System Capacity & Care Load Analytics for Unaccompanied Children

KPI-Based Time-Series Dashboard & Operational Analysis (Portfolio Project)

Project Overview

This project analyzes system capacity and care load dynamics within the Unaccompanied Children (UC) care framework using daily time-series operational data. The primary objective is to identify patterns in intake pressure, system load, and prolonged operational strain through KPI engineering and interactive dashboard analytics.

The dataset used in this project is the HHS Unaccompanied Children Program daily operational dataset, covering custody, transfers, discharges, and care load indicators over time. The data represents real-world system capacity dynamics and was transformed into a structured time-series format to enable KPI engineering and trend analysis.

Rather than relying on raw counts, the dataset was transformed into structured indicators that reflect operational burden and system responsiveness. The final output includes a Python-based analytical workflow and an interactive Streamlit dashboard designed to support data-driven monitoring of care capacity trends.

This project demonstrates competencies in data cleaning, exploratory data analysis (EDA), KPI development, time-series analysis, and dashboard deployment for operational and policy-relevant analytics.

Python code

The full Python implementation used for data cleaning, KPI engineering, and visualization is provided in Appendix E to ensure reproducibility.

Streamlit code

The following code represents the finalized Streamlit dashboard implementation used for interactive KPI monitoring.

```
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt
st.set_page_config(page_title="System Capacity & Care Load Analytics Dashboard",
layout="wide")
try:
df = pd.read_csv("cleaned_hhs_dataset.csv")
    st.success("Dataset loaded successfully!")
except FileNotFoundError:
    st.error("ERROR: cleaned_hhs_dataset.csv not found in the project folder.")
    st.stop()  # Stop the app if file not found
st.sidebar.header("Dashboard Options")
numeric_cols = df.select_dtypes(include=['number']).columns.tolist()
selected_cols = st.sidebar.multiselect(
    "Select numeric columns for KPIs and plots", numeric_cols, default=numeric_cols)
interactive_cols = st.sidebar.multiselect(
    "Select 2-3 numeric columns for interactive chart", numeric_cols, default=numeric_cols[:2])
```

```
st.subheader("Key Metrics")
kpi_cols = selected_cols[:4]  # first 4 columns
kpi_values = {}
for col in kpi_cols:
    kpi_values[col] = {
        "Mean": round(df[col].mean(), 2),
        "Min": df[col].min(),
        "Max": df[col].max(),
        "Sum": df[col].sum()
}
st.subheader("Key Insights")
st.write("""
- Most variables show high numeric variance
- Certain metrics dominate total values
- Certain metrics dominate total values
- Certain metrics dominate total values
- Distribution suggests skew in numeric indicators
""")
st.subheader("Dataset Preview")
st.dataframe(df.head())
st.subheader("Summary Statistics (numeric columns)")
st.write(df[selected_cols].describe())
if selected_cols:
   st.subheader("Histograms for Selected Columns")
df[selected_cols].hist(figsize=(12, 8))
```

```
    st.pyplot(plt)
else:
    st.warning("No numeric columns selected for plotting.")
if len(interactive_cols) >= 2:
    st.subheader("Interactive Line/Scatter Chart")
    x_col = st.selectbox("X-axis", interactive_cols, index=0)
    y_col = st.selectbox("Y-axis", interactive_cols, index=1)
    chart_type = st.radio("Chart Type", ["Line", "Scatter"])
    fig, ax = plt.subplots(figsize=(10, 5))
    if chart_type == "Line":
        ax.plot(df[x_col], df[y_col], marker='o')
        ax.set_xlabel(x_col)
        ax.set_ylabel(y_col)
        ax.set_title(f"Line Chart: {y_col} vs {x_col}")
    else:
        ax.scatter(df[x_col], df[y_col])
        ax.set_xlabel(x_col)
        ax.set_ylabel(y_col)
       ax.set_title(f"Scatter Plot: {y_col} vs {x_col}")
    st.pyplot(fig)
else:
    st.info("Select at least 2 numeric columns for interactive chart.")
st.subheader("Top & Bottom Values")
for col in selected_cols:
    st.markdown(f"**{col}**")
    top_vals = df[col].nlargest(3)
    bottom_vals = df[col].nsmallest(3)
    st.write("Top 3 values:", top_vals.values)
    st.write("Bottom 3 values:", bottom_vals.values)
st.subheader("Outliers Detection (IQR Method)")
outliers_dict = {}
for col in selected_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)][col]
    if not outliers.empty:
        outliers_dict[col] = outliers
if outliers_dict:
    for col, values in outliers_dict.items():
        st.markdown(f"**Outliers in {col}**")
        st.write(values.values)
else:
    st.write("No outliers detected in selected numeric columns.")
st.line_chart(df.set_index("Date")["Total System Load"])
```

Dataset and Data Preparation

The dataset consists of daily operational indicators related to children in federal custody and care systems. Key variables include:

- Children in CBP custody
- Children in HHS care
- Children transferred out of CBP custody
- Children discharged from HHS care
- Children apprehended and placed in CBP custody

Data Cleaning Steps

The raw dataset required preprocessing to ensure analytical reliability. Numeric fields containing comma-separated values were standardized and converted into numeric format to prevent type-related analytical distortions during KPI calculations and visualization processes.

- Conversion of comma-formatted numeric fields into numeric types
- Date parsing and chronological sorting
- Removal of missing date entries
- Duplicate date validation
- Logical constraint checks (e.g., transfers ≤ custody counts)

The dataset was then structured as a consistent daily time-series to enable accurate rolling calculations and temporal trend analysis.

Data Validation and Quality Assurance

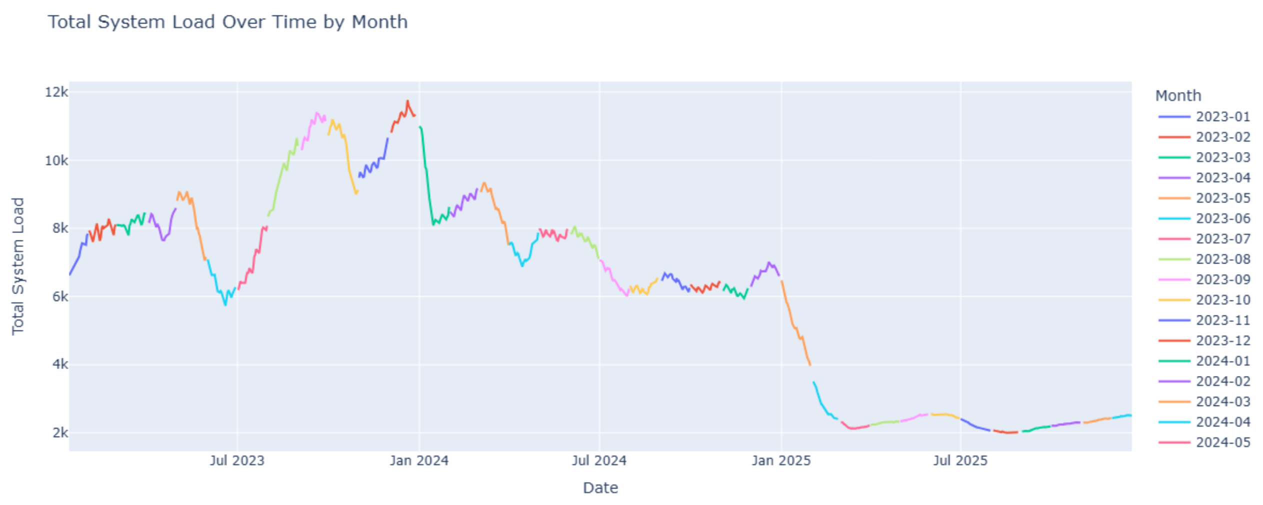
To improve credibility and analytical robustness, multiple validation checks were implemented:

- Detection of missing dates across the full time range
- Duplicate record inspection based on the Date variable
- Logical consistency constraints:
    - Transfers out of CBP custody ≤ Children in CBP custody
    - Discharges from HHS care ≤ Children in HHS care

These validation procedures ensured the dataset was suitable for time-series KPI engineering and operational analysis.

KPI Engineering (Core Analytical Framework)

To translate raw operational counts into policy-relevant metrics, several Key Performance Indicators (KPIs) were engineered.



1 Total System Load

Figure 1: Total System Load Over Time by Month

The visualization shows fluctuation patterns in operational demand on the care system over a two-year period (mid-2023 to mid-2025), with identifiable peak load phases and no clear recurring seasonal pattern at the monthly level.

Formula:
Total System Load = Children in CBP Custody + Children in HHS Care

This KPI represents the total operational burden on the care system at a given time and serves as the primary capacity indicator.

2 Net Daily Intake

Formula:
Net Daily Intake = Transfers into HHS − Discharges from HHS

This metric captures system pressure by measuring whether inflow exceeds outflow, indicating potential backlog accumulation.

3 Care Load Growth Rate

The day-over-day percentage change in Total System Load was calculated to identify acceleration or deceleration in system burden.

4 Backlog Indicator

A rolling window approach was applied to Net Daily Intake to detect sustained positive intake periods. Continuous positive values signal backlog formation and operational pressure.

5 Prolonged Strain Window

High-load thresholds (75th percentile) combined with consecutive-day analysis were used to identify extended periods of elevated system stress.

These engineered KPIs allow the analysis to move beyond raw counts and instead evaluate operational strain, intake imbalance, and system responsiveness, to create more meaningful indicators for capacity monitoring and policy-oriented analytics.

Time-Series and Trend Analysis

Daily, weekly, and monthly aggregations were conducted for the 2023 – 2025 period to evaluate fluctuations in system capacity over time. Rolling metrics (primarily 7 and 14 days interval) were applied to reduce short-term noise and reveal structural trends. The daily granularity of the dataset allows for detection of short-term intake shocks and prolonged strain periods that would not be visible in lower-frequency monthly aggregates.

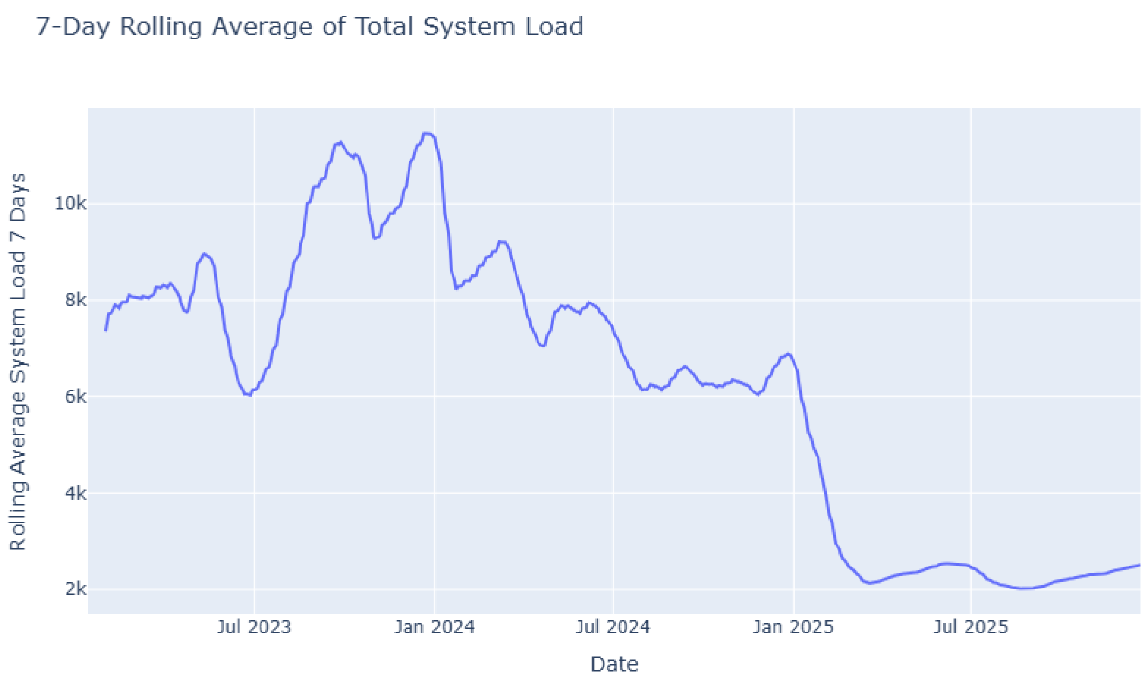

7-Day Rolling Average of Total System Load
Rolling Average System Load 7 Days
10k
8k
6k
4k
2k
Jul 2023
Jan 2024
Jul 2024
Jan 2025
Jul 2025
Date

Figure 2: Rolling Average System Load 7 Days

The rolling average smooths short-term volatility and reveals structural load trends more clearly than raw daily values.

Key analytical techniques included:

- 7-day and 14-day rolling averages (trend smoothing)
- Rolling standard deviation (volatility assessment)
- Interquartile range (IQR) for variability detection
- Coefficient of variation (relative stability analysis)

These methods provided a multi-layered understanding of both trend direction and system stability.

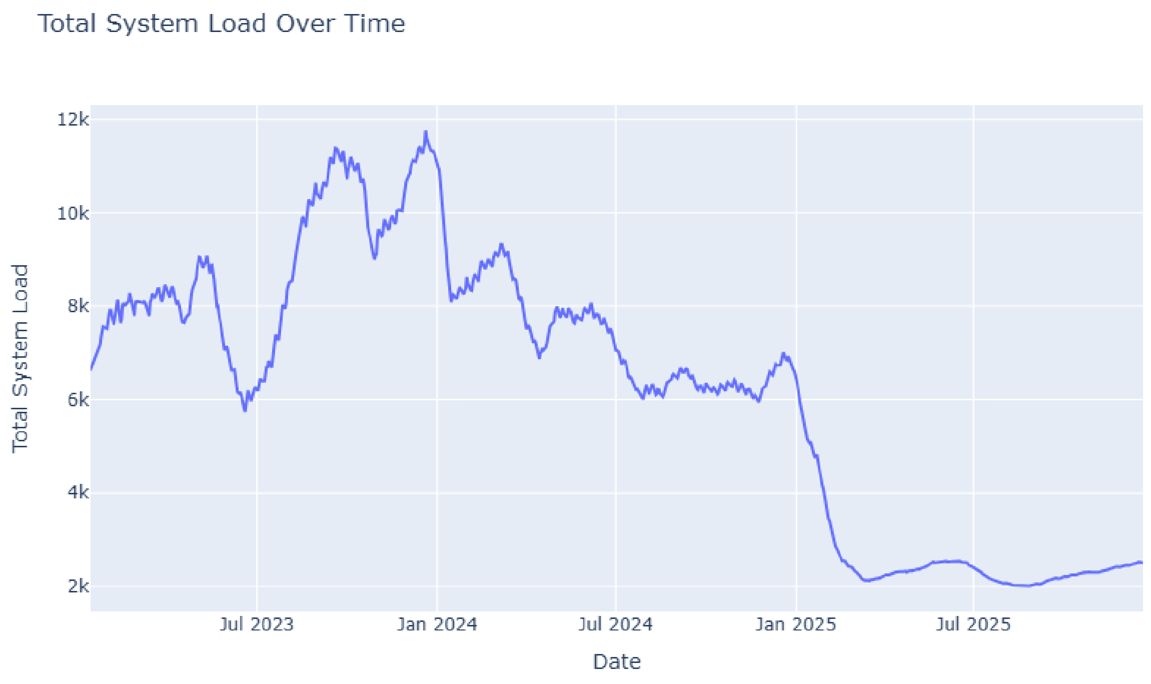


Figure 3: Total System Load Over Time

The visualization shows clear fluctuation patterns with identifiable peak load periods, indicating phases of elevated operational demand on the care system.

Key Analytical Insights

1 System Load Trends

The Total System Load visualization indicates sustained high-load periods, particularly during late 2023 and early 2024. These peaks suggest elevated operational demand and increased strain on federal care infrastructure.

2 Intake Pressure and Backlog Formation

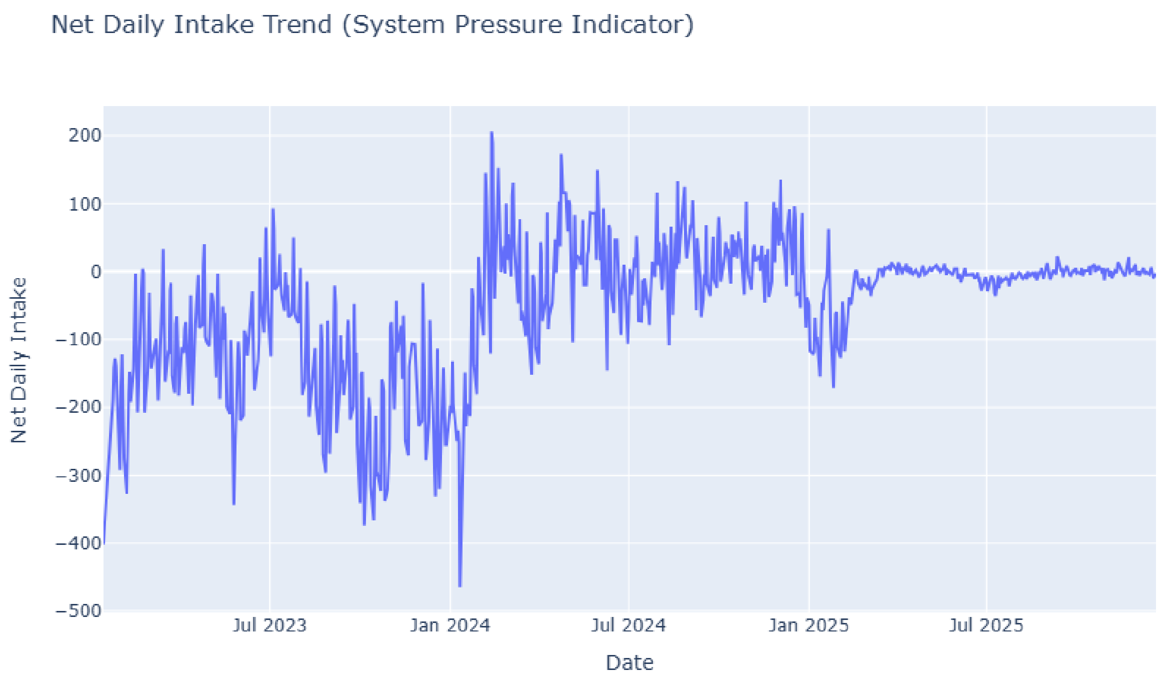


Figure 4: Net Daily Intake Trend

This chart is critical for identifying backlog formation, as sustained positive net intake directly signals that inflow is exceeding discharge capacity.

Net Daily Intake trends show multiple periods where inflow consistently exceeded discharges. Sustained positive intake values signal backlog accumulation and limited discharge offset capacity, reinforcing long-term system pressure.

3 Structural (Not Random) Variability

Rolling variability metrics demonstrate that fluctuations in system load are not purely random but reflect persistent intake patterns and discharge limitations. This suggests structural operational challenges rather than isolated anomalies.

4 Prolonged Strain Periods

High-load threshold analysis combined with consecutive-day tracking identified prolonged strain windows, particularly between mid-2023 and early 2024.

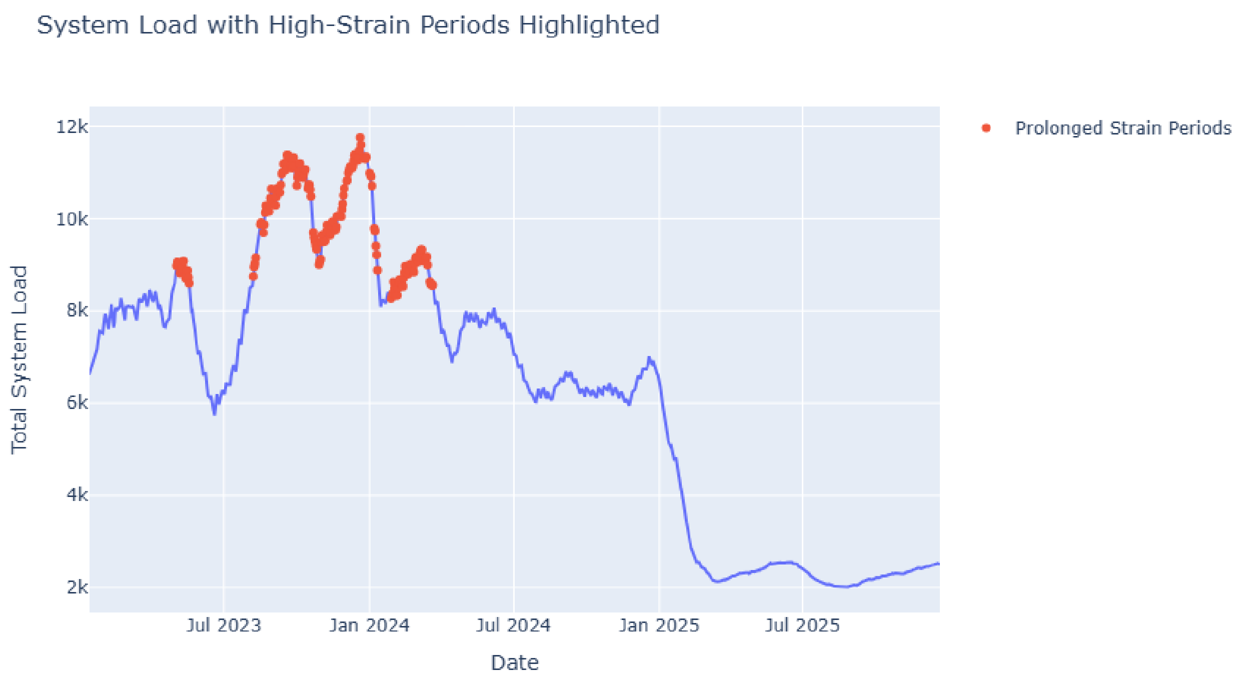


Figure 5: System Load with High_Strain Periods Highlighted.

These periods indicate sustained capacity stress rather than short-term spikes.

5 Correlation Insights

Correlation analysis between engineered KPIs reveals strong relationships between Total System Load, Net Intake, and rolling variability indicators, confirming that intake imbalance is a primary driver of system strain.

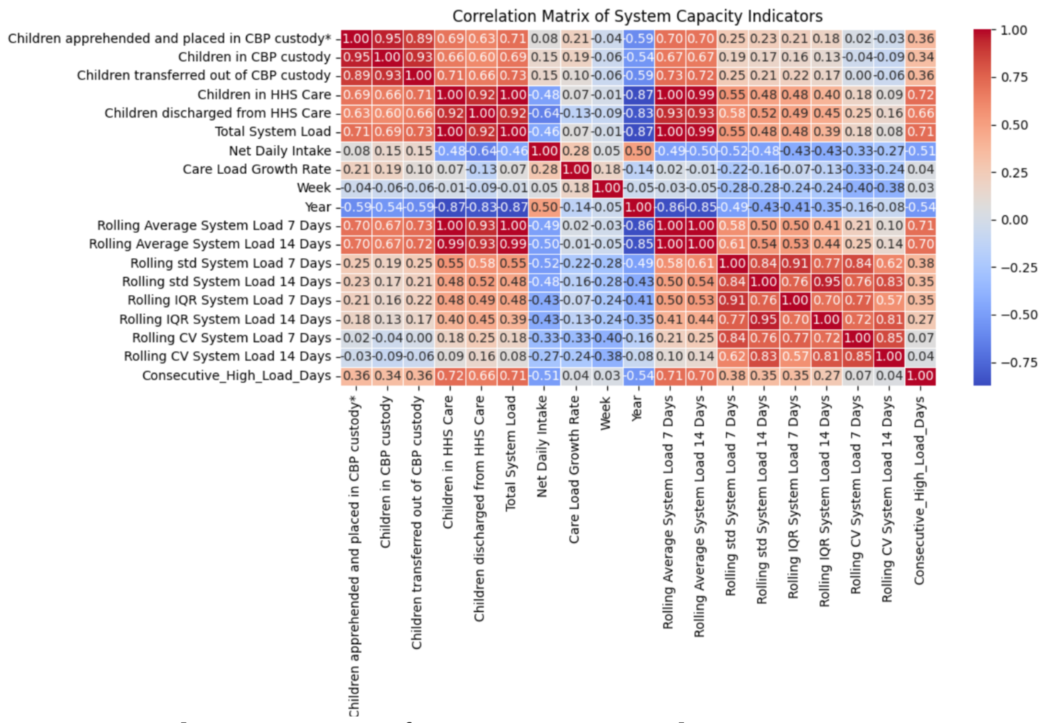


Figure 6: Correlation Heatmap of System Capacity Indicators

The heatmap confirms strong positive correlations between Total System Load, rolling averages, and intake-related KPIs, reinforcing that intake imbalance is a primary driver of prolonged system strain.

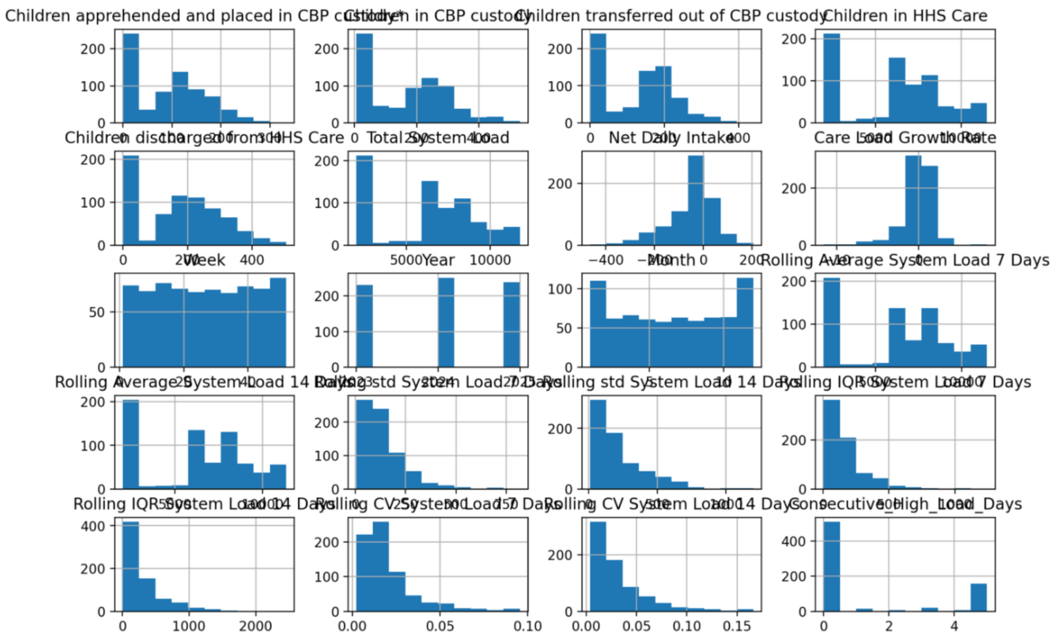
Interactive Dashboard Implementation (Streamlit)

An interactive dashboard was developed using Streamlit to operationalize the analytical findings into a real-time monitoring interface.

(Appendix A)

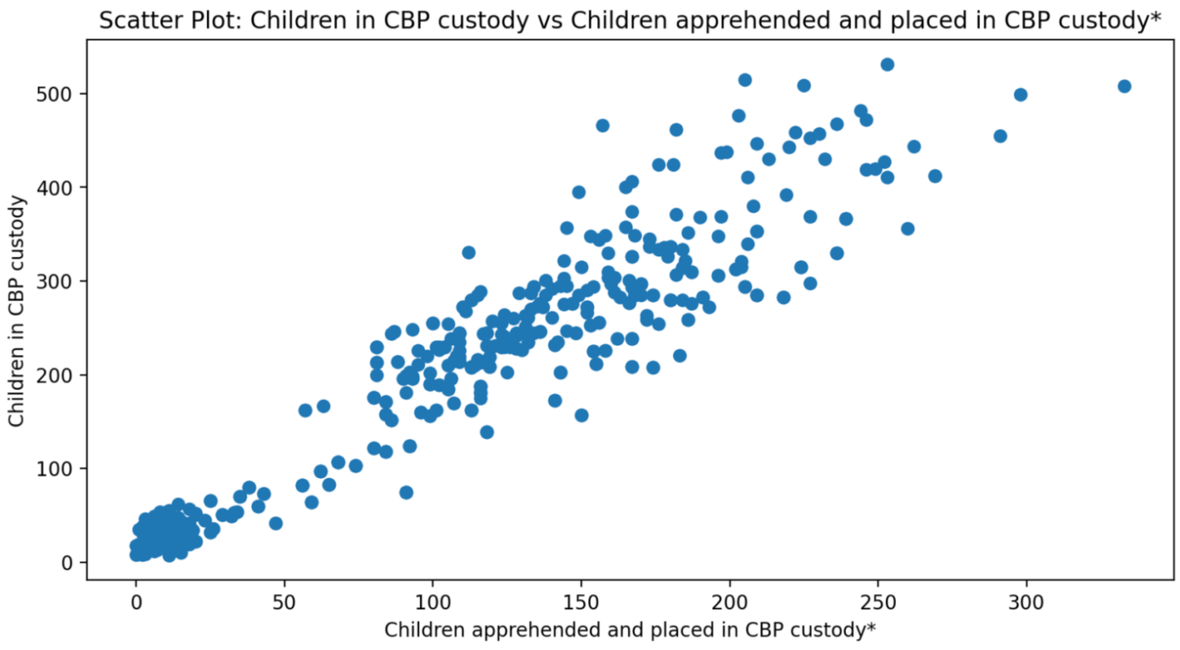
Core Dashboard Features:

- KPI summary metrics (load, intake, strain indicators)
- Time-series line charts for system load trends
- Histogram distribution analysis



- Interactive line and scatter visualizations

(Appendix B)



- Outlier detection using the IQR method

(Appendix C)

- Dataset preview and summary statistics

(Appendix D)

The dashboard enables dynamic exploration of system behavior and supports decision-oriented analysis through interactive visualization.

Policy and Operational Implications

The analysis suggests that system pressure is primarily driven by sustained positive net intake and slower discharge rates. During prolonged high-load periods, backlog accumulation increases operational burden and resource strain.

From an operational and policy perspective, the findings highlight the need for:

- Improved discharge processing efficiency
- Enhanced care capacity planning
- Proactive resource allocation during high-intake periods
- Continuous KPI-based monitoring of intake pressure and system load

Tracking Net Intake alongside Total System Load provides a more comprehensive framework for evaluating system resilience and responsiveness.

Technical Stack and Reproducibility

Programming & Tools:

- Python (Pandas, NumPy)

- Matplotlib & Seaborn (visualization)
- Streamlit (interactive dashboard)
- PowerShell (local execution)

Project structure:

- app.py – Streamlit dashboard
- cleaned_hhs_dataset.csv – Cleaned dataset
- Python scripts for data cleaning and KPI engineering

The project is fully reproducible and can be executed locally using:

```
python -m streamlit run app.py
```

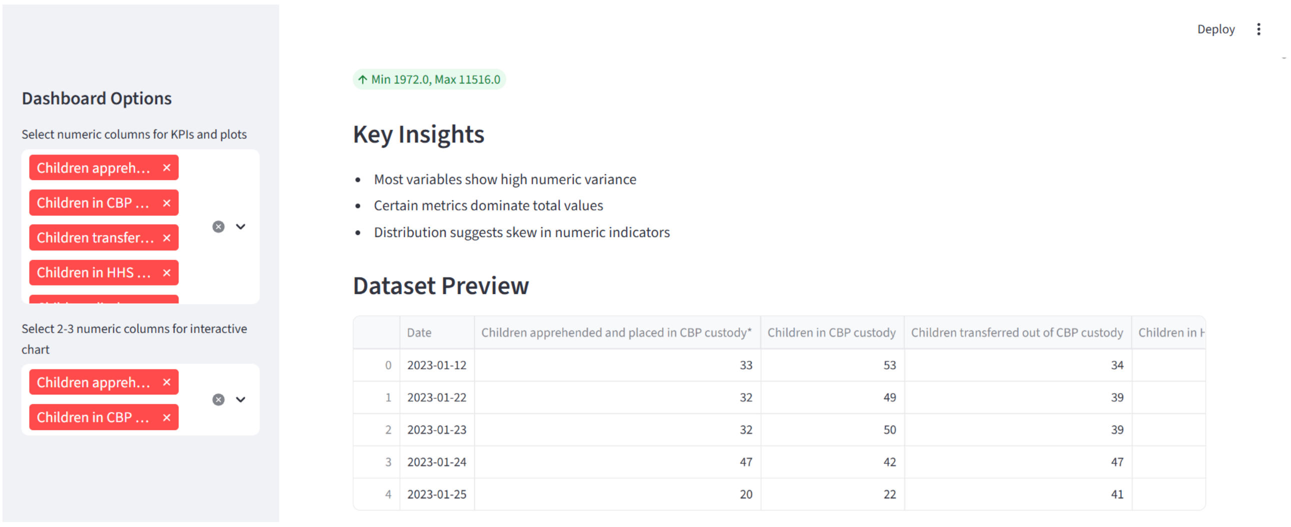
Conclusion

This project set out to transform raw operational data into a structured capacity analytics framework using KPI engineering, time-series analysis, and interactive dashboard visualization. The findings indicate that sustained intake pressure and limited discharge offset are key drivers of prolonged system strain.

By integrating data cleaning, validation, KPI development, and interactive analytics, the project provides a practical decision-support tool for monitoring care system capacity with proposed directions for improvement.

Appendix

Appendix A – Dashboard Visualizations



↑ Min 1972.0, Max 11516.0

Dashboard Options

Select numeric columns for KPIs and plots

Children appreh… ×
Children in CBP … ×
Children transfer… ×
Children in HHS … ×

Select 2-3 numeric columns for interactive chart

Children appreh… ×
Children in CBP … ×

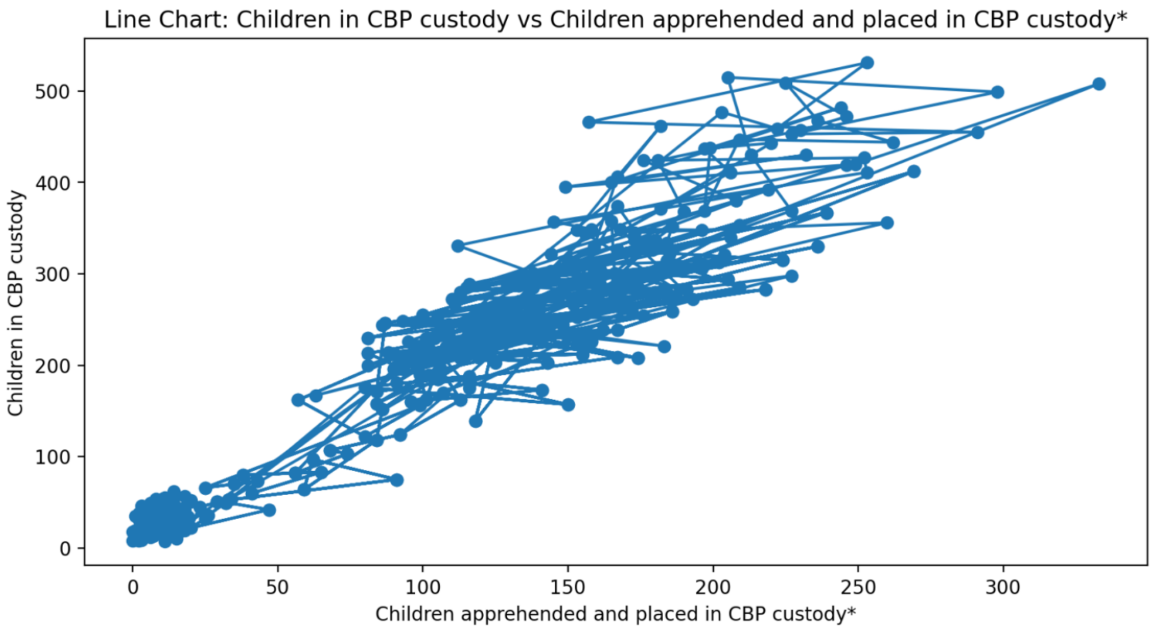
Key Insights

- Most variables show high numeric variance
- Certain metrics dominate total values
- Distribution suggests skew in numeric indicators

Dataset Preview

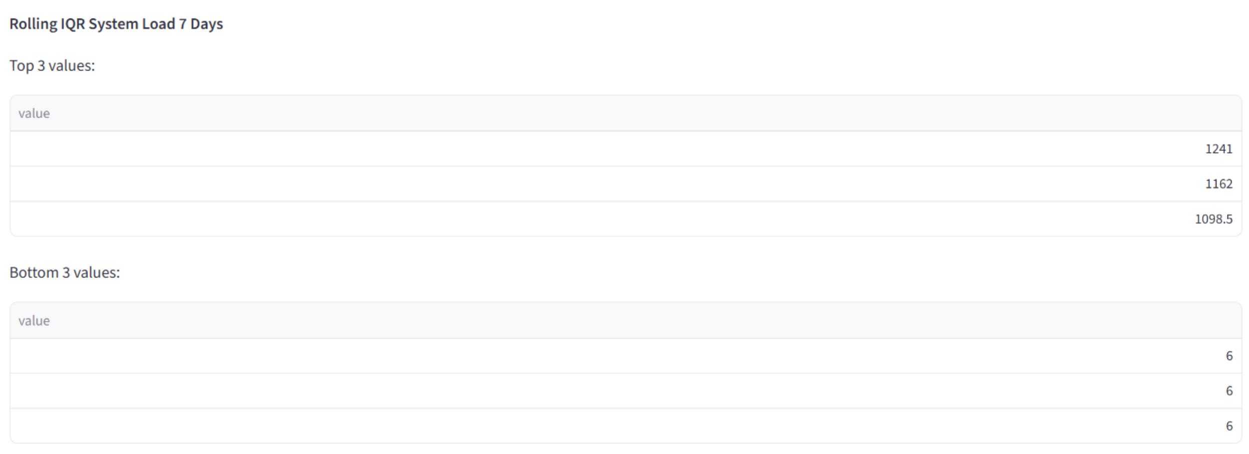
	Date	Children apprehended and placed in CBP custody*	Children in CBP custody	Children transferred out of CBP custody	Children in H
0	2023-01-12	33	53	34	
1	2023-01-22	32	49	39	
2	2023-01-23	32	50	39	
3	2023-01-24	47	42	47	
4	2023-01-25	20	22	41	

Appendix B – Interactive Line Chart



Appendix C – Additional KPI Outputs

Rolling IQR System Load 7 Days



Rolling IQR System Load 7 Days

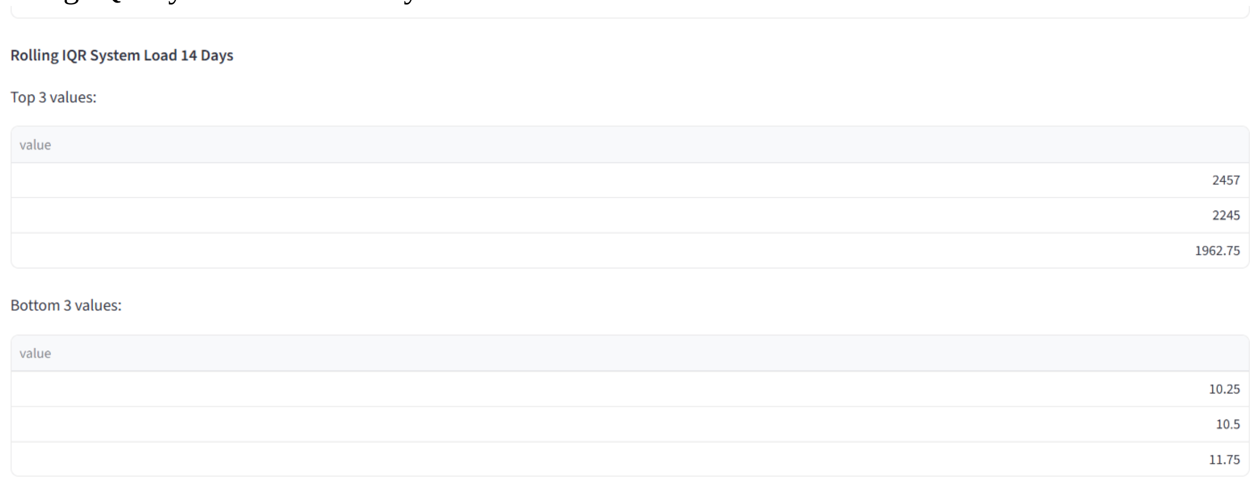
Top 3 values:

value
1241
1162
1098.5

Bottom 3 values:

value
6
6
6

Rolling IQR System Load 14 Days



Rolling IQR System Load 14 Days

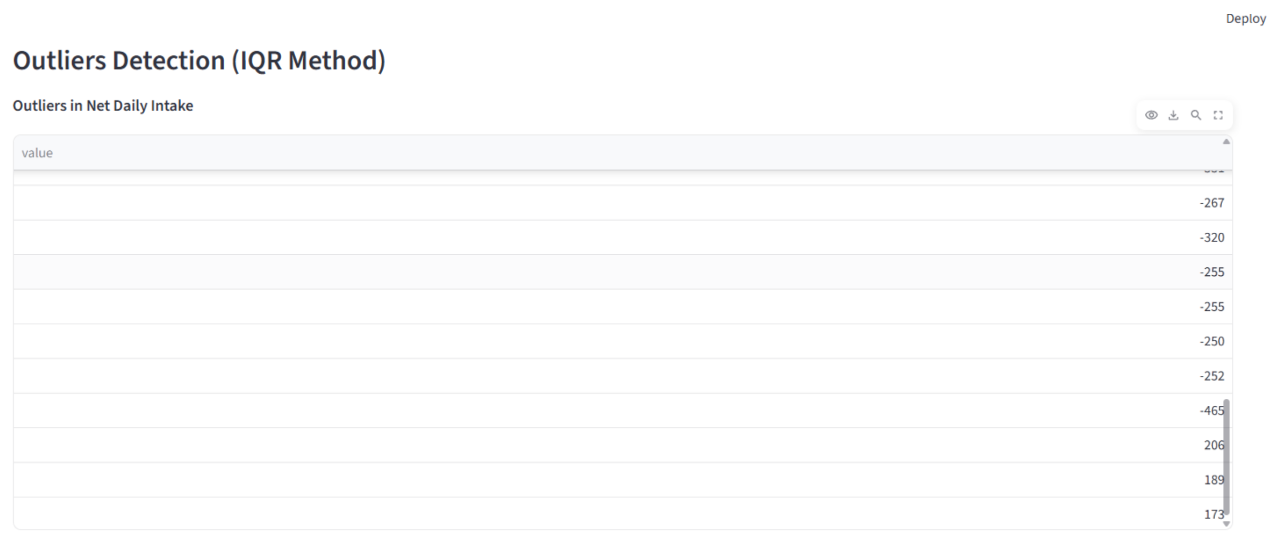
Top 3 values:

value
2457
2245
1962.75

Bottom 3 values:

value
10.25
10.5
11.75

Outliers Detection (IQR Method)

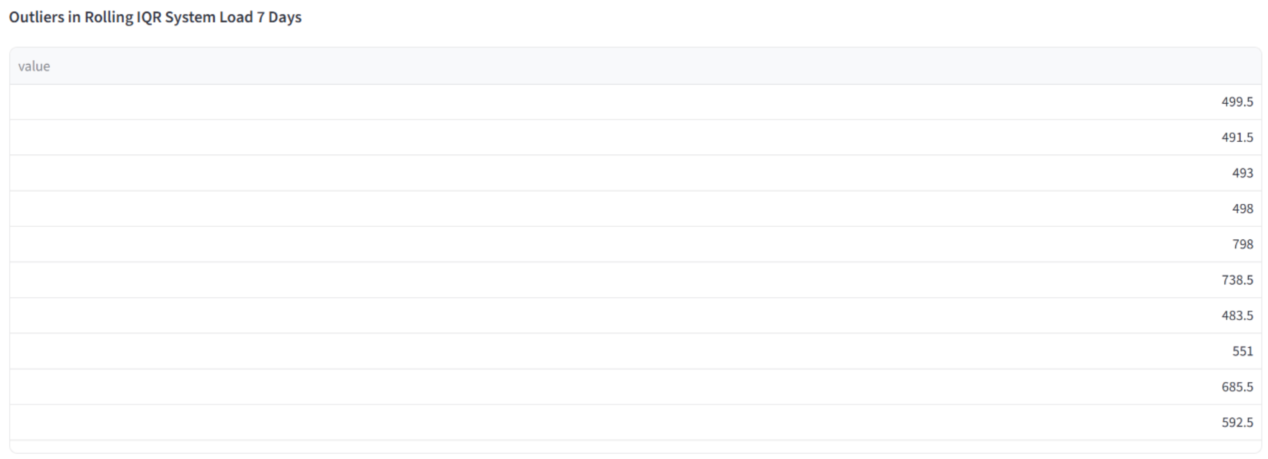


Outliers Detection (IQR Method)

Outliers in Net Daily Intake

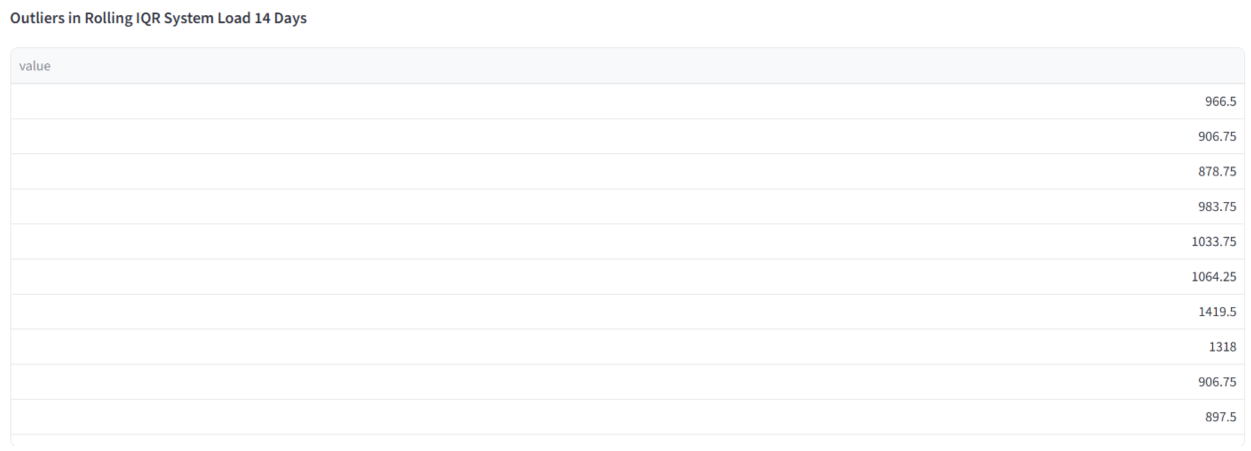
value
-267
-320
-255
-255
-250
-252
-465
206
189
173

Outliers in Rolling IQR System Load 7 Days



Outliers in Rolling IQR System Load 7 Days

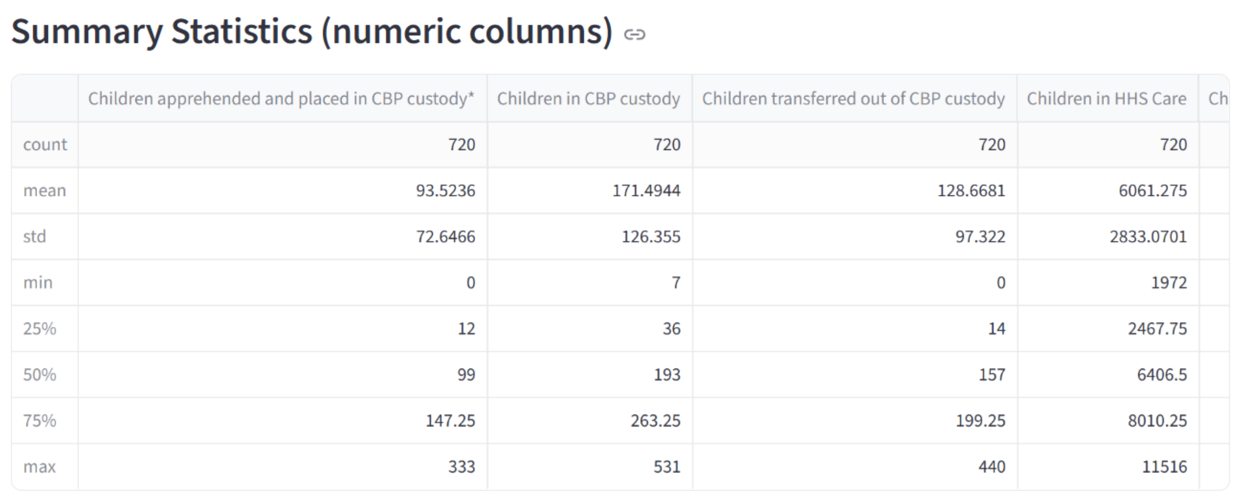
value
499.5
491.5
493
498
798
738.5
483.5
551
685.5
592.5



Outliers in Rolling IQR System Load 14 Days

value
966.5
906.75
878.75
983.75
1033.75
1064.25
1419.5
1318
906.75
897.5

Appendix D – Summary Statistics



Summary Statistics (numeric columns)

	Children apprehended and placed in CBP custody*	Children in CBP custody	Children transferred out of CBP custody	Children in HHS Care	Ch
count	720	720	720	720	
mean	93.5236	171.4944	128.6681	6061.275	
std	72.6466	126.355	97.322	2833.0701	
min	0	7	0	1972	
25%	12	36	14	2467.75	
50%	99	193	157	6406.5	
75%	147.25	263.25	199.25	8010.25	
max	333	531	440	11516	

Appendix E – Python Implementation

```
!pip install -U kaleido plotly
libcups2 libxcomposite1 libxdamage1 libxfixes3 libxrandr2 libgbm1 libxkbcommon0 libpango-1.0-
0 libcairo2 libasound2

import plotly.io as pio
import plotly.express as px

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('/content/HHS_Unaccompanied_Alien_Children_Program.csv')

print(df.head())

columns_to_convert = ['Children in CBP custody', 'Children transferred out of CBP custody',
'Children in HHS Care', 'Children discharged from HHS Care', 'Children apprehended and placed in
CBP custody*']

for col in columns_to_convert:
    df[col] = pd.to_numeric(df[col].astype(str).str.replace(',', ''), errors='coerce')

df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values(by='Date').reset_index(drop=True)

complete_index = pd.date_range(start = df['Date'].min(), end = df['Date'].max(), freq = 'D')

print("Initial Shape:", df.shape)
print("Columns:", df.columns.tolist())

missing_dates = complete_index.difference(df['Date'])

if missing_dates.empty:
  print("No missing dates found.")
else:
   print("Missing dates:", missing_dates)

duplicated_dates = df[df.duplicated(subset=['Date'])]
print(duplicated_dates)

df.dropna(subset=['Date'], inplace=True)

constraints_violated = (df['Children transferred out of CBP custody'] > df['Children in CBP
custody']) | (df['Children discharged from HHS Care'] > df['Children in HHS Care'])


if constraints_violated.any():
    print("Reporting Anomalies:")
    print(df[constraints_violated])
else:
    print("No reporting anomalies found.")

df['Total System Load'] = df['Children in CBP custody'] + df['Children in HHS Care']

print(df[['Date', 'Children in CBP custody', 'Children in HHS Care', 'Total System Load']].head())

df['Net Daily Intake'] = df['Children transferred out of CBP custody'] - df['Children discharged from
HHS Care']
print(df[['Date', 'Children transferred out of CBP custody', 'Children discharged from HHS Care']].
head())

df['Care Load Growth Rate'] = df['Total System Load'].pct_change(fill_method = None) * 100
print(df[['Date', 'Total System Load', 'Care Load Growth Rate']].head())

rolling_window = 3
df['Backlog Indicator'] = (df['Net Daily Intake'].rolling(window=rolling_window).sum() > 0)

print(df[['Date', 'Net Daily Intake', 'Backlog Indicator']].head(10))

rolling_window = 30
df['Backlog Indicator'] = (df['Net Daily Intake'].rolling(window=rolling_window).sum() > 0)
print(df[['Date', 'Net Daily Intake', 'Backlog Indicator']].head(10))

daily_care_load = df.groupby('Date')['Total System Load'].mean().reset_index()

daily_care_load.rename(columns={'Total System Load': 'Daily Avg Total System Load'},
inplace=True)
print("\nDaily Care Load Trends:")
print(daily_care_load.head())

df['Week'] = df['Date'].dt.isocalendar().week.astype(int)
df['Year'] = df['Date'].dt.year
weekly_care_load = df.groupby(['Year', 'Week'])['Total System Load'].mean().reset_index()
weekly_care_load.rename(columns={'Total System Load': 'Weekly Avg Total System Load'},
inplace=True)
print("\nWeekly Care Load Trends:")
print(weekly_care_load.head())

df['Month'] = df['Date'].dt.month
monthly_care_load = df.groupby(['Year', 'Month'])['Total System Load'].mean().reset_index()
monthly_care_load.rename(columns={'Total System Load': 'Monthly Avg Total System Load'},
inplace=True)
print("\nMonthly Care Load Trends:")
print(monthly_care_load.head())

df['Rolling Average System Load 7 Days'] = df['Total System Load'].rolling(window=7).mean()
print(df[['Date','Total System Load', 'Rolling Average System Load 7 Days']].head(10))

df['Rolling Average System Load 14 Days'] = df['Total System Load'].rolling(window=14).mean()
print(df[['Date','Total System Load', 'Rolling Average System Load 14 Days']].head(10))

df['Rolling std System Load 7 Days'] = df['Total System Load'].rolling(window=7).std()
```

```
print(df[['Date','Total System Load', 'Rolling std System Load 7 Days']].head(10))

df['Rolling std System Load 14 Days'] = df['Total System Load'].rolling(window=14).std()
print(df[['Date','Total System Load', 'Rolling std System Load 14 Days']].head(10))

df['Rolling IQR System Load 7 Days'] = df['Total System Load'].rolling(window=7).apply(lambda
x: x.quantile(0.75) - x.quantile(0.25))
print(df[['Date','Total System Load', 'Rolling IQR System Load 7 Days']].head(10))

df['Rolling IQR System Load 14 Days'] = df['Total System
Load'].rolling(window=14).apply(lambda x: x.quantile(0.75) - x.quantile(0.25))
print(df[['Date','Total System Load', 'Rolling IQR System Load 14 Days']].head(10))

df['Rolling CV System Load 7 Days'] = df['Total System Load'].rolling(window=7).std() / df['Total
System Load'].rolling(window=7).mean()
print(df[['Date','Total System Load', 'Rolling CV System Load 7 Days']].head(10))

df['Rolling CV System Load 14 Days'] = df['Total System Load'].rolling(window=14).std() /
df['Total System Load'].rolling(window=14).mean()
print(df[['Date','Total System Load', 'Rolling CV System Load 14 Days']].head(10))

high_load_threshold = df['Total System Load'].quantile(0.75)

consecutive_days_threshold = 5

df['Is_High_Load'] = df['Total System Load'] > high_load_threshold

df['Consecutive_High_Load_Days'] = df['Is_High_Load'].apply(lambda x: 1 if x else
0).rolling(window=consecutive_days_threshold).sum()
df['Prolonged_Strain_Window'] = df['Consecutive_High_Load_Days'] >=
consecutive_days_threshold

print(df[['Date', 'Total System Load', 'Is_High_Load', 'Prolonged_Strain_Window']].head(20))

df.to_csv('cleaned_hhs_dataset.csv', index=False)
print('Cleaned dataset saved to cleaned_hhs_dataset.csv')

df['Month'] = df['Date'].dt.to_period('M').astype(str)

fig1 = px.line(
    df,
    x='Date',
    y='Total System Load',
    color='Month',
    title='Total System Load Over Time by Month'
)
fig1.show()

df['Month'] = df['Date'].dt.to_period('M').astype(str)

fig2 = px.line(
    df,
    x='Date',
    y='Net Daily Intake',
    color='Month',
    title='Net Daily Intake Trend (System Pressure Indicator)'
)

fig2.show()
fig2.write_image("net_daily_intake.png")
fig2.write_html("net_daily_intake.html")

df['Month'] = df['Date'].dt.to_period('M').astype(str)

fig3 = px.line(
    df,
    x='Date',
    y='Rolling Average System Load 7 Days',
    color = 'Month',
    title='7-Day Rolling Average of Total System Load'
)

fig3.show()
fig3.write_image("rolling_avg_load.png")
fig3.write_html("rolling_avg_load.html")

fig4 = px.line(
    df,
    x='Date',
    y='Total System Load',
    title='System Load with High-Strain Periods Highlighted'
)

fig4.add_scatter(
    x=df[df['Prolonged_Strain_Window']]['Date'],
    y=df[df['Prolonged_Strain_Window']]['Total System Load'],
    mode='markers',
    name='Prolonged Strain Periods'
)

fig4.show()
fig4.write_image("strain_periods.png")
fig4.write_html("strain_periods.html")

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
correlation_matrix = df.select_dtypes(include=['number']).corr()

sns.heatmap(
    correlation_matrix,
    annot=True,
    fmt=".2f",
    cmap="coolwarm",
    linewidths=0.5
)

plt.title("Correlation Matrix of System Capacity Indicators")
plt.tight_layout()
plt.show()
```