In [18]:
```
#imports
import pandas as pd
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from plotly.graph_objs import Bar
import plotly.graph_objects as go
```

In [2]:
```
data = pd.read_csv('condensed_data_by_plants_02_01_2023.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	plantcube	plant_id	plant_title	slot	planted_on	harvested_on	growth_days	owner	customer_name	customer_email	customer_creation_date	share_1	share_2	share_3	share_4
0	0061d5de-d533-4f63-b889-468b97da2f7d	80.0	Tasty Mustard (CN)	a7	29.10.2022 15:04	29.11.2022 19:47	31.24	eu-central-1:eb379358-4ea0-42bf-b100-3087d8a476b9	Markus Kreikenbaum	m.kreikenbaum@ish.de	25.08.2022 20:33	NaN	NaN	NaN	NaN
1	0061d5de-d533-4f63-b889-468b97da2f7d	80.0	Tasty Mustard (CN)	a8	29.10.2022 15:04	04.12.2022 10:15	35.84	eu-central-1:eb379358-4ea0-42bf-b100-3087d8a476b9	Markus Kreikenbaum	m.kreikenbaum@ish.de	25.08.2022 20:33	NaN	NaN	NaN	NaN
2	0061d5de-d533-4f63-b889-468b97da2f7d	80.0	Tasty Mustard (CN)	a9	29.10.2022 15:04	04.12.2022 10:15	35.84	eu-central-1:eb379358-4ea0-42bf-b100-3087d8a476b9	Markus Kreikenbaum	m.kreikenbaum@ish.de	25.08.2022 20:33	NaN	NaN	NaN	NaN
3	0061d5de-d533-4f63-b889-468b97da2f7d	NaN	Currently Empty	a1	27.12.2022 14:22	NaN	NaN	eu-central-1:eb379358-4ea0-42bf-b100-3087d8a476b9	Markus Kreikenbaum	m.kreikenbaum@ish.de	25.08.2022 20:33	NaN	NaN	NaN	NaN
4	0061d5de-d533-4f63-b889-468b97da2f7d	NaN	Currently Empty	a2	27.12.2022 14:22	NaN	NaN	eu-central-1:eb379358-4ea0-42bf-b100-3087d8a476b9	Markus Kreikenbaum	m.kreikenbaum@ish.de	25.08.2022 20:33	NaN	NaN	NaN	NaN

In [4]:
```
df = data.copy()
```

In [5]:
```
df['customer_creation_date'] = pd.to_datetime(df['customer_creation_date'], format='%d.%m.%Y %H:%M')
df['planted_on'] = pd.to_datetime(df['planted_on'], format='%d.%m.%Y %H:%M')
df['harvested_on'] = pd.to_datetime(df['harvested_on'], format='%d.%m.%Y %H:%M')

df['difference_in_days'] = df['customer_creation_date'].apply(lambda x: (datetime.now() - x).days)
# If the difference in date is more than 1 year, we can take those records
```

In [6]:
```
## long term customers

long_term_customers = df[df['difference_in_days'] > 365]
long_term_customers_plantcubes = long_term_customers['plantcube'].unique()
long_term_customers_plantcubes_count = long_term_customers.plantcube.unique().size
# Plant cubes owned by long term customers
long_term_customers_plantcubes_count
```

Out[6]: 610

In [7]:
```
## active plantcubes
# Group the data by plantcube
grouped = df.groupby('plantcube')
active_plantcubes = pd.DataFrame(columns=df.columns)

for plantcube, group in grouped:
    earliest_planting = group['planted_on'].min()
    latest_harvest = group['harvested_on'].max()
    # Calculate the difference between the dates
    date_diff = latest_harvest - earliest_planting
     # Check if the difference is less than 1 year
    if date_diff.days > 365:
        # Add the group to the active plantcubes DataFrame
        active_plantcubes = pd.concat([active_plantcubes, group], axis=0, ignore_index=True)


active_plant_cubes = active_plantcubes['plantcube'].unique()
active_plantcubes_count = active_plantcubes.plantcube.unique().size
active_plantcubes_count
```

Out[7]: 277

In [8]:
```
# Find the intersection of the plantcubes in both long_term customers and active plantcubes
intersection = set(long_term_customers_plantcubes).intersection(active_plant_cubes)
```

In [9]:
```
# Extract only the plantcubes that are in the intersection
intersection_df = df[df['plantcube'].isin(intersection)]
```

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

Out[10]:

	plantcube	plant_id	plant_title	slot	planted_on	harvested_on	growth_days	owner	customer_name	customer_email	customer_creation_date	share_1	share_2	share_3	share_4	difference_in_days
67	00bc4f20-bd95-4b21-b154-64663084b66e	96.0	Micro radish mix	b4	2021-06-20 07:41:00	2021-06-28 16:16:00	8.36	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0
68	00bc4f20-bd95-4b21-b154-64663084b66e	75.0	Bronze Fennel	a1	2021-06-26 17:46:00	2021-07-27 17:43:00	31.00	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0
69	00bc4f20-bd95-4b21-b154-64663084b66e	27.0	Pak Choi (Red Lady F1)	b2	2021-06-26 17:48:00	2021-07-24 12:15:00	27.77	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0
70	00bc4f20-bd95-4b21-b154-64663084b66e	16.0	Tatsoi (Rozetto F1)	b3	2021-06-26 17:48:00	2021-07-24 12:15:00	27.77	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0
71	00bc4f20-bd95-4b21-b154-64663084b66e	96.0	Micro radish mix	b4	2021-06-30 12:43:00	2021-07-13 12:55:00	13.01	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0

In [11]:
```
dataf = intersection_df.copy()
```

In [12]:
```
dataf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32036 entries, 67 to 59276
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   plantcube               32036 non-null  object
 1   plant_id                27871 non-null  float64
 2   plant_title             32036 non-null  object
 3   slot                    32036 non-null  object
 4   planted_on              32036 non-null  datetime64[ns]
 5   harvested_on            27572 non-null  datetime64[ns]
 6   growth_days             27572 non-null  float64
 7   owner                   32036 non-null  object
 8   customer_name           32036 non-null  object
 9   customer_email          32036 non-null  object
 10  customer_creation_date  32036 non-null  datetime64[ns]
 11  share_1                 2486 non-null   object
 12  share_2                 213 non-null    object
 13  share_3                 0 non-null      object
 14  share_4                 0 non-null      float64
 15  difference_in_days      32036 non-null  float64
dtypes: datetime64[ns](3), float64(4), object(9)
memory usage: 4.2+ MB
```

In [13]:
```
dataf['day_of_week_planted'] = dataf['planted_on'].dt.day_name()
dataf['day_of_week_harvested'] = dataf['harvested_on'].dt.day_name()
```

In [14]:
```
dataf.head()
```

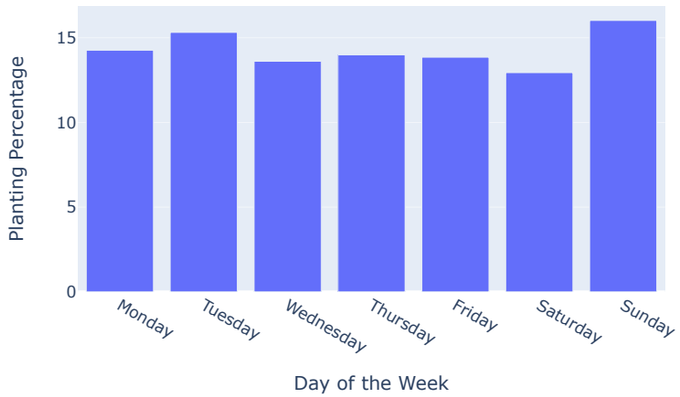
Out[14]:

	plantcube	plant_id	plant_title	slot	planted_on	harvested_on	growth_days	owner	customer_name	customer_email	customer_creation_date	share_1	share_2	share_3	share_4	difference_in_days	day_of_week_planted	day_of_w
67	00bc4f20-bd95-4b21-b154-64663084b66e	96.0	Micro radish mix	b4	2021-06-20 07:41:00	2021-06-28 16:16:00	8.36	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0	Sunday	
68	00bc4f20-bd95-4b21-b154-64663084b66e	75.0	Bronze Fennel	a1	2021-06-26 17:46:00	2021-07-27 17:43:00	31.00	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0	Saturday	
69	00bc4f20-bd95-4b21-b154-64663084b66e	27.0	Pak Choi (Red Lady F1)	b2	2021-06-26 17:48:00	2021-07-24 12:15:00	27.77	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0	Saturday	
70	00bc4f20-bd95-4b21-b154-64663084b66e	16.0	Tatsoi (Rozetto F1)	b3	2021-06-26 17:48:00	2021-07-24 12:15:00	27.77	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0	Saturday	
71	00bc4f20-bd95-4b21-b154-64663084b66e	96.0	Micro radish mix	b4	2021-06-30 12:43:00	2021-07-13 12:55:00	13.01	eu-central-1:6d6d50b2-7f13-4683-88b7-05c9c9840c38	Andrea Schöck	andreaschoeck@dtypical.com	2021-04-27 20:19:00	NaN	NaN	NaN	NaN	632.0	Wednesday	

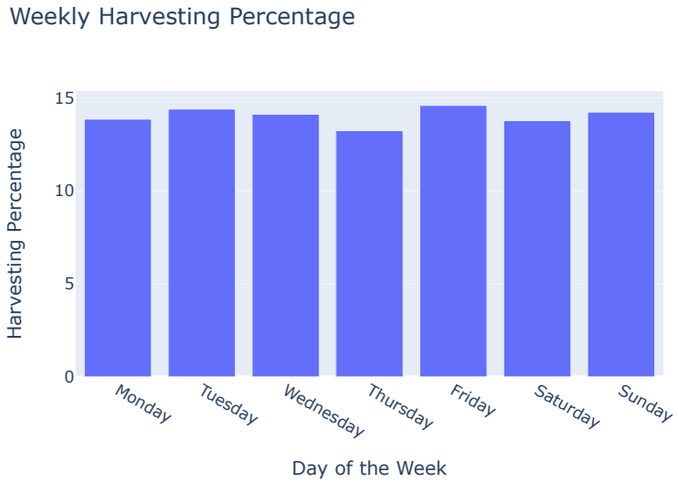
In [15]:
```
dataf = dataf[dataf['plant_title'] != 'Currently Empty']
dataf = dataf[dataf['plant_title'] != 'Not Found']
```

In [16]:
```
# analysis based on percentage.
```

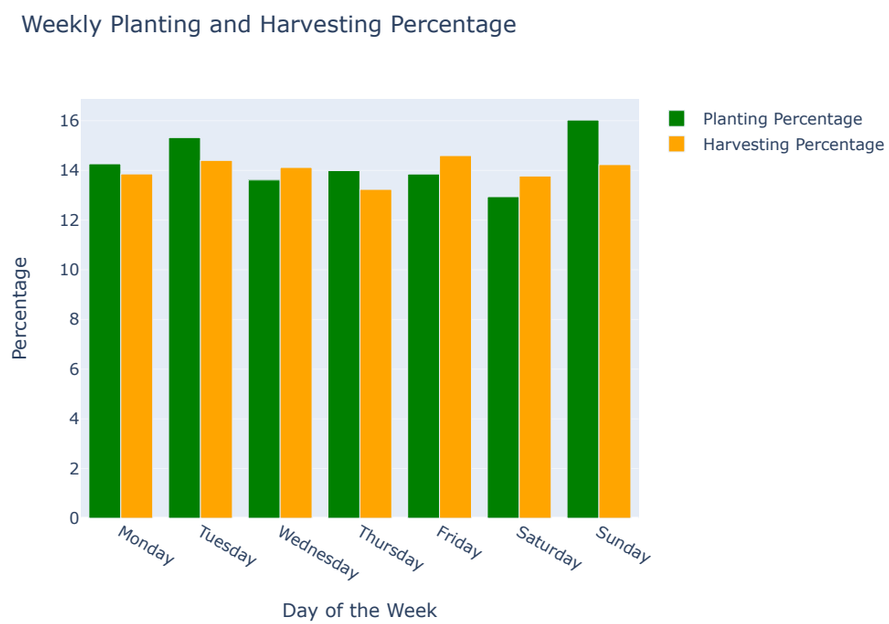
In [19]:
```
# updated graph

group_customer = dataf.groupby('customer_name')
count = 0
mon_percentage_list,tue_percentage_list,wed_percentage_list,thu_percentage_list,fri_percentage_list,sat_percentage_list,sun_percentage_list = [],[],[],[],[],[],[]
for customer, group in group_customer:
    count = count + 1
    mon_len = len(group[group.day_of_week_planted == 'Monday'])
    tue_len = len(group[group.day_of_week_planted == 'Tuesday'])
    wed_len = len(group[group.day_of_week_planted == 'Wednesday'])
    thu_len = len(group[group.day_of_week_planted == 'Thursday'])
    fri_len = len(group[group.day_of_week_planted == 'Friday'])
    sat_len = len(group[group.day_of_week_planted == 'Saturday'])
    sun_len = len(group[group.day_of_week_planted == 'Sunday'])
    total = len(group)
    percentage_monday = (mon_len/total)*100
    mon_percentage_list.append(percentage_monday)
    percentage_tuesday = (tue_len/total)*100
    tue_percentage_list.append(percentage_tuesday)
    percentage_wednesday = (wed_len/total)*100
    wed_percentage_list.append(percentage_wednesday)
    percentage_thursday = (thu_len/total)*100
    thu_percentage_list.append(percentage_thursday)
    percentage_friday = (fri_len/total)*100
    fri_percentage_list.append(percentage_friday)
    percentage_saturday = (sat_len/total)*100
    sat_percentage_list.append(percentage_saturday)
    percentage_sunday = (sun_len/total)*100
    sun_percentage_list.append(percentage_sunday)

average_percentage_planting_monday = sum(mon_percentage_list) / len(mon_percentage_list)
average_percentage_planting_tuesday = sum(tue_percentage_list) / len(tue_percentage_list)
average_percentage_planting_wednesday = sum(wed_percentage_list) / len(wed_percentage_list)
average_percentage_planting_thursday = sum(thu_percentage_list) / len(thu_percentage_list)
average_percentage_planting_friday = sum(fri_percentage_list) / len(fri_percentage_list)
average_percentage_planting_saturday = sum(sat_percentage_list) / len(sat_percentage_list)
average_percentage_planting_sunday = sum(sun_percentage_list) / len(sun_percentage_list)

# Create a bar graph
trace = Bar(x=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'], y=[average_percentage_planting_monday,average_percentage_planting_tuesday,average_percentage_planting_wednesday,average
layout = go.Layout(title='Weekly Planting Percentage', xaxis=dict(title='Day of the Week'), yaxis=dict(title='Planting Percentage'), width=600, height=400)
fig = go.Figure(data=[trace], layout=layout)
fig.show()
```


Weekly Planting Percentage
Planting Percentage
15
10
5
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of the Week

In [20]:
```
# updated graph

group_customer = dataf.groupby('customer_name')
mon_percentage_listh,tue_percentage_listh,wed_percentage_listh,thu_percentage_listh,fri_percentage_listh,sat_percentage_listh,sun_percentage_listh = [],[],[],[],[],[],[]
for customer, group in group_customer:
    mon_len = len(group[group.day_of_week_harvested == 'Monday'])
    tue_len = len(group[group.day_of_week_harvested == 'Tuesday'])
    wed_len = len(group[group.day_of_week_harvested == 'Wednesday'])
    thu_len = len(group[group.day_of_week_harvested == 'Thursday'])
    fri_len = len(group[group.day_of_week_harvested == 'Friday'])
    sat_len = len(group[group.day_of_week_harvested == 'Saturday'])
    sun_len = len(group[group.day_of_week_harvested == 'Sunday'])
    total = len(group)
    percentage_monday = (mon_len/total)*100
    mon_percentage_listh.append(percentage_monday)
    percentage_tuesday = (tue_len/total)*100
    tue_percentage_listh.append(percentage_tuesday)
    percentage_wednesday = (wed_len/total)*100
    wed_percentage_listh.append(percentage_wednesday)
    percentage_thursday = (thu_len/total)*100
    thu_percentage_listh.append(percentage_thursday)
    percentage_friday = (fri_len/total)*100
    fri_percentage_listh.append(percentage_friday)
    percentage_saturday = (sat_len/total)*100
    sat_percentage_listh.append(percentage_saturday)
    percentage_sunday = (sun_len/total)*100
    sun_percentage_listh.append(percentage_sunday)

average_percentage_harvesting_monday = sum(mon_percentage_listh) / len(mon_percentage_listh)
average_percentage_harvesting_tuesday = sum(tue_percentage_listh) / len(tue_percentage_listh)
average_percentage_harvesting_wednesday = sum(wed_percentage_listh) / len(wed_percentage_listh)
average_percentage_harvesting_thursday = sum(thu_percentage_listh) / len(thu_percentage_listh)
average_percentage_harvesting_friday = sum(fri_percentage_listh) / len(fri_percentage_listh)
average_percentage_harvesting_saturday = sum(sat_percentage_listh) / len(sat_percentage_listh)
average_percentage_harvesting_sunday = sum(sun_percentage_listh) / len(sun_percentage_listh)

# Create a bar graph
trace = Bar(x=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'], y=[average_percentage_harvesting_monday,average_percentage_harvesting_tuesday,average_percentage_harvesting_wednesday,a
layout = go.Layout(title='Weekly Harvesting Percentage', xaxis=dict(title='Day of the Week'), yaxis=dict(title='Harvesting Percentage'), width=600, height=400)
fig = go.Figure(data=[trace], layout=layout)
fig.show()
```


Weekly Harvesting Percentage
Harvesting Percentage
15
10
5
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of the Week

In [21]:
```
from plotly.graph_objs import Bar

# Define the data for the planting and harvesting traces
planting_trace = Bar(x=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'],
                     y=[average_percentage_planting_monday,average_percentage_planting_tuesday,average_percentage_planting_wednesday,average_percentage_planting_thursday,average_percentage_planting_friday,av
                     name='Planting Percentage',
                     marker=dict(color='green'))

harvesting_trace = Bar(x=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'],
                       y=[average_percentage_harvesting_monday,average_percentage_harvesting_tuesday,average_percentage_harvesting_wednesday,average_percentage_harvesting_thursday,average_percentage_harvesti
                       name='Harvesting Percentage',
                       marker=dict(color='orange'))

# Define the layout for the plot
layout = go.Layout(title='Weekly Planting and Harvesting Percentage',
                   xaxis=dict(title='Day of the Week'),
                   yaxis=dict(title='Percentage'),
                   barmode='group',width = 700,height = 500)

# Create the figure and add the data and layout
fig = go.Figure(data=[planting_trace, harvesting_trace], layout=layout)

# Show the plot
fig.show()
```

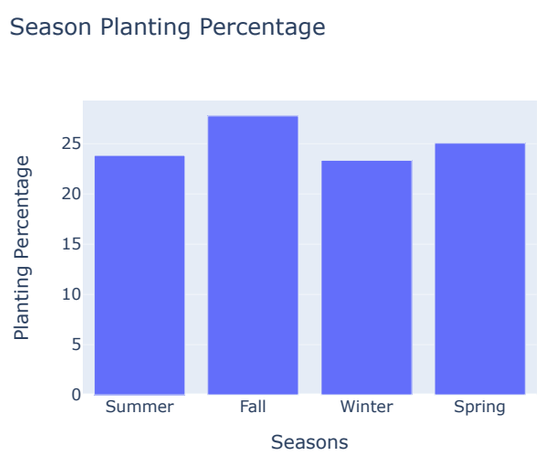


In [22]:
```
dataf['planted_month'] = dataf['planted_on'].dt.month
dataf['harvested_month'] = dataf['harvested_on'].dt.month
```

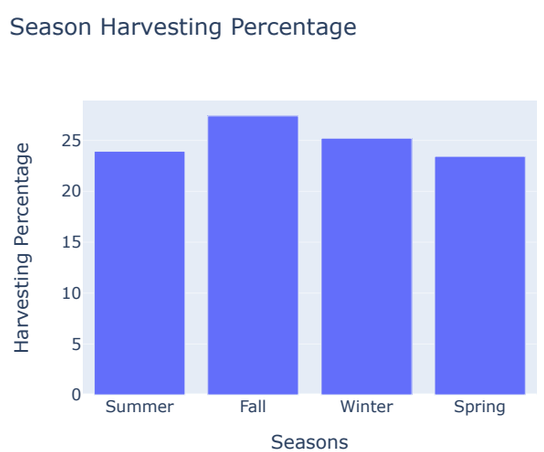
In [23]:
```
# how seasons affect the planting behaviour
# create a column for seasons_planted
dataf['season_planted'] = 'other'

# assign the season to each plant based on the month it was planted
dataf.loc[dataf['planted_month'].isin([6, 7, 8]), 'season_planted'] = 'summer'
dataf.loc[dataf['planted_month'].isin([9, 10, 11]), 'season_planted'] = 'fall'
dataf.loc[dataf['planted_month'].isin([12, 1, 2]), 'season_planted'] = 'winter'
dataf.loc[dataf['planted_month'].isin([3, 4, 5]), 'season_planted'] = 'spring'
```

In [24]:
```
# how seasons affect the harvested behaviour
# create a column for seasons_planted
dataf['season_harvested'] = 'other'

# assign the season to each plant based on the month it was planted
dataf.loc[dataf['harvested_month'].isin([6, 7, 8]), 'season_harvested'] = 'summer'
dataf.loc[dataf['harvested_month'].isin([9, 10, 11]), 'season_harvested'] = 'fall'
dataf.loc[dataf['harvested_month'].isin([12, 1, 2]), 'season_harvested'] = 'winter'
dataf.loc[dataf['harvested_month'].isin([3, 4, 5]), 'season_harvested'] = 'spring'
dataf1 = dataf[dataf['season_harvested'] != 'other']
```

In [25]:
```
# updated graph
group_customer = dataf.groupby('customer_name')
summer_percentage_list,fall_percentage_list,winter_percentage_list,spring_percentage_list = [],[],[],[]
for customer, group in group_customer:
    summer_len = len(group[group.season_planted == 'summer'])
    fall_len = len(group[group.season_planted == 'fall'])
    winter_len = len(group[group.season_planted == 'winter'])
    spring_len = len(group[group.season_planted == 'spring'])
    total = len(group)
    percentage_summer = (summer_len/total)*100
    summer_percentage_list.append(percentage_summer)
    percentage_fall = (fall_len/total)*100
    fall_percentage_list.append(percentage_fall)
    percentage_winter = (winter_len/total)*100
    winter_percentage_list.append(percentage_winter)
    percentage_spring = (spring_len/total)*100
    spring_percentage_list.append(percentage_spring)

average_percentage_planting_summer = sum(summer_percentage_list) / len(summer_percentage_list)
average_percentage_planting_fall = sum(fall_percentage_list) / len(fall_percentage_list)
average_percentage_planting_winter = sum(winter_percentage_list) / len(winter_percentage_list)
average_percentage_planting_spring = sum(spring_percentage_list) / len(spring_percentage_list)

# Create a bar graph
trace = Bar(x=['Summer','Fall','Winter','Spring'], y=[average_percentage_planting_summer,average_percentage_planting_fall,average_percentage_planting_winter,average_percentage_planting_spring])
layout = go.Layout(title='Season Planting Percentage', xaxis=dict(title='Seasons'), yaxis=dict(title='Planting Percentage'), width=500, height=400)
fig = go.Figure(data=[trace], layout=layout)
fig.show()
```



In [26]:
```
# updated graph
group_customer = dataf1.groupby('customer_name')
summer_percentage_listh,fall_percentage_listh,winter_percentage_listh,spring_percentage_listh = [],[],[],[]
for customer, group in group_customer:
    summer_len = len(group[group.season_harvested == 'summer'])
    fall_len = len(group[group.season_harvested == 'fall'])
    winter_len = len(group[group.season_harvested == 'winter'])
    spring_len = len(group[group.season_harvested == 'spring'])
    total = len(group)
    percentage_summer = (summer_len/total)*100
    summer_percentage_listh.append(percentage_summer)
    percentage_fall = (fall_len/total)*100
    fall_percentage_listh.append(percentage_fall)
    percentage_winter = (winter_len/total)*100
    winter_percentage_listh.append(percentage_winter)
    percentage_spring = (spring_len/total)*100
    spring_percentage_listh.append(percentage_spring)

average_percentage_harvesting_summer = sum(summer_percentage_listh) / len(summer_percentage_listh)
average_percentage_harvesting_fall = sum(fall_percentage_listh) / len(fall_percentage_listh)
average_percentage_harvesting_winter = sum(winter_percentage_listh) / len(winter_percentage_listh)
average_percentage_harvesting_spring = sum(spring_percentage_listh) / len(spring_percentage_listh)

# Create a bar graph
trace = Bar(x=['Summer','Fall','Winter','Spring'], y=[average_percentage_harvesting_summer,average_percentage_harvesting_fall,average_percentage_harvesting_winter,average_percentage_harvesting_spring])
layout = go.Layout(title='Season Harvesting Percentage', xaxis=dict(title='Seasons'), yaxis=dict(title='Harvesting Percentage'), width=500, height=400)
fig = go.Figure(data=[trace], layout=layout)
fig.show()
```



In [27]:
```
from plotly.graph_objs import Bar

# Define the data for the planting and harvesting traces
season_planting_trace = Bar(x=['Summer','Fall','Winter','Spring'],
                    y=[average_percentage_planting_summer,average_percentage_planting_fall,average_percentage_planting_winter,average_percentage_planting_spring],
                    name='Planting Percentage',
                    marker=dict(color='green'))

season_harvesting_trace = Bar(x=['Summer','Fall','Winter','Spring'],
                      y=[average_percentage_harvesting_summer,average_percentage_harvesting_fall,average_percentage_harvesting_winter,average_percentage_harvesting_spring],
                      name='Harvesting Percentage',
                      marker=dict(color='orange'))

# Define the layout for the plot
layout = go.Layout(title='Season Planting and Harvesting Percentage',
                   xaxis=dict(title='Seasons'),
                   yaxis=dict(title='Percentage'),
                   barmode='group',width = 700,height = 500)

# Create the figure and add the data and layout
fig = go.Figure(data=[season_planting_trace, season_harvesting_trace], layout=layout)

# Show the plot
fig.show()
```