1 Business Understanding

1.1 Project Overview

Analysis of Vaccination Patterns from the National 2009 H1N1 Flu Survey

1.2 Business problem

A vaccine for the H1N1 flu virus became publicly available in October 2009. In late 2009 and early 2010, the United States conducted the National 2009 H1N1 Flu Survey. This phone survey asked respondents whether they had received the H1N1 and seasonal flu vaccines, in conjunction with questions about themselves. These additional questions covered their social, economic, and demographic background, opinions on risks of illness and vaccine effectiveness, and behaviors towards mitigating transmission. A better understanding of how these characteristics are associated with personal vaccination patterns can provide guidance for future public health efforts.

1.3 Project objectives:

1.4 Main Objective

To analyze the demographic characteristics of respondents, including age, education, income, employment, and household composition.

1.5 Specific Objectives

1. **Age Distribution** – Examine the age group distribution among respondents.
   - **Feature Used:** age_group
2. **Educational Attainment** – Analyze the levels of education across different respondents.
   - **Feature Used:** education
3. **Income and Employment Status** – Assess variations in income levels and employment status.
   - **Features Used:** income_poverty, employment_status, employment_industry, employment_occupation
4. **Household Composition** – Investigate household structure based on marital status, homeownership, and number of adults/children.
   - **Features Used:** marital_status, rent_or_own, household_adults, household_children
5. **Geographic Demographics** – Identify demographic variations across different regions.
   - **Features Used:** hhs_geo_region, census_msa

2 Data Understanding

2.1 Data collection

The data for this competition comes from the National 2009 H1N1 Flu Survey (NHFS).

In their own words:

> The National 2009 H1N1 Flu Survey (NHFS) was sponsored by the National Center for Immunization and Respiratory Diseases (NCIRD) and conducted jointly by NCIRD and the National Center for Health Statistics (NCHS), Centers for Disease Control and Prevention (CDC). The NHFS was a list-assisted random-digit-dialing telephone survey of households, designed to monitor influenza immunization coverage in the 2009-10 season.

The target population for the NHFS was all persons 6 months or older living in the United States at the time of the interview. Data from the NHFS were used to produce timely estimates of vaccination coverage rates for both the monovalent pH1N1 and trivalent seasonal influenza vaccines.

The NHFS was conducted between October 2009 and June 2010. It was one-time survey designed specifically to monitor vaccination during the 2009-2010 flu season in response to the 2009 H1N1 pandemic. The CDC has other ongoing programs for annual phone surveys that continue to monitor seasonal flu vaccination.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV

warnings.filterwarnings("ignore")
```

```
C:\anaconda\lib\site-packages\pandas\core\arrays\masked.py:60: UserWarnin
g: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.
2' currently installed).
  from pandas.core import (
```

Loading Dataset

Features for training

In [2]:
```
# Features Training
train_features_df = pd.read_csv("data/training_set_features.csv")
```

Features for testing

In [3]:
```
#Features Test
test_features_df = pd.read_csv("data/test_set_features.csv")
```

Training Labels

In [4]:
```
train_labels_df = pd.read_csv("data/training_set_labels.csv")
```

training_set_features.csv

In [5]:
```
train_features_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 36 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   respondent_id                26707 non-null  int64  
 1   h1n1_concern                 26615 non-null  float64
 2   h1n1_knowledge               26591 non-null  float64
 3   behavioral_antiviral_meds    26636 non-null  float64
 4   behavioral_avoidance         26499 non-null  float64
 5   behavioral_face_mask         26688 non-null  float64
 6   behavioral_wash_hands        26665 non-null  float64
 7   behavioral_large_gatherings  26620 non-null  float64
 8   behavioral_outside_home      26625 non-null  float64
 9   behavioral_touch_face        26579 non-null  float64
 10  doctor_recc_h1n1             24547 non-null  float64
 11  doctor_recc_seasonal         24547 non-null  float64
 12  chronic_med_condition        25736 non-null  float64
 13  child_under_6_months         25887 non-null  float64
 14  health_worker                25903 non-null  float64
 15  health_insurance             14433 non-null  float64
 16  opinion_h1n1_vacc_effective  26316 non-null  float64
 17  opinion_h1n1_risk            26319 non-null  float64
 18  opinion_h1n1_sick_from_vacc  26312 non-null  float64
 19  opinion_seas_vacc_effective  26245 non-null  float64
 20  opinion_seas_risk            26193 non-null  float64
 21  opinion_seas_sick_from_vacc  26170 non-null  float64
 22  age_group                    26707 non-null  object 
 23  education                    25300 non-null  object 
 24  race                         26707 non-null  object 
 25  sex                          26707 non-null  object 
 26  income_poverty               22284 non-null  object 
 27  marital_status               25299 non-null  object 
 28  rent_or_own                  24665 non-null  object 
 29  employment_status            25244 non-null  object 
 30  hhs_geo_region               26707 non-null  object 
 31  census_msa                   26707 non-null  object 
 32  household_adults             26458 non-null  float64
 33  household_children           26458 non-null  float64
 34  employment_industry          13377 non-null  object 
 35  employment_occupation        13237 non-null  object 
dtypes: float64(23), int64(1), object(12)
memory usage: 7.3+ MB
```

2.2 Numerical Columns (26)

- respondent_id
- h1n1_concern
- h1n1_knowledge
- behavioral_antiviral_meds
- behavioral_avoidance
- behavioral_face_mask
- behavioral_wash_hands
- behavioral_large_gatherings
- behavioral_outside_home

- behavioral_touch_face
- doctor_recc_h1n1
- doctor_recc_seasonal
- chronic_med_condition
- child_under_6_months
- health_worker
- health_insurance
- opinion_h1n1_vacc_effective
- opinion_h1n1_risk
- opinion_h1n1_sick_from_vacc
- opinion_seas_vacc_effective
- opinion_seas_risk
- opinion_seas_sick_from_vacc
- household_adults
- household_children

2.3 Categorical Columns (10)

- age_group
- education
- race
- sex
- income_poverty
- marital_status
- rent_or_own
- employment_status
- hhs_geo_region
- census_msa
- employment_industry
- employment_occupation

Select the feautures

Features selected are to explore how demographic factors effect

In [6]:
```
train_df = pd.merge(train_features_df, train_labels_df, on="respondent_id")
train_df.drop(columns="respondent_id", inplace=True)
```

In [7]:
```
selected_features = ["age_group", "education", "income_poverty", "employmen
    "race", "sex", "marital_status", "rent_or_own","household_adults", "hou
train_df = train_df[selected_features]
```

In [8]:
```
train_df.describe()
```

Out[8]:

	household_adults	household_children	h1n1_vaccine	seasonal_vaccine
count	26458.000000	26458.000000	26707.000000	26707.000000
mean	0.886499	0.534583	0.212454	0.465608
std	0.753422	0.928173	0.409052	0.498825
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000
75%	1.000000	1.000000	0.000000	1.000000
max	3.000000	3.000000	1.000000	1.000000

In [9]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   age_group           26707 non-null  object 
 1   education           25300 non-null  object 
 2   income_poverty      22284 non-null  object 
 3   employment_status   25244 non-null  object 
 4   race                26707 non-null  object 
 5   sex                 26707 non-null  object 
 6   marital_status      25299 non-null  object 
 7   rent_or_own         24665 non-null  object 
 8   household_adults    26458 non-null  float64
 9   household_children  26458 non-null  float64
 10  hhs_geo_region      26707 non-null  object 
 11  census_msa          26707 non-null  object 
 12  h1n1_vaccine        26707 non-null  int64  
 13  seasonal_vaccine    26707 non-null  int64  
dtypes: float64(2), int64(2), object(10)
memory usage: 2.9+ MB
```

In [10]:
```
train_df.head()
```

Out[10]:

	age_group	education	income_poverty	employment_status	race	sex	marit
0	55 - 64 Years	< 12 Years	Below Poverty	Not in Labor Force	White	Female	
1	35 - 44 Years	12 Years	Below Poverty	Employed	White	Male	
2	18 - 34 Years	College Graduate	<= $75,000, Above Poverty	Employed	White	Male	
3	65+ Years	12 Years	Below Poverty	Not in Labor Force	White	Female	
4	45 - 54 Years	Some College	<= $75,000, Above Poverty	Employed	White	Female	

In [11]:
```
train_df.tail()
```

Out[11]:

	age_group	education	income_poverty	employment_status	race	sex	I
26702	65+ Years	Some College	<= $75,000, Above Poverty	Not in Labor Force	White	Female	
26703	18 - 34 Years	College Graduate	<= $75,000, Above Poverty	Employed	White	Male	
26704	55 - 64 Years	Some College	NaN	NaN	White	Female	
26705	18 - 34 Years	Some College	<= $75,000, Above Poverty	Employed	Hispanic	Female	
26706	65+ Years	Some College	<= $75,000, Above Poverty	Not in Labor Force	White	Male	

In [12]:
```
train_df.sample(5)
```

Out[12]:

	age_group	education	income_poverty	employment_status	race	sex
21187	65+ Years	College Graduate	> $75,000	Employed	White	Male
3964	35 - 44 Years	< 12 Years	Below Poverty	Employed	Hispanic	Female
18362	45 - 54 Years	College Graduate	<= $75,000, Above Poverty	Employed	Hispanic	Male
3405	18 - 34 Years	12 Years	<= $75,000, Above Poverty	Employed	Hispanic	Female
1620	45 - 54 Years	College Graduate	> $75,000	Employed	White	Female

3 Data Cleaning

3.1 Correct formats

training_set_features.csv

In [13]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   age_group          26707 non-null  object 
 1   education          25300 non-null  object 
 2   income_poverty     22284 non-null  object 
 3   employment_status  25244 non-null  object 
 4   race               26707 non-null  object 
 5   sex                26707 non-null  object 
 6   marital_status     25299 non-null  object 
 7   rent_or_own        24665 non-null  object 
 8   household_adults   26458 non-null  float64
 9   household_children 26458 non-null  float64
 10  hhs_geo_region     26707 non-null  object 
 11  census_msa         26707 non-null  object 
 12  h1n1_vaccine       26707 non-null  int64  
 13  seasonal_vaccine   26707 non-null  int64  
dtypes: float64(2), int64(2), object(10)
memory usage: 2.9+ MB
```

Formats are as expected

3.1.1 Missing Values

The follow are missing values: `education, income_poverty, employment_status marital status and rent_or_own` . Checking values

In [14]:
```
train_df.isna().sum()
```

Out[14]:
```
age_group                0
education             1407
income_poverty        4423
employment_status     1463
race                     0
sex                      0
marital_status        1408
rent_or_own           2042
household_adults       249
household_children     249
hhs_geo_region           0
census_msa               0
h1n1_vaccine             0
seasonal_vaccine         0
dtype: int64
```

In [15]:
```
missing = ["education", "income_poverty", "employment_status", "marital_sta
           ,"rent_or_own","household_adults","household_children"]
for col in missing:
    train_df[col].fillna(train_df[col].mode()[0], inplace=True)
```

In [16]:
```
train_df.isna().sum()
```

Out[16]:
```
age_group             0
education             0
income_poverty        0
employment_status     0
race                  0
sex                   0
marital_status        0
rent_or_own           0
household_adults      0
household_children    0
hhs_geo_region        0
census_msa            0
h1n1_vaccine          0
seasonal_vaccine      0
dtype: int64
```

In [17]:
```
test_features_df.isna().sum()
```

Out[17]:
```
respondent_id                      0
h1n1_concern                      85
h1n1_knowledge                   122
behavioral_antiviral_meds         79
behavioral_avoidance             213
behavioral_face_mask              19
behavioral_wash_hands             40
behavioral_large_gatherings       72
behavioral_outside_home           82
behavioral_touch_face            128
doctor_recc_h1n1                2160
doctor_recc_seasonal            2160
chronic_med_condition            932
child_under_6_months             813
health_worker                    789
health_insurance               12228
opinion_h1n1_vacc_effective      398
opinion_h1n1_risk                380
opinion_h1n1_sick_from_vacc      375
opinion_seas_vacc_effective      452
opinion_seas_risk                499
opinion_seas_sick_from_vacc      521
age_group                          0
education                       1407
race                               0
sex                                0
income_poverty                  4497
marital_status                  1442
rent_or_own                     2036
employment_status               1471
hhs_geo_region                     0
census_msa                         0
household_adults                 225
household_children               225
employment_industry            13275
employment_occupation          13426
dtype: int64
```

3.1.2 Changing Columns

In [18]:
```
train_df.columns
```

Out[18]:
```
Index(['age_group', 'education', 'income_poverty', 'employment_status', 'r
ace',
       'sex', 'marital_status', 'rent_or_own', 'household_adults',
       'household_children', 'hhs_geo_region', 'census_msa', 'h1n1_vaccin
e',
       'seasonal_vaccine'],
      dtype='object')
```

Columns are in the desired format

3.1.3 Checking Duplicates

In [19]:
```
#Checking there duplicates in training set
print(train_df.shape[0])
train_df.duplicated().sum()
```

```
26707
```

Out[19]:
```
5632
```

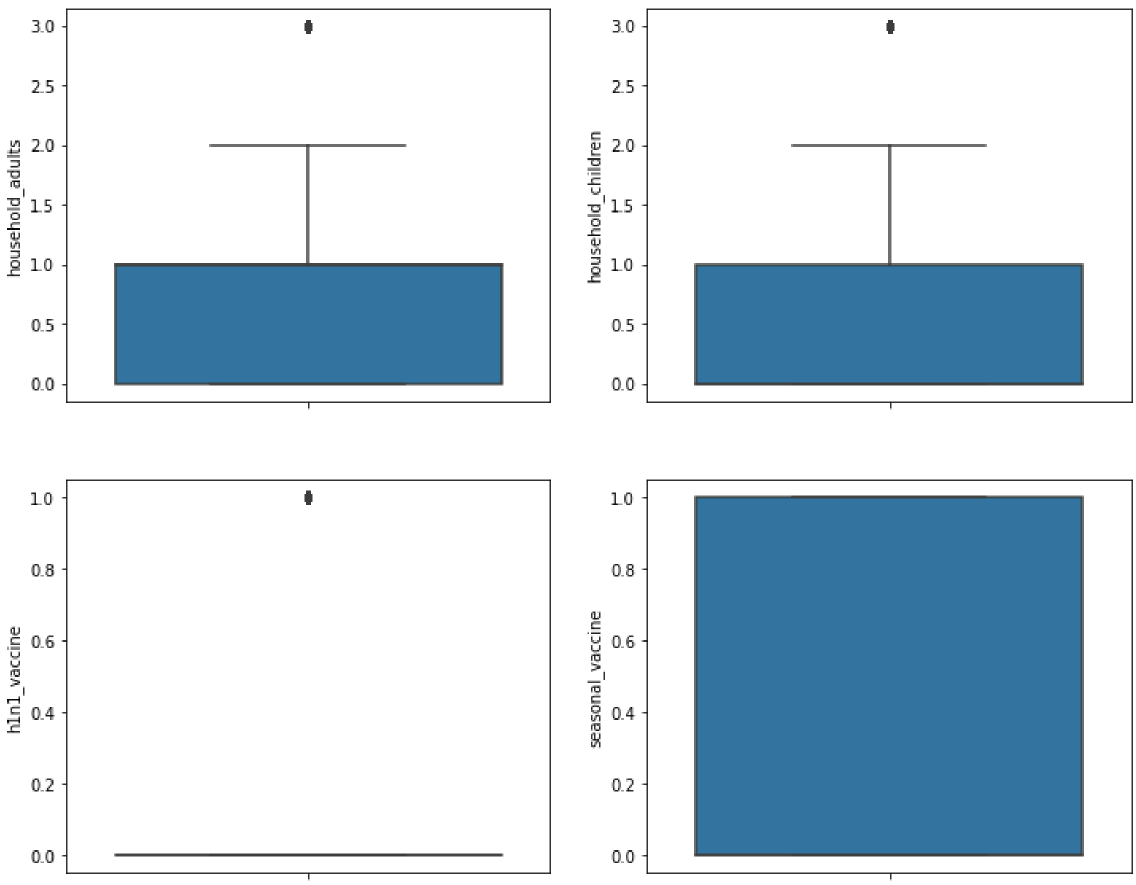
In [20]:
```
train_df.drop_duplicates(inplace=True)
print(train_df.shape[0])
```

```
21075
```

3.1.4 Checking Outliers

In [21]:
```
numeric_df =train_df.select_dtypes(include = ['number'])
```

In [22]:
```
#calculate the number of fig to fit height
grid=(numeric_df.shape[1]+1)//2
#allocating each plot a height of 5
plt.figure(figsize=(12, grid * 5))

count=0
for col in numeric_df:
    count += 1
    plt.subplot(grid,2,count)
    sns.boxplot(y=train_df[col])
```



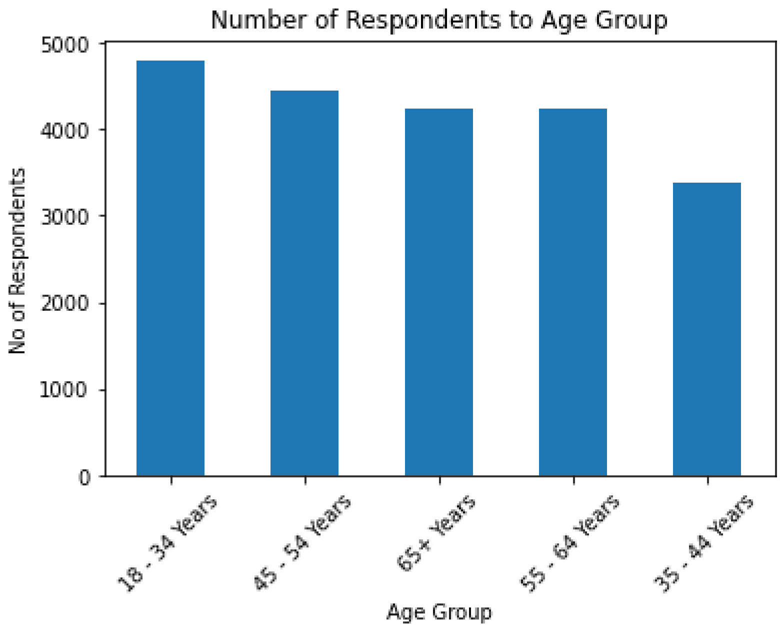
3.1.5 Saving Dataset

In [23]:
```
train_df.to_csv("train_clean.csv")
```

4 Explanatory Analysis

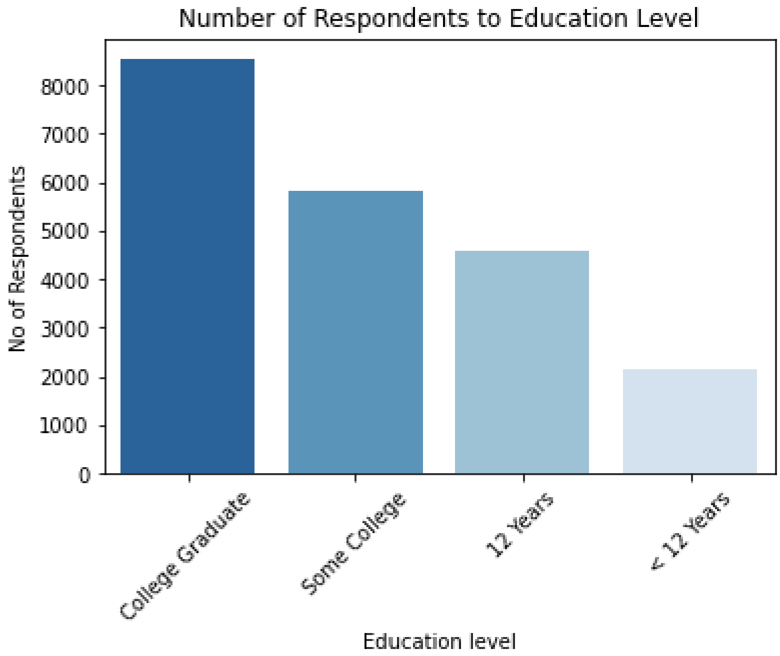
4.1 Univariate Analysis

In [24]:
```
age_count =  train_df['age_group'].value_counts()
age_count.plot(kind="bar")
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('Age Group')
plt.title("Number of Respondents to Age Group")
plt.show()
```



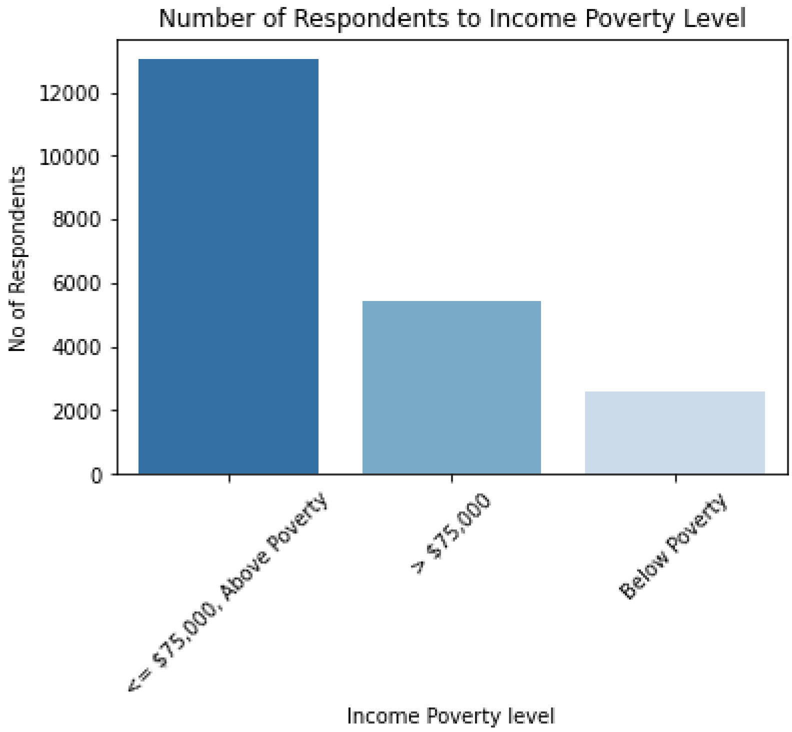
65+ years are the most respondents while **35-44 Years** are the least

In [25]:
```
education_count =  train_df['education'].value_counts()
sns.countplot(x=train_df['education'],order=education_count.index,palette='I
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('Education level')
plt.title("Number of Respondents to Education Level")
plt.show()
```



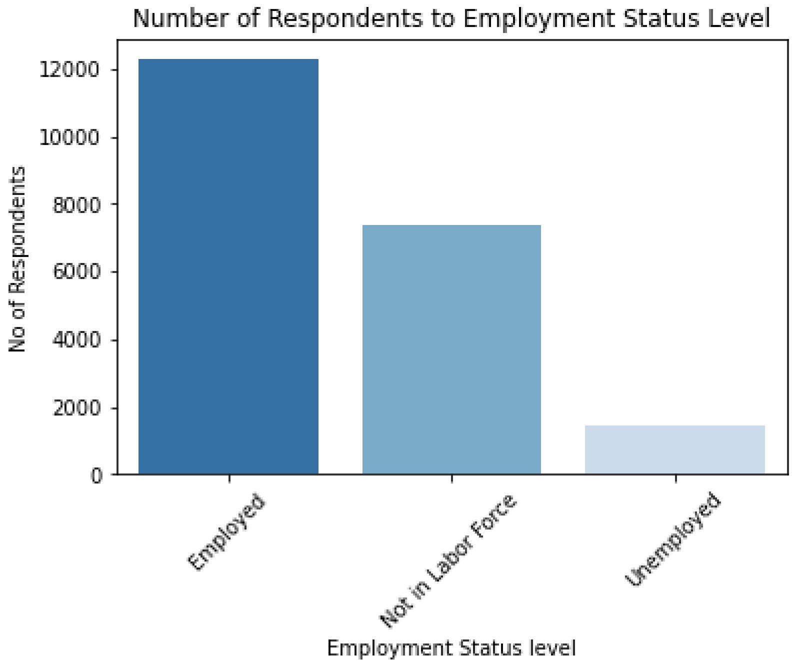
The survey had many **College Graduate** than other level.With the least be **<12 Years**

In [26]:
```
income_poverty_count =  train_df['income_poverty'].value_counts()
sns.countplot(x=train_df['income_poverty'],order=income_poverty_count.index
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('Income Poverty level')
plt.title("Number of Respondents to Income Poverty Level")
plt.show()
```



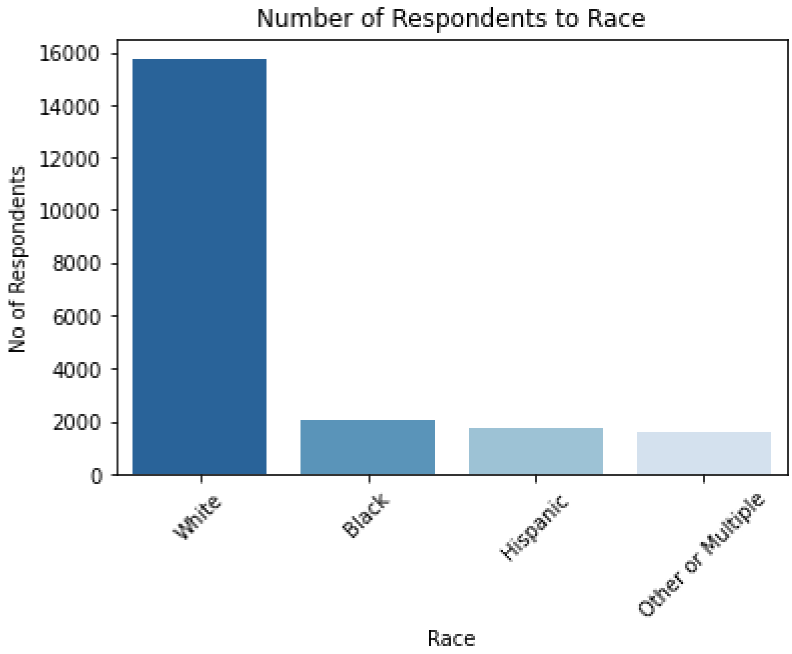
The survey had many **Above Poverty** respondents than other level.With the least be **Below Poverty**

In [27]:
```
employment_status_count =  train_df['employment_status'].value_counts()
sns.countplot(x=train_df['employment_status'],order=employment_status_count
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('Employment Status level')
plt.title("Number of Respondents to Employment Status Level")
plt.show()
```



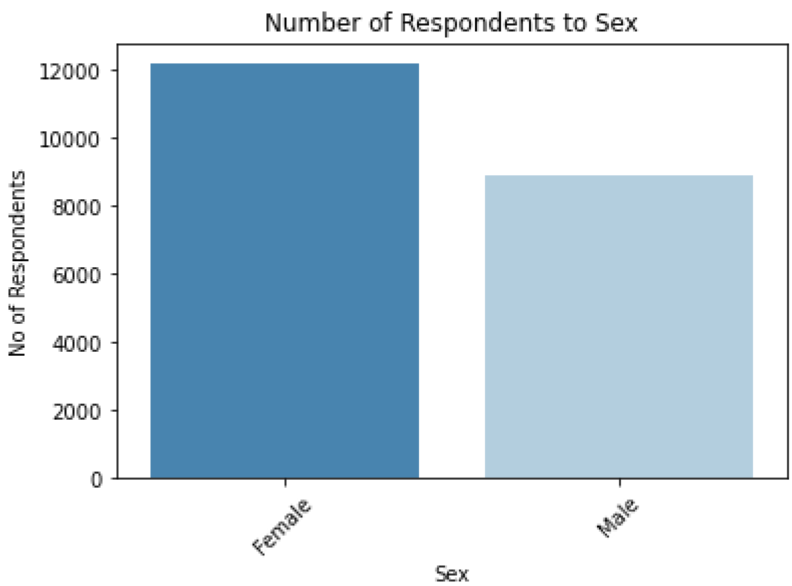
The survey had many **Employed** respondents than other level.With the least be **Unemployed**

In [28]:
```
race_count =  train_df['race'].value_counts()
sns.countplot(x=train_df['race'],order=race_count.index,palette='Blues_r')
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('Race')
plt.title("Number of Respondents to Race")
plt.show()
```



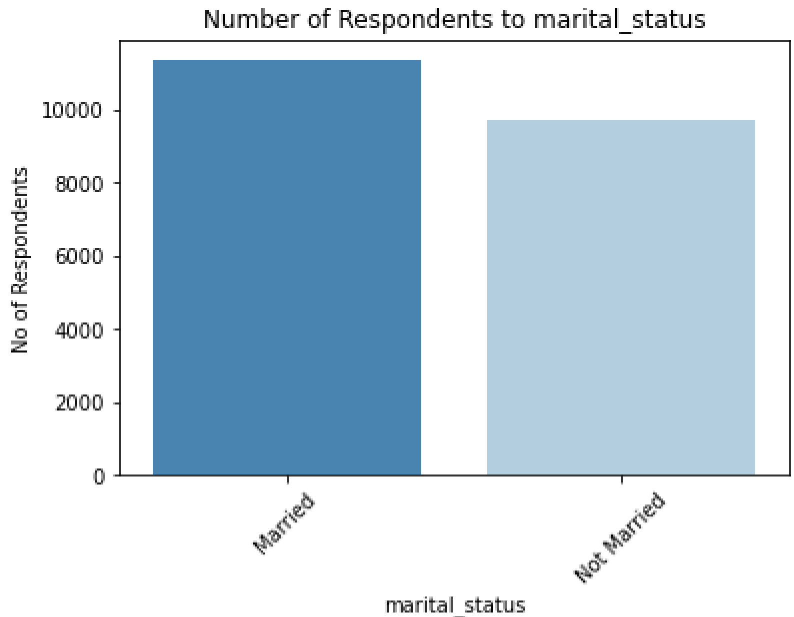
The survey had many **White** respondents than other level.With the least be **Other or Multiple**

In [29]:
```
sex_count =  train_df['sex'].value_counts()
sns.countplot(x=train_df['sex'],order=sex_count.index,palette='Blues_r')
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('Sex')
plt.title("Number of Respondents to Sex")
plt.show()
```



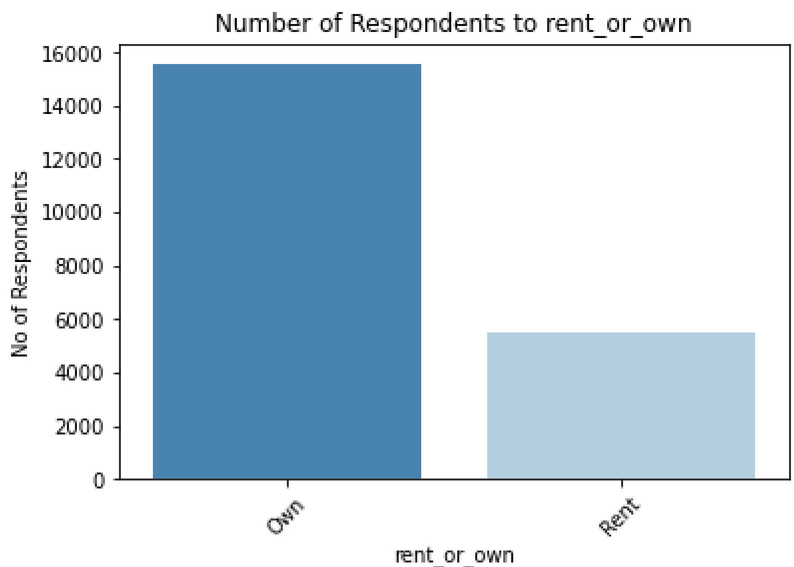
The survey had many **Female** respondents.With the least be **Male**

In [30]:
```
marital_status_count =  train_df['marital_status'].value_counts()
sns.countplot(x=train_df['marital_status'],order=marital_status_count.index
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('marital_status')
plt.title("Number of Respondents to marital_status")
plt.show()
```



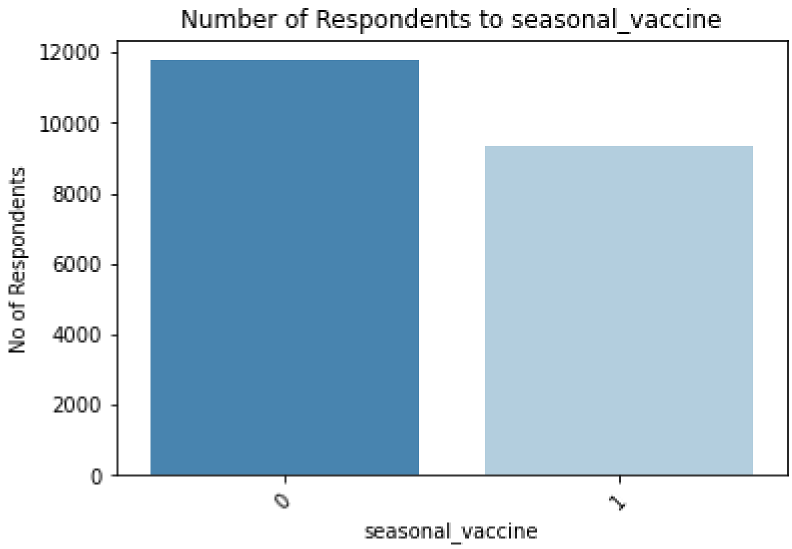
The survey had many **Married** respondents than other level.With the least be **Not Married**

In [31]:
```
rent_or_own_count =  train_df['rent_or_own'].value_counts()
sns.countplot(x=train_df['rent_or_own'],order=rent_or_own_count.index,palet
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('rent_or_own')
plt.title("Number of Respondents to rent_or_own")
plt.show()
```



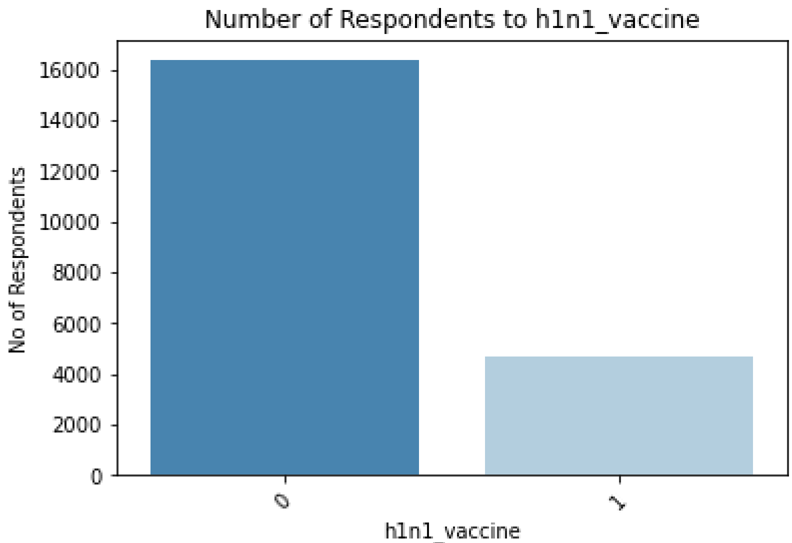
The survey had many **Own** respondents than other level.With the least be **Rent**

In [32]:
```
seasonal_vaccine_count =  train_df['seasonal_vaccine'].value_counts()
sns.countplot(x=train_df['seasonal_vaccine'],order=seasonal_vaccine_count.i
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('seasonal_vaccine')
plt.title("Number of Respondents to seasonal_vaccine")
plt.show()
```



The survey had many **not vaccinated** respondents for seasonal vaccine.With the least be **vaccinated**

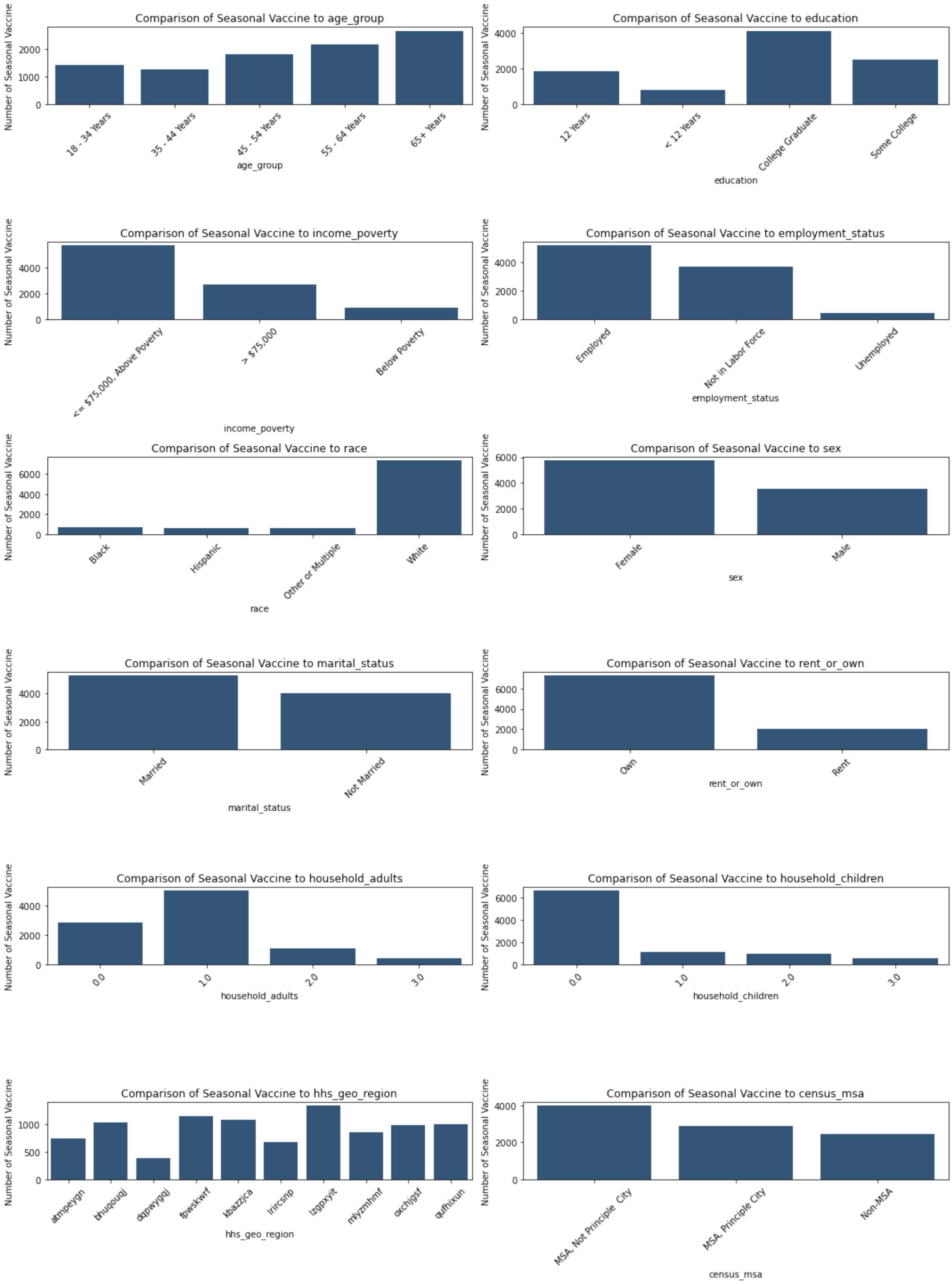
In [33]:
```
h1n1_vaccine_count =  train_df['h1n1_vaccine'].value_counts()
sns.countplot(x=train_df['h1n1_vaccine'],order=h1n1_vaccine_count.index,pal
plt.ylabel('No of Respondents')
plt.xticks(rotation=45)
plt.xlabel('h1n1_vaccine')
plt.title("Number of Respondents to h1n1_vaccine")
plt.show()
```



The survey had many **not vaccinated** respondents for h1n1 vaccine.With the least be

4.2 Bivariate Analysis

In [34]:
```
selected_features = ["age_group", "education", "income_poverty", "employmen
    "race", "sex", "marital_status", "rent_or_own","household_adults", "hou

fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(15, 20))

axes = axes.flatten()

# Loop through each column in the list
for i, feature in enumerate(selected_features):
    # creating pivot table to aggregate the feature by seasonal_vaccine
    feature_seasonal_vaccine = train_df.pivot_table(index=feature, values='


    sns.barplot(x=feature_seasonal_vaccine.index,
                y=feature_seasonal_vaccine['seasonal_vaccine'], color='#2a5
                ci=None, ax=axes[i])

    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Number of Seasonal Vaccine')
    axes[i].set_title(f'Comparison of Seasonal Vaccine to {feature}')
    axes[i].tick_params(axis='x', rotation=45)

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



1. Took most seasonal vaccines compared to other age groups.

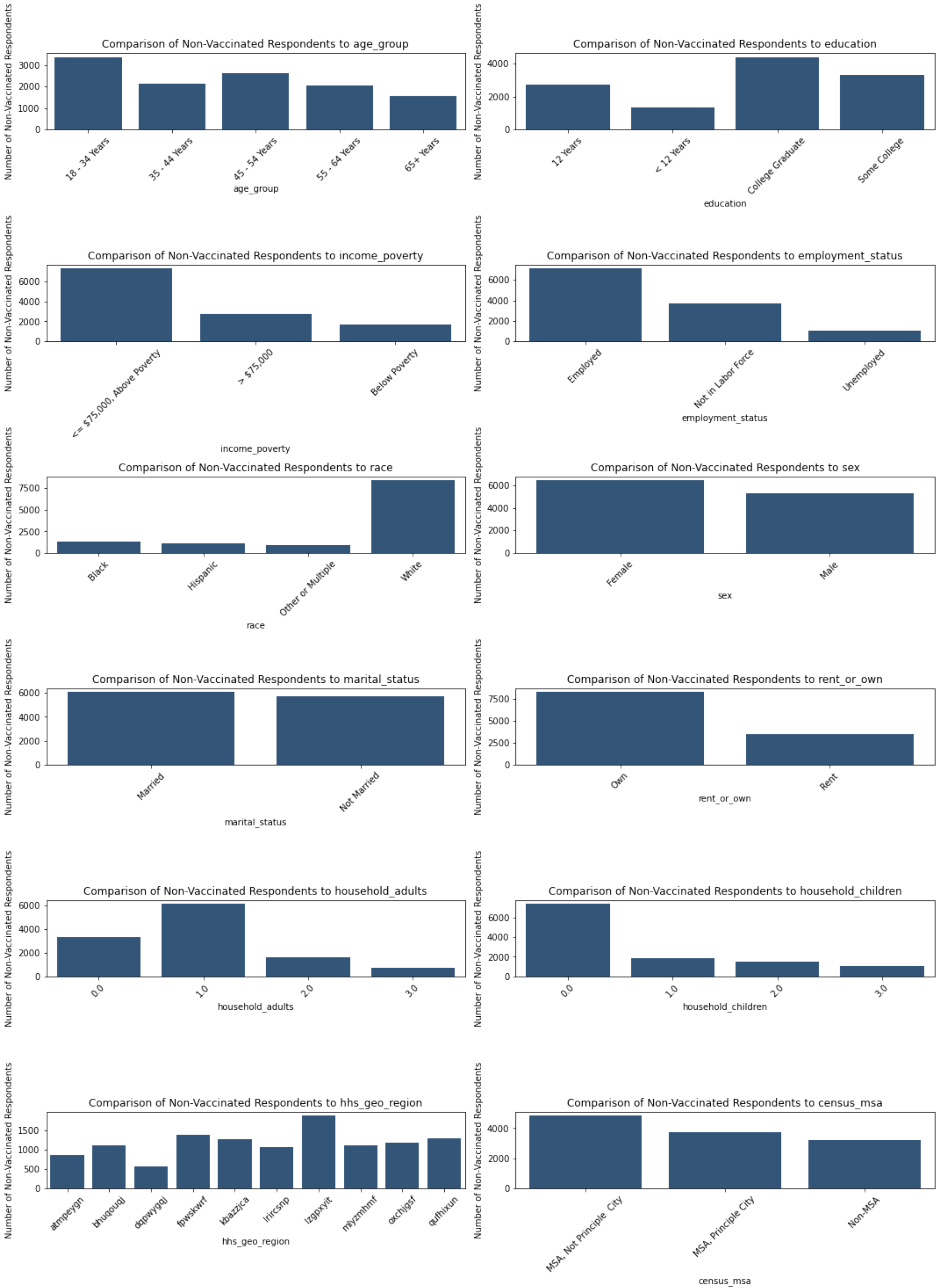
- **65+ Years**
- **College Graduate**
- **Female**
- **White**
- **Own House**
- **Married**
- **<=75000 Above Property**

2. Took least seasonal vaccines compared to other age groups.

- **35 - 44 Years**

- **<12 Years** of education
- **Male**
- **Black,Hispanic and Other**
- **Rent House**
- **Not Married**
- **Below Property**

In [35]:
```
 # Create subplots, setting the number of rows and columns
fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(15, 20))

# Flatten axes array for easy iteration
axes = axes.flatten()

# Loop through each column in the list
for i, feature in enumerate(selected_features):
    # Filter the train_df to only include rows where seasonal_vaccine == 0
    feature_seasonal_vaccine_0 = train_df[train_df['seasonal_vaccine'] == 0

    # Creating pivot table to aggregate the feature by seasonal_vaccine == 0
    feature_seasonal_vaccine = feature_seasonal_vaccine_0.pivot_table(index=

    # Plotting data
    sns.barplot(x=feature_seasonal_vaccine.index,
                y=feature_seasonal_vaccine['seasonal_vaccine'], color='#2a5
                ci=None, ax=axes[i])

    # Set labels and titles for each plot
    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Number of Non-Vaccinated Respondents')
    axes[i].set_title(f'Comparison of Non-Vaccinated Respondents to {featur
    axes[i].tick_params(axis='x', rotation=45)  # Rotate x-axis labels if n

# Adjust layout for better spacing
plt.tight_layout()
plt.show()
```

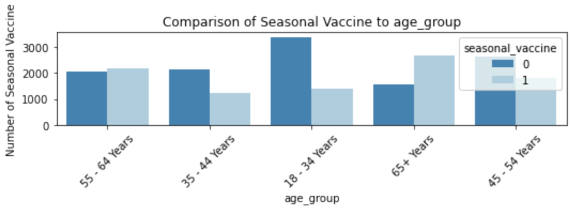


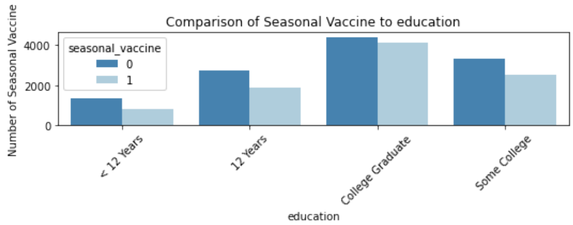
1. Most Non-vaccinated seasonal vaccines compared to other age groups.

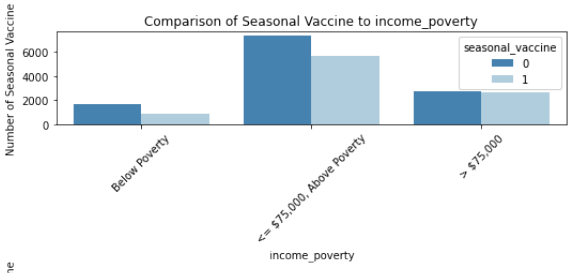
- **18 - 24 Years**
- **College Graduate**
- **Female**
- **White**
- **Employed**
- **Own House**
- **Married**
- **>=75000 Above Property**

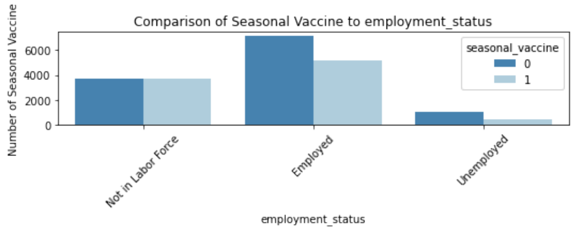
4.3 Multivariate Analysis

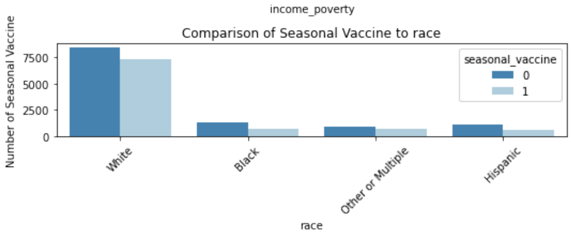
In [36]:
```
fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(15, 20))

axes = axes.flatten()

# loop through each column in the list
for i, feature in enumerate(selected_features):
    # creating pivot table to aggregate the feature by seasonal_vaccine

    sns.countplot(x=train_df[feature],hue=train_df['seasonal_vaccine'],pale
                  ax=axes[i])

    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Number of Seasonal Vaccine')
    axes[i].set_title(f'Comparison of Seasonal Vaccine to {feature}')
    axes[i].tick_params(axis='x', rotation=45)

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

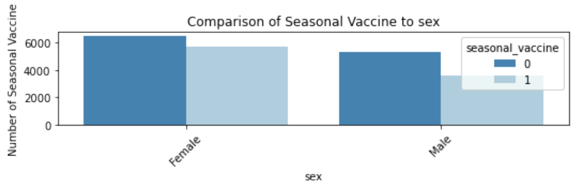


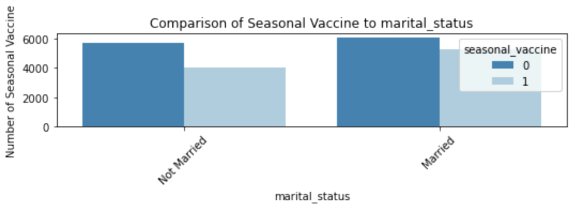


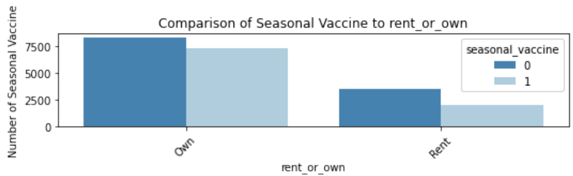


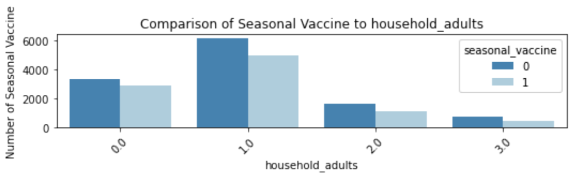


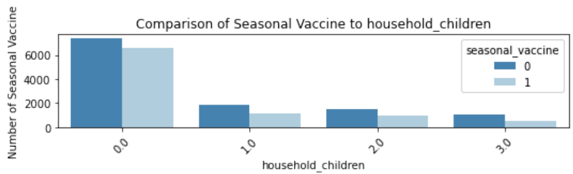


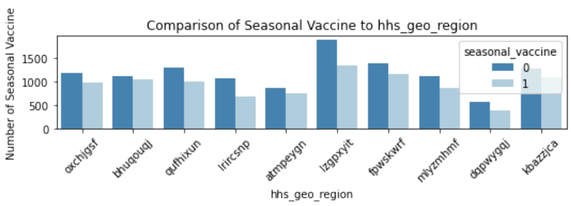


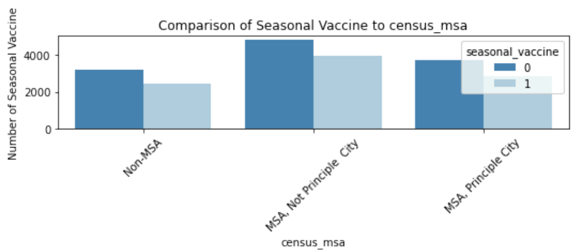






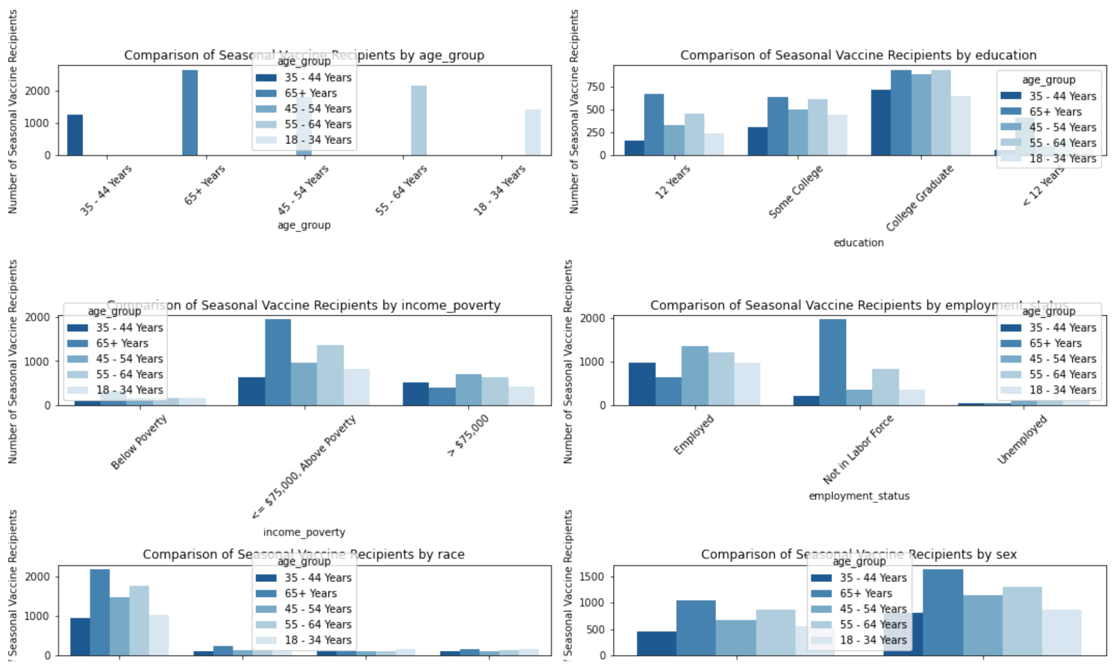




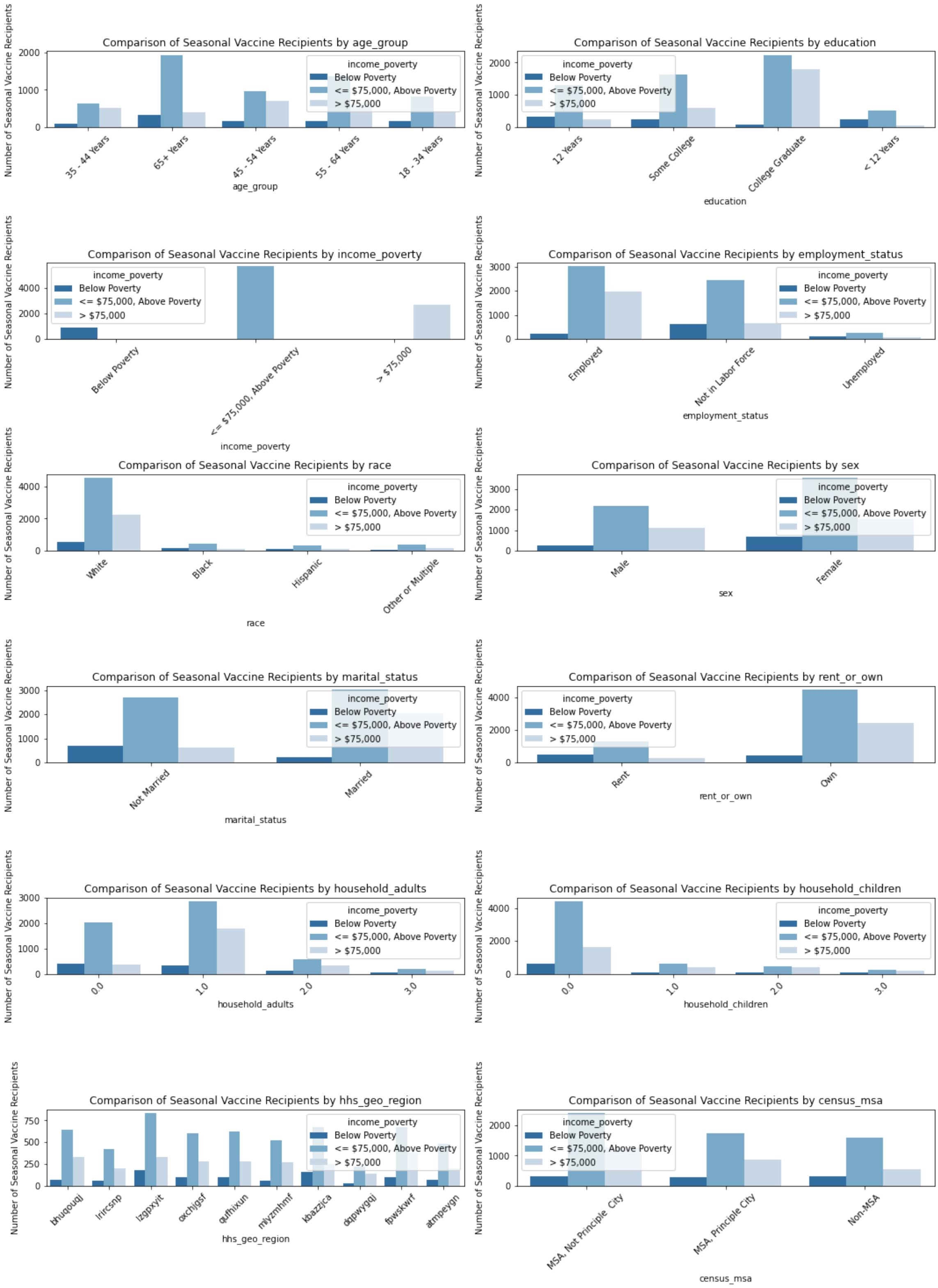


1. All age groups except 65 Years and 55 - 64 Years which have more are vaccinated than non vaccinated.
2. All education level except College Graduate which have more are vaccinated than non vaccinated.
3. All income levels which have more are non-vaccinated than vaccinated.
4. All employment status except not in labor force which have more are vaccinated than non vaccinated.
5. All sex, marital status and rent status which have more are non-vaccinated than vaccinated.

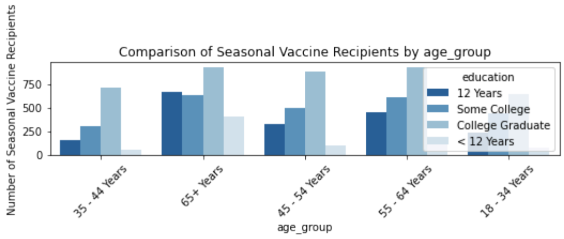
In [37]:
```
# filter dataset where seasonal_vaccine was given
df_vaccinated = train_df[train_df['seasonal_vaccine'] == 1]

fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(15, 20))

axes = axes.flatten()

# loop through each column in the list
for i, feature in enumerate(selected_features):
    sns.countplot(x=df_vaccinated[feature], hue=df_vaccinated['age_group'],
                  palette='Blues_r', ax=axes[i])

    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Number of Seasonal Vaccine Recipients')
    axes[i].set_title(f'Comparison of Seasonal Vaccine Recipients by {featu
    axes[i].tick_params(axis='x', rotation=45)

# remove any empty subplots
for j in range(len(col), len(axes)):
    fig.delaxes(axes[j])

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

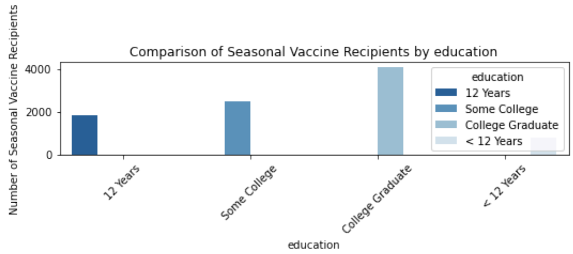


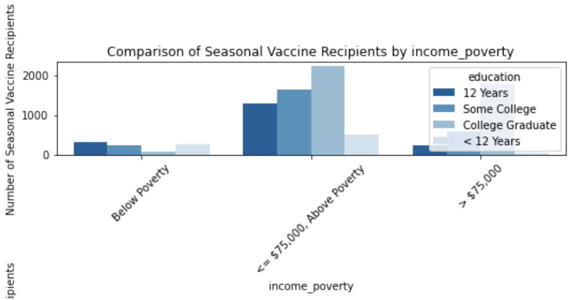
In [38]:
```
# filter dataset where seasonal_vaccine was given
df_vaccinated = train_df[train_df['seasonal_vaccine'] == 1]

fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(15, 20))
axes = axes.flatten()

# Loop through each column in the list
for i, feature in enumerate(selected_features):
    sns.countplot(x=df_vaccinated[feature], hue=df_vaccinated["income_pover
                  palette='Blues_r', ax=axes[i])

    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Number of Seasonal Vaccine Recipients')
    axes[i].set_title(f'Comparison of Seasonal Vaccine Recipients by {featu
    axes[i].tick_params(axis='x', rotation=45)

# remove any empty subplots
for j in range(len(col), len(axes)):
    fig.delaxes(axes[j])

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

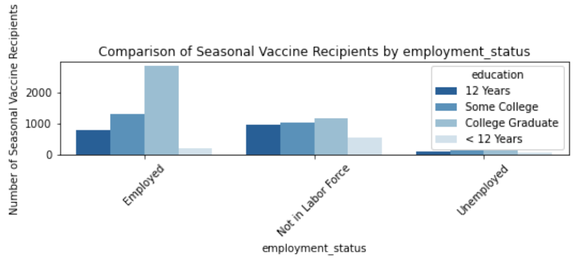

Comparison of Seasonal Vaccine Recipients by age_group
Number of Seasonal Vaccine Recipients
income_poverty
Below Poverty
<= $75,000, Above Poverty
> $75,000
35 - 44 Years
65+ Years
45 - 54 Years
55 - 64 Years
18 - 34 Years
age_group
Comparison of Seasonal Vaccine Recipients by education
12 Years
Some College
College Graduate
< 12 Years
education
Comparison of Seasonal Vaccine Recipients by income_poverty
income_poverty
Comparison of Seasonal Vaccine Recipients by employment_status
Employed
Not in Labor Force
Unemployed
employment_status
Comparison of Seasonal Vaccine Recipients by race
White
Black
Hispanic
Other or Multiple
race
Comparison of Seasonal Vaccine Recipients by sex
Male
Female
sex
Comparison of Seasonal Vaccine Recipients by marital_status
Not Married
Married
marital_status
Comparison of Seasonal Vaccine Recipients by rent_or_own
Rent
Own
rent_or_own
Comparison of Seasonal Vaccine Recipients by household_adults
0.0
1.0
2.0
3.0
household_adults
Comparison of Seasonal Vaccine Recipients by household_children
household_children
Comparison of Seasonal Vaccine Recipients by hhs_geo_region
bhuqouqj
lrircsnp
lzgpxyit
oxchjgsf
qufhixun
mlyzmhmf
kbazzjca
dqpwygqj
fpwskwrf
atmpeygn
hhs_geo_region
Comparison of Seasonal Vaccine Recipients by census_msa
MSA, Not Principle City
MSA, Principle City
Non-MSA
census_msa

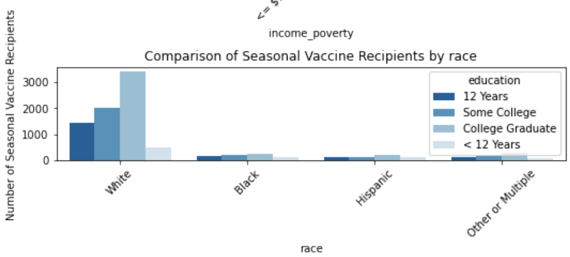
In [39]:
```
# filter dataset where seasonal_vaccine was given
df_vaccinated = train_df[train_df['seasonal_vaccine'] == 1]

fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(15, 20))
axes = axes.flatten()

# loop through each column in the list
for i, feature in enumerate(selected_features):
    sns.countplot(x=df_vaccinated[feature], hue=df_vaccinated["education"],
                  palette='Blues_r', ax=axes[i])

    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Number of Seasonal Vaccine Recipients')
    axes[i].set_title(f'Comparison of Seasonal Vaccine Recipients by {featu
    axes[i].tick_params(axis='x', rotation=45)

# remove any empty subplots
for j in range(len(col), len(axes)):
    fig.delaxes(axes[j])

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

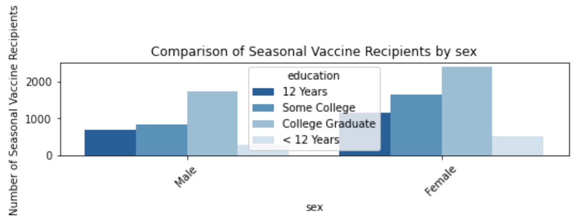

Comparison of Seasonal Vaccine Recipients by age_group
Number of Seasonal Vaccine Recipients
age_group
35 - 44 Years
65+ Years
45 - 54 Years
55 - 64 Years
18 - 34 Years
education
12 Years
Some College
College Graduate
< 12 Years

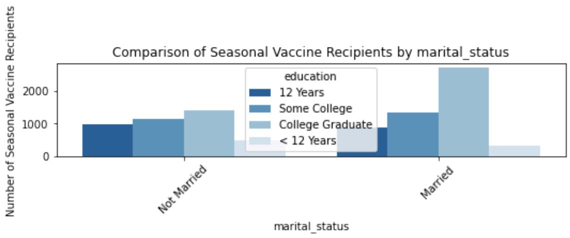

Comparison of Seasonal Vaccine Recipients by education
Number of Seasonal Vaccine Recipients
education
12 Years
Some College
College Graduate
< 12 Years

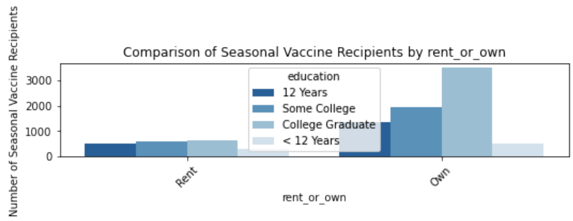

Comparison of Seasonal Vaccine Recipients by income_poverty
Number of Seasonal Vaccine Recipients
income_poverty
Below Poverty
<= $75,000, Above Poverty
> $75,000
education
12 Years
Some College
College Graduate
< 12 Years

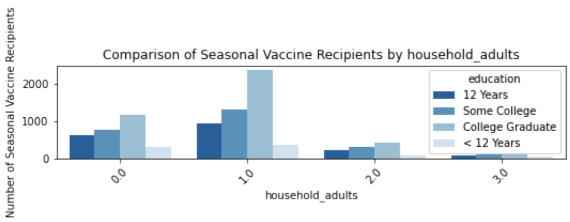

Comparison of Seasonal Vaccine Recipients by employment_status
Number of Seasonal Vaccine Recipients
employment_status
Employed
Not in Labor Force
Unemployed
education
12 Years
Some College
College Graduate
< 12 Years

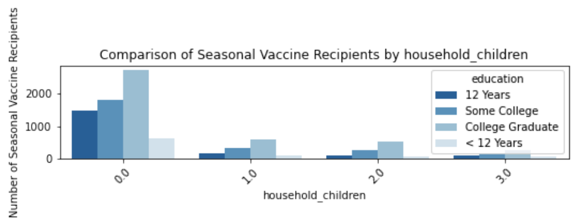

Comparison of Seasonal Vaccine Recipients by race
Number of Seasonal Vaccine Recipients
race
White
Black
Hispanic
Other or Multiple
education
12 Years
Some College
College Graduate
< 12 Years

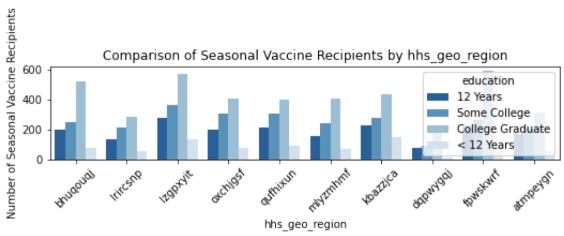

Comparison of Seasonal Vaccine Recipients by sex
Number of Seasonal Vaccine Recipients
sex
Male
Female
education
12 Years
Some College
College Graduate
< 12 Years

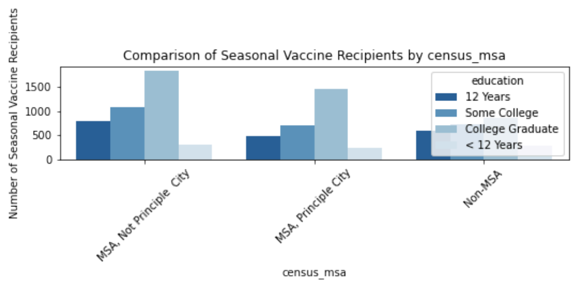

Comparison of Seasonal Vaccine Recipients by marital_status
Number of Seasonal Vaccine Recipients
marital_status
Not Married
Married
education
12 Years
Some College
College Graduate
< 12 Years


Comparison of Seasonal Vaccine Recipients by rent_or_own
Number of Seasonal Vaccine Recipients
rent_or_own
Rent
Own
education
12 Years
Some College
College Graduate
< 12 Years

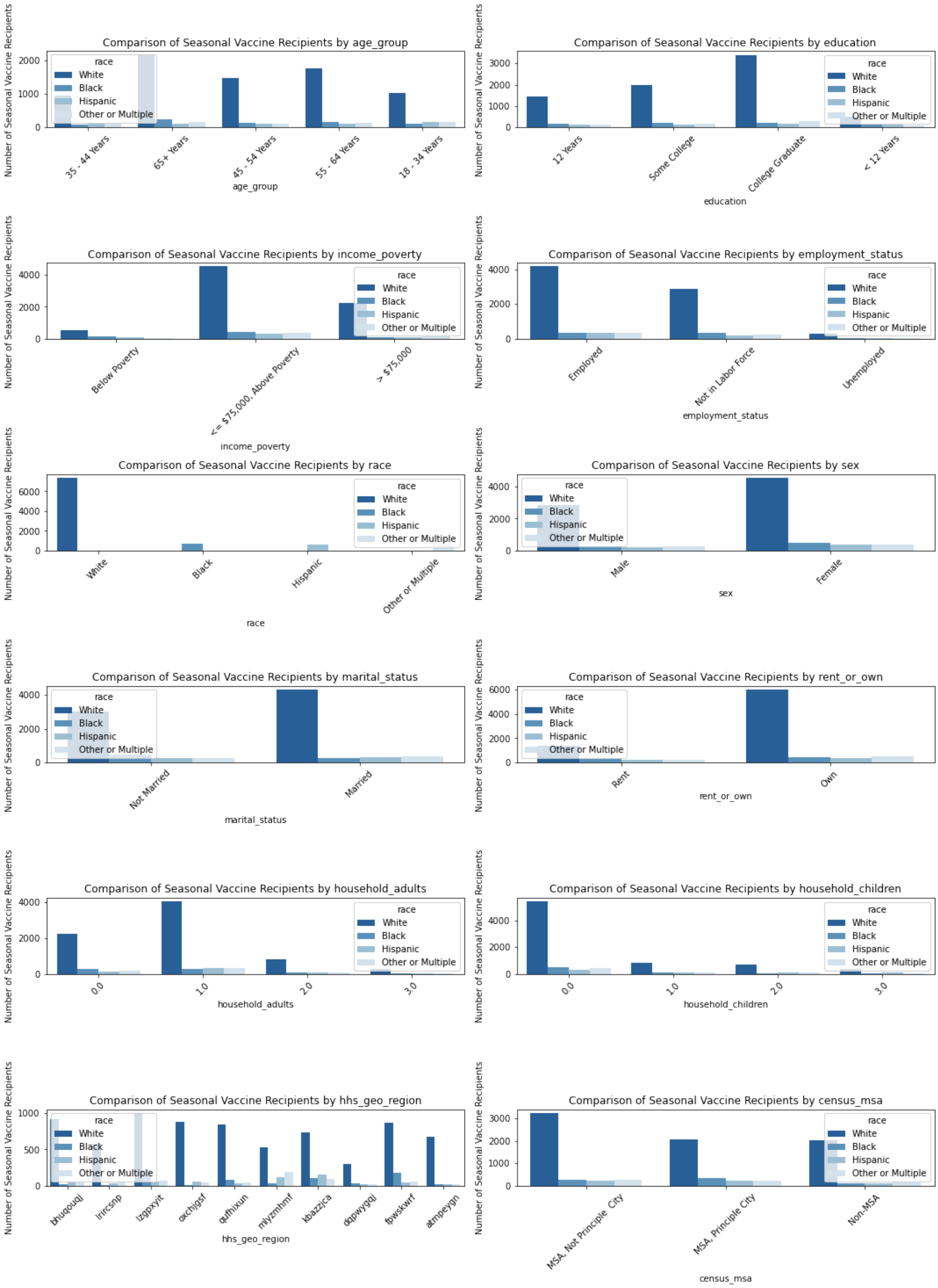

Comparison of Seasonal Vaccine Recipients by household_adults
Number of Seasonal Vaccine Recipients
household_adults
0.0
1.0
2.0
3.0
education
12 Years
Some College
College Graduate
< 12 Years


Comparison of Seasonal Vaccine Recipients by household_children
Number of Seasonal Vaccine Recipients
household_children
0.0
1.0
2.0
3.0
education
12 Years
Some College
College Graduate
< 12 Years


Comparison of Seasonal Vaccine Recipients by hhs_geo_region
Number of Seasonal Vaccine Recipients
hhs_geo_region
bhuqouqj
lrircsnp
lzgpxyit
oxchjgsf
qufhixun
mlyzmhmf
kbazzjca
dqpwygqj
fpwskwrf
atmpeygn
education
12 Years
Some College
College Graduate
< 12 Years


Comparison of Seasonal Vaccine Recipients by census_msa
Number of Seasonal Vaccine Recipients
census_msa
MSA, Not Principle City
MSA, Principle City
Non-MSA
education
12 Years
Some College
College Graduate
< 12 Years

In [40]:
```
# filter dataset where seasonal_vaccine was given
df_vaccinated = train_df[train_df['seasonal_vaccine'] == 1]


fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(15, 20))
axes = axes.flatten()

# loop through each column in the list
for i, feature in enumerate(selected_features):
    sns.countplot(x=df_vaccinated[feature], hue=df_vaccinated["race"],
                  palette='Blues_r', ax=axes[i])

    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Number of Seasonal Vaccine Recipients')
    axes[i].set_title(f'Comparison of Seasonal Vaccine Recipients by {featu
    axes[i].tick_params(axis='x', rotation=45)

# remove any empty subplots
for j in range(len(col), len(axes)):
    fig.delaxes(axes[j])

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



5 Preprocessing

In [41]:
```
# columns to encode
categorical_columns = ["age_group", "education", "income_poverty", "employm
                       "race", "sex", "marital_status", "rent_or_own","hhs_

label_encoders = {}

# encoding to each categorical column
for col in categorical_columns:
    encoder = LabelEncoder()
    encoder.fit(train_df[col])
    train_df[col] = encoder.transform(train_df[col])
    label_encoders[col] = encoder

train_df.head()
```

Out[41]:

	age_group	education	income_poverty	employment_status	race	sex	marit
0	3	1	2	1	3	0	
1	1	0	2	0	3	1	
2	0	2	0	0	3	1	
3	4	0	2	1	3	0	
4	2	3	0	0	3	0	

In [42]:
```
X = train_df.drop(columns=["seasonal_vaccine","h1n1_vaccine"])
y = train_df.seasonal_vaccine
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2, random_
```

6 Modeling

In [43]:
```
# train logreg model
logreg = LogisticRegression(random_state=42)
logreg.fit(X_train, y_train)
```

Out[43]:

```
▾ LogisticRegression
LogisticRegression(random_state=42)
```

(https://scikit-learn.org/1.6/modules/generated/sklearn.linear_mod

In [44]:
```
model = LogisticRegression()

# Define the parameter grid
param_grid = {
    'C': [0.1, 1, 10],
    'solver': ['liblinear', 'saga']
}

# Set up GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_

# Fit the model
grid_search.fit(X_train, y_train)

# Print the best parameters and score
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best cross-validation score: {grid_search.best_score_}")

# Use the best model to predict
logreg  = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters: {'C': 0.1, 'solver': 'liblinear'}
Best cross-validation score: 0.6198695136417557
```

In [45]:
```
# train dt model
dtc_model = DecisionTreeClassifier(random_state=42)
dtc_model.fit(X_train, y_train)
```

Out[45]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```
(https://scikit-learn.org/1.6/modules/generated/sklearn.tree.I

In [46]:
```
# train rf model
rfc_model = RandomForestClassifier(n_estimators=200, max_depth=10, random_s
rfc_model.fit(X_train, y_train)
```

Out[46]:
```
RandomForestClassifier
RandomForestClassifier(max_depth=10, n_estimators=200, random_state=42)
```
(https:
learn.o

In [47]:
```
# train svc model
svm_model = SVC(random_state=42)
svm_model.fit(X_train, y_train)
```

Out[47]:
```
SVC
SVC(random_state=42)
```
(https://scikit-learn.org/1.6/modules/generated/sklearn.svm.SVC.html)

In [48]:
```
model = SVC()

# Define the parameter grid
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}

# Set up GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_

# Fit the model
grid_search.fit(X_train, y_train)

# Print the best parameters and score
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best cross-validation score: {grid_search.best_score_}")

# Use the best model to predict
svm_mode = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Best parameters: {'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}
Best cross-validation score: 0.6201067615658363
```

In [49]:
```
# naive bayes classifier
nbc_model= MultinomialNB()
# Train the model
nbc_model.fit(X_train, y_train)
```

Out[49]:
```
▾ MultinomialNB
MultinomialNB()
```
(https://scikit-learn.org/1.6/modules/generated/sklearn.naive_bayes.MultinomialNB.

7 Evaluation

In [53]:

```
# logreg model
y_predict = logreg.predict(X_test)
# evaluate the logreg model
logreg_accuracy = accuracy_score(y_test,y_predict)

# dt model
y_pred = dtc_model.predict(X_test)
# Evaluate the dt model
dtc_model_accuracy = accuracy_score(y_test, y_pred)

#rf model
y_pred = rfc_model.predict(X_test)
# evaluate the rf model
rfc_model_accuracy = accuracy_score(y_test, y_pred)

# svm model
y_pred = svm_model.predict(X_test)
# evaluate the svm model
svm_model_accuracy = accuracy_score(y_test, y_pred)

# naive bayes classifier
y_pred = nbc_model.predict(X_test)
# evaluate the svm model
nbc_model_accuracy = accuracy_score(y_test, y_pred)
print("Logistic Regression Accuracy = ", logreg_accuracy )
print("Decision Tree Classifier Accuracy = ", dtc_model_accuracy)
print("Random Forest Accuracy = ",  rfc_model_accuracy)
print("SVM Accuracy = ", svm_model_accuracy)
print("Naive Bayes Classifier = ", nbc_model_accuracy)
```

```
Logistic Regression Accuracy =  0.6125741399762752
Decision Tree Classifier Accuracy =  0.4623962040332147
Random Forest Accuracy =  0.6166073546856465
SVM Accuracy =  0.6166073546856465
Naive Bayes Classifier =  0.6099644128113879
```

8 Conclusion

- **Logistic Regression**: 0.62 Accuracy
- **Decision Tree Classifier**: 0.59 Accuracy
- **Random Forest**: 0.60 Accuracy
- **Support Vector Machine (SVM)**: 0.62 Accuracy
- **Naive Bayes**: 0.62 Accuracy

Logistic Regression, Naive Bayes and SVM achieved the highest accuracy at **0.62**, making it the best-performing model. Decision Tree Classifier had the lowest accuracy at **0.59**. Random Forest, performed similarly with accuracy of **0.60**.