In [35]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all

import os
for dirname, _, filenames in os.walk('../CODES'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets pr
# You can also write temporary files to /kaggle/temp/, but they won't be saved outs
```

```
../CODES\1.1. MicrosoftMalwareDetection.ipynb
../CODES\21.1. PersonalizedCancerDiagnosis.ipynb
../CODES\AirlinesTwitterAnalysisProject1.ipynb
../CODES\Cust_segmentation_online_retail.ipynb
../CODES\features.csv
../CODES\File 2.ipynb
../CODES\Image background remover and periodictable.ipynb
../CODES\kc_house_data.csv
../CODES\NYC Final.ipynb
../CODES\sales-advanced-analysis-and-prediction.ipynb
../CODES\stores.csv
../CODES\test.csv
../CODES\test.txt
../CODES\train.csv
../CODES\Untitled.ipynb
../CODES\Untitled1.ipynb
../CODES\w.jpeg
../CODES\w4.jpeg
../CODES\win.jpg
../CODES\win1.png
../CODES\win2.png
../CODES\win24.png
../CODES\.ipynb_checkpoints\1.1. MicrosoftMalwareDetection-checkpoint.ipynb
../CODES\.ipynb_checkpoints\21.1. PersonalizedCancerDiagnosis-checkpoint.ipynb
../CODES\.ipynb_checkpoints\AirlinesTwitterAnalysisProject1-checkpoint.ipynb
../CODES\.ipynb_checkpoints\File 2-checkpoint.ipynb
../CODES\.ipynb_checkpoints\Image background remover and periodictable-checkpoint.
ipynb
../CODES\.ipynb_checkpoints\NYC Final-checkpoint.ipynb
../CODES\.ipynb_checkpoints\sales-advanced-analysis-and-prediction-checkpoint.ipyn
b
../CODES\.ipynb_checkpoints\Untitled-checkpoint.ipynb
../CODES\.ipynb_checkpoints\Untitled1-checkpoint.ipynb
../CODES\Predictive analytics\sales-advanced-analysis-and-prediction.ipynb
```

In [5]:
```
# Input data files are available in the "../input/" directory.
# First let us load the datasets into different Dataframes
def load_data(datapath):
    data = pd.read_csv(datapath)
   # Dimensions
    print('Shape:', data.shape)
    # Set of features we have are: date, store, and item
    display(data.sample(10))
    return data
```

```
traindf=load_data('train.csv')
testdf=load_data('test.csv')
featuresdf=load_data('features.csv')
storesdf=load_data('stores.csv')
```

```
Shape: (421570, 5)
```

	Store	Dept	Date	Weekly_Sales	IsHoliday
244265	25	67	2011-03-18	5778.26	False
127043	13	92	2010-03-26	140088.85	False
99608	11	13	2011-02-11	42844.65	True
347528	37	8	2010-08-06	14633.94	False
164755	17	67	2012-08-10	6436.01	False
329654	34	91	2011-07-22	35670.52	False
158484	17	8	2010-06-18	20617.58	False
342029	36	14	2012-01-27	1419.92	False
235885	24	90	2012-04-06	86581.06	False
297761	31	30	2011-06-17	3842.20	False

```
Shape: (115064, 4)
```

	Store	Dept	Date	IsHoliday
53108	20	67	2013-03-22	False
58424	22	71	2012-11-09	False
71077	27	30	2013-01-18	False
55354	21	42	2013-02-22	False
67884	26	17	2013-04-12	False
81010	31	27	2013-03-08	False
107910	42	79	2013-05-31	False
23544	9	81	2013-04-05	False
66766	25	80	2013-07-26	False
106290	41	93	2013-01-18	False

```
Shape: (8190, 12)
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
5953	33	2012-07-27	93.95	3.769	24.71	NaN	NaN	NaN
8050	45	2010-11-26	46.15	3.039	NaN	NaN	NaN	NaN
7183	40	2011-09-23	54.09	3.758	NaN	NaN	NaN	NaN
6833	38	2011-12-30	44.64	3.428	353.07	1926.94	NaN	25.20
2195	13	2010-04-23	55.66	2.936	NaN	NaN	NaN	NaN
2349	13	2013-04-05	53.84	3.547	25061.60	16527.33	715.36	3092.08
2601	15	2011-02-11	21.64	3.416	NaN	NaN	NaN	NaN
2387	14	2010-07-02	76.61	2.815	NaN	NaN	NaN	NaN
5627	31	2013-04-19	67.52	3.451	8848.14	792.60	84.09	937.55
6371	36	2010-02-12	46.11	2.539	NaN	NaN	NaN	NaN

Shape: (45, 3)

	Store	Type	Size
24	25	B	128107
0	1	A	151315
27	28	A	206302
17	18	B	120653
40	41	A	196321
18	19	A	203819
25	26	A	152513
16	17	B	93188
44	45	B	118221
8	9	B	125833

DATA PREPARATION & ANALYSIS

Merging the features and training data to get cumulative insights from overall

In [6]:
```
traindf1=traindf.merge(featuresdf,how='left',indicator=True).merge(storesdf,how='l
```

In [7]:
```
traindf1
```

Out[7]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Mark
0	1	1	2010-02-05	24924.50	False	42.31	2.572	NaN	
1	1	1	2010-02-12	46039.49	True	38.51	2.548	NaN	
2	1	1	2010-02-19	41595.55	False	39.93	2.514	NaN	
3	1	1	2010-02-26	19403.54	False	46.63	2.561	NaN	
4	1	1	2010-03-05	21827.90	False	46.50	2.625	NaN	
...	...	...	...	...	...	...	...	...	
421565	45	98	2012-09-28	508.37	False	64.88	3.997	4556.61	
421566	45	98	2012-10-05	628.10	False	64.89	3.985	5046.74	
421567	45	98	2012-10-12	1061.02	False	54.47	4.000	1956.28	
421568	45	98	2012-10-19	760.01	False	56.47	3.969	2004.02	
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	4018.91	

421570 rows × 17 columns

Markdown values are typically a promotional factors and it contains 58% null values,So here Im avoiding it to perform neat analysis.

In [8]:
```
traindf2=traindf1.drop(['MarkDown1','MarkDown2','MarkDown3','MarkDown4','MarkDown5
```

In [9]:
```
traindf2.isna().sum()
```

Out[9]:
```
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
_merge          0
Type            0
Size            0
dtype: int64
```

Let's check any outliers on sales values

In [10]:
```
traindf2.loc[traindf2['Weekly_Sales']<=0] #outliers
```

Out[10]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemp
846	1	6	2012-08-10	-139.65	False	85.05	3.494	221.958433	
2384	1	18	2012-05-04	-1.27	False	75.55	3.749	221.671800	
6048	1	47	2010-02-19	-863.00	False	39.93	2.514	211.289143	
6049	1	47	2010-03-12	-698.00	False	57.79	2.667	211.380643	
6051	1	47	2010-10-08	-58.00	False	63.93	2.633	211.746754	
...	...	...	...	...	...	...	...	...	
419597	45	80	2010-02-12	-0.43	True	27.73	2.773	181.982317	
419598	45	80	2010-02-19	-0.27	False	31.27	2.745	182.034782	
419603	45	80	2010-04-16	-1.61	False	54.28	2.899	181.692477	
419614	45	80	2010-07-02	-0.27	False	76.61	2.815	182.318780	
419640	45	80	2011-02-11	-0.24	True	30.30	3.239	183.701613	

1358 rows × 12 columns

In [11]:
```
traindf3=traindf2.loc[traindf2['Weekly_Sales']>0]
traindf4=traindf3.drop(['_merge'],axis=1)
```

In [12]:
```
traindf4.sort_values(by='Date')
```

Out[12]:

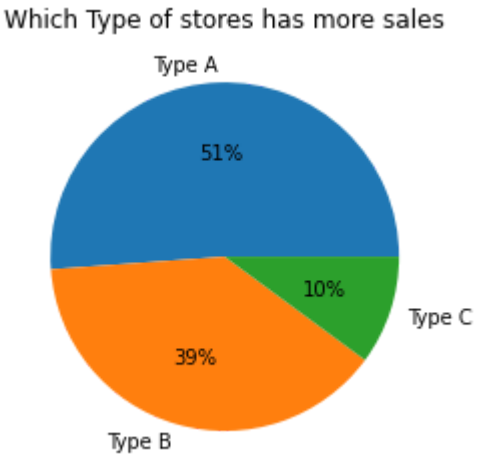
	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemp
0	1	1	2010-02-05	24924.50	False	42.31	2.572	211.096358	
140804	15	21	2010-02-05	3253.19	False	19.83	2.954	131.527903	
140661	15	20	2010-02-05	4606.90	False	19.83	2.954	131.527903	
140518	15	19	2010-02-05	1381.40	False	19.83	2.954	131.527903	
140408	15	18	2010-02-05	2239.25	False	19.83	2.954	131.527903	
...	...	...	...	...	...	...	...	...	
173673	18	52	2012-10-26	2226.10	False	56.09	3.917	138.728161	
342211	36	16	2012-10-26	564.50	False	74.39	3.494	222.113657	
390158	41	92	2012-10-26	131128.24	False	41.80	3.686	199.219532	
175485	18	81	2012-10-26	14036.52	False	56.09	3.917	138.728161	
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	192.308899	

420212 rows × 11 columns

In [13]:
```
traindf4['Type'].unique() #Store varities
```

Out[13]:
```
array(['A', 'B', 'C'], dtype=object)
```

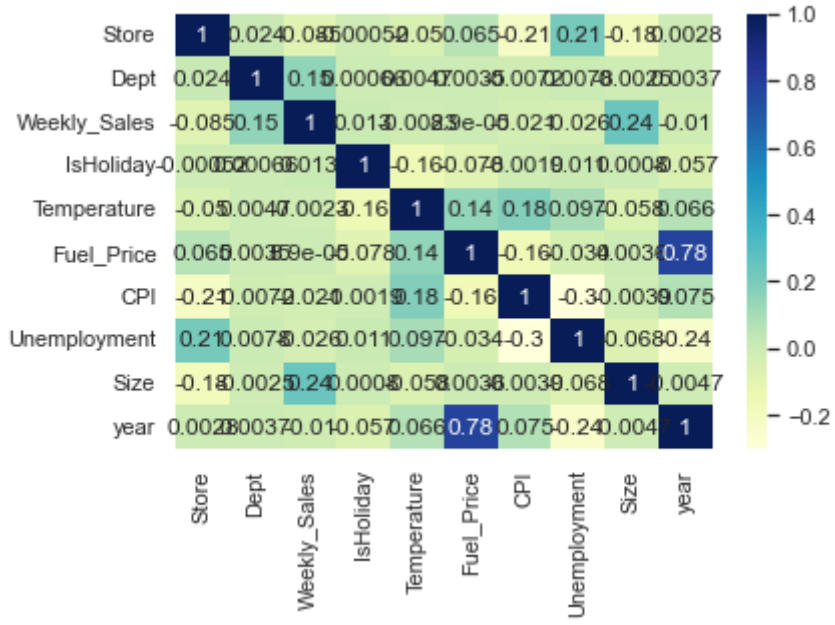
In [14]:
```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np


# Creating dataset
stores = ['Type A','Type B','Type C']

data = traindf4['Type'].value_counts()

# Creating plot
fig, ax = plt.subplots()
plt.pie(data, labels = stores,autopct='%.0f%%')
ax.set_title('Which Type of stores has more sales')
# show plot
plt.show()
```



In [15]:
```
traindf4['year'] = pd.DatetimeIndex(traindf4['Date']).year #Separating year data.
```

In [18]:
```
# import modules
import matplotlib.pyplot as mp
import pandas as pd
import seaborn as sns

# import file with data
data = traindf4

# prints data that will be plotted
# columns shown here are selected by corr() since
# they are ideal for the plot
print(data.corr())
sns.set_theme(style="whitegrid")
# plotting correlation heatmap
dataplot = sns.heatmap(data.corr(), cmap="YlGnBu", annot=True)
sns.set(rc = {'figure.figsize':(25,8)})

# displaying heatmap
mp.show()
```

```
                Store      Dept  Weekly_Sales  IsHoliday  Temperature  \
Store        1.000000  0.024258     -0.085117  -0.000522    -0.050230
Dept         0.024258  1.000000      0.148749   0.000663     0.004727
Weekly_Sales -0.085117  0.148749      1.000000   0.012843    -0.002339
IsHoliday    -0.000522  0.000663      0.012843   1.000000    -0.155775
Temperature  -0.050230  0.004727     -0.002339  -0.155775     1.000000
Fuel_Price    0.065321  0.003544      0.000089  -0.078155     0.143700
CPI          -0.211261 -0.007178     -0.021162  -0.001933     0.182223
Unemployment  0.208759  0.007787     -0.025806   0.010555     0.096768
Size         -0.182763 -0.002491      0.244117   0.000797    -0.058413
year          0.002831  0.003716     -0.010015  -0.056572     0.065712

              Fuel_Price       CPI  Unemployment      Size      year
Store           0.065321 -0.211261      0.208759 -0.182763  0.002831
Dept            0.003544 -0.007178      0.007787 -0.002491  0.003716
Weekly_Sales    0.000089 -0.021162     -0.025806  0.244117 -0.010015
IsHoliday      -0.078155 -0.001933      0.010555  0.000797 -0.056572
Temperature     0.143700  0.182223      0.096768 -0.058413  0.065712
Fuel_Price      1.000000 -0.164199     -0.033915  0.003632  0.779681
CPI            -0.164199  1.000000     -0.299887 -0.003903  0.074547
Unemployment   -0.033915 -0.299887      1.000000 -0.068335 -0.237210
Size            0.003632 -0.003903     -0.068335  1.000000 -0.004716
year            0.779681  0.074547     -0.237210 -0.004716  1.000000
```

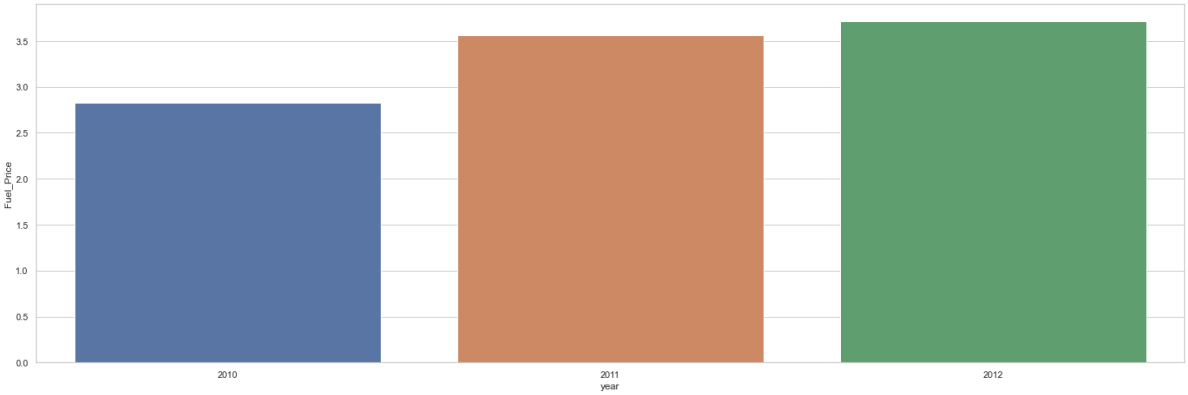


In [19]:
```
print(traindf4.dtypes)
```

```
Store             int64
Dept              int64
Date             object
Weekly_Sales    float64
IsHoliday          bool
Temperature     float64
Fuel_Price      float64
CPI             float64
Unemployment    float64
Type             object
Size              int64
year              int64
dtype: object
```

Year vs Fuel_price

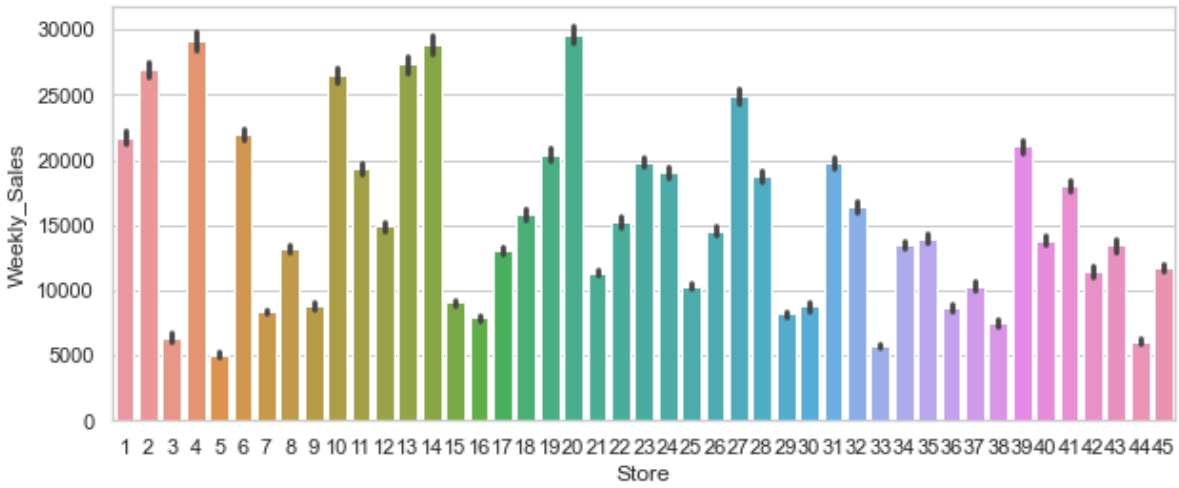
In [20]:
```
import seaborn as sns
sns.set_theme(style="whitegrid")
tips =traindf4
ax = sns.barplot(x="year", y="Fuel_Price", data=tips)
sns.set(rc = {'figure.figsize':(10,4)})
```



Weekly sales vs Store

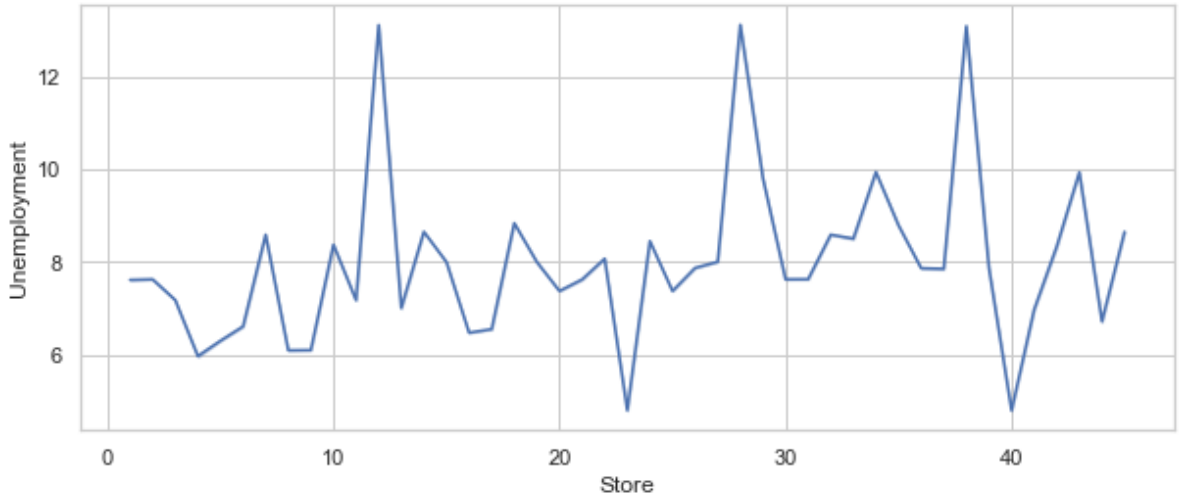
In [21]:
```
import seaborn as sns
sns.set_theme(style="whitegrid")
```

```
tips = traindf4
ax = sns.barplot(x='Store', y="Weekly_Sales", data=tips)
```



Store vs Unemployment

In [22]:
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data = traindf4

# draw lineplot
sns.lineplot(x="Store", y="Unemployment", data=data)
plt.show()
```



In [23]:
```
traindf4
```

Out[23]:

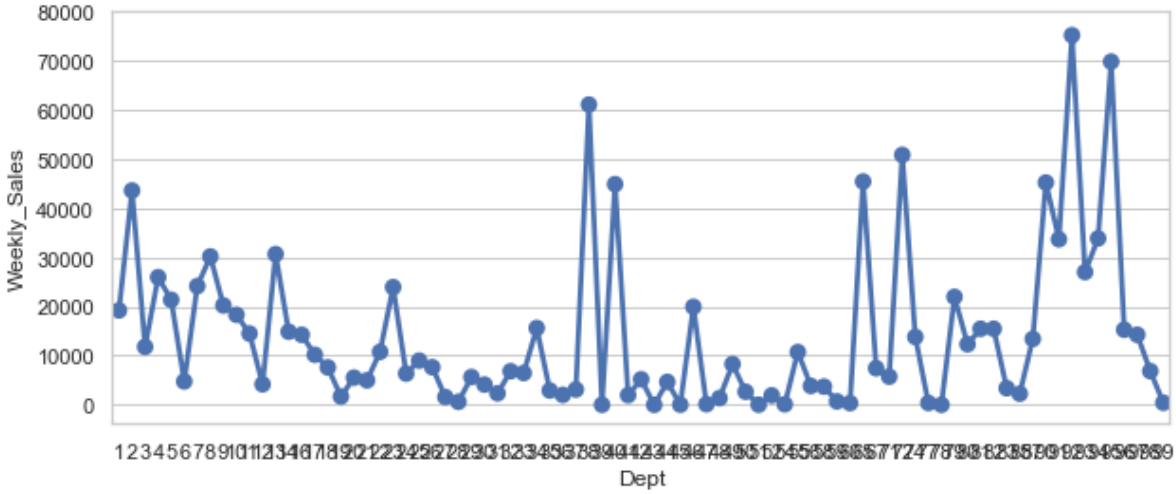
	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unem
0	1	1	2010-02-05	24924.50	False	42.31	2.572	211.096358	
1	1	1	2010-02-12	46039.49	True	38.51	2.548	211.242170	
2	1	1	2010-02-19	41595.55	False	39.93	2.514	211.289143	
3	1	1	2010-02-26	19403.54	False	46.63	2.561	211.319643	
4	1	1	2010-03-05	21827.90	False	46.50	2.625	211.350143	
...	...	...	...	...	...	...	...	...	
421565	45	98	2012-09-28	508.37	False	64.88	3.997	192.013558	
421566	45	98	2012-10-05	628.10	False	64.89	3.985	192.170412	
421567	45	98	2012-10-12	1061.02	False	54.47	4.000	192.327265	
421568	45	98	2012-10-19	760.01	False	56.47	3.969	192.330854	
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	192.308899	

420212 rows × 12 columns

In [24]:
```
traindf4['Dept'].unique()
```

Out[24]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 16, 17, 18,
       19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
       36, 37, 38, 40, 41, 42, 44, 45, 46, 47, 48, 49, 51, 52, 54, 55, 56,
       58, 59, 60, 67, 71, 72, 74, 77, 78, 79, 80, 81, 82, 83, 85, 87, 90,
       91, 92, 93, 94, 95, 96, 97, 98, 99, 39, 50, 43, 65], dtype=int64)
```

In [25]:
```
# importing required packages
import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data =traindf4

# draw pointplot
sns.pointplot(x ='Dept',
                        y = "Weekly_Sales",
                        data = data)
# show the plot
sns.set(rc = {'figure.figsize':(25,8)})
plt.show()
```



In [26]:
```
traindf4['month'] = pd.DatetimeIndex(traindf4['Date']).month #extract month data
```

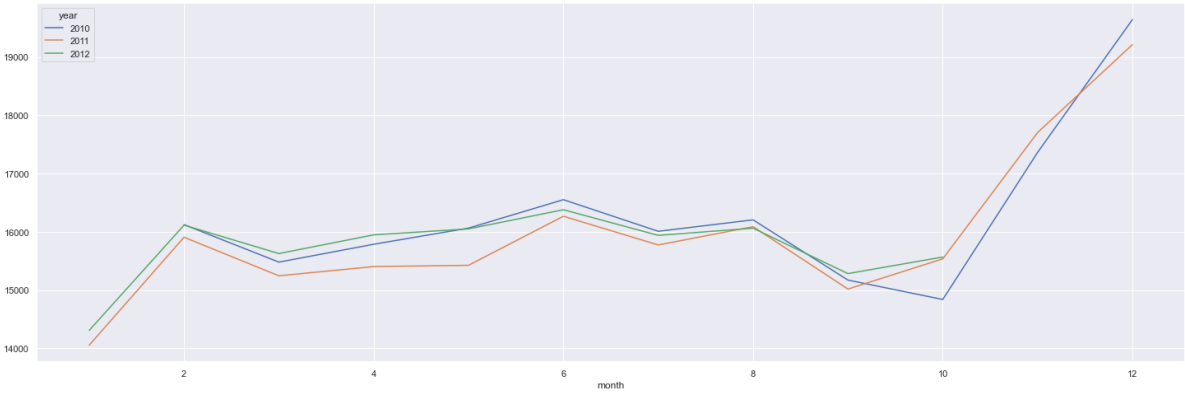
In [27]:
```
traindf4['week'] = pd.DatetimeIndex(traindf4['Date']).week #extract week data
```

```
C:\Users\shubh\AppData\Local\Temp\ipykernel_560\2593830182.py:1: FutureWarning: we
ekofyear and week have been deprecated, please use DatetimeIndex.isocalendar().wee
k instead, which returns a Series. To exactly reproduce the behavior of week and w
eekofyear and return an Index, you may call pd.Int64Index(idx.isocalendar().week)
  traindf4['week'] = pd.DatetimeIndex(traindf4['Date']).week #extract week data
```

In [28]:
```
traindf5=traindf4.drop(['Date'],axis=1)
```

In [29]:
```
month_wise_sales = pd.pivot_table(traindf5, values = "Weekly_Sales", columns = "ye
month_wise_sales.plot()
```

Out[29]:
```
<Axes: xlabel='month'>
```



Label encoding for Holiday column and Type

In [30]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
traindf5['IsHoliday']= label_encoder.fit_transform(traindf5['IsHoliday'])
traindf5['Type']= label_encoder.fit_transform(traindf5['Type'])

traindf5
```

Out[30]:

	Store	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemployme
0	1	1	24924.50	0	42.31	2.572	211.096358	8.10
1	1	1	46039.49	1	38.51	2.548	211.242170	8.10
2	1	1	41595.55	0	39.93	2.514	211.289143	8.10
3	1	1	19403.54	0	46.63	2.561	211.319643	8.10
4	1	1	21827.90	0	46.50	2.625	211.350143	8.10
...	...	...	...	...	...	...	...	
421565	45	98	508.37	0	64.88	3.997	192.013558	8.68
421566	45	98	628.10	0	64.89	3.985	192.170412	8.66
421567	45	98	1061.02	0	54.47	4.000	192.327265	8.66
421568	45	98	760.01	0	56.47	3.969	192.330854	8.66
421569	45	98	1076.80	0	58.85	3.882	192.308899	8.66

420212 rows × 13 columns

Correlation Map 2

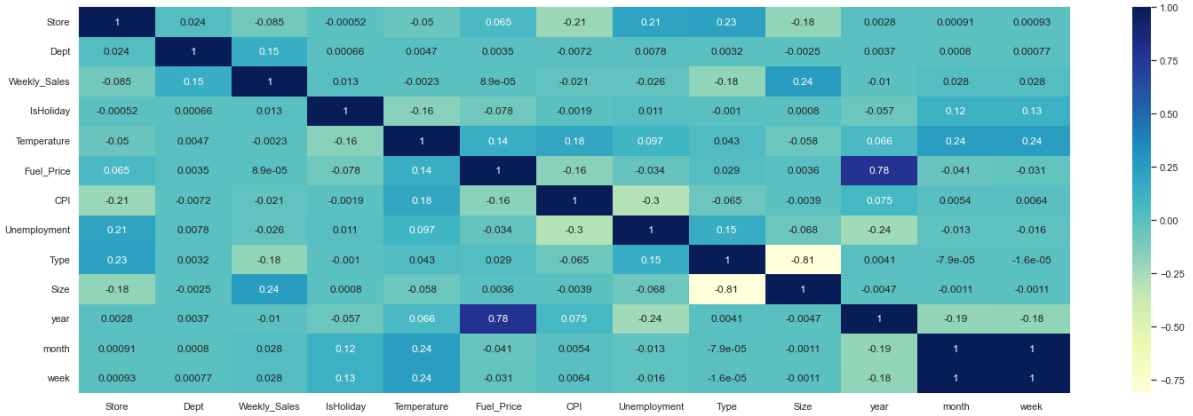
In [31]:

```
data = traindf5

# prints data that will be plotted
# columns shown here are selected by corr() since
# they are ideal for the plot
print(data.corr())
sns.set_theme(style="whitegrid")
# plotting correlation heatmap
dataplot = sns.heatmap(data.corr(), cmap="YlGnBu", annot=True)
sns.set(rc = {'figure.figsize':(25,8)})

# displaying heatmap
mp.show()
```

	Store	Dept	Weekly_Sales	IsHoliday	Temperature
Store	1.000000	0.024258	-0.085117	-0.000522	-0.050230
Dept	0.024258	1.000000	0.148749	0.000663	0.004727
Weekly_Sales	-0.085117	0.148749	1.000000	0.012843	-0.002339
IsHoliday	-0.000522	0.000663	0.012843	1.000000	-0.155775
Temperature	-0.050230	0.004727	-0.002339	-0.155775	1.000000
Fuel_Price	0.065321	0.003544	0.000089	-0.078155	0.143700
CPI	-0.211261	-0.007178	-0.021162	-0.001933	0.182223
Unemployment	0.208759	0.007787	-0.025806	0.010555	0.096768
Type	0.226352	0.003157	-0.182229	-0.001000	0.043035
Size	-0.182763	-0.002491	0.244117	0.000797	-0.058413
year	0.002831	0.003716	-0.010015	-0.056572	0.065712
month	0.000907	0.000800	0.028401	0.123058	0.235957
week	0.000926	0.000767	0.027659	0.127846	0.236256

	Fuel_Price	CPI	Unemployment	Type	Size
Store	0.065321	-0.211261	0.208759	0.226352	-0.182763
Dept	0.003544	-0.007178	0.007787	0.003157	-0.002491
Weekly_Sales	0.000089	-0.021162	-0.025806	-0.182229	0.244117
IsHoliday	-0.078155	-0.001933	0.010555	-0.001000	0.000797
Temperature	0.143700	0.182223	0.096768	0.043035	-0.058413
Fuel_Price	1.000000	-0.164199	-0.033915	0.029483	0.003632
CPI	-0.164199	1.000000	-0.299887	-0.065094	-0.003903
Unemployment	-0.033915	-0.299887	1.000000	0.148793	-0.068335
Type	0.029483	-0.065094	0.148793	1.000000	-0.811541
Size	0.003632	-0.003903	-0.068335	-0.811541	1.000000
year	0.779681	0.074547	-0.237210	0.004080	-0.004716
month	-0.040931	0.005366	-0.012562	-0.000079	-0.001051
week	-0.031191	0.006428	-0.015614	-0.000016	-0.001130

	year	month	week
Store	0.002831	0.000907	0.000926
Dept	0.003716	0.000800	0.000767
Weekly_Sales	-0.010015	0.028401	0.027659
IsHoliday	-0.056572	0.123058	0.127846
Temperature	0.065712	0.235957	0.236256
Fuel_Price	0.779681	-0.040931	-0.031191
CPI	0.074547	0.005366	0.006428
Unemployment	-0.237210	-0.012562	-0.015614
Type	0.004080	-0.000079	-0.000016
Size	-0.004716	-0.001051	-0.001130
year	1.000000	-0.194295	-0.181804
month	-0.194295	1.000000	0.996000
week	-0.181804	0.996000	1.000000



Feature Importance Test using various techniques

In [37]:
```
from sklearn.inspection import permutation_importance
from sklearn.ensemble import RandomForestRegressor
import shap
```

```
Using `tqdm.autonotebook.tqdm` in notebook mode. Use `tqdm.tqdm` instead to force
console mode (e.g. in jupyter console)
```

In [38]:
```
Features=traindf5.drop(['Weekly_Sales'],axis=1)
Target=traindf5['Weekly_Sales']
```

In [39]:
```
rf = RandomForestRegressor(n_estimators=100)
rf.fit(Features,Target)
```

Out[39]:
```
▾ RandomForestRegressor

RandomForestRegressor()
```

In [40]:
```
Features
```

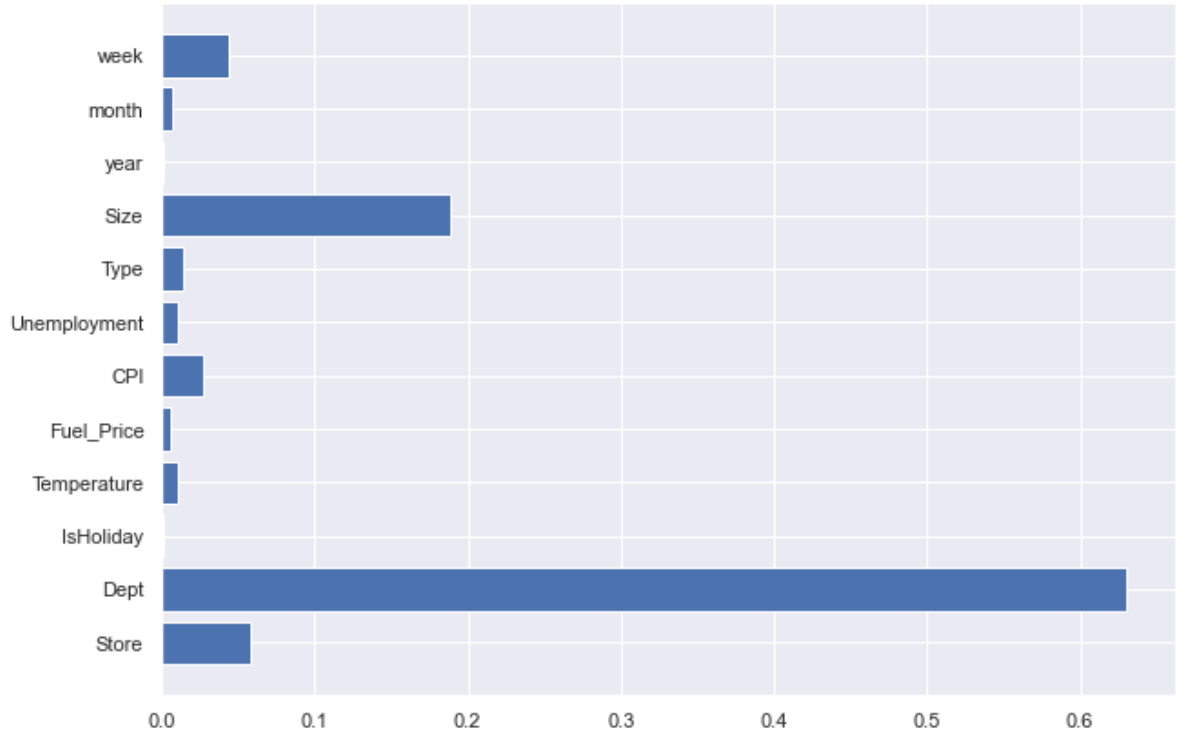
Out[40]:

	Store	Dept	IsHoliday	Temperature	Fuel_Price	CPI	Unemployment	Type	Si
0	1	1	0	42.31	2.572	211.096358	8.106	0	1513
1	1	1	1	38.51	2.548	211.242170	8.106	0	1513
2	1	1	0	39.93	2.514	211.289143	8.106	0	1513
3	1	1	0	46.63	2.561	211.319643	8.106	0	1513
4	1	1	0	46.50	2.625	211.350143	8.106	0	1513
...	...	...	...	...	...	...	...	...	
421565	45	98	0	64.88	3.997	192.013558	8.684	1	1182
421566	45	98	0	64.89	3.985	192.170412	8.667	1	1182
421567	45	98	0	54.47	4.000	192.327265	8.667	1	1182
421568	45	98	0	56.47	3.969	192.330854	8.667	1	1182
421569	45	98	0	58.85	3.882	192.308899	8.667	1	1182

420212 rows × 12 columns

In [41]:
```
f = plt.figure()
f.set_figwidth(10)
f.set_figheight(7)
plt.barh(Features.columns, rf.feature_importances_)
```

Out[41]:
```
<BarContainer object of 12 artists>
```



In [42]:
```
F=Features.drop(["IsHoliday",'year'],axis=1)
```

In [43]:
```
F
```

Out[43]:

	Store	Dept	Temperature	Fuel_Price	CPI	Unemployment	Type	Size	month
0	1	1	42.31	2.572	211.096358	8.106	0	151315	2
1	1	1	38.51	2.548	211.242170	8.106	0	151315	2
2	1	1	39.93	2.514	211.289143	8.106	0	151315	2
3	1	1	46.63	2.561	211.319643	8.106	0	151315	2
4	1	1	46.50	2.625	211.350143	8.106	0	151315	3
...	...	...	...	...	...	...	...	...	...
421565	45	98	64.88	3.997	192.013558	8.684	1	118221	9
421566	45	98	64.89	3.985	192.170412	8.667	1	118221	10
421567	45	98	54.47	4.000	192.327265	8.667	1	118221	10
421568	45	98	56.47	3.969	192.330854	8.667	1	118221	10
421569	45	98	58.85	3.882	192.308899	8.667	1	118221	10

420212 rows × 10 columns

In [44]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(F, Target, test_size= 0.25, ran
```

In [ ]:
```
```

In [45]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.metrics import r2_score,mean_squared_error
from math import sqrt
```

In [46]:
```
DTRmodel = DecisionTreeRegressor(max_depth=3,random_state=0)
DTRmodel.fit(x_train,y_train)
y_pred = DTRmodel.predict(x_test)
```

In [47]:
```
print("R2 score  :",r2_score(y_test, y_pred))
print("MSE score  :",mean_squared_error(y_test, y_pred))
print("RMSE: ",sqrt(mean_squared_error(y_test, y_pred)))
```

```
R2 score  : 0.3796166061691
MSE score  : 323184793.8287025
RMSE:  17977.34112233237
```

In [48]:
```
rf1 = RandomForestRegressor(n_estimators=50, random_state=42, n_jobs=-1, max_depth=
                            max_features = 'sqrt',min_samples_split = 10)
rf1.fit(x_train,y_train)
y_pred1 = rf1.predict(x_test)
```

In [49]:
```
print("R2 score  :",r2_score(y_test, y_pred))
print("MSE score  :",mean_squared_error(y_test, y_pred1))
print("RMSE: ",sqrt(mean_squared_error(y_test, y_pred1)))
```

```
R2 score  : 0.3796166061691
MSE score  : 62845166.705938146
RMSE:  7927.494352311968
```

In [54]:
```
from xgboost import XGBRegressor
model = XGBRegressor()
model.fit(x_train,y_train)
```

Out[54]:

▾ XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types
=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_typ
e=None,
             interaction_constraints=None, learning_rate=None, max_bin
=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
```

In [55]:
```
y_pred2 = model.predict(x_test)
```

In [56]:
```
print("R2 score  :",r2_score(y_test, y_pred2))
print("MSE score  :",mean_squared_error(y_test, y_pred2))
print("RMSE: ",sqrt(mean_squared_error(y_test, y_pred2)))
```

```
R2 score  : 0.9444449403000315
MSE score  : 28941055.95639064
RMSE:  5379.689206300922
```

In [57]:
```
y_pred2
```

Out[57]:
```
array([26847.85 ,   363.88 , 33762.836, ...,  8514.317,  5247.591,
        9374.635], dtype=float32)
```

In [58]:
```
#Regularization
from sklearn.linear_model import Ridge
rr_model = Ridge(alpha=0.5)
rr_model.fit(x_train,y_train)
```

Out[58]:
```
Ridge(alpha=0.5)
```

In [59]:
```
y_pred3 = model.predict(x_test)
```

In [60]:
```
y_pred3
```

Out[60]:
```
array([26847.85 ,   363.88 , 33762.836, ...,  8514.317,  5247.591,
        9374.635], dtype=float32)
```

In [61]:
```
print("R2 score  :",r2_score(y_test, y_pred3))
print("MSE score  :",mean_squared_error(y_test, y_pred3))
print("RMSE: ",sqrt(mean_squared_error(y_test, y_pred3)))
```

```
R2 score  : 0.9444449403000315
MSE score  : 28941055.95639064
RMSE:  5379.689206300922
```

In [62]:
```
y_test
```

Out[62]:
```
198556    18526.46
342491       84.00
267645    27025.56
169044    48324.54
45102      2968.68
             ...
323292     4156.69
22041     15252.97
261294     8461.31
212048      835.99
406506     6728.27
Name: Weekly_Sales, Length: 105053, dtype: float64
```

In [ ]:

In [ ]: