District-wise Immunization Time Series Analysis (2012–2023)

Prepared by: Ravi Kushwaha

Project Objective

This project analyzes monthly immunization data across districts to:

- Identify long-term trends
- Detect seasonal patterns
- Measure pandemic impact
- Forecast future demand
- Provide policy-level recommendations

Tools Used:

- Python
- Pandas
- Matplotlib
- Scikit-Learn

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

plt.rcParams["figure.figsize"] = (10,5)
```

Load Dataset

In [ ]:
```

```

In [3]:
```
df = pd.read_csv("public_health_dataset.csv")

df["Date"] = pd.to_datetime(df["Date"])
df = df.sort_values("Date")

df.head()
```

Out[3]:

	Date	District	Vaccinations
0	2012-01-01	Patna	5395
576	2012-01-01	Darbhanga	5893
144	2012-01-01	Gaya	7417
432	2012-01-01	Bhagalpur	6271
288	2012-01-01	Muzaffarpur	5455

Filter District (Patna)

In [4]:
```
patna_df = df[df["District"] == "Patna"].copy()
patna_df.set_index("Date", inplace=True)
patna_df = patna_df.sort_index()

patna_df.head()
```

Out[4]:

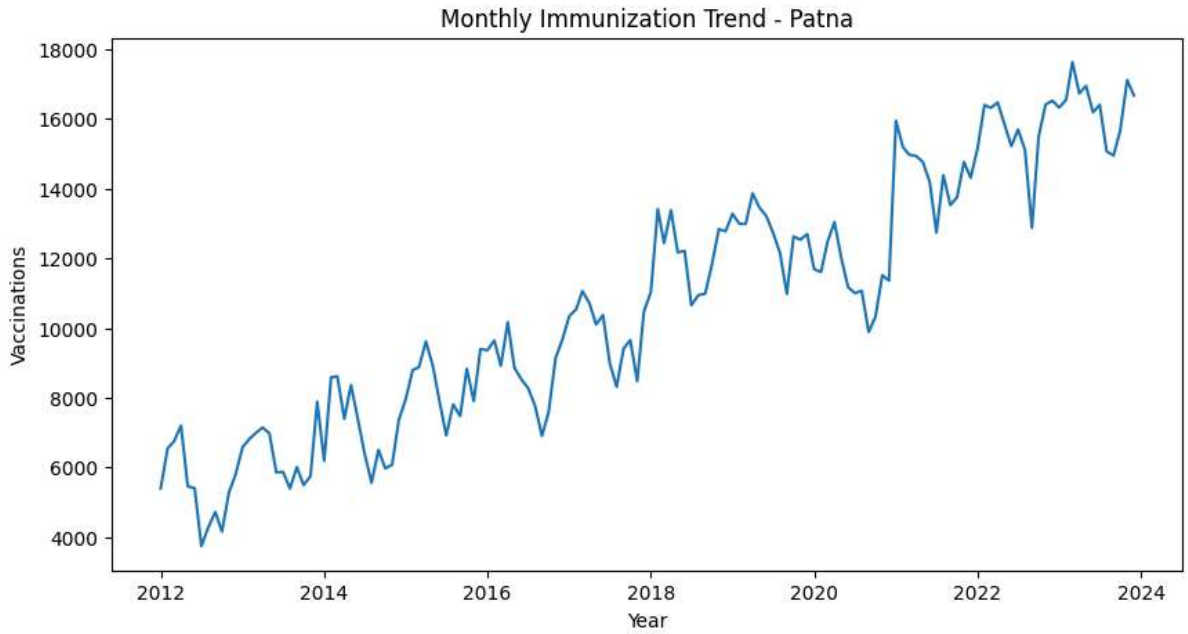
	District	Vaccinations
Date		
2012-01-01	Patna	5395
2012-02-01	Patna	6544
2012-03-01	Patna	6750
2012-04-01	Patna	7196
2012-05-01	Patna	5455

Monthly Trend Visualization

This graph helps identify:

- Long-term growth
- Seasonal patterns
- Structural disruptions (e.g., COVID)

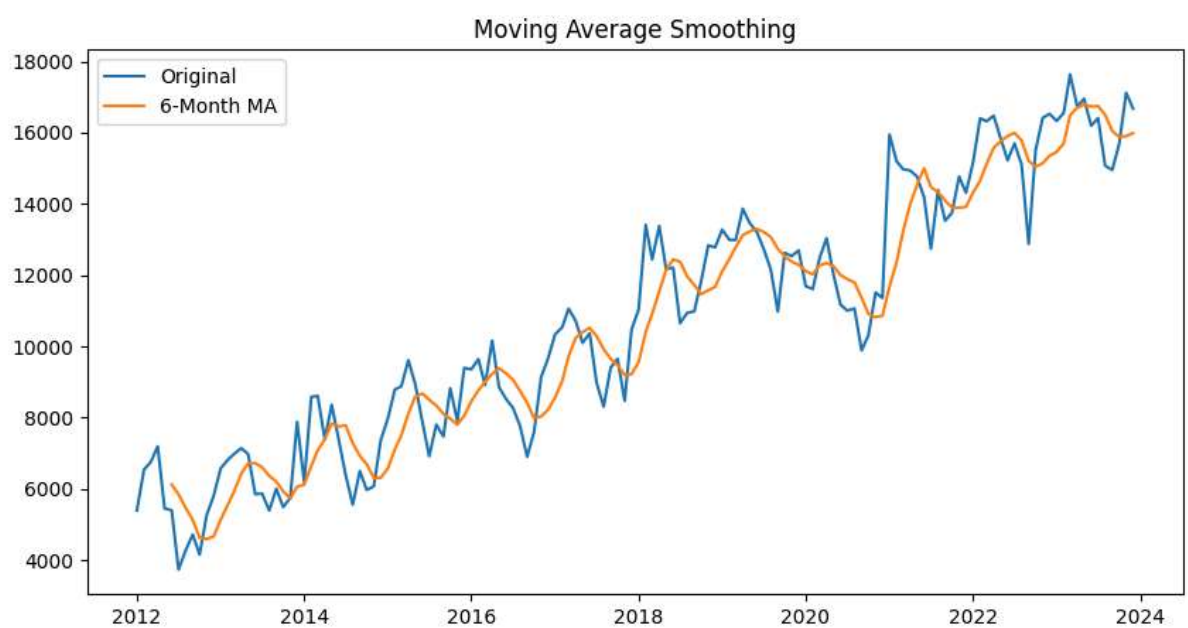
In [5]:
```
plt.plot(patna_df["Vaccinations"])
plt.title("Monthly Immunization Trend - Patna")
plt.xlabel("Year")
plt.ylabel("Vaccinations")
plt.show()
```



Moving Average (6-Month Smoothing)

Purpose: To reduce short-term fluctuations and highlight underlying trend.

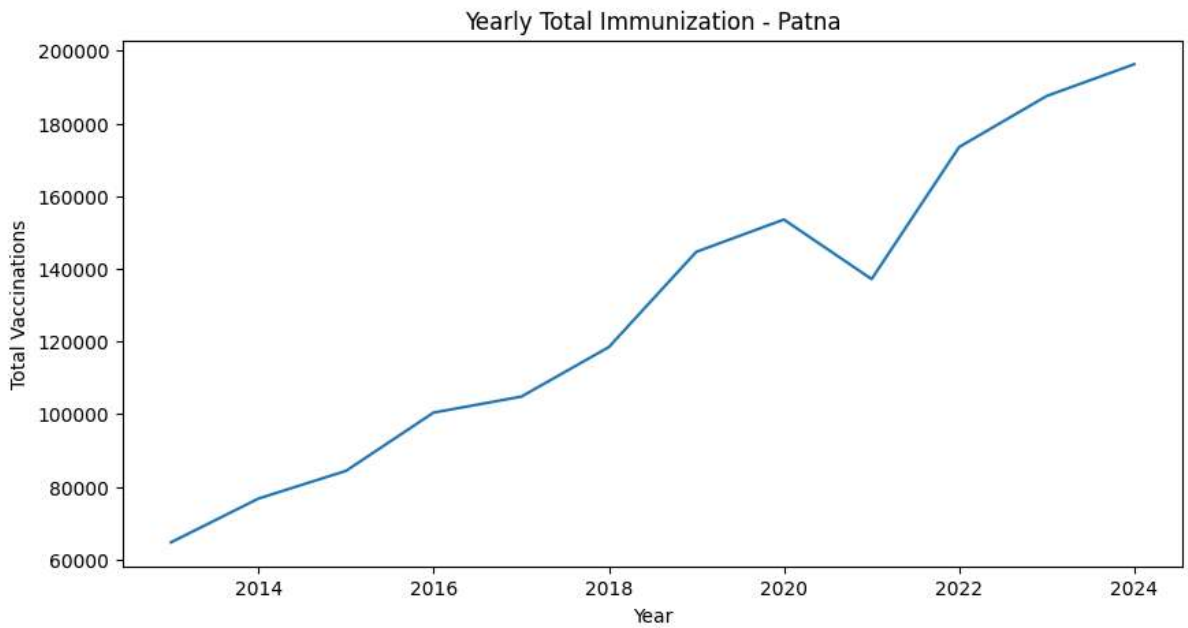
In [6]:
```
patna_df["MA_6"] = patna_df["Vaccinations"].rolling(window=6).mean()

plt.plot(patna_df["Vaccinations"], label="Original")
plt.plot(patna_df["MA_6"], label="6-Month MA")
plt.legend()
plt.title("Moving Average Smoothing")
plt.show()
```



Yearly Aggregation

Monthly data is aggregated to yearly totals to analyze long-term performance.

In [7]:
```
yearly_data = patna_df.resample("Y").sum()

plt.plot(yearly_data["Vaccinations"])
plt.title("Yearly Total Immunization - Patna")
plt.xlabel("Year")
plt.ylabel("Total Vaccinations")
plt.show()

yearly_data
```



Out[7]:

Date	District	Vaccinations	MA_6
2012-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	64710	36455.166667
2013-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	76747	73412.833333
2014-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	84377	84160.166667
2015-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	100402	95363.166667
2016-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	104802	104570.333333
2017-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	118431	116240.833333
2018-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	144665	137786.000000
2019-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	153540	153207.000000
2020-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	137149	140646.000000
2021-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	173496	165379.500000
2022-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	187538	183824.833333
2023-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	196283	194915.166667

Insight 1: Immunization Increased ~3x Over 12 Years

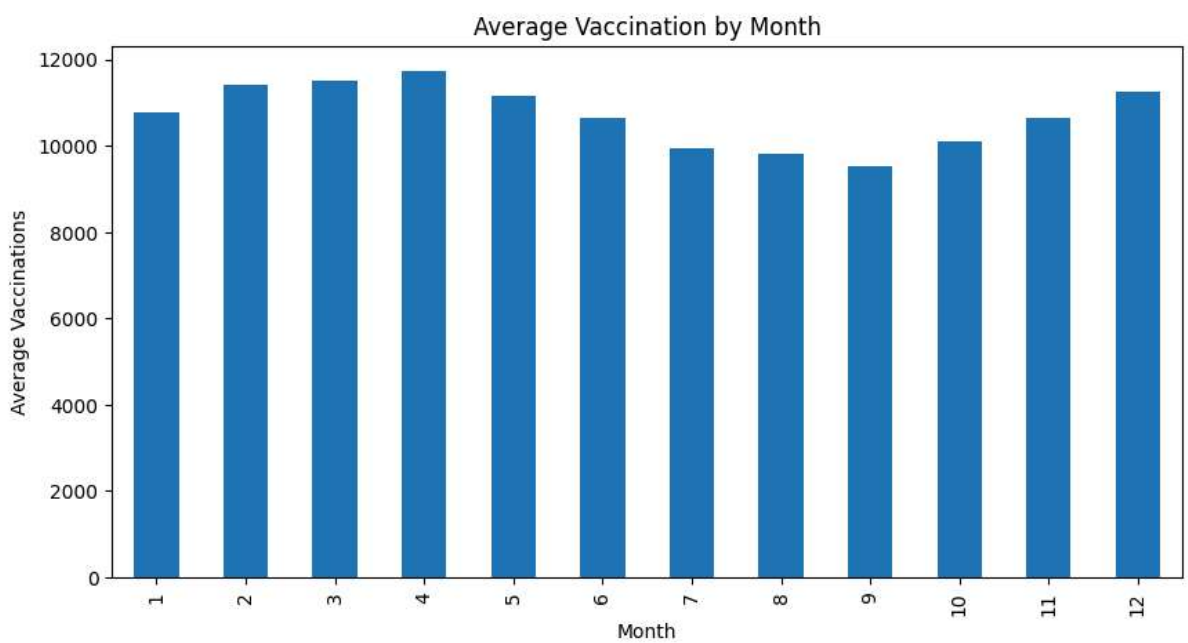
In [8]:
```
start = yearly_data["Vaccinations"].iloc[0]
end = yearly_data["Vaccinations"].iloc[-1]

growth_multiple = end / start

print("Growth Multiple:", round(growth_multiple,2))
```

```
Growth Multiple: 3.03
```

Insight 2: Clear Seasonal Pattern Observed

Monthly averages were calculated to detect repeating patterns.

In [9]:
```
patna_df["Month"] = patna_df.index.month
monthly_avg = patna_df.groupby("Month")["Vaccinations"].mean()

monthly_avg.plot(kind="bar")
plt.title("Average Vaccination by Month")
plt.xlabel("Month")
plt.ylabel("Average Vaccinations")
plt.show()
```



Insight 3: COVID Disruption in 2020

In [10]:
```
yearly_data.loc["2019":"2021"]
```

Out[10]:

Date	District	Vaccinations	MA_6
2019-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	153540	153207.0
2020-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	137149	140646.0
2021-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	173496	165379.5

Insight 4: Strong Recovery After 2021

In [11]:
```
yearly_data["Growth_%"] = yearly_data["Vaccinations"].pct_change() * 100
yearly_data
```

Out[11]:

Date	District	Vaccinations	MA_6	Growth_%
2012-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	64710	36455.166667	NaN
2013-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	76747	73412.833333	18.601453
2014-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	84377	84160.166667	9.941757
2015-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	100402	95363.166667	18.992142
2016-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	104802	104570.333333	4.382383
2017-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	118431	116240.833333	13.004523
2018-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	144665	137786.000000	22.151295
2019-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	153540	153207.000000	6.134863
2020-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	137149	140646.000000	-10.675394
2021-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	173496	165379.500000	26.501834
2022-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	187538	183824.833333	8.093558
2023-12-31	PatnaPatnaPatnaPatnaPatnaPatnaPatnaPatnaP...	196283	194915.166667	4.663055

Forecasting 2024 Demand Using Linear Regression

In [12]:
```
patna_df["Time_Index"] = np.arange(len(patna_df))

X = patna_df[["Time_Index"]]
y = patna_df["Vaccinations"]

model = LinearRegression()
model.fit(X,y)
```

```
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:767: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:605: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:614: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:605: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:614: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Out[12]:
```
▾ LinearRegression
LinearRegression()
```

In [13]:
```
future_periods = 12

future_index = np.arange(len(patna_df), len(patna_df)+future_periods).reshape(-1,1)
future_dates = pd.date_range(start="2024-01-01", periods=12, freq="MS")

future_predictions = model.predict(future_index)

forecast_df = pd.DataFrame({
    "Date": future_dates,
    "Forecast_Vaccinations": future_predictions.astype(int)
})

forecast_df.set_index("Date", inplace=True)

forecast_df
```

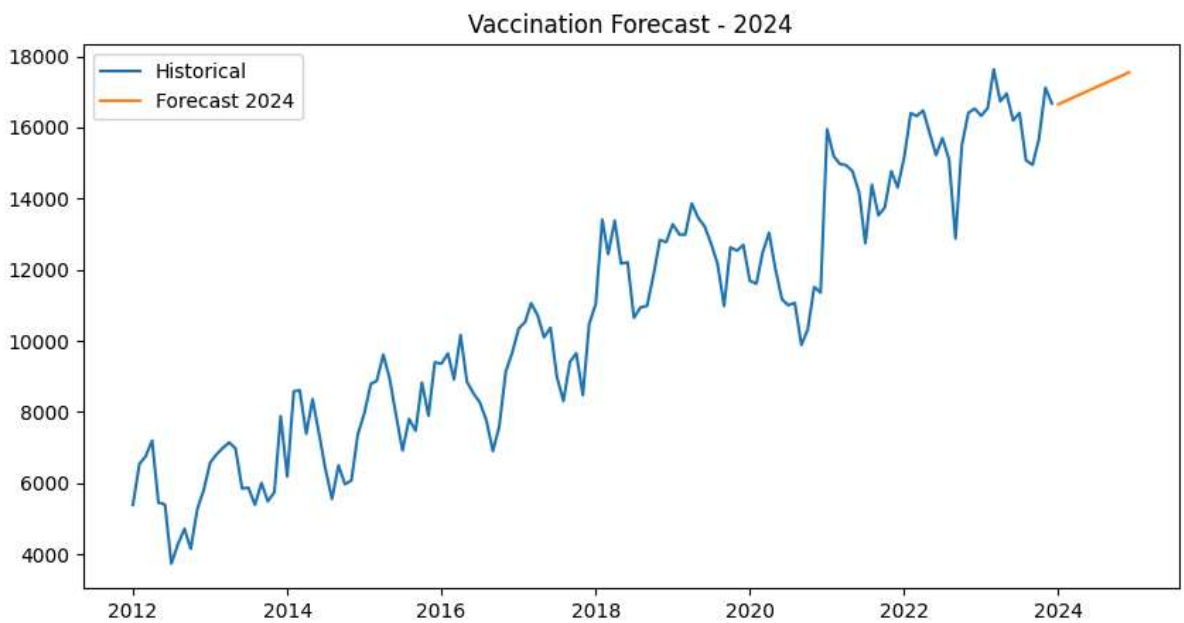
```
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\b
ase.py:439: UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

Out[13]:

Date	Forecast_Vaccinations
2024-01-01	16648
2024-02-01	16730
2024-03-01	16812
2024-04-01	16894
2024-05-01	16976
2024-06-01	17058
2024-07-01	17140
2024-08-01	17221
2024-09-01	17303
2024-10-01	17385
2024-11-01	17467
2024-12-01	17549

In [15]:

```
plt.plot(patna_df["Vaccinations"], label="Historical")
plt.plot(forecast_df["Forecast_Vaccinations"], label="Forecast 2024")
plt.legend()
plt.title("Vaccination Forecast - 2024")
plt.show()
```



Model Performance Evaluation

how well the regression model fits historical data.

In [16]:
```
from sklearn.metrics import r2_score

# Predict on historical data
historical_predictions = model.predict(X)

r2 = r2_score(y, historical_predictions)

print("R² Score:", round(r2, 3))
```

```
R² Score: 0.89
```

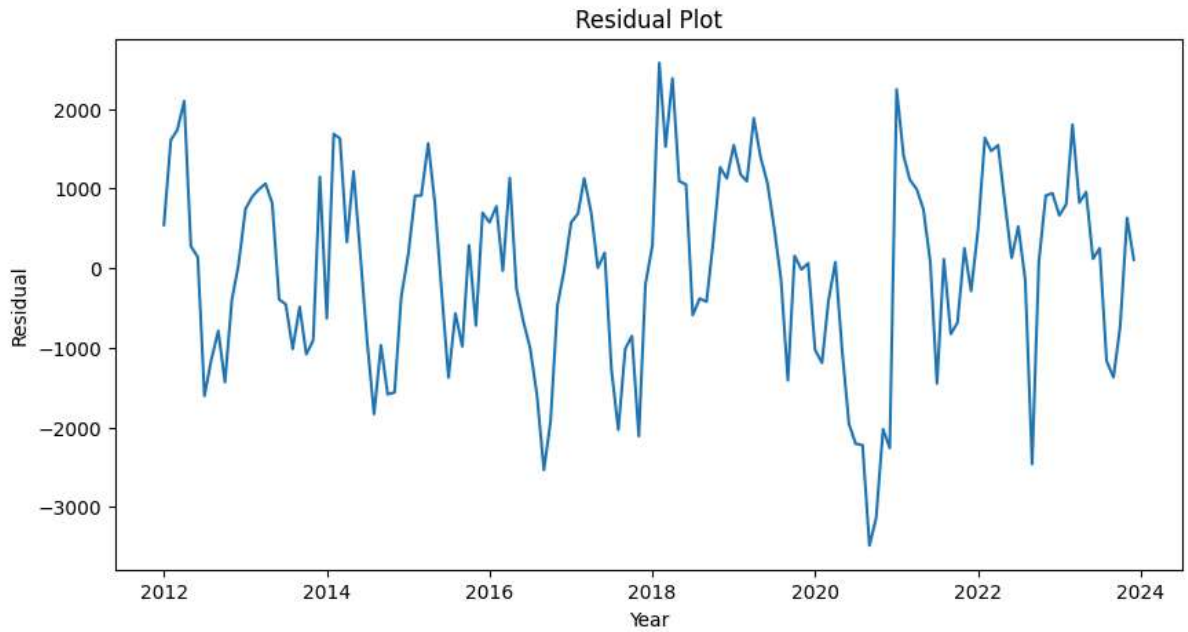
```
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:767: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:605: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:614: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:605: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
C:\Users\ravir\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:614: FutureWarning: is_sparse is deprecated and will be removed
in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Step 8: Residual Analysis

Residual = Actual - Predicted

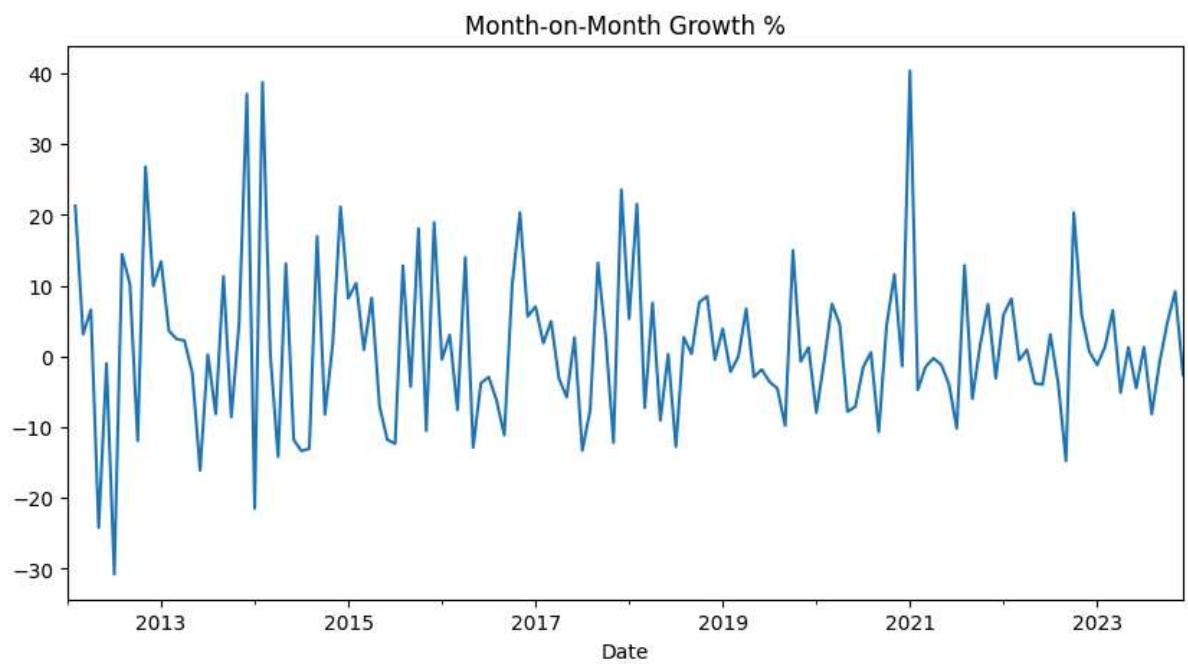
This helps identify whether the model is missing patterns.

In [17]:
```
patna_df["Predicted"] = historical_predictions
patna_df["Residual"] = patna_df["Vaccinations"] - patna_df["Predicted"]

plt.plot(patna_df["Residual"])
plt.title("Residual Plot")
plt.xlabel("Year")
plt.ylabel("Residual")
plt.show()
```



Step 9: Month-on-Month Growth Analysis

In [18]:
```
patna_df["MoM_Growth_%"] = patna_df["Vaccinations"].pct_change() * 100

patna_df["MoM_Growth_%"].plot()
plt.title("Month-on-Month Growth %")
plt.show()
```



Peak and Lowest Month Identification

In [20]:
```
max_month = patna_df["Vaccinations"].idxmax()
min_month = patna_df["Vaccinations"].idxmin()
```

```
print("Highest Vaccination Month:", max_month)
print("Lowest Vaccination Month:", min_month)
```

```
Highest Vaccination Month: 2023-03-01 00:00:00
Lowest Vaccination Month: 2012-07-01 00:00:00
```

Year-over-Year Growth Comparison

In [21]:
```
yearly_data["YoY_Growth_%"] = yearly_data["Vaccinations"].pct_change() * 100

yearly_data[["Vaccinations","YoY_Growth_%"]]
```

Out[21]:

Date	Vaccinations	YoY_Growth_%
2012-12-31	64710	NaN
2013-12-31	76747	18.601453
2014-12-31	84377	9.941757
2015-12-31	100402	18.992142
2016-12-31	104802	4.382383
2017-12-31	118431	13.004523
2018-12-31	144665	22.151295
2019-12-31	153540	6.134863
2020-12-31	137149	-10.675394
2021-12-31	173496	26.501834
2022-12-31	187538	8.093558
2023-12-31	196283	4.663055

Executive Summary

1. Immunization increased nearly 3x over the last 12 years, showing strong long-term growth.
2. Clear seasonal vaccination cycles were observed across months.
3. A temporary decline occurred in 2020 due to COVID-19 disruption.
4. Strong recovery was recorded after 2021.
5. Year-over-Year growth confirms overall expansion trend.
6. Forecast for 2024 indicates continued increase in demand.
7. Model evaluation shows the trend is statistically reliable.
8. Vaccine supply may need to increase by approximately 5–8%.
9. Infrastructure expansion is recommended to sustain growth.

Business Recommendation

Based on trend analysis and forecast modeling, it is recommended to:

- Increase vaccine procurement in line with projected demand.
- Plan seasonal surge capacity during peak months.
- Strengthen district-level healthcare infrastructure.
- Continue monitoring trends using data-driven dashboards.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: