Product_Sales_Forecasting

December 23, 2025

1 Product Sales Forecasting

Project Objective

Need and Use of Product Sales Forecasting Effective sales forecasting is fundamental for multiple aspects of retail management and operation, including:

1. **Inventory Management:** Accurate sales forecasts help ensure that stores maintain optimal inventory levels—enough to meet customer demand without overstocking, which can lead to increased costs or waste, especially in the case of perishable goods.
2. **Financial Planning:** Forecasting sales allows businesses to estimate future revenue and manage budgets more effectively. This is crucial for allocating resources to areas such as marketing, staffing, and capital investments.
3. **Marketing and Promotions:** Understanding when sales peaks and troughs are likely to occur enables retailers to plan effective marketing campaigns and promotional offers to boost revenue or manage customer flow.
4. **Supply Chain Optimization:** Sales forecasts inform production schedules, logistics, and distribution plans, ensuring that products are available where and when they are needed, thereby reducing transportation and storage costs.
5. **Strategic Decision Making:** Long-term sales forecasting supports broader business strategies, including store expansions, market entry, and other capital expenditures.

1.1 1. Dataset Loading

[3]:
```python
# Import Basic libraries
import pandas as pd
import numpy as np

# Import Visualization libraries
import plotly.graph_objs as go
import plotly.express as px
import plotly.io as pio
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```python
# Install MLflow
!pip install mlflow

# Import MLFLow libraries
import mlflow

# Suppress the specific ConvergenceWarning
from statsmodels.tools.sm_exceptions import ConvergenceWarning
warnings.simplefilter("ignore", ConvergenceWarning)
```

```
Collecting mlflow
  Downloading mlflow-3.8.0-py3-none-any.whl.metadata (31 kB)
Collecting mlflow-skinny==3.8.0 (from mlflow)
  Downloading mlflow_skinny-3.8.0-py3-none-any.whl.metadata (31 kB)
Collecting mlflow-tracing==3.8.0 (from mlflow)
  Downloading mlflow_tracing-3.8.0-py3-none-any.whl.metadata (19 kB)
Collecting Flask-CORS<7 (from mlflow)
  Downloading flask_cors-6.0.2-py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: Flask<4 in /usr/local/lib/python3.12/dist-
packages (from mlflow) (3.1.2)
Requirement already satisfied: alembic!=1.10.0,<2 in
/usr/local/lib/python3.12/dist-packages (from mlflow) (1.17.2)
Requirement already satisfied: cryptography<47,>=43.0.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow) (43.0.3)
Collecting docker<8,>=4.0.0 (from mlflow)
  Downloading docker-7.1.0-py3-none-any.whl.metadata (3.8 kB)
Collecting graphene<4 (from mlflow)
  Downloading graphene-3.4.3-py2.py3-none-any.whl.metadata (6.9 kB)
Collecting gunicorn<24 (from mlflow)
  Downloading gunicorn-23.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting huey<3,>=2.5.0 (from mlflow)
  Downloading huey-2.5.5-py3-none-any.whl.metadata (4.8 kB)
Requirement already satisfied: matplotlib<4 in /usr/local/lib/python3.12/dist-
packages (from mlflow) (3.10.0)
Requirement already satisfied: numpy<3 in /usr/local/lib/python3.12/dist-
packages (from mlflow) (2.0.2)
Requirement already satisfied: pandas<3 in /usr/local/lib/python3.12/dist-
packages (from mlflow) (2.2.2)
Requirement already satisfied: pyarrow<23,>=4.0.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow) (18.1.0)
Requirement already satisfied: scikit-learn<2 in /usr/local/lib/python3.12/dist-
packages (from mlflow) (1.6.1)
Requirement already satisfied: scipy<2 in /usr/local/lib/python3.12/dist-
packages (from mlflow) (1.16.3)
Requirement already satisfied: sqlalchemy<3,>=1.4.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow) (2.0.45)
```

```
Requirement already satisfied: cachetools<7,>=5.0.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(6.2.4)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.12/dist-
packages (from mlflow-skinny==3.8.0->mlflow) (8.3.1)
Requirement already satisfied: cloudpickle<4 in /usr/local/lib/python3.12/dist-
packages (from mlflow-skinny==3.8.0->mlflow) (3.1.2)
Collecting databricks-sdk<1,>=0.20.0 (from mlflow-skinny==3.8.0->mlflow)
  Downloading databricks_sdk-0.76.0-py3-none-any.whl.metadata (40 kB)
                           40.1/40.1 kB
3.1 MB/s eta 0:00:00
Requirement already satisfied: fastapi<1 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(0.123.10)
Requirement already satisfied: gitpython<4,>=3.1.9 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(3.1.45)
Requirement already satisfied: importlib_metadata!=4.7.0,<9,>=3.7.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(8.7.0)
Requirement already satisfied: opentelemetry-api<3,>=1.9.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(1.37.0)
Requirement already satisfied: opentelemetry-proto<3,>=1.9.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(1.37.0)
Requirement already satisfied: opentelemetry-sdk<3,>=1.9.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(1.37.0)
Requirement already satisfied: packaging<26 in /usr/local/lib/python3.12/dist-
packages (from mlflow-skinny==3.8.0->mlflow) (25.0)
Requirement already satisfied: protobuf<7,>=3.12.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(5.29.5)
Requirement already satisfied: pydantic<3,>=2.0.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(2.12.3)
Requirement already satisfied: python-dotenv<2,>=0.19.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(1.2.1)
Requirement already satisfied: pyyaml<7,>=5.1 in /usr/local/lib/python3.12/dist-
packages (from mlflow-skinny==3.8.0->mlflow) (6.0.3)
Requirement already satisfied: requests<3,>=2.17.3 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(2.32.4)
Requirement already satisfied: sqlparse<1,>=0.4.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(0.5.4)
```

```
Requirement already satisfied: typing-extensions<5,>=4.0.0 in
/usr/local/lib/python3.12/dist-packages (from mlflow-skinny==3.8.0->mlflow)
(4.15.0)
Requirement already satisfied: uvicorn<1 in /usr/local/lib/python3.12/dist-
packages (from mlflow-skinny==3.8.0->mlflow) (0.38.0)
Requirement already satisfied: Mako in /usr/local/lib/python3.12/dist-packages
(from alembic!=1.10.0,<2->mlflow) (1.3.10)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.12/dist-
packages (from cryptography<47,>=43.0.0->mlflow) (2.0.0)
Requirement already satisfied: urllib3>=1.26.0 in
/usr/local/lib/python3.12/dist-packages (from docker<8,>=4.0.0->mlflow) (2.5.0)
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.12/dist-
packages (from Flask<4->mlflow) (1.9.0)
Requirement already satisfied: itsdangerous>=2.2.0 in
/usr/local/lib/python3.12/dist-packages (from Flask<4->mlflow) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in /usr/local/lib/python3.12/dist-
packages (from Flask<4->mlflow) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in
/usr/local/lib/python3.12/dist-packages (from Flask<4->mlflow) (3.0.3)
Requirement already satisfied: werkzeug>=3.1.0 in
/usr/local/lib/python3.12/dist-packages (from Flask<4->mlflow) (3.1.4)
Collecting graphql-core<3.3,>=3.1 (from graphene<4->mlflow)
  Downloading graphql_core-3.2.7-py3-none-any.whl.metadata (11 kB)
Collecting graphql-relay<3.3,>=3.1 (from graphene<4->mlflow)
  Downloading graphql_relay-3.2.0-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: python-dateutil<3,>=2.7.0 in
/usr/local/lib/python3.12/dist-packages (from graphene<4->mlflow) (2.9.0.post0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib<4->mlflow) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-
packages (from matplotlib<4->mlflow) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib<4->mlflow) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib<4->mlflow) (1.4.9)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-
packages (from matplotlib<4->mlflow) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib<4->mlflow) (3.2.5)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-
packages (from pandas<3->mlflow) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
packages (from pandas<3->mlflow) (2025.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn<2->mlflow) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-learn<2->mlflow) (3.6.0)
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-
```

```
packages (from sqlalchemy<3,>=1.4.0->mlflow) (3.3.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-
packages (from cffi>=1.12->cryptography<47,>=43.0.0->mlflow) (2.23)
Requirement already satisfied: google-auth~=2.0 in
/usr/local/lib/python3.12/dist-packages (from databricks-sdk<1,>=0.20.0->mlflow-
skinny==3.8.0->mlflow) (2.43.0)
Requirement already satisfied: starlette<0.51.0,>=0.40.0 in
/usr/local/lib/python3.12/dist-packages (from fastapi<1->mlflow-
skinny==3.8.0->mlflow) (0.50.0)
Requirement already satisfied: annotated-doc>=0.0.2 in
/usr/local/lib/python3.12/dist-packages (from fastapi<1->mlflow-
skinny==3.8.0->mlflow) (0.0.4)
Requirement already satisfied: gitdb<5,>=4.0.1 in
/usr/local/lib/python3.12/dist-packages (from gitpython<4,>=3.1.9->mlflow-
skinny==3.8.0->mlflow) (4.0.12)
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.12/dist-
packages (from importlib_metadata!=4.7.0,<9,>=3.7.0->mlflow-
skinny==3.8.0->mlflow) (3.23.0)
Requirement already satisfied: opentelemetry-semantic-conventions==0.58b0 in
/usr/local/lib/python3.12/dist-packages (from opentelemetry-
sdk<3,>=1.9.0->mlflow-skinny==3.8.0->mlflow) (0.58b0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.12/dist-packages (from pydantic<3,>=2.0.0->mlflow-
skinny==3.8.0->mlflow) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in
/usr/local/lib/python3.12/dist-packages (from pydantic<3,>=2.0.0->mlflow-
skinny==3.8.0->mlflow) (2.41.4)
Requirement already satisfied: typing-inspection>=0.4.2 in
/usr/local/lib/python3.12/dist-packages (from pydantic<3,>=2.0.0->mlflow-
skinny==3.8.0->mlflow) (0.4.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-
packages (from python-dateutil<3,>=2.7.0->graphene<4->mlflow) (1.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.17.3->mlflow-
skinny==3.8.0->mlflow) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-
packages (from requests<3,>=2.17.3->mlflow-skinny==3.8.0->mlflow) (3.11)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.17.3->mlflow-
skinny==3.8.0->mlflow) (2025.11.12)
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.12/dist-
packages (from uvicorn<1->mlflow-skinny==3.8.0->mlflow) (0.16.0)
Requirement already satisfied: smmap<6,>=3.0.1 in
/usr/local/lib/python3.12/dist-packages (from
gitdb<5,>=4.0.1->gitpython<4,>=3.1.9->mlflow-skinny==3.8.0->mlflow) (5.0.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.12/dist-packages (from google-auth~=2.0->databricks-
sdk<1,>=0.20.0->mlflow-skinny==3.8.0->mlflow) (0.4.2)
```

```
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.12/dist-
packages (from google-auth~=2.0->databricks-sdk<1,>=0.20.0->mlflow-
skinny==3.8.0->mlflow) (4.9.1)
Requirement already satisfied: anyio<5,>=3.6.2 in
/usr/local/lib/python3.12/dist-packages (from
starlette<0.51.0,>=0.40.0->fastapi<1->mlflow-skinny==3.8.0->mlflow) (4.12.0)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in
/usr/local/lib/python3.12/dist-packages (from pyasn1-modules>=0.2.1->google-
auth~=2.0->databricks-sdk<1,>=0.20.0->mlflow-skinny==3.8.0->mlflow) (0.6.1)
Downloading mlflow-3.8.0-py3-none-any.whl (9.1 MB)
                         9.1/9.1 MB
80.9 MB/s eta 0:00:00
Downloading mlflow_skinny-3.8.0-py3-none-any.whl (2.5 MB)
                         2.5/2.5 MB
122.3 MB/s eta 0:00:00
Downloading mlflow_tracing-3.8.0-py3-none-any.whl (1.4 MB)
                         1.4/1.4 MB
86.6 MB/s eta 0:00:00
Downloading docker-7.1.0-py3-none-any.whl (147 kB)
                         147.8/147.8 kB
14.9 MB/s eta 0:00:00
Downloading flask_cors-6.0.2-py3-none-any.whl (13 kB)
Downloading graphene-3.4.3-py2.py3-none-any.whl (114 kB)
                         114.9/114.9 kB
10.2 MB/s eta 0:00:00
Downloading gunicorn-23.0.0-py3-none-any.whl (85 kB)
                         85.0/85.0 kB
8.4 MB/s eta 0:00:00
Downloading huey-2.5.5-py3-none-any.whl (76 kB)
                         76.9/76.9 kB
7.3 MB/s eta 0:00:00
Downloading databricks_sdk-0.76.0-py3-none-any.whl (774 kB)
                         774.7/774.7 kB
40.4 MB/s eta 0:00:00
Downloading graphql_core-3.2.7-py3-none-any.whl (207 kB)
                         207.3/207.3 kB
19.3 MB/s eta 0:00:00
Downloading graphql_relay-3.2.0-py3-none-any.whl (16 kB)
Installing collected packages: huey, gunicorn, graphql-core, graphql-relay,
docker, graphene, Flask-CORS, databricks-sdk, mlflow-tracing, mlflow-skinny,
mlflow
Successfully installed Flask-CORS-6.0.2 databricks-sdk-0.76.0 docker-7.1.0
graphene-3.4.3 graphql-core-3.2.7 graphql-relay-3.2.0 gunicorn-23.0.0 huey-2.5.5
mlflow-3.8.0 mlflow-skinny-3.8.0 mlflow-tracing-3.8.0
```

[4]:

```
# Read
train = pd.read_csv('/content/TRAIN.csv')
```

```
test = pd.read_csv('/content/TEST.csv')
```

[5]:
```
train.sample(5)
```

[5]:
```
              ID  Store_id Store_Type Location_Type Region_Code        Date  \
46883   T1046884        71         S2            L5          R1  2018-05-09
62624   T1062625       134         S1            L4          R1  2018-06-21
139396  T1139397       192         S1            L1          R3  2019-01-17
88778   T1088779        31         S1            L5          R2  2018-09-01
158244  T1158245       275         S4            L1          R1  2019-03-10

        Holiday Discount  #Order     Sales
46883         1       No      26  16903.08
62624         0      Yes      59  41364.00
139396        0      Yes      65  45756.00
88778         0      Yes      63  33240.00
158244        0      Yes     103  60051.00
```

[6]:
```
test.sample(5)
```

[6]:
```
             ID  Store_id Store_Type Location_Type Region_Code        Date  \
12131  T1200472       238         S1            L1          R2  2019-07-04
14568  T1202909       159         S2            L1          R3  2019-07-10
5793   T1194134       337         S4            L2          R1  2019-06-16
16307  T1204648       159         S2            L1          R3  2019-07-15
20526  T1208867        72         S1            L2          R3  2019-07-27

       Holiday Discount
12131        1      Yes
14568        0      Yes
5793         0       No
16307        0       No
20526        0       No
```

1.2 2. Observations on Data

[7]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188340 entries, 0 to 188339
Data columns (total 10 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   ID             188340 non-null  object
 1   Store_id       188340 non-null  int64
 2   Store_Type     188340 non-null  object
 3   Location_Type  188340 non-null  object
```

```
 4   Region_Code    188340 non-null  object
 5   Date           188340 non-null  object
 6   Holiday        188340 non-null  int64
 7   Discount       188340 non-null  object
 8   #Order         188340 non-null  int64
 9   Sales          188340 non-null  float64
dtypes: float64(1), int64(3), object(6)
memory usage: 14.4+ MB
```

[8]:
```python
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22265 entries, 0 to 22264
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   ID             22265 non-null  object
 1   Store_id       22265 non-null  int64
 2   Store_Type     22265 non-null  object
 3   Location_Type  22265 non-null  object
 4   Region_Code    22265 non-null  object
 5   Date           22265 non-null  object
 6   Holiday        22265 non-null  int64
 7   Discount       22265 non-null  object
dtypes: int64(2), object(6)
memory usage: 1.4+ MB
```

[9]:
```python
train.Date = pd.to_datetime(train.Date)
test.Date = pd.to_datetime(test.Date)
```

[10]:
```python
train.describe().T
```

[10]:
```
              count                           mean                  min  \
Store_id   188340.0                          183.0                  1.0
Date         188340  2018-09-15 12:00:00.000000256  2018-01-01 00:00:00
Holiday    188340.0                       0.131783                  0.0
#Order     188340.0                      68.205692                  0.0
Sales      188340.0                   42784.327982                  0.0

                           25%                  50%                  75%  \
Store_id                  92.0                183.0                274.0
Date       2018-05-09 18:00:00  2018-09-15 12:00:00  2019-01-22 06:00:00
Holiday                    0.0                  0.0                  0.0
#Order                    48.0                 63.0                 82.0
Sales                  30426.0              39678.0              51909.0

                           max           std
```

```
Store_id                 365.0     105.366308
Date       2019-05-31 00:00:00            NaN
Holiday                    1.0       0.338256
#Order                   371.0      30.467415
Sales                 247215.0   18456.708302
```

[11]:
```
test.describe().T
```

[11]:
```
             count                           mean                  min  \
Store_id   22265.0                          183.0                  1.0
Date         22265  2019-06-30 23:59:59.999999744  2019-06-01 00:00:00
Holiday    22265.0                       0.032787                  0.0

                           25%                  50%                  75%  \
Store_id                  92.0                183.0                274.0
Date       2019-06-16 00:00:00  2019-07-01 00:00:00  2019-07-16 00:00:00
Holiday                    0.0                  0.0                  0.0

                           max         std
Store_id                 365.0  105.368395
Date       2019-07-31 00:00:00         NaN
Holiday                    1.0    0.178082
```

1.3 3. Handling missing values and Preprocessing

[12]:
```
train_null = train.isna().sum().sum()
test_null = test.isna().sum().sum()
print(f'There are {train_null} nulls in train dataset and {test_null} nulls in␣
↪test dataset.')
```

```
There are 0 nulls in train dataset and 0 nulls in test dataset.
```

[13]:
```
# Define dataset type in separate column for train and test
train['Train'] = True
test['Train'] = False
```

[14]:
```
def decorator(func):
  def wrapper(*args, **kwargs):
    print('='*50)
    result = func(*args, **kwargs)
    print('='*50)
    return result
  return wrapper

@decorator
def df_size(df,typ):
  size = df.memory_usage().sum()/(1024**2)
```

```
    print(f'Size of {typ} data is: {size:.2f} MB')
    return size
```

[15]:
```
# Combine both the dataset into single dataframe
data = pd.concat([train, test])
raw_size = df_size(data, 'Non-Converted')
data.reset_index(drop=True, inplace=True)
# Change Datatypes to optimize sizes
# Store_id as unsigned integer 16 (Range is 1 to 371)
data.Store_id = data.Store_id.astype('uint16')
# Store_Type, Location_Type, Region_Code as categorical
data.Store_Type = data.Store_Type.astype('category')
data.Location_Type = data.Location_Type.astype('category')
data.Region_Code = data.Region_Code.astype('category')
# Holiday and Discount as Boolean
data.Holiday = data.Holiday.astype('bool')
data.replace({'Discount':{'Yes':True, 'No':False}}, inplace=True)
# Drop unnecessary column Transaction ID
data.pop('ID')
data.set_index('Date', inplace=True)
processed_size = df_size(data, 'Converted')
reduction = 100*(raw_size - processed_size)/raw_size
print(f'''Reduction in size after processing is: {reduction:.2f}%''')
print('='*50)
```

```
==================================================
Size of Non-Converted data is: 17.88 MB
==================================================
==================================================
Size of Converted data is: 6.43 MB
==================================================
Reduction in size after processing is: 64.04%
==================================================
```

[16]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 210605 entries, 2018-01-01 to 2019-07-31
Data columns (total 9 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Store_id       210605 non-null  uint16
 1   Store_Type     210605 non-null  category
 2   Location_Type  210605 non-null  category
 3   Region_Code    210605 non-null  category
 4   Holiday        210605 non-null  bool
 5   Discount       210605 non-null  bool
```

```
 6   #Order         188340 non-null  float64
 7   Sales          188340 non-null  float64
 8   Train          210605 non-null  bool
dtypes: bool(3), category(3), float64(2), uint16(1)
memory usage: 6.4 MB
```

[17]:
```
data.describe()
```

[17]:

	Store_id	#Order	Sales
count	210605.000000	188340.000000	188340.000000
mean	183.000000	68.205692	42784.327982
std	105.366279	30.467415	18456.708302
min	1.000000	0.000000	0.000000
25%	92.000000	48.000000	30426.000000
50%	183.000000	63.000000	39678.000000
75%	274.000000	82.000000	51909.000000
max	365.000000	371.000000	247215.000000

[18]:
```
# Rename #Order column name for ease of use
data.rename(columns={'#Order':'Order'}, inplace=True)
```

[19]:
```
# Assign index to Exogenous variable dataframe
exog_holiday = data.Holiday
exog_discount = data.Discount
exog = pd.concat([exog_holiday, exog_discount, data.Train], axis=1)
exog_train = exog[exog.Train == True][['Holiday','Discount']]
exog_test = exog[exog.Train == False][['Holiday','Discount']]
```

1.4 4. Feature Engineering

[20]:
```
# Developing Features from date
data['Year'] = data.index.year
data['Quarter'] = data.index.quarter
data['Month'] = data.index.month
data['MonthName'] = data.index.month_name()
data['Day'] = data.index.day
data['Week'] = data.index.isocalendar().week
data['Weekday'] = data.index.weekday
data['DayName'] = data.index.day_name()
data['Weekend'] = data.Weekday.apply(lambda x: 'Weekend' if x in␣
 ↪['Saturday','Sunday'] else 'Weekday')
```

[21]:
```
# Additional features
data['S/O'] = round(data.Sales/data.Order,2)
```

[22]:
```
data.sample(5)
```

[22]:
```
           Store_id Store_Type Location_Type Region_Code  Holiday  Discount  \
Date
2019-06-23       332         S2            L5          R1    False     False
2019-03-31       195         S4            L1          R1    False     False
2018-12-20        56         S2            L5          R3    False      True
2019-05-19       290         S1            L1          R3    False     False
2019-06-19       238         S1            L1          R2    False     False

           Order    Sales  Train  Year  Quarter  Month MonthName  Day  Week  \
Date
2019-06-23   NaN      NaN  False  2019        2      6      June   23    25
2019-03-31  89.0  44193.0   True  2019        1      3     March   31    13
2018-12-20  46.0  34593.0   True  2018        4     12  December   20    51
2019-05-19  45.0  31848.0   True  2019        2      5       May   19    20
2019-06-19   NaN      NaN  False  2019        2      6      June   19    25

           Weekday    DayName  Weekend     S/O
Date
2019-06-23        6     Sunday  Weekday     NaN
2019-03-31        6     Sunday  Weekday  496.55
2018-12-20        3   Thursday  Weekday  752.02
2019-05-19        6     Sunday  Weekday  707.73
2019-06-19        2  Wednesday  Weekday     NaN
```

[23]:
```
# Split the data into train and test before proceeding further
train = data[data.Train == True]
test = data[data.Train == False]
```

1.5 5. EDA

[24]:
```
# **Univariate Analysis: Distribution of numerical data**
fig = make_subplots(rows=1, cols=3, subplot_titles=('Order','Sales','Sales per␣
↪Order'))
fig.add_trace(go.Histogram(x=train.Order, marker_color='teal'), row=1, col=1)
fig.add_trace(go.Histogram(x=train.Sales, marker_color='orange'), row=1, col=2)
fig.add_trace(go.Histogram(x=train['S/O'], marker_color='purple'), row=1, col=3)
fig.update_layout(title='Distribution of target parameters', showlegend=False,␣
↪title_x=0.5, title_y=0.1)
fig.show()
```

[25]:
```
# **Bivariate Analysis: Bar Charts**
order_color = 'darkgreen'
sales_color = 'teal'
fig = make_subplots(rows=2, cols=4)
grouped = train.groupby('Store_Type').agg({'Order':'sum','Sales':'sum'})
fig.add_trace(go.Bar(x=grouped.index, y=grouped.Order,␣
↪marker=dict(color=order_color)), row=1, col=1)
```

```python
fig.add_trace(go.Bar(x=grouped.index, y=grouped.Sales,␣
↪marker=dict(color=sales_color)), row=2, col=1)
grouped = train.groupby('Location_Type').agg({'Order':'sum','Sales':'sum'})
fig.add_trace(go.Bar(x=grouped.index, y=grouped.Order,␣
↪marker=dict(color=order_color)), row=1, col=2)
fig.add_trace(go.Bar(x=grouped.index, y=grouped.Sales,␣
↪marker=dict(color=sales_color)), row=2, col=2)
grouped = train.groupby('Region_Code').agg({'Order':'sum','Sales':'sum'})
fig.add_trace(go.Bar(x=grouped.index, y=grouped.Order,␣
↪marker=dict(color=order_color)), row=1, col=3)
fig.add_trace(go.Bar(x=grouped.index, y=grouped.Sales,␣
↪marker=dict(color=sales_color)), row=2, col=3)
grouped = train.groupby(['Weekday','DayName']).agg({'Order':'sum','Sales':
↪'sum'}).reset_index()
fig.add_trace(go.Bar(x=grouped.DayName, y=grouped.Order,␣
↪marker=dict(color=order_color)), row=1, col=4)
fig.add_trace(go.Bar(x=grouped.DayName, y=grouped.Sales,␣
↪marker=dict(color=sales_color)), row=2, col=4)
fig.update_layout(title='Order, Sales and Sales/Order distribution',␣
↪showlegend=False, title_x=0.5, title_y=0.85)

fig.update_yaxes(title='Order Volume', row=1, col=1)
fig.update_yaxes(title='Sales Amount', row=2, col=1)

fig.update_xaxes(title='Store Type', row=2, col=1)
fig.update_xaxes(title='Location Type', row=2, col=2)
fig.update_xaxes(title='Region Code', row=2, col=3)

fig.show()
```

[26]:
```python
# **Top/Bottom 10s**
fig = make_subplots(rows=2, cols=1, subplot_titles=('Top/Bottom 10 Store IDs',␣
↪''))
top_order = train.groupby('Store_id').agg({'Order':'sum'}).sort_values('Order',␣
↪ascending=False).head(10)
top_sales = train.groupby('Store_id').agg({'Sales':'sum'}).sort_values('Sales',␣
↪ascending=False).head(10)
bottom_order = train.groupby('Store_id').agg({'Order':'sum'}).
↪sort_values('Order', ascending=False).tail(10)
bottom_sales = train.groupby('Store_id').agg({'Sales':'sum'}).
↪sort_values('Sales', ascending=False).tail(10)
tb_order = pd.concat([top_order, bottom_order])
tb_sales = pd.concat([top_sales, bottom_sales])
fig.add_trace(go.Bar(x=tb_order.index, y=tb_order.Order, name='Order',␣
↪marker=dict(color=order_color)), row=1, col=1)
```

```
fig.add_trace(go.Bar(x=tb_sales.index, y=tb_sales.Sales, name='Sales',␣
↪marker=dict(color=sales_color)), row=2, col=1)
fig.update_layout(xaxis=dict(type='category'),
                  xaxis2=dict(type='category'),
                  yaxis=dict(title='Order Volume'),
                  yaxis2=dict(title='Sales Amount'),
                  showlegend=False, width=500)
fig.show()
```

[27]:
```
def scatter_plots(df, column):
  categories = df[column].astype('category').unique().sort_values()
  fig = make_subplots(rows=1, cols=len(categories),
                      subplot_titles=[str(c) for c in categories])
  for i, category in enumerate(categories):
    fig.add_trace(go.Scatter(x=df[df[column] == category]['Order'],
                             y=df[df[column] == category]['Sales'],
                             mode='markers', marker=dict(size=2),␣
↪name=category,),row=1, col=i+1)
    fig.update_xaxes(range = [0,300], row=1, col=i+1)
    fig.update_yaxes(range = [0,250000], row=1, col=i+1)
  fig.update_layout(title=f'{column} wise Order v/s Sales Scatter Plot', height␣
↪= 400, showlegend=False, title_x=0.5)
  fig.show()
```

[28]:
```
ignore = False
if not ignore:
  scatter_plots(train,'Store_Type')
  scatter_plots(train,'Region_Code')
  scatter_plots(train,'Location_Type')
  scatter_plots(train,'Holiday')
  scatter_plots(train,'Discount')
```

1.5.1 Hypothesis Testing

[29]:
```
# **Chi-Square test for dependency**
from scipy.stats import chi2_contingency

@decorator
def chi2test(data, category1, category2, alpha=0.05):
  data = data.groupby(by=[category1, category2]).agg({'Order':'sum', 'Sales':
↪'sum'}).reset_index()
  test = chi2_contingency(data.
↪pivot(index=category1,columns=category2,values='Order').fillna(0))
  order_dependency = test.pvalue < alpha
  if order_dependency:
```

```
        print(f'Reject the Null Hypothesis. For Order volume, {category1} and␣
↪{category2} are dependent', end=" | ")
    else:
        print(f'Fail to reject the Null Hypothesis. For Order volume, {category1}␣
↪and {category2} are independent', end=" | ")
    print(f'Test statistics:{test.statistic},\tp-value:{test.pvalue}')

    test = chi2_contingency(data.
↪pivot(index=category1,columns=category2,values='Sales').fillna(0))
    sales_dependency = test.pvalue < alpha
    if sales_dependency:
        print(f'Reject the Null Hypothesis. For Sales amount, {category1} and␣
↪{category2} are dependent', end=" | ")
    else:
        print(f'Fail to reject the Null Hypothesis. For Sales amount, {category1}␣
↪and {category2} are independent', end=" | ")
    print(f'Test statistics:{test.statistic},\tp-value:{test.pvalue}')

    return {'C1':category1, 'C2':category2,'Order': order_dependency, 'Sales':␣
↪sales_dependency}
```

[30]:
```
from itertools import permutations
columns = ['Store_Type', 'Location_Type', 'Region_Code', 'Holiday', 'Discount',␣
↪'MonthName', 'DayName']
dependancy_summary = pd.DataFrame([chi2test(train,c1,c2) for c1,c2 in␣
↪list(permutations(columns,2))])
```

```
==================================================
Reject the Null Hypothesis. For Order volume, Store_Type and Location_Type are
dependent | Test statistics:8560447.197307907,   p-value:0.0
Reject the Null Hypothesis. For Sales amount, Store_Type and Location_Type are
dependent | Test statistics:5347008627.585614,   p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Store_Type and Region_Code are
dependent | Test statistics:2063286.1134297573,    p-value:0.0
Reject the Null Hypothesis. For Sales amount, Store_Type and Region_Code are
dependent | Test statistics:1263592484.9694943,    p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Store_Type and Holiday are
dependent | Test statistics:27.94049935419796, p-value:3.7379941858049926e-06
Reject the Null Hypothesis. For Sales amount, Store_Type and Holiday are
dependent | Test statistics:21093.7298558246,  p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Store_Type and Discount are
```

```
dependent | Test statistics:321.25338064274575,
p-value:2.497032993795113e-69
Reject the Null Hypothesis. For Sales amount, Store_Type and Discount are
dependent | Test statistics:329382.9310341213,        p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Store_Type and MonthName are
dependent | Test statistics:767.6808790550784,
p-value:4.541885984630596e-140
Reject the Null Hypothesis. For Sales amount, Store_Type and MonthName are
dependent | Test statistics:554539.2424337461,       p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Store_Type and DayName are
dependent | Test statistics:75.9649544550106,  p-value:4.312391037787697e-09
Reject the Null Hypothesis. For Sales amount, Store_Type and DayName are
dependent | Test statistics:52319.79147868339, p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Location_Type and Store_Type are
dependent | Test statistics:8560447.197307909,   p-value:0.0
Reject the Null Hypothesis. For Sales amount, Location_Type and Store_Type are
dependent | Test statistics:5347008627.585613,   p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Location_Type and Region_Code are
dependent | Test statistics:373016.05851354415, p-value:0.0
Reject the Null Hypothesis. For Sales amount, Location_Type and Region_Code are
dependent | Test statistics:228540888.13728154, p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Location_Type and Holiday are
dependent | Test statistics:21.222613057623803,
p-value:0.00028605478639954726
Reject the Null Hypothesis. For Sales amount, Location_Type and Holiday are
dependent | Test statistics:21296.647804881504,     p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Location_Type and Discount are
dependent | Test statistics:1260.3760430857506,
p-value:1.2971148482759126e-271
Reject the Null Hypothesis. For Sales amount, Location_Type and Discount are
dependent | Test statistics:724101.2360874555,     p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Location_Type and MonthName are
dependent | Test statistics:439.16646867455864,   p-value:1.3959817749676116e-66
Reject the Null Hypothesis. For Sales amount, Location_Type and MonthName are
```

```
dependent | Test statistics:307715.5842903906,    p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Location_Type and DayName are
dependent | Test statistics:50.63707711934928,
p-value:0.0011740360390016633
Reject the Null Hypothesis. For Sales amount, Location_Type and DayName are
dependent | Test statistics:38344.05165385948,      p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Region_Code and Store_Type are
dependent | Test statistics:2063286.1134297573,    p-value:0.0
Reject the Null Hypothesis. For Sales amount, Region_Code and Store_Type are
dependent | Test statistics:1263592484.9694943,    p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Region_Code and Location_Type are
dependent | Test statistics:373016.05851354403, p-value:0.0
Reject the Null Hypothesis. For Sales amount, Region_Code and Location_Type are
dependent | Test statistics:228540888.13728154, p-value:0.0
==================================================
==================================================
Fail to reject the Null Hypothesis. For Order volume, Region_Code and Holiday
are independent | Test statistics:5.22312907474964,
p-value:0.15616892563963025
Reject the Null Hypothesis. For Sales amount, Region_Code and Holiday are
dependent | Test statistics:4415.458954381414,        p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Region_Code and Discount are
dependent | Test statistics:726.5818658215513,
p-value:3.6135273295344166e-157
Reject the Null Hypothesis. For Sales amount, Region_Code and Discount are
dependent | Test statistics:567114.1191712606,       p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Region_Code and MonthName are
dependent | Test statistics:852.9791229987399,
p-value:6.956453417684306e-158
Reject the Null Hypothesis. For Sales amount, Region_Code and MonthName are
dependent | Test statistics:669584.5488212734,      p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Region_Code and DayName are
dependent | Test statistics:30.149954668791192,
p-value:0.03601386919680342
Reject the Null Hypothesis. For Sales amount, Region_Code and DayName are
dependent | Test statistics:29839.71437344468,        p-value:0.0
```

```
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Holiday and Store_Type are
dependent | Test statistics:27.94049935419796, p-value:3.7379941858049926e-06
Reject the Null Hypothesis. For Sales amount, Holiday and Store_Type are
dependent | Test statistics:21093.729855824597,        p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Holiday and Location_Type are
dependent | Test statistics:21.2226130576238,
p-value:0.00028605478639954 76
Reject the Null Hypothesis. For Sales amount, Holiday and Location_Type are
dependent | Test statistics:21296.647804881504,     p-value:0.0
==================================================
==================================================
Fail to reject the Null Hypothesis. For Order volume, Holiday and Region_Code
are independent | Test statistics:5.2231290747496395,
p-value:0.15616892563963033
Reject the Null Hypothesis. For Sales amount, Holiday and Region_Code are
dependent | Test statistics:4415.458954381414,        p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Holiday and Discount are dependent
| Test statistics:1531.4534101037711,  p-value:0.0
Reject the Null Hypothesis. For Sales amount, Holiday and Discount are dependent
| Test statistics:569983.2461383714,   p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Holiday and MonthName are
dependent | Test statistics:329278.4500446775,  p-value:0.0
Reject the Null Hypothesis. For Sales amount, Holiday and MonthName are
dependent | Test statistics:206015674.04294717, p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Holiday and DayName are dependent
| Test statistics:70922.4490624049,     p-value:0.0
Reject the Null Hypothesis. For Sales amount, Holiday and DayName are dependent
| Test statistics:40319768.89511568,    p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Discount and Store_Type are
dependent | Test statistics:321.2533806427457,
p-value:2.4970329937951835e-69
Reject the Null Hypothesis. For Sales amount, Discount and Store_Type are
dependent | Test statistics:329382.9310341213,        p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Discount and Location_Type are
```

```
dependent | Test statistics:1260.3760430857506,
p-value:1.2971148482759126e-271
Reject the Null Hypothesis. For Sales amount, Discount and Location_Type are
dependent | Test statistics:724101.2360874556,     p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Discount and Region_Code are
dependent | Test statistics:726.5818658215514,
p-value:3.61352732953421e-157
Reject the Null Hypothesis. For Sales amount, Discount and Region_Code are
dependent | Test statistics:567114.1191712606,       p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Discount and Holiday are dependent
| Test statistics:1531.4534101037711,  p-value:0.0
Reject the Null Hypothesis. For Sales amount, Discount and Holiday are dependent
| Test statistics:569983.2461383714,   p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Discount and MonthName are
dependent | Test statistics:97104.88049468104, p-value:0.0
Reject the Null Hypothesis. For Sales amount, Discount and MonthName are
dependent | Test statistics:59894831.49811946, p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, Discount and DayName are dependent
| Test statistics:14454.702562989041,  p-value:0.0
Reject the Null Hypothesis. For Sales amount, Discount and DayName are dependent
| Test statistics:9301326.292060012,   p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, MonthName and Store_Type are
dependent | Test statistics:767.6808790550785,
p-value:4.541885984630337e-140
Reject the Null Hypothesis. For Sales amount, MonthName and Store_Type are
dependent | Test statistics:554539.2424337475,       p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, MonthName and Location_Type are
dependent | Test statistics:439.1664686745586,    p-value:1.3959817749676514e-66
Reject the Null Hypothesis. For Sales amount, MonthName and Location_Type are
dependent | Test statistics:307715.5842903907,    p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, MonthName and Region_Code are
dependent | Test statistics:852.97912299874,
p-value:6.956453417683911e-158
Reject the Null Hypothesis. For Sales amount, MonthName and Region_Code are
```

```
dependent | Test statistics:669584.5488212737,      p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, MonthName and Holiday are
dependent | Test statistics:329278.4500446775,  p-value:0.0
Reject the Null Hypothesis. For Sales amount, MonthName and Holiday are
dependent | Test statistics:206015674.04294714, p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, MonthName and Discount are
dependent | Test statistics:97104.88049468104, p-value:0.0
Reject the Null Hypothesis. For Sales amount, MonthName and Discount are
dependent | Test statistics:59894831.4981195,  p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, MonthName and DayName are
dependent | Test statistics:115360.52602159233, p-value:0.0
Reject the Null Hypothesis. For Sales amount, MonthName and DayName are
dependent | Test statistics:77573972.75065999,  p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, DayName and Store_Type are
dependent | Test statistics:75.96495445501058, p-value:4.312391037787728e-09
Reject the Null Hypothesis. For Sales amount, DayName and Store_Type are
dependent | Test statistics:52319.79147868339, p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, DayName and Location_Type are
dependent | Test statistics:50.637077119349286,
p-value:0.0011740360390016587
Reject the Null Hypothesis. For Sales amount, DayName and Location_Type are
dependent | Test statistics:38344.05165385942,      p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, DayName and Region_Code are
dependent | Test statistics:30.149954668791196,
p-value:0.036013869196803355
Reject the Null Hypothesis. For Sales amount, DayName and Region_Code are
dependent | Test statistics:29839.71437344464,        p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, DayName and Holiday are dependent
| Test statistics:70922.4490624049,     p-value:0.0
Reject the Null Hypothesis. For Sales amount, DayName and Holiday are dependent
| Test statistics:40319768.89511567,    p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, DayName and Discount are dependent
```

```
| Test statistics:14454.70256298904,   p-value:0.0
Reject the Null Hypothesis. For Sales amount, DayName and Discount are dependent
| Test statistics:9301326.292059988,   p-value:0.0
==================================================
==================================================
Reject the Null Hypothesis. For Order volume, DayName and MonthName are
dependent | Test statistics:115360.52602159233, p-value:0.0
Reject the Null Hypothesis. For Sales amount, DayName and MonthName are
dependent | Test statistics:77573972.75065999,  p-value:0.0
==================================================
```

[31]:
```
pd.crosstab(dependancy_summary.C1, dependancy_summary.C2, dependancy_summary.
    ↪Order, aggfunc='max')
```

[31]:
```
C2             DayName Discount Holiday Location_Type MonthName Region_Code  \
C1
DayName            NaN     True    True          True      True        True
Discount          True      NaN    True          True      True        True
Holiday           True     True     NaN          True      True       False
Location_Type     True     True    True           NaN      True        True
MonthName         True     True    True          True       NaN        True
Region_Code       True     True   False          True      True         NaN
Store_Type        True     True    True          True      True        True

C2             Store_Type
C1
DayName              True
Discount             True
Holiday              True
Location_Type        True
MonthName            True
Region_Code          True
Store_Type            NaN
```

[32]:
```
pd.crosstab(dependancy_summary.C1, dependancy_summary.C2, dependancy_summary.
    ↪Sales, aggfunc='max').fillna(0)
```

[32]:
```
C2             DayName Discount Holiday Location_Type MonthName Region_Code  \
C1
DayName              0     True    True          True      True        True
Discount          True        0    True          True      True        True
Holiday           True     True       0          True      True        True
Location_Type     True     True    True             0      True        True
MonthName         True     True    True          True         0        True
Region_Code       True     True    True          True      True           0
Store_Type        True     True    True          True      True        True
```

```
C2            Store_Type
C1
DayName             True
Discount            True
Holiday             True
Location_Type       True
MonthName           True
Region_Code         True
Store_Type             0
```

[33]:
```
# **Mean similarity test**
from scipy.stats import f_oneway, kruskal, anderson, levene
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from itertools import combinations

def decorator(func):
  def wrapper(*args, **kwargs):
    print('~'*100)
    print('~'*100)
    result = func(*args, **kwargs)
    print('~'*100)
    print('~'*100)
    return result
  return wrapper

@decorator
def variance_test(data, category, target, alpha=0.05):
  d = data.groupby(by=category).agg(Mean=(target,'mean'), Count=(target,␣
↪'size')).reset_index()
  print(f'Hypothesis test whether Mean {target} is same for all {category} or␣
↪not.\n')
  print(d)
  print('='*53)
  cats = sorted(data[category].unique())
  groups = {}
  for cat in cats:
     groups[cat]=data[data[category] == cat][target]

  # Check for Normality test of all categories
  normality_test = True
  print('Criteria check for ANOVA')
  for cat,group in groups.items():
    if not anderson(group).fit_result.success:
      normality_test = False
      print(f'\033[31m \u274C Group {cat} is not normally distributed.\033[0m')
      break
  if normality_test:
```

```
  print('\033[32m \u2705 All groups are normally distributed.\033[0m')
# Check for levene test
levene_test = True
_, p_levene = levene(*groups.values())
if p_levene < alpha:
  levene_test = False
  print(f'\033[31m \u274C Variance of all groups are not same.\033[0m')
else:
  print(f'\033[32m \u2705 Variance of all groups are same.\033[0m')

# Perform One-way ANOVA if criteria meets otheriwse perform Kruskal
if normality_test and levene_test:
  print('One-Way ANOVA will be performed.')
  _, p_value = f_oneway(*groups.values())
else:
  print('All criterias not met for ANOVA. Kruskal test will be performed.')
  _, p_value = kruskal(*groups.values())

# Proceed for ttest_ind if one group has different mean
if p_value > alpha:
  print(f'p-Value is {p_value} > {alpha} Significance level.\nWe dont have enough evidence to  reject the Null Hypothesis. All means are same.')
  print('='*53)
  return None
else:
  print(f'p-Value is {p_value} < {alpha} Significance level.\nWe have enough evidence to reject the Null Hypothesis and at least one mean is different.')
  print('='*53)

tukey = pairwise_tukeyhsd(endog=data[target], groups=data[category], alpha=0.05)
print(tukey)
# Extract group1 and group2 using the Tukey object attributes
group1 = tukey.groupsunique[tukey._multicomp.pairindices[0]]
group2 = tukey.groupsunique[tukey._multicomp.pairindices[1]]
pair = [f'{a}-{b}' for a,b in list(zip(group1, group2))]
reject = tukey.reject

# Combine group1 and group2 into a DataFrame
group_pairs = pd.DataFrame({'pair': pair, 'reject': reject})
same_mean_pairs = group_pairs[group_pairs['reject'] == False]['pair']
different_mean_pairs = group_pairs[group_pairs['reject'] == True]['pair']
print(f'\033[34mPairs having different {target} mean are: {",".join(different_mean_pairs.values)}')
print(f'\033[35mPairs having same {target} mean are: {",".join(same_mean_pairs.values)}\033[0m')
```

```
  return None
```

[34]:
```
from itertools import product
for category, target in product(columns, ['Order', 'Sales']):
  variance_test(train, category, target)
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Order is same for all Store_Type or not.

  Store_Type        Mean  Count
0         S1   58.022095  88752
1         S2   40.472799  28896
2         S3   73.663396  24768
3         S4  102.392779  45924
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj  lower    upper   reject
-----------------------------------------------------
    S1     S2 -17.5493   0.0 -17.9275 -17.1711   True
    S1     S3  15.6413   0.0    15.24  16.0426   True
    S1     S4  44.3707   0.0  44.0497  44.6917   True
    S2     S3  33.1906   0.0   32.707  33.6742   True
    S2     S4    61.92   0.0  61.5007  62.3393   True
    S3     S4  28.7294   0.0  28.2891  29.1696   True
-----------------------------------------------------
Pairs having different Order mean are: S1-S2,S1-S3,S1-S4,S2-S3,S2-S4,S3-S4

Pairs having same Order mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
```

```
Hypothesis test whether Mean Sales is same for all Store_Type or not.

  Store_Type          Mean  Count
0         S1  37676.511694  88752
1         S2  27530.828222  28896
2         S3  47063.068209  24768
3         S4  59945.685926  45924
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
group1 group2   meandiff  p-adj     lower      upper    reject
-------------------------------------------------------------
    S1     S2 -10145.6835   0.0 -10402.8539  -9888.513   True
    S1     S3   9386.5565   0.0   9113.6974  9659.4156   True
    S1     S4  22269.1742   0.0  22050.9148 22487.4336   True
    S2     S3    19532.24   0.0  19203.4535 19861.0265   True
    S2     S4  32414.8577   0.0  32129.7514 32699.9641   True
    S3     S4  12882.6177   0.0  12583.2834 13181.9521   True
-------------------------------------------------------------
Pairs having different Sales mean are: S1-S2,S1-S3,S1-S4,S2-S3,S2-S4,S3-S4

Pairs having same Sales mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Order is same for all Location_Type or not.

  Location_Type       Mean  Count
0            L1  65.265938  85140
1            L2  94.851456  48504
2            L3  53.156943  29928
3            L4  47.386028  10836
4            L5  41.924131  13932
=====================================================
Criteria check for ANOVA
```

```
    All groups are normally distributed.
    Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj  lower    upper   reject
-----------------------------------------------------
    L1     L2  29.5855   0.0  29.1962  29.9749   True
    L1     L3  -12.109   0.0 -12.5689 -11.6491   True
    L1     L4 -17.8799   0.0  -18.578 -17.1819   True
    L1     L5 -23.3418   0.0 -23.9673 -22.7163   True
    L2     L3 -41.6945   0.0 -42.1976 -41.1914   True
    L2     L4 -47.4654   0.0 -48.1926 -46.7382   True
    L2     L5 -52.9273   0.0 -53.5852 -52.2695   True
    L3     L4  -5.7709   0.0  -6.5382  -5.0036   True
    L3     L5 -11.2328   0.0 -11.9347 -10.5309   True
    L4     L5  -5.4619   0.0  -6.3385  -4.5853   True
-----------------------------------------------------
Pairs having different Order mean are:

L1-L2,L1-L3,L1-L4,L1-L5,L2-L3,L2-L4,L2-L5,L3-L4,L3-L5,L4-L5

Pairs having same Order mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Sales is same for all Location_Type or not.

  Location_Type          Mean  Count
0            L1  41453.597889  85140
1            L2  59231.480373  48504
2            L3  33072.257756  29928
3            L4  29067.414313  10836
4            L5  25187.787261  13932
=====================================================
Criteria check for ANOVA
    All groups are normally distributed.
    Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
```

```
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
==============================================================
group1 group2   meandiff  p-adj    lower       upper    reject
--------------------------------------------------------------
    L1     L2  17777.8825   0.0  17546.9501  18008.8149   True
    L1     L3  -8381.3401   0.0  -8654.1357  -8108.5446   True
    L1     L4 -12386.1836   0.0 -12800.2268 -11972.1403   True
    L1     L5 -16265.8106   0.0 -16636.8052  -15894.816   True
    L2     L3 -26159.2226   0.0 -26457.6129 -25860.8323   True
    L2     L4 -30164.0661   0.0 -30595.4025 -29732.7296   True
    L2     L5 -34043.6931   0.0 -34433.8935 -33653.4927   True
    L3     L4  -4004.8434   0.0  -4459.9685  -3549.7184   True
    L3     L5  -7884.4705   0.0  -8300.8165  -7468.1245   True
    L4     L5  -3879.6271   0.0  -4399.5871   -3359.667   True
--------------------------------------------------------------
Pairs having different Sales mean are:

L1-L2,L1-L3,L1-L4,L1-L5,L2-L3,L2-L4,L2-L5,L3-L4,L3-L5,L4-L5

Pairs having same Sales mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Order is same for all Region_Code or not.

  Region_Code       Mean  Count
0          R1  79.626063  63984
1          R2  62.798616  54180
2          R3  63.882436  44376
3          R4  58.674031  25800
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
```

```
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj  lower    upper   reject
-----------------------------------------------------
    R1     R2 -16.8274   0.0  -17.267 -16.3879   True
    R1     R3 -15.7436   0.0 -16.2087 -15.2786   True
    R1     R4  -20.952   0.0 -21.5072 -20.3968   True
    R2     R3   1.0838   0.0   0.6018   1.5658   True
    R2     R4  -4.1246   0.0   -4.694  -3.5551   True
    R3     R4  -5.2084   0.0  -5.7978   -4.619   True
-----------------------------------------------------
Pairs having different Order mean are: R1-R2,R1-R3,R1-R4,R2-R3,R2-R4,R3-R4
Pairs having same Order mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Sales is same for all Region_Code or not.

  Region_Code          Mean  Count
0          R1  46765.488405  63984
1          R2  40054.847344  54180
2          R3  42144.517063  44376
3          R4  39743.434249  25800
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
group1 group2  meandiff  p-adj    lower      upper    reject
------------------------------------------------------------
    R1     R2 -6710.6411    0.0 -6983.8355 -6437.4467   True
    R1     R3 -4620.9713    0.0 -4910.0454 -4331.8973   True
    R1     R4 -7022.0542    0.0 -7367.1489 -6676.9594   True
    R2     R3  2089.6697    0.0  1790.0762  2389.2632   True
    R2     R4  -311.4131 0.1075  -665.3662      42.54  False
    R3     R4 -2401.0828    0.0 -2767.4316  -2034.734   True
```

```
------------------------------------------------------------
Pairs having different Sales mean are: R1-R2,R1-R3,R1-R4,R2-R3,R3-R4

Pairs having same Sales mean are: R2-R4
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Order is same for all Holiday or not.

   Holiday       Mean   Count
0    False  69.873379  163520
1     True  57.218574   24820
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj  lower    upper   reject
-----------------------------------------------------
 False   True -12.6548   0.0 -13.0576 -12.2521   True
-----------------------------------------------------
Pairs having different Order mean are: False-True

Pairs having same Order mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Sales is same for all Holiday or not.

   Holiday          Mean   Count
0    False  43897.288998  163520
1     True  35451.878930   24820
```

```
========================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
========================================================
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
==============================================================
group1 group2  meandiff  p-adj   lower      upper    reject
--------------------------------------------------------------
 False   True -8445.4101   0.0 -8688.8675 -8201.9526   True
--------------------------------------------------------------
Pairs having different Sales mean are: False-True
Pairs having same Sales mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Order is same for all Discount or not.

   Discount       Mean   Count
0     False  61.806153  104051
1      True  76.105637   84289
========================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
========================================================
Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj  lower   upper  reject
----------------------------------------------------
 False   True  14.2995   0.0 14.0304 14.5686   True
----------------------------------------------------
Pairs having different Order mean are: False-True
Pairs having same Order mean are:
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Sales is same for all Discount or not.

   Discount          Mean   Count
0     False  37403.679678  104051
1      True  49426.497620   84289
=====================================================
Criteria check for ANOVA
  All groups are normally distributed.
  Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
===========================================================
group1 group2  meandiff  p-adj   lower      upper    reject
-----------------------------------------------------------
 False   True 12022.8179   0.0 11864.2197 12181.4162   True
-----------------------------------------------------------
Pairs having different Sales mean are: False-True

Pairs having same Sales mean are:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Order is same for all MonthName or not.

    MonthName       Mean  Count
0       April  68.212968  21900
1      August  67.128502  11315
2    December  69.479806  11315
3    February  67.453474  20440
4     January  66.933672  22630
5        July  76.048873  11315
```

```
6        June  66.174155  10950
7       March  67.761688  22630
8         May  71.100044  22630
9    November  63.416438  10950
10    October  65.460009  11315
11  September  68.509954  10950
=====================================================
Criteria check for ANOVA
  All groups are normally distributed.
  Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 1.7657332953564353e-282 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
===========================================================
 group1    group2  meandiff p-adj   lower    upper   reject
-----------------------------------------------------------
   April    August  -1.0845 0.0855  -2.2328   0.0639  False
   April  December   1.2668 0.0163   0.1185   2.4152   True
   April  February  -0.7595  0.295  -1.7241   0.2051  False
   April   January  -1.2793 0.0005  -2.2195  -0.3391   True
   April      July   7.8359    0.0   6.6876   8.9842   True
   April      June  -2.0388    0.0  -3.1997  -0.8779   True
   April     March  -0.4513 0.9204  -1.3915   0.4889  False
   April       May   2.8871    0.0   1.9469   3.8272   True
   April  November  -4.7965    0.0  -5.9574  -3.6357   True
   April   October   -2.753    0.0  -3.9013  -1.6046   True
   April September    0.297 0.9996  -0.8639   1.4579  False
  August  December   2.3513    0.0   1.0326     3.67   True
  August  February    0.325  0.999  -0.8372   1.4872  False
  August   January  -0.1948    1.0  -1.3368   0.9472  False
  August      July   8.9204    0.0   7.6017   10.239   True
  August      June  -0.9543 0.4438  -2.2839   0.3753  False
  August     March   0.6332 0.8118  -0.5088   1.7752  False
  August       May   3.9715    0.0   2.8296   5.1135   True
  August  November  -3.7121    0.0  -5.0417  -2.3825   True
  August   October  -1.6685 0.0021  -2.9872  -0.3498   True
  August September   1.3815 0.0334   0.0519   2.7111   True
December  February  -2.0263    0.0  -3.1885  -0.8641   True
December   January  -2.5461    0.0  -3.6881  -1.4041   True
December      July   6.5691    0.0   5.2504   7.8877   True
December      June  -3.3057    0.0  -4.6353   -1.976   True
December     March  -1.7181 0.0001  -2.8601  -0.5761   True
December       May   1.6202 0.0002   0.4782   2.7622   True
December  November  -6.0634    0.0   -7.393  -4.7338   True
December   October  -4.0198    0.0  -5.3385  -2.7011   True
```

December	September	-0.9699	0.417	-2.2995	0.3598	False
February	January	-0.5198	0.8319	-1.4769	0.4373	False
February	July	8.5954	0.0	7.4332	9.7576	True
February	June	-1.2793	0.0192	-2.4539	-0.1047	True
February	March	0.3082	0.9964	-0.6489	1.2653	False
February	May	3.6466	0.0	2.6895	4.6037	True
February	November	-4.037	0.0	-5.2116	-2.8624	True
February	October	-1.9935	0.0	-3.1557	-0.8313	True
February	September	1.0565	0.1272	-0.1181	2.2311	False
January	July	9.1152	0.0	7.9732	10.2572	True
January	June	-0.7595	0.5865	-1.9141	0.3951	False
January	March	0.828	0.14	-0.1044	1.7604	False
January	May	4.1664	0.0	3.2339	5.0988	True
January	November	-3.5172	0.0	-4.6718	-2.3626	True
January	October	-1.4737	0.0015	-2.6157	-0.3317	True
January	September	1.5763	0.0005	0.4217	2.7309	True
July	June	-9.8747	0.0	-11.2043	-8.5451	True
July	March	-8.2872	0.0	-9.4292	-7.1452	True
July	May	-4.9488	0.0	-6.0908	-3.8068	True
July	November	-12.6324	0.0	-13.962	-11.3028	True
July	October	-10.5889	0.0	-11.9075	-9.2702	True
July	September	-7.5389	0.0	-8.8685	-6.2093	True
June	March	1.5875	0.0004	0.4329	2.7421	True
June	May	4.9259	0.0	3.7713	6.0805	True
June	November	-2.7577	0.0	-4.0982	-1.4173	True
June	October	-0.7141	0.8421	-2.0437	0.6155	False
June	September	2.3358	0.0	0.9953	3.6763	True
March	May	3.3384	0.0	2.4059	4.2708	True
March	November	-4.3452	0.0	-5.4999	-3.1906	True
March	October	-2.3017	0.0	-3.4437	-1.1597	True
March	September	0.7483	0.6098	-0.4063	1.9029	False
May	November	-7.6836	0.0	-8.8382	-6.529	True
May	October	-5.64	0.0	-6.782	-4.498	True
May	September	-2.5901	0.0	-3.7447	-1.4355	True
November	October	2.0436	0.0	0.714	3.3732	True
November	September	5.0935	0.0	3.7531	6.434	True
October	September	3.0499	0.0	1.7203	4.3795	True

```
Pairs having different Order mean are: April-December,April-January,April-
July,April-June,April-May,April-November,April-October,August-December,August-
July,August-May,August-November,August-October,August-September,December-
February,December-January,December-July,December-June,December-March,December-
May,December-November,December-October,February-July,February-June,February-
May,February-November,February-October,January-July,January-May,January-
November,January-October,January-September,July-June,July-March,July-May,July-
November,July-October,July-September,June-March,June-May,June-November,June-
September,March-May,March-November,March-October,May-November,May-October,May-
September,November-October,November-September,October-September
Pairs having same Order mean are: April-August,April-February,April-
March,April-September,August-February,August-January,August-June,August-
March,December-September,February-January,February-March,February-
September,January-June,January-March,June-October,March-September
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Sales is same for all MonthName or not.

    MonthName          Mean  Count
0       April  40773.681352  21900
1      August  40020.368869  11315
2    December  46477.110199  11315
3    February  40424.350645  20440
4     January  44979.147732  22630
5        July  46585.406232  11315
6        June  44705.726389  10950
7       March  40979.577286  22630
8         May  48115.830407  22630
9    November  38160.962496  10950
10    October  38988.407398  11315
11  September  41123.184822  10950
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
```

```
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
 group1    group2   meandiff  p-adj    lower      upper    reject
-----------------------------------------------------------------
   April    August  -753.3125  0.018 -1441.2331   -65.3918   True
   April  December  5703.4288    0.0  5015.5082  6391.3495   True
   April  February  -349.3307 0.7103  -927.2051   228.5437  False
   April   January  4205.4664    0.0  3642.2406  4768.6922   True
   April      July  5811.7249    0.0  5123.8042  6499.6455   True
   April      June   3932.045    0.0  3236.6059  4627.4842   True
   April     March   205.8959 0.9895  -357.3299   769.1218  False
   April       May  7342.1491    0.0  6778.9232  7905.3749   True
   April  November -2612.7189    0.0  -3308.158 -1917.2797   True
   April   October  -1785.274    0.0 -2473.1946 -1097.3533   True
   April September   349.5035 0.8934  -345.9356  1044.9426  False
  August  December  6456.7413    0.0  5666.7756  7246.7071   True
  August  February   403.9818  0.762  -292.2581  1100.2216  False
  August   January  4958.7789    0.0  4274.6485  5642.9093   True
  August      July  6565.0374    0.0  5775.0716  7355.0031   True
  August      June  4685.3575    0.0  3888.8359  5481.8791   True
  August     March   959.2084 0.0003    275.078  1643.3388   True
  August       May  8095.4615    0.0  7411.3311  8779.5919   True
  August  November -1859.4064    0.0  -2655.928 -1062.8848   True
  August   October -1031.9615 0.0012 -1821.9272  -241.9957   True
  August September   1102.816 0.0004   306.2944  1899.3376   True
December  February -6052.7596    0.0 -6748.9994 -5356.5197   True
December   January -1497.9625    0.0 -2182.0929  -813.8321   True
December      July    108.296    1.0  -681.6697   898.2618  False
December      June -1771.3838    0.0 -2567.9054  -974.8622   True
December     March -5497.5329    0.0 -6181.6633 -4813.4025   True
December       May  1638.7202    0.0   954.5898  2322.8506   True
December  November -8316.1477    0.0 -9112.6693 -7519.6261   True
December   October -7488.7028    0.0 -8278.6686  -6698.737   True
December September -5353.9254    0.0  -6150.447 -4557.4038   True
February   January  4554.7971    0.0  3981.4399  5128.1543   True
February      July  6161.0556    0.0  5464.8157  6857.2954   True
February      June  4281.3757    0.0  3577.7063  4985.0452   True
February     March   555.2266 0.0682   -18.1305  1128.5838  False
February       May  7691.4798    0.0  7118.1226  8264.8369   True
February  November -2263.3881    0.0 -2967.0576 -1559.7187   True
February   October -1435.9432    0.0 -2132.1831  -739.7034   True
February September   698.8342 0.0536    -4.8353  1402.5036  False
 January      July  1606.2585    0.0   922.1281  2290.3889   True
```

January	June	-273.4213	0.9803	-965.1114	418.2687	False
January	March	-3999.5704	0.0	-4558.1606	-3440.9803	True
January	May	3136.6827	0.0	2578.0925	3695.2728	True
January	November	-6818.1852	0.0	-7509.8753	-6126.4951	True
January	October	-5990.7403	0.0	-6674.8707	-5306.6099	True
January	September	-3855.9629	0.0	-4547.653	-3164.2728	True
July	June	-1879.6798	0.0	-2676.2014	-1083.1582	True
July	March	-5605.8289	0.0	-6289.9594	-4921.6985	True
July	May	1530.4242	0.0	846.2938	2214.5546	True
July	November	-8424.4437	0.0	-9220.9653	-7627.9221	True
July	October	-7596.9988	0.0	-8386.9646	-6807.0331	True
July	September	-5462.2214	0.0	-6258.743	-4665.6998	True
June	March	-3726.1491	0.0	-4417.8392	-3034.459	True
June	May	3410.104	0.0	2718.4139	4101.7941	True
June	November	-6544.7639	0.0	-7347.7878	-5741.74	True
June	October	-5717.319	0.0	-6513.8406	-4920.7974	True
June	September	-3582.5416	0.0	-4385.5655	-2779.5176	True
March	May	7136.2531	0.0	6577.663	7694.8433	True
March	November	-2818.6148	0.0	-3510.3049	-2126.9247	True
March	October	-1991.1699	0.0	-2675.3003	-1307.0395	True
March	September	143.6075	0.9999	-548.0826	835.2976	False
May	November	-9954.8679	0.0	-10646.558	-9263.1778	True
May	October	-9127.423	0.0	-9811.5534	-8443.2926	True
May	September	-6992.6456	0.0	-7684.3357	-6300.9555	True
November	October	827.4449	0.0334	30.9233	1623.9665	True
November	September	2962.2223	0.0	2159.1984	3765.2462	True
October	September	2134.7774	0.0	1338.2558	2931.299	True

```
Pairs having different Sales mean are: April-August,April-December,April-
January,April-July,April-June,April-May,April-November,April-October,August-
December,August-January,August-July,August-June,August-March,August-May,August-
November,August-October,August-September,December-February,December-
January,December-June,December-March,December-May,December-November,December-
October,December-September,February-January,February-July,February-
June,February-May,February-November,February-October,January-July,January-
March,January-May,January-November,January-October,January-September,July-
June,July-March,July-May,July-November,July-October,July-September,June-
March,June-May,June-November,June-October,June-September,March-May,March-
November,March-October,May-November,May-October,May-September,November-
October,November-September,October-September
Pairs having same Sales mean are: April-February,April-March,April-
September,August-February,December-July,February-March,February-
September,January-June,March-September
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Order is same for all DayName or not.

     DayName       Mean  Count
0     Friday  63.507812  27010
1     Monday  66.164939  27010
2   Saturday  75.887934  26645
3     Sunday  77.694389  26645
4   Thursday  64.140244  27010
5    Tuesday  65.198001  27010
6  Wednesday  65.078563  27010
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
```

```
different.
=====================================================
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
===========================================================
 group1    group2  meandiff p-adj   lower    upper   reject
-----------------------------------------------------------
  Friday    Monday   2.6571    0.0   1.8967   3.4176   True
  Friday  Saturday  12.3801    0.0  11.6171  13.1431   True
  Friday    Sunday  14.1866    0.0  13.4236  14.9496   True
  Friday  Thursday   0.6324 0.1772   -0.128   1.3929  False
  Friday   Tuesday   1.6902    0.0   0.9298   2.4506   True
  Friday Wednesday   1.5708    0.0   0.8103   2.3312   True
  Monday  Saturday    9.723    0.0     8.96   10.486   True
  Monday    Sunday  11.5295    0.0  10.7664  12.2925   True
  Monday  Thursday  -2.0247    0.0  -2.7851  -1.2643   True
  Monday   Tuesday  -0.9669 0.0034  -1.7274  -0.2065   True
  Monday Wednesday  -1.0864 0.0005  -1.8468   -0.326   True
Saturday    Sunday   1.8065    0.0   1.0408   2.5721   True
Saturday  Thursday -11.7477    0.0 -12.5107 -10.9847   True
Saturday   Tuesday -10.6899    0.0  -11.453  -9.9269   True
Saturday Wednesday -10.8094    0.0 -11.5724 -10.0463   True
  Sunday  Thursday -13.5541    0.0 -14.3172 -12.7911   True
  Sunday   Tuesday -12.4964    0.0 -13.2594 -11.7334   True
  Sunday Wednesday -12.6158    0.0 -13.3788 -11.8528   True
Thursday   Tuesday   1.0578 0.0008   0.2973   1.8182   True
Thursday Wednesday   0.9383 0.0051   0.1779   1.6987   True
 Tuesday Wednesday  -0.1194 0.9993  -0.8799    0.641  False
-----------------------------------------------------------
Pairs having different Order mean are: Friday-Monday,Friday-
Saturday,Friday-Sunday,Friday-Tuesday,Friday-Wednesday,Monday-Saturday,Monday-
Sunday,Monday-Thursday,Monday-Tuesday,Monday-Wednesday,Saturday-Sunday,Saturday-
Thursday,Saturday-Tuesday,Saturday-Wednesday,Sunday-Thursday,Sunday-
Tuesday,Sunday-Wednesday,Thursday-Tuesday,Thursday-Wednesday
Pairs having same Order mean are: Friday-Thursday,Tuesday-Wednesday
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Hypothesis test whether Mean Sales is same for all DayName or not.

     DayName          Mean  Count
```

```
0     Friday  39701.020376  27010
1     Monday  42291.175854  27010
2   Saturday  46729.798143  26645
3     Sunday  49044.051947  26645
4   Thursday  40231.985963  27010
5    Tuesday  40802.966220  27010
6  Wednesday  40827.205395  27010
=====================================================
Criteria check for ANOVA
   All groups are normally distributed.
   Variance of all groups are not same.
All criterias not met for ANOVA. Kruskal test will be performed.
p-Value is 0.0 < 0.05 Significance level.
We have enough evidence to reject the Null Hypothesis and at least one mean is
different.
=====================================================
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
 group1    group2   meandiff  p-adj    lower      upper    reject
-----------------------------------------------------------------
  Friday    Monday  2590.1555    0.0  2129.6495  3050.6614   True
  Friday  Saturday  7028.7778    0.0  6566.6975  7490.8581   True
  Friday    Sunday  9343.0316    0.0  8880.9513  9805.1119   True
  Friday  Thursday   530.9656  0.012    70.4597   991.4715   True
  Friday   Tuesday  1101.9458    0.0   641.4399  1562.4518   True
  Friday Wednesday   1126.185    0.0   665.6791   1586.691   True
  Monday  Saturday  4438.6223    0.0   3976.542  4900.7026   True
  Monday    Sunday  6752.8761    0.0  6290.7958  7214.9564   True
  Monday  Thursday -2059.1899    0.0 -2519.6958  -1598.684   True
  Monday   Tuesday -1488.2096    0.0 -1948.7156 -1027.7037   True
  Monday Wednesday -1463.9705    0.0 -1924.4764 -1003.4645   True
Saturday    Sunday  2314.2538    0.0  1850.6045  2777.9032   True
Saturday  Thursday -6497.8122    0.0 -6959.8925 -6035.7319   True
Saturday   Tuesday -5926.8319    0.0 -6388.9122 -5464.7516   True
Saturday Wednesday -5902.5927    0.0 -6364.6731 -5440.5124   True
  Sunday  Thursday  -8812.066    0.0 -9274.1463 -8349.9857   True
  Sunday   Tuesday -8241.0857    0.0  -8703.166 -7779.0054   True
  Sunday Wednesday -8216.8466    0.0 -8678.9269 -7754.7662   True
Thursday   Tuesday   570.9803 0.0048   110.4743  1031.4862   True
Thursday Wednesday   595.2194 0.0026   134.7135  1055.7254   True
 Tuesday Wednesday    24.2392    1.0  -436.2668   484.7451  False
-----------------------------------------------------------------
```

```
Pairs having different Sales mean are: Friday-Monday,Friday-
Saturday,Friday-Sunday,Friday-Thursday,Friday-Tuesday,Friday-Wednesday,Monday-
Saturday,Monday-Sunday,Monday-Thursday,Monday-Tuesday,Monday-Wednesday,Saturday-
Sunday,Saturday-Thursday,Saturday-Tuesday,Saturday-Wednesday,Sunday-
Thursday,Sunday-Tuesday,Sunday-Wednesday,Thursday-Tuesday,Thursday-Wednesday
Pairs having same Sales mean are: Tuesday-Wednesday
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
```

1.6 6. Data Preperation for modeling

[35]:
```
# Data Preperation for modeling
# Data for Sales forecastig model training
overall_sales = train.groupby(level=0).agg({'Sales':'sum'})
id_wise_sales = pd.crosstab(index=train.index, columns=train.Store_id, values
    =train.Sales, aggfunc='sum')
store_type_wise_sales = pd.crosstab(index=train.index, columns=train.
    Store_Type, values =train.Sales, aggfunc='sum')
location_wise_sales = pd.crosstab(index=train.index, columns=train.
    Location_Type, values =train.Sales, aggfunc='sum')
region_wise_sales = pd.crosstab(index=train.index, columns=train.Region_Code,
    values =train.Sales, aggfunc='sum')

# Data for Order forecastig model training
overall_order = train.groupby(level=0).agg({'Order':'sum'})
id_wise_order = pd.crosstab(index=train.index, columns=train.Store_id, values
    =train.Order, aggfunc='sum')
store_type_wise_order = pd.crosstab(index=train.index, columns=train.
    Store_Type, values =train.Order, aggfunc='sum')
location_wise_order = pd.crosstab(index=train.index, columns=train.
    Location_Type, values =train.Order, aggfunc='sum')
region_wise_order = pd.crosstab(index=train.index, columns=train.Region_Code,
    values =train.Order, aggfunc='sum')

# Create a Single DataFrame for Sales and Order
train_sales = pd.concat([overall_sales, id_wise_sales, store_type_wise_sales,
    location_wise_sales, region_wise_sales], axis=1)
train_order = pd.concat([overall_order, id_wise_order, store_type_wise_order,
    location_wise_order, region_wise_order], axis=1)

exog_train_holiday = train.groupby(train.index).mean('Holiday')['Holiday']
exog_test_holiday = test.groupby(test.index).mean('Holiday')['Holiday']
```

[36]:
```
train_sales.sample(5)
```

[36]:
```
                  Sales        1        2          3        4        5  \
2018-09-23  15356484.00  50787.0  45009.0   79581.00  37530.0  56091.0
2018-04-01  18166922.52  54228.3  63533.4   66489.96  49104.0  42486.0
2018-07-06  19560669.00  55365.0  75336.0  100575.00  55824.0  54261.0
2019-03-12  15125649.00  17169.0  48828.0   41235.00  40383.0  31425.0
2019-04-22  14214354.00  49641.0  37809.0   62280.00  33180.0  51990.0

                   6         7         8        9  …         S4         L1  \
2018-09-23  22146.00  33870.00  61662.00  26868.0  …  5362617.0  6643680.0
2018-04-01  51044.82  46874.04  68442.75  37654.2  …  6153951.6  7935110.1
2018-07-06  62088.00  60111.00  32301.00  60669.0  …  6868077.0  8374755.0
2019-03-12  44379.00  44469.00  30996.00  40953.0  …  4962132.0  6785559.0
2019-04-22  18630.00  29235.00  47628.00  42177.0  …  5095695.0  6166440.0

                    L2          L3        L4         L5          R1  \
2018-09-23  5646750.00  1900068.00  549048.0  616938.00  5640360.00
2018-04-01  6379018.11  2287515.24  763146.0  802133.07  6791009.28
2018-07-06  7179198.00  2455737.00  701085.0  849894.00  7496133.00
2019-03-12  5155902.00  1847976.00  635424.0  700788.00  5659689.00
2019-04-22  5201874.00  1783437.00  514428.0  548175.00  5423253.00

                   R2          R3          R4
2018-09-23  4184151.0  3560130.00  1971843.00
2018-04-01  5011791.9  4137095.07  2227026.27
2018-07-06  4778820.0  4823478.00  2462238.00
2019-03-12  4096827.0  3458814.00  1910319.00
2019-04-22  3758352.0  3272553.00  1760196.00

[5 rows x 379 columns]
```

1.7 7. Time series plots

[37]:
```
# Function to plot the data
def plot_sales(data, code):
  fig = go.Figure()
  fig.add_trace(go.Scatter(x=data.index, y=data[code], mode='lines', name=code))
  fig.add_trace(go.Bar(x=data.index, y=exog_train_holiday, name='campaign',␣
↪yaxis='y2', opacity=1))
  fig.update_layout(title=f'Timeseries for {code}', showlegend=False, title_x=0.
↪12,
      yaxis=dict(title='Sales Amount'),
      yaxis2=dict(overlaying='y', showline=False, showgrid=False,␣
↪showticklabels=False, side='right'))
  return fig
```

```
def plot_order(data, code):
  fig = go.Figure()
  fig.add_trace(go.Scatter(x=data.index, y=data[code], mode='lines', name=code))
  fig.add_trace(go.Bar(x=data.index, y=exog_train_holiday, name='campaign',␣
↪yaxis='y2', opacity=1))
  fig.update_layout(title=f'Timeseries for {code}', showlegend=False, title_x=0.
↪12,
      yaxis=dict(title='Order Volume'),
      yaxis2=dict(overlaying='y', showline=False, showgrid=False,␣
↪showticklabels=False, side='right'))
  return fig
```

[38]:
```
fig = make_subplots(rows=2, cols=1, specs=[[{"secondary_y": True}],␣
↪[{"secondary_y": True}]])
for trace in plot_order(overall_order, 'Order').data:
  secondary_y = "yaxis" in trace and trace["yaxis"] == "y2"
  fig.add_trace(trace, row=1, col=1)
for trace in plot_sales(overall_sales, 'Sales').data:
  secondary_y = "yaxis" in trace and trace["yaxis"] == "y2"
  fig.add_trace(trace, row=2, col=1)
fig.show()
```

[39]:
```
df_order = region_wise_order
df_sales = region_wise_sales
for code in df_order.columns:
  plot_order(df_order, code).show()
  plot_sales(df_sales, code).show()
```

[40]:
```
df_order = location_wise_order
df_sales = location_wise_sales
for code in df_order.columns:
  plot_order(df_order, code).show()
  plot_sales(df_sales, code).show()
```

[41]:
```
df_order = store_type_wise_order
df_sales = store_type_wise_sales
for code in df_order.columns:
  plot_order(df_order, code).show()
  plot_sales(df_sales, code).show()
```

2 B. Stationarity, decomposition, detrending, ACF, and PACF

2.1 8. Stationarity test and decomposition

Most of timeseries model (like **AR, MA, ARIMA**) works on assumption of Stationarity, which makes it easier to predict future values, estimate model parameters, and perform statistical tests.

By transforming non-stationary data into a stationary form, analysts can apply a broader range of statistical tools and achieve more reliable results.

To check stationarity of timeseries we will use Augmanted Dickey-Duller test with 5% significance level as threshold.

[42]:
```
# Import ACF/PACF plotting modules
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Import Dickey-Fuller test
from statsmodels.tsa.stattools import adfuller
```

[43]:
```
# Print Dickey-Fuller test insights
def adf_test(dataset):
  print(f'Results of Dickey-Fuller Test:')
  for column in dataset.columns:
    pvalue = adfuller(dataset[column])[1]
    if pvalue <= 0.05:
      print(f'\033[32mTimeseries for "{column}" is stationary', end='.\t')
    else:
      print(f'\033[31mTimeseries for "{column}" is not stationary', end='.')
    print(f'\tp-value is {pvalue}\033[0m')
```

[44]:
```
adf_test(train_order)
adf_test(train_sales)
```

```
Results of Dickey-Fuller Test:
Timeseries for "Order" is stationary.              p-value is

0.00019001472288566557
Timeseries for "1" is stationary.          p-value is

3.3930512375433676e-07
Timeseries for "2" is stationary.          p-value is

3.487864501563372e-05
Timeseries for "3" is stationary.          p-value is

0.0026307597340207416
Timeseries for "4" is stationary.          p-value is

2.1803807823009623e-12
Timeseries for "5" is stationary.          p-value is

0.0001419446382003186
Timeseries for "6" is stationary.          p-value is

0.00205035109867991
Timeseries for "7" is stationary.          p-value is

0.00013323300549812953
```

```
Timeseries for "8" is stationary.          p-value is
2.2959953942007774e-07
Timeseries for "9" is stationary.          p-value is
3.2347862976079523e-17
Timeseries for "10" is stationary.         p-value is
7.441225128758886e-06
Timeseries for "11" is stationary.         p-value is
3.628125417229516e-05
Timeseries for "12" is stationary.         p-value is
0.0013883312595671436
Timeseries for "13" is stationary.         p-value is
0.013189837370918795
Timeseries for "14" is stationary.         p-value is
0.00013405682296692347
Timeseries for "15" is stationary.         p-value is
5.111294828052101e-05
Timeseries for "16" is stationary.         p-value is
0.0009199590224398585
Timeseries for "17" is stationary.         p-value is
1.0711379964565977e-06
Timeseries for "18" is stationary.         p-value is
0.00012238000145299083
Timeseries for "19" is stationary.         p-value is
2.1112891772215627e-06
Timeseries for "20" is stationary.         p-value is
0.02374224801394575
Timeseries for "21" is stationary.         p-value is
0.00023624055901910664
Timeseries for "22" is stationary.         p-value is
3.8756263276873324e-05
Timeseries for "23" is stationary.         p-value is
0.0008614546829018442
Timeseries for "24" is stationary.         p-value is
0.0008344652715878641
Timeseries for "25" is stationary.         p-value is
1.1717640571394116e-24
```

```
Timeseries for "26" is stationary.        p-value is
3.8959077828157970e-07
Timeseries for "27" is stationary.        p-value is
4.21062260155192e-06
Timeseries for "28" is stationary.        p-value is
3.401045038842587e-05
Timeseries for "29" is stationary.        p-value is
8.314714296307071e-07
Timeseries for "30" is stationary.        p-value is
3.6441956726965804e-05
Timeseries for "31" is stationary.        p-value is
0.004680006836468052
Timeseries for "32" is stationary.        p-value is
7.348732211260025e-07
Timeseries for "33" is stationary.        p-value is
7.890619917291206e-13
Timeseries for "34" is stationary.        p-value is
4.512693454886711e-06
Timeseries for "35" is stationary.        p-value is
5.309425335125379e-07
Timeseries for "36" is stationary.        p-value is
2.1693219921700006e-05
Timeseries for "37" is stationary.        p-value is
2.8247390142701275e-11
Timeseries for "38" is stationary.        p-value is
0.010751046397342333
Timeseries for "39" is stationary.        p-value is
1.9876552667426326e-09
Timeseries for "40" is stationary.        p-value is
8.119240331224733e-05
Timeseries for "41" is stationary.        p-value is
1.580021909188928e-05
Timeseries for "42" is stationary.        p-value is
0.0005572886373698547
Timeseries for "43" is stationary.        p-value is
1.2638437278931261e-06
```

```
Timeseries for "44" is stationary.        p-value is
1.8030462937170134e-06
Timeseries for "45" is stationary.        p-value is
3.8124949892148184e-05
Timeseries for "46" is not stationary.    p-value is
0.05535856530253811
Timeseries for "47" is stationary.        p-value is
1.8108070324666628e-06
Timeseries for "48" is stationary.        p-value is
2.5546797099157627e-06
Timeseries for "49" is stationary.        p-value is
0.0001435214955483385
Timeseries for "50" is stationary.        p-value is
5.993430215771944e-05
Timeseries for "51" is stationary.        p-value is
9.894957738616657e-06
Timeseries for "52" is stationary.        p-value is
7.393797204959565e-05
Timeseries for "53" is stationary.        p-value is
0.0007221302732318259
Timeseries for "54" is stationary.        p-value is
8.5513366100919e-06
Timeseries for "55" is stationary.        p-value is
2.5123285495137178e-05
Timeseries for "56" is stationary.        p-value is
1.4953534519962917e-06
Timeseries for "57" is stationary.        p-value is
1.2041706531829316e-06
Timeseries for "58" is not stationary.    p-value is
0.09093201528108513
Timeseries for "59" is stationary.        p-value is
5.998154721746872e-05
Timeseries for "60" is stationary.        p-value is
1.698930370119066e-09
Timeseries for "61" is stationary.        p-value is
2.4968497740175623e-06
```

```
Timeseries for "62" is stationary.        p-value is
1.497010164466012e-07
Timeseries for "63" is stationary.        p-value is
0.0013646209488178507
Timeseries for "64" is stationary.        p-value is
6.460736779407831e-05
Timeseries for "65" is not stationary.    p-value is
0.5386965878127822
Timeseries for "66" is stationary.        p-value is
0.0002364189250287246
Timeseries for "67" is stationary.        p-value is
6.013978922957296e-06
Timeseries for "68" is stationary.        p-value is
1.6717487828797033e-13
Timeseries for "69" is stationary.        p-value is
1.3551035044719921e-05
Timeseries for "70" is stationary.        p-value is
0.007720591150134667
Timeseries for "71" is stationary.        p-value is
1.4830684877403486e-05
Timeseries for "72" is stationary.        p-value is
4.6820858273089173e-08
Timeseries for "73" is stationary.        p-value is
1.5741391779406382e-06
Timeseries for "74" is stationary.        p-value is
0.0043133262978212565
Timeseries for "75" is stationary.        p-value is
2.697111660497253e-05
Timeseries for "76" is stationary.        p-value is
2.5708528258692893e-06
Timeseries for "77" is stationary.        p-value is
3.8155980754113376e-05
Timeseries for "78" is stationary.        p-value is
0.00016147255591651057
Timeseries for "79" is stationary.        p-value is
2.605161896558134e-05
```

```
Timeseries for "80" is stationary.        p-value is
4.83481290686305e-05
Timeseries for "81" is stationary.        p-value is
3.737655970894863e-05
Timeseries for "82" is stationary.        p-value is
0.001993493594374998
Timeseries for "83" is stationary.        p-value is
7.004283797447677e-15
Timeseries for "84" is stationary.        p-value is
5.571904889755887e-05
Timeseries for "85" is stationary.        p-value is
0.0021295531650791922
Timeseries for "86" is stationary.        p-value is
4.372334149494939e-06
Timeseries for "87" is stationary.        p-value is
1.4046418027945153e-06
Timeseries for "88" is stationary.        p-value is
0.038354241685291696
Timeseries for "89" is stationary.        p-value is
0.002571840765244099
Timeseries for "90" is stationary.        p-value is
9.538544858757939e-05
Timeseries for "91" is stationary.        p-value is
0.000336206416799257
Timeseries for "92" is stationary.        p-value is
0.0017260076252782856
Timeseries for "93" is stationary.        p-value is
0.001474124682377123
Timeseries for "94" is stationary.        p-value is
4.575949601838265e-07
Timeseries for "95" is stationary.        p-value is
0.001317343984144627
Timeseries for "96" is stationary.        p-value is
0.0006942335774146915
Timeseries for "97" is stationary.        p-value is
0.00012180161288027802
```

```
Timeseries for "98" is stationary.        p-value is
8.74848672780359e-08
Timeseries for "99" is stationary.        p-value is
3.955645993497474e-06
Timeseries for "100" is stationary.               p-value is
0.0006469654542312705
Timeseries for "101" is stationary.               p-value is
2.0153101703932955e-07
Timeseries for "102" is stationary.               p-value is
9.487147439371601e-11
Timeseries for "103" is stationary.               p-value is
6.565326305306632e-07
Timeseries for "104" is stationary.               p-value is
3.6991285343036747e-07
Timeseries for "105" is stationary.               p-value is
0.011071898631391009
Timeseries for "106" is stationary.               p-value is
3.155230372582543e-05
Timeseries for "107" is stationary.               p-value is
0.00026624851390524433
Timeseries for "108" is stationary.               p-value is
1.853967025189231e-10
Timeseries for "109" is stationary.               p-value is
8.749008795477004e-05
Timeseries for "110" is stationary.               p-value is
3.149313591033713e-06
Timeseries for "111" is stationary.               p-value is
1.0091425117159596e-05
Timeseries for "112" is stationary.               p-value is
6.520659064953944e-05
Timeseries for "113" is stationary.               p-value is
1.7865455026889918e-10
Timeseries for "114" is stationary.               p-value is
8.301087012248881e-12
Timeseries for "115" is stationary.               p-value is
0.00012583256779253316
```

```
Timeseries for "116" is stationary.               p-value is
1.1834075230958227e-10
Timeseries for "117" is stationary.               p-value is
1.9718918986153662e-06
Timeseries for "118" is stationary.               p-value is
1.0933309998271666e-10
Timeseries for "119" is stationary.               p-value is
0.00014708357883481274
Timeseries for "120" is stationary.               p-value is
9.913511150912327e-06
Timeseries for "121" is stationary.               p-value is
2.723225101853458e-06
Timeseries for "122" is stationary.               p-value is
1.1752964130149005e-05
Timeseries for "123" is stationary.               p-value is
2.139101441311718e-05
Timeseries for "124" is stationary.               p-value is
7.896420187636915e-07
Timeseries for "125" is stationary.               p-value is
0.00011947851846840942
Timeseries for "126" is stationary.               p-value is
3.8620535186139247e-07
Timeseries for "127" is stationary.               p-value is
3.5670950681081125e-07
Timeseries for "128" is stationary.               p-value is
0.010633702829589706
Timeseries for "129" is stationary.               p-value is
5.203292864137039e-07
Timeseries for "130" is stationary.               p-value is
1.8908153585059535e-06
Timeseries for "131" is stationary.               p-value is
0.0007262619422735948
Timeseries for "132" is stationary.               p-value is
8.333259467777589e-07
Timeseries for "133" is stationary.               p-value is
6.789011652574819e-05
```

```
Timeseries for "134" is stationary.               p-value is
2.076297711073456e-07
Timeseries for "135" is stationary.               p-value is
3.2544088194635824e-06
Timeseries for "136" is stationary.               p-value is
0.0016451747085213385
Timeseries for "137" is stationary.               p-value is
2.0090805067845517e-06
Timeseries for "138" is stationary.               p-value is
4.270008858444917e-14
Timeseries for "139" is stationary.               p-value is
1.6747504360262133e-05
Timeseries for "140" is stationary.               p-value is
1.7306650304429188e-07
Timeseries for "141" is stationary.               p-value is
2.916748711073959e-05
Timeseries for "142" is stationary.               p-value is
3.1496676944661386e-08
Timeseries for "143" is stationary.               p-value is
2.3086234268581383e-06
Timeseries for "144" is stationary.               p-value is
1.5799794695725325e-11
Timeseries for "145" is stationary.               p-value is
4.299464534141279e-06
Timeseries for "146" is stationary.               p-value is
0.002468515341846608
Timeseries for "147" is stationary.               p-value is
0.00016865675111165732
Timeseries for "148" is stationary.               p-value is
4.5403535839328955e-05
Timeseries for "149" is stationary.               p-value is
8.295417566991276e-05
Timeseries for "150" is stationary.               p-value is
1.0767614434951178e-05
Timeseries for "151" is stationary.               p-value is
3.969608207812673e-10
```

```
Timeseries for "152" is stationary.               p-value is
5.0078497482456014e-08
Timeseries for "153" is stationary.               p-value is
8.444282283952105e-05
Timeseries for "154" is stationary.               p-value is
3.383442764215231e-07
Timeseries for "155" is stationary.               p-value is
0.00017102218547661131
Timeseries for "156" is stationary.               p-value is
5.258887943929971e-08
Timeseries for "157" is stationary.               p-value is
6.929853876090441e-08
Timeseries for "158" is stationary.               p-value is
6.928076436982305e-05
Timeseries for "159" is stationary.               p-value is
0.0035456517565745213
Timeseries for "160" is stationary.               p-value is
0.0009795640542862624
Timeseries for "161" is stationary.               p-value is
7.780640147016751e-06
Timeseries for "162" is stationary.               p-value is
1.254621575298791e-07
Timeseries for "163" is stationary.               p-value is
2.58094892281611e-12
Timeseries for "164" is stationary.               p-value is
0.0004686842928708999
Timeseries for "165" is stationary.               p-value is
0.00016959509179191538
Timeseries for "166" is stationary.               p-value is
8.785738747623313e-06
Timeseries for "167" is stationary.               p-value is
0.0002595137175460508
Timeseries for "168" is stationary.               p-value is
7.077563909724307e-12
Timeseries for "169" is stationary.               p-value is
2.7700044055212924e-06
```

```
Timeseries for "170" is stationary.               p-value is
7.933104481738624e-07
Timeseries for "171" is stationary.               p-value is
4.870367624715471e-07
Timeseries for "172" is stationary.               p-value is
5.4642657233765346e-05
Timeseries for "173" is stationary.               p-value is
1.7146183668627356e-08
Timeseries for "174" is stationary.               p-value is
1.4697060127673197e-07
Timeseries for "175" is stationary.               p-value is
0.011386226098472138
Timeseries for "176" is stationary.               p-value is
9.684310863336179e-08
Timeseries for "177" is stationary.               p-value is
4.6954083274973815e-05
Timeseries for "178" is stationary.               p-value is
8.344840987786177e-05
Timeseries for "179" is stationary.               p-value is
2.444337187103508e-06
Timeseries for "180" is stationary.               p-value is
0.024807871541731644
Timeseries for "181" is stationary.               p-value is
5.246482640609067e-10
Timeseries for "182" is stationary.               p-value is
0.008143302537358305
Timeseries for "183" is stationary.               p-value is
1.4470425609233592e-08
Timeseries for "184" is stationary.               p-value is
0.00029333044859244315
Timeseries for "185" is stationary.               p-value is
5.79664288666121e-05
Timeseries for "186" is stationary.               p-value is
1.4051496255533592e-05
Timeseries for "187" is stationary.               p-value is
4.418015767340524e-07
```

```
Timeseries for "188" is stationary.               p-value is
0.0007513225966590528
Timeseries for "189" is stationary.               p-value is
0.00025605639308622487
Timeseries for "190" is stationary.               p-value is
1.90552253482218e-11
Timeseries for "191" is stationary.               p-value is
2.9336876492075247e-06
Timeseries for "192" is stationary.               p-value is
8.765088353742073e-10
Timeseries for "193" is stationary.               p-value is
8.055484375320195e-11
Timeseries for "194" is stationary.               p-value is
8.901743696472675e-05
Timeseries for "195" is stationary.               p-value is
1.260124858225439e-26
Timeseries for "196" is stationary.               p-value is
7.910690793339057e-12
Timeseries for "197" is stationary.               p-value is
1.2289931875136164e-06
Timeseries for "198" is stationary.               p-value is
8.6523959256508e-07
Timeseries for "199" is stationary.               p-value is
2.2316505781938132e-06
Timeseries for "200" is stationary.               p-value is
2.0404072737453138e-07
Timeseries for "201" is stationary.               p-value is
0.003151423510302445
Timeseries for "202" is stationary.               p-value is
4.227563608810947e-05
Timeseries for "203" is stationary.               p-value is
4.795911712537781e-09
Timeseries for "204" is stationary.               p-value is
4.3890601708509006e-14
Timeseries for "205" is stationary.               p-value is
5.856848465711828e-05
```

```
Timeseries for "206" is not stationary.    p-value is
0.10946694123579304
Timeseries for "207" is stationary.                p-value is
2.129647218943059e-06
Timeseries for "208" is stationary.                p-value is
2.5326725612245584e-06
Timeseries for "209" is stationary.                p-value is
6.311216582776823e-05
Timeseries for "210" is stationary.                p-value is
0.0017479461434251858
Timeseries for "211" is stationary.                p-value is
0.00024956553972248934
Timeseries for "212" is stationary.                p-value is
6.3276157098852926e-15
Timeseries for "213" is stationary.                p-value is
3.135286071523618e-05
Timeseries for "214" is stationary.                p-value is
3.717417906211807e-05
Timeseries for "215" is stationary.                p-value is
1.3190527302977789e-05
Timeseries for "216" is stationary.                p-value is
0.00021826854006605113
Timeseries for "217" is stationary.                p-value is
6.022004410721664e-06
Timeseries for "218" is stationary.                p-value is
7.364911445760457e-07
Timeseries for "219" is stationary.                p-value is
0.0001888642065283942
Timeseries for "220" is stationary.                p-value is
0.0004068319867345256
Timeseries for "221" is stationary.                p-value is
0.007102411382209182
Timeseries for "222" is stationary.                p-value is
5.947031939111964e-09
Timeseries for "223" is stationary.                p-value is
2.439924239963443e-05
```

```
Timeseries for "224" is stationary.               p-value is
0.0030196886119002292
Timeseries for "225" is stationary.               p-value is
4.567402188286042e-10
Timeseries for "226" is stationary.               p-value is
6.191692893768323e-07
Timeseries for "227" is stationary.               p-value is
2.819149099769977e-06
Timeseries for "228" is stationary.               p-value is
1.443445471807939e-06
Timeseries for "229" is stationary.               p-value is
0.0002227329582254047
Timeseries for "230" is stationary.               p-value is
5.9341160057158775e-06
Timeseries for "231" is stationary.               p-value is
1.9731989584590604e-06
Timeseries for "232" is stationary.               p-value is
4.862555599428091e-07
Timeseries for "233" is stationary.               p-value is
0.011116867819018121
Timeseries for "234" is stationary.               p-value is
1.2037130383192282e-05
Timeseries for "235" is stationary.               p-value is
0.000417775521394403
Timeseries for "236" is stationary.               p-value is
1.268816540835271e-06
Timeseries for "237" is stationary.               p-value is
3.3672373338658524e-10
Timeseries for "238" is stationary.               p-value is
5.658643960415718e-08
Timeseries for "239" is stationary.               p-value is
0.00030691128484766124
Timeseries for "240" is stationary.               p-value is
0.0025517073586012588
Timeseries for "241" is stationary.               p-value is
4.3461070113121813e-05
```

```
Timeseries for "242" is stationary.                p-value is
4.280415364093271e-06
Timeseries for "243" is stationary.                p-value is
0.00082853608587924
Timeseries for "244" is stationary.                p-value is
8.59350255222756e-06
Timeseries for "245" is stationary.                p-value is
4.730780677154208e-06
Timeseries for "246" is stationary.                p-value is
1.2161075846856997e-06
Timeseries for "247" is stationary.                p-value is
0.000131212453389353
Timeseries for "248" is stationary.                p-value is
1.7918944997759566e-07
Timeseries for "249" is stationary.                p-value is
0.00036607129773350347
Timeseries for "250" is stationary.                p-value is
1.681901006718479e-05
Timeseries for "251" is stationary.                p-value is
3.16610699463011e-07
Timeseries for "252" is stationary.                p-value is
3.9917513298946077e-05
Timeseries for "253" is not stationary.    p-value is
0.06421249505051625
Timeseries for "254" is stationary.                p-value is
1.0883878110193443e-07
Timeseries for "255" is stationary.                p-value is
1.2025255429467576e-10
Timeseries for "256" is stationary.                p-value is
1.47893177255516e-05
Timeseries for "257" is stationary.                p-value is
1.4119363821673375e-12
Timeseries for "258" is stationary.                p-value is
4.992812079165035e-12
Timeseries for "259" is stationary.                p-value is
2.7924333928363728e-06
```

```
Timeseries for "260" is stationary.                p-value is
4.262313061592368e-05
Timeseries for "261" is stationary.                p-value is
0.0006518111638001428
Timeseries for "262" is stationary.                p-value is
2.1527522559680643e-06
Timeseries for "263" is stationary.                p-value is
7.46919653800032e-08
Timeseries for "264" is stationary.                p-value is
1.961118300271593e-06
Timeseries for "265" is not stationary.    p-value is
0.1747960816443197
Timeseries for "266" is not stationary.    p-value is
0.058133744122539535
Timeseries for "267" is not stationary.    p-value is
0.22647562713131197
Timeseries for "268" is stationary.                p-value is
0.00020972166741867673
Timeseries for "269" is stationary.                p-value is
0.018873041818966534
Timeseries for "270" is stationary.                p-value is
0.000327202401245754
Timeseries for "271" is stationary.                p-value is
0.00015498715561106764
Timeseries for "272" is stationary.                p-value is
1.6026841135698594e-05
Timeseries for "273" is stationary.                p-value is
1.777096064991541e-05
Timeseries for "274" is stationary.                p-value is
4.891416538817896e-06
Timeseries for "275" is stationary.                p-value is
3.6676556111398138e-28
Timeseries for "276" is stationary.                p-value is
2.4324285363767784e-06
Timeseries for "277" is stationary.                p-value is
0.00027601745945045623
```

```
Timeseries for "278" is stationary.               p-value is
0.0003373884023617O5
Timeseries for "279" is stationary.               p-value is
3.904208654283419e-05
Timeseries for "280" is stationary.               p-value is
0.00013857330387773327
Timeseries for "281" is stationary.               p-value is
7.932930575163268e-10
Timeseries for "282" is stationary.               p-value is
1.047579127750594e-05
Timeseries for "283" is stationary.               p-value is
0.0005137027537466457
Timeseries for "284" is stationary.               p-value is
0.0008325047934977306
Timeseries for "285" is stationary.               p-value is
8.313392586855961e-05
Timeseries for "286" is stationary.               p-value is
4.556157778610793e-05
Timeseries for "287" is stationary.               p-value is
7.263260514975754e-07
Timeseries for "288" is stationary.               p-value is
0.00011650204650465217
Timeseries for "289" is stationary.               p-value is
1.4920622861657612e-05
Timeseries for "290" is stationary.               p-value is
5.074459555452562e-05
Timeseries for "291" is stationary.               p-value is
0.005381741594836949
Timeseries for "292" is stationary.               p-value is
6.874333642198158e-05
Timeseries for "293" is stationary.               p-value is
0.00011136058526701835
Timeseries for "294" is stationary.               p-value is
0.0002833971345018345
Timeseries for "295" is stationary.               p-value is
1.1453974485699648e-06
```

```
Timeseries for "296" is stationary.               p-value is
0.028491948965722713
Timeseries for "297" is stationary.               p-value is
4.2705561064375764e-05
Timeseries for "298" is stationary.               p-value is
9.480148443769051e-08
Timeseries for "299" is stationary.               p-value is
1.0146305387454682e-07
Timeseries for "300" is not stationary.    p-value is
0.18630095937928448
Timeseries for "301" is stationary.               p-value is
3.2775535933734434e-07
Timeseries for "302" is stationary.               p-value is
5.261582469974119e-06
Timeseries for "303" is stationary.               p-value is
0.005420856557244605
Timeseries for "304" is stationary.               p-value is
0.006520182864179445
Timeseries for "305" is stationary.               p-value is
8.500317481400646e-06
Timeseries for "306" is stationary.               p-value is
5.760695095601388e-14
Timeseries for "307" is stationary.               p-value is
3.6096933501381394e-05
Timeseries for "308" is stationary.               p-value is
4.449914013860301e-05
Timeseries for "309" is stationary.               p-value is
0.0004323393777942778
Timeseries for "310" is stationary.               p-value is
0.004787697776453726
Timeseries for "311" is stationary.               p-value is
0.0002560641465854551
Timeseries for "312" is stationary.               p-value is
0.0005534276875624203
Timeseries for "313" is stationary.               p-value is
1.8485663934821887e-06
```

```
Timeseries for "314" is stationary.               p-value is
0.0003996736239977625
Timeseries for "315" is stationary.               p-value is
0.049945737852137115
Timeseries for "316" is stationary.               p-value is
5.580870911314861e-10
Timeseries for "317" is stationary.               p-value is
1.9995511777153038e-24
Timeseries for "318" is stationary.               p-value is
4.330003152273656e-05
Timeseries for "319" is stationary.               p-value is
0.00026576526709938883
Timeseries for "320" is stationary.               p-value is
0.0005206065180896851
Timeseries for "321" is stationary.               p-value is
0.0018378495059394606
Timeseries for "322" is stationary.               p-value is
0.0027337126834068334
Timeseries for "323" is stationary.               p-value is
0.00025392437959206166
Timeseries for "324" is stationary.               p-value is
0.000347430239172436
Timeseries for "325" is stationary.               p-value is
0.017294976083193546
Timeseries for "326" is stationary.               p-value is
7.976834333173641e-06
Timeseries for "327" is stationary.               p-value is
0.0018461029029841654
Timeseries for "328" is stationary.               p-value is
0.0004832099037848997
Timeseries for "329" is stationary.               p-value is
3.960684881695652e-05
Timeseries for "330" is stationary.               p-value is
0.0033753156780492117
Timeseries for "331" is stationary.               p-value is
3.992483857195217e-07
```

```
Timeseries for "332" is stationary.              p-value is
1.304637100924014e-06
Timeseries for "333" is stationary.              p-value is
4.323510712103889e-07
Timeseries for "334" is stationary.              p-value is
1.062355460743779e-05
Timeseries for "335" is stationary.              p-value is
0.009073321731381\83
Timeseries for "336" is stationary.              p-value is
0.00011682094299079566
Timeseries for "337" is stationary.              p-value is
2.5499577221748048e-05
Timeseries for "338" is stationary.              p-value is
1.0902150461238297e-05
Timeseries for "339" is stationary.              p-value is
0.011717640533577376
Timeseries for "340" is stationary.              p-value is
5.511231332364279e-06
Timeseries for "341" is stationary.              p-value is
7.241594054128154e-06
Timeseries for "342" is stationary.              p-value is
6.888647997770799e-07
Timeseries for "343" is stationary.              p-value is
8.800446125133676e-05
Timeseries for "344" is stationary.              p-value is
3.073854857697815e-08
Timeseries for "345" is stationary.              p-value is
1.5548309912031497e-25
Timeseries for "346" is stationary.              p-value is
1.9026313860071634e-05
Timeseries for "347" is stationary.              p-value is
1.8563510927320425e-06
Timeseries for "348" is stationary.              p-value is
4.410469136597884e-05
Timeseries for "349" is stationary.              p-value is
3.3725610944696004e-06
```

```
Timeseries for "350" is stationary.               p-value is
0.00048950590573241
Timeseries for "351" is stationary.               p-value is
1.3554110180357414e-05
Timeseries for "352" is stationary.               p-value is
0.03502712286435989
Timeseries for "353" is stationary.               p-value is
4.438801287468647e-06
Timeseries for "354" is stationary.               p-value is
8.730174857855914e-08
Timeseries for "355" is stationary.               p-value is
2.321036507600439e-05
Timeseries for "356" is stationary.               p-value is
0.0003218863348237397
Timeseries for "357" is stationary.               p-value is
0.010030744366767177
Timeseries for "358" is stationary.               p-value is
2.8263736825744944e-06
Timeseries for "359" is stationary.               p-value is
0.0008662720360079804
Timeseries for "360" is stationary.               p-value is
8.98195210246438e-08
Timeseries for "361" is stationary.               p-value is
8.809429002921898e-06
Timeseries for "362" is stationary.               p-value is
0.000663340062058788
Timeseries for "363" is stationary.               p-value is
0.0010172862630813118
Timeseries for "364" is stationary.               p-value is
2.4175579825030357e-06
Timeseries for "365" is stationary.               p-value is
0.0021034988056674894
Timeseries for "S1" is stationary.        p-value is
7.793448064208875e-05
Timeseries for "S2" is stationary.        p-value is
3.19465531365585e-05
```

```
Timeseries for "S3" is stationary.          p-value is
0.0004455172797190973
Timeseries for "S4" is stationary.          p-value is
0.0005447205594067498
Timeseries for "L1" is stationary.          p-value is
0.00014331514294486087
Timeseries for "L2" is stationary.          p-value is
0.00044985543030531615
Timeseries for "L3" is stationary.          p-value is
6.657730739788191e-05
Timeseries for "L4" is stationary.          p-value is
5.520188415032601e-06
Timeseries for "L5" is stationary.          p-value is
2.446372236792949e-05
Timeseries for "R1" is stationary.          p-value is
0.0004224908491593281
Timeseries for "R2" is stationary.          p-value is
4.5432256312175485e-05
Timeseries for "R3" is stationary.          p-value is
0.00017724010576170132
Timeseries for "R4" is stationary.          p-value is
0.00017453874458951745
Results of Dickey-Fuller Test:
Timeseries for "Sales" is stationary.              p-value is
0.007386718711362291
Timeseries for "1" is stationary.          p-value is
0.0010072301346594694
Timeseries for "2" is stationary.          p-value is
0.00031561292823509697
Timeseries for "3" is stationary.          p-value is
0.008991511921083089
Timeseries for "4" is stationary.          p-value is
0.0002737671033380586
Timeseries for "5" is stationary.          p-value is
0.005051306139498418
Timeseries for "6" is stationary.          p-value is
0.013888862311180786
```

```
Timeseries for "7" is stationary.          p-value is
0.008717157545251193
Timeseries for "8" is stationary.          p-value is
0.00013217108310095338
Timeseries for "9" is stationary.          p-value is
2.9868967309125026e-21
Timeseries for "10" is stationary.         p-value is
9.122877817511299e-05
Timeseries for "11" is stationary.         p-value is
0.00023967290176901514
Timeseries for "12" is stationary.         p-value is
0.009512150715144845
Timeseries for "13" is stationary.         p-value is
0.022684796879694213
Timeseries for "14" is stationary.         p-value is
0.0001438788771060245
Timeseries for "15" is stationary.         p-value is
0.0005642453559567487
Timeseries for "16" is stationary.         p-value is
0.0008708998374719632
Timeseries for "17" is stationary.         p-value is
0.00015253072153556218
Timeseries for "18" is stationary.         p-value is
0.002310062738161719
Timeseries for "19" is stationary.         p-value is
0.0002075112312322143
Timeseries for "20" is stationary.         p-value is
0.013183196680306977
Timeseries for "21" is stationary.         p-value is
0.0013863982414371134
Timeseries for "22" is stationary.         p-value is
0.0006521497642114568
Timeseries for "23" is stationary.         p-value is
0.005768868464092041
Timeseries for "24" is stationary.         p-value is
0.016916048119491083
```

```
Timeseries for "25" is stationary.        p-value is
3.558101260958532e-24
Timeseries for "26" is stationary.        p-value is
0.00017876734181824014
Timeseries for "27" is stationary.        p-value is
0.0003157634407573397
Timeseries for "28" is stationary.        p-value is
0.0010849474336122329
Timeseries for "29" is stationary.        p-value is
2.2234749436377456e-05
Timeseries for "30" is stationary.        p-value is
0.0005491655112836941
Timeseries for "31" is stationary.        p-value is
0.0074353945297359406
Timeseries for "32" is stationary.        p-value is
0.0023147416981024642
Timeseries for "33" is stationary.        p-value is
0.0002011310512751143
Timeseries for "34" is stationary.        p-value is
7.249322540028292e-05
Timeseries for "35" is stationary.        p-value is
0.0003736584947946565
Timeseries for "36" is stationary.        p-value is
0.0003777959298406816
Timeseries for "37" is stationary.        p-value is
1.393012778080807e-11
Timeseries for "38" is stationary.        p-value is
0.022930015602661512
Timeseries for "39" is stationary.        p-value is
9.569520115462759e-07
Timeseries for "40" is stationary.        p-value is
0.001904343486754834
Timeseries for "41" is stationary.        p-value is
0.0001738481437417984
Timeseries for "42" is stationary.        p-value is
0.007219768463646842
```

```
Timeseries for "43" is stationary.        p-value is
2.7440674341048e-10
Timeseries for "44" is stationary.        p-value is
2.8660386173796303e-05
Timeseries for "45" is stationary.        p-value is
0.0015535020882591192
Timeseries for "46" is stationary.        p-value is
0.029745293195268505
Timeseries for "47" is stationary.        p-value is
0.0005264761742407543
Timeseries for "48" is stationary.        p-value is
0.0009533733239574719
Timeseries for "49" is stationary.        p-value is
0.00018768351929052625
Timeseries for "50" is stationary.        p-value is
0.001821013315630373
Timeseries for "51" is stationary.        p-value is
9.709833091280135e-06
Timeseries for "52" is stationary.        p-value is
0.00010669142753664644
Timeseries for "53" is stationary.        p-value is
0.016944351315085673
Timeseries for "54" is stationary.        p-value is
0.00039906920191827295
Timeseries for "55" is stationary.        p-value is
0.0004199617709412446
Timeseries for "56" is stationary.        p-value is
0.0006155764926102539
Timeseries for "57" is stationary.        p-value is
0.0031306500293198517
Timeseries for "58" is not stationary.    p-value is
0.05599412739299651
Timeseries for "59" is stationary.        p-value is
0.0005049044323103629
Timeseries for "60" is stationary.        p-value is
5.958124430320324e-06
```

```
Timeseries for "61" is stationary.         p-value is
3.718347132257486e-05
Timeseries for "62" is stationary.         p-value is
0.0006939322440744843
Timeseries for "63" is stationary.         p-value is
0.005152732990119278
Timeseries for "64" is stationary.         p-value is
0.0041408112601858576
Timeseries for "65" is not stationary.     p-value is
0.18483372921324598
Timeseries for "66" is stationary.         p-value is
0.0110642570537213
Timeseries for "67" is stationary.         p-value is
0.0001858253429276891
Timeseries for "68" is stationary.         p-value is
0.00021692105928102737
Timeseries for "69" is stationary.         p-value is
0.0002476726160297314
Timeseries for "70" is stationary.         p-value is
0.01610110687786486
Timeseries for "71" is stationary.         p-value is
0.0011138623056447929
Timeseries for "72" is stationary.         p-value is
0.0002259581081959068
Timeseries for "73" is stationary.         p-value is
0.00010750310980277186
Timeseries for "74" is stationary.         p-value is
0.028771409803396188
Timeseries for "75" is stationary.         p-value is
0.0008917292983730971
Timeseries for "76" is stationary.         p-value is
5.2358450002276005e-05
Timeseries for "77" is stationary.         p-value is
2.9809608530756844e-05
Timeseries for "78" is stationary.         p-value is
0.0009624697563829815
```

```
Timeseries for "79" is stationary.         p-value is
0.00016455991284978206
Timeseries for "80" is stationary.         p-value is
8.778855813241938e-05
Timeseries for "81" is stationary.         p-value is
0.00028144423489681505
Timeseries for "82" is stationary.         p-value is
0.005831163927923531
Timeseries for "83" is stationary.         p-value is
5.130846582256076e-05
Timeseries for "84" is stationary.         p-value is
0.0010651714308486657
Timeseries for "85" is stationary.         p-value is
2.363522318342324e-20
Timeseries for "86" is stationary.         p-value is
2.213457504886807e-05
Timeseries for "87" is stationary.         p-value is
5.4862893774895915e-05
Timeseries for "88" is stationary.         p-value is
0.04143993154659144
Timeseries for "89" is stationary.         p-value is
0.015928260812085347
Timeseries for "90" is stationary.         p-value is
0.00023417991573657947
Timeseries for "91" is stationary.         p-value is
0.001730456899608241
Timeseries for "92" is stationary.         p-value is
0.003953046849771409
Timeseries for "93" is stationary.         p-value is
3.525918966730799e-09
Timeseries for "94" is stationary.         p-value is
0.002427024960420313
Timeseries for "95" is stationary.         p-value is
2.376101389913979e-05
Timeseries for "96" is stationary.         p-value is
0.003137381413218535
```

```
Timeseries for "97" is stationary.        p-value is
0.00107602356931371
Timeseries for "98" is stationary.        p-value is
0.0004866827000706573
Timeseries for "99" is stationary.        p-value is
1.082660925284785e-05
Timeseries for "100" is stationary.                p-value is
0.001790434955505629
Timeseries for "101" is stationary.                p-value is
0.00012110899978042064
Timeseries for "102" is stationary.                p-value is
0.0010391184787552944
Timeseries for "103" is stationary.                p-value is
2.0553471181387548e-05
Timeseries for "104" is stationary.                p-value is
2.9894270303828405e-05
Timeseries for "105" is stationary.                p-value is
0.033011775202366186
Timeseries for "106" is stationary.                p-value is
0.0006272473174448875
Timeseries for "107" is stationary.                p-value is
0.007526035876384321
Timeseries for "108" is stationary.                p-value is
4.568705663622652e-10
Timeseries for "109" is stationary.                p-value is
0.0009411371765543806
Timeseries for "110" is stationary.                p-value is
6.564392374203068e-05
Timeseries for "111" is stationary.                p-value is
0.0001936772455416642
Timeseries for "112" is stationary.                p-value is
0.00042618085530415407
Timeseries for "113" is stationary.                p-value is
0.00040523942375613343
Timeseries for "114" is stationary.                p-value is
0.0018411896874280392
```

```
Timeseries for "115" is stationary.               p-value is
0.00025459123912125473
Timeseries for "116" is stationary.               p-value is
6.718188975618646e-09
Timeseries for "117" is stationary.               p-value is
8.361888003537213e-05
Timeseries for "118" is stationary.               p-value is
3.940266672700867e-05
Timeseries for "119" is stationary.               p-value is
0.0016617027675708542
Timeseries for "120" is stationary.               p-value is
0.00048028169836101647
Timeseries for "121" is stationary.               p-value is
3.854991998881068e-05
Timeseries for "122" is stationary.               p-value is
0.0004979907744716396
Timeseries for "123" is stationary.               p-value is
0.0014677193942636392
Timeseries for "124" is stationary.               p-value is
5.194863134877565e-05
Timeseries for "125" is stationary.               p-value is
0.0040296816181112916
Timeseries for "126" is stationary.               p-value is
3.3447916782392196e-05
Timeseries for "127" is stationary.               p-value is
0.0008367289602497126
Timeseries for "128" is stationary.               p-value is
0.020410203453832387
Timeseries for "129" is stationary.               p-value is
2.0264426392745064e-05
Timeseries for "130" is stationary.               p-value is
0.0018858058564864026
Timeseries for "131" is stationary.               p-value is
0.0046206644375212675
Timeseries for "132" is stationary.               p-value is
0.0003563715598358019
```

```
Timeseries for "133" is stationary.               p-value is
0.005435325827521715
Timeseries for "134" is stationary.               p-value is
5.342157715564607e-06
Timeseries for "135" is stationary.               p-value is
0.00014829951342225866
Timeseries for "136" is stationary.               p-value is
0.004012635832407312
Timeseries for "137" is stationary.               p-value is
5.242918624304251e-05
Timeseries for "138" is stationary.               p-value is
1.6197594623572468e-05
Timeseries for "139" is stationary.               p-value is
0.00024690079368640434
Timeseries for "140" is stationary.               p-value is
6.0552756376033996e-05
Timeseries for "141" is stationary.               p-value is
1.3940381999497495e-05
Timeseries for "142" is stationary.               p-value is
3.1771690956939645e-05
Timeseries for "143" is stationary.               p-value is
0.0013764220384307369
Timeseries for "144" is stationary.               p-value is
0.0011179628381382566
Timeseries for "145" is stationary.               p-value is
6.816240842177759e-05
Timeseries for "146" is stationary.               p-value is
0.0024038829584976484
Timeseries for "147" is stationary.               p-value is
0.0004342759107233961
Timeseries for "148" is stationary.               p-value is
0.0003721526304682833
Timeseries for "149" is stationary.               p-value is
0.00033353783438898934
Timeseries for "150" is stationary.               p-value is
0.001320918604322212
```

```
Timeseries for "151" is stationary.               p-value is
1.3941164437570472e-23
Timeseries for "152" is stationary.               p-value is
1.3245681031290174e-05
Timeseries for "153" is stationary.               p-value is
0.00045288873118110175
Timeseries for "154" is stationary.               p-value is
2.971020683912829e-05
Timeseries for "155" is stationary.               p-value is
0.0006735608962806927
Timeseries for "156" is stationary.               p-value is
6.066785295908553e-06
Timeseries for "157" is stationary.               p-value is
0.00285220447419756
Timeseries for "158" is stationary.               p-value is
2.474079904845501e-06
Timeseries for "159" is stationary.               p-value is
0.0023826624953484387
Timeseries for "160" is stationary.               p-value is
0.0010601568081331778
Timeseries for "161" is stationary.               p-value is
0.0009006657928950012
Timeseries for "162" is stationary.               p-value is
0.00010029404247567932
Timeseries for "163" is stationary.               p-value is
0.00014355492991076837
Timeseries for "164" is stationary.               p-value is
0.001044396002093798
Timeseries for "165" is stationary.               p-value is
0.001030960509203344
Timeseries for "166" is stationary.               p-value is
3.2511195999710264e-05
Timeseries for "167" is stationary.               p-value is
0.0008504456432767575
Timeseries for "168" is stationary.               p-value is
1.7328311146079027e-11
```

```
Timeseries for "169" is stationary.            p-value is
2.767696645833784e-05
Timeseries for "170" is stationary.            p-value is
2.52699783654674e-05
Timeseries for "171" is stationary.            p-value is
1.7115680935026434e-05
Timeseries for "172" is stationary.            p-value is
0.002854645341767548
Timeseries for "173" is stationary.            p-value is
0.00037183650634175
Timeseries for "174" is stationary.            p-value is
0.0002977941654444389
Timeseries for "175" is stationary.            p-value is
0.02952885424160673
Timeseries for "176" is stationary.            p-value is
1.8652537044153936e-05
Timeseries for "177" is stationary.            p-value is
1.547311263471623e-05
Timeseries for "178" is stationary.            p-value is
2.3314004656250497e-05
Timeseries for "179" is stationary.            p-value is
5.050030649808855e-05
Timeseries for "180" is not stationary.    p-value is
0.0704381670914891
Timeseries for "181" is stationary.            p-value is
4.945394454609212e-07
Timeseries for "182" is stationary.            p-value is
0.003012509403693294
Timeseries for "183" is stationary.            p-value is
2.3972410694476644e-06
Timeseries for "184" is stationary.            p-value is
0.0017036132869262238
Timeseries for "185" is stationary.            p-value is
0.003141264862993032
Timeseries for "186" is stationary.            p-value is
0.0008272973123651762
```

```
Timeseries for "187" is stationary.               p-value is
4.708091770182434e-05
Timeseries for "188" is stationary.               p-value is
0.0006248862296784562
Timeseries for "189" is stationary.               p-value is
0.0004549796057056625
Timeseries for "190" is stationary.               p-value is
5.330849934641119e-10
Timeseries for "191" is stationary.               p-value is
0.004964974511928851
Timeseries for "192" is stationary.               p-value is
9.930872498345495e-05
Timeseries for "193" is stationary.               p-value is
0.0012670189977772334
Timeseries for "194" is stationary.               p-value is
0.00024337166072774857
Timeseries for "195" is stationary.               p-value is
1.7231655740755665e-23
Timeseries for "196" is stationary.               p-value is
0.002368185711247773
Timeseries for "197" is stationary.               p-value is
0.00017823194694497425
Timeseries for "198" is stationary.               p-value is
0.00010982930721878995
Timeseries for "199" is stationary.               p-value is
0.00023593185533639865
Timeseries for "200" is stationary.               p-value is
1.0567546809902824e-05
Timeseries for "201" is stationary.               p-value is
0.006998287872963017
Timeseries for "202" is stationary.               p-value is
0.0005080831505495583
Timeseries for "203" is stationary.               p-value is
6.398004977933699e-05
Timeseries for "204" is stationary.               p-value is
9.457535216944182e-14
```

```
Timeseries for "205" is stationary.               p-value is
0.0002576764452714263
Timeseries for "206" is not stationary.    p-value is
0.11490127651432774
Timeseries for "207" is stationary.               p-value is
0.0001155650001373095
Timeseries for "208" is stationary.               p-value is
2.1900987700222402e-05
Timeseries for "209" is stationary.               p-value is
0.00027185106615009943
Timeseries for "210" is stationary.               p-value is
0.006298983612997602
Timeseries for "211" is stationary.               p-value is
0.0030994427336336968
Timeseries for "212" is stationary.               p-value is
1.8384467255317023e-08
Timeseries for "213" is stationary.               p-value is
0.008280798571854726
Timeseries for "214" is stationary.               p-value is
0.0004324746095029078
Timeseries for "215" is stationary.               p-value is
3.011969529994633e-05
Timeseries for "216" is stationary.               p-value is
0.00034023366659356166
Timeseries for "217" is stationary.               p-value is
1.0426360674689158e-13
Timeseries for "218" is stationary.               p-value is
0.0002038366445523663
Timeseries for "219" is stationary.               p-value is
0.0014283352492701015
Timeseries for "220" is stationary.               p-value is
0.0004235424962425248
Timeseries for "221" is stationary.               p-value is
0.012636122090964205
Timeseries for "222" is stationary.               p-value is
2.7145152599663654e-05
```

```
Timeseries for "223" is stationary.               p-value is
0.00027199637367506587
Timeseries for "224" is stationary.               p-value is
0.002366804395631395
Timeseries for "225" is stationary.               p-value is
9.301170925790265e-06
Timeseries for "226" is stationary.               p-value is
3.774615392240189e-05
Timeseries for "227" is stationary.               p-value is
0.0003031420987914876
Timeseries for "228" is stationary.               p-value is
1.0055917243420243e-05
Timeseries for "229" is stationary.               p-value is
0.001197972329319524
Timeseries for "230" is stationary.               p-value is
0.0010340078587901392
Timeseries for "231" is stationary.               p-value is
1.7913763569179506e-05
Timeseries for "232" is stationary.               p-value is
6.014281109071207e-06
Timeseries for "233" is stationary.               p-value is
0.010232386336641242
Timeseries for "234" is stationary.               p-value is
3.519016261255216e-05
Timeseries for "235" is stationary.               p-value is
0.011869727260044585
Timeseries for "236" is stationary.               p-value is
6.156473227097317e-05
Timeseries for "237" is stationary.               p-value is
0.00036215914799563465
Timeseries for "238" is stationary.               p-value is
0.00010138689453172003
Timeseries for "239" is stationary.               p-value is
0.004431032640689593
Timeseries for "240" is stationary.               p-value is
0.01071496275530863
```

```
Timeseries for "241" is stationary.                p-value is
0.00036218095880287697
Timeseries for "242" is stationary.                p-value is
1.1734247111458108e-05
Timeseries for "243" is stationary.                p-value is
0.00027508538682600276
Timeseries for "244" is stationary.                p-value is
0.0004002338454727453
Timeseries for "245" is stationary.                p-value is
9.233905576736461e-05
Timeseries for "246" is stationary.                p-value is
0.0002848468706414233
Timeseries for "247" is stationary.                p-value is
0.0005428265295455665
Timeseries for "248" is stationary.                p-value is
6.21331919358916e-05
Timeseries for "249" is stationary.                p-value is
0.0031294865525661013
Timeseries for "250" is stationary.                p-value is
0.00011577524298764726
Timeseries for "251" is stationary.                p-value is
6.502472970219795e-06
Timeseries for "252" is stationary.                p-value is
0.00297014133205754
Timeseries for "253" is not stationary.    p-value is
0.06608166489018202
Timeseries for "254" is stationary.                p-value is
2.1890747666875984e-05
Timeseries for "255" is stationary.                p-value is
2.939232253840054e-10
Timeseries for "256" is stationary.                p-value is
0.00017599749674439765
Timeseries for "257" is stationary.                p-value is
1.6436838320565628e-12
Timeseries for "258" is stationary.                p-value is
0.0009504319588403772
```

```
Timeseries for "259" is stationary.               p-value is
0.0016216262832032798
Timeseries for "260" is stationary.               p-value is
0.0011722661783458478
Timeseries for "261" is stationary.               p-value is
0.0005235278577295648
Timeseries for "262" is stationary.               p-value is
1.0318144047496912e-05
Timeseries for "263" is stationary.               p-value is
4.400555566004078e-05
Timeseries for "264" is stationary.               p-value is
3.108065694738563e-05
Timeseries for "265" is not stationary.    p-value is
0.08932256866593069
Timeseries for "266" is stationary.               p-value is
0.006060811129853634
Timeseries for "267" is not stationary.    p-value is
0.21304952243086084
Timeseries for "268" is stationary.               p-value is
0.0003218253261987438
Timeseries for "269" is stationary.               p-value is
0.00896151731908972
Timeseries for "270" is stationary.               p-value is
0.0068005208411403536
Timeseries for "271" is stationary.               p-value is
0.0006382871948541686
Timeseries for "272" is stationary.               p-value is
2.9372764717430668e-05
Timeseries for "273" is stationary.               p-value is
0.0004842498602359248
Timeseries for "274" is stationary.               p-value is
2.5258729215995463e-05
Timeseries for "275" is stationary.               p-value is
5.9645714026137255e-25
Timeseries for "276" is stationary.               p-value is
6.5745443216935096e-06
```

```
Timeseries for "277" is stationary.               p-value is
0.00206096963439632
Timeseries for "278" is stationary.               p-value is
0.0013725775393979439
Timeseries for "279" is stationary.               p-value is
0.0019025742504396872
Timeseries for "280" is stationary.               p-value is
0.000842249239911077
Timeseries for "281" is stationary.               p-value is
7.884684312223885e-05
Timeseries for "282" is stationary.               p-value is
0.0003244246637642427
Timeseries for "283" is stationary.               p-value is
0.0010257519898195966
Timeseries for "284" is stationary.               p-value is
0.006683033168869181
Timeseries for "285" is stationary.               p-value is
0.00035279633702756287
Timeseries for "286" is stationary.               p-value is
0.00011239460818445546
Timeseries for "287" is stationary.               p-value is
2.6020055037444134e-05
Timeseries for "288" is stationary.               p-value is
0.00010793626885476344
Timeseries for "289" is stationary.               p-value is
0.000494607978341371
Timeseries for "290" is stationary.               p-value is
0.002578818632838419
Timeseries for "291" is not stationary.    p-value is
0.05660839298732183
Timeseries for "292" is stationary.               p-value is
0.00138631804295186
Timeseries for "293" is stationary.               p-value is
9.227170846012794e-05
Timeseries for "294" is stationary.               p-value is
0.0019322849111849656
```

```
Timeseries for "295" is stationary.               p-value is
0.0003483226840811138
Timeseries for "296" is not stationary.    p-value is
0.08522175424013323
Timeseries for "297" is stationary.               p-value is
0.0009513523148761597
Timeseries for "298" is stationary.               p-value is
2.3154900772002134e-07
Timeseries for "299" is stationary.               p-value is
8.198964138962852e-05
Timeseries for "300" is stationary.               p-value is
0.02960701081753662
Timeseries for "301" is stationary.               p-value is
0.0008024291068340855
Timeseries for "302" is stationary.               p-value is
8.288561721439884e-05
Timeseries for "303" is stationary.               p-value is
0.007511206148330425
Timeseries for "304" is stationary.               p-value is
0.016506561303227273
Timeseries for "305" is stationary.               p-value is
0.0004695374483105809
Timeseries for "306" is stationary.               p-value is
9.10652662005277e-14
Timeseries for "307" is stationary.               p-value is
0.0003212892838848065
Timeseries for "308" is stationary.               p-value is
0.0003516318904385247
Timeseries for "309" is stationary.               p-value is
8.144569755515096e-26
Timeseries for "310" is stationary.               p-value is
0.023236368768485848
Timeseries for "311" is stationary.               p-value is
0.001849704106143876
Timeseries for "312" is stationary.               p-value is
0.0031234616557759996
```

```
Timeseries for "313" is stationary.               p-value is
6.769873735967843e-05
Timeseries for "314" is stationary.               p-value is
0.00242119827219045
Timeseries for "315" is not stationary.    p-value is
0.07098414793954455
Timeseries for "316" is stationary.               p-value is
0.00032883048073315533
Timeseries for "317" is stationary.               p-value is
0.008526634021654383
Timeseries for "318" is stationary.               p-value is
0.004451253636279219
Timeseries for "319" is stationary.               p-value is
0.0019990276271246873
Timeseries for "320" is stationary.               p-value is
0.002321073375819277
Timeseries for "321" is stationary.               p-value is
0.0043801553799993845
Timeseries for "322" is stationary.               p-value is
0.0029190152014397946
Timeseries for "323" is stationary.               p-value is
0.0055571501156199046
Timeseries for "324" is stationary.               p-value is
0.00407112225375758
Timeseries for "325" is not stationary.    p-value is
0.05424739627652914
Timeseries for "326" is stationary.               p-value is
0.0001318294963165366
Timeseries for "327" is stationary.               p-value is
0.0040422211288417345
Timeseries for "328" is stationary.               p-value is
0.0005944383550889401
Timeseries for "329" is stationary.               p-value is
0.00027879216831228304
Timeseries for "330" is stationary.               p-value is
0.000998860758398374
```

```
Timeseries for "331" is stationary.                p-value is
1.3350525607786505e-05
Timeseries for "332" is stationary.                p-value is
5.378603187944803e-05
Timeseries for "333" is stationary.                p-value is
0.0006380804496381146
Timeseries for "334" is stationary.                p-value is
2.896472891471825e-05
Timeseries for "335" is stationary.                p-value is
0.017967412288814648
Timeseries for "336" is stationary.                p-value is
0.0006222975405897576
Timeseries for "337" is stationary.                p-value is
0.0006886776027446046
Timeseries for "338" is stationary.                p-value is
0.0003975014114795488
Timeseries for "339" is stationary.                p-value is
0.007551842387699304
Timeseries for "340" is stationary.                p-value is
5.8278185854456127e-05
Timeseries for "341" is stationary.                p-value is
0.024458939513174457
Timeseries for "342" is stationary.                p-value is
8.80299947584453e-22
Timeseries for "343" is stationary.                p-value is
0.0025455537863433918
Timeseries for "344" is stationary.                p-value is
0.000171545237044495
Timeseries for "345" is stationary.                p-value is
0.0004347580821028395
Timeseries for "346" is stationary.                p-value is
4.027542336281311e-05
Timeseries for "347" is stationary.                p-value is
9.323362886751891e-05
Timeseries for "348" is stationary.                p-value is
0.001596161922824127
```

```
Timeseries for "349" is stationary.               p-value is
2.599161815081659e-05
Timeseries for "350" is stationary.               p-value is
0.002818972880596073
Timeseries for "351" is stationary.               p-value is
0.00013922679432018566
Timeseries for "352" is stationary.               p-value is
0.04481384730031847
Timeseries for "353" is stationary.               p-value is
6.203586391737858e-05
Timeseries for "354" is stationary.               p-value is
0.0018002150858307112
Timeseries for "355" is stationary.               p-value is
0.003388497575288476
Timeseries for "356" is stationary.               p-value is
0.0019667757787619963
Timeseries for "357" is stationary.               p-value is
0.00254489802701704
Timeseries for "358" is stationary.               p-value is
0.0021694578498596675
Timeseries for "359" is stationary.               p-value is
0.0007575061228127756
Timeseries for "360" is stationary.               p-value is
0.00021343745292771302
Timeseries for "361" is stationary.               p-value is
5.7338625887903014e-05
Timeseries for "362" is stationary.               p-value is
0.005608341846038478
Timeseries for "363" is stationary.               p-value is
0.0014778227742358158
Timeseries for "364" is stationary.               p-value is
0.00035559780959084524
Timeseries for "365" is stationary.               p-value is
0.012319364620433915
Timeseries for "S1" is stationary.        p-value is
0.0056604416023610755
```

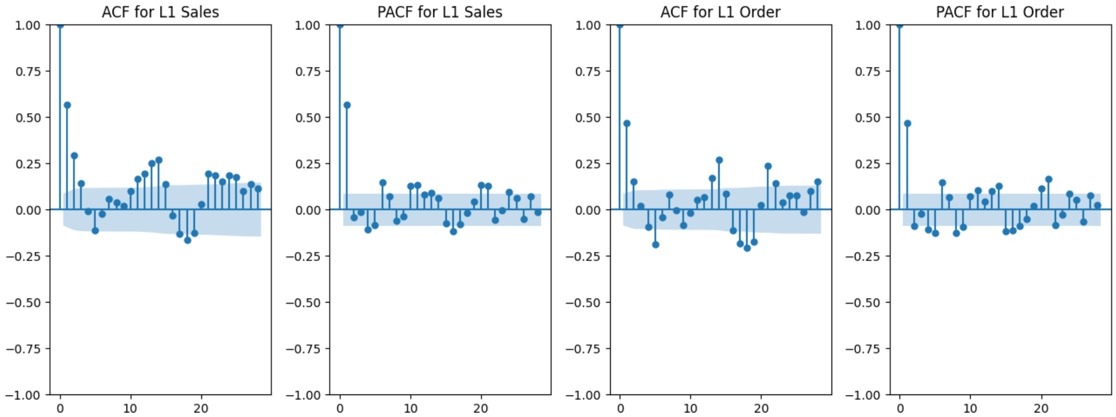
```
Timeseries for "S2" is stationary.         p-value is
0.003571295858539642
Timeseries for "S3" is stationary.         p-value is
0.011274338394491178
Timeseries for "S4" is stationary.         p-value is
0.009377000210500017
Timeseries for "L1" is stationary.         p-value is
0.007358214508015907
Timeseries for "L2" is stationary.         p-value is
0.009166726145432423
Timeseries for "L3" is stationary.         p-value is
0.004101073644607693
Timeseries for "L4" is stationary.         p-value is
0.0028948095280306304
Timeseries for "L5" is stationary.         p-value is
0.004235997270914649
Timeseries for "R1" is stationary.         p-value is
0.008887459005395964
Timeseries for "R2" is stationary.         p-value is
0.006066321673956619
Timeseries for "R3" is stationary.         p-value is
0.0059852511474317895
Timeseries for "R4" is stationary.         p-value is
0.0057033709981046225
```

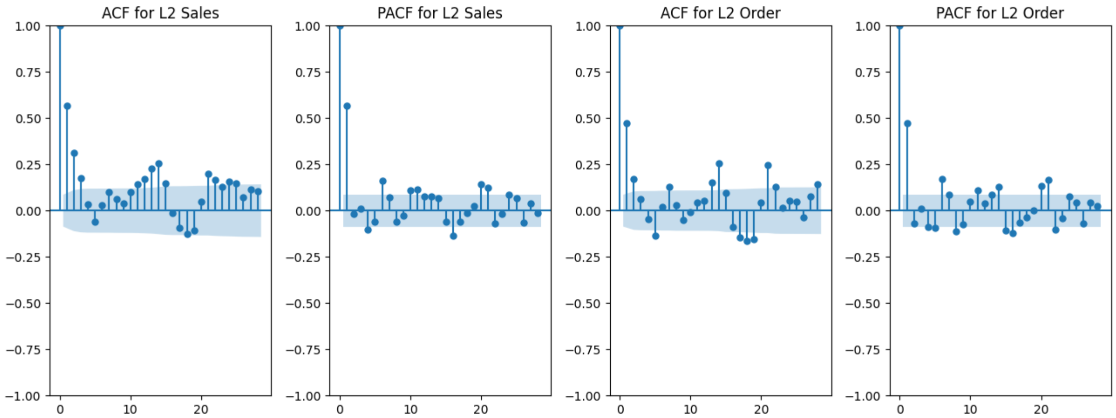
2.2 9. ACF/PACF Charts

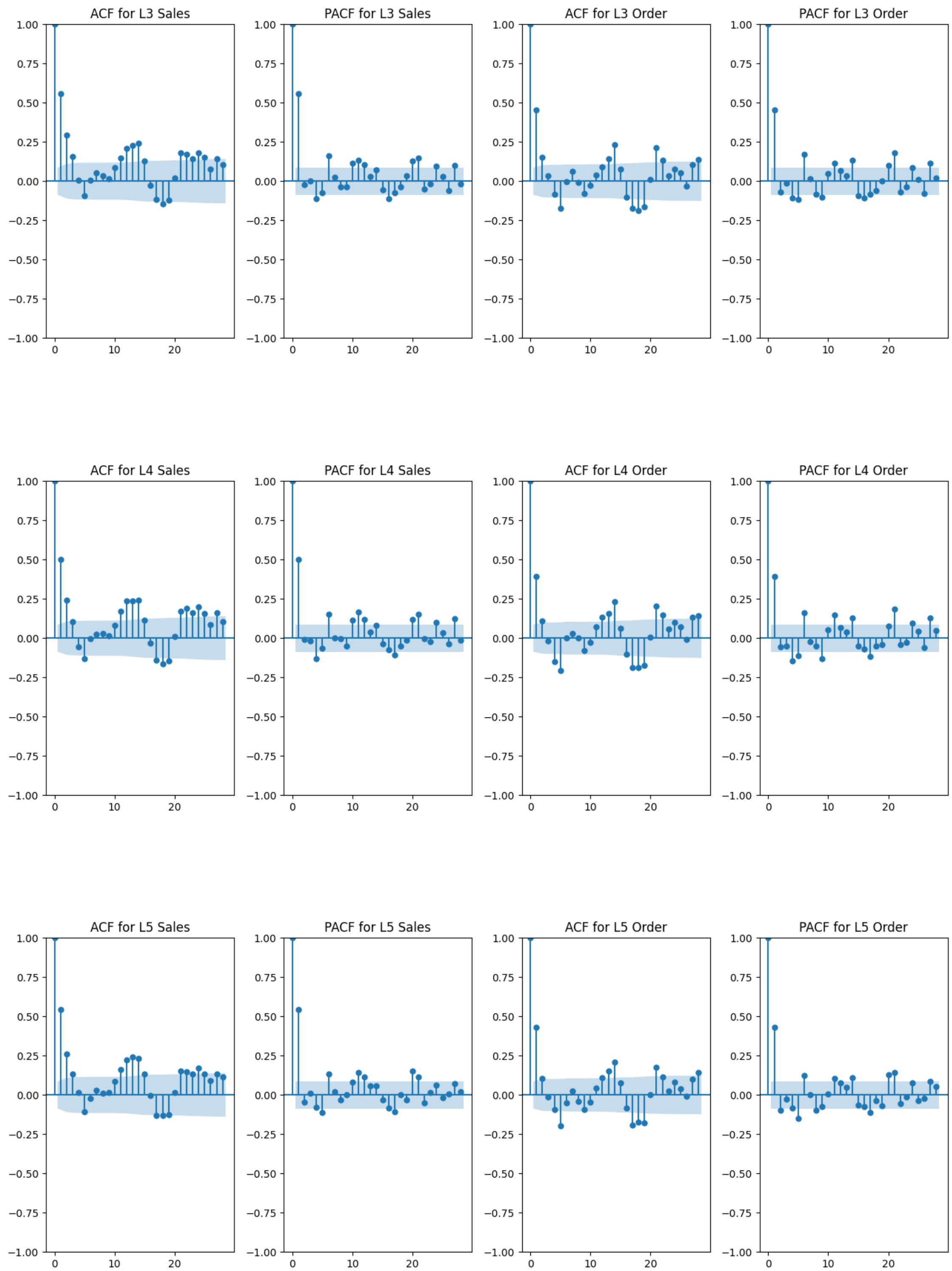
[45]:
```
# ACF v/s PACF Plot
def acf_pacf_plot(series_sales:pd.Series,series_order:pd.Series)->None:
  fig, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(13, 5))
  # Plot Sales ACF in first cell
  plot_acf(series_sales, ax=ax1)
  ax1.set_title(f'ACF for {series_sales.name} Sales')
  # Plot Sales PACF in second cell
  plot_pacf(series_sales, ax=ax2)
  ax2.set_title(f'PACF for {series_sales.name} Sales')

  # Plot Sales ACF in first cell
  plot_acf(series_order, ax=ax3)
  ax3.set_title(f'ACF for {series_sales.name} Order')
```

```python
    # Plot Sales PACF in second cell
    plot_pacf(series_order, ax=ax4)
    ax4.set_title(f'PACF for {series_sales.name} Order')

    # Adjust layout
    plt.tight_layout()
    plt.show()
```

[46]:
```python
# Location Wise Sales and Order ACF/PACF Plot
data_sales = location_wise_sales
data_order = location_wise_order
for column in data_sales.columns:
    acf_pacf_plot(data_sales[column], data_order[column])
```

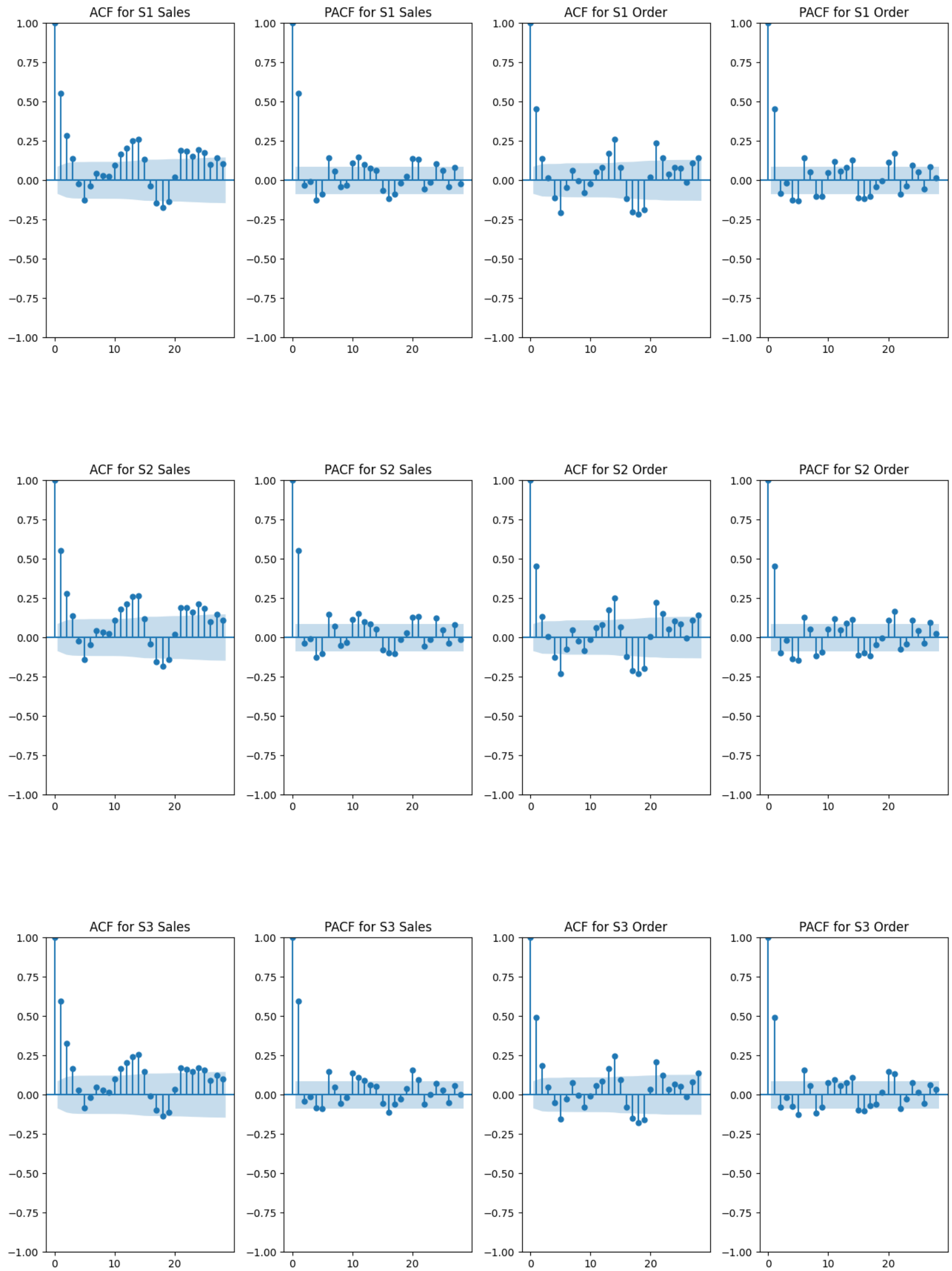


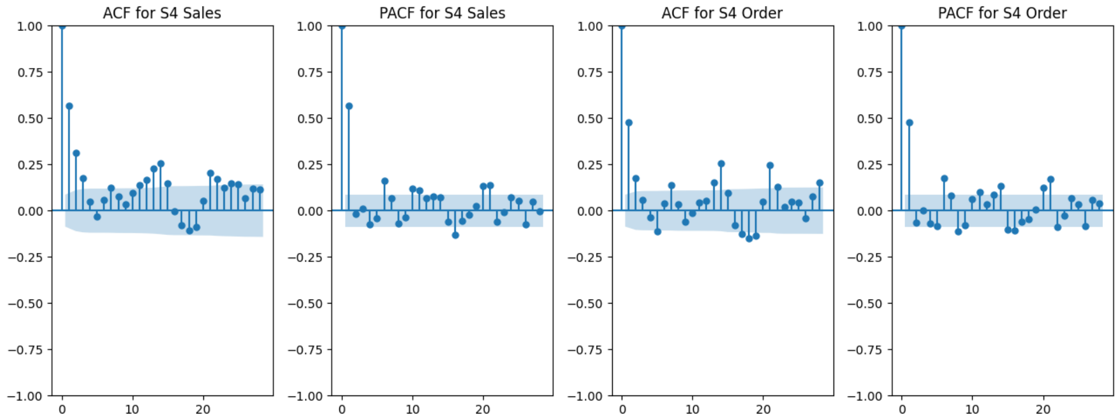




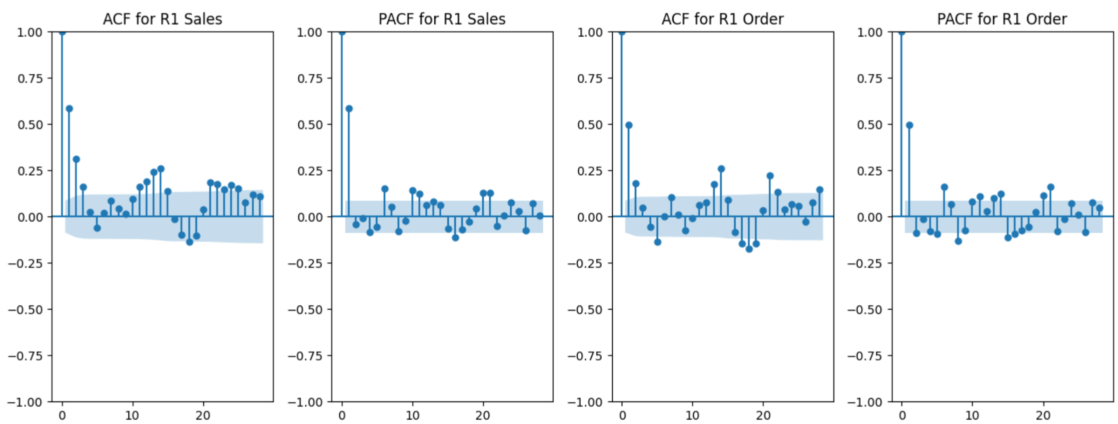
[47]:
```
# Store type Wise Sales and Order ACF/PACF Plot
data_sales = store_type_wise_sales
```

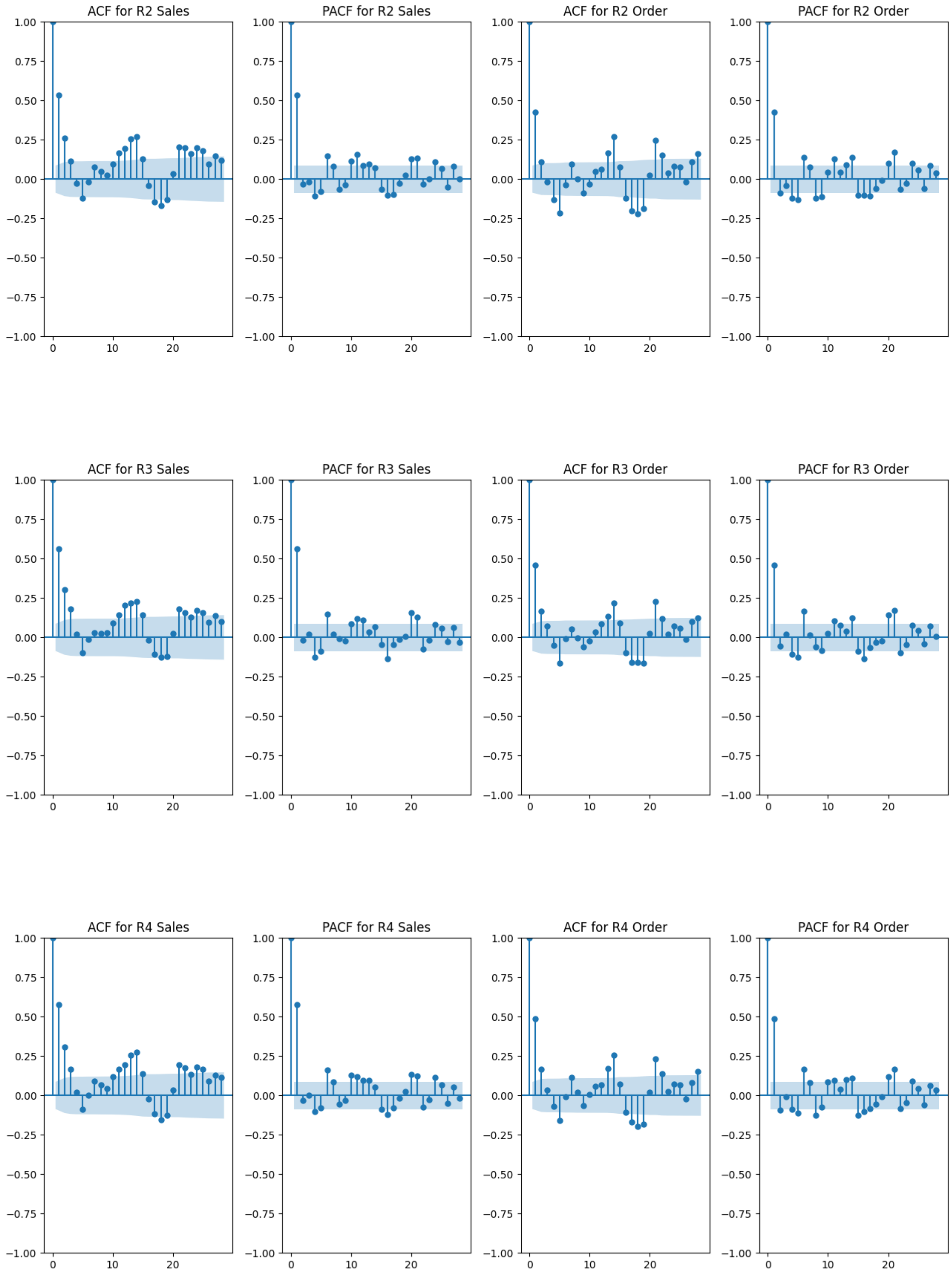
```python
data_order = store_type_wise_order
for column in data_sales.columns:
  acf_pacf_plot(data_sales[column], data_order[column])
```





[48]:
```python
# Region Wise Sales and Order ACF/PACF Plot
data_sales = region_wise_sales
data_order = region_wise_order
for column in data_sales.columns:
  acf_pacf_plot(data_sales[column], data_order[column])
```

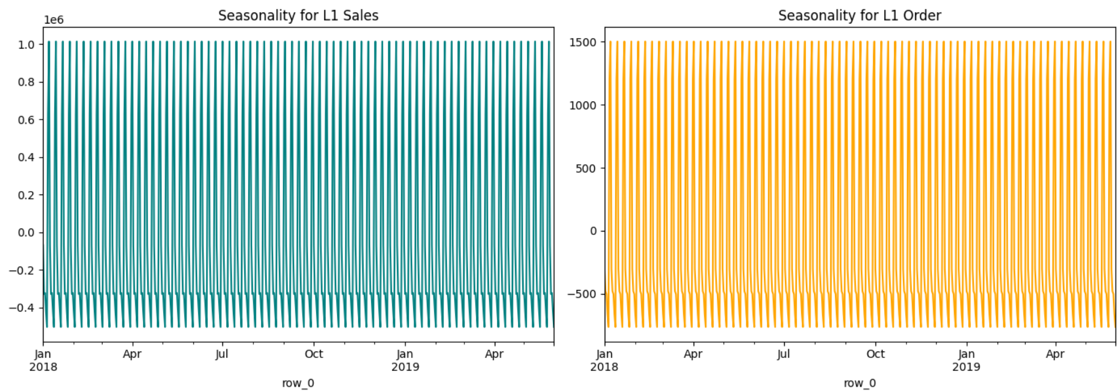


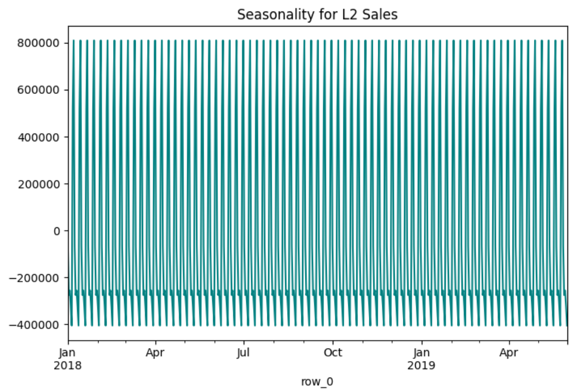

ACF for R2 Sales
PACF for R2 Sales
ACF for R2 Order
PACF for R2 Order
ACF for R3 Sales
PACF for R3 Sales
ACF for R3 Order
PACF for R3 Order
ACF for R4 Sales
PACF for R4 Sales
ACF for R4 Order
PACF for R4 Order

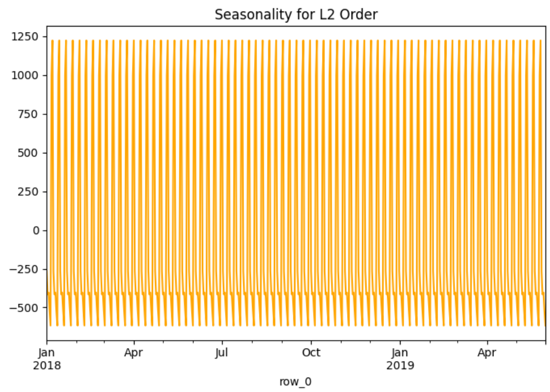
2.3 10. Seasonality Charts

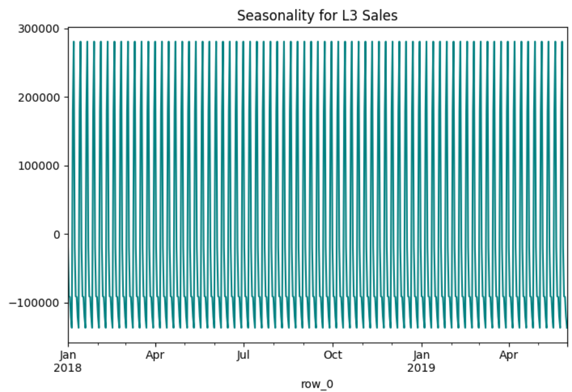
[49]:
```
# Seasonality Charts function
from statsmodels.tsa.seasonal import seasonal_decompose

def seasonal_chart(series_sales, series_order):

  fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))
  # Plot Sales Seasonality in first cell
  result = seasonal_decompose(series_sales, model='additive', period=None)
  result.seasonal.plot(ax=ax1, color='teal')
  ax1.set_title(f'Seasonality for {series_sales.name} Sales')
  # Plot Order Seasonality in second cell
  result = seasonal_decompose(series_order, model='additive', period=None)
  result.seasonal.plot(ax=ax2, color='Orange')
  ax2.set_title(f'Seasonality for {series_order.name} Order')

  # Adjust layout
  plt.tight_layout()
  plt.show()
```

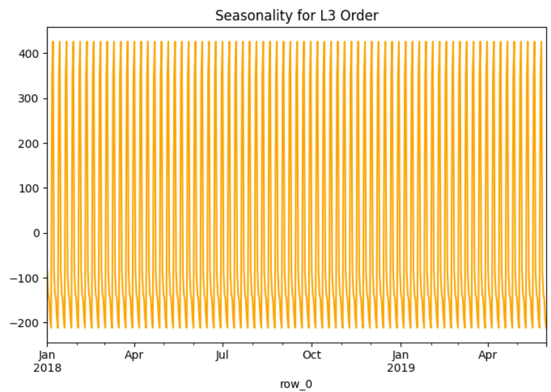
[50]:
```
# Location Wise Sales and Order Seasonality Plot
data_sales = location_wise_sales
data_order = location_wise_order
for column in data_sales.columns:
  seasonal_chart(data_sales[column], data_order[column])
```

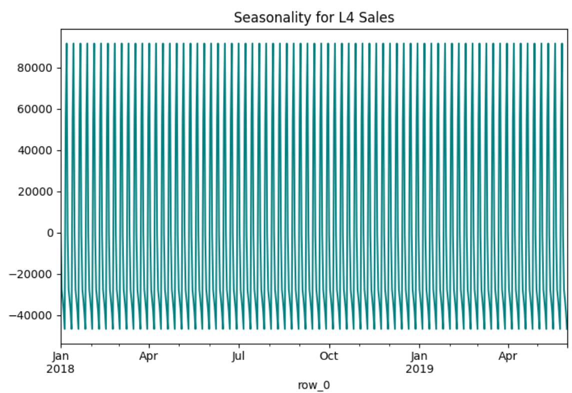


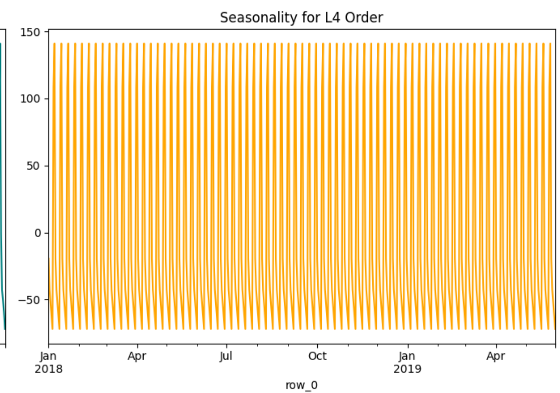

Seasonality for L2 Sales
800000
600000
400000
200000
0
−200000
−400000
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0

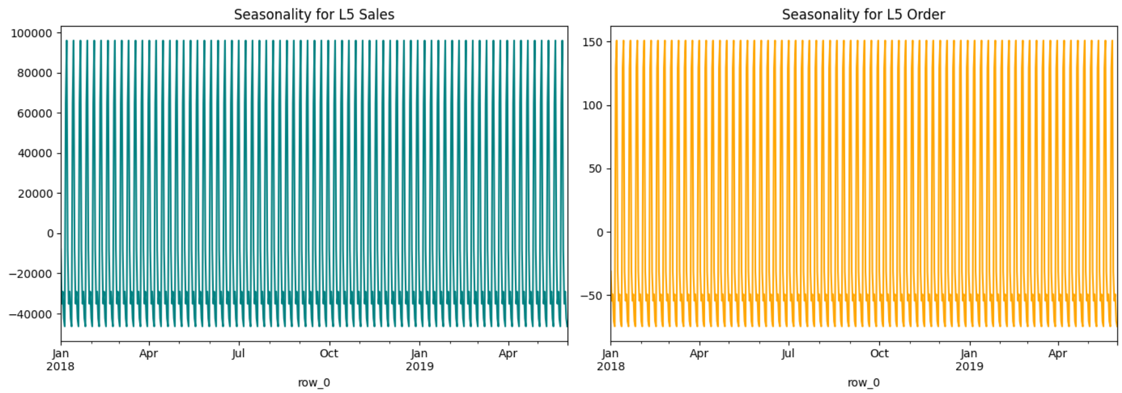

Seasonality for L2 Order
1250
1000
750
500
250
0
−250
−500
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0


Seasonality for L3 Sales
300000
200000
100000
0
−100000
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0

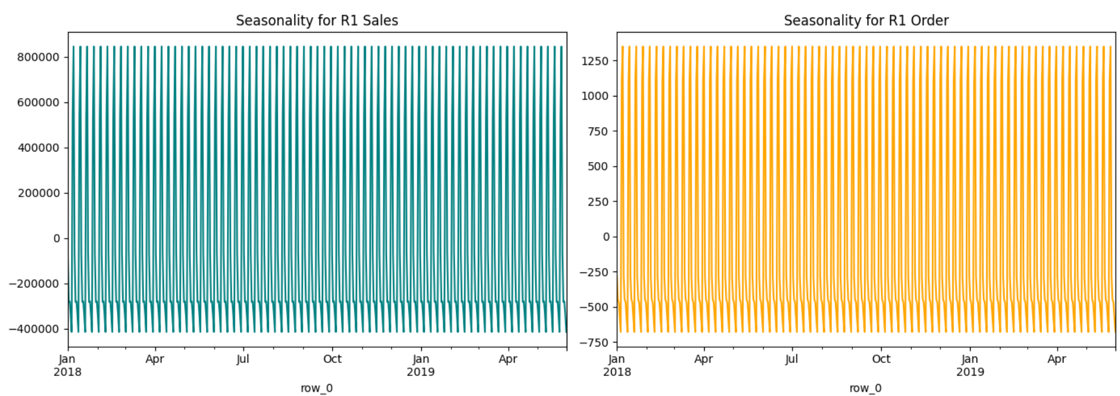

Seasonality for L3 Order
400
300
200
100
0
−100
−200
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0

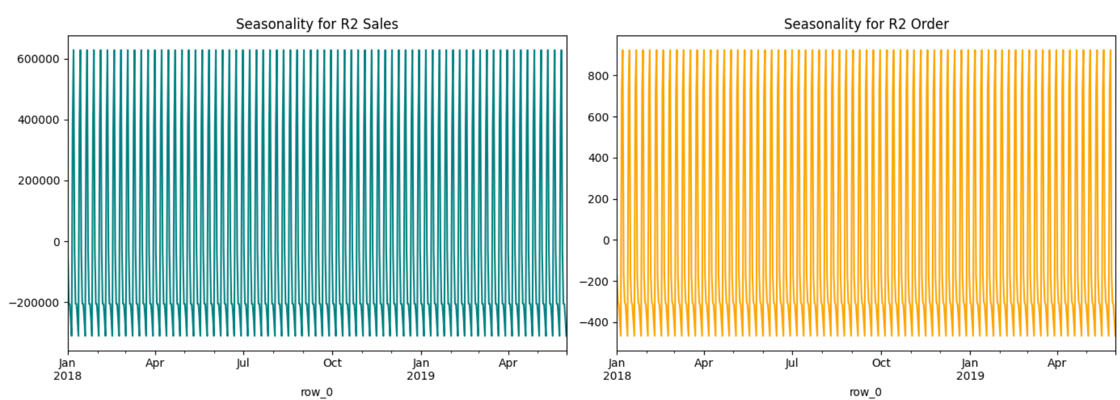

Seasonality for L4 Sales
80000
60000
40000
20000
0
−20000
−40000
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0

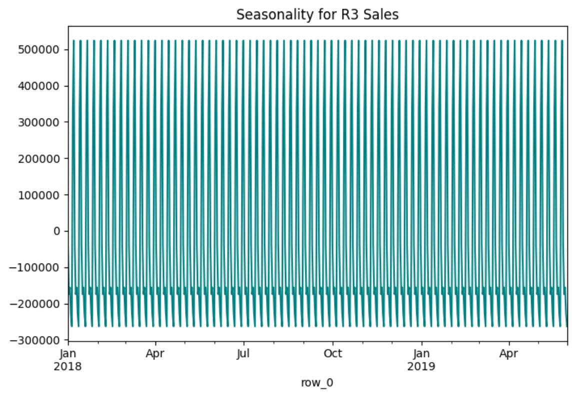

Seasonality for L4 Order
150
100
50
0
−50
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0

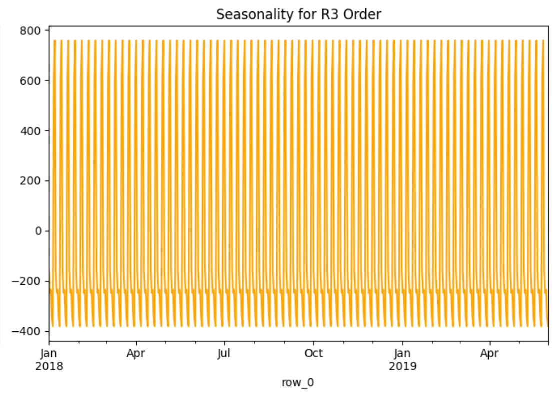


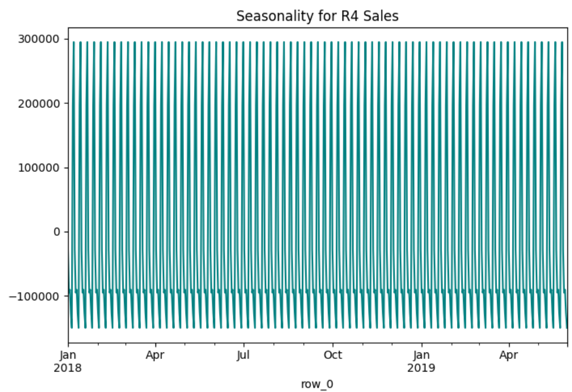
[51]:
```python
# REgion Wise Sales and Order Seasonality Plot
data_sales = region_wise_sales
data_order = region_wise_order
for column in data_sales.columns:
  seasonal_chart(data_sales[column], data_order[column])
```

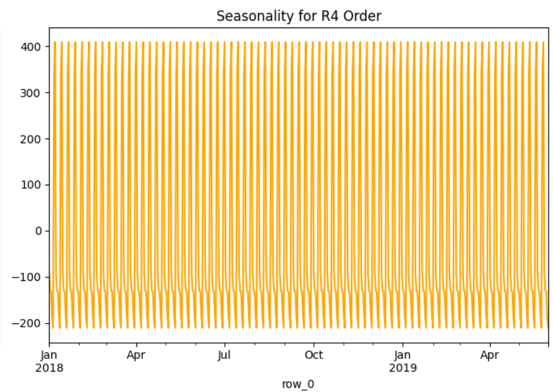






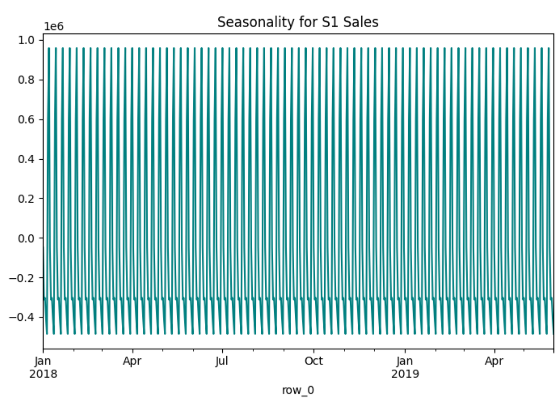


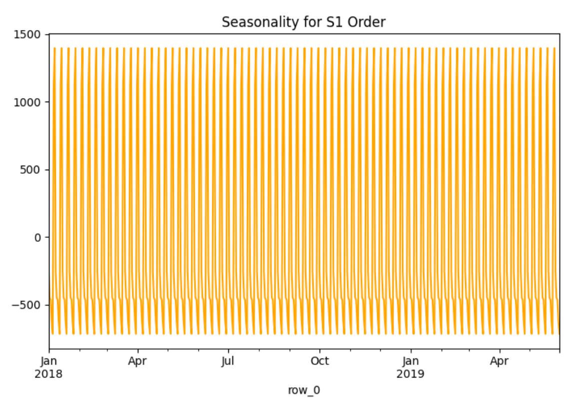


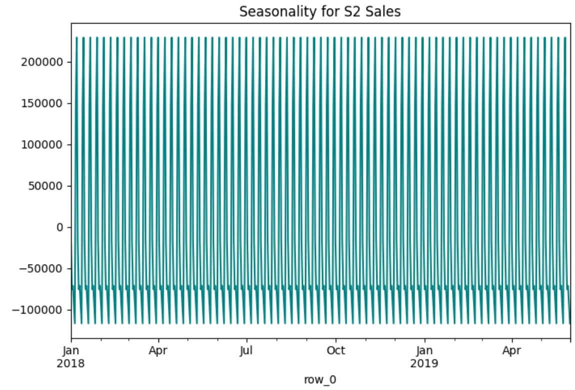


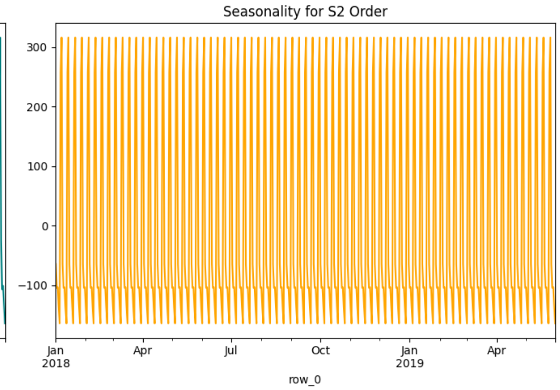
[52]:

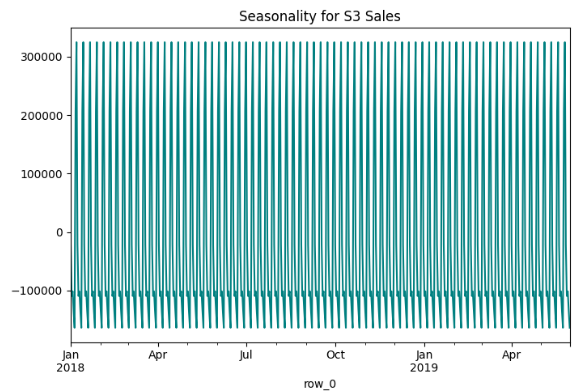
```
# StoreType Wise Sales and Order Seasonality Plot
data_sales = store_type_wise_sales
data_order = store_type_wise_order
for column in data_sales.columns:
  seasonal_chart(data_sales[column], data_order[column])
```

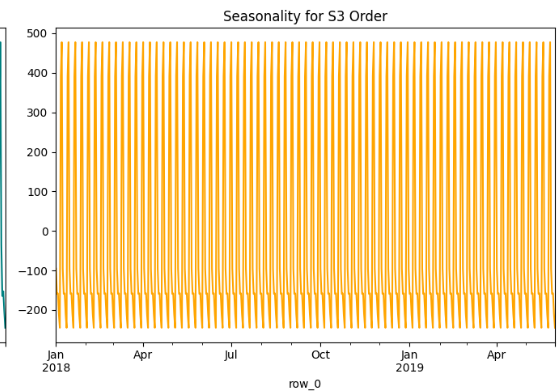


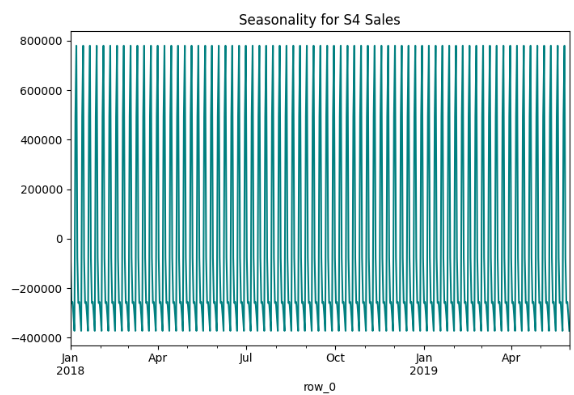


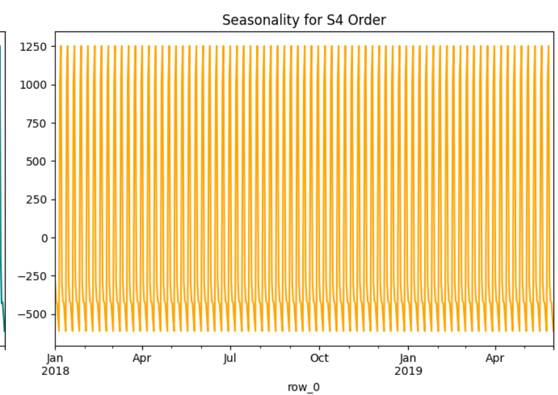

Seasonality for S2 Sales
200000
150000
100000
50000
0
−50000
−100000
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0


Seasonality for S2 Order
300
200
100
0
−100
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0


Seasonality for S3 Sales
300000
200000
100000
0
−100000
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0


Seasonality for S3 Order
500
400
300
200
100
0
−100
−200
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0


Seasonality for S4 Sales
800000
600000
400000
200000
0
−200000
−400000
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0


Seasonality for S4 Order
1250
1000
750
500
250
0
−250
−500
Jan 2018
Apr
Jul
Oct
Jan 2019
Apr
row_0

3 C. Model building and Evaluation

3.1 10. Data splitting

To build a model we will split data into 80:20 ratio. First 80% rows will be train data whereas remaininf 20% will be test data.

[53]:
```
# Train/Test splitting stage of pipeline
from sklearn.base import BaseEstimator, TransformerMixin

class TimeSeriesSplitter(BaseEstimator, TransformerMixin):
    def __init__(self, test_size=0.2):
        self.test_size = test_size
        self.train_data, self.test_data = None, None

    def fit(self, X, y=None):
        """No fitting required; just for compatibility."""
        return self

    def transform(self, X, y=None):
        """Split the single-column time series into train and test sets."""
        n_rows = len(X)
        split_index = int(n_rows * (1 - self.test_size))
        self.train_data = X.iloc[:split_index].asfreq('D')
        self.test_data = X.iloc[split_index:].asfreq('D')
        return self.train_data, self.test_data
```

3.2 11. SARIMAX Model Training

To have better accuracy in forecasting we will use exogenous variable

3.2.1 Model building preperation

[54]:
```
# Seasonality Factor function
def get_seasonal_factor_fft(data:pd.Series)->int:
    """
    Automatically detects seasonality using FFT.
    Args:
        data (pd.Series): Time series data.
    Returns:
        int: Seasonal factor (dominant period).
    """
    fft = np.fft.fft(data - np.mean(data))  # Remove mean for better results
    freqs = np.fft.fftfreq(len(data))
    magnitudes = np.abs(fft)
    dominant_freq = freqs[np.argmax(magnitudes[1:]) + 1]  # Ignore zero␣
↪frequency
```

```
    seasonal_period = int(round(1 / dominant_freq)) if dominant_freq != 0 else␣
↪None
    return abs(seasonal_period)
```

[55]:
```
for column in train_sales.columns:
    seasonal_factor_fft = get_seasonal_factor_fft(train_sales[column])
    print(f"Detected seasonal factor (FFT) {column}: {seasonal_factor_fft}")
```

```
Detected seasonal factor (FFT) Sales: 12
Detected seasonal factor (FFT) 1: 6
Detected seasonal factor (FFT) 2: 12
Detected seasonal factor (FFT) 3: 172
Detected seasonal factor (FFT) 4: 12
Detected seasonal factor (FFT) 5: 7
Detected seasonal factor (FFT) 6: 13
Detected seasonal factor (FFT) 7: 172
Detected seasonal factor (FFT) 8: 7
Detected seasonal factor (FFT) 9: 12
Detected seasonal factor (FFT) 10: 12
Detected seasonal factor (FFT) 11: 12
Detected seasonal factor (FFT) 12: 12
Detected seasonal factor (FFT) 13: 12
Detected seasonal factor (FFT) 14: 12
Detected seasonal factor (FFT) 15: 7
Detected seasonal factor (FFT) 16: 12
Detected seasonal factor (FFT) 17: 12
Detected seasonal factor (FFT) 18: 7
Detected seasonal factor (FFT) 19: 12
Detected seasonal factor (FFT) 20: 12
Detected seasonal factor (FFT) 21: 172
Detected seasonal factor (FFT) 22: 172
Detected seasonal factor (FFT) 23: 12
Detected seasonal factor (FFT) 24: 3
Detected seasonal factor (FFT) 25: 172
Detected seasonal factor (FFT) 26: 172
Detected seasonal factor (FFT) 27: 12
Detected seasonal factor (FFT) 28: 12
Detected seasonal factor (FFT) 29: 12
Detected seasonal factor (FFT) 30: 12
Detected seasonal factor (FFT) 31: 12
Detected seasonal factor (FFT) 32: 172
Detected seasonal factor (FFT) 33: 12
Detected seasonal factor (FFT) 34: 12
Detected seasonal factor (FFT) 35: 7
Detected seasonal factor (FFT) 36: 13
Detected seasonal factor (FFT) 37: 12
Detected seasonal factor (FFT) 38: 12
```

```
Detected seasonal factor (FFT) 39: 13
Detected seasonal factor (FFT) 40: 12
Detected seasonal factor (FFT) 41: 12
Detected seasonal factor (FFT) 42: 172
Detected seasonal factor (FFT) 43: 12
Detected seasonal factor (FFT) 44: 12
Detected seasonal factor (FFT) 45: 13
Detected seasonal factor (FFT) 46: 12
Detected seasonal factor (FFT) 47: 12
Detected seasonal factor (FFT) 48: 12
Detected seasonal factor (FFT) 49: 7
Detected seasonal factor (FFT) 50: 12
Detected seasonal factor (FFT) 51: 12
Detected seasonal factor (FFT) 52: 12
Detected seasonal factor (FFT) 53: 3
Detected seasonal factor (FFT) 54: 12
Detected seasonal factor (FFT) 55: 12
Detected seasonal factor (FFT) 56: 12
Detected seasonal factor (FFT) 57: 103
Detected seasonal factor (FFT) 58: 516
Detected seasonal factor (FFT) 59: 12
Detected seasonal factor (FFT) 60: 3
Detected seasonal factor (FFT) 61: 12
Detected seasonal factor (FFT) 62: 12
Detected seasonal factor (FFT) 63: 12
Detected seasonal factor (FFT) 64: 7
Detected seasonal factor (FFT) 65: 516
Detected seasonal factor (FFT) 66: 13
Detected seasonal factor (FFT) 67: 13
Detected seasonal factor (FFT) 68: 12
Detected seasonal factor (FFT) 69: 7
Detected seasonal factor (FFT) 70: 7
Detected seasonal factor (FFT) 71: 12
Detected seasonal factor (FFT) 72: 12
Detected seasonal factor (FFT) 73: 12
Detected seasonal factor (FFT) 74: 13
Detected seasonal factor (FFT) 75: 12
Detected seasonal factor (FFT) 76: 12
Detected seasonal factor (FFT) 77: 12
Detected seasonal factor (FFT) 78: 13
Detected seasonal factor (FFT) 79: 13
Detected seasonal factor (FFT) 80: 12
Detected seasonal factor (FFT) 81: 12
Detected seasonal factor (FFT) 82: 172
Detected seasonal factor (FFT) 83: 12
Detected seasonal factor (FFT) 84: 12
Detected seasonal factor (FFT) 85: 12
Detected seasonal factor (FFT) 86: 7
```

```
Detected seasonal factor (FFT) 87: 12
Detected seasonal factor (FFT) 88: 516
Detected seasonal factor (FFT) 89: 172
Detected seasonal factor (FFT) 90: 13
Detected seasonal factor (FFT) 91: 12
Detected seasonal factor (FFT) 92: 7
Detected seasonal factor (FFT) 93: 172
Detected seasonal factor (FFT) 94: 12
Detected seasonal factor (FFT) 95: 12
Detected seasonal factor (FFT) 96: 172
Detected seasonal factor (FFT) 97: 7
Detected seasonal factor (FFT) 98: 172
Detected seasonal factor (FFT) 99: 12
Detected seasonal factor (FFT) 100: 12
Detected seasonal factor (FFT) 101: 12
Detected seasonal factor (FFT) 102: 12
Detected seasonal factor (FFT) 103: 12
Detected seasonal factor (FFT) 104: 12
Detected seasonal factor (FFT) 105: 172
Detected seasonal factor (FFT) 106: 12
Detected seasonal factor (FFT) 107: 3
Detected seasonal factor (FFT) 108: 12
Detected seasonal factor (FFT) 109: 3
Detected seasonal factor (FFT) 110: 172
Detected seasonal factor (FFT) 111: 12
Detected seasonal factor (FFT) 112: 7
Detected seasonal factor (FFT) 113: 172
Detected seasonal factor (FFT) 114: 12
Detected seasonal factor (FFT) 115: 13
Detected seasonal factor (FFT) 116: 172
Detected seasonal factor (FFT) 117: 12
Detected seasonal factor (FFT) 118: 172
Detected seasonal factor (FFT) 119: 12
Detected seasonal factor (FFT) 120: 12
Detected seasonal factor (FFT) 121: 12
Detected seasonal factor (FFT) 122: 12
Detected seasonal factor (FFT) 123: 12
Detected seasonal factor (FFT) 124: 12
Detected seasonal factor (FFT) 125: 172
Detected seasonal factor (FFT) 126: 12
Detected seasonal factor (FFT) 127: 3
Detected seasonal factor (FFT) 128: 12
Detected seasonal factor (FFT) 129: 3
Detected seasonal factor (FFT) 130: 12
Detected seasonal factor (FFT) 131: 13
Detected seasonal factor (FFT) 132: 3
Detected seasonal factor (FFT) 133: 172
Detected seasonal factor (FFT) 134: 12
```

```
Detected seasonal factor (FFT) 135: 12
Detected seasonal factor (FFT) 136: 12
Detected seasonal factor (FFT) 137: 12
Detected seasonal factor (FFT) 138: 12
Detected seasonal factor (FFT) 139: 172
Detected seasonal factor (FFT) 140: 12
Detected seasonal factor (FFT) 141: 12
Detected seasonal factor (FFT) 142: 12
Detected seasonal factor (FFT) 143: 12
Detected seasonal factor (FFT) 144: 7
Detected seasonal factor (FFT) 145: 12
Detected seasonal factor (FFT) 146: 172
Detected seasonal factor (FFT) 147: 172
Detected seasonal factor (FFT) 148: 7
Detected seasonal factor (FFT) 149: 13
Detected seasonal factor (FFT) 150: 13
Detected seasonal factor (FFT) 151: 12
Detected seasonal factor (FFT) 152: 12
Detected seasonal factor (FFT) 153: 172
Detected seasonal factor (FFT) 154: 3
Detected seasonal factor (FFT) 155: 172
Detected seasonal factor (FFT) 156: 12
Detected seasonal factor (FFT) 157: 7
Detected seasonal factor (FFT) 158: 12
Detected seasonal factor (FFT) 159: 516
Detected seasonal factor (FFT) 160: 7
Detected seasonal factor (FFT) 161: 12
Detected seasonal factor (FFT) 162: 12
Detected seasonal factor (FFT) 163: 12
Detected seasonal factor (FFT) 164: 13
Detected seasonal factor (FFT) 165: 12
Detected seasonal factor (FFT) 166: 12
Detected seasonal factor (FFT) 167: 103
Detected seasonal factor (FFT) 168: 12
Detected seasonal factor (FFT) 169: 12
Detected seasonal factor (FFT) 170: 12
Detected seasonal factor (FFT) 171: 12
Detected seasonal factor (FFT) 172: 12
Detected seasonal factor (FFT) 173: 12
Detected seasonal factor (FFT) 174: 12
Detected seasonal factor (FFT) 175: 13
Detected seasonal factor (FFT) 176: 12
Detected seasonal factor (FFT) 177: 12
Detected seasonal factor (FFT) 178: 13
Detected seasonal factor (FFT) 179: 172
Detected seasonal factor (FFT) 180: 12
Detected seasonal factor (FFT) 181: 3
Detected seasonal factor (FFT) 182: 172
```

```
Detected seasonal factor (FFT) 183: 12
Detected seasonal factor (FFT) 184: 12
Detected seasonal factor (FFT) 185: 12
Detected seasonal factor (FFT) 186: 12
Detected seasonal factor (FFT) 187: 12
Detected seasonal factor (FFT) 188: 12
Detected seasonal factor (FFT) 189: 172
Detected seasonal factor (FFT) 190: 172
Detected seasonal factor (FFT) 191: 12
Detected seasonal factor (FFT) 192: 172
Detected seasonal factor (FFT) 193: 12
Detected seasonal factor (FFT) 194: 7
Detected seasonal factor (FFT) 195: 12
Detected seasonal factor (FFT) 196: 12
Detected seasonal factor (FFT) 197: 3
Detected seasonal factor (FFT) 198: 3
Detected seasonal factor (FFT) 199: 12
Detected seasonal factor (FFT) 200: 12
Detected seasonal factor (FFT) 201: 172
Detected seasonal factor (FFT) 202: 12
Detected seasonal factor (FFT) 203: 12
Detected seasonal factor (FFT) 204: 7
Detected seasonal factor (FFT) 205: 12
Detected seasonal factor (FFT) 206: 516
Detected seasonal factor (FFT) 207: 12
Detected seasonal factor (FFT) 208: 12
Detected seasonal factor (FFT) 209: 12
Detected seasonal factor (FFT) 210: 7
Detected seasonal factor (FFT) 211: 172
Detected seasonal factor (FFT) 212: 7
Detected seasonal factor (FFT) 213: 12
Detected seasonal factor (FFT) 214: 12
Detected seasonal factor (FFT) 215: 12
Detected seasonal factor (FFT) 216: 7
Detected seasonal factor (FFT) 217: 12
Detected seasonal factor (FFT) 218: 12
Detected seasonal factor (FFT) 219: 172
Detected seasonal factor (FFT) 220: 12
Detected seasonal factor (FFT) 221: 516
Detected seasonal factor (FFT) 222: 13
Detected seasonal factor (FFT) 223: 3
Detected seasonal factor (FFT) 224: 12
Detected seasonal factor (FFT) 225: 12
Detected seasonal factor (FFT) 226: 3
Detected seasonal factor (FFT) 227: 12
Detected seasonal factor (FFT) 228: 12
Detected seasonal factor (FFT) 229: 12
Detected seasonal factor (FFT) 230: 12
```

```
Detected seasonal factor (FFT) 231: 12
Detected seasonal factor (FFT) 232: 12
Detected seasonal factor (FFT) 233: 516
Detected seasonal factor (FFT) 234: 12
Detected seasonal factor (FFT) 235: 12
Detected seasonal factor (FFT) 236: 12
Detected seasonal factor (FFT) 237: 516
Detected seasonal factor (FFT) 238: 12
Detected seasonal factor (FFT) 239: 172
Detected seasonal factor (FFT) 240: 172
Detected seasonal factor (FFT) 241: 12
Detected seasonal factor (FFT) 242: 12
Detected seasonal factor (FFT) 243: 12
Detected seasonal factor (FFT) 244: 12
Detected seasonal factor (FFT) 245: 12
Detected seasonal factor (FFT) 246: 13
Detected seasonal factor (FFT) 247: 12
Detected seasonal factor (FFT) 248: 12
Detected seasonal factor (FFT) 249: 12
Detected seasonal factor (FFT) 250: 12
Detected seasonal factor (FFT) 251: 12
Detected seasonal factor (FFT) 252: 12
Detected seasonal factor (FFT) 253: 172
Detected seasonal factor (FFT) 254: 12
Detected seasonal factor (FFT) 255: 12
Detected seasonal factor (FFT) 256: 12
Detected seasonal factor (FFT) 257: 12
Detected seasonal factor (FFT) 258: 12
Detected seasonal factor (FFT) 259: 12
Detected seasonal factor (FFT) 260: 7
Detected seasonal factor (FFT) 261: 12
Detected seasonal factor (FFT) 262: 12
Detected seasonal factor (FFT) 263: 3
Detected seasonal factor (FFT) 264: 12
Detected seasonal factor (FFT) 265: 516
Detected seasonal factor (FFT) 266: 12
Detected seasonal factor (FFT) 267: 516
Detected seasonal factor (FFT) 268: 13
Detected seasonal factor (FFT) 269: 12
Detected seasonal factor (FFT) 270: 12
Detected seasonal factor (FFT) 271: 13
Detected seasonal factor (FFT) 272: 12
Detected seasonal factor (FFT) 273: 7
Detected seasonal factor (FFT) 274: 12
Detected seasonal factor (FFT) 275: 12
Detected seasonal factor (FFT) 276: 13
Detected seasonal factor (FFT) 277: 172
Detected seasonal factor (FFT) 278: 172
```

```
Detected seasonal factor (FFT) 279: 7
Detected seasonal factor (FFT) 280: 172
Detected seasonal factor (FFT) 281: 7
Detected seasonal factor (FFT) 282: 12
Detected seasonal factor (FFT) 283: 13
Detected seasonal factor (FFT) 284: 74
Detected seasonal factor (FFT) 285: 12
Detected seasonal factor (FFT) 286: 13
Detected seasonal factor (FFT) 287: 12
Detected seasonal factor (FFT) 288: 7
Detected seasonal factor (FFT) 289: 172
Detected seasonal factor (FFT) 290: 12
Detected seasonal factor (FFT) 291: 7
Detected seasonal factor (FFT) 292: 12
Detected seasonal factor (FFT) 293: 13
Detected seasonal factor (FFT) 294: 7
Detected seasonal factor (FFT) 295: 12
Detected seasonal factor (FFT) 296: 516
Detected seasonal factor (FFT) 297: 12
Detected seasonal factor (FFT) 298: 172
Detected seasonal factor (FFT) 299: 12
Detected seasonal factor (FFT) 300: 172
Detected seasonal factor (FFT) 301: 12
Detected seasonal factor (FFT) 302: 12
Detected seasonal factor (FFT) 303: 7
Detected seasonal factor (FFT) 304: 3
Detected seasonal factor (FFT) 305: 12
Detected seasonal factor (FFT) 306: 12
Detected seasonal factor (FFT) 307: 12
Detected seasonal factor (FFT) 308: 7
Detected seasonal factor (FFT) 309: 172
Detected seasonal factor (FFT) 310: 6
Detected seasonal factor (FFT) 311: 13
Detected seasonal factor (FFT) 312: 12
Detected seasonal factor (FFT) 313: 12
Detected seasonal factor (FFT) 314: 12
Detected seasonal factor (FFT) 315: 258
Detected seasonal factor (FFT) 316: 12
Detected seasonal factor (FFT) 317: 516
Detected seasonal factor (FFT) 318: 12
Detected seasonal factor (FFT) 319: 12
Detected seasonal factor (FFT) 320: 12
Detected seasonal factor (FFT) 321: 12
Detected seasonal factor (FFT) 322: 12
Detected seasonal factor (FFT) 323: 172
Detected seasonal factor (FFT) 324: 12
Detected seasonal factor (FFT) 325: 12
Detected seasonal factor (FFT) 326: 12
```

```
Detected seasonal factor (FFT) 327: 12
Detected seasonal factor (FFT) 328: 7
Detected seasonal factor (FFT) 329: 172
Detected seasonal factor (FFT) 330: 12
Detected seasonal factor (FFT) 331: 12
Detected seasonal factor (FFT) 332: 12
Detected seasonal factor (FFT) 333: 12
Detected seasonal factor (FFT) 334: 12
Detected seasonal factor (FFT) 335: 172
Detected seasonal factor (FFT) 336: 12
Detected seasonal factor (FFT) 337: 12
Detected seasonal factor (FFT) 338: 3
Detected seasonal factor (FFT) 339: 172
Detected seasonal factor (FFT) 340: 12
Detected seasonal factor (FFT) 341: 12
Detected seasonal factor (FFT) 342: 12
Detected seasonal factor (FFT) 343: 13
Detected seasonal factor (FFT) 344: 7
Detected seasonal factor (FFT) 345: 12
Detected seasonal factor (FFT) 346: 12
Detected seasonal factor (FFT) 347: 12
Detected seasonal factor (FFT) 348: 7
Detected seasonal factor (FFT) 349: 12
Detected seasonal factor (FFT) 350: 12
Detected seasonal factor (FFT) 351: 12
Detected seasonal factor (FFT) 352: 13
Detected seasonal factor (FFT) 353: 12
Detected seasonal factor (FFT) 354: 12
Detected seasonal factor (FFT) 355: 12
Detected seasonal factor (FFT) 356: 172
Detected seasonal factor (FFT) 357: 172
Detected seasonal factor (FFT) 358: 12
Detected seasonal factor (FFT) 359: 516
Detected seasonal factor (FFT) 360: 12
Detected seasonal factor (FFT) 361: 12
Detected seasonal factor (FFT) 362: 172
Detected seasonal factor (FFT) 363: 12
Detected seasonal factor (FFT) 364: 12
Detected seasonal factor (FFT) 365: 12
Detected seasonal factor (FFT) S1: 12
Detected seasonal factor (FFT) S2: 12
Detected seasonal factor (FFT) S3: 12
Detected seasonal factor (FFT) S4: 12
Detected seasonal factor (FFT) L1: 12
Detected seasonal factor (FFT) L2: 12
Detected seasonal factor (FFT) L3: 12
Detected seasonal factor (FFT) L4: 12
Detected seasonal factor (FFT) L5: 12
```

```
Detected seasonal factor (FFT) R1: 12
Detected seasonal factor (FFT) R2: 12
Detected seasonal factor (FFT) R3: 12
Detected seasonal factor (FFT) R4: 12
```

[56]:
```python
# Model training stage
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.base import BaseEstimator
from sklearn.metrics import (make_scorer, mean_squared_error as mse,␣
↪mean_absolute_error as mae, mean_absolute_percentage_error as mape)
import mlflow.statsmodels # Import mlflow.statsmodels

class SARIMAXEstimator(BaseEstimator):
    def __init__(self, order=(1,0,1), seasonal_order = (1,0,1,12)):
        self.order = order
        self.seasonal_order = seasonal_order
        self.model_ = None

    def fit(self, X, exog=None):
      self.exog_train=exog
      try:
        if isinstance(self.exog_train, pd.Series):
          self.model_ = SARIMAX(X, exog=self.exog_train, order=self.order,␣
↪seasonal_order=self.seasonal_order).fit(disp=False)
        else:
          self.model_ = SARIMAX(X, order=self.order, seasonal_order=self.
↪seasonal_order).fit(disp=False)
      except Exception as e:
        print(f"Skipping: order={self.order}, seasonal_order={self.
↪seasonal_order}. Error: {e}")
        self.model_ = None
      return self

    def predict(self, n_steps, exog=None):
      if self.model_ is None:
        return np.full(n_steps, 1E-10)
      try:
        if not isinstance(self.exog_train, pd.Series):
          return self.model_.forecast(steps=n_steps)
        elif isinstance(exog, pd.Series):
          return self.model_.forecast(steps=n_steps, exog=exog[:n_steps])
        else:
          raise ValueError('No exog data provided')
      except Exception as e:
        print(e)
      return None
```

```
    def score(self, X, exog=None):
        n_steps = len(X)
        predictions = self.predict(n_steps, exog)
        return mape(X, predictions)
```

[57]:
```
#@title Define the parameter grid
from itertools import product
p,d,q = range(1,2), [1], range(1, 2)
order = list(product(p, d, q))

# seasonal_parameters P, D, Q, S
P, D, Q, S = range(1, 2), [0], range(1, 2), [7]
seasonal_order = list(product(P, D, Q, S))

param_grid = list(product(order, seasonal_order))
```

3.2.2 Model Training with MLFLow

[58]:
```
# Model training for sales forecasting with MLFLow
import cloudpickle
import tempfile
import os

mlflow.set_experiment("Sore ID Sales Forecasting-1.1.0")
for column in train_sales.columns[0:1]:
  splitter = TimeSeriesSplitter()
  X_train, X_test = splitter.fit_transform(train_sales[column])
  X_train_exog, X_test_exog = splitter.fit_transform(exog_train_holiday)
  for order, seasonal in param_grid:
    with mlflow.start_run():
      sarimax_estimator = SARIMAXEstimator(order=order, seasonal_order=seasonal)
      sarimax_estimator.fit(X=X_train, exog=X_train_exog)

      if sarimax_estimator.model_ is not None:
          model_score = sarimax_estimator.score(X=X_test, exog=X_test_exog)
          mlflow.set_tag('data', column)
          mlflow.log_params({'order': order, 'seasonal_order': seasonal})
          mlflow.log_metric('mape', model_score)

          # Manually save the model as a pickle file and log it as an artifact
          with tempfile.TemporaryDirectory() as tmpdir:
              model_path = os.path.join(tmpdir, "model.pkl")
              with open(model_path, "wb") as f:
                  cloudpickle.dump(sarimax_estimator.model_, f)
              # Log the model.pkl directly at the root of the artifacts
              mlflow.log_artifact(model_path)
```

```
    else:
        print(f"Model fitting failed for column {column} with order={order},␣
↪seasonal_order={seasonal}. Skipping logging of model artifact.")
```

```
2025/12/23 16:45:03 INFO mlflow.store.db.utils: Creating initial MLflow database
tables…
2025/12/23 16:45:03 INFO mlflow.store.db.utils: Updating database tables
2025/12/23 16:45:03 INFO alembic.runtime.migration: Context impl SQLiteImpl.
2025/12/23 16:45:03 INFO alembic.runtime.migration: Will assume non-
transactional DDL.
2025/12/23 16:45:03 INFO alembic.runtime.migration: Running upgrade  ->
451aebb31d03, add metric step
2025/12/23 16:45:03 INFO alembic.runtime.migration: Running upgrade 451aebb31d03
-> 90e64c465722, migrate user column to tags
2025/12/23 16:45:03 INFO alembic.runtime.migration: Running upgrade 90e64c465722
-> 181f10493468, allow nulls for metric values
2025/12/23 16:45:03 INFO alembic.runtime.migration: Running upgrade 181f10493468
-> df50e92ffc5e, Add Experiment Tags Table
2025/12/23 16:45:03 INFO alembic.runtime.migration: Running upgrade df50e92ffc5e
-> 7ac759974ad8, Update run tags with larger limit
2025/12/23 16:45:03 INFO alembic.runtime.migration: Running upgrade 7ac759974ad8
-> 89d4b8295536, create latest metrics table
2025/12/23 16:45:03 INFO alembic.runtime.migration: Running upgrade 89d4b8295536
-> 2b4d017a5e9b, add model registry tables to db
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 2b4d017a5e9b
-> cfd24bdc0731, Update run status constraint with killed
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade cfd24bdc0731
-> 0a8213491aaa, drop_duplicate_killed_constraint
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 0a8213491aaa
-> 728d730b5ebd, add registered model tags table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 728d730b5ebd
-> 27a6a02d2cf1, add model version tags table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 27a6a02d2cf1
-> 84291f40a231, add run_link to model_version
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 84291f40a231
-> a8c4a736bde6, allow nulls for run_id
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade a8c4a736bde6
-> 39d1c3be5f05, add_is_nan_constraint_for_metrics_tables_if_necessary
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 39d1c3be5f05
-> c48cb773bb87, reset_default_value_for_is_nan_in_metrics_table_for_mysql
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade c48cb773bb87
-> bd07f7e963c5, create index on run_uuid
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade bd07f7e963c5
-> 0c779009ac13, add deleted_time field to runs table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 0c779009ac13
-> cc1f77228345, change param value length to 500
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade cc1f77228345
```

```
-> 97727af70f4d, Add creation_time and last_update_time to experiments table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 97727af70f4d
-> 3500859a5d39, Add Model Aliases table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 3500859a5d39
-> 7f2a7d5fae7d, add datasets inputs input_tags tables
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 7f2a7d5fae7d
-> 2d6e25af4d3e, increase max param val length from 500 to 8000
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 2d6e25af4d3e
-> acf3f17fdcc7, add storage location field to model versions
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade acf3f17fdcc7
-> 867495a8f9d4, add trace tables
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 867495a8f9d4
-> 5b0e9adcef9c, add cascade deletion to trace tables foreign keys
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 5b0e9adcef9c
-> 4465047574b1, increase max dataset schema size
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 4465047574b1
-> f5a4f2784254, increase run tag value limit to 8000
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade f5a4f2784254
-> 0584bdc529eb, add cascading deletion to datasets from experiments
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 0584bdc529eb
-> 400f98739977, add logged model tables
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 400f98739977
-> 6953534de441, add step to inputs table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 6953534de441
-> bda7b8c39065, increase_model_version_tag_value_limit
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade bda7b8c39065
-> cbc13b556ace, add V3 trace schema columns
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade cbc13b556ace
-> 770bee3ae1dd, add assessments table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 770bee3ae1dd
-> a1b2c3d4e5f6, add spans table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade a1b2c3d4e5f6
-> de4033877273, create entity_associations table
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade de4033877273
-> 1a0cddfcaa16, Add webhooks and webhook_events tables
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 1a0cddfcaa16
-> 534353b11cbc, add scorer tables
2025/12/23 16:45:04 INFO alembic.runtime.migration: Running upgrade 534353b11cbc
-> 71994744cf8e, add evaluation datasets
2025/12/23 16:45:05 INFO alembic.runtime.migration: Running upgrade 71994744cf8e
-> 3da73c924c2f, add outputs to dataset record
2025/12/23 16:45:05 INFO alembic.runtime.migration: Running upgrade 3da73c924c2f
-> bf29a5ff90ea, add jobs table
2025/12/23 16:45:05 INFO alembic.runtime.migration: Running upgrade bf29a5ff90ea
-> 1bd49d398cd23, add secrets tables
2025/12/23 16:45:05 INFO alembic.runtime.migration: Context impl SQLiteImpl.
2025/12/23 16:45:05 INFO alembic.runtime.migration: Will assume non-
transactional DDL.
```

```
2025/12/23 16:45:05 INFO mlflow.tracking.fluent: Experiment with name 'Sore ID
Sales Forecasting-1.1.0' does not exist. Creating a new experiment.
```

[59]:
```
# Model training for order forecasting with MLFLow
import cloudpickle
import tempfile
import os

mlflow.set_experiment("Sore ID Order Forecasting-1.1.0")
for column in train_order.columns[0:1]:
  splitter = TimeSeriesSplitter()
  X_train, X_test = splitter.fit_transform(train_order[column])
  X_train_exog, X_test_exog = splitter.fit_transform(exog_train_holiday)
  for order, seasonal in param_grid:
    with mlflow.start_run():
      sarimax_estimator = SARIMAXEstimator(order=order, seasonal_order=seasonal)
      sarimax_estimator.fit(X=X_train, exog=X_train_exog)

      if sarimax_estimator.model_ is not None:
          model_score = sarimax_estimator.score(X=X_test, exog=X_test_exog)
          mlflow.set_tag('data', column)
          mlflow.log_params({'order': order, 'seasonal_order': seasonal})
          mlflow.log_metric('mape', model_score)

          # Manually save the model as a pickle file and log it as an artifact
          with tempfile.TemporaryDirectory() as tmpdir:
              model_path = os.path.join(tmpdir, "model.pkl")
              with open(model_path, "wb") as f:
                  cloudpickle.dump(sarimax_estimator.model_, f)
              # Log the model.pkl directly at the root of the artifacts
              mlflow.log_artifact(model_path)

      else:
          print(f"Model fitting failed for column {column} with order={order},␣
↪seasonal_order={seasonal}. Skipping logging of model artifact.")
```

```
2025/12/23 16:45:06 INFO mlflow.tracking.fluent: Experiment with name 'Sore ID
Order Forecasting-1.1.0' does not exist. Creating a new experiment.
```

4 D. Forecasting Example

[60]:
```
import mlflow
import pandas as pd
import shutil
import os
import cloudpickle # Import cloudpickle for loading
```

```python
# Create 'models' directory if it's not exist
if not os.path.exists('models'):
    os.makedirs('models')

# Get the MLflow client
client = mlflow.tracking.MlflowClient()

# Get the experiment by name for Order Forecasting
order_experiment = client.get_experiment_by_name("Sore ID Order Forecasting-1.1.0")

if order_experiment is None:
    print("MLflow experiment 'Sore ID Order Forecasting-1.1.0' not found. Please ensure the model training step was executed.")
else:
    order_exp_id = order_experiment.experiment_id

    # Search for runs within this experiment, ordered by MAPE ascending
    # Assuming 'Order' is the column name for the overall order forecasting
    runs = client.search_runs(
        experiment_ids=[order_exp_id],
        filter_string="tags.data = 'Order'", # Filter for the specific 'Order' data tag
        order_by=["metrics.mape ASC"], # Order by MAPE to find the best model
        max_results=1 # We only need the best one
    )

    if runs:
        best_run = runs[0]
        run_id = best_run.info.run_id
        data_tag = best_run.data.tags.get('data') # This should be 'Order'

        print(f"Found best run for '{data_tag}' forecasting: Run ID {run_id}, MAPE {best_run.data.metrics.get('mape')}")

        # Define a temporary path to download artifacts for this run
        temp_download_dir = os.path.join('mlflow_temp_artifacts', run_id)
        os.makedirs(temp_download_dir, exist_ok=True)

        # Download the 'model.pkl' artifact (logged directly at root)
        artifact_uri_to_download = client.get_run(run_id).info.artifact_uri + '/model.pkl'
        mlflow.artifacts.download_artifacts(artifact_uri=artifact_uri_to_download, dst_path=temp_download_dir)
```

```
        # The actual model file is now at temp_download_dir/model.pkl
        source_model_path = os.path.join(temp_download_dir, 'model.pkl')
        destination_filename = os.path.join('models', f'{data_tag.
↪lower()}_order.pkl')

        # Move the model.pkl to its final destination
        shutil.move(source_model_path, destination_filename)
        print(f"Model for '{data_tag}' saved to {destination_filename}")

        # Clean up the temporary download directory
        shutil.rmtree(temp_download_dir)

    else:
        print("No runs found for 'Order' in 'Sore ID Order Forecasting-1.1.0'␣
↪experiment. Cannot save model.")
```

```
Found best run for 'Order' forecasting: Run ID 73d81e6dbd01429aa5b060f8e6de65f2,
MAPE 0.15094690023597634

Downloading artifacts:   0%|          | 0/1 [00:00<?, ?it/s]

Model for 'Order' saved to models/order_order.pkl
```

[61]:
```
import mlflow
import pandas as pd
import shutil
import os
import cloudpickle

# Get the MLflow client
client = mlflow.tracking.MlflowClient()

# Get the experiment by name for Sales Forecasting
sales_experiment = client.get_experiment_by_name("Sore ID Sales Forecasting-1.1.
↪0")

if sales_experiment is None:
    print("MLflow experiment 'Sore ID Sales Forecasting-1.1.0' not found.␣
↪Please ensure the model training step was executed.")
else:
    sales_exp_id = sales_experiment.experiment_id

    # Search for runs within this experiment, ordered by MAPE ascending
    # Assuming 'Sales' is the column name for the overall sales forecasting
    runs = client.search_runs(
        experiment_ids=[sales_exp_id],
        filter_string="tags.data = 'Sales'", # Filter for the specific 'Sales'␣
↪data tag
```

```
        order_by=["metrics.mape ASC"], # Order by MAPE to find the best model
        max_results=1 # We only need the best one
    )

    if runs:
        best_run = runs[0]
        run_id = best_run.info.run_id
        data_tag = best_run.data.tags.get('data') # This should be 'Sales'

        print(f"Found best run for '{data_tag}' forecasting: Run ID {run_id},␣
↪MAPE {best_run.data.metrics.get('mape')}")

        # Define a temporary path to download artifacts for this run
        temp_download_dir = os.path.join('mlflow_temp_artifacts', run_id)
        os.makedirs(temp_download_dir, exist_ok=True)

        # Download the 'model.pkl' artifact (logged directly at root)
        artifact_uri_to_download = client.get_run(run_id).info.artifact_uri + '/
↪model.pkl'
        mlflow.artifacts.
↪download_artifacts(artifact_uri=artifact_uri_to_download,␣
↪dst_path=temp_download_dir)

        # The actual model file is now at temp_download_dir/model.pkl
        source_model_path = os.path.join(temp_download_dir, 'model.pkl')
        destination_filename = os.path.join('models', f'{data_tag.
↪lower()}_sales.pkl')

        # Move the model.pkl to its final destination
        shutil.move(source_model_path, destination_filename)
        print(f"Model for '{data_tag}' saved to {destination_filename}")

        # Clean up the temporary download directory
        shutil.rmtree(temp_download_dir)

    else:
        print("No runs found for 'Sales' in 'Sore ID Sales Forecasting-1.1.0'␣
↪experiment. Cannot save model.")
```

```
Found best run for 'Sales' forecasting: Run ID 4c5193f3575947ccb13eb8f9aef568a9,
MAPE 0.19729424250763192

Downloading artifacts:   0%|          | 0/1 [00:00<?, ?it/s]

Model for 'Sales' saved to models/sales_sales.pkl
```

[62]:
```
import cloudpickle as pickle
with open('models/sales_sales.pkl', 'rb') as file:
```

```
    model = pickle.load(file)
model.summary()
```

[62]:

Dep. Variable:	Sales	**No. Observations:**	412
Model:	SARIMAX(1, 1, 1)x(1, 0, 1, 7)	**Log Likelihood**	-6643.432
Date:	Tue, 23 Dec 2025	**AIC**	13298.864
Time:	16:45:06	**BIC**	13322.975
Sample:	01-01-2018 - 02-16-2019	**HQIC**	13308.402
Covariance Type:	opg		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Holiday	-2.225e+06	1.32e-09	-1.68e+15	0.000	-2.23e+06	-2.23e+06
ar.L1	0.5814	0.042	13.805	0.000	0.499	0.664
ma.L1	-0.9989	0.021	-46.864	0.000	-1.041	-0.957
ar.S.L7	0.9963	0.013	74.994	0.000	0.970	1.022
ma.S.L7	-0.9707	0.056	-17.266	0.000	-1.081	-0.860
sigma2	8.338e+12	3.67e-15	2.27e+27	0.000	8.34e+12	8.34e+12

Ljung-Box (L1) (Q):	0.55	**Jarque-Bera (JB):**	507.31
Prob(Q):	0.46	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.85	**Skew:**	-0.65
Prob(H) (two-sided):	0.34	**Kurtosis:**	8.29

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 5.91e+41. Standard errors may be unstable.

[63]:

```
# Forecasting values
# Define the number of days to forecast
n = 50

# Re-run splitter to get consistent train/test sets for 'Sales'
# This ensures that X_train_sales, X_test_sales, and X_test_exog_sales align␣
↪with how the model was trained and evaluated.
splitter = TimeSeriesSplitter()
X_train_sales, X_test_sales = splitter.fit_transform(overall_sales['Sales'])
_, X_test_exog_sales = splitter.fit_transform(exog_train_holiday) # Only need␣
↪the test part of exog_train_holiday

# Prepare data for plotting
df_plot = X_train_sales
test_plot = X_test_sales.iloc[:n] # Take the first 'n' days of the actual test␣
↪data
exog_for_pred = X_test_exog_sales.iloc[:n] # Exogenous variables for the␣
↪prediction period
```

```
# Generate predictions using the loaded model
test_plot['pred'] = model.forecast(steps=n, exog=exog_for_pred)

# Create and display the plot
fig = go.Figure()
fig.add_trace(go.Scatter(x=df_plot.index, y=df_plot, mode='lines', name='Train␣
 ↪values', line=dict(color='green')))
fig.add_trace(go.Scatter(x=test_plot.index, y=test_plot, mode='lines',␣
 ↪name='Test values', line=dict(color='orange')))
fig.add_trace(go.Scatter(x=test_plot.index, y=test_plot['pred'], mode='lines',␣
 ↪name='Forecasting', line=dict(color='red')))
fig.update_layout(title_text='Forecasting of overall Sales', # Updated title␣
 ↪for clarity
                  title_x=0.5, title_y=0.85,
                  legend_x=0)
fig.show()
```

[64]:
```
import cloudpickle as pickle
with open('models/sales_sales.pkl', 'rb') as file:
    model = pickle.load(file)
model.summary()
```

[64]:

Dep. Variable:	Sales	**No. Observations:**	412
Model:	SARIMAX(1, 1, 1)x(1, 0, 1, 7)	**Log Likelihood**	-6643.432
Date:	Tue, 23 Dec 2025	**AIC**	13298.864
Time:	16:45:07	**BIC**	13322.975
Sample:	01-01-2018 - 02-16-2019	**HQIC**	13308.402
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Holiday	-2.225e+06	1.32e-09	-1.68e+15	0.000	-2.23e+06	-2.23e+06
ar.L1	0.5814	0.042	13.805	0.000	0.499	0.664
ma.L1	-0.9989	0.021	-46.864	0.000	-1.041	-0.957
ar.S.L7	0.9963	0.013	74.994	0.000	0.970	1.022
ma.S.L7	-0.9707	0.056	-17.266	0.000	-1.081	-0.860
sigma2	8.338e+12	3.67e-15	2.27e+27	0.000	8.34e+12	8.34e+12

Ljung-Box (L1) (Q):	0.55	**Jarque-Bera (JB):**	507.31
Prob(Q):	0.46	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.85	**Skew:**	-0.65
Prob(H) (two-sided):	0.34	**Kurtosis:**	8.29

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 5.91e+41. Standard errors may be unstable.

[65]:
```
sub_to_forecast = 'Sales' # Changed to 'Sales' to match the loaded overall␣
    ↪sales model
n = 50              # Number of days

# Create splitter instance
splitter = TimeSeriesSplitter()

# Split the overall sales data and the exogenous holiday data for the overall␣
    ↪sales model
X_train_data, X_test_data = splitter.
    ↪fit_transform(overall_sales[sub_to_forecast])
_, X_test_exog_data = splitter.fit_transform(exog_train_holiday)

# Use the 'n' variable to define the length of the test data for plotting and␣
    ↪forecasting
test_plot_data = X_test_data.iloc[:n] # Actual test values for the first 'n'␣
    ↪days
exog_for_forecast = X_test_exog_data.iloc[:n] # Exogenous variables for the␣
    ↪forecast period

# The model 'model' is already loaded from 'models/sales_sales.pkl' in cell␣
    ↪PYOujvvmwZaw

# Generate predictions using the loaded model
test_plot_data['pred'] = model.forecast(steps=n, exog=exog_for_forecast)

fig = go.Figure()
fig.add_trace(go.Scatter(x=X_train_data.index, y=X_train_data, mode='lines',␣
    ↪name='Train values', line=dict(color='green')))
fig.add_trace(go.Scatter(x=test_plot_data.index, y=test_plot_data,␣
    ↪mode='lines', name='Test values', line=dict(color='orange')))
fig.add_trace(go.Scatter(x=test_plot_data.index, y=test_plot_data['pred'],␣
    ↪mode='lines', name='Forecasting', line=dict(color='red')))
fig.update_layout(title_text=f'Forecasting of {sub_to_forecast}',
                  title_x=0.5,title_y=0.85,
                  legend_x=0)
fig.show()
```

[66]:
```
import cloudpickle as pickle
with open('models/order_order.pkl', 'rb') as file:
    model = pickle.load(file)
model.summary()
```

[66]:

Dep. Variable:	Order	**No. Observations:**	412
Model:	SARIMAX(1, 1, 1)x(1, 0, 1, 7)	**Log Likelihood**	-3938.587
Date:	Tue, 23 Dec 2025	**AIC**	7889.174
Time:	16:45:07	**BIC**	7913.286
Sample:	01-01-2018	**HQIC**	7898.713
	- 02-16-2019		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Holiday	-3412.0755	640.458	-5.328	0.000	-4667.350	-2156.801
ar.L1	0.5009	0.041	12.166	0.000	0.420	0.582
ma.L1	-0.9982	0.011	-91.624	0.000	-1.020	-0.977
ar.S.L7	0.9961	0.013	79.606	0.000	0.972	1.021
ma.S.L7	-0.9644	0.059	-16.368	0.000	-1.080	-0.849
sigma2	1.711e+07	0.094	1.83e+08	0.000	1.71e+07	1.71e+07

Ljung-Box (L1) (Q):	0.25	**Jarque-Bera (JB):**	908.66
Prob(Q):	0.62	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.77	**Skew:**	-0.99
Prob(H) (two-sided):	0.12	**Kurtosis:**	10.01

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 8.16e+24. Standard errors may be unstable.

[67]:
```
# Forecasting values
sub_to_forecast = 'Order' # Changed to 'Order' to match the loaded overall␣
↪order model
n = 50                # Number of days

# Create splitter instance
splitter = TimeSeriesSplitter()

# Split the overall order data and the exogenous holiday data for the overall␣
↪order model
X_train_data_order, X_test_data_order = splitter.
↪fit_transform(overall_order[sub_to_forecast])
_, X_test_exog_data_order = splitter.fit_transform(exog_train_holiday) # Only␣
↪need the test part of exog_train_holiday

# Use the 'n' variable to define the length of the test data for plotting and␣
↪forecasting
test_plot_data_order = X_test_data_order.iloc[:n] # Actual test values for the␣
↪first 'n' days
exog_for_forecast_order = X_test_exog_data_order.iloc[:n] # Exogenous variables␣
↪for the forecast period
```

```python
# The model 'model' is already loaded from 'models/order_order.pkl' in cell␣
↪hnZfqILrwZaw

# Generate predictions using the loaded model
test_plot_data_order['pred'] = model.forecast(steps=n,␣
↪exog=exog_for_forecast_order)

fig = go.Figure()
fig.add_trace(go.Scatter(x=X_train_data_order.index, y=X_train_data_order,␣
↪mode='lines', name='Train values', line=dict(color='green')))
fig.add_trace(go.Scatter(x=test_plot_data_order.index, y=test_plot_data_order,␣
↪mode='lines', name='Test values', line=dict(color='orange')))
fig.add_trace(go.Scatter(x=test_plot_data_order.index,␣
↪y=test_plot_data_order['pred'], mode='lines', name='Forecasting',␣
↪line=dict(color='red')))
fig.update_layout(title_text=f'Forecasting of overall {sub_to_forecast}', #␣
↪Updated title for clarity
                  title_x=0.5,title_y=0.85,
                  legend_x=0)
fig.show()
```

4.1 Observations

- Sales and Orders move together, when orders increase, sales also increase.
- Discounts lead to higher customer activity compared to non-discount days.
- Sales and orders drop noticeably on holidays.
- Performance varies significantly across Store Type, Location Type, and Region.
- Clear seasonal patterns exist across months and days of the week.
- Time-series statistical tests show the data is suitable for forecasting.

4.2 Key Insights

- **Discounts are effective** : They significantly boost both sales and order volume.
- **Holidays negatively impact business** : Lower sales and fewer orders are recorded during holidays.
- **Store characteristics matter** : Store Type, Region, and Location strongly influence performance.
- **Strong seasonality exists** : Sales follow predictable monthly and weekly patterns.
- **Reliable forecasting potential** : Data supports accurate time-series forecasting.

[69]:

```python
```