TSF Final Assignment

Murat Tirkeshov

Kaggle competition

In [62]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.arima.model import ARIMA
from pmdarima import auto_arima
from sklearn.metrics import mean_absolute_error,mean_squared_error
from statsmodels.tsa.arima_process import ArmaProcess
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.stats.diagnostic import acorr_ljungbox, acorr_breu
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.api import VAR
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import arma_order_select_ic
from arch import arch_model
from statsmodels.tsa.stattools import kpss
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.deterministic import CalendarFourier, Determin
from tbats import TBATS
from statsmodels.stats.diagnostic import acorr_ljungbox
from scipy.stats import norm
import pmdarima as pm
from prophet import Prophet
from itertools import product
from statsmodels.tsa.stattools import grangercausalitytests
import itertools
import plotly.express as px

from prophet import Prophet

import tensorflow as tf
from tensorflow.keras.layers import Dense, Input, GlobalMaxPooling1
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten,
from tensorflow.keras.models import Model
from sklearn.metrics import mean_absolute_percentage_error
from tensorflow.keras.callbacks import ModelCheckpoint
from sklearn.metrics import mean_absolute_percentage_error
import seaborn as sns
```

```
from statsmodels.stats.diagnostic import acorr_ljungbox
import scipy.stats as stats
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
```

Data

Merging given data sets

In [3]:
```
df = pd.read_csv('store-sales-time-series-forecasting/train.csv', p
submission = pd.read_csv('store-sales-time-series-forecasting/test.
stores = pd.read_csv('store-sales-time-series-forecasting/stores.cs
oil = pd.read_csv('store-sales-time-series-forecasting/oil.csv', pa
transactions = pd.read_csv('store-sales-time-series-forecasting/tra
holidays = pd.read_csv('store-sales-time-series-forecasting/holiday
```

In [4]:
```
df.head()
```

Out[4]:

	id	date	store_nbr	family	sales	onpromotion
0	0	2013-01-01	1	AUTOMOTIVE	0.0	0
1	1	2013-01-01	1	BABY CARE	0.0	0
2	2	2013-01-01	1	BEAUTY	0.0	0
3	3	2013-01-01	1	BEVERAGES	0.0	0
4	4	2013-01-01	1	BOOKS	0.0	0

In [5]:
```
df = df.merge(stores, on='store_nbr', how='left')
submission = submission.merge(stores, on='store_nbr', how='left')
df.head()
```

Out[5]:

	id	date	store_nbr	family	sales	onpromotion	city	state
0	0	2013-01-01	1	AUTOMOTIVE	0.0	0	Quito	Pichincha
1	1	2013-01-01	1	BABY CARE	0.0	0	Quito	Pichincha
2	2	2013-01-01	1	BEAUTY	0.0	0	Quito	Pichincha
3	3	2013-01-01	1	BEVERAGES	0.0	0	Quito	Pichincha
4	4	2013-01-01	1	BOOKS	0.0	0	Quito	Pichincha

In [6]:
```
df = df.merge(transactions, on=['date', 'store_nbr'], how='left')
submission = submission.merge(transactions, on=['date', 'store_nbr'
df.head()
```

Out[6]:

	id	date	store_nbr	family	sales	onpromotion	city	state
0	0	2013-01-01	1	AUTOMOTIVE	0.0	0	Quito	Pichincha
1	1	2013-01-01	1	BABY CARE	0.0	0	Quito	Pichincha
2	2	2013-01-01	1	BEAUTY	0.0	0	Quito	Pichincha
3	3	2013-01-01	1	BEVERAGES	0.0	0	Quito	Pichincha
4	4	2013-01-01	1	BOOKS	0.0	0	Quito	Pichincha

In [7]:

```
oil = oil.set_index('date').resample('D').mean().interpolate()
oil.reset_index(inplace=True)
df = df.merge(oil, on='date', how='left')
submission = submission.merge(oil, on='date', how='left')
df.head()
```

Out[7]:

	id	date	store_nbr	family	sales	onpromotion	city	state
0	0	2013-01-01	1	AUTOMOTIVE	0.0	0	Quito	Pichincha
1	1	2013-01-01	1	BABY CARE	0.0	0	Quito	Pichincha
2	2	2013-01-01	1	BEAUTY	0.0	0	Quito	Pichincha
3	3	2013-01-01	1	BEVERAGES	0.0	0	Quito	Pichincha
4	4	2013-01-01	1	BOOKS	0.0	0	Quito	Pichincha

In [8]:

```
def process_holidays(df):
    df = df.copy()
    df['is_holiday'] = 1
    df = df[['date', 'is_holiday', 'type', 'locale']]
    df = df.drop_duplicates('date')
    df = df.pivot_table(index='date',
                        values='is_holiday',
                        aggfunc='max').reset_index()
    return df

holiday_features = process_holidays(holidays)
df = df.merge(holiday_features, on='date', how='left')
submission = submission.merge(holiday_features, on='date', how='lef


df.head()
```

Out[8]:

	id	date	store_nbr	family	sales	onpromotion	city	state
0	0	2013-01-01	1	AUTOMOTIVE	0.0	0	Quito	Pichincha
1	1	2013-01-01	1	BABY CARE	0.0	0	Quito	Pichincha
2	2	2013-01-01	1	BEAUTY	0.0	0	Quito	Pichincha
3	3	2013-01-01	1	BEVERAGES	0.0	0	Quito	Pichincha
4	4	2013-01-01	1	BOOKS	0.0	0	Quito	Pichincha

Missing values

In [9]:

```
df['transactions'] = df['transactions'].fillna(df['transactions'].m
submission['transactions'] = submission['transactions'].fillna(subm
df['dcoilwtico'] = df['dcoilwtico'].fillna(df['dcoilwtico'].mean())
submission['dcoilwtico'] = submission['dcoilwtico'].fillna(submissi
```

Adding aditional variables

In [10]:

```
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
df['day_of_week'] = df['date'].dt.dayofweek
df['is_weekend'] = (df['day_of_week'] >= 5).astype(int)
df['is_weekend_and_holiday'] = ((df['is_weekend'] == 1) & (df['is_h
submission['day'] = submission['date'].dt.day
submission['month'] = submission['date'].dt.month
submission['year'] = submission['date'].dt.year
submission['day_of_week'] = submission['date'].dt.dayofweek
submission['is_weekend'] = (submission['day_of_week'] >= 5).astype(
submission['is_weekend_and_holiday'] = ((submission['is_weekend'] =
```

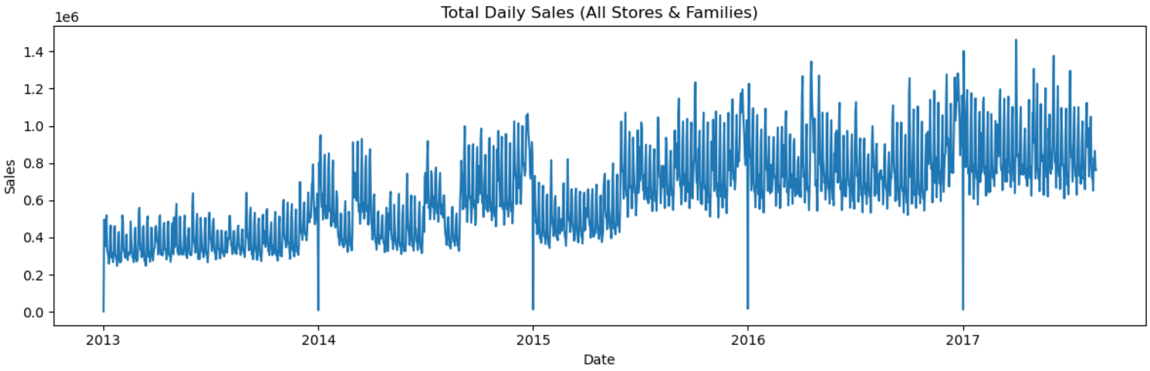
In [11]:

```
df['is_holiday'] = df['is_holiday'].fillna(0)
submission['is_holiday'] = submission['is_holiday'].fillna(0)
```

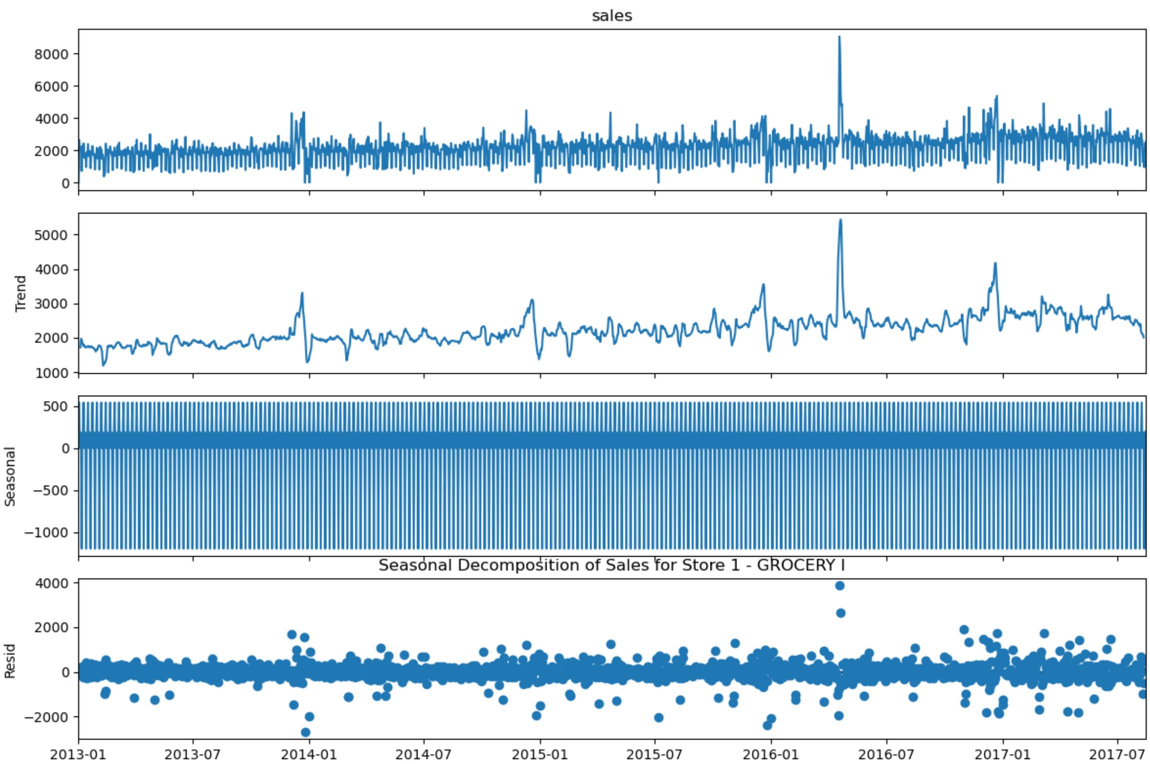
Data Visualization

In [12]:

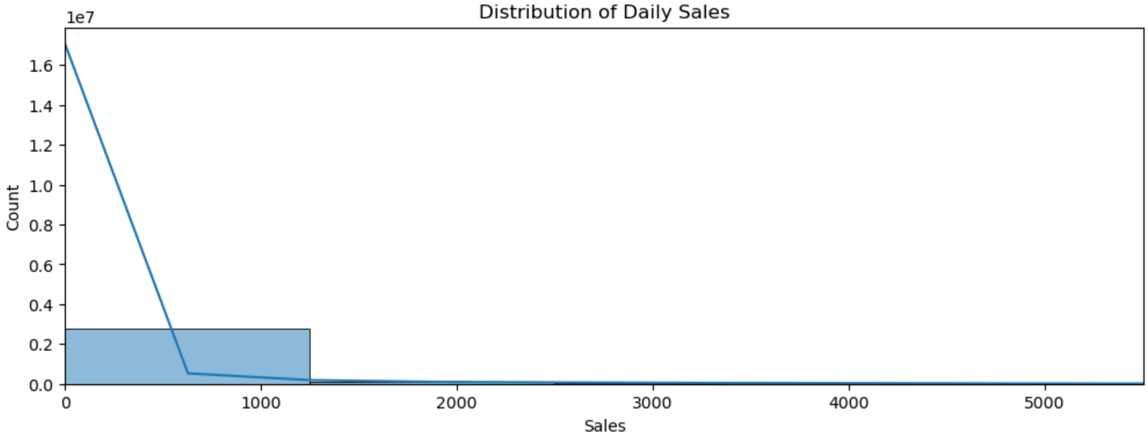
```
daily = df.groupby('date')['sales'].sum().reset_index()
plt.figure(figsize=(12,4))
plt.plot(daily['date'], daily['sales'])
plt.title('Total Daily Sales (All Stores & Families)')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.tight_layout()
plt.show()
```

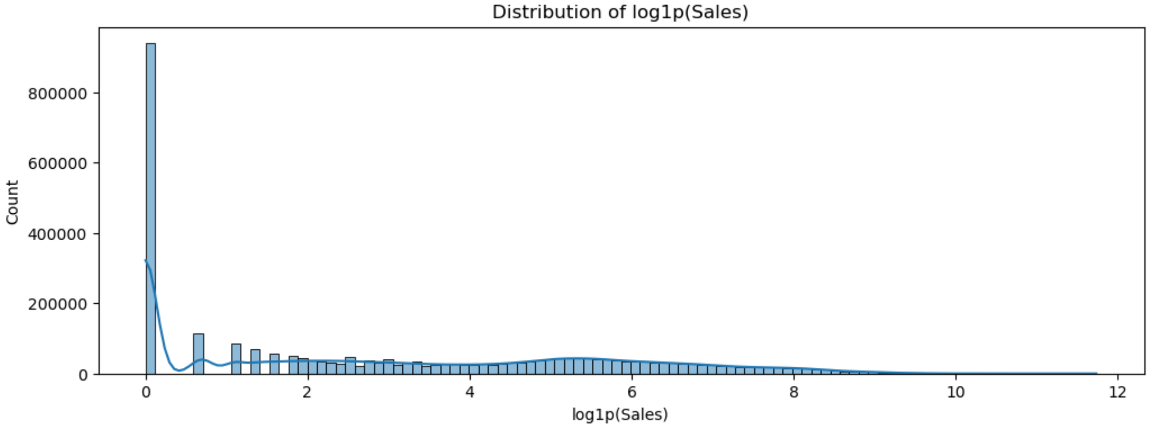


In [63]:
```
mask = (df['store_nbr']==1) & (df['family']=='GROCERY I')
ts = df[mask].set_index('date')['sales'].asfreq('D').fillna(0)
decomp = seasonal_decompose(ts, model='additive', period=7)
fig = decomp.plot()
fig.set_size_inches(12,8)
plt.tight_layout()
plt.title('Seasonal Decomposition of Sales for Store 1 - GROCERY I'
plt.show()
```

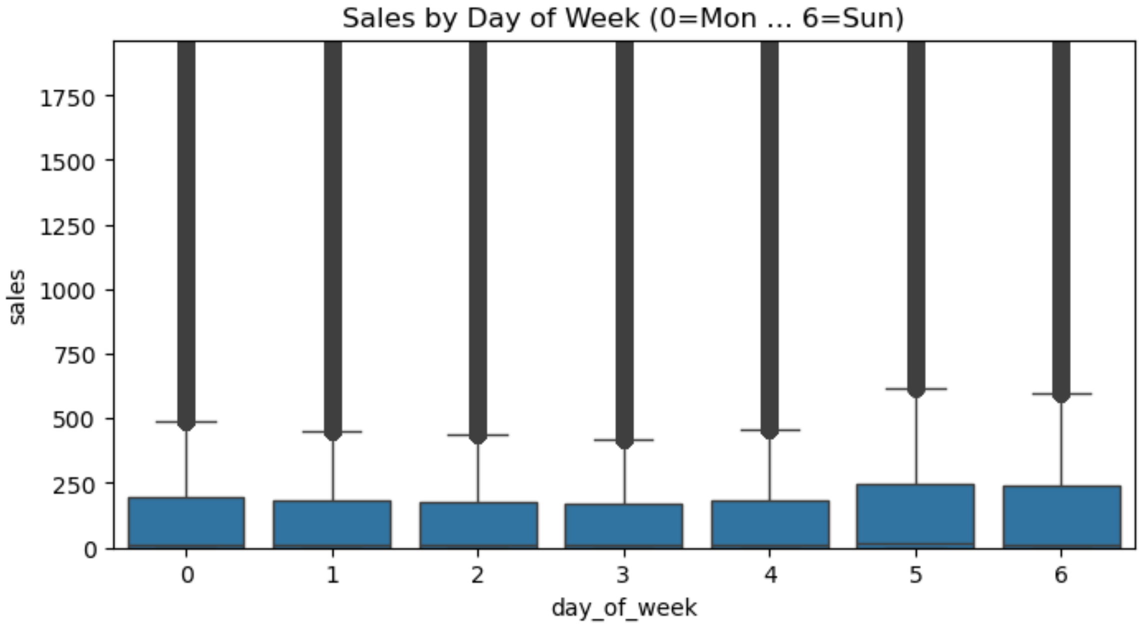


In [14]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['sales'], bins=100, kde=True)
plt.xlim(0, df['sales'].quantile(0.99))
plt.title('Distribution of Daily Sales')
plt.xlabel('Sales')
plt.show()

plt.figure(figsize=(12,4))
sns.histplot(np.log1p(df['sales']), bins=100, kde=True)
plt.title('Distribution of log1p(Sales)')
plt.xlabel('log1p(Sales)')
plt.show()
```



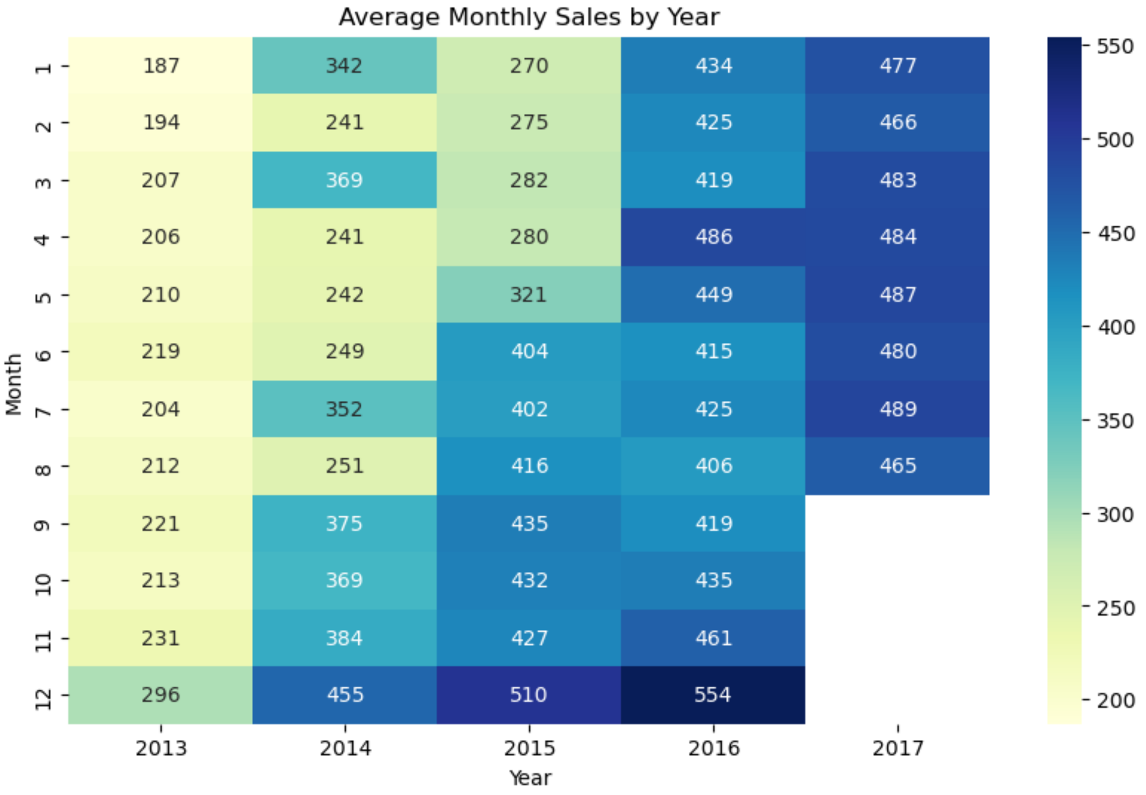


In [15]:
```
plt.figure(figsize=(8,4))
sns.boxplot(x='day_of_week', y='sales', data=df)
plt.ylim(0, df['sales'].quantile(0.95))
plt.title('Sales by Day of Week (0=Mon … 6=Sun)')
plt.show()
```

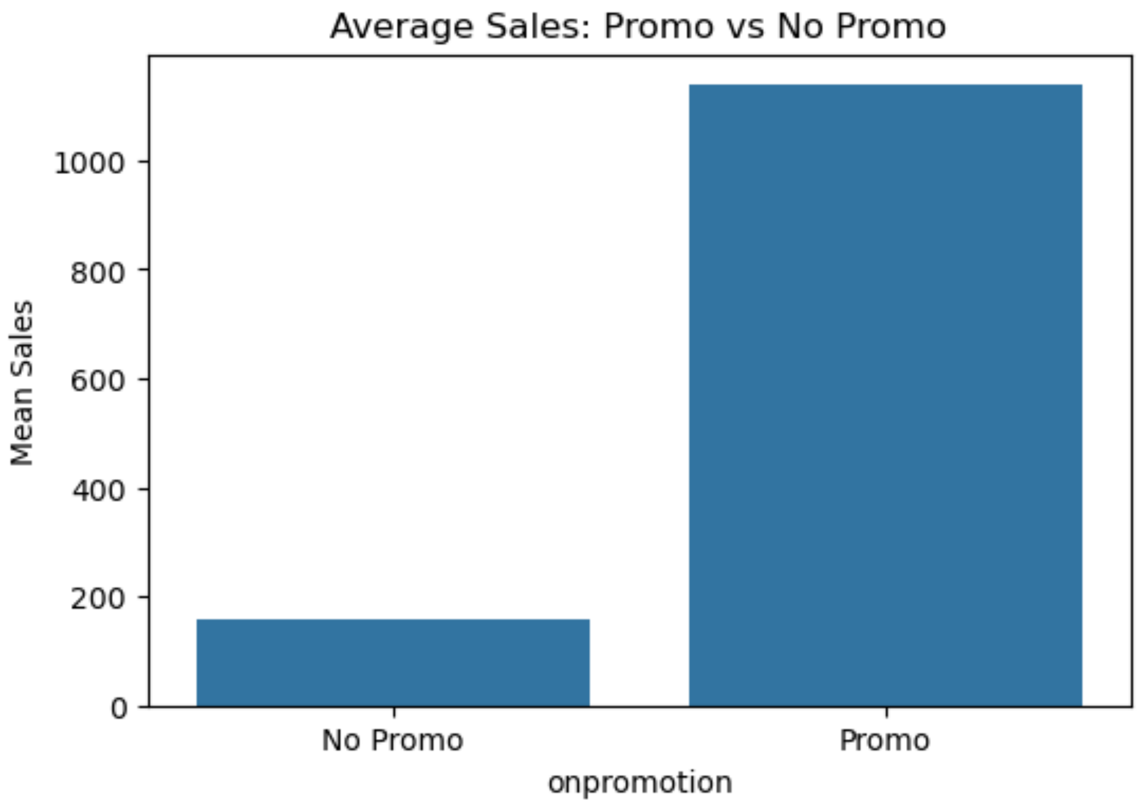


In [16]:
```
pivot = df.groupby(['year','month'])['sales'].mean().unstack(level=
plt.figure(figsize=(10,6))
sns.heatmap(pivot, annot=True, fmt=".0f", cmap='YlGnBu')
plt.title('Average Monthly Sales by Year')
plt.ylabel('Month')
plt.xlabel('Year')
```

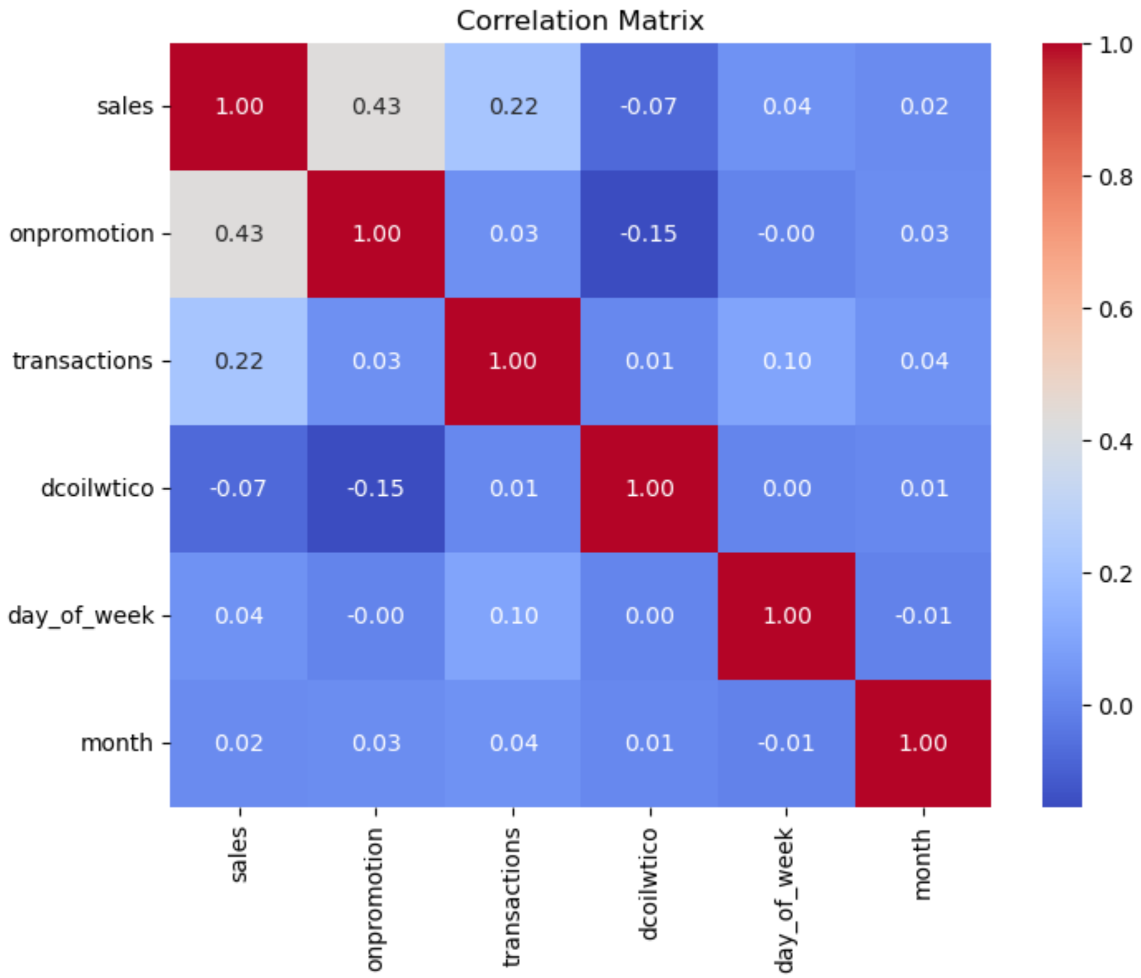
```
plt.show()
```



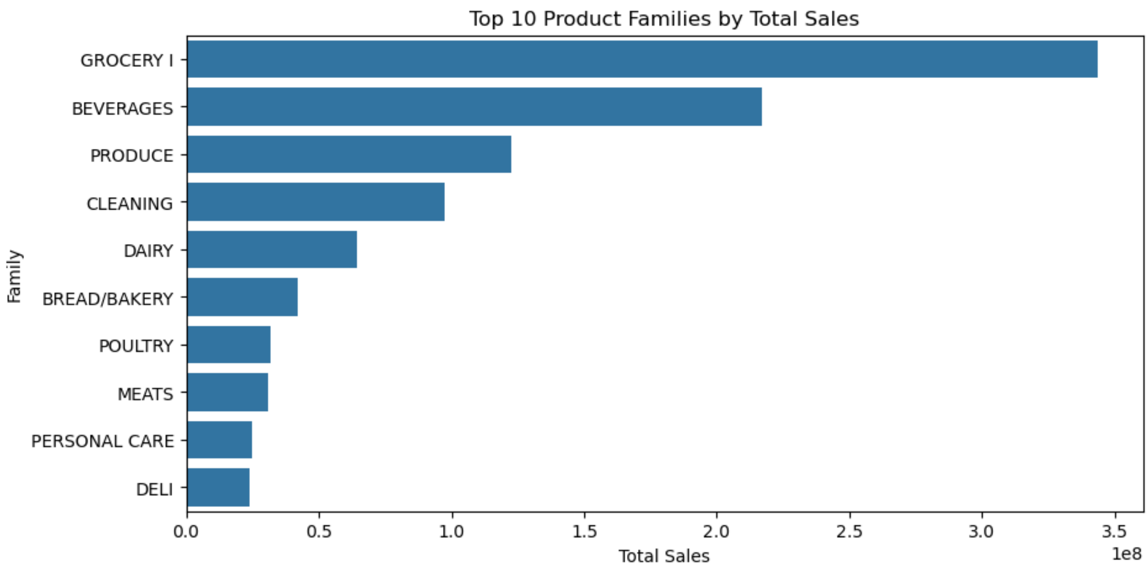
In [17]:
```
promo = df.copy()
promo['onpromotion'] = promo['onpromotion'] > 0
mean_sales = promo.groupby('onpromotion')['sales'].mean().reset_ind
plt.figure(figsize=(6,4))
sns.barplot(x='onpromotion', y='sales', data=mean_sales)
plt.xticks([0,1], ['No Promo','Promo'])
plt.title('Average Sales: Promo vs No Promo')
plt.ylabel('Mean Sales')
plt.show()
```



In [18]:
```
num_cols = ['sales','onpromotion','transactions','dcoilwtico','day_
corr = df[num_cols].corr()
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



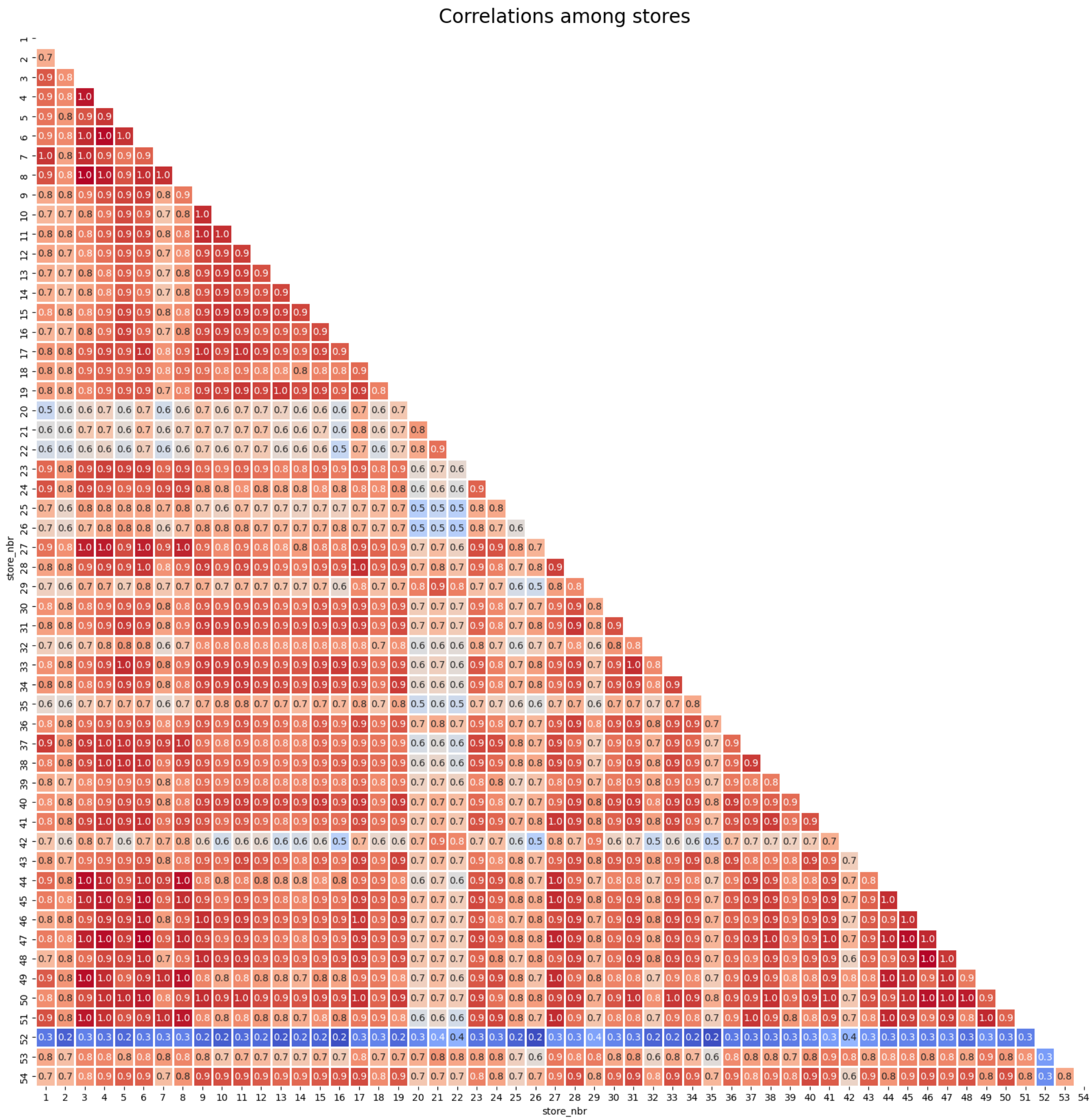
In [19]:
```
family_sales = df.groupby('family')['sales'].sum().sort_values(asce
plt.figure(figsize=(10,5))
sns.barplot(x=family_sales.values, y=family_sales.index)
plt.title('Top 10 Product Families by Total Sales')
plt.xlabel('Total Sales')
plt.ylabel('Family')
plt.show()
```



In [20]:
```
cluster_sales = df.groupby('cluster')['sales'].sum().sort_values(as
plt.figure(figsize=(8,4))
sns.barplot(x=cluster_sales.index, y=cluster_sales.values)
```

```
plt.title('Total Sales by Store Cluster')
plt.xlabel('Cluster')
plt.ylabel('Sales')
plt.show()
```

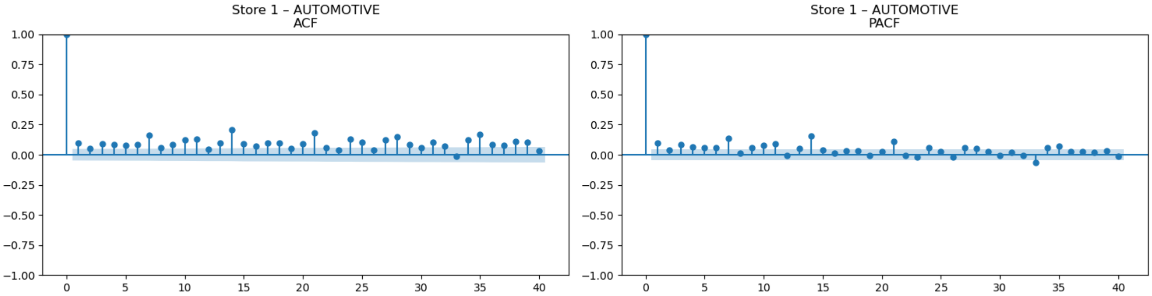
In [21]:
```
a = df[["store_nbr", "sales"]]
a["ind"] = 1
a["ind"] = a.groupby("store_nbr").ind.cumsum().values
a = pd.pivot(a, index = "ind", columns = "store_nbr", values = "sal
mask = np.triu(a.corr())
plt.figure(figsize=(20, 20))
sns.heatmap(a,
        annot=True,
        fmt='.1f',
        cmap='coolwarm',
        square=True,
        mask=mask,
        linewidths=1,
        cbar=False)
plt.title("Correlations among stores",fontsize = 20)
plt.show()
```



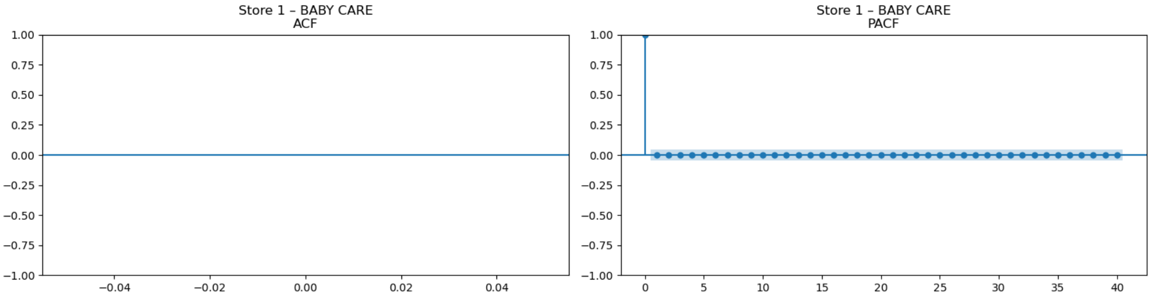
ACF and PACF for store 1

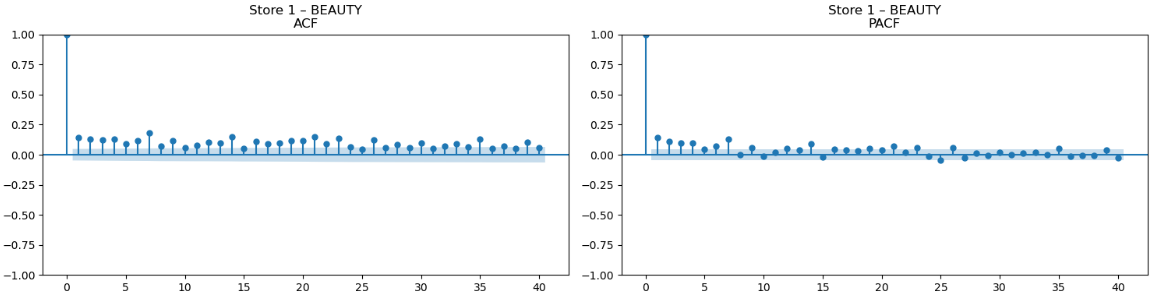
In [22]:
```
agg1 = (
    df[(df['store_nbr'] == 1) & df['sales'].notnull()]
    .groupby(['family', 'date'])['sales']
    .mean()
    .reset_index()
)

# 3. Loop over each family in store 1
for fam, grp in agg1.groupby('family'):
    ts = grp.set_index('date')['sales']

    try:
        fig, axes = plt.subplots(1, 2, figsize=(15, 4))
        sm.graphics.tsa.plot_acf(ts, lags=40, ax=axes[0])
        axes[0].set_title(f'Store 1 — {fam}\nACF')
        sm.graphics.tsa.plot_pacf(ts, lags=40, ax=axes[1])
        axes[1].set_title(f'Store 1 — {fam}\nPACF')
        plt.tight_layout()
        plt.show()
```

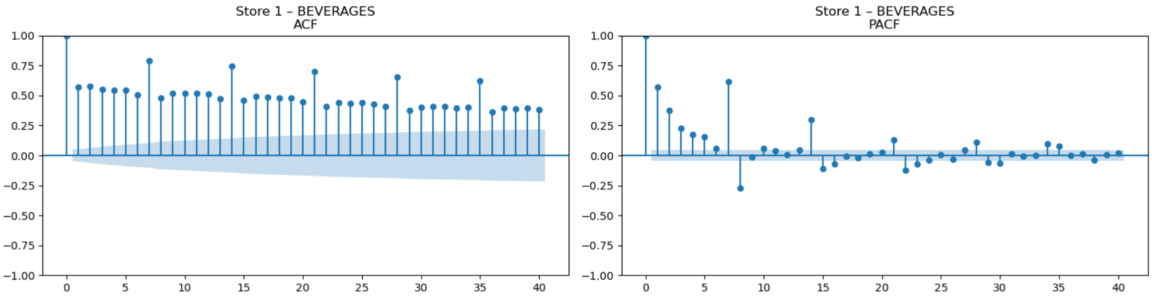
```
    except Exception as e:
        print(f"Could not plot family {fam}: {e}")
```

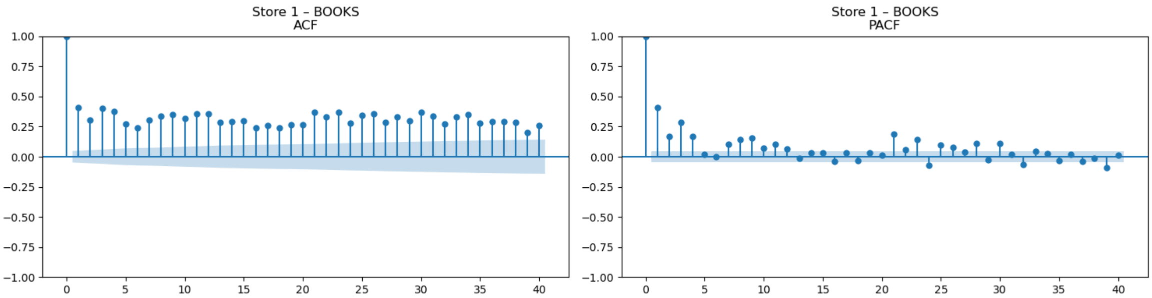


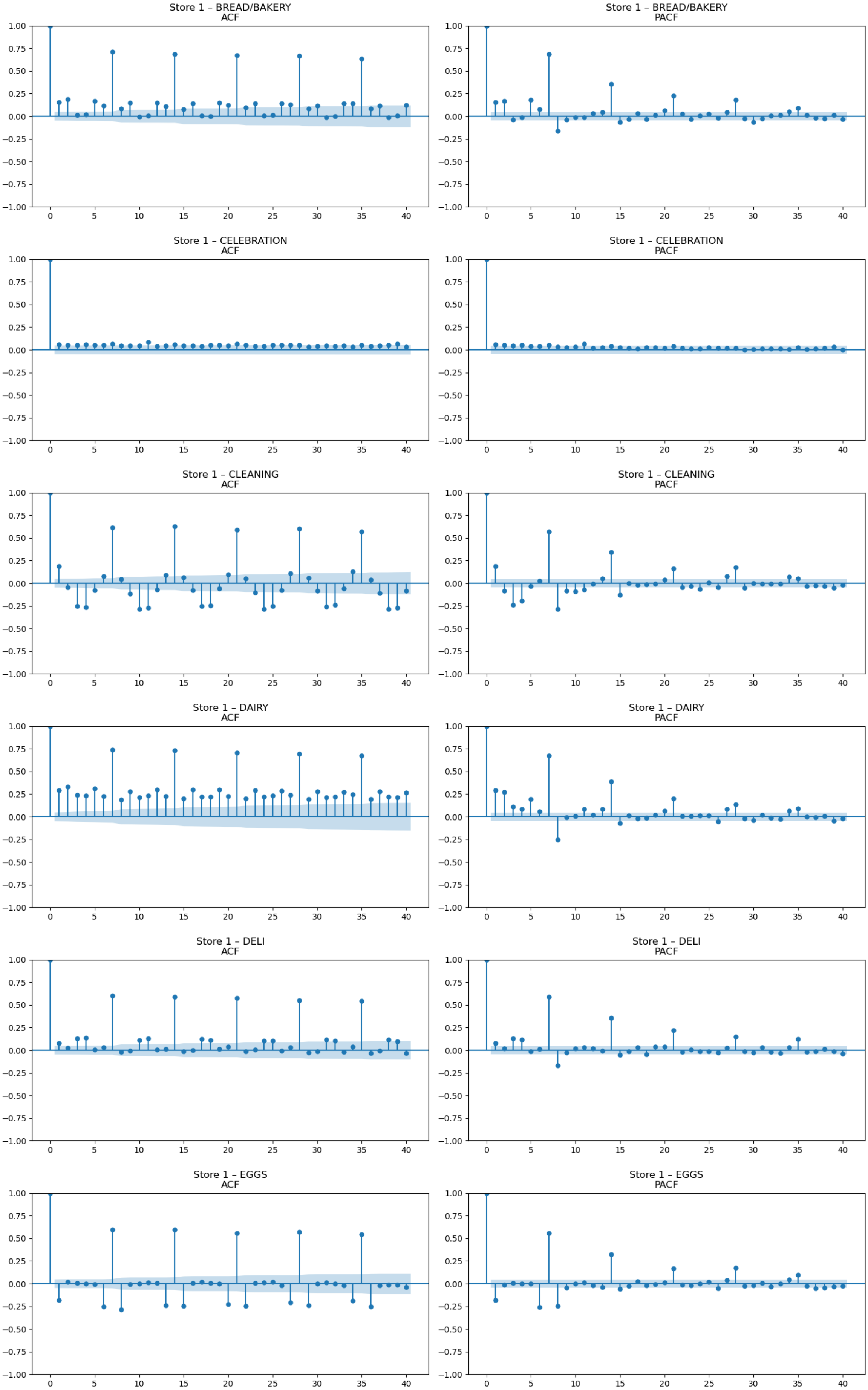

```
/opt/anaconda3/lib/python3.12/site-packages/statsmodels/regression/l
inear_model.py:1491: ValueWarning: Matrix is singular. Using pinv.
  warnings.warn("Matrix is singular. Using pinv.", ValueWarning)
```

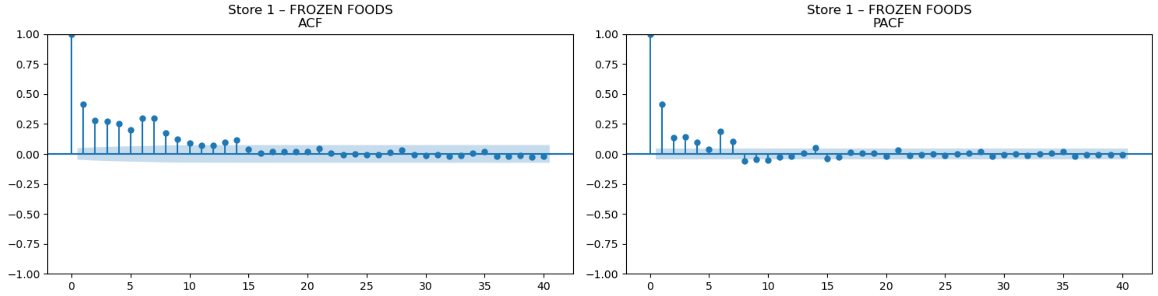


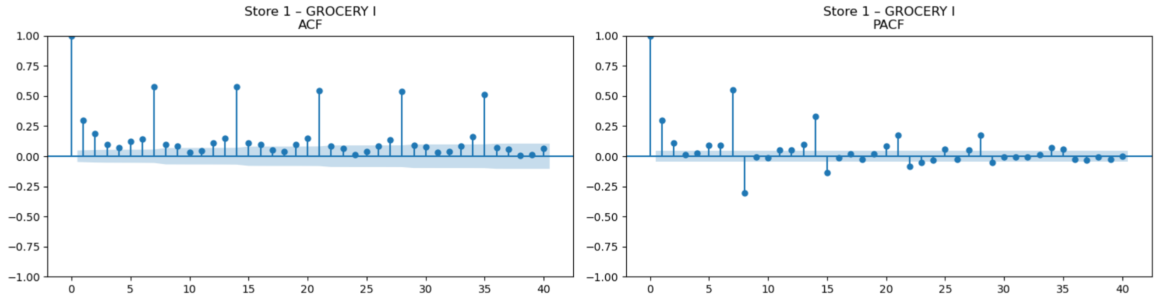


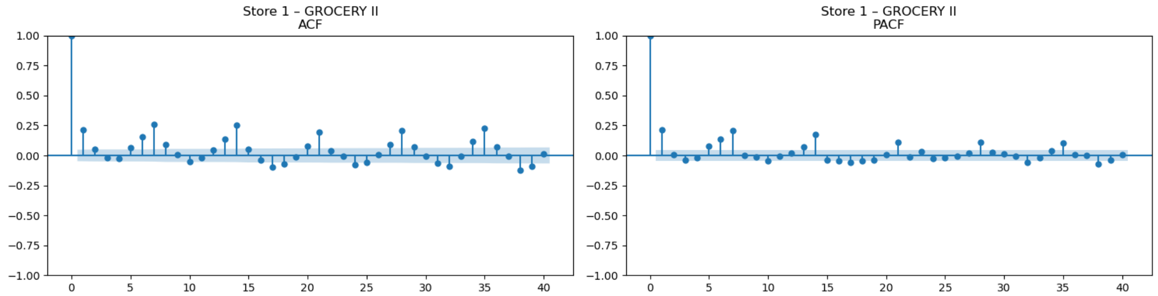


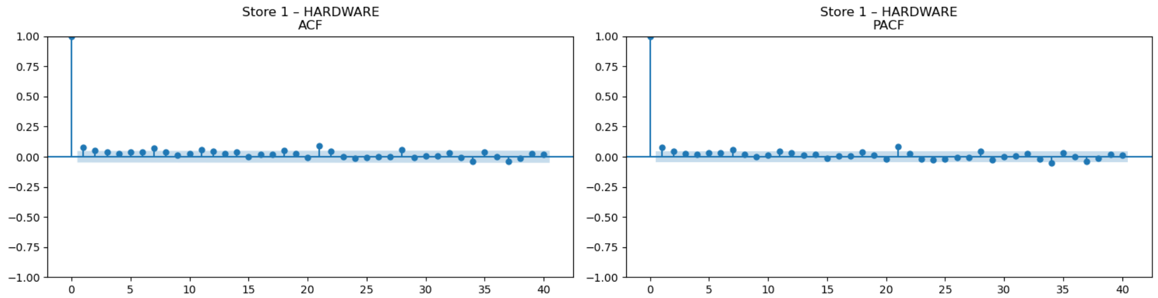


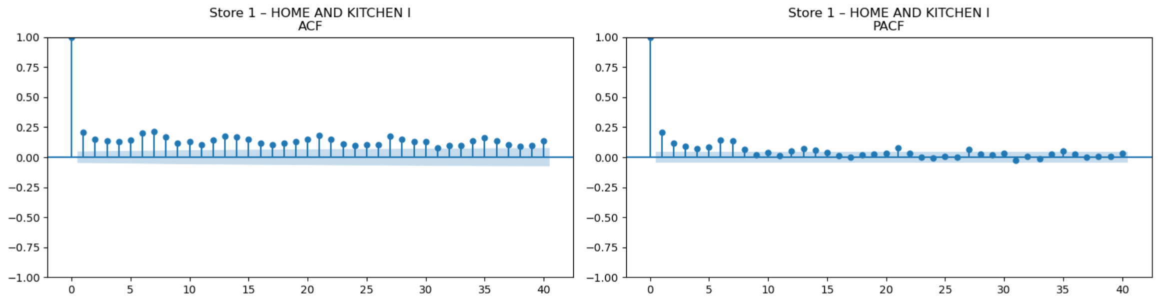

Store 1 – BREAD/BAKERY
ACF
Store 1 – BREAD/BAKERY
PACF
Store 1 – CELEBRATION
ACF
Store 1 – CELEBRATION
PACF
Store 1 – CLEANING
ACF
Store 1 – CLEANING
PACF
Store 1 – DAIRY
ACF
Store 1 – DAIRY
PACF
Store 1 – DELI
ACF
Store 1 – DELI
PACF
Store 1 – EGGS
ACF
Store 1 – EGGS
PACF

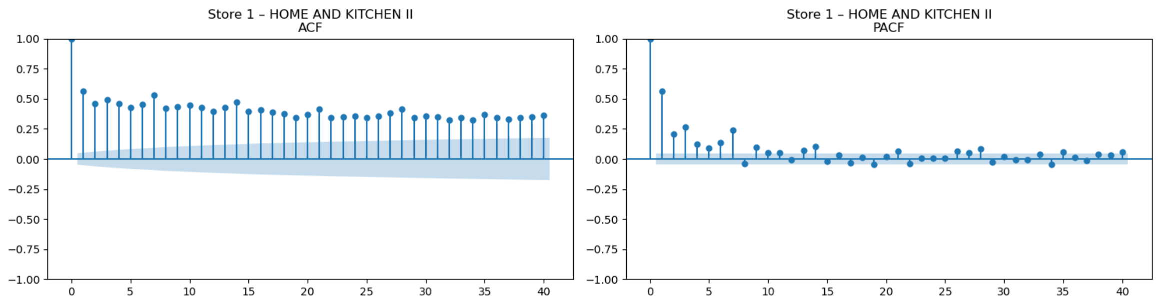

Store 1 – FROZEN FOODS
ACF
Store 1 – FROZEN FOODS
PACF

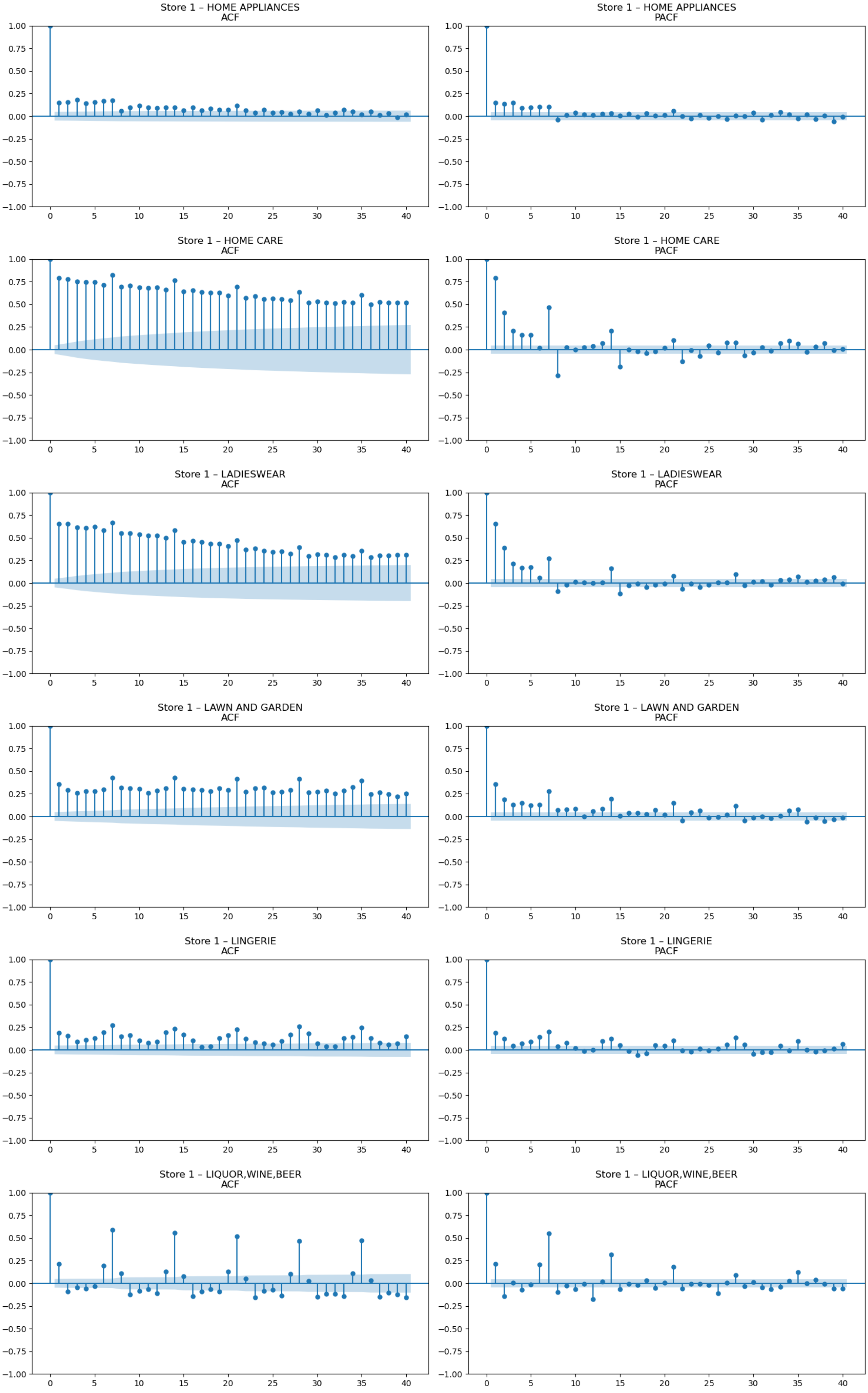

Store 1 – GROCERY I
ACF
Store 1 – GROCERY I
PACF

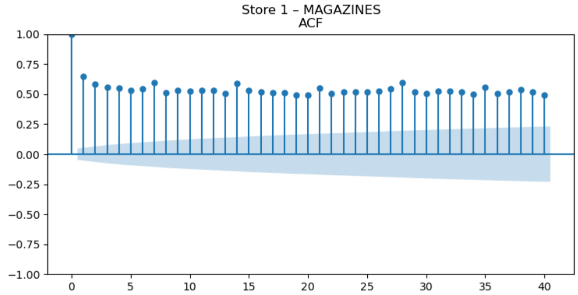

Store 1 – GROCERY II
ACF
Store 1 – GROCERY II
PACF

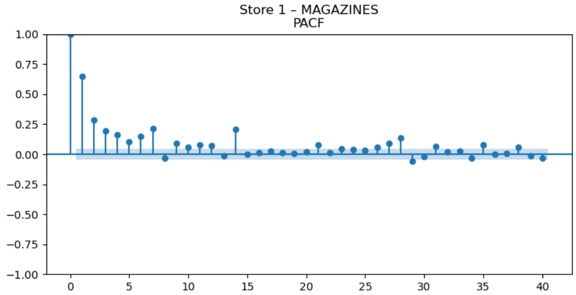

Store 1 – HARDWARE
ACF
Store 1 – HARDWARE
PACF

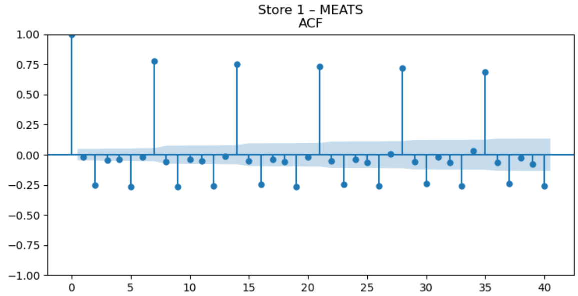

Store 1 – HOME AND KITCHEN I
ACF
Store 1 – HOME AND KITCHEN I
PACF

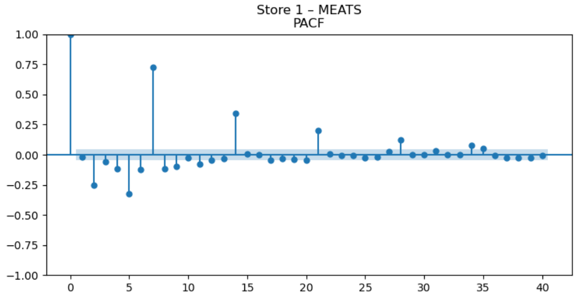

Store 1 – HOME AND KITCHEN II
ACF
Store 1 – HOME AND KITCHEN II
PACF

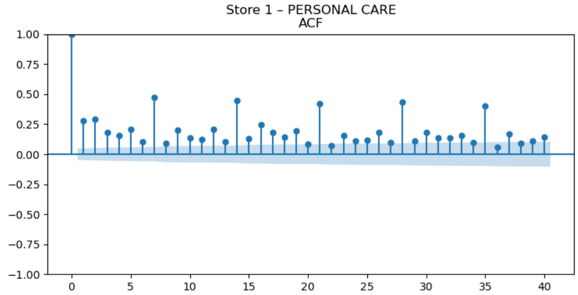

Store 1 – HOME APPLIANCES
ACF
Store 1 – HOME APPLIANCES
PACF
Store 1 – HOME CARE
ACF
Store 1 – HOME CARE
PACF
Store 1 – LADIESWEAR
ACF
Store 1 – LADIESWEAR
PACF
Store 1 – LAWN AND GARDEN
ACF
Store 1 – LAWN AND GARDEN
PACF
Store 1 – LINGERIE
ACF
Store 1 – LINGERIE
PACF
Store 1 – LIQUOR,WINE,BEER
ACF
Store 1 – LIQUOR,WINE,BEER
PACF

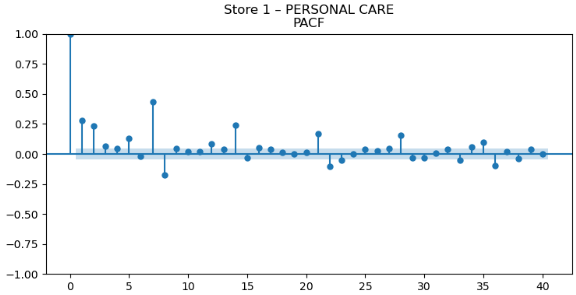

Store 1 – MAGAZINES
ACF

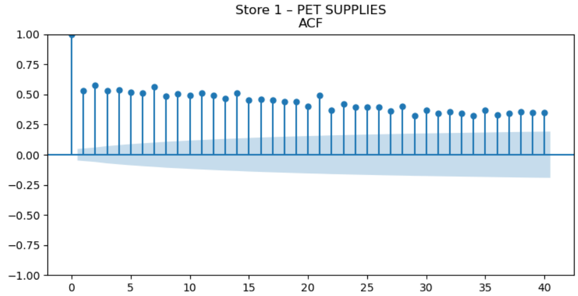

Store 1 – MAGAZINES
PACF

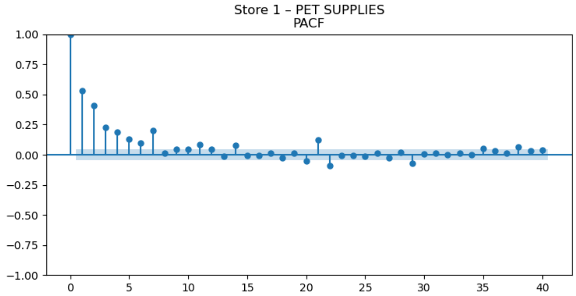

Store 1 – MEATS
ACF

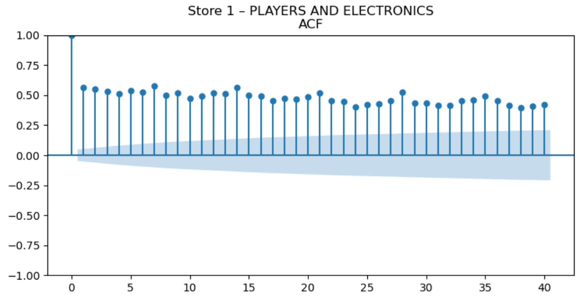

Store 1 – MEATS
PACF

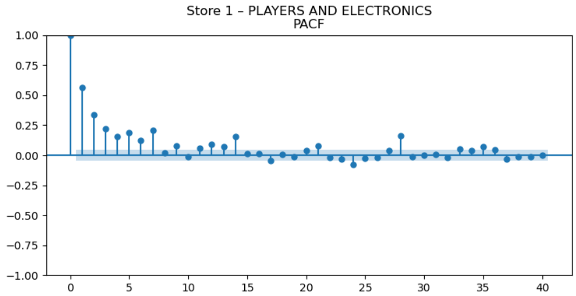

Store 1 – PERSONAL CARE
ACF

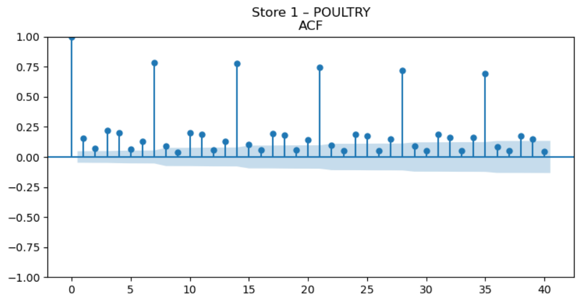

Store 1 – PERSONAL CARE
PACF

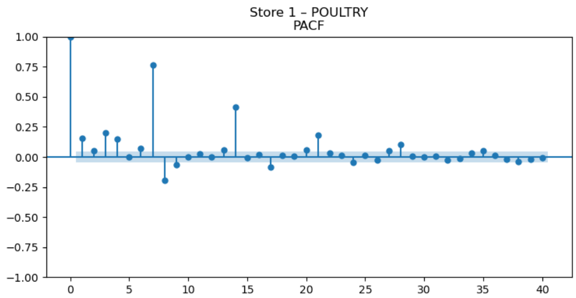

Store 1 – PET SUPPLIES
ACF

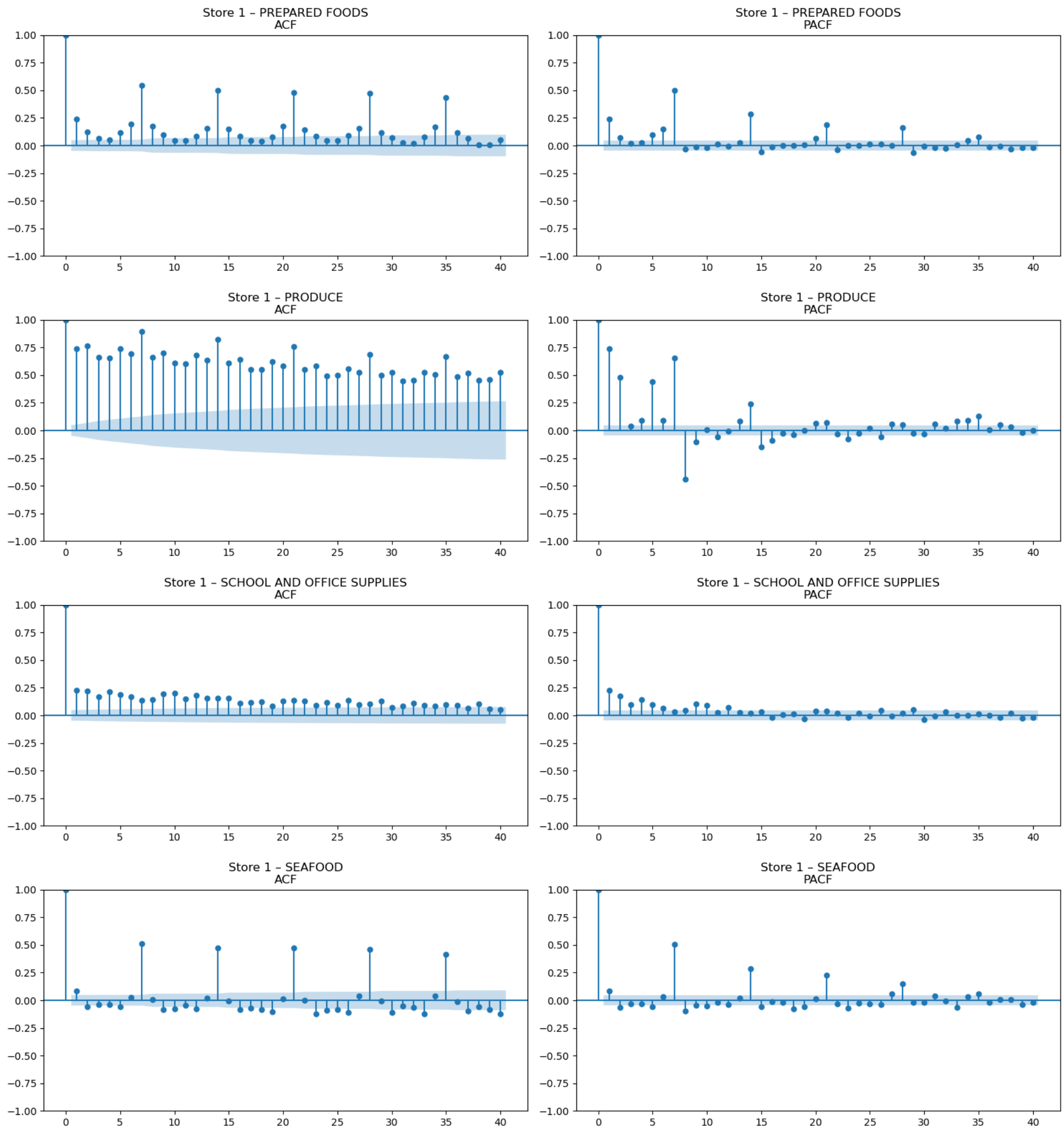

Store 1 – PET SUPPLIES
PACF


Store 1 – PLAYERS AND ELECTRONICS
ACF


Store 1 – PLAYERS AND ELECTRONICS
PACF


Store 1 – POULTRY
ACF


Store 1 – POULTRY
PACF



Model training

In [23]:
```
train = df[df['date'] <= '2017-07-31']
test = df[(df['date'] > '2017-07-31') & (df['date'] <= '2017-08-15'

print("Training period:", train['date'].min(), "to", train['date'].
print("Validation period:", test['date'].min(), "to", test['date'].
print("Train size:", len(train), "rows,", "Validation size:", len(t
train.head()
```

```
Training period: 2013-01-01 00:00:00 to 2017-07-31 00:00:00
Validation period: 2017-08-01 00:00:00 to 2017-08-15 00:00:00
Train size: 2974158 rows, Validation size: 26730 rows
```

Out [23]:

	id	date	store_nbr	family	sales	onpromotion	city	state
0	0	2013-01-01	1	AUTOMOTIVE	0.0	0	Quito	Pichincha
1	1	2013-01-01	1	BABY CARE	0.0	0	Quito	Pichincha
2	2	2013-01-01	1	BEAUTY	0.0	0	Quito	Pichincha
3	3	2013-01-01	1	BEVERAGES	0.0	0	Quito	Pichincha
4	4	2013-01-01	1	BOOKS	0.0	0	Quito	Pichincha

SAIMAX model

In [24]:

```
series_key = (1, 'GROCERY I')
train_series = train[(train['store_nbr']==series_key[0]) &
                     (train['family']==series_key[1])]
val_series = test[(test['store_nbr']==series_key[0]) &
                  (test['family']==series_key[1])]


y_train = train_series['sales']
exog_features = ['onpromotion', 'is_holiday', 'dcoilwtico', 'transa
exog_train = train_series[exog_features]

sarimax_model = SARIMAX(y_train, exog=exog_train, order=(1,1,1), se
                        enforce_stationarity=False, enforce_inverti
sarimax_result = sarimax_model.fit(disp=False)
print(sarimax_result.summary().tables[1])
n_val = len(val_series)
exog_val = val_series[exog_features]
sarimax_forecast = sarimax_result.forecast(steps=n_val, exog=exog_v
def rmsle(y_true, y_pred):
    return np.sqrt(np.mean(np.square(np.log1p(y_pred) - np.log1p(y_

series_rmsle = rmsle(val_series['sales'].values, sarimax_forecast.v
print(f"SARIMAX validation RMSLE for store {series_key[0]}, family
```

```
==============================================================================
============
                  coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------------
onpromotion     5.3473      0.610      8.769      0.000       4.152
6.542
is_holiday    -80.8851     32.132     -2.517      0.012    -143.862
-17.908
dcoilwtico    -33.1567      5.893     -5.627      0.000     -44.706
-21.608
transactions    1.2234      0.029     42.494      0.000       1.167
1.280
ar.L1           0.3605      0.014     26.169      0.000       0.333
0.387
ma.L1          -1.0507      0.009   -116.925      0.000      -1.068
-1.033
ar.S.L7        -0.0022      0.024     -0.093      0.926      -0.049
0.044
ma.S.L7        -0.8748      0.017    -50.302      0.000      -0.909
-0.841
sigma2       1.485e+05   2766.743     53.656      0.000    1.43e+05
1.54e+05
==============================================================================
============
SARIMAX validation RMSLE for store 1, family GROCERY I: 0.1419
```

In [56]:
```python
results = []
combinations = train[['store_nbr', 'family']].drop_duplicates()

for _, row in combinations.iterrows():
    store, family = row['store_nbr'], row['family']

    train_series = train[(train['store_nbr'] == store) & (train['fa
    val_series   = test[(test['store_nbr'] == store) & (test['famil

    if len(train_series) < 30 or len(val_series) == 0:
        continue

    try:
        y_train = train_series['sales']
        exog_train = train_series[exog_features]
        exog_val   = val_series[exog_features]

        model = SARIMAX(
            y_train, exog=exog_train,
            order=(1, 1, 1), seasonal_order=(1, 1, 1, 7),
            enforce_stationarity=False, enforce_invertibility=False
        )
        result = model.fit(disp=False)
        forecast = result.forecast(steps=len(val_series), exog=exog_

        rmsle_score = rmsle(val_series['sales'].values, forecast.va
        results.append({'store_nbr': store, 'family': family, 'rmsl
    except Exception as e:
```

```
            print(f"Skipped ({store}, {family}) due to error: {e}")
            continue


results_df = pd.DataFrame(results)
results_df.to_csv("sarimax_rmsle_results.csv", index=False)
print(results_df.sort_values('rmsle').head(10))
```

```
      store_nbr      family  rmsle
829          32       BOOKS    0.0
1354         47   BABY CARE    0.0
679          28  LADIESWEAR    0.0
37           10       BOOKS    0.0
877          33  LADIESWEAR    0.0
664          28       BOOKS    0.0
1240         43  LADIESWEAR    0.0
250          16  LADIESWEAR    0.0
1141         40  LADIESWEAR    0.0
928          35       BOOKS    0.0
```

In [61]:
```
results = pd.DataFrame(results)
overall_rmsle = np.sqrt(np.mean(results['rmsle'] ** 2))

print("Overall RMSLE:", overall_rmsle)
```

```
Overall RMSLE: 0.42663089438997875
```

CatBoost

In [50]:
```
X_train = train.drop(columns=['date', 'sales'])
X_test = test.drop(columns=['date', 'sales'])
y_train = train['sales']
y_test = test['sales']

from catboost import CatBoostRegressor
cat_feature_names = ['family', 'city', 'state', 'type']
cat_feature_indices = [X_train.columns.get_loc(col) for col in cat_


model = CatBoostRegressor(
    iterations=600,
    learning_rate=0.1,
    depth=7,
    verbose=100
)

model.fit(
    X_train,
    y_train,
    cat_features=cat_feature_indices
)

preds = model.predict(X_test)
```

```
0:      learn: 1054.9797107     total: 638ms    remaining: 6m 21s
100:    learn: 358.6449079      total: 47.3s    remaining: 3m 53s
200:    learn: 323.2161544      total: 1m 40s   remaining: 3m 19s
300:    learn: 301.4173992      total: 2m 37s   remaining: 2m 36s
400:    learn: 287.4052151      total: 3m 32s   remaining: 1m 45s
500:    learn: 276.0157785      total: 4m 26s   remaining: 52.7s
599:    learn: 267.6864603      total: 5m 20s   remaining: 0us
```

In [51]:

```
rmsle_cat = rmsle(y_test, preds)
print(f"CatBoost validation RMSLE: {rmsle_cat:.4f}")
```

```
CatBoost validation RMSLE: 0.9997
```

In [ ]:

```
```

Deeplearning Models

ANN

In [43]:

```
for col in cat_feature_names:
    le = LabelEncoder()
    X_train[col] = le.fit_transform(X_train[col].astype(str))
    X_test[col] = le.transform(X_test[col].astype(str))

num_cols = X_train.columns.difference(cat_feature_names).tolist()
scaler = StandardScaler()
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])


input_dim = X_train.shape[1]

model_ann = Sequential([
    Dense(128, activation='relu', input_dim=input_dim),
    Dropout(0.2),
    Dense(64,  activation='relu'),
    Dropout(0.1),
    Dense(1)
])
model_ann.compile(optimizer='adam', loss='mse')

history = model_ann.fit(
    X_train.values, y_train,
    validation_data=(X_test.values, y_test),
    epochs=50,
    batch_size=32,
    verbose=1
)

ann_preds = model_ann.predict(X_test.values).flatten().clip(0)
print("ANN RMSLE:", rmsle(y_test, ann_preds))
```

```
Epoch 1/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 375us/step - loss: 724965.4375
- val_loss: 427405.8125
```

```
Epoch 2/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 383us/step - loss: 395747.2812
- val_loss: 325056.9375
Epoch 3/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 384us/step - loss: 318025.2812
- val_loss: 225053.8281
Epoch 4/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 37s 398us/step - loss: 270169.2812
- val_loss: 214891.3594
Epoch 5/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 38s 405us/step - loss: 256036.3281
- val_loss: 199257.5781
Epoch 6/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 380us/step - loss: 251677.5469
- val_loss: 201097.6875
Epoch 7/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 367us/step - loss: 226579.2031
- val_loss: 186281.5000
Epoch 8/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 362us/step - loss: 224704.2344
- val_loss: 173584.9688
Epoch 9/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 369us/step - loss: 226496.9844
- val_loss: 171293.4531
Epoch 10/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 389us/step - loss: 210443.5625
- val_loss: 167952.7812
Epoch 11/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 380us/step - loss: 218922.2031
- val_loss: 192884.3750
Epoch 12/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 391us/step - loss: 207073.8906
- val_loss: 155555.0156
Epoch 13/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 379us/step - loss: 217547.2031
- val_loss: 157898.2812
Epoch 14/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 390us/step - loss: 200611.4375
- val_loss: 192516.2656
Epoch 15/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 384us/step - loss: 213970.6406
- val_loss: 161398.6250
Epoch 16/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 385us/step - loss: 196884.9375
- val_loss: 155616.1250
Epoch 17/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 390us/step - loss: 198663.9219
- val_loss: 193898.9844
Epoch 18/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 376us/step - loss: 192877.5781
- val_loss: 246439.1875
Epoch 19/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 389us/step - loss: 190941.3750
- val_loss: 145925.8750
Epoch 20/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 384us/step - loss: 205003.0000
```

```
- val_loss: 160612.0156
Epoch 21/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 382us/step - loss: 196211.1250
- val_loss: 166851.0938
Epoch 22/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 37s 392us/step - loss: 197947.5000
- val_loss: 165686.4375
Epoch 23/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 369us/step - loss: 189226.0156
- val_loss: 173440.6875
Epoch 24/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 33s 359us/step - loss: 177975.5938
- val_loss: 163216.2188
Epoch 25/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 381us/step - loss: 186733.8594
- val_loss: 147762.7812
Epoch 26/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 363us/step - loss: 189749.2500
- val_loss: 149810.2969
Epoch 27/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 360us/step - loss: 178990.6406
- val_loss: 153051.8750
Epoch 28/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 370us/step - loss: 186934.2969
- val_loss: 161705.8125
Epoch 29/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 391us/step - loss: 177098.0000
- val_loss: 135121.9844
Epoch 30/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 364us/step - loss: 170950.6875
- val_loss: 139069.6250
Epoch 31/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 362us/step - loss: 172360.5781
- val_loss: 148676.0469
Epoch 32/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 33s 359us/step - loss: 172744.6250
- val_loss: 168409.1719
Epoch 33/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 372us/step - loss: 174352.8906
- val_loss: 149469.8594
Epoch 34/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 33s 360us/step - loss: 175414.9375
- val_loss: 140922.4688
Epoch 35/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 367us/step - loss: 172557.2500
- val_loss: 146215.8438
Epoch 36/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 33s 357us/step - loss: 179342.8281
- val_loss: 129323.3438
Epoch 37/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 33s 355us/step - loss: 168879.8281
- val_loss: 148916.5938
Epoch 38/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 34s 366us/step - loss: 172394.5156
- val_loss: 160249.1406
Epoch 39/50
```

```
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 36s 382us/step - loss: 186995.5625
- val_loss: 137815.1875
Epoch 40/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 376us/step - loss: 170306.4219
- val_loss: 139160.8750
Epoch 41/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 377us/step - loss: 171504.1094
- val_loss: 148385.6094
Epoch 42/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 378us/step - loss: 164788.8438
- val_loss: 152800.0625
Epoch 43/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 377us/step - loss: 178498.2188
- val_loss: 132143.8906
Epoch 44/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 378us/step - loss: 170506.1875
- val_loss: 136636.1875
Epoch 45/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 377us/step - loss: 186354.8750
- val_loss: 135458.5781
Epoch 46/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 377us/step - loss: 166900.4531
- val_loss: 135461.5625
Epoch 47/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 376us/step - loss: 162601.3438
- val_loss: 130549.3125
Epoch 48/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 374us/step - loss: 169067.8906
- val_loss: 131386.4844
Epoch 49/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 379us/step - loss: 166145.7188
- val_loss: 152522.2656
Epoch 50/50
92943/92943 ━━━━━━━━━━━━━━━━━━━━ 35s 376us/step - loss: 168264.9844
- val_loss: 156142.3750
836/836 ━━━━━━━━━━━━━━━━━━━━ 0s 219us/step
ANN RMSLE: 1.7919142241944601
```

CNN

In [ ]:
```python
full = pd.concat([train, test]) \
         .sort_values(['store_nbr','family','date']) \
         .reset_index(drop=True)


le_store = LabelEncoder()
full['store_enc'] = le_store.fit_transform(full['store_nbr'])
le_fam   = LabelEncoder()
full['fam_enc']   = le_fam.fit_transform(full['family'])


seq_feats = [
    'sales', 'onpromotion', 'is_holiday',
    'dcoilwtico', 'transactions',
    'store_enc', 'fam_enc'
```

```
]

N = 14
train_max_date = train['date'].max()

X_tr_seq, y_tr_seq = [], []
X_te_seq, y_te_seq = [], []


for (_, _), grp in full.groupby(['store_nbr','family']):
    grp = grp.sort_values('date').reset_index(drop=True)
    for i in range(N, len(grp)):
        window = grp.loc[i-N:i-1, seq_feats].values
        target_date = grp.loc[i, 'date']
        target_sale = grp.loc[i, 'sales']

        if target_date <= train_max_date:
            X_tr_seq.append(window);  y_tr_seq.append(target_sale)
        else:
            X_te_seq.append(window);  y_te_seq.append(target_sale)

X_tr_seq = np.array(X_tr_seq)
y_tr_seq = np.array(y_tr_seq)
X_te_seq = np.array(X_te_seq)
y_te_seq = np.array(y_te_seq)

print("train seq:", X_tr_seq.shape, "test seq:", X_te_seq.shape)
```

```
train seq: (2949210, 14, 7) test seq: (26730, 14, 7)
```

```
--------------------------------------------------------------------
-------
NameError                                 Traceback (most recent cal
l last)
Cell In[57], line 53
     46 print("train seq:", X_tr_seq.shape, "test seq:", X_te_seq.sh
ape)
     48 #
─
────────────────────────────────────────────────────────────────────
─
     49 # 2) Define & train the CNN
     50 model_cnn = Sequential([
     51     Conv1D(32, 3, activation='relu', input_shape=(N, len(seq
_feats))),
     52     MaxPooling1D(2),
---> 53     Flatten(),
     54     Dense(64, activation='relu'),
     55     Dense(1)   # linear output
     56 ])
     57 model_cnn.compile(optimizer='adam', loss='mse')
     59 history = model_cnn.fit(
     60     X_tr_seq, y_tr_seq,
     61     validation_data=(X_te_seq, y_te_seq),
   (...)
     64     verbose=1
     65 )

NameError: name 'Flatten' is not defined
```

In [59]:
```
model_cnn = Sequential([
    Conv1D(32, 3, activation='relu', input_shape=(N, len(seq_feats)
    MaxPooling1D(2),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1)
])
model_cnn.compile(optimizer='adam', loss='mse')

history = model_cnn.fit(
    X_tr_seq, y_tr_seq,
    validation_data=(X_te_seq, y_te_seq),
    epochs=20,
    batch_size=64,
    verbose=1
)



cnn_preds = model_cnn.predict(X_te_seq).flatten().clip(0)
def rmsle(y_true, y_pred):
    return np.sqrt(np.mean((np.log1p(y_pred)-np.log1p(y_true))**2))

print("CNN RMSLE:", rmsle(y_te_seq, cnn_preds))
```

```
Epoch 1/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 32s 689us/step - loss: 146681.0156
```

```
- val_loss: 90892.6797
Epoch 2/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 29s 625us/step - loss: 76827.0547 -
val_loss: 64996.5391
Epoch 3/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 606us/step - loss: 82135.6719 -
val_loss: 57614.9102
Epoch 4/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 606us/step - loss: 72212.2578 -
val_loss: 62621.5430
Epoch 5/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 607us/step - loss: 77133.5234 -
val_loss: 59538.0586
Epoch 6/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 612us/step - loss: 70197.7500 -
val_loss: 72716.6484
Epoch 7/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 1409s 31ms/step - loss: 73327.0391
- val_loss: 53333.1328
Epoch 8/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 27s 575us/step - loss: 70030.2266 -
val_loss: 79379.4531
Epoch 9/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 27s 593us/step - loss: 76086.3125 -
val_loss: 55868.1289
Epoch 10/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 617us/step - loss: 67867.5391 -
val_loss: 50817.9297
Epoch 11/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 616us/step - loss: 59951.6719 -
val_loss: 57692.3242
Epoch 12/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 613us/step - loss: 68322.5938 -
val_loss: 59911.7773
Epoch 13/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 616us/step - loss: 73027.4141 -
val_loss: 54790.2188
Epoch 14/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 615us/step - loss: 67984.0312 -
val_loss: 56354.3984
Epoch 15/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 29s 619us/step - loss: 77199.9453 -
val_loss: 57063.7305
Epoch 16/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 29s 620us/step - loss: 66860.2734 -
val_loss: 57783.9414
Epoch 17/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 28s 616us/step - loss: 63030.0547 -
val_loss: 56720.0547
Epoch 18/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 30s 641us/step - loss: 56338.0195 -
val_loss: 63579.8828
Epoch 19/20
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 47s 1ms/step - loss: 65576.3906 - v
al_loss: 56965.6836
Epoch 20/20
```

```
46082/46082 ━━━━━━━━━━━━━━━━━━━━ 29s 620us/step - loss: 71273.6797 -
val_loss: 57971.0625
836/836 ━━━━━━━━━━━━━━━━━━━━ 0s 259us/step
CNN RMSLE: 1.018394351235643
```

The best model CatBoost

training submission data

In [ ]:
```
features = df.drop(columns=['date', 'sales'])
cat_feature_indices = [features.columns.get_loc(col) for col in cat
model = CatBoostRegressor(
    iterations=600,
    learning_rate=0.1,
    depth=7,
    verbose=100
)
model.fit(
    features,
    df['sales'],
    cat_features=cat_feature_indices
)
```

```
0:      learn: 1056.2901431     total: 601ms    remaining: 5m 59s
100:    learn: 367.3552426      total: 47s      remaining: 3m 52s
200:    learn: 326.4408369      total: 1m 35s   remaining: 3m 10s
300:    learn: 304.3553557      total: 2m 24s   remaining: 2m 23s
400:    learn: 290.4660154      total: 3m 12s   remaining: 1m 35s
500:    learn: 279.5679376      total: 4m 6s    remaining: 48.6s
599:    learn: 270.6415999      total: 5m 1s    remaining: 0us
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent cal
l last)
File _catboost.pyx:2547, in _catboost.get_float_feature()

File _catboost.pyx:1226, in _catboost._FloatOrNan()

File _catboost.pyx:1021, in _catboost._FloatOrNanFromString()

TypeError: Cannot convert 'b'AUTOMOTIVE'' to float

During handling of the above exception, another exception occurred:

CatBoostError                             Traceback (most recent cal
l last)
Cell In[54], line 14
      3 model = CatBoostRegressor(
      4     iterations=600,
      5     learning_rate=0.1,
      6     depth=7,
      7     verbose=100
      8 )
      9 model.fit(
```

```
     10     features,
     11     df['sales'],
     12     cat_features=cat_feature_indices
     13 )
---> 14 preds = model.predict(submission.drop(columns=['date', 'i
d']))

File /opt/anaconda3/lib/python3.12/site-packages/catboost/core.py:59
24, in CatBoostRegressor.predict(self, data, prediction_type, ntree_
start, ntree_end, thread_count, verbose, task_type)
   5922 if prediction_type is None:
   5923     prediction_type = self._get_default_prediction_type()
-> 5924 return self._predict(data, prediction_type, ntree_start, ntr
ee_end, thread_count, verbose, 'predict', task_type)

File /opt/anaconda3/lib/python3.12/site-packages/catboost/core.py:26
20, in CatBoost._predict(self, data, prediction_type, ntree_start, n
tree_end, thread_count, verbose, parent_method_name, task_type)
   2618 if verbose is None:
   2619     verbose = False
-> 2620 data, data_is_single_object = self._process_predict_input_da
ta(data, parent_method_name, thread_count)
   2621 self._validate_prediction_type(prediction_type)
   2623 predictions = self._base_predict(data, prediction_type, ntre
e_start, ntree_end, thread_count, verbose, task_type)

File /opt/anaconda3/lib/python3.12/site-packages/catboost/core.py:26
00, in CatBoost._process_predict_input_data(self, data, parent_metho
d_name, thread_count, label)
   2598 is_single_object = _is_data_single_object(data)
   2599 if not isinstance(data, Pool):
-> 2600     data = Pool(
   2601         data=[data] if is_single_object else data,
   2602         label=label,
   2603         cat_features=self._get_cat_feature_indices() if not
isinstance(data, FeaturesData) else None,
   2604         text_features=self._get_text_feature_indices() if no
t isinstance(data, FeaturesData) else None,
   2605         embedding_features=self._get_embedding_feature_indic
es() if not isinstance(data, FeaturesData) else None,
   2606         thread_count=thread_count
   2607     )
   2608 return data, is_single_object

File /opt/anaconda3/lib/python3.12/site-packages/catboost/core.py:85
5, in Pool.__init__(self, data, label, cat_features, text_features,
embedding_features, embedding_features_data, column_description, pai
rs, graph, delimiter, has_header, ignore_csv_quoting, weight, group_
id, group_weight, subgroup_id, pairs_weight, baseline, timestamp, fe
ature_names, feature_tags, thread_count, log_cout, log_cerr, data_ca
n_be_none)
    849         if isinstance(feature_names, PATH_TYPES):
    850             raise CatBoostError(
    851                 "feature_names must be None or have non-stri
ng type when the pool is created from "
    852                 "python objects."
```

```
    853              )
--> 855         self._init(data, label, cat_features, text_features, embedding_features, embedding_features_data, pairs, graph, weight,
    856                     group_id, group_weight, subgroup_id, pairs_weight, baseline, timestamp, feature_names, feature_tags, thread_count)
    857 elif not data_can_be_none:
    858     raise CatBoostError("'data' parameter can't be None")

File /opt/anaconda3/lib/python3.12/site-packages/catboost/core.py:1491, in Pool._init(self, data, label, cat_features, text_features, embedding_features, embedding_features_data, pairs, graph, weight, group_id, group_weight, subgroup_id, pairs_weight, baseline, timestamp, feature_names, feature_tags, thread_count)
   1489 if feature_tags is not None:
   1490     feature_tags = self._check_transform_tags(feature_tags, feature_names)
-> 1491 self._init_pool(data, label, cat_features, text_features, embedding_features, embedding_features_data, pairs, graph, weight,
   1492                 group_id, group_weight, subgroup_id, pairs_weight, baseline, timestamp, feature_names, feature_tags, thread_count)

File _catboost.pyx:4339, in _catboost._PoolBase._init_pool()

File _catboost.pyx:4391, in _catboost._PoolBase._init_pool()

File _catboost.pyx:4200, in _catboost._PoolBase._init_features_order_layout_pool()

File _catboost.pyx:3127, in _catboost._set_features_order_data_pd_data_frame()

File _catboost.pyx:2591, in _catboost.create_num_factor_data()

File _catboost.pyx:2549, in _catboost.get_float_feature()

CatBoostError: Bad value for num_feature[non_default_doc_idx=0,feature_idx=1]="AUTOMOTIVE": Cannot convert 'b'AUTOMOTIVE'' to float
```

In [ ]:

```
X_submission = submission[features.columns]
preds = model.predict(X_submission)
sub = submission[['id']].copy()
sub['sales'] = preds.clip(0)
sub.to_csv('submission_catboost.csv', index=False)
```

Random Forest

In [66]:

```
X_train_enc = train.drop(columns=['date', 'sales']).copy()
y_train = train['sales']

X_test_enc  = test.drop(columns=['date', 'sales']).copy()
y_test = test['sales']
```

```
# Identify object columns (categoricals)
cat_cols = X_train_enc.select_dtypes(include=['object', 'category']

# Label encode them
for col in cat_cols:
    le = LabelEncoder()
    X_train_enc[col] = le.fit_transform(X_train_enc[col].astype(str
    X_test_enc[col]  = le.transform(X_test_enc[col].astype(str))

# Now you can fit the RandomForest
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(random_state=42)
rf.fit(X_train_enc, y_train)

# Predict
rf_preds = rf.predict(X_test_enc)

# Evaluate
def rmsle(y_true, y_pred):
    return np.sqrt(np.mean((np.log1p(y_pred) - np.log1p(y_true))**2

print("Random Forest RMSLE:", rmsle(y_test, rf_preds))
```

```
Random Forest RMSLE: 0.49665852946033556
```

In [67]:

```
target = df['sales']

cat_cols = features.select_dtypes(include=['object', 'category']).c
submission_enc = X_submission.copy()

features_enc = features.copy()
for col in cat_cols:
    le = LabelEncoder()
    features_enc[col] = le.fit_transform(features_enc[col].astype(s
    submission_enc[col] = le.transform(submission_enc[col].astype(s


rf = RandomForestRegressor(random_state=42,)
rf.fit(features_enc, target)
submission_preds = rf.predict(submission_enc)
```

In [69]:

```
submission_preds = pd.DataFrame({
    'id': submission['id'],
    'sales': submission_preds.clip(0)
})
submission_preds.to_csv('submission_pred.csv', index=False)
```

At the end i tried Random forest without parameter tuning, because i thought we were not using everything possible with

In [ ]: