In [198]:

```
#imports
import pandas as pd
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
```

In [199]:

```
data = pd.read_csv('condensed_data_by_plants_02_01_2023.csv')
```

In [200]:

```
data.head()
```

Out[200]:

	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 [201]:

```
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 [202]:

```
df = data.copy()
```

In [203]:
```
actual_df_count = df.plantcube.unique().size
actual_df_count
```

Out[203]:
```
962
```

Long term customers

In [204]:
```
df['customer_creation_date'] = pd.to_datetime(df['customer_creation_date'], format='%d.%m.%Y %H:%M')
```

In [205]:
```
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 [206]:
```
df['difference_in_days'] = df['customer_creation_date'].apply(lambda x: (datetime.now() - x).days)
```

In [207]:
```
df.head()
```

Out[207]:

	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	131.0
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	131.0
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	131.0
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	131.0
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	131.0

In [208]:

```
# 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 [209]:

```
long_term_customers
```

Out[209]:

	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	791.0
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	791.0
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	791.0
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	791.0
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	791.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	1125.0
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	1125.0
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	1125.0
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	1125.0
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	1125.0

45877 rows × 16 columns

In [210]:

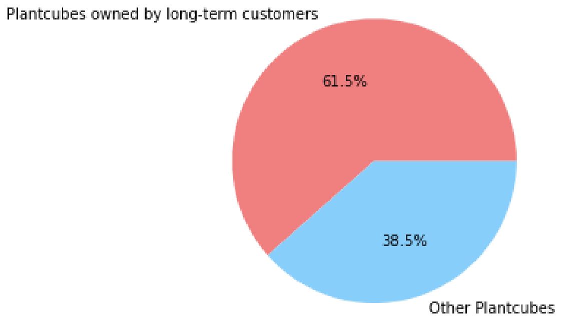
```
long_term_customers_count = long_term_customers.plantcube.unique().size
long_term_customers_count
```

Out[210]:

```
592
```

In [211]:

```
#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 [212]:

```
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 [213]:

```
active_plant_cubes_df1.head()
```

Out[213]:

	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	569
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	569
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	569
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	569
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	569

In [214]:

```
active_plant_cubes_df1.plantcube.unique().size
```

Out[214]:

```
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 [215]:

```
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 [216]:

```
active_plant_cubes_df2.head()
```

Out[216]:

	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	616.0	563	
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	616.0	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	616.0	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	616.0	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	616.0	553	

In [217]:

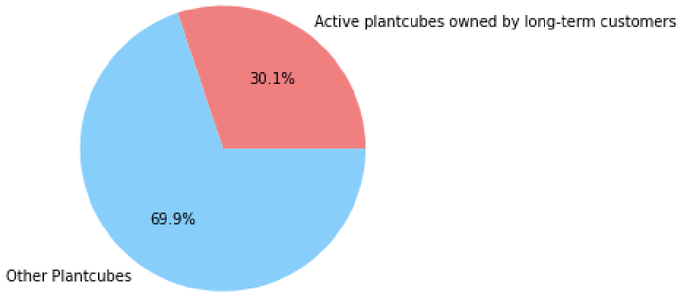
```
active_plant_cubes_df2.plantcube.unique().size
```

Out[217]:

```
290
```

In [218]:

```
#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 [219]:

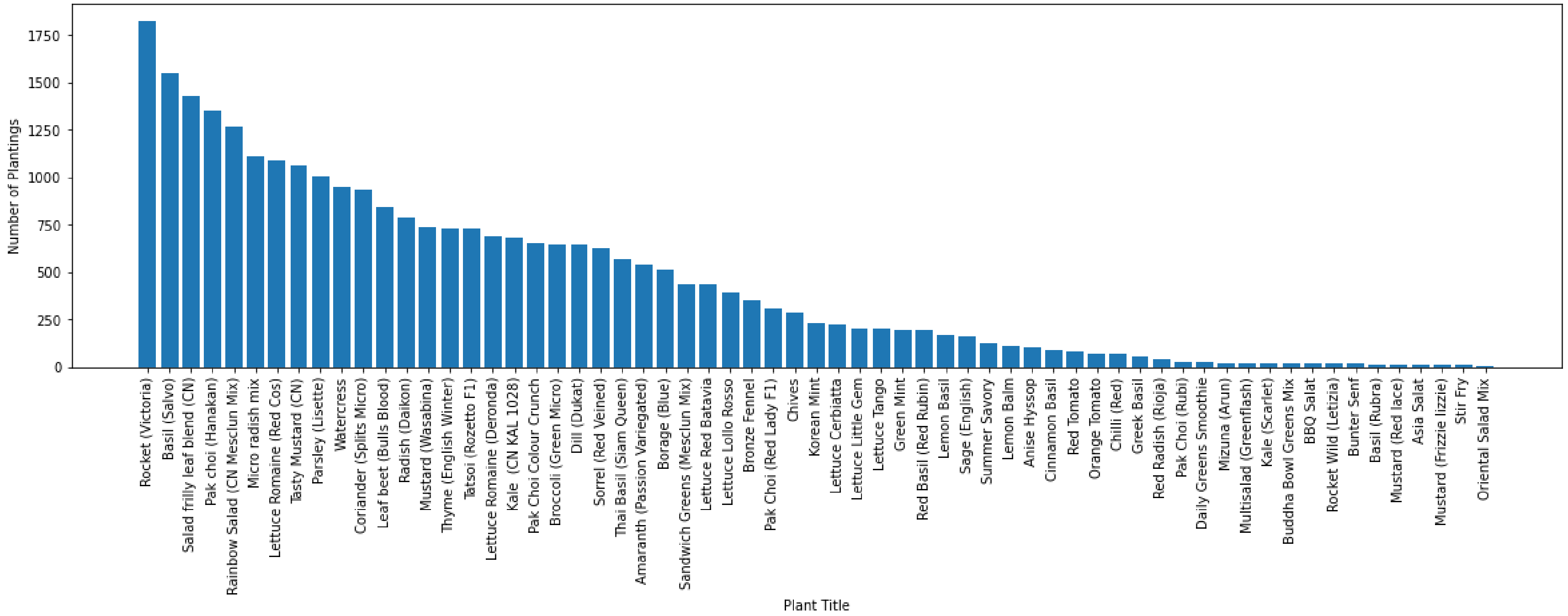
```
df3 = active_plant_cubes_df2.copy()

# 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[219]:

```
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 [220]:

```
# 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()
```

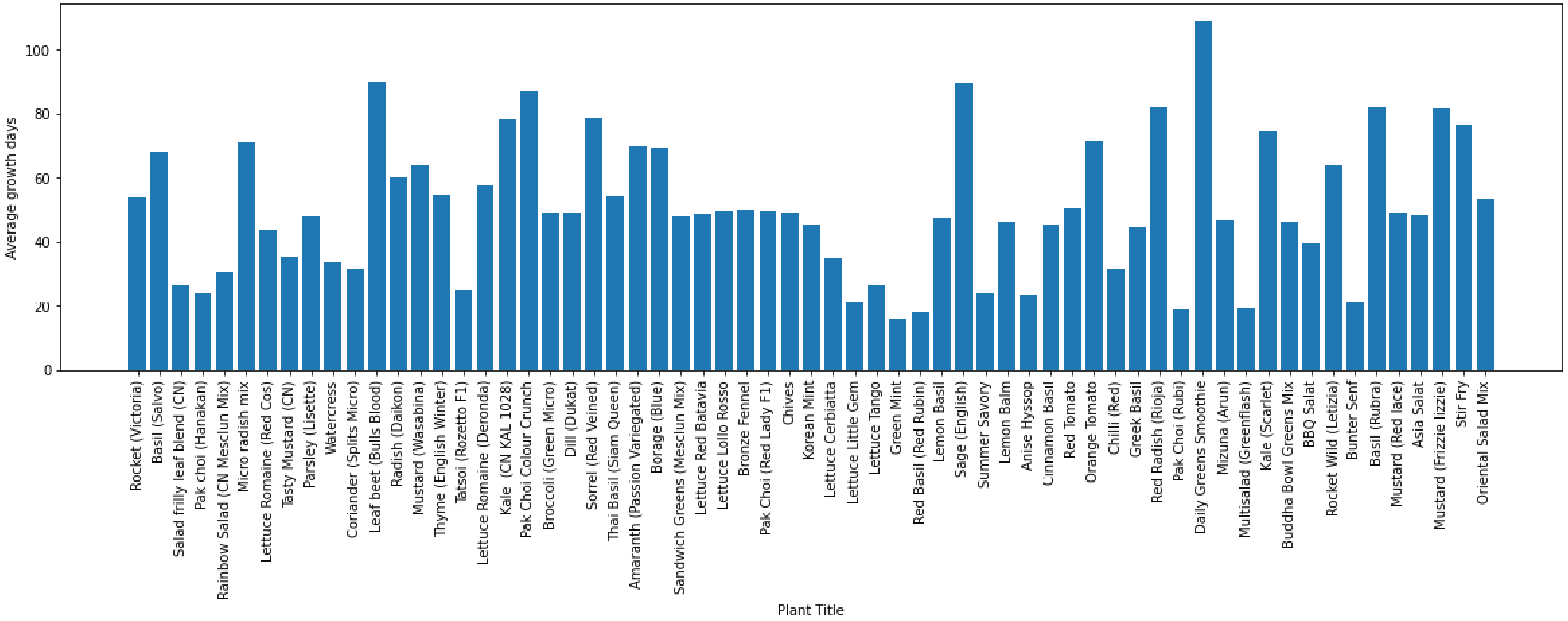


In [221]:

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

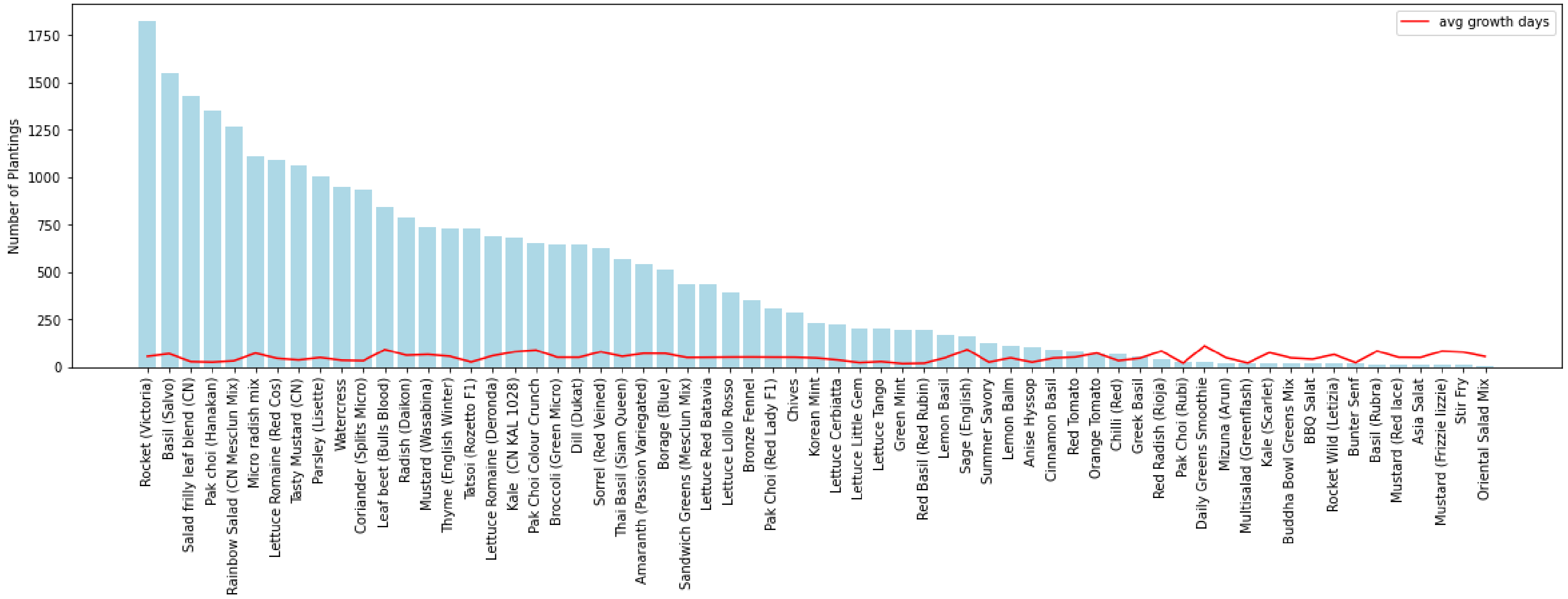
In [222]:

```
# 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 [234]:

```
# 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 [235]:

```
#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: 275
```

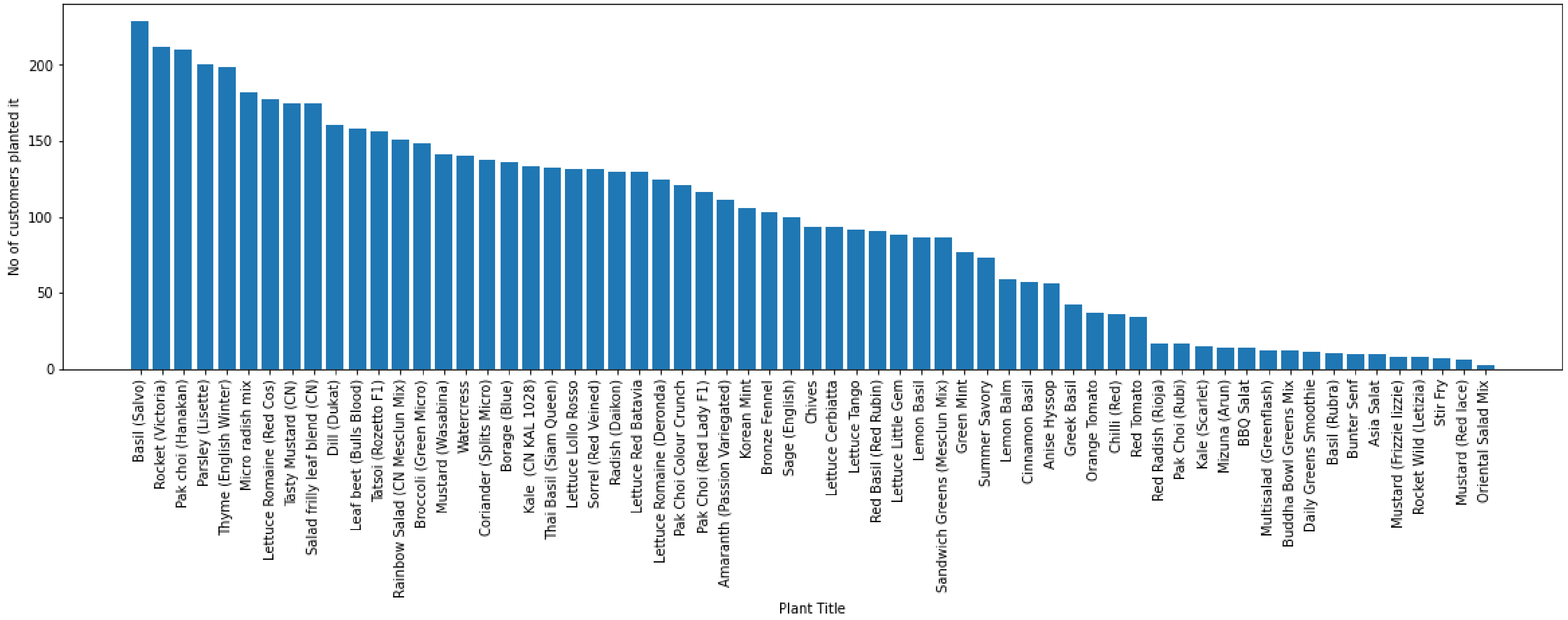
In [240]:

```
# 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[240]:

```
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 [241]:
```
# 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 [251]:

```
# Group the data by customer name and calculate the mean planting frequency for each group
growth_days_by_customer = df3.groupby('customer_name')['growth_days'].mean()

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

growth_days_by_customer
```

Out[251]:

```
customer_name
Thomas Richter             306.451818
Bert Fraeye                272.411667
Klara Behrendt             269.800000
Manuela Sickl              245.053611
Sascha Mattick             231.405556
                              ...    
Christopher  Harker         23.137778
Tim Kallas                  14.110000
Steve Sastalla              13.124118
KicheConcept Bertrange       3.290000
Firma XXXLutz Heilbronn           NaN
Name: growth_days, Length: 275, dtype: float64
```

In [ ]:

```

```