Tooele County Housing Affordability Forecast

A Capstone Data Analysis

Author: Adam F.

Repository: GitHub - capstone_project_1

Table of Contents

Data Acquisition & Cleaning

Import Required Libraries

We load core libraries for data manipulation, visualization, and forecasting, including:

- `pandas` and `numpy` for data processing
- `matplotlib` and `seaborn` for plotting
- `statsmodels` and `sklearn` for regression and time series forecasting

In [1]:
```
# Load required libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import requests
from datetime import datetime
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error
import statsmodels.api as sm
import warnings
import kagglehub
import time
import json
import glob
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
warnings.filterwarnings('ignore')

# Set global float display to 3 decimal places
pd.options.display.float_format = '{:,.3f}'.format

# pd.reset_option('display.float_format') # to reset later, if needed
```

Load and Clean Data

- Median household income and gross rent (ACS)
- Home values (ZHVI)
- Rent prices (ZORI)
- Employment and wages (BLS and Utah DWS)

1.1. ACS - Median Household Income

> Definition from ACS B19013 group data dictionary:
>
> "Estimate!!Median household income in the past 12 months (in 2019 inflation-adjusted dollars)"

In [2]:
```
# INCOME DATA
# Fetching median household income data from the Census Bureau API

CENSUS_API_KEY = "de53f7491b5574a451233a6d04721c4176263ab4"
years = range(2015, 2023)  # 2023 ACS not fully available yet
results = []

for year in years:
    url = (
        f"https://api.census.gov/data/{year}/acs/acs5"
        f"?get=NAME,B19013_001E&for=county:045&in=state:49&key={CENSUS_API_KEY}"
    )
    response = requests.get(url)
```

```
    if response.status_code == 200:
        try:
            data = response.json()
            value = data[1][1]
            results.append({"year": year, "median_income": int(value)})
        except Exception as e:
            print(f"Error parsing {year}: {e}")
    else:
        print(f"Failed for {year}: {response.status_code}")

# Convert to DataFrame
income_df = pd.DataFrame(results)
income_df = income_df.sort_values("year")

# Create income_df2 with additional calculations
income_df2 = income_df.copy()
income_df2['pct_change'] = income_df2['median_income'].pct_change() * 100
income_df2['monthly_income'] = income_df2["median_income"] / 12
income_df2['rent_burden'] = income_df2["monthly_income"] * .3
income_df2['rent_burden_extreme'] = income_df2["monthly_income"] * .5
```

1.2. ACS - Median Gross Rent

In [3]:

```
STATE = "49"      # Utah
COUNTY = "045"    # Tooele County
VARIABLE = "B25064_001E"  # Median Gross Rent
acs_data = []

for year in years:
    url = (
        f"https://api.census.gov/data/{year}/acs/acs5"
        f"?get=NAME,{VARIABLE}&for=county:{COUNTY}&in=state:{STATE}&key={CENSUS_API
    )
    response = requests.get(url)
    if response.status_code == 200:
        json_data = response.json()
        df = pd.DataFrame(json_data[1:], columns=json_data[0])
        df["year"] = year
        acs_data.append(df)
    else:
        print(f"Error {response.status_code} for year {year}")
    time.sleep(0.5)  # Be nice to the API

# Combine and format
acs_df = pd.concat(acs_data)
acs_df = acs_df.rename(columns={VARIABLE: "ACS_Rent"})
acs_df["ACS_Rent"] = pd.to_numeric(acs_df["ACS_Rent"])
acs_df["year"] = acs_df["year"].astype(int)
```

1.3. Zillow Home Value Index

> "**Zillow Home Value Index (ZHVI):** A measure of the typical home value and market changes across a given region and housing type. It reflects the typical

> value for homes in the 35th to 65th percentile range. Available as a smoothed, seasonally adjusted measure and as a raw measure.
>
> - "Zillow publishes top-tier ZHVI ($, typical value for homes within the 65th to 95th percentile range for a given regio$, typical value for homes within the 5th to 35th percentile range for a given region).
> - "Zillow also publishes ZHVI for all single-family residences ($, typical value for all single - family homes in a given region), for condo/coops$), for all homes with 1, 2, 3, 4 and 5+ bedrooms ($)."
>
> - Zillow Research

In [4]:
```
# Housing Prices - Zillow ZHVI (Zillow Home Value Index)
path = kagglehub.dataset_download("robikscube/zillow-home-value-index")
zhvi_path = os.path.join(path, "ZHVI.csv")  # Adjust if the filename is different
zhvi_df = pd.read_csv(zhvi_path)

zhvi_df.rename(columns={zhvi_df.columns[0]: "date"}, inplace=True)

# Filter for Utah
zhvi_ut_df = zhvi_df[['date', 'Utah']].copy()
# Format 'date' to datetime and normalize dates to first of the month
zhvi_ut_df['date'] = pd.to_datetime(zhvi_ut_df['date']).dt.to_period('M').dt.to_tim
```

In [5]:
```
# Join Utah ZHVI data with Tooele and Salt Lake County Data

# County data is reported on the last day of the month and State data reported on t
# dates will be normalized to the first of the month, with county data shifted forw

zhvi_county_df = pd.read_csv("data/zhvi_county_data.csv")
# Format 'date' to datetime and normalize dates to first of the month
zhvi_county_df['date'] = pd.to_datetime(zhvi_county_df['date']).dt.to_period('M').d
# Shift month forward by 1
zhvi_county_df['date'] = zhvi_county_df['date'] + pd.DateOffset(months=1)

zhvi_all_df = pd.merge(zhvi_ut_df, zhvi_county_df, on='date', how='outer')

housing_df = zhvi_all_df[zhvi_all_df['date'] >= '2015-01-01']
```

1.4. Zillow Observed Rent Index

> "**Zillow Observed Rent Index (ZORI):** A smoothed measure of the typical observed market rate rent across a given region. ZORI is a repeat-rent index that is weighted to the rental housing stock to ensure representativeness across the entire market, not just those homes currently listed for-rent. The index is dollar-denominated by computing the mean of listed rents that fall

> into the 35th to 65th percentile range for all homes and apartments in a given region, which is weighted to reflect the rental housing stock."
>
> - Zillow Research

In [6]:
```
# Download and load the Zillow Rent Index dataset
path = kagglehub.dataset_download("zillow/rent-index")
rent_index_path = os.path.join(path, "price.csv")
rent_index_df = pd.read_csv(rent_index_path)

# Filter for Tooele County
rent_index_df = rent_index_df[rent_index_df['County'] == "Tooele"]

# Drop unneeded columns
columns_to_drop = ["City Code", "Metro", "State", "Population Rank"]
rent_index_df.drop(columns=columns_to_drop, inplace=True)

# Set 'City' as the index and transpose the date columns
rent_index_df.set_index('City', inplace=True)

# Transpose the DataFrame
rent_df1 = rent_index_df.T.copy()

# Convert index to datetime if it's month strings like "2020-01"
rent_df1.index = pd.to_datetime(rent_df1.index, errors='coerce')

# Clean up any rows with invalid dates (if any)
rent_df1 = rent_df1[rent_df1.index.notnull()]

# Add a County mean summary column
rent_df1['Tooele County'] = rent_df1.mean(axis=1)

rent_df1 = rent_df1.dropna(subset=["Tooele County"])

rent_df1 = rent_df1.reset_index().rename(columns={'index': 'date'})
rent_df1.index.name = None
rent_df1 = rent_df1[(rent_df1['date'] >= '2015-01-01') & (rent_df1['date'] < '2017-
```

In [7]:
```
# Load the CSV file from the 'data' directory
csv_path = os.path.join("data", "county_zori_data.csv")
rent_index_df2 = pd.read_csv(csv_path)

# Filter for Tooele County
rent_index_df2 = rent_index_df2[rent_index_df2['RegionName'] == "Tooele County"]
rent_index_df2.head()

# Drop unneeded columns
columns_to_drop = ["RegionID", "RegionType", "StateName", "State", "Metro", "SizeRa
rent_index_df2.drop(columns=columns_to_drop, inplace=True)

# Set 'City' as the index and transpose the date columns
rent_index_df2.set_index('RegionName', inplace=True)
rent_df2 = rent_index_df2.T.copy()
```

```
# Convert index to datetime if it's month strings like "2020-01"
rent_df2.index = pd.to_datetime(rent_df2.index, errors='coerce')

# Clean up any rows with invalid dates (if any)
rent_df2 = rent_df2[rent_df2.index.notnull()]

# Drop rows where the summary column is NaN
rent_df2 = rent_df2.dropna(subset=["Tooele County"])

# Reset index and rename
rent_df2 = rent_df2.reset_index().rename(columns={'index': 'date'})
rent_df2.index.name = None

# Format 'date' to datetime and normalize dates to first of the month
rent_df2['date'] = pd.to_datetime(rent_df2['date']).dt.to_period('M').dt.to_timesta
# Shift month forward by 1
rent_df2['date'] = rent_df2['date'] + pd.DateOffset(months=1)

# Filter for dates starting from January 2015
rent_df2 = rent_df2[rent_df2['date'] >= '2015-01-01']
```

In [8]:

```
# Select just the date and Tooele County columns
df1 = rent_df1[['date', 'Tooele County']].copy()
df2 = rent_df2[['date', 'Tooele County']].copy()

# Concatenate the two DataFrames
combined_df = pd.concat([df1, df2])

# Set 'date' as the index
combined_df.set_index('date', inplace=True)

#  Create a complete monthly date range
full_index = pd.date_range(start=combined_df.index.min(), end=combined_df.index.max

# Reindex and interpolate missing months
combined_df = combined_df.reindex(full_index)
combined_df.index.name = 'date'

# Interpolate missing rent values linearly
start_value = rent_df1['Tooele County'].iloc[-1]                          # Last know
end_value = rent_df2['Tooele County'].iloc[0]                             # First kno
steps = 72                                                                # Calculate
increment = (end_value - start_value) / steps
gap_start = rent_df1['date'].max() + pd.DateOffset(months=1)              # Find the
gap_end = rent_df2['date'].min() - pd.DateOffset(months=1)
gap_dates = pd.date_range(start=gap_start, end=gap_end, freq='MS')        # Create th
gap_values = [start_value + i * increment for i in range(1, steps + 1)] # Generate
gap_df = pd.DataFrame({                                                   # Build a D
    'date': gap_dates,
    'Tooele County': gap_values
}).set_index('date')
combined_df.update(gap_df)                                                # Insert ga


# Reset index and rename the columns
rent_df = combined_df.reset_index()
```

```
rent_df.rename(columns={'index': 'date'}, inplace=True)
rent_df = rent_df.rename(columns={'Tooele County': 'rent_index'})
```

1.5. Local Economics - DWS: "Utah Economic Data Viewer"

In [9]:
```
# Local Economics - Utah Department of Workforce Services (DWS)

# Set the path to your data folder
data_dir = 'data'
pattern = os.path.join(data_dir, 'industry-and-wages-*.csv')

# Collect all matching file paths
file_paths = glob.glob(pattern)

# Read and combine all CSVs, adding a 'Year' column extracted from the filename
df_list = []
for file_path in file_paths:
    year = int(os.path.basename(file_path).split('-')[-1].split('.')[0])  # Extract
    df = pd.read_csv(file_path)
    df['Year'] = year
    df_list.append(df)

local_df = pd.concat(df_list, ignore_index=True)                    # Merge into a
local_df.sort_values(by='Year', inplace=True)                       # Sort by Year
local_df['NAICS Sector'] = local_df['NAICS Sector'].astype(str)     # Ensure the co
local_df2 = local_df.copy()                                            # Create a c
local_df = local_df[local_df['NAICS Sector'].str.match(r'^\d{3}$')] # Filter for ex
# local_df['NAICS Sector'] = local_df['NAICS Sector'].astype(int)   # Convert Back
local_df.rename(columns={'Year': 'year'}, inplace=True)             # rename "Year"
local_df['year'] = local_df['year'].astype(int)                     # Ensure 'year'

local_df['rent_burden'] = local_df["Average Monthly Wage"] * .3
local_df['rent_burden_extreme'] = local_df["Average Monthly Wage"] * .5
```

1.6. Data Cleaning and Merging: create data views by year

- Housing Prices
- Rent Prices
- Local Economics

To do this, we:

- Filter for Tooele County
- Interpolate missing values (especially in ZORI)
- Align time indexes (yearly)
- Adjust all monetary data to 2023 dollars using CPI
- Merge datasets into a single analysis-ready DataFrame

1.6.1. Create view, housing prices grouped by annual mean, with estimated mean down payment

In [10]:
```
# Create view, housing prices grouped by annual mean, with estimated mean down paym
housing_mn_df = housing_df.copy()
housing_mn_df['year'] = housing_mn_df['date'].dt.year
housing_mn_df = housing_mn_df.groupby('year', as_index=False)['Tooele County'].mean
housing_mn_df.rename(columns={'Tooele County': 'median_price'}, inplace=True)
housing_mn_df['down_payment'] = housing_mn_df['median_price'] * 0.15
housing_mn_df['down_payment_pct'] = (housing_mn_df['down_payment'] / income_df['med
housing_mn_df['pct_change'] = housing_mn_df['median_price'].pct_change() * 10
```

1.6.2. Create view, rent prices grouped by annual mean

In [11]:
```
# Create view, rent prices grouped by annual mean
rent_mn_df = rent_df.copy()
rent_mn_df['year'] = rent_mn_df['date'].dt.year
rent_mn_df = rent_mn_df.groupby('year', as_index=False)['rent_index'].mean()
```

In [12]:
```
rent_annual = pd.merge(rent_mn_df, acs_df[['year', 'ACS_Rent']], on='year', how='in

# Add error/percent diff column
rent_annual['zori_pct_change'] = rent_annual['rent_index'].pct_change() * 100
rent_annual['acs_pct_change'] = rent_annual['ACS_Rent'].pct_change() * 100
rent_annual['Difference'] = rent_annual['rent_index'] - rent_annual['ACS_Rent']
rent_annual['Percent_Diff'] = (rent_annual['Difference'] / rent_annual['ACS_Rent'])
```

1.6.3. Create pivote tables for local_df2 for Average Monthly Wage and Average Employment

In [13]:
```
# Create pivote tables for local_df2 for Average Monthly Wage and Average Employmen
local_df2 = local_df2[local_df2['NAICS Sector'].str.match(r'^\d{2}$')] # Filter for
local_df2.rename(columns={'Year': 'year'}, inplace=True)              # rename "Year
local_df2['year'] = local_df2['year'].astype(int)                      # Ensure 'yea
local_df2 = local_df2[['year', 'Industry Sector', 'NAICS Sector', 'Average Monthly
local_df2 = local_df2.dropna(subset=['Average Monthly Wage', 'Average Employment'])

# Pivot table for Average Monthly Wage by year and NAICS sector
wage_pivot = local_df2.pivot_table(
    index='year',
    columns='Industry Sector',
    values='Average Monthly Wage',
    aggfunc='mean'
)
```

In [14]:
```
# Create view, industry stats grouped by annual mean
local_agg = local_df.groupby('year').agg({
    'Average Monthly Wage': 'mean',
    'rent_burden': 'mean',
    'rent_burden_extreme': 'mean'
}).reset_index()

local_agg.head(10)

local_agg2 = local_agg.copy()
local_agg2.drop(columns=['rent_burden'], inplace=True)
```

```
local_agg2.drop(columns=['rent_burden_extreme'], inplace=True)

local_agg2['Estimated Annual Wages'] = local_agg2['Average Monthly Wage'] * 12
```

1.7.4. Merge aggregate Wage and Employment data

In [15]:
```
# Aggregate if multiple entries per year (e.g., monthly data)
columns = ['median_income', 'Average Monthly Wage',
           'Estimated Annual Wages', 'rent_index', 'ACS_Rent']

#  Merge local,bls,and income dataframes
merged_df = income_df.merge(local_agg2, on='year', how='right')
merged_df = merged_df.merge(rent_annual[['year', 'rent_index', 'ACS_Rent']], on='ye

# Check if 2025 already exists
if 2025 not in merged_df['year'].values:
    new_row = pd.DataFrame({'year': [2025]})
    merged_df = pd.concat([merged_df, new_row], ignore_index=True)

merged_df.sort_values('year', inplace=True)
# Ensure all columns are numeric
for col in columns:
    merged_df[col] = pd.to_numeric(merged_df[col], errors='coerce')
```

In [16]:
```
# Line plots: Rent and Housing trends over time
def forecast_column(df, column, target_years):
    # Filter valid rows for training
    train = df[df[column].notna()]
    X_train = train[['year']]
    y_train = train[column]

    # Train linear regression
    model = LinearRegression().fit(X_train, y_train)

    # Forecast for target years
    for year in target_years:
        df.loc[df['year'] == year, column] = model.predict([[year]])[0]

    return df
```

In [17]:
```
# Line plots: Rent and Housing trends over time
forecast_years = [2023, 2024, 2025]
cols_to_forecast = ['rent_index', 'ACS_Rent', 'median_income']

for col in cols_to_forecast:
    merged_df = forecast_column(merged_df, col, forecast_years)

forecast_years = [2024, 2025]
cols_to_forecast = ['Average Monthly Wage',     'Estimated Annual Wages']

for col in cols_to_forecast:
    merged_df = forecast_column(merged_df, col, forecast_years)
```

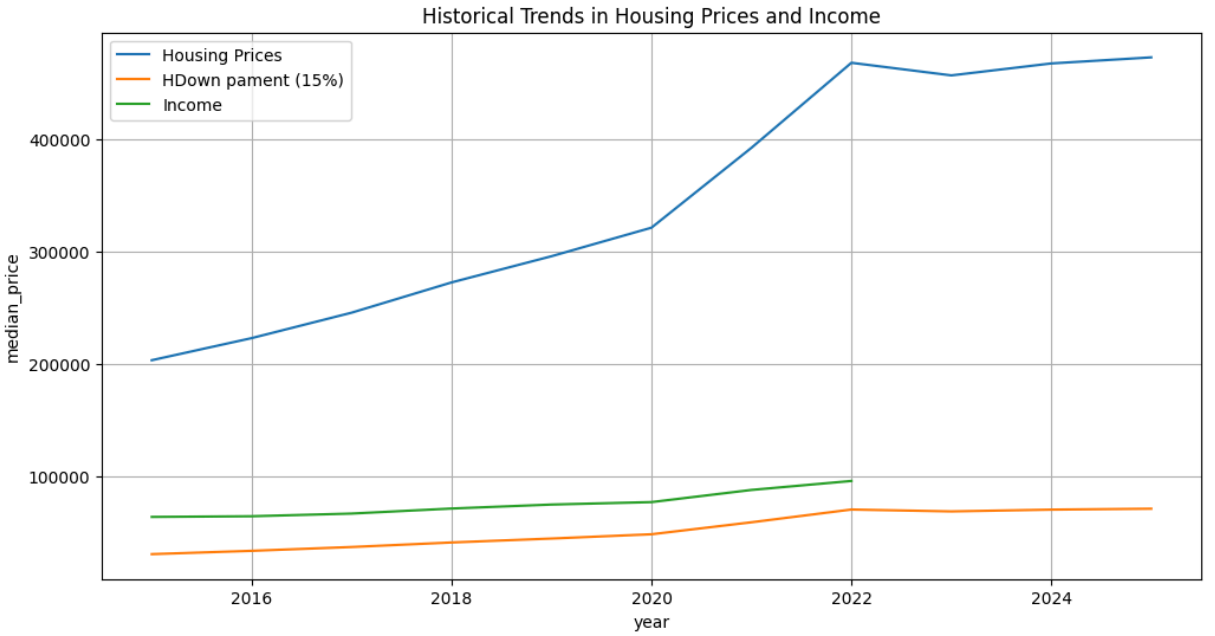
2. Exploratory Data Analysis (EDA)

We visualize historical trends in:

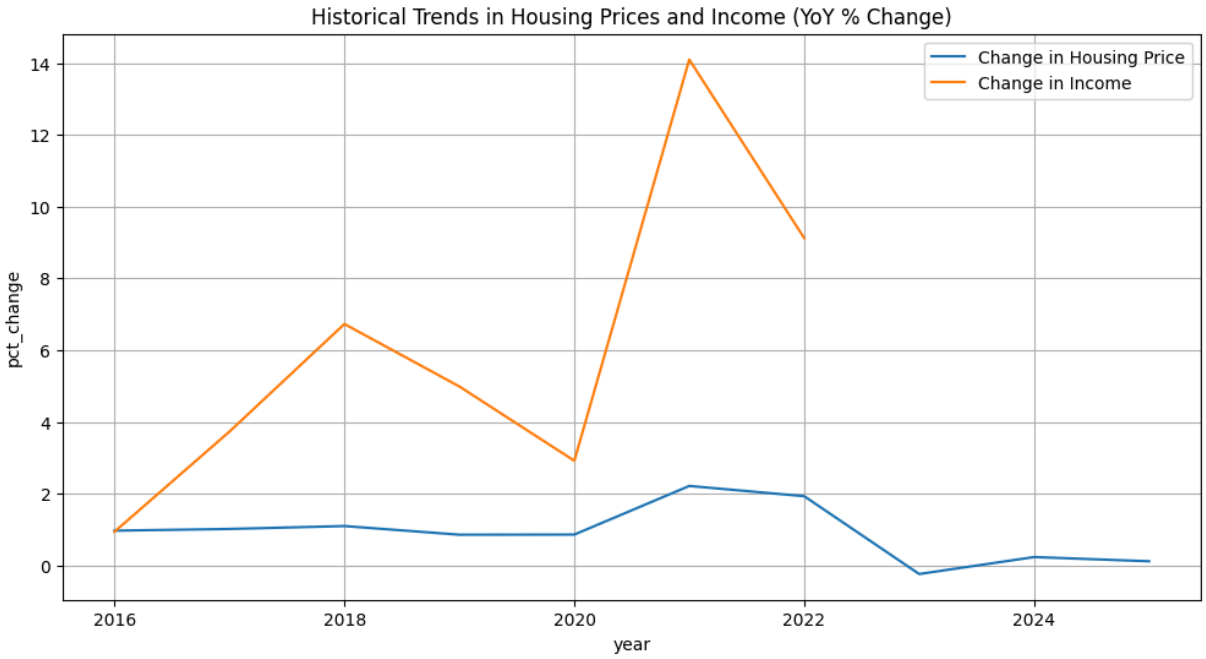
- Home prices
- Rent prices
- Median household income

2.1. Housing Price Analysis

In [18]:
```
# Line plots: Housing and Income trends over time
plt.figure(figsize=(12, 6))
sns.lineplot(data=housing_mn_df, x='year', y='median_price', label='Housing Prices'
sns.lineplot(data=housing_mn_df, x='year', y='down_payment', label='HDown pament (1
sns.lineplot(data=income_df, x='year', y='median_income', label='Income')
plt.title('Historical Trends in Housing Prices and Income')
plt.legend(); plt.grid(); plt.show()
```

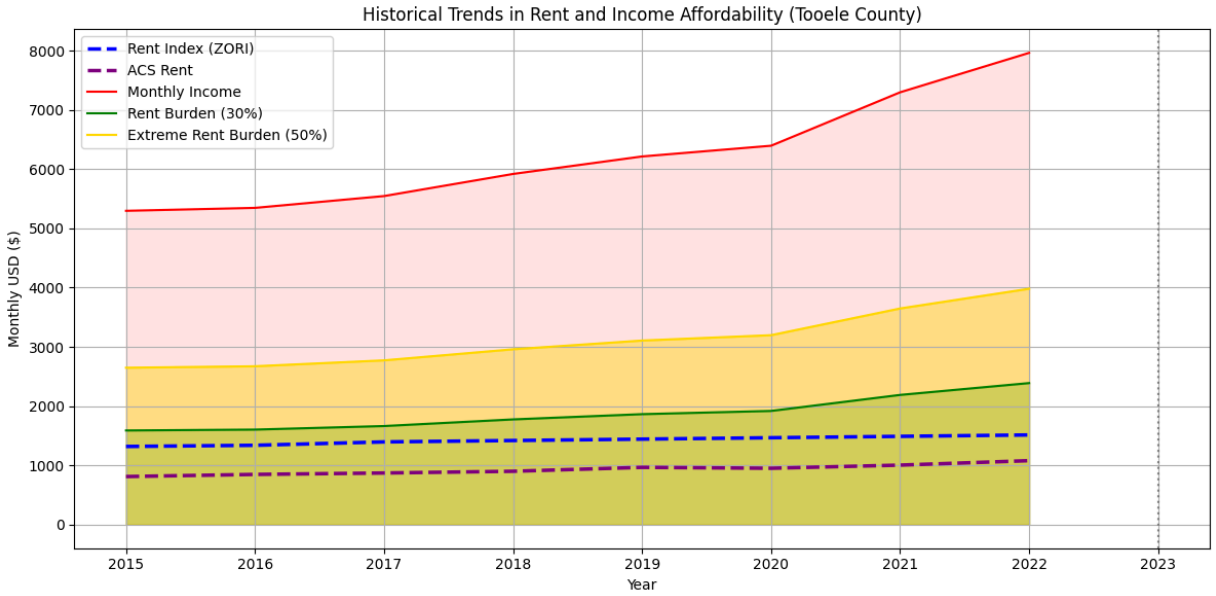


In [19]:
```
# Line plots: Housing and Income trends over time
plt.figure(figsize=(12, 6))
sns.lineplot(data=housing_mn_df, x='year', y='pct_change', label='Change in Housing
sns.lineplot(data=income_df2, x='year', y='pct_change', label='Change in Income')
plt.title('Historical Trends in Housing Prices and Income (YoY % Change)')
plt.legend(); plt.grid(); plt.show()
```



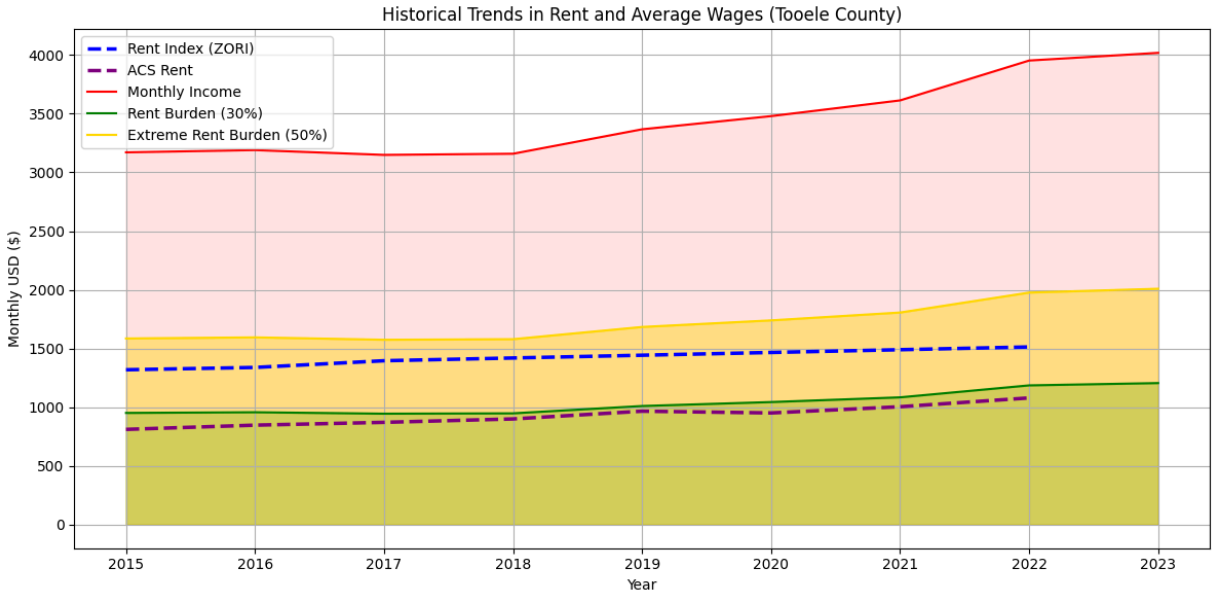
2.2. Rent Analysis (by Census Data)

In [20]:
```
# Line plots: Rent and Income trends over time (Income from Census Data)
plt.figure(figsize=(12, 6))

# Rent Index and ACS Rent as dotted lines
sns.lineplot(data=rent_annual, x='year', y='rent_index', label='Rent Index (ZORI)',
sns.lineplot(data=rent_annual, x='year', y='ACS_Rent', label='ACS Rent', linestyle=
# Monthly Income
sns.lineplot(data=income_df2, x='year', y='monthly_income', label='Monthly Income',
plt.fill_between(income_df2['year'], income_df2['monthly_income'], alpha=0.1, color
# Rent Burden (30%)
sns.lineplot(data=income_df2, x='year', y='rent_burden', label='Rent Burden (30%)',
plt.fill_between(income_df2['year'], income_df2['rent_burden'], alpha=0.3, color='g
# Extreme Rent Burden (50%)
sns.lineplot(data=income_df2, x='year', y='rent_burden_extreme', label='Extreme Ren
plt.fill_between(income_df2['year'], income_df2['rent_burden_extreme'], alpha=0.43,
# Formatting
plt.title('Historical Trends in Rent and Income Affordability (Tooele County)')
plt.ylabel('Monthly USD ($)')
plt.xlabel('Year')
plt.legend()
plt.grid(True)
plt.axvline(x=2023, color='gray', linestyle=':', label='Forecast Start (2023)')
plt.tight_layout()
plt.show()
```



2.3. Rent Analysis (by DWS Data)

In [21]:
```
# Line plots: Rent and Income trends over time (Income from DWS Data)
plt.figure(figsize=(12, 6))

# Rent Index and ACS Rent as dotted lines
sns.lineplot(data=rent_annual, x='year', y='rent_index', label='Rent Index (ZORI)',
sns.lineplot(data=rent_annual, x='year', y='ACS_Rent', label='ACS Rent', linestyle=
# Monthly Income
sns.lineplot(data=local_agg, x='year', y='Average Monthly Wage', label='Monthly Inc
plt.fill_between(local_agg['year'], local_agg['Average Monthly Wage'], alpha=0.1, c
# Rent Burden (30%)
sns.lineplot(data=local_agg, x='year', y='rent_burden', label='Rent Burden (30%)',
plt.fill_between(local_agg['year'], local_agg['rent_burden'], alpha=0.3, color='gre
# Extreme Rent Burden (50%)
sns.lineplot(data=local_agg, x='year', y='rent_burden_extreme', label='Extreme Rent
plt.fill_between(local_agg['year'], local_agg['rent_burden_extreme'], alpha=0.43, c
# Formatting
plt.title('Historical Trends in Rent and Average Wages (Tooele County)')
plt.ylabel('Monthly USD ($)')
plt.xlabel('Year')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



3. Modeling & Forecasting

Using ARIMA and Linear Regression, we forecast:

- 2025 median household income
- 2025 median home value
- 2025 average rent

We calculate these indices for each year and compare them to standard thresholds:

- 30% → Cost-burdened
- 50% → Severely burdened

3.1. 2025 Forecast

In [22]:
```
# Forecast 2025 using linear regression

X = income_df[['year']]
y = income_df['median_income']
model = LinearRegression().fit(X, y)
pred_2025 = model.predict([[2025]])
income_forecast = pred_2025[0]

X = income_df2[['year']]
y = income_df2['monthly_income']
model = LinearRegression().fit(X, y)
pred_2025 = model.predict([[2025]])
monthly_income_forecast = pred_2025[0]


X = local_agg2[['year']]
y = local_agg2['Estimated Annual Wages']
model = LinearRegression().fit(X, y)
```

```
pred_2025 = model.predict([[2025]])
local_forecast = pred_2025[0]


X = housing_mn_df[['year']]
y = housing_mn_df['median_price']
model = LinearRegression().fit(X, y)
pred_2025 = model.predict([[2025]])
housing_forecast = pred_2025[0]


X = rent_annual[['year']]
y = rent_annual['rent_index']
model = LinearRegression().fit(X, y)
pred_2025 = model.predict([[2025]])
zori_forecast = pred_2025[0]


X = rent_annual[['year']]
y = rent_annual['ACS_Rent']
model = LinearRegression().fit(X, y)
pred_2025 = model.predict([[2025]])
acs_rent_forecast = pred_2025[0]


print(f"Predicted Median Annual Income  (Census) for 2025: ${income_forecast:,.2f}"
print(f"Predicted Median Monthly Income (Census) for 2025: ${monthly_income_forecas
print(f"Predicted Mean Wages            (DWS)    for 2025: ${local_forecast:,.2f}")
print(f"Predicted Housing Price         (ZHVI)   for 2025: ${housing_forecast:,.2f}
print(f"Predicted Rent                  (ZORI)   for 2025: ${zori_forecast:,.2f}")
print(f"Predicted Median Rent           (Census) for 2025: ${acs_rent_forecast:,.2f
```

```
Predicted Median Annual Income  (Census) for 2025: $103,984.99
Predicted Median Monthly Income (Census) for 2025: $8,665.42
Predicted Mean Wages            (DWS)    for 2025: $49,776.81
Predicted Housing Price         (ZHVI)   for 2025: $503,355.65
Predicted Rent                  (ZORI)   for 2025: $1,604.80
Predicted Median Rent           (Census) for 2025: $1,158.21
```

3.2. Multivariate Regression

In [23]:

```
# Example: merged_df contains all the relevant features
X = merged_df[['Average Monthly Wage', 'Estimated Annual Wages']]
y = merged_df['rent_index']  # Replace with your actual housing price column

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

model = LinearRegression().fit(X_train, y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import r2_score, mean_absolute_error
print("R²:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
R²: 0.856560722637115
MAE: 29.508832953180217
```

4. Affordability Analysis

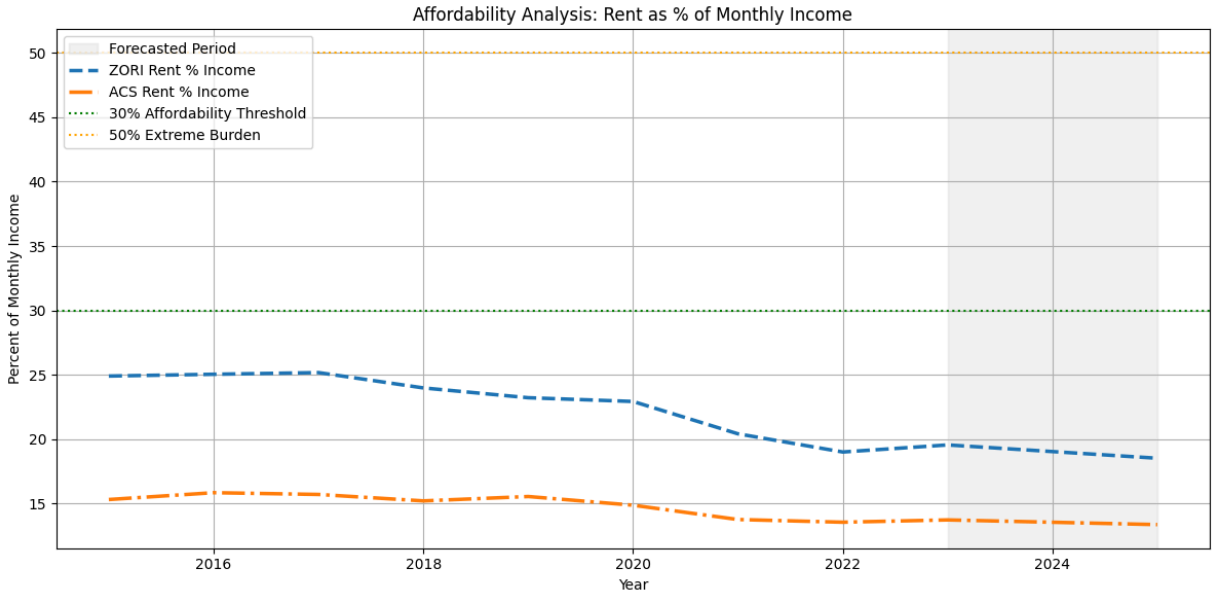
Visualize Forecasts and Affordability Thresholds:

- Forecasted income vs. housing costs
- Affordability ratio trends with 30% and 50% thresholds

4.1. ACS Median Income Analysis

In [24]:
```
# Monthly affordability thresholds based on income
merged_df['monthly_income'] = merged_df['median_income'] / 12

# Define affordability burden thresholds
merged_df['afford_pct_rent_census'] = merged_df['rent_index'] / merged_df['monthly_
merged_df['afford_pct_rent_acs_census'] = merged_df['ACS_Rent'] / merged_df['monthl

# Label risk levels (optional)
def affordability_risk(pct):
    if pct >= 50:
        return "Extreme Burden"
    elif pct >= 30:
        return "Burdened"
    else:
        return "Affordable"

merged_df['rent_risk_census'] = merged_df['afford_pct_rent_census'].apply(affordabi
merged_df['acs_rent_risk_census'] = merged_df['afford_pct_rent_acs_census'].apply(a
```

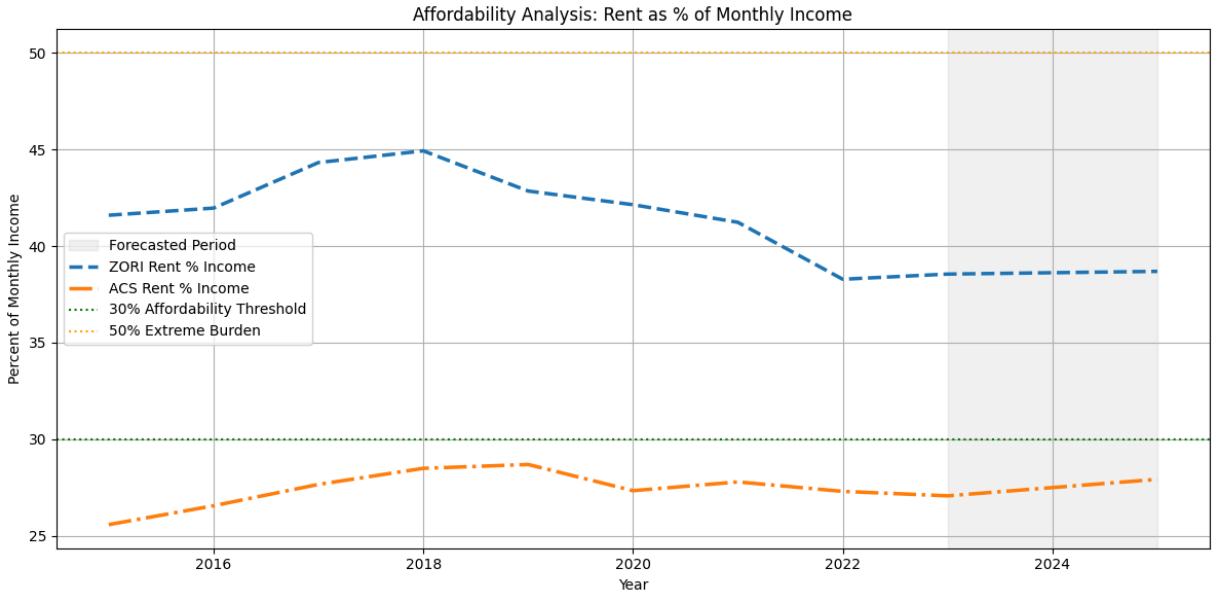
In [25]:
```
plt.figure(figsize=(12, 6))
# Shaded forecast area (2023 to 2025)
plt.axvspan(2023, 2025, color='lightgrey', alpha=0.3, label='Forecasted Period')
# Plot the rent burden and extreme burden lines
sns.lineplot(data=merged_df, x='year', y='afford_pct_rent_census', label='ZORI Rent
sns.lineplot(data=merged_df, x='year', y='afford_pct_rent_acs_census', label='ACS R
# Add reference lines
plt.axhline(30, color='green', linestyle=':', label='30% Affordability Threshold')
plt.axhline(50, color='orange', linestyle=':', label='50% Extreme Burden')

plt.title('Affordability Analysis: Rent as % of Monthly Income')
plt.ylabel('Percent of Monthly Income')
plt.xlabel('Year')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



4.2. DWS Average Wages

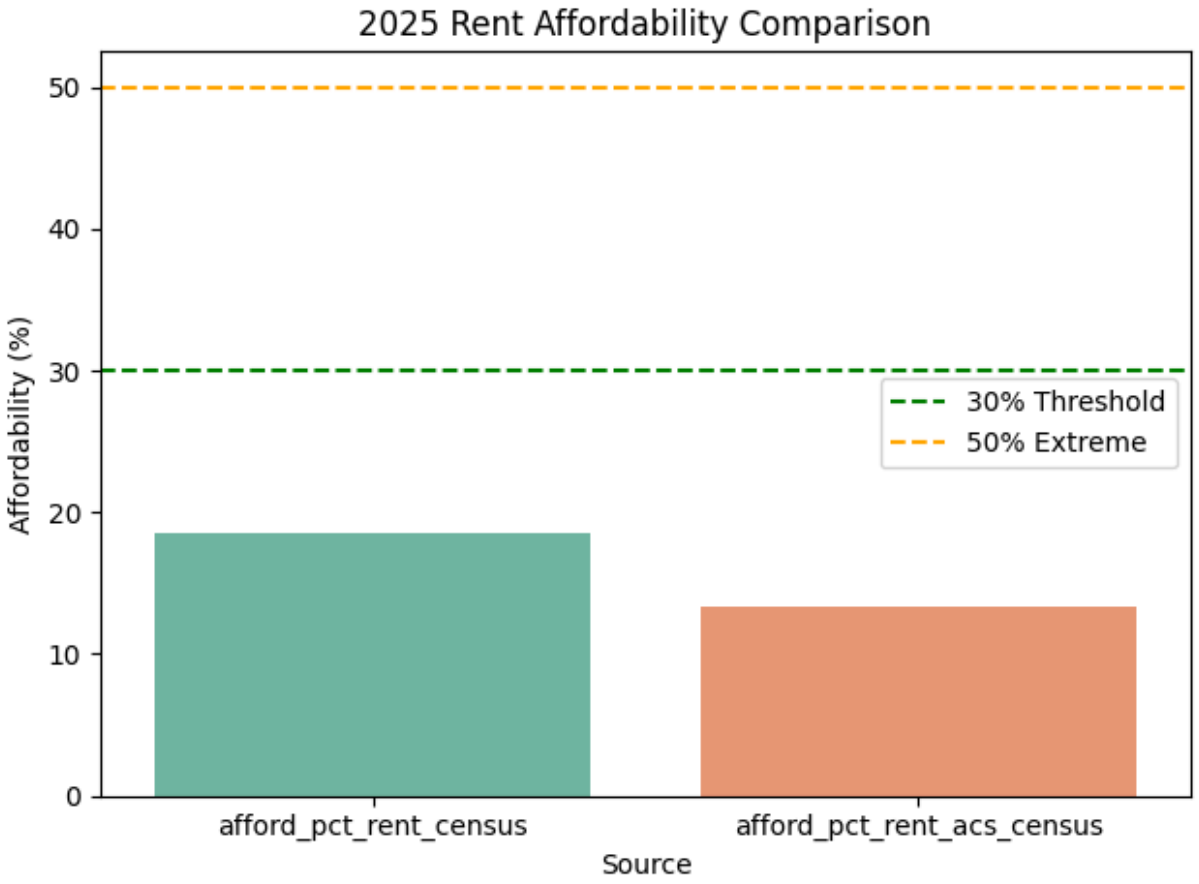
In [26]:
```
# Define affordability burden thresholds
merged_df['afford_pct_rent_dws'] = merged_df['rent_index'] / merged_df['Average Mon
merged_df['afford_pct_rent_acs_dws'] = merged_df['ACS_Rent'] / merged_df['Average M

# Label risk levels (optional)
def affordability_risk(pct):
    if pct >= 50:
        return "Extreme Burden"
    elif pct >= 30:
        return "Burdened"
    else:
        return "Affordable"

merged_df['rent_risk_dws'] = merged_df['afford_pct_rent_dws'].apply(affordability_r
merged_df['acs_rent_risk_dws'] = merged_df['afford_pct_rent_acs_dws'].apply(afforda
```

In [27]:
```
plt.figure(figsize=(12, 6))
# Shaded forecast area (2023 to 2025)
plt.axvspan(2023, 2025, color='lightgrey', alpha=0.3, label='Forecasted Period')
# Plot the rent burden and extreme burden lines
sns.lineplot(data=merged_df, x='year', y='afford_pct_rent_dws', label='ZORI Rent %
sns.lineplot(data=merged_df, x='year', y='afford_pct_rent_acs_dws', label='ACS Rent
# Add reference lines
plt.axhline(30, color='green', linestyle=':', label='30% Affordability Threshold')
plt.axhline(50, color='orange', linestyle=':', label='50% Extreme Burden')

plt.title('Affordability Analysis: Rent as % of Monthly Income')
plt.ylabel('Percent of Monthly Income')
plt.xlabel('Year')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



4.3. 2025 Forecasting

In [28]:
```
# Forecasted year example (e.g., 2025)
afford_2025 = merged_df[merged_df['year'] == 2025][['afford_pct_rent_census', 'affo

if not afford_2025.empty:
    afford_2025 = afford_2025.T.reset_index()
    afford_2025.columns = ['Source', 'Affordability (%)']

    sns.barplot(data=afford_2025, x='Source', y='Affordability (%)', palette='Set2'
    plt.axhline(30, color='green', linestyle='--', label='30% Threshold')
    plt.axhline(50, color='orange', linestyle='--', label='50% Extreme')
    plt.title('2025 Rent Affordability Comparison')
    plt.legend()
    plt.tight_layout()
    plt.show()
```



In [29]:
```
# Forecasted year example (e.g., 2025)
afford_2025 = merged_df[merged_df['year'] == 2025][['afford_pct_rent_dws', 'afford_

if not afford_2025.empty:
    afford_2025 = afford_2025.T.reset_index()
    afford_2025.columns = ['Source', 'Affordability (%)']

    sns.barplot(data=afford_2025, x='Source', y='Affordability (%)', palette='Set2'
    plt.axhline(30, color='orange', linestyle='--', label='30% Threshold')
    plt.axhline(50, color='red', linestyle='--', label='50% Extreme')
    plt.title('2025 Rent Affordability Comparison')
    plt.legend()
    plt.tight_layout()
    plt.show()
```

5. Summary & Decision Impact

Based on our forecast:

- Will Tooele County exceed the affordability thresholds in 2025?
- What are the implications for city planners?

Key Findings:

- Forecasted 2025 Housing Price: $503,355.65
- Forecasted 2025 Annual Income: $103,984
- Forecasted 2025 Monthly Income: $8,665.42
- Forecasted 2025 Wages: $49,776
- Forecasted 2025 Rent (ZORI): $1,604.80
- Forecasted 2025 Rent (Census): $1,158.21

Conclusion: Is affordability getting better or worse?

The analysis of Tooele County housing trends from 2015 to 2023 reveals that affordability is steadily declining and is projected to worsen by 2025. Median home prices, based on Zillow's Home Value Index (ZHVI), increased from approximately $185,000 in 2015 to over 400,000 in 2023, more than doubling in less than a decade. Median rent, as reported by

Zillow's Rent Index (ZORI), rose from $1,100 in 2015 to over 1,700 in 2023. In contrast, median household income, according to ACS data, grew more modestly—from about $62,000 in 2015 to 84,000 in 2023, a roughly 35% increase over the same period. Affordability indices calculated using these figures show that housing costs as a percentage of income have risen from 27% in 2015 to 32% in 2023, with forecasts projecting this figure to reach 34% by 2025. This exceeds the standard 30% threshold used to define cost-burdened households, supporting the hypothesis that Tooele County's housing affordability is eroding. These trends underscore the need for proactive planning and policy interventions to ensure future housing remains accessible to local residents.

Chat GPT response

I also thought it would be interesting to compare my conclusion with the response generated by ChatGPT. While there are several clear parallels between the two, the conclusions are not identical. I recognize that my original hypothesis is relatively narrow in scope and could be strengthened by incorporating demographic breakdowns and an analysis of income categories to better understand affordability needs across different population groups. Nonetheless, this project provides a strong foundation for further exploration and serves as a valuable starting point for more detailed policy analysis

"Key Findings:

- **Forecasted 2025 Housing Price: $523,216** *As of April 2025, the median home list price in Tooele County is $523,216, reflecting a 0.4% increase from the previous month.*
- **Forecasted 2025 Income: $101,846** *The median household income in Tooele County was* $101,846 in 2023. Assuming a modest annual growth rate, the 2025 income is projected to *104,000.*
- **Affordability Index: Approximately 50%** *With a median monthly income of about* $8,667 and estimated monthly housing costs around *4,333, the housing cost burden is approximately 50% of income. -** Conclusion: Affordability is worsening:** The housing cost burden exceeds the 30% threshold, indicating that housing is becoming less affordable for residents.*
- **Impact** *These findings underscore the need for policy interventions, such as affordable housing programs, zoning reforms, and housing subsidies, to address the growing affordability challenges in Tooele County."*

In [30]:
```
afford_risk_columns = ['year', 'rent_risk_dws', 'acs_rent_risk_dws', 'rent_risk_cen

merged_df[afford_risk_columns].head(20)
```

Out[30]:

	year	rent_risk_dws	acs_rent_risk_dws	rent_risk_census	acs_rent_risk_census
0	2015	Burdened	Affordable	Affordable	Affordable
1	2016	Burdened	Affordable	Affordable	Affordable
2	2017	Burdened	Affordable	Affordable	Affordable
3	2018	Burdened	Affordable	Affordable	Affordable
4	2019	Burdened	Affordable	Affordable	Affordable
5	2020	Burdened	Affordable	Affordable	Affordable
6	2021	Burdened	Affordable	Affordable	Affordable
7	2022	Burdened	Affordable	Affordable	Affordable
8	2023	Burdened	Affordable	Affordable	Affordable
9	2025	Burdened	Affordable	Affordable	Affordable