```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
import pandas as pd

df = pd.read_csv("QVI_purchase_behaviour.csv")   # use CSV file, not .py

# 1. Check if any nulls exist
print(df.isnull().values.any())

# 2. Count nulls in each column
print(df.isnull().sum())

# 3. Count total nulls in dataset
print(df.isnull().sum().sum())

# 4. Show only rows with null values
print(df[df.isnull().any(axis=1)])
```

```
False
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
0
Empty DataFrame
Columns: [LYLTY_CARD_NBR, LIFESTAGE, PREMIUM_CUSTOMER]
Index: []
```

```
# 1. Count total duplicate rows
print(df.duplicated().sum())

# 2. Show duplicate rows (if any)
print(df[df.duplicated()])

# 3. Drop duplicates (if needed)
df_clean = df.drop_duplicates()
```

```
0
Empty DataFrame
Columns: [DATE, STORE_NBR, LYLTY_CARD_NBR, TXN_ID, PROD_NBR, PROD_NAME, PROD_QTY, TOT_SALES]
Index: []
```

```
print(df.dtypes)  # customer Behaviour # ???
```

```
DATE              object
STORE_NBR          int64
```

```
LYLTY_CARD_NBR      int64
TXN_ID              int64
PROD_NBR            int64
PROD_NAME          object
PROD_QTY            int64
TOT_SALES         float64
Zscore            float64
dtype: object
```

```
import pandas as pd

df = pd.read_csv("QVI_transaction_data.csv")   # use CSV file, not .py

# 1. Check if any nulls exist
print(df.isnull().values.any())

# 2. Count nulls in each column
print(df.isnull().sum())

# 3. Count total nulls in dataset
print(df.isnull().sum().sum())

# 4. Show only rows with null values
print(df[df.isnull().any(axis=1)])
```

```
False
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
dtype: int64
0
Empty DataFrame
Columns: [DATE, STORE_NBR, LYLTY_CARD_NBR, TXN_ID, PROD_NBR, PROD_NAME, PROD_QTY, TOT_SALES]
Index: []
```

```
# 1. Count total duplicate rows
print(df.duplicated().sum())

# 2. Show duplicate rows (if any)
print(df[df.duplicated()])

# 3. Drop duplicates (if needed)
df_clean = df.drop_duplicates()
```

```
1
              DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
124845  10/1/2018        107          107024  108462        45

                                     PROD_NAME  PROD_QTY  TOT_SALES
124845  Smiths Thinly Cut   Roast Chicken 175g         2        6.0
```

```
df = df.drop_duplicates(keep='first')
```

```
# convert QTY column to numeric
df["TOT_SALES"] = pd.to_numeric(df["TOT_SALES"], errors="coerce")
```

```
# convert QTY column to numeric
df["PROD_QTY"] = pd.to_numeric(df["PROD_QTY"], errors="coerce")
```

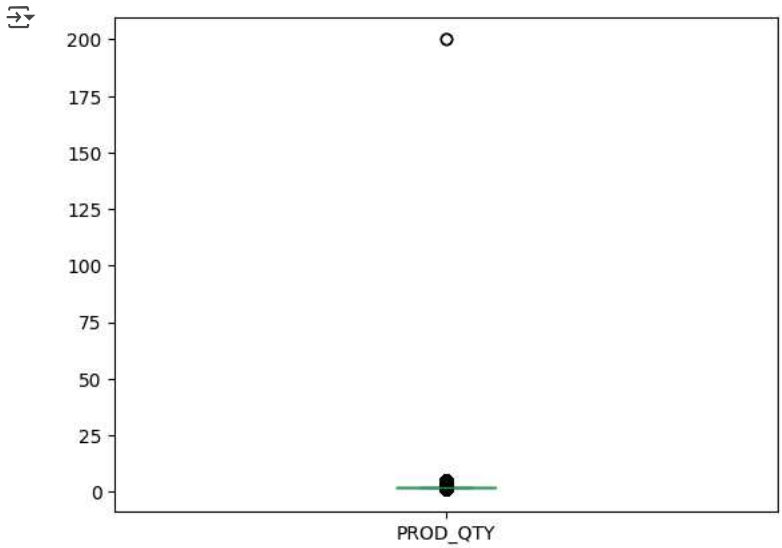
```
pd.set_option('display.max_rows', 100)
print(df)
```

```
              DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0       2018-10-17          1            1000       1         5
2       2019-05-20          1            1343     383        61
9       2018-08-18          7            7150    6900        52
12      2019-05-18          9            9208    8634        15
```

```
24     2018-08-15         38           38142   34181       108
...           ...        ...             ...     ...       ...
264830 2018-11-12        272          272319  270087        44
264831 2019-03-09        272          272319  270088        89
264833 2018-11-06        272          272379  270187        51
264834 2018-12-27        272          272379  270188        42
264835 2018-09-22        272          272380  270189        74

                                  PROD_NAME  PROD_QTY  TOT_SALES    Zscore
0        Natural Chip        Compny SeaSaltg         2        6.0  0.144009
2        Smiths Crinkle Cut  Chips Chicken g         2        2.9  0.144009
9          Grain Waves Sour    CreamChives G         2        7.2  0.144009
12                    Twisties Cheese     g         2        9.2  0.144009
24      Kettle Tortilla ChpsHnyJlpno Chili g         2        9.2  0.144009
...                                     ...       ...        ...       ...
264830           Thins Chips Light  Tangy g         2        6.6  0.144009
264831  Kettle Sweet Chilli And Sour Cream g         2       10.8  0.144009
264833                Doritos Mexicana    g         2        8.8  0.144009
264834  Doritos Corn Chip Mexican Jalapeno g         2        7.8  0.144009
264835           Tostitos Splash Of  Lime g         2        8.8  0.144009

[220069 rows x 9 columns]
```

```
# Z score

outliers = []
def detect_outliers(data):
  threshold = 3 # 3rd standard deviation #
  mean = np.mean(data)
  std = np.std(data)


  for i in data:
    z_score = (i-mean)/std
    if np.abs(z_score) > threshold:
      outliers.append(i)
  return outliers
```

```
import matplotlib.pyplot as plt
df["PROD_QTY"].plot(kind="box")
plt.show()
```




```
Q1 = df["PROD_QTY"].quantile(0.25)
Q3 = df["PROD_QTY"].quantile(0.75)
IQR = Q3 - Q1

outliers = df[(df["PROD_QTY"] < (Q1 - 1.5 * IQR)) | (df["PROD_QTY"] > (Q3 + 1.5 * IQR))]

# Show only distinct outlier values
print(outliers["PROD_QTY"].unique())
```

```
[  3   5   1   4 200]
```

```
Q1 = df["PROD_QTY"].quantile(0.25)
Q3 = df["PROD_QTY"].quantile(0.75)
IQR = Q3 - Q1

# keep only non-outliers
df = df[(df["PROD_QTY"] >= (Q1 - 1.5 * IQR)) & (df["PROD_QTY"] <= (Q3 + 1.5 * IQR))]


Q1 = df["TOT_SALES"].quantile(0.25)
Q3 = df["TOT_SALES"].quantile(0.75)
IQR = Q3 - Q1

outliers = df[(df["TOT_SALES"] < (Q1 - 1.5 * IQR)) | (df["TOT_SALES"] > (Q3 + 1.5 * IQR))]

# Show only distinct outlier values
print(outliers["TOT_SALES"].unique())
```

```
[]
```

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

```
['Natural Chip        Compny SeaSaltg'
 'Smiths Crinkle Cut  Chips Chicken g' 'Grain Waves Sour    CreamChives G'
 'Twisties Cheese     g' 'Kettle Tortilla ChpsHnyJlpno Chili g'
 'Smiths Chip Thinly  Cut Original g' 'Thins Chips Light  Tangy g'
 'Smiths Crinkle Cut  Salt  Vinegar g'
 'Doritos Corn Chip Mexican Jalapeno g' 'Pringles Sthrn FriedChicken g'
 'Infuzions Thai SweetChili PotatoMix g' 'Kettle g Swt Pot Sea Salt'
 'Pringles SourCream  Onion g' 'Dorito Corn Chp     Supreme g'
 'Twisties Chickeng' 'Cobs Popd Sea Salt  Chips g'
 'WW Original Corn    Chips g' 'WW Crinkle Cut      Chicken g'
 'Smiths Crnkle Chip  Orgnl Big Bag g' 'WW Original Stacked Chips g'
 'Doritos Corn Chip Southern Chicken g' 'Cheezels Cheese g'
 'Kettle Sensations   Siracha Lime g' 'Cheetos Puffs g' 'Kettle Chilli g'
 'Smiths Crinkle Cut  Chips ChsOniong' 'Smiths Crinkle      Original g'
 'Smith Crinkle Cut   Mac N Cheese g' 'Burger Rings g'
 'Doritos Corn Chips  Nacho Cheese g'
 'Tyrrells Crisps     Lightly Salted g'
 'Infzns Crn Crnchers Tangy Gcamole g' 'Pringles Original   Crisps g'
 'Thins Chips Seasonedchicken g' 'Grain Waves         Sweet Chilli g'
 'Kettle Mozzarella   Basil  Pesto g' 'Kettle Honey Soy    Chicken g'
 'Thins Potato Chips  Hot  Spicy g' 'Tostitos Lightly    Salted g'
 'Infuzions SourCreamHerbs Veg Strws g' 'Kettle Sensations   BBQMaple g'
 'Kettle Sensations   Camembert  Fig g'
 'Infuzions BBQ Rib   Prawn Crackers g' 'Kettle Tortilla ChpsFetaGarlic g'
 'Doritos Mexicana    g' 'Thins Chips Salt   Vinegar g'
 'Kettle Original g' 'Tyrrells Crisps     Ched  Chives g'
 'Pringles Barbeque   g' 'Kettle Tortilla ChpsBtrootRicotta g'
 'Smiths Thinly Cut   Roast Chicken g' 'CCs Nacho Cheese    g'
 'RRD Sweet Chilli   Sour Cream g' 'CCs Tasty Cheese    g'
 'Red Rock Deli Thai  ChilliLime g' 'GrnWves Plus Btroot  Chilli Jam g'
 'WW D/Style Chip     Sea Salt g' 'RRD Chilli         Coconut g'
 'Cheetos Chs  Bacon Balls g' 'Natural Chip Co     Tmato HrbSpce g'
 'Infuzions Mango     Chutny Papadums g'
 'RRD Steak          Chimuchurri g'
 'Smiths Crinkle Cut  French OnionDip g' 'Tostitos Smoked     Chipotle g'
 'RRD Honey Soy       Chicken g' 'Smiths Chip Thinly  S/CreamOnion g'
 'Cobs Popd Swt/Chlli Sr/Cream Chips g'
 'Red Rock Deli Sp    Salt  Truffle G'
 'Sunbites Whlegrn    Crisps Frch/Onin g' 'CCs Original g'
 'Tostitos Splash Of  Lime g' 'Doritos Cheese      Supreme g'
 'Twisties Cheese     Burger g' 'Pringles Mystery    Flavour g'
 'Smiths Crinkle Chips Salt  Vinegar g' 'Pringles Chicken    Salt Crips g'
 'Smiths Crinkle Cut  SnagSauce g' 'Woolworths Cheese   Rings g'
 'Pringles Slt Vingar g' 'WW Sour Cream OnionStacked Chips g'
 'Smiths Crinkle Cut  Chips Original g'
 'Smiths Thinly       Swt ChliS/CreamG'
 'Cobs Popd Sour Crm  Chives Chips g' 'Cheezels Cheese Box g'
 'NCC Sour Cream     Garden Chives g'
 'Kettle Sweet Chilli And Sour Cream g'
 'Doritos Corn Chips  Cheese Supreme g' 'RRD Salt  Vinegar  g'
 'Doritos Corn Chips  Original g' 'Pringles SweetSpcy BBQ g'
 'WW Supreme Cheese   Corn Chips g' 'Natural ChipCo Sea  Salt  Vinegr g'
 'Snbts Whlgrn Crisps CheddrMstrd g' 'Kettle Sea Salt     And Vinegar g'
 'Smiths Chip Thinly  CutSalt/Vinegrg' 'Smith Crinkle Cut   Bolognese g'
 'Red Rock Deli ChiknGarlic Aioli g' 'Thins Chips         Originl saltd g'
 'RRD Lime  Pepper   g' 'French Fries Potato Chips g'
 'Natural ChipCo      Hony Soy Chckng' 'RRD Pc Sea Salt     g'
```

```
       'Smiths Crinkle Cut  Chips Barbecue g' 'RRD SR Slow Rst     Pork Belly g'
```

```
df = df[~df["PROD_NAME"].str.contains("salsa", case=False, na=False)]


# Remove digits and '&' from product names
df["PROD_NAME"] = df["PROD_NAME"].str.replace(r'\d+', '', regex=True)   # remove digits
df["PROD_NAME"] = df["PROD_NAME"].str.replace('&', '', regex=False)     # remove '&'

# Optional: remove extra spaces left behind
df["PROD_NAME"] = df["PROD_NAME"].str.strip()


transactions_per_date = df.groupby('DATE')['TXN_ID'].count().reset_index()
transactions_per_date.rename(columns={'TXN_ID': 'Total_Transactions'}, inplace=True)

print(transactions_per_date)
```

```
           DATE  Total_Transactions
0     1/1/2019                 573
1    1/10/2019                 620
2    1/11/2019                 574
3    1/12/2019                 625
4    1/13/2019                 587
..         ...                 ...
359   9/5/2018                 633
360   9/6/2018                 680
361   9/7/2018                 612
362   9/8/2018                 621
363   9/9/2018                 652

[364 rows x 2 columns]
```

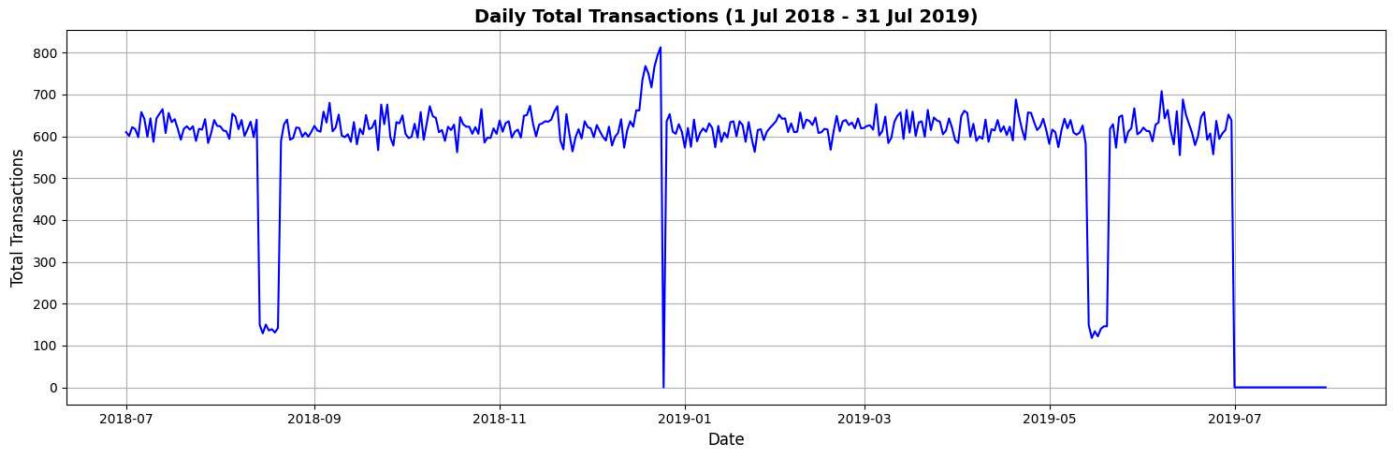
```
date_seq = pd.date_range(start="2018-07-01", end="2019-06-30", freq="D")
print(date_seq)
```

```
DatetimeIndex(['2018-07-01', '2018-07-02', '2018-07-03', '2018-07-04',
               '2018-07-05', '2018-07-06', '2018-07-07', '2018-07-08',
               '2018-07-09', '2018-07-10',
               ...
               '2019-06-21', '2019-06-22', '2019-06-23', '2019-06-24',
               '2019-06-25', '2019-06-26', '2019-06-27', '2019-06-28',
               '2019-06-29', '2019-06-30'],
              dtype='datetime64[ns]', length=365, freq='D')
```

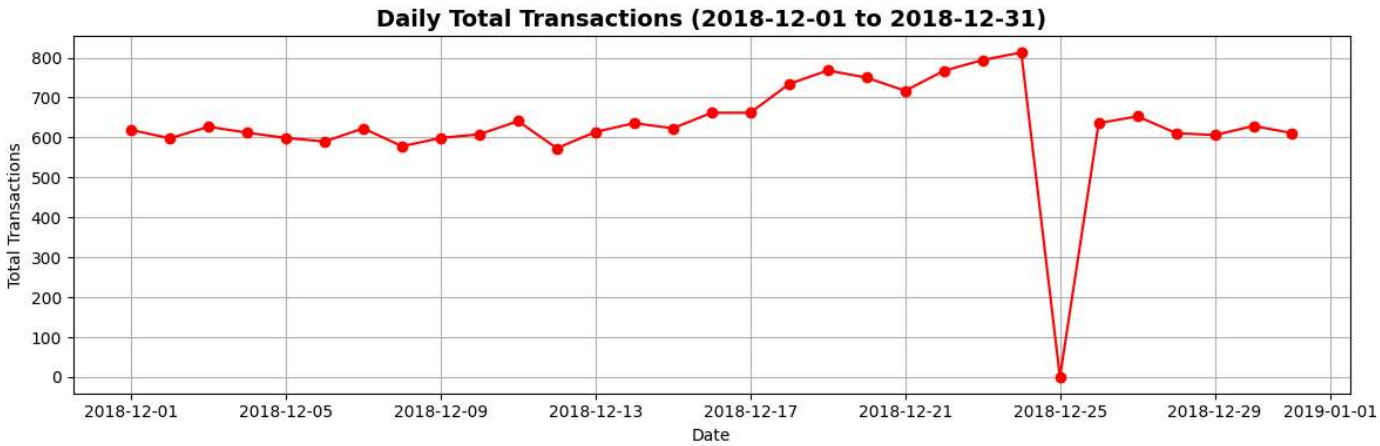
```
# Ensure DATE column is datetime
df['DATE'] = pd.to_datetime(df['DATE'])

# Create full date sequence
full_dates = pd.date_range(start='2018-07-01', end='2019-07-31')

# Count transactions per date
daily_counts = df.groupby('DATE').size().reset_index(name='TOTAL_TRANSACTIONS')

# Merge with full dates to include missing dates
daily_counts_full = pd.DataFrame({'DATE': full_dates}).merge(daily_counts, on='DATE', how='left')

# Fill missing counts with 0
daily_counts_full['TOTAL_TRANSACTIONS'] = daily_counts_full['TOTAL_TRANSACTIONS'].fillna(0)

print(daily_counts_full)
```

```
           DATE  TOTAL_TRANSACTIONS
0    2018-07-01               610.0
1    2018-07-02               601.0
2    2018-07-03               622.0
3    2018-07-04               617.0
4    2018-07-05               598.0
..          ...                 ...
391  2019-07-27                 0.0
392  2019-07-28                 0.0
393  2019-07-29                 0.0
394  2019-07-30                 0.0
395  2019-07-31                 0.0

[396 rows x 2 columns]
```

```
import matplotlib.pyplot as plt

# Plot total transactions over time
plt.figure(figsize=(15,5))
plt.plot(daily_counts_full['DATE'], daily_counts_full['TOTAL_TRANSACTIONS'], marker='', color='blue')
plt.title('Daily Total Transactions (1 Jul 2018 - 31 Jul 2019)', fontsize=14, fontweight='bold')
plt.xlabel('Date', fontsize=12)
plt.ylabel('Total Transactions', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
# Example: zoom into 1 Jan 2019 to 31 Jan 2019  """  the increase in sales occurs in dec due to christmas but on christmas sales dropped as

start_date = '2018-12-01'
end_date = '2018-12-31'

# Filter the data
zoomed_data = daily_counts_full[(daily_counts_full['DATE'] >= start_date) &
                                (daily_counts_full['DATE'] <= end_date)]

# Plot the zoomed-in data
plt.figure(figsize=(12,4))
plt.plot(zoomed_data['DATE'], zoomed_data['TOTAL_TRANSACTIONS'], marker='o', color='red')
plt.title(f'Daily Total Transactions ({start_date} to {end_date})', fontsize=14, fontweight='bold')
plt.xlabel('Date')
plt.ylabel('Total Transactions')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
# packet sizes #
import pandas as pd

# Import the file
df = pd.read_csv('QVI_transaction_data.csv')



# Extract weight (e.g., anything ending with g or kg) from product name
df['PACKET_SIZE'] = df['PROD_NAME'].str.extract(r'(\d+\s*(?:kg|g))', expand=False)

# Optional: drop rows where no weight was found
df = df.dropna(subset=['PACKET_SIZE'])

print(df[['PROD_NAME', 'PACKET_SIZE']])
```

```
                                     PROD_NAME PACKET_SIZE
0         Natural Chip        Compny SeaSalt175g        175g
1                       CCs Nacho Cheese    175g        175g
2         Smiths Crinkle Cut  Chips Chicken 170g        170g
3         Smiths Chip Thinly  S/Cream&Onion 175g        175g
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g        150g
...                                      ...         ...
264831   Kettle Sweet Chilli And Sour Cream 175g        175g
264832             Tostitos Splash Of  Lime 175g        175g
264833                   Doritos Mexicana    170g        170g
264834   Doritos Corn Chip Mexican Jalapeno 150g        150g
264835             Tostitos Splash Of  Lime 175g        175g

[258772 rows x 2 columns]
```

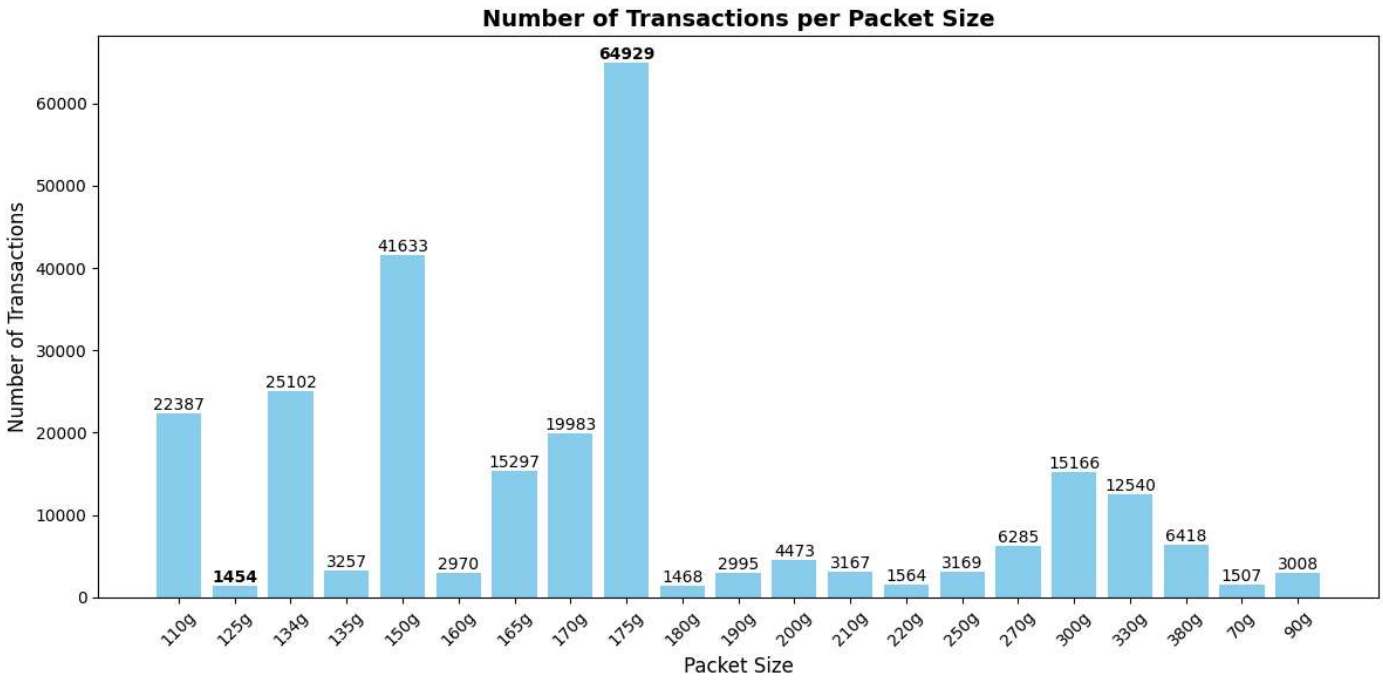
```
def size_in_grams(size):
    size = size.replace(" ", "").lower()
    if "kg" in size:
        return float(size.replace("kg","")) * 1000
    elif "g" in size:
        return float(size.replace("g",""))
    else:
        return None

# Create a temporary numeric column for comparison
df['SIZE_GRAMS'] = df['PACKET_SIZE'].apply(size_in_grams)

# Find min and max packet size strings
min_packet = df.loc[df['SIZE_GRAMS'].idxmin(), 'PACKET_SIZE']
max_packet = df.loc[df['SIZE_GRAMS'].idxmax(), 'PACKET_SIZE']

print(f"Minimum packet size: {min_packet}")
print(f"Maximum packet size: {max_packet}")
```

```
Minimum packet size: 70g
Maximum packet size: 380g
```

```
# 1️⃣ Count number of transactions per packet size
txn_counts = df.groupby('PACKET_SIZE')['TXN_ID'].count().reset_index()

# 2️⃣ Rename the column for clarity
txn_counts.columns = ['PACKET_SIZE', 'TOTAL_TRANSACTIONS']

print(txn_counts)
```

```
    PACKET_SIZE  TOTAL_TRANSACTIONS
0          110g               22387
1          125g                1454
2          134g               25102
3          135g                3257
4          150g               41633
5          160g                2970
6          165g               15297
7          170g               19983
8          175g               64929
9          180g                1468
10         190g                2995
11         200g                4473
12         210g                3167
13         220g                1564
14         250g                3169
15         270g                6285
16         300g               15166
17         330g               12540
18         380g                6418
19          70g                1507
20          90g                3008
```

```
import matplotlib.pyplot as plt

# Find max and min transactions
max_txn = txn_counts['TOTAL_TRANSACTIONS'].max()
min_txn = txn_counts['TOTAL_TRANSACTIONS'].min()

plt.figure(figsize=(12,6))
bars = plt.bar(txn_counts['PACKET_SIZE'], txn_counts['TOTAL_TRANSACTIONS'], color='skyblue')

# Add data labels
for bar in bars:
    height = bar.get_height()
    label_weight = 'bold' if height == max_txn or height == min_txn else 'normal'
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.5,  # 0.5 for slight gap
             f'{int(height)}', ha='center', va='bottom', fontweight=label_weight)

plt.title('Number of Transactions per Packet Size', fontsize=14, fontweight='bold')
plt.xlabel('Packet Size', fontsize=12)
plt.ylabel('Number of Transactions', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
# Extract first word as Brand Name
df['BRAND_NAME'] = df['PROD_NAME'].str.split().str[0]

print(df[['PROD_NAME', 'BRAND_NAME']])
```

```
                                     PROD_NAME BRAND_NAME
0         Natural Chip        Compny SeaSalt175g    Natural
1                       CCs Nacho Cheese    175g        CCs
2         Smiths Crinkle Cut  Chips Chicken 170g     Smiths
3         Smiths Chip Thinly  S/Cream&Onion 175g     Smiths
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g     Kettle
...                                        ...        ...
264831   Kettle Sweet Chilli And Sour Cream 175g     Kettle
264832             Tostitos Splash Of  Lime 175g   Tostitos
264833                  Doritos Mexicana    170g    Doritos
264834   Doritos Corn Chip Mexican Jalapeno 150g    Doritos
264835             Tostitos Splash Of  Lime 175g   Tostitos

[258772 rows x 2 columns]
```

```
# Get unique brand names
unique_brands = df['BRAND_NAME'].unique()

# Show first 100
print(unique_brands[:100])
```

```
['Natural' 'CCs' 'Smiths' 'Kettle' 'Old' 'Grain' 'Doritos' 'Twisties' 'WW'
 'Thins' 'Burger' 'NCC' 'Cheezels' 'Infzns' 'Red' 'Pringles' 'Dorito'
 'Infuzions' 'Smith' 'GrnWves' 'Tyrrells' 'Cobs' 'Woolworths' 'French'
 'RRD' 'Tostitos' 'Cheetos' 'Snbts' 'Sunbites']
```

```
# Example unique brands
brands = ['Natural', 'CCs', 'Smiths', 'Kettle', 'Old', 'Grain', 'Doritos',
          'Twisties', 'WW', 'Thins', 'Burger', 'NCC', 'Cheezels', 'Infzns',
          'Red', 'Pringles', 'Dorito', 'Infuzions', 'Smith', 'GrnWves',
          'Tyrrells', 'Cobs', 'Woolworths', 'French', 'RRD', 'Tostitos',
          'Cheetos', 'Snbts', 'Sunbites']

# Create a mapping table (old → new)
replace_dict = {
    'Red': 'RRD',
```

```
    'Infzns': 'Infuzions',
    'Smith': 'Smiths',
    'Snbts': 'Sunbites',
    'WW': 'Woolworths',
    'Tostitos': 'Cheetos'
}

# Apply replacement
df['BRAND_NAME'] = df['BRAND_NAME'].replace(replace_dict)

# Check cleaned unique brand names
print(df['BRAND_NAME'].unique())
```

```
['Natural' 'CCs' 'Smiths' 'Kettle' 'Old' 'Grain' 'Doritos' 'Twisties'
 'Woolworths' 'Thins' 'Burger' 'NCC' 'Cheezels' 'Infuzions' 'RRD'
 'Pringles' 'Dorito' 'GrnWves' 'Tyrrells' 'Cobs' 'French' 'Cheetos'
 'Sunbites']
```

```
# Assuming your DataFrames are named transactionData and customerData
# Merge transaction data to customer data using left join
purchase_df = pd.read_csv("QVI_purchase_behaviour.csv")
data = df.merge(purchase_df, on='LYLTY_CARD_NBR', how='left')

# Check result
print(data.head())
```

```
          DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0  10/17/2018          1            1000       1         5
1   5/14/2019          1            1307     348        66
2   5/20/2019          1            1343     383        61
3   8/17/2018          2            2373     974        69
4   8/18/2018          2            2426    1038       108

                                PROD_NAME  PROD_QTY  TOT_SALES PACKET_SIZE  \
0    Natural Chip        Compny SeaSalt175g         2        6.0        175g
1                  CCs Nacho Cheese    175g         3        6.3        175g
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9        170g
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0        175g
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8        150g

   PACKET_SIZE_KG  SIZE_GRAMS BRAND_NAME              LIFESTAGE  \
0           0.175       175.0    Natural   YOUNG SINGLES/COUPLES
1           0.175       175.0        CCs  MIDAGE SINGLES/COUPLES
2           0.170       170.0     Smiths  MIDAGE SINGLES/COUPLES
3           0.175       175.0     Smiths  MIDAGE SINGLES/COUPLES
4           0.150       150.0     Kettle  MIDAGE SINGLES/COUPLES

  PREMIUM_CUSTOMER
0          Premium
1           Budget
2           Budget
3           Budget
4           Budget
```

```
import pandas as pd

data['TOT_SALES'] = pd.to_numeric(data['TOT_SALES'], errors='coerce')

# ✅ Group by Lifestage and sum total sales
sales_by_lifestage = (
    data.groupby('LIFESTAGE')['TOT_SALES']
    .sum()
    .reset_index()
    .sort_values(by='TOT_SALES', ascending=False)
)

print(sales_by_lifestage)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-3606330634.py in <cell line: 0>()
      1 import pandas as pd
      2
----> 3 data['TOT_SALES'] = pd.to_numeric(data['TOT_SALES'], errors='coerce')
      4
      5 # ✅ Group by Lifestage and sum total sales

NameError: name 'data' is not defined
```

```
transactions = pd.read_csv("QVI_transaction_data.csv")
customers = pd.read_csv("QVI_purchase_behaviour.csv")

# Merge them
data = transactions.merge(customers, on="LYLTY_CARD_NBR", how="left")
```

```
from google.colab import files

uploaded = files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
from google.colab import files

uploaded = files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
import pandas as pd

transactions = pd.read_csv("QVI_transaction_data.csv")
customers = pd.read_csv("QVI_purchase_behaviour.csv")

# Merge on customer card number
data = transactions.merge(customers, on="LYLTY_CARD_NBR", how="left")

print(data.head())
```

```
         DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0  10/17/2018          1            1000       1         5
1   5/14/2019          1            1307     348        66
2   5/20/2019          1            1343     383        61
3   8/17/2018          2            2373     974        69
4   8/18/2018          2            2426    1038       108

                                PROD_NAME  PROD_QTY  TOT_SALES  \
0    Natural Chip        Compny SeaSalt175g         2        6.0
1                  CCs Nacho Cheese    175g         3        6.3
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8

                LIFESTAGE PREMIUM_CUSTOMER
0   YOUNG SINGLES/COUPLES          Premium
1  MIDAGE SINGLES/COUPLES           Budget
2  MIDAGE SINGLES/COUPLES           Budget
3  MIDAGE SINGLES/COUPLES           Budget
4  MIDAGE SINGLES/COUPLES           Budget
```

```
# Check if any null values exist
print(data.isnull().sum())

# Quick check: are there ANY nulls in the dataset?
print("\nTotal null values:", data.isnull().sum().sum())
```

```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
```

```
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64

Total null values: 0
```

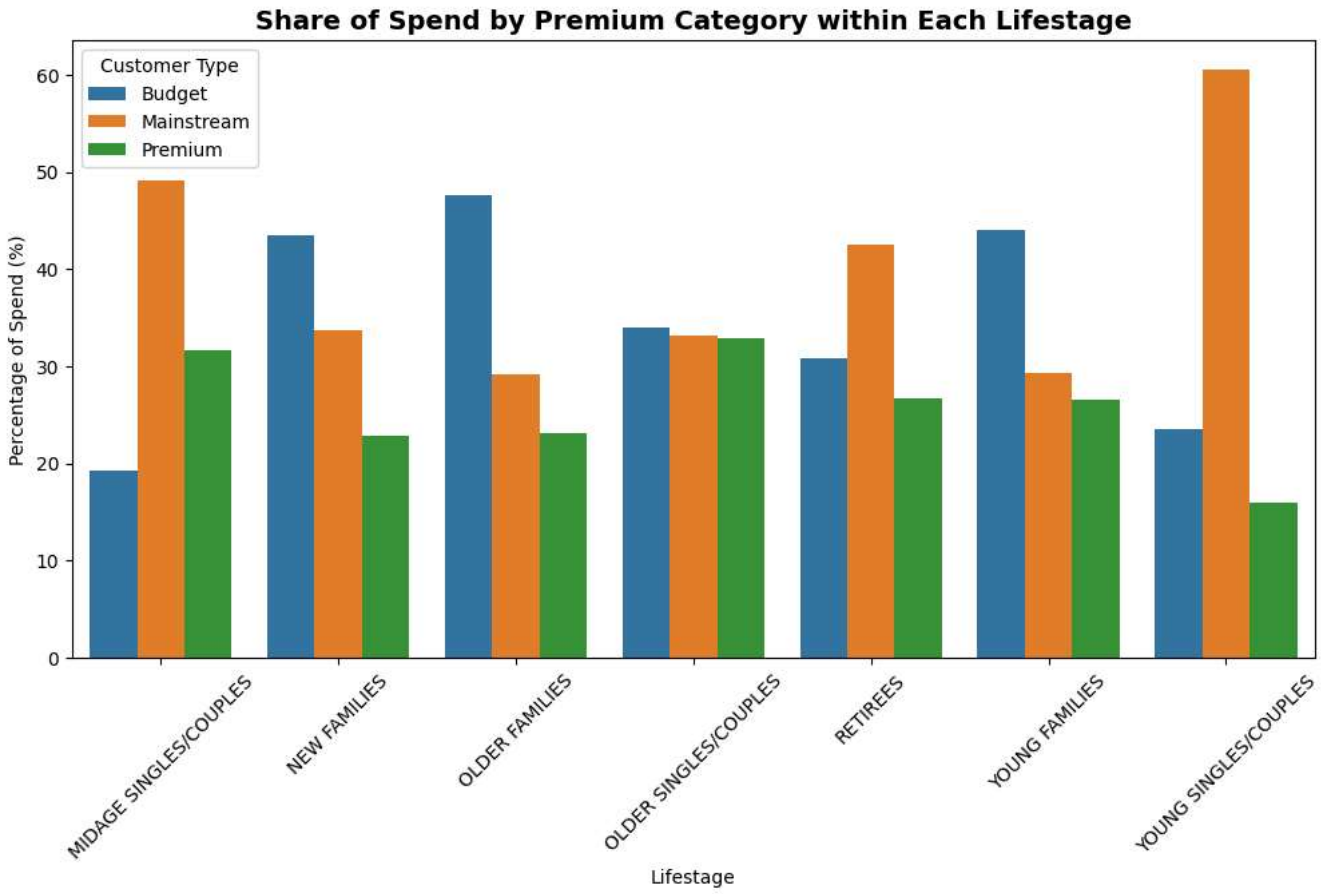
```
# Make sure TOT_SALES is numeric
data['TOT_SALES'] = pd.to_numeric(data['TOT_SALES'], errors='coerce')

# Group by Lifestage and calculate total spend
sales_by_lifestage = (
    data.groupby('LIFESTAGE')['TOT_SALES']
    .sum()
    .reset_index()
    .sort_values(by='TOT_SALES', ascending=False)
)

print(sales_by_lifestage)
```

```
                LIFESTAGE  TOT_SALES
3   OLDER SINGLES/COUPLES  402426.75
4               RETIREES  366470.90
2          OLDER FAMILIES  353767.20
5          YOUNG FAMILIES  316160.10
6   YOUNG SINGLES/COUPLES  260405.30
0  MIDAGE SINGLES/COUPLES  184751.30
1            NEW FAMILIES   50433.45
```

```
# Group total sales by lifestage & premium status
premium_split = (
    data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES']
    .sum()
    .reset_index()
)

# Calculate total sales per lifestage
lifestage_totals = premium_split.groupby('LIFESTAGE')['TOT_SALES'].sum().reset_index()
lifestage_totals.rename(columns={'TOT_SALES': 'LIFESTAGE_TOTAL_SALES'}, inplace=True)

# Merge lifestage totals back to the premium_split dataframe
premium_split = premium_split.merge(lifestage_totals, on='LIFESTAGE', how='left')

# Calculate percentage share
premium_split['PCT'] = (premium_split['TOT_SALES'] / premium_split['LIFESTAGE_TOTAL_SALES']) * 100

# Drop the intermediate lifestage total column
premium_split.drop(columns=['LIFESTAGE_TOTAL_SALES'], inplace=True)


print(premium_split)
```

```
                 LIFESTAGE PREMIUM_CUSTOMER  TOT_SALES        PCT
0   MIDAGE SINGLES/COUPLES          Budget   35514.80  19.223031
1   MIDAGE SINGLES/COUPLES      Mainstream   90803.85  49.149235
2   MIDAGE SINGLES/COUPLES         Premium   58432.65  31.627734
3             NEW FAMILIES          Budget   21928.45  43.479972
4             NEW FAMILIES      Mainstream   17013.90  33.735348
5             NEW FAMILIES         Premium   11491.10  22.784680
6           OLDER FAMILIES          Budget  168363.25  47.591538
7           OLDER FAMILIES      Mainstream  103445.55  29.241137
8           OLDER FAMILIES         Premium   81958.40  23.167326
9    OLDER SINGLES/COUPLES          Budget  136769.80  33.986260
10  OLDER SINGLES/COUPLES      Mainstream  133393.80  33.147349
11  OLDER SINGLES/COUPLES         Premium  132263.15  32.866391
12              RETIREES          Budget  113147.80  30.874975
13              RETIREES      Mainstream  155677.05  42.480058
14              RETIREES         Premium   97646.05  26.644967
15          YOUNG FAMILIES          Budget  139345.85  44.074458
16          YOUNG FAMILIES      Mainstream   92788.75  29.348659
17          YOUNG FAMILIES         Premium   84025.50  26.576883
18  YOUNG SINGLES/COUPLES          Budget   61141.60  23.479399
19  YOUNG SINGLES/COUPLES      Mainstream  157621.60  60.529336
20  YOUNG SINGLES/COUPLES         Premium   41642.10  15.991264
```

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12,6))
sns.barplot(data=premium_split,
            x='LIFESTAGE',
            y='PCT',
            hue='PREMIUM_CUSTOMER')

plt.title("Share of Spend by Premium Category within Each Lifestage", fontsize=14, fontweight="bold")
plt.ylabel("Percentage of Spend (%)")
plt.xlabel("Lifestage")
plt.xticks(rotation=45)
plt.legend(title="Customer Type")
plt.show()
```




```
# count of customers per lifestage
data.groupby("LIFESTAGE")["LYLTY_CARD_NBR"].nunique().reset_index(name="total_customers")
```

| | LIFESTAGE | total_customers |
|---|---|---|
| 0 | MIDAGE SINGLES/COUPLES | 7275 |
| 1 | NEW FAMILIES | 2549 |
| 2 | OLDER FAMILIES | 9780 |
| 3 | OLDER SINGLES/COUPLES | 14609 |
| 4 | RETIREES | 14805 |
| 5 | YOUNG FAMILIES | 9178 |
| 6 | YOUNG SINGLES/COUPLES | 14441 |

```
# average price per unit by lifestage and premium customer
average_price_per_unit = (
    data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES'].sum() /
    data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['PROD_QTY'].sum()
).reset_index(name='AVG_PRICE_PER_UNIT')
```

```
print(average_price_per_unit)
```

```
                LIFESTAGE PREMIUM_CUSTOMER  AVG_PRICE_PER_UNIT
0   MIDAGE SINGLES/COUPLES           Budget            3.739975
1   MIDAGE SINGLES/COUPLES       Mainstream            4.000346
2   MIDAGE SINGLES/COUPLES          Premium            3.763535
3              NEW FAMILIES           Budget            3.936178
4              NEW FAMILIES       Mainstream            3.939315
5              NEW FAMILIES          Premium            3.886067
6           OLDER FAMILIES           Budget            3.736009
7           OLDER FAMILIES       Mainstream            3.726962
8           OLDER FAMILIES          Premium            3.696649
9    OLDER SINGLES/COUPLES           Budget            3.883299
10   OLDER SINGLES/COUPLES       Mainstream            3.811578
11   OLDER SINGLES/COUPLES          Premium            3.891695
12                RETIREES           Budget            3.933660
13                RETIREES       Mainstream            3.842170
14                RETIREES          Premium            3.924050
15          YOUNG FAMILIES           Budget            3.754840
16          YOUNG FAMILIES       Mainstream            3.705029
17          YOUNG FAMILIES          Premium            3.750134
18   YOUNG SINGLES/COUPLES           Budget            3.667542
19   YOUNG SINGLES/COUPLES       Mainstream            4.080079
20   YOUNG SINGLES/COUPLES          Premium            3.675060
```

```
# total chips bought per customer by segment
data.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["PROD_QTY"].sum().reset_index(name="total_chips_bought")
```

| | LIFESTAGE | PREMIUM_CUSTOMER | total_chips_bought |
|---|---|---|---|
| 0 | MIDAGE SINGLES/COUPLES | Budget | 9496 |
| 1 | MIDAGE SINGLES/COUPLES | Mainstream | 22699 |
| 2 | MIDAGE SINGLES/COUPLES | Premium | 15526 |
| 3 | NEW FAMILIES | Budget | 5571 |
| 4 | NEW FAMILIES | Mainstream | 4319 |
| 5 | NEW FAMILIES | Premium | 2957 |
| 6 | OLDER FAMILIES | Budget | 45065 |
| 7 | OLDER FAMILIES | Mainstream | 27756 |
| 8 | OLDER FAMILIES | Premium | 22171 |
| 9 | OLDER SINGLES/COUPLES | Budget | 35220 |
| 10 | OLDER SINGLES/COUPLES | Mainstream | 34997 |
| 11 | OLDER SINGLES/COUPLES | Premium | 33986 |
| 12 | RETIREES | Budget | 28764 |
| 13 | RETIREES | Mainstream | 40518 |
| 14 | RETIREES | Premium | 24884 |
| 15 | YOUNG FAMILIES | Budget | 37111 |
| 16 | YOUNG FAMILIES | Mainstream | 25044 |
| 17 | YOUNG FAMILIES | Premium | 22406 |
| 18 | YOUNG SINGLES/COUPLES | Budget | 16671 |
| 19 | YOUNG SINGLES/COUPLES | Mainstream | 38632 |
| 20 | YOUNG SINGLES/COUPLES | Premium | 11331 |

```
import pandas as pd

# Ensure DATE is in datetime format
data['DATE'] = pd.to_datetime(data['DATE'])

# Extract month/year for period analysis
data['YearMonth'] = data['DATE'].dt.to_period('M')

# Total spend per customer per period
total_spend_per_period = data.groupby(['LYLTY_CARD_NBR', 'YearMonth'])['TOT_SALES'].sum().reset_index(name='customer_total_spend')
```

```
# Total chip spend per customer per period
chip_spend_per_period = data[data['PROD_NAME'].str.contains('chip', case=False, na=False)] \
    .groupby(['LYLTY_CARD_NBR', 'YearMonth'])['TOT_SALES'].sum().reset_index(name='chip_total_spend')

# Merge both to get proportion
spend_summary = total_spend_per_period.merge(chip_spend_per_period, on=['LYLTY_CARD_NBR','YearMonth'], how='left').fillna(0)
spend_summary['chips_proportion'] = spend_summary['chip_total_spend'] / spend_summary['customer_total_spend']

print(spend_summary.head())
```

```
   LYLTY_CARD_NBR YearMonth  customer_total_spend  chip_total_spend  \
0            1000   2018-10                   6.0               6.0
1            1002   2018-09                   2.7               0.0
2            1003   2019-03                   6.6               3.0
3            1004   2018-11                   1.9               1.9
4            1005   2018-12                   2.8               0.0

   chips_proportion
0          1.000000
1          0.000000
2          0.454545
3          1.000000
4          0.000000
```

```
import pandas as pd

def lifestage_chip_comparison(data):
    # Overall transaction proportion by lifestage
    overall = data.groupby('LIFESTAGE')['TXN_ID'].nunique().reset_index()
    overall['proportion_overall'] = overall['TXN_ID'] / overall['TXN_ID'].sum()

    # Chip-buying transaction proportion by lifestage
    chips = data[data['PROD_NAME'].str.contains('Chips', case=False, na=False)]
    chips = chips.groupby('LIFESTAGE')['TXN_ID'].nunique().reset_index()
    chips['proportion_chips'] = chips['TXN_ID'] / chips['TXN_ID'].sum()

    # Merge for comparison
    comparison = overall.merge(chips, on='LIFESTAGE', how='left').fillna(0)

    return comparison

# Example usage
comparison_table = lifestage_chip_comparison(data)
print(comparison_table)
```

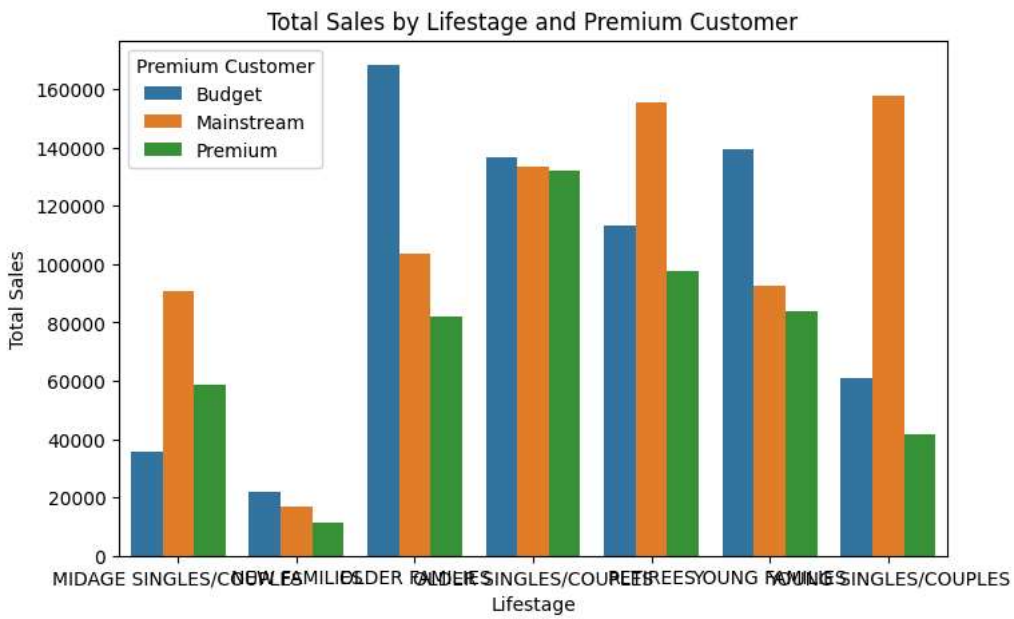
```
                LIFESTAGE  TXN_ID_x  proportion_overall  TXN_ID_y  \
0  MIDAGE SINGLES/COUPLES     24949            0.094817      4727
1            NEW FAMILIES      6896            0.026208      1288
2          OLDER FAMILIES     48126            0.182900      9084
3   OLDER SINGLES/COUPLES     54147            0.205783     10255
4                RETIREES     49512            0.188168      9334
5          YOUNG FAMILIES     43242            0.164339      8145
6   YOUNG SINGLES/COUPLES     36255            0.137785      6879

   proportion_chips
0          0.095088
1          0.025909
2          0.182733
3          0.206288
4          0.187762
5          0.163844
6          0.138377
```

```
import pandas as pd

# Calculate total sales per segment
tot_sales = data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES'].sum().reset_index()

print(tot_sales)
```

```
                 LIFESTAGE PREMIUM_CUSTOMER  TOT_SALES
0   MIDAGE SINGLES/COUPLES           Budget   35514.80
1   MIDAGE SINGLES/COUPLES       Mainstream   90803.85
2   MIDAGE SINGLES/COUPLES          Premium   58432.65
3             NEW FAMILIES           Budget   21928.45
4             NEW FAMILIES       Mainstream   17013.90
5             NEW FAMILIES          Premium   11491.10
6           OLDER FAMILIES           Budget  168363.25
```

```
7           OLDER FAMILIES       Mainstream  103445.55
8           OLDER FAMILIES          Premium   81958.40
9    OLDER SINGLES/COUPLES           Budget  136769.80
10   OLDER SINGLES/COUPLES       Mainstream  133393.80
11   OLDER SINGLES/COUPLES          Premium  132263.15
12                RETIREES           Budget  113147.80
13                RETIREES       Mainstream  155677.05
14                RETIREES          Premium   97646.05
15          YOUNG FAMILIES           Budget  139345.85
16          YOUNG FAMILIES       Mainstream   92788.75
17          YOUNG FAMILIES          Premium   84025.50
18   YOUNG SINGLES/COUPLES           Budget   61141.60
19   YOUNG SINGLES/COUPLES       Mainstream  157621.60
20   YOUNG SINGLES/COUPLES          Premium   41642.10
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot
plt.figure(figsize=(8,5))
sns.barplot(data=tot_sales, x='LIFESTAGE', y='TOT_SALES', hue='PREMIUM_CUSTOMER')
plt.title('Total Sales by Lifestage and Premium Customer')
plt.ylabel('Total Sales')
plt.xlabel('Lifestage')
plt.legend(title='Premium Customer')
plt.show()
```




```
# Total quantity purchased per LIFESTAGE
total_qty_per_segment = data.groupby("LIFESTAGE")["PROD_QTY"].sum().reset_index(name="TOTAL_QTY")
total_qty_per_segment
```

| | LIFESTAGE | TOTAL_QTY |
|---|---|---|
| 0 | MIDAGE SINGLES/COUPLES | 47721 |
| 1 | NEW FAMILIES | 12847 |
| 2 | OLDER FAMILIES | 94992 |
| 3 | OLDER SINGLES/COUPLES | 104203 |
| 4 | RETIREES | 94166 |
| 5 | YOUNG FAMILIES | 84561 |
| 6 | YOUNG SINGLES/COUPLES | 66634 |

```
import pandas as pd
from scipy.stats import ttest_ind

# Assuming your merged dataframe is named data
# Step 1: Compute unit price
```

```
data['UNIT_PRICE'] = data['TOT_SALES'] / data['PROD_QTY']

# Step 2: Function to perform t-test for each LIFESTAGE & household type
def ttest_by_segment(lifestage):
    mainstream = data[(data['PREMIUM_CUSTOMER']=='Mainstream') & (data['LIFESTAGE']==lifestage)]['UNIT_PRICE']
    budget_premium = data[(data['PREMIUM_CUSTOMER'].isin(['Budget','Premium'])) & (data['LIFESTAGE']==lifestage)]['UNIT_PRICE']

    if len(mainstream) > 0 and len(budget_premium) > 0:
        t_stat, p_value = ttest_ind(mainstream, budget_premium, equal_var=False)
        signif = "Yes" if p_value < 0.05 else "No"
        return pd.Series([p_value, signif])
    else:
        return pd.Series([None, "Not enough data"])

# Step 3: Apply for unique LIFESTAGE values
results = pd.DataFrame(data['LIFESTAGE'].unique(), columns=['LIFESTAGE'])
results[['p_value','Significant']] = results['LIFESTAGE'].apply(lambda x: ttest_by_segment(x))

print(results)
```

```
                 LIFESTAGE        p_value Significant
0   YOUNG SINGLES/COUPLES  3.717918e-297         Yes
1  MIDAGE SINGLES/COUPLES   7.444769e-77         Yes
2            NEW FAMILIES   6.178404e-01          No
3          OLDER FAMILIES   7.599237e-01          No
4   OLDER SINGLES/COUPLES   6.230241e-15         Yes
5                RETIREES   1.231591e-19         Yes
6          YOUNG FAMILIES   9.493935e-05         Yes
```

```
import pandas as pd

# Filter for LIFESTAGE = 'YOUNG SINGLES/COUPLES'
young_group = data[data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES']

# Aggregate total quantity purchased per PROD_NAME
brand_pref = young_group.groupby('PROD_NAME')['PROD_QTY'].sum().reset_index(name='TOTAL_QTY')

# Sort descending to see the most popular brands
brand_pref = brand_pref.sort_values(by='TOTAL_QTY', ascending=False)

print(brand_pref)
```

```
                                  PROD_NAME  TOTAL_QTY
33     Kettle Mozzarella   Basil & Pesto 175g        956
98              Tostitos Splash Of  Lime 175g        945
15    Doritos Corn Chips  Cheese Supreme 170g        909
53           Pringles Mystery    Flavour 134g        906
37   Kettle Sensations   Camembert & Fig 150g        889
..                                        ...        ...
43     NCC Sour Cream &    Garden Chives 175g        282
105         WW Crinkle Cut      Original 175g        282
71         Smith Crinkle Cut   Bolognese 150g        280
65       RRD Steak &         Chimuchurri 150g        277
88     Smiths Thinly Cut   Roast Chicken 175g        274

[114 rows x 2 columns]
```

```
import pandas as pd

# Create first word column, strip spaces, make consistent capitalization
data['BRAND_FIRST'] = data['PROD_NAME'].str.split().str[0].str.strip().str.capitalize()

# Updated replacements
brand_replacements = {
    "Dorito": "Doritos",
    "Tostitos": "Cheetos",
    "Tyrrells": "Cheezels",
    "Infzns": "Infuzions",
    "Smith": "Smiths",
    "Grnwves": "Grain",
    "Red": "RRD",
    "Rrd": "RRD",
    "Sbnts": "Sunbites",
    "Snbts": "Sunbites",
    "Ww": "Woolworths"
}

data['BRAND_FIRST'] = data['BRAND_FIRST'].replace(brand_replacements)
```

```
# Aggregate total quantity purchased per updated first word
brand_pref = data.groupby('BRAND_FIRST')['PROD_QTY'].sum().reset_index(name='TOTAL_QTY')

# Sort descending
brand_pref = brand_pref.sort_values(by='TOTAL_QTY', ascending=False)

print(brand_pref)
```

```
    BRAND_FIRST  TOTAL_QTY
9       Kettle      79051
15      Smiths      60339
5      Doritos      54216
13    Pringles      48019
14         RRD      33646
19  Woolworths      27856
8    Infuzions      27119
17       Thins      26929
2      Cheetos      23664
3     Cheezels      21045
4         Cobs      18571
18    Twisties      18118
12         Old      17805
7        Grain      14726
10     Natural      11424
1          Ccs       8609
16    Sunbites       5692
0       Burger       2970
11         Ncc       2682
6       French       2643
```

```
# Aggregate total quantity purchased per updated first word
brand_pref = data.groupby('BRAND_FIRST')['PROD_QTY'].sum().reset_index(name='TOTAL_QTY')

# Sort descending by TOTAL_QTY and take top 3
top_3_brands = brand_pref.sort_values(by='TOTAL_QTY', ascending=False).head(3)

print(top_3_brands)
```

```
    BRAND_FIRST  TOTAL_QTY
9       Kettle      79051
15      Smiths      60339
5      Doritos      54216
```

```
import pandas as pd

# Extract numbers followed by optional letters (e.g., 75g, 120kg, 200g)
data['PACK_WEIGHT'] = data['PROD_NAME'].str.extract(r'(\d+\s*(?:g|kg|ml|l))', expand=False)

# Strip any extra spaces
data['PACK_WEIGHT'] = data['PACK_WEIGHT'].str.replace(' ', '')

# Preview
print(data[['PROD_NAME', 'PACK_WEIGHT']].head(10))
```

```
                                PROD_NAME PACK_WEIGHT
0    Natural Chip        Compny SeaSalt175g        175g
1                 CCs Nacho Cheese    175g        175g
2    Smiths Crinkle Cut  Chips Chicken 170g        170g
3    Smiths Chip Thinly  S/Cream&Onion 175g        175g
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g        150g
5  Old El Paso Salsa   Dip Tomato Mild 300g        300g
6  Smiths Crinkle Chips Salt & Vinegar 330g        330g
7     Grain Waves         Sweet Chilli 210g        210g
8   Doritos Corn Chip Mexican Jalapeno 150g        150g
9     Grain Waves Sour    Cream&Chives 210G         NaN
```

```
import re
import pandas as pd

# 1. Extract pack weight from PROD_NAME
df["PACK_WEIGHT"] = df["PROD_NAME"].str.extract(r"(\d+\s?[gGmM][lL]?\b|\d+\s?[kK][gG])")

# 2. Clean pack weight (standardize - remove spaces, lower case)
```

```
df["PACK_WEIGHT"] = df["PACK_WEIGHT"].str.replace(" ", "").str.lower()

# 3. Split population into Young Singles/Couples vs Rest
young = df[df["LIFESTAGE"] == "YOUNG SINGLES/COUPLES"]
```