Capstone Projekt Rossmann

XDi - Certified Data Scientist

Christoph Gödecke

Reports

Individual Store Reports

Definition of Information and KPIs to be reported

Information

- Period
- Store ID
- Store Type
- Assortment
- Competition Distance
- Competition Open Since
- Promo2Since
- Promo2Interval

KPIs

- Line plot with sales
- Line plot with SalesPerOpenDay
- Line plot with sales per customer
- Line plot with customers
- Line plot with CustomersPerOpenDay
- Line plot with IsPromo
- Line plot with Promo2Active
- Line plot with SchoolHoliday
- Line plot with NumStateHoliday
- Rolling mean
- Compare with other storeTypes and assortments
- weekly and monthly

Input Fields

- StoreId
- StartDate
- EndDate

Store Report

Input and pre setup

In [1]:
```
import pandas as pd
import numpy as np
from datetime import datetime

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from pandas.api.types import infer_dtype
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import warnings
pd.set_option('display.max_columns', None)
import plotly.io as pio
pio.renderers.default = 'notebook_connected'
```

In [2]:
```
# Input fields to define the store and the date range
StoreId = 836
StartDate = "2013-01-01"
EndDate = "2013-12-31"

StartDate = datetime.strptime(StartDate, "%Y-%m-%d")
EndDate = datetime.strptime(EndDate, "%Y-%m-%d")
```

In [3]:
```
df = pd.read_csv('weekly_sales_with_store_info.csv', parse_dates=['Date'])
```

In [4]:
```
print(df.info())
df.sample(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150525 entries, 0 to 150524
Data columns (total 31 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Store                      150525 non-null  int64
 1   Date                       150525 non-null  datetime64[ns]
 2   CW                         150525 non-null  int64
 3   Month                      150525 non-null  int64
 4   Year                       150525 non-null  int64
 5   DayOfWeek                  150525 non-null  int64
 6   Sales                      150525 non-null  int64
 7   SalesPerCustomer           145809 non-null  float64
 8   SalesPerOpenDay            145815 non-null  float64
 9   Customers                  150525 non-null  int64
 10  CustomersPerOpenDay        145815 non-null  float64
 11  Open                       150525 non-null  int64
 12  Promo                      150525 non-null  int64
 13  IsPromo                    150525 non-null  int64
 14  StateHoliday               150525 non-null  object
 15  IsStateHoliday             150525 non-null  int64
 16  SchoolHoliday              150525 non-null  int64
 17  IsSchoolHoliday            150525 non-null  int64
 18  NumStateHoliday            150525 non-null  int64
 19  StoreType                  150525 non-null  object
 20  Assortment                 150525 non-null  object
 21  CompetitionDistance        150120 non-null  float64
 22  CompetitionOpenSinceMonth  102735 non-null  float64
 23  CompetitionOpenSinceYear   102735 non-null  float64
 24  IsCompetition              150525 non-null  int64
 25  Promo2                     150525 non-null  int64
 26  Promo2SinceWeek            77085 non-null   float64
 27  Promo2SinceYear            77085 non-null   float64
 28  PromoInterval              77085 non-null   object
 29  Promo2Member               150525 non-null  int64
 30  Promo2Active               150525 non-null  int64
dtypes: datetime64[ns](1), float64(8), int64(18), object(4)
memory usage: 35.6+ MB
None
```

Out[4]:

	Store	Date	CW	Month	Year	DayOfWeek	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	StateHoliday	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	N
30119	224	2013-04-14	15	4	2013	6	33940	13.089086	5656.666667	2593	432.166667	6	5	1	0	0	0	0	
105560	782	2015-05-31	22	5	2015	6	26164	10.567044	5232.800000	2476	495.200000	5	0	0	a	1	0	0	
142086	1053	2014-04-13	15	4	2014	6	42533	8.050918	7088.833333	5283	880.500000	6	0	0	0	0	7	1	
80650	598	2014-01-26	4	1	2014	6	36010	7.942214	6001.666667	4534	755.666667	6	5	1	0	0	0	0	
28356	211	2013-02-17	7	2	2013	6	71882	8.102119	11980.333333	8872	1478.666667	6	0	0	0	0	0	0	

In [5]:
```
# Generate dataframes based on input fields
df_store = df[(df['Store'] == StoreId) & (df['Date'] >= StartDate) & (df['Date'] <= EndDate)]
StoreType = df_store['StoreType'].iloc[0]
Assortment = df_store['Assortment'].iloc[0]

# Dataframe for comparison with all stores with the same StoreType and Assortment exept the selected one
df_store_compare = df[(df['Store'] != StoreId) & (df['Date'] >= StartDate) & (df['Date'] <= EndDate) & (df['StoreType'] == StoreType) & (df['Assortment'] == Assortment)]
df_store.head()
```

Out[5]:

	Store	Date	CW	Month	Year	DayOfWeek	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	StateHoliday	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	N
112725	836	2013-01-06	1	1	2013	6	17937	7.171931	4484.250000	2501	625.250000	4	0	0	a	1	2	1	
112726	836	2013-01-13	2	1	2013	6	38907	8.141243	6484.500000	4779	796.500000	6	5	1	0	0	0	0	
112727	836	2013-01-20	3	1	2013	6	27071	7.213163	4511.833333	3753	625.500000	6	0	0	0	0	0	0	
112728	836	2013-01-27	4	1	2013	6	36644	8.402660	6107.333333	4361	726.833333	6	5	1	0	0	0	0	
112729	836	2013-02-03	5	2	2013	6	29513	7.618224	4918.833333	3874	645.666667	6	0	0	0	0	0	0	

In [6]:
```
# Create a mean dataframe out of all stores with the same StoreType and Assortment
columns_to_average = ['Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers',
                      'CustomersPerOpenDay', 'IsPromo', 'Promo2Active',
                      'SchoolHoliday', 'NumStateHoliday']

# Group df_store_compare by 'Date' and calculate the mean of the specified columns
mean_df = df_store_compare.groupby('Date')[columns_to_average].mean().reset_index()
mean_df.head()
```

Out[6]:

	Date	Sales	SalesPerOpenDay	SalesPerCustomer	Customers	CustomersPerOpenDay	IsPromo	Promo2Active	SchoolHoliday	NumStateHoliday
0	2013-01-06	22175.478947	5543.869737	8.151861	2782.386842	695.596711	0.0	0.178947	3.997368	1.286842
1	2013-01-13	43826.200000	7304.366667	9.168636	4897.513158	816.252193	1.0	0.178947	0.355263	0.000000
2	2013-01-20	30347.055263	5057.842544	7.941036	3904.978947	650.829825	0.0	0.178947	0.000000	0.000000
3	2013-01-27	40892.684211	6815.447368	9.130798	4600.573684	766.762281	1.0	0.178947	0.000000	0.000000
4	2013-02-03	33601.405263	5600.234211	8.276263	4156.610526	692.768421	0.0	0.092105	0.139474	0.000000

Store Report - Weekly Basis

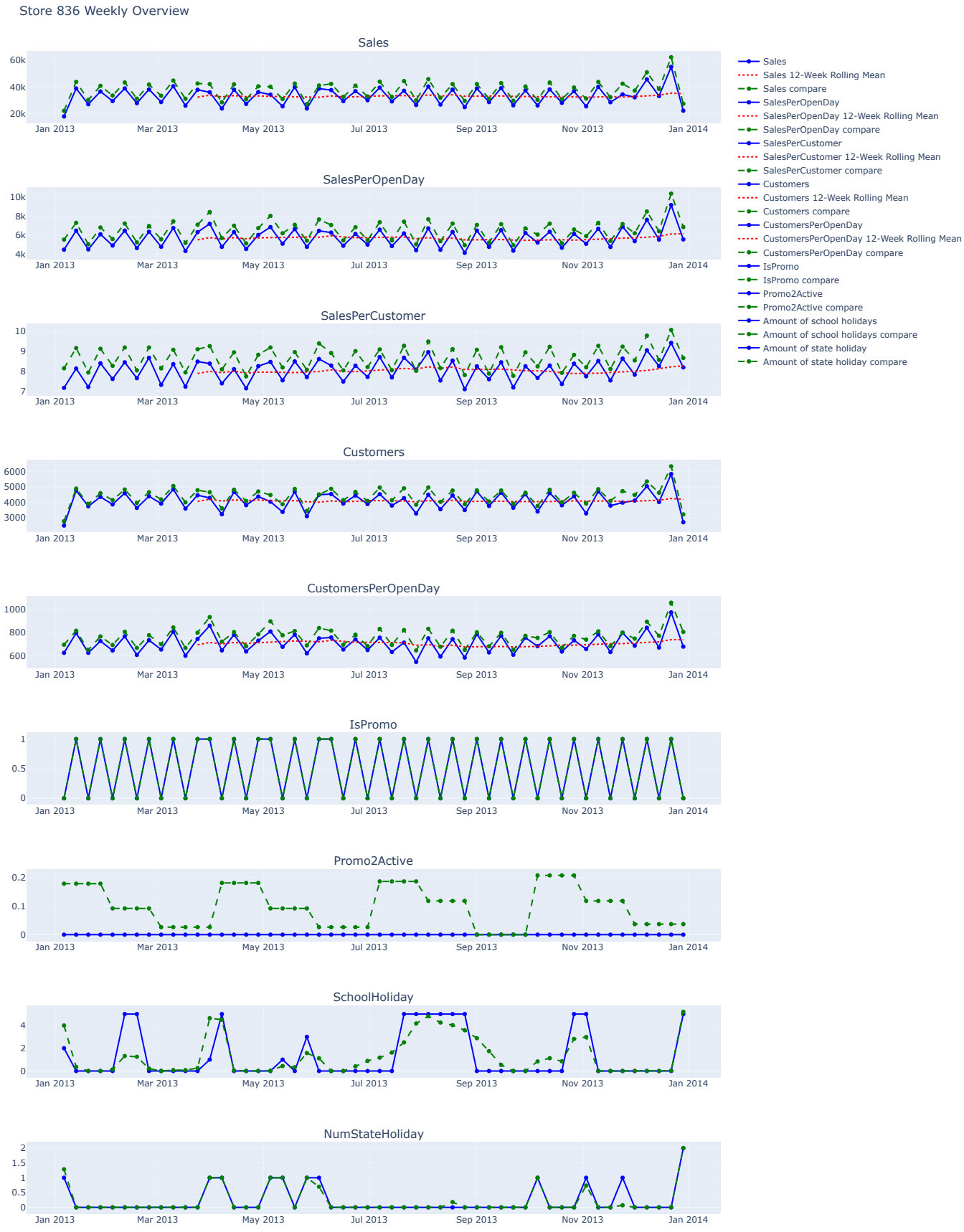
In [7]:
```
# Print report header

print(f"Period: {StartDate.strftime('%Y-%m-%d')} to {EndDate.strftime('%Y-%m-%d')}")
print(f"Store ID: {StoreId}")
print(f"Store Type: {StoreType}")
print(f"Assortment: {Assortment}")

if df_store['IsCompetition'].iloc[0] == 1:
        print(f"Competition Distance: {int(df_store['CompetitionDistance'].iloc[0])}")
        CompetitionOpenSince = datetime(int(df_store['CompetitionOpenSinceYear'].iloc[0]), int(df_store['CompetitionOpenSinceMonth'].iloc[0]), 1)
        print(f"Competition Open Since: {CompetitionOpenSince.strftime('%Y-%m-%d')}")
else:
        print("Competition: None")

if df_store['Promo2'].iloc[0] == 1:
        Promo2Since = datetime.strptime(f"{int(df_store['Promo2SinceYear'].iloc[0])} {int(df_store['Promo2SinceWeek'].iloc[0])} 1", '%G %V %u').date()
        print(f"Promo2Since: {Promo2Since.strftime('%Y-%m-%d')}")
        print(f"PromoInterval: {df_store['PromoInterval'].iloc[0]}")
else:
        print("Promo2: None")
```

```
Period: 2013-01-01 to 2013-12-31
Store ID: 836
Store Type: a
Assortment: a
Competition Distance: 2720
Competition Open Since: 2012-09-01
Promo2: None
```

In [8]:

```python
# plot the weekly overview

rolling_mean_window = 12

# Suppress all warnings in the current cell
with warnings.catch_warnings():
        warnings.simplefilter("ignore")
        # Calculate the rolling mean for the last X weeks for each column
        df_store['Sales_rolling_mean'] = df_store['Sales'].rolling(window=rolling_mean_window).mean()
        df_store['SalesPerOpenDay_rolling_mean'] = df_store['SalesPerOpenDay'].rolling(window=rolling_mean_window).mean()
        df_store['SalesPerCustomer_rolling_mean'] = df_store['SalesPerCustomer'].rolling(window=rolling_mean_window).mean()
        df_store['Customers_rolling_mean'] = df_store['Customers'].rolling(window=rolling_mean_window).mean()
        df_store['CustomersPerOpenDay_rolling_mean'] = df_store['CustomersPerOpenDay'].rolling(window=rolling_mean_window).mean()

# Create a subplot grid
fig = make_subplots(rows=9, cols=1, subplot_titles=('Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers', 'CustomersPerOpenDay', 'IsPromo', 'Promo2Active', 'SchoolHoliday', 'NumStateHoliday'))

# Add the original and rolling mean plots to the subplot grid
# Sales
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Sales'], mode='lines+markers', name='Sales', line=dict(color='blue')), row=1, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Sales_rolling_mean'], mode='lines', name=f'Sales {rolling_mean_window}-Week Rolling Mean', line=dict(dash='dot', color='red')), row=1, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['Sales'], mode='lines+markers', name='Sales compare', line=dict(dash='dash', color='green')), row=1, col=1)

# SalesPerOpenDay
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerOpenDay'], mode='lines+markers', name='SalesPerOpenDay', line=dict(color='blue')), row=2, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerOpenDay_rolling_mean'], mode='lines', name=f'SalesPerOpenDay {rolling_mean_window}-Week Rolling Mean', line=dict(dash='dot', color='red')),
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['SalesPerOpenDay'], mode='lines+markers', name='SalesPerOpenDay compare', line=dict(dash='dash', color='green')), row=2, col=1)

# SalesPerCustomer
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerCustomer'], mode='lines+markers', name='SalesPerCustomer', line=dict(color='blue')), row=3, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerCustomer_rolling_mean'], mode='lines', name=f'SalesPerCustomer {rolling_mean_window}-Week Rolling Mean', line=dict(dash='dot', color='red')
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['SalesPerCustomer'], mode='lines+markers', name='SalesPerCustomer compare', line=dict(dash='dash', color='green')), row=3, col=1)

# Customers
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Customers'], mode='lines+markers', name='Customers', line=dict(color='blue')), row=4, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Customers_rolling_mean'], mode='lines', name=f'Customers {rolling_mean_window}-Week Rolling Mean', line=dict(dash='dot', color='red')), row=4, col=
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['Customers'], mode='lines+markers', name='Customers compare', line=dict(dash='dash', color='green')), row=4, col=1)

# CustomersPerOpenDay
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['CustomersPerOpenDay'], mode='lines+markers', name='CustomersPerOpenDay', line=dict(color='blue')), row=5, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['CustomersPerOpenDay_rolling_mean'], mode='lines', name=f'CustomersPerOpenDay {rolling_mean_window}-Week Rolling Mean', line=dict(dash='dot', color=
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['CustomersPerOpenDay'], mode='lines+markers', name='CustomersPerOpenDay compare', line=dict(dash='dash', color='green')), row=5, col=1)

# IsPromo
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['IsPromo'], mode='lines+markers', name='IsPromo', line=dict(color='blue')), row=6, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['IsPromo'], mode='lines+markers', name='IsPromo compare', line=dict(dash='dash', color='green')), row=6, col=1)

# Promo2Active
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Promo2Active'], mode='lines+markers', name='Promo2Active', line=dict(color='blue')), row=7, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['Promo2Active'], mode='lines+markers', name='Promo2Active compare', line=dict(dash='dash', color='green')), row=7, col=1)

# SchoolHoliday
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SchoolHoliday'], mode='lines+markers', name='Amount of school holidays', line=dict(color='blue')), row=8, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['SchoolHoliday'], mode='lines+markers', name='Amount of school holidays compare', line=dict(dash='dash', color='green')), row=8, col=1)

# NumStateHoliday
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['NumStateHoliday'], mode='lines+markers', name='Amount of state holiday', line=dict(color='blue')), row=9, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['NumStateHoliday'], mode='lines+markers', name='Amount of state holiday compare', line=dict(dash='dash', color='green')), row=9, col=1)

# Update layout (set layout properties and show the legend)
fig.update_layout(height=1800, width=1400, title_text=f"Store {StoreId} Weekly Overview", showlegend=True)

# Show the figure
print( '\033[1m' + 'Info: The compare numbers are the average of the same store type and assortment'+ '\033[0m')
fig.show()
```

```
Info: The compare numbers are the average of the same store type and assortment
```



Store Report - Monthly Basis

In [9]:

```
# Prepare monthly dataframe

monthly_summary_df_store = df_store.groupby(['Year', 'Month']).agg(
    {
        'Sales': 'sum',
        'SalesPerOpenDay': 'mean',
        'SalesPerCustomer': 'mean',
        'Customers': 'sum',
        'CustomersPerOpenDay': 'mean',
        'IsPromo': 'sum',
        'Promo2Active': 'sum',
        'SchoolHoliday': 'sum',
        'NumStateHoliday': 'sum'
    }
).reset_index()

# Add a new 'Date' column representing the first day of each month
monthly_summary_df_store['Date'] = pd.to_datetime(monthly_summary_df_store['Year'].astype(str) + '-' + monthly_summary_df_store['Month'].astype(str) + '-01')

# Reorder columns
monthly_summary_df_store = monthly_summary_df_store[['Date', 'Year', 'Month', 'Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers', 'CustomersPerOpenDay', 'IsPromo', 'Promo2Active', 'SchoolHoliday'
```

```
monthly_summary_df_store.head()
```

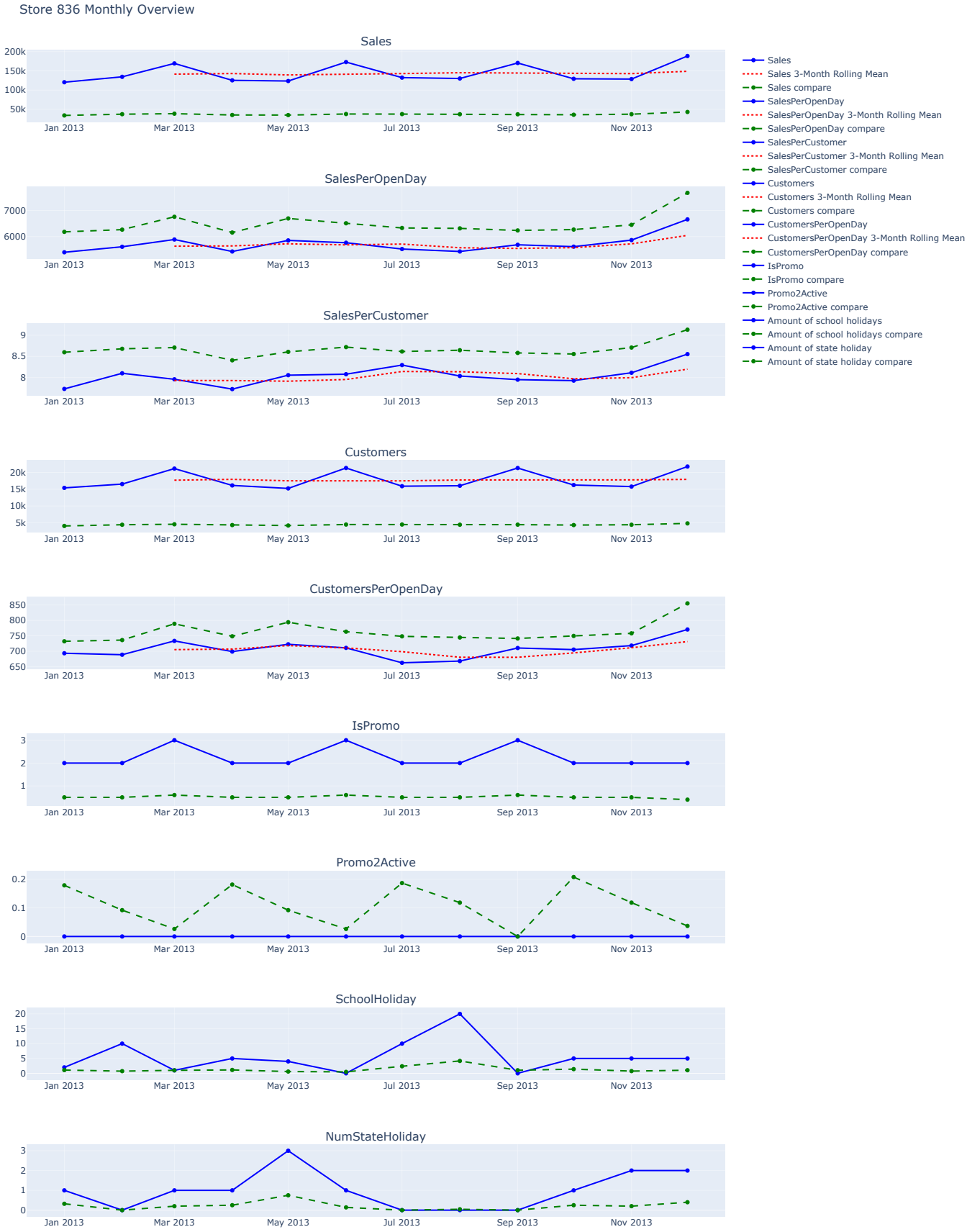
Out[9]:

	Date	Year	Month	Sales	SalesPerOpenDay	SalesPerCustomer	Customers	CustomersPerOpenDay	IsPromo	Promo2Active	SchoolHoliday	NumStateHoliday
0	2013-01-01	2013	1	120559	5396.979167	7.732249	15394	693.520833	2	0	2	1
1	2013-02-01	2013	2	134587	5607.791667	8.101283	16531	688.791667	2	0	10	0
2	2013-03-01	2013	3	169388	5886.620000	7.959337	21149	733.613333	3	0	1	1
3	2013-04-01	2013	4	125464	5426.933333	7.727423	16130	699.025000	2	0	5	1
4	2013-05-01	2013	5	123694	5852.008333	8.056547	15235	722.600000	2	0	4	3

In [10]:

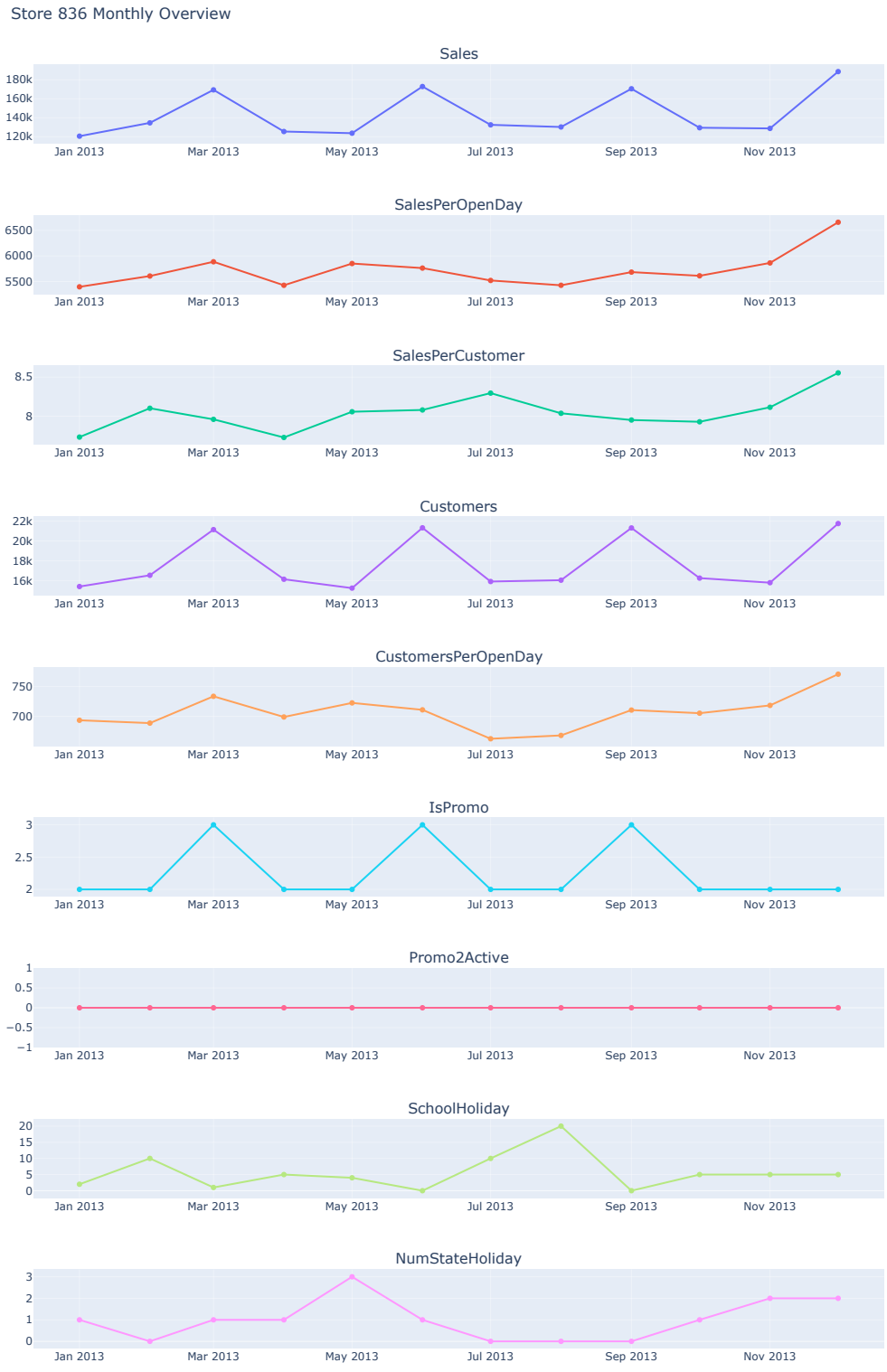
```
# plot the monthly overview with comparison

df_store = monthly_summary_df_store

df_store_compare = df[(df['Store'] != StoreId) & (df['Date'] >= StartDate) & (df['Date'] <= EndDate) & (df['StoreType'] == StoreType) & (df['Assortment'] == Assortment)]


monthly_summary_df_store_all = df_store_compare.groupby(['Year', 'Month']).agg(
    {
        'Sales': 'mean',
        'SalesPerOpenDay': 'mean',
        'SalesPerCustomer': 'mean',
        'Customers': 'mean',
        'CustomersPerOpenDay': 'mean',
        'IsPromo': 'mean',
        'Promo2Active': 'mean',
        'SchoolHoliday': 'mean',
        'NumStateHoliday': 'mean'
    }
).reset_index()

# Add a new 'Date' column representing the first day of each month
monthly_summary_df_store_all['Date'] = pd.to_datetime(monthly_summary_df_store_all['Year'].astype(str) + '-' + monthly_summary_df_store_all['Month'].astype(str) + '-01')

# Reorder columns
monthly_summary_df_store_all = monthly_summary_df_store_all[['Date', 'Year', 'Month', 'Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers', 'CustomersPerOpenDay', 'IsPromo', 'Promo2Active', 'School
mean_df = monthly_summary_df_store_all


rolling_mean_window = 3

# Suppress all warnings in the current cell
with warnings.catch_warnings():
        warnings.simplefilter("ignore")
        # Calculate the rolling mean for the last X Months for each column
        df_store['Sales_rolling_mean'] = df_store['Sales'].rolling(window=rolling_mean_window).mean()
        df_store['SalesPerOpenDay_rolling_mean'] = df_store['SalesPerOpenDay'].rolling(window=rolling_mean_window).mean()
        df_store['SalesPerCustomer_rolling_mean'] = df_store['SalesPerCustomer'].rolling(window=rolling_mean_window).mean()
        df_store['Customers_rolling_mean'] = df_store['Customers'].rolling(window=rolling_mean_window).mean()
        df_store['CustomersPerOpenDay_rolling_mean'] = df_store['CustomersPerOpenDay'].rolling(window=rolling_mean_window).mean()

# Create a subplot grid
fig = make_subplots(rows=9, cols=1, subplot_titles=('Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers', 'CustomersPerOpenDay', 'IsPromo', 'Promo2Active', 'SchoolHoliday', 'NumStateHoliday'))

# Add the original and rolling mean plots to the subplot grid
# Sales
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Sales'], mode='lines+markers', name='Sales', line=dict(color='blue')), row=1, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Sales_rolling_mean'], mode='lines', name=f'Sales {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot', color='red')), row=1, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['Sales'], mode='lines+markers', name='Sales compare', line=dict(dash='dash', color='green')), row=1, col=1)

# SalesPerOpenDay
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerOpenDay'], mode='lines+markers', name='SalesPerOpenDay', line=dict(color='blue')), row=2, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerOpenDay_rolling_mean'], mode='lines', name=f'SalesPerOpenDay {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot', color='red'))
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['SalesPerOpenDay'], mode='lines+markers', name='SalesPerOpenDay compare', line=dict(dash='dash', color='green')), row=2, col=1)

# SalesPerCustomer
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerCustomer'], mode='lines+markers', name='SalesPerCustomer', line=dict(color='blue')), row=3, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SalesPerCustomer_rolling_mean'], mode='lines', name=f'SalesPerCustomer {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot', color='red'
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['SalesPerCustomer'], mode='lines+markers', name='SalesPerCustomer compare', line=dict(dash='dash', color='green')), row=3, col=1)

# Customers
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Customers'], mode='lines+markers', name='Customers', line=dict(color='blue')), row=4, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Customers_rolling_mean'], mode='lines', name=f'Customers {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot', color='red')), row=4, col
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['Customers'], mode='lines+markers', name='Customers compare', line=dict(dash='dash', color='green')), row=4, col=1)

# CustomersPerOpenDay
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['CustomersPerOpenDay'], mode='lines+markers', name='CustomersPerOpenDay', line=dict(color='blue')), row=5, col=1)
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['CustomersPerOpenDay_rolling_mean'], mode='lines', name=f'CustomersPerOpenDay {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot', color
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['CustomersPerOpenDay'], mode='lines+markers', name='CustomersPerOpenDay compare', line=dict(dash='dash', color='green')), row=5, col=1)

# IsPromo
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['IsPromo'], mode='lines+markers', name='IsPromo', line=dict(color='blue')), row=6, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['IsPromo'], mode='lines+markers', name='IsPromo compare', line=dict(dash='dash', color='green')), row=6, col=1)

# Promo2Active
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['Promo2Active'], mode='lines+markers', name='Promo2Active', line=dict(color='blue')), row=7, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['Promo2Active'], mode='lines+markers', name='Promo2Active compare', line=dict(dash='dash', color='green')), row=7, col=1)

# SchoolHoliday
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['SchoolHoliday'], mode='lines+markers', name='Amount of school holidays', line=dict(color='blue')), row=8, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['SchoolHoliday'], mode='lines+markers', name='Amount of school holidays compare', line=dict(dash='dash', color='green')), row=8, col=1)

# NumStateHoliday
fig.add_trace(go.Scatter(x=df_store['Date'], y=df_store['NumStateHoliday'], mode='lines+markers', name='Amount of state holiday', line=dict(color='blue')), row=9, col=1)
fig.add_trace(go.Scatter(x=mean_df['Date'], y=mean_df['NumStateHoliday'], mode='lines+markers', name='Amount of state holiday compare', line=dict(dash='dash', color='green')), row=9, col=1)

# Update Layout (set layout properties and show the legend)
fig.update_layout(height=1800, width=1400, title_text=f"Store {StoreId} Monthly Overview", showlegend=True)

# Show the figure
print( '\033[1m' + 'Info: The compare numbers are the average of the same store type and assortment'+ '\033[0m')
fig.show()
```

Info: The compare numbers are the average of the same store type and assortment



In [11]:

```
# plot the monthly overview

# Create a subplot grid
fig = make_subplots(rows=9, cols=1, subplot_titles=('Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers', 'CustomersPerOpenDay', 'IsPromo', 'Promo2Active', 'SchoolHoliday', 'NumStateHoliday'))

# Add each plot to the subplot grid with specific names
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['Sales'], mode='lines+markers', name='Sales'), row=1, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['SalesPerOpenDay'], mode='lines+markers', name='SalesPerOpenDay'), row=2, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['SalesPerCustomer'], mode='lines+markers', name='SalesPerCustomer'), row=3, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['Customers'], mode='lines+markers', name='Customers'), row=4, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['CustomersPerOpenDay'], mode='lines+markers', name='CustomersPerOpenDay'), row=5, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['IsPromo'], mode='lines+markers', name='IsPromo'), row=6, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['Promo2Active'], mode='lines+markers', name='Promo2Active'), row=7, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['SchoolHoliday'], mode='lines+markers', name='Amount of school holidays'), row=8, col=1)
fig.add_trace(go.Scatter(x=monthly_summary_df_store['Date'], y=monthly_summary_df_store['NumStateHoliday'], mode='lines+markers', name='Amount of state holiday'), row=9, col=1)

# Update Layout (set layout properties and hide the legend)
fig.update_layout(height=1600, width=1100, title_text=f"Store {StoreId} Monthly Overview", showlegend=False)

# Show the figure
fig.show()
```



Overall Report

Definition of Information and KPIs to be reported

Information

- Period
- Number of Stores
- Number of Stores in each Store Type
- Number of Stores in each Assortment
- Number of StoreType in each Assortment
- Number of Assortment in each StoreType
- Number of Stores with Promo
- Number of Stores without Promo
- Number of Stores with Promo2
- Number of Stores without Promo2
- Number of Stores with Competition

KPIs

- Line plot with sales
- Line plot with SalesPerOpenDay
- Line plot with sales per customer
- Line plot with customers
- Line plot with CustomersPerOpenDay
- Line plot with IsPromo

- Line plot with Promo2Active
- Line plot with SchoolHoliday
- Line plot with NumStateHoliday
- Rolling mean

Input Fields

- StartDate
- EndDate
- (StoreType)
- (Assortment)

Input and pre setup

In [12]:
```
# Input fields to define the store and the date range

StartDate = "2013-01-01"
EndDate = "2013-12-31"
StartDate = datetime.strptime(StartDate, "%Y-%m-%d")
EndDate = datetime.strptime(EndDate, "%Y-%m-%d")
```

In [13]:
```
df_all_stores = df[(df['Date'] >= StartDate) & (df['Date'] <= EndDate)]
```

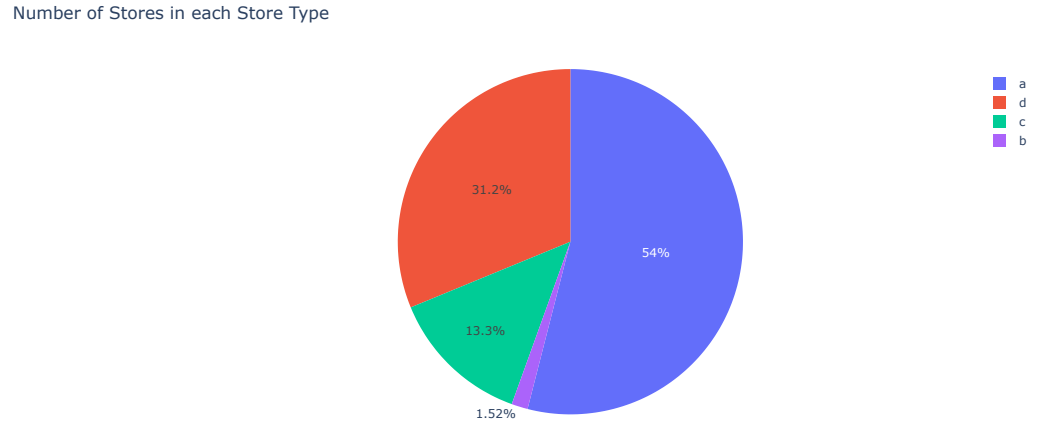
In [14]:
```
monthly_summary = df_all_stores.groupby(['Year', 'Month']).agg(
    {
        'Sales': 'sum',
        'SalesPerOpenDay': 'mean',
        'SalesPerCustomer': 'mean',
        'Customers': 'sum',
        'CustomersPerOpenDay': 'mean',
        'IsPromo': 'mean',
        'Promo2Active': 'sum',
        'SchoolHoliday': 'mean',
        'NumStateHoliday': 'mean'
    }
).reset_index()

# Add a new 'Date' column representing the first day of each month
monthly_summary['Date'] = pd.to_datetime(monthly_summary['Year'].astype(str) + '-' + monthly_summary['Month'].astype(str) + '-01')

# Reorder columns
monthly_summary = monthly_summary[['Date', 'Year', 'Month', 'Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers', 'CustomersPerOpenDay', 'IsPromo', 'Promo2Active', 'SchoolHoliday', 'NumStateHoliday

monthly_summary.head()
```

Out[14]:

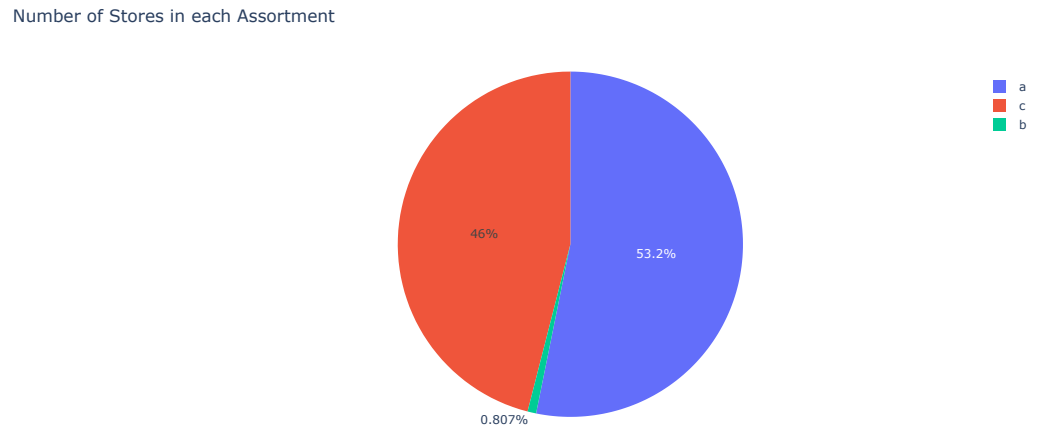
	Date	Year	Month	Sales	SalesPerOpenDay	SalesPerCustomer	Customers	CustomersPerOpenDay	IsPromo	Promo2Active	SchoolHoliday	NumStateHoliday
0	2013-01-01	2013	1	155822491	6322.836886	9.273179	17471528	709.885841	0.5	912	1.141928	0.319058
1	2013-02-01	2013	2	171439913	6427.544368	9.342311	19190788	717.751004	0.5	392	0.604933	0.000000
2	2013-03-01	2013	3	225584447	7033.858730	9.451784	24932037	773.210937	0.6	339	1.020448	0.200000
3	2013-04-01	2013	4	162826157	6324.883565	9.090353	18786879	728.266352	0.5	952	1.192601	0.250000
4	2013-05-01	2013	5	164459743	6956.306618	9.295173	18548458	781.232953	0.5	408	0.562556	0.750000

In [15]:
```
# Print report header

#- Period
print(f"Period: {StartDate.strftime('%Y-%m-%d')} to {EndDate.strftime('%Y-%m-%d')}"     )

#- Number of Stores
print(f"Number of Stores: {df_all_stores['Store'].nunique()}")

#- Number of Stores in each Store Type
print("Number of Stores in each Store Type:", df_all_stores.groupby('StoreType')['Store'].nunique())
store_counts = df_all_stores.groupby('StoreType')['Store'].nunique().reset_index(name='NumberOfStores')
fig = px.pie(store_counts, names='StoreType', values='NumberOfStores', title='Number of Stores in each Store Type')
fig.update_traces(hovertemplate='%{label}: %{value} (<b>%{percent}</b>)')
fig.show()

#- Number of Stores in each Assortment
print("Number of Stores in each Assortment:", df_all_stores.groupby('Assortment')['Store'].nunique())
assortment_counts = df_all_stores.groupby('Assortment')['Store'].nunique().reset_index(name='NumberOfStores')
fig = px.pie(assortment_counts, names='Assortment', values='NumberOfStores', title='Number of Stores in each Assortment')
fig.update_traces(hovertemplate='%{label}: %{value} (<b>%{percent}</b>)')
fig.show()

#- Number of Stores with Promo2
print(f"Number of Stores with Promo2: {df_all_stores.groupby('Promo2')['Store'].nunique().loc[1]}")

#- Number of Stores without Promo2
print(f"Number of Stores without Promo2: {df_all_stores.groupby('Promo2')['Store'].nunique().loc[0]}")

#- Number of Stores with Competition
print(f"Number of Stores with Competition: {df_all_stores.groupby('IsCompetition')['Store'].nunique().loc[1]}")

#- Number of Stores without Competition
print(f"Number of Stores without Competition: {df_all_stores.groupby('IsCompetition')['Store'].nunique().loc[0]}")
```

```
Period: 2013-01-01 to 2013-12-31
Number of Stores: 1115
Number of Stores in each Store Type: StoreType
a    602
b     17
c    148
d    348
Name: Store, dtype: int64
```




```
Number of Stores in each Assortment: Assortment
a    593
b      9
c    513
Name: Store, dtype: int64
```




```
Number of Stores with Promo2: 571
Number of Stores without Promo2: 544
Number of Stores with Competition: 653
Number of Stores without Competition: 545
```

In [16]:

```python
# Plot monthly overview

rolling_mean_window = 3
import warnings

# Suppress all warnings in the current cell
with warnings.catch_warnings():
        warnings.simplefilter("ignore")
        # Calculate the rolling mean for the last X Months for each column
        monthly_summary['Sales_rolling_mean'] = monthly_summary['Sales'].rolling(window=rolling_mean_window).mean()
        monthly_summary['SalesPerOpenDay_rolling_mean'] = monthly_summary['SalesPerOpenDay'].rolling(window=rolling_mean_window).mean()
        monthly_summary['SalesPerCustomer_rolling_mean'] = monthly_summary['SalesPerCustomer'].rolling(window=rolling_mean_window).mean()
        monthly_summary['Customers_rolling_mean'] = monthly_summary['Customers'].rolling(window=rolling_mean_window).mean()
        monthly_summary['CustomersPerOpenDay_rolling_mean'] = monthly_summary['CustomersPerOpenDay'].rolling(window=rolling_mean_window).mean()

# Create a subplot grid
fig = make_subplots(rows=10, cols=1, subplot_titles=('Sales', 'SalesPerOpenDay', 'SalesPerCustomer', 'Customers', 'CustomersPerOpenDay', 'IsPromo', 'Promo2Active', 'SchoolHoliday', 'NumStateHoliday'))

# Add the original and rolling mean plots to the subplot grid
# Sales
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['Sales'], mode='lines+markers', name='Sales', line=dict(color='blue')), row=1, col=1)
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['Sales_rolling_mean'], mode='lines', name=f'Sales {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot', color='red')), row=

# SalesPerOpenDay
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['SalesPerOpenDay'], mode='lines+markers', name='SalesPerOpenDay', line=dict(color='blue')), row=2, col=1)
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['SalesPerOpenDay_rolling_mean'], mode='lines', name=f'SalesPerOpenDay {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot',

# SalesPerCustomer
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['SalesPerCustomer'], mode='lines+markers', name='SalesPerCustomer', line=dict(color='blue')), row=3, col=1)
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['SalesPerCustomer_rolling_mean'], mode='lines', name=f'SalesPerCustomer {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot

# Customers
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['Customers'], mode='lines+markers', name='Customers', line=dict(color='blue')), row=4, col=1)
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['Customers_rolling_mean'], mode='lines', name=f'Customers {rolling_mean_window}-Month Rolling Mean', line=dict(dash='dot', color='red'

# CustomersPerOpenDay
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['CustomersPerOpenDay'], mode='lines+markers', name='CustomersPerOpenDay', line=dict(color='blue')), row=5, col=1)
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['CustomersPerOpenDay_rolling_mean'], mode='lines', name=f'CustomersPerOpenDay {rolling_mean_window}-Month Rolling Mean', line=dict(das

# IsPromo
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['IsPromo'], mode='lines+markers', name='IsPromo', line=dict(color='blue')), row=6, col=1)

# Promo2Active
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['Promo2Active'], mode='lines+markers', name='Promo2Active', line=dict(color='blue')), row=7, col=1)

# SchoolHoliday
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['SchoolHoliday'], mode='lines+markers', name='Amount of school holidays', line=dict(color='blue')), row=8, col=1)

# NumStateHoliday
fig.add_trace(go.Scatter(x=monthly_summary['Date'], y=monthly_summary['NumStateHoliday'], mode='lines+markers', name='Amount of state holiday', line=dict(color='blue')), row=9, col=1)

# Update layout (set layout properties and show the legend)
fig.update_layout(height=1800, width=1400, title_text=f"Overall Monthly Overview", showlegend=True)

# Show the figure
fig.show()
```

Overall Monthly Overview