```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
file_path_1 = "/content/drive/MyDrive/Projects/Quantium DA projects/QVI_purchase_behaviour.csv"
purchase_behaviour = pd.read_csv(file_path_1)
purchase_behaviour.head()
```

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream
4	1005	MIDAGE SINGLES/COUPLES	Mainstream

Next steps: Generate code with purchase_behaviour View recommended plots New interactive sheet

```
file_path_2="/content/drive/MyDrive/Projects/Quantium DA projects/QVI_transaction_data.csv"
transaction_data = pd.read_csv(file_path_2)
transaction_data.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8

```
merged_data = pd.merge(transaction_data, purchase_behaviour, on="LYLTY_CARD_NBR", how="inner")
merged_data.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUSTOMER
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	YOUNG SINGLES/COUPLES	Premium
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	MIDAGE SINGLES/COUPLES	Budget
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	MIDAGE SINGLES/COUPLES	Budget
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0	MIDAGE SINGLES/COUPLES	Budget
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8	MIDAGE SINGLES/COUPLES	Budget

```
merged_data.shape
```

```
(264836, 10)
```

```
merged_data.dropna()
merged_data.shape
```

```
(264836, 10)
```

```
merged_data["LYLTY_CARD_NBR"].unique().shape
```

```
(72637,)
```

```
merged_data["PROD_NAME"].unique()
```

```
       'Thins Chips Seasonedchicken 175g',
       'Smiths Crinkle Cut  Salt & Vinegar 170g',
       'Infuzions BBQ Rib   Prawn Crackers 110g',
       'GrnWves Plus Btroot & Chilli Jam 180g',
       'Tyrrells Crisps     Lightly Salted 165g',
       'Kettle Sweet Chilli And Sour Cream 175g',
       'Doritos Salsa       Medium 300g', 'Kettle 135g Swt Pot Sea Salt',
       'Pringles SourCream  Onion 134g',
       'Cheetos Chs & Bacon Balls 190g', 'Pringles Slt Vingar 134g',
       'Infuzions SourCream&Herbs Veg Strws 110g',
       'Kettle Tortilla ChpsFeta&Garlic 150g',
       'Infuzions Mango     Chutny Papadums 70g',
       'RRD Steak &         Chimuchurri 150g',
       'RRD Honey Soy       Chicken 165g',
       'Sunbites Whlegrn    Crisps Frch/Onin 90g',
       'RRD Salt & Vinegar  165g', 'Doritos Cheese      Supreme 330g',
       'Smiths Crinkle Cut  Snag&Sauce 150g',
       'WW Sour Cream &OnionStacked Chips 160g',
       'RRD Lime & Pepper   165g',
       'Natural ChipCo Sea  Salt & Vinegr 175g',
       'Red Rock Deli Chikn&Garlic Aioli 150g',
```

```
df=merged_data.copy()
df.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUSTOMER
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	YOUNG SINGLES/COUPLES	Premium
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	MIDAGE SINGLES/COUPLES	Budget
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	MIDAGE SINGLES/COUPLES	Budget
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0	MIDAGE SINGLES/COUPLES	Budget
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8	MIDAGE SINGLES/COUPLES	Budget

```
df["PROD_NAME"].unique()
```

```
       'Thins Chips Seasonedchicken 175g',
       'Smiths Crinkle Cut  Salt & Vinegar 170g',
       'Infuzions BBQ Rib   Prawn Crackers 110g',
       'GrnWves Plus Btroot & Chilli Jam 180g',
       'Tyrrells Crisps     Lightly Salted 165g',
       'Kettle Sweet Chilli And Sour Cream 175g',
       'Doritos Salsa       Medium 300g', 'Kettle 135g Swt Pot Sea Salt',
       'Pringles SourCream  Onion 134g',
       'Doritos Corn Chips  Original 170g',
       'Twisties Cheese     Burger 250g',
       'Old El Paso Salsa   Dip Chnky Tom Ht300g',
       'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
       'Woolworths Mild     Salsa 300g',
       'Natural Chip Co     Tmato Hrb&Spce 175g',
       'Smiths Crinkle Cut  Chips Original 170g',
       'Cobs Popd Sea Salt  Chips 110g',
       'Smiths Crinkle Cut  Chips Chs&Onion170g',
       'French Fries Potato Chips 175g',
       'Old El Paso Salsa   Dip Tomato Med 300g',
       'Doritos Corn Chips  Cheese Supreme 170g',
       'Pringles Original   Crisps 134g',
       'RRD Chilli&         Coconut 150g',
       'WW Original Corn    Chips 200g',
       'Thins Potato Chips  Hot & Spicy 175g',
       'Cobs Popd Sour Crm  &Chives Chips 110g',
       'Smiths Crnkle Chip  Orgnl Big Bag 380g',
       'Doritos Corn Chips  Nacho Cheese 170g',
       'Kettle Sensations   BBQ&Maple 150g',
       'WW D/Style Chip     Sea Salt 200g',
       'Pringles Chicken    Salt Crips 134g',
       'WW Original Stacked Chips 160g',
       'Smiths Chip Thinly  CutSalt/Vinegr175g', 'Cheezels Cheese 330g',
       'Tostitos Lightly    Salted 175g',
       'Thins Chips Salt &  Vinegar 175g',
       'Smiths Crinkle Cut  Chips Barbecue 170g', 'Cheetos Puffs 165g',
       'RRD Sweet Chilli &  Sour Cream 165g',
       'WW Crinkle Cut      Original 175g',
       'Tostitos Splash Of  Lime 175g', 'Woolworths Medium   Salsa 300g',
       'Kettle Tortilla ChpsBtroot&Ricotta 150g',
       'CCs Tasty Cheese    175g', 'Woolworths Cheese   Rings 190g',
       'Tostitos Smoked     Chipotle 175g', 'Pringles Barbeque   134g',
       'WW Supreme Cheese   Corn Chips 200g',
       'Pringles Mystery    Flavour 134g',
       'Tyrrells Crisps     Ched & Chives 165g',
       'Snbts Whlgrn Crisps Cheddr&Mstrd 90g',
       'Cheetos Chs & Bacon Balls 190g', 'Pringles Slt Vingar 134g',
       'Infuzions SourCream&Herbs Veg Strws 110g',
       'Kettle Tortilla ChpsFeta&Garlic 150g',
       'Infuzions Mango     Chutny Papadums 70g',
       'RRD Steak &         Chimuchurri 150g',
       'RRD Honey Soy       Chicken 165g',
       'Sunbites Whlegrn    Crisps Frch/Onin 90g',
       'RRD Salt & Vinegar  165g', 'Doritos Cheese      Supreme 330g',
       'Smiths Crinkle Cut  Snag&Sauce 150g',
       'WW Sour Cream &OnionStacked Chips 160g',
       'RRD Lime & Pepper   165g',
       'Natural ChipCo Sea  Salt & Vinegr 175g',
       'Red Rock Deli Chikn&Garlic Aioli 150g',
       'RRD SR Slow Rst     Pork Belly 150g', 'RRD Pc Sea Salt     165g',
```

```
brand_dict = {
    "Natural": "Natural Chip Company",
    "CCs": "CCs",
    "Smiths": "Smiths",
    "Kettle": "Kettle",
    "Old": "Old El Paso",
    "Grain": "Grain Waves",
    "Doritos": "Doritos",
    "Twisties": "Twisties",
    "WW": "Woolworths",
    "Pringles": "Pringles",
    "Red": "Red Rock Deli",
    "Cheezels": "Cheezels",
    "Tyrrells": "Tyrrells",
    "Cheetos": "Cheetos",
    "Tostitos": "Tostitos",
    "Infuzions": "Infuzions",
    "Sunbites": "Sunbites",
    "Burger": "Burger Rings",

    "Infzns": "Infuzions",
    "NCC": "Natural Chip Company",
    "RRD": "Red Rock Deli",
    "Snbts": "Sunbites",
    "GrnWves": "Grain Waves",
    "Dorito": "Doritos",
    "Smith": "Smiths"



}
```

```
def get_company_name(prod_name):
    first_word = prod_name.split()[0]
    return brand_dict.get(first_word, first_word)


df["COMPANY_NAME"] = df["PROD_NAME"].apply(get_company_name)


df.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUSTOMER	COMPANY_NAME
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	YOUNG SINGLES/COUPLES	Premium	Natural Chip Company
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	MIDAGE SINGLES/COUPLES	Budget	CCs
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	MIDAGE SINGLES/COUPLES	Budget	Smiths
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0	MIDAGE SINGLES/COUPLES	Budget	Smiths
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8	MIDAGE SINGLES/COUPLES	Budget	Kettle

```
df['COMPANY_NAME'].unique().shape
```

```
(21,)
```

```
new_df = df[df["COMPANY_NAME"] != "French"]
```

```
new_df.shape
```

```
(263418, 11)
```

```
new_df["COMPANY_NAME"].unique().shape
```

```
(20,)
```

```
def percentage(a,b):
  return (a/b)*100
```

```
new_df['COMPANY_NAME'].value_counts().head(5)
```

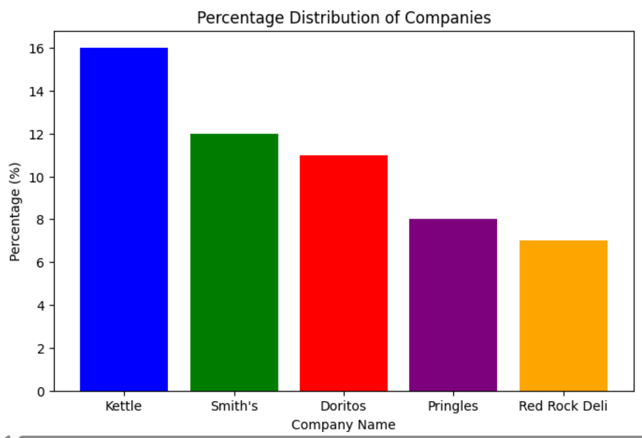
COMPANY_NAME	count
Kettle	41288
Smiths	31823
Doritos	28147
Pringles	25102
Red Rock Deli	17779

dtype: int64

```
a= percentage(new_df[new_df['COMPANY_NAME']=="Kettle"].shape[0],new_df.shape[0])
b= percentage(new_df[new_df['COMPANY_NAME']=="Smith's"].shape[0],new_df.shape[0])
c= percentage(new_df[new_df['COMPANY_NAME']=="Doritos"].shape[0],new_df.shape[0])
d= percentage(new_df[new_df['COMPANY_NAME']=="Pringles"].shape[0],new_df.shape[0])
e= percentage(new_df[new_df['COMPANY_NAME']=="Red Rock Deli"].shape[0],new_df.shape[0])
a,b,c,d,e
```

```
(15.673947869925367,
 0.0,
 10.685298650813536,
 9.529341199158752,
 6.749348943504241)
```

```
companies = ["Kettle", "Smith's", "Doritos", "Pringles", "Red Rock Deli"]
percentages = [16, 12, 11, 8, 7]

plt.figure(figsize=(8, 5))
plt.bar(companies, percentages, color=['blue', 'green', 'red', 'purple', 'orange'])


plt.xlabel("Company Name")
plt.ylabel("Percentage (%)")
plt.title("Percentage Distribution of Companies")

plt.show()
```



```
new_df['LIFESTAGE'].value_counts().head(5)
```

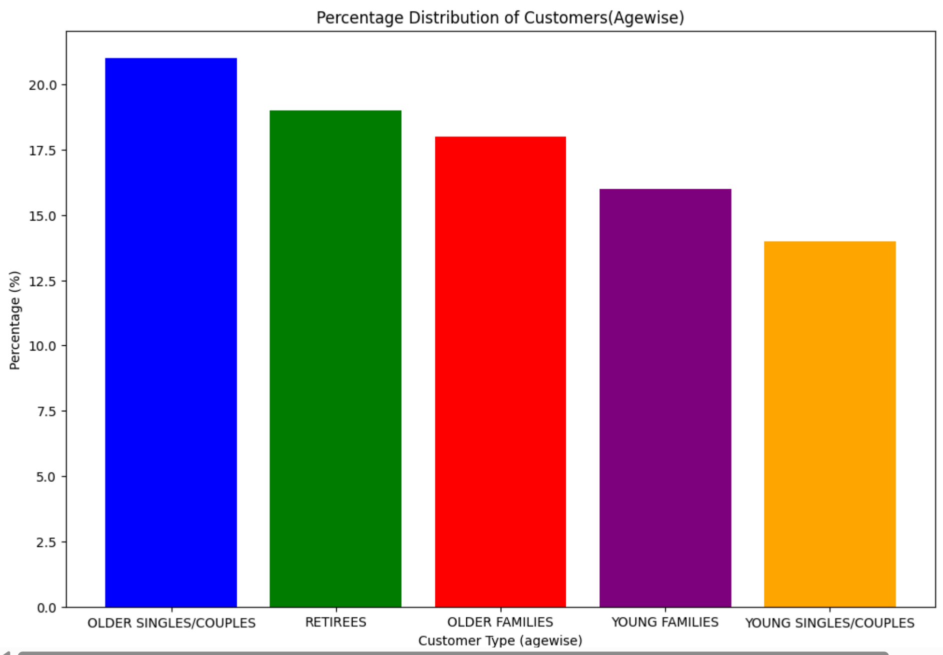
LIFESTAGE	count
OLDER SINGLES/COUPLES	54193
RETIREES	49535
OLDER FAMILIES	48313
YOUNG FAMILIES	43314
YOUNG SINGLES/COUPLES	36183

dtype: int64

```
a= percentage(new_df[new_df['LIFESTAGE']=="OLDER SINGLES/COUPLES"].shape[0],new_df.shape[0])
b= percentage(new_df[new_df['LIFESTAGE']=="RETIREES"].shape[0],new_df.shape[0])
c= percentage(new_df[new_df['LIFESTAGE']=="OLDER FAMILIES"].shape[0],new_df.shape[0])
d= percentage(new_df[new_df['LIFESTAGE']=="YOUNG FAMILIES"].shape[0],new_df.shape[0])
e= percentage(new_df[new_df['LIFESTAGE']=="YOUNG SINGLES/COUPLES"].shape[0],new_df.shape[0])
a,b,c,d,e
```

```
(20.573005641224213,
 18.804713421254434,
 18.340811941477046,
 16.44306767191308,
 13.735963373801335)
```

```
Customer_Type_agewise = ["OLDER SINGLES/COUPLES", "RETIREES", "OLDER FAMILIES", "YOUNG FAMILIES", "YOUNG SINGLES/COUPLES"]
percentages = [21, 19, 18, 16, 14]

plt.figure(figsize=(12, 8))
plt.bar(Customer_Type_agewise, percentages, color=['blue', 'green', 'red', 'purple', 'orange'])


plt.xlabel("Customer Type (agewise)")
plt.ylabel("Percentage (%)")
plt.title("Percentage Distribution of Customers(Agewise)")

plt.show()
```



```
new_df['PREMIUM_CUSTOMER'].value_counts().head(5)
```

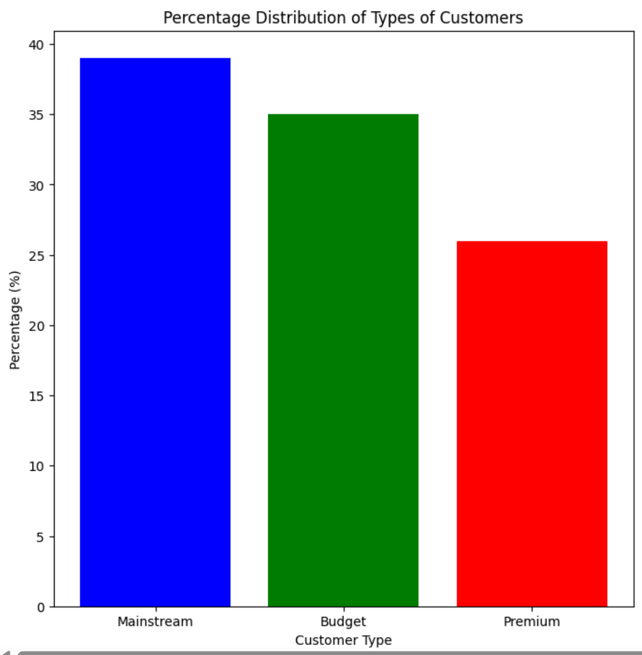
PREMIUM_CUSTOMER	count
Mainstream	101481
Budget	92618
Premium	69319

dtype: int64

```
a= percentage(new_df[new_df['PREMIUM_CUSTOMER']=="Mainstream"].shape[0],new_df.shape[0])
b= percentage(new_df[new_df['PREMIUM_CUSTOMER']=="Budget"].shape[0],new_df.shape[0])
c= percentage(new_df[new_df['PREMIUM_CUSTOMER']=="Premium"].shape[0],new_df.shape[0])
a,b,c
```

```
(38.524702184360976, 35.160087769248875, 26.31521004639015)
```

```
Customer_Type = ["Mainstream", "Budget", "Premium"]
percentages = [39, 35, 26]

plt.figure(figsize=(8, 8))
plt.bar(Customer_Type, percentages, color=['blue', 'green', 'red'])


plt.xlabel("Customer Type")
plt.ylabel("Percentage (%)")
plt.title("Percentage Distribution of Types of Customers")

plt.show()
```



```
new_df
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUSTOMER	COMPANY_NAME
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	YOUNG SINGLES/COUPLES	Premium	Natural Chip Company
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	MIDAGE SINGLES/COUPLES	Budget	CCs
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	MIDAGE SINGLES/COUPLES	Budget	Smiths
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0	MIDAGE SINGLES/COUPLES	Budget	Smiths
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8	MIDAGE SINGLES/COUPLES	Budget	Kettle
...	...	...	...	...	...	...	...	...	...	...	...
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2	10.8	YOUNG SINGLES/COUPLES	Premium	Kettle
264832	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	1	4.4	YOUNG SINGLES/COUPLES	Premium	Tostitos
264833	43410	272	272379	270187	51	Doritos Mexicana 170g	2	8.8	YOUNG SINGLES/COUPLES	Premium	Doritos
264834	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2	7.8	YOUNG SINGLES/COUPLES	Premium	Doritos
264835	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	2	8.8	YOUNG SINGLES/COUPLES	Premium	Tostitos

263418 rows × 11 columns

```
new_df['DATE'] = pd.to_datetime(new_df['DATE'])
new_df['MONTH'] = new_df['DATE'].dt.to_period('M')
new_df.head()
```

```
<ipython-input-253-b347b00396ee>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['DATE'] = pd.to_datetime(new_df['DATE'])
<ipython-input-253-b347b00396ee>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['MONTH'] = new_df['DATE'].dt.to_period('M')
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUSTOMER	COMPANY_NAME	MONTH
0	1970-01-01 00:00:00.000043390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	YOUNG SINGLES/COUPLES	Premium	Natural Chip Company	1970-01
1	1970-01-01 00:00:00.000043599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	MIDAGE SINGLES/COUPLES	Budget	CCs	1970-01
2	1970-01-01 00:00:00.000043605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	MIDAGE SINGLES/COUPLES	Budget	Smiths	1970-01

```
file_path_3 = "/content/drive/MyDrive/Projects/Quantium DA projects/QVI_data.csv"
df_2 = pd.read_csv(file_path_3)
df_2.head()
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	Premium
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	Mainstream
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	Budget
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	3.0	175	NATURAL	YOUNG FAMILIES	Budget
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	1.9	160	WOOLWORTHS	OLDER SINGLES/COUPLES	Mainstream

```
df_2.shape
```

```
(264834, 12)
```

```
brand_dict = {
    "Natural": "Natural Chip Company",
    "CCs": "CCs",
    "Smiths": "Smiths",
    "Kettle": "Kettle",
    "Old": "Old El Paso",
    "Grain": "Grain Waves",
    "Doritos": "Doritos",
    "Twisties": "Twisties",
    "WW": "Woolworths",
    "Pringles": "Pringles",
    "Red": "Red Rock Deli",
    "Cheezels": "Cheezels",
    "Tyrrells": "Tyrrells",
    "Cheetos": "Cheetos",
    "Tostitos": "Tostitos",
    "Infuzions": "Infuzions",
    "Sunbites": "Sunbites",
    "Burger": "Burger Rings",

    "Infzns": "Infuzions",
    "NCC": "Natural Chip Company",
    "RRD": "Red Rock Deli",
    "Snbts": "Sunbites",
    "GrnWves": "Grain Waves",
    "Dorito": "Doritos",
    "Smith": "Smiths"



}


def get_company_name(PROD_NAME):
    first_word = PROD_NAME.split()[0]
    return brand_dict.get(first_word, first_word)


df_2["BRAND"] = df_2["PROD_NAME"].apply(get_company_name)
df_2.head()
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	Natural Chip Company	YOUNG SINGLES/COUPLES	Premium
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	Red Rock Deli	YOUNG SINGLES/COUPLES	Mainstream
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	Grain Waves	YOUNG FAMILIES	Budget

```
df_2["BRAND"].unique()
```

```
array(['Natural Chip Company', 'Red Rock Deli', 'Grain Waves',
       'Woolworths', 'Cheetos', 'Infuzions', 'Doritos', 'Old El Paso',
       'Smiths', 'Kettle', 'CCs', 'Tostitos', 'Cobs', 'Burger Rings',
       'Thins', 'Tyrrells', 'Cheezels', 'Twisties', 'Sunbites',
       'Pringles', 'French'], dtype=object)
```

```
df_2['BRAND'].unique().shape
```

```
(21,)
```

```
new_df_2 = df_2[df_2["BRAND"] != "French"]
new_df_2.shape
```

```
(263416, 12)
```

```
new_df_2['BRAND'].unique().shape
```

```
(20,)
```

```
new_df_2.head()
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	Natural Chip Company	YOUNG SINGLES/COUPLES	Premium
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	Red Rock Deli	YOUNG SINGLES/COUPLES	Mainstream
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	Grain Waves	YOUNG FAMILIES	Budget

```
# Convert the DATE column to datetime if it's not already
new_df_2['DATE'] = pd.to_datetime(new_df_2['DATE'])

# Filter data for the period July 2018 - June 2019
filtered_data_2 = new_df_2[(new_df_2['DATE'] >= '2018-07-01') & (new_df_2['DATE'] <= '2019-06-30')]
```

```
<ipython-input-262-72d7c2ac3890>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df_2['DATE'] = pd.to_datetime(new_df_2['DATE'])
```

```
# Split data into pre-trial and trial periods
pre_trial_data = filtered_data_2[(filtered_data_2['DATE'] >= '2018-07-01') & (filtered_data_2['DATE'] < '2019-02-01')]
trial_data = filtered_data_2[(filtered_data_2['DATE'] >= '2019-02-01') & (filtered_data_2['DATE'] <= '2019-04-30')]
```

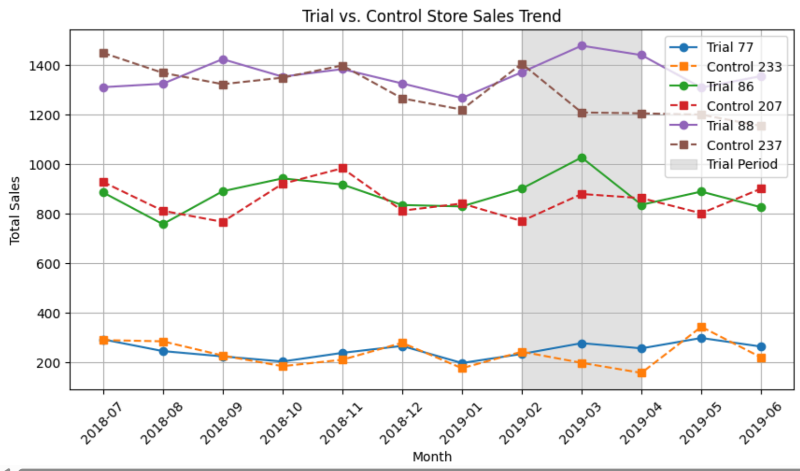
```
# Aggregate data for each store in the pre-trial period
pre_store_metrics = pre_trial_data.groupby("STORE_NBR").agg(
    total_sales=("TOT_SALES", "sum"),
    total_customers=("LYLTY_CARD_NBR", "nunique"),  # Unique customers per store
    avg_txn_per_customer=("TOT_SALES", "mean")  # Avg sales per transaction
).reset_index()

# Aggregate data for each store in the trial period
trial_store_metrics = trial_data.groupby("STORE_NBR").agg(
    total_sales=("TOT_SALES", "sum"),
    total_customers=("LYLTY_CARD_NBR", "nunique"),
    avg_txn_per_customer=("TOT_SALES", "mean")
).reset_index()

print(pre_store_metrics.head())  # Pre-trial data summary
print(trial_store_metrics.head())  # Trial data summary
```

```
   STORE_NBR  total_sales  total_customers  avg_txn_per_customer
0          1      1383.90              245              4.180967
1          2      1110.50              218              3.951957
2          3      7526.15              334              8.571925
3          4      9127.00              350              8.742337
4          5      5697.70              231              7.025524
   STORE_NBR  total_sales  total_customers  avg_txn_per_customer
0          1        611.2              125              4.157823
1          2        525.0              109              4.166667
2          3       3242.1              230              8.554354
3          4       3306.3              243              8.840374
4          5       2124.2              172              6.830225
```

```
from scipy.spatial.distance import euclidean
from scipy.stats import pearsonr

# List of trial stores
trial_stores = [77, 86, 88]

# Function to find the best control store
def find_control_store(trial_store):
    trial_values = pre_store_metrics.loc[pre_store_metrics['STORE_NBR'] == trial_store,
                                         ['total_sales', 'total_customers', 'avg_txn_per_customer']].values.flatten()

    best_store, best_dist, best_corr = None, float("inf"), -1

    for store in pre_store_metrics['STORE_NBR']:
        if store == trial_store:
            continue  # Skip the trial store itself

        control_values = pre_store_metrics.loc[pre_store_metrics['STORE_NBR'] == store,
                                               ['total_sales', 'total_customers', 'avg_txn_per_customer']].values.flatten()

        dist = euclidean(trial_values, control_values)  # Magnitude distance
        corr, _ = pearsonr(trial_values, control_values)  # Pearson correlation

        # Choose the store with the lowest distance
        if dist < best_dist:
            best_store, best_dist, best_corr = store, dist, corr

    return best_store, best_dist, best_corr

# Find control stores for each trial store
control_stores = {trial: find_control_store(trial) for trial in trial_stores}

# Print results
for trial, (control, dist, corr) in control_stores.items():
    print(f"Trial Store {trial} -> Control Store {control} (Distance: {dist:.2f}, Correlation: {corr:.2f})")
```

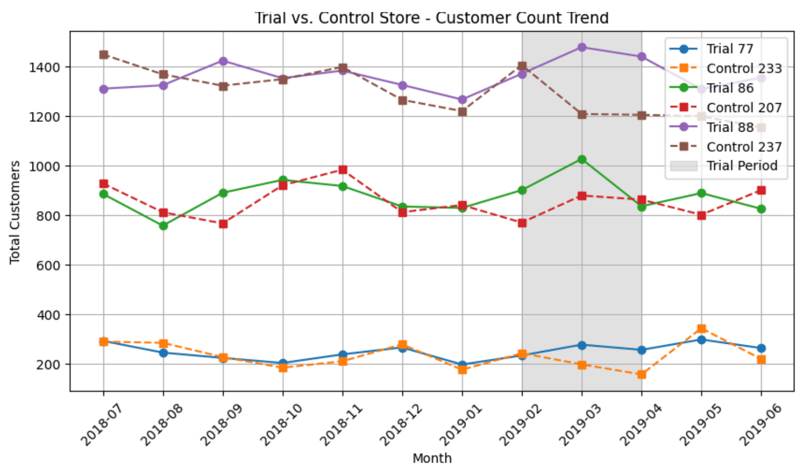
```
Trial Store 77 -> Control Store 233 (Distance: 16.00, Correlation: 1.00)
Trial Store 86 -> Control Store 207 (Distance: 5.62, Correlation: 1.00)
Trial Store 88 -> Control Store 237 (Distance: 19.55, Correlation: 1.00)
```

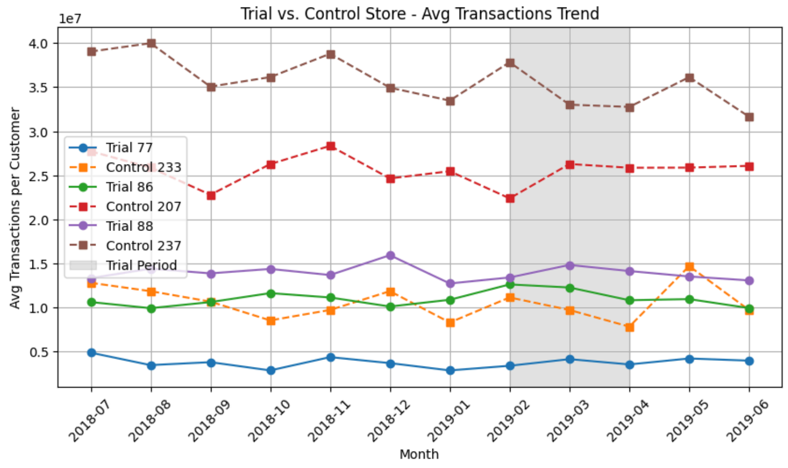
```
def get_monthly_sales(data, store):
    return data[data['STORE_NBR'] == store].groupby(data['DATE'].dt.to_period('M'))['TOT_SALES'].sum()

# Plot sales trends
plt.figure(figsize=(10, 5))

for trial_store, control_store in control_stores.items():
    trial_sales = get_monthly_sales(new_df_2, trial_store)
    control_sales = get_monthly_sales(new_df_2, control_store[0])  # control_store[0] is store number

    plt.plot(trial_sales.index.astype(str), trial_sales.values, label=f"Trial {trial_store}", linestyle='-', marker='o')
    plt.plot(control_sales.index.astype(str), control_sales.values, label=f"Control {control_store[0]}", linestyle='--', marker='s')

# Highlight trial period
plt.axvspan('2019-02', '2019-04', color='gray', alpha=0.2, label="Trial Period")

# Labels and title
plt.xlabel("Month")
plt.ylabel("Total Sales")
plt.title("Trial vs. Control Store Sales Trend")
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```



```
import matplotlib.pyplot as plt

# Function to get monthly metric data
def get_monthly_metric(data, store, metric):
    # Check if the metric column exists in the DataFrame
    if metric not in data.columns:
        raise KeyError(f"Column not found: {metric}. Available columns: {data.columns.tolist()}")
    return data[data['STORE_NBR'] == store].groupby(data['DATE'].dt.to_period('M'))[metric].sum()
```

```
# Function to plot trends
def plot_metric_trend(data, metric, ylabel, title):
    plt.figure(figsize=(10, 5))

    for trial_store, control_store in control_stores.items():
        trial_metric = get_monthly_metric(data, trial_store, metric)
        control_metric = get_monthly_metric(data, control_store[0], metric)

        plt.plot(trial_metric.index.astype(str), trial_metric.values, label=f"Trial {trial_store}", linestyle='-', marker='o')
        plt.plot(control_metric.index.astype(str), control_metric.values, label=f"Control {control_store[0]}", linestyle='--', marker='s')

    # Highlight trial period
    plt.axvspan('2019-02', '2019-04', color='gray', alpha=0.2, label="Trial Period")

    # Labels and title
    plt.xlabel("Month")
    plt.ylabel(ylabel)
    plt.title(title)
    plt.xticks(rotation=45)
    plt.legend()
    plt.grid(True)
    plt.show()

# Plot total customers trend
# Assuming 'TOT_SALES' represents total customers here
plot_metric_trend(new_df_2, 'TOT_SALES', "Total Customers", "Trial vs. Control Store - Customer Count Trend")

# Assuming 'TXN_ID' represents the number of transactions per customer here
plot_metric_trend(new_df_2, 'TXN_ID', "Avg Transactions per Customer", "Trial vs. Control Store - Avg Transactions Trend")
```





Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.