In [370]:
```
#imports
import pandas as pd
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx
```

In [371]:
```
data = pd.read_csv('condensed_data_by_plants_02_01_2023.csv')
```

In [372]:
```
data.head()
```

Out[372]:

	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 [373]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59298 entries, 0 to 59297
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   plantcube              59298 non-null  object
 1   plant_id               43309 non-null  float64
 2   plant_title            59298 non-null  object
 3   slot                   59298 non-null  object
 4   planted_on             59298 non-null  object
 5   harvested_on           42000 non-null  object
 6   growth_days            42000 non-null  float64
 7   owner                  56219 non-null  object
 8   customer_name          55758 non-null  object
 9   customer_email         55758 non-null  object
 10  customer_creation_date  55758 non-null  object
 11  share_1                4188 non-null   object
 12  share_2                412 non-null    object
 13  share_3                69 non-null     object
 14  share_4                0 non-null      float64
dtypes: float64(3), object(12)
memory usage: 6.8+ MB
```

In [374]:
```
df = data.copy()
```

In [375]:
```
actual_df_count = df.plantcube.unique().size
actual_df_count
```

Out[375]:
```
962
```

Long term customers

In [376]:
```
df['customer_creation_date'] = pd.to_datetime(df['customer_creation_date'], format='%d.%m.%Y %H:%M')
```

In [377]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59298 entries, 0 to 59297
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   plantcube              59298 non-null  object
 1   plant_id               43309 non-null  float64
 2   plant_title            59298 non-null  object
 3   slot                   59298 non-null  object
 4   planted_on             59298 non-null  object
 5   harvested_on           42000 non-null  object
 6   growth_days            42000 non-null  float64
 7   owner                  56219 non-null  object
 8   customer_name          55758 non-null  object
 9   customer_email         55758 non-null  object
 10  customer_creation_date  55758 non-null  datetime64[ns]
 11  share_1                4188 non-null   object
 12  share_2                412 non-null    object
 13  share_3                69 non-null     object
 14  share_4                0 non-null      float64
dtypes: datetime64[ns](1), float64(3), object(11)
memory usage: 6.8+ MB
```

In [378]:
```
df['difference_in_days'] = df['customer_creation_date'].apply(lambda x: (datetime.now() - x).days)
```

In [379]:
```
df.head()
```

Out[379]:

	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
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	2022-08-25 20:33:00	NaN	NaN	NaN	NaN	132.0
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	2022-08-25 20:33:00	NaN	NaN	NaN	NaN	132.0
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	2022-08-25 20:33:00	NaN	NaN	NaN	NaN	132.0
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	2022-08-25 20:33:00	NaN	NaN	NaN	NaN	132.0
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	2022-08-25 20:33:00	NaN	NaN	NaN	NaN	132.0

In [380]:
```
# If the difference in date is more than 1 year, we can take those records

long_term_customers = df[df['difference_in_days'] > 365]
```

In [381]:
```
long_term_customers
```

Out[381]:

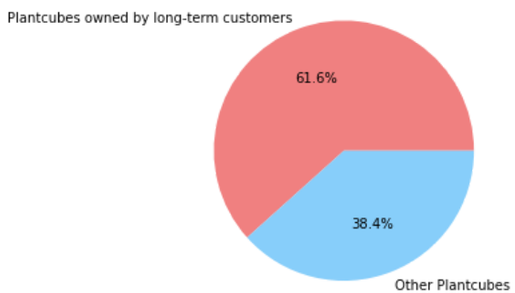
	plantcube	plant_id	plant_title	slot	planted_on	harvested_on	growth_days	owner	customer_name	customer_email	customer_creation_date	share_1	share_2	share_3	share_4	difference_in_days
21	00950202-abc4-43e9-acd2-25d269b63a3e	4.0	Mustard (Frizzie lizzie)	b2	21.12.2020 18:12	13.01.2021 09:44	22.65	eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e	Olaf Neumann	o.neumann@kuechenhelfer.de	2020-11-04 03:02:00	NaN	NaN	NaN	NaN	792.0
22	00950202-abc4-43e9-acd2-25d269b63a3e	4.0	Mustard (Frizzie lizzie)	b3	21.12.2020 18:15	12.01.2021 23:43	22.23	eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e	Olaf Neumann	o.neumann@kuechenhelfer.de	2020-11-04 03:02:00	NaN	NaN	NaN	NaN	792.0
23	00950202-abc4-43e9-acd2-25d269b63a3e	14.0	Kale (Scarlet)	b1	21.12.2020 18:22	21.01.2021 13:36	30.80	eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e	Olaf Neumann	o.neumann@kuechenhelfer.de	2020-11-04 03:02:00	NaN	NaN	NaN	NaN	792.0
24	00950202-abc4-43e9-acd2-25d269b63a3e	10.0	Basil (Salvo)	b4	21.12.2020 18:23	13.01.2021 09:45	22.64	eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e	Olaf Neumann	o.neumann@kuechenhelfer.de	2020-11-04 03:02:00	NaN	NaN	NaN	NaN	792.0
25	00950202-abc4-43e9-acd2-25d269b63a3e	4.0	Mustard (Frizzie lizzie)	a2	23.12.2020 16:30	23.12.2020 16:32	0.00	eu-central-1:7e51bbfe-5541-4bc7-981c-177d0666627e	Olaf Neumann	o.neumann@kuechenhelfer.de	2020-11-04 03:02:00	NaN	NaN	NaN	NaN	792.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59272	ffdbb1ca-89ff-4a5a-bdcb-5009a15c5a75	NaN	Currently Empty	b5	09.04.2022 13:03	NaN	NaN	eu-central-1:6b74aeca-8fac-4db6-9904-1e40b45d9ce6	Sabrina Gutmann	sabrina.gutmann@hotmail.com	2019-12-05 20:14:00	NaN	NaN	NaN	NaN	1126.0
59273	ffdbb1ca-89ff-4a5a-bdcb-5009a15c5a75	NaN	Currently Empty	b6	09.04.2022 13:03	NaN	NaN	eu-central-1:6b74aeca-8fac-4db6-9904-1e40b45d9ce6	Sabrina Gutmann	sabrina.gutmann@hotmail.com	2019-12-05 20:14:00	NaN	NaN	NaN	NaN	1126.0
59274	ffdbb1ca-89ff-4a5a-bdcb-5009a15c5a75	NaN	Currently Empty	b7	09.04.2022 13:03	NaN	NaN	eu-central-1:6b74aeca-8fac-4db6-9904-1e40b45d9ce6	Sabrina Gutmann	sabrina.gutmann@hotmail.com	2019-12-05 20:14:00	NaN	NaN	NaN	NaN	1126.0
59275	ffdbb1ca-89ff-4a5a-bdcb-5009a15c5a75	NaN	Currently Empty	b8	09.04.2022 13:03	NaN	NaN	eu-central-1:6b74aeca-8fac-4db6-9904-1e40b45d9ce6	Sabrina Gutmann	sabrina.gutmann@hotmail.com	2019-12-05 20:14:00	NaN	NaN	NaN	NaN	1126.0
59276	ffdbb1ca-89ff-4a5a-bdcb-5009a15c5a75	NaN	Currently Empty	b9	09.04.2022 13:03	NaN	NaN	eu-central-1:6b74aeca-8fac-4db6-9904-1e40b45d9ce6	Sabrina Gutmann	sabrina.gutmann@hotmail.com	2019-12-05 20:14:00	NaN	NaN	NaN	NaN	1126.0

45905 rows × 16 columns

In [382]:
```
long_term_customers_count = long_term_customers.plantcube.unique().size
long_term_customers_count
```

Out[382]: 593

In [383]:
```
#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()
```



Active plantcubes

Assumption 1: Active Plantcube: At least 1 planting of installed Plantcube within last 60 days https://agrilution.atlassian.net/wiki/spaces/MAR/pages/1851719789/Def.+of+Active+Plantcube+and+Utilization (https://agrilution.atlassian.net/wiki/spaces/MAR/pages/1851719789/Def.+of+Active+Plantcube+and+Utilization)

In [384]:
```
df1 = long_term_customers.copy()

# Parse the planted_on and harvested_on columns to date objects
df1['planted_on'] = pd.to_datetime(df1['planted_on'], format='%d.%m.%Y %H:%M')
df1['harvested_on'] = pd.to_datetime(df1['harvested_on'], format='%d.%m.%Y %H:%M')

# Calculate the difference between today's date and the planted_on date in days
df1['difference_in_days'] = df1['planted_on'].apply(lambda x: (datetime.now() - x).days)

# Filter the rows where the difference is less than 60 days and the harvested_on date is NaT
recently_planted = df1[(df1['difference_in_days'] < 60) & (df1['harvested_on'].isnull())]

# Group the plant cubes by their ID
grouped = recently_planted.groupby('plantcube')

# Count the number of slots for each plant cube
counts = grouped['plant_id'].count()

# Filter the plant cubes that have at least one slot with a plant ID
active_plant_cubes1 = counts[counts > 0]

active_plant_cube_ids1 = active_plant_cubes1.index

active_plant_cubes_df1 = df1[df1['plantcube'].isin(active_plant_cube_ids1)]
```

In [385]:
```
active_plant_cubes_df1.head()
```

Out[385]:

	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
2138	092d10c0-abb5-4cdb-9efa-fa95a6776172	31.0	Thai Basil (Siam Queen)	b1	2021-06-14 10:38:00	2021-10-19 21:06:00	127.44	eu-central-1:9a313c76-0bae-4279-b8a2-49f0ce200f60	Uwe Handke	kontakt@restaurant-gruenspecht.de	2019-12-20 12:39:00	NaN	NaN	NaN	NaN	570
2139	092d10c0-abb5-4cdb-9efa-fa95a6776172	31.0	Thai Basil (Siam Queen)	b4	2021-06-14 10:38:00	2021-10-19 21:05:00	127.44	eu-central-1:9a313c76-0bae-4279-b8a2-49f0ce200f60	Uwe Handke	kontakt@restaurant-gruenspecht.de	2019-12-20 12:39:00	NaN	NaN	NaN	NaN	570
2140	092d10c0-abb5-4cdb-9efa-fa95a6776172	31.0	Thai Basil (Siam Queen)	b7	2021-06-14 10:38:00	2021-10-19 21:07:00	127.44	eu-central-1:9a313c76-0bae-4279-b8a2-49f0ce200f60	Uwe Handke	kontakt@restaurant-gruenspecht.de	2019-12-20 12:39:00	NaN	NaN	NaN	NaN	570
2141	092d10c0-abb5-4cdb-9efa-fa95a6776172	92.0	Amaranth (Passion Variegated)	b2	2021-06-14 10:39:00	2021-07-16 16:31:00	32.24	eu-central-1:9a313c76-0bae-4279-b8a2-49f0ce200f60	Uwe Handke	kontakt@restaurant-gruenspecht.de	2019-12-20 12:39:00	NaN	NaN	NaN	NaN	570
2142	092d10c0-abb5-4cdb-9efa-fa95a6776172	92.0	Amaranth (Passion Variegated)	b5	2021-06-14 10:39:00	2021-07-16 16:31:00	32.24	eu-central-1:9a313c76-0bae-4279-b8a2-49f0ce200f60	Uwe Handke	kontakt@restaurant-gruenspecht.de	2019-12-20 12:39:00	NaN	NaN	NaN	NaN	570

In [386]:
```
active_plant_cubes_df1.plantcube.unique().size
```

Out[386]: 41

Assumption 2: inactive plant cubes are plantcubes that have not had a planting or harvesting within the last 6 months. If there is a planting or harvesting take place in the plantcube, in the last 6 months, then it is active

In [387]:
```
df2 = long_term_customers.copy()

# Parse the planted_on and harvested_on columns to date objects
df2['planted_on'] = pd.to_datetime(df2['planted_on'], format='%d.%m.%Y %H:%M')
df2['harvested_on'] = pd.to_datetime(df2['harvested_on'], format='%d.%m.%Y %H:%M')

# Calculate the difference between today's date and the planted_on and harvested_on dates in days
df2['planted_difference_in_days'] = df2['planted_on'].apply(lambda x: (datetime.now() - x).days)
df2['harvested_difference_in_days'] = df2['harvested_on'].apply(lambda x: (datetime.now() - x).days)

# Filter the rows where the difference is less than 180 days for either planted_on or harvested_on
# 6 months = 180 days
active = df2[(df2['planted_difference_in_days'] < 180) | (df2['harvested_difference_in_days'] < 180)]

# Group the plant cubes by their ID
active_grouped = active.groupby('plantcube')

# Count the number of slots for each plant cube
active_counts = active_grouped['plant_id'].count()

# Filter the plant cubes that have at least one slot with a plant ID
active_plant_cubes2 = active_counts[active_counts > 0]

active_plant_cube_ids2 = active_plant_cubes2.index

active_plant_cubes_df2 = df2[df2['plantcube'].isin(active_plant_cube_ids2)]
```

In [388]:
```
active_plant_cubes_df2.head()
```

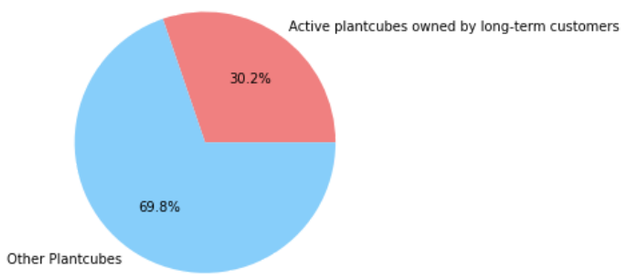
Out[388]:

	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	planted_difference_in_days	har
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	617.0	564	
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	617.0	557	
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	617.0	557	
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	617.0	557	
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	617.0	553	

In [389]:
```
active_plant_cubes_df2.plantcube.unique().size
```

Out[389]: 291

In [390]:
```
#visualization

# total plant cubes
total_plant_cubes = df['plantcube'].nunique()
# Number of plantcubes owned by long term customers
active_long_term_plant_cubes = active_plant_cubes_df2['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()
```

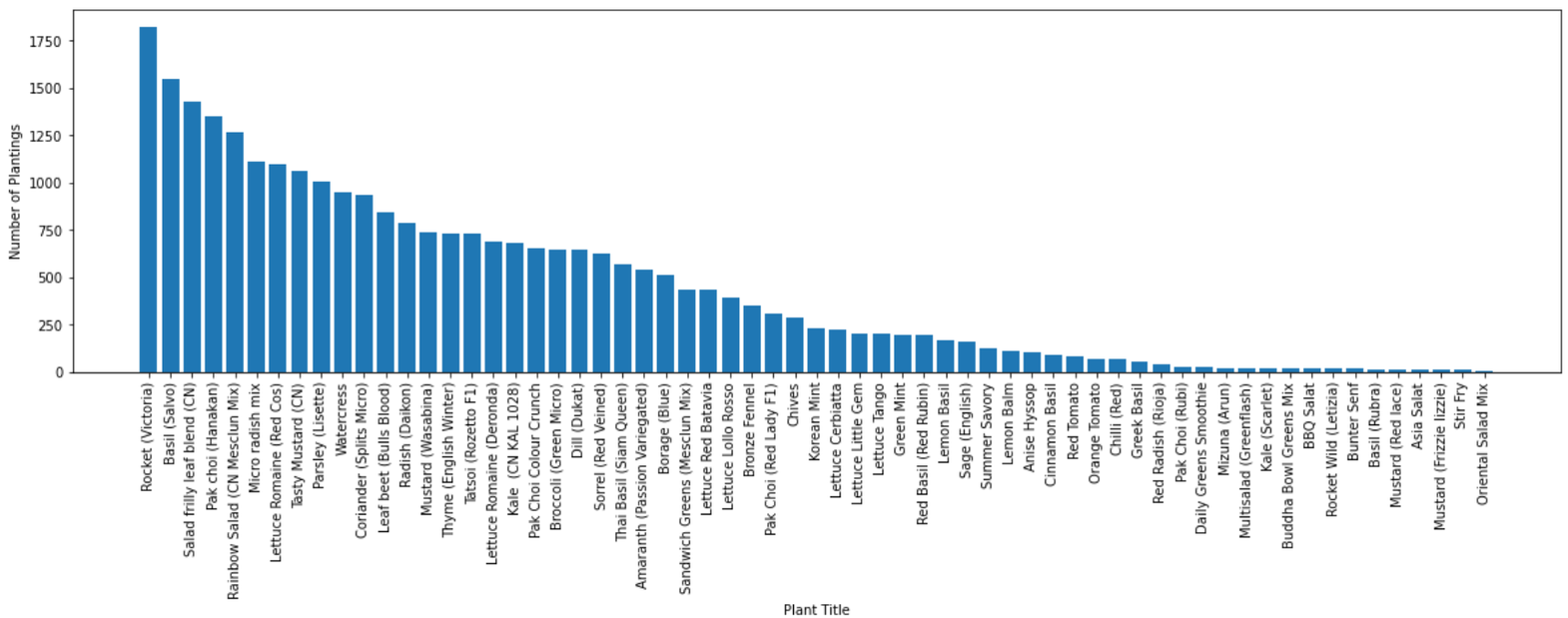


Analysis: what is planted more frequently and how often is it harvested

In [391]:
```
df3 = active_plant_cubes_df2.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']

# 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
```

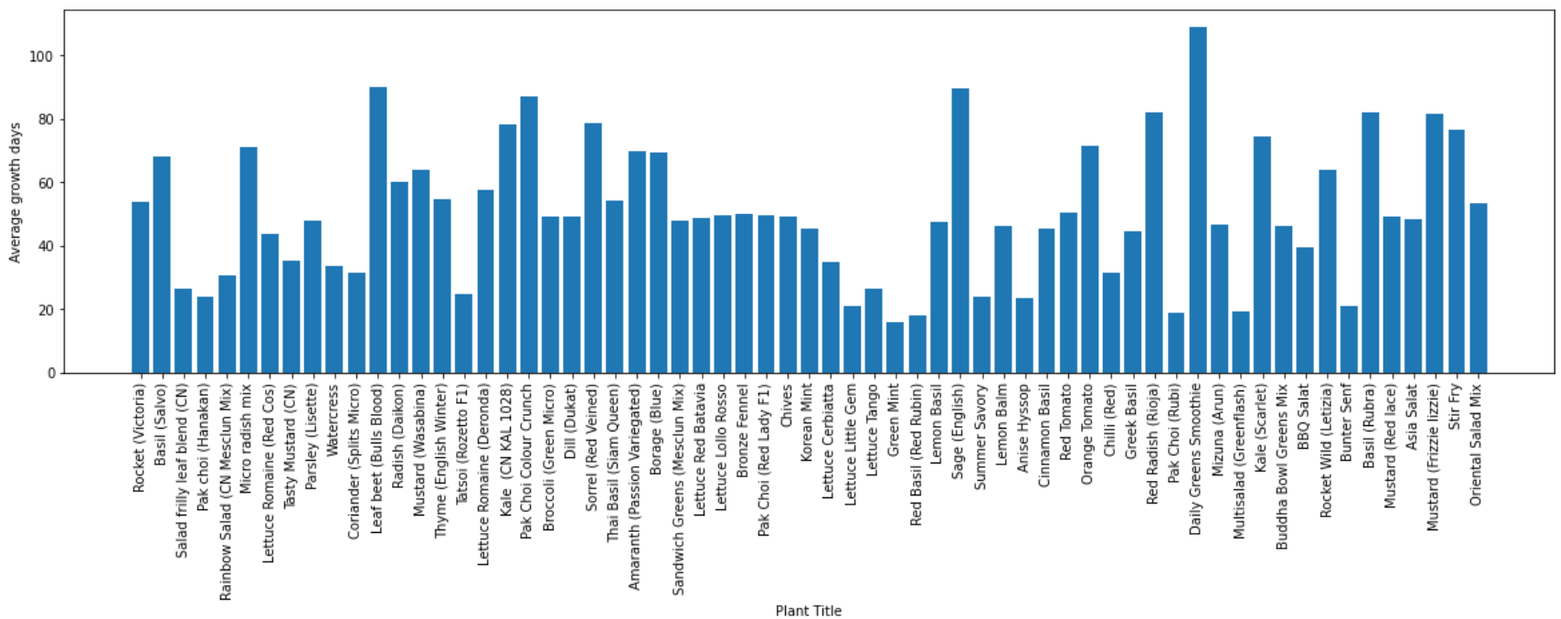
Out[391]:
```
plant_title
Rocket (Victoria)                1823
Basil (Salvo)                    1545
Salad frilly leaf blend (CN)     1428
Pak choi (Hanakan)               1349
Rainbow Salad (CN Mesclun Mix)   1265
                                 ...
Mustard (Red lace)                 12
Asia Salat                         12
Mustard (Frizzie lizzie)           11
Stir Fry                            9
Oriental Salad Mix                  2
Name: plant_title, Length: 63, dtype: int64
```

In [392]:
```
# 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()
```

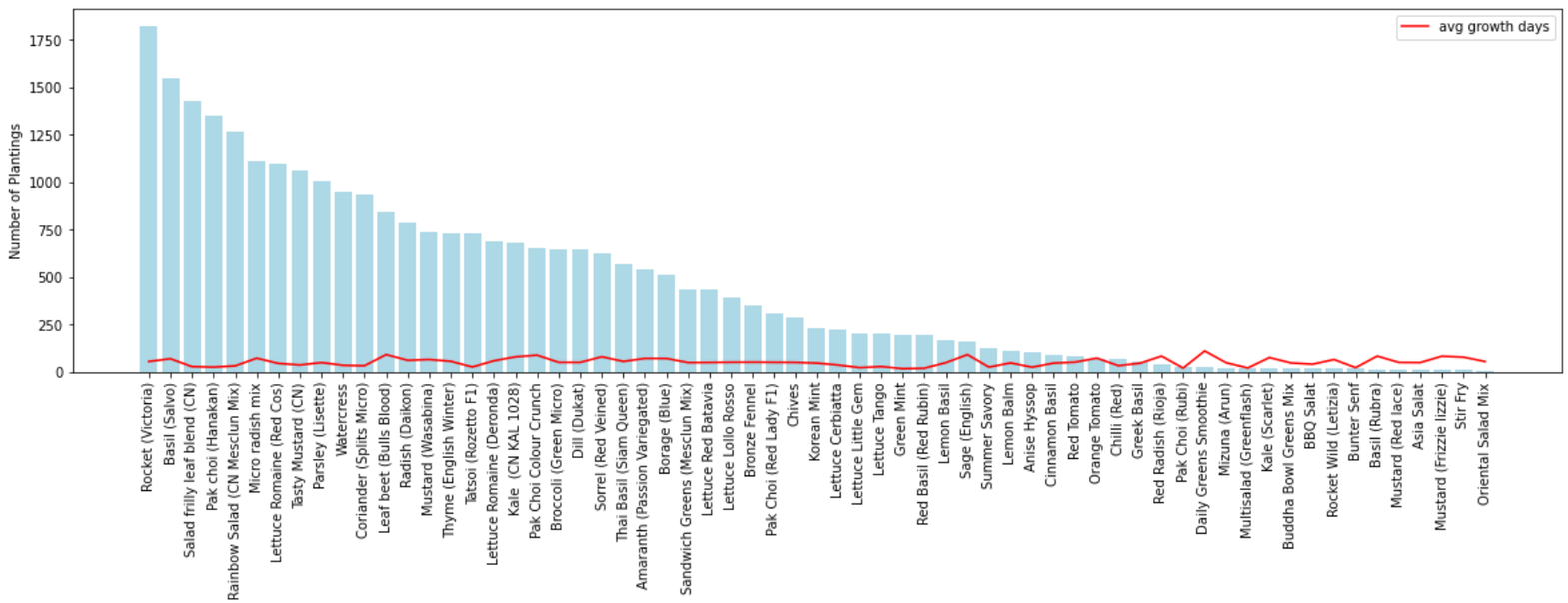


In [393]:
```
# Calculate the average number of growth days for each plant
avg_growth_days = group_plant_title['growth_days'].mean()
```

In [394]:
```
# 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 [395]:
```
# 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
ax.set_ylabel('Number of Plantings')

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

plt.legend(loc="upper right")

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

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



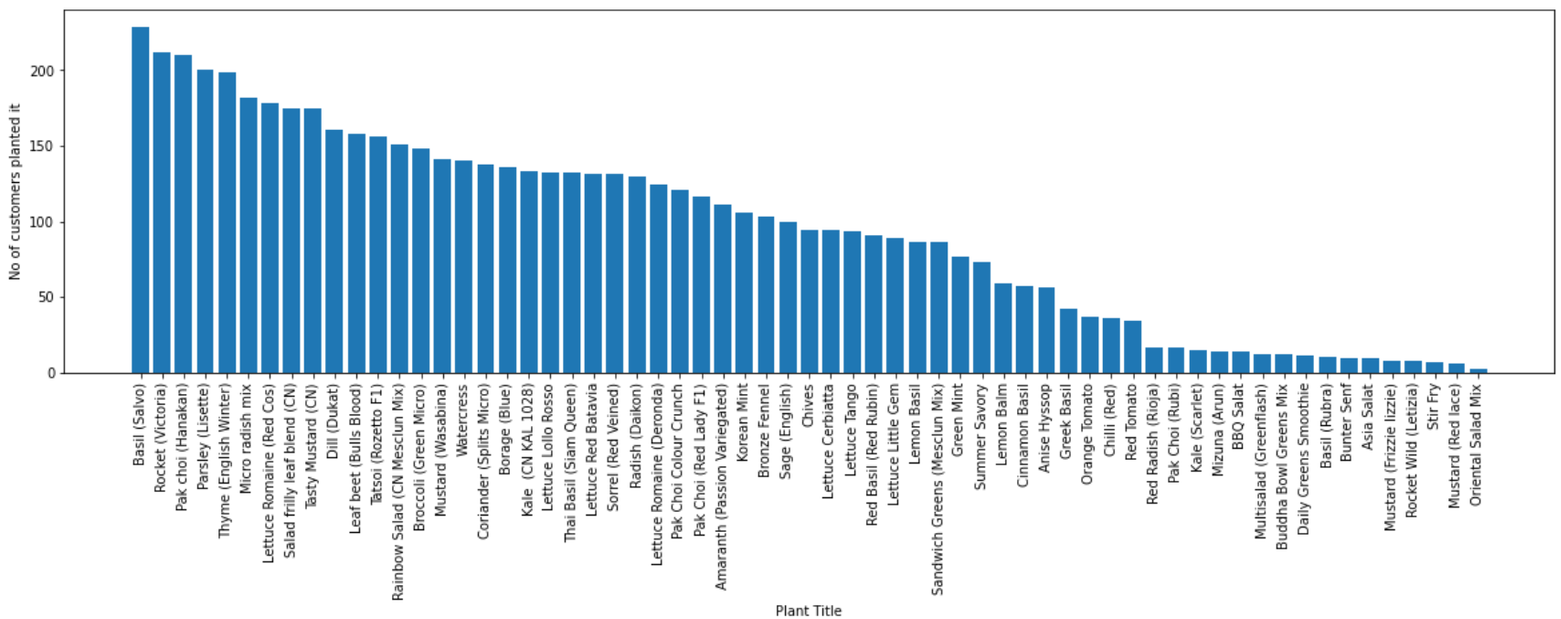
In [396]:
```
#customer name

# 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: 276
```

In [397]:
```
# 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[397]:
```
plant_title
Basil (Salvo)               229
Rocket (Victoria)           212
Pak choi (Hanakan)          210
Parsley (Lisette)           200
Thyme (English Winter)      199
                           ...
Mustard (Frizzie lizzie)      8
Rocket Wild (Letizia)         8
Stir Fry                      7
Mustard (Red lace)            6
Oriental Salad Mix            2
Name: customer_name, Length: 63, dtype: int64
```

In [398]:
```
# 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()
```



In [399]:
```
df3
```

Out[399]:

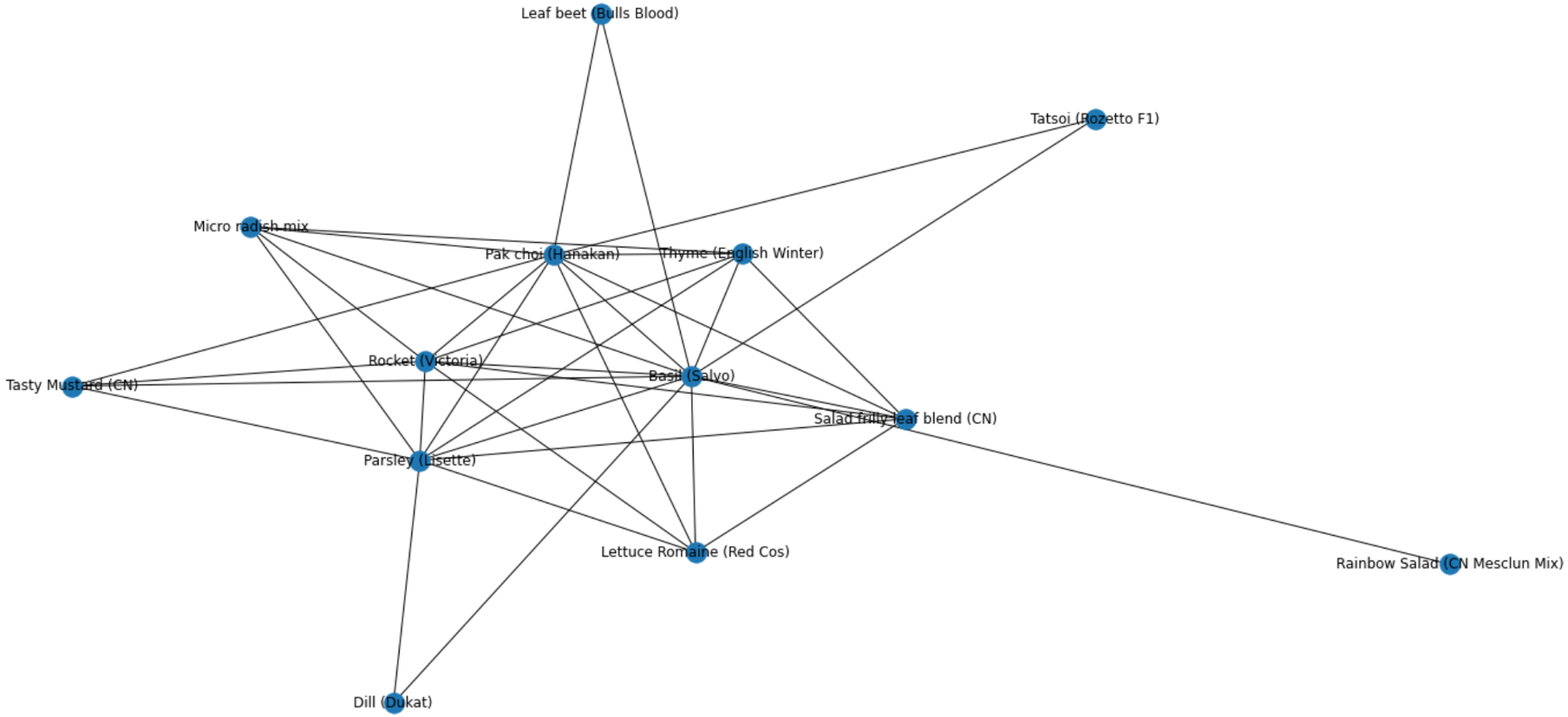
	plantcube	plant_id	plant_title	slot	planted_on	harvested_on	growth_days	owner	customer_name	customer_email	customer_creation_date	share_1	share_2	share_3	share_4	difference_in_days	planted_difference_in_days
67	00bc4f20-bd95-4b21-b154-64663084b66e	96.0	Micro radish mix	b4	2021-06-20 07:41:00	2021-06-28 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	617.0	564
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	617.0	557
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	617.0	557
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	617.0	557
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	617.0	553
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
58797	feba2f01-2a01-4479-be9d-36bdad2c367c	13.0	Kale (CN KAL 1028)	a8	2022-06-27 20:03:00	2022-09-17 14:38:00	81.77	eu-central-1:5ce93387-d5f0-443c-a0f9-7ac2a0714fb7	artur penkala	a.penkala@lok-loewen.de	2021-07-08 20:16:00	NaN	NaN	NaN	NaN	545.0	191
58798	feba2f01-2a01-4479-be9d-36bdad2c367c	19.0	Rocket (Victoria)	b7	2022-07-17 20:50:00	2022-09-29 18:08:00	73.89	eu-central-1:5ce93387-d5f0-443c-a0f9-7ac2a0714fb7	artur penkala	a.penkala@lok-loewen.de	2021-07-08 20:16:00	NaN	NaN	NaN	NaN	545.0	171
58799	feba2f01-2a01-4479-be9d-36bdad2c367c	100.0	Korean Mint	a9	2022-07-17 20:52:00	2022-09-17 14:37:00	61.74	eu-central-1:5ce93387-d5f0-443c-a0f9-7ac2a0714fb7	artur penkala	a.penkala@lok-loewen.de	2021-07-08 20:16:00	NaN	NaN	NaN	NaN	545.0	171
58800	feba2f01-2a01-4479-be9d-36bdad2c367c	64.0	Pak Choi Colour Crunch	a6	2022-07-17 20:52:00	2022-09-17 14:38:00	61.74	eu-central-1:5ce93387-d5f0-443c-a0f9-7ac2a0714fb7	artur penkala	a.penkala@lok-loewen.de	2021-07-08 20:16:00	NaN	NaN	NaN	NaN	545.0	171
58801	feba2f01-2a01-4479-be9d-36bdad2c367c	9.0	Pak choi (Hanakan)	a4	2022-07-17 20:53:00	2022-09-17 14:38:00	61.74	eu-central-1:5ce93387-d5f0-443c-a0f9-7ac2a0714fb7	artur penkala	a.penkala@lok-loewen.de	2021-07-08 20:16:00	NaN	NaN	NaN	NaN	545.0	171

27681 rows × 18 columns

Plants planted together by the customers

In [400]:
```
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 [401]:
```
import matplotlib.pyplot as plt

# Total customers - 276. Half the customers - 138
# create a subgraph with only the edges with a weight >= 138
subgraph = G.edge_subgraph([(u, v) for u, v, w in G.edges(data='weight') if w >= 138])

# 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()
```


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

In [402]:
```
# Plants planted together by half of the customers(138 or more customers)
index = 0
for u, v in subgraph.edges():
    print(f'{u} - {v}')
    index += 1
print('Total combinations',index)
```

```
Tasty Mustard (CN) - Rocket (Victoria)
Tasty Mustard (CN) - Pak choi (Hanakan)
Tasty Mustard (CN) - Basil (Salvo)
Tasty Mustard (CN) - Parsley (Lisette)
Tatsoi (Rozetto F1) - Pak choi (Hanakan)
Tatsoi (Rozetto F1) - Basil (Salvo)
Lettuce Romaine (Red Cos) - Salad frilly leaf blend (CN)
Lettuce Romaine (Red Cos) - Rocket (Victoria)
Lettuce Romaine (Red Cos) - Pak choi (Hanakan)
Lettuce Romaine (Red Cos) - Basil (Salvo)
Lettuce Romaine (Red Cos) - Parsley (Lisette)
Parsley (Lisette) - Micro radish mix
Parsley (Lisette) - Salad frilly leaf blend (CN)
Parsley (Lisette) - Rocket (Victoria)
Parsley (Lisette) - Pak choi (Hanakan)
Parsley (Lisette) - Basil (Salvo)
Parsley (Lisette) - Thyme (English Winter)
Parsley (Lisette) - Dill (Dukat)
Thyme (English Winter) - Micro radish mix
Thyme (English Winter) - Salad frilly leaf blend (CN)
Thyme (English Winter) - Rocket (Victoria)
Thyme (English Winter) - Pak choi (Hanakan)
Thyme (English Winter) - Basil (Salvo)
Pak choi (Hanakan) - Micro radish mix
Pak choi (Hanakan) - Salad frilly leaf blend (CN)
Pak choi (Hanakan) - Rocket (Victoria)
Pak choi (Hanakan) - Basil (Salvo)
Pak choi (Hanakan) - Leaf beet (Bulls Blood)
Micro radish mix - Rocket (Victoria)
Micro radish mix - Basil (Salvo)
Rocket (Victoria) - Salad frilly leaf blend (CN)
Rocket (Victoria) - Basil (Salvo)
Leaf beet (Bulls Blood) - Basil (Salvo)
Salad frilly leaf blend (CN) - Basil (Salvo)
Dill (Dukat) - Basil (Salvo)
Rainbow Salad (CN Mesclun Mix) - Basil (Salvo)
Total combinations 36
```

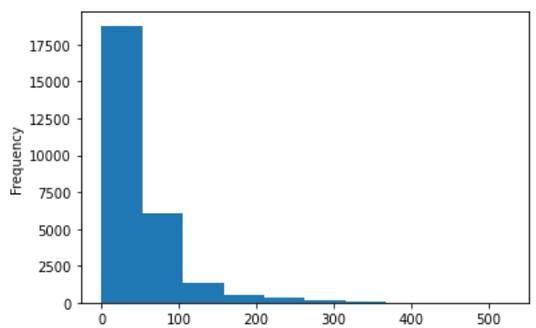
In [403]:
```
# Analyze customer behavior: how long the customers keep their plants before harvesting them, or how often they plant new plants.
df5 = df3.copy()

df5['duration'] = (df5['harvested_on'] - df5['planted_on']).dt.days

df5['duration'].describe()
```

Out[403]:
```
count    27341.000000
mean        52.371128
std         48.707570
min          0.000000
25%         27.000000
50%         37.000000
75%         61.000000
max        525.000000
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 [404]:
```
# distribution of the durations.
df5['duration'].plot.hist()
```

Out[404]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [405]:
```
df5.groupby('customer_name')['duration'].mean()
```

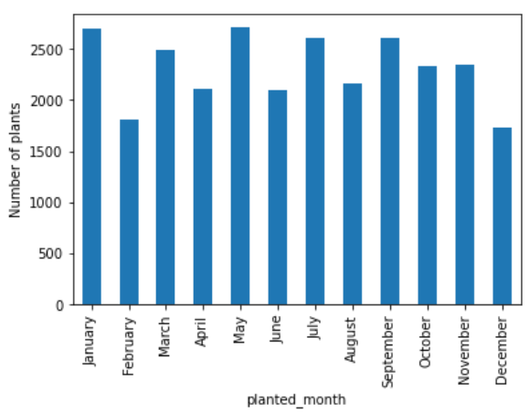
Out[405]:
```
customer_name
AL Küchen                       86.337079
Aalber van Aggelen              50.137931
Adrian Winkler                  39.550388
Al Kuechen                      65.500000
Alexander Groba                 60.333333
                                  ...
sven klingelhoefer - demand     36.337900
tim kriese                     120.363636
tischlerei Schöpker             60.422535
titi titu                       69.614458
valerius Kachniaschwili         45.942029
Name: duration, Length: 276, dtype: float64
```

In [406]:
```
# 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()

plant_counts_by_month
```

Out[406]:
```
planted_month
1     2693
2     1812
3     2489
4     2105
5     2711
6     2095
7     2604
8     2157
9     2611
10    2332
11    2344
12    1728
Name: plant_id, dtype: int64
```

In [407]:
```
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)
```

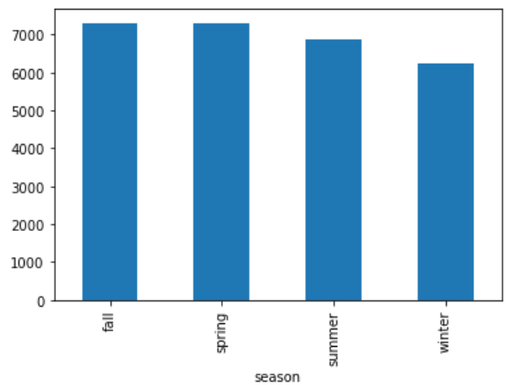
Out[407]:
```
([<matplotlib.axis.XTick at 0x15f8a636970>,
  <matplotlib.axis.XTick at 0x15f8a636940>,
  <matplotlib.axis.XTick at 0x15f91997c10>,
  <matplotlib.axis.XTick at 0x15f8b3d4bb0>,
  <matplotlib.axis.XTick at 0x15f8b3de340>,
  <matplotlib.axis.XTick at 0x15f91997e20>,
  <matplotlib.axis.XTick at 0x15f8b3dec40>,
  <matplotlib.axis.XTick at 0x15f8b3e43d0>,
  <matplotlib.axis.XTick at 0x15f8a63dfd0>,
  <matplotlib.axis.XTick at 0x15f90496310>,
  <matplotlib.axis.XTick at 0x15f90496a60>,
  <matplotlib.axis.XTick at 0x15f90496820>],
 [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')])
```



From this graph, we can infer higher counts in the spring and fall and lower counts in the winter and summer.This may be due to the change in the weather conditions

In [408]:
```
# 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[408]:
```
<AxesSubplot:xlabel='season'>
```

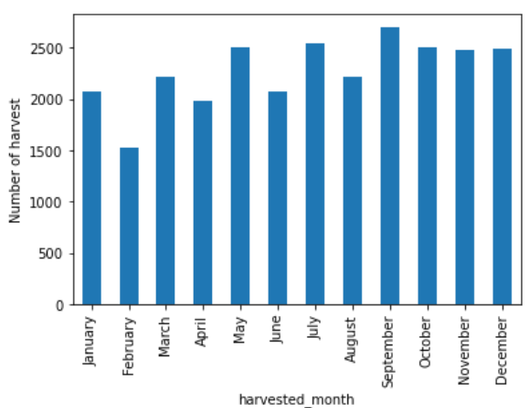


In [409]:
```
# Identifying plant popularity over time
# harvested on
df6['harvested_month'] = df6['harvested_on'].dt.month

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

harvested_counts_by_month
```

Out[409]:
```
harvested_month
1.0     2078
2.0     1525
3.0     2224
4.0     1989
5.0     2508
6.0     2079
7.0     2545
8.0     2216
9.0     2700
10.0    2507
11.0    2478
12.0    2492
Name: plant_id, dtype: int64
```

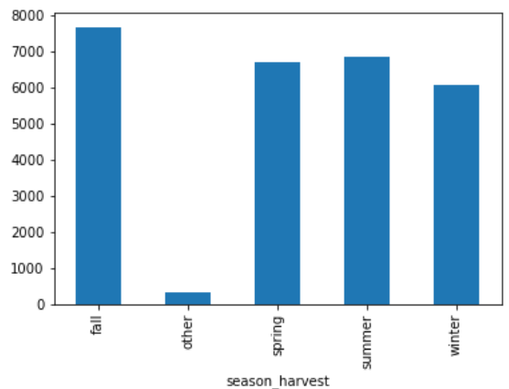
In [410]:
```
month_names = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']
harvested_counts_by_month.plot.bar()
plt.ylabel('Number of harvest')
plt.xticks(range(len(month_names)), month_names)
```

Out[410]:
```
([<matplotlib.axis.XTick at 0x15f8c7c4580>,
  <matplotlib.axis.XTick at 0x15f8c7c4550>,
  <matplotlib.axis.XTick at 0x15f85b20790>,
  <matplotlib.axis.XTick at 0x15f8b00f790>,
  <matplotlib.axis.XTick at 0x15f8b02d070>,
  <matplotlib.axis.XTick at 0x15f8b02d670>,
  <matplotlib.axis.XTick at 0x15f8b02ddc0>,
  <matplotlib.axis.XTick at 0x15f8b02de20>,
  <matplotlib.axis.XTick at 0x15f8c7d11f0>,
  <matplotlib.axis.XTick at 0x15f8750d7f0>,
  <matplotlib.axis.XTick at 0x15f874eb0a0>,
  <matplotlib.axis.XTick at 0x15f874eb6d0>],
 [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 [411]:
```
# 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'

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

harvest_counts_by_season.plot.bar()
```

Out[411]:
```
<AxesSubplot:xlabel='season_harvest'>
```



In [412]:
```
harvest_counts_by_season
```

Out[412]:
```
season_harvest
fall      7685
other      340
spring    6721
summer    6840
winter    6095
Name: plant_title, dtype: int64
```

we have higher harvest in fall and summer

In [ ]:
```

```