# Retail Forecasting

Data Forecasting Team

# Problem Description:

The large beverage company in Australia needs to forecast demand for each of their products at the item level, on a weekly basis. Their sales are influenced by various factors including promotions, holidays, and seasonality. The company currently uses an in-house software solution for forecasting, but it often produces unreliable results. They want to explore AI/ML-based forecasting to replace their current system.

# Business Understanding

**Business Objectives:**

- Develop an AI/ML-based forecasting model that can accurately predict weekly demand for each product
- Improve forecast accuracy compared to the current in-house solution
- Account for various influencing factors such as promotions, holidays, and seasonality
- Enable better inventory management and production planning
- Optimize promotional strategies based on accurate demand predictions

**Target Audience:**

- Sales and Operations Planning team
- Inventory Management team
- Production Planning team
- Marketing team (for promotional planning)

# Data Understanding

The dataset we are analyzing is the “forecasting_case_study.xlsx” file, which contains 1218 rows and 12 features, comprising of “Product”, “date”, “Sales”, “Price Discount (%) ”, “In-Store Promo”, “Catalogue Promo”, “Store End Promo”, “Google_Mobility”, “Covid_Flag”, “V_DAY”, “EASTER” and “CHRISTMAS”. Surprisingly, there were no missing data in the file

The dataset contained six unique Products: SKU1, SKU2, SKU3, SKU4, SKU5, SKU6

# Data Understanding

1. **Product**: SKU identifier (SKU1, SKU2, etc.) **Dtype: object**

2. **Date**: Weekly dates from 2017 to 2020 **Dtype: object**

3. **Sales**: Numeric sales figures **Dtype: int64**

4. **Price Discount (%):** Percentage of price discount offered **Dtype: object**

5. **In-Store Promo**: Binary (0 or 1) indicating in-store promotions **Dtype: int64**

6. **Catalogue Promo**: Binary (0 or 1) indicating catalogue promotions **Dtype: int64**

7. **Store End Promo**: Binary (0 or 1) indicating store end promotions **Dtype: int64**

8. **Google_Mobility**: Numeric values (possibly related to Google's mobility reports during COVID-19) **Dtype: float64**

9. **Covid_Flag**: Binary (0 or 1) indicating COVID-19 period **Dtype: int64**

10. **V_DAY**: Binary (0 or 1) possibly indicating Valentine's Day **Dtype: int64**

11. **EASTER**: Binary (0 or 1) indicating Easter holiday **Dtype: int64**

12. **CHRISTMAS**: Binary (0 or 1) indicating Christmas holiday **Dtype: int64**

# Data Understanding

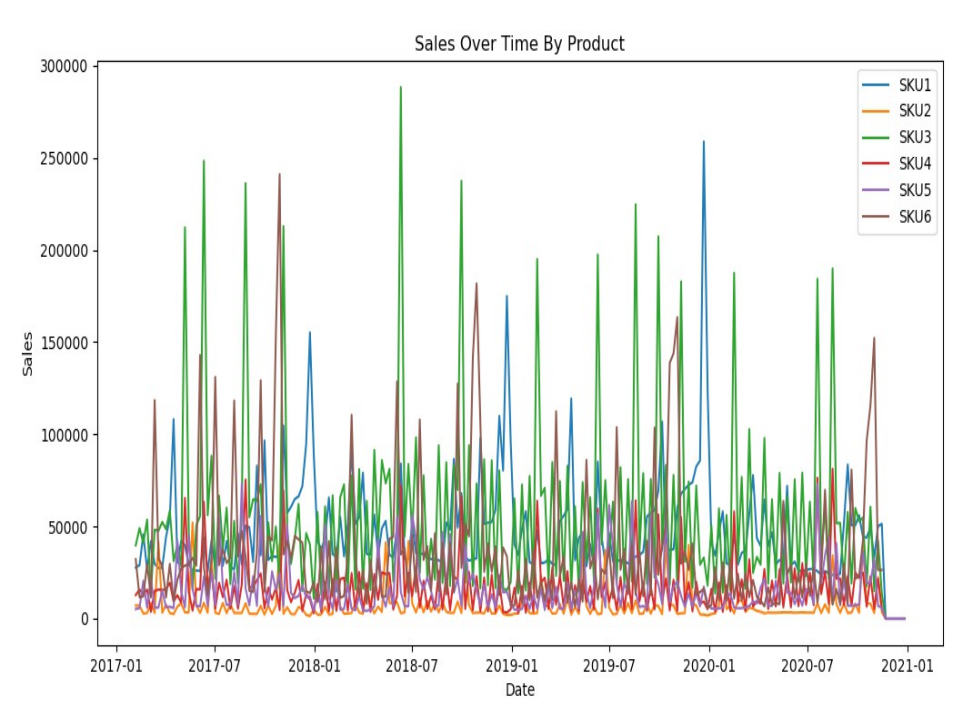
## Sales Data Analysis: Insights and Trends

- Dataset: Sales data from February 2017 to December 2020
- Products: SKU1, SKU2, SKU3, SKU4, SKU5, SKU6
- Key variables: Sales, Promotions, Holidays, and COVID-19 impact

# Sales Overview

## Sales Data Analysis: Insights and Trends

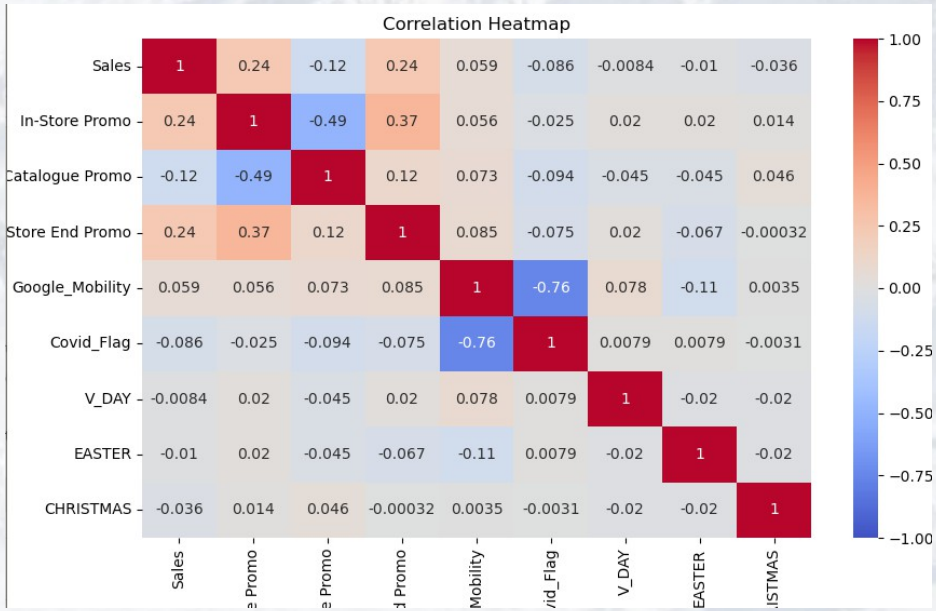
- Overall Increasing Trend In sales from Feb 2017 to Dec 2020
- Significant spikes in sales, likely due to promotional activities
- Noticeable seasonality, with higher sales during certain periods



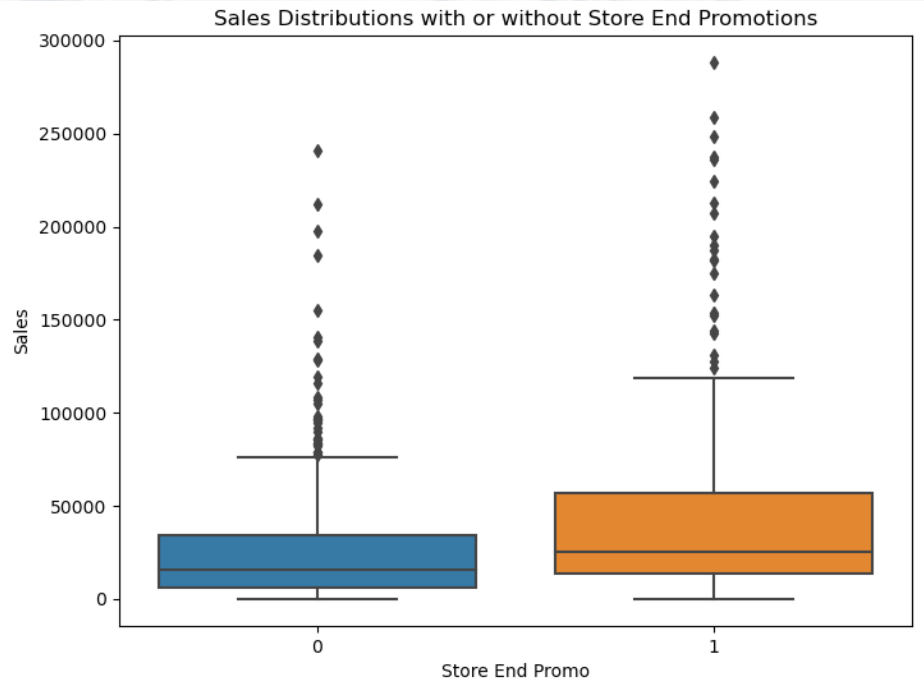
# Key Insights

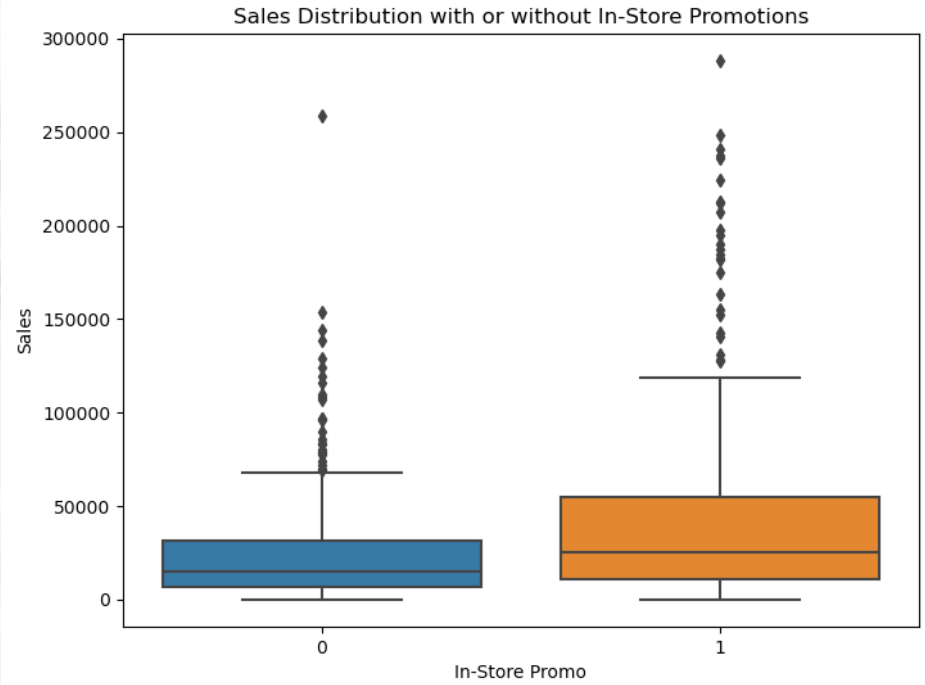
## Sales Drivers and Patterns

- Promotions Impact: In-store promotions (correlation: 0.24) and Store End promotions (correlation: 0.24) have a positive effect on sales
- Seasonal Effects: Most of the seasons have negative correlations with Easter (-0.01), V_Day (-0.0084) and negative Christmas (-0.036)
- COVID-19 Impact: Negative correlation (-0.086) between sales and COVID-19 flag, indicating a slight decrease in sales during the pandemic



# How Promotions Effect Sales

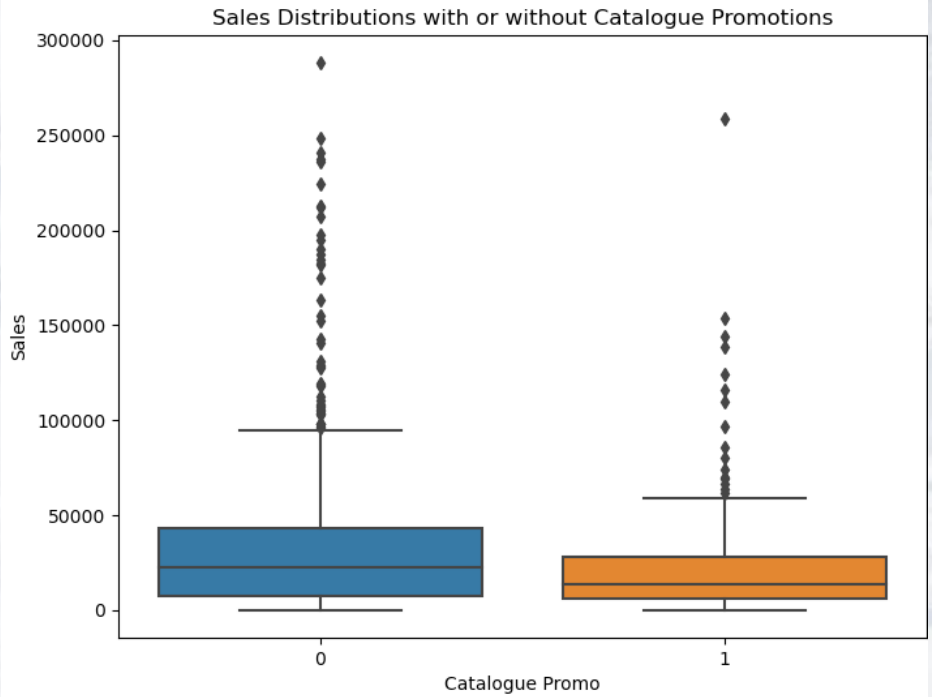


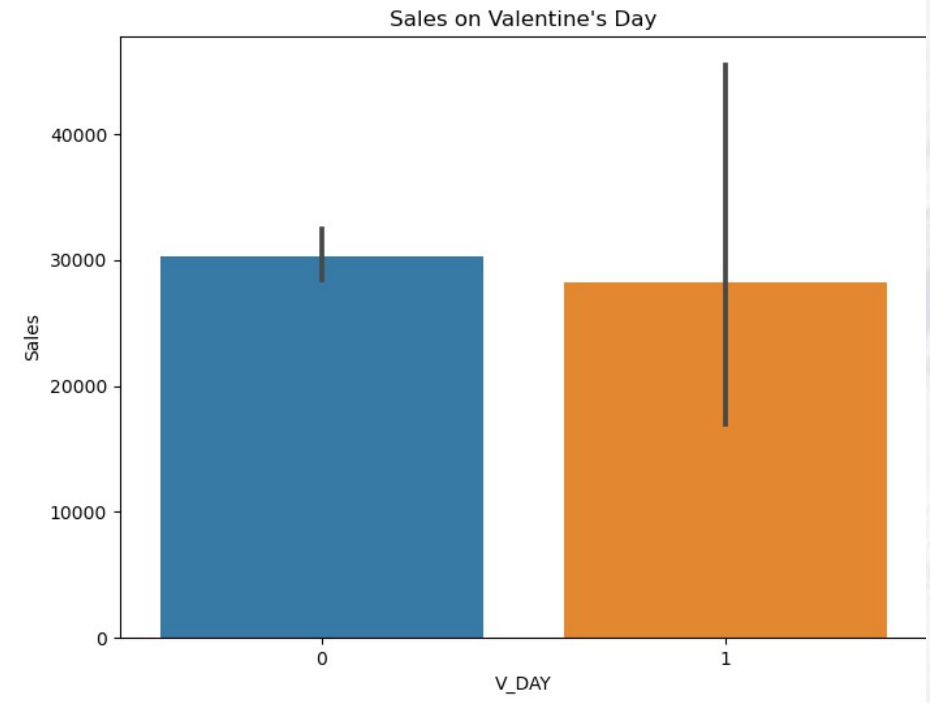
# How Promotions Effect Sales

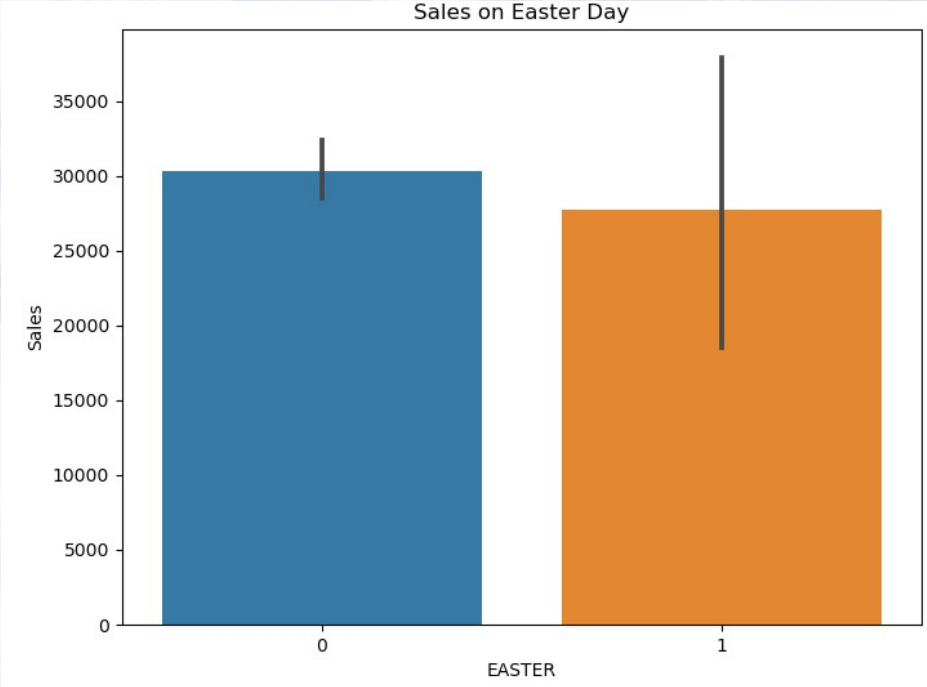
In the first two box plots, we noticed there was an increase in Sales on the In-Store and End Store Promotion, but the opposite effect for Catalogue Promotions.

This indicates that sellers should be more invested in these type of promotions



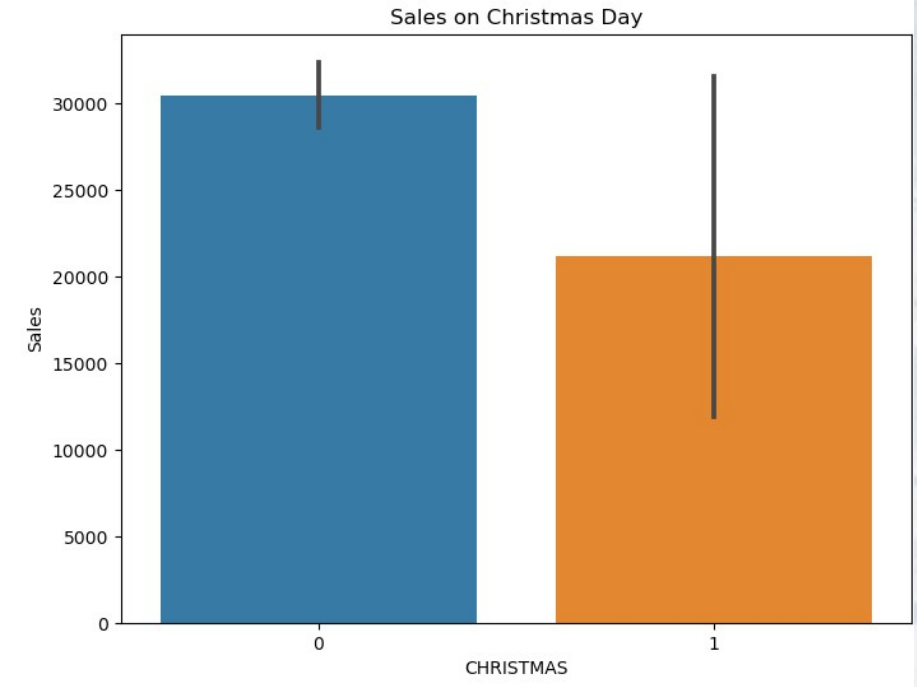
# Seasonal Effects of Sales



# Seasonal Effects of Sales

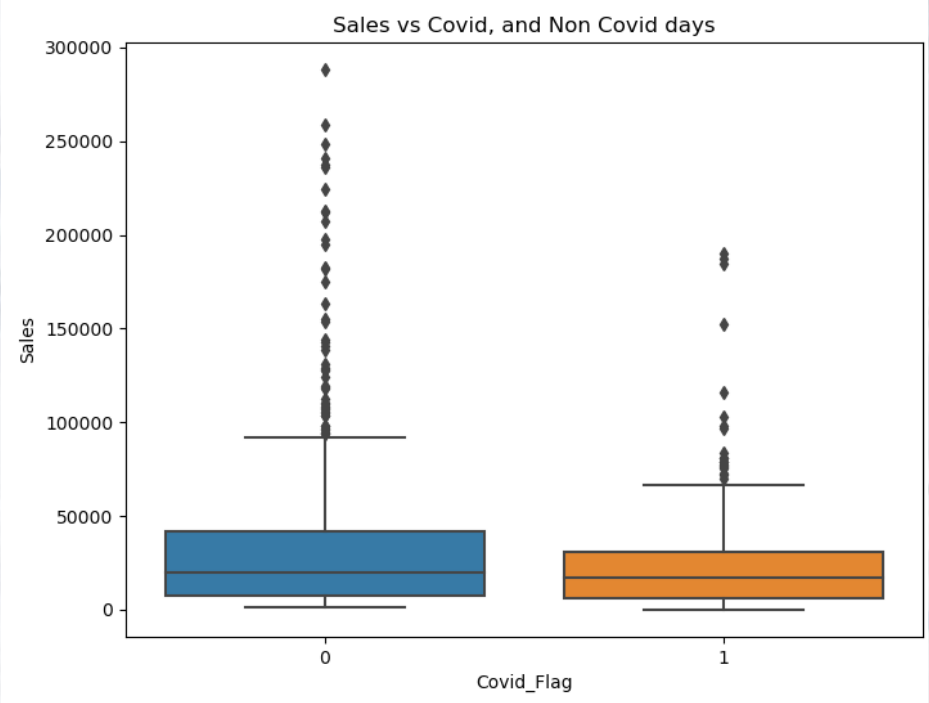
From our box plot, we can see that instead of an increase of Sales on Festive Seasons, there was a decrease, signifying that the holidays doesn't improve the sales of these products.



# COVID-19 Impact on Sales

We can observe from the box plot on the right that COVID has a negative influence on the Sales of such product.

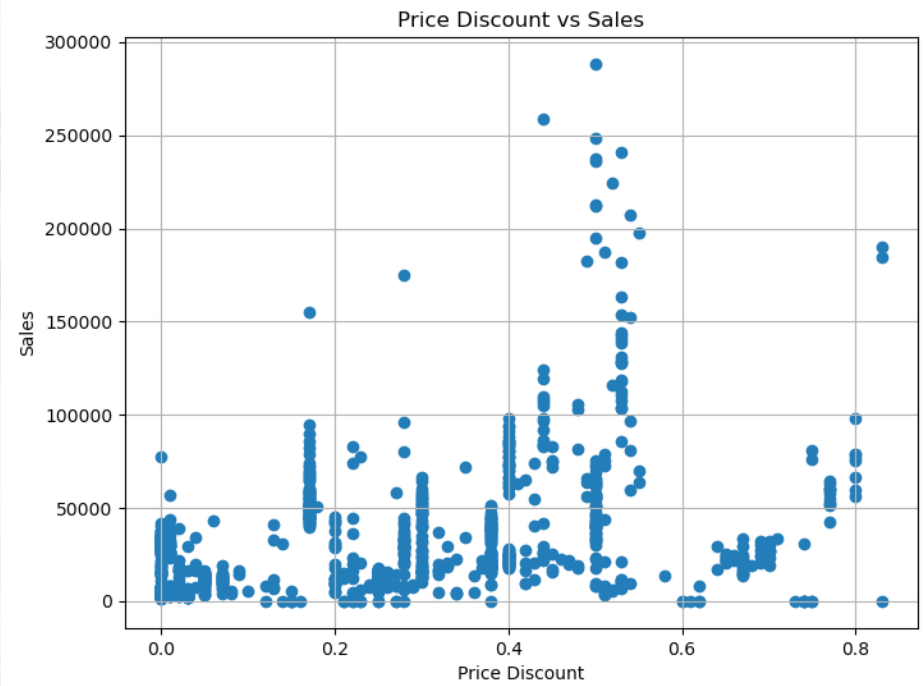
- This helps us understand that a pandemic could affect sales in a negative way.



# Other External Factors

## Price Discount

- From the scatter plot on the right, we see that the densely populated % is between 0 to 0.58. After that there's little to no correlation with the Sales



# Recommendations

- From my observations on the products, SKU3 has the highest spikes, indicating that it's influenced by Promotions.
- SKU2 product has the least spike, even though it made the most sales throughout the 3 years, so its value isn't influenced by seasonality or promotions
- Even though SKU6 made the lowest sales, it has the same pattern of spikes of SKU3, also signifying how its sales are moved by promotions.
- The Seller should focus more on giving In-Store and End Promotions for most of their products, and maintain the price discount to 0.2.

# Model Recommendation

## One should use a Linear Regression Model

### Benefits of Using a Linear Regression Model

- Strength: Focuses on the relationship between sales and external predictors like promotions, discounts, or events. It assumes these relationships are linear and consistent over time.
- Implication: A relatively low RMSE(root mean square error) implies the external factors are strong predictors of sales, and time-dependent patterns (like seasonality or trends) may not dominate the data.

# Model Recommendation

## Advisable to Use Random Forest Regressor

### Benefits of Using Random Forest Regressor

- Strength: Handles non-linear relationships and interactions between variables well. It doesn't assume a fixed functional form like Linear Regression.
- Implication: A slightly worse RMSE suggests that the data does not have substantial non-linear relationships or interactions that Random Forest could exploit.

# Model Recommendation

## Time Dependent Models such as AMIRA

### Benefits of Using ARIMA

- Strength: Focuses on temporal dependencies and trends by modeling the relationship between sales and its own past values (lags) and changes.
- Implication: ARIMA is useful if sales show a strong temporal pattern (like steady growth, periodic fluctuations, or trends). The poor performance of ARIMA (as indicated by potential issues in its residuals) suggests that external factors play a more significant role in driving sales than just its past values.

# Information for Technical Users

```
# Evaluating Model Performance with a Linear Regression model

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

# Select features and target variables

X = forecasting_df[["In-Store Promo", "Catalogue Promo", "Store End Promo", "Google_Mobility", "Covid_Flag", "V_DAY",
"EASTER", "CHRISTMAS"]]

y = forecasting_df["Sales"]

# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a linear regression model

lg = LinearRegression()

lg.fit(X_train, y_train)

# print(X_train, "\n:",y_train)
```

# Information for Technical Users

```
# Predict the test set
y_pred = lg.predict(X_test)
# print(y_pred)
# Calculate the Root Mean Square Error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"RMSE: {rmse}")
# Evaluating Model Performance with a Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
# Train a Random Regressor model
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

# Information for Technical Users

```
# Predict the test set
y_pred_rf = rf.predict(X_test)
# Calculate RMSE for the Random Forest Regressor model
rf_rmse = np.sqrt(mean_squared_error(y_test, y_pred_rf))
print(f"rfRMSE: {rf_rmse}")
# Using Time Series Forecasting with ARIMA
from statsmodels.tsa.arima.model  import ARIMA
# ARIMA model on sales data
sales_series = forecasting_df["Sales"]
ts = ARIMA(sales_series, order=(5,1,0))
model_fit = ts.fit()
print(model_fit.summary())
```