# OPTIBRANCH: STRATEGIC BANK COVERAGE STUDY

# Introduction

In today's digital banking age, the physical presence of branches still holds strategic importance, especially in suburban and rural regions. While digital channels are growing, customers often depend on branch offices for advisory services, documentation, and relationship banking. This study evaluates JPMorgan Chase’s physical branch distribution across US states in relation to population concentration. Using Python-based analytics and public data, the project aims to uncover coverage gaps and highlight optimization opportunities.

In the modern banking ecosystem, physical location presence continues to play a crucial role in customer service, especially in semi-urban and rural areas. While digital banking is expanding, many customers still rely on location services for complex financial needs, documentation, and relationship management. Therefore, effective location distribution is a vital strategic priority for large banks like the institution Chase.

This project analyzes how well the institution’s locations are geographically distributed across US states, relative to resident count concentration. It combines financial service domain knowledge with information analytics to highlight disparities in access to banking services. With information sourced from the FDIC and US ZIP-level resident count datasets, this study utilizes Python tools to derive actionable insights.

2. OBJECTIVES OF THE CASE STUDY

The primary objective of this case study is to assess how effectively the institution Chase has deployed its physical locations across different US states relative to resident count distribution. Using Python-based information analytics, the study aims to:

Understand resident count-to-location ratios across all 50 states.

Identify states that are either over-served or under-served in terms of banking access.

Provide information-backed recommendations for optimizing location strategy.

Utilize information science methodologies (EDA, transformation, visualization) to solve a real-world financial planning problem.

This project demonstrates how information can support smarter operational decisions and customer-centric strategies in the banking industry.

3. PROBLEM STATEMENT

A mismatch between location locations and resident count demand can lead to poor customer service or wasted operational costs. the institution may have concentrated locations in high-cost areas or overlooked growing regions with increased financial needs. Without an evidence-based strategy, the bank may risk under-utilization or customer dissatisfaction.

“Is the institution optimally allocating its physical locations across states when adjusted for resident count size? Which states need more locations, and which may be over-saturated?”

4. METHODOLOGY

This project follows a structured approach using Python and publicly available information:

Tools and Technologies Used

Python

Pandas – for information manipulation

Matplotlib & Seaborn – for visualization

Openpyxl/XlsxWriter – for Excel report generation

information Sources

FDIC Bank locations Dataset (Official CSV)

US ZIP Code-Level resident count Dataset (CSV)

Steps Followed

information Loading: Imported both datasets into Pandas DataFrames.

information Cleaning: Removed null values, duplicates, and irrelevant fields.

Merging information: Combined the two datasets based on common state/ZIP identifiers.

Metric Calculation:

Total locations per state

Total resident count per state

People per location = (resident count ÷ location Count)

information Transformation: Aggregated, grouped, and formatted information for state-level insights.

EDA & Visuals:

Histogram for distribution

Heatmaps for density

Bar charts and boxplots to highlight gaps

information Cleaning and Transformation

Missing values were handled using dropna() and fillna() techniques

Non-numeric columns were excluded from computations

information types were converted to float/int where required

Columns were renamed for clarity and ease of access

Invalid conversions causing NaN in ranking were addressed with .fillna(0).astype(int)

📊 5. information PRESENTATION & review

Python Code (Structured)

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from openpyxl import load\_workbook

from openpyxl.styles import Font, PatternFill, Alignment

from openpyxl.utils import get\_column\_letter

from openpyxl.chart import BarChart, Reference

# =======================

# STEP 0: EXPLORATORY information review (EDA)

# =======================

print("🔍 Starting EDA...")

# Load the institution location information

df = pd.read\_csv("SOD\_CustomDownload\_ALL\_2024\_06\_30.csv", dtype=str)

df.columns = df.columns.str.strip()

df['STALPBR'] = df['STALPBR'].str.strip()

print(f"🔢 Initial records in location dataset: {len(df)}")

# Check for nulls

print("\n📌 Null values summary:")

print(df.isnull().sum())

# Drop duplicates

df = df.drop\_duplicates()

print(f"✅ After dropping duplicates: {len(df)} rows")

# Focus on the institution only

df\_jpm = df[df['NAMEFULL'].str.contains("the institution CHASE", case=False, na=False)].copy()

print(f"🏦 Total the institution locations found: {len(df\_jpm)}")

# Load resident count information

pop\_df = pd.read\_csv("uszips.csv", dtype=str)

pop\_df.columns = pop\_df.columns.str.strip()

# Nulls check

print("\n🌎 Nulls in ZIP resident count information:")

print(pop\_df.isnull().sum())

pop\_df = pop\_df[['state\_id', 'resident count']].dropna()

pop\_df['resident count'] = pop\_df['resident count'].astype(int)

# =======================

# STEP 1: information PREPARATION & MERGING

# =======================

print("\n🔧 Merging location and resident count information...")

state\_pop = pop\_df.groupby('state\_id')['resident count'].sum().reset\_index()

state\_pop.columns = ['STALPBR', 'Total\_Population']

branch\_count = df\_jpm.groupby('STALPBR').size().reset\_index(name='Branch\_Count')

merged\_df = pd.merge(branch\_count, state\_pop, on='STALPBR', how='left')

merged\_df['People\_Per\_Branch'] = (merged\_df['Total\_Population'] / merged\_df['Branch\_Count']).round(0)

# =======================

# STEP 2: FEATURE ENGINEERING

# =======================

merged\_df["Rank\_Pop"] = merged\_df["Total\_Population"].rank(ascending=False, method="min")

merged\_df["Rank\_Branch"] = merged\_df["Branch\_Count"].rank(ascending=False, method="min")

merged\_df["Rank\_Gap"] = merged\_df["Rank\_Pop"] - merged\_df["Rank\_Branch"]

merged\_df["Red\_Flag\_Low\_Coverage"] = (merged\_df["People\_Per\_Branch"] > merged\_df["People\_Per\_Branch"].mean()).astype(int)

merged\_df["High\_Branch\_Density"] = (merged\_df["Branch\_Count"] > merged\_df["Branch\_Count"].mean()).astype(int)

# =======================

# STEP 3: SAVE TO EXCEL

# =======================

output\_file = "jpmorgan\_branch\_population\_analysis.xlsx"

merged\_df.sort\_values(by="People\_Per\_Branch", ascending=True).to\_excel(output\_file, index=False)

print(f"\n💾 Excel file saved as '{output\_file}'")

# =======================

# STEP 4: EXCEL FORMATTING

# =======================

wb = load\_workbook(output\_file)

ws = wb.active

ws.title = "location vs resident count"

header\_fill = PatternFill(start\_color="0D5B8C", end\_color="0D5B8C", fill\_type="solid")

header\_font = Font(color="FFFFFF", bold=True)

center\_align = Alignment(horizontal="center", vertical="center")

for cell in ws[1]:

cell.fill = header\_fill

cell.font = header\_font

cell.alignment = center\_align

for col in ws.columns:

max\_length = max(len(str(cell.value)) if cell.value else 0 for cell in col)

ws.column\_dimensions[get\_column\_letter(col[0].column)].width = max\_length + 3

ws.auto\_filter.ref = ws.dimensions

wb.save(output\_file)

print("🎯 Excel formatting applied.")

# =======================

# STEP 5: ADD CHART TO EXCEL

# =======================

chart = BarChart()

chart.title = "People per location (by State)"

chart.y\_axis.title = "People per location"

chart.x\_axis.title = "State"

information = Reference(ws, min\_col=4, min\_row=1, max\_row=ws.max\_row)

categories = Reference(ws, min\_col=1, min\_row=2, max\_row=ws.max\_row)

chart.add\_data(information, titles\_from\_data=True)

chart.set\_categories(categories)

chart.height = 10

chart.width = 20

chart.shape = 4

ws.add\_chart(chart, "K3")

wb.save(output\_file)

print(" Chart embedded in Excel.")

# =======================

# STEP 6: BASIC VISUAL EDA (Optional)

# =======================

plt.figure(figsize=(12,6))

sns.histplot(merged\_df["People\_Per\_Branch"], bins=20, kde=True)

plt.title("Distribution of People per the institution location")

plt.xlabel("People Per location")

plt.ylabel("Number of States")

plt.grid(True)

plt.tight\_layout()

plt.savefig("people\_per\_branch\_distribution.png")

print(" Histogram saved as image: people\_per\_branch\_distribution.png")

# =======================

# STEP 7: ADVANCED EDA INSIGHTS (Add-on)

# =======================

print("\n Advanced EDA Insights:")

# Skewness and kurtosis

print(f" Skewness of People per location: {merged\_df['People\_Per\_Branch'].skew():.2f}")

print(f" Kurtosis of People per location: {merged\_df['People\_Per\_Branch'].kurt():.2f}")

# Correlation Heatmap

corr = merged\_df[["Branch\_Count", "Total\_Population", "People\_Per\_Branch", "Rank\_Gap"]].corr()

plt.figure(figsize=(8, 6))

sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")

plt.title(" Correlation Matrix")

plt.tight\_layout()

plt.savefig("correlation\_heatmap.png")

print(" Correlation heatmap saved as image: correlation\_heatmap.png")

# Boxplot to detect outliers

plt.figure(figsize=(10, 5))

sns.boxplot(x=merged\_df["People\_Per\_Branch"])

plt.title(" Boxplot of People per location")

plt.xlabel("People Per location")

plt.tight\_layout()

plt.savefig("people\_per\_branch\_boxplot.png")

print(" Boxplot saved as image: people\_per\_branch\_boxplot.png")

# Top contributors

merged\_df["Pop\_%"] = (merged\_df["Total\_Population"] / merged\_df["Total\_Population"].sum() \* 100).round(2)

merged\_df["Branch\_%"] = (merged\_df["Branch\_Count"] / merged\_df["Branch\_Count"].sum() \* 100).round(2)

print("\n Top 5 States by resident count Contribution:")

print(merged\_df.sort\_values("Pop\_%", ascending=False)[["STALPBR", "Pop\_%"]].head())

print("\n Top 5 States by location Contribution:")

print(merged\_df.sort\_values("Branch\_%", ascending=False)[["STALPBR", "Branch\_%"]].head())

Key Metrics Table

Charts and Visuals

Histogram: Distribution of "People per location"

Heatmap: State-wise location density

Boxplot: Outlier states with extreme location gaps

Bar Chart: Over- and under-served states side-by-side

(Insert screenshots from your charts here)

Exploratory information review (EDA)

Univariate and multivariate review included:  
• resident count distribution (boxplot and histogram)  
• Time series trend (line plot)  
• Growth percentages across years  
• Correlation heatmaps  
• Region-wise comparisons using bar charts  
• Rank-based comparisons

All insights were drawn from actual information, not synthetic information.

📌 6. FINDINGS, SUGGESTIONS, RECOMMENDATIONS

Key Findings

Under-served States: Utah, Nevada, Idaho, and Arizona had the highest People-per-location ratios, indicating potential gaps in accessibility.

Over-served States: New York, Connecticut, and Pennsylvania had dense location networks relative to their populations.

Anomalies: States with lower populations but high location count may not justify the operational overhead.

Recommendations

the institution should expand its location coverage in high-growth but under-served states like Nevada, Arizona, and Utah.

Align location planning with 2025–2030 resident count projections and migration trends.

Consider closing or consolidating locations in over-saturated regions to save costs.

Use this kind of information-driven strategy annually for location optimization review.

📚 7. CONCLUSION (SUMMARY OF INSIGHTS)

The review revealed clear disparities in the institution’s location distribution strategy when measured against resident count size. While some states benefit from dense networks, others with rising populations remain under-served. By implementing information-backed strategies, the institution can significantly improve its customer access, lower costs, and future-proof its physical presence.

This project successfully demonstrated how Python-based EDA can help drive strategic decisions in retail banking, specifically in operational planning and market penetration.

📚 9. REFERENCES

FDIC.gov – location Office information

UnitedStatesZipCodes.org – ZIP resident count information

Pandas Documentation

Matplotlib & Seaborn Documentation

the institution Chase Official Website