Hector Sanchez - TVHC SKILLS TEST - DATA ANALYSIS

The purpose of this notebook is to provide a clear and easy to follow document that outlines my data analysis, and allows for reproducibility

Load Data and Initial Inspections

In [4]:
```
# IMPORT NECESSARY PACKAGES
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

# LOAD THE DATASET
file_path = 'C:/Users/hecsa/Documents/TVHC SKILLS TEST/Skills_Test_Data_Set_202507_1.xlsx'
df = pd.read_excel(file_path)
```

In [5]:
```
# INSPECT THE STRUCTURE OF THE DATASET
df.head()
```

Out[5]:

	Patient_ID	Site	Age	Sex	Race_Ethnicity	Insurance	BP_Systolic_Pre	BP_Diastolic_Pre	BP_Date_Pre	BP_Systolic_Post	BF
0	2001	Site A	69	M	Black	Medicare	129	92	2023-03-19	121	
1	2002	Site C	41	M	White	Medicaid	135	91	2023-03-24	124	
2	2003	Site A	75	M	White	Uninsured	114	95	2023-03-17	105	
3	2004	Site C	32	M	Hispanic	Commercial	155	89	2023-03-19	143	
4	2005	Site B	68	F	Black	Medicaid	135	100	2023-03-18	131	

In [6]:
```
# CHECK BASIC INFORMATION
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Patient_ID          100 non-null    int64
 1   Site                100 non-null    object
 2   Age                 100 non-null    int64
 3   Sex                 100 non-null    object
 4   Race_Ethnicity      100 non-null    object
 5   Insurance           100 non-null    object
 6   BP_Systolic_Pre     100 non-null    int64
 7   BP_Diastolic_Pre    100 non-null    int64
 8   BP_Date_Pre         100 non-null    datetime64[ns]
 9   BP_Systolic_Post    100 non-null    int64
 10  BP_Diastolic_Post   100 non-null    int64
 11  BP_Date_Post        100 non-null    datetime64[ns]
 12  BP_Controlled_Pre   100 non-null    int64
 13  BP_Controlled_Post  100 non-null    int64
 14  Intervention        100 non-null    object
dtypes: datetime64[ns](2), int64(8), object(5)
memory usage: 11.8+ KB
```

In [7]:
```
# CHECK FOR MISSING VALUES
df.isnull().sum()
```

Out[7]:
```
Patient_ID           0
Site                 0
Age                  0
Sex                  0
Race_Ethnicity       0
Insurance            0
BP_Systolic_Pre      0
BP_Diastolic_Pre     0
BP_Date_Pre          0
BP_Systolic_Post     0
BP_Diastolic_Post    0
BP_Date_Post         0
BP_Controlled_Pre    0
BP_Controlled_Post   0
Intervention         0
dtype: int64
```

In [8]:
```python
# CHECK DATA TYPES AND UNIQUE VALUES
df.describe(include='all')
```

Out[8]:

	Patient_ID	Site	Age	Sex	Race_Ethnicity	Insurance	BP_Systolic_Pre	BP_Diastolic_Pre	BP_Date_Pre	BP_Sys
count	100.000000	100	100.000000	100	100	100	100.000000	100.000000	100	1
unique	NaN	3	NaN	2	5	4	NaN	NaN	NaN	
top	NaN	Site A	NaN	M	Other	Uninsured	NaN	NaN	NaN	
freq	NaN	41	NaN	58	28	30	NaN	NaN	NaN	
mean	2050.500000	NaN	52.660000	NaN	NaN	NaN	139.150000	90.190000	2023-03-15 18:00:00	1
min	2001.000000	NaN	18.000000	NaN	NaN	NaN	114.000000	78.000000	2023-03-01 00:00:00	1
25%	2025.750000	NaN	37.750000	NaN	NaN	NaN	134.000000	87.750000	2023-03-08 00:00:00	1
50%	2050.500000	NaN	53.000000	NaN	NaN	NaN	139.000000	90.000000	2023-03-15 00:00:00	1
75%	2075.250000	NaN	69.000000	NaN	NaN	NaN	145.000000	93.000000	2023-03-25 00:00:00	1
max	2100.000000	NaN	84.000000	NaN	NaN	NaN	161.000000	100.000000	2023-03-30 00:00:00	1
std	29.011492	NaN	18.416736	NaN	NaN	NaN	10.395585	4.670334	NaN	

In [9]:
```python
# CHECK HOW MANY DUPLICATED ROWS THERE ARE
duplicate_count = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_count}")
```

```
Number of duplicate rows: 0
```

In [10]:
```python
# DISPLAY THE ACTUAL DUPLIATE ROWS, IF ANY
duplicates = df[df.duplicated()]
print(duplicates)
```

```
Empty DataFrame
Columns: [Patient_ID, Site, Age, Sex, Race_Ethnicity, Insurance, BP_Systolic_Pre, BP_Diastolic_Pre, BP_Date_Pre, BP_
Systolic_Post, BP_Diastolic_Post, BP_Date_Post, BP_Controlled_Pre, BP_Controlled_Post, Intervention]
Index: []
```

CLEAN AND PREPARE THE DATA

The previous outputs confirm the following about the dataset:

- No missing values
- No duplicated rows
- daatetime columns (BP_Date_Pre, BP_Date_Post) are the correct data type

- Column names are consistent and clear

Next Steps for Cleaning & Preparing the Data:

1. Ensure that Categorical Columns have the correct data type

Make sure that categorical variables are stored as category types in pandas to allow for cleaner analysis and more efficient grouping

In [15]:
```
# CONVERT CATEGORICAL COLUMNS TO 'CATEGORY' DATA TYPE
categorical_cols = ['Site', 'Sex', 'Race_Ethnicity', 'Insurance', 'Intervention']
for col in categorical_cols:
    df[col] = df[col].astype('category')
```

In [16]:
```
# CHECK UPDATED COLUMN DATA TYPES
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Patient_ID          100 non-null    int64
 1   Site                100 non-null    category
 2   Age                 100 non-null    int64
 3   Sex                 100 non-null    category
 4   Race_Ethnicity      100 non-null    category
 5   Insurance           100 non-null    category
 6   BP_Systolic_Pre     100 non-null    int64
 7   BP_Diastolic_Pre    100 non-null    int64
 8   BP_Date_Pre         100 non-null    datetime64[ns]
 9   BP_Systolic_Post    100 non-null    int64
 10  BP_Diastolic_Post   100 non-null    int64
 11  BP_Date_Post        100 non-null    datetime64[ns]
 12  BP_Controlled_Pre   100 non-null    int64
 13  BP_Controlled_Post  100 non-null    int64
 14  Intervention        100 non-null    category
dtypes: category(5), datetime64[ns](2), int64(8)
memory usage: 9.3 KB
```

2. Create a new BP Control Status Column (Post-Intervention)

Although we already have a **'BP_Controlled_Post'** column, I'm creating a fresh derived column, **'BP_Controlled_calculated'**, to verify and ensure data integrity, based on the definition: **Controlled = Systolic < 140 and Diastolic < 90**

In [18]:
```
# CREATE NEW 'BP_Controlled_Calculated' COLUMN
df['BP_Controlled_Calculated'] = ((df['BP_Systolic_Post'] < 140) & (df['BP_Diastolic_Post'] < 90)).astype(int)
```

In [19]:
```
# INSPECT DATASET WITH THE ADDITION OF 'BP_Controlled_Calculated'
df.head()
```

Out[19]:

	Patient_ID	Site	Age	Sex	Race_Ethnicity	Insurance	BP_Systolic_Pre	BP_Diastolic_Pre	BP_Date_Pre	BP_Systolic_Post	BP
0	2001	Site A	69	M	Black	Medicare	129	92	2023-03-19	121	
1	2002	Site C	41	M	White	Medicaid	135	91	2023-03-24	124	
2	2003	Site A	75	M	White	Uninsured	114	95	2023-03-17	105	
3	2004	Site C	32	M	Hispanic	Commercial	155	89	2023-03-19	143	
4	2005	Site B	68	F	Black	Medicaid	135	100	2023-03-18	131	

In [20]:
```
# COMPARE NEW COLUMN WITH THE ORIGINAL COLUMN
```

```python
bp_check = (df['BP_Controlled_Calculated'] == df['BP_Controlled_Post']).value_counts()
print(bp_check)
```

```
True     75
False    25
Name: count, dtype: int64
```

In order to verify data quality, I compared the calculated BP control values against the original recorded values. I found that they matched in only 75% of cases, meaning that for 25% of patients, there is a mismatch between what the data says and what the BP control status actually appears to be based on standard definitions.

This inconsistency raises a flag about possible data entry errors or logic flaws in how BP control was assessed in the original dataset. If this is not corrected, the discrepancies could lead to patients being misclassified.

Misclassifying patients could potentially affect the interventions they receive. For example, a patient whose BP is not truly controlled may be recorded as 'controlled' and therefore may not receive needed support such as medication adjustment or coaching.

In [22]:
```python
# EVALUATE HOW THE 25 MISMATCHES ARE DISTRIBUTED

# 1. ISOLATE MISMATCHES VIA A BOOLEAN MASK
mismatched_mask = df['BP_Controlled_Calculated'] != df['BP_Controlled_Post']

# 2. CREATE A NEW DATAFRAME WITH ONLY MISMATCHED ROWS
mismatched_df = df[mismatched_mask]

# 3. CHECK HOW MISMATCHES ARE DISTRIBUTED BY INTERVENTION
intervention_mismatches = mismatched_df['Intervention'].value_counts()
print("Mismatch Count by Intervention:")
print(intervention_mismatches)

# 4. CHECK HOW MISMATCHES ARE DISTRIBUTED BY SIZE
site_mismatches = mismatched_df['Site'].value_counts()
print("\nMismatch Count by Site:")
print(site_mismatches)

# 5. CROSSTAB TO SEE MISTACH DISTRIBUTION ACROSS BOTH INTERVENTION AND SITE
intervention_site_ct = pd.crosstab(mismatched_df['Site'], mismatched_df['Intervention'])
print("\nMismatch Distribution by Site and Intervention:")
print(intervention_site_ct)
```

```
Mismatch Count by Intervention:
Intervention
Care Team Outreach          6
Medication Adjustment       6
Health Coaching             5
Clinical Pharmacy Program   4
Home BP Monitoring          4
Name: count, dtype: int64

Mismatch Count by Site:
Site
Site A    14
Site C     6
Site B     5
Name: count, dtype: int64

Mismatch Distribution by Site and Intervention:
Intervention  Care Team Outreach  Clinical Pharmacy Program  Health Coaching  \
Site
Site A                         3                          1                3
Site B                         1                          1                1
Site C                         2                          2                1

Intervention  Home BP Monitoring  Medication Adjustment
Site
Site A                         3                      4
Site B                         1                      1
Site C                         0                      1
```

3. Calculate Delta in BP Pre vs. Post AND Calculate Percentage Change

Creating **Delta** variables will help me clearly demonstrate improvement per patient. This tells us the **raw difference in mmHg**("How much did the BP drop?")

Calculating **Percentage Change** tells us **how much did BP drop relative to their starting point**("Was it a big improvement relative to where they started?)

In [24]:
```
# CALCULATE THE DELTA(ABSOLUTE CHANGE)
df['Delta_Systolic'] = df['BP_Systolic_Pre'] - df['BP_Systolic_Post']
df['Delta_Diastolic'] = df['BP_Diastolic_Pre'] - df['BP_Diastolic_Post']
```

In [25]:
```
# INSPECT A FEW ROWS TO VERIFY
df[['BP_Systolic_Pre', 'BP_Systolic_Post', 'Delta_Systolic',
    'BP_Diastolic_Pre', 'BP_Diastolic_Post', 'Delta_Diastolic']].head()
```

Out[25]:

	BP_Systolic_Pre	BP_Systolic_Post	Delta_Systolic	BP_Diastolic_Pre	BP_Diastolic_Post	Delta_Diastolic
0	129	121	8	92	91	1
1	135	124	11	91	86	5
2	114	105	9	95	90	5
3	155	143	12	89	86	3
4	135	131	4	100	101	-1

In [26]:
```
# CALCULATE THE PERCENTAGE CHANGE

# AVOID DIVIDING BY ZERO
df = df[df['BP_Systolic_Pre'] != 0]
df = df[df['BP_Diastolic_Pre'] != 0]

# PERCENT CHANGE
df['Pct_Change_Systolic'] = ((df['BP_Systolic_Pre'] - df['BP_Systolic_Post']) / df['BP_Systolic_Pre']) * 100
df['Pct_Change_Diastolic'] = ((df['BP_Diastolic_Pre'] - df['BP_Diastolic_Post']) / df['BP_Diastolic_Pre']) * 100

# ROUND FOR CLEAN PRESENTATION
df['Pct_Change_Systolic'] = df['Pct_Change_Systolic'].round(2)
df['Pct_Change_Diastolic'] = df['Pct_Change_Diastolic'].round(2)
```

In [27]:
```
# INSPECT A FEW ROWS TO VERIFY
df[['BP_Systolic_Pre', 'BP_Systolic_Post', 'Pct_Change_Systolic',
    'BP_Diastolic_Pre', 'BP_Diastolic_Post', 'Pct_Change_Diastolic']].head()
```

Out[27]:

	BP_Systolic_Pre	BP_Systolic_Post	Pct_Change_Systolic	BP_Diastolic_Pre	BP_Diastolic_Post	Pct_Change_Diastolic
0	129	121	6.20	92	91	1.09
1	135	124	8.15	91	86	5.49
2	114	105	7.89	95	90	5.26
3	155	143	7.74	89	86	3.37
4	135	131	2.96	100	101	-1.00

Descriptive Statistics and Demographic Trends

Calculate Percentage of Patients Achieving BP Control

Compute the percentage of patients who achieved **BP control post-intervention** where:

- Systolic < 140
- Diastolic < 90

In [30]:
```
# CALCULATE PERCENT OF BP CONTROLLED PATIENTS
pct_controlled = df['BP_Controlled_Calculated'].mean() * 100
print(f"Percentage of patients with BP controlled post-intervention: {pct_controlled: .2f}%")
```

```
Percentage of patients with BP controlled post-intervention:  60.00%
```

In [31]:
```
# CALL .value_counts() TO CHECK ACCURACY OF PERCENT CALCULATION
df['BP_Controlled_Calculated'].value_counts()
```

Out[31]:
```
BP_Controlled_Calculated
1    60
0    40
Name: count, dtype: int64
```

In [32]:
```
# CALL .info() TO INSPECT THE NEW COLUMN DATA TYPES BEFORE PROCEEDING.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 20 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Patient_ID                100 non-null    int64
 1   Site                      100 non-null    category
 2   Age                       100 non-null    int64
 3   Sex                       100 non-null    category
 4   Race_Ethnicity            100 non-null    category
 5   Insurance                 100 non-null    category
 6   BP_Systolic_Pre           100 non-null    int64
 7   BP_Diastolic_Pre          100 non-null    int64
 8   BP_Date_Pre               100 non-null    datetime64[ns]
 9   BP_Systolic_Post          100 non-null    int64
 10  BP_Diastolic_Post         100 non-null    int64
 11  BP_Date_Post              100 non-null    datetime64[ns]
 12  BP_Controlled_Pre         100 non-null    int64
 13  BP_Controlled_Post        100 non-null    int64
 14  Intervention              100 non-null    category
 15  BP_Controlled_Calculated  100 non-null    int32
 16  Delta_Systolic            100 non-null    int64
 17  Delta_Diastolic           100 non-null    int64
 18  Pct_Change_Systolic       100 non-null    float64
 19  Pct_Change_Diastolic      100 non-null    float64
dtypes: category(5), datetime64[ns](2), float64(2), int32(1), int64(10)
memory usage: 12.8 KB
```

Group by Demographics or Provider Fields, then Compare Outcomes

Bucket Age Groups to help analyze trends by Age

In [35]:
```
# BUCKET AGE GROUPS
bins = [0, 29, 44, 59, 74, 120]
labels = ['<30', '30-44', '45-59', '60-74', '75+']
df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels)
```

In [36]:
```
# LIST OF DEMOGRAPHIC AND PROVIDER-RELATED VARIABLES
group_columns = [
    'Site',             # Provider-related
    'Sex',              # Demographic
    'Race_Ethnicity',   # Demographic
    'Insurance',        # Demographic
    'Intervention',     # Provider-related (or treatment-level)
    'Age_Group'         # Demographic (bucketed age)
]

# LOOP THROUGH EACH COLUMN AND PRINT % CONTROLLED
for col in group_columns:
    print(f"\n--- BP Control Rate by {col} ---")
    grouped_pct = df.groupby(col)['BP_Controlled_Calculated'].mean() * 100
    for group, pct in grouped_pct.items():
        print(f"{col} = {group}: {pct:.2f}%")
```

```
--- BP Control Rate by Site ---
Site = Site A: 70.73%
Site = Site B: 53.33%
Site = Site C: 51.72%

--- BP Control Rate by Sex ---
Sex = F: 54.76%
Sex = M: 63.79%

--- BP Control Rate by Race_Ethnicity ---
Race_Ethnicity = Asian: 56.25%
Race_Ethnicity = Black: 50.00%
Race_Ethnicity = Hispanic: 64.29%
Race_Ethnicity = Other: 67.86%
Race_Ethnicity = White: 60.00%

--- BP Control Rate by Insurance ---
Insurance = Commercial: 63.33%
Insurance = Medicaid: 63.64%
Insurance = Medicare: 55.56%
Insurance = Uninsured: 56.67%

--- BP Control Rate by Intervention ---
Intervention = Care Team Outreach: 30.77%
Intervention = Clinical Pharmacy Program: 81.82%
Intervention = Health Coaching: 76.92%
Intervention = Home BP Monitoring: 40.00%
Intervention = Medication Adjustment: 75.00%

--- BP Control Rate by Age_Group ---
Age_Group = <30: 71.43%
Age_Group = 30-44: 73.91%
Age_Group = 45-59: 54.55%
Age_Group = 60-74: 48.28%
Age_Group = 75+: 58.33%
```

In [37]:

```
# LOOP THROUGH AND PRINT CROSSTABS FOR EACH COLUMN
for col in group_columns:
    print(f"\n--- BP Control % by {col} ---")
    ctab = pd.crosstab(df[col], df['BP_Controlled_Calculated'], normalize='index') * 100
    ctab.columns = ['Not Controlled %', 'Controlled %']  # Rename columns for clarity
    display(ctab)  # Use display() in Jupyter to show tables
```

```
--- BP Control % by Site ---
```

Site	Not Controlled %	Controlled %
Site A	29.268293	70.731707
Site B	46.666667	53.333333
Site C	48.275862	51.724138

```
--- BP Control % by Sex ---
```

Sex	Not Controlled %	Controlled %
F	45.238095	54.761905
M	36.206897	63.793103

```
--- BP Control % by Race_Ethnicity ---
```

	Not Controlled %	Controlled %
Race_Ethnicity		
Asian	43.750000	56.250000
Black	50.000000	50.000000
Hispanic	35.714286	64.285714
Other	32.142857	67.857143
White	40.000000	60.000000

```
--- BP Control % by Insurance ---
```

	Not Controlled %	Controlled %
Insurance		
Commercial	36.666667	63.333333
Medicaid	36.363636	63.636364
Medicare	44.444444	55.555556
Uninsured	43.333333	56.666667

```
--- BP Control % by Intervention ---
```

	Not Controlled %	Controlled %
Intervention		
Care Team Outreach	69.230769	30.769231
Clinical Pharmacy Program	18.181818	81.818182
Health Coaching	23.076923	76.923077
Home BP Monitoring	60.000000	40.000000
Medication Adjustment	25.000000	75.000000

```
--- BP Control % by Age_Group ---
```

	Not Controlled %	Controlled %
Age_Group		
<30	28.571429	71.428571
30-44	26.086957	73.913043
45-59	45.454545	54.545455
60-74	51.724138	48.275862
75+	41.666667	58.333333

Intervention Effectiveness

In [39]:

```
# COLUMNS TO GROUP BY
group_vars = ['Site', 'Race_Ethnicity', 'Sex', 'Insurance', 'Age_Group']

# FOR EACH VARIABLE, COMPARE BP CONTROL RATES BY INTERVENTION
for var in group_vars:
    print(f"\n--- {var}: BP Control % by Intervention ---")
    subgroup_result = df.groupby([var, 'Intervention'])['BP_Controlled_Calculated'].mean().unstack().round(3) * 100
    display(subgroup_result)
```

```
--- Site: BP Control % by Intervention ---
```

Intervention	Care Team Outreach	Clinical Pharmacy Program	Health Coaching	Home BP Monitoring	Medication Adjustment
Site					
Site A	50.0	87.5	80.0	62.5	75.0
Site B	20.0	100.0	75.0	20.0	85.7
Site C	25.0	70.0	75.0	0.0	60.0

```
--- Race_Ethnicity: BP Control % by Intervention ---
```

Intervention	Care Team Outreach	Clinical Pharmacy Program	Health Coaching	Home BP Monitoring	Medication Adjustment
Race_Ethnicity					
Asian	50.0	66.7	66.7	0.0	80.0
Black	0.0	100.0	80.0	100.0	50.0
Hispanic	33.3	85.7	NaN	0.0	66.7
Other	57.1	83.3	100.0	33.3	85.7
White	28.6	66.7	66.7	75.0	100.0

```
--- Sex: BP Control % by Intervention ---
```

Intervention	Care Team Outreach	Clinical Pharmacy Program	Health Coaching	Home BP Monitoring	Medication Adjustment
Sex					
F	20.0	72.7	80.0	57.1	100.0
M	45.5	90.9	75.0	25.0	70.0

```
--- Insurance: BP Control % by Intervention ---
```

Intervention	Care Team Outreach	Clinical Pharmacy Program	Health Coaching	Home BP Monitoring	Medication Adjustment
Insurance					
Commercial	42.9	87.5	66.7	25.0	80.0
Medicaid	33.3	100.0	100.0	50.0	100.0
Medicare	0.0	66.7	100.0	100.0	60.0
Uninsured	33.3	83.3	66.7	33.3	66.7

```
--- Age_Group: BP Control % by Intervention ---
```

Intervention	Care Team Outreach	Clinical Pharmacy Program	Health Coaching	Home BP Monitoring	Medication Adjustment
Age_Group					
<30	33.3	80.0	100.0	NaN	80.0
30-44	0.0	66.7	100.0	66.7	90.0
45-59	44.4	66.7	50.0	66.7	66.7
60-74	22.2	100.0	66.7	16.7	60.0
75+	33.3	100.0	100.0	33.3	0.0

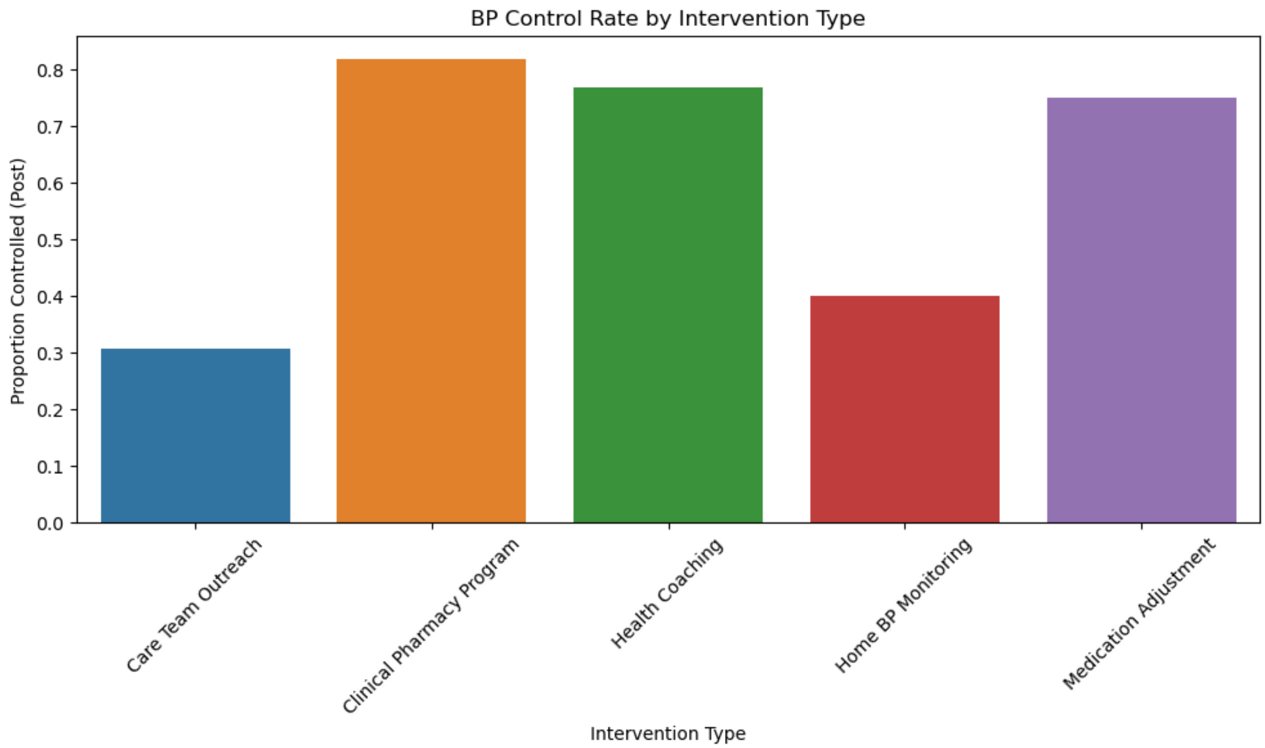
Visualization

In [41]:
```
# IMPORT VISUALIZATION PACKAGES
import seaborn as sns
import matplotlib.pyplot as plt
```

Bar Plot - Effectiveness of Interventions

In [43]:
```
# AGGREGATE BP CONTROL RATE BY INTERVENTION
intervention_effectiveness = df.groupby('Intervention')['BP_Controlled_Calculated'].mean().reset_index()
```

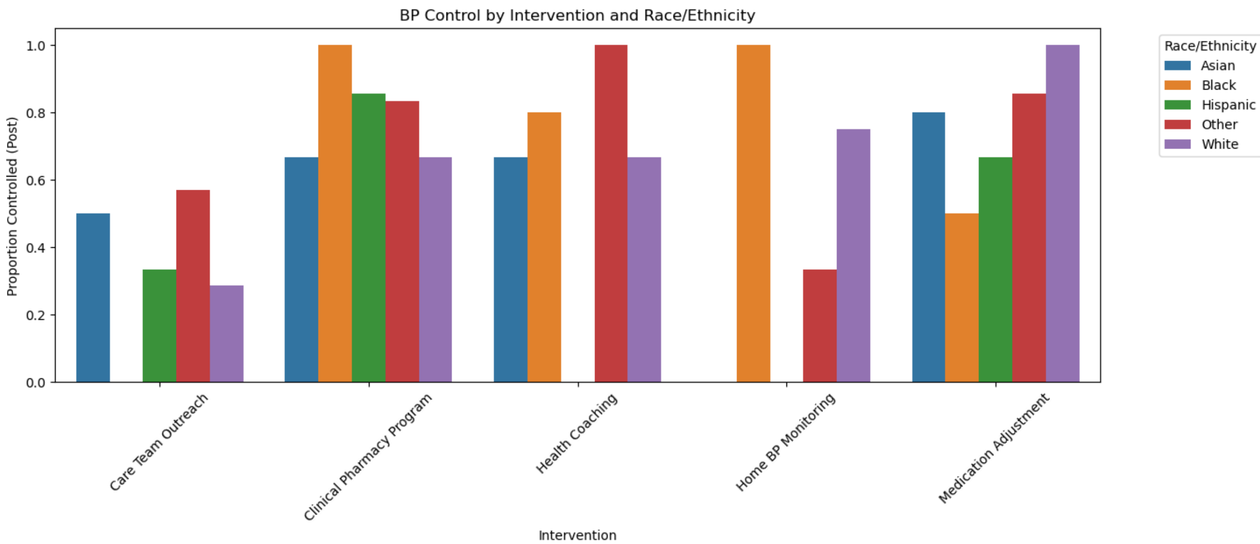
```
plt.figure(figsize=(10,6))
sns.barplot(data=intervention_effectiveness, x='Intervention', y='BP_Controlled_Calculated')
plt.title('BP Control Rate by Intervention Type')
plt.ylabel('Proportion Controlled (Post)')
plt.xlabel('Intervention Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



This bar chart shows the average blood pressue control rate for each type of intervention. We can immediately see which intervention was the most effective overall in improving BP control across all patients. For example, since 'Clinical Pharmacy Program' has the highest bar, it suggests that patients who received this program were more likely to achieve BP control compared to those who did not.

Grouped Bar Plot - Intervention Effectiveness by Race_Ethnicity

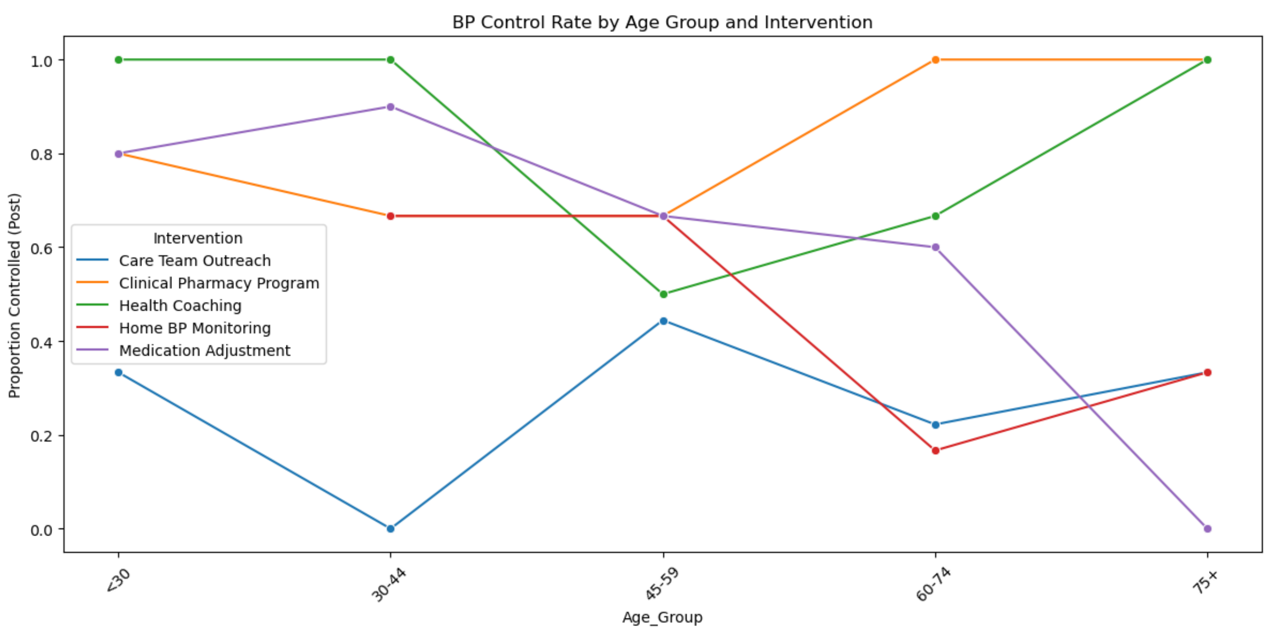
In [46]:
```
# CREATE A GRROUPED DATAFRAME
grouped = df.groupby(['Intervention', 'Race_Ethnicity'])['BP_Controlled_Calculated'].mean().reset_index()

plt.figure(figsize=(12,6))
sns.barplot(data=grouped, x='Intervention', y='BP_Controlled_Calculated', hue='Race_Ethnicity')
plt.title('BP Control by Intervention and Race/Ethnicity')
plt.ylabel('Proportion Controlled (Post)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend(title='Race/Ethnicity', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



Here, we've broken down the intervention outcomes by race and ethnicity. This grouped bar plot helps us identify whether certain interventions were more or less effective depending on the patient's racial or ethnic background. This is important for highlighting disparities in health outcomes and tailoring future interventions to be more inclusive and equitable.

Line Plot - BP Control Rate by Age Group per Intervention

In [49]:
```
age_grouped = df.groupby(['Age_Group', 'Intervention'])['BP_Controlled_Calculated'].mean().reset_index()

plt.figure(figsize=(12,6))
sns.lineplot(data=age_grouped, x='Age_Group', y='BP_Controlled_Calculated', hue='Intervention', marker='o')
plt.title('BP Control Rate by Age Group and Intervention')
plt.ylabel('Proportion Controlled (Post)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

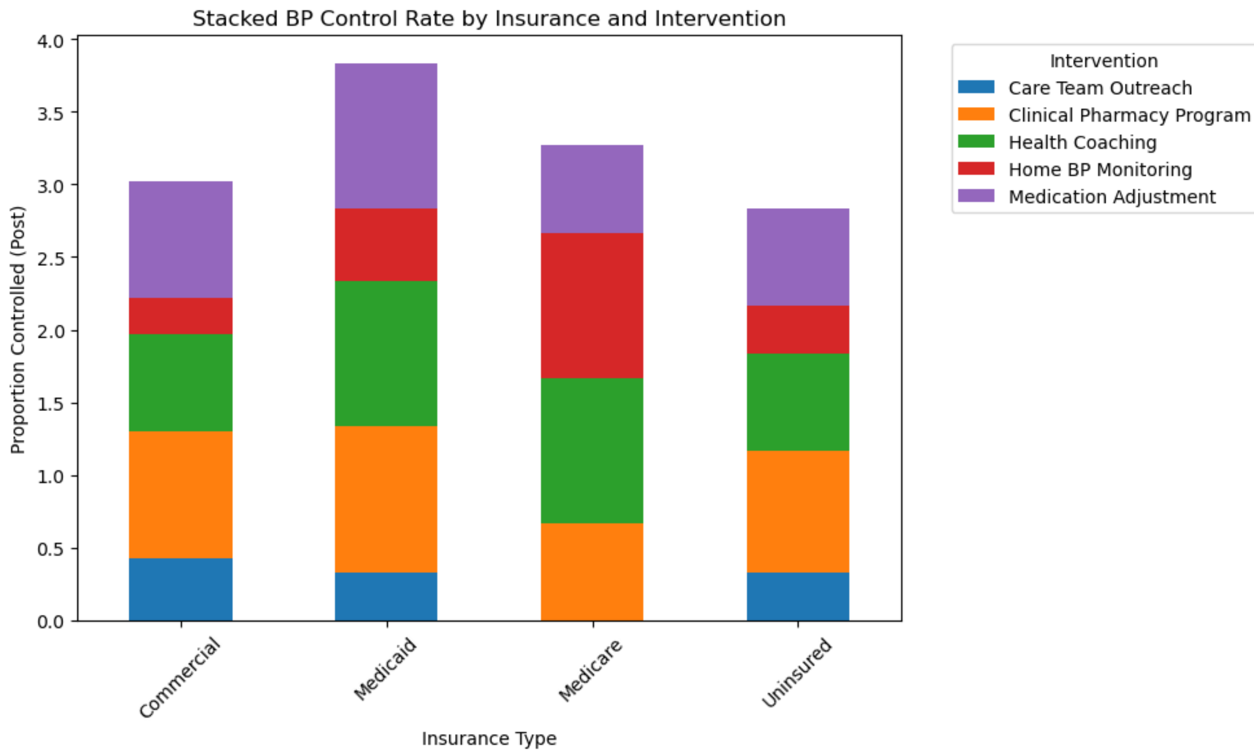


This line chart tracks how each intervention performs across different age groups. Each line represents one intervention, and the points show the BP control rate by age category. This helps us understand which interventions work better for younger versus older patients, potentially guiding age targeted strategies.

Stacked Bar Plot - BP Control by Insurance Type and Intervention

In [52]:
```
# CALCULATE BP CONTROL RATES
pivot_df = df.groupby(['Insurance', 'Intervention'])['BP_Controlled_Calculated'].mean().unstack()
```

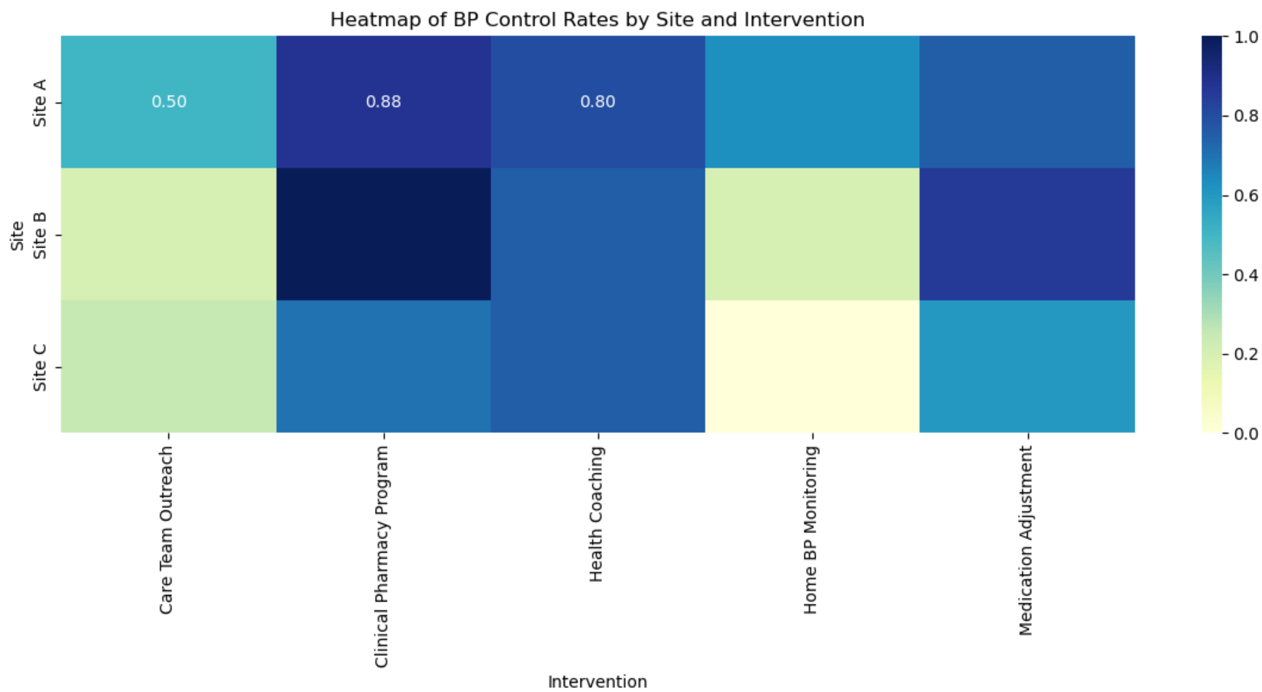
```
pivot_df.plot(kind='bar', stacked=True, figsize=(10,6))
plt.title('Stacked BP Control Rate by Insurance and Intervention')
plt.ylabel('Proportion Controlled (Post)')
plt.xlabel('Insurance Type')
plt.xticks(rotation=45)
plt.legend(title='Intervention', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



This stacked bar chart compares how different insurance groups responded to each intervention. For example, we can see how patients with private insurance versus those on Medicaid or uninsured experienced BP improvements under various approaches. This is especially useful for identifying whether access to resources plays a role in intervention success.

Heatmap - BP Control by Site and Intervention

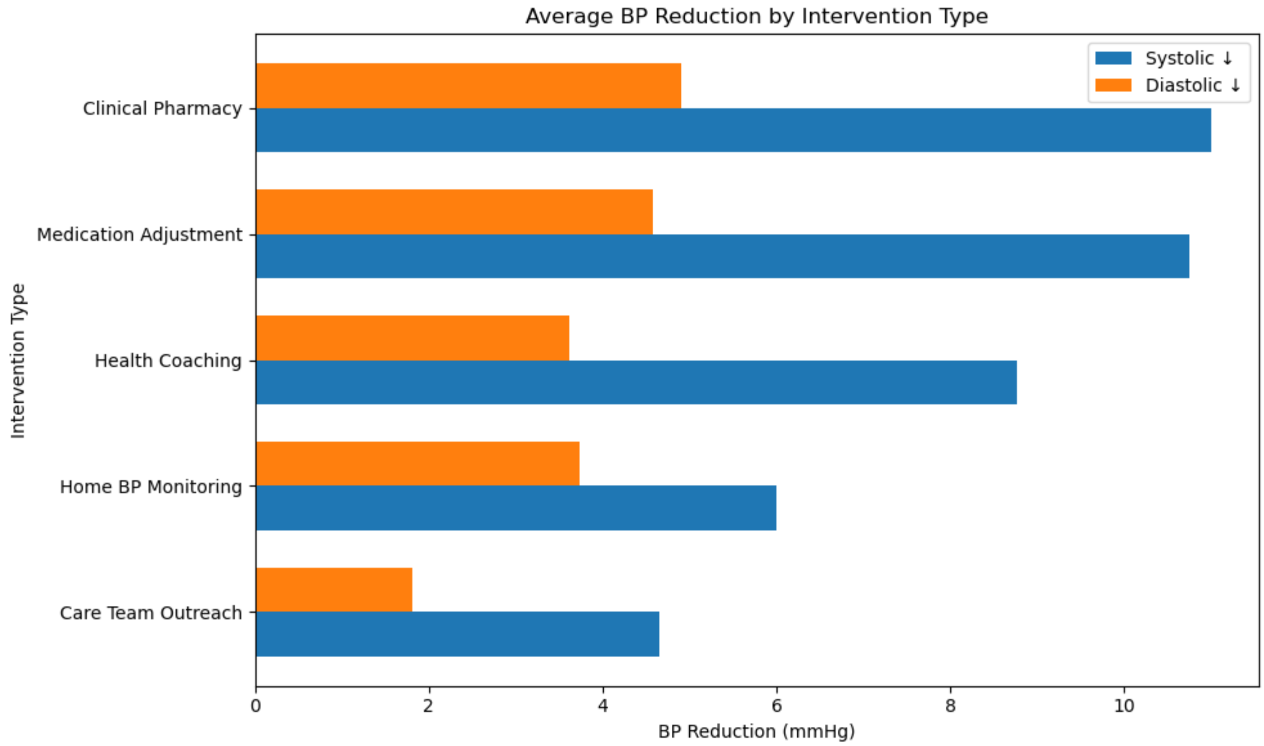
In [55]:
```
site_heatmap = df.groupby(['Site', 'Intervention'])['BP_Controlled_Calculated'].mean().unstack()

plt.figure(figsize=(12,6))
sns.heatmap(site_heatmap, annot=True, fmt=".2f", cmap='YlGnBu')
plt.title('Heatmap of BP Control Rates by Site and Intervention')
plt.ylabel('Site')
plt.xlabel('Intervention')
plt.tight_layout()
plt.show()
```



This heatmap gives us a high level view of how each site performed under different interventions. Darker colors represent higher BP control rates. This visualization helps us identify which sites are excelling or struggling with specific interventions, and can inform operational or training improvements.

Grouped Bar Chart - Average BP Reduction by Intervention Type

In [58]:
```
# CREATE DATA
data = {
    'Intervention': [
        'Clinical Pharmacy', 'Medication Adjustment', 'Health Coaching',
        'Home BP Monitoring', 'Care Team Outreach'
    ],
    'Systolic (mmHg)': [11.00, 10.75, 8.77, 6.00, 4.65],
    'Diastolic (mmHg)': [4.91, 4.58, 3.62, 3.73, 1.81]
}

df_viz = pd.DataFrame(data)

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

bar_height = 0.35
y = range(len(df_viz))

ax.barh([i + bar_height for i in y], df_viz['Systolic (mmHg)'], height=bar_height, label='Systolic ↓', align='cente
ax.barh(y, df_viz['Diastolic (mmHg)'], height=bar_height, label='Diastolic ↓', align='center')

# LABELING
ax.set_ylabel('Intervention Type')
ax.set_xlabel('BP Reduction (mmHg)')
ax.set_title('Average BP Reduction by Intervention Type')
ax.set_yticks([i + bar_height / 2 for i in y])
ax.set_yticklabels(df_viz['Intervention'])
ax.invert_yaxis()  # Optional: puts top intervention at top
ax.legend()

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



This chart shows the **average reduction in blood pressure** by intervention type. For each intervention, we've broken down the results into **Systolic** and **Diastolic** reductions. The longer the bar, the greater the improvement

Final Thoughts

Summary Metrics

In [62]:
```
# OVERALL BP CONTROL RATE
overall_bp_control_rate = df['BP_Controlled_Calculated'].mean()
print(f"Overall BP Control Rate: {overall_bp_control_rate:.2%}")

# BP CONTROL RATE BY INTERVENTION
bp_by_intervention = df.groupby('Intervention')['BP_Controlled_Calculated'].mean().sort_values(ascending=False) * 1
bp_by_intervention = bp_by_intervention.round(2)
print("\nBP Control Rate by Intervention (%):\n", bp_by_intervention.astype(str) + '%')

# MATCH RATE BETWEEN CALCUALTED AND RECORDED BP CONTROL
bp_check_counts = (df['BP_Controlled_Calculated'] == df['BP_Controlled_Post']).value_counts(normalize=True) * 100
bp_check_counts = bp_check_counts.round(2)
print("\nMatch Rate between Calculated and Recorded BP Control (%):\n", bp_check_counts.astype(str) + '%')

# MEAN DELTA AND PERCENT CHANGE FOR SYSTOLIC BP
mean_delta_sys = df['Delta_Systolic'].mean()
mean_pct_change_sys = df['Pct_Change_Systolic'].mean()
print(f"\nMean Delta Systolic BP (Post - Pre): {mean_delta_sys:.2f}")
print(f"Mean Percent Change in Systolic BP: {mean_pct_change_sys:.2f}%")

# MEAN DELTA AND PERCENT CHANGE FOR DIASTOLIC BP
mean_delta_dia = df['Delta_Diastolic'].mean()
mean_pct_change_dia = df['Pct_Change_Diastolic'].mean()
print(f"\nMean Delta Diastolic BP (Post - Pre): {mean_delta_dia:.2f}")
print(f"Mean Percent Change in Diastolic BP: {mean_pct_change_dia:.2f}%")

# BP DELTA AND PERCENT CHANGE BY INTERVENTION
delta_by_intervention = df.groupby('Intervention')[['Delta_Systolic', 'Delta_Diastolic',
                                                    'Pct_Change_Systolic', 'Pct_Change_Diastolic']].mean()

# ROUND PERCENT CHANGE COLUMNS AND SHOW AS PERCENTAGE
delta_by_intervention['Pct_Change_Systolic'] = (delta_by_intervention['Pct_Change_Systolic']).round(2).astype(str)
delta_by_intervention['Pct_Change_Diastolic'] = (delta_by_intervention['Pct_Change_Diastolic']).round(2).astype(str
```

```python
# ROUND DELTAS AS WELL
delta_by_intervention['Delta_Systolic'] = delta_by_intervention['Delta_Systolic'].round(2)
delta_by_intervention['Delta_Diastolic'] = delta_by_intervention['Delta_Diastolic'].round(2)

print("\nAverage BP Delta and Percent Change by Intervention:\n", delta_by_intervention)
```

```
Overall BP Control Rate: 60.00%

BP Control Rate by Intervention (%):
 Intervention
Clinical Pharmacy Program    81.82%
Health Coaching              76.92%
Medication Adjustment         75.0%
Home BP Monitoring            40.0%
Care Team Outreach           30.77%
Name: BP_Controlled_Calculated, dtype: object

Match Rate between Calculated and Recorded BP Control (%):
 True     75.0%
False    25.0%
Name: proportion, dtype: object

Mean Delta Systolic BP (Post - Pre): 8.25
Mean Percent Change in Systolic BP: 5.99%

Mean Delta Diastolic BP (Post - Pre): 3.68
Mean Percent Change in Diastolic BP: 4.11%

Average BP Delta and Percent Change by Intervention:
                            Delta_Systolic  Delta_Diastolic  \
Intervention
Care Team Outreach                   4.65             1.81
Clinical Pharmacy Program           11.00             4.91
Health Coaching                      8.77             3.62
Home BP Monitoring                   6.00             3.73
Medication Adjustment               10.75             4.58

                          Pct_Change_Systolic Pct_Change_Diastolic
Intervention
Care Team Outreach                      3.36%                2.01%
Clinical Pharmacy Program               8.06%                5.52%
Health Coaching                         6.26%                 4.0%
Home BP Monitoring                      4.34%                4.18%
Medication Adjustment                   7.81%                5.11%
```

Key Findings

1. Discrepancies in Reported BP Control

- **25% of paitents** were misclassified IN the recorded **BP_Controlled_Post** variable. This meant they **their BP control status was incorrectly reported, which could prevent patients from receiving much needed interventions.**
- This discrepancy indicates that **1 in 4 patients** may not have received appropriate follow-up or interventions due to incorrect status reporting.
- **Mismatch rates varied by site**(ex. **Site A accounted for 56%** of all mismatches) and were also present across all interventions types.

2. Effectiveness of Interventions Varies Greatly

- **Clinical Pharmacy Program** had the highest BP Control rate (**81.82%**) and the largest average systolic and diastolic BP reductions.
- **Health Coaching** and **Medication Adjustment** also performed well, with control rates of **76.92%** and **75.00%,** respectively.
- Interventions like **Care Team Outreach (30.77%)** and **Home BP Monitoring(40%)** were considerably less effective.
- This suggests that **more intensive or personalized interventions may yield better outcomes**

3. Site Level Variation Suggest Opportunity for ReTraining or Operational Refinement

- BP Control Rates differed by site:
- **Site A:** 70.73%

- **Site B:** 53.33%
- **Site C:** 51.72%
- These differences, along with mismatch variation by site, suggest opportunites for **targeted process improvements or staff trainings**

4. Demographic Disparities Exist in BP Control

- **Sex:** Males had higher BP control (63.79%) than females (54.76%)
- **Race/Ethnicity:** BP Control was lowest among **Black patients (50.00%)** and highest among **Hispnnic (64.29%)** and **Other (67.86%)** groups.
- **Insurance:** Patients on **Medicare (55.56%)** and the **uninsured (56.67%)** had lower control rates than those with **Medicaid (63.64%** or **Commercial insurance (63.33%)**
- **Age:** BP Control declined with age, dropping from **~74% in 30-44 group** to **~48% in 60-74 group**

5. Overall BP Improvements Achieved, But Modest

- **60.00% of patients** had controlled BP post-intervention
- This suggests progress, but **40% remain uncontrolled**, indicating **room for improvement** in both patient engagement and intervention delivery.
- Average **systolic BP reduction** post-intervention: **8.25 mmHg**
- Average **diastolic BP reduction: 3.68 mmHg**
- Mean **percent decrease** in systolic and diastolic pressure was **6% and 4%,** respectively which is a meanfingul but modest improvement.

Recommendations

1. Improve Data Quality and Accuracy

- **Why:** There is a large discrepancy between the reported **BP_Controlled_Post** variables and the actual BP values after intervention.
- **Action:** Review and standardize the criteria used to classify BP control across all sites to ensure data reliability and consistency in reporting

2. Expand Access to Effective Interventions

- **Why:** Certain interventions (ex: Medication Adjustment and Health Coaching) showed significantly better improvements in systolic and diastolic BP.
- **Action:** Expand the use of higher-performing interventions across all patients groups, with special focus on patients who did **not** receive any targeted follow up post-visit.

3. Target Outreach to At-Risk Populations

- **Why:** Some patient subgroups had significantly lower BP control rates which suggests uneven outcomes.
- **Acton:** Identify high risk segments (by Sex, Race/Ethnicity, Insurance, and Age) and tailor intervention strategies to address their specific barriers

4. Standardie Best Practices Across Sites

- **Why:** Some sites consistently performed better in reducing BP or applying effective interventions, while others lagged behind.
- **Action::** Investigate workflows, staffing, and follow-up processes at top-performing sites and apply learnings to underperforming locations.

5. Improve Reporting with Clearer Metrics

- **Why:** Tracking average changes in BP offers more nuance than binary "controlled/uncontrolled" outcomes and helps highlight meaningful improvements.
- **Action:** Integrate mean BP change and percent improvement metrics into clinical dashboards to support data-driven decision making.