Capstone Projekt Rossmann

XDi - Certified Data Scientist

Christoph Gödecke

Intro and EDA

1. Project definition

1.1 Background

This project aims to investigate what insights the drugstore chain Rossmann can gain from its historical sales and advertising data, data on school vacations and public holidays and competitor data. The project will address the question of how this data can be used to optimize store operations and management with the aim of generating more sales. It will also determine how these data sets can be used to predict weekly sales (revenue) for each store with a sufficient level of accuracy. In addition, the possibility of creating automated reports at store level and an overall report is to be created.

Rossmann operates over 4,300 drugstores in 9 European countries. Currently, Rossmann store managers are tasked with preparing their weekly sales forecasts up to eight weeks in advance. Store sales results are influenced by many factors, including promotions, competitive intensity, school and national vacations, seasonal changes and location conditions. Because thousands of individual store managers create sales forecasts based on their individual circumstances, the accuracy of the results can vary widely. Therefore, the company's data science team is on a new mission to create a unified modeling methodology for store managers to predict weekly results with greater accuracy. The management team also needs an overall report with feasible or actionable strategies to understand the overall performance of all stores and find a way to optimize future sales performance (i.e. revenue). Finally, an individual store performance report needs to be provided to each store manager.

1.2 General task from the client

- Determine the key factors that influence sales and revenue
- Create strategies for the management to improve the business
- Build a standardized forecasting model which can predict the sales figures for the next eight weeks for each store.
- Generate a report with information on the overall performance of the 1115 stores as well as individual performance reports for each store.

1.3 Definition of use cases (background/business purpose)

- The analyses should help to better understand the current sales figures and the impact of e.g. promotions, product range, distance to competitors, etc. on sales.
- The resulting strategy and optimization proposals for increasing sales will then be discussed and implemented at management level.
- The standardized forecast model of sales figures per store for the next eight weeks should also be more accurate than the current individual forecasts of the individual store managers themselves, as well as saving corresponding time, monetary and capacity resources.
- The overall report as well as the individual store reports should also contribute to the mentioned objectives and also provide a quick and automated overview of the current performance.

1.4 Formulating hypotheses

1. promotions
   1.1 In weeks with promotions, sales, number of customers and sales per customer are higher.
2. promotion2
   2.1 In months with Promo2, sales, number of customers and sales per customer are higher.
   2.2 Since the store has been participating in Promo2, sales, number of customers and sales per customer are higher, even outside of Promo2 promotions.
3. school holidays
   3.1 Sales are lower in weeks with school vacations.
   3.2 In weeks with school vacations, sales per customer are lower.
4. public holidays
   4.1 Sales are lower in weeks with public holidays.
   4.2 In weeks with public holidays, more purchases are made on the other days.
5. day of the week
   5.1 Fewer purchases are made at the weekend.
6. assortment
   6.1 Stores with an extra assortment have higher sales than basic stores
   6.2 Stores with an extended assortment have higher sales than basic and extra stores
7. CompetitionDistance
   7.1 The closer the competitor, the lower the sales.
   7.2 Sales have been lower since the competitor opened.

1.5 Further interesting questions

1. promotions
   1.1 Is a promotion worthwhile in weeks with public holidays?
2. promotion2
3. SchoolHolidays
4. StateHolidays
5. day of the week
   5.1 Is a promotion on the weekend worthwhile?
6. Assortment
   6.1 What is the ratio of basic, extra and extended assortment?
   6.2 What is the ratio of basic, extra and extended assortment in the individual store types?
   6.3 What is the average turnover of the individual assortments?
   6.4 How many promotion weeks are there in the individual assortments?
   6.5 Do promotion weeks perform better in individual assortments than in others?
   7 CompetitionDistance
7. StoreType
   8.1 How many stores are there of which type?
   8.2 How many promotion weeks are there in the individual store types?
   8.3 Do promotion weeks perform better in individual store types than in others?

Data Analysis

Data preparing for EDA/ Row data preparation

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

import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.io as pio
pio.renderers.default = 'notebook_connected'

# matplotlib inline notebook




pd.set_option('display.max_columns', None)
```

Store Data

In [2]:
```
df_store = pd.read_csv("store.csv", dtype={0:int, 1:object, 2:object, 3:float, 4:float, 5:float, 6:int, 7:float, 8:float, 9:object})
```

In [3]:
```
df_store.sample(10)
```

Out[3]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
736	737	a	a	100.0	5.0	2007.0	1	31.0	2013.0	Jan,Apr,Jul,Oct
490	491	d	c	4680.0	NaN	NaN	1	22.0	2012.0	Mar,Jun,Sept,Dec
898	899	d	a	2590.0	NaN	NaN	1	13.0	2010.0	Jan,Apr,Jul,Oct
752	753	d	c	540.0	11.0	2012.0	1	35.0	2010.0	Mar,Jun,Sept,Dec
715	716	d	a	3200.0	1.0	2008.0	1	22.0	2011.0	Jan,Apr,Jul,Oct
681	682	b	a	150.0	9.0	2006.0	0	NaN	NaN	NaN
18	19	a	c	3240.0	NaN	NaN	1	22.0	2011.0	Mar,Jun,Sept,Dec
407	408	c	a	1560.0	NaN	NaN	1	45.0	2009.0	Feb,May,Aug,Nov
954	955	d	c	1690.0	7.0	2009.0	1	36.0	2013.0	Mar,Jun,Sept,Dec
1062	1063	a	c	6250.0	NaN	NaN	0	NaN	NaN	NaN

In [ ]:

In [4]:
```
df_store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Store                      1115 non-null   int32  
 1   StoreType                  1115 non-null   object 
 2   Assortment                 1115 non-null   object 
 3   CompetitionDistance        1112 non-null   float64
 4   CompetitionOpenSinceMonth  761 non-null    float64
 5   CompetitionOpenSinceYear   761 non-null    float64
 6   Promo2                     1115 non-null   int32  
 7   Promo2SinceWeek            571 non-null    float64
 8   Promo2SinceYear            571 non-null    float64
 9   PromoInterval              571 non-null    object 
dtypes: float64(5), int32(2), object(3)
memory usage: 78.5+ KB
```

In [ ]:

In [5]:
```
for column in df_store.select_dtypes(include=['object']).columns:
    print(f"Unique values in '{column}': {df_store[column].unique()}")
```

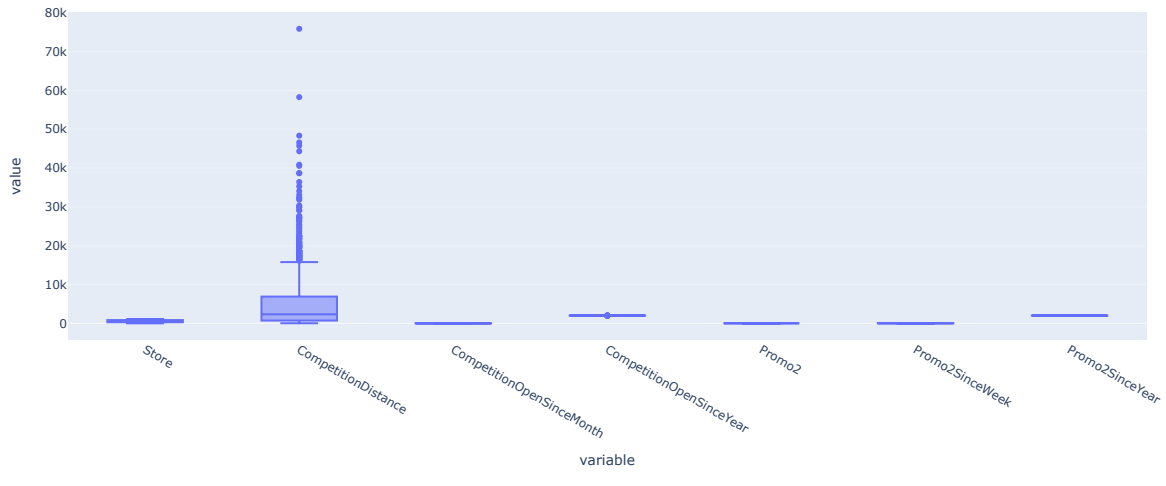
```
Unique values in 'StoreType': ['c' 'a' 'd' 'b']
Unique values in 'Assortment': ['a' 'c' 'b']
Unique values in 'PromoInterval': [nan 'Jan,Apr,Jul,Oct' 'Feb,May,Aug,Nov' 'Mar,Jun,Sept,Dec']
```

In [6]:
```
df_store.describe()
```

Out[6]:

	Store	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear
count	1115.00000	1112.000000	761.000000	761.000000	1115.000000	571.000000	571.000000
mean	558.00000	5404.901079	7.224704	2008.668857	0.512108	23.595447	2011.763573
std	322.01708	7663.174720	3.212348	6.195983	0.500078	14.141984	1.674935
min	1.00000	20.000000	1.000000	1900.000000	0.000000	1.000000	2009.000000
25%	279.50000	717.500000	4.000000	2006.000000	0.000000	13.000000	2011.000000
50%	558.00000	2325.000000	8.000000	2010.000000	1.000000	22.000000	2012.000000
75%	836.50000	6882.500000	10.000000	2013.000000	1.000000	37.000000	2013.000000
max	1115.00000	75860.000000	12.000000	2015.000000	1.000000	50.000000	2015.000000

In [7]:
```
px.box(df_store.select_dtypes(include=['number']))
```



Train Data

In [8]:
```
df_train = pd.read_csv("train.csv", parse_dates=[2], dtype={7:object})
```

In [9]:
```
# Group by store and aggregate to week
df_train.set_index('Date', inplace=True)
weekly_sales_by_store = df_train.groupby('Store').resample('W').agg({'Sales': 'sum', 'Customers': 'sum', 'Open': 'sum', 'Promo': 'sum', 'StateHoliday': 'sum', 'SchoolHoliday': 'sum'}).reset_index()
weekly_sales_by_store
```

Out[9]:

	Store	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	2013-01-06	19340	2500	4	0	a00000	6
1	1	2013-01-13	32952	3918	6	5	0000000	5
2	1	2013-01-20	25978	3417	6	0	0000000	0
3	1	2013-01-27	33071	3862	6	5	0000000	0
4	1	2013-02-03	28693	3561	6	0	0000000	0
...	...	...	...	...	...	...	...	...
150520	1115	2015-07-05	48130	2982	6	5	0000000	0
150521	1115	2015-07-12	36233	2531	6	0	0000000	0
150522	1115	2015-07-19	45927	3057	6	5	0000000	0
150523	1115	2015-07-26	35362	2504	6	0	0000000	0
150524	1115	2015-08-02	43551	2621	5	5	00000	5

150525 rows × 8 columns

In [10]:
```
# Check non mixing state holidays
print("Unique values in 'StateHoliday':", weekly_sales_by_store.StateHoliday.unique())
# Create NumStateHoliday with how many state holidays are in the week
weekly_sales_by_store['StateHoliday'] = weekly_sales_by_store['StateHoliday'].astype(str)
weekly_sales_by_store['NumStateHoliday'] = weekly_sales_by_store['StateHoliday'].apply(lambda x: x.count('a') + x.count('b') + x.count('c'))
print("Unique values in 'NumStateHoliday':", weekly_sales_by_store.NumStateHoliday.unique())
# Remove aggregated 0 and doubled labels
weekly_sales_by_store['StateHoliday'] = weekly_sales_by_store['StateHoliday'].str.extract('(a|b|c)', expand=False).fillna('0')
print("Unique values in 'StateHoliday':", weekly_sales_by_store.StateHoliday.unique())
```

```
Unique values in 'StateHoliday': ['a00000' '0000000' '0000b00' 'b000000' '00a0000' '000a000' 'a000000'
 '00cc000' '0000a00' '000cc00' '00000' '00000a0' 'a0000a' '0' 0 'a000'
 '0a00000']
Unique values in 'NumStateHoliday': [1 0 2]
Unique values in 'StateHoliday': ['a' '0' 'b' 'c']
```

In [11]:
```
# Create Month and Year columns
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Date') + 1, 'CW', weekly_sales_by_store['Date'].dt.isocalendar().week)
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Date') + 2, 'Month', weekly_sales_by_store['Date'].dt.month)
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Date') + 3, 'Year', weekly_sales_by_store['Date'].dt.year)
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Date') + 4, 'DayOfWeek', weekly_sales_by_store['Date'].dt.dayofweek)

# Create SalesPerCustomer column
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Sales') + 1, 'SalesPerCustomer', weekly_sales_by_store['Sales'] / weekly_sales_by_store['Customers'])
# Create sales per open day
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Sales') + 2, 'SalesPerOpenDay', weekly_sales_by_store['Sales'] / weekly_sales_by_store['Open'])
# Create customers per open day
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Customers') + 1, 'CustomersPerOpenDay', weekly_sales_by_store['Customers'] / weekly_sales_by_store['Open'])

weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('Promo') + 1, 'IsPromo', np.where(weekly_sales_by_store['Promo'] > 0, 1, 0))
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('StateHoliday') + 1, 'IsStateHoliday', np.where(weekly_sales_by_store['StateHoliday'] != '0', 1, 0))
weekly_sales_by_store.insert(weekly_sales_by_store.columns.get_loc('SchoolHoliday') + 1, 'IsSchoolHoliday', np.where(weekly_sales_by_store['SchoolHoliday'] > 0, 1, 0))
```

In [12]:
```
# Merge df_train with df_store
weekly_sales_with_store_info = pd.merge(weekly_sales_by_store, df_store, on='Store', how='left')
# Check if all sores were found
print("Amount of none found stores:", weekly_sales_with_store_info.Date.isna().sum())
```

```
Amount of none found stores: 0
```

In [13]:
```
# Create Promo2Member column too see if the store joined the Promo2 program
def is_promo2_active(row):
    if row['Promo2'] == 1:
        # Construct the starting date of Promo2
        promo2_start_date = datetime.fromisocalendar(int(row['Promo2SinceYear']), int(row['Promo2SinceWeek']), 1)
        # Check if the promo was active on the 'Date'
        return 1 if promo2_start_date <= row['Date'] else 0
    else:
        # If Promo2 is not active, return False
        return 0

# Apply the function across the DataFrame rows
weekly_sales_with_store_info['Promo2Member'] = weekly_sales_with_store_info.apply(is_promo2_active, axis=1)
```

In [14]:
```
# Create Promo2Active if current month is in PromoInterval
month_dict = {1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug', 9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'}

def is_promo2_active_month(row):
        if row['Promo2Member'] == 1:
                # Check if the promo was active on the 'Date'
                return 1 if month_dict[row['Month']] in row['PromoInterval'].split(',') else 0
        else:
                # If Promo2 is not active, return False
                return 0

# Apply the function across the DataFrame rows
weekly_sales_with_store_info['Promo2Active'] = weekly_sales_with_store_info.apply(is_promo2_active_month, axis=1)
```

In [15]:
```
# Create IsCompetition column

def is_competition_active(row):
        if row['CompetitionOpenSinceYear'] > 0:
                # Construct the starting date of competition
                competition_start_date = datetime(int(row['CompetitionOpenSinceYear']), int(row['CompetitionOpenSinceMonth']), 1)
                # Check if the competition was active on the 'Date'
                return 1 if competition_start_date <= row['Date'] else 0
        else:
                # If competition is not active, return False
                return 0

# Apply the function across the DataFrame rows
weekly_sales_with_store_info.insert(weekly_sales_with_store_info.columns.get_loc('CompetitionOpenSinceYear') + 1, 'IsCompetition', weekly_sales_with_store_info.apply(is_competition_active, axis=1))
```

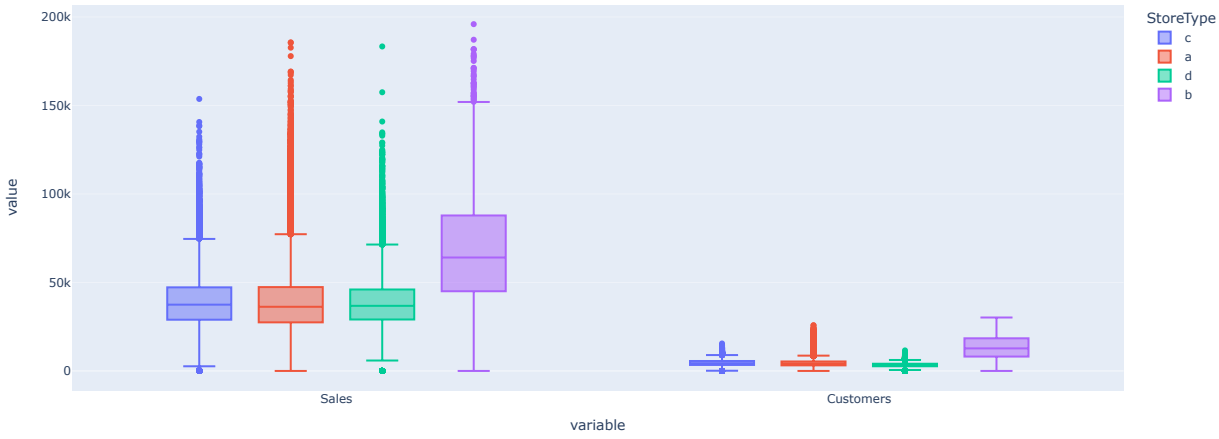
In [16]:
```
print(weekly_sales_with_store_info.info())
weekly_sales_with_store_info.sample(5)
```

```
<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  UInt32
 3   Month                      150525 non-null  int32
 4   Year                       150525 non-null  int32
 5   DayOfWeek                  150525 non-null  int32
 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  int32
 14  StateHoliday               150525 non-null  object
 15  IsStateHoliday             150525 non-null  int32
 16  SchoolHoliday              150525 non-null  int64
 17  IsSchoolHoliday            150525 non-null  int32
 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  int32
 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: UInt32(1), datetime64[ns](1), float64(8), int32(7), int64(10), object(4)
memory usage: 31.2+ MB
None
```

Out[16]:

	Store	Date	CW	Month	Year	DayOfWeek	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	StateHoliday	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	N
75405	559	2014-06-15	24	6	2014	6	24673	7.153668	4934.600000	3449	689.800000	5	0	0	a	1	0	0	
124870	925	2015-07-05	27	7	2015	6	46245	5.727644	7707.500000	8074	1345.666667	6	5	1	0	0	0	0	
8583	64	2014-07-06	27	7	2014	6	78434	14.573393	13072.333333	5382	897.000000	6	5	1	0	0	0	0	
130811	969	2015-07-12	28	7	2015	6	19326	5.114051	3221.000000	3779	629.833333	6	0	0	0	0	0	0	
67971	504	2014-04-13	15	4	2014	6	34430	6.635190	5738.333333	5189	864.833333	6	0	0	0	0	7	1	

In [17]:
```
used_df = weekly_sales_with_store_info[['StoreType', 'Sales', 'Customers']]
px.box(used_df, color='StoreType')
```



In [18]:
```
#save df to csv
weekly_sales_with_store_info.to_csv("weekly_sales_with_store_info.csv", index=False)
```

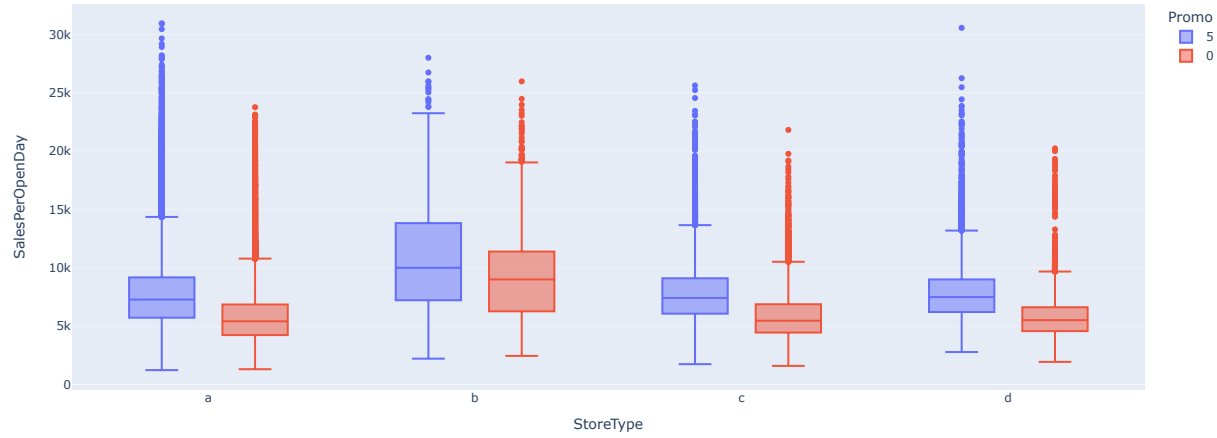
EDA

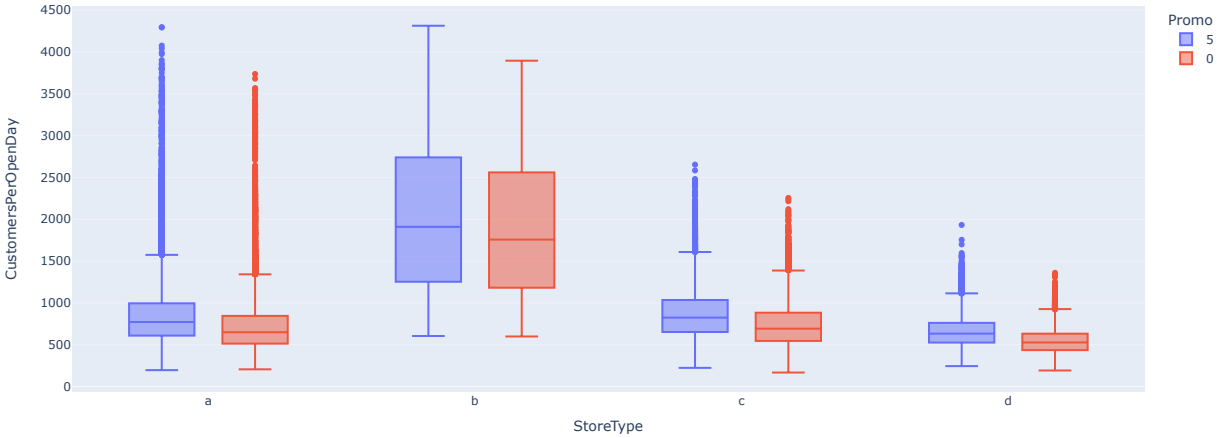
Analyse Hypotheses

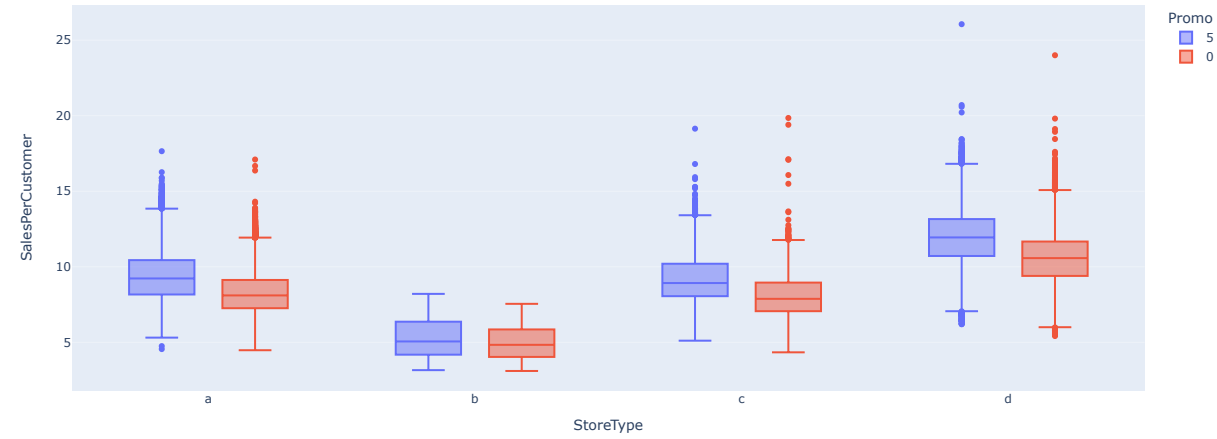
1. Promotions

1.1 In weeks with promotions, sales, number of customers and sales per customer are higher?

In [19]:
```
# To be compareble only weeks with >5 open days are used
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Open'] >= 5)]
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerOpenDay', color='Promo')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='CustomersPerOpenDay', color='Promo')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerCustomer', color='Promo')
fig.show()

median_values = used_df.groupby(['StoreType', 'Promo']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='Promo', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDayDiff%'
median_values_pivot
```



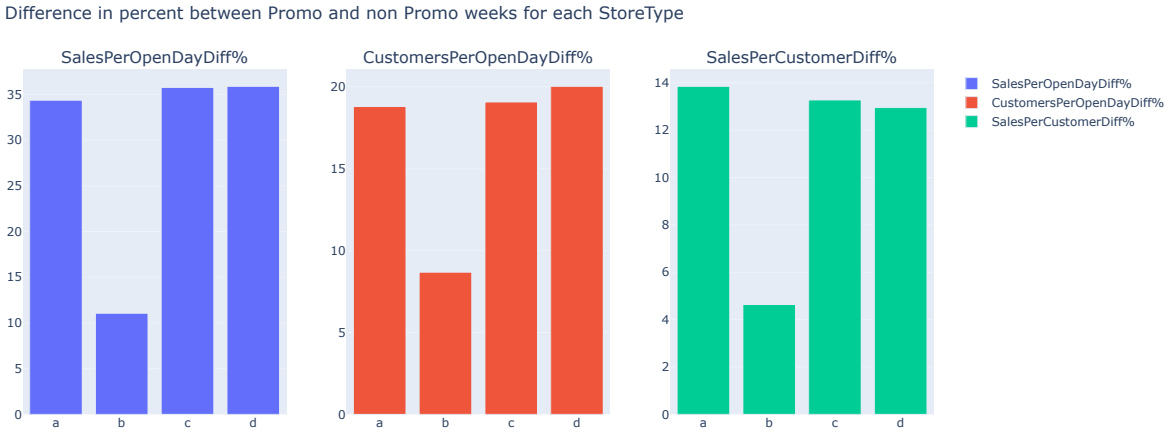




Out[19]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Promo	**0**	**5**	**5**	**0**	**5**	**5**	**0**	**5**	**5**
StoreType									
a	650.00	772.17	18.79	8.11	9.23	13.85	5412.50	7272.25	34.36
b	1757.07	1909.57	8.68	4.83	5.06	4.64	9000.79	9996.00	11.06
c	692.50	824.58	19.07	7.88	8.92	13.27	5459.50	7411.83	35.76
d	527.80	633.50	20.03	10.57	11.94	12.95	5510.50	7487.63	35.88

In [20]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between Promo and non Promo weeks for each StoreType")
fig.show()
```



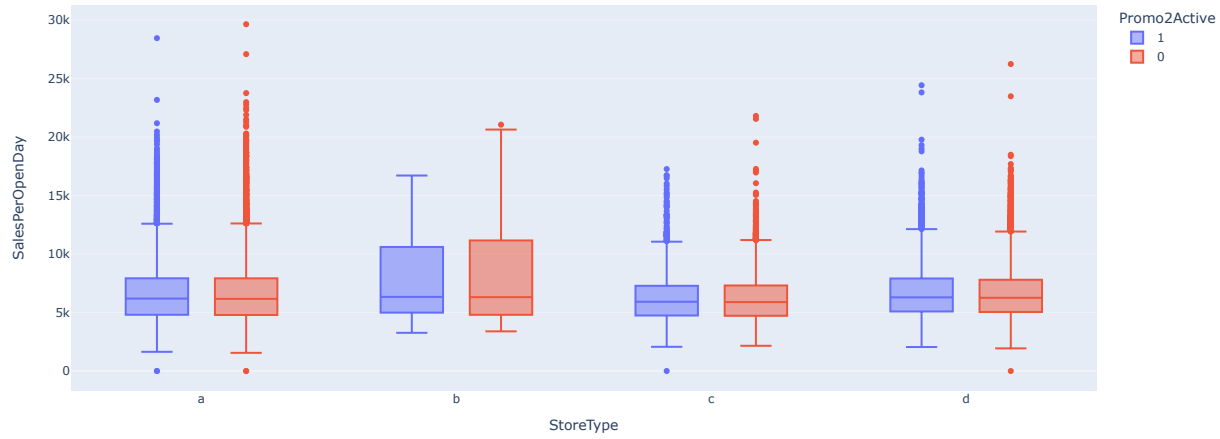
Result: Hypothesis is true. The sales, customers and sales per customer are higher in weeks with promotions.

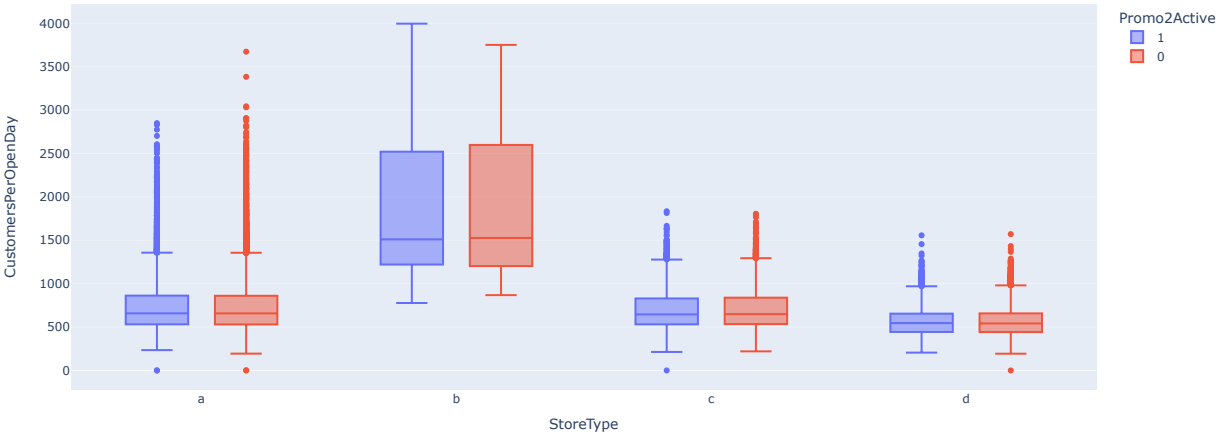
- Sales per day is ~ 35% higher
- Customers per day is ~ 19% higher
- Sales per customer is ~ 13% higher
- Stortype b is quite different. It just has 1/3 of the promotion impact.

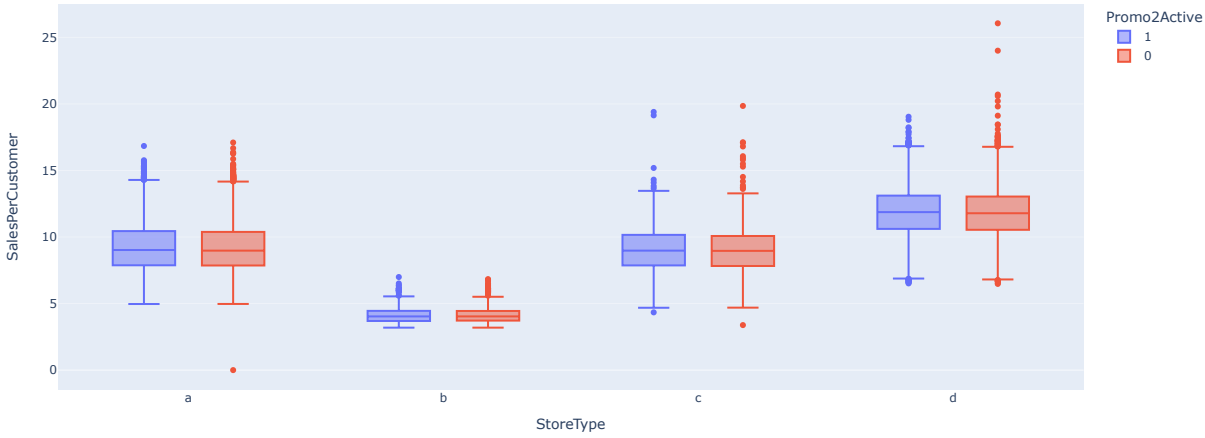
2. Promotion2

2.1 In months with Promo2, sales, number of customers and sales per customer are higher?

In [21]:
```
# To be compareble only weeks with >5 open days are used
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Promo2'] == 1) & (weekly_sales_with_store_info['Promo2Member'] == 1)]

fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerOpenDay', color='Promo2Active')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='CustomersPerOpenDay', color='Promo2Active')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerCustomer', color='Promo2Active')
fig.show()

median_values = used_df.groupby(['StoreType', 'Promo2Active']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='Promo2Active', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDa
median_values_pivot
```







Out[21]:

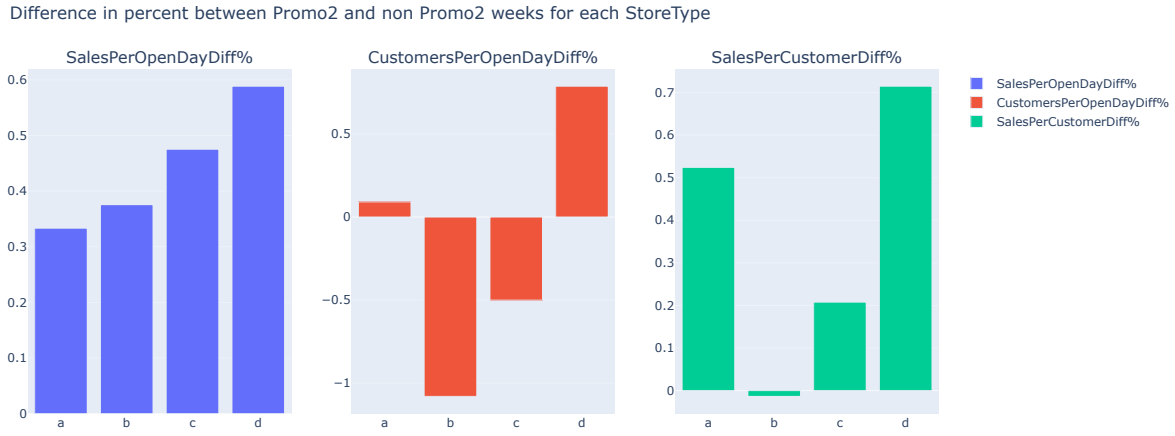
	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Promo2Active	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**
StoreType									
a	657.40	658.00	0.09	8.98	9.02	0.52	6170.17	6190.73	0.33
b	1526.86	1510.36	-1.08	4.03	4.03	-0.01	6312.50	6336.21	0.38
c	648.92	645.67	-0.50	8.96	8.98	0.21	5891.50	5919.50	0.48
d	541.40	545.67	0.79	11.79	11.87	0.71	6258.17	6295.00	0.59

In [22]:
```
# see amount of data
median_values_amount = used_df.groupby(['StoreType', 'Promo2Active']).agg({
    'SalesPerOpenDay': ['median', 'count'],
    'CustomersPerOpenDay': ['median', 'count'],
    'SalesPerCustomer': ['median', 'count']
}).reset_index()
median_values_amount
```

Out[22]:

	StoreType	Promo2Active	SalesPerOpenDay		CustomersPerOpenDay		SalesPerCustomer	
			median	count	median	count	median	count
0	a	0	6170.166667	20784	657.400000	20784	8.977074	20783
1	a	1	6190.733333	9924	658.000000	9924	9.024130	9922
2	b	0	6312.500000	404	1526.857143	404	4.034303	404
3	b	1	6336.214286	190	1510.357143	190	4.033754	190
4	c	0	5891.500000	6164	648.916667	6164	8.959205	6164
5	c	1	5919.500000	2855	645.666667	2855	8.977817	2854
6	d	0	6258.166667	15469	541.400000	15469	11.786457	15468
7	d	1	6295.000000	7323	545.666667	7323	11.870706	7323

In [23]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between Promo2 and non Promo2 weeks for each StoreType")
fig.show()
```



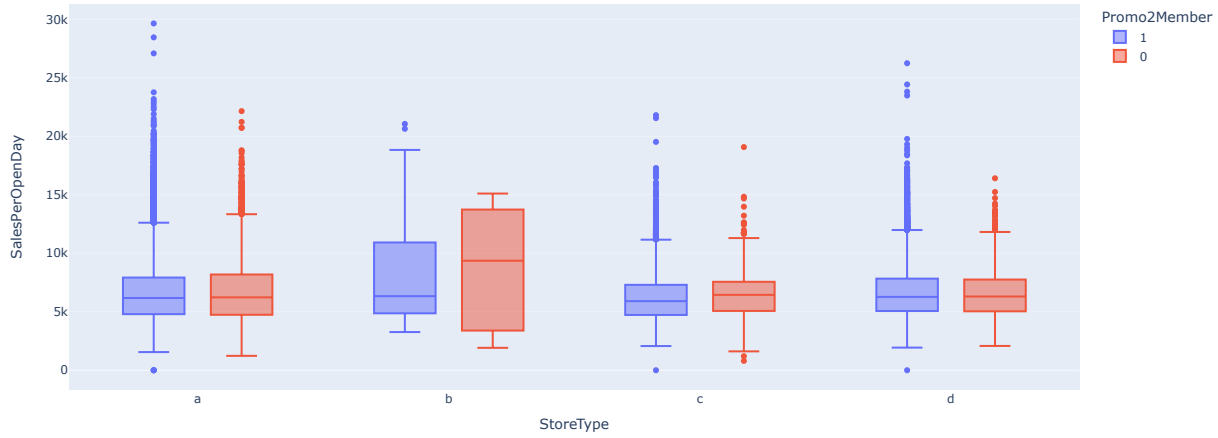
Result: Hypothesis is not really true.

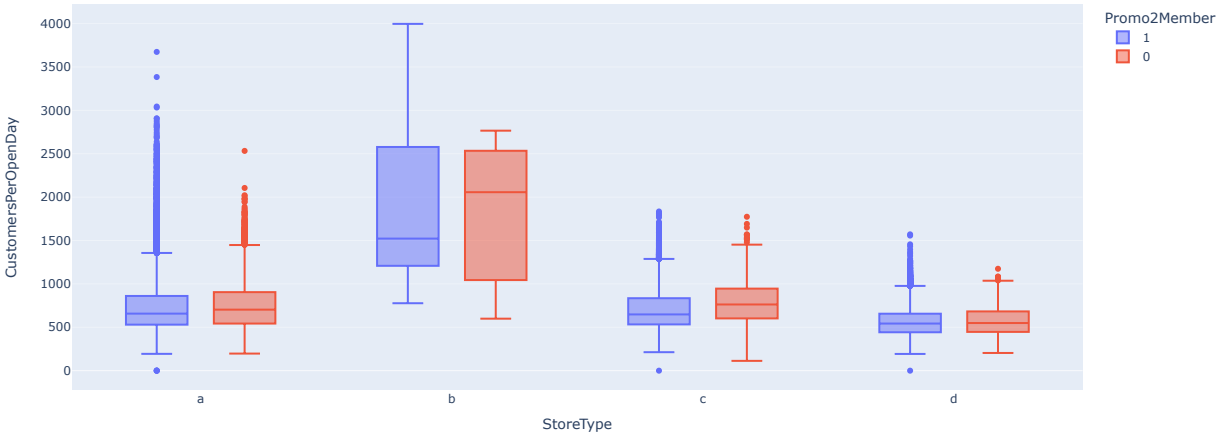
- Sales per day is ~ 0.4% higher
- Customers per day is the nearly the same as without Promo2
- Sales per customer is ~ 0.4% higher

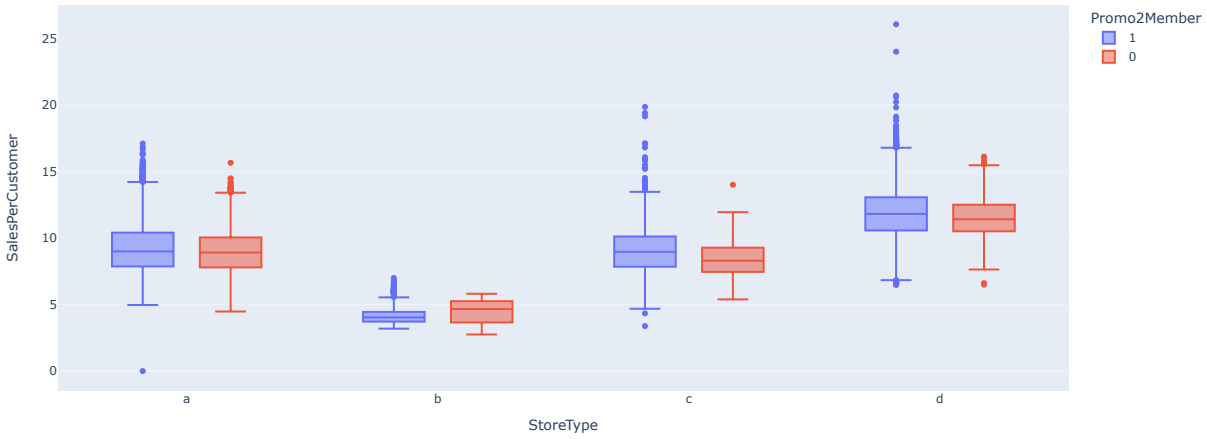
2.2 Since the store has been participating in Promo2, sales, number of customers and sales per customer are higher, even outside of Promo2 promotions?

In [24]:
```
# To be compareble only weeks with >5 open days are used
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Promo2'] == 1)]

fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerOpenDay', color='Promo2Member')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='CustomersPerOpenDay', color='Promo2Member')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerCustomer', color='Promo2Member')
```

```
fig.show()

median_values = used_df.groupby(['StoreType', 'Promo2Member']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='Promo2Member', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDa
median_values_pivot
```







Out[24]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Promo2Member	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**
StoreType									
a	703.33	657.50	-6.52	8.91	8.99	0.91	6230.50	6176.83	-0.86
b	2056.86	1522.21	-25.99	4.66	4.03	-13.49	9361.14	6331.43	-32.36
c	762.83	648.17	-15.03	8.30	8.96	8.04	6443.92	5904.00	-8.38
d	549.00	543.00	-1.09	11.41	11.81	3.50	6302.50	6271.17	-0.50

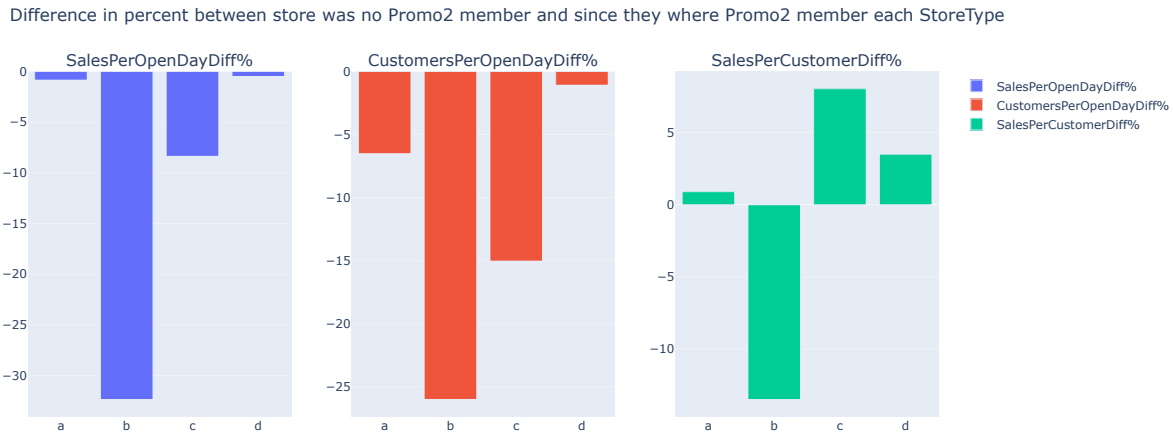
In [25]:

```
median_values_amount = used_df.groupby(['StoreType', 'Promo2Member']).agg({
    'SalesPerOpenDay': ['median', 'count'],
    'CustomersPerOpenDay': ['median', 'count'],
    'SalesPerCustomer': ['median', 'count']
}).reset_index()
median_values_amount
```

Out[25]:

StoreType	Promo2Member	SalesPerOpenDay		CustomersPerOpenDay		SalesPerCustomer	
		median	count	median	count	median	count
0	a	0	6230.500000	6301	703.333333	6301	8.910295	6301
1	a	1	6176.833333	30708	657.500000	30708	8.991142	30705
2	b	0	9361.142857	55	2056.857143	55	4.663068	55
3	b	1	6331.428571	594	1522.214286	594	4.034183	594
4	c	0	6443.916667	756	762.833333	756	8.296721	756
5	c	1	5904.000000	9019	648.166667	9019	8.963461	9018
6	d	0	6302.500000	2873	549.000000	2873	11.411541	2873
7	d	1	6271.166667	22792	543.000000	22792	11.811266	22791

In [26]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between store was no Promo2 member and since they where Promo2 member each StoreType")
fig.show()
```



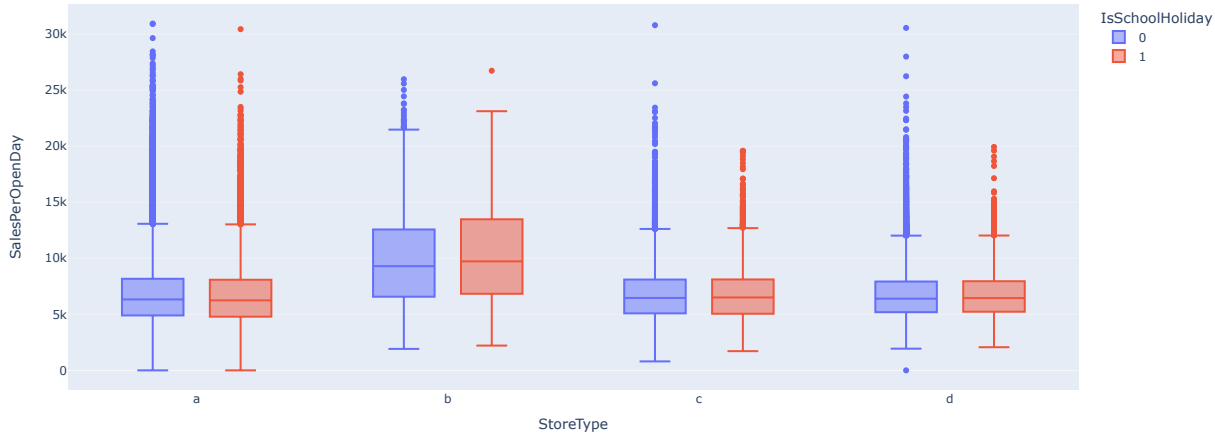
Result: Hypothesis is not true at all.

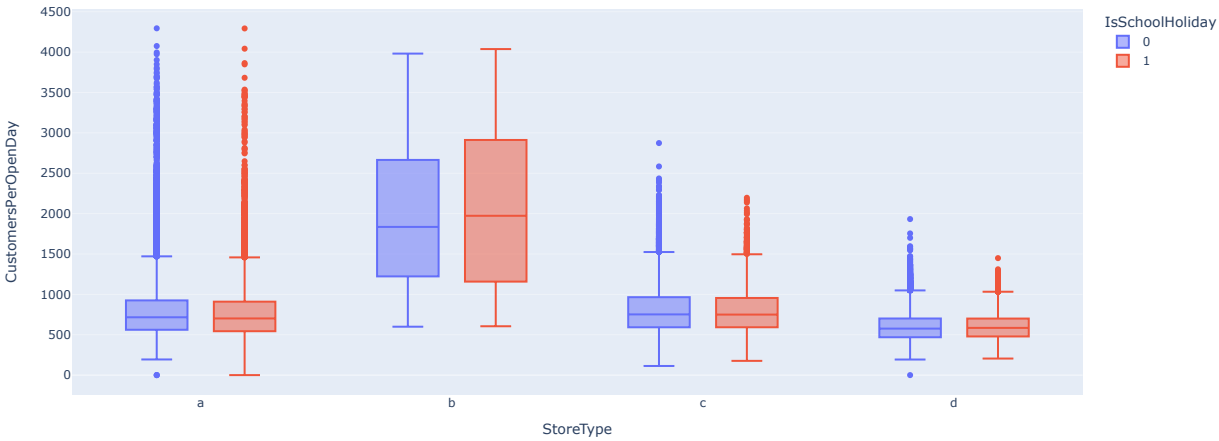
- Sales per day is ~ -10% lower
- Customers per day is ~ -12% lower
- Sales per customer is ~ -0,3% lower

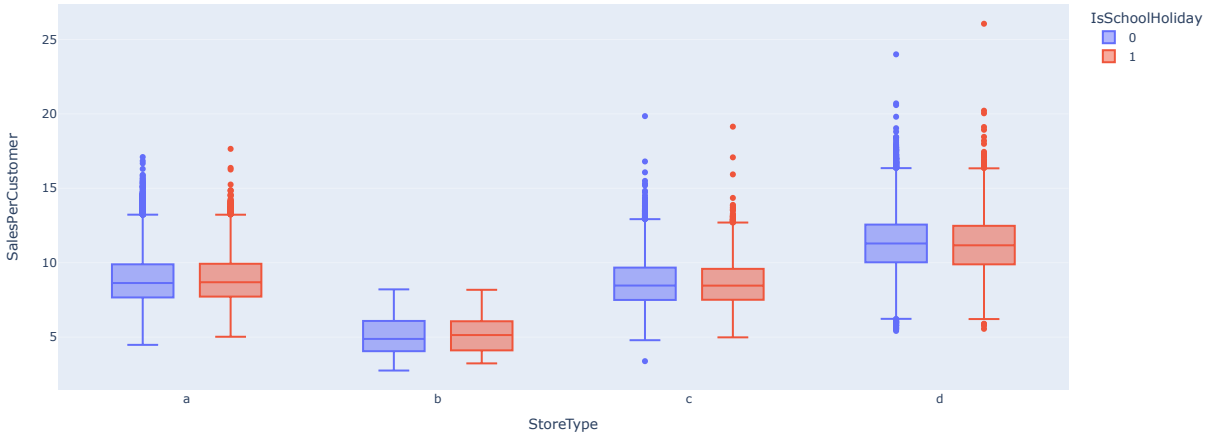
3. SchoolHolidays

3.1 Sales are lower in weeks with school vacations?

In [27]:
```
# To be unbiased only weeks where is no StateHolidays.
used_df = weekly_sales_with_store_info[weekly_sales_with_store_info['IsStateHoliday'] == 0]
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerOpenDay', color='IsSchoolHoliday')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='CustomersPerOpenDay', color='IsSchoolHoliday')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerCustomer', color='IsSchoolHoliday')
fig.show()

median_values = used_df.groupby(['StoreType', 'IsSchoolHoliday']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='IsSchoolHoliday', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpe
median_values_pivot
```



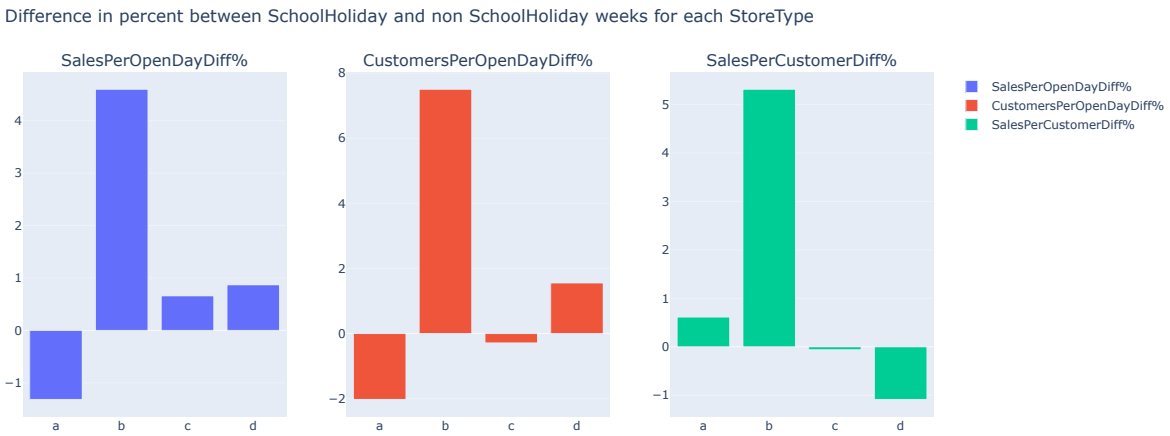




Out[27]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
IsSchoolHoliday	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**
StoreType									
a	716.17	701.67	-2.02	8.63	8.69	0.61	6327.83	6244.50	-1.32
b	1835.29	1973.00	7.50	4.88	5.14	5.32	9295.43	9723.00	4.60
c	751.67	749.50	-0.29	8.46	8.46	-0.06	6460.33	6503.00	0.66
d	575.83	584.79	1.56	11.29	11.17	-1.09	6394.00	6449.67	0.87

In [28]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between SchoolHoliday and non SchoolHoliday weeks for each StoreType")
fig.show()
```



Result: Hypothesis is just true for storetype a.

- For Storetype a the sales on Schooldays are 1.3% lower.
- For Storetype b the sales on Schooldays are 4.6% higher.
- For Storetype c and d the sales on Schooldays are nearly the same higher with 0.7% and 0.9%.

3.2 In weeks with school vacations, sales per customer are lower?

Result: Hypothesis just true for storetype d.

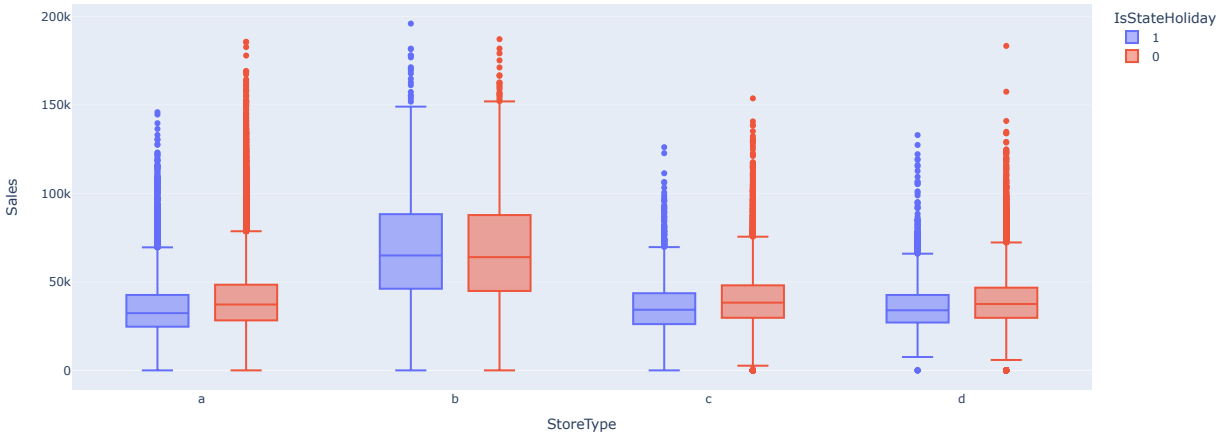
- For Storetype a the sales per customer on Schooldays are 0.6& higher.
- For Storetype b the sales per customer on Schooldays are 5.3% higher.
- For Storetype c there is no differnce compared no other days.

- For Storetype d the sales per customer on Schooldays are 1.1% lower.

4. StateHolidays

4.1 Sales are lower in weeks with public holidays?

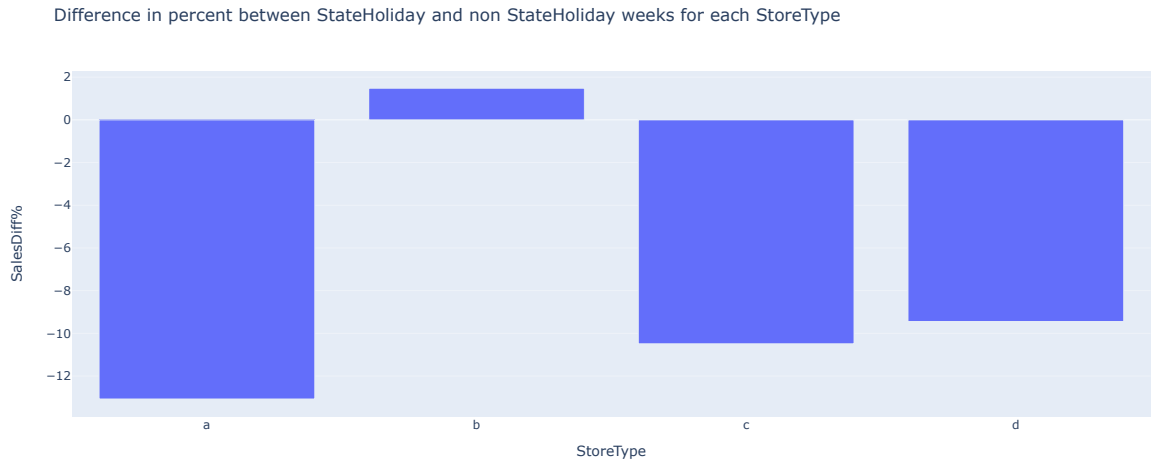
In [29]:
```
used_df = weekly_sales_with_store_info

fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='Sales', color='IsStateHoliday')
fig.show()

median_values = used_df.groupby(['StoreType', 'IsStateHoliday']).agg({'Sales': 'median'}).reset_index()
median_values['SalesDiff%'] = median_values.groupby('StoreType')['Sales'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='IsStateHoliday', values=['Sales', 'SalesDiff%']).style.format("{:.2f}")
median_values_pivot
```



Out[29]:

	Sales	SalesDiff%
IsStateHoliday	**0**	**1**	**1**
StoreType			
a	37202.50	32337.50	-13.08
b	63979.00	64926.00	1.48
c	38265.00	34251.50	-10.49
d	37498.00	33956.00	-9.45

In [30]:
```
px.bar(median_values, x='StoreType', y='SalesDiff%', title='Difference in percent between StateHoliday and non StateHoliday weeks for each StoreType')
```



In [31]:
```
# Check how many days store b is open if in the week is a StateHoliday
weekly_sales_with_store_info[(weekly_sales_with_store_info['StoreType'] == 'b') & (weekly_sales_with_store_info['IsStateHoliday'] == 1)]['Open'].value_counts()
```

Out[31]:
```
Open
7    398
6     15
0      6
3      2
4      1
Name: count, dtype: int64
```

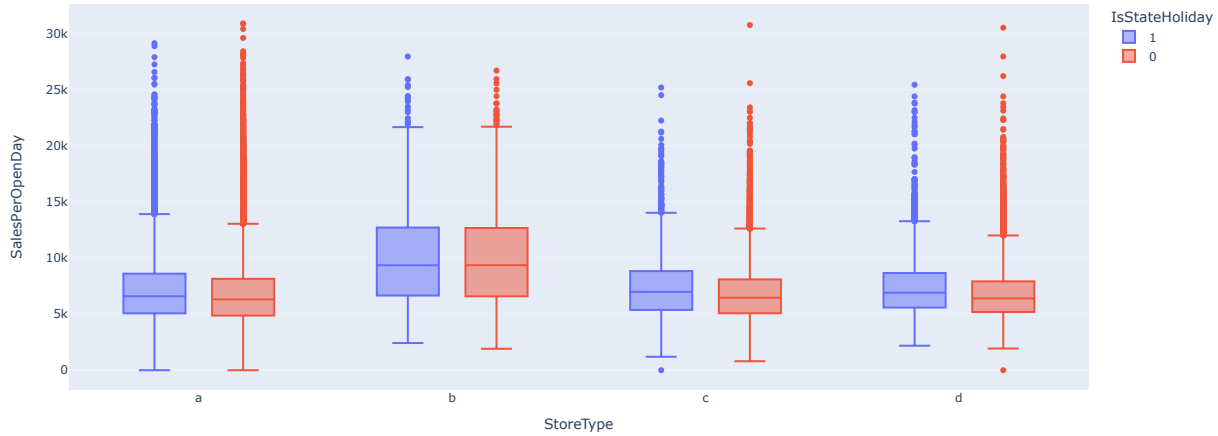
Result Hypothesis is true for all storetypes exept storetype b.

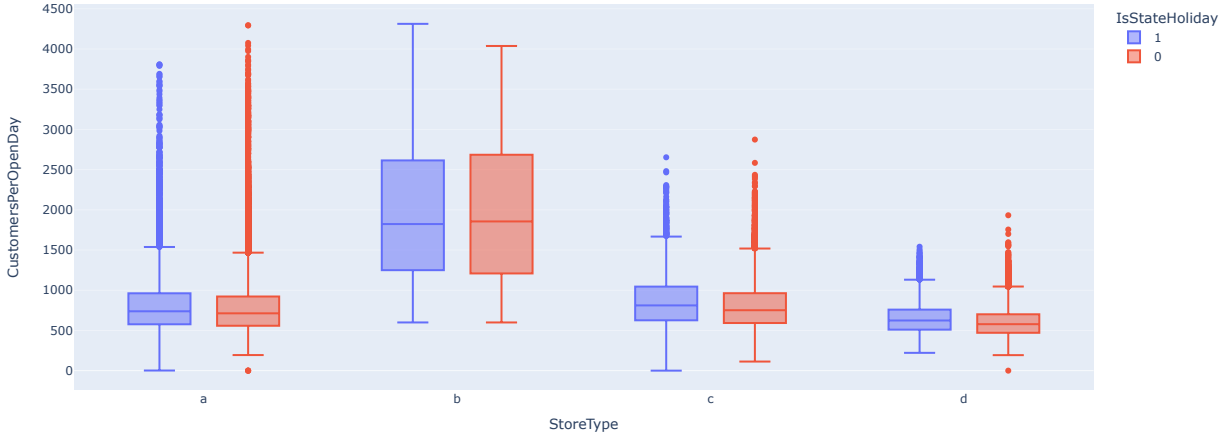
- For Storetype a the sales on holidays are 13% lower.
- For Storetype b the sales on holidays are 1.5% higher, but this is due to that the stores are mostly opend all 7 days a week.
- For Storetype c the sales on holidays are 10.5% lower.
- For Storetype d the sales on holidays are 9.4% lower.

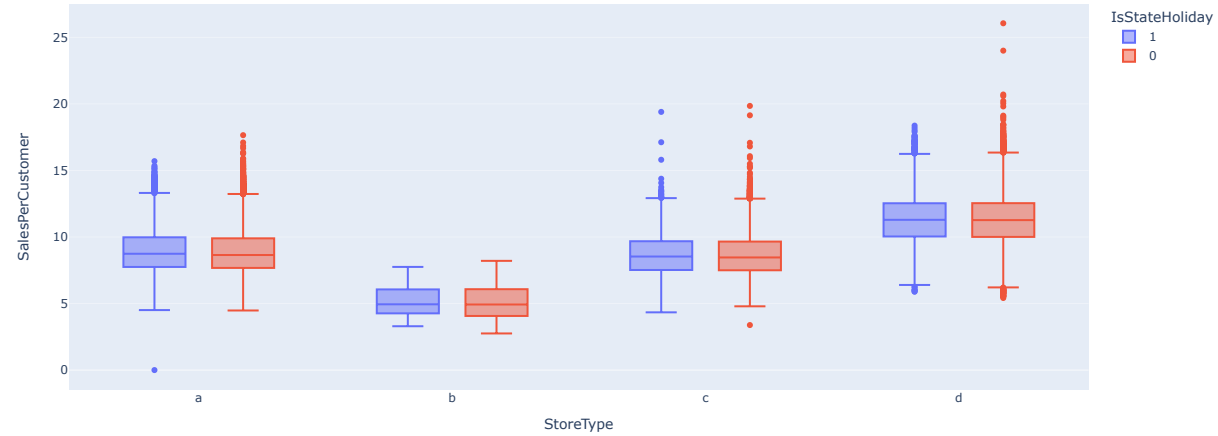
4.2 In weeks with public holidays, more purchases are made on the other days?

In [32]:
```
used_df = weekly_sales_with_store_info

fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerOpenDay', color='IsStateHoliday')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='CustomersPerOpenDay', color='IsStateHoliday')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerCustomer', color='IsStateHoliday')
fig.show()
```

```
median_values = used_df.groupby(['StoreType', 'IsStateHoliday']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='IsStateHoliday', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpen
median_values_pivot
```



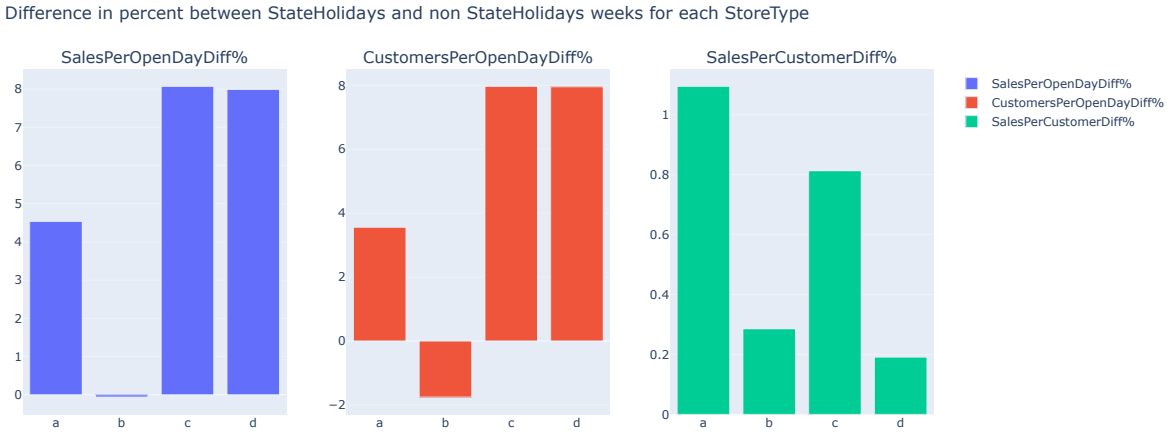




Out[32]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
IsStateHoliday	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**
StoreType									
a	712.83	738.20	3.56	8.65	8.74	1.09	6311.75	6598.20	4.54
b	1855.43	1822.86	-1.76	4.93	4.94	0.29	9369.43	9363.71	-0.06
c	751.33	811.20	7.97	8.46	8.53	0.81	6466.75	6988.82	8.07
d	577.83	623.80	7.96	11.27	11.29	0.19	6406.00	6918.00	7.99

In [33]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between StateHolidays and non StateHolidays weeks for each StoreType")
fig.show()
```



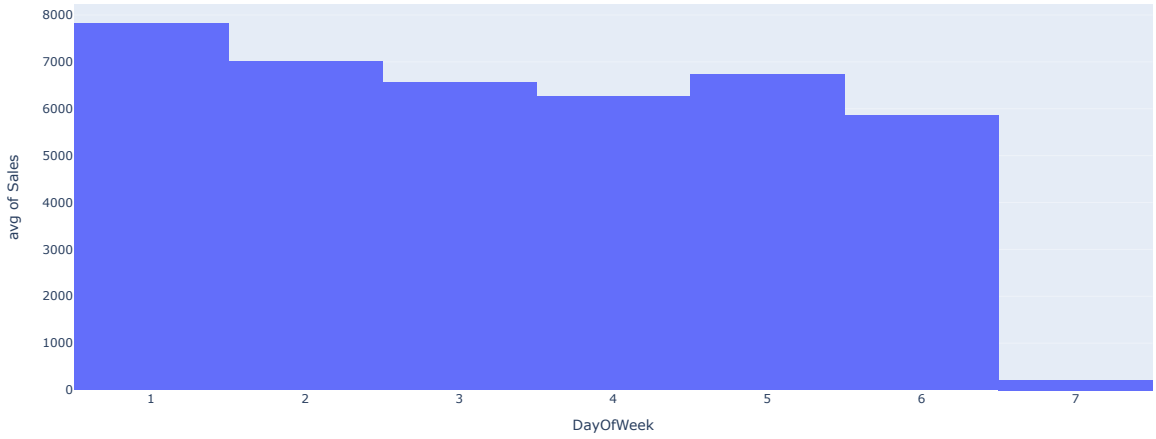
Result Hypothesis is true for all storetypes exept storetype b.

- For Storetype a the sales on a open day within a holiday week are 4.5% higher.
- For Storetype b the sales are equal but it is due to the fact that the stores are mostly opend all 7 days a week.
- For Storetype c and d the sales on a open day within a holiday week are 8.0% higher.

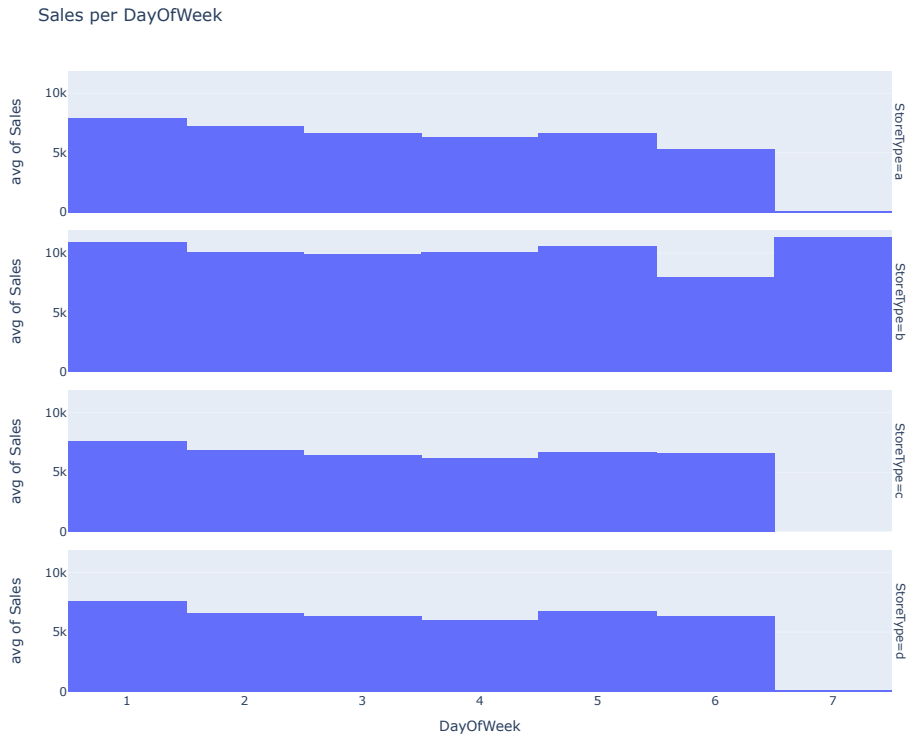
5. DayOfWeek

5.1 Fewer purchases are made at the weekend?

In [34]:
```
fig = px.histogram(df_train, x='DayOfWeek', y='Sales', histfunc='avg')
fig.show()
```



In [35]:
```
df_train_merged = pd.merge(df_train, df_store, on='Store', how='left').sort_values(by='StoreType')

plt.figure(figsize=(20, 10))
fig = px.histogram(df_train_merged, x='DayOfWeek', y='Sales', histfunc='avg', title='Sales per DayOfWeek', facet_row='StoreType')
fig.update_layout(width=1000, height=800)
fig.show()
```




```
<Figure size 2000x1000 with 0 Axes>
```

In [36]:
```
(df_train_merged[(df_train_merged['StoreType'] == 'a') & (df_train_merged['DayOfWeek'] == 6)]['Sales'].mean() - df_train_merged[(df_train_merged['StoreType'] == 'a') & (df_train_merged['DayOfWeek'] >= 1)
```

Out[36]:
```
-0.30272726562654123
```

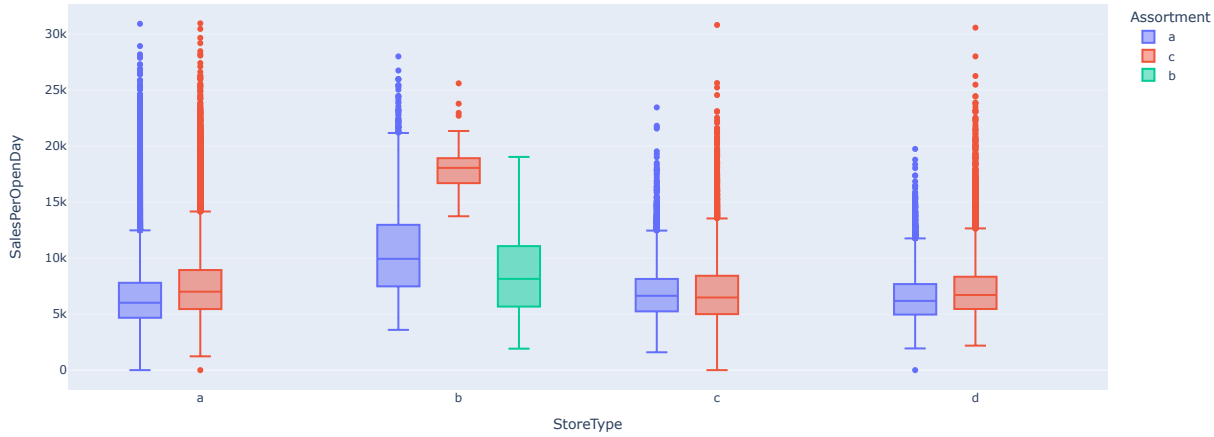
Result Hypothesis is true except for StoreType b.

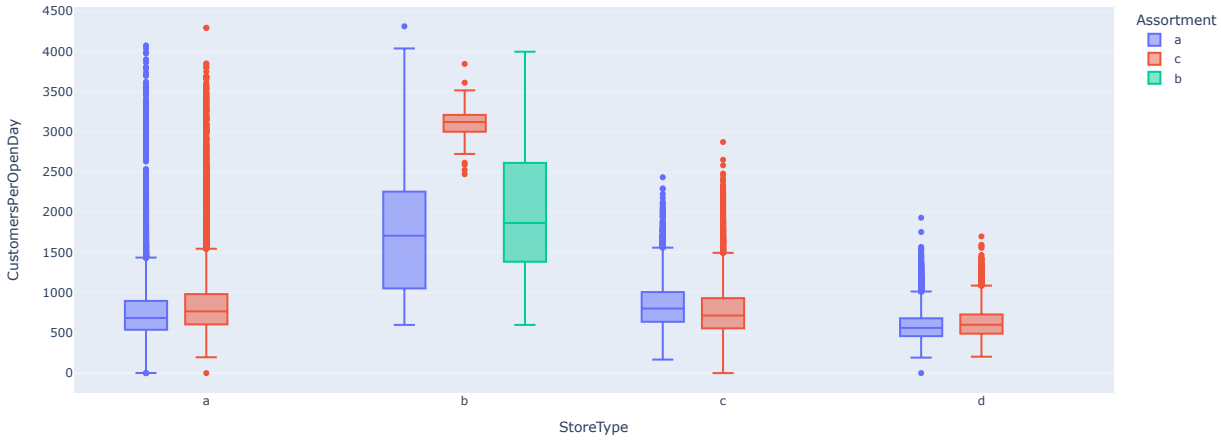
- In average, on a Saturday there is 30% less sales as within the week

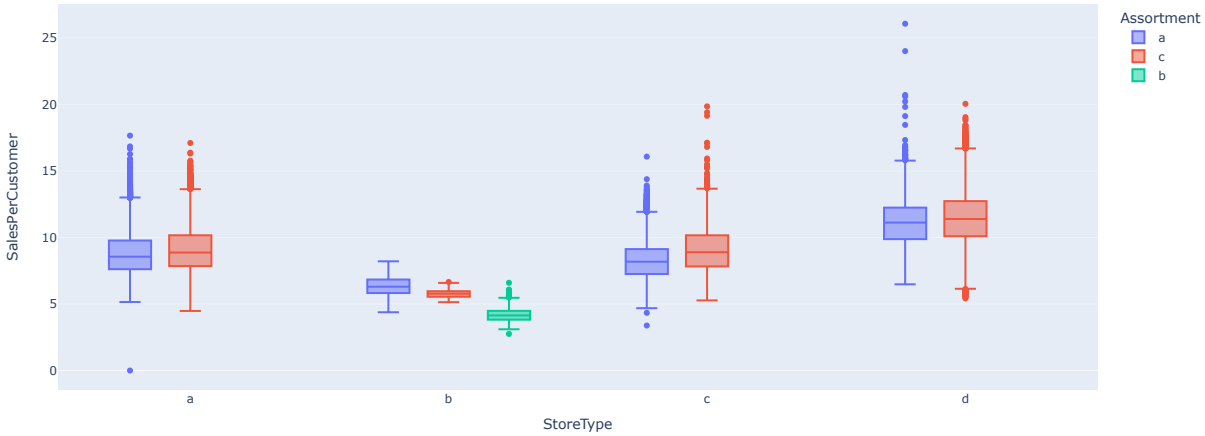
6. Assortment

6.1 Stores with an extra assortment have higher sales than basic stores?

In [37]:
```
used_df = weekly_sales_with_store_info

fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerOpenDay', color='Assortment')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='CustomersPerOpenDay', color='Assortment')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerCustomer', color='Assortment')
fig.show()
```





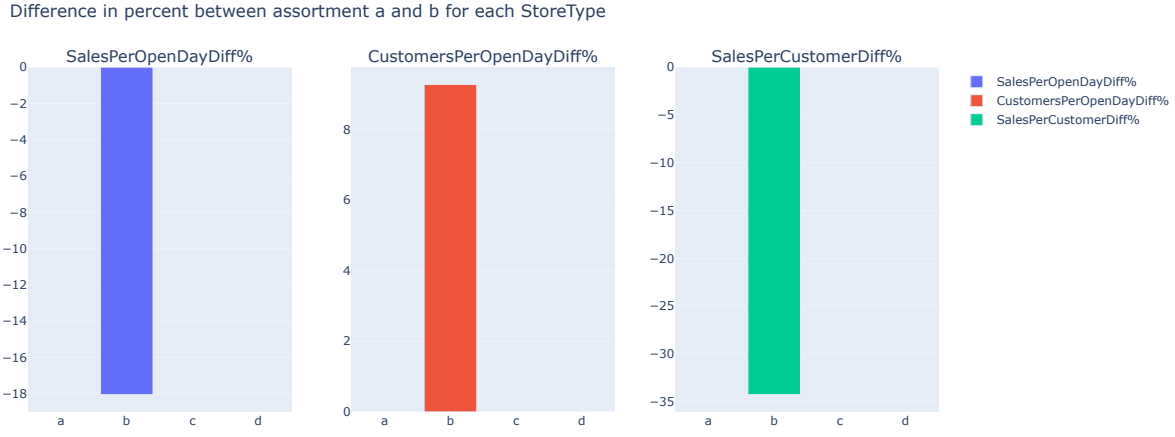


In [38]:
```
# Calculate difference between assortment a and b
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Assortment'] == 'a') | (weekly_sales_with_store_info['Assortment'] == 'b')]

median_values = used_df.groupby(['StoreType', 'Assortment']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='Assortment', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDayD
median_values_pivot
```

Out[38]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Assortment	**a**	**b**	**b**	**a**	**b**	**b**	**a**	**b**	**b**
StoreType									
a	685.80	nan	nan	8.55	nan	nan	6009.60	nan	nan
b	1708.57	1867.14	9.28	6.30	4.14	-34.27	9931.14	8139.29	-18.04
c	803.00	nan	nan	8.18	nan	nan	6632.73	nan	nan
d	561.80	nan	nan	11.11	nan	nan	6177.67	nan	nan

In [39]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between assortment a and b for each StoreType")
fig.show()
```



Result Hypothesis is false.

- Only Storetype b has the assortment caregory extra(b).

- The sales in Storetype b for assortment extra is 18% less then for assortment basic(a).
- They have 9.3% more customers but 34% less sales per customer.

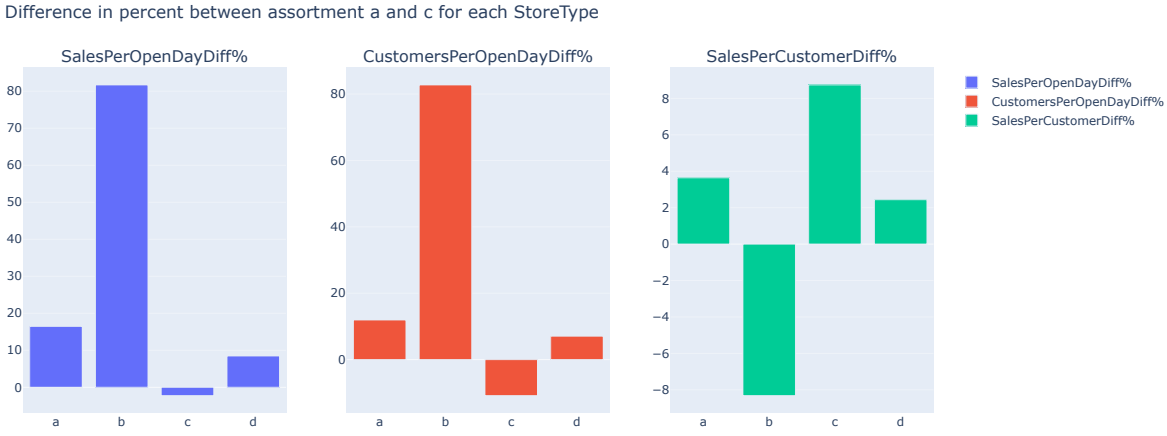
6.2 Stores with an extended assortment have higher sales than basic and extra stores?

In [40]:
```
# Calculate difference between assortment a and c
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Assortment'] == 'a') | (weekly_sales_with_store_info['Assortment'] == 'c')]

median_values = used_df.groupby(['StoreType', 'Assortment']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='Assortment', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDayD
median_values_pivot
```

Out[40]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Assortment	**a**	**c**	**c**	**a**	**c**	**c**	**a**	**c**	**c**
StoreType									
a	685.80	767.80	11.96	8.55	8.87	3.64	6009.60	6998.83	16.46
b	1708.57	3122.57	82.76	6.30	5.78	-8.32	9931.14	18045.71	81.71
c	803.00	715.83	-10.86	8.18	8.89	8.74	6632.73	6483.08	-2.26
d	561.80	601.50	7.07	11.11	11.39	2.44	6177.67	6701.08	8.47

In [41]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between assortment a and c for each StoreType")
fig.show()
```

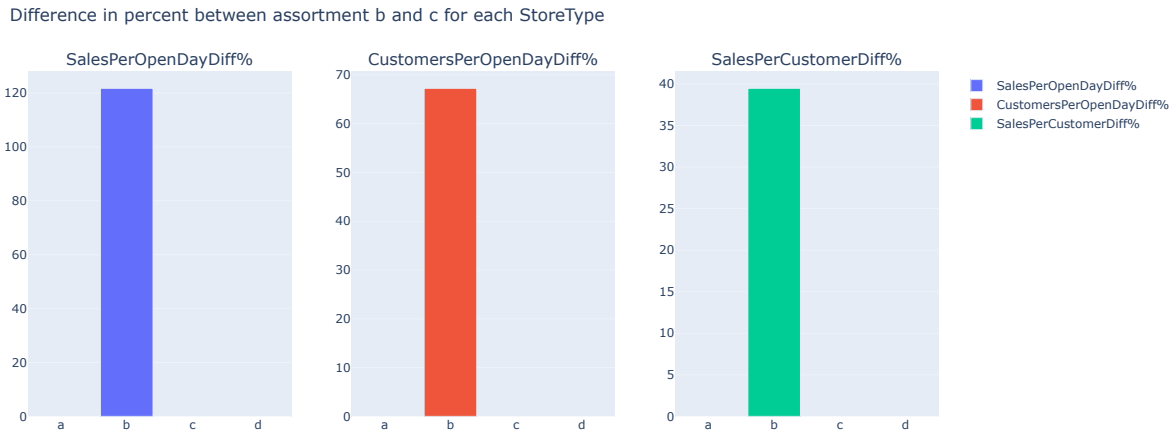


In [42]:
```
# Calculate difference between assortment b and c
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Assortment'] == 'b') | (weekly_sales_with_store_info['Assortment'] == 'c')]

median_values = used_df.groupby(['StoreType', 'Assortment']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='StoreType', columns='Assortment', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDayD
median_values_pivot
```

Out[42]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Assortment	**b**	**c**	**c**	**b**	**c**	**c**	**b**	**c**	**c**
StoreType									
a	nan	767.80	nan	nan	8.87	nan	nan	6998.83	nan
b	1867.14	3122.57	67.24	4.14	5.78	39.47	8139.29	18045.71	121.71
c	nan	715.83	nan	nan	8.89	nan	nan	6483.08	nan
d	nan	601.50	nan	nan	11.39	nan	nan	6701.08	nan

In [43]:
```
# show the diff columns in a plotly bar plot unsing subplots
fig = make_subplots(rows=1, cols=3, subplot_titles=("SalesPerOpenDayDiff%", "CustomersPerOpenDayDiff%", "SalesPerCustomerDiff%"))
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerOpenDayDiff%'], name='SalesPerOpenDayDiff%'), row=1, col=1)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['CustomersPerOpenDayDiff%'], name='CustomersPerOpenDayDiff%'), row=1, col=2)
fig.add_trace(go.Bar(x=median_values['StoreType'], y=median_values['SalesPerCustomerDiff%'], name='SalesPerCustomerDiff%'), row=1, col=3)
fig.update_layout(title_text="Difference in percent between assortment b and c for each StoreType")
fig.show()
```



Result Hypothesis is mostly true.

- Only Storetype b has the assortment caregory extra(b).
- The sales for assortment erweitert(c) is in average 26%(16%,81%,-2%,8%) higher then for assortment basic(a).
- The sales for assortment erweitert(c) is 121% higher then for assortment extra(b).

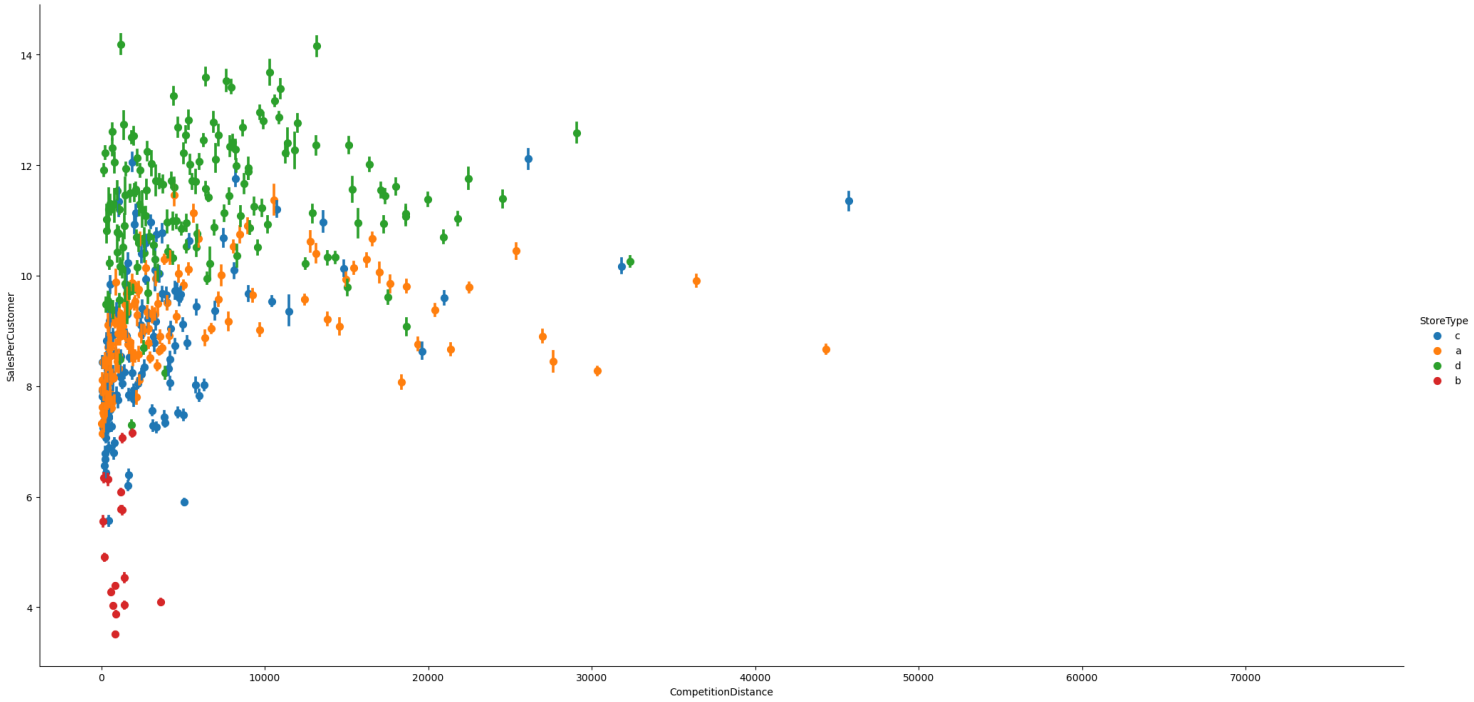
7. CompetitionDistance

7.1 The closer the competitor, the lower the sales?

In [44]:
```
sns.lmplot(x='CompetitionDistance', y='SalesPerCustomer', data=weekly_sales_with_store_info, hue='StoreType', fit_reg=False, x_bins=150, height=10
           , aspect=2
           )
```

```
c:\Users\Chris\anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```

Out[44]:
```
<seaborn.axisgrid.FacetGrid at 0x258aeafdaf0>
```



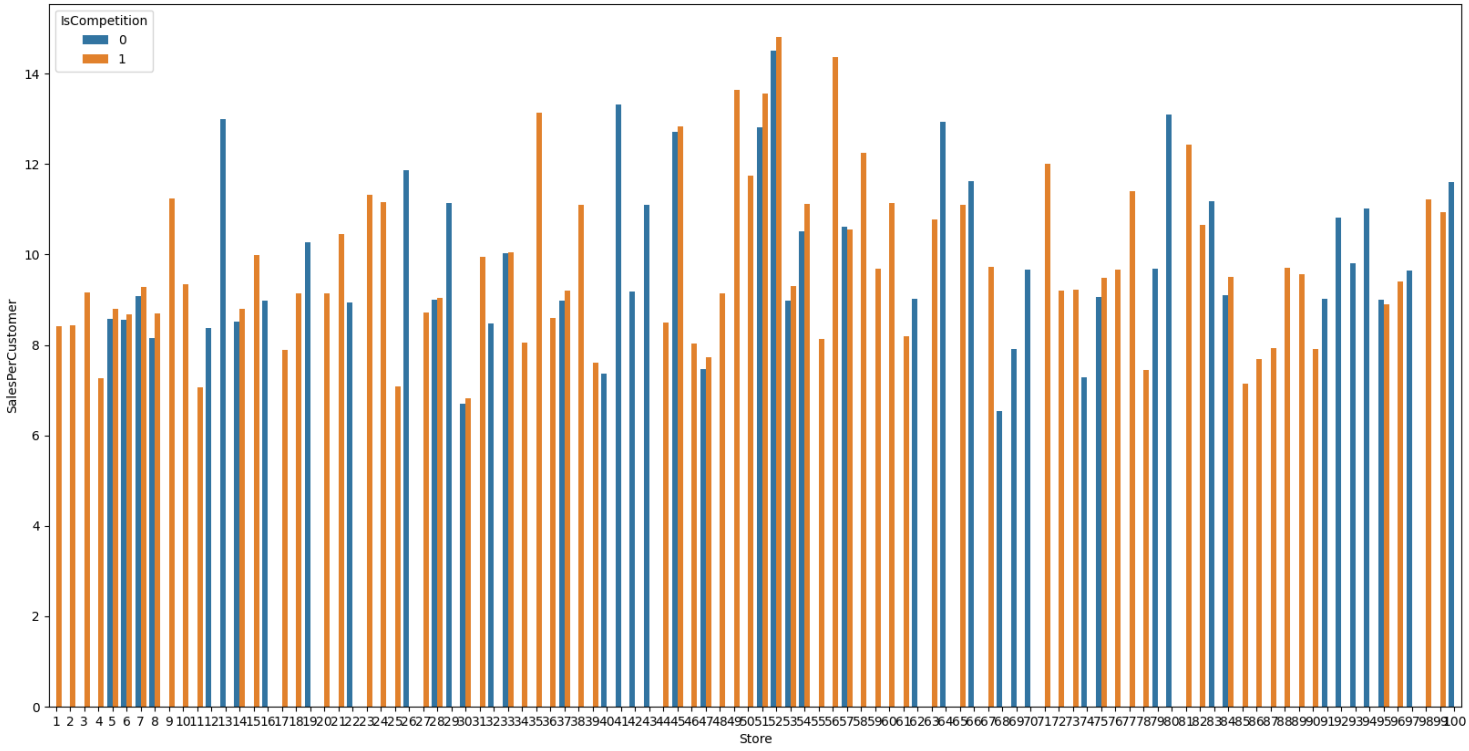
Result Hypothesis is nearly true for storeType a, c, d.

- The sales per customer is rising if the distance to the competeter is getting up to 10km. After that the sales per customer is falling until 20km and is rising again after 20km.

7.2 Sales have been lower since the competitor opened?

In [45]:
```
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Store'] >= 0) & (weekly_sales_with_store_info['Store'] <= 100) & (weekly_sales_with_store_info['IsCompetition'].mean() > 0)].sort_valu
plt.figure(figsize=(20, 10))
sns.barplot(x='Store', y='SalesPerCustomer', data=used_df, hue='IsCompetition', errorbar=None)
```

Out[45]:
```
<Axes: xlabel='Store', ylabel='SalesPerCustomer'>
```



In [46]:
```
def returnSalesBeforeCompetition(df, storeId):
        store = df[(df['Store'] == storeId) & (df['IsCompetition'] == 0)]
        competition = df[(df['Store'] == storeId) & (df['IsCompetition'] == 1)]
        return store['Sales'].mean(), competition['Sales'].mean()

arr_sales_before_competition = []
arr_sales_after_competition = []

for id in range(1, 1116):
        store = weekly_sales_with_store_info[(weekly_sales_with_store_info['Store'] == id)]
        if (store['IsCompetition'].mean() > 0) and (store['IsCompetition'].mean() < 1):
                store_sales, competition_sales = returnSalesBeforeCompetition(weekly_sales_with_store_info, id)
                arr_sales_before_competition.append(store_sales)
                arr_sales_after_competition.append(competition_sales)

print(f"Mean sales before competition: {np.mean(arr_sales_before_competition)}")
print(f"Mean sales after competition: {np.mean(arr_sales_after_competition)}")
```

```
Mean sales before competition: 42475.15042881677
Mean sales after competition: 40241.59236485943
```

Result Hypothesis is true.

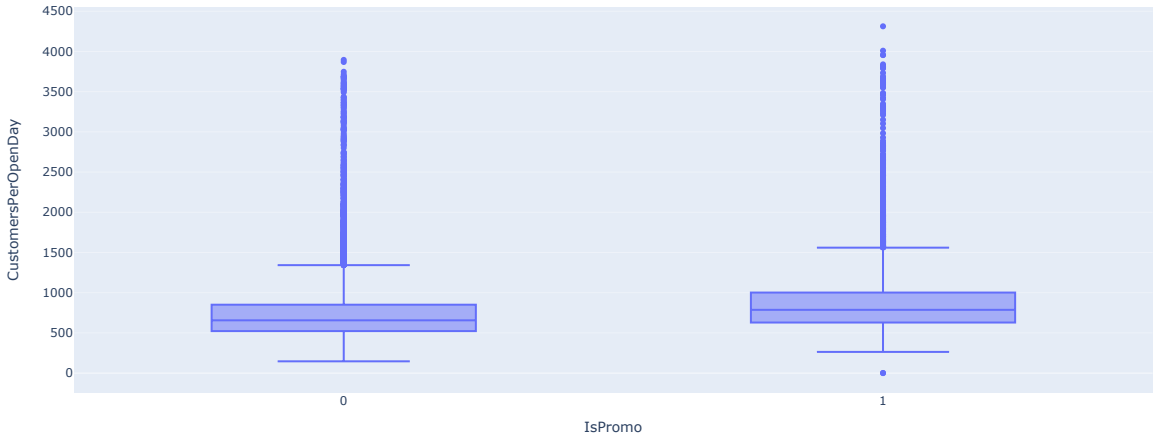
- The sales is in average 5,5% less after the competeter has opened.

Answers Further interesting questions

1. Promotions

1.1 Is a promotion worthwhile in weeks with state holidays?

In [47]:
```
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['IsStateHoliday'] == 1)]

fig = px.box(used_df.sort_values('IsPromo'), x='IsPromo', y='SalesPerOpenDay')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('IsPromo'), x='IsPromo', y='CustomersPerOpenDay')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('IsPromo'), x='IsPromo', y='SalesPerCustomer')
fig.show()
```



In [48]:
```
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['IsStateHoliday'] == 1)]

median_values = used_df.groupby(['IsPromo']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='IsPromo', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDayDiff%', 'SalesPerCustomer
median_values_pivot
```

Out[48]:

IsPromo	CustomersPerOpenDay	CustomersPerOpenDayDiff%	SalesPerCustomer	SalesPerCustomerDiff%	SalesPerOpenDay	SalesPerOpenDayDiff%
0	656.40	nan	8.92	nan	6011.60	nan
1	786.80	19.87	10.32	15.61	8274.60	37.64

Result

- Also in a week with state holidays the promotions have a positive effect.
- The sales per day is 38% higher
- The customers per day is 20% higher
- The sales per customer is 16% higher

5. DayOfWeek

5.1 Is a promotion on the weekend worthwhile?

In [49]:
```
df_train_day = df_train.reset_index()
df_train_day
```

Out[49]:

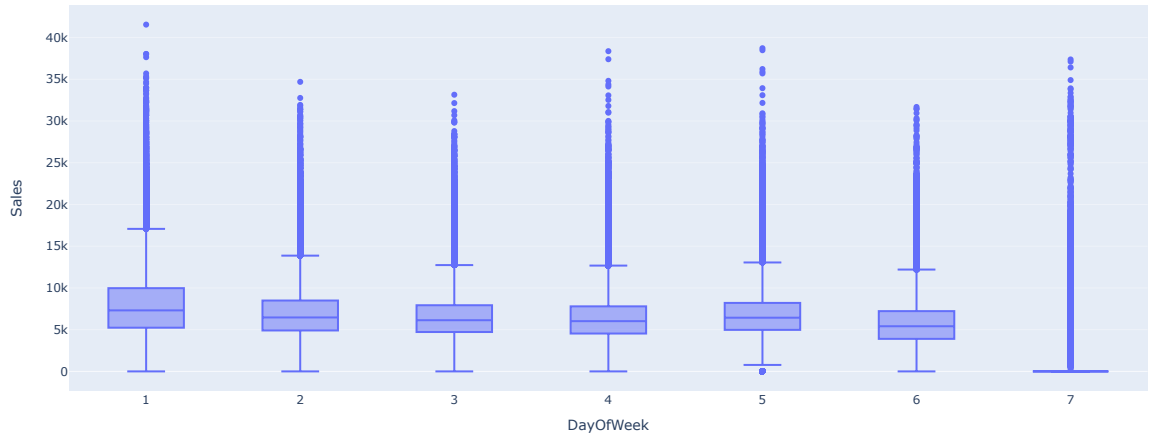
	Date	Store	DayOfWeek	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	2015-07-31	1	5	5263	555	1	1	0	1
1	2015-07-31	2	5	6064	625	1	1	0	1
2	2015-07-31	3	5	8314	821	1	1	0	1
3	2015-07-31	4	5	13995	1498	1	1	0	1
4	2015-07-31	5	5	4822	559	1	1	0	1
...	...	...	...	...	...	...	...	...	...
1017204	2013-01-01	1111	2	0	0	0	0	a	1
1017205	2013-01-01	1112	2	0	0	0	0	a	1
1017206	2013-01-01	1113	2	0	0	0	0	a	1
1017207	2013-01-01	1114	2	0	0	0	0	a	1
1017208	2013-01-01	1115	2	0	0	0	0	a	1

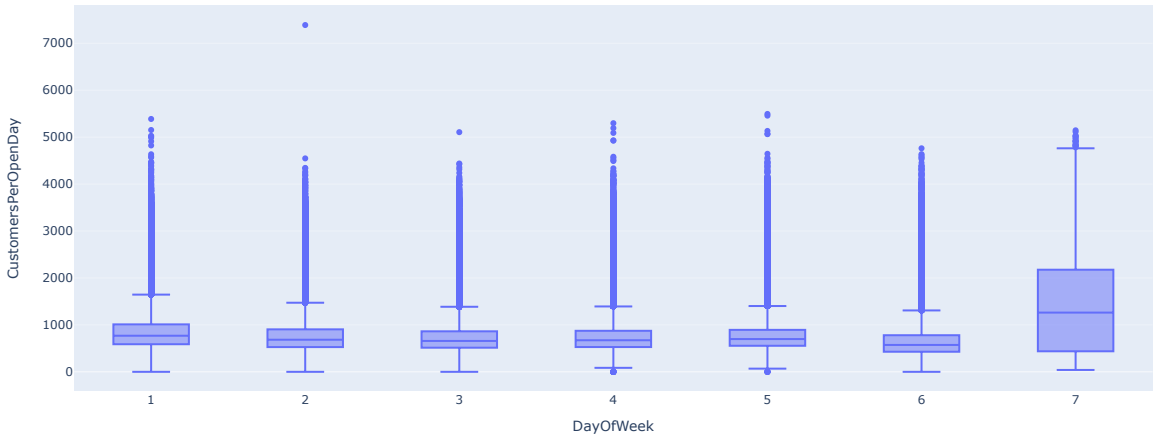
1017209 rows × 9 columns

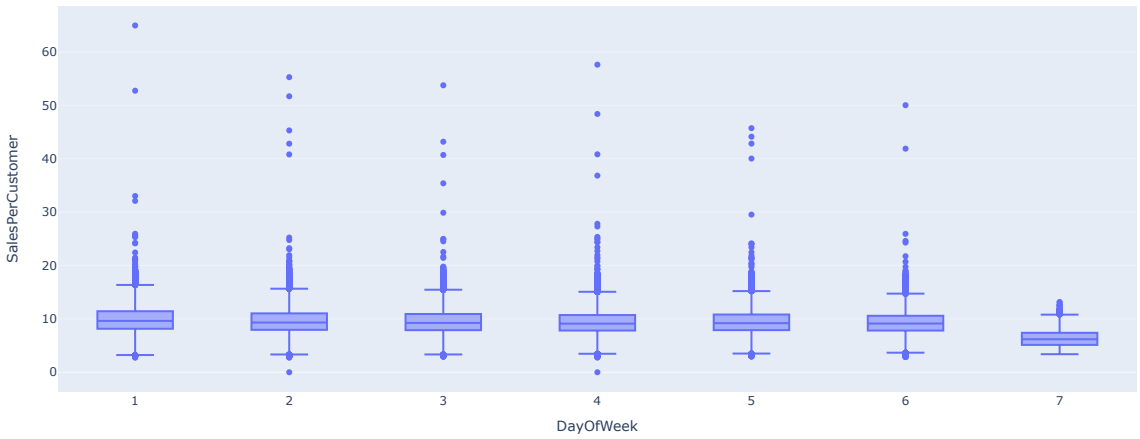
In [50]:
```
# Create Month and Year columns
df_train_day.insert(df_train_day.columns.get_loc('Date') + 1, 'CW', df_train_day['Date'].dt.isocalendar().week)
df_train_day.insert(df_train_day.columns.get_loc('Date') + 2, 'Month', df_train_day['Date'].dt.month)
df_train_day.insert(df_train_day.columns.get_loc('Date') + 3, 'Year', df_train_day['Date'].dt.year)

# Create SalesPerCustomer column
df_train_day.insert(df_train_day.columns.get_loc('Sales') + 1, 'SalesPerCustomer', df_train_day['Sales'] / df_train_day['Customers'])
# Create customers per open day
df_train_day.insert(df_train_day.columns.get_loc('Customers') + 1, 'CustomersPerOpenDay', df_train_day['Customers'] / df_train_day['Open'])
```

```
df_train_day.insert(df_train_day.columns.get_loc('Promo') + 1, 'IsPromo', np.where(df_train_day['Promo'] > 0, 1, 0))
df_train_day.insert(df_train_day.columns.get_loc('StateHoliday') + 1, 'IsStateHoliday', np.where(df_train_day['StateHoliday'] != '0', 1, 0))
df_train_day.insert(df_train_day.columns.get_loc('SchoolHoliday') + 1, 'IsSchoolHoliday', np.where(df_train_day['SchoolHoliday'] > 0, 1, 0))
```

In [51]:
```
used_df = df_train_day

fig = px.box(used_df.sort_values('DayOfWeek'), x='DayOfWeek', y='Sales')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('DayOfWeek'), x='DayOfWeek', y='CustomersPerOpenDay')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('DayOfWeek'), x='DayOfWeek', y='SalesPerCustomer')
fig.show()

median_values = used_df.groupby('DayOfWeek').agg({'Sales': 'median', 'Customers': 'median', 'SalesPerCustomer': 'median'}).reset_index()

median_values_pivot = median_values.pivot_table(index='DayOfWeek', values=['Sales', 'Customers', 'SalesPerCustomer']).style.format("{:.2f}")
median_values_pivot
```







Out[51]:

DayOfWeek	Customers	Sales	SalesPerCustomer
1	748.00	7310.00	9.62
2	680.00	6463.00	9.31
3	651.00	6133.00	9.24
4	646.00	6020.00	9.11
5	682.00	6434.00	9.19
6	571.00	5410.00	9.13
7	0.00	0.00	6.19

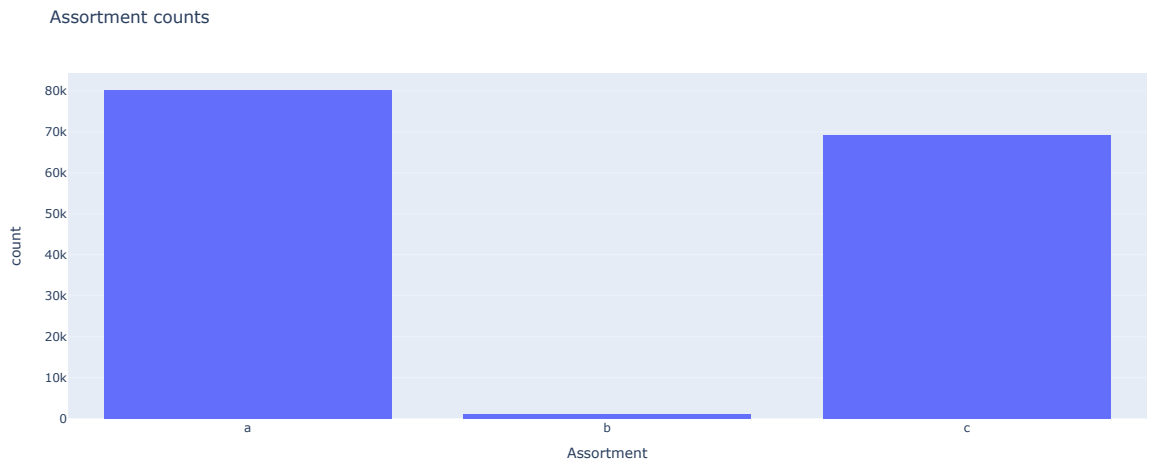
Result:

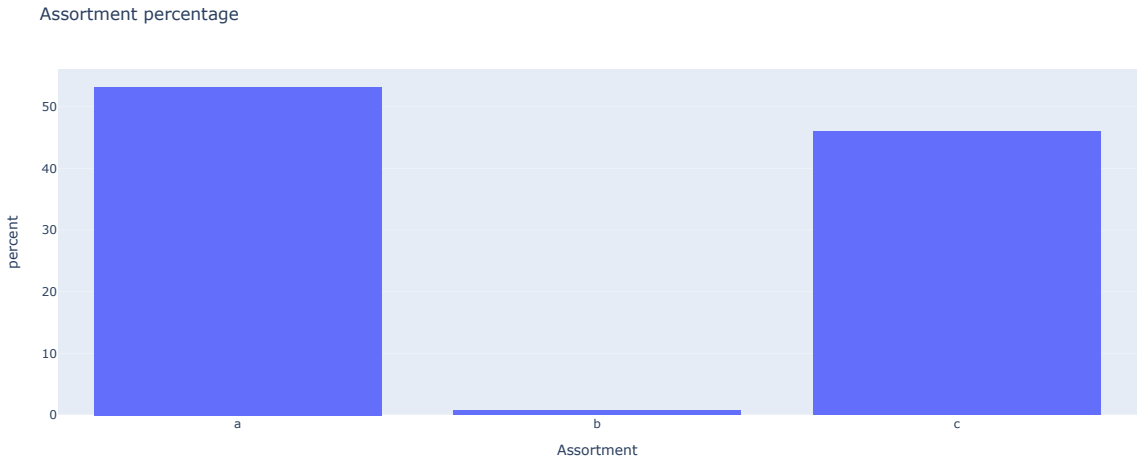
- The amount of customers on a Saturday is -22% lower then within the week.
- The sales on a Saturday is ~ -20% lower then within the week.
- The sales per customer on a Saturday is ~ 2% lower then within the week.

6. Assortment

6.1 What is the ratio of basic, extra and extended assortment?

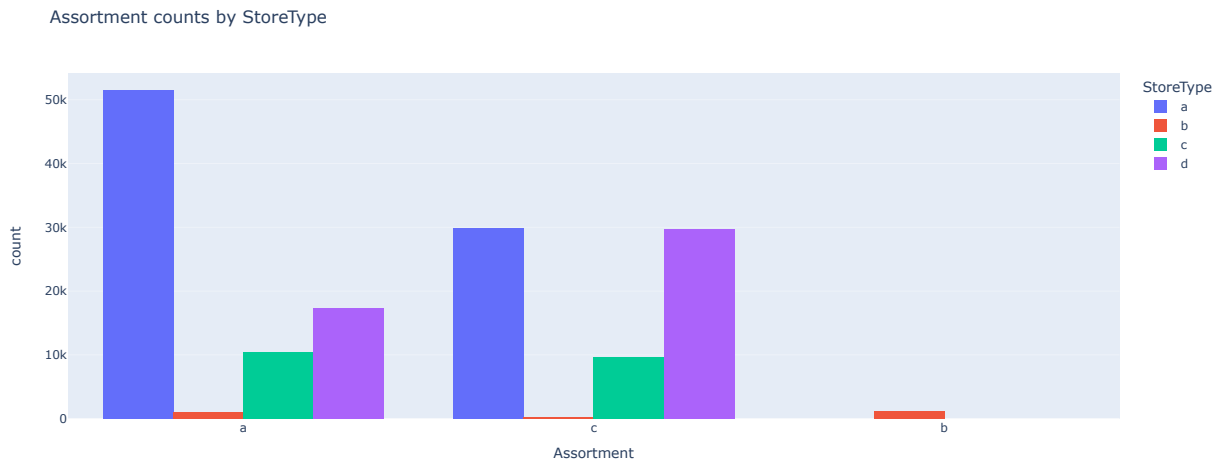
In [52]:
```
# counts of assortment a, b and c and there percentage
fig = px.histogram(weekly_sales_with_store_info.sort_values('Assortment'), x='Assortment', title='Assortment counts')
fig.show()

fig = px.histogram(weekly_sales_with_store_info.sort_values('Assortment'), x='Assortment', title='Assortment percentage', histnorm='percent')
fig.show()
```

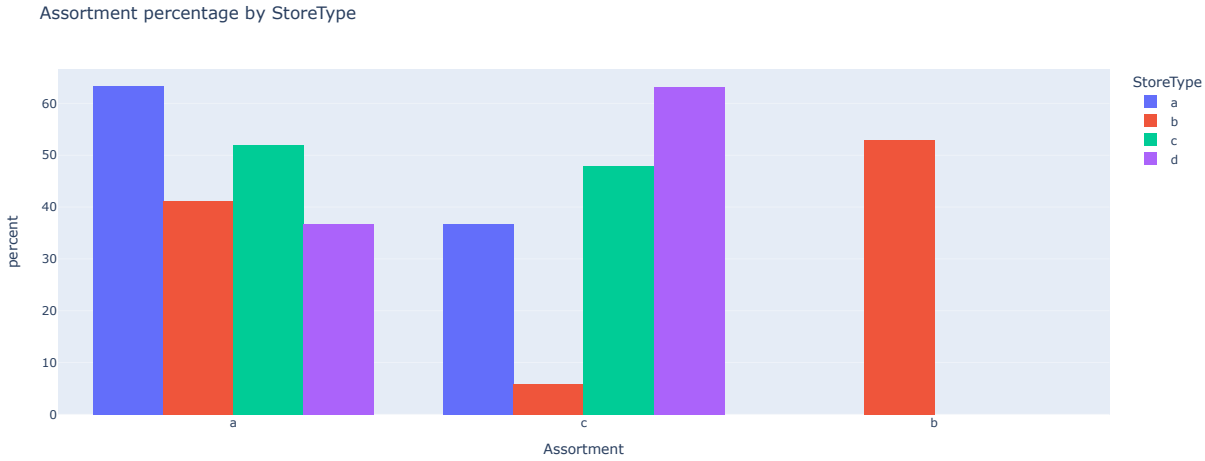




6.2 What is the ratio of basic, extra and extended assortment in the individual store types?

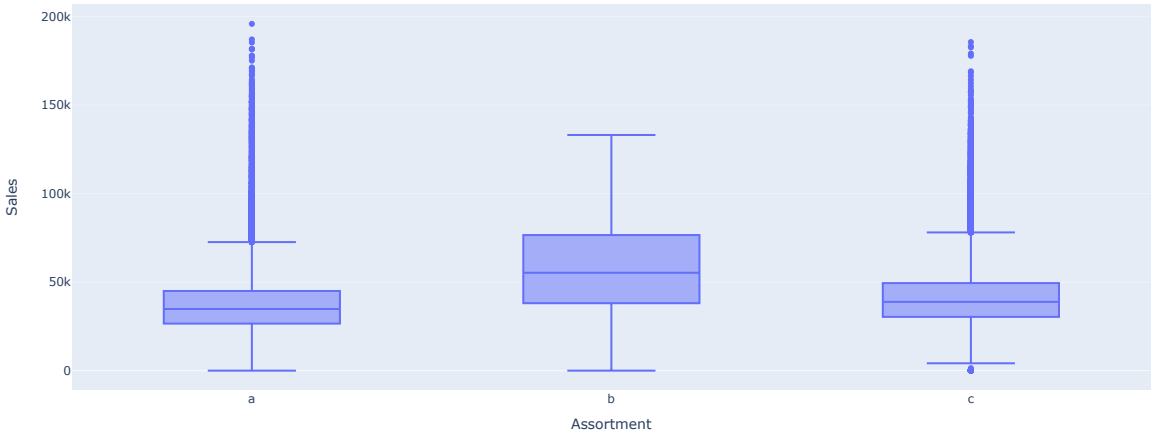
In [53]:
```
fig = px.histogram(weekly_sales_with_store_info.sort_values(by=['Assortment', 'StoreType']), x='Assortment', title='Assortment counts by StoreType', color='StoreType', barmode='group')
fig.show()

fig = px.histogram(weekly_sales_with_store_info.sort_values(by=['Assortment', 'StoreType']), x='Assortment', title='Assortment percentage by StoreType', histnorm='percent', color='StoreType', barmode='gro
fig.show()
```



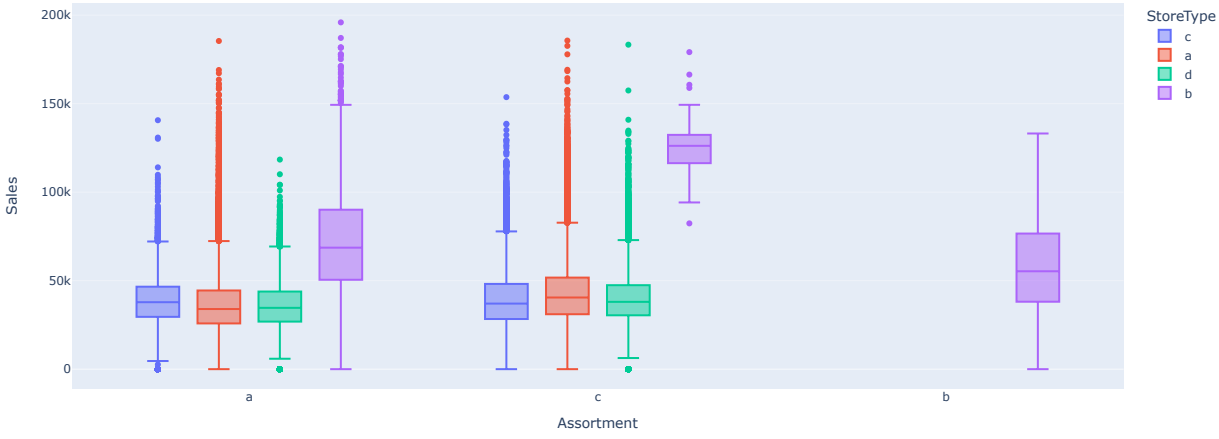


6.3 What is the average turnover of the individual assortments?

In [54]:
```
used_df = weekly_sales_with_store_info

fig = px.box(used_df.sort_values('Assortment'), x='Assortment', y='Sales')
fig.show()
```



In [55]:
```
used_df = weekly_sales_with_store_info

fig = px.box(used_df.sort_values('Assortment'), x='Assortment', y='Sales', color='StoreType')
fig.show()
```



6.4 How many promotion weeks are there in the individual assortments?

In [56]:
```
# How many promo weeks are there in each year for each assortment
promo_data = weekly_sales_with_store_info[weekly_sales_with_store_info['IsPromo'] == 1]

# Group by Year, Assortment, and Calendar Week, then count the number of weeks
promo_weeks = promo_data.groupby(['Year', 'Assortment'])['CW'].nunique().reset_index(name='PromoWeeksCount')
promo_weeks
```

Out[56]:

	Year	Assortment	PromoWeeksCount
0	2013	a	27
1	2013	b	27
2	2013	c	27
3	2014	a	28
4	2014	b	28
5	2014	c	28
6	2015	a	17
7	2015	b	17
8	2015	c	17

In [57]:
```
# How many promo weeks are there in each year for each StoreType
promo_data = weekly_sales_with_store_info[weekly_sales_with_store_info['IsPromo'] == 1]

# Group by Year, Assortment, and Calendar Week, then count the number of weeks
promo_weeks = promo_data.groupby(['Year', 'StoreType'])['CW'].nunique().reset_index(name='PromoWeeksCount')
promo_weeks
```

Out[57]:

	Year	StoreType	PromoWeeksCount
0	2013	a	27
1	2013	b	27
2	2013	c	27
3	2013	d	27
4	2014	a	28
5	2014	b	28
6	2014	c	28
7	2014	d	28
8	2015	a	17
9	2015	b	17
10	2015	c	17
11	2015	d	17

In [58]:
```
# How many promo weeks are there in each month
promo_data = weekly_sales_with_store_info[weekly_sales_with_store_info['IsPromo'] == 1]

promo_weeks = promo_data.groupby(['Year', 'Month'])['CW'].nunique().reset_index(name='PromoWeeksCount')
promo_weeks
```

Out[58]:

	Year	Month	PromoWeeksCount
0	2013	1	2
1	2013	2	2
2	2013	3	3
3	2013	4	2
4	2013	5	2
5	2013	6	3
6	2013	7	2
7	2013	8	2
8	2013	9	3
9	2013	10	2
10	2013	11	2
11	2013	12	2
12	2014	1	2
13	2014	2	2
14	2014	3	2
15	2014	4	2
16	2014	5	3
17	2014	6	2
18	2014	7	2
19	2014	8	3
20	2014	9	2
21	2014	10	3
22	2014	11	3
23	2014	12	2
24	2015	1	2
25	2015	2	3
26	2015	3	2
27	2015	4	2
28	2015	5	3
29	2015	6	2
30	2015	7	2
31	2015	8	1

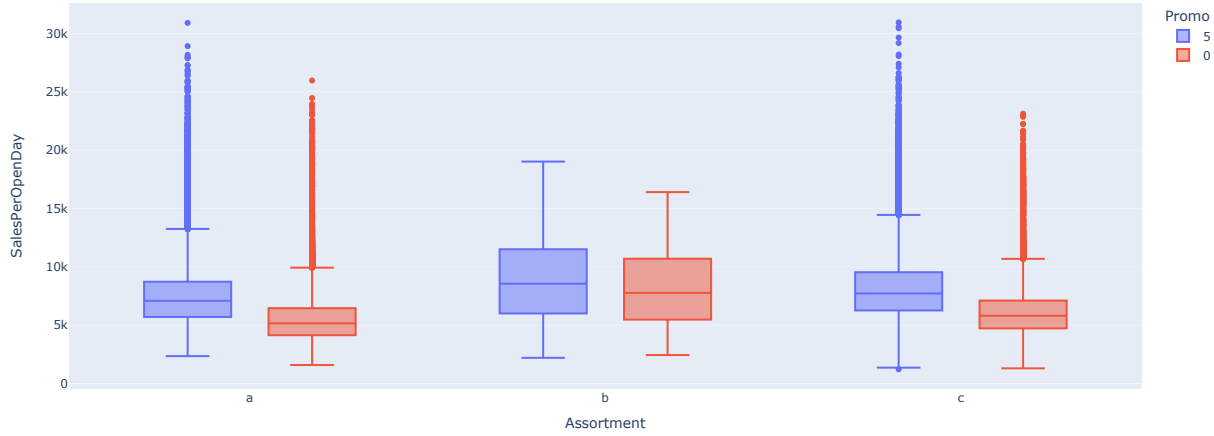
Result

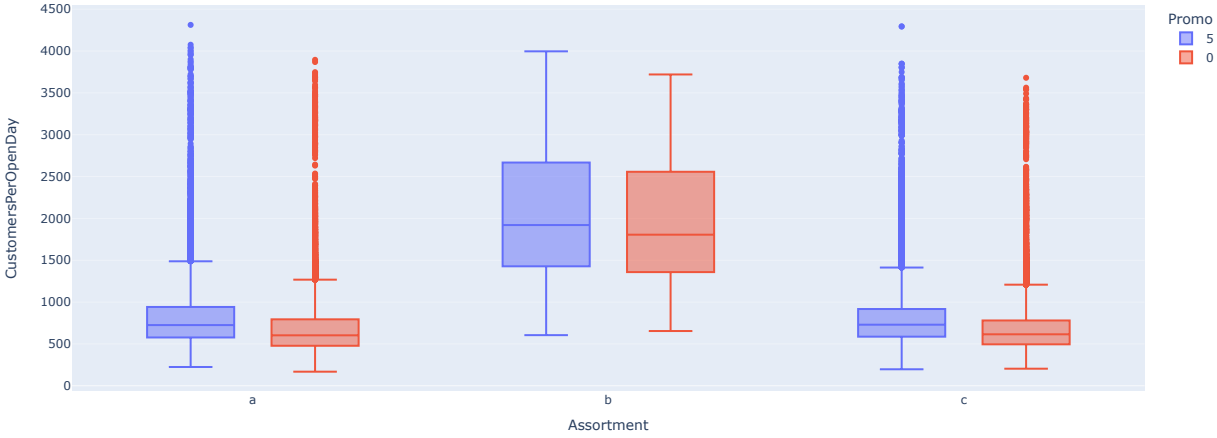
- Every second week for all assortment types.

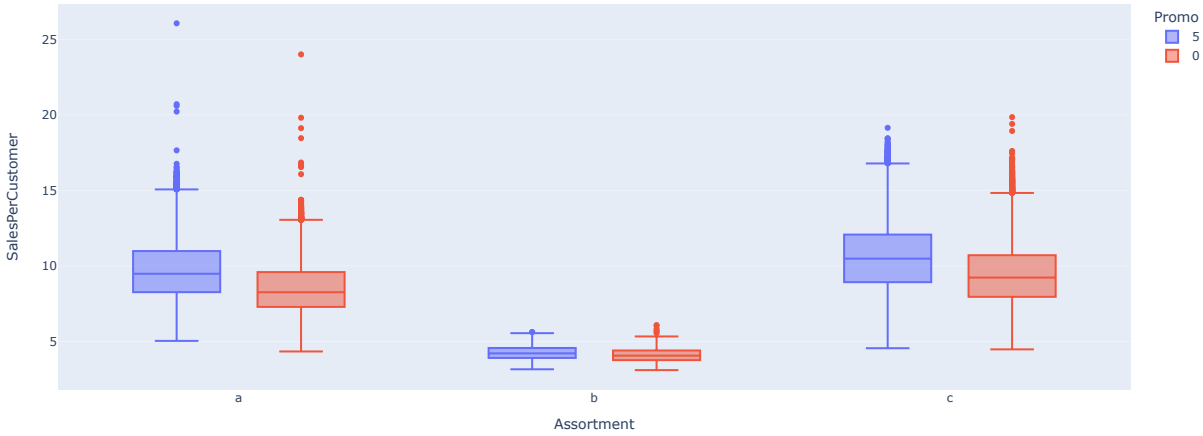
6.5 Do promotion weeks perform better in individual assortments than in others?

In [59]:
```
# To be compareble only weeks with >5 open days are used
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Open'] >= 5)]
fig = px.box(used_df.sort_values('Assortment'), x='Assortment', y='SalesPerOpenDay', color='Promo')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('Assortment'), x='Assortment', y='CustomersPerOpenDay', color='Promo')
fig.show()
# SalesPerCustomer
```

```
fig = px.box(used_df.sort_values('Assortment'), x='Assortment', y='SalesPerCustomer', color='Promo')
fig.show()

median_values = used_df.groupby(['Assortment', 'Promo']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()
median_values['SalesPerOpenDayDiff%'] = median_values.groupby('Assortment')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('Assortment')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('Assortment')['SalesPerCustomer'].pct_change() * 100

median_values_pivot = median_values.pivot_table(index='Assortment', columns='Promo', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDayDiff%
median_values_pivot
```







Out[59]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Promo	**0**	**5**	**5**	**0**	**5**	**5**	**0**	**5**	**5**
Assortment									
a	602.80	725.17	20.30	8.26	9.48	14.81	5161.83	7096.67	37.48
b	1805.43	1920.57	6.38	4.06	4.21	3.74	7771.21	8559.71	10.15
c	615.00	729.80	18.67	9.23	10.48	13.59	5810.55	7721.17	32.88

Result

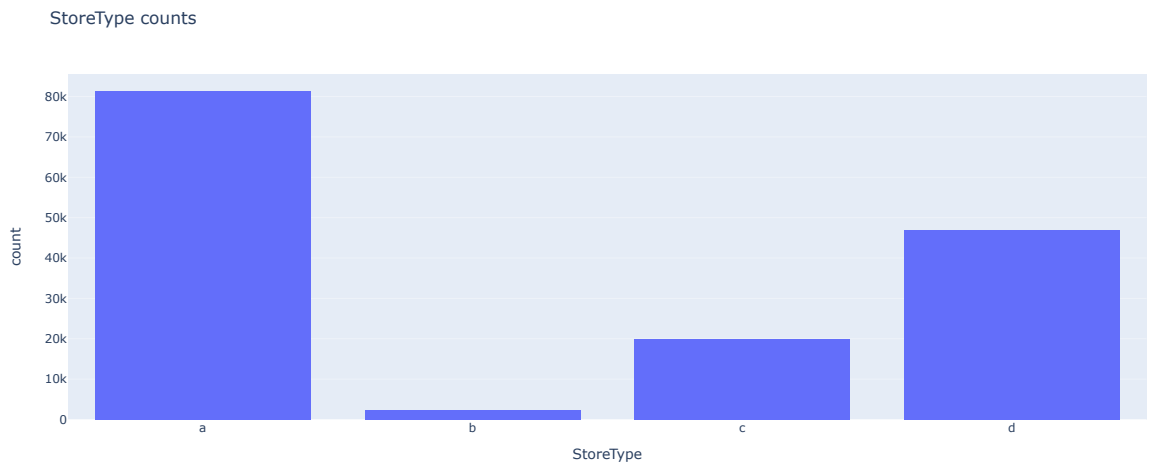
- The promotion weeks are in Assortment a and c three times more effective then in assortment b.

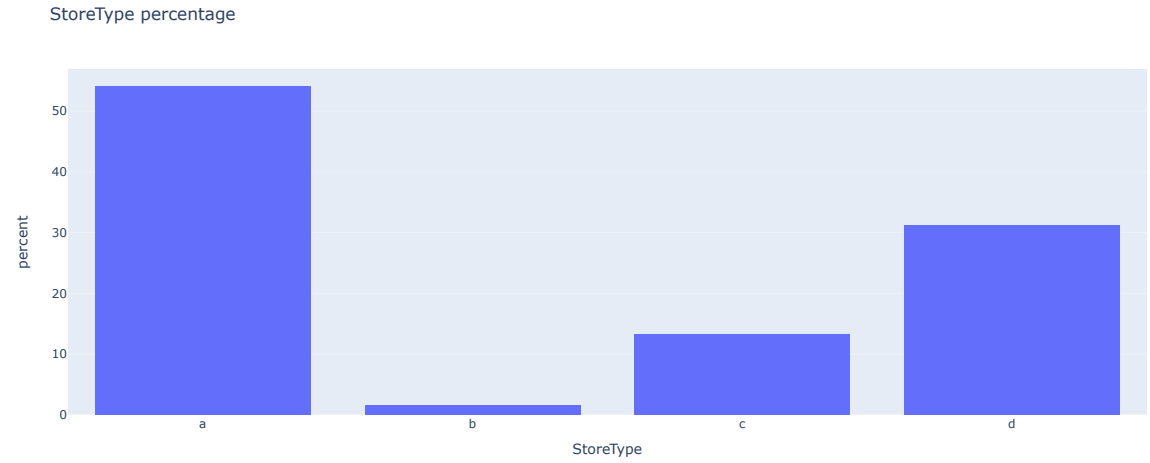
8. Storetype

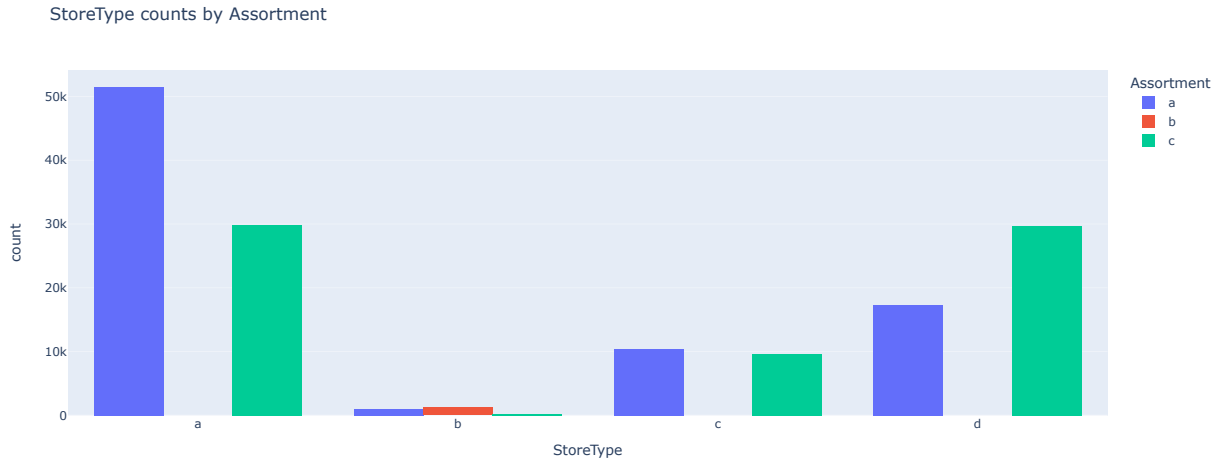
8.1 How many stores are there of which type?

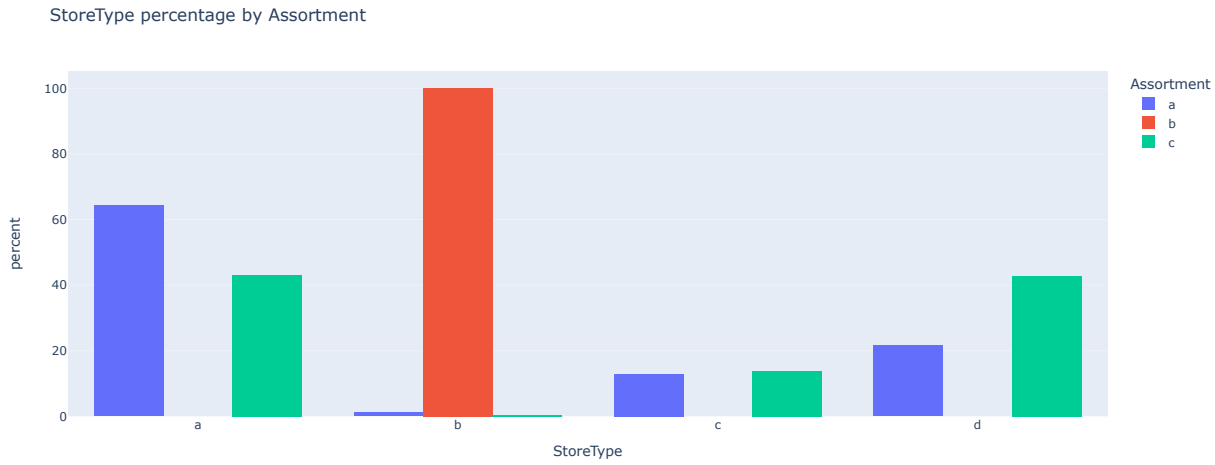
In [60]:
```
fig = px.histogram(weekly_sales_with_store_info.sort_values(by=['StoreType']), x='StoreType', title='StoreType counts')
fig.show()

fig = px.histogram(weekly_sales_with_store_info.sort_values(by=['StoreType']), x='StoreType', title='StoreType percentage', histnorm='percent')
fig.show()
```

```
fig = px.histogram(weekly_sales_with_store_info.sort_values(by=['Assortment', 'StoreType']), x='StoreType', title='StoreType counts by Assortment', color='Assortment', barmode='group')
fig.show()

fig = px.histogram(weekly_sales_with_store_info.sort_values(by=['Assortment', 'StoreType']), x='StoreType', title='StoreType percentage by Assortment', histnorm='percent', color='Assortment', barmode='gro
fig.show()
```









8.2 How many promotion weeks are there in the individual store types?

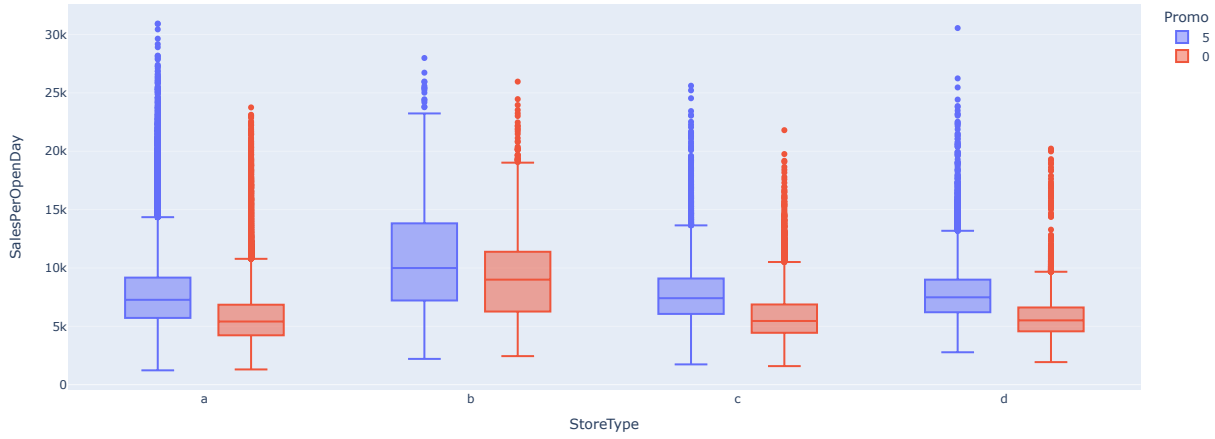
In [61]:
```
# How many promo weeks are there in each year for each StoreType
promo_data = weekly_sales_with_store_info[weekly_sales_with_store_info['IsPromo'] == 1]

# Group by Year, Assortment, and Calendar Week, then count the number of weeks
promo_weeks = promo_data.groupby(['Year', 'StoreType'])['CW'].nunique().reset_index(name='PromoWeeksCount')
promo_weeks
```

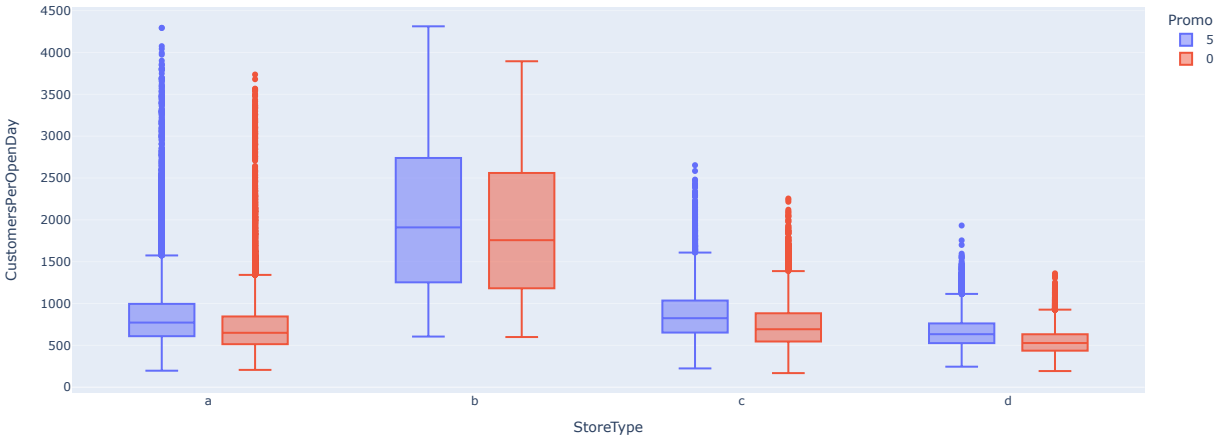
Out[61]:

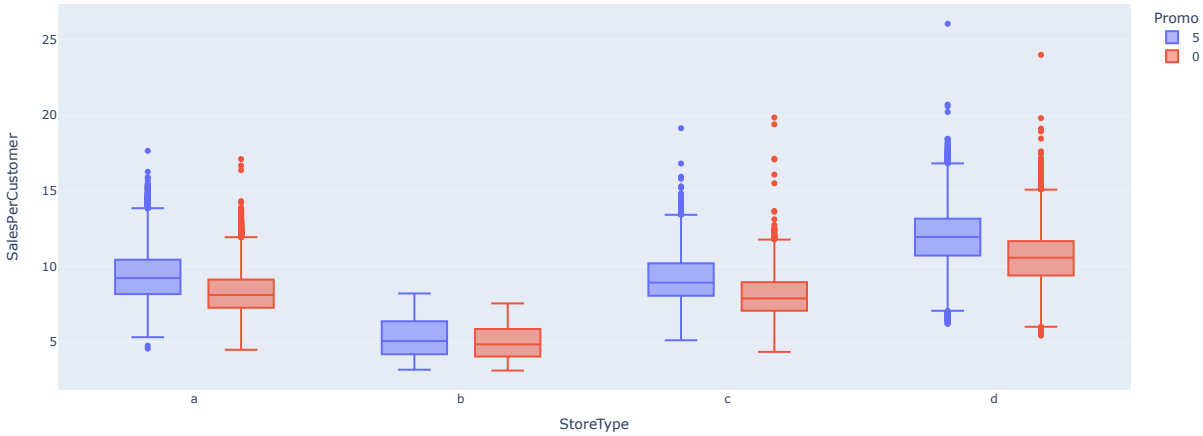
	Year	StoreType	PromoWeeksCount
0	2013	a	27
1	2013	b	27
2	2013	c	27
3	2013	d	27
4	2014	a	28
5	2014	b	28
6	2014	c	28
7	2014	d	28
8	2015	a	17
9	2015	b	17
10	2015	c	17
11	2015	d	17

8.3 Do promotion weeks perform better in individual store types than in others?

In [62]:
```
# To be compareble only weeks with >5 open days are used
used_df = weekly_sales_with_store_info[(weekly_sales_with_store_info['Open'] >= 5)]
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerOpenDay', color='Promo')
fig.show()
# CustomersPerOpenDay
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='CustomersPerOpenDay', color='Promo')
fig.show()
# SalesPerCustomer
fig = px.box(used_df.sort_values('StoreType'), x='StoreType', y='SalesPerCustomer', color='Promo')
fig.show()

median_values = used_df.groupby(['StoreType', 'Promo']).agg({'SalesPerOpenDay': 'median', 'CustomersPerOpenDay': 'median', 'SalesPerCustomer': 'median'}).reset_index()

median_values['SalesPerOpenDayDiff%'] = median_values.groupby('StoreType')['SalesPerOpenDay'].pct_change() * 100
median_values['CustomersPerOpenDayDiff%'] = median_values.groupby('StoreType')['CustomersPerOpenDay'].pct_change() * 100
median_values['SalesPerCustomerDiff%'] = median_values.groupby('StoreType')['SalesPerCustomer'].pct_change() * 100

# pivot table to show the mean values of the SalesPerOpenDay, CustomersPerOpenDay and SalesPerCustomer for each StoreType and Promo
median_values_pivot = median_values.pivot_table(index='StoreType', columns='Promo', values=['SalesPerOpenDay', 'CustomersPerOpenDay', 'SalesPerCustomer', 'SalesPerOpenDayDiff%', 'CustomersPerOpenDayDiff%'
median_values_pivot
```







Out[62]:

	CustomersPerOpenDay		CustomersPerOpenDayDiff%	SalesPerCustomer		SalesPerCustomerDiff%	SalesPerOpenDay		SalesPerOpenDayDiff%
Promo	**0**	**5**	**5**	**0**	**5**	**5**	**0**	**5**	**5**
StoreType									
a	650.00	772.17	18.79	8.11	9.23	13.85	5412.50	7272.25	34.36
b	1757.07	1909.57	8.68	4.83	5.06	4.64	9000.79	9996.00	11.06
c	692.50	824.58	19.07	7.88	8.92	13.27	5459.50	7411.83	35.76
d	527.80	633.50	20.03	10.57	11.94	12.95	5510.50	7487.63	35.88

Result

- The promotion weeks are in StoreType a, c and d three times more effective then in assortment b.