```python
import pandas as pd
df = pd.read_csv(r"C:\Users\USER\OneDrive\Desktop\sql\programming\R STUDIO\QVI_data.csv")

 # Show first 5 rows
print(df.head())
```

```
   LYLTY_CARD_NBR        DATE  STORE_NBR  TXN_ID  PROD_NBR  \
0            1000  2018-10-17          1       1         5
1            1002  2018-09-16          1       2        58
2            1003  2019-03-07          1       3        52
3            1003  2019-03-08          1       4       106
4            1004  2018-11-02          1       5        96

                                PROD_NAME  PROD_QTY  TOT_SALES
PACK_SIZE  \
0  Natural Chip        Compny SeaSalt175g         2        6.0
175
1   Red Rock Deli Chikn&Garlic Aioli 150g         1        2.7
150
2   Grain Waves Sour    Cream&Chives 210G         1        3.6
210
3  Natural ChipCo      Hony Soy Chckn175g         1        3.0
175
4          WW Original Stacked Chips 160g         1        1.9
160

        BRAND              LIFESTAGE PREMIUM_CUSTOMER
0     NATURAL  YOUNG SINGLES/COUPLES          Premium
1         RRD  YOUNG SINGLES/COUPLES       Mainstream
2     GRNWVES         YOUNG FAMILIES           Budget
3     NATURAL         YOUNG FAMILIES           Budget
4  WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream
```

```python
#Check basic info (columns, datatypes, missing values)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264834 non-null  int64
 1   DATE              264834 non-null  object
 2   STORE_NBR         264834 non-null  int64
 3   TXN_ID            264834 non-null  int64
 4   PROD_NBR          264834 non-null  int64
 5   PROD_NAME         264834 non-null  object
 6   PROD_QTY          264834 non-null  int64
```

```
 7   TOT_SALES         264834 non-null  float64
 8   PACK_SIZE         264834 non-null  int64
 9   BRAND             264834 non-null  object
 10  LIFESTAGE         264834 non-null  object
 11  PREMIUM_CUSTOMER  264834 non-null  object
dtypes: float64(1), int64(6), object(5)
memory usage: 24.2+ MB
None
```

```
 #View column names
print(df.columns)
```

```
Index(['LYLTY_CARD_NBR', 'DATE', 'STORE_NBR', 'TXN_ID', 'PROD_NBR',
       'PROD_NAME', 'PROD_QTY', 'TOT_SALES', 'PACK_SIZE', 'BRAND',
'LIFESTAGE',
       'PREMIUM_CUSTOMER'],
      dtype='object')
```

```
#Summary statistics
print(df.describe())
```

```
       LYLTY_CARD_NBR      STORE_NBR        TXN_ID       PROD_NBR  \
count    2.648340e+05  264834.000000  2.648340e+05  264834.000000
mean     1.355488e+05     135.079423  1.351576e+05      56.583554
std      8.057990e+04      76.784063  7.813292e+04      32.826444
min      1.000000e+03       1.000000  1.000000e+00       1.000000
25%      7.002100e+04      70.000000  6.760050e+04      28.000000
50%      1.303570e+05     130.000000  1.351365e+05      56.000000
75%      2.030940e+05     203.000000  2.026998e+05      85.000000
max      2.373711e+06     272.000000  2.415841e+06     114.000000

            PROD_QTY      TOT_SALES      PACK_SIZE
count  264834.000000  264834.000000  264834.000000
mean        1.905813       7.299346     182.425512
std         0.343436       2.527241      64.325148
min         1.000000       1.500000      70.000000
25%         2.000000       5.400000     150.000000
50%         2.000000       7.400000     170.000000
75%         2.000000       9.200000     175.000000
max         5.000000      29.500000     380.000000
```

```
#Check for missing values?
print(df.isnull().sum())
```

```
LYLTY_CARD_NBR      0
DATE                0
STORE_NBR           0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
```

```
TOT_SALES           0
PACK_SIZE           0
BRAND               0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

```
#We'll focus on data before February 2019.

df=pre_trial_df = df[df['DATE'] < '2019-02-01']

df['DATE'] = pd.to_datetime(df['DATE'])
df = pre_trial_df = df[df['DATE'] < '2019-02-01']

#extract month for grouping
pre_trial_df['MONTH'] = pre_trial_df['DATE'].dt.to_period('M')

#Aggregate Monthly Metrics
monthly_metrics = pre_trial_df.groupby(['STORE_NBR', 'MONTH']).agg(
    total_sales=('TOT_SALES', 'sum'),
    num_customers=('LYLTY_CARD_NBR', pd.Series.nunique),
    num_transactions=('TXN_ID', 'nunique')
).reset_index()

 #Add transactions per customer
monthly_metrics['txn_per_cust'] = monthly_metrics['num_transactions']
/ monthly_metrics['num_customers']

 #Average Monthly Metrics per Store

store_summary = monthly_metrics.groupby('STORE_NBR').agg({
    'total_sales': 'mean',
    'num_customers': 'mean',
    'txn_per_cust': 'mean'
}).reset_index()

#Identify Control Stores
#For each trial store, compute similarity (e.g., Euclidean distance)
to all other stores:

from sklearn.metrics.pairwise import euclidean_distances

trial_stores = [77, 86, 88]
trial_data =
store_summary[store_summary['STORE_NBR'].isin(trial_stores)]
potential_controls =
store_summary[~store_summary['STORE_NBR'].isin(trial_stores)]

for _, trial_row in trial_data.iterrows():
    trial_vector = trial_row[['total_sales', 'num_customers',
'txn_per_cust']].values.reshape(1, -1)
```

```
    control_vectors = potential_controls[['total_sales',
'num_customers', 'txn_per_cust']].values
    distances = euclidean_distances(trial_vector, control_vectors)

    best_match_idx = distances.argmin()
    matched_store = potential_controls.iloc[best_match_idx]
['STORE_NBR']

    print(f"Trial store {trial_row['STORE_NBR']} best matches with
Control store {matched_store}")
#Let me know if you'd like help visualizing this, exporting the
matched pairs, or validating store operational periods. I can provide
plots or checks too.
```

```
Trial store 77.0 best matches with Control store 188.0
Trial store 86.0 best matches with Control store 196.0
Trial store 88.0 best matches with Control store 237.0
```