0. Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

1. Load and Explore The Dataset

First let us understand what we are doing, what's the purpose of our project?. We're analyzing sales data from Walmart to develop an anomaly detection system.our goal is to identify unusual sales patterns across stores and departments without labeled data, making it an unsupervised machine learning task.by detecting anomalies,we aim to improve Walmart's operational efficiency and decision-making regarding inventory management and sales forecasting.

In [25]:
```
data = pd.read_csv('./walmart_cleaned.csv')
data.head()
```

Out[25]:

	Unnamed: 0	Store	Date	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1
0	0	1	2010-02-05	0	1.0	24924.50	42.31	2.572	0.0
1	1	1	2010-02-05	0	26.0	11737.12	42.31	2.572	0.0
2	2	1	2010-02-05	0	17.0	13223.76	42.31	2.572	0.0
3	3	1	2010-02-05	0	45.0	37.44	42.31	2.572	0.0
4	4	1	2010-02-05	0	28.0	1085.29	42.31	2.572	0.0

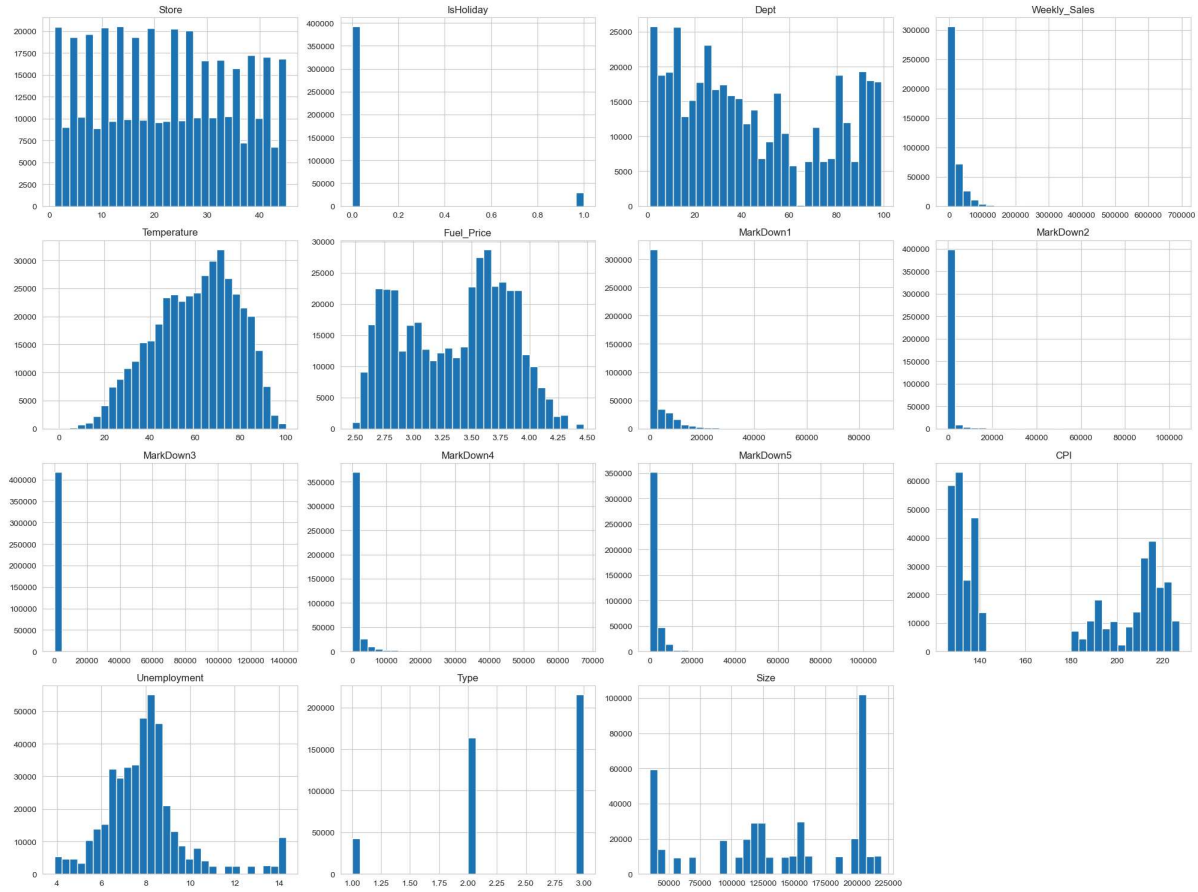
In [4]:
```
data.describe()
```

Out[4]:

	Unnamed: 0	Store	IsHoliday	Dept	Weekly_Sales	Temperature
count	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000
mean	211611.321278	22.200546	0.070358	44.260317	15981.258123	60.090059
std	122195.149363	12.785297	0.255750	30.492054	22711.183519	18.447931
min	0.000000	1.000000	0.000000	1.000000	-4988.940000	-2.060000
25%	105782.250000	11.000000	0.000000	18.000000	2079.650000	46.680000
50%	211603.500000	22.000000	0.000000	37.000000	7612.030000	62.090000
75%	317424.750000	33.000000	0.000000	74.000000	20205.852500	74.280000
max	423285.000000	45.000000	1.000000	99.000000	693099.360000	100.140000

In [26]:
```
data=data.drop(['Unnamed: 0'],axis=1)
```

In [19]:
```
sns.set_style("whitegrid")

data.hist(bins=30, figsize=(20, 15))
plt.tight_layout()
plt.show()
```



Data Analysis -Markdown Features (MarkDown1 to MarkDown5): These features are predominantly zero, suggesting that markdowns are infrequent. Most weeks do not have markdowns, and only a few weeks show significant promotional activity. -Holiday Indicator (IsHoliday): As expected, the majority of the data points are non-holiday weeks. -Economic Indicators (CPI, Temperature, Unemployment): CPI: This feature shows a relatively balanced distribution, indicating a stable variation in consumer price index values. -Temperature: The data reveals a slight left skew, hinting that cooler temperatures are more common. -Unemployment: Similar to

CPI, this feature is fairly symmetrically distributed, reflecting consistent unemployment rates over time. -Fuel Price (Fuel_Price): The distribution is mildly right-skewed, indicating that lower fuel prices are more common while higher prices are less frequent. -Weekly Sales (Weekly_Sales): Sales data is heavily right-skewed, meaning most weeks have lower sales figures, with high sales being relatively rare. Store Characteristics (Size and Type): Size: Larger stores are more prevalent, suggesting that the dataset is dominated by bigger Walmart stores. Type: Type 3 stores are the most common, which might indicate a specific classification of stores that are more frequent in the dataset.

In [7]:
```
### we need to see the types of the data to see if anything need to be converted
pd.DataFrame(data.dtypes, columns=['Type']).T
```

Out[7]:

	Store	Date	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	Markl
Type	int64	object	int64	float64	float64	float64	float64	float64	

In [5]:
```
### we can notice from the previous output that all of the columns are int64 or float64 except
### for the Date column, we'll solve this in the next section (preprocessing)
```

2. Data Preprocessing

-As we already saw in the previous section, the date column is an object we need to convert it either to numeric value or date time format, both will work just fine because we'are dealing with a time series problem so the date will be our index. -In addition to that, we need to remove all the weekly_sales instances that has negative values. -Hence There Are no categorical values so no need to Encoding

In [27]:
```
# Remove rows with negative Weekly_Sales
data = data[data["Weekly_Sales"] >= 0]
```

In [28]:
```
###Converting the Date column in Date Time Format
data['Date'] = pd.to_datetime(data['Date'])
```

In [10]:
```
data.head()
```

Out[10]:

	Store	Date	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown
0	1	2010-02-05	0	1.0	24924.50	42.31	2.572	0.0	0.
1	1	2010-02-05	0	26.0	11737.12	42.31	2.572	0.0	0.
2	1	2010-02-05	0	17.0	13223.76	42.31	2.572	0.0	0.
3	1	2010-02-05	0	45.0	37.44	42.31	2.572	0.0	0.
4	1	2010-02-05	0	28.0	1085.29	42.31	2.572	0.0	0.

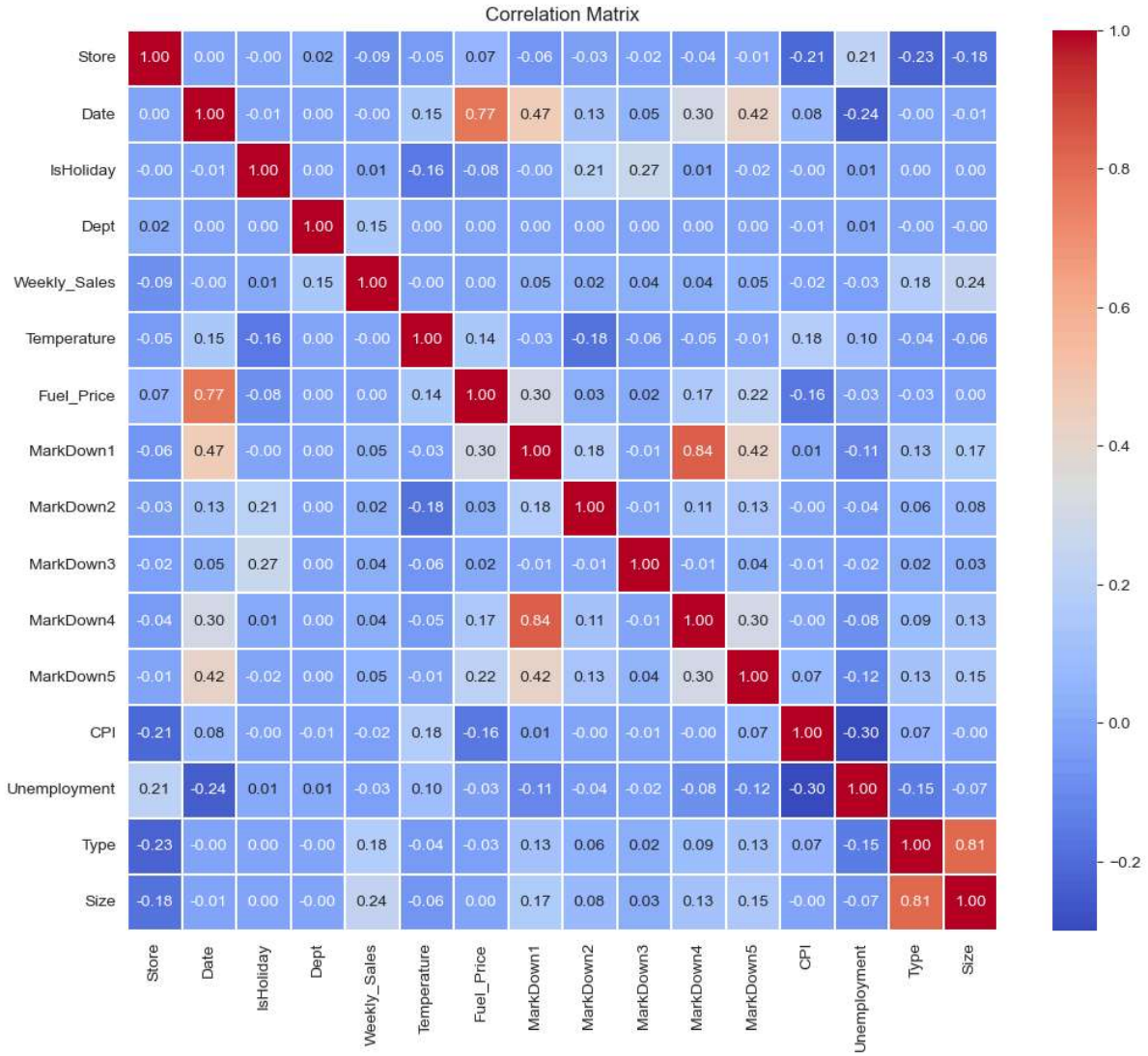
5 rows × 21 columns

In [11]:
```
### Cheaking NA values
data.isna().sum()
```

Out[11]:
```
Store           0
Date            0
IsHoliday       0
Dept            0
Weekly_Sales    0
Temperature     0
Fuel_Price      0
MarkDown1       0
MarkDown2       0
MarkDown3       0
MarkDown4       0
MarkDown5       0
CPI             0
Unemployment    0
Type            0
Size            0
Year            0
Month           0
Week            0
DayOfTheWeek    0
Day             0
dtype: int64
```

In [10]:
```
### No NA values in our data
```

In [29]:
```
# Plot the heatmap of the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f",
linewidths=0.1)
plt.title('Correlation Matrix')
plt.show()
```



I will drop the Type column as it is Quite correlated with Size and we don't really understand the type column and it's purpose, so Size alone is sufficient. Additionally, I will drop the Store and Dept columns since anomaly detection needs to be applied across all stores and departments, making these identifiers unnecessary. Additionally We Can Notice That MarkDown1 and MarkDown4 are Highly Correlated (the correlation factor is 0.84) , But It's less than 0.9. So it's up to us to decide weather we want to drop one of them or not. I'm Going To Stick with Both Of Them.

In [10]:
```
###Selecting a subset of the dataset
col_features=['Date', 'IsHoliday', 'Weekly_Sales',
       'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2',
'MarkDown3',
       'MarkDown4', 'MarkDown5', 'CPI', 'Unemployment', 'Size']
df = data[col_features]
```

In [12]:
```
### in time series problem it's most likely to set the Date As Index,
but it's not necessary
df.set_index('Date', inplace=True)
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 420285 entries, 2010-02-05 to 2012-10-26
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   IsHoliday     420285 non-null  int64  
 1   Weekly_Sales  420285 non-null  float64
 2   Temperature   420285 non-null  float64
 3   Fuel_Price    420285 non-null  float64
 4   MarkDown1     420285 non-null  float64
 5   MarkDown2     420285 non-null  float64
 6   MarkDown3     420285 non-null  float64
 7   MarkDown4     420285 non-null  float64
 8   MarkDown5     420285 non-null  float64
 9   CPI           420285 non-null  float64
 10  Unemployment  420285 non-null  float64
 11  Size          420285 non-null  int64  
dtypes: float64(10), int64(2)
memory usage: 41.7 MB
```

Now The Main Quetsion Is Do we Need Scaling? Answer:For anomaly detection Isolation Forest, scaling the data may not be helpful.Isolation Forest creates random splits, so scaling may not significantly impact it's performance. However, for DBSCAN, scaling could be beneficial as it relies on density-based clustering and is sensitive to the scale of the features, ensuring more accurate clustering.

3. Fit the dataset using base models

In this section, we will apply three anomaly detection algorithms: KNN (k-Nearest Neighbors), Isolation Forest, and DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Our primary focus is to detect unusual sales patterns in the weekly sales data from Walmart. -KNN: K-nearest neighbors is a simple algorithm that classifies a data point based on the majority class of its k-nearest neighbors. -Isolation Forest: Isolation Forest is an ensemble learning algorithm that isolates anomalies by randomly selecting features and splitting data points. -DBSCAN: Density-Based Spatial Clustering of Applications with Noise identifies clusters based on density and is effective in separating dense regions from sparse regions in the data. By applying these algorithms, we aim to identify anomalous sales behavior and improve Walmart's decision-making processes related to inventory management and sales forecasting.

3.1. KNN

In [17]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors

df_scaled=df.copy()
# Initialize the StandardScaler
scaler = StandardScaler()
### No Need To Scale IsHoliday because it's already between 0 and 1
features_to_be_scaled=['Weekly_Sales',
       'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2',
'MarkDown3',
       'MarkDown4', 'MarkDown5', 'CPI', 'Unemployment', 'Size']
df_scaled[features_to_be_scaled] =
scaler.fit_transform(df_scaled[features_to_be_scaled])
```

```
### we will try different n_neighbors to reach the optimum number
#nbrs = NearestNeighbors(n_neighbors = 3)
#nbrs = NearestNeighbors(n_neighbors = 5)
nbrs = NearestNeighbors(n_neighbors = 10)
# fit model
nbrs.fit(df_scaled.values)
```

Out[17]:

```
NearestNeighbors(n_neighbors=10)
```

In [18]:

```
# distances and indexes of k-neaighbors from model outputs
distances, indexes = nbrs.kneighbors(df_scaled.values)
```

Now we need to determine the threshold(cut_off) we will apply some statistical references on the distances to determine the threshold

In [19]:

```
distances = pd.DataFrame(distances)
distances_mean = distances.mean(axis =1)
distances_mean
```

Out[19]:

```
0         0.144475
1         0.038536
2         0.050681
3         0.026852
4         0.024361
             ...
420280    0.225526
420281    0.023152
420282    0.028520
420283    0.015128
420284    0.009492
Length: 420285, dtype: float64
```

In [20]:

```
distances_mean.describe()
```

Out[20]:

```
count    420285.000000
mean          0.083142
std           0.139485
min           0.000184
25%           0.020020
50%           0.041113
75%           0.090400
max          14.133607
dtype: float64
```

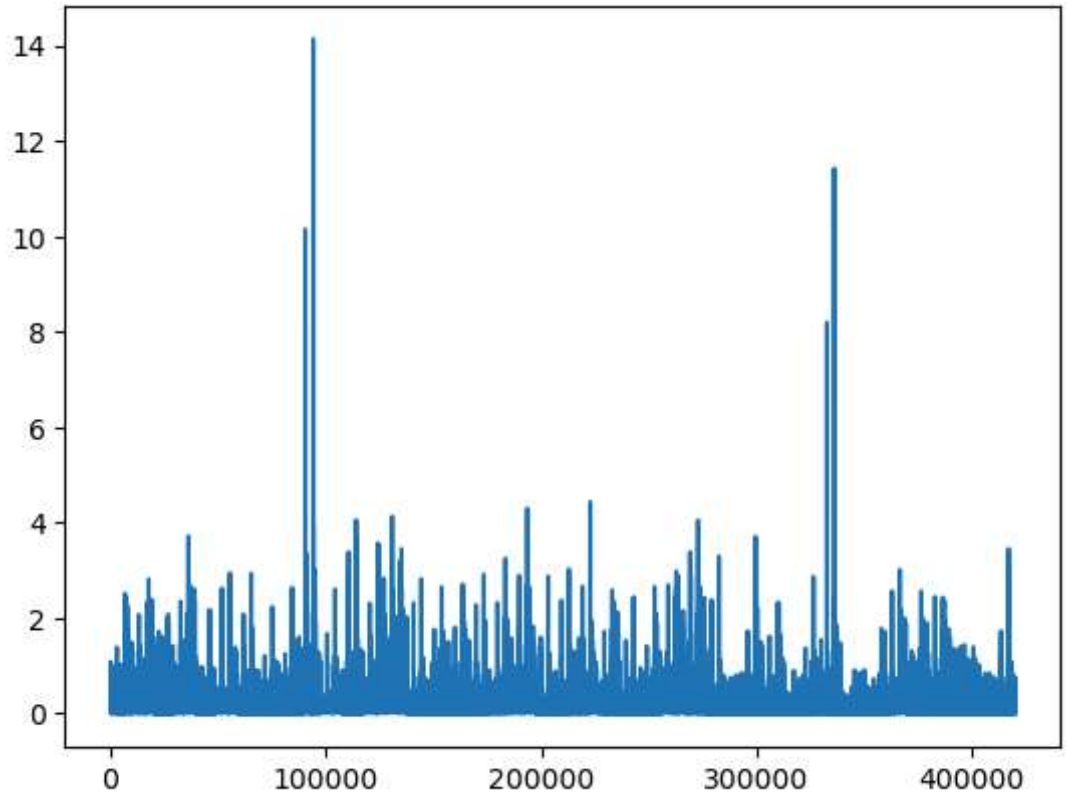
Since 75th percentile is 0.1.however we can see that the max is 14.13 , so we need to see it visually.

In [18]:

```
plt.plot(distances_mean)
```

Out[18]:

```
[<matplotlib.lines.Line2D at 0x1df07d86910>]
```



In [19]:

```
### visualy we can see that most of the points have an average distance
<=2
th=2
outlier_index = np.where(distances_mean > th)
outlier_index
```

Out[19]:

```
(array([  6747,   6759,   7471,   7749,  13225,  16838,  16960,  16975,
        17385,  17697,  19094,  19133,  19527,  26709,  32552,  36105,
        36227,  36242,  36317,  36514,  36603,  36694,  36970,  36980,
        37020,  37092,  38867,  38873,  39194,  39224,  45665,  45740,
        46302,  51722,  55478,  61808,  65296,  75154,  84346,  90433,
        90742,  90752,  94024,  94158,  94188,  94190,  94236,  94452,
        94503,  94547,  94876,  94946,  95073, 104250, 110545, 110774,
       114051, 114295, 120427, 124123, 124225, 124245, 124247, 124269,
       124276, 124287, 124316, 124433, 124726, 125022, 125095, 125150,
       125233, 125297, 125810, 126913, 126915, 127266, 130749, 131045,
       133606, 134264, 134374, 134400, 134488, 134721, 134791, 134824,
       135098, 135180, 135226, 135369, 135841, 137444, 140694, 144282,
       153774, 163511, 163521, 169767, 173323, 179672, 183233, 183365,
       189785, 193368, 193491, 193503, 193551, 193568, 193966, 194266,
       194321, 194389, 194433, 194511, 203236, 209371, 212872, 219141,
       222767, 222772, 222793, 232923, 232957, 233663, 233999, 235582,
       242861, 252631, 253352, 259001, 262624, 262690, 262774, 262775,
       263445, 263778, 265907, 269178, 269490, 272698, 272842, 272843,
       273291, 273574, 273849, 275807, 279114, 282531, 299396, 299564,
       299568, 300301, 309725, 310431, 310476, 326432, 332663, 336064,
       336269, 336402, 362892, 366488, 367177, 376420, 377113, 382811,
       386429, 387105, 417030], dtype=int64),)
```

In [20]:
```
outlier_values = df.iloc[outlier_index]
outlier_values
```

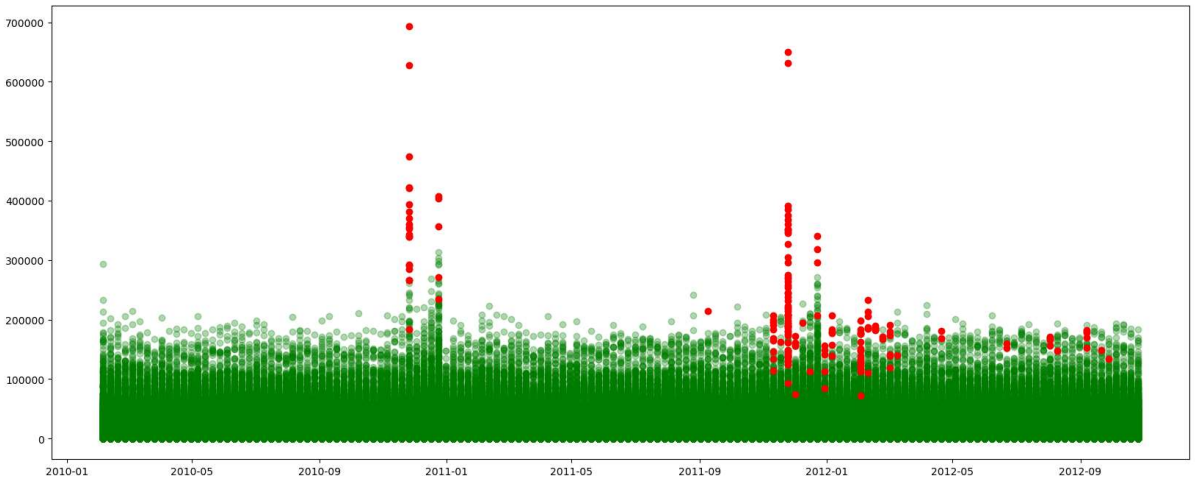
Out[20]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2011-11-25	1	186516.35	60.14	3.236	410.31	98.00	55805.51
2011-11-25	1	203670.47	60.14	3.236	410.31	98.00	55805.51
2012-02-03	0	147179.82	56.55	3.360	34577.06	3579.21	160.53
2012-03-02	0	138638.98	60.96	3.630	15441.40	1569.00	10.80
2010-11-26	1	285353.53	62.98	2.735	0.00	0.00	0.00
...	...	...	...	...	...	...	...
2012-02-03	0	113179.16	27.86	3.633	37980.28	2731.05	108.41
2010-11-26	1	290809.17	25.30	2.742	0.00	0.00	0.00
2011-11-25	1	271392.45	36.37	3.424	256.95	1053.98	79621.20
2012-02-03	0	130392.27	31.65	3.031	42353.77	4696.86	231.92
2011-11-25	1	198041.11	48.71	3.492	140.87	384.82	26961.99

171 rows × 12 columns

In [21]:
```
# plot data
plt.figure(figsize=(20,8))
plt.scatter(df.index,df["Weekly_Sales"] ,color = "g",alpha=0.3)
# plot outlier values (where outlier values= df.iloc[outlier_index])
otlr_val = df.iloc[outlier_index]
plt.scatter(otlr_val.index,otlr_val["Weekly_Sales"], color = "r")
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x1df081c0bd0>
```



In [21]:

```
### let's try lower threshold
th=1.5
outlier_index = np.where(distances_mean > th)
outlier_index
```

Out[21]:

```
(array([  6624,   6728,   6747,   6759,   6869,   7158,   7178,   7467,
         7471,   7550,   7637,   7749,   7757,  13225,  16838,  16844,
        16960,  16975,  17031,  17119,  17319,  17385,  17429,  17448,
        17657,  17697,  17752,  17835,  17940,  19094,  19133,  19527,
        19557,  26364,  26383,  26442,  26709,  26763,  27054,  32552,
        32817,  36076,  36105,  36227,  36242,  36252,  36265,  36288,
        36317,  36331,  36371,  36503,  36514,  36529,  36603,  36666,
        36694,  36720,  36750,  36944,  36970,  36980,  37008,  37020,
        37034,  37092,  37097,  37118,  38867,  38873,  45665,  45740,
        46302,  51722,  55435,  55478,  55819,  61808,  65296,  65607,
        65662,  75154,  84346,  84986,  90433,  90675,  90742,  90752,
        94024,  94074,  94158,  94188,  94190,  94236,  94316,  94452,
        94503,  94547,  94876,  94946,  95073,  95132,  95218, 104250,
       110545, 110774, 113989, 114051, 114231, 114295, 120427, 124081,
       124098, 124123, 124225, 124245, 124247, 124269, 124276, 124287,
       124316, 124370, 124427, 124433, 124523, 124609, 124652, 124726,
       124784, 125005, 125006, 125022, 125060, 125095, 125140, 125150,
       125233, 125235, 125253, 125260, 125297, 125729, 125744, 125808,
       125810, 125821, 126865, 126913, 126915, 127808, 130749, 131040,
       131045, 133465, 134264, 134308, 134374, 134383, 134400, 134488,
       134543, 134721, 134758, 134791, 134824, 134925, 135072, 135098,
       135180, 135226, 135295, 135369, 135428, 135718, 135774, 135784,
       135841, 135875, 136000, 136994, 137014, 137200, 137289, 137444,
       140694, 144282, 150402, 153774, 154422, 154485, 159960, 163511,
       163521, 163552, 164188, 164341, 164506, 166003, 169767, 173311,
       173323, 174110, 179672, 183233, 183365, 183381, 183819, 184062,
       184374, 185948, 189785, 190102, 190107, 193349, 193368, 193491,
       193503, 193517, 193520, 193545, 193551, 193568, 193652, 193950,
       193966, 193967, 194021, 194033, 194231, 194266, 194284, 194321,
       194331, 194389, 194390, 194433, 194472, 194511, 194525, 196068,
       196080, 203236, 209371, 212872, 217738, 219141, 219400, 219417,
       222767, 222772, 222793, 223026, 223098, 223173, 223497, 223572,
       223808, 232780, 232923, 232957, 232980, 233052, 233338, 233663,
       233709, 233857, 233999, 234271, 234479, 235563, 235582, 235810,
       242861, 252631, 253014, 253352, 253657, 259001, 262602, 262624,
       262660, 262690, 262774, 262775, 262853, 263123, 263179, 263236,
       263337, 263436, 263445, 263502, 263624, 263685, 263778, 265342,
       265346, 265907, 269178, 269490, 272698, 272842, 272843, 272925,
       272988, 273178, 273291, 273293, 273574, 273637, 273849, 273891,
       275401, 275422, 275807, 279114, 282531, 299396, 299564, 299568,
       300005, 300301, 302099, 309558, 309708, 309725, 310431, 310476,
       310541, 310623, 310691, 310758, 326397, 326432, 332663, 335900,
       336000, 336027, 336045, 336064, 336110, 336229, 336269, 336402,
       336729, 336784, 336839, 336902, 362892, 366476, 366488, 366812,
       367177, 367328, 367462, 368524, 368554, 368607, 368634, 368679,
       368718, 368752, 368823, 368844, 368899, 369230, 369265, 369291,
       376420, 376437, 376827, 377113, 377181, 377305, 377398, 377464,
       382811, 386261, 386419, 386429, 387105, 387210, 387264, 387327,
       387395, 387516, 388962, 389020, 389730, 413480, 417005, 417030],
      dtype=int64),)
```

In [22]:

```
outlier_values = df.iloc[outlier_index]
outlier_values
```

Out[22]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2011-11-11	0	165723.32	59.11	3.297	10382.90	6115.67	215.07
2011-11-25	1	112688.97	60.14	3.236	410.31	98.00	55805.51
2011-11-25	1	186516.35	60.14	3.236	410.31	98.00	55805.51
2011-11-25	1	203670.47	60.14	3.236	410.31	98.00	55805.51
2011-12-09	0	165633.02	43.93	3.158	4640.65	19.00	105.02
...	...	...	...	...	...	...	...
2012-08-03	0	128724.44	73.16	3.528	40493.45	83.46	44.56
2012-10-12	0	145984.90	39.38	3.760	1570.23	0.00	26.31
2010-11-26	1	240758.86	46.15	3.039	0.00	0.00	0.00
2011-11-25	1	84018.86	48.71	3.492	140.87	384.82	26961.99
2011-11-25	1	198041.11	48.71	3.492	140.87	384.82	26961.99

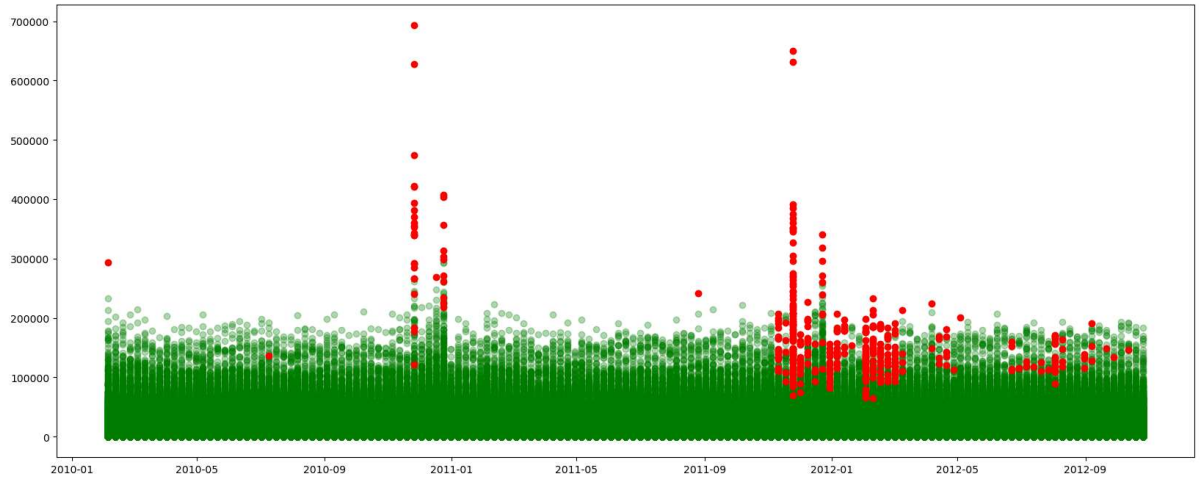
392 rows × 12 columns

In [23]:

```
# plot data
plt.figure(figsize=(20,8))
plt.scatter(df.index,df["Weekly_Sales"] ,color = "g",alpha=0.3)
# plot outlier values (where outlier values= df.iloc[outlier_index])
otlr_val = df.iloc[outlier_index]
plt.scatter(otlr_val.index,otlr_val["Weekly_Sales"], color = "r")
```

Out[23]:

```
<matplotlib.collections.PathCollection at 0x1ff42a7e3d0>
```



Analyzing KNN Results for Weekly Sales Anomaly Detection

The KNN algorithm effectively identifies anomalies in weekly sales, particularly highlighting increases in sales trends. However, it may overlook anomalies associated with small weekly sales figures. This aligns with the primary goal of detecting significant deviations to prevent stockouts. Despite limited detection of minor anomalies, KNN provides valuable insights for proactive inventory management and strategic decision-making. However, leveraging the power of isolation forest and other algorithms can significantly enhance our analysis by capturing a broader range of anomalies.

3.2. Isolation Forest

3.2.1. Univariate Isolation Forest on Weekly_sales

Here we initialize the isolation forest model with some hyperparameters assuming the proportion of outliers to be 10% of the total data (using the contamination setting) Note:The Contamination is actually a hyperparameter. Here i don't know exactly how much we expect anomalies from our data , if there's a way to know from the data engineer or from other factors it could be better.

In [22]:
```
from sklearn.ensemble import IsolationForest
model=IsolationForest(n_estimators=1000,max_samples='auto',contamination=
model.fit(df[["Weekly_Sales"]])
```

Out[22]:
```
IsolationForest
IsolationForest(contamination=0.1, n_estimators=1000, random_state=42)
```

In [23]:
```
df2 = df.copy()
df2['anomaly_scores']=model.decision_function(df2[['Weekly_Sales']])
df2['anomaly']=model.predict(df[['Weekly_Sales']])
df2.head(50)
```

Out[23]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2010-02-05	0	24924.50	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	11737.12	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	13223.76	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	37.44	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	1085.29	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	46729.77	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	21249.31	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	32229.38	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	7659.97	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	21084.08	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	5711.19	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	16930.99	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	35972.49	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	14350.83	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	1947.05	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	3508.04	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	12251.94	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	115564.35	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	64494.87	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	24146.49	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	10217.55	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	106690.06	42.31	2.572	0.0	0.0	0.0

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2010-02-05	0	50605.27	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	2293.00	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	1.16	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	39954.04	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	15694.17	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	7024.95	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	2567.36	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	3455.92	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	8589.77	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	40129.01	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	3825.78	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	8414.14	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	20837.77	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	76419.47	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	8449.54	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	439.00	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	19466.91	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	56655.39	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	32153.04	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	8366.71	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	11501.46	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	70.00	42.31	2.572	0.0	0.0	0.0

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2010-02-05	0	62424.14	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	13740.12	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	8907.63	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	11875.84	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	30052.75	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	5491.00	42.31	2.572	0.0	0.0	0.0

we can notice that every data that has a negative anomaly score will be considered anomaly.

In [24]:
```
# percantage. of Outliers(expected to be 10%)
df2[df2['anomaly']== -1].shape[0]/df.shape[0]*100
```

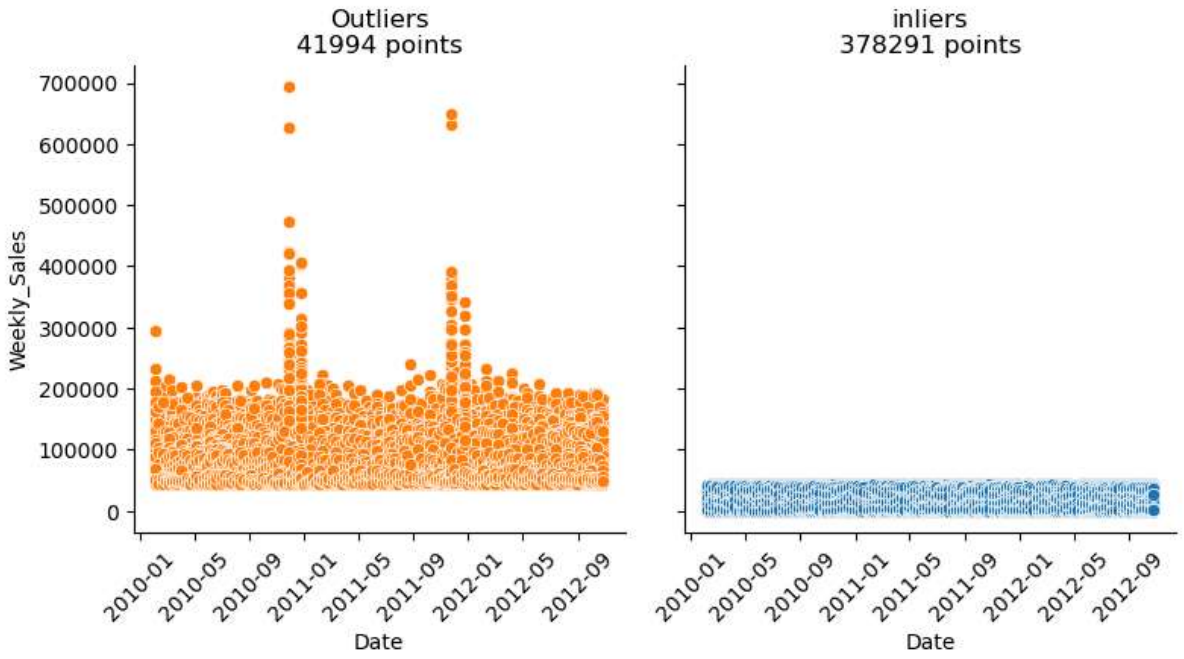
Out[24]:
```
9.991791284485528
```

In [25]:
```
###outlier_plot
def outlier_plot(data,outlier_method_name,x_var,y_var,
                 xaxis_limits=[0,1],yaxis_limits=[0,1]):
    print(f'Outlier Method: {outlier_method_name}')
    print(f"Number of Anomalous Values
{len(data[data['anomaly']==-1])}")
    print(f"Number of Non Anomalous Values
{len(data[data['anomaly']==1])}")
    print(f"Total Number Of Values {len(data)}")

    g=sns.FacetGrid(data,col='anomaly',height=4,hue='anomaly',hue_order=
[1,-1])
    g.map(sns.scatterplot,x_var,y_var)
    axes=g.axes.flatten()
    axes[0].set_title(f"Outliers\n{len(data[data['anomaly']==-1])}
points")
    axes[1].set_title(f"inliers\n{len(data[data['anomaly']==1])}
points")
    # Rotate x-axis labels
    for ax in axes:
        ax.set_xticklabels(ax.get_xticklabels(), rotation=45)
    return g
```

In [26]:
```
# Convert DatetimeIndex to a column
df2.reset_index(inplace=True)

# Plot the data
outlier_plot(df2, "Isolation Forest", "Date", "Weekly_Sales")
```

```
Outlier Method: Isolation Forest
Number of Anomalous Values 41994
Number of Non Anomalous Values 378291
Total Number Of Values 420285
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x1df0827aa90>



3.2.2. Multivariate Isolation Forest on Weekly_Sales

In [27]:
```
model=IsolationForest(n_estimators=1000,max_samples='auto',contamination=
model.fit(df)
```

Out[27]:
```
IsolationForest(contamination=0.1, max_features=12, n_estimators=1000,
                random_state=42)
```

In [28]:
```
df3=df.copy()
df3['scores']=model.decision_function(df)
df3['anomaly']=model.predict(df)
df3.head()
```

Out[28]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2010-02-05	0	24924.50	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	11737.12	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	13223.76	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	37.44	42.31	2.572	0.0	0.0	0.0
2010-02-05	0	1085.29	42.31	2.572	0.0	0.0	0.0

In [29]:

```
###let's us see some of the anomalies detected to sence it with our human interpretation
df3[df3['anomaly']== -1]
```

Out[29]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2010-02-12	1	36080.86	38.51	2.548	0.00	0.0	0.00
2010-02-12	1	36476.40	38.51	2.548	0.00	0.0	0.00
2010-02-12	1	39254.57	38.51	2.548	0.00	0.0	0.00
2010-02-12	1	35351.21	38.51	2.548	0.00	0.0	0.00
2010-02-12	1	70202.02	38.51	2.548	0.00	0.0	0.00
...	...	...	...	...	...	...	...
2012-09-07	1	128.20	75.70	3.911	11024.45	12.8	52.63
2012-09-07	1	3938.63	75.70	3.911	11024.45	12.8	52.63
2012-09-07	1	751.80	75.70	3.911	11024.45	12.8	52.63
2012-09-07	1	12927.36	75.70	3.911	11024.45	12.8	52.63
2012-09-07	1	11.94	75.70	3.911	11024.45	12.8	52.63

42029 rows × 14 columns

In [30]:

```
df3[df3['anomaly']== -1][1000:1500]
```

Out[30]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2010-02-12	1	30285.16	38.49	2.548	0.00	0.00	0.0
2010-02-12	1	42135.66	38.49	2.548	0.00	0.00	0.0
2010-02-12	1	106948.27	38.49	2.548	0.00	0.00	0.0
2010-02-12	1	65487.46	38.49	2.548	0.00	0.00	0.0
2010-02-12	1	92680.72	38.49	2.548	0.00	0.00	0.0
...	...	...	...	...	...	...	...
2012-02-03	0	40186.60	55.21	3.360	75149.79	3818.85	221.4
2012-02-03	0	4676.00	55.21	3.360	75149.79	3818.85	221.4
2012-02-03	0	6497.19	55.21	3.360	75149.79	3818.85	221.4
2012-02-03	0	31790.24	55.21	3.360	75149.79	3818.85	221.4
2012-02-03	0	3455.59	55.21	3.360	75149.79	3818.85	221.4

500 rows × 14 columns

In [31]:
```
df3[df3['anomaly']== -1][30000:30050]
```

Out[31]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2012-05-04	0	41352.37	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	16466.08	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	41826.19	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	11542.40	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	20788.25	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	5981.59	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	1868.97	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	4728.79	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	9722.20	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	35364.85	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	2091.23	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	492.12	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	14277.57	76.03	4.171	16196.98	0.0	45.99
2012-05-04	0	59953.46	76.03	4.171	16196.98	0.0	45.99
2012-05-11	0	20492.79	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	62334.08	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	18758.11	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	68.92	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	6326.17	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	92860.95	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	20359.60	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	2580.23	77.27	4.186	14847.64	392.0	84.72

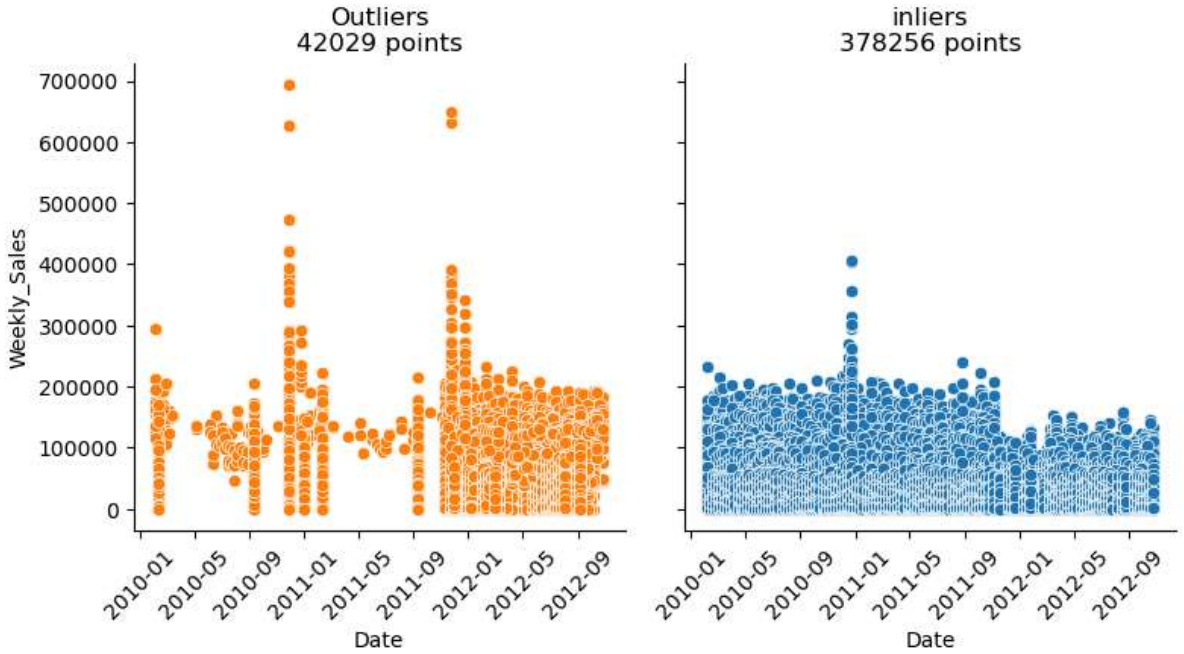
Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2012-05-11	0	6009.27	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	35.91	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	8975.09	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	19053.91	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	44612.81	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	6215.02	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	9067.04	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	12735.69	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	14839.76	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	32016.74	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	22650.74	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	43130.66	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	6858.02	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	7182.27	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	17353.68	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	1130.07	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	6994.67	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	2001.87	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	95121.98	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	20822.64	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	558.00	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	35179.59	77.27	4.186	14847.64	392.0	84.72

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2012-05-11	0	11733.75	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	3986.61	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	15758.03	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	2430.00	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	23114.22	77.27	4.186	14847.64	392.0	84.72
2012-05-11	0	16084.03	77.27	4.186	14847.64	392.0	84.72

In [32]:
```
# Convert DatetimeIndex to a column
df3.reset_index(inplace=True)

# Plot the data
outlier_plot(df3, "Isolation Forest", "Date", "Weekly_Sales")
```

```
Outlier Method: Isolation Forest
Number of Anomalous Values 42029
Number of Non Anomalous Values 378256
Total Number Of Values 420285
```

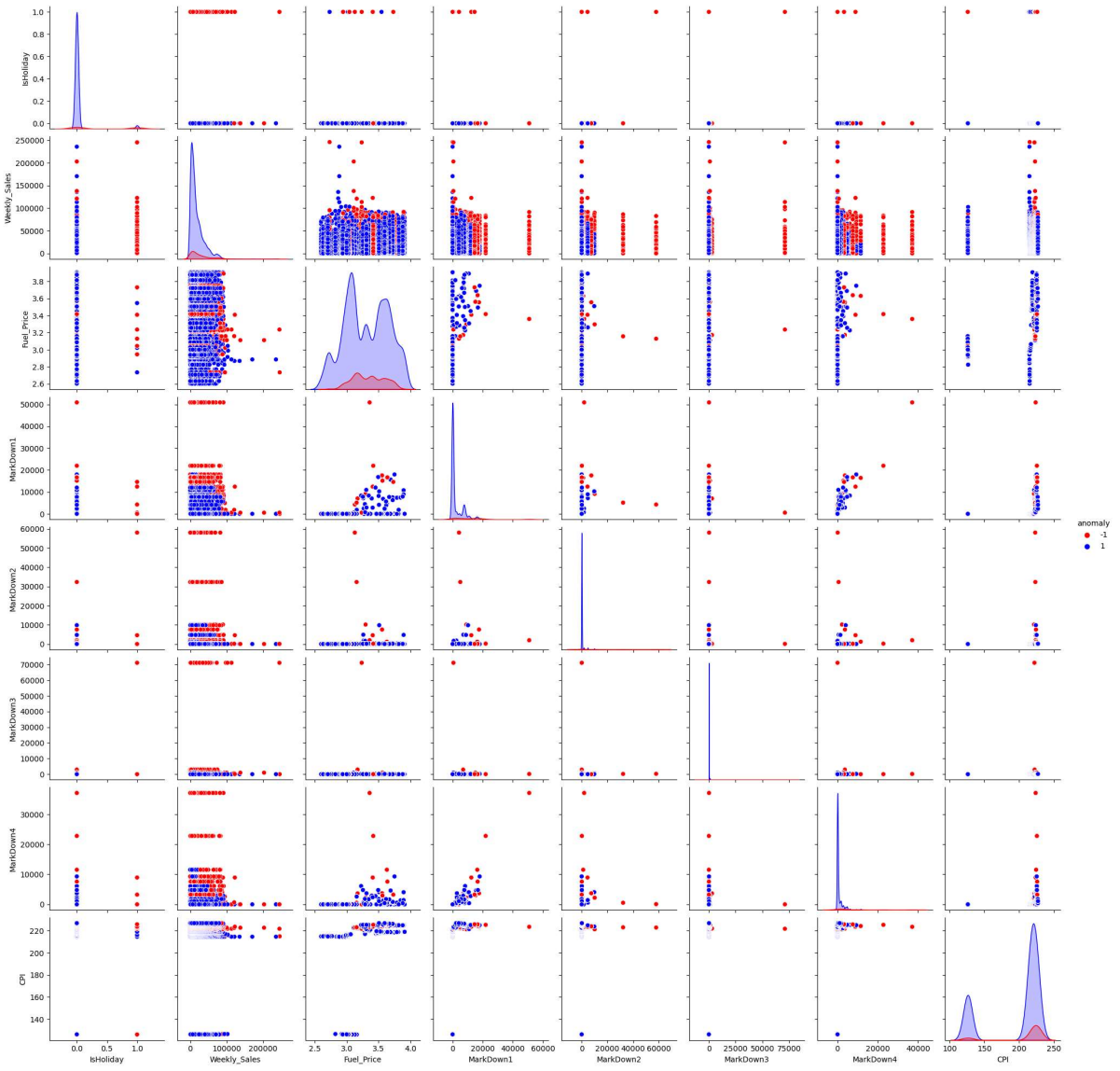
Out[32]:
```
<seaborn.axisgrid.FacetGrid at 0x1df7b0d5c50>
```



Notice that how we've got anomaly points within the main cluster , that's expected because we're plotting a 2D function for a multivariate isolation forecast So instead of looking at two-variables we can look at all the variables(features) that we have.

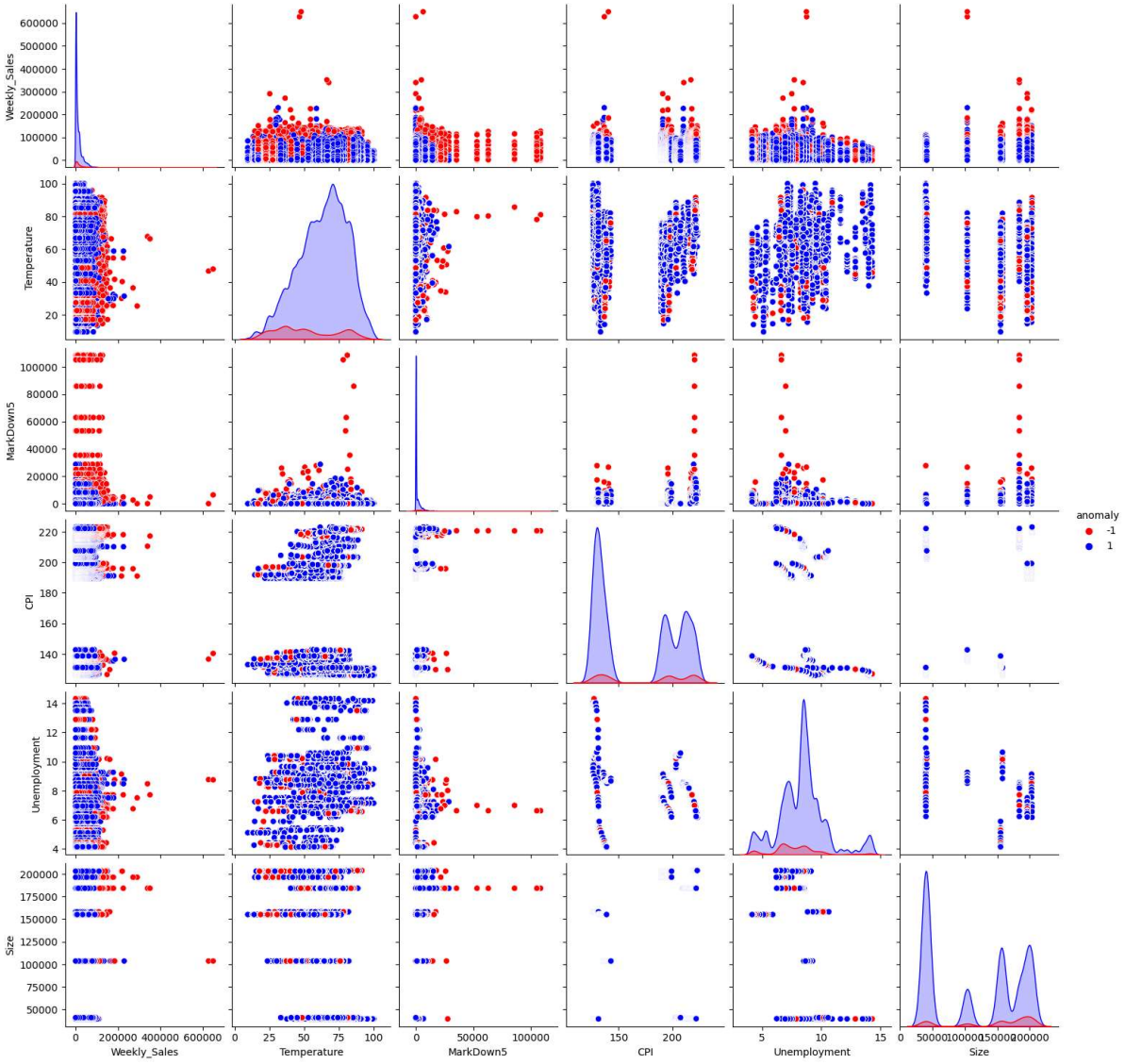
In [43]:
```
columns_to_plot = ['IsHoliday', 'Weekly_Sales', 'Fuel_Price',
'MarkDown1',
                   'MarkDown2', 'MarkDown3', 'MarkDown4', 'CPI']
sns.pairplot(data=df3[100000:110000], vars=columns_to_plot,
hue="anomaly", palette=['red','blue'])
```

Out[43]:
```
<seaborn.axisgrid.PairGrid at 0x1df30129210>
```



In [44]:
```
columns_to_plot = [ 'Weekly_Sales', 'Temperature','MarkDown5', 'CPI',
'Unemployment', 'Size']
sns.pairplot(data=df3[300000:400000], vars=columns_to_plot,
hue="anomaly", palette=['red','blue'])
```

Out[44]:
```
<seaborn.axisgrid.PairGrid at 0x1df36c61750>
```



Analyzing Isolation Forest Results for Weekly Sales Anomaly Detection

Upon reviewing the results, a discernible pattern emerges that aligns with our expectations. For instance, when examining the relationship between weekly sales and the "IsHoliday" feature, it becomes evident that the majority of anomalies occur during holiday periods, which intuitively makes sense. Similarly, there's a noticeable increase in anomalies when the markdown initiatives (1 to 5) are high, indicating a correlation between promotional activities and anomaly occurrences. Isolation Forest often outperforms other anomaly detection methods due to its ability to efficiently isolate anomalies in high-dimensional data spaces. Its effectiveness stems from its capability to identify outliers by exploiting the inherent sparsity of anomalies, making it particularly well-suited for various anomaly detection tasks across diverse datasets.

3.3. DBSCAN

In [9]:
```
from sklearn.cluster import DBSCAN
```

In [11]:
```
# DBSCAN model fitting
db = DBSCAN(eps=0.5, min_samples=5)
```

```
db.fit(df_scaled)
```

Out[11]:

▾ DBSCAN

DBSCAN()

In [12]:

```
# Adding the cluster labels to the dataframe
df4=df.copy()
df4['DBSCAN_Labels'] = db.labels_
```

In [13]:

```
unique, counts = np.unique(db.labels_, return_counts=True)
cluster_counts = dict(zip(unique, counts))

print("Cluster counts:", cluster_counts)
```

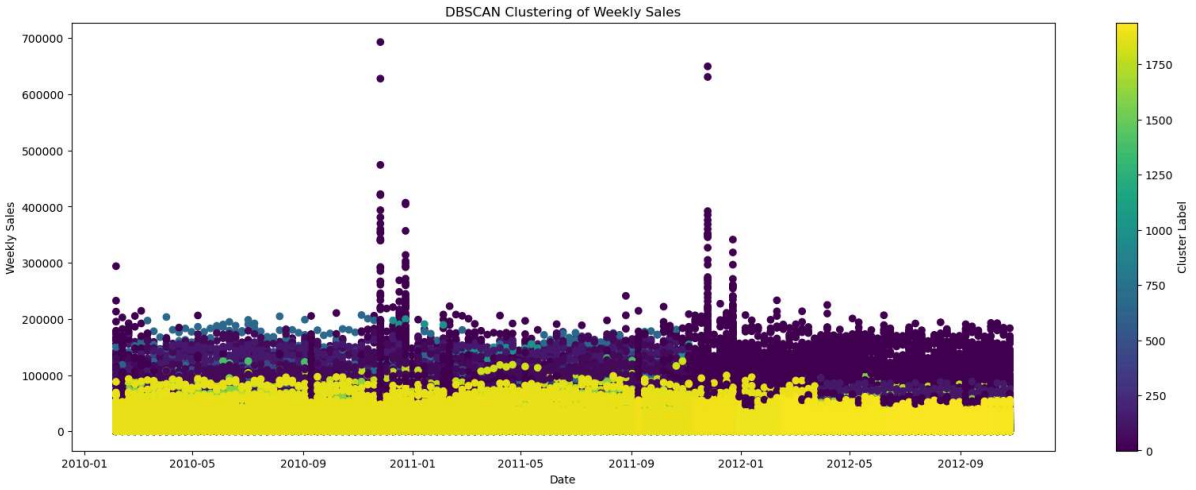
```
Cluster counts: {-1: 5683, 0: 28792, 1: 129, 2: 7, 3: 130, 4: 6, 5: 202, 6: 135,
7: 136, 8: 66, 9: 131, 10: 64, 11: 64, 12: 67, 13: 69, 14: 68, 15: 68, 16: 70, 17:
63, 18: 63, 19: 133, 20: 68, 21: 63, 22: 6, 23: 64, 24: 6, 25: 70, 26: 68, 27: 64,
28: 67, 29: 130, 30: 5, 31: 67, 32: 66, 33: 132, 34: 4, 35: 126, 36: 12, 37: 66, 3
8: 131, 39: 5, 40: 137, 41: 407, 42: 205, 43: 338, 44: 66, 45: 62, 46: 65, 47: 13
5, 48: 67, 49: 64, 50: 132, 51: 69, 52: 69, 53: 65, 54: 64, 55: 137, 56: 65, 57: 1
37, 58: 42093, 59: 203, 60: 7, 61: 350, 62: 471, 63: 550, 64: 204, 65: 350, 66: 6
3, 67: 5, 68: 70, 69: 66, 70: 67, 71: 64, 72: 4, 73: 68, 74: 276, 75: 66, 76: 68,
77: 134, 78: 8, 79: 61, 80: 280, 81: 62, 82: 62, 83: 62, 84: 7, 85: 59, 86: 5, 87:
61, 88: 7, 89: 60, 90: 8, 91: 207, 92: 257, 93: 20, 94: 9, 95: 59, 96: 141, 97: 6
5, 98: 5, 99: 409, 100: 271, 101: 69, 102: 199, 103: 7, 104: 138, 105: 61, 106: 6,
107: 343, 108: 62, 109: 6, 110: 206, 111: 60, 112: 8, 113: 1115, 114: 62, 115: 5,
116: 61, 117: 8, 118: 61, 119: 8, 120: 276, 121: 59, 122: 9, 123: 70, 124: 8, 125:
60, 126: 346, 127: 61, 128: 8, 129: 62, 130: 6, 131: 271, 132: 34944, 133: 62, 13
4: 40, 135: 264, 136: 265, 137: 604, 138: 4, 139: 24, 140: 203, 141: 64, 142: 61,
143: 65, 144: 64, 145: 64, 146: 63, 147: 62, 148: 62, 149: 62, 150: 64, 151: 60, 1
52: 63, 153: 63, 154: 61, 155: 6, 156: 61, 157: 211, 158: 30724, 159: 144, 160: 7
0, 161: 65, 162: 134, 163: 5, 164: 67, 165: 67, 166: 54, 167: 12, 168: 66, 169: 7
1, 170: 67, 171: 66, 172: 58, 173: 5, 174: 66, 175: 3, 176: 59, 177: 11, 178: 69,
179: 59, 180: 10, 181: 58, 182: 7, 183: 59, 184: 7, 185: 66, 186: 69, 187: 65, 18
8: 4, 189: 67, 190: 68, 191: 3, 192: 66, 193: 65, 194: 138, 195: 68, 196: 71, 197:
58, 198: 10, 199: 70, 200: 59, 201: 11, 202: 69, 203: 68, 204: 67, 205: 69, 206: 6
9, 207: 70, 208: 68, 209: 68, 210: 61, 211: 7, 212: 68, 213: 67, 214: 69, 215: 60,
216: 7, 217: 67, 218: 69, 219: 69, 220: 69, 221: 58, 222: 9, 223: 67, 224: 66, 22
5: 138, 226: 61, 227: 9, 228: 70, 229: 64, 230: 62, 231: 61, 232: 62, 233: 62, 23
4: 65, 235: 63, 236: 66, 237: 65, 238: 62, 239: 62, 240: 63, 241: 63, 242: 65, 24
3: 64, 244: 63, 245: 62, 246: 180, 247: 60, 248: 60, 249: 60, 250: 62, 251: 62, 25
2: 133, 253: 67, 254: 67, 255: 68, 256: 66, 257: 67, 258: 3, 259: 68, 260: 67, 26
1: 69, 262: 67, 263: 67, 264: 66, 265: 66, 266: 67, 267: 135, 268: 67, 269: 67, 27
0: 66, 271: 66, 272: 66, 273: 70, 274: 68, 275: 68, 276: 67, 277: 69, 278: 132, 27
9: 6, 280: 66, 281: 68, 282: 67, 283: 67, 284: 67, 285: 66, 286: 66, 287: 69, 288:
68, 289: 208, 290: 67, 291: 67, 292: 68, 293: 69, 294: 67, 295: 67, 296: 71, 297:
69, 298: 5686, 299: 68, 300: 133, 301: 66, 302: 68, 303: 67, 304: 67, 305: 68, 30
6: 65, 307: 69, 308: 208, 309: 68, 310: 69, 311: 68, 312: 68, 313: 70, 314: 70, 31
5: 67, 316: 70, 317: 68, 318: 69, 319: 69, 320: 69, 321: 67, 322: 69, 323: 69, 32
4: 138, 325: 406, 326: 67, 327: 67, 328: 68, 329: 68, 330: 69, 331: 67, 332: 406,
333: 67, 334: 67, 335: 68, 336: 68, 337: 137, 338: 136, 339: 69, 340: 68, 341: 67,
342: 67, 343: 68, 344: 65, 345: 65, 346: 67, 347: 66, 348: 69, 349: 67, 350: 67, 3
51: 66, 352: 67, 353: 70, 354: 68, 355: 66, 356: 67, 357: 65, 358: 67, 359: 67, 36
0: 68, 361: 66, 362: 65, 363: 67, 364: 133, 365: 137, 366: 67, 367: 68, 368: 65, 3
69: 67, 370: 65, 371: 65, 372: 65, 373: 132, 374: 6, 375: 194, 376: 7, 377: 130, 3
78: 6, 379: 67, 380: 66, 381: 66, 382: 65, 383: 67, 384: 133, 385: 68, 386: 65, 38
7: 134, 388: 134, 389: 6, 390: 66, 391: 66, 392: 61, 393: 59, 394: 57, 395: 60, 39
6: 60, 397: 10, 398: 5, 399: 62, 400: 125, 401: 62, 402: 65, 403: 62, 404: 124, 40
5: 62, 406: 63, 407: 61, 408: 61, 409: 61, 410: 61, 411: 60, 412: 62, 413: 61, 41
4: 60, 415: 60, 416: 62, 417: 61, 418: 185, 419: 61, 420: 62, 421: 61, 422: 59, 42
3: 798, 424: 244, 425: 62, 426: 61, 427: 60, 428: 61, 429: 60, 430: 61, 431: 62, 4
32: 43813, 433: 54, 434: 15, 435: 65, 436: 5, 437: 403, 438: 67, 439: 64, 440: 67,
441: 68, 442: 65, 443: 5, 444: 63, 445: 6, 446: 63, 447: 7, 448: 120, 449: 15, 45
0: 71, 451: 66, 452: 67, 453: 66, 454: 69, 455: 67, 456: 66, 457: 67, 458: 66, 45
9: 68, 460: 402, 461: 134, 462: 7, 463: 66, 464: 52, 465: 16, 466: 68, 467: 66, 46
8: 65, 469: 66, 470: 5, 471: 66, 472: 64, 473: 140, 474: 64, 475: 67, 476: 61, 47
7: 5, 478: 65, 479: 5, 480: 66, 481: 66, 482: 64, 483: 63, 484: 7, 485: 62, 486: 6
1, 487: 67, 488: 66, 489: 63, 490: 63, 491: 65, 492: 66, 493: 66, 494: 68, 495: 6
8, 496: 67, 497: 5, 498: 67, 499: 69, 500: 70, 501: 68, 502: 72, 503: 71, 504: 66,
505: 59, 506: 9, 507: 70, 508: 69, 509: 68, 510: 67, 511: 69, 512: 65, 513: 5, 51
```

```
4: 66, 515: 6, 516: 69, 517: 67, 518: 66, 519: 5, 520: 208, 521: 137, 522: 139, 52
3: 69, 524: 68, 525: 68, 526: 70, 527: 64, 528: 6, 529: 66, 530: 67, 531: 67, 532:
5770, 533: 194, 534: 67, 535: 66, 536: 67, 537: 66, 538: 66, 539: 66, 540: 69, 54
1: 65, 542: 5, 543: 12, 544: 55, 545: 63, 546: 65, 547: 67, 548: 66, 549: 68, 550:
67, 551: 67, 552: 67, 553: 67, 554: 65, 555: 66, 556: 67, 557: 67, 558: 67, 559: 6
7, 560: 66, 561: 69, 562: 68, 563: 68, 564: 67, 565: 67, 566: 67, 567: 67, 568: 6
5, 569: 66, 570: 66, 571: 69, 572: 68, 573: 67, 574: 132, 575: 67, 576: 65, 577: 6
7, 578: 68, 579: 66, 580: 66, 581: 66, 582: 65, 583: 66, 584: 66, 585: 66, 586: 6
7, 587: 64, 588: 67, 589: 277, 590: 422, 591: 6, 592: 70, 593: 64, 594: 6, 595: 6
0, 596: 9, 597: 69, 598: 69, 599: 64, 600: 7, 601: 60, 602: 5, 603: 58, 604: 8, 60
5: 70, 606: 66, 607: 62, 608: 63, 609: 6, 610: 64, 611: 6, 612: 65, 613: 5, 614: 6
6, 615: 5, 616: 63, 617: 8, 618: 65, 619: 5, 620: 6, 621: 64, 622: 6, 623: 65, 62
4: 64, 625: 6, 626: 64, 627: 6, 628: 5, 629: 63, 630: 64, 631: 6, 632: 6, 633: 62,
634: 65, 635: 6, 636: 64, 637: 5, 638: 128, 639: 9, 640: 67, 641: 64, 642: 5, 643:
137, 644: 140, 645: 63, 646: 7, 647: 61, 648: 5, 649: 69, 650: 66, 651: 5, 652: 20
7, 653: 64, 654: 5, 655: 65, 656: 5, 657: 61, 658: 7, 659: 7, 660: 63, 661: 64, 66
2: 5, 663: 65, 664: 6, 665: 63, 666: 7, 667: 64, 668: 6, 669: 65, 670: 6, 671: 64,
672: 6, 673: 62, 674: 6, 675: 12022, 676: 272, 677: 9, 678: 5, 679: 135, 680: 62,
681: 8, 682: 128, 683: 6, 684: 44, 685: 58, 686: 5, 687: 4, 688: 62, 689: 63, 690:
65, 691: 61, 692: 6, 693: 63, 694: 132, 695: 48, 696: 14, 697: 62, 698: 56, 699:
8, 700: 56, 701: 8, 702: 57, 703: 5, 704: 63, 705: 57, 706: 5, 707: 57, 708: 5, 70
9: 64, 710: 62, 711: 61, 712: 62, 713: 56, 714: 5, 715: 266, 716: 64, 717: 54, 71
8: 9, 719: 55, 720: 5, 721: 6, 722: 64, 723: 58, 724: 6, 725: 68, 726: 540, 727: 6
4, 728: 66, 729: 68, 730: 65, 731: 131, 732: 67, 733: 68, 734: 4, 735: 67, 736: 5
8, 737: 7, 738: 67, 739: 62, 740: 67, 741: 66, 742: 66, 743: 63, 744: 3, 745: 64,
746: 64, 747: 65, 748: 64, 749: 68, 750: 267, 751: 542, 752: 264, 753: 69, 754: 6
9, 755: 68, 756: 66, 757: 68, 758: 68, 759: 68, 760: 68, 761: 68, 762: 69, 763: 6
8, 764: 68, 765: 68, 766: 70, 767: 69, 768: 133, 769: 68, 770: 67, 771: 69, 772: 6
9, 773: 67, 774: 68, 775: 529, 776: 205, 777: 66, 778: 67, 779: 206, 780: 5, 781:
69, 782: 201, 783: 67, 784: 6959, 785: 196, 786: 62, 787: 65, 788: 64, 789: 66, 79
0: 66, 791: 68, 792: 67, 793: 67, 794: 68, 795: 66, 796: 65, 797: 66, 798: 65, 79
9: 66, 800: 67, 801: 66, 802: 66, 803: 65, 804: 67, 805: 66, 806: 67, 807: 68, 80
8: 5, 809: 66, 810: 67, 811: 72, 812: 135, 813: 65, 814: 66, 815: 66, 816: 66, 81
7: 66, 818: 69, 819: 3526, 820: 196, 821: 65, 822: 68, 823: 2334, 824: 68, 825: 6
7, 826: 135, 827: 67, 828: 68, 829: 70, 830: 61, 831: 7, 832: 68, 833: 66, 834: 6
6, 835: 65, 836: 127, 837: 12, 838: 62, 839: 5, 840: 67, 841: 70, 842: 68, 843: 6
2, 844: 5, 845: 68, 846: 68, 847: 69, 848: 70, 849: 69, 850: 64, 851: 550, 852: 6
4, 853: 202, 854: 340, 855: 133, 856: 68, 857: 66, 858: 63, 859: 4, 860: 61, 861:
5, 862: 65, 863: 68, 864: 69, 865: 68, 866: 196, 867: 66, 868: 63, 869: 5, 870: 6
7, 871: 62, 872: 8, 873: 8, 874: 61, 875: 68, 876: 64, 877: 71, 878: 64, 879: 6, 8
80: 64, 881: 6, 882: 65, 883: 5, 884: 66, 885: 6, 886: 64, 887: 6, 888: 63, 889:
5, 890: 63, 891: 5, 892: 64, 893: 6, 894: 63, 895: 6, 896: 6, 897: 65, 898: 5, 89
9: 70, 900: 65, 901: 6, 902: 138, 903: 66, 904: 5, 905: 65, 906: 5, 907: 5, 908: 6
4, 909: 5, 910: 66, 911: 5, 912: 68, 913: 65, 914: 5, 915: 65, 916: 5, 917: 68, 91
8: 65, 919: 6, 920: 64, 921: 5, 922: 61, 923: 7, 924: 64, 925: 6, 926: 64, 927: 5,
928: 6, 929: 61, 930: 202, 931: 6, 932: 62, 933: 5, 934: 200, 935: 7, 936: 421, 93
7: 208, 938: 66, 939: 68, 940: 66, 941: 66, 942: 65, 943: 67, 944: 134, 945: 65, 9
46: 67, 947: 63, 948: 66, 949: 132, 950: 64, 951: 66, 952: 67, 953: 68, 954: 68, 9
55: 66, 956: 135, 957: 754, 958: 66, 959: 68, 960: 140, 961: 68, 962: 69, 963: 34
4, 964: 66, 965: 136, 966: 68, 967: 68, 968: 68, 969: 65, 970: 474, 971: 5, 972: 6
6, 973: 66, 974: 68, 975: 70, 976: 68, 977: 67, 978: 63, 979: 5, 980: 69, 981: 69,
982: 6, 983: 6, 984: 63, 985: 6, 986: 132, 987: 7, 988: 61, 989: 5, 990: 21, 991:
65, 992: 5, 993: 63, 994: 3, 995: 60, 996: 6, 997: 65, 998: 6, 999: 61, 1000: 67,
1001: 61, 1002: 6, 1003: 65, 1004: 5, 1005: 66, 1006: 64, 1007: 65, 1008: 58, 100
9: 6, 1010: 60, 1011: 5, 1012: 62, 1013: 64, 1014: 61, 1015: 5, 1016: 62, 1017: 6
2, 1018: 63, 1019: 5, 1020: 60, 1021: 65, 1022: 66, 1023: 69, 1024: 62, 1025: 64,
```

```
1026: 64, 1027: 5, 1028: 68, 1029: 63, 1030: 6, 1031: 135, 1032: 203, 1033: 67, 10
34: 61, 1035: 5, 1036: 66, 1037: 62, 1038: 5, 1039: 63, 1040: 5, 1041: 7, 1042: 6
2, 1043: 63, 1044: 5, 1045: 63, 1046: 4, 1047: 61, 1048: 6, 1049: 67, 1050: 68, 10
51: 63, 1052: 126, 1053: 13, 1054: 62, 1055: 63, 1056: 6, 1057: 64, 1058: 5, 1059:
64, 1060: 63, 1061: 67, 1062: 68, 1063: 64, 1064: 68, 1065: 68, 1066: 68, 1067: 6
7, 1068: 67, 1069: 69, 1070: 66, 1071: 68, 1072: 67, 1073: 69, 1074: 68, 1075: 68,
1076: 68, 1077: 68, 1078: 67, 1079: 67, 1080: 67, 1081: 67, 1082: 67, 1083: 68, 10
84: 67, 1085: 69, 1086: 66, 1087: 65, 1088: 66, 1089: 68, 1090: 67, 1091: 67, 109
2: 67, 1093: 68, 1094: 67, 1095: 67, 1096: 62, 1097: 5, 1098: 62, 1099: 65, 1100:
60, 1101: 7, 1102: 68, 1103: 66, 1104: 68, 1105: 66, 1106: 67, 1107: 59, 1108: 5,
1109: 62, 1110: 5, 1111: 61, 1112: 5, 1113: 62, 1114: 61, 1115: 65, 1116: 62, 111
7: 64, 1118: 127, 1119: 10, 1120: 62, 1121: 5, 1122: 67, 1123: 68, 1124: 5, 1125:
62, 1126: 62, 1127: 5, 1128: 59, 1129: 7, 1130: 62, 1131: 65, 1132: 67, 1133: 65,
1134: 66, 1135: 62, 1136: 5, 1137: 133, 1138: 60, 1139: 8, 1140: 65, 1141: 61, 114
2: 6, 1143: 5, 1144: 64, 1145: 5, 1146: 70, 1147: 67, 1148: 3528, 1149: 66, 1150:
5, 1151: 61, 1152: 5, 1153: 138, 1154: 2405, 1155: 65, 1156: 5, 1157: 69, 1158: 6
9, 1159: 65, 1160: 67, 1161: 66, 1162: 63, 1163: 6, 1164: 66, 1165: 70, 1166: 67,
1167: 4, 1168: 64, 1169: 64, 1170: 6, 1171: 63, 1172: 6, 1173: 65, 1174: 67, 1175:
70, 1176: 66, 1177: 67, 1178: 137, 1179: 66, 1180: 4, 1181: 70, 1182: 70, 1183: 7
1, 1184: 70, 1185: 139, 1186: 68, 1187: 141, 1188: 68, 1189: 207, 1190: 70, 1191:
69, 1192: 208, 1193: 67, 1194: 70, 1195: 63, 1196: 4, 1197: 67, 1198: 69, 1199: 6
3, 1200: 6, 1201: 65, 1202: 5, 1203: 70, 1204: 66, 1205: 140, 1206: 69, 1207: 69,
1208: 70, 1209: 66, 1210: 67, 1211: 69, 1212: 68, 1213: 67, 1214: 67, 1215: 67, 12
16: 68, 1217: 67, 1218: 67, 1219: 67, 1220: 67, 1221: 68, 1222: 68, 1223: 66, 122
4: 69, 1225: 71, 1226: 69, 1227: 68, 1228: 70, 1229: 69, 1230: 68, 1231: 66, 1232:
133, 1233: 67, 1234: 66, 1235: 69, 1236: 70, 1237: 68, 1238: 69, 1239: 68, 1240: 6
6, 1241: 70, 1242: 69, 1243: 69, 1244: 68, 1245: 61, 1246: 8, 1247: 69, 1248: 69,
1249: 69, 1250: 68, 1251: 136, 1252: 69, 1253: 68, 1254: 274, 1255: 68, 1256: 67,
1257: 138, 1258: 70, 1259: 68, 1260: 70, 1261: 70, 1262: 67, 1263: 70, 1264: 68, 1
265: 69, 1266: 67, 1267: 67, 1268: 68, 1269: 70, 1270: 70, 1271: 69, 1272: 69, 127
3: 69, 1274: 70, 1275: 68, 1276: 68, 1277: 68, 1278: 136, 1279: 69, 1280: 137, 128
1: 68, 1282: 205, 1283: 138, 1284: 134, 1285: 68, 1286: 71, 1287: 207, 1288: 68, 1
289: 138, 1290: 68, 1291: 133, 1292: 67, 1293: 69, 1294: 69, 1295: 68, 1296: 69, 1
297: 67, 1298: 69, 1299: 69, 1300: 69, 1301: 62, 1302: 63, 1303: 64, 1304: 62, 130
5: 133, 1306: 65, 1307: 63, 1308: 65, 1309: 65, 1310: 66, 1311: 65, 1312: 130, 131
3: 67, 1314: 65, 1315: 65, 1316: 66, 1317: 65, 1318: 65, 1319: 67, 1320: 66, 1321:
65, 1322: 65, 1323: 65, 1324: 199, 1325: 728, 1326: 65, 1327: 65, 1328: 5, 1329: 6
1, 1330: 66, 1331: 65, 1332: 193, 1333: 65, 1334: 64, 1335: 66, 1336: 66, 1337: 6
6, 1338: 65, 1339: 66, 1340: 65, 1341: 66, 1342: 64, 1343: 6, 1344: 65, 1345: 64,
1346: 68, 1347: 66, 1348: 64, 1349: 6, 1350: 63, 1351: 66, 1352: 68, 1353: 66, 135
4: 64, 1355: 67, 1356: 69, 1357: 65, 1358: 4, 1359: 70, 1360: 68, 1361: 71, 1362:
67, 1363: 64, 1364: 69, 1365: 64, 1366: 65, 1367: 5, 1368: 6, 1369: 66, 1370: 69,
1371: 68, 1372: 139, 1373: 65, 1374: 65, 1375: 4, 1376: 64, 1377: 66, 1378: 65, 13
79: 66, 1380: 67, 1381: 67, 1382: 69, 1383: 6, 1384: 63, 1385: 68, 1386: 67, 1387:
69, 1388: 69, 1389: 6009, 1390: 207, 1391: 68, 1392: 68, 1393: 5, 1394: 64, 1395:
69, 1396: 68, 1397: 67, 1398: 69, 1399: 71, 1400: 141, 1401: 71, 1402: 64, 1403: 1
38, 1404: 69, 1405: 67, 1406: 70, 1407: 69, 1408: 67, 1409: 61, 1410: 8, 1411: 70,
1412: 69, 1413: 68, 1414: 69, 1415: 66, 1416: 69, 1417: 69, 1418: 68, 1419: 67, 14
20: 137, 1421: 69, 1422: 138, 1423: 67, 1424: 68, 1425: 68, 1426: 67, 1427: 69, 14
28: 68, 1429: 66, 1430: 65, 1431: 63, 1432: 69, 1433: 137, 1434: 68, 1435: 65, 143
6: 69, 1437: 68, 1438: 66, 1439: 68, 1440: 67, 1441: 68, 1442: 66, 1443: 63, 1444:
130, 1445: 67, 1446: 67, 1447: 64, 1448: 65, 1449: 133, 1450: 65, 1451: 65, 1452:
65, 1453: 67, 1454: 133, 1455: 131, 1456: 64, 1457: 65, 1458: 64, 1459: 67, 1460:
67, 1461: 65, 1462: 384, 1463: 130, 1464: 65, 1465: 66, 1466: 198, 1467: 256, 146
8: 67, 1469: 66, 1470: 66, 1471: 66, 1472: 67, 1473: 259, 1474: 64, 1475: 130, 147
6: 48, 1477: 50, 1478: 49, 1479: 47, 1480: 48, 1481: 46, 1482: 61, 1483: 6, 1484:
```

61, 1485: 5, 1486: 56, 1487: 10, 1488: 67, 1489: 67, 1490: 68, 1491: 61, 1492: 5,
1493: 8, 1494: 59, 1495: 63, 1496: 62, 1497: 6, 1498: 67, 1499: 68, 1500: 61, 150
1: 5, 1502: 62, 1503: 6, 1504: 64, 1505: 5, 1506: 66, 1507: 62, 1508: 61, 1509: 5,
1510: 60, 1511: 5, 1512: 124, 1513: 11, 1514: 61, 1515: 5, 1516: 63, 1517: 5, 151
8: 61, 1519: 5, 1520: 61, 1521: 6, 1522: 61, 1523: 5, 1524: 62, 1525: 6, 1526: 60,
1527: 7, 1528: 62, 1529: 6, 1530: 64, 1531: 61, 1532: 6, 1533: 61, 1534: 5, 1535:
62, 1536: 7, 1537: 63, 1538: 6, 1539: 69, 1540: 66, 1541: 60, 1542: 7, 1543: 69, 1
544: 67, 1545: 66, 1546: 71, 1547: 61, 1548: 62, 1549: 5, 1550: 62, 1551: 65, 155
2: 63, 1553: 6, 1554: 63, 1555: 7, 1556: 9, 1557: 62, 1558: 62, 1559: 7, 1560: 63,
1561: 5, 1562: 63, 1563: 62, 1564: 6, 1565: 65, 1566: 64, 1567: 3, 1568: 62, 1569:
5, 1570: 65, 1571: 5, 1572: 68, 1573: 7, 1574: 207, 1575: 69, 1576: 68, 1577: 63,
1578: 5, 1579: 69, 1580: 68, 1581: 62, 1582: 7, 1583: 68, 1584: 69, 1585: 60, 158
6: 8, 1587: 60, 1588: 8, 1589: 62, 1590: 6, 1591: 69, 1592: 187, 1593: 20, 1594: 6
3, 1595: 7, 1596: 64, 1597: 5, 1598: 69, 1599: 7845, 1600: 84, 1601: 80, 1602: 8,
1603: 267, 1604: 91, 1605: 45, 1606: 11125, 1607: 191, 1608: 97, 1609: 46, 1610: 5
184, 1611: 69, 1612: 69, 1613: 70, 1614: 143, 1615: 987, 1616: 69, 1617: 68, 1618:
68, 1619: 67, 1620: 70, 1621: 69, 1622: 138, 1623: 3, 1624: 71, 1625: 68, 1626: 6
8, 1627: 68, 1628: 67, 1629: 68, 1630: 69, 1631: 70, 1632: 69, 1633: 69, 1634: 68,
1635: 136, 1636: 69, 1637: 67, 1638: 68, 1639: 68, 1640: 67, 1641: 68, 1642: 70, 1
643: 206, 1644: 66, 1645: 67, 1646: 67, 1647: 412, 1648: 12, 1649: 71, 1650: 70, 1
651: 70, 1652: 69, 1653: 68, 1654: 69, 1655: 69, 1656: 68, 1657: 6, 1658: 69, 165
9: 68, 1660: 135, 1661: 64, 1662: 65, 1663: 64, 1664: 65, 1665: 64, 1666: 65, 166
7: 60, 1668: 64, 1669: 66, 1670: 64, 1671: 64, 1672: 65, 1673: 66, 1674: 64, 1675:
66, 1676: 64, 1677: 67, 1678: 64, 1679: 63, 1680: 65, 1681: 63, 1682: 65, 1683: 6
4, 1684: 64, 1685: 67, 1686: 63, 1687: 66, 1688: 64, 1689: 67, 1690: 64, 1691: 66,
1692: 65, 1693: 60, 1694: 65, 1695: 64, 1696: 64, 1697: 66, 1698: 64, 1699: 92, 17
00: 94, 1701: 44, 1702: 4349, 1703: 152, 1704: 49, 1705: 45, 1706: 48, 1707: 782,
1708: 99, 1709: 2, 1710: 49, 1711: 49, 1712: 258, 1713: 1201, 1714: 10, 1715: 5, 1
716: 48, 1717: 163, 1718: 66, 1719: 67, 1720: 66, 1721: 65, 1722: 63, 1723: 64, 17
24: 64, 1725: 63, 1726: 64, 1727: 66, 1728: 65, 1729: 132, 1730: 66, 1731: 66, 173
2: 67, 1733: 65, 1734: 66, 1735: 64, 1736: 66, 1737: 66, 1738: 67, 1739: 65, 1740:
65, 1741: 67, 1742: 65, 1743: 66, 1744: 64, 1745: 65, 1746: 65, 1747: 68, 1748: 6
5, 1749: 64, 1750: 67, 1751: 64, 1752: 67, 1753: 65, 1754: 67, 1755: 58, 1756: 6,
1757: 67, 1758: 65, 1759: 66, 1760: 66, 1761: 60, 1762: 6, 1763: 66, 1764: 66, 176
5: 65, 1766: 64, 1767: 66, 1768: 2771, 1769: 66, 1770: 65, 1771: 775, 1772: 139, 1
773: 2345, 1774: 70, 1775: 68, 1776: 68, 1777: 67, 1778: 68, 1779: 69, 1780: 66, 1
781: 133, 1782: 66, 1783: 66, 1784: 67, 1785: 67, 1786: 66, 1787: 67, 1788: 68, 17
89: 68, 1790: 68, 1791: 66, 1792: 133, 1793: 541, 1794: 68, 1795: 69, 1796: 333, 1
797: 10, 1798: 69, 1799: 68, 1800: 69, 1801: 68, 1802: 66, 1803: 67, 1804: 67, 180
5: 67, 1806: 67, 1807: 65, 1808: 68, 1809: 67, 1810: 67, 1811: 68, 1812: 68, 1813:
68, 1814: 199, 1815: 66, 1816: 10, 1817: 69, 1818: 69, 1819: 67, 1820: 66, 1821: 6
8, 1822: 68, 1823: 67, 1824: 66, 1825: 69, 1826: 64, 1827: 3, 1828: 69, 1829: 135,
1830: 62, 1831: 6, 1832: 69, 1833: 70, 1834: 68, 1835: 70, 1836: 68, 1837: 68, 183
8: 67, 1839: 274, 1840: 69, 1841: 68, 1842: 61, 1843: 6, 1844: 69, 1845: 68, 1846:
138, 1847: 68, 1848: 138, 1849: 67, 1850: 69, 1851: 69, 1852: 69, 1853: 68, 1854:
70, 1855: 68, 1856: 67, 1857: 63, 1858: 6, 1859: 62, 1860: 7, 1861: 68, 1862: 137,
1863: 68, 1864: 68, 1865: 68, 1866: 69, 1867: 4, 1868: 42, 1869: 4556, 1870: 39, 1
871: 38, 1872: 45, 1873: 82, 1874: 6, 1875: 40, 1876: 47, 1877: 45, 1878: 42, 187
9: 281, 1880: 43, 1881: 47, 1882: 50, 1883: 146, 1884: 51, 1885: 46, 1886: 46, 188
7: 50, 1888: 51, 1889: 5762, 1890: 67, 1891: 67, 1892: 65, 1893: 134, 1894: 67, 18
95: 68, 1896: 67, 1897: 65, 1898: 68, 1899: 68, 1900: 66, 1901: 68, 1902: 67, 190
3: 64, 1904: 63, 1905: 68, 1906: 66, 1907: 67, 1908: 68, 1909: 68, 1910: 68, 1911:
68, 1912: 67, 1913: 69, 1914: 334, 1915: 67, 1916: 66, 1917: 66, 1918: 68, 1919: 6
6, 1920: 65, 1921: 133, 1922: 67, 1923: 63, 1924: 65, 1925: 66, 1926: 65, 1927: 6
4, 1928: 63, 1929: 65, 1930: 67, 1931: 68, 1932: 66, 1933: 64, 1934: 67, 1935: 68,
1936: 66, 1937: 65, 1938: 64, 1939: 134}

In [14]:
```
plt.figure(figsize=(20, 7))
plt.scatter(data['Date'], df4['Weekly_Sales'], c=df4['DBSCAN_Labels'],
cmap='viridis')
plt.title('DBSCAN Clustering of Weekly Sales')
plt.xlabel('Date')
plt.ylabel('Weekly Sales')
plt.colorbar(label='Cluster Label')
plt.show()
```



In [15]:
```
pd.value_counts(df4['DBSCAN_Labels'])
```

Out[15]:
```
DBSCAN_Labels
432     43813
58      42093
132     34944
158     30724
0       28792
        ...
994         3
175         3
191         3
744         3
1709        2
Name: count, Length: 1941, dtype: int64
```

In [15]:
```
pd.value_counts(df4['DBSCAN_Labels']==-1)
```

Out[15]:
```
DBSCAN_Labels
False    414602
True       5683
Name: count, dtype: int64
```

In [17]:
```
df4[df4['DBSCAN_Labels'] == -1]
```

Out[17]:

Date	IsHoliday	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
2010-02-05	0	139884.94	42.31	2.572	0.00	0.00	0.00
2010-02-12	1	94136.35	38.51	2.548	0.00	0.00	0.00
2010-02-12	1	111390.36	38.51	2.548	0.00	0.00	0.00
2010-02-12	1	143081.42	38.51	2.548	0.00	0.00	0.00
2010-02-19	0	135066.75	39.93	2.514	0.00	0.00	0.00
...	...	...	...	...	...	...	...
2012-09-14	0	53711.96	67.87	3.948	11407.95	0.00	4.30
2012-09-14	0	51602.46	67.87	3.948	11407.95	0.00	4.30
2012-09-21	0	50977.95	65.32	4.038	8452.20	92.28	63.24
2012-09-21	0	58339.44	65.32	4.038	8452.20	92.28	63.24
2012-09-21	0	54160.50	65.32	4.038	8452.20	92.28	63.24

5683 rows × 13 columns

In [18]:
```
# percantage. of Outliers
sum(df4['DBSCAN_Labels'] == -1)/df.shape[0]*100
```

Out[18]:
```
1.3521776889491655
```

In [33]:
```
### let us now choose another eps and min_samples
db = DBSCAN(eps=0.4, min_samples=10)
db.fit(df_scaled)
```

Out[33]:
```
DBSCAN
DBSCAN(eps=0.4, min_samples=10)
```

In [34]:
```
df5=df.copy()
df5['DBSCAN_Labels'] = db.labels_
```

In [35]:
```
unique, counts = np.unique(db.labels_, return_counts=True)
cluster_counts = dict(zip(unique, counts))
```

```
print("Cluster counts:", cluster_counts)
```

```
Cluster counts: {-1: 19694, 0: 9371, 1: 129, 2: 23, 3: 39, 4: 130, 5: 122, 6: 131,
7: 125, 8: 4714, 9: 28, 10: 130, 11: 64, 12: 64, 13: 62, 14: 63, 15: 64, 16: 61, 1
7: 193, 18: 63, 19: 63, 20: 62, 21: 65, 22: 56, 23: 57, 24: 62, 25: 61, 26: 61, 2
7: 61, 28: 61, 29: 61, 30: 62, 31: 63, 32: 14, 33: 47, 34: 129, 35: 126, 36: 50, 3
7: 11, 38: 130, 39: 122, 40: 63, 41: 62, 42: 129, 43: 132, 44: 60, 45: 61, 46: 63,
47: 63, 48: 63, 49: 61, 50: 62, 51: 64, 52: 62, 53: 54, 54: 7, 55: 11, 56: 53, 57:
12, 58: 52, 59: 56, 60: 117, 61: 65, 62: 62, 63: 63, 64: 24027, 65: 121, 66: 46, 6
7: 34, 68: 197, 69: 132, 70: 29, 71: 134, 72: 122, 73: 14431, 74: 28, 75: 253, 76:
15, 77: 60, 78: 60, 79: 56, 80: 58, 81: 58, 82: 54, 83: 112, 84: 17, 85: 60, 86: 5
8, 87: 57, 88: 59, 89: 189, 90: 60, 91: 62, 92: 58, 93: 58, 94: 60, 95: 60, 96: 12
5, 97: 60, 98: 59, 99: 53, 100: 60, 101: 58, 102: 61, 103: 58, 104: 127, 105: 119,
106: 11, 107: 51, 108: 8, 109: 118, 110: 60, 111: 59, 112: 60, 113: 248, 114: 52,
115: 9, 116: 123, 117: 61, 118: 61, 119: 60, 120: 122, 121: 57, 122: 57, 123: 56,
124: 59, 125: 60, 126: 127, 127: 61, 128: 61, 129: 59, 130: 187, 131: 12, 132: 312
09, 133: 62, 134: 13, 135: 62, 136: 59, 137: 61, 138: 60, 139: 13, 140: 197, 141:
63, 142: 61, 143: 65, 144: 64, 145: 64, 146: 63, 147: 62, 148: 62, 149: 62, 150: 6
4, 151: 60, 152: 62, 153: 62, 154: 64, 155: 64, 156: 61, 157: 123, 158: 62, 159: 6
1, 160: 62, 161: 24633, 162: 58, 163: 59, 164: 54, 165: 58, 166: 345, 167: 55, 16
8: 113, 169: 15, 170: 57, 171: 4371, 172: 27, 173: 59, 174: 205, 175: 129, 176: 5
6, 177: 54, 178: 52, 179: 13, 180: 52, 181: 47, 182: 49, 183: 10, 184: 46, 185: 1
0, 186: 52, 187: 12, 188: 58, 189: 58, 190: 57, 191: 58, 192: 58, 193: 55, 194: 5
6, 195: 56, 196: 58, 197: 52, 198: 49, 199: 8, 200: 57, 201: 117, 202: 48, 203: 1
0, 204: 60, 205: 57, 206: 53, 207: 58, 208: 57, 209: 44, 210: 10, 211: 7, 212: 48,
213: 7, 214: 50, 215: 7, 216: 46, 217: 11, 218: 46, 219: 10, 220: 5, 221: 44, 222:
14, 223: 47, 224: 8, 225: 59, 226: 57, 227: 55, 228: 56, 229: 56, 230: 57, 231: 5
8, 232: 56, 233: 59, 234: 58, 235: 59, 236: 48, 237: 49, 238: 9, 239: 58, 240: 54,
241: 53, 242: 64, 243: 16, 244: 61, 245: 61, 246: 62, 247: 62, 248: 125, 249: 61,
250: 62, 251: 62, 252: 65, 253: 63, 254: 66, 255: 132, 256: 65, 257: 62, 258: 62,
259: 63, 260: 63, 261: 65, 262: 64, 263: 63, 264: 62, 265: 314, 266: 123, 267: 62,
268: 121, 269: 669, 270: 60, 271: 60, 272: 119, 273: 60, 274: 60, 275: 62, 276: 6
2, 277: 63, 278: 67, 279: 131, 280: 190, 281: 66, 282: 65, 283: 57, 284: 64, 285:
66, 286: 50, 287: 15, 288: 63, 289: 64, 290: 65, 291: 49, 292: 10, 293: 66, 294: 5
6, 295: 57, 296: 58, 297: 63, 298: 59, 299: 63, 300: 57, 301: 10, 302: 61, 303: 6,
304: 60, 305: 58, 306: 9, 307: 62, 308: 58, 309: 8, 310: 64, 311: 56, 312: 10, 31
3: 66, 314: 58, 315: 10, 316: 64, 317: 67, 318: 132, 319: 65, 320: 65, 321: 66, 32
2: 132, 323: 66, 324: 62, 325: 56, 326: 10, 327: 67, 328: 67, 329: 57, 330: 9, 33
1: 66, 332: 66, 333: 66, 334: 57, 335: 57, 336: 57, 337: 10, 338: 67, 339: 58, 34
0: 9, 341: 64, 342: 57, 343: 9, 344: 66, 345: 69, 346: 67, 347: 2926, 348: 63, 34
9: 132, 350: 201, 351: 65, 352: 63, 353: 65, 354: 2484, 355: 66, 356: 67, 357: 68,
358: 65, 359: 69, 360: 68, 361: 65, 362: 67, 363: 67, 364: 64, 365: 64, 366: 64, 3
67: 66, 368: 66, 369: 64, 370: 66, 371: 65, 372: 66, 373: 64, 374: 64, 375: 63, 37
6: 66, 377: 63, 378: 138, 379: 65, 380: 63, 381: 67, 382: 265, 383: 63, 384: 63, 3
85: 63, 386: 64, 387: 124, 388: 62, 389: 193, 390: 62, 391: 64, 392: 64, 393: 64,
394: 130, 395: 67, 396: 66, 397: 67, 398: 67, 399: 198, 400: 65, 401: 9, 402: 316
8, 403: 66, 404: 62, 405: 63, 406: 63, 407: 130, 408: 1250, 409: 30, 410: 16, 411:
132, 412: 965, 413: 64, 414: 64, 415: 63, 416: 64, 417: 63, 418: 65, 419: 64, 420:
61, 421: 66, 422: 67, 423: 64, 424: 65, 425: 63, 426: 62, 427: 65, 428: 64, 429: 6
5, 430: 63, 431: 63, 432: 67, 433: 66, 434: 64, 435: 67, 436: 68, 437: 65, 438: 6
5, 439: 66, 440: 65, 441: 65, 442: 65, 443: 66, 444: 130, 445: 66, 446: 64, 447: 6
7, 448: 66, 449: 66, 450: 64, 451: 65, 452: 66, 453: 67, 454: 67, 455: 62, 456: 6
5, 457: 65, 458: 63, 459: 63, 460: 65, 461: 64, 462: 64, 463: 65, 464: 2917, 465:
61, 466: 17, 467: 57, 468: 56, 469: 59, 470: 59, 471: 118, 472: 1895, 473: 183, 47
4: 60, 475: 13, 476: 59, 477: 61, 478: 62, 479: 62, 480: 64, 481: 62, 482: 123, 48
3: 62, 484: 63, 485: 61, 486: 60, 487: 61, 488: 59, 489: 59, 490: 61, 491: 59, 49
2: 59, 493: 59, 494: 61, 495: 59, 496: 61, 497: 60, 498: 60, 499: 61, 500: 60, 50
1: 59, 502: 60, 503: 61, 504: 245, 505: 246, 506: 58, 507: 60, 508: 59, 509: 122,
```

```
510: 61, 511: 60, 512: 59, 513: 60, 514: 59, 515: 60, 516: 59, 517: 62, 518: 123,
519: 17381, 520: 54, 521: 10, 522: 64, 523: 131, 524: 52, 525: 11, 526: 11904, 52
7: 58, 528: 53, 529: 60, 530: 56, 531: 56, 532: 7, 533: 51, 534: 50, 535: 19, 536:
35, 537: 57, 538: 59, 539: 60, 540: 53, 541: 11, 542: 66, 543: 62, 544: 51, 545:
9, 546: 55, 547: 52, 548: 54, 549: 10, 550: 56, 551: 58, 552: 54, 553: 51, 554: 1
1, 555: 52, 556: 52, 557: 52, 558: 52, 559: 50, 560: 11, 561: 52, 562: 61, 563: 5
4, 564: 54, 565: 51, 566: 51, 567: 8, 568: 50, 569: 8, 570: 60, 571: 53, 572: 6, 5
73: 57, 574: 59, 575: 57, 576: 57, 577: 58, 578: 56, 579: 62, 580: 59, 581: 60, 58
2: 55, 583: 6, 584: 60, 585: 56, 586: 56, 587: 57, 588: 6, 589: 54, 590: 55, 591:
10, 592: 63, 593: 64, 594: 128, 595: 60, 596: 64, 597: 64, 598: 63, 599: 61, 600:
59, 601: 59, 602: 64, 603: 62, 604: 62, 605: 64, 606: 64, 607: 61, 608: 63, 609: 6
5, 610: 63, 611: 127, 612: 65, 613: 62, 614: 63, 615: 63, 616: 64, 617: 62, 618: 6
3, 619: 65, 620: 65, 621: 63, 622: 64, 623: 65, 624: 63, 625: 64, 626: 62, 627: 6
3, 628: 62, 629: 62, 630: 63, 631: 122, 632: 64, 633: 64, 634: 63, 635: 63, 636: 6
3, 637: 3385, 638: 60, 639: 20, 640: 60, 641: 128, 642: 57, 643: 59, 644: 330, 64
5: 1579, 646: 16, 647: 60, 648: 268, 649: 57, 650: 6, 651: 58, 652: 59, 653: 66, 6
54: 59, 655: 57, 656: 10, 657: 55, 658: 63, 659: 60, 660: 57, 661: 57, 662: 61, 66
3: 60, 664: 59, 665: 61, 666: 57, 667: 56, 668: 59, 669: 60, 670: 60, 671: 59, 67
2: 60, 673: 56, 674: 58, 675: 60, 676: 59, 677: 59, 678: 57, 679: 59, 680: 8, 681:
58, 682: 56, 683: 57, 684: 58, 685: 60, 686: 60, 687: 58, 688: 58, 689: 59, 690: 5
7, 691: 59, 692: 56, 693: 59, 694: 58, 695: 58, 696: 59, 697: 58, 698: 60, 699: 5
9, 700: 56, 701: 60, 702: 57, 703: 66, 704: 61, 705: 263, 706: 124, 707: 10, 708:
254, 709: 23, 710: 61, 711: 58, 712: 20, 713: 61, 714: 57, 715: 58, 716: 60, 717:
62, 718: 53, 719: 9, 720: 51, 721: 52, 722: 63, 723: 60, 724: 60, 725: 60, 726: 5
8, 727: 58, 728: 60, 729: 60, 730: 61, 731: 58, 732: 61, 733: 58, 734: 58, 735: 5
7, 736: 60, 737: 58, 738: 61, 739: 57, 740: 63, 741: 59, 742: 61, 743: 58, 744: 12
3, 745: 55, 746: 10, 747: 62, 748: 58, 749: 56, 750: 58, 751: 56, 752: 62, 753: 5
9, 754: 62, 755: 57, 756: 60, 757: 62, 758: 59, 759: 61, 760: 65, 761: 63, 762: 6
0, 763: 61, 764: 59, 765: 62, 766: 6100, 767: 267, 768: 13, 769: 23, 770: 118, 77
1: 50, 772: 5627, 773: 37, 774: 120, 775: 28, 776: 11, 777: 54, 778: 52, 779: 55,
780: 51, 781: 55, 782: 49, 783: 111, 784: 19, 785: 48, 786: 62, 787: 55, 788: 55,
789: 57, 790: 56, 791: 55, 792: 56, 793: 57, 794: 57, 795: 56, 796: 57, 797: 56, 7
98: 182, 799: 55, 800: 54, 801: 55, 802: 54, 803: 58, 804: 59, 805: 54, 806: 48, 8
07: 48, 808: 51, 809: 56, 810: 53, 811: 54, 812: 57, 813: 55, 814: 57, 815: 58, 81
6: 58, 817: 58, 818: 56, 819: 57, 820: 56, 821: 57, 822: 58, 823: 55, 824: 55, 82
5: 57, 826: 58, 827: 53, 828: 65, 829: 67, 830: 64, 831: 64, 832: 394, 833: 68, 83
4: 130, 835: 69, 836: 68, 837: 66, 838: 65, 839: 65, 840: 68, 841: 68, 842: 68, 84
3: 67, 844: 66, 845: 67, 846: 67, 847: 68, 848: 68, 849: 68, 850: 68, 851: 131, 85
2: 383, 853: 68, 854: 67, 855: 69, 856: 69, 857: 68, 858: 65, 859: 68, 860: 68, 86
1: 199, 862: 66, 863: 134, 864: 129, 865: 66, 866: 61, 867: 131, 868: 63, 869: 66,
870: 68, 871: 128, 872: 67, 873: 127, 874: 66, 875: 134, 876: 69, 877: 66, 878: 6
5, 879: 68, 880: 68, 881: 2812, 882: 189, 883: 61, 884: 436, 885: 63, 886: 2626, 8
87: 61, 888: 65, 889: 65, 890: 68, 891: 66, 892: 66, 893: 68, 894: 63, 895: 66, 89
6: 60, 897: 61, 898: 60, 899: 62, 900: 64, 901: 62, 902: 63, 903: 63, 904: 64, 90
5: 63, 906: 63, 907: 206, 908: 269, 909: 68, 910: 135, 911: 69, 912: 132, 913: 65,
914: 61, 915: 61, 916: 62, 917: 68, 918: 63, 919: 62, 920: 62, 921: 62, 922: 62, 9
23: 62, 924: 66, 925: 68, 926: 3413, 927: 190, 928: 63, 929: 15, 930: 57, 931: 62,
932: 2302, 933: 12, 934: 58, 935: 61, 936: 61, 937: 62, 938: 61, 939: 63, 940: 61,
941: 61, 942: 63, 943: 60, 944: 63, 945: 127, 946: 62, 947: 62, 948: 62, 949: 62,
950: 61, 951: 63, 952: 62, 953: 62, 954: 63, 955: 64, 956: 65, 957: 63, 958: 63, 9
59: 62, 960: 64, 961: 24, 962: 374, 963: 127, 964: 187, 965: 60, 966: 63, 967: 14,
968: 62, 969: 61, 970: 62, 971: 62, 972: 61, 973: 55, 974: 62, 975: 65, 976: 62, 9
77: 64, 978: 63, 979: 59, 980: 128, 981: 61, 982: 59, 983: 127, 984: 126, 985: 11
8, 986: 61, 987: 61, 988: 60, 989: 61, 990: 61, 991: 56, 992: 54, 993: 64, 994: 6
3, 995: 63, 996: 63, 997: 66, 998: 63, 999: 62, 1000: 63, 1001: 64, 1002: 62, 100
3: 63, 1004: 64, 1005: 63, 1006: 62, 1007: 66, 1008: 24, 1009: 63, 1010: 63, 1011:
```

```
63, 1012: 64, 1013: 64, 1014: 65, 1015: 61, 1016: 65, 1017: 63, 1018: 62, 1019: 6
2, 1020: 63, 1021: 62, 1022: 61, 1023: 61, 1024: 62, 1025: 61, 1026: 62, 1027: 62,
1028: 61, 1029: 62, 1030: 199, 1031: 187, 1032: 12, 1033: 403, 1034: 17, 1035: 20
7, 1036: 63, 1037: 58, 1038: 65, 1039: 60, 1040: 61, 1041: 59, 1042: 118, 1043: 6
0, 1044: 58, 1045: 59, 1046: 60, 1047: 58, 1048: 60, 1049: 64, 1050: 60, 1051: 62,
1052: 62, 1053: 61, 1054: 61, 1055: 392, 1056: 61, 1057: 59, 1058: 60, 1059: 134,
1060: 62, 1061: 61, 1062: 263, 1063: 65, 1064: 65, 1065: 59, 1066: 62, 1067: 67, 1
068: 58, 1069: 65, 1070: 63, 1071: 129, 1072: 4, 1073: 123, 1074: 60, 1075: 60, 10
76: 60, 1077: 60, 1078: 60, 1079: 62, 1080: 62, 1081: 60, 1082: 62, 1083: 62, 108
4: 255, 1085: 15, 1086: 57, 1087: 130, 1088: 55, 1089: 59, 1090: 10, 1091: 15, 109
2: 60, 1093: 54, 1094: 59, 1095: 52, 1096: 44, 1097: 14, 1098: 47, 1099: 13, 1100:
49, 1101: 10, 1102: 48, 1103: 63, 1104: 57, 1105: 58, 1106: 56, 1107: 55, 1108: 5
6, 1109: 58, 1110: 54, 1111: 58, 1112: 59, 1113: 58, 1114: 58, 1115: 61, 1116: 57,
1117: 52, 1118: 59, 1119: 56, 1120: 60, 1121: 59, 1122: 62, 1123: 61, 1124: 59, 11
25: 50, 1126: 121, 1127: 61, 1128: 44, 1129: 14, 1130: 43, 1131: 59, 1132: 59, 113
3: 59, 1134: 57, 1135: 57, 1136: 57, 1137: 59, 1138: 57, 1139: 57, 1140: 60, 1141:
57, 1142: 51, 1143: 6, 1144: 49, 1145: 55, 1146: 57, 1147: 55, 1148: 61, 1149: 62,
1150: 64, 1151: 61, 1152: 63, 1153: 63, 1154: 62, 1155: 63, 1156: 64, 1157: 61, 11
58: 64, 1159: 60, 1160: 61, 1161: 62, 1162: 62, 1163: 62, 1164: 62, 1165: 63, 116
6: 63, 1167: 65, 1168: 62, 1169: 64, 1170: 63, 1171: 62, 1172: 123, 1173: 63, 117
4: 62, 1175: 62, 1176: 63, 1177: 64, 1178: 62, 1179: 63, 1180: 62, 1181: 64, 1182:
60, 1183: 62, 1184: 63, 1185: 62, 1186: 62, 1187: 63, 1188: 63, 1189: 63, 1190: 6
5, 1191: 64, 1192: 63, 1193: 127, 1194: 123, 1195: 58, 1196: 60, 1197: 58, 1198: 5
8, 1199: 56, 1200: 55, 1201: 53, 1202: 63, 1203: 59, 1204: 57, 1205: 60, 1206: 60,
1207: 60, 1208: 59, 1209: 60, 1210: 60, 1211: 62, 1212: 125, 1213: 62, 1214: 63, 1
215: 60, 1216: 62, 1217: 59, 1218: 61, 1219: 61, 1220: 123, 1221: 11, 1222: 51, 12
23: 61, 1224: 56, 1225: 54, 1226: 62, 1227: 58, 1228: 61, 1229: 59, 1230: 59, 123
1: 60, 1232: 59, 1233: 58, 1234: 58, 1235: 56, 1236: 59, 1237: 59, 1238: 60, 1239:
61, 1240: 59, 1241: 61, 1242: 67, 1243: 67, 1244: 2765, 1245: 61, 1246: 60, 1247:
743, 1248: 138, 1249: 209, 1250: 2048, 1251: 129, 1252: 65, 1253: 63, 1254: 62, 12
55: 64, 1256: 63, 1257: 60, 1258: 58, 1259: 58, 1260: 69, 1261: 66, 1262: 64, 126
3: 64, 1264: 63, 1265: 65, 1266: 66, 1267: 63, 1268: 66, 1269: 67, 1270: 65, 1271:
64, 1272: 65, 1273: 65, 1274: 62, 1275: 63, 1276: 65, 1277: 63, 1278: 63, 1279: 6
2, 1280: 64, 1281: 60, 1282: 131, 1283: 61, 1284: 59, 1285: 63, 1286: 63, 1287: 6
4, 1288: 62, 1289: 61, 1290: 63, 1291: 61, 1292: 61, 1293: 60, 1294: 58, 1295: 63,
1296: 63, 1297: 63, 1298: 131, 1299: 63, 1300: 64, 1301: 61, 1302: 11, 1303: 66, 1
304: 63, 1305: 67, 1306: 62, 1307: 61, 1308: 60, 1309: 7, 1310: 56, 1311: 67, 131
2: 62, 1313: 61, 1314: 62, 1315: 62, 1316: 61, 1317: 61, 1318: 61, 1319: 61, 1320:
63, 1321: 63, 1322: 61, 1323: 61, 1324: 62, 1325: 61, 1326: 62, 1327: 61, 1328: 6
0, 1329: 60, 1330: 59, 1331: 61, 1332: 60, 1333: 58, 1334: 8, 1335: 63, 1336: 64,
1337: 68, 1338: 63, 1339: 61, 1340: 63, 1341: 61, 1342: 61, 1343: 60, 1344: 61, 13
45: 62, 1346: 63, 1347: 62, 1348: 60, 1349: 62, 1350: 124, 1351: 11, 1352: 62, 135
3: 63, 1354: 60, 1355: 274, 1356: 63, 1357: 62, 1358: 67, 1359: 738, 1360: 131, 13
61: 63, 1362: 546, 1363: 61, 1364: 1699, 1365: 61, 1366: 64, 1367: 64, 1368: 66, 1
369: 62, 1370: 60, 1371: 62, 1372: 62, 1373: 64, 1374: 63, 1375: 63, 1376: 64, 137
7: 63, 1378: 62, 1379: 62, 1380: 62, 1381: 61, 1382: 61, 1383: 61, 1384: 63, 1385:
64, 1386: 136, 1387: 63, 1388: 63, 1389: 135, 1390: 134, 1391: 62, 1392: 65, 1393:
62, 1394: 66, 1395: 61, 1396: 66, 1397: 64, 1398: 63, 1399: 67, 1400: 123, 1401: 6
1, 1402: 64, 1403: 63, 1404: 62, 1405: 62, 1406: 61, 1407: 62, 1408: 62, 1409: 62,
1410: 125, 1411: 11, 1412: 61, 1413: 59, 1414: 60, 1415: 1930, 1416: 1435, 1417: 1
0, 1418: 33, 1419: 13, 1420: 60, 1421: 61, 1422: 125, 1423: 455, 1424: 125, 1425:
262, 1426: 1416, 1427: 13, 1428: 62, 1429: 59, 1430: 62, 1431: 65, 1432: 66, 1433:
63, 1434: 64, 1435: 65, 1436: 67, 1437: 61, 1438: 62, 1439: 62, 1440: 61, 1441: 6
2, 1442: 65, 1443: 61, 1444: 61, 1445: 62, 1446: 62, 1447: 61, 1448: 65, 1449: 12
9, 1450: 62, 1451: 390, 1452: 61, 1453: 62, 1454: 62, 1455: 60, 1456: 128, 1457: 6
3, 1458: 63, 1459: 62, 1460: 65, 1461: 63, 1462: 62, 1463: 62, 1464: 64, 1465: 63,
```

```
1466: 62, 1467: 62, 1468: 65, 1469: 62, 1470: 62, 1471: 62, 1472: 61, 1473: 11, 14
74: 62, 1475: 62, 1476: 59, 1477: 63, 1478: 15, 1479: 48, 1480: 49, 1481: 13, 148
2: 10, 1483: 67, 1484: 60, 1485: 59, 1486: 58, 1487: 62, 1488: 59, 1489: 62, 1490:
62, 1491: 62, 1492: 62, 1493: 60, 1494: 61, 1495: 65, 1496: 61, 1497: 64, 1498: 6
4, 1499: 64, 1500: 64, 1501: 62, 1502: 57, 1503: 63, 1504: 130, 1505: 58, 1506: 6,
1507: 53, 1508: 8, 1509: 63, 1510: 62, 1511: 64, 1512: 59, 1513: 61, 1514: 62, 151
5: 61, 1516: 61, 1517: 63, 1518: 61, 1519: 58, 1520: 61, 1521: 61, 1522: 3485, 152
3: 66, 1524: 21, 1525: 61, 1526: 134, 1527: 56, 1528: 60, 1529: 343, 1530: 1643, 1
531: 14, 1532: 60, 1533: 270, 1534: 64, 1535: 60, 1536: 62, 1537: 64, 1538: 60, 15
39: 63, 1540: 133, 1541: 66, 1542: 61, 1543: 62, 1544: 59, 1545: 61, 1546: 59, 154
7: 62, 1548: 58, 1549: 60, 1550: 60, 1551: 60, 1552: 59, 1553: 60, 1554: 64, 1555:
60, 1556: 60, 1557: 62, 1558: 65, 1559: 60, 1560: 134, 1561: 60, 1562: 66, 1563: 6
1, 1564: 60, 1565: 64, 1566: 66, 1567: 63, 1568: 61, 1569: 64, 1570: 63, 1571: 58,
1572: 63, 1573: 66, 1574: 134, 1575: 65, 1576: 58, 1577: 61, 1578: 58, 1579: 59, 1
580: 62, 1581: 60, 1582: 65, 1583: 62, 1584: 65, 1585: 2626, 1586: 63, 1587: 63, 1
588: 718, 1589: 130, 1590: 450, 1591: 1454, 1592: 65, 1593: 65, 1594: 62, 1595: 6
3, 1596: 66, 1597: 65, 1598: 63, 1599: 60, 1600: 65, 1601: 65, 1602: 65, 1603: 64,
1604: 64, 1605: 63, 1606: 63, 1607: 64, 1608: 64, 1609: 61, 1610: 195, 1611: 255,
1612: 128, 1613: 65, 1614: 66, 1615: 60, 1616: 66, 1617: 64, 1618: 66, 1619: 65, 1
620: 63, 1621: 64, 1622: 252, 1623: 65, 1624: 61, 1625: 63, 1626: 64, 1627: 63, 16
28: 63, 1629: 124, 1630: 65, 1631: 63, 1632: 63, 1633: 128, 1634: 63, 1635: 64, 16
36: 125, 1637: 42, 1638: 130, 1639: 134, 1640: 217, 1641: 42, 1642: 43, 1643: 48,
1644: 47, 1645: 45, 1646: 144, 1647: 42, 1648: 48, 1649: 46, 1650: 47, 1651: 42, 1
652: 45, 1653: 141, 1654: 44, 1655: 8, 1656: 59, 1657: 58, 1658: 56, 1659: 58, 166
0: 57, 1661: 60, 1662: 60, 1663: 55, 1664: 59, 1665: 57, 1666: 60, 1667: 58, 1668:
61, 1669: 59, 1670: 60, 1671: 59, 1672: 58, 1673: 62, 1674: 60, 1675: 58, 1676: 5
8, 1677: 60, 1678: 60, 1679: 57, 1680: 61, 1681: 61, 1682: 58, 1683: 55, 1684: 60,
1685: 60, 1686: 57, 1687: 60, 1688: 58, 1689: 60, 1690: 60, 1691: 59, 1692: 60, 16
93: 59, 1694: 59, 1695: 60, 1696: 61, 1697: 59, 1698: 59, 1699: 60, 1700: 58, 170
1: 59, 1702: 62, 1703: 62, 1704: 61, 1705: 62, 1706: 59, 1707: 60, 1708: 62, 1709:
56, 1710: 53, 1711: 10, 1712: 64, 1713: 59, 1714: 61, 1715: 61, 1716: 61, 1717: 6
2, 1718: 63, 1719: 62, 1720: 62, 1721: 63, 1722: 63, 1723: 62, 1724: 62, 1725: 25
0, 1726: 61, 1727: 62, 1728: 62, 1729: 61, 1730: 65, 1731: 61, 1732: 23, 1733: 61,
1734: 63, 1735: 62, 1736: 63, 1737: 62, 1738: 62, 1739: 61, 1740: 62, 1741: 63, 17
42: 61, 1743: 60, 1744: 60, 1745: 62, 1746: 62, 1747: 63, 1748: 124, 1749: 63, 175
0: 63, 1751: 62, 1752: 61, 1753: 4534, 1754: 75, 1755: 40, 1756: 174, 1757: 84, 17
58: 3149, 1759: 87, 1760: 42, 1761: 41, 1762: 40, 1763: 179, 1764: 1160, 1765: 94,
1766: 2963, 1767: 42, 1768: 3669, 1769: 60, 1770: 8, 1771: 68, 1772: 65, 1773: 13
6, 1774: 1473, 1775: 987, 1776: 63, 1777: 11, 1778: 67, 1779: 66, 1780: 66, 1781:
69, 1782: 67, 1783: 133, 1784: 67, 1785: 65, 1786: 61, 1787: 66, 1788: 63, 1789: 6
1, 1790: 61, 1791: 59, 1792: 61, 1793: 63, 1794: 60, 1795: 61, 1796: 63, 1797: 63,
1798: 59, 1799: 65, 1800: 67, 1801: 63, 1802: 63, 1803: 70, 1804: 131, 1805: 63, 1
806: 66, 1807: 62, 1808: 62, 1809: 134, 1810: 68, 1811: 65, 1812: 139, 1813: 67, 1
814: 63, 1815: 59, 1816: 65, 1817: 66, 1818: 67, 1819: 61, 1820: 63, 1821: 65, 182
2: 60, 1823: 58, 1824: 67, 1825: 10, 1826: 64, 1827: 65, 1828: 59, 1829: 62, 1830:
61, 1831: 64, 1832: 60, 1833: 63, 1834: 64, 1835: 63, 1836: 64, 1837: 64, 1838: 6
1, 1839: 63, 1840: 65, 1841: 63, 1842: 65, 1843: 59, 1844: 59, 1845: 62, 1846: 63,
1847: 61, 1848: 60, 1849: 59, 1850: 62, 1851: 61, 1852: 126, 1853: 58, 1854: 60, 1
855: 63, 1856: 61, 1857: 58, 1858: 64, 1859: 62, 1860: 65, 1861: 57, 1862: 65, 186
3: 57, 1864: 59, 1865: 60, 1866: 62, 1867: 60, 1868: 61, 1869: 56, 1870: 60, 1871:
76, 1872: 84, 1873: 91, 1874: 87, 1875: 41, 1876: 4, 1877: 2529, 1878: 46, 1879: 4
6, 1880: 95, 1881: 43, 1882: 44, 1883: 238, 1884: 18, 1885: 1239, 1886: 47, 1887:
188, 1888: 12, 1889: 45, 1890: 144, 1891: 46, 1892: 145, 1893: 49, 1894: 48, 1895:
342, 1896: 48, 1897: 50, 1898: 46, 1899: 241, 1900: 10, 1901: 690, 1902: 10, 1903:
20, 1904: 98, 1905: 312, 1906: 48, 1907: 47, 1908: 154, 1909: 62, 1910: 53, 1911:
60, 1912: 58, 1913: 53, 1914: 51, 1915: 5, 1916: 55, 1917: 54, 1918: 56, 1919: 7,
```

```
1920: 54, 1921: 58, 1922: 49, 1923: 10, 1924: 46, 1925: 12, 1926: 58, 1927: 56, 19
28: 55, 1929: 113, 1930: 10, 1931: 55, 1932: 55, 1933: 58, 1934: 54, 1935: 54, 193
6: 51, 1937: 54, 1938: 57, 1939: 52, 1940: 52, 1941: 50, 1942: 54, 1943: 53, 1944:
52, 1945: 51, 1946: 51, 1947: 53, 1948: 51, 1949: 10, 1950: 51, 1951: 53, 1952: 5
5, 1953: 55, 1954: 50, 1955: 55, 1956: 51, 1957: 54, 1958: 53, 1959: 54, 1960: 56,
1961: 57, 1962: 57, 1963: 55, 1964: 56, 1965: 54, 1966: 53, 1967: 55, 1968: 56, 19
69: 54, 1970: 62, 1971: 65, 1972: 1238, 1973: 1500, 1974: 65, 1975: 65, 1976: 760,
1977: 131, 1978: 462, 1979: 133, 1980: 1669, 1981: 37, 1982: 10, 1983: 69, 1984: 6
7, 1985: 68, 1986: 67, 1987: 65, 1988: 67, 1989: 66, 1990: 66, 1991: 66, 1992: 65,
1993: 65, 1994: 66, 1995: 62, 1996: 62, 1997: 64, 1998: 65, 1999: 65, 2000: 62, 20
01: 60, 2002: 62, 2003: 65, 2004: 65, 2005: 65, 2006: 66, 2007: 64, 2008: 134, 200
9: 66, 2010: 64, 2011: 132, 2012: 64, 2013: 66, 2014: 66, 2015: 67, 2016: 65, 201
7: 67, 2018: 65, 2019: 64, 2020: 134, 2021: 66, 2022: 66, 2023: 65, 2024: 68, 202
5: 67, 2026: 65, 2027: 66, 2028: 133, 2029: 64, 2030: 68, 2031: 197, 2032: 10, 203
3: 36, 2034: 61, 2035: 128, 2036: 59, 2037: 61, 2038: 125, 2039: 872, 2040: 20, 20
41: 1355, 2042: 22, 2043: 59, 2044: 62, 2045: 59, 2046: 58, 2047: 8, 2048: 59, 204
9: 59, 2050: 61, 2051: 52, 2052: 11, 2053: 63, 2054: 60, 2055: 62, 2056: 60, 2057:
62, 2058: 60, 2059: 64, 2060: 62, 2061: 62, 2062: 64, 2063: 62, 2064: 62, 2065: 6
1, 2066: 273, 2067: 58, 2068: 56, 2069: 54, 2070: 61, 2071: 53, 2072: 60, 2073: 5
4, 2074: 137, 2075: 56, 2076: 56, 2077: 62, 2078: 62, 2079: 56, 2080: 58, 2081: 5
7, 2082: 57, 2083: 63, 2084: 60, 2085: 58, 2086: 125, 2087: 61, 2088: 60, 2089: 6
0, 2090: 58, 2091: 60, 2092: 40, 2093: 42, 2094: 43, 2095: 42, 2096: 45, 2097: 9,
2098: 14, 2099: 44, 2100: 42, 2101: 2135, 2102: 39, 2103: 17, 2104: 38, 2105: 40,
2106: 44, 2107: 38, 2108: 159, 2109: 39, 2110: 1588, 2111: 37, 2112: 25, 2113: 38,
2114: 42, 2115: 43, 2116: 167, 2117: 41, 2118: 43, 2119: 41, 2120: 81, 2121: 38, 2
122: 41, 2123: 40, 2124: 210, 2125: 15, 2126: 1077, 2127: 95, 2128: 40, 2129: 40,
2130: 9, 2131: 2290, 2132: 43, 2133: 46, 2134: 92, 2135: 44, 2136: 140, 2137: 44,
2138: 43, 2139: 2872, 2140: 47, 2141: 584, 2142: 44, 2143: 45, 2144: 47, 2145: 45,
2146: 44, 2147: 522, 2148: 48, 2149: 267, 2150: 63, 2151: 2469, 2152: 61, 2153: 6
2, 2154: 3006, 2155: 126, 2156: 61, 2157: 59, 2158: 59, 2159: 62, 2160: 60, 2161:
61, 2162: 55, 2163: 57, 2164: 64, 2165: 62, 2166: 60, 2167: 61, 2168: 62, 2169: 6
1, 2170: 61, 2171: 61, 2172: 61, 2173: 60, 2174: 62, 2175: 63, 2176: 63, 2177: 64,
2178: 63, 2179: 63, 2180: 65, 2181: 125, 2182: 61, 2183: 64, 2184: 127, 2185: 60,
2186: 63, 2187: 65, 2188: 65, 2189: 63, 2190: 60, 2191: 61, 2192: 62, 2193: 61, 21
94: 62, 2195: 62, 2196: 60, 2197: 61, 2198: 61, 2199: 62, 2200: 62, 2201: 61, 220
2: 128, 2203: 60, 2204: 61}
```

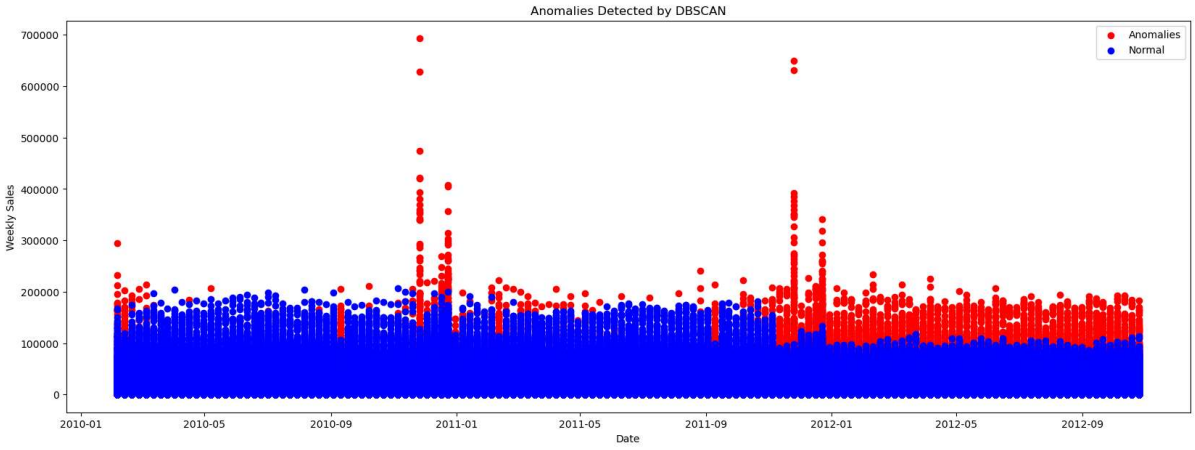
In [36]:
```
pd.value_counts(df4['DBSCAN_Labels']==-1)
```

Out[36]:
```
DBSCAN_Labels
False    414602
True       5683
Name: count, dtype: int64
```

In [37]:
```
### same results
```

In [39]:
```
### visualize the anomalies resulted from the DBSCAN
df4.reset_index(inplace=True)
plt.figure(figsize=(20, 7))

# Plot anomalies (label = -1) as red
plt.scatter(df4[df4['DBSCAN_Labels'] == -1]['Date'],
            df4[df4['DBSCAN_Labels'] == -1]['Weekly_Sales'],
```

```
            c='red', label='Anomalies')

# Plot normal data points as blue
plt.scatter(df4[df4['DBSCAN_Labels'] != -1]['Date'],
            df4[df4['DBSCAN_Labels'] != -1]['Weekly_Sales'],
            c='blue', label='Normal')

plt.title('Anomalies Detected by DBSCAN')
plt.xlabel('Date')
plt.ylabel('Weekly Sales')
plt.legend()
plt.show()
```



Analyzing DBSCAN Results for Weekly Sales Anomaly Detection

DBSCAN shows promising results in anomaly detection for our Walmart dataset. However, like KNN, it struggles with identifying anomalies in significant decreases in weekly sales. This limitation may be acceptable depending on our application's focus, such as detecting improvement trends.

Comparing Anomaly Detection Models

Isolation Forest outperforms both KNN and DBSCAN in our Walmart dataset for anomaly detection. KNN lags behind but can improve with the right K value. DBSCAN falls between KNN and Isolation Forest, offering better performance than KNN but not matching the robustness of Isolation Forest. Isolation Forest's strength lies in its ability to handle high-dimensional data and identify anomalies efficiently, especially when the data is well understood. Additionally, Isolation Forest allows for adjusting the contamination parameter, which represents the expected percentage of anomalies in the data. This flexibility enables fine-tuning to match the specific anomaly detection requirements of the dataset. KNN's performance suffers due to the curse of dimensionality, but tuning the K parameter can enhance its results. Overall, Isolation Forest is the top choice, with KNN and DBSCAN being viable alternatives depending on the specific dataset characteristics and requirements.