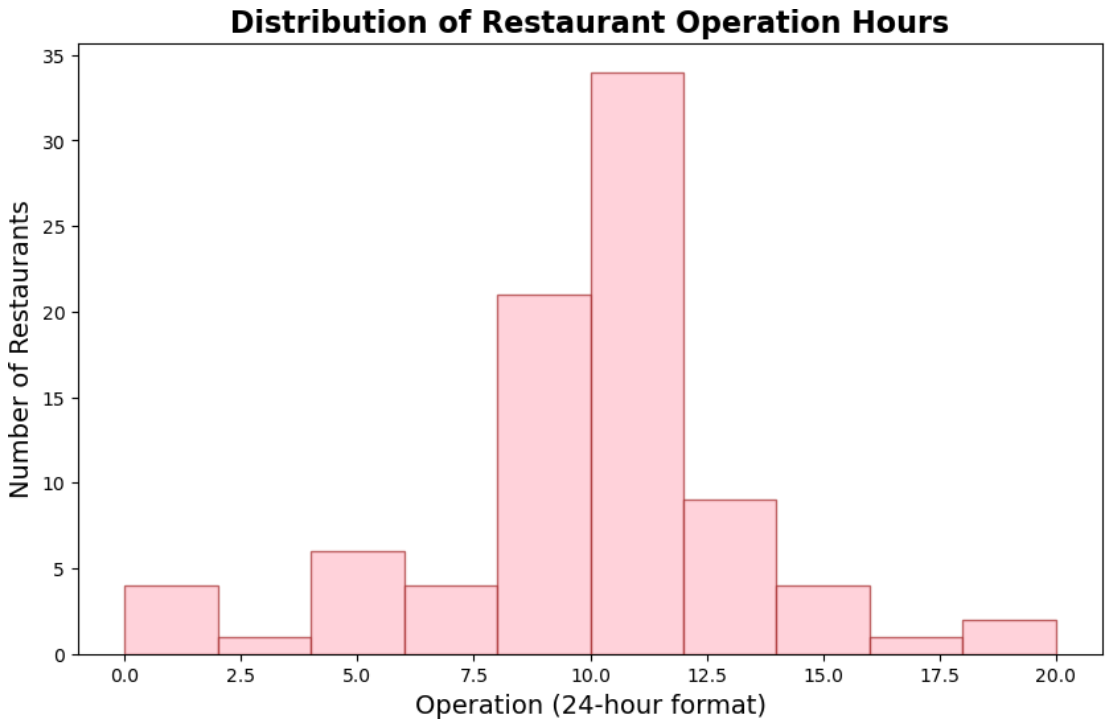
```
# Convert to DataFrame
df = pd.DataFrame(ice_cream_hours)

# Drop NA values
df = df.dropna()

# Convert 'HH:MM' to float (e.g., 08:30 → 8.5, 19:45 → 19.75)
df['saturday_open'] = df['saturday_open'].apply(lambda x: int(x.split(":")[0])
    + int(x.split(":")[1])/60)
df['saturday_close'] = df['saturday_close'].apply(lambda x: int(x.split(":
    ")[0]) + int(x.split(":")[1])/60)

open_hours = df["saturday_open"]
close_hours = df["saturday_close"]
operation_hours = df["saturday_hours"]
```

[40]:
```
# 3 Plot Histogram
plt.figure(figsize=(10, 6))
plt.hist(open_hours, bins=10, color='pink', edgecolor='brown', alpha=0.7)

# Labels and Title
plt.xlabel("Opening Hour (24-hour format)", fontsize=14)
plt.ylabel("Number of Restaurants", fontsize=14)
plt.title("Distribution of Restaurant Opening Hours", fontsize=16,
    fontweight='bold')

# Plot Histogram
plt.figure(figsize=(10, 6))
plt.hist(close_hours, bins=10, color='pink', edgecolor='brown', alpha=0.7)

# Labels and Title
plt.xlabel("Close Hour (24-hour format)", fontsize=14)
plt.ylabel("Number of Restaurants", fontsize=14)
plt.title("Distribution of Restaurant Close Hours", fontsize=16,
    fontweight='bold')

# Plot Histogram
plt.figure(figsize=(10, 6))
plt.hist(operation_hours, bins=10, color='pink', edgecolor='brown', alpha=0.7)

# Labels and Title
plt.xlabel("Operation (24-hour format)", fontsize=14)
plt.ylabel("Number of Restaurants", fontsize=14)
plt.title("Distribution of Restaurant Operation Hours", fontsize=16,
    fontweight='bold')
```

[40] : Text(0.5, 1.0, 'Distribution of Restaurant Operation Hours')







[41]:
```
# Compute overall average operation hours for Saturday
median_open_hour = open_hours.dropna().median()
median_close_hour = close_hours.dropna().median()
median_saturday_hours = operation_hours.dropna().median()
# Print the DataFrame
# Print the average Saturday operation hours for all ice cream shops
print(f"\nMedian Saturday Open Hour for Ice Cream Shops in SF:␣
  {median_open_hour:.2f} hours")
print(f"\nMedian Saturday Close Hour for Ice Cream Shops in SF:␣
  {median_close_hour:.2f}  hours")
print(f"\nMedian Saturday Operation Hours for Ice Cream Shops in SF:␣
  {median_saturday_hours:.2f}  hours")
```

Median Saturday Open Hour for Ice Cream Shops in SF: 11.00 hours

Median Saturday Close Hour for Ice Cream Shops in SF: 21.00 hours

Median Saturday Operation Hours for Ice Cream Shops in SF: 10.00 hours

4 Suggestion using Google AI API

[43]:
```
pip install -q -U google-genai
```

Note: you may need to restart the kernel to use updated packages.

[44]:
```
## Test Code
from google import genai
client = genai.Client(api_key="##get your key")

### Validate Test Code
try:

    response = client.models.generate_content(
        model="gemini-1.5-flash",
        contents="Explain how AI works"
    )

    # If the code executes successfully, print valid
    print("Test code is valid.")

except Exception as e:
    # If there is an error, print invalid and show the error message
    print("Test code is invalid. Error:", e)
```

Test code is valid.

4.1 Method 1: Direct Output Responses

[46]:
```
from google import genai
client = genai.Client(api_key="##get your key ")

response = client.models.generate_content(
    model="gemini-1.5-flash", contents=
    """Provide business strategy for opening an icecream shop in San Francisco.␣
␣You should consider following information:
    1. According to heatmap, the place with traffic-heavy place are Inner␣
␣Richmand, Mission District, Union Square, and Financial District.
    2. According to heatmap, the place with most popular icecream are Inner␣
␣Richmand, Middle Richmand, North Beach.
    3. The average operating hours is 9.41, and the most frequent opening time␣
␣is 11:00, the most frequent closing time is 21:00.
    4. The regional data shows that Top50 rating icecream are in: Mission: 19,␣
␣Downtown: 15, North Beach: 9, Richmond: 8, Nob Hill: 4, Marina: 4, Sunset:␣
␣2, Financial District: 1
    """
)
```

```
print(response.text)
```

Business Strategy: Ice Cream Shop in San Francisco

This strategy leverages the provided data to maximize the chances of success for a new ice cream shop in San Francisco.

I. Target Market & Location Selection:

The data reveals a clear tension: high foot traffic areas (Inner Richmond, Mission District, Union Square, Financial District) don't always correlate with high ice cream demand (Inner Richmond, Middle Richmond, North Beach). Therefore, a nuanced approach is needed.

Option 1 (High Traffic, Moderate Competition):

* **Location:** Inner Richmond. This area balances high foot traffic with existing ice cream popularity, suggesting a receptive market. It avoids the extreme competition of the Mission District.
* **Strategy:** Focus on differentiation. Instead of directly competing on price or standard flavors, specialize in a unique niche (e.g., organic, vegan, artisanal, international flavors). Strong branding and social media marketing will be crucial to stand out.

Option 2 (High Demand, Higher Competition):

* **Location:** Mission District. This area boasts high ice cream popularity and significant foot traffic, but requires a more aggressive competitive strategy.
* **Strategy:** Offer premium ice cream with exceptional quality and potentially higher price points. Invest heavily in creating a memorable brand experience (e.g., unique atmosphere, exceptional customer service). Partner with local businesses or events for cross-promotion.

Option 3 (Strategic Niche):

* **Location:** Financial District. Despite lower ice cream popularity ratings, this location offers immense foot traffic, primarily during weekday business hours.
* **Strategy:** Cater to the working professional. Offer quick service, convenient options (e.g., single scoops, pre-packaged items), and potentially catering services for office lunches or events. Opening hours should prioritize weekday lunch and after-work periods (adjusting slightly from the average).

II. Operational Strategy:

* **Hours of Operation:** While the average is 9.41 hours, consider adjusting

based on the chosen location. The 11:00 AM to 9:00 PM range is a good starting point but might need adjustments (e.g., earlier opening for the Financial District, later closing for areas with evening activity). Weekend hours should likely be extended.
* **Menu:** Offer a core menu of classic flavors along with seasonal specials and unique offerings tied to the chosen niche (e.g., unique vegan options, locally sourced ingredients, international flavor profiles).
* **Pricing:** Conduct thorough market research to determine competitive pricing. Consider premium pricing if offering a high-quality, differentiated product.
* **Supply Chain:** Establish reliable relationships with high-quality ice cream suppliers and ensure efficient inventory management.
* **Staffing:** Hire friendly and efficient staff capable of handling peak periods effectively. Training on customer service and product knowledge is crucial.

III. Marketing & Sales:

* **Branding:** Develop a strong brand identity reflecting the chosen niche and target market.
* **Online Presence:** Build a user-friendly website and active social media presence (Instagram is particularly relevant for ice cream). Utilize online ordering and delivery services.
* **Local Partnerships:** Collaborate with nearby businesses, event organizers, and community groups for cross-promotion.
* **Loyalty Programs:** Implement a loyalty program to incentivize repeat business.
* **Public Relations:** Seek opportunities to get positive media coverage in local publications and blogs.

IV. Financial Planning:

* **Detailed Budget:** Create a comprehensive budget encompassing startup costs, operating expenses, and projected revenue.
* **Funding Sources:** Explore funding options, including loans, investors, or personal savings.
* **Profitability Analysis:** Conduct thorough financial forecasting to ensure profitability.

V. Risk Mitigation:

* **Competition:** Analyze existing competitors to understand their strengths and weaknesses. Develop a clear differentiation strategy.
* **Seasonality:** San Francisco enjoys relatively mild weather, but sales might fluctuate seasonally. Plan accordingly with adjusted menus and promotions.

* **Location Specific Risks:** Consider factors such as rent costs, permits, and potential construction or events impacting foot traffic.

This strategy provides a framework. Detailed market research specific to the chosen location is crucial before making final decisions. Thorough competitive analysis and a robust financial plan are paramount for success.

4.2 Method 2: Gradio UI + Direct Prompt

[48]:
```
%%capture
!pip install gradio google-generativeai
import gradio as gr
from google import genai
import pandas as pd
from bs4 import BeautifulSoup
```

[49]:
```
client = genai.Client(api_key="##get your key ")
def generate_suggestions(user_prompt):
    """Provide business strategy for opening an icecream shop in San Francisco. You should consider following information:
    1. According to heatmap, the place with traffic-heavy place are Inner Richmand, Mission District, Union Square, and Financial District.
    2. According to heatmap, the place with most popular icecream are Inner Richmand, Middle Richmand, North Beach.
    3. The average operating hours is 9.41, and the most frequent opening time is 11:00, the most frequent closing time is 21:00.
    4. The regional data shows that Top50 rating icecream are in: Mission: 19, Downtown: 15, North Beach: 9, Richmond: 8, Nob Hill: 4, Marina: 4, Sunset: 2, Financial District: 1
    """

    try:
        prompt = f"""{user_prompt}""" # Embed the user's prompt directly.

        response = client.models.generate_content(
            model="gemini-1.5-flash",
            contents=prompt,
        )

        return response.text # Return the generated text.

    except Exception as e:
        return f"Error generating suggestions: {e}"    # Handle errors.
```

```
iface = gr.Interface(
    fn=generate_suggestions,
    inputs=gr.Textbox(label="Your Prompt/Request",
                      placeholder="Summarize business strategy for opening an
 icecream shop in San Francisco downtown according to given information."),
 # Clearer input label and placeholder
    outputs="text",
    title="Gemini Ice Cream Shop Suggestion Generator",
    description="Get business strategies, ideas, and suggestions for your ice
 cream shop.  Enter your prompt or request below.",    # Improved description
)

iface.launch()
```

```
* Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

<IPython.core.display.HTML object>

[49]:

4.3 Method 3: Gradio UI + Input File (Future Work)

[51]:
```
def generate_suggestions(html_input, text_input, df_file, avg_operating_hours,
 shop_concept, target_audience, budget):
    prompt = f"""## Ice Cream Shop Planning - Seeking Suggestions

I'm planning to open an ice cream shop and need suggestions. I have the
 following information:

**Shop Concept:** {shop_concept}

**Target Audience:** {target_audience}

**Budget:** ${budget}

**Average Competitor Operating Hours:** {avg_operating_hours}

"""

    if df_file:   # Check if a file was uploaded
        try:
            df = pd.read_csv(df_file.name)
            df_string = df.to_markdown(index=False)
            prompt += f"**Competitor Data (from DataFrame):
 **\n```\n{df_string}\n```\n"
```

```
            except Exception as e:
                return f"Error parsing DataFrame: {e}"

        if html_input: # Check if HTML input was provided
            try:
                soup = BeautifulSoup(html_input, "html.parser")
                extracted_text = soup.get_text(strip=True)
                prompt += f"**Population Heatmap (from HTML):**\n{extracted_text}\n"
            except Exception as e:
                return f"Error parsing HTML: {e}"

        if text_input: # Check if text input was provided
            prompt += f"**Additional Information:**\n{text_input}\n"

        prompt += """
        """
```

[52]:
```
iface = gr.Interface(
    fn=generate_suggestions,
    inputs=[
        gr.Textbox(label="HTML Content "),
        gr.Textbox(label="Additional Text Information"),
        gr.File(label="DataFrame (Upload CSV)"),
        gr.Number(label="Average Competitor Operating Hours"),
        gr.Textbox(label="Shop Concept"),
        gr.Textbox(label="Target Audience"),
        gr.Number(label="Budget"),
    ],
    outputs="text",
    title="Gemini Ice Cream Shop Suggestion Generator",
    description="Generate suggestions for your ice cream shop using Gemini.",
)

iface.launch()
```

```
* Running on local URL:  http://127.0.0.1:7861

To create a public link, set `share=True` in `launch()`.
```

<IPython.core.display.HTML object>

[52]:

Prompt:

- Additional Text Information: The place with most popular icecream is Union Square and Hayes Valley area.
- Average Competitor Operating Hours: 9.8
- Target Audience: Young people between 20 to 30 years old.

- HTML Link: https://sf.eater.com/maps/best-ice-cream-san-francisco