In [1]:
```
#   Importing the necessary libraries/modules.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import xlrd
%matplotlib inline

#   Ignoring any warnings.

import warnings
```

In [2]:
```
transaction_data=pd.read_excel("QVI_transaction_data.xlsx") #   Reading the Ex
transaction_data
```

Out[2]:

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

264836 rows × 8 columns

In [3]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DATE            264836 non-null  int64
 1   STORE_NBR       264836 non-null  int64
 2   LYLTY_CARD_NBR  264836 non-null  int64
 3   TXN_ID          264836 non-null  int64
 4   PROD_NBR        264836 non-null  int64
 5   PROD_NAME       264836 non-null  object
 6   PROD_QTY        264836 non-null  int64
 7   TOT_SALES       264836 non-null  float64
dtypes: float64(1), int64(6), object(1)
memory usage: 16.2+ MB
```

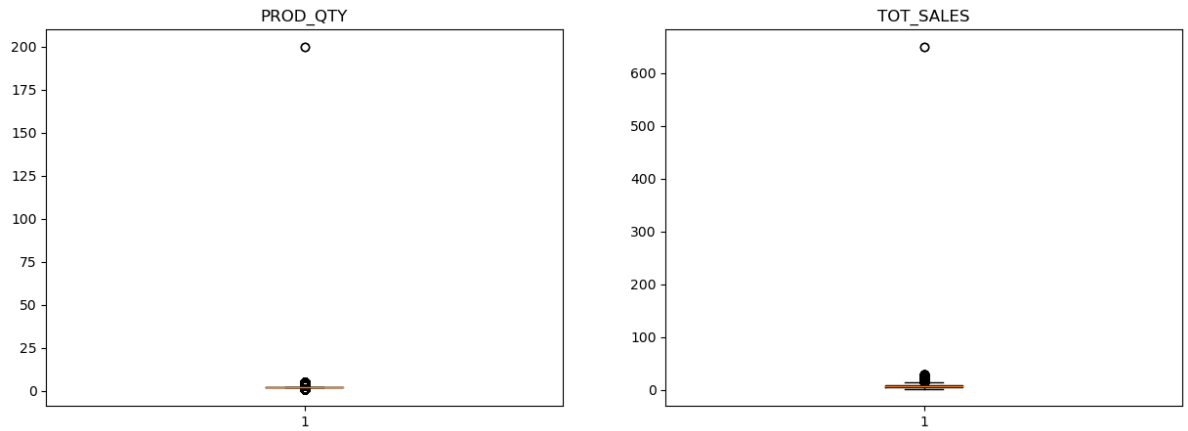
In [4]:

Out[4]:

```
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
```

In [5]:

```
#   Checking for any outliers in the pandas.DataFrame using a box plot of the

figure, axis=plt.subplots(1, 2, figsize=(15, 5))
axis[0].boxplot(transaction_data["PROD_QTY"])
axis[1].boxplot(transaction_data["TOT_SALES"])
axis[0].set_title("PROD_QTY")
axis[1].set_title("TOT_SALES")
plt.show()
```



In [8]:
```
#   Removing the outliers from the pandas.DataFrame.

transaction_data=transaction_data[transaction_data["PROD_QTY"]<100]
transaction_data=transaction_data[transaction_data["TOT_SALES"]<500]
transaction_data=transaction_data.reset_index(drop=True)    #   Resetting the
transaction_data
```

Out[8]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QT
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	43329	2	2373	974	69	Smiths Chip Thinly S/ Cream&Onion 175g	
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	
...	...	...	...	...	...	...	
264829	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
264830	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	
264831	43410	272	272379	270187	51	Doritos Mexicana 170g	
264832	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
264833	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	

264834 rows × 8 columns

In [9]:
```
date=transaction_data["DATE"].tolist()  #   Storing the DATE column as a list.

#   Converting the Microsoft Excel serial date format to the datetime format.

for i in range(len(date)):
    date[i]=xlrd.xldate_as_datetime(date[i], 0)

transaction_data["DATE"]=date   #   Replacing the DATE column with its corresp
transaction_data
```

Out[9]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROI
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/ Cream&Onion 175g	
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	
...	...	...	...	...	...	...	
264829	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
264830	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	
264831	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	
264832	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
264833	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	

264834 rows × 8 columns

Purchase Dataset

In [10]:
```
purchase_behaviour=pd.read_csv("QVI_purchase_behaviour.csv")    #   Reading th
purchase_behaviour
```

Out[10]:

	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
...	...	...	...
72632	2370651	MIDAGE SINGLES/COUPLES	Mainstream
72633	2370701	YOUNG FAMILIES	Mainstream
72634	2370751	YOUNG FAMILIES	Premium
72635	2370961	OLDER FAMILIES	Budget
72636	2373711	YOUNG SINGLES/COUPLES	Mainstream

72637 rows × 3 columns

In [11]:
```

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72637 entries, 0 to 72636
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   LYLTY_CARD_NBR    72637 non-null  int64 
 1   LIFESTAGE         72637 non-null  object
 2   PREMIUM_CUSTOMER  72637 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.7+ MB
```

In [12]:
```

```

Out[12]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [13]:
```
dataframe=pd.merge(transaction_data, purchase_behaviour, on="LYLTY_CARD_NBR")
dataframe
```

Out[13]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips 160g	
3	2019-03-09	1	1307	347	54	CCs Original 175g	
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
...	...	...	...	...	...	...	
264829	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
264830	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	
264831	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	
264832	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
264833	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	

264834 rows × 10 columns

In [14]:
```
#Distinct products that customers purchased
unique_products=list(dataframe["PROD_NAME"].unique())   #   Storing the distin
print("Total Distinct Products:", len(unique_products))
```

```
Total Distinct Products: 114
```

In [15]:
```
dataframe["PROD_NAME_CLEAN"]=dataframe["PROD_NAME"].str.replace("\d+g", "") #
dataframe["PROD_SIZE"]=dataframe["PROD_NAME"].str.extract("(\d+)")    #   Extr
dataframe["PROD_NAME"]=dataframe["PROD_NAME_CLEAN"] #   Assigning the PROD_NAM
dataframe=dataframe.drop("PROD_NAME_CLEAN", axis=1) #   Dropping the PROD_NAME_
dataframe["BRAND_NAME"]=dataframe["PROD_NAME"].str.split().str[0]    #   Extra
dataframe=dataframe.loc[:, ["DATE", "STORE_NBR", "LYLTY_CARD_NBR", "TXN_ID", "
dataframe
```

Out[15]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese	
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips	
3	2019-03-09	1	1307	347	54	CCs Original	
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken	
...	...	...	...	...	...	...	
264829	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream	
264830	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime	
264831	2018-11-06	272	272379	270187	51	Doritos Mexicana	
264832	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno	
264833	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime	

264834 rows × 12 columns

In [16]:

```
                                    ()
```

Out[16]:

```
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_SIZE           0
BRAND_NAME          0
PROD_QTY            0
TOT_SALES           0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [17]:

```
dataframe=dataframe.sort_values(by="DATE")  #   Sorting the pandas.DataFrame i
dataframe=dataframe.reset_index(drop=True)  #   Resetting the index of the pan
dataframe
```

Out[17]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_
0	2018-07-01	27	27181	24218	70	Tyrrells Crisps Lightly Salted	
1	2018-07-01	191	191099	192367	103	RRD Steak & Chimuchurri	
2	2018-07-01	257	257010	255769	24	Grain Waves Sweet Chilli	
3	2018-07-01	48	48129	43842	114	Kettle Sensations Siracha Lime	
4	2018-07-01	203	203013	202339	23	Cheezels Cheese	
...	...	...	...	...	...	...	
264829	2019-06-30	67	67129	64592	57	Old El Paso Salsa Dip Tomato Mild	
264830	2019-06-30	133	133121	136776	44	Thins Chips Light& Tangy	
264831	2019-06-30	257	257195	256935	83	WW D/Style Chip Sea Salt	
264832	2019-06-30	45	45057	40739	91	CCs Tasty Cheese	
264833	2019-06-30	199	199122	198088	42	Doritos Corn Chip Mexican Jalapeno	

264834 rows × 12 columns

In [18]:
```
pd.date_range(start="2018-07-01", end="2019-06-30").difference(dataframe["DATE
```

Out[18]:
```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

The missing value is on 2018-12-25 , it might be the premise was closed since it was christmas.

In [19]:
```
#   Filling missing values for date

dataframe=dataframe.append({"DATE": pd.to_datetime("2018-12-25"), "STORE_NBR":
dataframe=dataframe.sort_values(by="DATE")  #   Sorting the pandas.DataFrame i
dataframe=dataframe.reset_index(drop=True)  #   Resetting the index of the pan
dataframe
```

Out[19]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_
0	2018-07-01	27	27181	24218	70	Tyrrells Crisps Lightly Salted	
1	2018-07-01	180	180179	182143	46	Kettle Original	
2	2018-07-01	164	164069	164212	56	Cheezels Cheese Box	
3	2018-07-01	179	179216	180709	24	Grain Waves Sweet Chilli	
4	2018-07-01	18	18221	15451	80	Natural ChipCo Sea Salt & Vinegr	
...	...	...	...	...	...	...	
264830	2019-06-30	230	230022	232028	77	Doritos Corn Chips Nacho Cheese	
264831	2019-06-30	101	101071	100462	12	Natural Chip Co Tmato Hrb&Spce	
264832	2019-06-30	141	141226	142472	47	Doritos Corn Chips Original	
264833	2019-06-30	162	162118	162544	42	Doritos Corn Chip Mexican Jalapeno	
264834	2019-06-30	27	27288	24377	25	Pringles SourCream Onion	

264835 rows × 12 columns

In [20]:

Out[20]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_
129324	2018-12-25	0	0	0	0	None	

Data Analysis

Change in total sales over time

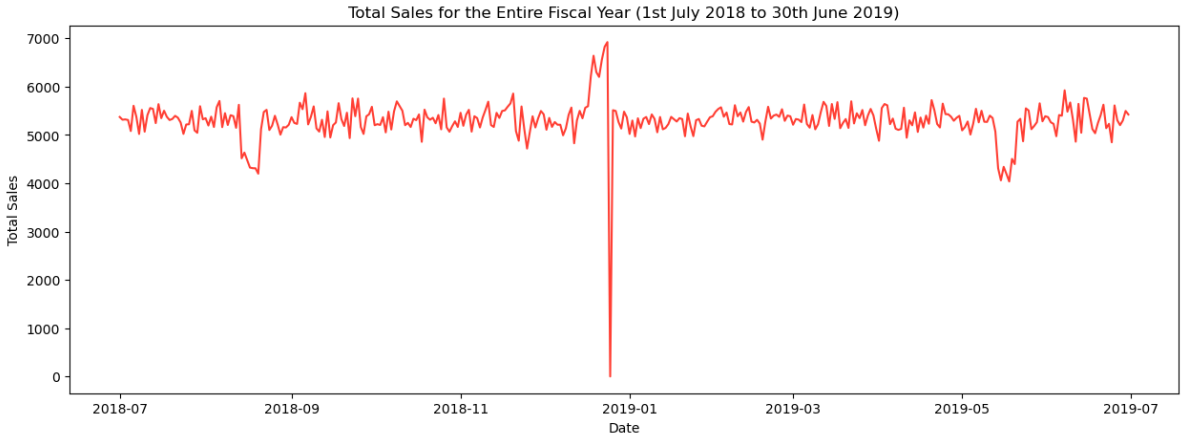
In [21]:
```
date_sales=dataframe.groupby("DATE")["TOT_SALES"].sum().reset_index()    #    Gr
date_sales
```

Out[21]:

	DATE	TOT_SALES
0	2018-07-01	5372.2
1	2018-07-02	5315.4
2	2018-07-03	5321.8
3	2018-07-04	5309.9
4	2018-07-05	5080.9
...	...	...
360	2019-06-26	5305.0
361	2019-06-27	5202.8
362	2019-06-28	5299.6
363	2019-06-29	5497.6
364	2019-06-30	5423.4

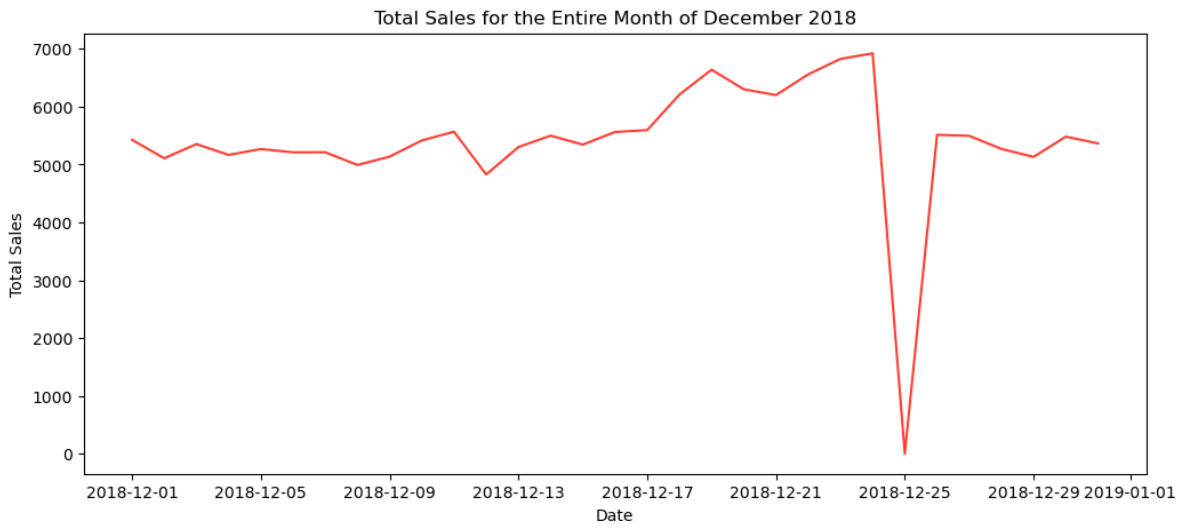
365 rows × 2 columns

In [22]:
```
#   Plotting a line graph of the total sales for each date over the entire rec

plt.figure(figsize=(15, 5))
plt.plot(date_sales["DATE"], date_sales["TOT_SALES"], color="#ff3f34")
plt.title("Total Sales for the Entire Fiscal Year (1st July 2018 to 30th June
plt.xlabel("Date")
plt.ylabel("Total Sales")
```



As we can see from the line graph, the sales drop to zero on a certain date, which is 25th December 2018 — Christmas Day, which we manually set to zero. However, the sales also reached an all-time high right before that, so we would need to analyse the transaction data for December 2018 to find out more about the sales.

In [23]:
```
#   Plotting a line graph of the total sales for each recorded date during Dec

plt.figure(figsize=(12, 5))
plt.plot(date_sales["DATE"][date_sales["DATE"].dt.month==12], date_sales["TOT_
plt.title("Total Sales for the Entire Month of December 2018")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.show()
```



As suspected, the sales reached an all-time high the day before Christmas Day, which makes

sense because people tend to purchase food items more when approaching holiday season. We can also see a consistent rise in the line graph between 21st December and 24th December, which means that these are the dates the store could target with promotions and discounts to increase the sales even more.

If the store does want to target these dates, it would be important to know which package sizes sell the most to create promotions and discounts around them.

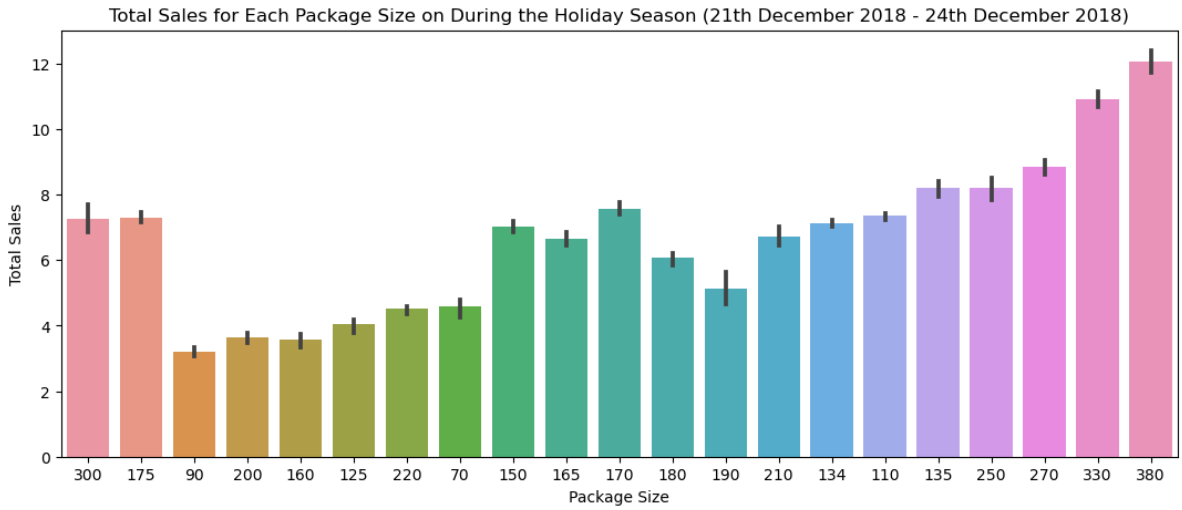
In [24]:
```
holiday_sales=dataframe[(dataframe["DATE"]>="2018-12-21") & (dataframe["DATE"]
holiday_sales=holiday_sales.sort_values(by="TOT_SALES") #   Sorting the pandas
holiday_sales=holiday_sales.reset_index(drop=True)  #   Resetting the index of
holiday_sales
```

Out[24]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_SI
0	2018-12-24	38	38005	34012	35	Woolworths Mild Salsa	3
1	2018-12-22	127	127448	130458	76	Woolworths Medium Salsa	3
2	2018-12-22	136	136114	138499	35	Woolworths Mild Salsa	3
3	2018-12-23	255	255077	254619	76	Woolworths Medium Salsa	3
4	2018-12-22	186	186218	188613	76	Woolworths Medium Salsa	3
...	...	...	...	...	...	...	
3608	2018-12-24	40	40152	36819	4	Dorito Corn Chp Supreme	3
3609	2018-12-24	217	217332	217772	4	Dorito Corn Chp Supreme	3
3610	2018-12-23	238	238351	243296	4	Dorito Corn Chp Supreme	3
3611	2018-12-24	3	3270	2289	4	Dorito Corn Chp Supreme	3
3612	2018-12-21	250	250213	252361	4	Dorito Corn Chp Supreme	3

3613 rows × 12 columns

In [25]:
```
#   Plotting a bar graph of the total sales for each package size between 21st

plt.figure(figsize=(13, 5))
sns.barplot(x="PROD_SIZE", y="TOT_SALES", data=holiday_sales)
plt.title("Total Sales for Each Package Size on During the Holiday Season (21t
plt.xlabel("Package Size")
plt.ylabel("Total Sales")
plt.show()
```



It seems like customers mostly purchased the 380 gramme package size (the largest one in the store) when approaching the holiday season.

Additionally, we can also find the brands that sold the most during the particular dates for brand-specific campaigns.

In [26]:
```
holiday_brands=holiday_sales.groupby("BRAND_NAME")["TOT_SALES"].sum().reset_in
holiday_brands=holiday_brands.reset_index(drop=True)    #   Resetting the inde
holiday_brands
```

Out[26]:

	BRAND_NAME	TOT_SALES
0	Kettle	4940.0
1	Doritos	2948.5
2	Smiths	2914.5
3	Pringles	2290.3
4	Thins	1343.1

We can see that KETTLE was the highest-selling brand during the holiday season, so it'd be wise to surround promotions and discounts around it to drive sales even more.

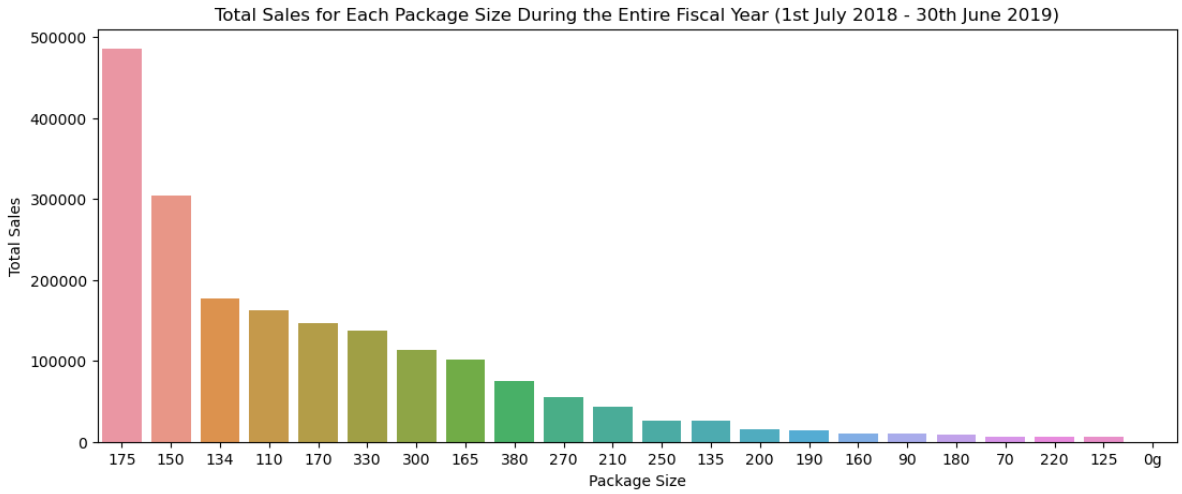
Let's see if our holiday season statistics match with the ones during the entire duration of the recorded sales.

In [27]:
```
package_sales=dataframe.groupby("PROD_SIZE")["TOT_SALES"].sum().reset_index().
package_sales=package_sales.reset_index(drop=True)    #   Resetting the index
package_sales
```

Out[27]:

	PROD_SIZE	TOT_SALES
0	175	485437.4
1	150	304288.5
2	134	177655.5
3	110	162765.4
4	170	146673.0
5	330	136794.3
6	300	113330.6
7	165	101360.6
8	380	75419.6
9	270	55425.4
10	210	43048.8
11	250	26096.7
12	135	26090.4
13	200	16007.5
14	190	14412.9
15	160	10647.6
16	90	9676.4
17	180	8568.4
18	70	6852.0
19	220	6831.0
20	125	5733.0
21	0g	0.0

In [28]:
```
#   Plotting a bar graph of the total sales for each package size during the e

plt.figure(figsize=(13, 5))
sns.barplot(x="PROD_SIZE", y="TOT_SALES", data=package_sales)
plt.title("Total Sales for Each Package Size During the Entire Fiscal Year (1s
plt.xlabel("Package Size")
plt.ylabel("Total Sales")
plt.show()
```



As we can see, the 175 gramme package size was the highest-selling one over the entire duration of the recorded sales, and even that by nearly 37% from the second highest-selling package size. Hence, it's clear that the 175 gramme package size is a customer favourite!

Likewise, we can also check for the highest-selling brands during the entire duration of the recorded sales.

In [29]:
```
brands_sales=dataframe.groupby("BRAND_NAME")["TOT_SALES"].sum().reset_index().
brands_sales=brands_sales.reset_index(drop=True)    #   Resetting the index of
brands_sales
```

Out[29]:

	BRAND_NAME	TOT_SALES
0	Kettle	390239.8
1	Smiths	210076.8
2	Doritos	201538.9
3	Pringles	177655.5
4	Old	90785.1

Just like the holiday season sales, KETTLE® remained the highest-selling brand during the entire duration of the recorded sales.

With the brand and product analysis done, we can move onto the customer analysis now. The first part would be analyse which sort of customers are the most loyal to the store, which would also be the ones that have the most purchases from it.

In [30]:

```
                                    ["LIFESTAGE"].value_counts()
```

Out[30]:

```
OLDER SINGLES/COUPLES     54479
RETIREES                  49763
OLDER FAMILIES            48594
YOUNG FAMILIES            43592
YOUNG SINGLES/COUPLES     36377
MIDAGE SINGLES/COUPLES    25110
NEW FAMILIES               6919
None                          1
Name: LIFESTAGE, dtype: int64
```

In [31]:

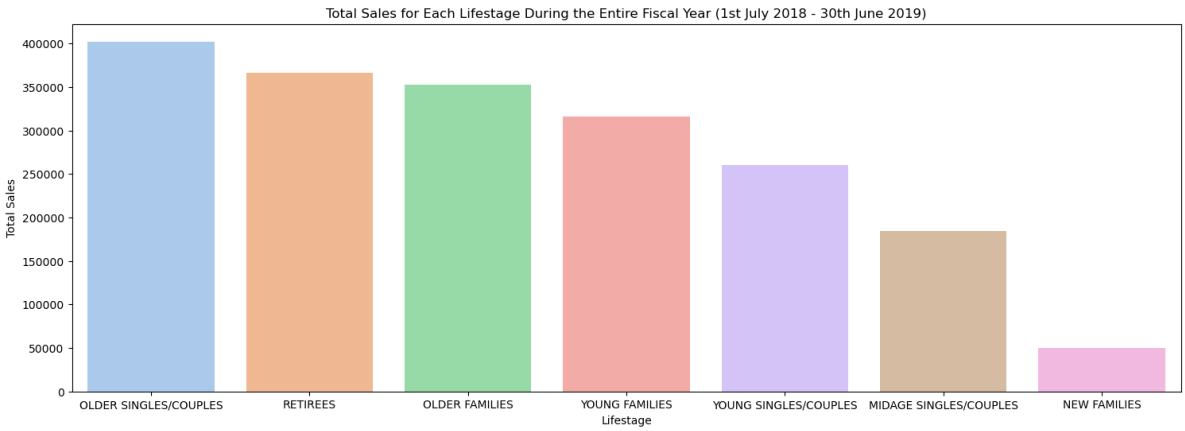
```
customer_sales=dataframe.groupby("LIFESTAGE")["TOT_SALES"].sum().reset_index()
customer_sales=customer_sales.reset_index(drop=True)    #   Resetting the inde
customer_sales
```

Out[31]:

	LIFESTAGE	TOT_SALES
0	OLDER SINGLES/COUPLES	402426.75
1	RETIREES	366470.90
2	OLDER FAMILIES	352467.20
3	YOUNG FAMILIES	316160.10
4	YOUNG SINGLES/COUPLES	260405.30
5	MIDAGE SINGLES/COUPLES	184751.30
6	NEW FAMILIES	50433.45

In [32]:

```
#   Plotting a bar graph of the total sales for each lifestage during the enti

plt.figure(figsize=(18, 6))
sns.barplot(x="LIFESTAGE", y="TOT_SALES", data=customer_sales, palette="pastel
plt.title("Total Sales for Each Lifestage During the Entire Fiscal Year (1st J
plt.xlabel("Lifestage")
plt.ylabel("Total Sales")
plt.show()
```



It seems like OLDER SINGLES/COUPLES are the most loyal customers of the store and NEW FAMILIES are the least. Interestingly, we can see a decreasing trend of purchases according to

age in the first half of the bar graph, with customers that are the most likely to spend the most time at home also having the most purchases, even though snack items wouldn't logically be associated with an age demographic.

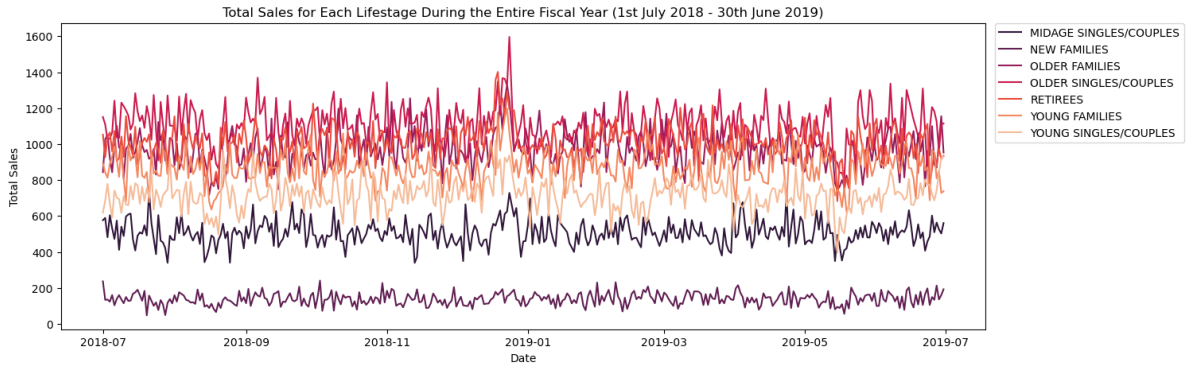
In [33]:
```
lifestage_sales=dataframe.groupby(["LIFESTAGE", "DATE"])["TOT_SALES"].sum().re
lifestage_sales=lifestage_sales[lifestage_sales["LIFESTAGE"]!="None"]   #   Re
lifestage_sales
```

Out[33]:

	LIFESTAGE	DATE	TOT_SALES
0	MIDAGE SINGLES/COUPLES	2018-07-01	576.8
1	MIDAGE SINGLES/COUPLES	2018-07-02	589.5
2	MIDAGE SINGLES/COUPLES	2018-07-03	482.2
3	MIDAGE SINGLES/COUPLES	2018-07-04	604.5
4	MIDAGE SINGLES/COUPLES	2018-07-05	531.6
...	...	...	...
2544	YOUNG SINGLES/COUPLES	2019-06-26	687.4
2545	YOUNG SINGLES/COUPLES	2019-06-27	743.4
2546	YOUNG SINGLES/COUPLES	2019-06-28	840.7
2547	YOUNG SINGLES/COUPLES	2019-06-29	924.5
2548	YOUNG SINGLES/COUPLES	2019-06-30	929.9

2548 rows × 3 columns

In [34]:
```
#   Plotting a multi-line graph of the total sales for each lifestage during t

plt.figure(figsize=(15, 5))
sns.lineplot(x="DATE", y="TOT_SALES", hue="LIFESTAGE", data=lifestage_sales, p
plt.title("Total Sales for Each Lifestage During the Entire Fiscal Year (1st J
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.legend(bbox_to_anchor=(1.01, 1), loc=2, borderaxespad=0.)
plt.show()
```



Like the holiday season statistics, we can see an increase in sales right before Christmas Day for all age demographics, except NEW FAMILIES, which remains consistent throughout the entire recorded duration. As new families are more inclined toward their careers and developing

their newly established home, it's unlikely for them to spend on snack items frequently.

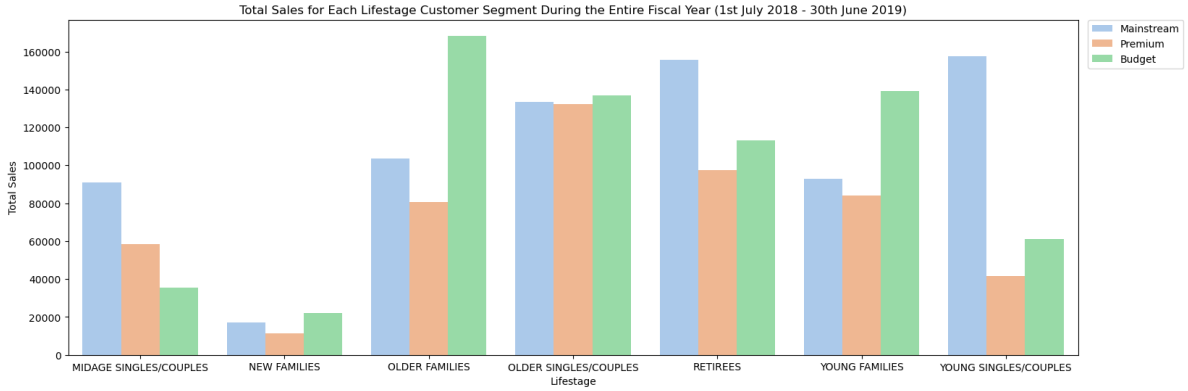
Let's see what sort of purchase behaviour each age demographic has!

In [35]:
```
lifestage_segment=dataframe.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["TOT_SA
lifestage_segment=lifestage_segment[lifestage_segment["LIFESTAGE"]!="None"] #
lifestage_segment=lifestage_segment.reset_index(drop=True)  #   Resetting the
lifestage_segment
```

Out[35]:

	LIFESTAGE	PREMIUM_CUSTOMER	TOT_SALES
0	MIDAGE SINGLES/COUPLES	Mainstream	90803.85
1	MIDAGE SINGLES/COUPLES	Premium	58432.65
2	MIDAGE SINGLES/COUPLES	Budget	35514.80
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	80658.40
9	OLDER SINGLES/COUPLES	Budget	136769.80
10	OLDER SINGLES/COUPLES	Mainstream	133393.80
11	OLDER SINGLES/COUPLES	Premium	132263.15
12	RETIREES	Mainstream	155677.05
13	RETIREES	Budget	113147.80
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	Mainstream	157621.60
19	YOUNG SINGLES/COUPLES	Budget	61141.60
20	YOUNG SINGLES/COUPLES	Premium	41642.10

In [36]:
```
#   Plotting a bar graph of the total sales for each lifestage and whether it

plt.figure(figsize=(18, 6))
sns.barplot(x="LIFESTAGE", y="TOT_SALES", hue="PREMIUM_CUSTOMER", data=lifesta
plt.title("Total Sales for Each Lifestage Customer Segment During the Entire F
plt.xlabel("Lifestage")
plt.ylabel("Total Sales")
plt.legend(bbox_to_anchor=(1.01, 1), loc=2, borderaxespad=0.)
plt.show()
```



We can see that MIDAGE SINGLES/COUPLES had the highest Mainstream and Premium purchases of all their purchases, while all the others had the least Premium purchases, which means that this would be the age demographic to target for payment plans and promotions to drive sales even more since they're more likely to pay more per packet of chips than the others.

Now that we know which age demographic the store should target to drive sales more, let's find which brand and package size each customer segment for MIDAGE SINGLES/COUPLES is the most inclined to.

In [37]:
```
lifestage_brands=dataframe.groupby(["LIFESTAGE", "BRAND_NAME", "PREMIUM_CUSTOM
lifestage_brands=lifestage_brands[lifestage_brands["LIFESTAGE"]!="None"]    #
lifestage_brands=lifestage_brands.reset_index(drop=True)    #   Resetting the
midage=lifestage_brands[lifestage_brands["LIFESTAGE"]=="MIDAGE SINGLES/COUPLES
midage
```

Out[37]:

	LIFESTAGE	BRAND_NAME	PREMIUM_CUSTOMER	PROD_SIZE	TOT_SALES
0	MIDAGE SINGLES/ COUPLES	Kettle	Mainstream	175	10557.0
1	MIDAGE SINGLES/ COUPLES	Kettle	Mainstream	150	8381.2
2	MIDAGE SINGLES/ COUPLES	Pringles	Mainstream	134	8177.0
3	MIDAGE SINGLES/ COUPLES	Kettle	Premium	175	5815.8
4	MIDAGE SINGLES/ COUPLES	Pringles	Premium	134	5538.9
...	...	...	...	...	...
133	MIDAGE SINGLES/ COUPLES	Snbts	Mainstream	90	120.7
134	MIDAGE SINGLES/ COUPLES	Cheezels	Budget	125	105.0
135	MIDAGE SINGLES/ COUPLES	Sunbites	Mainstream	90	103.7
136	MIDAGE SINGLES/ COUPLES	Sunbites	Budget	90	96.9
137	MIDAGE SINGLES/ COUPLES	Woolworths	Budget	190	81.0

138 rows × 5 columns

With this, we can see that MIDAGE SINGLES/COUPLES prefer KETTLE® and 175 gramme package size the most in both the Mainstream and Premium customer segment.

In [40]:
```
# Saving finalised dataframe to csv
```

Conclusion

Generally, sales gradually increase during the holiday season and are the highest the day before Christmas Day, but suddenly decrease right after, so this would be the ideal time for any promotional campaigns or discounts.

The 380 gramme package size, also the largest in the store, is the highest-selling package size during the holiday season with KETTLE® being the highest-selling brand.

KETTLE® is the also the highest-selling brand during the entire year, but the 175 gramme package size is the highest-selling package size, on average, with a difference of nearly 37%

from the second highest-selling package size.

OLDER SINGLES/COUPLES are the most loyal customers of the store and NEW FAMILIES are the least.

MIDAGE SINGLES/COUPLES had the highest Mainstream and Premium purchases of all their purchases, while all the others had the least Premium purchases, which means that they're more likely to pay more per packet of chips than the others.

MIDAGE SINGLES/COUPLES prefer KETTLE® and 175 gramme package size the most in both the Mainstream and Premium customer segment.

Oldest customers may be most valuable to the store and the recent ones may likely be at risk of churning.