In [173]:
```
#imports
import pandas as pd
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx
import plotly.express as px
```

In [174]:
```
data = pd.read_csv('condensed_data_by_plants_02_01_2023.csv')
```

In [175]:
```
data.head()
```

Out[175]:

	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 [176]:
```
# different plant titles in the data
data.plant_title.unique()
```

Out[176]:
```
array(['Tasty Mustard (CN)', 'Currently Empty',
       'Mustard (Frizzie lizzie)', 'Kale (Scarlet)', 'Basil (Salvo)',
       'Pak Choi (Rubi)', 'Mustard (Red lace)', 'Pak choi (Hanakan)',
       'Red Radish (Rioja)', 'Rocket (Victoria)', 'Tatsoi (Rozetto F1)',
       'Sandwich Greens (Mesclun Mix)', 'Watercress', 'Parsley (Lisette)',
       'Dill (Dukat)', 'Micro radish mix', 'Bronze Fennel',
       'Pak Choi (Red Lady F1)', 'Lettuce Romaine (Deronda)',
       'Cinnamon Basil', 'Lemon Basil', 'Red Basil (Red Rubin)',
       'Rainbow Salad (CN Mesclun Mix)', 'Salad frilly leaf blend (CN)',
       'Sorrel (Red Veined)', 'Kale  (CN KAL 1028)', 'Borage (Blue)',
       'Korean Mint', 'Anise Hyssop', 'Sage (English)',
       'Lettuce Romaine (Red Cos)', 'Broccoli (Green Micro)',
       'Thai Basil (Siam Queen)', 'Lettuce Red Batavia', 'Lemon Balm',
       'Thyme (English Winter)', 'Lettuce Lollo Rosso',
       'Lettuce Little Gem', 'Lettuce Cerbiatta', 'Lettuce Tango',
       'Pak Choi Colour Crunch', 'Leaf beet (Bulls Blood)',
       'Radish (Daikon)', 'BBQ Salat', 'Asia Salat',
       'Multisalad (Greenflash)', 'Bunter Senf', 'Rocket Wild (Letizia)',
       'Basil (Rubra)', 'Mustard (Wasabina)',
       'Amaranth (Passion Variegated)', 'Coriander (Splits Micro)',
       'Summer Savory', 'Chives', 'Orange Tomato', 'Red Tomato',
       'Chilli (Red)', 'Green Mint', 'Greek Basil',
       'Daily Greens Smoothie', 'Stir Fry', 'Mizuna (Arun)', 'Not Found',
       'Buddha Bowl Greens Mix', 'Oriental Salad Mix', 'Basil Cinnamon',
       'Basil Lemon'], dtype=object)
```

In [27]:
```
df = data.copy()
```

In [28]:
```
actual_df_count = df.plantcube.unique().size
actual_df_count
```

Out[28]: 962

Long Term customers with active plantcubes

In [29]:
```
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 [30]:
```
## 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[30]: 603

In [31]:
```
## 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[31]: 485

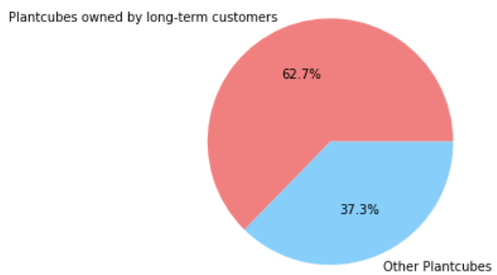
In [33]:
```
# Find the intersection of the plantcubes in both long_term customers and active plantcubes
intersection = set(long_term_customers_plantcubes).intersection(active_plant_cubes)
```

In [34]:
```
# Extract only the plantcubes that are in the intersection
intersection_df = df[df['plantcube'].isin(intersection)]
```

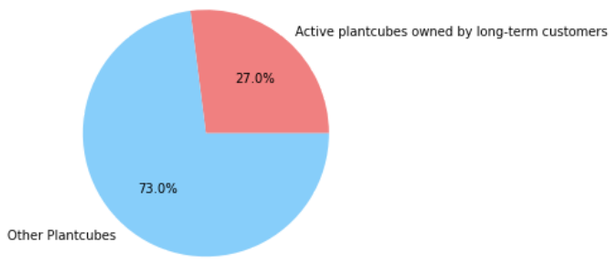
In [36]:
```
# long term customers with active plantcubes
intersection_df.plantcube.unique().size
```

Out[36]:
```
260
```

In [37]:
```
#visualization

# total plant cubes
total_plant_cubes = df['plantcube'].nunique()
# Number of plantcubes owned by long term customers
long_term_plant_cubes = long_term_customers['plantcube'].nunique()
# Percentage
long_term_plant_cubes_percentage = long_term_plant_cubes / total_plant_cubes * 100

# Create the pie chart
labels = ['Plantcubes owned by long-term customers', 'Other Plantcubes']
sizes = [long_term_plant_cubes_percentage, 100 - long_term_plant_cubes_percentage]
colors = ['lightcoral', 'lightskyblue']

fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%')
ax.axis('equal')

plt.show()
```



In [38]:
```
#visualization

# total plant cubes
total_plant_cubes = df['plantcube'].nunique()
# Number of plantcubes owned by long term customers
active_long_term_plant_cubes = intersection_df['plantcube'].nunique()
# Percentage
active_long_term_plant_cubes_percentage = active_long_term_plant_cubes / total_plant_cubes * 100

# Create the pie chart
labels = ['Active plantcubes owned by long-term customers', 'Other Plantcubes']
sizes = [active_long_term_plant_cubes_percentage, 100 - active_long_term_plant_cubes_percentage]
colors = ['lightcoral', 'lightskyblue']

fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%')
ax.axis('equal')

plt.show()
```



Analysis: what is planted more frequently and how often is it harvested

In [39]:
```
df3 = intersection_df.copy()

# Filter the rows where the plant_title column is not equal to "currently empty"
df3 = df3[df3['plant_title'] != 'Currently Empty']
df3 = df3[df3['plant_title'] != 'Not Found']
```

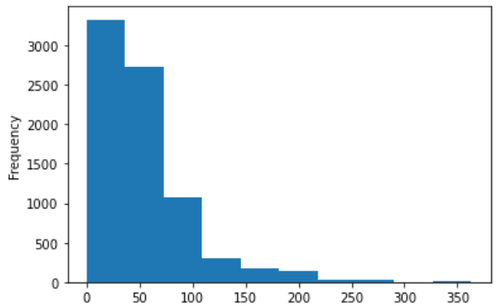
In [40]:
```
# how long the customers keep their plants before harvesting them, or how often they plant new plants.

df3['duration'] = (df3['harvested_on'] - df3['planted_on']).dt.days

df3['duration'].describe()
```

Out[40]:
```
count    7810.000000
mean       54.510755
std        43.427945
min         0.000000
25%        28.000000
50%        42.000000
75%        67.000000
max       363.000000
Name: duration, dtype: float64
```

The summary of the duration column tells that the mean duration of the plants is about 52 days, with a standard deviation of 48.7 days. This indicates that most of the plants were kept for a duration of time that is within about 49 days of the mean.

In [41]:
```
# distribution of the durations.
df3['duration'].plot.hist()
```

Out[41]:
```
<AxesSubplot:ylabel='Frequency'>
```

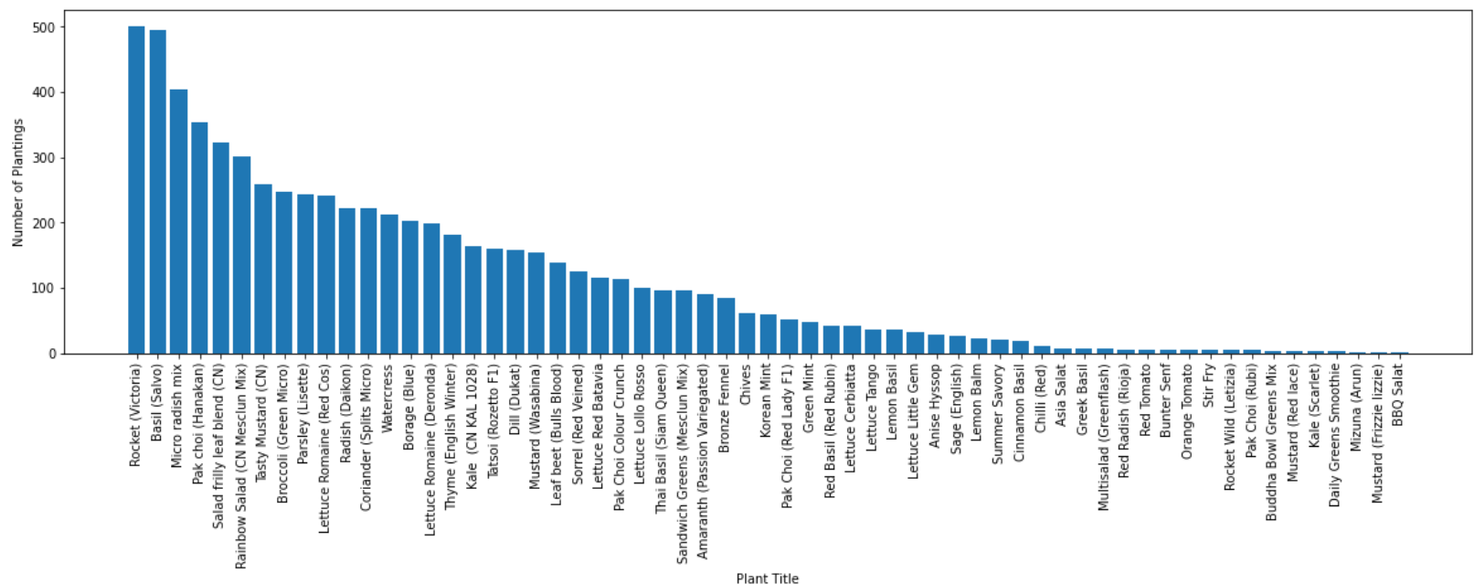


In [42]:
```
# create a boolean condition for the growth days column
cond = (df3["growth_days"] >= 1) & (df3["growth_days"] <= 100)

# extract the rows that match the conditions
df3 = df3[cond]
```

In [44]:
```
# frequently planted plant

# Group the data by the plant_title column
group_plant_title = df3.groupby('plant_title')

# Count the number of plantings happened for each plant
plantings_count = group_plant_title['plant_title'].count()

# Planting should have taken place more than 1 time
frequenly_planted = plantings_count[plantings_count > 1]

# Sort the frequently planted plants in descending order
frequenly_planted = frequenly_planted.sort_values(ascending=False)

frequenly_planted
```

Out[44]:
```
plant_title
Rocket (Victoria)              501
Basil (Salvo)                  495
Micro radish mix               404
Pak choi (Hanakan)             354
Salad frilly leaf blend (CN)   322
                              ...
Kale (Scarlet)                   3
Daily Greens Smoothie            3
Mizuna (Arun)                    2
Mustard (Frizzie lizzie)         2
BBQ Salat                        2
Name: plant_title, Length: 61, dtype: int64
```

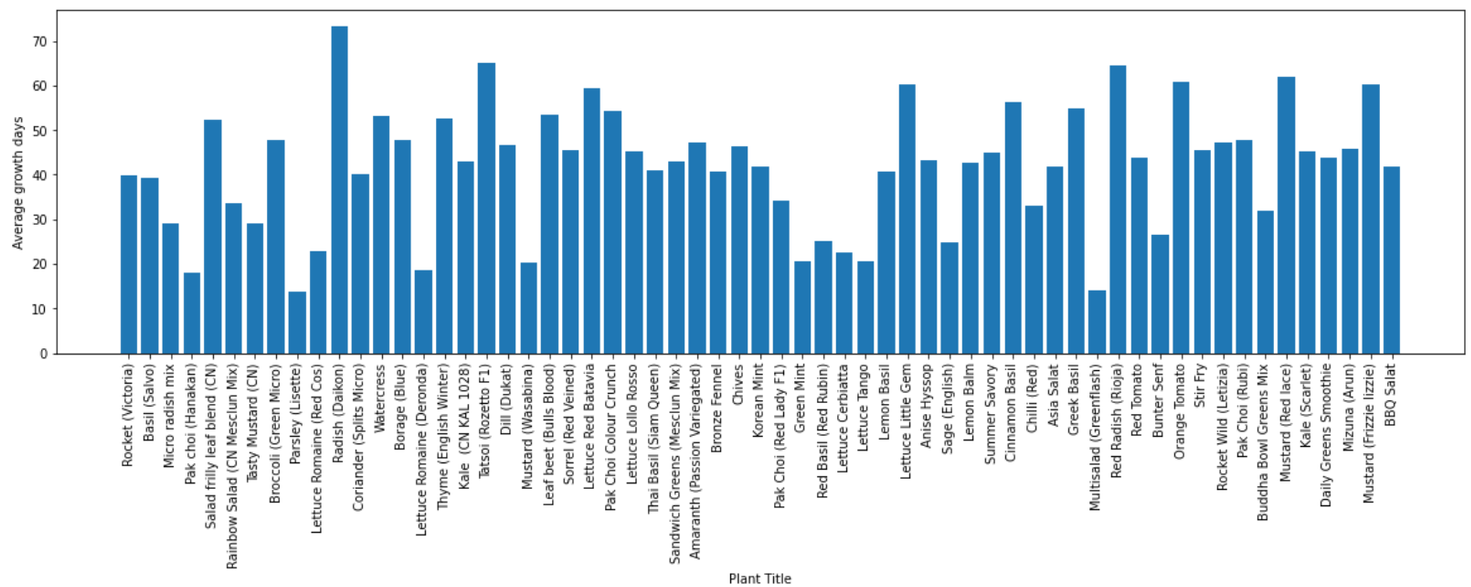
In [45]:
```
# visualization

# Get the plant titles and the number of plantings as lists
titles = frequenly_planted.index.tolist()
number_of_plantings = frequenly_planted.values.tolist()

# Create the bar chart
fig, ax = plt.subplots(figsize=(20, 5))
ax.bar(titles, number_of_plantings)

# Set the x-axis label
ax.set_xlabel('Plant Title')

# Set the y-axis label
ax.set_ylabel('Number of Plantings')

# Rotate the x-axis labels
plt.xticks(titles, titles, rotation=90)

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



In [46]:
```
# Calculate the average number of growth days for each plant
avg_growth_days = group_plant_title['growth_days'].mean()
avg_growth_days
```

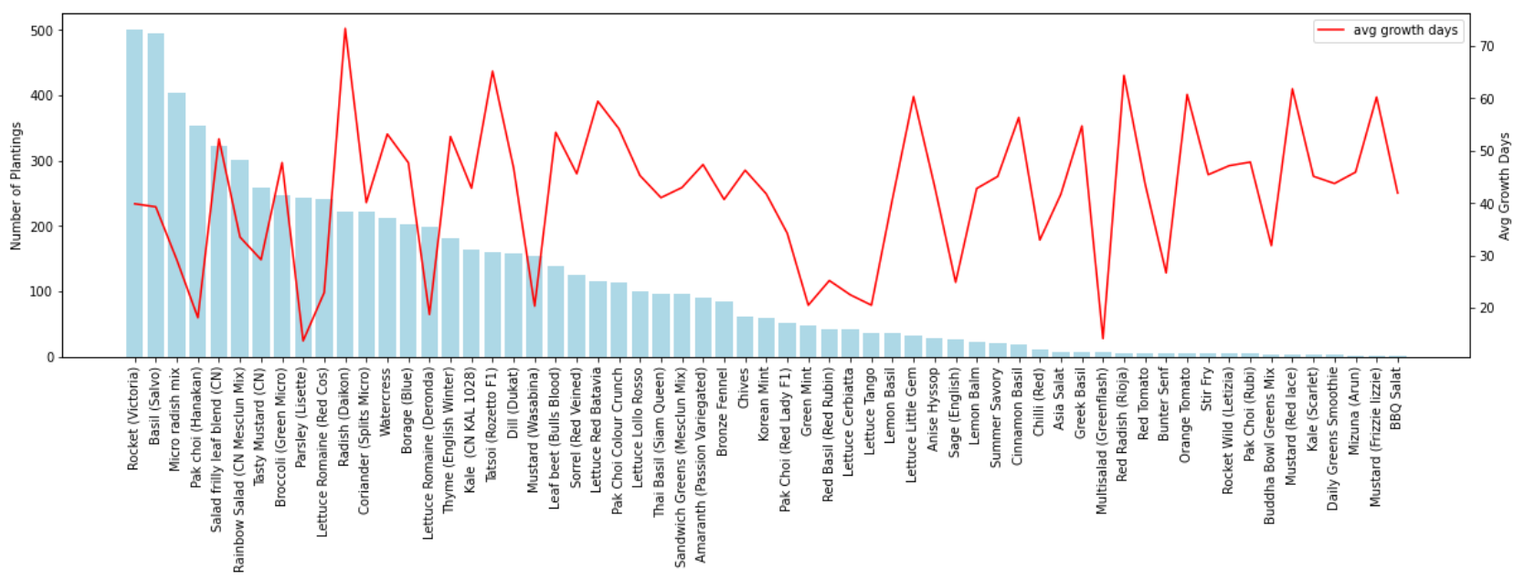
Out[46]:
```
plant_title
Amaranth (Passion Variegated)    39.852667
Anise Hyssop                     39.265172
Asia Salat                       29.118571
BBQ Salat                        18.080000
Basil (Salvo)                    52.241354
                                   ...
Tasty Mustard (CN)               45.103992
Tatsoi (Rozetto F1)              43.726000
Thai Basil (Siam Queen)          45.895833
Thyme (English Winter)           60.252747
Watercress                       41.905377
Name: growth_days, Length: 61, dtype: float64
```

In [47]:
```
# visualization

# Get the plant titles and the number of plantings as lists
titles = frequenly_planted.index.tolist()
avg_growth_days = avg_growth_days.values.tolist()

# Create the bar chart
fig, ax = plt.subplots(figsize=(20, 5))
ax.bar(titles, avg_growth_days)

# Set the x-axis label
ax.set_xlabel('Plant Title')

# Set the y-axis label
ax.set_ylabel('Average growth days')

# Rotate the x-axis labels
plt.xticks(titles, titles, rotation=90)

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



In [49]:
```
# Ignore the warnings
pd.options.mode.chained_assignment = None

# Create the figure and the subplot
fig, ax = plt.subplots(figsize=(20, 5))

# Create the bar chart
ax.bar(titles,number_of_plantings,color='lightblue')

# Set the y-axis label for the bar chart
ax.set_ylabel('Number of Plantings')

# Create the second y-axis
ax2 = ax.twinx()

# Create the line chart
ax2.plot(titles, avg_growth_days,color='red',label="avg growth days")

# Set the y-axis label for the line chart
ax2.set_ylabel('Avg Growth Days')
plt.legend(loc="upper right")
# Rotate the x-axis labels
ax.set_xticklabels(titles,rotation=90)

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

```
C:\Users\Nila\AppData\Local\Temp\ipykernel_61632\2071756675.py:23: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(titles,rotation=90)
```



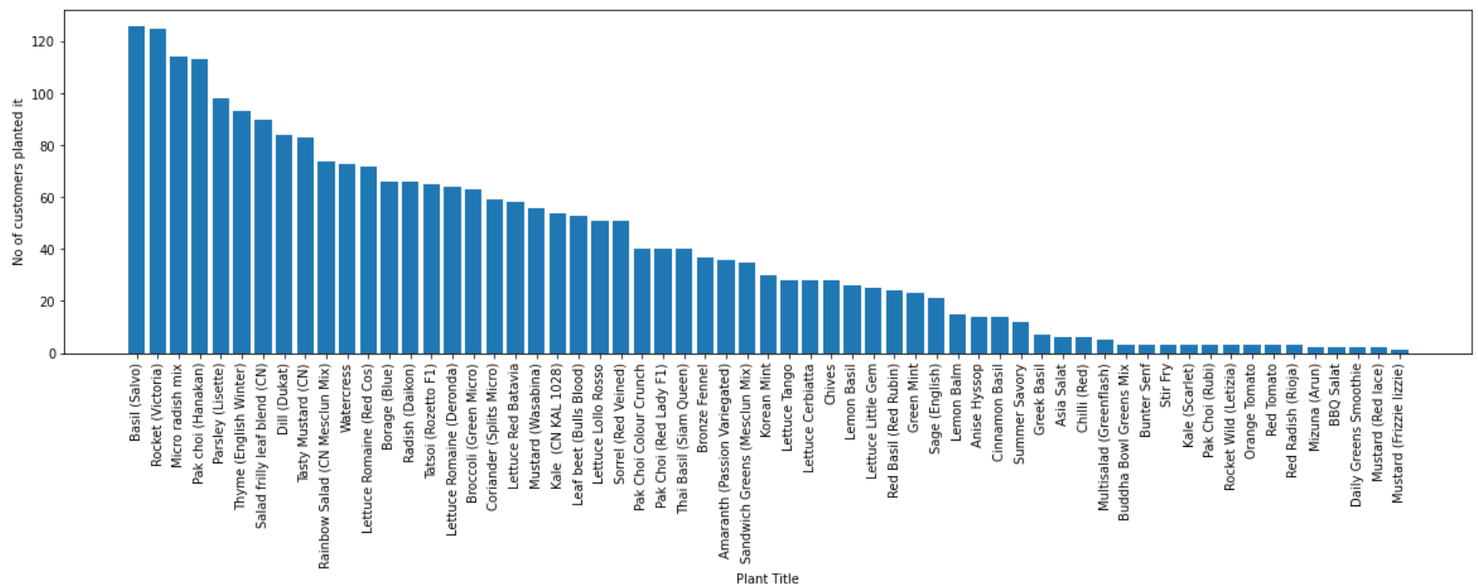
In [50]:
```
#customer name

# Select the customer name column
customer_names = df3['customer_name']

# Count the number of unique customer names
num_customers = customer_names.nunique()

# Print the result
print(f'Total number of customers: {num_customers}')
```

```
Total number of customers: 236
```

In [51]:
```
# which plant titles are the most popular, based on how many customers have planted them.
# most commonly planted plant by most customers

# Select the plant title and customer name columns
plant_titles_customers = df3[['plant_title', 'customer_name']]

# Drop duplicate rows
plant_titles_customers = plant_titles_customers.drop_duplicates()

# Count the number of unique customer names for each plant title
plant_counts_customer = plant_titles_customers.groupby('plant_title')['customer_name'].nunique()

# Sort by descending order
plant_counts_customer = plant_counts_customer.sort_values(ascending=False)

plant_counts_customer
```

Out[51]:
```
plant_title
Basil (Salvo)               126
Rocket (Victoria)           125
Micro radish mix            114
Pak choi (Hanakan)          113
Parsley (Lisette)            98
                           ...
Mizuna (Arun)                 2
BBQ Salat                     2
Daily Greens Smoothie         2
Mustard (Red lace)            2
Mustard (Frizzie lizzie)      1
Name: customer_name, Length: 61, dtype: int64
```

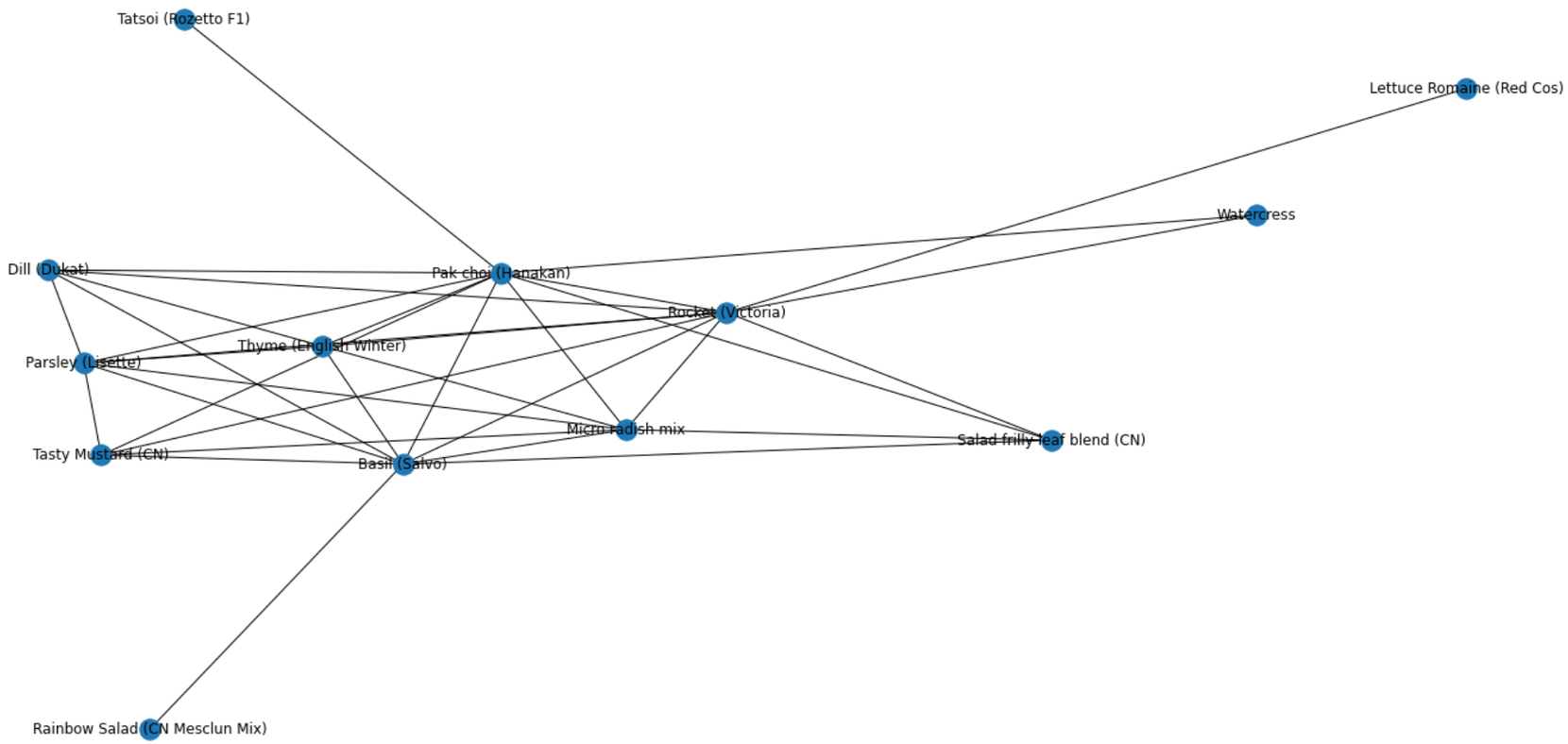
In [52]:
```
# visualization

# Get the plant titles and the number of plantings as lists
titles = plant_counts_customer.index.tolist()
no_of_customers = plant_counts_customer.values.tolist()

# Create the bar chart
fig, ax = plt.subplots(figsize=(20, 5))
ax.bar(titles, no_of_customers)

# Set the x-axis label
ax.set_xlabel('Plant Title')

# Set the y-axis label
ax.set_ylabel('No of customers planted it')

# Rotate the x-axis labels
plt.xticks(titles, titles, rotation=90)

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



Plants planted together by the customers

In [53]:
```
import pandas as pd
import networkx as nx

# read the data into a DataFrame
df4 = df3.copy()

# create an empty graph
G = nx.Graph()

# create a dictionary to store the connections between plant titles
connections = {}

# iterate through the rows in the DataFrame
for i, row in df4.iterrows():
    # get the plant title and customer name for this row
    plant_title = row['plant_title']
    customer_name = row['customer_name']

    # add the plant title as a node in the graph
    G.add_node(plant_title)

    # add the connection to the dictionary

    # if the plant title is already a key in the connections dictionary
    if plant_title in connections:
        connections[plant_title].add(customer_name)
    # if it is not, add the key as well as value
    else:
        connections[plant_title] = {customer_name}
    #print(connections)

# iterate through the connections in the dictionary
for plant_title, customer_names in connections.items():
    # connect the plant title to all other plant titles used by the same customer
    for other_plant_title, other_customer_names in connections.items():
        # Plant titles are not same, but customer names are same, then there is a connection
        if plant_title != other_plant_title and customer_names & other_customer_names:
            # there is at least one customer in common between the two plant titles
            G.add_edge(plant_title, other_plant_title, weight=len(customer_names & other_customer_names))

# create a figure with a larger size
#plt.figure(figsize=(20, 10))

# draw the graph using the spring layout
#pos = nx.spring_layout(G)
#nx.draw(G, pos, with_labels=True)

# show the plot
#plt.show()
```

In [57]:
```
import matplotlib.pyplot as plt

# Total customers - 236.
# create a subgraph with only the edges with a weight >= 50
subgraph = G.edge_subgraph([(u, v) for u, v, w in G.edges(data='weight') if w >= 50])

# create a figure with a larger size
plt.figure(figsize=(20, 10))

# draw the subgraph using the spring layout
pos = nx.spring_layout(subgraph)
nx.draw(subgraph, pos, with_labels=True)

# show the plot
plt.show()
```

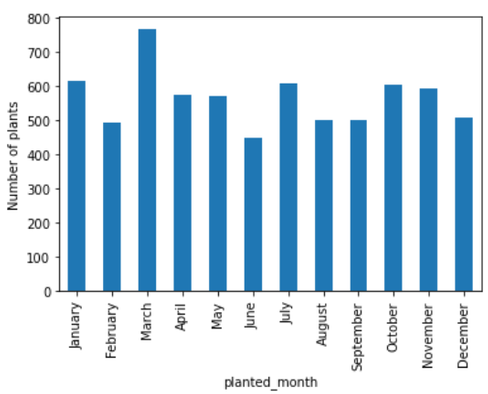


In [59]:
```
# Ignore the warnings
pd.options.mode.chained_assignment = None

# Initialize an empty DataFrame
data = {'Plant 1': [], 'Plant 2': [], 'Weight': []}
combination = pd.DataFrame(data)
# Iterate through the edges of the subgraph
for u, v, weight in subgraph.edges(data='weight'):
    # Add a new row to the DataFrame with the plant titles and weight value
    new_row = pd.DataFrame({'Plant 1': [u], 'Plant 2': [v], 'Weight': [weight]})
    combination = pd.concat([combination, new_row], ignore_index=True)

# Print the resulting DataFrame
print(combination)
# weight represents number of customers who planted that combination.
```

	Plant 1	Plant 2	Weight
0	Rainbow Salad (CN Mesclun Mix)	Basil (Salvo)	58.0
1	Micro radish mix	Basil (Salvo)	67.0
2	Micro radish mix	Pak choi (Hanakan)	72.0
3	Micro radish mix	Rocket (Victoria)	69.0
4	Micro radish mix	Parsley (Lisette)	58.0
5	Micro radish mix	Salad frilly leaf blend (CN)	51.0
6	Micro radish mix	Tasty Mustard (CN)	59.0
7	Micro radish mix	Thyme (English Winter)	55.0
8	Lettuce Romaine (Red Cos)	Rocket (Victoria)	50.0
9	Basil (Salvo)	Pak choi (Hanakan)	71.0
10	Basil (Salvo)	Rocket (Victoria)	81.0
11	Basil (Salvo)	Parsley (Lisette)	85.0
12	Basil (Salvo)	Dill (Dukat)	64.0
13	Basil (Salvo)	Salad frilly leaf blend (CN)	51.0
14	Basil (Salvo)	Tasty Mustard (CN)	55.0
15	Basil (Salvo)	Thyme (English Winter)	74.0
16	Tatsoi (Rozetto F1)	Pak choi (Hanakan)	54.0
17	Rocket (Victoria)	Pak choi (Hanakan)	72.0
18	Rocket (Victoria)	Watercress	58.0
19	Rocket (Victoria)	Parsley (Lisette)	65.0
20	Rocket (Victoria)	Dill (Dukat)	56.0
21	Rocket (Victoria)	Salad frilly leaf blend (CN)	63.0
22	Rocket (Victoria)	Tasty Mustard (CN)	55.0
23	Rocket (Victoria)	Thyme (English Winter)	67.0
24	Thyme (English Winter)	Pak choi (Hanakan)	55.0
25	Thyme (English Winter)	Parsley (Lisette)	65.0
26	Thyme (English Winter)	Dill (Dukat)	53.0
27	Salad frilly leaf blend (CN)	Pak choi (Hanakan)	50.0
28	Pak choi (Hanakan)	Watercress	51.0
29	Pak choi (Hanakan)	Parsley (Lisette)	60.0
30	Pak choi (Hanakan)	Dill (Dukat)	51.0
31	Pak choi (Hanakan)	Tasty Mustard (CN)	58.0
32	Dill (Dukat)	Parsley (Lisette)	64.0
33	Parsley (Lisette)	Tasty Mustard (CN)	50.0

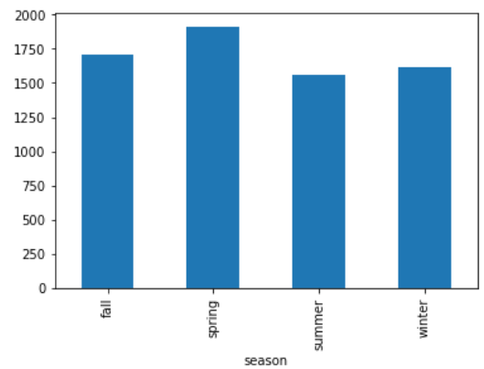
In [60]:
```
# Identifying plant popularity over time
# planted on
df6 = df3.copy()

df6['planted_month'] = df6['planted_on'].dt.month

plant_counts_by_month = df6.groupby('planted_month')['plant_id'].count()

month_names = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']
plant_counts_by_month.plot.bar()
plt.ylabel('Number of plants')
plt.xticks(range(len(month_names)), month_names)
```

Out[60]:
```
([<matplotlib.axis.XTick at 0x22b9f892b80>,
  <matplotlib.axis.XTick at 0x22b9f892d00>,
  <matplotlib.axis.XTick at 0x22b9f5689a0>,
  <matplotlib.axis.XTick at 0x22b99fe4310>,
  <matplotlib.axis.XTick at 0x22b9f568850>,
  <matplotlib.axis.XTick at 0x22b99fe4e80>,
  <matplotlib.axis.XTick at 0x22b9a015670>,
  <matplotlib.axis.XTick at 0x22b9a015b20>,
  <matplotlib.axis.XTick at 0x22b9f8923d0>,
  <matplotlib.axis.XTick at 0x22b9a1aafd0>,
  <matplotlib.axis.XTick at 0x22b9a1aaaf0>,
  <matplotlib.axis.XTick at 0x22b9a015ee0>],
 [Text(0, 0, 'January'),
  Text(1, 0, 'February'),
  Text(2, 0, 'March'),
  Text(3, 0, 'April'),
  Text(4, 0, 'May'),
  Text(5, 0, 'June'),
  Text(6, 0, 'July'),
  Text(7, 0, 'August'),
  Text(8, 0, 'September'),
  Text(9, 0, 'October'),
  Text(10, 0, 'November'),
  Text(11, 0, 'December')])
```



In [61]:
```
# how seasons affect the planting behaviour
# create a column for seasons
df6['season'] = 'other'

# assign the season to each plant based on the month it was planted
df6.loc[df6['planted_month'].isin([6, 7, 8]), 'season'] = 'summer'
df6.loc[df6['planted_month'].isin([9, 10, 11]), 'season'] = 'fall'
df6.loc[df6['planted_month'].isin([12, 1, 2]), 'season'] = 'winter'
df6.loc[df6['planted_month'].isin([3, 4, 5]), 'season'] = 'spring'

plant_counts_by_season = df6.groupby('season')['plant_title'].count()

plant_counts_by_season.plot.bar()
```

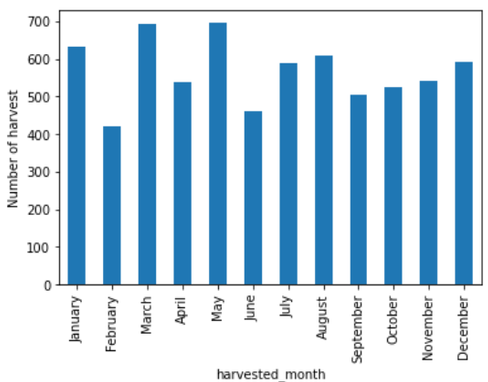
Out[61]: <AxesSubplot:xlabel='season'>



In [62]:
```
# Identifying plant popularity over time
# harvested on
df6['harvested_month'] = df6['harvested_on'].dt.month

harvested_counts_by_month = df6.groupby('harvested_month')['plant_id'].count()
```

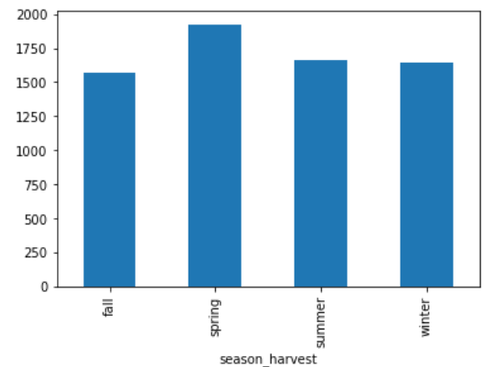
In [63]:
```
month_names = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']
harvested_counts_by_month.plot.bar()
plt.ylabel('Number of harvest')
plt.xticks(range(len(month_names)), month_names)
```

Out[63]:
```
([<matplotlib.axis.XTick at 0x22b996793a0>,
  <matplotlib.axis.XTick at 0x22b99679670>,
  <matplotlib.axis.XTick at 0x22b9e0e7640>,
  <matplotlib.axis.XTick at 0x22b9eca5ac0>,
  <matplotlib.axis.XTick at 0x22b99f2a3d0>,
  <matplotlib.axis.XTick at 0x22b99f2abe0>,
  <matplotlib.axis.XTick at 0x22ba1ac3070>,
  <matplotlib.axis.XTick at 0x22ba1ac3dc0>,
  <matplotlib.axis.XTick at 0x22b9e0bb130>,
  <matplotlib.axis.XTick at 0x22b99e04700>,
  <matplotlib.axis.XTick at 0x22b99e04e50>,
  <matplotlib.axis.XTick at 0x22b99dfe5e0>],
 [Text(0, 0, 'January'),
  Text(1, 0, 'February'),
  Text(2, 0, 'March'),
  Text(3, 0, 'April'),
  Text(4, 0, 'May'),
  Text(5, 0, 'June'),
  Text(6, 0, 'July'),
  Text(7, 0, 'August'),
  Text(8, 0, 'September'),
  Text(9, 0, 'October'),
  Text(10, 0, 'November'),
  Text(11, 0, 'December')])
```



In [64]:
```
# how seasons affect the harvest behaviour
# create a column for seasons
df6['season_harvest'] = 'other'

# assign the season to each plant based on the month it was planted
df6.loc[df6['harvested_month'].isin([6, 7, 8]), 'season_harvest'] = 'summer'
df6.loc[df6['harvested_month'].isin([9, 10, 11]), 'season_harvest'] = 'fall'
df6.loc[df6['harvested_month'].isin([12, 1, 2]), 'season_harvest'] = 'winter'
df6.loc[df6['harvested_month'].isin([3, 4, 5]), 'season_harvest'] = 'spring'
df6 = df6[df6['season_harvest'] != 'other']

harvest_counts_by_season = df6.groupby('season_harvest')['plant_title'].count()

harvest_counts_by_season.plot.bar()
```

Out[64]:
```
<AxesSubplot:xlabel='season_harvest'>
```



Identifying combinations based on planted_on date

In [ ]:
```
edges_list = []

# Iterate through the dataframe and add edges between nodes that are planted at the same time
for i, row in dfk.iterrows():
    for j, row2 in dfk.iterrows():
        if (row["planted_on"].date() == row2["planted_on"].date()) & (row["plant_title"] != row2["plant_title"]):
            edges_list.append({"plant1": row["plant_title"], "plant2": row2["plant_title"],"planted on": row2["planted_on"].date()})

# Create a dataframe to store the edges
if edges_list:
    edges_df = pd.DataFrame(edges_list)
else:
    edges_df = pd.DataFrame(columns=["plant1", "plant2","planted on"])
```

In [79]:
```
frames = []
for cube in intersection:
    dfk = df3[df3.plantcube == str(cube)]
    edges_list = []

    # Iterate through the dataframe and add edges between nodes that are planted at the same time
    for i, row in dfk.iterrows():
        for j, row2 in dfk.iterrows():
            if (row["planted_on"].date() == row2["planted_on"].date()) & (row["plant_title"] != row2["plant_title"]):
                edges_list.append({"plant1": row["plant_title"], "plant2": row2["plant_title"],"slot1": row["slot"], "slot2": row2["slot"],"planted on": row2["planted_on"].date(),"customer": row2["customer_name"],"plantcube

    # Create a dataframe to store the edges
    if edges_list:
        edges_df = pd.DataFrame(edges_list)

    frames.append(edges_df)

all_cubes = pd.concat(frames,axis=0, ignore_index=True)
```

In [84]:
```
all_cubes.head(10)
```

Out[84]:

	plant1	plant2	slot1	slot2	planted on	customer	plantcube
0	Rocket (Victoria)	Basil (Salvo)	b1	b3	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
1	Rocket (Victoria)	Basil (Salvo)	b1	b6	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
2	Rocket (Victoria)	Basil (Salvo)	b1	b9	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
3	Rocket (Victoria)	Basil (Salvo)	b1	a7	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
4	Rocket (Victoria)	Parsley (Lisette)	b1	a4	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
5	Rocket (Victoria)	Thyme (English Winter)	b1	a1	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
6	Rocket (Victoria)	Parsley (Lisette)	b1	a5	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
7	Rocket (Victoria)	Basil (Salvo)	b1	a8	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
8	Rocket (Victoria)	Dill (Dukat)	b1	a2	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
9	Rocket (Victoria)	Basil (Salvo)	b1	a3	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825

In [109]:
```
all_cubesr = all_cubes[['plant1','plant2','planted on','customer','plantcube']]
```

In [110]:
```
all_cubesr
```

Out[110]:

	plant1	plant2	planted on	customer	plantcube
0	Rocket (Victoria)	Basil (Salvo)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
1	Rocket (Victoria)	Basil (Salvo)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
2	Rocket (Victoria)	Basil (Salvo)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
3	Rocket (Victoria)	Basil (Salvo)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
4	Rocket (Victoria)	Parsley (Lisette)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
...	...	...	...	...	...
57099	Borage (Blue)	Summer Savory	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57100	Borage (Blue)	Red Basil (Red Rubin)	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57101	Borage (Blue)	Thyme (English Winter)	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57102	Borage (Blue)	Thyme (English Winter)	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57103	Borage (Blue)	Thyme (English Winter)	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48

57104 rows × 5 columns

In [111]:
```
# dropping the duplicate values

all_cubesr.drop_duplicates(keep='first', inplace=True)
```

In [112]:
```
all_cubesr
```

Out[112]:

	plant1	plant2	planted on	customer	plantcube
0	Rocket (Victoria)	Basil (Salvo)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
4	Rocket (Victoria)	Parsley (Lisette)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
5	Rocket (Victoria)	Thyme (English Winter)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
8	Rocket (Victoria)	Dill (Dukat)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
66	Basil (Salvo)	Rocket (Victoria)	2021-11-06	Patrick Löffler	5574dd8f-f66c-4905-a6ba-19f17ec8c825
...	...	...	...	...	...
57062	Borage (Blue)	Thai Basil (Siam Queen)	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57067	Borage (Blue)	Lemon Basil	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57068	Borage (Blue)	Red Basil (Red Rubin)	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57069	Borage (Blue)	Summer Savory	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48
57071	Borage (Blue)	Thyme (English Winter)	2021-11-25	Martina Hohenlohe	1ba27b1c-bd1f-4e78-bdd0-c36b26161e48

19524 rows × 5 columns

In [158]:
```
combination_dict = {}
for _, row in all_cubesr.iterrows():
    combination = (row['plant1'], row['plant2'])
    customer = row['customer']
    if combination in combination_dict:
        combination_dict[combination].append(customer)
    else:
        combination_dict[combination] = [customer]

#print(list(combination_dict.items())[0])
print(combination_dict)
```

a)): ['Patrick Löffler ', 'kuechen vb', 'Al Kuechen', 'Al Kuechen', 'Al Kuechen', 'Josef Raß ', 'Simon Brenzinger', 'Florian Schafhäutl', 'Florian Schafhäutl', 'Florian Schafhäutl', 'Florian Schafhäutl', 'Daniel Meyer', 'Ina Bernstein', 'Bettina Schmid', 'Nicole Bohn', 'Georg Harenberg', 'Sascha Breuer', 'Sascha Breuer', 'Alexander Hagen', 'Alexander Hagen', 'Martin Stettler', 'gabriele volmer', 'alexander negoschanu', 'Claudia Erlebach', 'Frank Montagnese ', 'Gordon McKenna', 'Andrey Vorontsov', 'Alex Regg', 'Dirk Ahlbrecht', 'tim kriese', 'Jörg Ehebrecht', 'Jörg Ehebrecht', 'dani n.', 'dani n.', 'Samyra Rücksties', 'Kirstin Flüssmeyer ', 'Ralf Thome ', 'Benjamin Ochocki ', 'Nicolas Holder ', 'Friedrich Büse', 'Celine Dyhr', 'Celine Dyhr', 'Harry Briel', 'Harry Briel', 'Martina Hohenlohe', 'Martina Hohenlohe'], ('Thyme (English Winter)', 'Basil (Salvo)'): ['Patrick Löffler ', 'peter Steinecker-Moog ', 'kevin von Holt', 'kuechen vb', 'Al Kuechen', 'Al Kuechen', 'Al Kuechen', 'Heike Müller', 'Heike Müller', 'Heike Müller', 'Florian Schafhäutl', 'Florian Schafhäutl', 'Daniel Meyer', 'Team SVYT Munich', 'Ina Bernstein', 'Nicole Bohn', 'Georg Harenberg', 'Marcus Kemper', 'Sascha Breuer', 'Alexander Hagen', 'Alexander Hagen', 'Martin Stettler', 'gabriele volmer', 'Matthias Fischbacher', 'Jörg Adrian', 'alexander negoschanu', 'Claudia Erlebach', 'Frank Montagnese ', 'Roderik Lauchner', 'Sebastian Schäfer ', 'Küchenwerk Simon', 'plant cube', 'Mitja Birlo', 'Jens Lehnemann', 'Ann-Julie Trepesch ', 'Ann-Julie Trepesch ', 'Andrey Vorontsov', 'B Keiner', 'stef cappelle', 'Alex Regg', 'Daye Bünting ', 'Dirk Ahlbrecht', 'Tobias Schmidt', 'Marco Maier', 'Marco Maier', 'Marco Maier', 'Dietmar Spath', 'Michael Winter', 'Michael Winter', 'Jörg Ehebrecht', 'Jörg Ehebrecht', 'dani n.', 'Samyra Rücksties', 'Ingo Wagner', 'Ralf Thome ', 'Monika Höger ', 'valerius Kachniaschwili', 'Susanne Fitz', 'Dirk Meyer', 'Peter Küstermann ', 'Sabine Morasch ', 'Sabine Morasch ', 'Maria Vogt', 'Maria Vogt', 'simone kring', 'Küchenhaus Ehrmann', 'AL Küchen', 'Celine Dyhr', 'Herta Neuhauser ', 'Jürgen Pannier', 'Harry Briel', 'Harry Briel'], ('Thyme (English Winter)', 'Parsley (Lisette)'): ['Patrick Löffler ', 'peter Steinecker-Moog ', 'kevin von Holt', 'Al Kuechen', 'Al Kuechen', 'Al Kuechen', 'Josef Raß ', 'Simon Brenzinger', 'Heike Müller', 'Heike Müller', 'Florian Schafhäutl', 'Florian Schafhäutl', 'Christian Baatz ', 'Daniel Meyer', 'Team SVYT Munich', 'Ina Bernstein', 'Michaela Lins-Dollhopf', 'Sascha Breuer', 'Alexander Hagen', 'Alexander Hagen', 'gabriele volmer', 'Matthias Fischbacher', 'Jörg Adrian', 'philip Hessl ', 'Frank Montagnese ', 'Roderik Lauchner', 'Sebastian Schäfer ', 'Küchenwerk Simon', 'Jennifer Lieberwirth', 'Mitja Birlo', 'Jens Lehnemann', 'Ann-Julie Trepesch ', 'Andrey Vorontsov', 'B Keiner', 'Dirk Ahlbrecht', 'Dirk Ahlbrecht', 'Tobias Schmidt', 'Marco Maier', 'Marco Maier', 'Marco Maier', 'Dietmar Spath', 'Jörg Ehebrecht', 'dani n.', 'fabienne biner', 'Samyra Rücksties', 'Ingo Wagner', 'Ralf Thome ', 'Ralf Thome ', 'Monika Höger ', 'Nicolas Holder ', 'valerius Kachniaschwili', 'Dirk Meyer', 'Sabine Morasch ', 'simone kring', 'Küchenhaus Ehrmann', 'AL Küchen', 'Herta Neuhauser ', 'Harry Briel', 'Harry Briel'], ('Thyme (English Winter)', 'Dill (Dukat)'): ['Patrick Löffler ', 'Josef Raß ', 'Josef Raß ', 'Heike Müller', 'Florian Schafhäutl', 'Florian Schafhäutl', 'Christian Baatz ', 'Team SVYT Munich', 'Luise Brandl', 'Sascha Breuer', 'Jörg Adrian', 'Roderik Lauchner', 'Gordon McKenna', 'Küchenwerk Simon', 'Ann-Julie Trepesch ', 'Ann-Julie Trepesch ', 'Andrey Vorontsov', 'Alex Regg', 'Dirk Ahlbrecht', 'Tobias Schmidt', 'Marco Maier', 'Marco Maier', 'Anton Farthofer1', 'Dietmar Spath', 'dani n.', 'Dirk Meyer', 'Sabine Morasch ', 'Maria Vogt', 'Celine Dyhr', 'Herta Neuhauser '], ('Dill (Dukat)', 'Rocket (Victoria)'): ['Patrick Löffler ', 'peter Steinecker-Moog ', 'peter Steinecker-Moog ', 'Jan Kuschnik', 'Benjamin Maerz', 'Josef Raß ', 'Simon Brenzinger', 'Frank Schneider ', 'Florian Schafhäutl', 'Florian Schafhäutl', 'Michaela Lins-Dollhopf', 'Nicole Bohn', 'Sascha Breuer', 'Björn Matschke', 'Gordon McKenna', 'Sebastian Schäfer ', 'Jörg Schröter', 'Jörg Schröter', 'plant cube', 'Andrey Vorontsov', 'Alex Regg', 'Heiko Himmelreich ', 'Dirk Ahlbrecht', 'Dagmar Rudolph-Weibezahl', 'c

In [113]:
```
group = all_cubesr.groupby(['plant1', 'plant2'])
grouped_df = group.agg({'planted on':'count', 'customer':'nunique'}).reset_index()
grouped_df
```

Out[113]:

	plant1	plant2	planted on	customer
0	Amaranth (Passion Variegated)	Anise Hyssop	1	1
1	Amaranth (Passion Variegated)	Basil (Salvo)	15	11
2	Amaranth (Passion Variegated)	Borage (Blue)	18	14
3	Amaranth (Passion Variegated)	Broccoli (Green Micro)	15	12
4	Amaranth (Passion Variegated)	Bronze Fennel	18	12
...	...	...	...	...
1903	Watercress	Summer Savory	1	1
1904	Watercress	Tasty Mustard (CN)	23	16
1905	Watercress	Tatsoi (Rozetto F1)	23	17
1906	Watercress	Thai Basil (Siam Queen)	6	5
1907	Watercress	Thyme (English Winter)	18	15

1908 rows × 4 columns

Network graph based on number of plantings for that combination

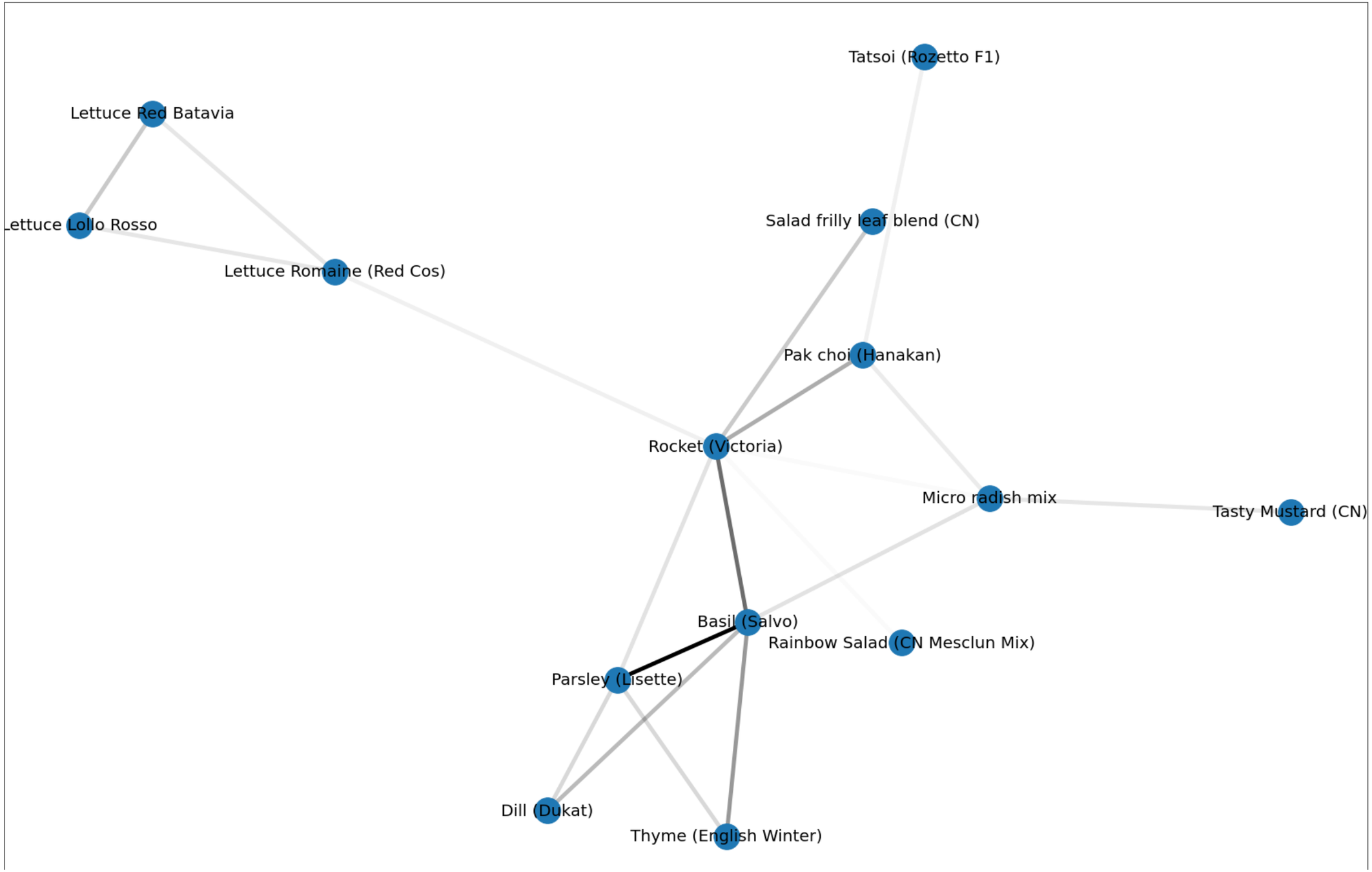
In [132]:
```
grouped_df.sort_values('planted on', ascending=False)
```

Out[132]:

	plant1	plant2	planted on	customer
1265	Parsley (Lisette)	Basil (Salvo)	103	72
113	Basil (Salvo)	Parsley (Lisette)	103	72
117	Basil (Salvo)	Rocket (Victoria)	81	58
1440	Rocket (Victoria)	Basil (Salvo)	81	58
1820	Thyme (English Winter)	Basil (Salvo)	72	58
...	...	...	...	...
1251	Pak choi (Hanakan)	Red Radish (Rioja)	1	1
1253	Pak choi (Hanakan)	Rocket Wild (Letizia)	1	1
334	Cinnamon Basil	Kale (CN KAL 1028)	1	1
1258	Pak choi (Hanakan)	Summer Savory	1	1
0	Amaranth (Passion Variegated)	Anise Hyssop	1	1

1908 rows × 4 columns

In [130]:
```
import matplotlib

# Create an empty graph
G = nx.Graph()

# Add the nodes to the graph
for plant in set(grouped_df['plant1'].tolist() + grouped_df['plant2'].tolist()):
    G.add_node(plant)

# Add the edges to the graph
for _, row in grouped_df.iterrows():
    G.add_edge(row['plant1'], row['plant2'], weight=row['planted on'])

# create a figure with a larger size
plt.figure(figsize=(30, 20))

# Draw the graph
#nx.draw(G, with_labels=True)

# use filter_edge function to filter edges with weight>20
filtered_graph = nx.Graph((u, v, d) for u, v, d in G.edges(data=True) if d['weight'] > 50)
# Draw the filtered graph
#nx.draw(filtered_graph, with_labels=True)

# Draw the filtered graph
pos = nx.spring_layout(filtered_graph)
nx.draw_networkx_nodes(filtered_graph, pos,node_size=1000)
nx.draw_networkx_labels(filtered_graph, pos, font_size = 20)

# retrieve edge weights
edge_weights = [d['weight'] for u, v, d in filtered_graph.edges(data=True)]

# normalize edge weights for alpha values
norm = matplotlib.colors.Normalize(vmin=min(edge_weights), vmax=max(edge_weights))

# draw edges with transparency
for u, v, d in filtered_graph.edges(data=True):
    alpha = norm(d['weight'])
    nx.draw_networkx_edges(filtered_graph, pos, edgelist=[(u, v)], alpha=alpha,width=5)

plt.show()
```


Tatsoi (Rozetto F1)
Lettuce Red Batavia
Lettuce Lollo Rosso
Salad frilly leaf blend (CN)
Lettuce Romaine (Red Cos)
Pak choi (Hanakan)
Rocket (Victoria)
Micro radish mix
Tasty Mustard (CN)
Basil (Salvo)
Rainbow Salad (CN Mesclun Mix)
Parsley (Lisette)
Dill (Dukat)
Thyme (English Winter)

In [162]:
```
#From the graph above, we can see strong connection between parsley and Basil, we can identify
#customer group who planted this combination from the combination dict

combination_dict.get(('Parsley (Lisette)', 'Basil (Salvo)'))
```

Out[162]:
```
['Patrick Löffler ',
 'peter Steinecker-Moog ',
 'kevin  von Holt',
 'Benjamin Maerz',
 'Al Kuechen',
 'Al Kuechen',
 'Al Kuechen',
 'Josef  Raß ',
 'Josef  Raß ',
 'Heike Müller',
 'Heike Müller',
 'Heike Müller',
 'Florian Schafhäutl',
 'Florian Schafhäutl',
 'Daniel Meyer',
 'Team SVYT Munich',
 'Klaus Malin K10-design',
 'Ina Bernstein',
 'Einrichtungshaus Gitterle',
 'Nicole Bohn',
 'Nicole Bohn',
 'Segmüller Pulheim',
 'Segmüller Pulheim',
 'Sascha Breuer',
 'Alexander Hagen',
 'Alexander Hagen',
 'gabriele volmer',
 'Matthias Fischbacher',
 'Jörg Adrian',
 'Jörg Adrian',
 'philip Hessl ',
 'michael schermann',
 'michael schermann',
 'michael schermann',
 'Frank  Montagnese ',
 'Olaf Neumann',
 'Atilla Akinli',
 'Martin Seipp',
 'Björn Matschke',
 'Roderik Lauchner',
 'Sebastian Schäfer ',
 'Sebastian Schäfer ',
 'Thomas Meusel',
 'Küchenwerk Simon',
 'Mike Hamel',
 'plant cube',
 'plant cube',
 'Mitja Birlo',
 'Jens Lehnemann',
 'Jens Lehnemann',
 'Ann-Julie Trepesch ',
 'Steve Sastalla',
 'Andrey Vorontsov',
 'B Keiner',
 'Heiko Himmelreich ',
 'Dirk Ahlbrecht',
 'Esther Kahl',
 'Esther Kahl',
 'Daniel Kanschat',
 'Tobias Schmidt',
 'Marco Maier',
 'Marco Maier',
```

```
 'Marco Maier',
 'Dietmar  Spath',
 'christian  jonas',
 'christian  jonas',
 'Jörg Ehebrecht',
 'Jörg Ehebrecht',
 'dani n.',
 'dani n.',
 'dani n.',
 'dani n.',
 'fabienne biner',
 'Samyra Rücksties',
 'Ingo Wagner',
 'Mauro Peter',
 'Andrea Altner',
 'Monika  Höger ',
 'Reddy  Reddy',
 'sabine Bohn ',
 'valerius Kachniaschwili',
 'valerius Kachniaschwili',
 'Friedrich Büse',
 'Susanne Fitz',
 'Dirk Meyer',
 'Sabine Morasch ',
 'Maria Vogt',
 'Maria Vogt',
 'Daniel Aschoff',
 'simone kring',
 'simone kring',
 'simone kring',
 'Küchenhaus Ehrmann',
 'AL Küchen',
 'AL Küchen',
 'norbert pasch',
 'Herta Neuhauser ',
 'Sven Reimers ',
 'Sven Reimers ',
 'Sven Reimers ',
 'manu bunke',
 'Harry Briel',
 'Harry Briel']
```

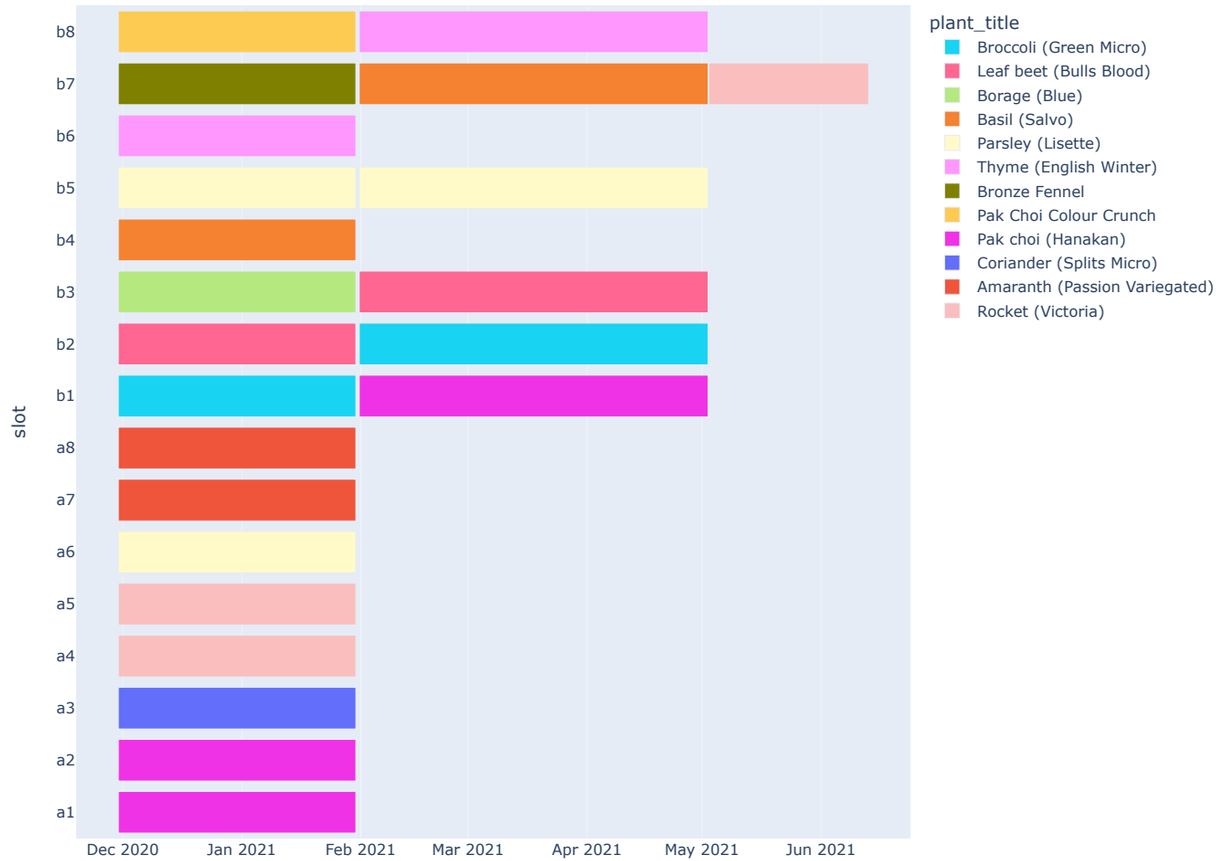
In [168]:
```
customer1_plantcubes = df3[df3['customer_name'] == 'Harry Briel']
customer1_plantcubes.plantcube.unique()
cus1_plantcube = df3[df3.plantcube == '775a49df-fc28-4579-bbf7-5c4eb7c9f402']
```

In [169]:
```
customer2_plantcubes = df3[df3['customer_name'] == 'manu bunke']
customer2_plantcubes.plantcube.unique()
cus2_plantcube = df3[df3.plantcube == 'c8c34875-b118-46c7-bedf-37d10594b300']
```

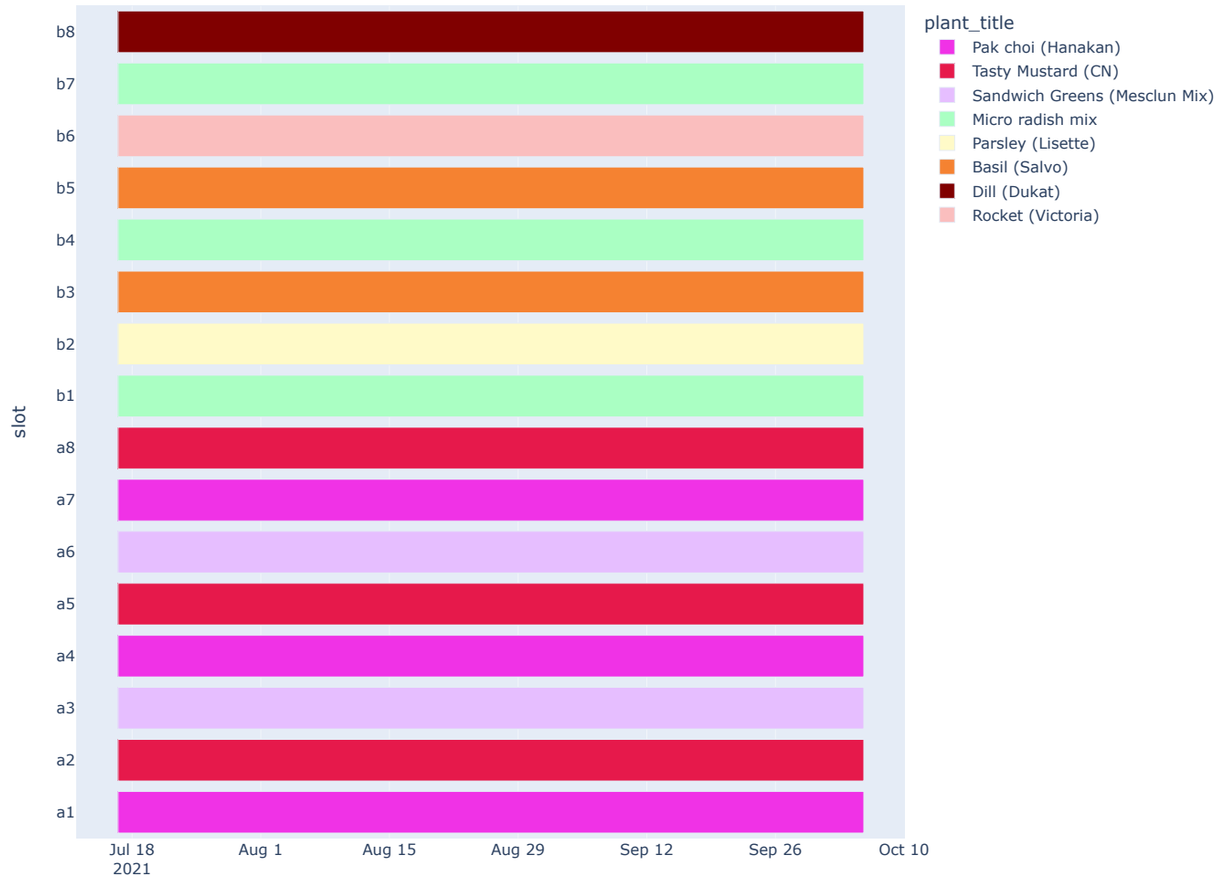
In [178]:
```
# different colour for different planttitles

plant_colors = {'Tasty Mustard (CN)': '#e6194b',
                'Mustard (Frizzie lizzie)': '#ffe119',
                'Kale (Scarlet)': '#4363d8',
                'Basil (Salvo)': '#f58231',
                'Pak Choi (Rubi)': '#911eb4',
                'Mustard (Red lace)': '#46f0f0',
                'Pak choi (Hanakan)': '#f032e6',
                'Red Radish (Rioja)': '#bcf60c',
                'Rocket (Victoria)': '#fabebe',
                'Tatsoi (Rozetto F1)': '#008080',
                'Sandwich Greens (Mesclun Mix)': '#e6beff',
                'Watercress': '#9a6324',
                'Parsley (Lisette)': '#fffac8',
                'Dill (Dukat)': '#800000',
                'Micro radish mix': '#aaffc3',
                'Bronze Fennel': '#808000',
                'Pak Choi (Red Lady F1)': '#ffd8b1',
                'Lettuce Romaine (Deronda)': '#000075',
                'Cinnamon Basil': '#a9a9a9',
                'Lemon Basil': '#ffffff',
                'Red Basil (Red Rubin)': '#000000',
                'Rainbow Salad (CN Mesclun Mix)': '#ff0000',
                'Salad frilly leaf blend (CN)': '#00ff00',
                'Sorrel (Red Veined)': '#0000ff',
                'Kale  (CN KAL 1028)': '#ff00ff'}
```

In [188]:
```
slot_order = ['a1','a2','a3','a4','a5','a6','a7','a8','b1','b2','b3','b4','b5','b6','b7','b8']
```

In [191]:
```
fig1 = px.timeline(cus1_plantcube, x_start="planted_on", x_end="harvested_on", y="slot",
      color='plant_title', color_discrete_map=plant_colors,height=800, width=1000)
fig1.update_yaxes(categoryorder = "array", categoryarray = slot_order)
fig1.show()
```



In [192]:
```
fig2 = px.timeline(cus2_plantcube, x_start="planted_on", x_end="harvested_on", y="slot",
      color='plant_title',  color_discrete_map=plant_colors,height=800, width=1000)
fig2.update_yaxes(categoryorder = "array", categoryarray = slot_order)
fig2.show()
```



Network graph based on number of customer for that combination

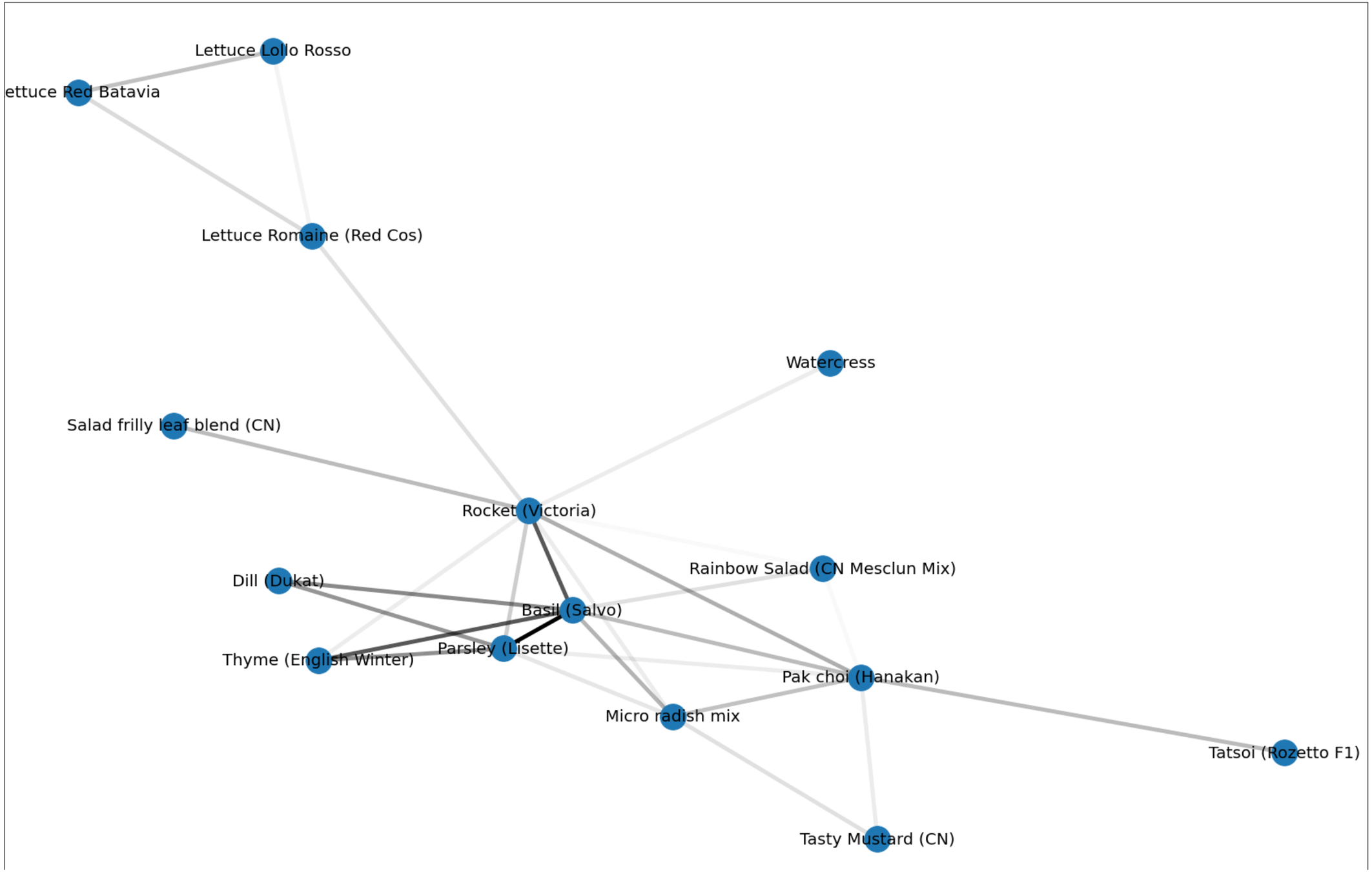
In [133]:
```
grouped_df.sort_values('customer', ascending=False)
```

Out[133]:

	plant1	plant2	planted on	customer
113	Basil (Salvo)	Parsley (Lisette)	103	72
1265	Parsley (Lisette)	Basil (Salvo)	103	72
1440	Rocket (Victoria)	Basil (Salvo)	81	58
1820	Thyme (English Winter)	Basil (Salvo)	72	58
117	Basil (Salvo)	Rocket (Victoria)	81	58
...	...	...	...	...
1069	Mustard (Red lace)	Bunter Senf	1	1
1070	Mustard (Red lace)	Pak Choi (Rubi)	1	1
1073	Mustard (Wasabina)	Asia Salat	1	1
1074	Mustard (Wasabina)	BBQ Salat	1	1
0	Amaranth (Passion Variegated)	Anise Hyssop	1	1

1908 rows × 4 columns

In [129]:
```
# Create an empty graph
G = nx.Graph()

# Add the nodes to the graph
for plant in set(grouped_df['plant1'].tolist() + grouped_df['plant2'].tolist()):
    G.add_node(plant)

# Add the edges to the graph
for _, row in grouped_df.iterrows():
    G.add_edge(row['plant1'], row['plant2'], weight=row['customer'])

# create a figure with a larger size
plt.figure(figsize=(30,20))

# Draw the graph
#nx.draw(G, with_labels=True)

# use filter_edge function to filter edges with weight>20
filtered_graph = nx.Graph((u, v, d) for u, v, d in G.edges(data=True) if d['weight'] > 30)
# Draw the filtered graph
#nx.draw(filtered_graph, with_labels=True)

# Draw the filtered graph
pos = nx.spring_layout(filtered_graph)
nx.draw_networkx_nodes(filtered_graph, pos,node_size=1000)
nx.draw_networkx_labels(filtered_graph, pos,font_size=20)

# retrieve edge weights
edge_weights = [d['weight'] for u, v, d in filtered_graph.edges(data=True)]

# normalize edge weights for alpha values
norm = matplotlib.colors.Normalize(vmin=min(edge_weights), vmax=max(edge_weights))

# draw edges with transparency
for u, v, d in filtered_graph.edges(data=True):
    alpha = norm(d['weight'])
    nx.draw_networkx_edges(filtered_graph, pos, edgelist=[(u, v)], alpha=alpha,width=5)

plt.show()
```


Lettuce Lollo Rosso
ettuce Red Batavia
Lettuce Romaine (Red Cos)
Watercress
Salad frilly leaf blend (CN)
Rocket (Victoria)
Rainbow Salad (CN Mesclun Mix)
Dill (Dukat)
Basil (Salvo)
Parsley (Lisette)
Thyme (English Winter)
Pak choi (Hanakan)
Micro radish mix
Tatsoi (Rozetto F1)
Tasty Mustard (CN)

Identifying combinations based on plants inside the cube at the same time

In [221]:
```
dfn = df3.copy()

# convert planted_on and harvested_on to datetime
dfn['planted_on'] = pd.to_datetime(dfn['planted_on'])
dfn['harvested_on'] = pd.to_datetime(dfn['harvested_on'])

# create a column for month
dfn['month'] = dfn['planted_on'].dt.to_period("M")

print(dfn)

# group by plantcube, month, and get the list of plant titles
# df_grouped = dfn.groupby(['plantcube','month'])['plant_title'].agg(lambda x: list(set(x))).reset_index()
```

```
                                plantcube  plant_id  \
21     00950202-abc4-43e9-acd2-25d269b63a3e       4.0
22     00950202-abc4-43e9-acd2-25d269b63a3e       4.0
23     00950202-abc4-43e9-acd2-25d269b63a3e      14.0
24     00950202-abc4-43e9-acd2-25d269b63a3e      10.0
27     00950202-abc4-43e9-acd2-25d269b63a3e      25.0
...                                     ...       ...
59016  ff20c1b6-748f-40a4-af4f-26512ace1e36      23.0
59017  ff20c1b6-748f-40a4-af4f-26512ace1e36      23.0
59018  ff20c1b6-748f-40a4-af4f-26512ace1e36      76.0
59019  ff20c1b6-748f-40a4-af4f-26512ace1e36      76.0
59020  ff20c1b6-748f-40a4-af4f-26512ace1e36      76.0

                   plant_title slot          planted_on        harvested_on  \
21     Mustard (Frizzie lizzie)   b2 2020-12-21 18:12:00 2021-01-13 09:44:00
22     Mustard (Frizzie lizzie)   b3 2020-12-21 18:15:00 2021-01-12 23:43:00
23               Kale (Scarlet)   b1 2020-12-21 18:22:00 2021-01-21 13:36:00
24                Basil (Salvo)   b4 2020-12-21 18:23:00 2021-01-13 09:45:00
27              Pak Choi (Rubi)   a1 2020-12-23 16:32:00 2021-01-20 19:35:00
...                         ...  ...                 ...                 ...
59016       Sorrel (Red Veined)   a5 2021-09-11 11:59:00 2021-10-10 17:54:00
59017       Sorrel (Red Veined)   a6 2021-09-11 11:59:00 2021-10-10 17:54:00
59018                Anise Hyssop   a7 2021-09-11 11:59:00 2021-10-10 17:54:00
59019                Anise Hyssop   a8 2021-09-11 11:59:00 2021-10-10 17:54:00
59020                Anise Hyssop   a9 2021-09-11 11:59:00 2021-10-10 17:54:00

       growth_days                                          owner  \
21           22.65  eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e
22           22.23  eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e
23           30.80  eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e
24           22.64  eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e
27           28.13  eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e
...            ...                                            ...
59016        29.25  eu-central-1:217cb235-7953-433a-bc8b-f676856e369e
59017        29.25  eu-central-1:217cb235-7953-433a-bc8b-f676856e369e
59018        29.25  eu-central-1:217cb235-7953-433a-bc8b-f676856e369e
59019        29.25  eu-central-1:217cb235-7953-433a-bc8b-f676856e369e
59020        29.25  eu-central-1:217cb235-7953-433a-bc8b-f676856e369e

       customer_name               customer_email customer_creation_date  \
21      Olaf Neumann    o.neumann@kuechenhelfer.de    2020-11-04 03:02:00
22      Olaf Neumann    o.neumann@kuechenhelfer.de    2020-11-04 03:02:00
23      Olaf Neumann    o.neumann@kuechenhelfer.de    2020-11-04 03:02:00
24      Olaf Neumann    o.neumann@kuechenhelfer.de    2020-11-04 03:02:00
27      Olaf Neumann    o.neumann@kuechenhelfer.de    2020-11-04 03:02:00
...              ...                           ...                    ...
59016  Marcus Kemper  familie@arbeitsplatz.digital    2019-06-18 18:16:00
59017  Marcus Kemper  familie@arbeitsplatz.digital    2019-06-18 18:16:00
59018  Marcus Kemper  familie@arbeitsplatz.digital    2019-06-18 18:16:00
59019  Marcus Kemper  familie@arbeitsplatz.digital    2019-06-18 18:16:00
59020  Marcus Kemper  familie@arbeitsplatz.digital    2019-06-18 18:16:00

      share_1 share_2 share_3  share_4  difference_in_days  duration    month
21        NaN     NaN     NaN      NaN               803.0      22.0  2020-12
22        NaN     NaN     NaN      NaN               803.0      22.0  2020-12
23        NaN     NaN     NaN      NaN               803.0      30.0  2020-12
24        NaN     NaN     NaN      NaN               803.0      22.0  2020-12
27        NaN     NaN     NaN      NaN               803.0      28.0  2020-12
...       ...     ...     ...      ...                 ...       ...      ...
59016     NaN     NaN     NaN      NaN              1307.0      29.0  2021-09
59017     NaN     NaN     NaN      NaN              1307.0      29.0  2021-09
59018     NaN     NaN     NaN      NaN              1307.0      29.0  2021-09
```

```
59019     NaN     NaN     NaN     NaN             1307.0      29.0  2021-09
59020     NaN     NaN     NaN     NaN             1307.0      29.0  2021-09

[6797 rows x 18 columns]
```

In [200]:
```
df_grouped
```

Out[200]:

	plantcube	month	plant_title
0	00950202-abc4-43e9-acd2-25d269b63a3e	2020-12	[Pak Choi (Rubi), Mustard (Frizzie lizzie), Ba...
1	00950202-abc4-43e9-acd2-25d269b63a3e	2021-01	[Pak choi (Hanakan), Tatsoi (Rozetto F1), Red ...
2	00950202-abc4-43e9-acd2-25d269b63a3e	2021-07	[Pak choi (Hanakan), Watercress, Sandwich Gree...
3	00950202-abc4-43e9-acd2-25d269b63a3e	2021-08	[Parsley (Lisette), Dill (Dukat), Basil (Salvo)]
4	03b0c1ac-20b7-49ad-8fab-21a77e47a00b	2022-01	[Pak choi (Hanakan), Rainbow Salad (CN Mesclun...
...	...	...	...
802	ff20c1b6-748f-40a4-af4f-26512ace1e36	2020-12	[Rocket (Victoria), Pak choi (Hanakan), Mustar...
803	ff20c1b6-748f-40a4-af4f-26512ace1e36	2021-01	[Basil (Salvo), Tatsoi (Rozetto F1), Rocket (V...
804	ff20c1b6-748f-40a4-af4f-26512ace1e36	2021-02	[Tatsoi (Rozetto F1)]
805	ff20c1b6-748f-40a4-af4f-26512ace1e36	2021-07	[Bronze Fennel, Tatsoi (Rozetto F1), Kale (CN...
806	ff20c1b6-748f-40a4-af4f-26512ace1e36	2021-09	[Tatsoi (Rozetto F1), Sorrel (Red Veined), Pak...

807 rows × 3 columns

In [ ]:
```
#itertools makes combinations like this
#[('Pak Choi (Rubi)', 'Mustard (Frizzie lizzie)'), ('Pak Choi (Rubi)', 'Basil (Salvo)'), ('Pak Choi (Rubi)', 'Kale (Scarlet)'),
#('Pak Choi (Rubi)', 'Mustard (Red Lace)'), ('Mustard (Frizzie lizzie)', 'Basil (Salvo)'), ('Mustard (Frizzie lizzie)', 'Kale (Scarlet)'),
#('Mustard (Frizzie lizzie)', 'Mustard (Red Lace)'), ('Basil (Salvo)', 'Kale (Scarlet)'), ('Basil (Salvo)', 'Mustard (Red Lace)'),
#('Kale (Scarlet)', 'Mustard (Red Lace)')]
```

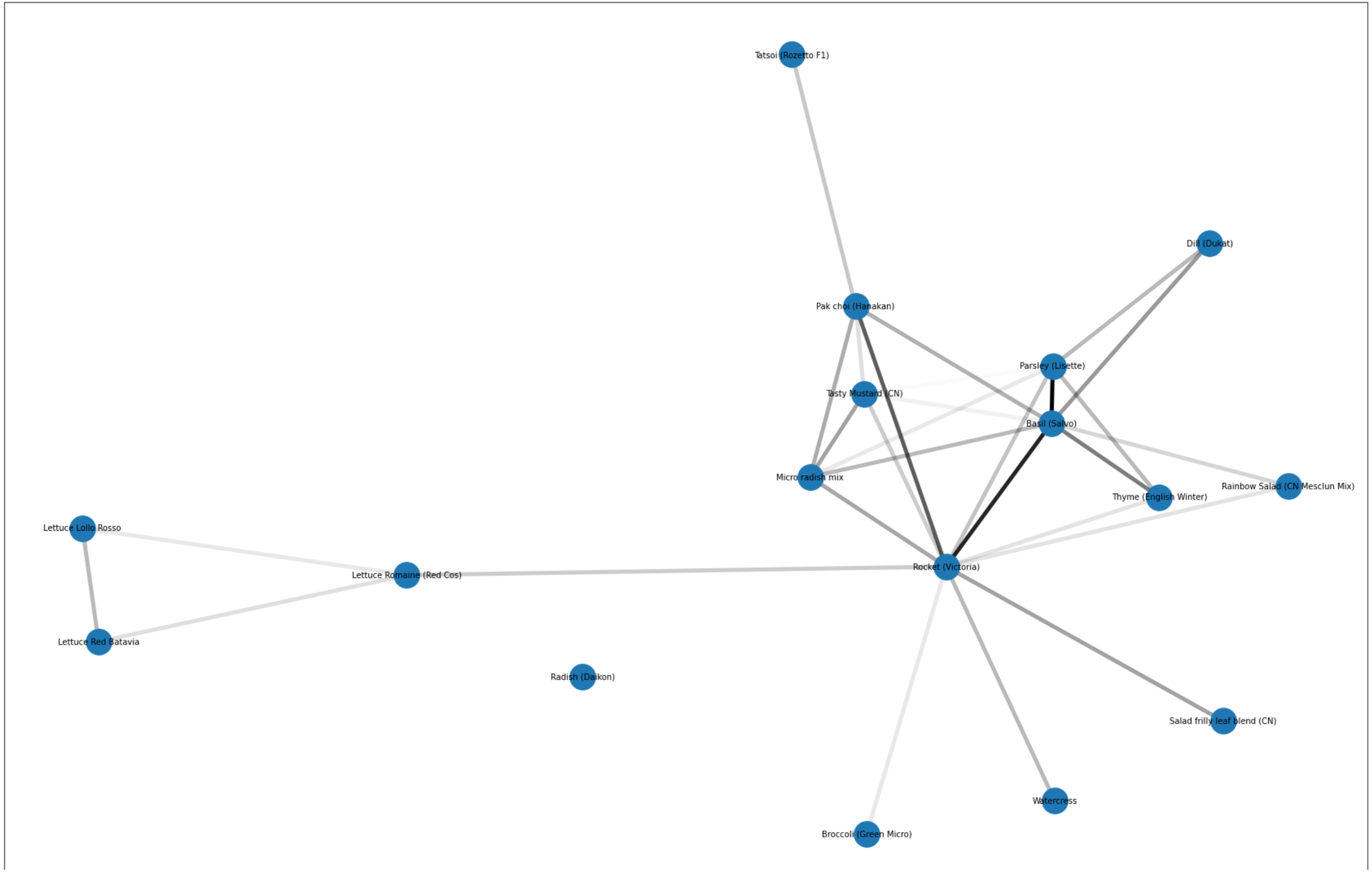
In [207]:
```
import itertools

combinations = []
for i, row in df_grouped.iterrows():
    plant_titles = row['plant_title']
    combs = list(itertools.combinations(plant_titles, 2))
    #print(combs)
    #break
    combinations.extend(combs)
```

In [208]:
```
print(combinations)
```

```
[('Pak Choi (Rubi)', 'Mustard (Frizzie lizzie)'), ('Pak Choi (Rubi)', 'Basil (Salvo)'), ('Pak Choi (Rubi)', 'Kale (Scarlet)'), ('Pak Choi (Rubi)', 'Mustard (Red lace)'), ('Mustard (Frizzie lizzie)', 'Basil (Salvo)'),
('Mustard (Frizzie lizzie)', 'Kale (Scarlet)'), ('Mustard (Frizzie lizzie)', 'Mustard (Red lace)'), ('Basil (Salvo)', 'Kale (Scarlet)'), ('Basil (Salvo)', 'Mustard (Red lace)'), ('Kale (Scarlet)', 'Mustard (Red lace)'),
('Pak choi (Hanakan)', 'Tatsoi (Rozetto F1)'), ('Pak choi (Hanakan)', 'Red Radish (Rioja)'), ('Pak choi (Hanakan)', 'Rocket (Victoria)'), ('Tatsoi (Rozetto F1)', 'Red Radish (Rioja)'), ('Tatsoi (Rozetto F1)', 'Rocket (V
ictoria)'), ('Red Radish (Rioja)', 'Rocket (Victoria)'), ('Pak choi (Hanakan)', 'Watercress'), ('Pak choi (Hanakan)', 'Sandwich Greens (Mesclun Mix)'), ('Watercress', 'Sandwich Greens (Mesclun Mix)'), ('Parsley (Lisett
e)', 'Dill (Dukat)'), ('Parsley (Lisette)', 'Basil (Salvo)'), ('Dill (Dukat)', 'Basil (Salvo)'), ('Pak choi (Hanakan)', 'Rainbow Salad (CN Mesclun Mix)'), ('Parsley (Lisette)', 'Basil (Salvo)'), ('Watercress', 'Lettuce
Romaine (Red Cos)'), ('Anise Hyssop', 'Rocket (Victoria)'), ('Anise Hyssop', 'Basil (Salvo)'), ('Rocket (Victoria)', 'Basil (Salvo)'), ('Pak choi (Hanakan)', 'Parsley (Lisette)'), ('Pak choi (Hanakan)', 'Dill (Dukat)'),
('Pak choi (Hanakan)', 'Basil (Salvo)'), ('Parsley (Lisette)', 'Dill (Dukat)'), ('Parsley (Lisette)', 'Basil (Salvo)'), ('Dill (Dukat)', 'Basil (Salvo)'), ('Bronze Fennel', 'Dill (Dukat)'), ('Tatsoi (Rozetto F1)', 'Pak
Choi (Red Lady F1)'), ('Salad frilly leaf blend (CN)', 'Tasty Mustard (CN)'), ('Basil (Salvo)', 'Sorrel (Red Veined)'), ('Basil (Salvo)', 'Rocket (Victoria)'), ('Basil (Salvo)', 'Thyme (English Winter)'), ('Basil (Salv
o)', 'Salad frilly leaf blend (CN)'), ('Basil (Salvo)', 'Lettuce Romaine (Deronda)'), ('Basil (Salvo)', 'Parsley (Lisette)'), ('Sorrel (Red Veined)', 'Rocket (Victoria)'), ('Sorrel (Red Veined)', 'Thyme (English Winte
r)'), ('Sorrel (Red Veined)', 'Salad frilly leaf blend (CN)'), ('Sorrel (Red Veined)', 'Lettuce Romaine (Deronda)'), ('Sorrel (Red Veined)', 'Parsley (Lisette)'), ('Rocket (Victoria)', 'Thyme (English Winter)'), ('Rocke
t (Victoria)', 'Salad frilly leaf blend (CN)'), ('Rocket (Victoria)', 'Lettuce Romaine (Deronda)'), ('Rocket (Victoria)', 'Parsley (Lisette)'), ('Thyme (English Winter)', 'Salad frilly leaf blend (CN)'), ('Thyme (Englis
h Winter)', 'Lettuce Romaine (Deronda)'), ('Thyme (English Winter)', 'Parsley (Lisette)'), ('Salad frilly leaf blend (CN)', 'Lettuce Romaine (Deronda)'), ('Salad frilly leaf blend (CN)', 'Parsley (Lisette)'), ('Lettuce
Romaine (Deronda)', 'Parsley (Lisette)'), ('Lettuce Tango', 'Lettuce Romaine (Red Cos)'), ('Lettuce Tango', 'Basil (Salvo)'), ('Lettuce Tango', 'Lettuce Red Batavia'), ('Lettuce Tango', 'Rocket (Victoria)'), ('Lettuce T
ango', 'Lettuce Cerbiatta'), ('Lettuce Tango', 'Watercress'), ('Lettuce Tango', 'Lettuce Little Gem'), ('Lettuce Tango', 'Lettuce Lollo Rosso'), ('Lettuce Romaine (Red Cos)', 'Basil (Salvo)'), ('Lettuce Romaine (Red Co
s)', 'Lettuce Red Batavia'), ('Lettuce Romaine (Red Cos)', 'Rocket (Victoria)'), ('Lettuce Romaine (Red Cos)', 'Lettuce Cerbiatta'), ('Lettuce Romaine (Red Cos)', 'Watercress'), ('Lettuce Romaine (Red Cos)', 'Lettuce Li
ttle Gem'), ('Lettuce Romaine (Red Cos)', 'Lettuce Lollo Rosso'), ('Basil (Salvo)', 'Lettuce Red Batavia'), ('Basil (Salvo)', 'Rocket (Victoria)'), ('Basil (Salvo)', 'Lettuce Cerbiatta'), ('Basil (Salvo)', 'Watercres
s'), ('Basil (Salvo)', 'Lettuce Little Gem'), ('Basil (Salvo)', 'Lettuce Lollo Rosso'), ('Lettuce Red Batavia', 'Rocket (Victoria)'), ('Lettuce Red Batavia', 'Lettuce Cerbiatta'), ('Lettuce Red Batavia', 'Watercress'),
('Lettuce Red Batavia', 'Lettuce Little Gem'), ('Lettuce Red Batavia', 'Lettuce Lollo Rosso'), ('Rocket (Victoria)', 'Lettuce Cerbiatta'), ('Rocket (Victoria)', 'Watercress'), ('Rocket (Victoria)', 'Lettuce Little Ge
m'), ('Rocket (Victoria)', 'Lettuce Lollo Rosso'), ('Lettuce Cerbiatta', 'Watercress'), ('Lettuce Cerbiatta', 'Lettuce Little Gem'), ('Lettuce Cerbiatta', 'Lettuce Lollo Rosso'), ('Watercress', 'Lettuce Little Gem'),
```

In [220]:

```
import networkx as nx

# Create a new DataFrame with the plant combinations
df_combinations = pd.DataFrame(combinations, columns=['plant1', 'plant2'])

# Create an empty graph
G = nx.Graph()

# Add nodes to the graph
for plant in set(df_combinations['plant1'].tolist() + df_combinations['plant2'].tolist()):
    G.add_node(plant)

# Add edges to the graph with weight as the count of the combination
for i, row in df_combinations.iterrows():
    plant1, plant2 = row['plant1'], row['plant2']
    if G.has_edge(plant1, plant2):
        G[plant1][plant2]['weight'] += 1
    else:
        G.add_edge(plant1, plant2, weight=1)

# create a figure with a larger size
plt.figure(figsize=(30, 20))

# Draw the graph
#nx.draw(G, with_labels=True)

# use filter_edge function to filter edges with weight>30
filtered_graph = nx.Graph((u, v, d) for u, v, d in G.edges(data=True) if d['weight'] > 50)
# Draw the filtered graph
#nx.draw(filtered_graph, with_labels=True)

# Draw the filtered graph
pos = nx.spring_layout(filtered_graph)
nx.draw_networkx_nodes(filtered_graph, pos,node_size=1000)
nx.draw_networkx_labels(filtered_graph, pos,font_size=10)

# retrieve edge weights
edge_weights = [d['weight'] for u, v, d in filtered_graph.edges(data=True)]

# normalize edge weights for alpha values
norm = matplotlib.colors.Normalize(vmin=min(edge_weights), vmax=max(edge_weights))

# draw edges with transparency
for u, v, d in filtered_graph.edges(data=True):
    alpha = norm(d['weight'])
    nx.draw_networkx_edges(filtered_graph, pos, edgelist=[(u, v)], alpha=alpha,width=5)

plt.show()
```


Tatsoi (Rozetto F1)
Dill (Dukat)
Pak choi (Hanakan)
Parsley (Lisette)
Tasty Mustard (CN)
Basil (Salvo)
Micro radish mix
Thyme (English Winter)
Rainbow Salad (CN Mesclun Mix)
Lettuce Lollo Rosso
Lettuce Romaine (Red Cos)
Rocket (Victoria)
Lettuce Red Batavia
Radish (Daikon)
Salad frilly leaf blend (CN)
Watercress
Broccoli (Green Micro)

In [ ]: