Table of Contents

Section 1

Overview

This report explores the aggregated data regarding the prices of the products found in different supermarkets (described here as Banner) in different regions of the United States.

The final form of the dataset is obtained after joining together three separate files namely stores.json, prices.csv and auditors.csv. The dataset contains 6 different supermarket chains namely Walmart, Wegmans, Whole Foods Trader Joes, Safeway and Kroger and 5 different regions namely New York, Northern California, Kansas, Texas and Hawaii. The data is recorded over a period of 14 days from 16th October 2017 to 29th October 2017.

Data Preprocessing

Loading required libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.api as sm
from matplotlib.collections import PatchCollection
import matplotlib.patches as mpatches
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
plt.style.use('classic')
plt.style.use('ggplot')
plt.style.use('seaborn-whitegrid')
plt.style.use('seaborn')
```

Loading 'stores' json file

In [2]:
```
stores = pd.read_json("stores.json")
```

In [3]:
```
stores.head()
```

Out[3]:

	Banner	Region	Store ID
0	Walmart	Northern California	66999
1	Trader Joes	Northern California	4698
2	Safeway	Northern California	39482
3	Whole Foods	Northern California	34957
4	Walmart	New York	12837

Loading the 'prices' and 'auditors' csv files

In [4]:
```
auditors = pd.read_csv("auditors.csv")
prices = pd.read_csv("prices.csv")
```

In [5]:
```
auditors.head()
```

Out[5]:

	Auditor ID	First	Last	Region
0	234	Sue	Smith	Northern California
1	536	Bob	Smith	Northern California
2	98	Jack	Smith	New York
3	203	Jill	Smith	New York
4	304	Jerry	Johnson	Texas

In [6]:
```
prices.head()
```

Out[6]:

	Auditor ID	Date	Price	Store ID	UPC
0	234	10/18/17	24.95	66999	268588472
1	234	10/27/17	49.71	66999	475245085
2	234	10/20/17	25.75	66999	126967843
3	234	10/23/17	18.81	66999	708930835
4	234	10/23/17	33.32	66999	325885139

Creating an outer join on auditors and prices dataframes on the key 'Auditor ID'

In [7]:
```
auditors_prices_merged = pd.merge(auditors, prices, on = "Auditor ID", how = "outer")
```

In [8]:
```
auditors_prices_merged.head()
```

Out[8]:

	Auditor ID	First	Last	Region	Date	Price	Store ID	UPC
0	234	Sue	Smith	Northern California	10/18/17	24.95	66999	268588472
1	234	Sue	Smith	Northern California	10/27/17	49.71	66999	475245085
2	234	Sue	Smith	Northern California	10/20/17	25.75	66999	126967843
3	234	Sue	Smith	Northern California	10/23/17	18.81	66999	708930835
4	234	Sue	Smith	Northern California	10/23/17	33.32	66999	325885139

Creating an outer join on the previously merged dataframe 'auditors_prices_merged' and stores dataframe on the key 'Store ID'

In [9]:
```
final_df = pd.merge(auditors_prices_merged, stores, on = "Store ID", how = "outer")
```

Converting UPC and Auditor ID from float type to int type

In [10]:
```
final_df['UPC'] = final_df['UPC'].fillna(0.0).astype(int)
final_df['Auditor ID'] = final_df['Auditor ID'].fillna(0.0).astype(int)
```

In [11]:
```
final_df.loc[final_df['Region_y'] == 'Hawaii']
```

Out[11]:

	Auditor ID	First	Last	Region_x	Date	Price	Store ID	UPC	Banner	Region_y
12319	0	NaN	NaN	NaN	NaN	NaN	29342	0	Walmart	Hawaii
12320	0	NaN	NaN	NaN	NaN	NaN	11123	0	Trader Joes	Hawaii
12321	0	NaN	NaN	NaN	NaN	NaN	39284	0	Wegmans	Hawaii
12322	0	NaN	NaN	NaN	NaN	NaN	40582	0	Safeway	Hawaii
12323	0	NaN	NaN	NaN	NaN	NaN	59582	0	Whole Foods	Hawaii
12324	0	NaN	NaN	NaN	NaN	NaN	50692	0	Kroger	Hawaii

We see that the location 'Hawaii' more or less has all its values as NA values. Hence, I have decided to exclude these rows

In [12]:
```
final_df.loc[final_df['Banner'] == 'Kroger']
```

Out[12]:

	Auditor ID	First	Last	Region_x	Date	Price	Store ID	UPC	Banner	Region_y
12315	0	NaN	NaN	NaN	NaN	NaN	6746	0	Kroger	Northern California
12316	0	NaN	NaN	NaN	NaN	NaN	77955	0	Kroger	New York
12317	0	NaN	NaN	NaN	NaN	NaN	66356	0	Kroger	Texas
12318	0	NaN	NaN	NaN	NaN	NaN	99997	0	Kroger	Kansas
12324	0	NaN	NaN	NaN	NaN	NaN	50692	0	Kroger	Hawaii

We see that the banner 'Kroger' more or less has all its values as NA values. Hence, I have decided to exclude these rows

In [13]:
```
final_df = final_df[final_df.Banner != 'Kroger']
final_df = final_df[final_df.Region_y != 'Hawaii']
```

After joining all the dataframes together, I decided to drop the 'Region_x' column, the reason being the column 'Region_y' comes from the 'stores' dataframe and hence will provide a more accurate location of a store, as compared to the column 'Region_x' column which only gives the location of the store visited by the Auditor.

Also, for Question 1, I created a new dataframe called 'answer_df' and to obtain the desired output, I have dropped the additional columns of 'First', 'Last', 'Date'

In [14]:
```
answer_df = final_df.drop(columns=['Region_x', 'First', 'Last'])
final_df = final_df.drop(columns=['Region_x'])
```

In [15]:
```
final_df.head()
```

Out[15]:

	Auditor ID	First	Last	Date	Price	Store ID	UPC	Banner	Region_y
0	234	Sue	Smith	10/18/17	24.95	66999	268588472	Walmart	Northern California
1	234	Sue	Smith	10/27/17	49.71	66999	475245085	Walmart	Northern California
2	234	Sue	Smith	10/20/17	25.75	66999	126967843	Walmart	Northern California
3	234	Sue	Smith	10/23/17	18.81	66999	708930835	Walmart	Northern California
4	234	Sue	Smith	10/23/17	33.32	66999	325885139	Walmart	Northern California

Renaming the 'Region_x' column to 'Region'

In [16]:
```
final_df = final_df.rename(index=str, columns={"Region_y": "Region"})
answer_df = answer_df.rename(index=str, columns={"Region_y": "Region"})
answer_df_copy = answer_df
```

In [17]:
```
banner_region_groupby = final_df.groupby(['Banner', 'Region']).mean()
banner_region_groupby
```

Out[17]:

Banner	Region	Auditor ID	Price	Store ID	UPC
Safeway	**Kansas**	1326.0	30.694151	39485	5.017671e+08
	Northern California	98.0	35.292941	39482	5.211833e+08
	Texas	304.0	30.619833	29382	5.106342e+08
Trader Joes	**Kansas**	1326.0	29.201024	29384	5.098010e+08
	New York	203.0	30.521937	9487	4.952290e+08
	Northern California	536.0	34.376035	4698	5.201408e+08
	Texas	63.0	29.321849	40586	5.059505e+08
Walmart	**Kansas**	1326.0	27.646769	40593	5.109277e+08
	New York	203.0	28.427648	12837	5.135045e+08
	Northern California	234.0	32.854515	66999	4.932582e+08
	Texas	63.0	27.958877	50495	5.004522e+08
Wegmans	**Kansas**	713.0	30.276889	3948	5.054505e+08
	New York	98.0	30.885284	2938	5.111262e+08
	Texas	304.0	30.121931	98638	5.055896e+08
Whole Foods	**Kansas**	713.0	1.989452	39287	5.072501e+08
	New York	98.0	33.834301	50948	5.107961e+08
	Northern California	536.0	38.417573	34957	5.197973e+08
	Texas	304.0	32.823461	98736	5.061820e+08

Creating a pivot table to get the cross-tabulation of regional prices

In [18]:
```
answer_df = pd.pivot_table(answer_df, values='Price', index=['UPC', 'Banner'],
columns = ['Region']).reset_index()

answer_df.head()
```

Out[18]:

Region	UPC	Banner	Kansas	New York	Northern California	Texas
0	11873171	Safeway	NaN	NaN	6.09	5.19
1	11873171	Trader Joes	NaN	NaN	NaN	4.99
2	11873171	Walmart	NaN	NaN	5.53	4.75
3	11873171	Wegmans	NaN	5.19	NaN	5.09
4	11873171	Whole Foods	1.99	5.69	NaN	5.49

Rearranging the columns

In [19]:
```
answer_df = answer_df[['Banner', 'UPC', 'Northern California', 'New York', 'Kansas', 'Texas']]
answer_df.head()
```

Out[19]:

Region	Banner	UPC	Northern California	New York	Kansas	Texas
0	Safeway	11873171	6.09	NaN	NaN	5.19
1	Trader Joes	11873171	NaN	NaN	NaN	4.99
2	Walmart	11873171	5.53	NaN	NaN	4.75
3	Wegmans	11873171	NaN	5.19	NaN	5.09
4	Whole Foods	11873171	NaN	5.69	1.99	5.49

The NA values are introduced in the answer_df dataframes because a particular Banner is not present in the given location. If we observe the results of the banner_region_groupby dataframe, we can clearly see, for example, that the banner 'Safeway' is not present in the region 'Northern California'

In [20]:
```
final_df.loc[(final_df['Banner'] == 'Safeway') & (final_df['Region'] == 'New York')]
```

Out[20]:

	Auditor ID	First	Last	Date	Price	Store ID	UPC	Banner	Region

Checking the data types of the columns

In [21]:
```
answer_df.dtypes
```

Out[21]:
```
Region
Banner                 object
UPC                     int64
Northern California    float64
New York               float64
Kansas                 float64
Texas                  float64
dtype: object
```

Exporting the dataframe to an excel file

In [22]:
```
answer_df.to_excel("output.xlsx")
```

Section 2

In [23]:
```
final_df.head()
```

Out[23]:

	Auditor ID	First	Last	Date	Price	Store ID	UPC	Banner	Region
0	234	Sue	Smith	10/18/17	24.95	66999	268588472	Walmart	Northern California
1	234	Sue	Smith	10/27/17	49.71	66999	475245085	Walmart	Northern California
2	234	Sue	Smith	10/20/17	25.75	66999	126967843	Walmart	Northern California
3	234	Sue	Smith	10/23/17	18.81	66999	708930835	Walmart	Northern California
4	234	Sue	Smith	10/23/17	33.32	66999	325885139	Walmart	Northern California

Checking the data types of the columns

In [24]:
```
final_df.dtypes
```

Out[24]:
```
Auditor ID      int32
First          object
Last           object
Date           object
Price         float64
Store ID        int64
UPC             int32
Banner         object
Region         object
dtype: object
```

Converting the 'Date' column to date datatype

In [25]:
```
final_df['Date']= pd.to_datetime(final_df['Date'])
```

In [26]:
```
final_df.dtypes
```

Out[26]:
```
Auditor ID             int32
First                 object
Last                  object
Date          datetime64[ns]
Price                float64
Store ID               int64
UPC                    int32
Banner                object
Region                object
dtype: object
```

Checking for NA values in the dataset

In [27]:
```
final_df[final_df.isnull().any(axis=1)]
```

Out[27]:

	Auditor ID	First	Last	Date	Price	Store ID	UPC	Banner	Region
813	536	Bob	Smith	2017-10-19	48.69	60957	340260209	NaN	NaN
814	536	Bob	Smith	2017-10-29	55.79	60957	749133422	NaN	NaN
815	536	Bob	Smith	2017-10-25	5.99	60957	16999755	NaN	NaN
816	536	Bob	Smith	2017-10-28	11.09	60957	673299284	NaN	NaN
817	536	Bob	Smith	2017-10-27	32.79	60957	204071291	NaN	NaN
818	536	Bob	Smith	2017-10-20	11.79	60957	736014357	NaN	NaN
819	536	Bob	Smith	2017-10-27	15.19	60957	252888653	NaN	NaN
820	536	Bob	Smith	2017-10-21	43.79	60957	676939567	NaN	NaN
821	536	Bob	Smith	2017-10-23	44.59	60957	433536637	NaN	NaN
822	536	Bob	Smith	2017-10-18	60.29	60957	83141336	NaN	NaN
823	536	Bob	Smith	2017-10-20	65.19	60957	797375865	NaN	NaN
824	536	Bob	Smith	2017-10-27	47.99	60957	831639757	NaN	NaN
825	536	Bob	Smith	2017-10-23	20.59	60957	24764365	NaN	NaN
826	536	Bob	Smith	2017-10-25	46.99	60957	870309366	NaN	NaN
827	536	Bob	Smith	2017-10-18	30.29	60957	352180692	NaN	NaN
828	536	Bob	Smith	2017-10-21	23.69	60957	888713764	NaN	NaN
829	536	Bob	Smith	2017-10-17	5.89	60957	11873171	NaN	NaN
830	536	Bob	Smith	2017-10-20	2.19	60957	606513778	NaN	NaN
831	536	Bob	Smith	2017-10-29	5.19	60957	266112120	NaN	NaN
832	536	Bob	Smith	2017-10-25	41.49	60957	774342015	NaN	NaN
833	536	Bob	Smith	2017-10-23	18.49	60957	740153739	NaN	NaN
834	536	Bob	Smith	2017-10-29	46.59	60957	853687097	NaN	NaN
835	536	Bob	Smith	2017-10-25	63.79	60957	987411986	NaN	NaN
836	536	Bob	Smith	2017-10-28	21.99	60957	115104280	NaN	NaN
837	536	Bob	Smith	2017-10-26	39.79	60957	980564443	NaN	NaN
838	536	Bob	Smith	2017-10-18	36.79	60957	130358296	NaN	NaN
839	536	Bob	Smith	2017-10-18	41.69	60957	848280790	NaN	NaN
840	536	Bob	Smith	2017-10-26	62.29	60957	504139198	NaN	NaN
841	536	Bob	Smith	2017-10-26	51.89	60957	764458897	NaN	NaN
842	536	Bob	Smith	2017-10-25	4.29	60957	47174618	NaN	NaN
...	...	...	...	...	...	...	...	...	...
5433	203	Jill	Smith	2017-10-29	34.99	38472	599921977	NaN	NaN
5434	203	Jill	Smith	2017-10-28	59.19	38472	945973686	NaN	NaN
5435	203	Jill	Smith	2017-10-25	25.39	38472	374144563	NaN	NaN
5436	203	Jill	Smith	2017-10-22	60.89	38472	421428384	NaN	NaN

	Auditor ID	First	Last	Date	Price	Store ID	UPC	Banner	Region
5437	203	Jill	Smith	2017-10-25	14.19	38472	456975266	NaN	NaN
5438	203	Jill	Smith	2017-10-27	18.59	38472	16482322	NaN	NaN
5439	203	Jill	Smith	2017-10-17	55.19	38472	81313305	NaN	NaN
5440	203	Jill	Smith	2017-10-18	7.89	38472	672429356	NaN	NaN
5441	203	Jill	Smith	2017-10-19	33.69	38472	107865183	NaN	NaN
5442	203	Jill	Smith	2017-10-28	58.19	38472	357064541	NaN	NaN
5443	203	Jill	Smith	2017-10-23	55.59	38472	778006625	NaN	NaN
5444	203	Jill	Smith	2017-10-29	28.89	38472	45509167	NaN	NaN
5445	203	Jill	Smith	2017-10-25	27.69	38472	361415663	NaN	NaN
5446	203	Jill	Smith	2017-10-16	10.09	38472	673299284	NaN	NaN
5447	203	Jill	Smith	2017-10-19	19.69	38472	723906184	NaN	NaN
5448	203	Jill	Smith	2017-10-18	25.49	38472	576244412	NaN	NaN
5449	203	Jill	Smith	2017-10-29	18.39	38472	363663062	NaN	NaN
5450	203	Jill	Smith	2017-10-28	58.09	38472	917901226	NaN	NaN
5451	203	Jill	Smith	2017-10-29	59.69	38472	495847520	NaN	NaN
5452	203	Jill	Smith	2017-10-23	57.09	38472	514987291	NaN	NaN
5453	203	Jill	Smith	2017-10-20	45.09	38472	723419379	NaN	NaN
5454	203	Jill	Smith	2017-10-22	9.29	38472	526069074	NaN	NaN
5455	203	Jill	Smith	2017-10-19	14.99	38472	125829867	NaN	NaN
5456	203	Jill	Smith	2017-10-16	0.99	38472	278739906	NaN	NaN
5457	203	Jill	Smith	2017-10-23	39.39	38472	55404606	NaN	NaN
5458	203	Jill	Smith	2017-10-16	32.59	38472	915191199	NaN	NaN
5459	203	Jill	Smith	2017-10-20	8.89	38472	949435750	NaN	NaN
5460	203	Jill	Smith	2017-10-17	12.59	38472	561425157	NaN	NaN
5461	203	Jill	Smith	2017-10-18	47.39	38472	963229417	NaN	NaN
5462	203	Jill	Smith	2017-10-27	37.89	38472	376851918	NaN	NaN

804 rows × 9 columns

Classifying the missing values under the column 'Banner' as 'Undefined'

In [28]:
```
final_df['Banner'] = final_df['Banner'].fillna('Undefined')
```

In [29]:
```
final_df['Region'] = final_df['Region'].fillna('Unknown')
```

Rechecking for any remaining NA values

In [30]:
```
final_df[final_df.isnull().any(axis=1)]
```

Out[30]:

Auditor ID	First	Last	Date	Price	Store ID	UPC	Banner	Region

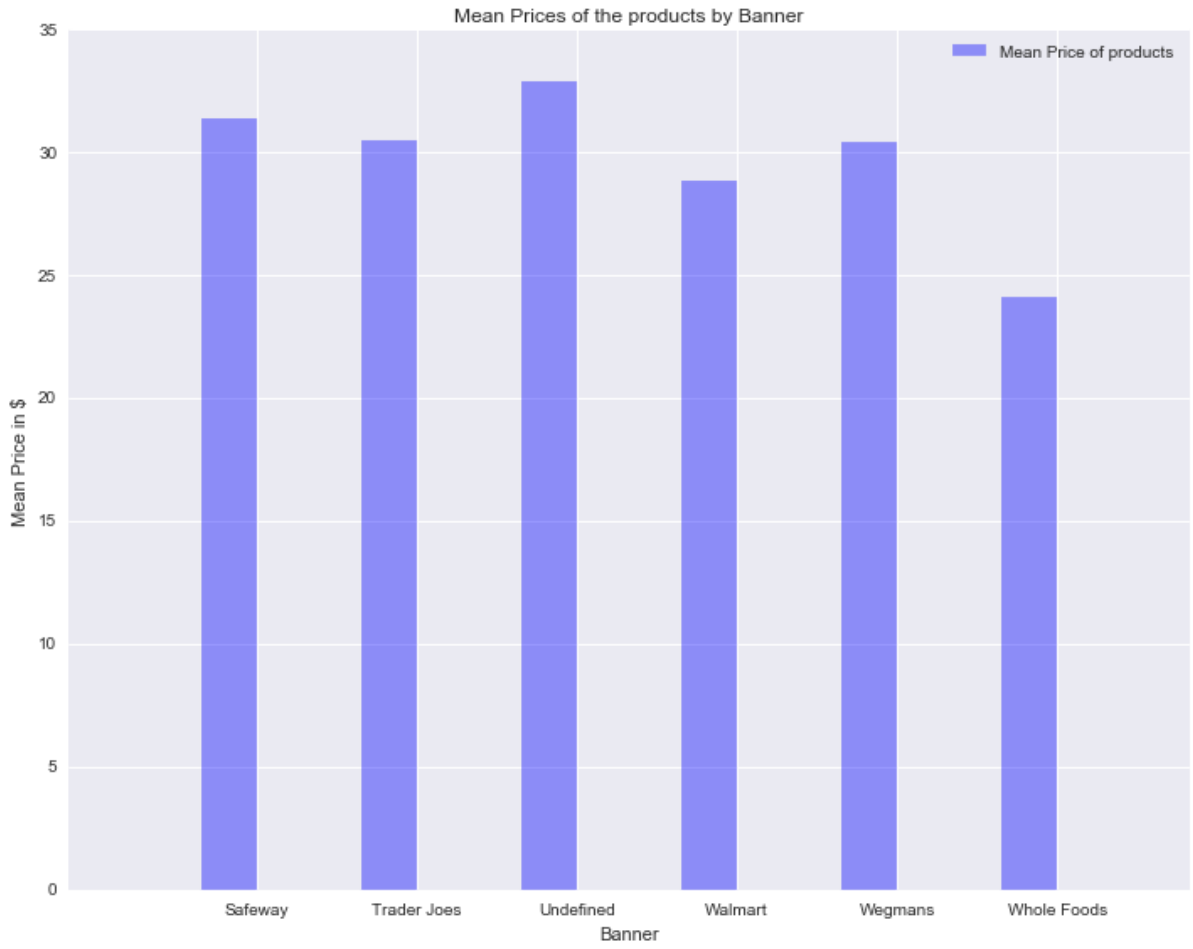
Graph of Price vs Banner

In [31]:
```
banner_groupby = final_df.groupby(['Banner']).mean()
banner_groupby = banner_groupby.drop(columns=['Auditor ID', 'Store ID', 'UPC'
])
banner_groupby
```

Out[31]:

	Price
Banner	
Safeway	31.379302
Trader Joes	30.504743
Undefined	32.920100
Walmart	28.877602
Wegmans	30.465411
Whole Foods	24.111734

In [33]:
```
n_groups = 6

mean_price = banner_groupby['Price']

fig, ax = plt.subplots()

index = np.arange(n_groups)
bar_width = 0.35

opacity = 0.4
error_config = {'ecolor': '0.3'}

rects1 = ax.bar(index, mean_price, bar_width,
                alpha=opacity, color='b',
                label='Mean Price of products')

ax.set_xlabel('Banner')
ax.set_ylabel('Mean Price in $')
ax.set_title('Mean Prices of the products by Banner')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(('Safeway', 'Trader Joes', 'Undefined', 'Walmart', 'Wegman
s', 'Whole Foods'))
ax.legend()

fig.tight_layout()

fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 10
fig_size[1] = 8
plt.rcParams["figure.figsize"] = fig_size

plt.show()
```



Count of Occurrences of each Banner in the dataset

In [34]:
```
final_df['Banner'].value_counts()
```

Out[34]:
```
Whole Foods    2802
Walmart        2435
Wegmans        2249
Trader Joes    2062
Safeway        1963
Undefined       804
Name: Banner, dtype: int64
```

Whole Foods is the cheapest supermarket according to this graph and it also has the most occurences. There is a high number of undefined values.

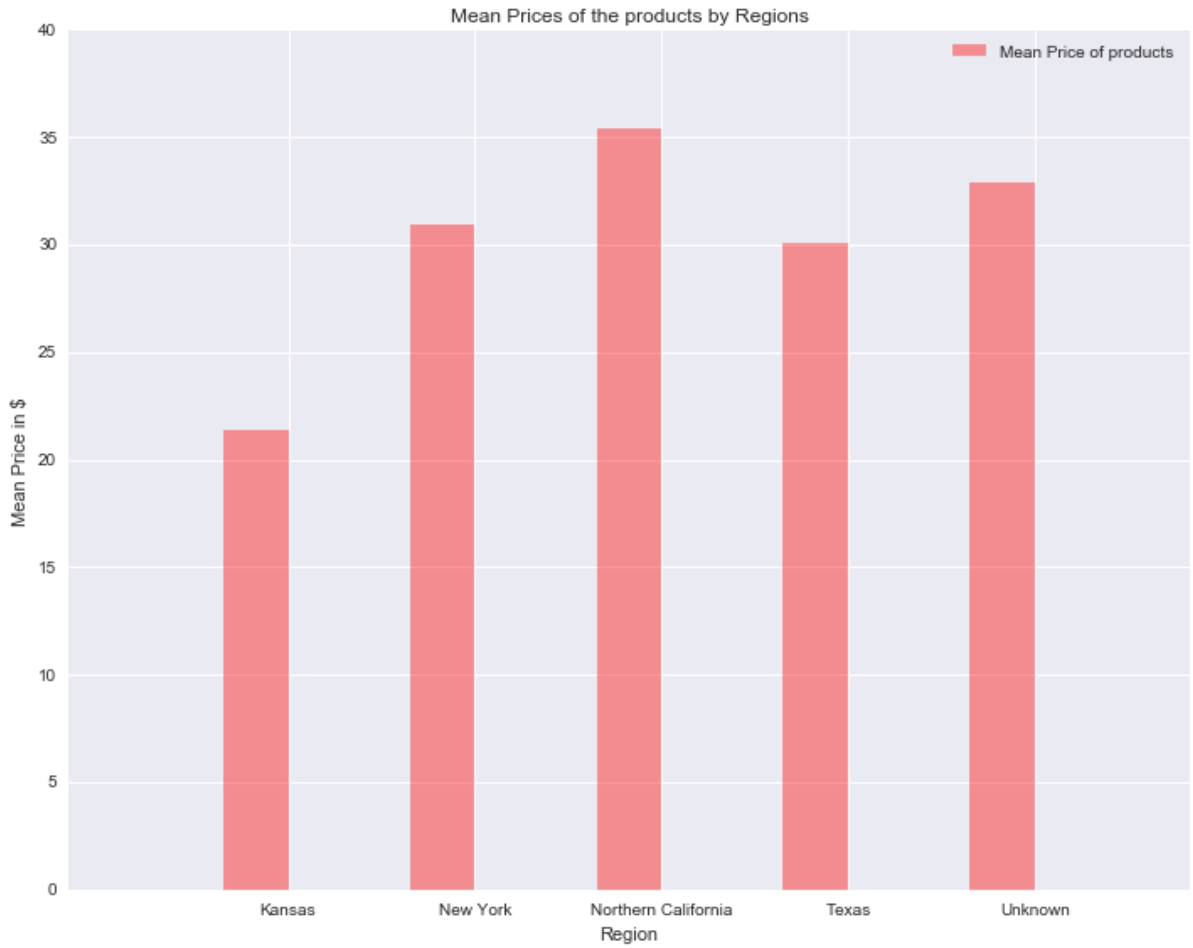
Graph of Price vs Regions

In [35]:
```
region_groupby = final_df.groupby(['Region']).mean()
region_groupby = region_groupby.drop(columns=['Auditor ID', 'Store ID', 'UPC'
])
region_groupby
```

Out[35]:

Region	Price
Kansas	21.390198
New York	30.971137
Northern California	35.437907
Texas	30.084550
Unknown	32.920100

In [36]:
```
n_groups = 5

mean_price = region_groupby['Price']

fig, ax = plt.subplots()

index = np.arange(n_groups)
bar_width = 0.35

opacity = 0.4
error_config = {'ecolor': '0.3'}

rects1 = ax.bar(index, mean_price, bar_width,
                alpha=opacity, color='r',
                label='Mean Price of products')

ax.set_xlabel('Region')
ax.set_ylabel('Mean Price in $')
ax.set_title('Mean Prices of the products by Regions')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(('Kansas', 'New York', 'Northern California', 'Texas', 'Unk
nown'))
ax.legend()

fig.tight_layout()

fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 10
fig_size[1] = 8
plt.rcParams["figure.figsize"] = fig_size

plt.show()
```



Count of Occurences of each region in the dataset

In [37]:
```
final_df['Region'].value_counts()
```

Out[37]:
```
Texas                  3765
Kansas                 3087
New York               3025
Northern California    1634
Unknown                 804
Name: Region, dtype: int64
```

Northern California is the most expensive state, but the number of values for Northern California in the dataset are less than half of that in Texas and nearly half of values in Texas and Kansas

Graph of Price vs Banner and Region

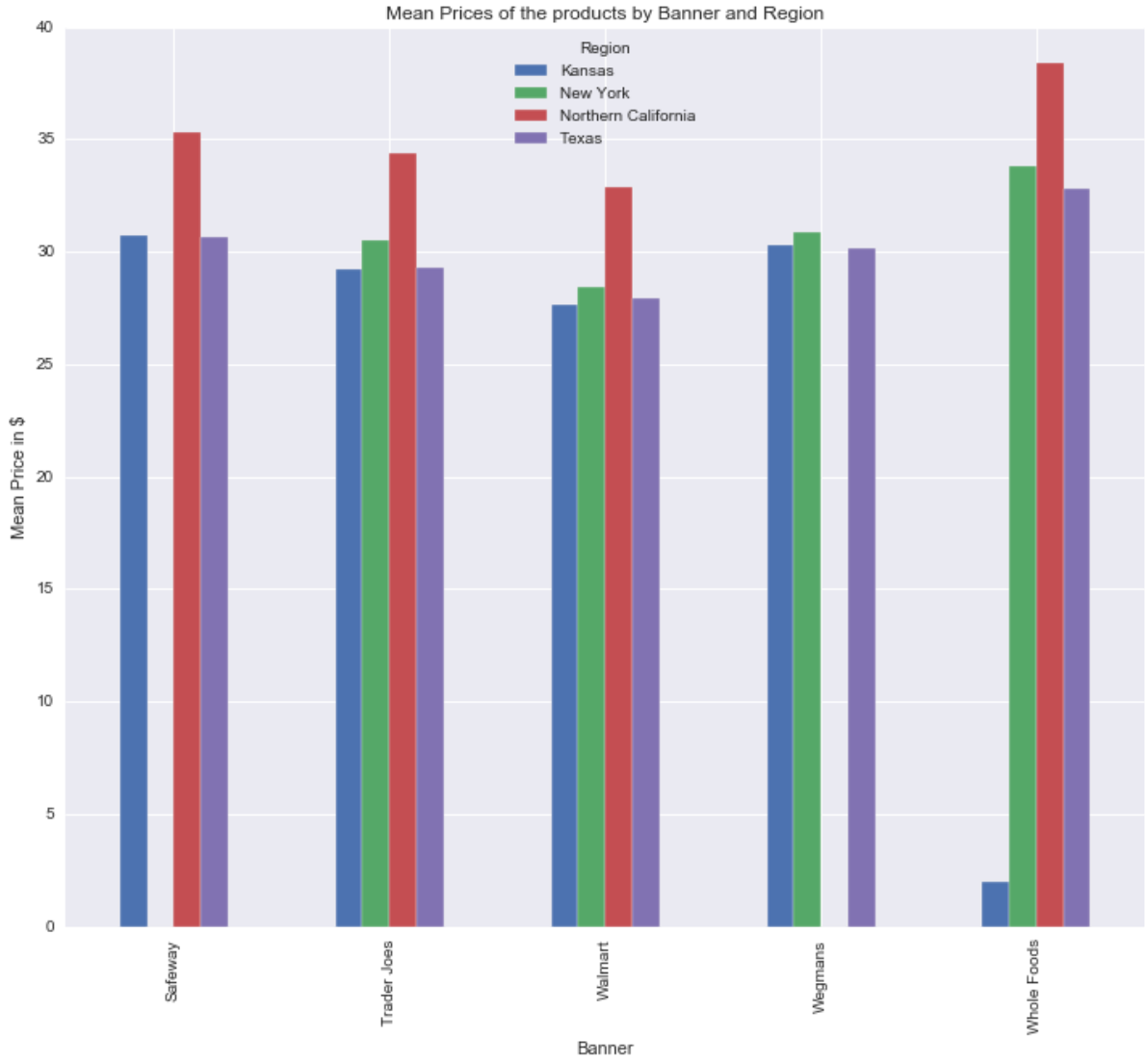
In [38]:
```
banner_region_groupby = banner_region_groupby.drop(columns=['Auditor ID', 'Store ID', 'UPC'])
banner_region_groupby
```

Out[38]:

		Price
Banner	**Region**	
Safeway	**Kansas**	30.694151
	Northern California	35.292941
	Texas	30.619833
Trader Joes	**Kansas**	29.201024
	New York	30.521937
	Northern California	34.376035
	Texas	29.321849
Walmart	**Kansas**	27.646769
	New York	28.427648
	Northern California	32.854515
	Texas	27.958877
Wegmans	**Kansas**	30.276889
	New York	30.885284
	Texas	30.121931
Whole Foods	**Kansas**	1.989452
	New York	33.834301
	Northern California	38.417573
	Texas	32.823461

In [39]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 12
fig_size[1] = 10
plt.rcParams["figure.figsize"] = fig_size

ax = banner_region_groupby.groupby(['Banner', 'Region'])['Price'].mean().unstack().plot(kind = "bar")
ax.set_xlabel("Banner")
ax.set_ylabel("Mean Price in $")
ax.set_title('Mean Prices of the products by Banner and Region')
```

Out[39]: Text(0.5, 1.0, 'Mean Prices of the products by Banner and Region')



Whole foods which had the cheapest mean price, turns out to be the most expensive supermarket chain in this graph. The scaling factors involved in the price calculations may have affected this increase.

Graph of Price vs Banner, Region and Date

In [40]:
```
store_state_perday_df = pd.pivot_table(answer_df_copy, values='Price', index=[
'Banner'], columns = ['Region', 'Date'])
store_state_perday_df
```

Out[40]:

Region	Kansas							
Date	10/16/17	10/17/17	10/18/17	10/19/17	10/20/17	10/21/17	10/22/17	10/23/17
Banner								
Safeway	30.485082	31.099836	29.465000	30.476275	30.662414	33.213729	32.208605	30.63717
Trader Joes	26.681304	28.752500	27.879286	34.132105	27.790000	34.134118	29.751290	29.3400(
Walmart	30.457949	26.186216	29.314444	24.540250	24.598293	25.154889	28.331915	32.74714
Wegmans	29.529394	36.446667	31.405385	28.997407	35.731379	26.601429	29.458966	30.3766(
Whole Foods	1.990000	1.990000	1.990000	1.990000	1.990000	1.990000	1.990000	1.9900(

5 rows × 56 columns

In [41]:
```
store_state_perday_df.columns
```

Out[41]:
```
MultiIndex(levels=[['Kansas', 'New York', 'Northern California', 'Texas'],
['10/16/17', '10/17/17', '10/18/17', '10/19/17', '10/20/17', '10/21/17', '10/
22/17', '10/23/17', '10/24/17', '10/25/17', '10/26/17', '10/27/17', '10/28/1
7', '10/29/17']],
           labels=[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 0, 1, 2, 3, 4, 5, 6, 7, 8,
9, 10, 11, 12, 13, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]],
           names=['Region', 'Date'])
```

In [42]:
```
store_state_perday_df_temp = store_state_perday_df
store_state_perday_df_temp.columns = store_state_perday_df_temp.columns.droplevel(0)
store_state_perday_df_temp.head()
```

Out[42]:

Date	10/16/17	10/17/17	10/18/17	10/19/17	10/20/17	10/21/17	10/22/17	10/23/17
Banner								
Safeway	30.485082	31.099836	29.465000	30.476275	30.662414	33.213729	32.208605	30.63717
Trader Joes	26.681304	28.752500	27.879286	34.132105	27.790000	34.134118	29.751290	29.34000
Walmart	30.457949	26.186216	29.314444	24.540250	24.598293	25.154889	28.331915	32.74714
Wegmans	29.529394	36.446667	31.405385	28.997407	35.731379	26.601429	29.458966	30.37666
Whole Foods	1.990000	1.990000	1.990000	1.990000	1.990000	1.990000	1.990000	1.99000

5 rows × 56 columns

In [43]:
```
store_state_perday_df_Kansas = store_state_perday_df_temp.iloc[:,0:14]
store_state_perday_df_New_York = store_state_perday_df_temp.iloc[:,14:28]
store_state_perday_df_Northern_California = store_state_perday_df_temp.iloc[:,28:42]
store_state_perday_df_Texas = store_state_perday_df_temp.iloc[:,42:]

store_state_perday_df_Kansas['Banner'] = ['Safeway' , 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Foods']
store_state_perday_df_New_York['Banner'] = ['Safeway' , 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Foods']
store_state_perday_df_Northern_California['Banner'] = ['Safeway' , 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Foods']
store_state_perday_df_Texas['Banner'] = ['Safeway' , 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Foods']
```

Graph of Price vs Banner, Region and Date for Kansas

In [44]:
```
store_state_perday_df_Kansas = store_state_perday_df_Kansas.transpose().reset_index()
store_state_perday_df_Kansas.head()
```

Out[44]:

Banner	Date	Safeway	Trader Joes	Walmart	Wegmans	Whole Foods
0	10/16/17	30.4851	26.6813	30.4579	29.5294	1.99
1	10/17/17	31.0998	28.7525	26.1862	36.4467	1.99
2	10/18/17	29.465	27.8793	29.3144	31.4054	1.99
3	10/19/17	30.4763	34.1321	24.5403	28.9974	1.99
4	10/20/17	30.6624	27.79	24.5983	35.7314	1.99

In [45]:
```
store_state_perday_df_Kansas.columns
```

Out[45]:
```
Index(['Date', 'Safeway', 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Foods'], dtype='object', name='Banner')
```

In [46]:
```
for col in store_state_perday_df_Kansas.columns:
    if (col != 'Date'):
        store_state_perday_df_Kansas[col] = pd.to_numeric(store_state_perday_df_Kansas[col], errors='coerce')
store_state_perday_df_Kansas.dtypes
```

Out[46]:
```
Banner
Date            object
Safeway        float64
Trader Joes    float64
Walmart        float64
Wegmans        float64
Whole Foods    float64
dtype: object
```

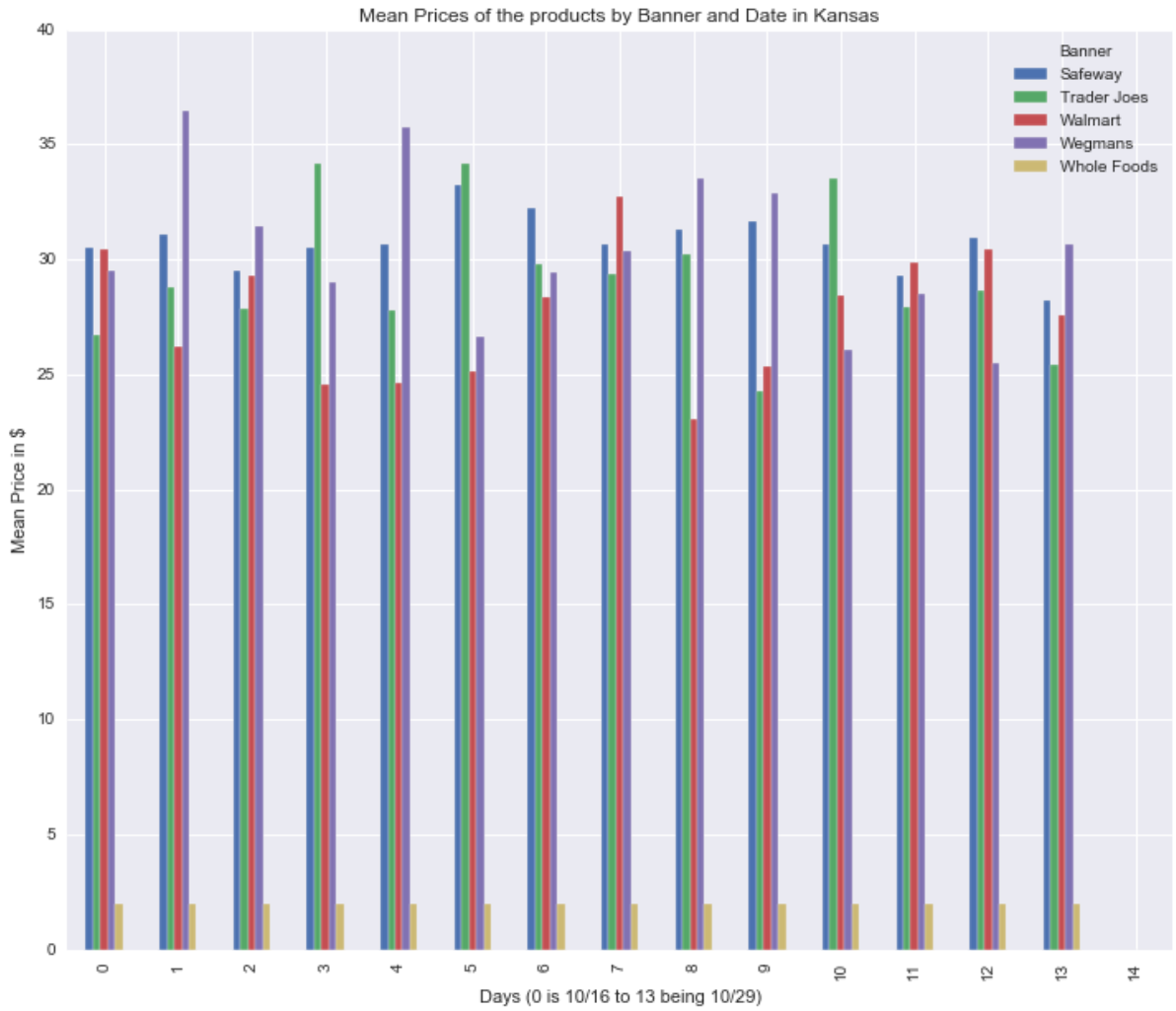
In [47]:
```
store_state_perday_df_Kansas.head()
```

Out[47]:

Banner	Date	Safeway	Trader Joes	Walmart	Wegmans	Whole Foods
0	10/16/17	30.485082	26.681304	30.457949	29.529394	1.99
1	10/17/17	31.099836	28.752500	26.186216	36.446667	1.99
2	10/18/17	29.465000	27.879286	29.314444	31.405385	1.99
3	10/19/17	30.476275	34.132105	24.540250	28.997407	1.99
4	10/20/17	30.662414	27.790000	24.598293	35.731379	1.99

In [48]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 12
fig_size[1] = 10
plt.rcParams["figure.figsize"] = fig_size

ax = store_state_perday_df_Kansas.plot(kind = "bar")
ax.set_xlabel("Days (0 is 10/16 to 13 being 10/29)")
ax.set_ylabel("Mean Price in $")
ax.set_title('Mean Prices of the products by Banner and Date in Kansas')
```

Out[48]: Text(0.5, 1.0, 'Mean Prices of the products by Banner and Date in Kansas')



The graph suggests that the data for Whole Foods has some anomalies. Trader Joe has always been on the expensive end while Walmart is on the cheaper end.

Graph of Price vs Banner, Region and Date for New York

In [49]:
```
store_state_perday_df_New_York = store_state_perday_df_New_York.transpose().reset_index()
store_state_perday_df_New_York.head()
```

Out[49]:

Banner	Date	Safeway	Trader Joes	Walmart	Wegmans	Whole Foods
0	10/16/17	NaN	26.8094	31.502	33.2407	27.8192
1	10/17/17	NaN	36.4757	27.316	31.4883	36.5313
2	10/18/17	NaN	31.3823	28.5716	30.0504	32.8985
3	10/19/17	NaN	30.73	28.8984	33.2305	37.7304
4	10/20/17	NaN	26.4689	28.1167	33.59	33.8802

In [50]:
```
store_state_perday_df_New_York.columns
```

Out[50]:
```
Index(['Date', 'Safeway', 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Foods'], dtype='object', name='Banner')
```

In [51]:
```
for col in store_state_perday_df_New_York.columns:
    if (col != 'Date'):
        store_state_perday_df_New_York[col] = pd.to_numeric(store_state_perday_df_New_York[col], errors='coerce')
store_state_perday_df_New_York.dtypes
```

Out[51]:
```
Banner
Date           object
Safeway       float64
Trader Joes   float64
Walmart       float64
Wegmans       float64
Whole Foods   float64
dtype: object
```

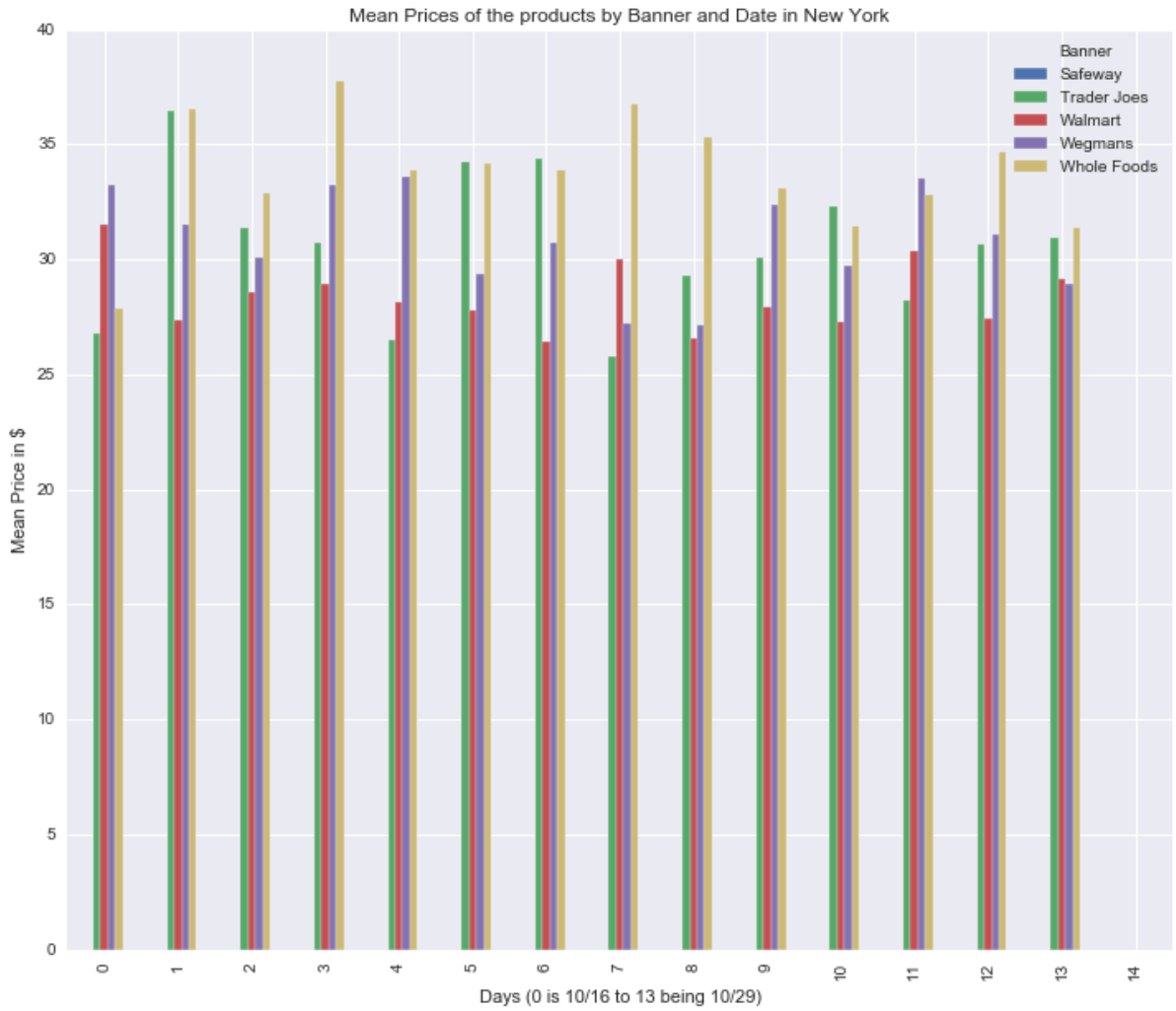
In [52]:
```
store_state_perday_df_New_York.head()
```

Out[52]:

Banner	Date	Safeway	Trader Joes	Walmart	Wegmans	Whole Foods
0	10/16/17	NaN	26.809355	31.501967	33.240685	27.819167
1	10/17/17	NaN	36.475714	27.315965	31.488333	36.531270
2	10/18/17	NaN	31.382308	28.571562	30.050377	32.898475
3	10/19/17	NaN	30.730000	28.898393	33.230506	37.730351
4	10/20/17	NaN	26.468947	28.116667	33.590000	33.880164

In [53]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 12
fig_size[1] = 10
plt.rcParams["figure.figsize"] = fig_size

ax = store_state_perday_df_New_York.plot(kind = "bar")
ax.set_xlabel("Days (0 is 10/16 to 13 being 10/29)")
ax.set_ylabel("Mean Price in $")
ax.set_title('Mean Prices of the products by Banner and Date in New York')
```

Out[53]: Text(0.5, 1.0, 'Mean Prices of the products by Banner and Date in New York')



Whole Foods is the most expensive of all. Wegmans and Trader Joe are approximately the similar priced supermarkets.

Graph of Price vs Banner, Region and Date for Northern California

In [54]:
```
store_state_perday_df_Northern_California = store_state_perday_df_Northern_Cal
ifornia.transpose().reset_index()
store_state_perday_df_Northern_California.head()
```

Out[54]:

Banner	Date	Safeway	Trader Joes	Walmart	Wegmans	Whole Foods
0	10/16/17	31.7627	28.115	33.1339	NaN	38.8724
1	10/17/17	39.09	35.675	31.4156	NaN	40.4438
2	10/18/17	33.1061	34.5307	31.4621	NaN	39.0442
3	10/19/17	35.8712	34.1355	34.7579	NaN	32.69
4	10/20/17	34.7639	35.859	36.9828	NaN	36.0726

In [55]:
```
store_state_perday_df_Northern_California.columns
```

Out[55]:
```
Index(['Date', 'Safeway', 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Food
s'], dtype='object', name='Banner')
```

In [56]:
```
for col in store_state_perday_df_Northern_California.columns:
    if (col != 'Date'):
        store_state_perday_df_Northern_California[col] = pd.to_numeric(store_s
tate_perday_df_Northern_California[col], errors='coerce')
store_state_perday_df_Northern_California.dtypes
```

Out[56]:
```
Banner
Date            object
Safeway        float64
Trader Joes    float64
Walmart        float64
Wegmans        float64
Whole Foods    float64
dtype: object
```

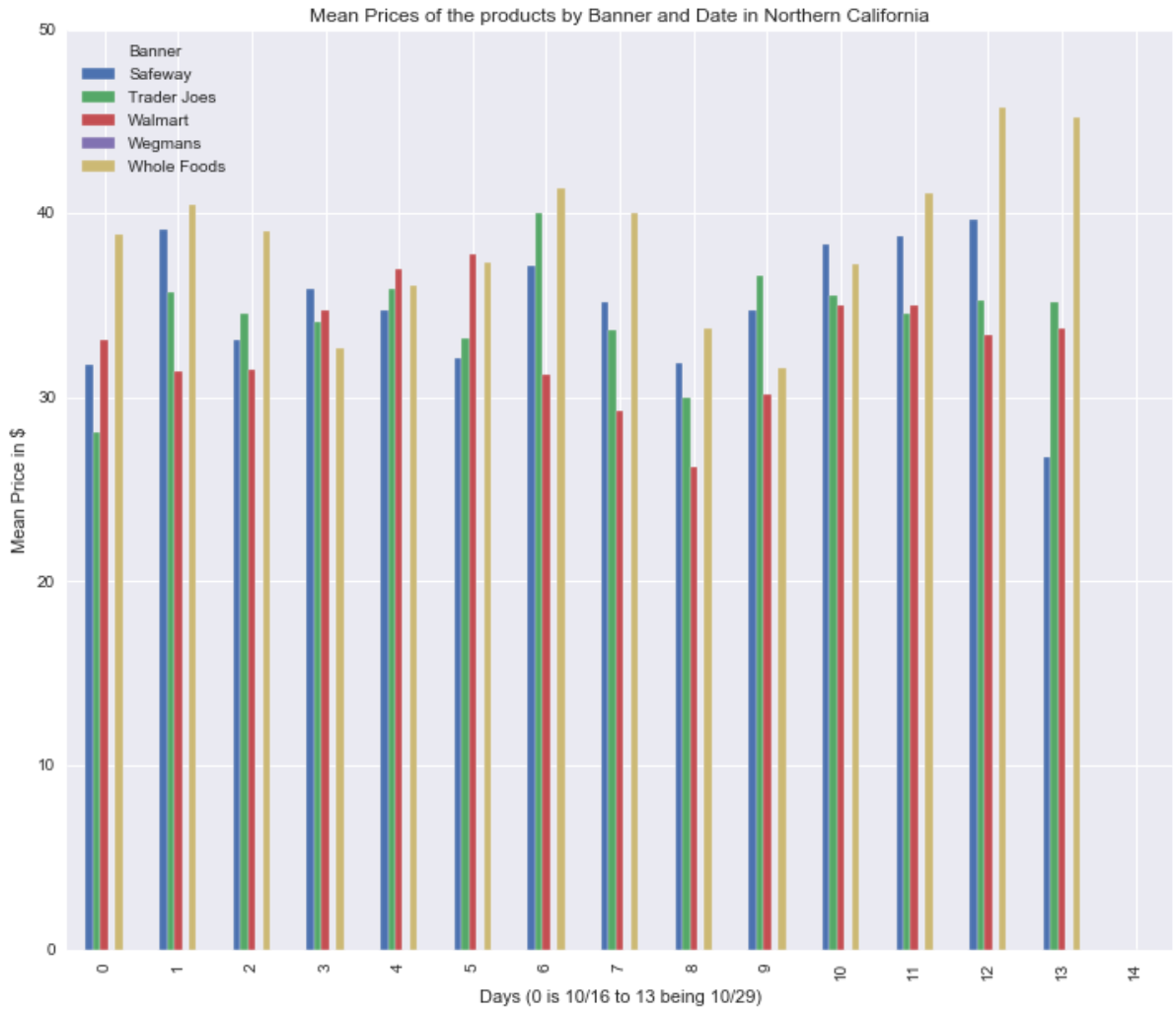
In [57]:
```
store_state_perday_df_Northern_California.head()
```

Out[57]:

Banner	Date	Safeway	Trader Joes	Walmart	Wegmans	Whole Foods
0	10/16/17	31.762727	28.115000	33.133939	NaN	38.872353
1	10/17/17	39.090000	35.675000	31.415600	NaN	40.443846
2	10/18/17	33.106129	34.530741	31.462059	NaN	39.044167
3	10/19/17	35.871250	34.135455	34.757857	NaN	32.690000
4	10/20/17	34.763913	35.858966	36.982813	NaN	36.072609

In [58]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 12
fig_size[1] = 10
plt.rcParams["figure.figsize"] = fig_size

ax = store_state_perday_df_Northern_California.plot(kind = "bar")
ax.set_xlabel("Days (0 is 10/16 to 13 being 10/29)")
ax.set_ylabel("Mean Price in $")
ax.set_title('Mean Prices of the products by Banner and Date in Northern Califo
rnia')
```

Out[58]:
```
Text(0.5, 1.0, 'Mean Prices of the products by Banner and Date in Northern Ca
lifornia')
```



Safeway appears to be the cheapest while Walmart and Trader Joe's prices are comparable over 14 days.

Graph of Price vs Banner, Region and Date for Texas

In [59]:
```
store_state_perday_df_Texas = store_state_perday_df_Texas.transpose().reset_index()
store_state_perday_df_Texas.head()
```

Out[59]:

Banner	Date	Safeway	Trader Joes	Walmart	Wegmans	Whole Foods
0	10/16/17	28.3466	31.5797	23.9248	30.6997	32.321
1	10/17/17	30.9824	27.2004	23.9733	31.1567	27.4217
2	10/18/17	29.7918	31.2957	29.0126	29.9064	30.89
3	10/19/17	31.9621	30.6268	30.4021	31.1829	36.4423
4	10/20/17	29.6278	26.3485	31.4683	27.6915	35.5997

In [60]:
```
store_state_perday_df_Texas.columns
```

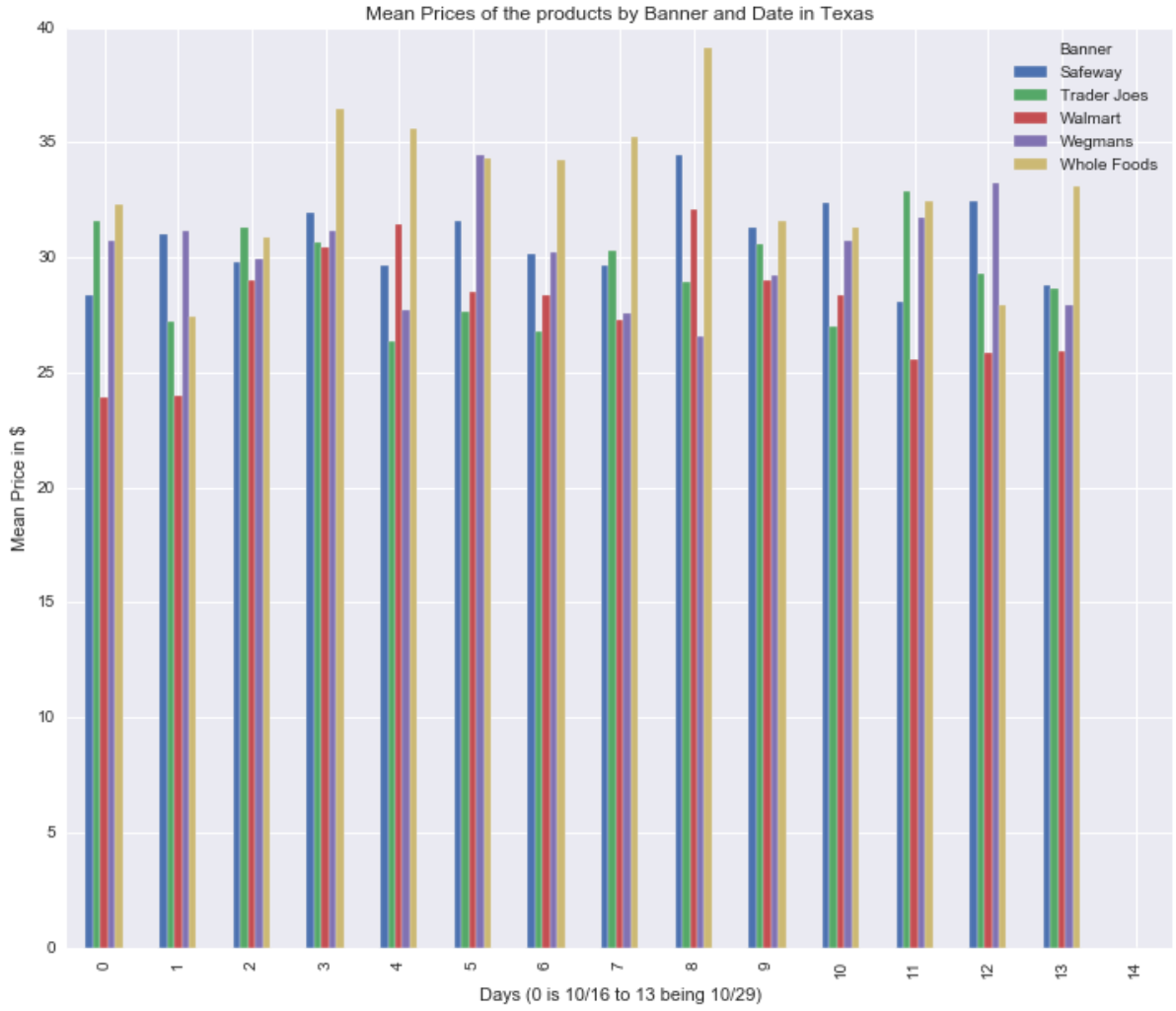
Out[60]:
```
Index(['Date', 'Safeway', 'Trader Joes', 'Walmart', 'Wegmans', 'Whole Foods'], dtype='object', name='Banner')
```

In [61]:
```
for col in store_state_perday_df_Texas.columns:
    if (col != 'Date'):
        store_state_perday_df_Texas[col] = pd.to_numeric(store_state_perday_df_Texas[col], errors='coerce')
store_state_perday_df_Texas.dtypes
```

Out[61]:
```
Banner
Date            object
Safeway        float64
Trader Joes    float64
Walmart        float64
Wegmans        float64
Whole Foods    float64
dtype: object
```

In [62]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 12
fig_size[1] = 10
plt.rcParams["figure.figsize"] = fig_size

ax = store_state_perday_df_Texas.plot(kind = "bar")
ax.set_xlabel("Days (0 is 10/16 to 13 being 10/29)")
ax.set_ylabel("Mean Price in $")
ax.set_title('Mean Prices of the products by Banner and Date in Texas')
```

Out[62]:
```
Text(0.5, 1.0, 'Mean Prices of the products by Banner and Date in Texas')
```



From all the four graphs we can conlcude that the prices fell by the end of the 14th day. In 31st October 2017 was Halloween's and that could be one of the reasons so as to attract customers on the weekend just before it.

Conclusions

1) More information on store scaling factor and regional scaling factor can give us more insights on a few things observed in the graphs above. For example, the mean price of Whole Foods is the lowest of all, but when grouped with region, Whole Foods is the most expensive retail store. The reason behind this could be clearer given the above mentioned scaling factors.

2) With more details about a particular product, we can be able derive more insights from the data, to see if a particular product category is more expensive in other states even if other categories in that state are cheaper and so on.