In [1]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
vaccine = pd.read_csv('h1n1_vaccine_prediction.csv')
vaccine.head()
```

Out[2]:

	unique_id	h1n1_worry	h1n1_awareness	antiviral_medication	contact_avoidance	bought_face
0	0	1.0	0.0	0.0	0.0	
1	1	3.0	2.0	0.0	1.0	
2	2	1.0	1.0	0.0	1.0	
3	3	1.0	1.0	0.0	1.0	
4	4	2.0	1.0	0.0	1.0	

5 rows × 34 columns

In [3]:
```
vaccine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 34 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   unique_id                    26707 non-null  int64  
 1   h1n1_worry                   26615 non-null  float64
 2   h1n1_awareness               26591 non-null  float64
 3   antiviral_medication         26636 non-null  float64
 4   contact_avoidance            26499 non-null  float64
 5   bought_face_mask             26688 non-null  float64
 6   wash_hands_frequently        26665 non-null  float64
 7   avoid_large_gatherings       26620 non-null  float64
 8   reduced_outside_home_cont    26625 non-null  float64
 9   avoid_touch_face             26579 non-null  float64
 10  dr_recc_h1n1_vacc            24547 non-null  float64
 11  dr_recc_seasonal_vacc        24547 non-null  float64
 12  chronic_medic_condition      25736 non-null  float64
 13  cont_child_undr_6_mnths      25887 non-null  float64
 14  is_health_worker             25903 non-null  float64
 15  has_health_insur             14433 non-null  float64
 16  is_h1n1_vacc_effective       26316 non-null  float64
 17  is_h1n1_risky                26319 non-null  float64
 18  sick_from_h1n1_vacc          26312 non-null  float64
 19  is_seas_vacc_effective       26245 non-null  float64
 20  is_seas_risky                26193 non-null  float64
 21  sick_from_seas_vacc          26170 non-null  float64
 22  age_bracket                  26707 non-null  object 
 23  qualification                25300 non-null  object 
 24  race                         26707 non-null  object 
 25  sex                          26707 non-null  object 
 26  income_level                 22284 non-null  object 
 27  marital_status               25299 non-null  object 
 28  housing_status               24665 non-null  object 
 29  employment                   25244 non-null  object 
 30  census_msa                   26707 non-null  object 
 31  no_of_adults                 26458 non-null  float64
 32  no_of_children               26458 non-null  float64
 33  h1n1_vaccine                 26707 non-null  int64  
dtypes: float64(23), int64(2), object(9)
memory usage: 6.9+ MB
```

In [4]:
```
vaccine.shape
```

Out[4]:
```
(26707, 34)
```

In [5]:
```
vaccine.dtypes.value_counts()
```

Out[5]:
```
float64    23
object      9
int64       2
Name: count, dtype: int64
```

In [6]:
```
vaccine.columns
```

Out[6]:
```
Index(['unique_id', 'h1n1_worry', 'h1n1_awareness', 'antiviral_medication',
       'contact_avoidance', 'bought_face_mask', 'wash_hands_frequently',
       'avoid_large_gatherings', 'reduced_outside_home_cont',
       'avoid_touch_face', 'dr_recc_h1n1_vacc', 'dr_recc_seasonal_vacc',
       'chronic_medic_condition', 'cont_child_undr_6_mnths',
       'is_health_worker', 'has_health_insur', 'is_h1n1_vacc_effective',
       'is_h1n1_risky', 'sick_from_h1n1_vacc', 'is_seas_vacc_effective',
       'is_seas_risky', 'sick_from_seas_vacc', 'age_bracket', 'qualificatio
n',
       'race', 'sex', 'income_level', 'marital_status', 'housing_status',
       'employment', 'census_msa', 'no_of_adults', 'no_of_children',
       'h1n1_vaccine'],
      dtype='object')
```

In [7]:
```
vaccine.drop(['unique_id'], inplace = True, axis = 1)
```

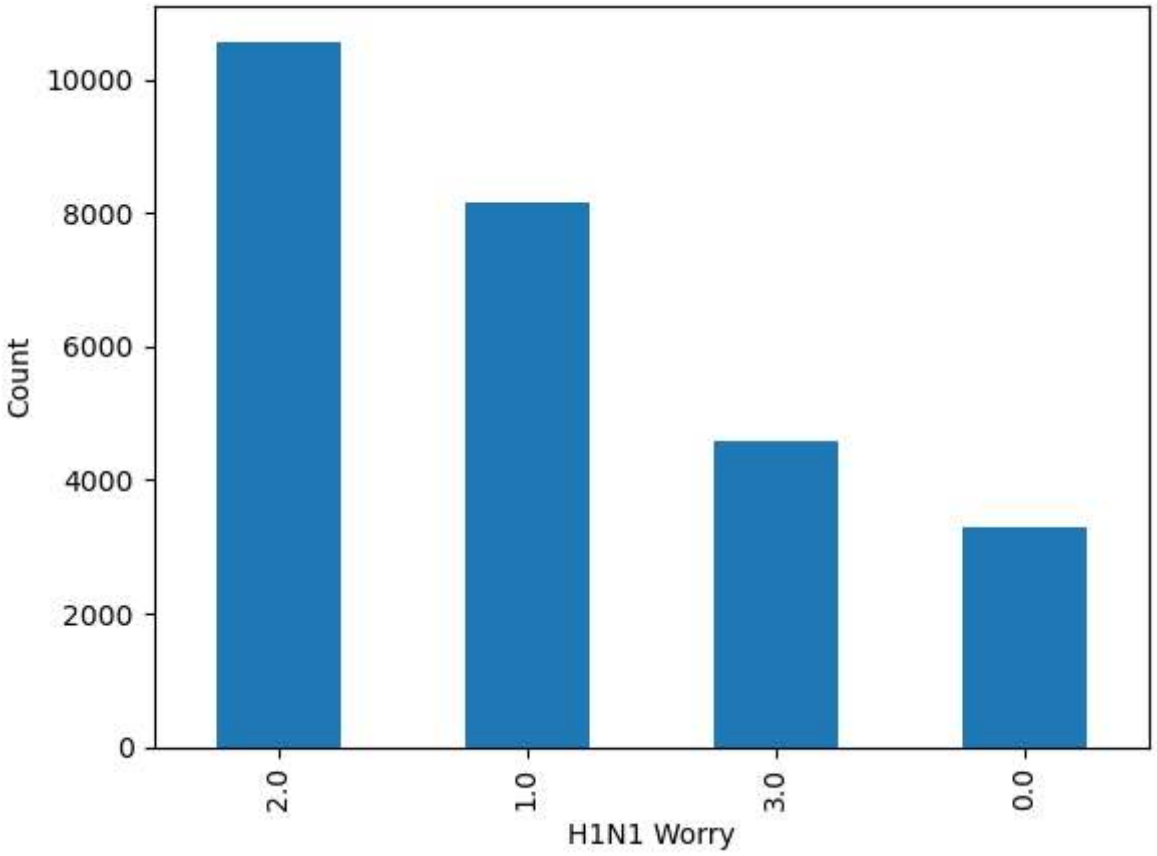
In [8]:
```
vaccine.isnull().sum()
```

Out[8]:
```
h1n1_worry                     92
h1n1_awareness                116
antiviral_medication           71
contact_avoidance             208
bought_face_mask               19
wash_hands_frequently          42
avoid_large_gatherings         87
reduced_outside_home_cont      82
avoid_touch_face              128
dr_recc_h1n1_vacc            2160
dr_recc_seasonal_vacc        2160
chronic_medic_condition       971
cont_child_undr_6_mnths       820
is_health_worker              804
has_health_insur            12274
is_h1n1_vacc_effective        391
is_h1n1_risky                 388
sick_from_h1n1_vacc           395
is_seas_vacc_effective        462
is_seas_risky                 514
sick_from_seas_vacc           537
age_bracket                     0
qualification                1407
race                            0
sex                             0
income_level                 4423
marital_status               1408
housing_status               2042
employment                   1463
census_msa                      0
no_of_adults                  249
no_of_children                249
h1n1_vaccine                    0
dtype: int64
```

In [9]:
```
vaccine['h1n1_worry'].unique()
# 0=Not worried at all, 1=Not very worried, 2=Somewhat worried, 3=Very worried
```

Out[9]:
```
array([ 1.,  3.,  2.,  0., nan])
```

In [10]:
```
vaccine['h1n1_worry'].value_counts().plot(kind = 'bar')
plt.xlabel('H1N1 Worry')
plt.ylabel('Count')
plt.show()
```



In [11]:
```
display(vaccine['h1n1_worry'].mode(), vaccine['h1n1_worry'].median())
```

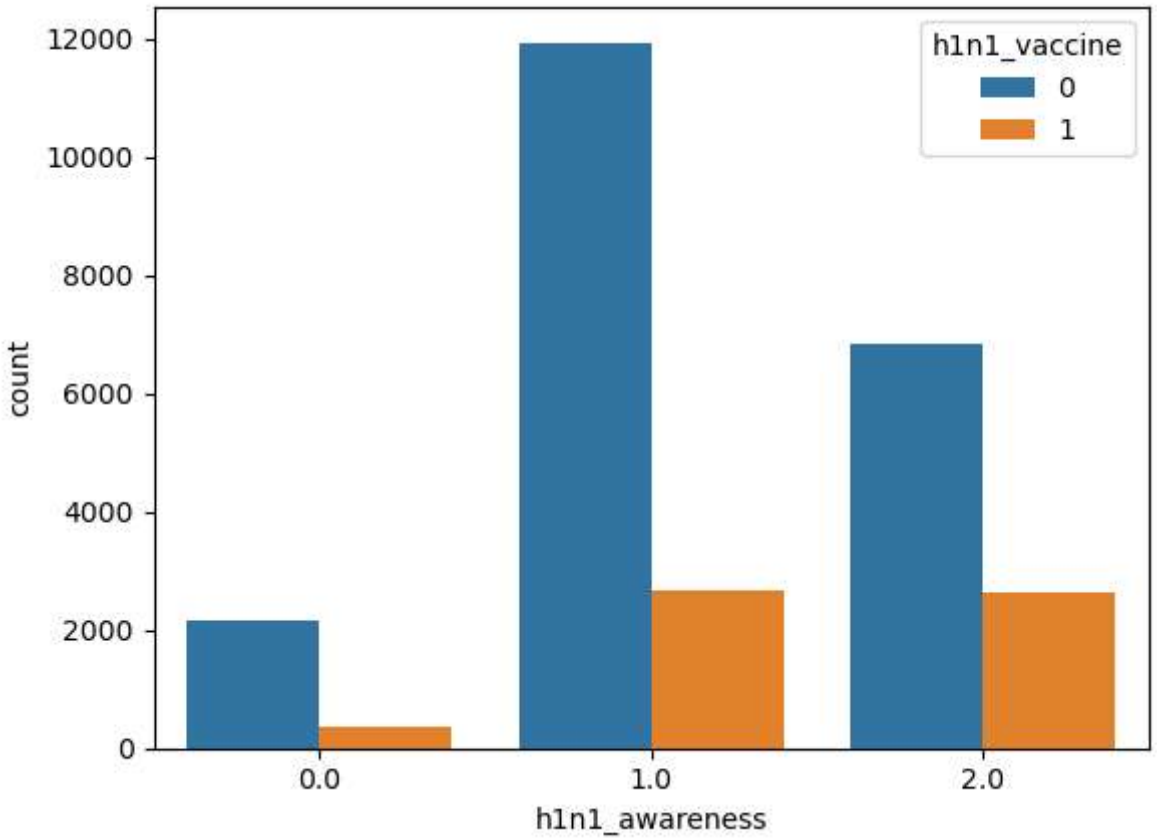
```
0    2.0
Name: h1n1_worry, dtype: float64

2.0
```

In [12]:
```
# Filling 92 missing values by mode
vaccine['h1n1_worry'].fillna(vaccine['h1n1_worry'].mode()[0], inplace = True)
```

In [13]:
```
vaccine['h1n1_awareness'].unique()
# 0=No knowledge, 1=Little knowledge, 2=good knowledge
```

Out[13]:
```
array([ 0.,  2.,  1., nan])
```

In [14]:
```
sns.countplot(x = 'h1n1_awareness', data = vaccine, hue = 'h1n1_vaccine')
plt.show()
```



In [15]:
```
vaccine['h1n1_awareness'].mode()
```

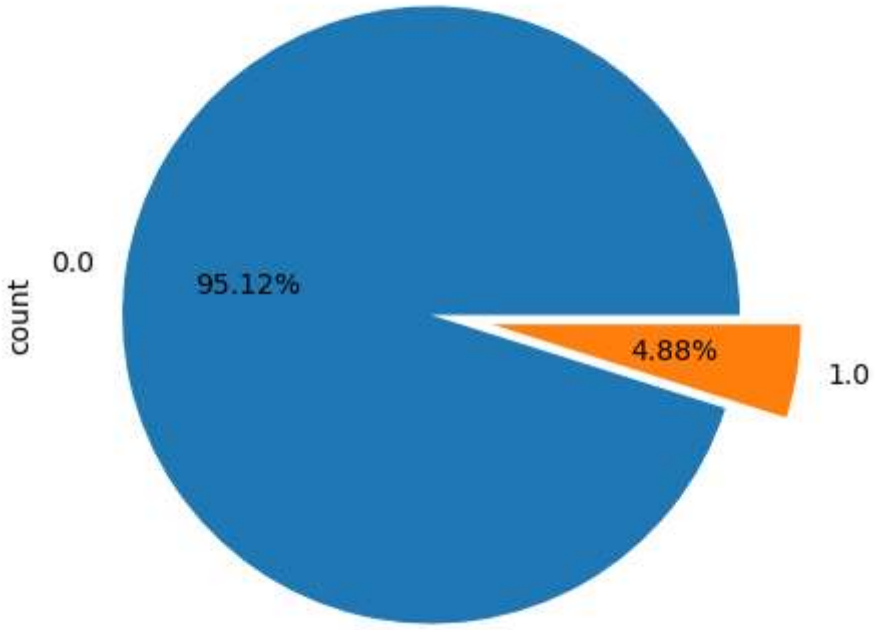
Out[15]:
```
0    1.0
Name: h1n1_awareness, dtype: float64
```

In [16]:
```
# Filling 192 missing values by mode
vaccine['h1n1_awareness'].fillna(vaccine['h1n1_awareness'].mode()[0], inplace
```

In [17]:
```
vaccine['antiviral_medication'].unique()
# 0=no, 1=yes
```

Out[17]:
```
array([ 0.,  1., nan])
```

In [18]:
```
vaccine['antiviral_medication'].value_counts().plot(kind = 'pie', autopct = '%
plt.show()
```



In [19]:
```
display(vaccine['antiviral_medication'].mode())
# missing 71 missing values by mode
vaccine['antiviral_medication'].fillna(vaccine['antiviral_medication'].mode()[
```

```
0    0.0
Name: antiviral_medication, dtype: float64
```

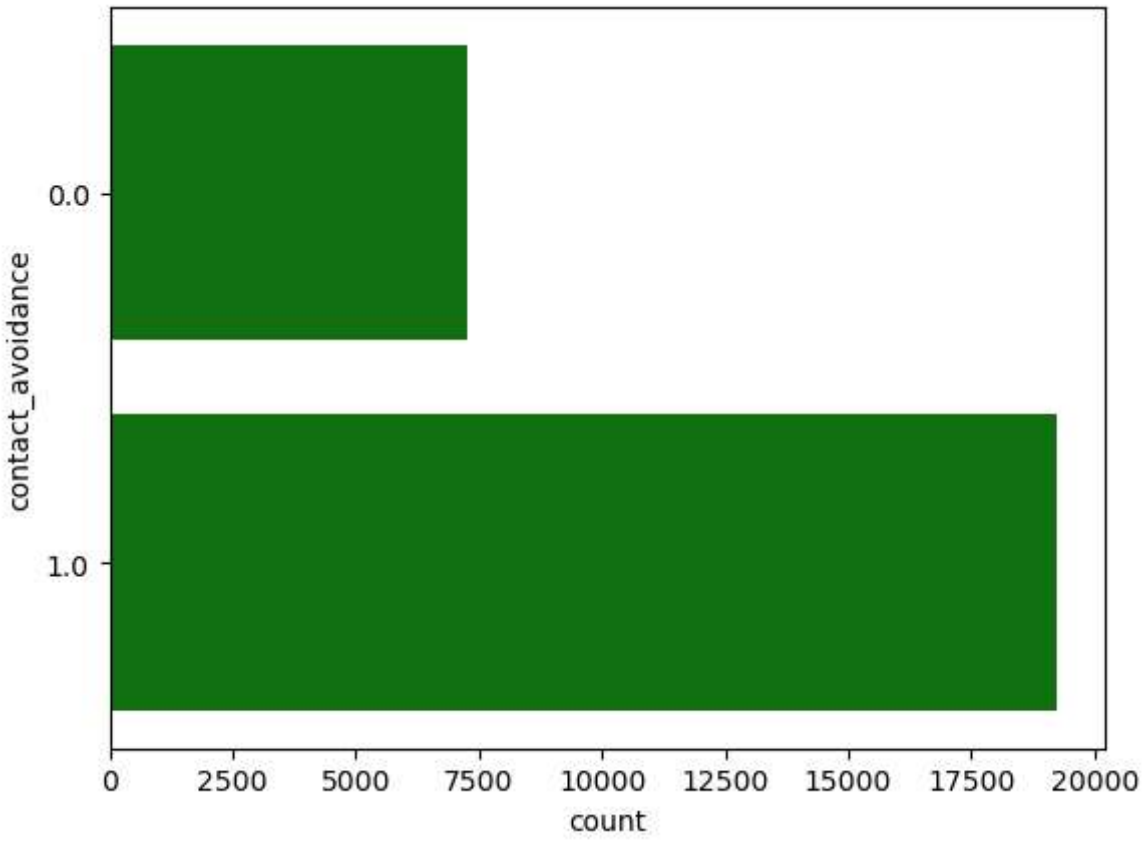
In [20]:
```
vaccine['contact_avoidance'].unique()
# 0=no, 1=yes
```

Out[20]:
```
array([ 0.,  1., nan])
```

In [21]:
```
vaccine['contact_avoidance'].value_counts()
```

Out[21]:
```
contact_avoidance
1.0    19228
0.0     7271
Name: count, dtype: int64
```

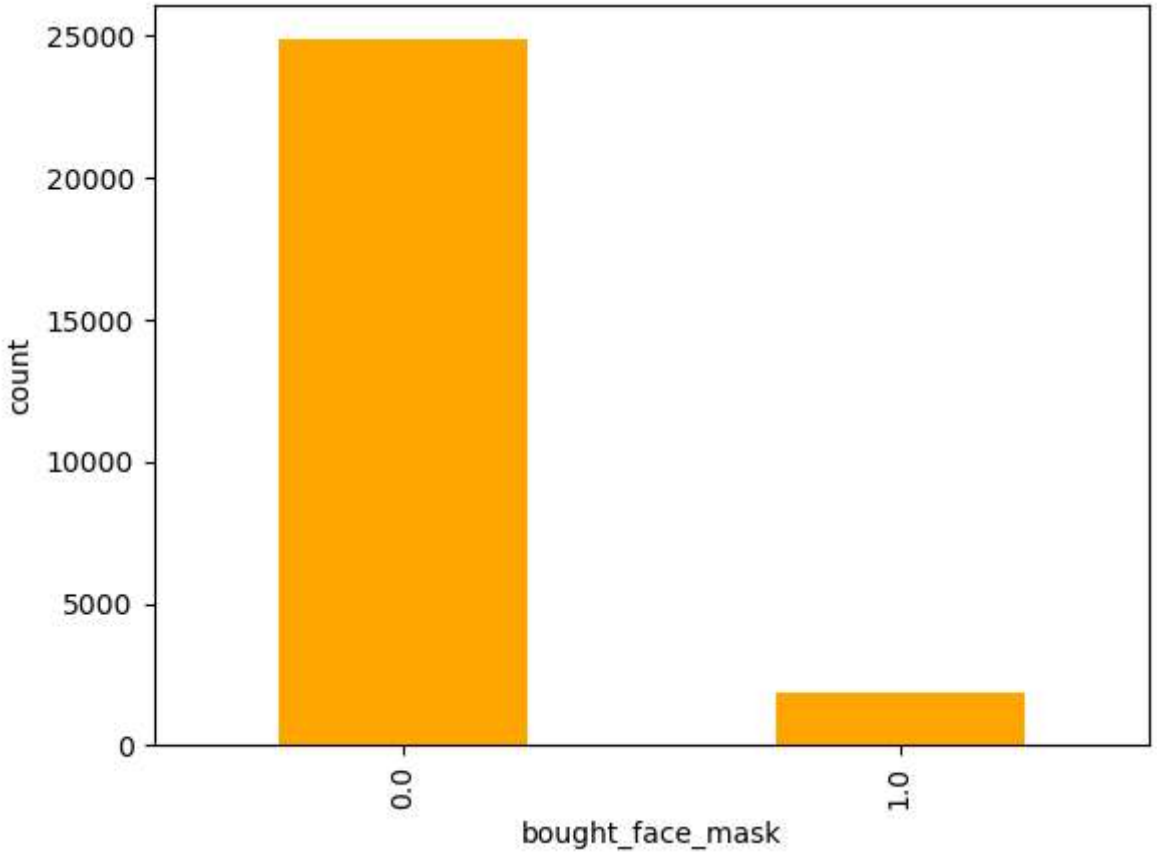
In [22]:
```
sns.countplot(y = 'contact_avoidance', data = vaccine, color = 'green')
plt.show()
```



In [23]:
```
vaccine['bought_face_mask'].unique()
# 0=no, 1=yes
```

Out[23]:
```
array([ 0.,  1., nan])
```

In [24]:
```
vaccine['bought_face_mask'].value_counts().plot(kind = 'bar', color = 'orange'
plt.xlabel('bought_face_mask')
plt.ylabel('count')
plt.show()
```

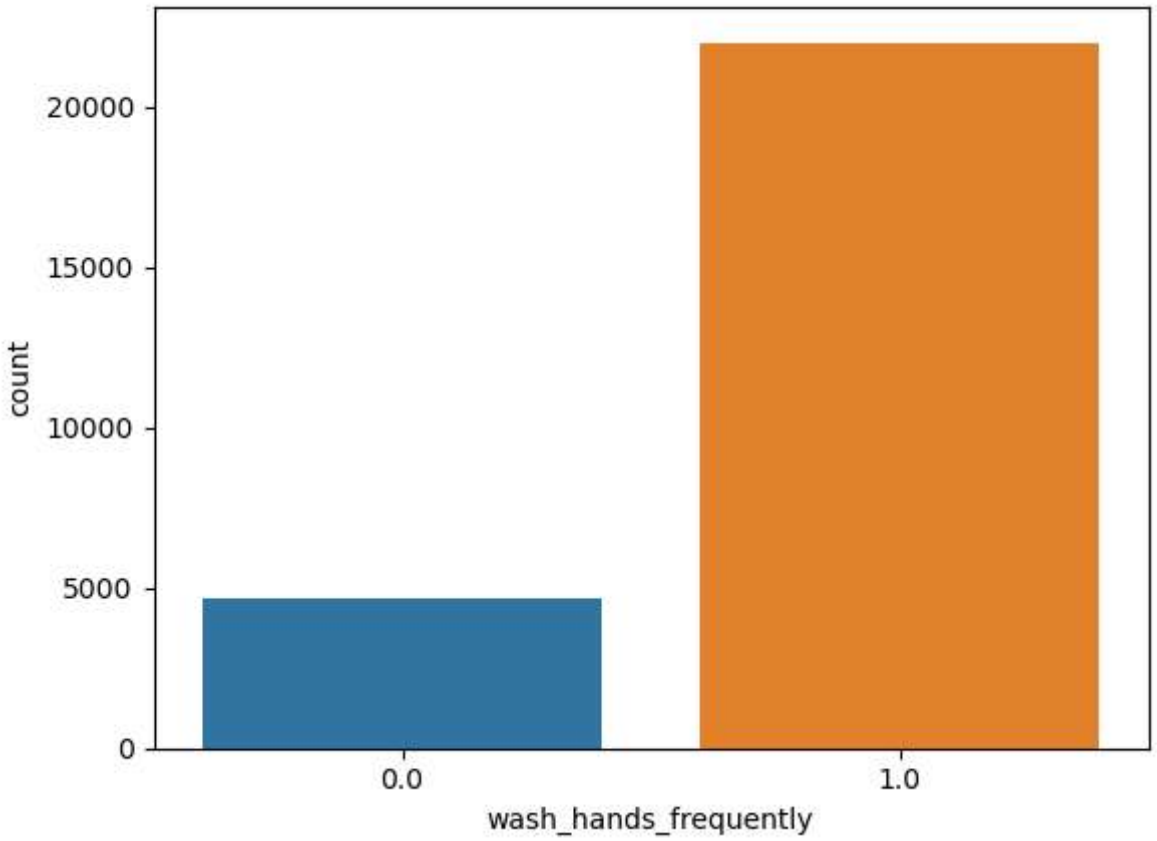


In [25]:
```
# filling 19 missing values by mode
vaccine['bought_face_mask'].fillna(vaccine['bought_face_mask'].mode()[0], inpl
```

In [26]:
```
vaccine['wash_hands_frequently'].unique()
# 0-Washes hands frequently, 1=uses hand sanitizer -
```

Out[26]:
```
array([ 0.,  1., nan])
```

In [27]:
```
sns.countplot(x = 'wash_hands_frequently', data = vaccine)
plt.show()
```



In [28]:
```
# filling 42 missing values by mode
vaccine['wash_hands_frequently'].fillna(vaccine['wash_hands_frequently'].mode(
```

In [29]:
```
vaccine['avoid_large_gatherings'].unique()
# 0=no, 1=yes
```

Out[29]:
```
array([ 0.,  1., nan])
```

In [30]:
```
vaccine['avoid_large_gatherings'].value_counts()
```

Out[30]:
```
avoid_large_gatherings
0.0    17073
1.0     9547
Name: count, dtype: int64
```

In [31]:
```
# filling 87 missing values by mode
vaccine['avoid_large_gatherings'].fillna(vaccine['avoid_large_gatherings'].mod
```

In [32]:
```
vaccine['reduced_outside_home_cont'].unique()
# 0=no, 1=yes
```

Out[32]:
```
array([ 1.,  0., nan])
```

In [33]:
```
vaccine['reduced_outside_home_cont'].value_counts()
```

Out[33]:
```
reduced_outside_home_cont
0.0    17644
1.0     8981
Name: count, dtype: int64
```

In [34]:
```
# filling 82 missing values by mode
vaccine['reduced_outside_home_cont'].fillna(vaccine['reduced_outside_home_cont
```

In [35]:
```
vaccine['avoid_touch_face'].unique()
# 0=no, 1=yes
```

Out[35]:
```
array([ 1.,  0., nan])
```

In [36]:
```
vaccine['avoid_touch_face'].value_counts()
```

Out[36]:
```
avoid_touch_face
1.0    18001
0.0     8578
Name: count, dtype: int64
```

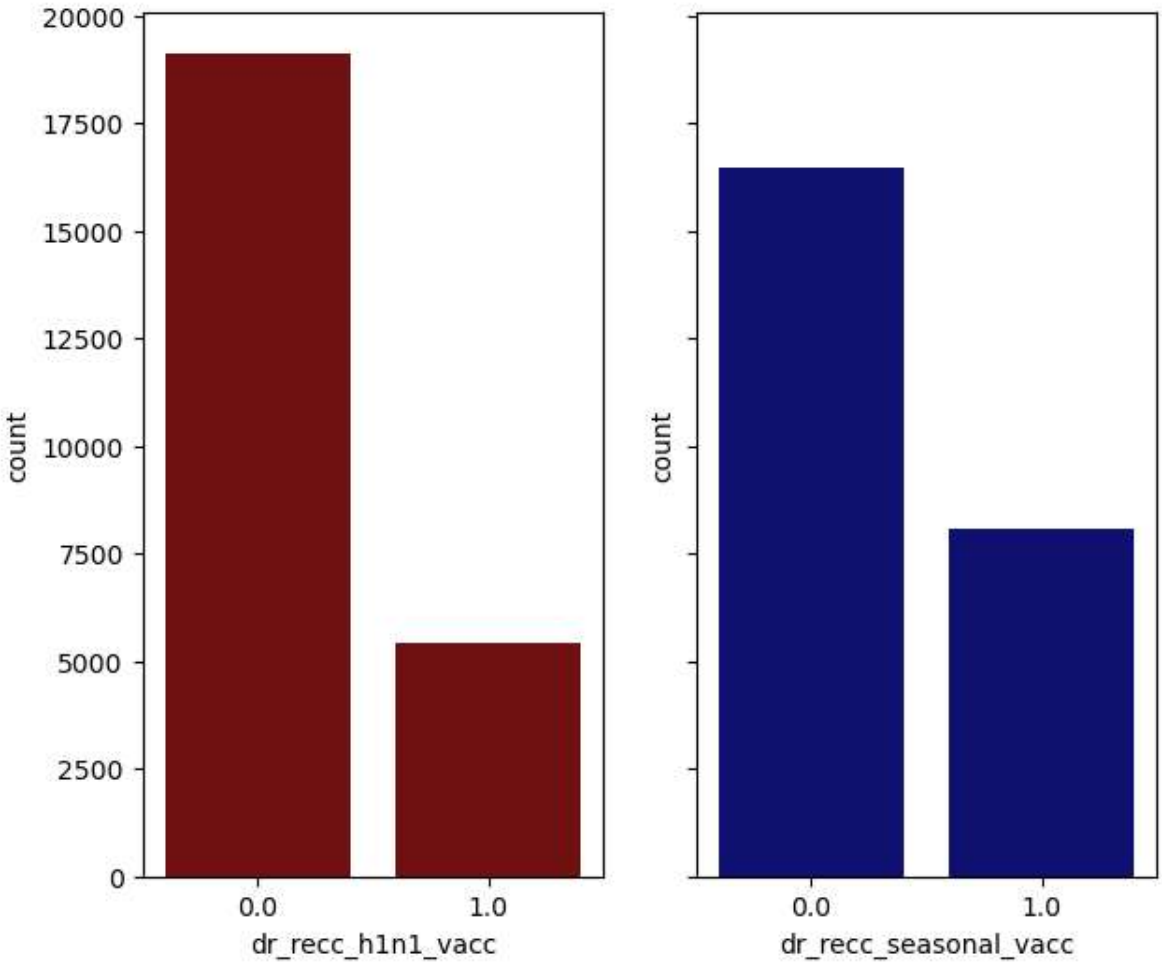
In [37]:
```
# filling 128 missing values by mode
vaccine['avoid_touch_face'].fillna(vaccine['avoid_touch_face'].mode()[0], inpl
```

In [38]:
```
display(vaccine['dr_recc_h1n1_vacc'].unique(), vaccine['dr_recc_seasonal_vacc'
# 0=no, 1=yes
```

```
array([ 0., nan,  1.])
```

```
array([ 0., nan,  1.])
```

In [39]:
```
fig, ax = plt.subplots(1,2, figsize = [7,6], sharey = True )
sns.countplot(x = 'dr_recc_h1n1_vacc', data = vaccine, color = 'maroon', ax=ax
sns.countplot(x = 'dr_recc_seasonal_vacc', data = vaccine, color = 'navy', ax=
plt.show()
```



In [40]:
```
# filling 2160 missing values by mode
vaccine['dr_recc_h1n1_vacc'].fillna(vaccine['dr_recc_h1n1_vacc'].mode()[0], in
vaccine['dr_recc_seasonal_vacc'].fillna(vaccine['dr_recc_seasonal_vacc'].mode(
```

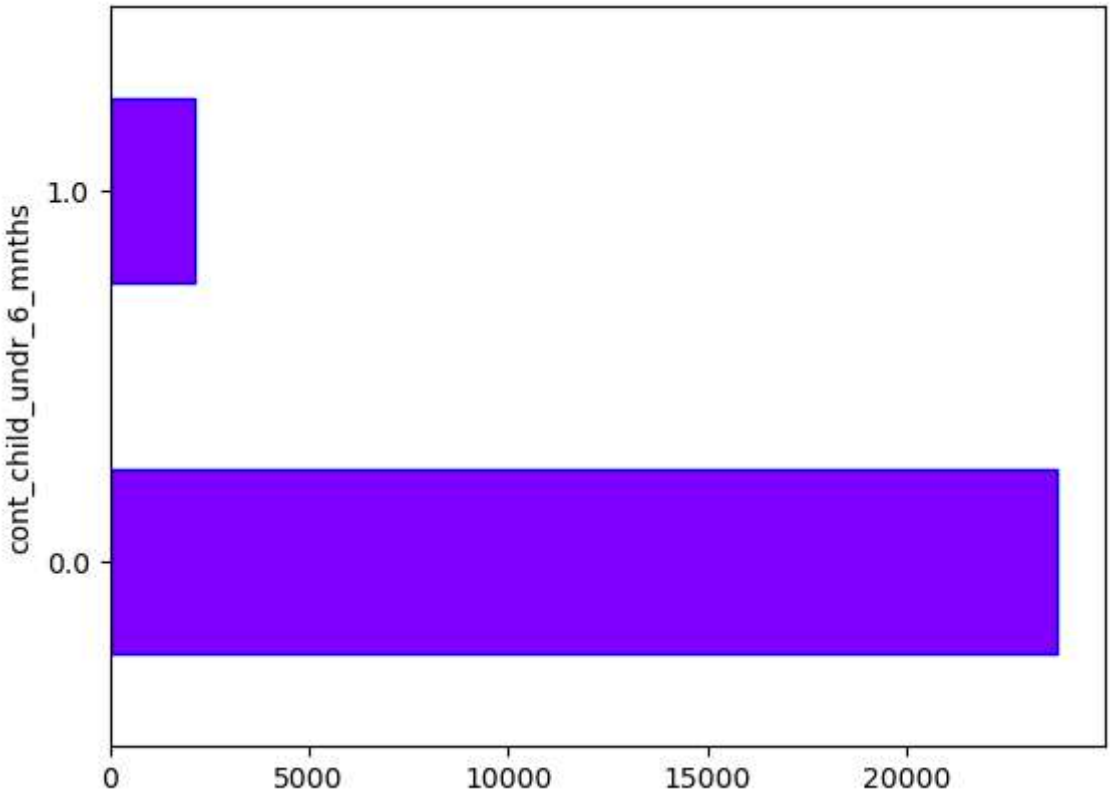
In [41]:
```
vaccine['chronic_medic_condition'].value_counts()
```

Out[41]:
```
chronic_medic_condition
0.0    18446
1.0     7290
Name: count, dtype: int64
```

In [42]:
```
# filling 971 missing values by mode
vaccine['chronic_medic_condition'].fillna(vaccine['chronic_medic_condition'].m
```

In [43]:
```
vaccine['cont_child_undr_6_mnths'].value_counts().plot(kind = 'barh', cmap = '
```

Out[43]: <Axes: ylabel='cont_child_undr_6_mnths'>



In [44]:
```
# filling 820 missing values by mode
vaccine['cont_child_undr_6_mnths'].fillna(vaccine['cont_child_undr_6_mnths'].m
```

In [45]:
```
vaccine['is_health_worker'].value_counts()
```

Out[45]:
```
is_health_worker
0.0    23004
1.0     2899
Name: count, dtype: int64
```

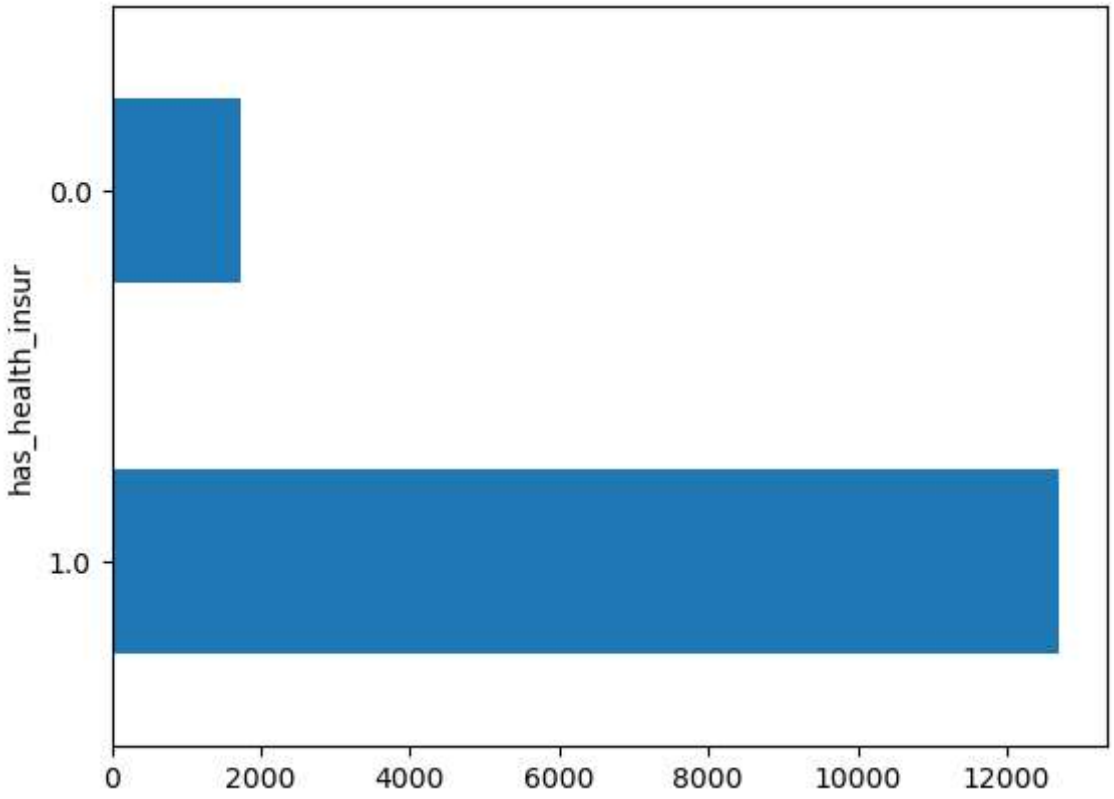
In [46]:
```
# filling 804 missing values by mode
vaccine['is_health_worker'].fillna(vaccine['is_health_worker'].mode()[0], inpl
```

In [47]:
```
vaccine['has_health_insur'].unique()
# 0=no, 1=yes
```

Out[47]:
```
array([ 1., nan,  0.])
```

In [48]:
```
vaccine['has_health_insur'].value_counts().plot(kind = 'barh')
```

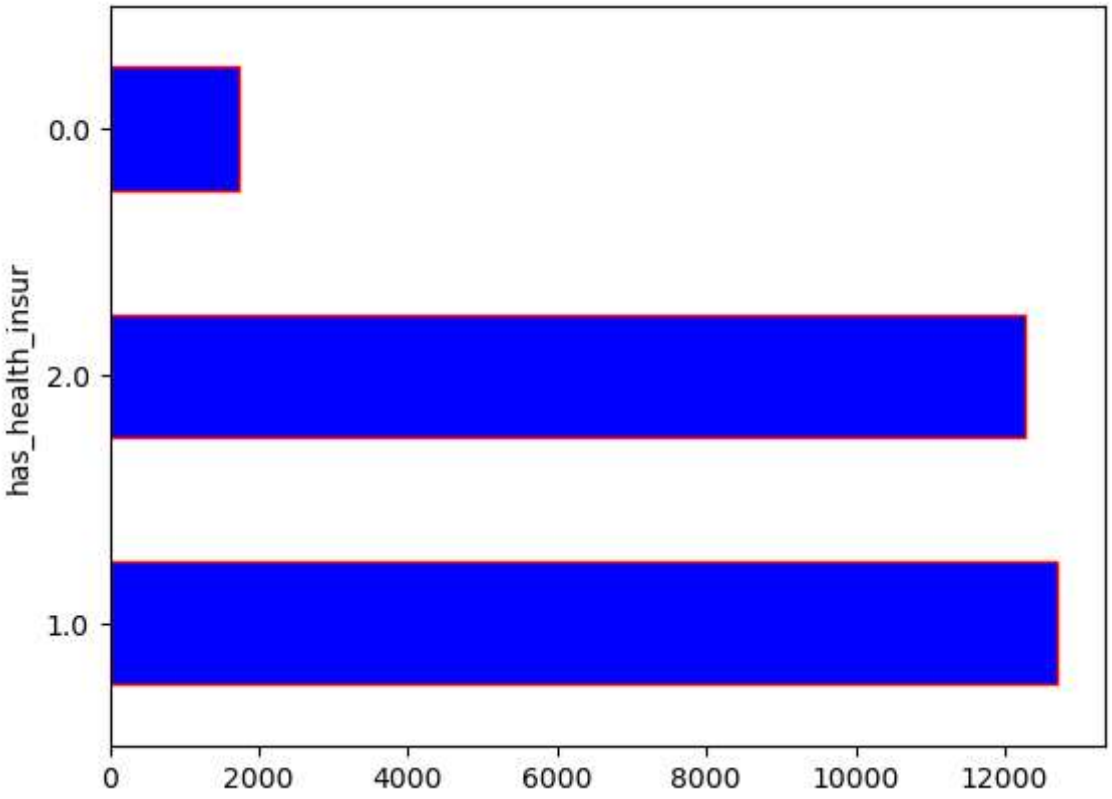
Out[48]: <Axes: ylabel='has_health_insur'>



In [49]:
```
vaccine['has_health_insur'].fillna(2.0, inplace = True)
```

In [50]:
```
vaccine['has_health_insur'].value_counts().plot(kind = 'barh', color = 'b', ed
```

Out[50]: <Axes: ylabel='has_health_insur'>

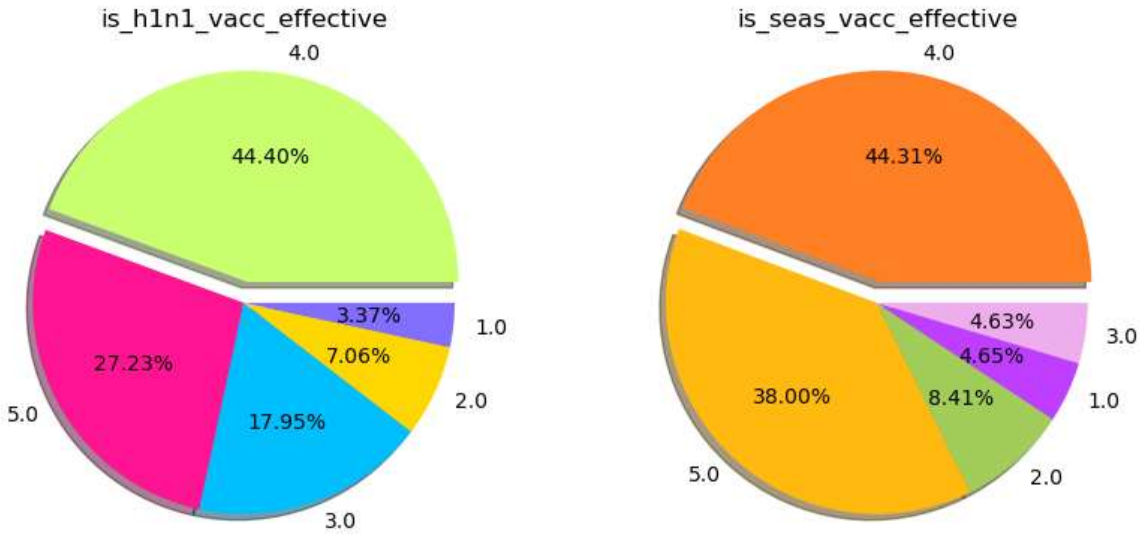


In [51]:
```
display(vaccine['is_h1n1_vacc_effective'].unique(), vaccine['is_seas_vacc_effe
# 1=Thinks not effective at all, 2=Thinks it is not very effective, 3=Doesn't
# 4=Thinks it is somewhat effective, 5=Thinks it is highly effective
```

```
array([ 3.,  5.,  4.,  2.,  1., nan])
```

```
array([ 2.,  4.,  5.,  3.,  1., nan])
```

In [52]:
```
colors = ['#CAFF70', '#FF1493', '#00BFFF', '#FFD700', '#836FFF']
colors1 = ['#FF7F24', '#FFB90F', '#A2CD5A', '#BF3EFF', '#EEAEEE']
fig, (ax1,ax2) = plt.subplots(1,2, figsize = [10,10])

ax1.pie(vaccine['is_h1n1_vacc_effective'].value_counts(), labels = vaccine['is
        autopct = '%0.2f%%', explode= [0.1,0,0,0,0], colors = colors, shadow =

ax2.pie(vaccine['is_seas_vacc_effective'].value_counts(), labels = vaccine['is
        autopct = '%0.2f%%', explode= [0.1,0,0,0,0], colors = colors1, shadow

ax1.set_title('is_h1n1_vacc_effective')
ax2.set_title('is_seas_vacc_effective')

plt.show()
```

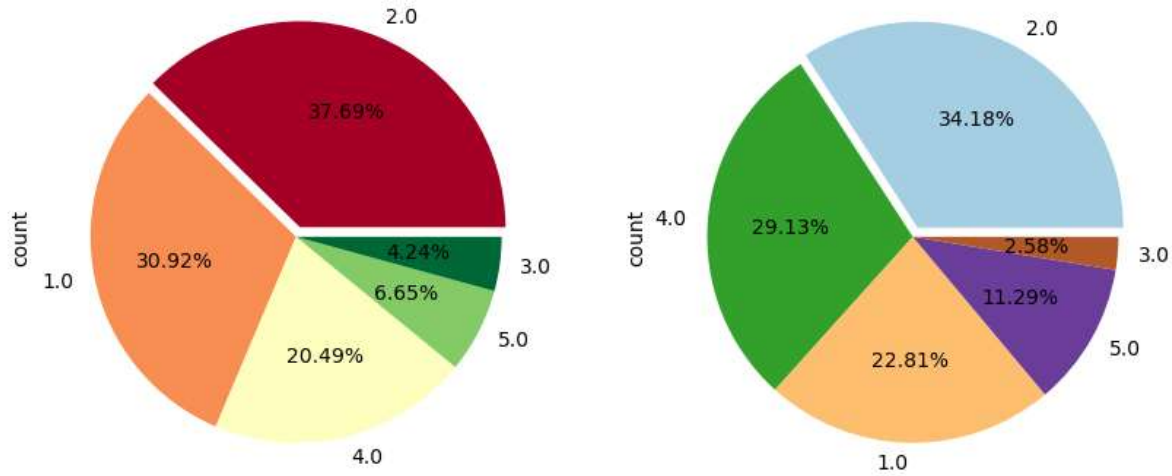


In [53]:
```
# filling 391 and 462 missing values respectively by mode
vaccine['is_h1n1_vacc_effective'].fillna(vaccine['is_h1n1_vacc_effective'].mod
vaccine['is_seas_vacc_effective'].fillna(vaccine['is_seas_vacc_effective'].mod
```

In [54]:
```
display(vaccine['is_h1n1_risky'].unique(), vaccine['is_seas_risky'].unique())
# 1=Thinks it is not very low risk, 2=Thinks it is somewhat low risk, 3=don't
# 4=Thinks it is a somewhat high risk, 5=Thinks it is very highly risky
```

```
array([ 1.,  4.,  3.,  2.,  5., nan])

array([ 1.,  2.,  4.,  3.,  5., nan])
```

In [55]:
```
fig, ax = plt.subplots(1, 2, figsize = [10,10])
vaccine['is_h1n1_risky'].value_counts().plot(kind = 'pie', autopct = '%0.2f%%'
vaccine['is_seas_risky'].value_counts().plot(kind = 'pie', autopct = '%0.2f%%'
plt.show()
```



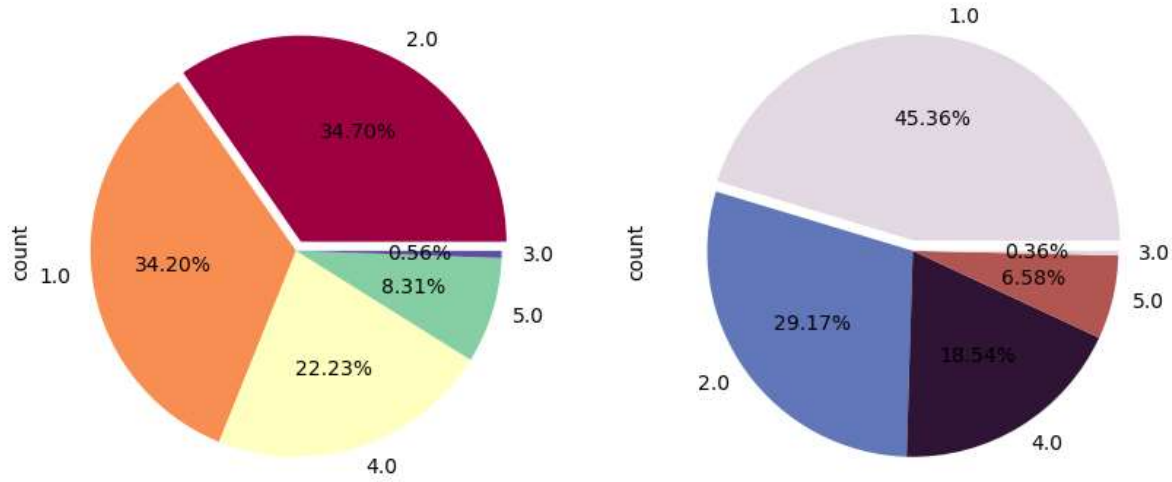
In [56]:
```
# filling 388 and 514 missing values respectively by mode
vaccine['is_h1n1_risky'].fillna(vaccine['is_h1n1_risky'].mode()[0], inplace =
vaccine['is_seas_risky'].fillna(vaccine['is_seas_risky'].mode()[0], inplace =
```

In [57]:
```
display(vaccine['sick_from_h1n1_vacc'].unique(), vaccine['sick_from_seas_vacc'
# 1=Respondent not worried at all, 2=Respondent is not very worried, 3=Doesn't
# 5Respondent is very worried
```

```
array([ 2.,  4.,  1.,  5.,  3., nan])
```

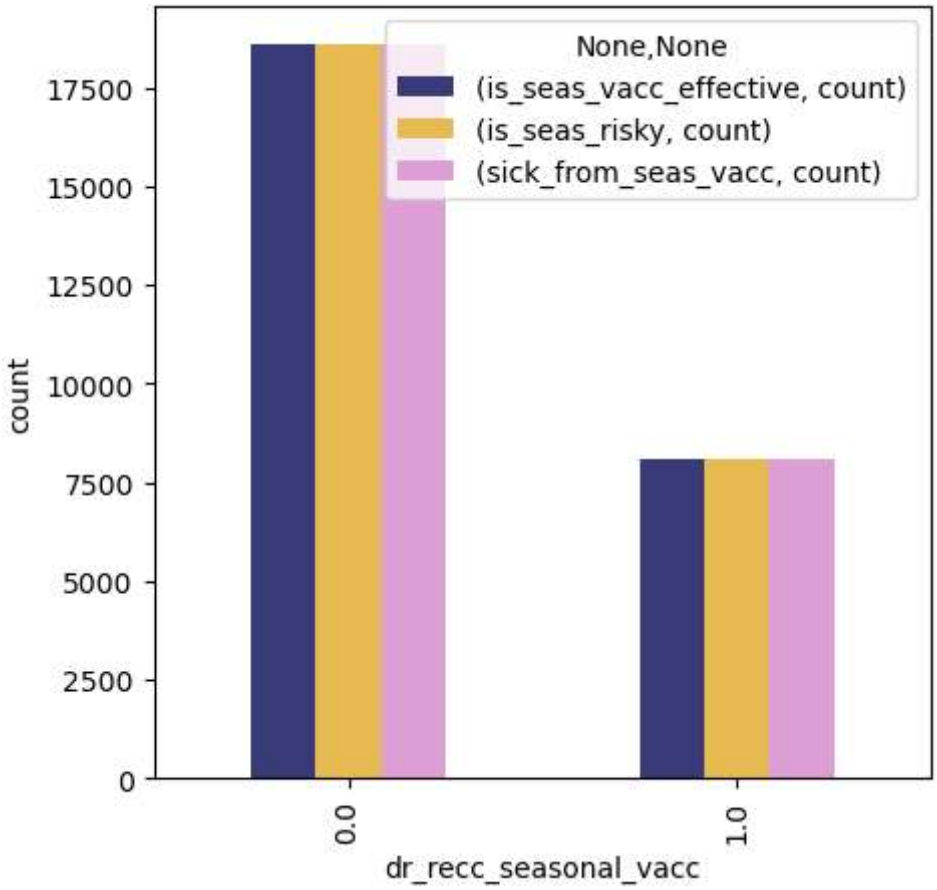
```
array([ 2.,  4.,  1.,  5., nan,  3.])
```

In [58]:
```
fig, ax = plt.subplots(1,2, figsize = [10,10])
vaccine['sick_from_h1n1_vacc'].value_counts().plot(kind = 'pie', autopct = '%0
vaccine['sick_from_seas_vacc'].value_counts().plot(kind = 'pie', autopct = '%0
plt.show()
```

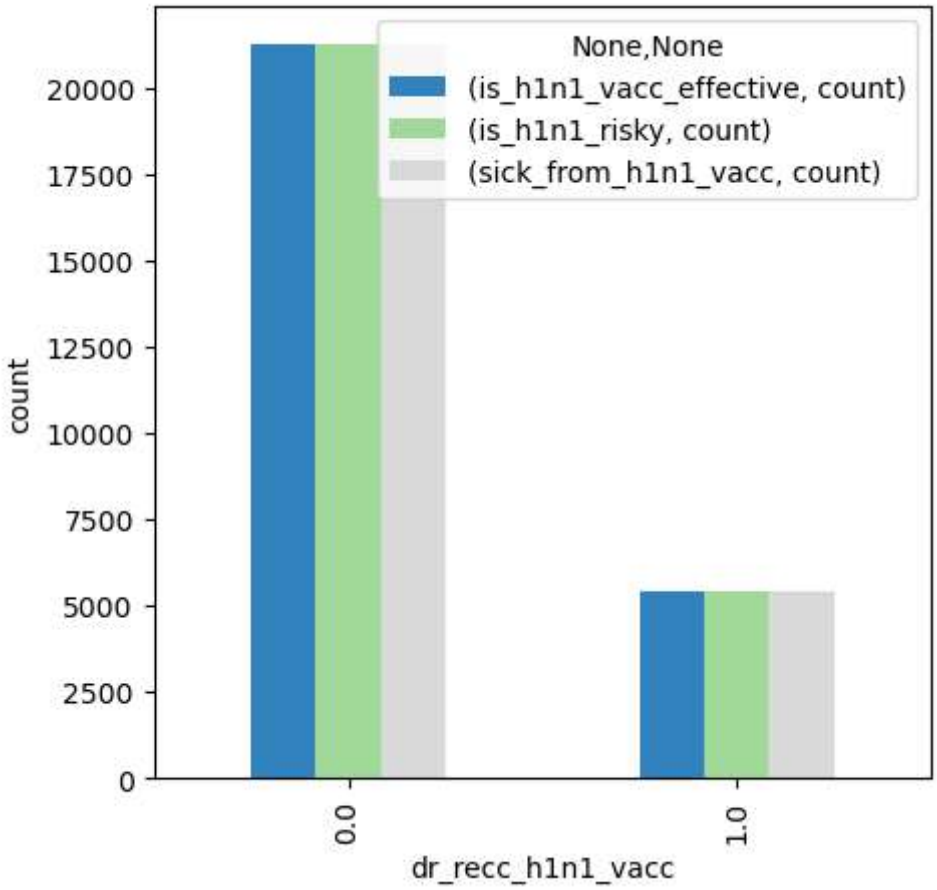


In [59]:
```
# filling 395 and 537 missing values respectively by mode
vaccine['sick_from_h1n1_vacc'].fillna(vaccine['sick_from_h1n1_vacc'].mode()[0]
vaccine['sick_from_seas_vacc'].fillna(vaccine['sick_from_seas_vacc'].mode()[0]
```

In [60]:
```
vaccine.groupby(['dr_recc_seasonal_vacc']).agg({'is_seas_vacc_effective' : ['c
                                                'is_seas_risky' :['count'],
                                                'sick_from_seas_vacc' : ['count']}).plot
plt.ylabel('count')
plt.show()
```



In [61]:
```
vaccine.groupby(['dr_recc_h1n1_vacc']).agg({'is_h1n1_vacc_effective' : ['count
                                            'is_h1n1_risky' :['count'],
                                            'sick_from_h1n1_vacc' : ['count']}).plot
plt.ylabel('count')
plt.show()
```



In [62]:
```
vaccine['qualification'].unique()
```

Out[62]:
```
array(['< 12 Years', '12 Years', 'College Graduate', 'Some College', nan],
      dtype=object)
```

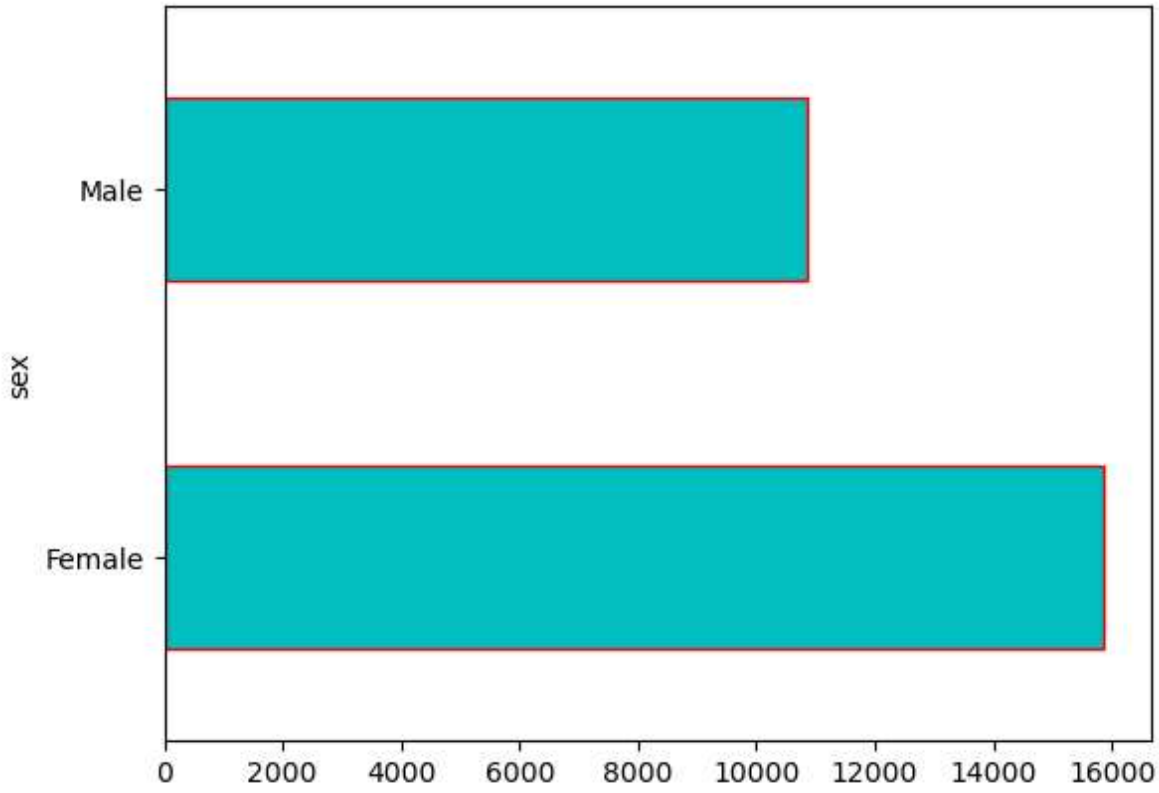
In [63]:
```
vaccine['qualification'].value_counts()
```

Out[63]:
```
qualification
College Graduate    10097
Some College         7043
12 Years             5797
< 12 Years           2363
Name: count, dtype: int64
```

In [64]:
```
# filling 1407 missing values by mode
vaccine['qualification'].fillna(vaccine['qualification'].mode()[0], inplace =
```

In [65]:
```
vaccine['sex'].value_counts().plot(kind = 'barh', color = 'c', edgecolor = 'r'
# no nan values
```

Out[65]:
```
<Axes: ylabel='sex'>
```

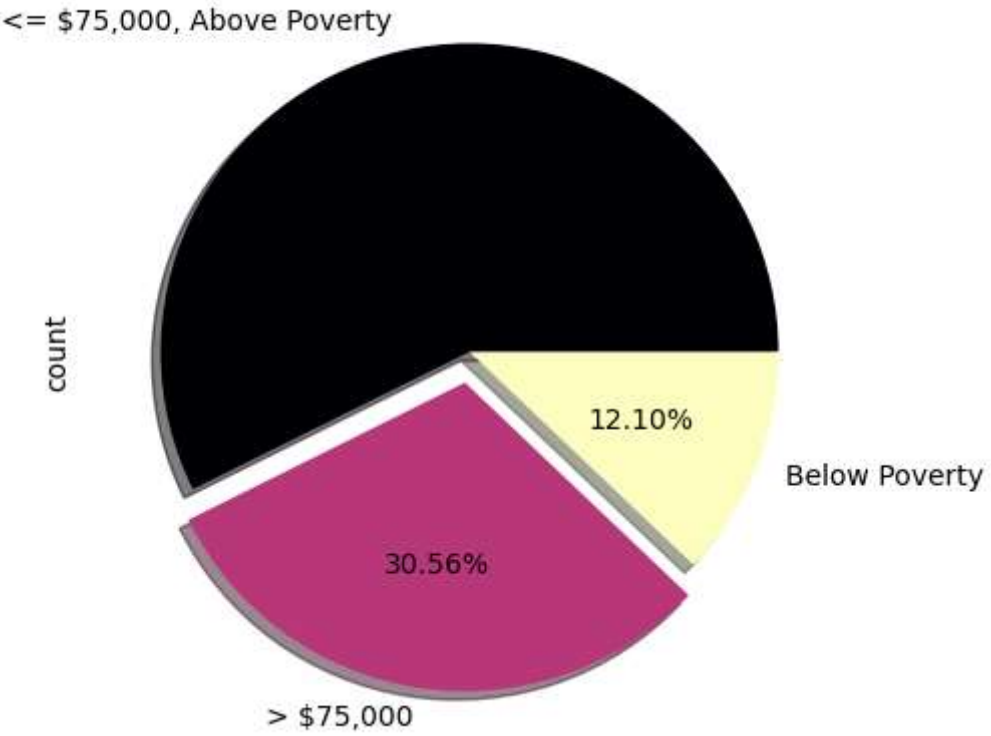


In [66]:
```
vaccine['income_level'].unique()
```

Out[66]:
```
array(['Below Poverty', '<= $75,000, Above Poverty', '> $75,000', nan],
      dtype=object)
```

In [67]:
```
vaccine['income_level'].value_counts().plot(kind = 'pie', autopct = '%0.2f%%',
                                            explode = [0,0.1,0], figsize = [5,
```

Out[67]: <Axes: ylabel='count'>



In [68]:
```
# no of missing values is 4423, better to create a new category as 'Unknown' a
vaccine['income_level'].fillna('Unknown', inplace = True)
```

In [69]:
```
vaccine['marital_status'].unique()
```

Out[69]: array(['Not Married', 'Married', nan], dtype=object)

In [70]:
```
vaccine['marital_status'].value_counts()
```

Out[70]:
```
marital_status
Married        13555
Not Married    11744
Name: count, dtype: int64
```

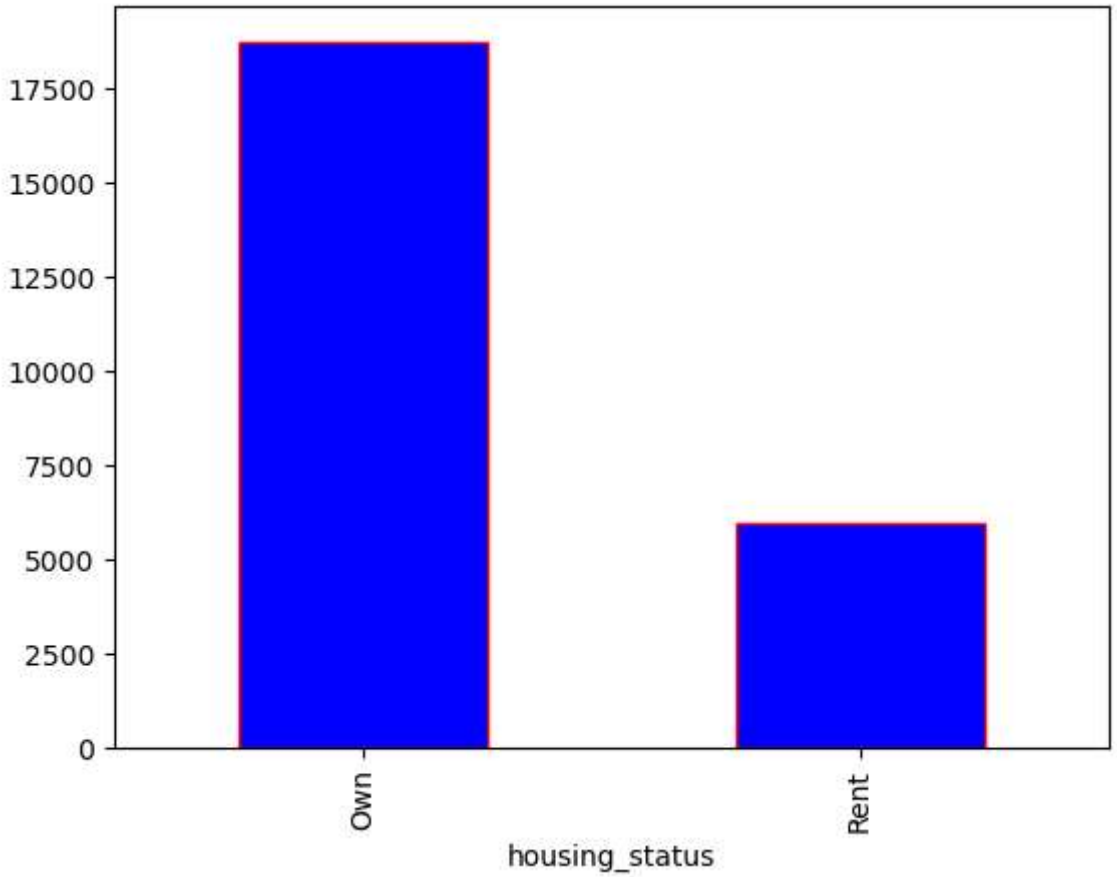
In [71]:
```
# filling 1408 missing values by mode
vaccine['marital_status'].fillna(vaccine['marital_status'].mode()[0], inplace
```

In [72]:
```
vaccine['housing_status'].unique()
```

Out[72]: array(['Own', 'Rent', nan], dtype=object)

In [73]:
```
vaccine['housing_status'].value_counts().plot(kind = 'bar', color = 'b', edgec
```

Out[73]:
```
<Axes: xlabel='housing_status'>
```



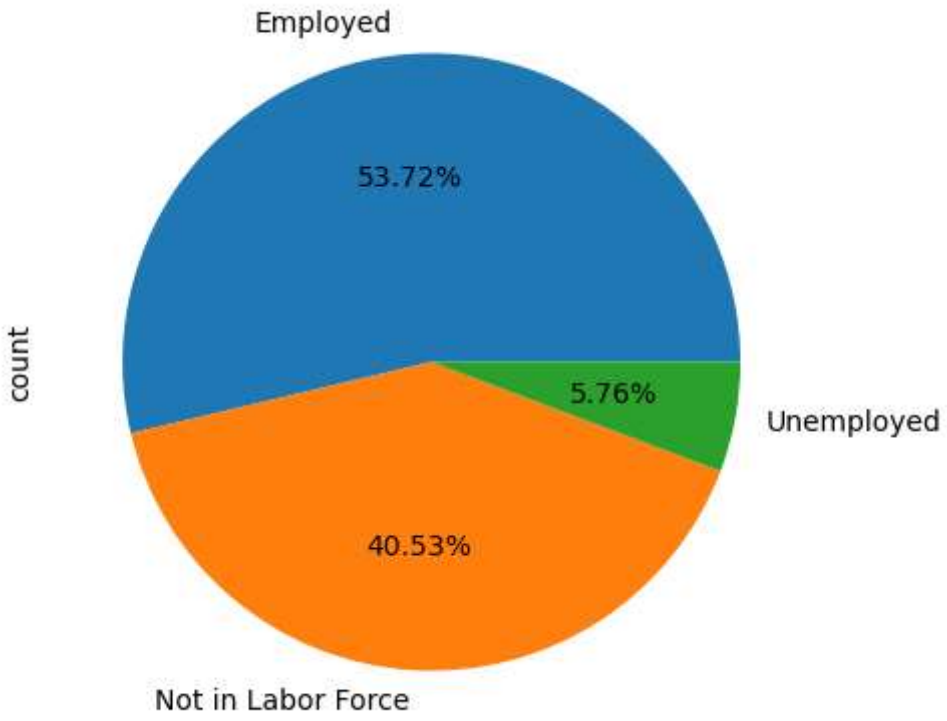
In [74]:
```
# filling 2402 missing values by mode
vaccine['housing_status'].fillna(vaccine['housing_status'].mode()[0], inplace
```

In [75]:
```
vaccine['employment'].unique()
```

Out[75]:
```
array(['Not in Labor Force', 'Employed', 'Unemployed', nan], dtype=object)
```

In [76]:
```
vaccine['employment'].value_counts().plot(kind = 'pie', autopct = '%0.2f%%', f
```

Out[76]: <Axes: ylabel='count'>

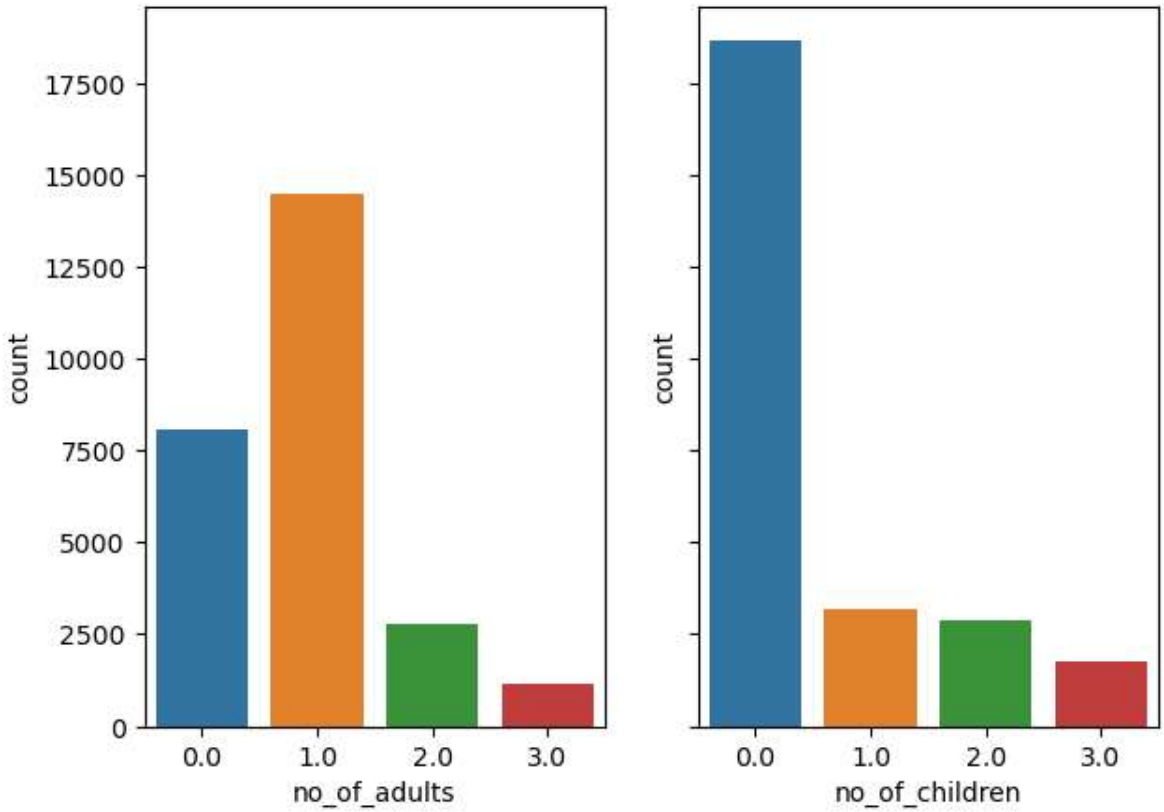


In [77]:
```
# filling 1463 missing values by mode
vaccine['employment'].fillna(vaccine['employment'].mode()[0], inplace = True)
```

In [78]:
```
display(vaccine['no_of_adults'].unique(), vaccine['no_of_children'].unique())
```

```
array([ 0.,  2.,  1.,  3., nan])

array([ 0.,  3.,  2.,  1., nan])
```

In [79]:
```
fig, ax = plt.subplots(1,2, figsize = [7,5], sharey = True)
sns.countplot( x = 'no_of_adults', data = vaccine, ax = ax[0])
sns.countplot( x = 'no_of_children', data = vaccine, ax = ax[1])
plt.show()
```



In [80]:
```
 #filling 249 nan values by mode
vaccine['no_of_adults'].fillna(vaccine['no_of_adults'].mode()[0], inplace = Tr
vaccine['no_of_children'].fillna(vaccine['no_of_children'].mode()[0], inplace
```

In [81]:
```
vaccine['age_bracket'].value_counts()
```

Out[81]:
```
age_bracket
65+ Years        6843
55 - 64 Years    5563
45 - 54 Years    5238
18 - 34 Years    5215
35 - 44 Years    3848
Name: count, dtype: int64
```

In [82]:
```
vaccine.isnull().sum()
```

Out[82]:
```
h1n1_worry                     0
h1n1_awareness                 0
antiviral_medication           0
contact_avoidance            208
bought_face_mask               0
wash_hands_frequently          0
avoid_large_gatherings         0
reduced_outside_home_cont      0
avoid_touch_face               0
dr_recc_h1n1_vacc              0
dr_recc_seasonal_vacc          0
chronic_medic_condition        0
cont_child_undr_6_mnths        0
is_health_worker               0
has_health_insur               0
is_h1n1_vacc_effective         0
is_h1n1_risky                  0
sick_from_h1n1_vacc            0
is_seas_vacc_effective         0
is_seas_risky                  0
sick_from_seas_vacc            0
age_bracket                    0
qualification                  0
race                           0
sex                            0
income_level                   0
marital_status                 0
housing_status                 0
employment                     0
census_msa                     0
no_of_adults                   0
no_of_children                 0
h1n1_vaccine                   0
dtype: int64
```

In [83]:
```
vaccine1 = vaccine.astype({'h1n1_worry' : str, 'h1n1_awareness': str, 'antivir
       'contact_avoidance': str, 'bought_face_mask': str, 'wash_hands_frequent
       'avoid_large_gatherings': str, 'reduced_outside_home_cont': str,
       'avoid_touch_face': str, 'dr_recc_h1n1_vacc': str, 'dr_recc_seasonal_va
       'chronic_medic_condition': str, 'cont_child_undr_6_mnths': str,
       'is_health_worker': str, 'has_health_insur': str, 'is_h1n1_vacc_effecti
       'is_h1n1_risky': str, 'sick_from_h1n1_vacc': str, 'is_seas_vacc_effecti
       'is_seas_risky': str, 'sick_from_seas_vacc': str, 'no_of_adults' : str,
        'h1n1_vaccine' : str})
vaccine1.dtypes.value_counts()
```

Out[83]:
```
object    33
Name: count, dtype: int64
```

In [84]:
```
vaccine_dummy = pd.get_dummies(vaccine1, drop_first= True )
```

In [85]:
```
vaccine_dummy.shape
```

Out[85]:
```
(26707, 71)
```

In [86]:
```
vaccine_dummy['h1n1_vaccine_1'].value_counts(normalize = True)
```

Out[86]:
```
h1n1_vaccine_1
False    0.787546
True     0.212454
Name: proportion, dtype: float64
```

In [87]:
```
y=vaccine_dummy['h1n1_vaccine_1']
x=vaccine_dummy.drop(['h1n1_vaccine_1'], axis=1)
```

In [88]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
vif_data['Var'] = x.columns
vif_data['VIF'] = [variance_inflation_factor(x.values, i) for i in range(len(x
```

In [89]:
```
vif_data.sort_values(by = ['VIF'], ascending = False)
```

Out[89]:

	Var	VIF
41	sick_from_seas_vacc_3.0	0.011573
29	sick_from_h1n1_vacc_3.0	0.007401
7	contact_avoidance_nan	0.005001
37	is_seas_risky_3.0	0.002132
33	is_seas_vacc_effective_3.0	0.001959
...	...	...
63	census_msa_Non-MSA	0.000229
12	avoid_touch_face_1.0	0.000225
62	census_msa_MSA, Principle City	0.000218
15	chronic_medic_condition_1.0	0.000216
54	sex_Male	0.000167

70 rows × 2 columns

In [90]:
```
x_train, x_test, y_train , y_test=train_test_split(x,y, test_size=.25, random_
```

In [91]:
```
log = LogisticRegression()

log.fit(x_train,y_train)

print(f"Training Score : {log.score(x_train, y_train)}")
print(f"Testing Score : {log.score(x_test, y_test)}")
```

```
Training Score : 0.8558662006989516
Testing Score : 0.8440916579302081
```

In [92]:
```
pred_train=log.predict(x_train)
pred_test=log.predict(x_test)
```

In [93]:
```
cnfTrain=pd.DataFrame(metrics.confusion_matrix(y_train , pred_train), columns=
                index=["Act_0", "Act_1"])
cnfTrain
```

Out[93]:

	Pred_0	Pred_1
Act_0	15032	772
Act_1	2115	2111

In [94]:
```
print(metrics.classification_report(y_train, pred_train))
```

```
              precision    recall  f1-score   support

       False       0.88      0.95      0.91     15804
        True       0.73      0.50      0.59      4226

    accuracy                           0.86     20030
   macro avg       0.80      0.73      0.75     20030
weighted avg       0.85      0.86      0.85     20030
```

In [95]:
```
cnfTest=pd.DataFrame(metrics.confusion_matrix(y_test , pred_test), columns=["P
                index=["Act_0", "Act_1"])
cnfTest
```

Out[95]:

	Pred_0	Pred_1
Act_0	4985	244
Act_1	797	651

In [96]:
```
print(metrics.classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

       False       0.86      0.95      0.91      5229
        True       0.73      0.45      0.56      1448

    accuracy                           0.84      6677
   macro avg       0.79      0.70      0.73      6677
weighted avg       0.83      0.84      0.83      6677
```

In [97]:
```
pd.DataFrame(log.predict_proba(x_train), columns=["Prob_0", "Prob_1"])
```

Out[97]:

	Prob_0	Prob_1
0	0.945491	0.054509
1	0.965750	0.034250
2	0.404915	0.595085
3	0.988873	0.011127
4	0.799867	0.200133
...	...	...
20025	0.963766	0.036234
20026	0.935824	0.064176
20027	0.735807	0.264193
20028	0.775539	0.224461
20029	0.962844	0.037156

20030 rows × 2 columns

In [98]:
```
x_train1=x_train.copy()

x_train1["Actual_Default"]=y_train
x_train1["Prob_Default"]=log.predict_proba(x_train)[:, 1]

x_train1
```

Out[98]:

	h1n1_worry_1.0	h1n1_worry_2.0	h1n1_worry_3.0	h1n1_awareness_1.0	h1n1_awareness_2
9353	True	False	False	True	Fals
6317	False	True	False	True	Fals
13542	False	True	False	True	Fals
23986	False	True	False	False	Tru
16025	True	False	False	False	Tru
...	...	...	...	...	
24938	False	False	False	False	Tru
2481	False	False	True	True	Fals
4047	False	False	True	True	Fals
6432	True	False	False	True	Fals
26584	False	True	False	True	Fals

20030 rows × 72 columns

In [99]:
```
# Decile Analysis
def profile_decile(X,y,trained_model):
    X_1=X.copy()
    y_1=y.copy()
    y_pred1=trained_model.predict(X_1)
    X_1["Prob_Event"]=trained_model.predict_proba(X_1)[:,1]
    X_1["Y_actual"]=y_1
    X_1["Y_pred"]=y_pred1
    X_1["Rank"]=pd.qcut(X_1["Prob_Event"], 10, labels=np.arange(0,10,1))
    X_1["numb"]=10
    X_1["Decile"]=X_1["numb"]-X_1["Rank"].astype("int")

    profile=pd.DataFrame(X_1.groupby("Decile") \
                         .apply(lambda x: pd.Series({
        'min_score'   : x["Prob_Event"].min(),
        'max_score'   : x["Prob_Event"].max(),
        'Event'       : x["Y_actual"].sum(),
        'Non_event'   : x["Y_actual"].count()-x["Y_actual"].sum(),
        'Total'       : x["Y_actual"].count() })))
    return profile
```

In [100]:
```
# train_profile=profile_decile(x_train, y_train, log)
# train_profile.to_excel("train_profile.xlsx")
```

In [101]:
```
# test_profile=profile_decile(x_test, y_test, log)
# test_profile.to_excel("test_profile.xlsx")
```

In [102]:
```
newtrain_pred=np.where(log.predict_proba(x_train)[:,1] > 0.232280104, 1, 0) #
```

In [103]:
```
cnfNewTrain=pd.DataFrame(metrics.confusion_matrix(y_train , newtrain_pred), co
                index=["Act_0", "Act_1"])
cnfNewTrain
```

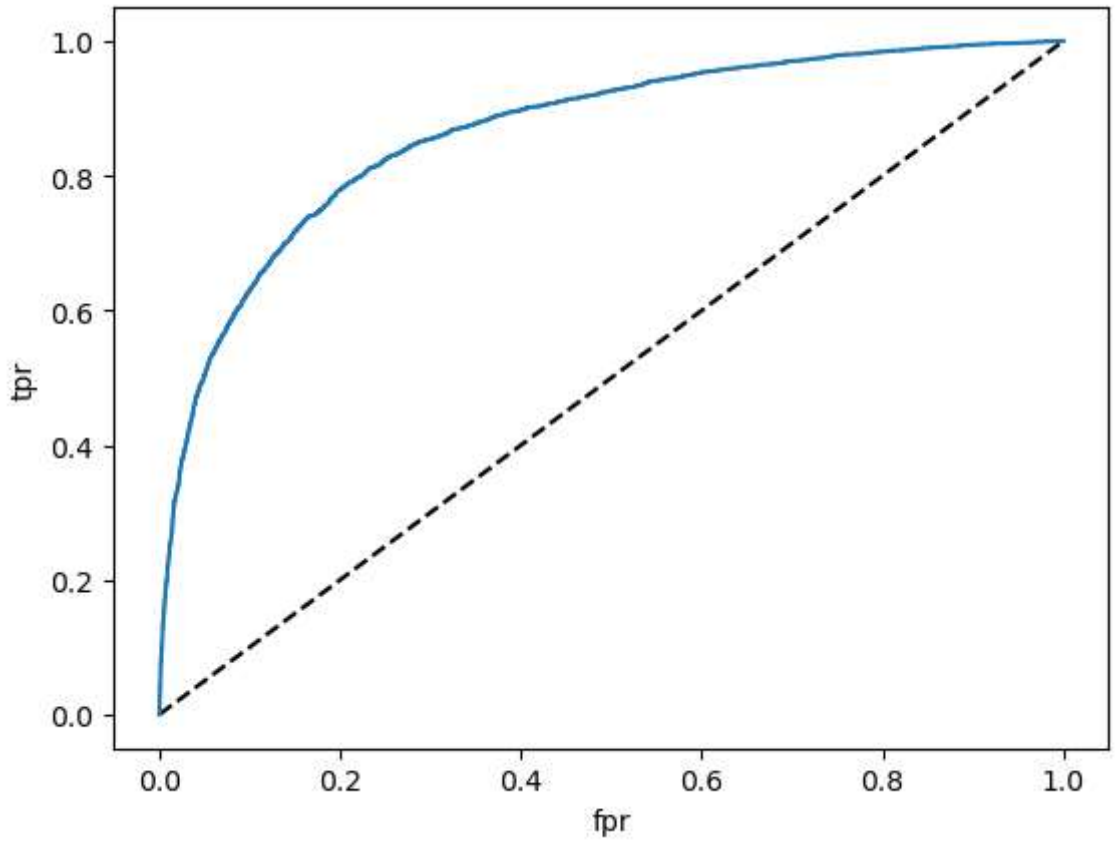
Out[103]:

	Pred_0	Pred_1
Act_0	12965	2839
Act_1	1055	3171

In [104]:
```
print(metrics.classification_report(y_train , newtrain_pred))
```

```
              precision    recall  f1-score   support

       False       0.92      0.82      0.87     15804
        True       0.53      0.75      0.62      4226

    accuracy                           0.81     20030
   macro avg       0.73      0.79      0.74     20030
weighted avg       0.84      0.81      0.82     20030
```

In [105]:
```
# Receiver Operating Characterestics

probs=log.predict_proba(x_train)[:,1]

fpr, tpr, threshold=metrics.roc_curve(y_train,probs )
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr, label='logistic')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.show()
```



In [106]:
```
metrics.roc_auc_score(y_train,probs)
```

Out[106]:
```
0.8646619144146652
```

In [112]:
```
newtest_pred=np.where(log.predict_proba(x_test)[:,1] > 0.229209326, 1, 0) # ba
```

In [117]:
```
import pandas as pd
from sklearn import metrics

# Assuming you have already defined y_test and newtest_pred

# Create a DataFrame with the information
model_logR_report = pd.DataFrame({
    'Model': ["Logical Regression"],
    'ROC Score': [metrics.roc_auc_score(y_test, newtest_pred)],
    'Precision Score': [metrics.precision_score(y_test, newtest_pred)],
    'Recall Score': [metrics.recall_score(y_test, newtest_pred)],
    'F1 Score': [metrics.f1_score(y_test, newtest_pred)],
    'Accuracy Score': [metrics.accuracy_score(y_test, newtest_pred)]
})

# Display the DataFrame
model_logR_report
```

Out[117]:

	Model	ROC Score	Precision Score	Recall Score	F1 Score	Accuracy Score
0	Logical Regression	0.769775	0.522478	0.722376	0.606377	0.796615

In [ ]:
```
In this case, Recall Score is important as we dont want to miss out on people
```