Importing Pandas and uploading file

In [1]:
```
import pandas as pd
df = df = pd.read_csv(r"C:\Users\shahz\Downloads\Greetings H acquisition Insur
df.head()
```

Out[1]:

	Country	Year	Month	Health insurance category	Revenue (£000)	Customer volume	Unnamed: 6	Unnamed: 7
0	India	2022	1	Basic Health Insurance	4,639	5,944	NaN	NaN
1	India	2022	1	Comprehensive Health Insurance	10,376	15,788	NaN	NaN
2	India	2022	1	Critical Illness Insurance	20,631	28,605	NaN	NaN
3	India	2022	1	Dental Insurance	5,860	29,162	NaN	NaN
4	India	2022	1	Vision Insurance	5,249	14,860	NaN	NaN

In [2]:
```
df['Health insurance category'] = df['Health insurance category'].replace({
    'Comprehensive Health Ins$$$$$$$': 'Comprehensive Health Insurance'
})
```

In [3]:
```
print(df['Health insurance category'].unique())
```

```
['Basic Health Insurance ' 'Comprehensive Health Insurance'
 'Critical Illness Insurance' 'Dental Insurance' 'Vision Insurance'
 'Travel Health Insurance' 'Vision Insurance ']
```

Check missing values

In [4]:
```
df.isnull().sum()
df.head()
```

Out[4]:

	Country	Year	Month	Health insurance category	Revenue (£000)	Customer volume	Unnamed: 6	Unnamed: 7
0	India	2022	1	Basic Health Insurance	4,639	5,944	NaN	NaN
1	India	2022	1	Comprehensive Health Insurance	10,376	15,788	NaN	NaN
2	India	2022	1	Critical Illness Insurance	20,631	28,605	NaN	NaN
3	India	2022	1	Dental Insurance	5,860	29,162	NaN	NaN
4	India	2022	1	Vision Insurance	5,249	14,860	NaN	NaN

Dropping rows with missing values and fixing column names

In [5]:
```
df = df.drop_duplicates()
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')
df = df.map(lambda x: x.strip().title() if isinstance(x, str) else x)
df = df.dropna(subset=['revenue_(£000)', 'customer_volume'])  # Replace with y
df.head()
```

Out[5]:

	country	year	month	health_insurance_category	revenue_(£000)	customer_volum
0	India	2022	1	Basic Health Insurance	4,639	5,94
1	India	2022	1	Comprehensive Health Insurance	10,376	15,78
2	India	2022	1	Critical Illness Insurance	20,631	28,60
3	India	2022	1	Dental Insurance	5,860	29,16
4	India	2022	1	Vision Insurance	5,249	14,86

In [6]:
```
before = df.shape[0]

df = df[~((df['revenue_(£000)'] == '-') & (df['customer_volume'] == '-'))]

after = df.shape[0]
print("Rows removed:", before - after)
```

```
Rows removed: 5
```

Removing negative values from 'revenue_(£000)' and 'customer_volume' column

In [7]:
```
# Step 1: Strip spaces and commas
df['revenue_(£000)'] = df['revenue_(£000)'].astype(str).str.replace(',', '').s
df['customer_volume'] = df['customer_volume'].astype(str).str.replace(',', '')

# Step 2: Convert to numeric
df['revenue_(£000)'] = pd.to_numeric(df['revenue_(£000)'], errors='coerce')
df['customer_volume'] = pd.to_numeric(df['customer_volume'], errors='coerce')
```

In [8]:
```
before = df.shape[0]

# Drop rows where both are negative or NaN
df = df[(df['revenue_(£000)'] >= 0) & (df['customer_volume'] >= 0)]

after = df.shape[0]
print("Rows before:", before)
print("Rows after:", after)
print("Rows removed:", before - after)
```

```
Rows before: 643
Rows after: 643
Rows removed: 0
```

In [9]:
```
df = df.dropna(axis=1, how='all')
df.shape
```

Out[9]: (643, 6)

In [10]:
```
print(" Descriptive statistics:")
print(df.describe())
```

```
📊 Descriptive statistics:
              year       month  revenue_(£000)  customer_volume
count   643.000000  643.000000      643.000000       643.000000
mean   2023.001555    6.496112    16292.748056     38584.213064
std       0.817448    3.447592    12119.026938     40180.701917
min    2022.000000    1.000000      195.000000       185.000000
25%    2022.000000    4.000000     5981.000000     11145.000000
50%    2023.000000    6.000000    14405.000000     26562.000000
75%    2024.000000    9.000000    23561.000000     50616.500000
max    2024.000000   12.000000    59577.000000    235345.000000
```

In [11]:
```
print(" 📃 Column names in the dataset:")
print(df.columns.tolist())
```

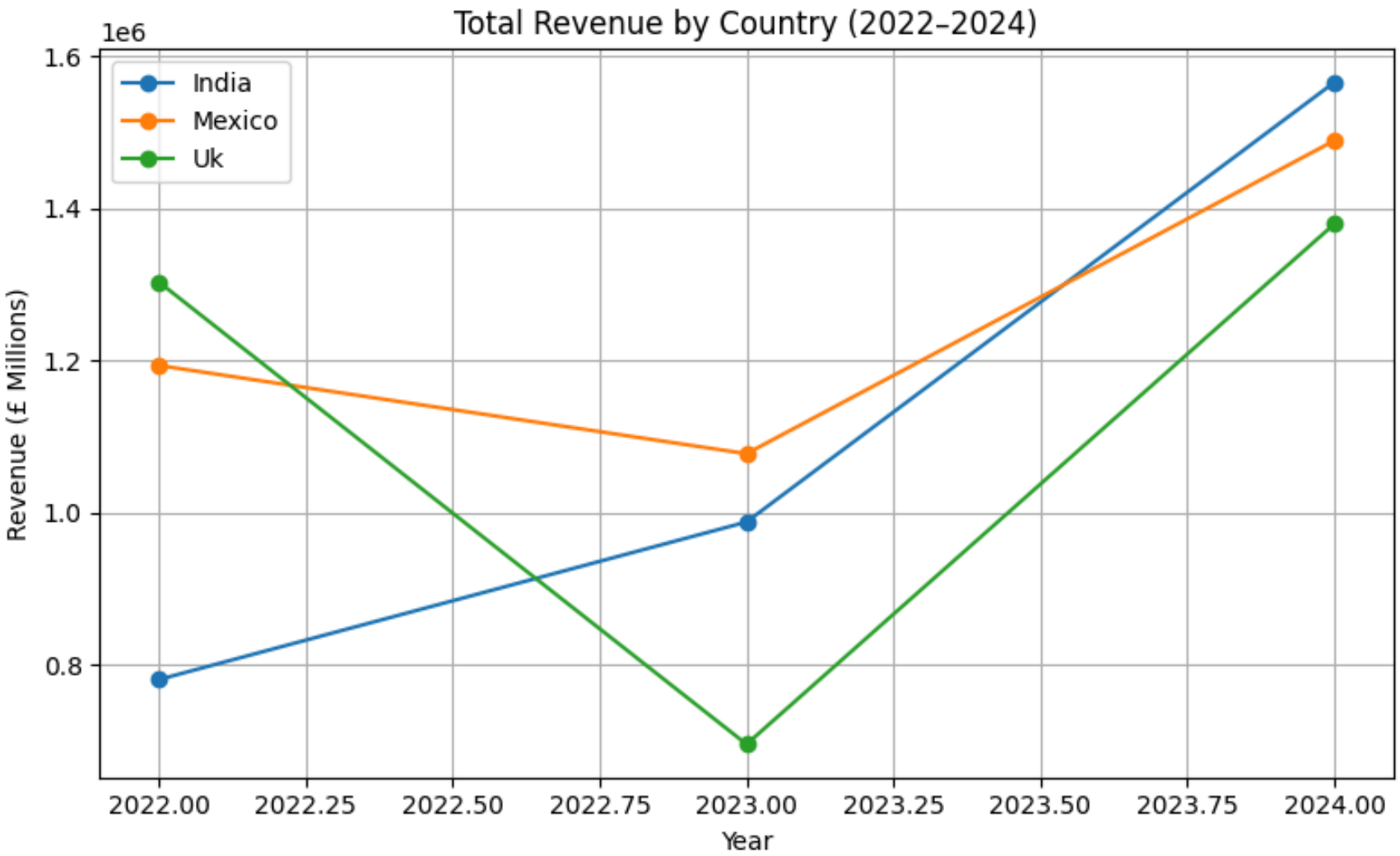
```
📃 Column names in the dataset:
['country', 'year', 'month', 'health_insurance_category', 'revenue_(£000)', 'cu
stomer_volume']
```

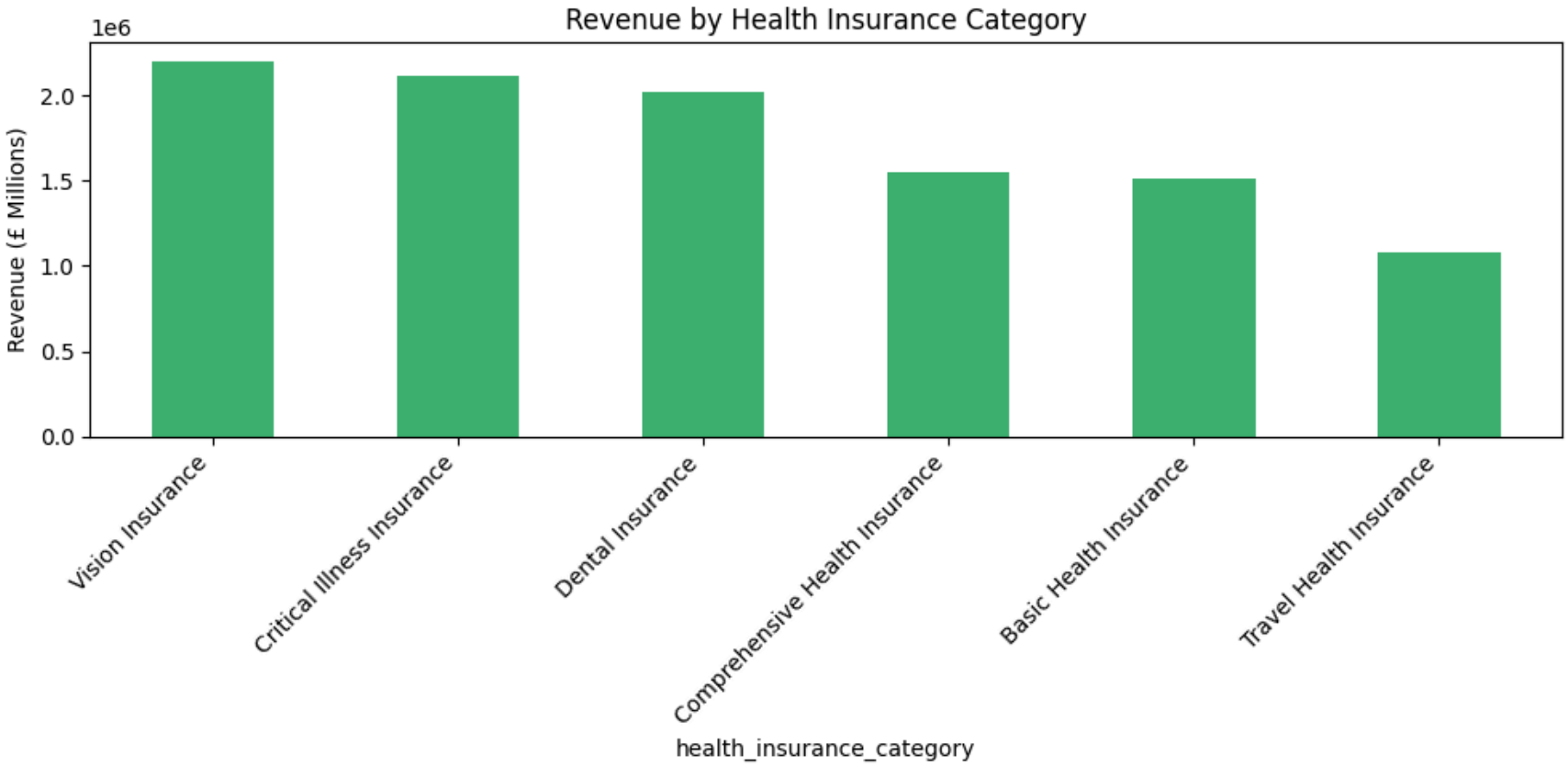
In [19]:
```
df.to_excel('cleaned_insurance_data.xlsx', index=False)
```

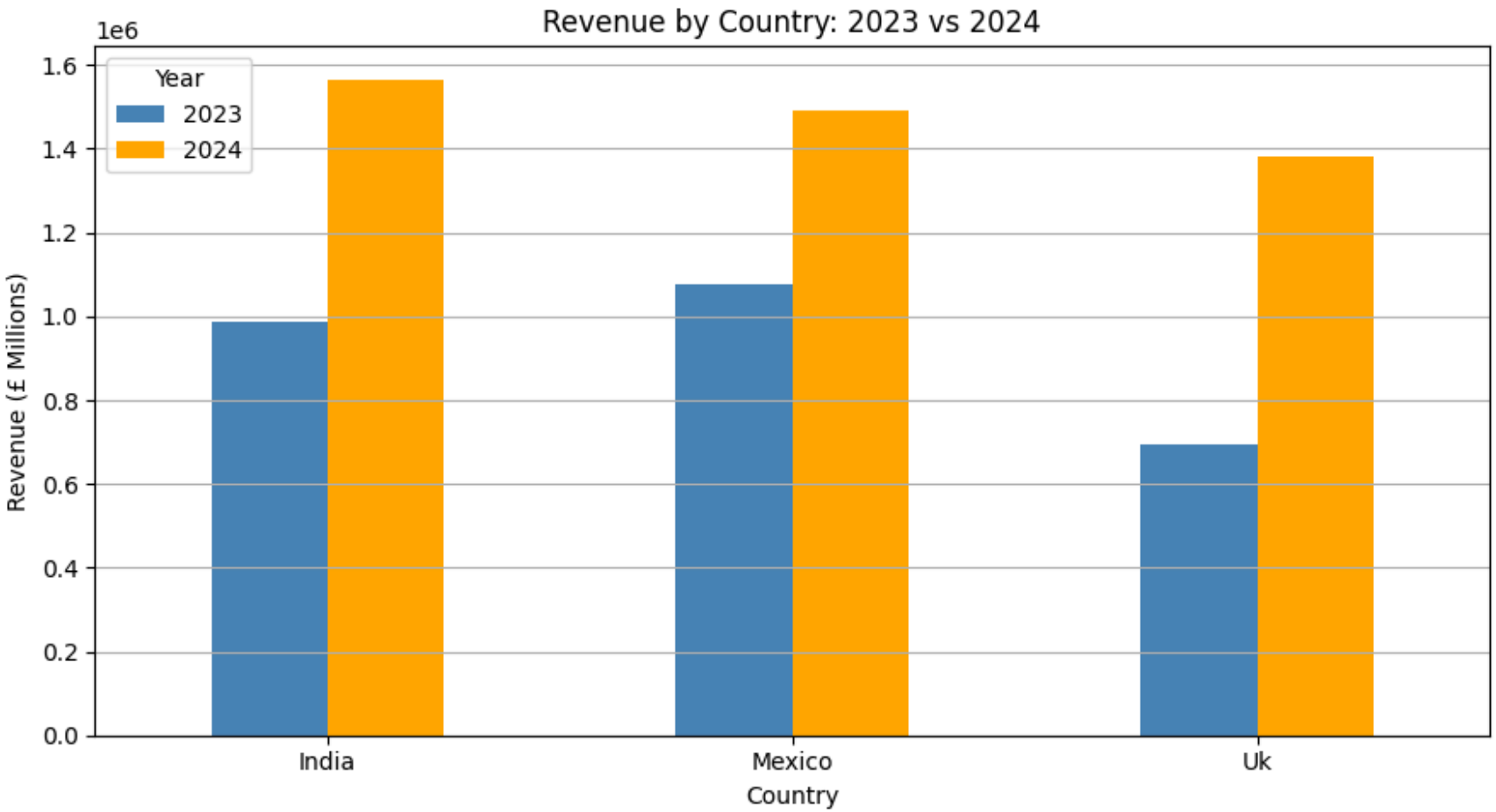
In [12]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker


# ------------------  1. Revenue by Country (2022–2024) ------------------
countries = df['country'].unique()
country_year = df.groupby(['country', 'year'])[['revenue_(£000)', 'customer_vo


plt.figure(figsize=(8, 5))
for country in countries:
    temp = country_year[country_year['country'] == country]
    plt.plot(temp['year'], temp['revenue_(£000)'], marker='o', label=country)

plt.title('Total Revenue by Country (2022–2024)')
plt.xlabel('Year')
plt.ylabel('Revenue (£ Millions)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# ------------------  2. Revenue by Health Insurance Category ----------------
product_summary = df.groupby('health_insurance_category')[['revenue_(£000)', '
```

```
product_summary_sorted = product_summary.sort_values(by='revenue_(£000)', asce

plt.figure(figsize=(10, 5))
product_summary_sorted['revenue_(£000)'].plot(kind='bar', color='mediumseagree
plt.title('Revenue by Health Insurance Category')
plt.ylabel('Revenue (£ Millions)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# ------------------  3. Rebranding Impact (2023 vs 2024) ------------------
# Filter only 2023 and 2024 data
df_23_24 = df[df['year'].isin([2023, 2024])]

# Group by Country & Year
country_rebrand = df_23_24.groupby(['country', 'year'])['revenue_(£000)'].sum(

# Plot grouped bar chart
country_rebrand.plot(kind='bar', figsize=(9, 5), color=['steelblue', 'orange']


plt.title('Revenue by Country: 2023 vs 2024')
plt.xlabel('Country')
plt.ylabel('Revenue (£ Millions)')
plt.xticks(rotation=0)
plt.legend(title='Year')
plt.grid(axis='y')
```

```
plt.tight_layout()
plt.show()
```




Revenue by Health Insurance Category
1e6
2.0
1.5
1.0
0.5
0.0
Revenue (£ Millions)
Vision Insurance
Critical Illness Insurance
Dental Insurance
Comprehensive Health Insurance
Basic Health Insurance
Travel Health Insurance
health_insurance_category



In [16]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Pivot and calculate metrics
category_pivot = df.pivot_table(
    index='health_insurance_category',
    columns='year',
```

```
    values='revenue_(£000)',
    aggfunc='sum'
).reset_index()

category_pivot.columns.name = None
category_pivot = category_pivot.rename(columns={
    'health_insurance_category': 'Category',
    2022: '2022 Revenue',
    2023: '2023 Revenue',
    2024: '2024 Revenue'
})

category_pivot['Growth Rate (%)'] = (
    (category_pivot['2024 Revenue'] - category_pivot['2022 Revenue']) /
    category_pivot['2022 Revenue']
) * 100

category_pivot['Average Revenue'] = category_pivot[
    ['2022 Revenue', '2023 Revenue', '2024 Revenue']
].mean(axis=1)

category_pivot = category_pivot.round(2)

# Plot visual table
fig, ax = plt.subplots(figsize=(14, 0.6 * len(category_pivot) + 2))
ax.axis('tight')
ax.axis('off')
```

```
table = ax.table(
    cellText=category_pivot.values,
    colLabels=category_pivot.columns,
    loc='center',
    cellLoc='center'
)

table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.5)

# Optional: Fit wide text
try:
    table.auto_set_column_width(col=list(range(len(category_pivot.columns))))
except:
    pass

#  Title closer to table
plt.title('Insurance Category Performance (2022-2024)', fontsize=14, fontweigh

#  Skip tight_layout for cleaner spacing
plt.show()

import matplotlib.pyplot as plt

# STEP 1: Group by Country & Year
```

```
summary = df.groupby(['country', 'year'])[['revenue_(£000)', 'customer_volume'

# STEP 2: Rename columns
summary.columns = ['Country', 'Year', 'Revenue (£000)', 'Customer Volume']

# STEP 3: Revenue Growth (YoY) per Country
summary['Revenue Growth (%)'] = summary.groupby('Country')['Revenue (£000)'].p
summary['Revenue Growth (%)'] = summary['Revenue Growth (%)'].round(2)

# STEP 4: Market Share (%)
total_revenue_per_year = summary.groupby('Year')['Revenue (£000)'].transform('
summary['Market Share (%)'] = (summary['Revenue (£000)'] / total_revenue_per_y
summary['Market Share (%)'] = summary['Market Share (%)'].round(2)

# STEP 5: Visual Table
fig_height = 0.5 * len(summary) + 2
fig, ax = plt.subplots(figsize=(12, fig_height))
ax.axis('off')

# Title
ax.text(0.5, 1.01, 'Country Performance (2022–2024)', fontsize=14, fontweight=
        ha='center', va='bottom', transform=ax.transAxes)

# Table
table = ax.table(
    cellText=summary.values,
    colLabels=summary.columns,
```

```
    cellLoc='center',
    loc='center'
)

table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.3)

plt.tight_layout()
plt.show()
import calendar
import matplotlib.pyplot as plt

# STEP 1: Add Quarter and Month Name
monthly_summary = df.copy()
monthly_summary['Quarter'] = monthly_summary['month'].astype(int).apply(lambda
monthly_summary['Month Name'] = monthly_summary['month'].astype(int).apply(lam

# STEP 2: Group and summarize
summary = monthly_summary.groupby(['year', 'Quarter', 'Month Name'])[['revenue

# STEP 3: Clean column names
summary = summary.rename(columns={
    'year': 'Year',
    'Month Name': 'Month',
    'revenue_(£000)': 'Total Revenue (£000)',
    'customer_volume': 'Customer Volume'
```

```
})

# STEP 4: Add YoY Growth
summary['YoY Growth (%)'] = summary.groupby(['Month'])['Total Revenue (£000)']
summary['YoY Growth (%)'] = summary['YoY Growth (%)'].round(2)

# Reorder columns
summary = summary[['Year', 'Quarter', 'Month', 'Total Revenue (£000)', 'YoY Gr

# STEP 5: Plot as visual table
fig_height = 0.5 * len(summary) + 2
fig, ax = plt.subplots(figsize=(15, fig_height))
ax.axis('off')

#  Title placed using ax.text() for better spacing
ax.text(0.5, 1.03, 'Monthly/Quarterly Trends (2022–2024)', fontsize=14, fontwe
        ha='center', transform=ax.transAxes)

# Render table
table = ax.table(
    cellText=summary.values,
    colLabels=summary.columns,
    loc='center',
    cellLoc='center',
    colLoc='center'
)
```

```
# Style
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.3)

plt.subplots_adjust(top=0.9)  # ⬅️ Pulls title closer to the table
plt.show()
# Convert month number to full month name (if not already)
df['Month Name'] = df['month'].astype(int).apply(lambda x: calendar.month_name
df['Date'] = pd.to_datetime(df['year'].astype(str) + '-' + df['month'].astype(

# Group monthly revenue by Country
monthly_country = df.groupby(['Date', 'country'])['revenue_(£000)'].sum().rese

# Plot
plt.figure(figsize=(12, 6))
for country in df['country'].unique():
    temp = monthly_country[monthly_country['country'] == country]
    plt.plot(temp['Date'], temp['revenue_(£000)'], marker='o', label=country)

plt.title('Monthly Revenue Trends by Country (2022–2024)', fontsize=14, fontwe
plt.xlabel('Date')
plt.ylabel('Revenue (£000)')
plt.legend(title='Country')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
# Create 'Quarter' column
df['Quarter'] = df['month'].astype(int).apply(lambda x: (x - 1) // 3 + 1)
df['Year-Quarter'] = df['year'].astype(str) + ' Q' + df['Quarter'].astype(str)

# Group by Quarter and Insurance Category
quarterly_cat = df.groupby(['Year-Quarter', 'health_insurance_category'])['rev

# Plot
quarterly_cat.plot(kind='bar', stacked=True, figsize=(14, 6), colormap='Set3')
plt.title('Quarterly Revenue by Health Insurance Category', fontsize=14, fontw
plt.xlabel('Quarter')
plt.ylabel('Revenue (£ Millions)')
plt.xticks(rotation=45)
plt.legend(title='Category', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
# Group by category
category_data = df.groupby('health_insurance_category')[['revenue_(£000)', 'cu

# Bar chart for revenue
fig, ax1 = plt.subplots(figsize=(10, 6))

bars = ax1.bar(category_data['health_insurance_category'], category_data['reve
               color='skyblue', label='Revenue (£000)')

# Line chart for customer volume
ax2 = ax1.twinx()
```

```
line = ax2.plot(category_data['health_insurance_category'], category_data['cus
                color='darkorange', marker='o', label='Customer Volume')

# Titles and labels
ax1.set_ylabel('Revenue (£ Millions)', color='skyblue')  # ✅ Correct
ax2.set_ylabel('Customer Volume', color='darkorange')
plt.title('Revenue vs Customer Volume by Category', fontsize=14, fontweight='b
ax1.set_xticklabels(category_data['health_insurance_category'], rotation=45, h

# Combine legends
lines_labels = bars + line
labels = [bar.get_label() for bar in lines_labels]
plt.legend(lines_labels, labels, loc='upper right')

plt.tight_layout()
plt.show()
```

Insurance Category Performance (2022–2024)

Category	2022 Revenue	2023 Revenue	2024 Revenue	Growth Rate (%)	Average Revenue
Basic Health Insurance	529754	365840	612187	15.56	502593.67
Comprehensive Health Insurance	405473	341587	801927	97.78	516329.0
Critical Illness Insurance	848929	577888	692248	-18.46	706355.0
Dental Insurance	636969	463220	917663	44.07	672617.33
Travel Health Insurance	331364	284701	466854	40.89	360973.0
Vision Insurance	526059	728598	944976	79.63	733211.0

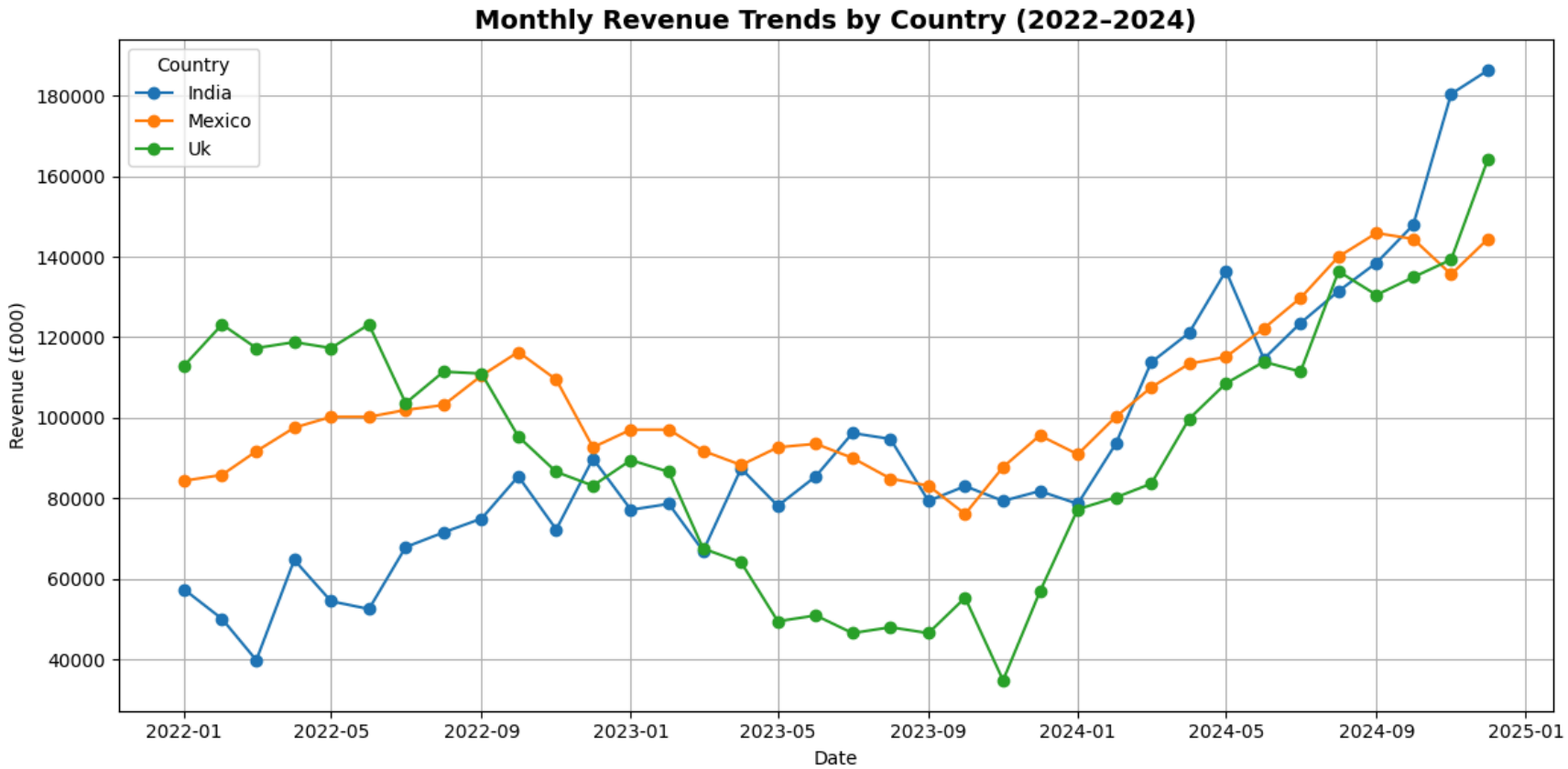
Country Performance (2022–2024)

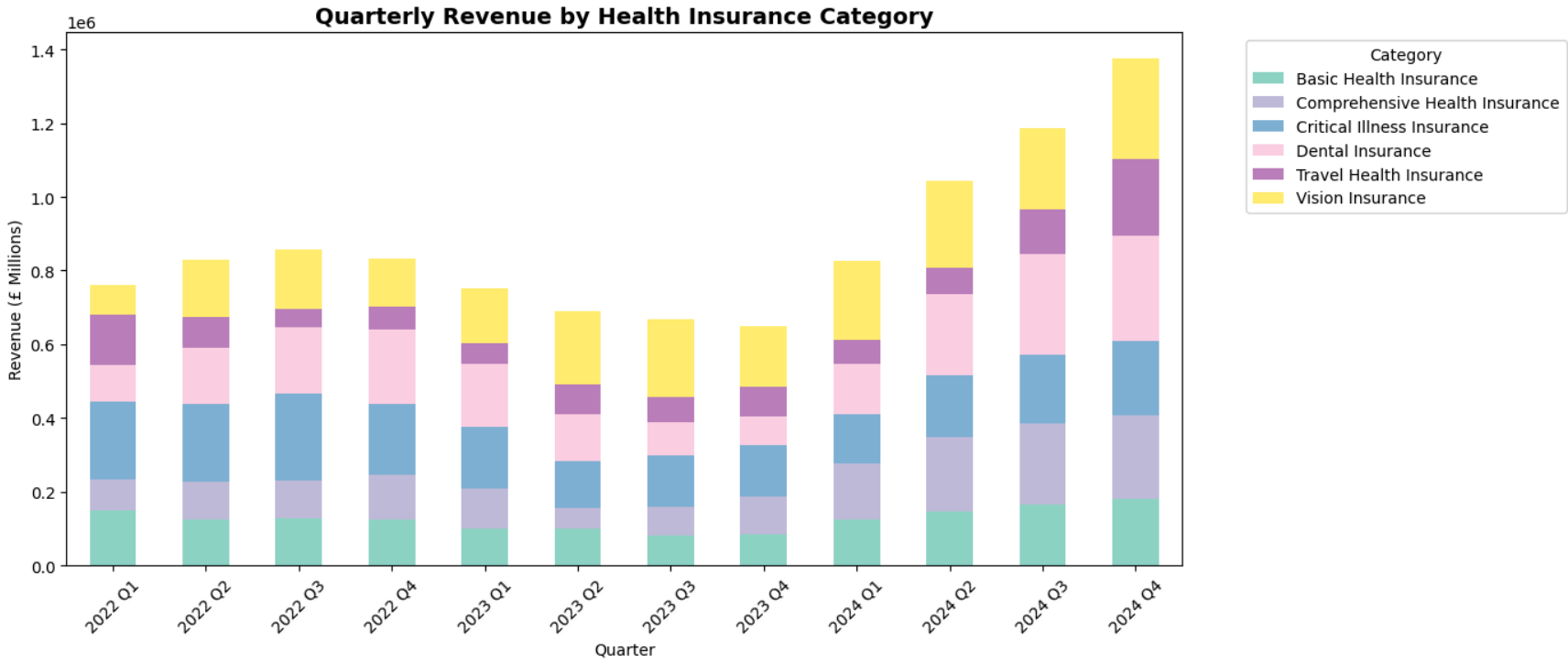
Country	Year	Revenue (£000)	Customer Volume	Revenue Growth (%)	Market Share (%)
India	2022	780801	1709986	nan	23.82
India	2023	988098	2413238	26.55	35.78
India	2024	1566244	4008827	58.51	35.31
Mexico	2022	1193911	2765420	nan	36.42
Mexico	2023	1077880	2730684	-9.72	39.03
Mexico	2024	1489596	3399570	38.2	33.58
Uk	2022	1303836	2935017	nan	39.77
Uk	2023	695856	1407029	-46.63	25.2
Uk	2024	1380015	3439878	98.32	31.11

Monthly/Quarterly Trends (2022–2024)

Year	Quarter	Month	Total Revenue (£000)	YoY Growth (%)	Customer Volume
2022	1	February	259053	nan	512293
2022	1	January	254658	nan	515822
2022	1	March	248798	nan	499476
2022	2	April	281028	nan	591421
2022	2	June	275900	nan	658849
2022	2	May	271994	nan	623741
2022	3	August	286155	nan	671109
2022	3	July	273458	nan	663307
2022	3	September	296410	nan	700085
2022	4	December	265645	nan	656248
2022	4	November	268307	nan	643989
2022	4	October	297142	nan	674083
2023	1	February	262228	1.23	598478
2023	1	January	263693	3.55	612598
2023	1	March	226189	-9.09	582506
2023	2	April	239764	-14.68	586963
2023	2	June	229899	-16.67	565601
2023	2	May	220230	-19.03	537554
2023	3	August	227604	-20.46	559657
2023	3	July	232732	-14.89	557431
2023	3	September	208999	-29.49	473285
2023	4	December	234172	-11.85	514893
2023	4	November	201905	-24.75	488888
2023	4	October	214419	-27.84	473097
2024	1	February	273947	4.47	666278
2024	1	January	246820	-6.4	582878
2024	1	March	304956	34.82	739092
2024	2	April	334256	39.41	786829
2024	2	June	350762	52.57	905154
2024	2	May	360138	63.53	858158

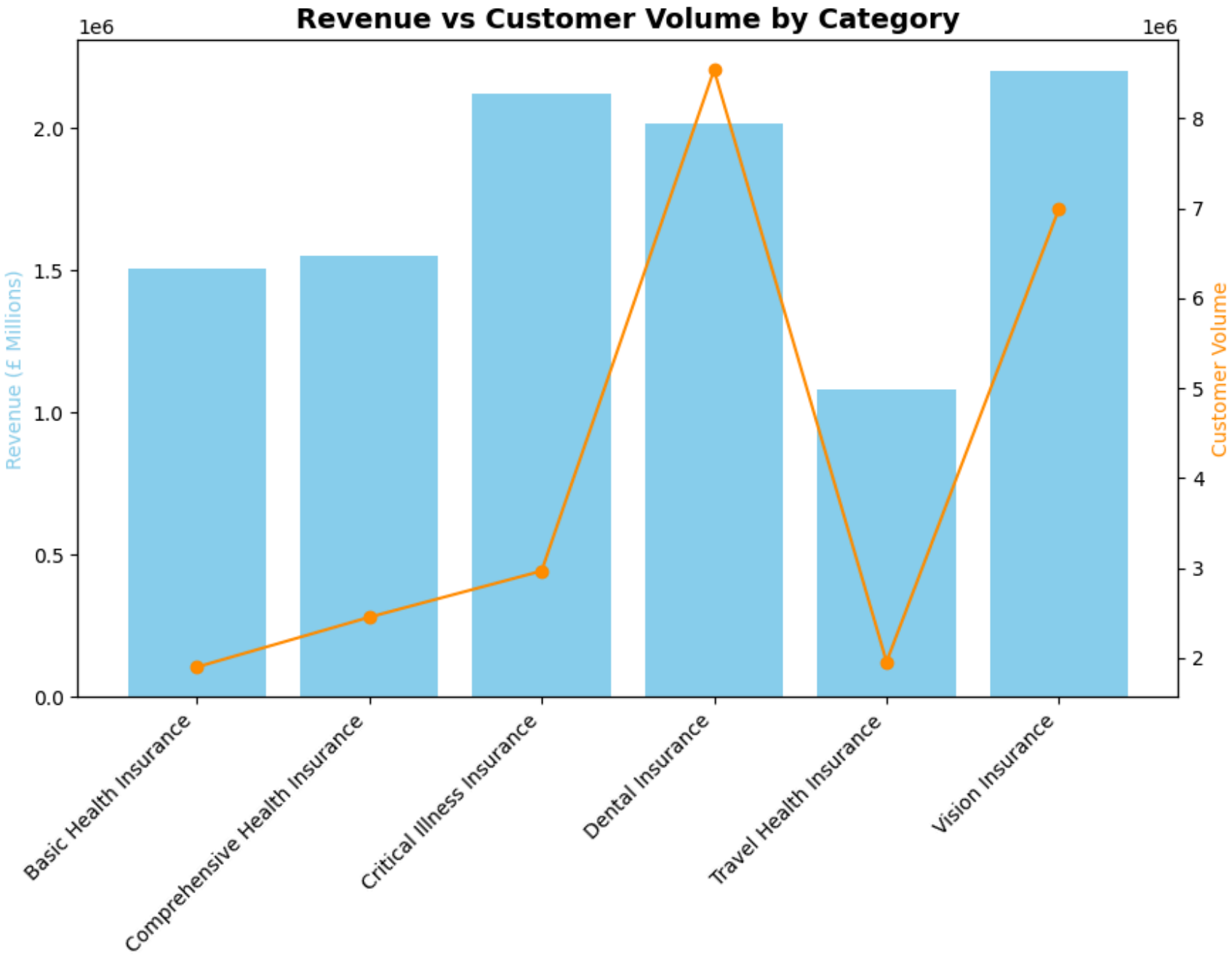
2024	3	August	407849	79.19	982796
2024	3	July	364778	56.74	888250
2024	3	September	414881	98.51	1032392
2024	4	December	494918	111.35	1232444
2024	4	November	455363	125.53	1096289
2024	4	October	427187	99.23	1077715


Monthly Revenue Trends by Country (2022–2024)
Country
India
Mexico
Uk
Revenue (£000)
180000
160000
140000
120000
100000
80000
60000
40000
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
Date

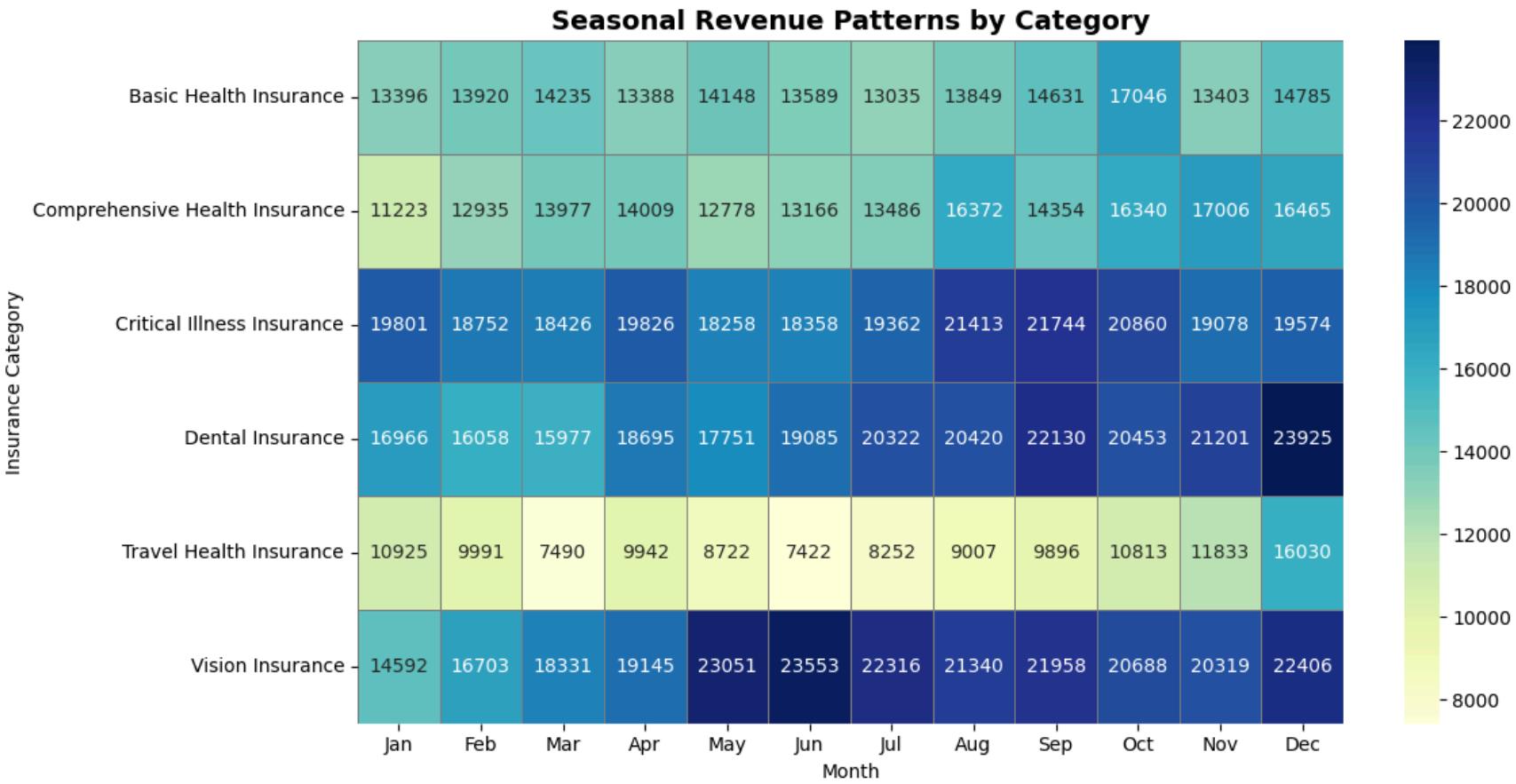



```
C:\Users\shahz\AppData\Local\Temp\ipykernel_34836\3150380487.py:204: UserWarnin
g: set_ticklabels() should only be used with a fixed number of ticks, i.e. afte
r set_ticks() or using a FixedLocator.
  ax1.set_xticklabels(category_data['health_insurance_category'], rotation=45,
ha='right')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 207
    204 ax1.set_xticklabels(category_data['health_insurance_category'], rotatio
n=45, ha='right')
    206 # Combine legends
--> 207 lines_labels = bars + line
    208 labels = [bar.get_label() for bar in lines_labels]
    209 plt.legend(lines_labels, labels, loc='upper right')

TypeError: can only concatenate tuple (not "list") to tuple
```


Revenue vs Customer Volume by Category
1e6
1e6
Revenue (£ Millions)
Customer Volume
0.0
0.5
1.0
1.5
2.0
2
3
4
5
6
7
8
Basic Health Insurance
Comprehensive Health Insurance
Critical Illness Insurance
Dental Insurance
Travel Health Insurance
Vision Insurance

In [18]:
```
import seaborn as sns

# Group: average monthly revenue per category
seasonal = df.groupby(['health_insurance_category', 'month'])['revenue_(£000)'

# Convert month numbers to names
seasonal.columns = [calendar.month_abbr[int(m)] for m in seasonal.columns]

# Plot heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(seasonal, cmap='YlGnBu', annot=True, fmt=".0f", linewidths=0.5, li
plt.title('Seasonal Revenue Patterns by Category', fontsize=14, fontweight='bo
plt.xlabel('Month')
plt.ylabel('Insurance Category')
plt.tight_layout()
plt.show()
```


Seasonal Revenue Patterns by Category
Insurance Category
Basic Health Insurance
Comprehensive Health Insurance
Critical Illness Insurance
Dental Insurance
Travel Health Insurance
Vision Insurance
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Month
13396 13920 14235 13388 14148 13589 13035 13849 14631 17046 13403 14785
11223 12935 13977 14009 12778 13166 13486 16372 14354 16340 17006 16465
19801 18752 18426 19826 18258 18358 19362 21413 21744 20860 19078 19574
16966 16058 15977 18695 17751 19085 20322 20420 22130 20453 21201 23925
10925 9991 7490 9942 8722 7422 8252 9007 9896 10813 11833 16030
14592 16703 18331 19145 23051 23553 22316 21340 21958 20688 20319 22406
22000
20000
18000
16000
14000
12000
10000
8000

In [ ]: