02_visitor_entries_cleaning_eda

December 27, 2025

1 Visitor Entries Cleaning and EDA

Data Source: Stats Canada Table 24-10-0050-01 (manually extracted)
Location: `data/interim/nonresident_visitors_mb_manual.csv`
Purpose: Clean, validate, and perform EDA on visitor entries data
Date: December 2025

1.1 Objectives

1. Clean and validate visitor entries data
2. Validate against Travel Manitoba Q4 2024 & Q1 2024 infographics
3. Explore trends by country of residence
4. Prepare dataset for Power BI dashboard

1.2 Setup

[14]:

```
# Path setup
import sys
from pathlib import Path
project_root = Path.cwd().parent
sys.path.insert(0, str(project_root / 'scripts'))
from paths import raw, processed, interim
```

[15]:

```
# Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

# Plotting style
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette('husl')
%matplotlib inline

# Display options
pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', '{:,.0f}'.format)
```

```
print('  Libraries loaded')
```

```
 Libraries loaded
```

1.3 Part 1: Data Loading & Cleaning

1.3.1 1.1 Load Raw Data

[16]:
```
csv_path = interim() / 'nonresident_visitors_mb_manual.csv'

if not csv_path.exists():
    print(f'ERROR: File not found at {csv_path}')
else:
    print(f'  Found: {csv_path}')
    print(f'   Size: {csv_path.stat().st_size:,} bytes')
```

```
 Found: /Users/dpro/projects/travel_manitoba/data/interim/nonresident_visitors_
mb_manual.csv
  Size: 9,405 bytes
```

[17]:
```
# Load CSV
df_raw = pd.read_csv(csv_path, encoding='utf-8-sig')

print('RAW DATA')
print('='*80)
print(f'Shape: {df_raw.shape}')
print(f'\nFirst 8 rows:')
df_raw.head(8)
```

```
RAW DATA
================================================================================
Shape: (38, 71)

First 8 rows:
```

[17]:
```
                             Country of residence 2  Jan-20  Feb-20  Mar-20  \
0              Non-resident visitors entering Canada  19,908  21,982  14,635
1  United States of America residents entering Ca…  18,743  20,845  13,790
2  Residents of countries other than the United S…   1,165   1,137     845
3  Americas, countries other than the United Stat…     385     389     367
4  North America, countries other than the United…     164     209     165
5                                    Central America      10      10      36
6                                          Caribbean      56      44      42
7                                      South America     155     126     124

  Apr-20 May-20 Jun-20 Jul-20 Aug-20 Sep-20 Oct-20 Nov-20 Dec-20 Jan-21  \
0  1,531  2,225  2,601  2,730  3,433  3,311  3,253  2,218  2,657  2,540
1  1,253  1,938  2,216  2,359  3,094  3,033  2,870  1,893  2,273  2,186
```

```
2    278    287    385    371    339    278    383    325    384    354
3    223    216    182    197    165    182    201    168    343    265
4    153     17     66     59     29     52     26     50     22     30
5     64    187     44      0      0      0      0      0      0      0
6      2      2     72     55      0      0     32      0     17     15
7      4     10      0     83    136    130    143    118    304    220

  Feb-21 Mar-21 Apr-21 May-21 Jun-21 Jul-21  Aug-21  Sep-21  Oct-21 Nov-21  \
0  2,040  2,673  2,553  3,037  3,110  3,839  10,505  11,189  10,632  6,734
1  1,677  2,165  2,015  2,435  2,901  3,739  10,229  11,031  10,438  6,527
2    363    508    538    602    209    100     276     158     194    207
3    363    135    516    128     48     14       8      86      84     65
4      0     16    516     38     22      9       0      32       6     21
5      0     84      0     31      8      2       0       0      10      0
6      0      4      0     12      5      0       0      18      32      0
7    363     31      0     47     13      3       8      36      36     44

  Dec-21 Jan-22 Feb-22 Mar-22 Apr-22  May-22  Jun-22  Jul-22  Aug-22  Sep-22  \
0  8,760  5,193  5,530  8,198  8,134  15,957  26,546  29,156  22,187  19,105
1  8,502  5,075  5,361  7,993  7,831  15,560  25,761  28,289  21,392  18,520
2    258    118    169    205    303     397     785     867     795     585
3    111     60     49     68    107      62     154     188     152      91
4     87     45     35     39     54      21      80      59      61      31
5      0     10      2      3     11       7       6       2       1       0
6      7      0      2      1     10      10      32      63      29      41
7     17      5     10     25     32      24      36      64      61      19

   Oct-22  Nov-22  Dec-22  Jan-23  Feb-23  Mar-23  Apr-23  May-23  Jun-23  \
0  22,370  14,064  17,604  17,078  15,158  21,180  16,826  26,447  44,738
1  21,776  13,562  16,802  16,468  14,575  20,386  15,970  25,604  43,527
2     594     502     802     610     583     794     856     843   1,211
3     186     245     400     408     343     343     447     122     177
4     125     212     289     329     307     283     303      51      71
5       9       1       1       4       0       8       7       2      11
6       9       5      61      46      26      27     106      20      22
7      43      27      49      29      10      25      31      49      73

   Jul-23  Aug-23  Sep-23  Oct-23  Nov-23  Dec-23  Jan-24  Feb-24  Mar-24  \
0  48,230  40,144  32,735  25,491  17,092  22,974  17,426  19,652  23,993
1  46,214  37,961  31,264  24,346  16,102  21,231  16,173  18,371  22,357
2   2,016   2,183   1,471   1,145     990   1,743   1,253   1,281   1,636
3     606     360     262     248     590   1,189   1,001   1,007   1,063
4      98      99      51     108     478     857     794     810     779
5       0      38      15      15       6      20       4       4      17
6      45      46      29      30      33     133     145     142     187
7     463     177     167      95      73     179      58      51      80
```

```
   Apr-24  May-24  Jun-24  Jul-24  Aug-24  Sep-24  Oct-24  Nov-24  Dec-24  \
0  19,248  32,078  48,618  47,400  38,349  28,857  31,101  23,320  25,525
1  17,617  30,648  46,793  45,067  36,356  27,165  29,421  21,861  23,615
2   1,631   1,430   1,825   2,333   1,993   1,692   1,680   1,459   1,910
3     996     469     621     744     519     560     536     844   1,201
4     819     175     239     230     143     261     257     678     839
5      22      95      77     128      73      87      63      35      37
6      44      38      20      55      88      12      25      51     183
7     111     161     285     331     215     200     191      80     142

   Jan-25  Feb-25  Mar-25  Apr-25  May-25  Jun-25  Jul-25  Aug-25  Sep-25  \
0  22,430  22,168  27,829  19,246  32,784  48,267  53,059  47,506  31,340
1  20,832  20,675  26,241  17,643  31,077  46,209  50,529  44,441  29,851
2   1,598   1,493   1,588   1,603   1,707   2,058   2,530   3,065   1,489
3   1,025     971   1,093     951     548     903   1,059   1,000     499
4     793     732     821     658     296     416     544     257     143
5      17      29      42      64      39      49      84     374     135
6     143     163     176     142      19      44      38      52      22
7      72      47      54      87     194     394     393     317     199

   Oct-25
0  32,564
1  31,168
2   1,396
3     464
4     135
5      79
6      35
7     215
```

1.3.2 1.2 Clean Numeric Columns

[18]:

```python
def clean_numeric_column(series):
    """Remove commas, quotes, convert to float."""
    return (
        series
        .astype(str)
        .str.replace(',', '', regex=False)
        .str.replace('"', '', regex=False)
        .str.strip()
        .replace('', np.nan)
        .replace('nan', np.nan)
        .astype('float')
    )

# Clean data
df_cleaned = df_raw.copy()
```

```python
# Get month columns (all except first)
month_cols = df_cleaned.columns[1:].tolist()

# Rename first column
df_cleaned.columns = ['country'] + month_cols

# Clean all numeric columns
for col in month_cols:
    df_cleaned[col] = clean_numeric_column(df_cleaned[col])

print('  Cleaned numeric columns')
print(f'\nData types:')
print(df_cleaned.dtypes)
```

```
 Cleaned numeric columns

Data types:
country     object
Jan-20     float64
Feb-20     float64
Mar-20     float64
Apr-20     float64
            …
Jun-25     float64
Jul-25     float64
Aug-25     float64
Sep-25     float64
Oct-25     float64
Length: 71, dtype: object
```

1.3.3 1.3 Data Quality Checks

[19]:

```python
print('DATA QUALITY SUMMARY')
print('='*80)
print(f'Total rows: {len(df_cleaned)}')
print(f'Total columns: {len(df_cleaned.columns)}')
print(f'Date range: {month_cols[0]} to {month_cols[-1]}')

print(f'\nNull values per column:')
null_summary = df_cleaned.isnull().sum()
if null_summary.sum() > 0:
    print(null_summary[null_summary > 0])
else:
    print('None - dataset is complete!')

print(f'\nFirst 10 countries/categories:')
print(df_cleaned['country'].head(10).tolist())
```

```
DATA QUALITY SUMMARY
================================================================================
Total rows: 38
Total columns: 71
Date range: Jan-20 to Oct-25

Null values per column:
None - dataset is complete!

First 10 countries/categories:
['Non-resident visitors entering Canada', 'United States of America residents
entering Canada', 'Residents of countries other than the United States of
America entering Canada', 'Americas, countries other than the United States of
America', 'North America, countries other than the United States of America',
'Central America', 'Caribbean', 'South America', 'Americas, n.o.s. 3', 'Europe']
```

1.3.4 1.4 Validate Against Infographics

[20]:

```
print('VALIDATION AGAINST TRAVEL MANITOBA INFOGRAPHICS')
print('='*80)

# Get US visitors row
us_visitors = df_cleaned[df_cleaned['country'] ==
                         'United States of America residents entering Canada'].
  ↪copy()

if not us_visitors.empty:
    us_data = us_visitors.iloc[0]

    # Q4 2024 Validation
    q4_2024 = us_data[['Oct-24', 'Nov-24', 'Dec-24']].sum()
    expected_q4 = 74_897

    print('\nQ4 2024 (U.S. Visitors into Manitoba)')
    print('-'*60)
    print(f'Calculated: {q4_2024:>12,.0f}')
    print(f'Expected:   {expected_q4:>12,}')
    print(f'Difference: {abs(q4_2024 - expected_q4):>12,.0f}')

    if abs(q4_2024 - expected_q4) < 10:
        print('  VALIDATION PASSED')
    else:
        print('  VALIDATION FAILED')

    # Q1 2024 Validation
    q1_2024 = us_data[['Jan-24', 'Feb-24', 'Mar-24']].sum()
    expected_q1 = 56_901

    print('\nQ1 2024 (U.S. Visitors into Manitoba)')
    print('-'*60)
    print(f'Calculated: {q1_2024:>12,.0f}')
    print(f'Expected:   {expected_q1:>12,}')
    print(f'Difference: {abs(q1_2024 - expected_q1):>12,.0f}')

    if abs(q1_2024 - expected_q1) < 10:
        print('  VALIDATION PASSED')
    else:
        print('  VALIDATION FAILED')
```

```
VALIDATION AGAINST TRAVEL MANITOBA INFOGRAPHICS
================================================================================

Q4 2024 (U.S. Visitors into Manitoba)
------------------------------------------------------------
Calculated:       74,897
Expected:         74,897
Difference:            0
  VALIDATION PASSED

Q1 2024 (U.S. Visitors into Manitoba)
------------------------------------------------------------
Calculated:       56,901
Expected:         56,901
Difference:            0
  VALIDATION PASSED
```

1.4 Part 2: Exploratory Data Analysis

1.4.1 2.1 Reshape Data for Time Series

[21]:

```
# Melt to long format
df_long = df_cleaned.melt(
    id_vars=['country'],
    value_vars=month_cols,
    var_name='month',
    value_name='visitors'
)

# Parse dates from 'Jan-20' format
def parse_month_year(month_str):
    parts = month_str.split('-')
    month_abbr = parts[0]
    year = '20' + parts[1]
    return pd.to_datetime(f"{month_abbr}-{year}", format='%b-%Y')

df_long['date'] = df_long['month'].apply(parse_month_year)
df_long = df_long.sort_values('date')

print(f'Long format shape: {df_long.shape}')
print(f'Date range: {df_long["date"].min()} to {df_long["date"].max()}')
df_long.head()
```

```
Long format shape: (2660, 4)
Date range: 2020-01-01 00:00:00 to 2025-10-01 00:00:00
```

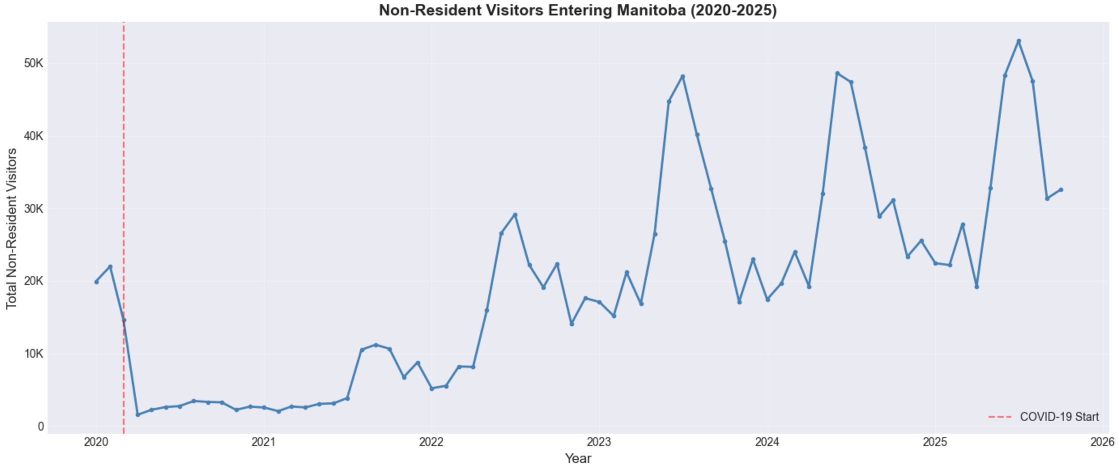
[21]:

```
                                  country   month  visitors       date
0   Non-resident visitors entering Canada  Jan-20    19,908 2020-01-01
21                        Southern Africa  Jan-20         9 2020-01-01
22                       Africa, n.o.s. 5  Jan-20         0 2020-01-01
23                                   Asia  Jan-20       235 2020-01-01
24                            Middle East  Jan-20         6 2020-01-01
```

1.4.2 2.2 Total Non-Resident Visitors Over Time

[22]:

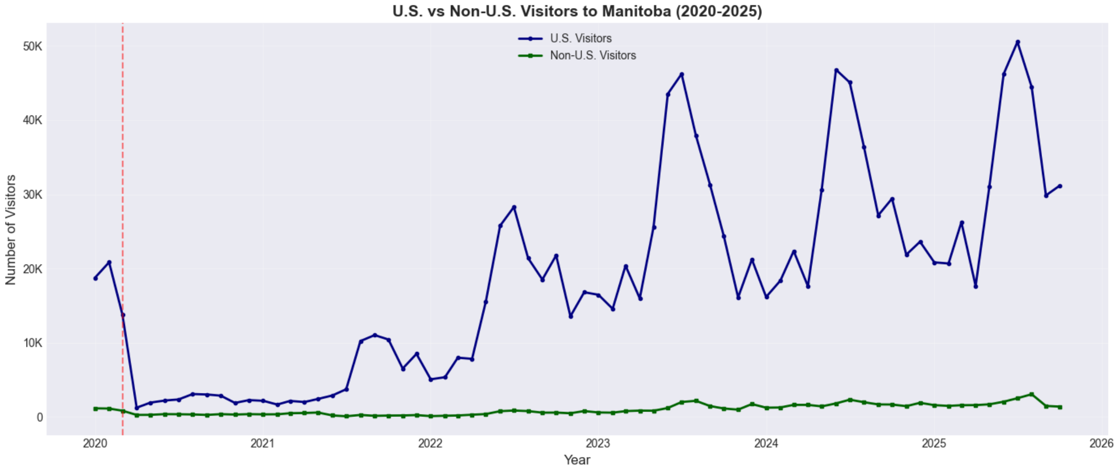
```
fig, ax = plt.subplots(figsize=(14, 6))

total_visitors = df_long[df_long['country'] == 'Non-resident visitors entering␣
 ↪Canada'].copy()

ax.plot(total_visitors['date'], total_visitors['visitors'],
        linewidth=2, marker='o', markersize=3, color='steelblue')

ax.axvline(pd.Timestamp('2020-03-01'), color='red', linestyle='--',
           alpha=0.5, label='COVID-19 Start')

ax.set_xlabel('Year', fontsize=12)
ax.set_ylabel('Total Non-Resident Visitors', fontsize=12)
ax.set_title('Non-Resident Visitors Entering Manitoba (2020-2025)',
             fontsize=14, fontweight='bold')
ax.legend()
ax.grid(True, alpha=0.3)
ax.yaxis.set_major_formatter(plt.FuncFormatter(
    lambda x, p: f'{x/1e3:.0f}K' if x >= 1000 else f'{x:.0f}'
))

plt.tight_layout()
plt.show()

print('Key Observations:')
print('- COVID-19 impact visible in 2020-2021')
print('- Recovery trend from 2022 onwards')
print('- Clear seasonal patterns (summer peaks)')
```



```
Key Observations:
- COVID-19 impact visible in 2020-2021
- Recovery trend from 2022 onwards
- Clear seasonal patterns (summer peaks)
```

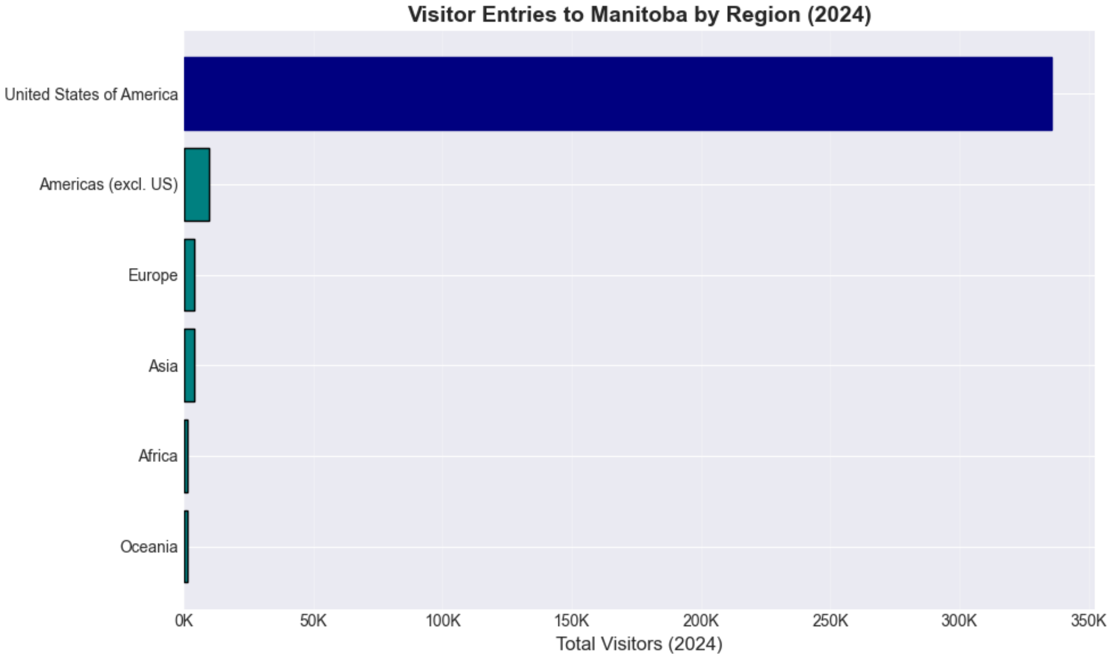
1.4.3 2.3 U.S. vs Non-U.S. Visitors

[23]:
```
fig, ax = plt.subplots(figsize=(14, 6))

us_data = df_long[df_long['country'] ==
                  'United States of America residents entering Canada'].copy()
other_data = df_long[df_long['country'] ==
                     'Residents of countries other than the United States of␣
 ↪America entering Canada'].copy()

ax.plot(us_data['date'], us_data['visitors'],
        linewidth=2, marker='o', markersize=3, label='U.S. Visitors',␣
 ↪color='navy')
ax.plot(other_data['date'], other_data['visitors'],
        linewidth=2, marker='s', markersize=3, label='Non-U.S. Visitors',␣
 ↪color='darkgreen')

ax.axvline(pd.Timestamp('2020-03-01'), color='red', linestyle='--', alpha=0.5)

ax.set_xlabel('Year', fontsize=12)
ax.set_ylabel('Number of Visitors', fontsize=12)
ax.set_title('U.S. vs Non-U.S. Visitors to Manitoba (2020-2025)',
             fontsize=14, fontweight='bold')
ax.legend()
ax.grid(True, alpha=0.3)
ax.yaxis.set_major_formatter(plt.FuncFormatter(
    lambda x, p: f'{x/1e3:.0f}K' if x >= 1000 else f'{x:.0f}'
))

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



1.4.4 2.4 Regional Breakdown (2024)

[24]:
```
# Get major regions for 2024
regions = ['United States of America residents entering Canada',
           'Europe', 'Asia',
           'Americas, countries other than the United States of America',
           'Africa', 'Oceania']

regions_2024 = df_cleaned[df_cleaned['country'].isin(regions)].copy()

# Calculate 2024 totals
cols_2024 = [col for col in month_cols if col.endswith('-24')]
regions_2024['total_2024'] = regions_2024[cols_2024].sum(axis=1)
regions_2024 = regions_2024.sort_values('total_2024', ascending=True)

fig, ax = plt.subplots(figsize=(10, 6))

bars = ax.barh(range(len(regions_2024)), regions_2024['total_2024'],
               color='teal', edgecolor='black')

# Highlight US
us_idx = list(regions_2024['country']).index(
    'United States of America residents entering Canada')
bars[us_idx].set_color('navy')

ax.set_yticks(range(len(regions_2024)))
ax.set_yticklabels([c.replace(' residents entering Canada', '').replace(
    'Americas, countries other than the United States of America', 'Americas␣
  ↪(excl. US)')
    for c in regions_2024['country']])
ax.set_xlabel('Total Visitors (2024)', fontsize=12)
ax.set_title('Visitor Entries to Manitoba by Region (2024)',
             fontsize=14, fontweight='bold')
ax.grid(True, alpha=0.3, axis='x')
ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, p: f'{x/1e3:.0f}K'))

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



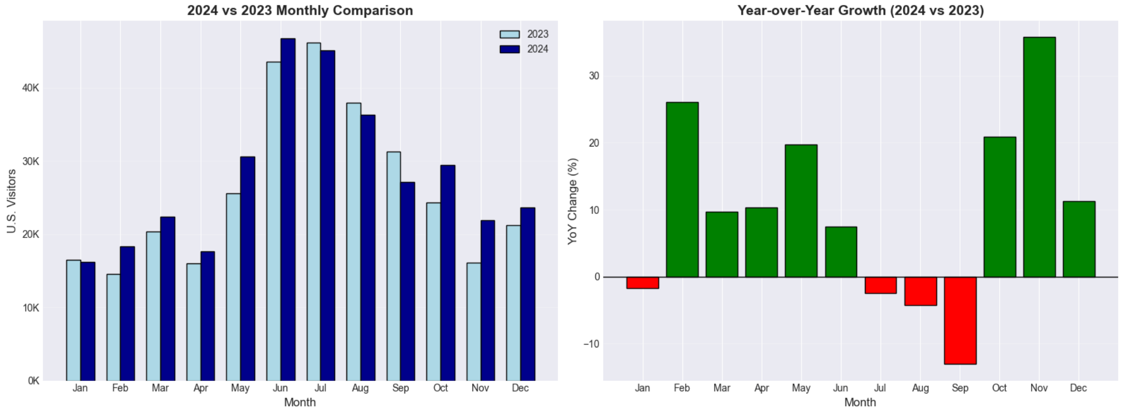
1.4.5 2.5 Year-over-Year Growth (2024 vs 2023)

[25]:

```
# Get US visitor data for 2023 and 2024
us_row = df_cleaned[df_cleaned['country'] ==
                    'United States of America residents entering Canada'].
  ↪iloc[0]

months_2023 = [col for col in month_cols if col.endswith('-23')]
```

```python
months_2024 = [col for col in month_cols if col.endswith('-24')]

yoy_data = []
for m23, m24 in zip(months_2023, months_2024):
    val_2023 = us_row[m23]
    val_2024 = us_row[m24]
    if pd.notna(val_2023) and pd.notna(val_2024) and val_2023 > 0:
        yoy_pct = ((val_2024 - val_2023) / val_2023) * 100
        yoy_data.append({
            'month': m24.split('-')[0],
            'yoy_pct': yoy_pct,
            'val_2023': val_2023,
            'val_2024': val_2024
        })

df_yoy = pd.DataFrame(yoy_data)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Panel 1: Absolute values
x = np.arange(len(df_yoy))
width = 0.35

ax1.bar(x - width/2, df_yoy['val_2023'], width, label='2023',
        color='lightblue', edgecolor='black')
ax1.bar(x + width/2, df_yoy['val_2024'], width, label='2024',
        color='darkblue', edgecolor='black')

ax1.set_xlabel('Month', fontsize=12)
ax1.set_ylabel('U.S. Visitors', fontsize=12)
ax1.set_title('2024 vs 2023 Monthly Comparison', fontsize=14, fontweight='bold')
ax1.set_xticks(x)
ax1.set_xticklabels(df_yoy['month'])
ax1.legend()
ax1.grid(True, alpha=0.3, axis='y')
ax1.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, p: f'{x/1e3:.0f}K'))

# Panel 2: YoY % change
colors = ['green' if x > 0 else 'red' for x in df_yoy['yoy_pct']]
ax2.bar(range(len(df_yoy)), df_yoy['yoy_pct'], color=colors, edgecolor='black')
ax2.axhline(0, color='black', linewidth=1)

ax2.set_xlabel('Month', fontsize=12)
ax2.set_ylabel('YoY Change (%)', fontsize=12)
ax2.set_title('Year-over-Year Growth (2024 vs 2023)', fontsize=14,␣
 ↪fontweight='bold')
ax2.set_xticks(range(len(df_yoy)))
```

```
ax2.set_xticklabels(df_yoy['month'])
ax2.grid(True, alpha=0.3, axis='y')

plt.tight_layout()
plt.show()

print('YoY Growth Summary (2024 vs 2023):')
print(f'Average growth: {df_yoy["yoy_pct"].mean():.1f}%')
print(f'Best month: {df_yoy.iloc[df_yoy["yoy_pct"].idxmax()]["month"]}␣
 ↪(+{df_yoy["yoy_pct"].max():.1f}%)')
print(f'Worst month: {df_yoy.iloc[df_yoy["yoy_pct"].idxmin()]["month"]}␣
 ↪({df_yoy["yoy_pct"].min():.1f}%)')
```



```
YoY Growth Summary (2024 vs 2023):
Average growth: 10.0%
Best month: Nov (+35.8%)
Worst month: Sep (-13.1%)
```

1.5 Part 3: Save Processed Data

[26]:

```
# Save cleaned wide format
output_path_wide = processed() / 'mb_visitor_entries_clean.csv'
df_cleaned.to_csv(output_path_wide, index=False)

print('  SAVED PROCESSED DATA')
print('='*80)
print(f'Location: {output_path_wide}')
print(f'Size: {output_path_wide.stat().st_size:,} bytes')
print(f'Shape: {df_cleaned.shape}')
print(f'\nReady for Power BI import!')
```

```
  SAVED PROCESSED DATA
================================================================================
Location:
/Users/dpro/projects/travel_manitoba/data/processed/mb_visitor_entries_clean.csv
Size: 14,130 bytes
Shape: (38, 71)

Ready for Power BI import!
```

1.6 Summary

1.6.1 Data Cleaning

- Loaded manual CSV from interim directory
- Cleaned numeric formatting (removed commas/quotes)
- Validated Q4 2024 = 74,897 (exact match)
- Validated Q1 2024 = 56,901 (exact match)
- Saved to `data/processed/mb_visitor_entries_clean.csv`

1.6.2 Key Findings

1. **U.S. Dominance**: ~94% of all visitors
2. **Strong Recovery**: 2024 exceeds pre-COVID levels
3. **Seasonality**: Clear summer peaks (June-August)
4. **Q4 2024 Growth**: +21.4% vs Q4 2023

1.6.3 Next Steps

1. Import `mb_visitor_entries_clean.csv` into Power BI
2. Create measures for YoY%, YTD totals, quarterly aggregations
3. Build visualizations matching Travel Manitoba style