jupyter Vancouver_Real_Estate_Price_Prediction Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Markdown

Vancouver Real Estate Price Prediction 2022 ¶

In [35]:

```
import io
import pandas as pd
import numpy as np
from sklearn import linear_model

# Get the data for apartments for all years
apartments = pd.read_csv('E:/All_Vancouver_Apartments.csv')

apartments.head()
```

Out[35]:

	Year	Date	Bed	Bath	Sqft	price	Address
0	2014	28-Feb-14	1	1	625	249900	#211 680 East 5th Avenue
1	2014	28-Feb-14	1	2	938	525000	#113 West 43rd Avenue
2	2014	28-Feb-14	1	1	718	384900	#104 1775 West 11th Avenue
3	2014	28-Feb-14	1	1	719	389000	#207 1775 West 11th Avenue
4	2014	01-Mar-14	1	1	645	309000	#207 550 East 7th Avenue

In [36]:

```
# Train the model. Here the independent variables are Year, Bedroom, Bathroom, and Sqft and dependent variable is price
model_apartments = linear_model.LinearRegression()
model_apartments.fit(apartments.drop(['price','Date','Address'], axis=1), apartments.price)
```

Out[36]:

```
LinearRegression()
```

In [37]:

```
prediction_apartments = pd.read_csv('E:/predict_apartments_2022.csv') # Read the testing file
prediction_apartments.head()
```

Out[37]:

	Year	Bed	Bath	Sqft
0	2022	1	1	695
1	2022	1	2	675
2	2022	1	1	1050
3	2022	1	1	638
4	2022	1	1	692

In [38]:

```
predicted_price_apt = model_apartments.predict(prediction_apartments)      # Predict the prices for prediction file data using
predicted_price_apt
```

Out[38]:

```
array([ 1110722.65798277,  1381811.43700638,  1998170.11315507,
         968231.09475791,  1103223.10202354,  1278212.74107161,
        1223215.99737081,   810740.41961464,  1060725.61825475,
        1095723.54606435,  1060725.61825475,  2150661.08432552,
         973230.7987307 ,  2410645.6909112 ,  1805681.5102022 ,
         910734.49907067,  1273213.03709882,  1319315.13734636,
        1229320.46583593,   985730.05866271,  1023227.83845872,
        1175718.80962917,  1298211.55696282,  1050726.21030915,
        1050726.21030915,   690747.52426741,  1363207.70860925,
        1493200.01190209,  1526802.85221764,  1223215.99737081,
        1662503.83767796,  3887372.10557473,  1350022.33937782,
        1802495.5489164 ,  1412518.63903785,  1180032.40430257,
        1282526.33574501,  2372461.80181578,  1585008.42609951,
        2144975.27105331,  2307465.65016937,  2422458.84154382,
        1737499.39726996,  2652445.2242927 ,  4425945.04714325,
        1275026.77978581,  2131080.92362723,  1889990.36844042,
        1507513.01452109,  3620992.70752215,  1190031.81224817,
        1595007.83404511,  1575009.01815391,  1771392.56058726,
        1231424.53152466,  2978530.74701715,  2247469.20249575,
        5533379.47711882,  1311419.79508948,  1528906.91790637,
        3933078.41952461,  2849247.42191896,  3165623.85969955,
        3878081.67582378,  1793205.11916545,  4165564.65425989,
        7278985.1418213 ,  1986798.48661065,  3480605.20998603,
        3998074.57117102,  1341836.67411923, 11862318.59738642,
        3775587.74438134,  3835584.19205496,  2818958.87508041,
        1620715.33210379,  1955695.49828151,  2841747.86595976,
        3772797.56939322,  6127948.46358457,  2639259.85506129,
        7090101.15534717,  1908198.3105399 ,  3106732.17651823,
        2579553.73038939,  2555659.97501773,  6257940.76687741,
        2148184.10123438,  2486768.88389084,  2951741.3533614 ])
```

Testing the accuracy of the Model for Test Data for Apartments

In [39]:

```
prediction_apartments['price'] = predicted_price_apt          # Update the prediction file with the prices predicted
prediction_apartments
```

Out[39]:

	Year	Bed	Bath	Sqft	price
0	2022	1	1	695	1.110723e+06
1	2022	1	2	675	1.381811e+06
2	2022	1	1	1050	1.998170e+06
3	2022	1	1	638	9.682311e+05
4	2022	1	1	692	1.103223e+06
...	...	...	...	...	...
85	2022	3	2	1530	2.555660e+06
86	2022	3	2	3011	6.257941e+06
87	2022	3	2	1367	2.148184e+06
88	2022	3	3	1374	2.486769e+06
89	2022	3	3	1560	2.951741e+06

90 rows × 5 columns

In [40]:

```
prediction_apartments.to_csv('apt-predictions-2022.csv')         # write the prediction file data into a new file
predicted_apt_result = pd.read_csv('apt-predictions-2022.csv')   # read the new file
```

In [41]:

```
temp1 = pd.read_csv('apt-predictions-2022.csv',usecols=['Year','Bed','Bath','Sqft'])   # read the independent variables in te
x_test = [list(row) for row in temp1.values]                                        # store the temp1 values as list of lists
temp2 = pd.read_csv('apt-predictions-2022.csv',usecols=['price'])                        # read the target variable in temp2
y_test = [list(row) for row in temp2.values]                                        # store the temp2 values as list of lists
```

In [42]:

```
# Test the accuracy of the model. This model is accurate since it is getting 100% accuracy
model_apartments.score(x_test,y_test)
```

Out[42]:

```
1.0
```

In [43]:

```
# Check for the residual sum square error. Since model is accurate so no error
print('Residual sum of squares: %.2f' % np.mean((model_apartments.predict(x_test) - y_test)) ** 2)
```

```
Residual sum of squares: 0.00
```

Price Prediction for Houses - non-uniform data

In [44]:

```
# Get the data for houses for all years
houses = pd.read_csv('E:/All_Vancouver_Houses.csv')

houses.head()
```

Out[44]:

	Year	Date	Bed	Bath	Sqft	price	Address
0	2014	07-Feb-14	1	1	603	379000	3083 West 4th Avenue
1	2014	28-Feb-14	2	3	1476	1088000	281 Smithe Street
2	2014	01-Mar-14	2	2	1055	769000	2483 West 8th Avenue
3	2014	28-May-14	2	3	1070	988000	2315 Balsam Street
4	2014	30-Sep-14	2	2	1548	419900	3-3333 South Main Street

In [45]:

```
# Train the model. Here the independent variables are Year, Bedroom, Bathroom, and Sqft and dependent variable is price
model_houses = linear_model.LinearRegression()
model_houses.fit(houses.drop(['price', 'Date', 'Address'], axis=1), houses.price)
```

Out[45]:

```
LinearRegression()
```

In [46]:

```
prediction_houses = pd.read_csv('E:/predict_houses_2022.csv')   # Read the testing file
prediction_houses
```

Out[46]:

	Year	Bed	Bath	Sqft
0	2022	1	1	692
1	2022	1	2	848
2	2022	1	2	994
3	2022	1	3	1258
4	2022	1	2	813
5	2022	2	3	1145

```
In [47]: print(len(prediction_houses))                    # Check the number of rows present for the prediction file
         print(len(prediction_houses.columns))            # Check the number of columns present for the prediction file

         50
         4
```

```
In [48]: predicted_price_house = model_houses.predict(prediction_houses)     # Predict the prices for prediction file data using the m
         predicted_price_house

Out[48]: array([ 910341.67714572, 1219056.39495385, 1439558.24104583,
                1911383.91349736, 1166196.36335646, 1403525.02906278,
                2029677.72721998, 1972286.83577138,  906918.52281198,
                3945337.40558891, 1959524.17097051, 6543521.84453785,
                2732790.91890955, 3040273.14086533, 3884523.35980657,
                5281752.14742142, 2414611.9002994 , 2212511.28437693,
                3624967.72849737, 1494231.10257855, 2423271.03890221,
                2270115.84265211, 1635517.67271867, 1662980.62259062,
                2033955.54886003, 4676216.09046866, 2179712.31245458,
                2454306.68599632, 3443822.21094084, 3162292.65223993,
                1081947.60867243,  959892.18345459, 3438610.97922532,
                4860962.5153423 , 4268652.37068677, 3845369.53680801,
                 581917.94808789, 2873638.99410684, 1993480.35351743,
                3010181.03756918, 1460224.38735738, 2850026.53222179,
                4763221.218886  ,  556456.74242429,  917415.24390365,
                5665414.45326388, 4300329.01826061, 7312054.1995216 ,
                9632684.35809143, 8486928.93972456])
```

```
In [49]: prediction_houses['price'] = predicted_price_house           # Update the prediction file with the prices predicted
         prediction_houses
```

Out[49]:

	Year	Bed	Bath	Sqft	price
0	2022	1	1	692	9.103417e+05
1	2022	1	2	848	1.219056e+06
2	2022	1	2	994	1.439558e+06
3	2022	1	3	1258	1.911384e+06
4	2022	1	2	813	1.166196e+06
5	2022	2	3	1145	1.403525e+06
6	2022	2	2	1608	2.029678e+06
7	2022	2	2	1570	1.972287e+06
8	2022	2	1	913	9.069185e+05
9	2022	2	3	2828	3.945337e+06
10	2022	3	4	1688	1.959524e+06
11	2022	3	2	4820	6.543522e+06
12	2022	3	4	2200	2.732791e+06
13	2022	3	3	2452	3.040273e+06

Testing the accuracy of the Model for Test Data for Houses

```
In [50]: prediction_houses.to_csv('houses-predictions-2022.csv')                   # write the prediction file data into a new
         predicted_houses_result = pd.read_csv('houses-predictions-2022.csv')      # read the new file
         predicted_houses_result
```

Out[50]:

	Unnamed: 0	Year	Bed	Bath	Sqft	price
0	0	2022	1	1	692	9.103417e+05
1	1	2022	1	2	848	1.219056e+06
2	2	2022	1	2	994	1.439558e+06
3	3	2022	1	3	1258	1.911384e+06
4	4	2022	1	2	813	1.166196e+06
5	5	2022	2	3	1145	1.403525e+06
6	6	2022	2	2	1608	2.029678e+06
7	7	2022	2	2	1570	1.972287e+06
8	8	2022	2	1	913	9.069185e+05
9	9	2022	2	3	2828	3.945337e+06
10	10	2022	3	4	1688	1.959524e+06
11	11	2022	3	2	4820	6.543522e+06

```
In [51]: temp1 = pd.read_csv('houses-predictions-2022.csv',usecols=["Year", "Bed", "Bath", "Sqft"])      # read the independent v
         x_test = [list(row) for row in temp1.values]                                      # store the temp1 values as list of lists
         temp2 = list(predicted_houses_result['price'])                                     # read the target variable in temp2
         y_test = [[i] for i in temp2]                                                      # store the temp2 values as list of lists
```

In [52]:
```
# Test the accuracy of the model. The model is accurate since it is getting 100% accuracy
model_houses.score(x_test,y_test)
```

Out[52]:
```
1.0
```

In [53]:
```
# Check for the residual sum square error. Since model is accurate so no error
print('Residual sum of squares: %.2f' % np.mean((model_houses.predict(x_test) - y_test)) ** 2)
```

```
Residual sum of squares: 0.00
```

Price Prediction for Houses - uniform data

In [54]:
```
# Get the data for houses for all years
house_uniform = pd.read_csv('E:/All_Vancouver_Houses - Uniform data.csv')

house_uniform.head()
```

Out[54]:

	Year	Date	Bed	Bath	Sqft	price	Address
0	2014	02-Jul-14	2	3	1755	795500	2967 Wall Street
1	2014	01-Mar-14	2	2	1055	769000	2483 West 8th Avenue
2	2014	28-May-14	2	3	1070	988000	2315 Balsam Street
3	2014	02-Jul-14	2	2	1193	645900	2977 Wall Street
4	2014	26-Nov-14	2	3	2069	1488000	2176 West 15th Avenue

In [55]:
```
# Train the model. Here the independent variables are Year, Bedroom, Bathroom, and Sqft and dependent variable is price
model_houses_unif = linear_model.LinearRegression()
model_houses_unif.fit(house_uniform.drop(['price', 'Date', 'Address'], axis=1), house_uniform.price)
```

Out[55]:
```
LinearRegression()
```

In [56]:
```
prediction_houses_unif = pd.read_csv('E:/predict_houses_2022 - Uniform data.csv')    # Read the testing file
prediction_houses_unif
```

Out[56]:

	Year	Bed	Bath	Sqft
0	2022	2	1	692
1	2022	2	2	848
2	2022	2	2	994
3	2022	2	3	1258
4	2022	2	2	813
5	2022	2	3	1145
6	2022	2	2	1608
7	2022	2	2	1570
8	2022	2	1	913
9	2022	2	4	2828
10	2022	3	1	1688

In [57]:
```
print(len(prediction_houses_unif))                              # Check the number of rows present for the prediction file
print(len(prediction_houses_unif.columns))
```

```
60
4
```

In [58]:
```
predicted_price_house_unif = model_houses_unif.predict(prediction_houses_unif)     # Predict the prices for prediction file a
predicted_price_house_unif
```

Out[58]:
```
array([ 955963.19200608, 1259148.42764348, 1449231.18372732,
       1893025.58139935, 1213580.64365077, 1745906.73593718,
       2248620.3086279 , 2199146.71457869, 1243691.19950286,
       4037149.290342  , 1967780.0560897 , 6145528.86779249,
       2934620.97716412, 3162625.90921313, 3890408.51641089,
       5422621.9831658 , 2913956.00228149, 2965867.45761627,
       4094480.55186436, 2083486.40901423, 2927187.4285821 ,
       2758101.67155179, 1984917.29244679, 2234723.12397847,
       2602673.1697202 , 5460258.18232009, 3007907.50308347,
       3218655.23404995, 4097628.50885186, 3817878.13355356,
       2104409.49070248, 1925074.25227362, 4173020.87188581,
       5287976.71093521, 5151745.23521528, 4777285.50604597,
       1889921.96176496, 3902550.06547153, 3054823.27112174,
       4257480.74435821, 2526781.06042284, 4036197.55518231,
       5574282.8170211 , 1910798.99609312, 2221961.863929  ,
       6891012.68232569, 5941900.8385919 , 8465552.17135864,
       3040880.03911218, 3083677.93764842, 1199017.70065242,
       1454031.2789261 , 3771312.56704104, 4252863.12859273,
       4274830.04016086, 5132640.30382377, 3786271.75892523,
       4608929.73207948, 6060423.12433302, 4778513.52536616])
```

In [59]:

```
prediction_houses_unif['price'] = predicted_price_house_unif           # Update the prediction file with the prices predicte
prediction_houses_unif
```

Out[59]:

	Year	Bed	Bath	Sqft	price
0	2022	2	1	692	9.559632e+05
1	2022	2	2	848	1.259148e+06
2	2022	2	2	994	1.449231e+06
3	2022	2	3	1258	1.893026e+06
4	2022	2	2	813	1.213581e+06
5	2022	2	3	1145	1.745907e+06
6	2022	2	2	1608	2.248620e+06
7	2022	2	2	1570	2.199147e+06
8	2022	2	1	913	1.243691e+06
9	2022	2	4	2828	4.037149e+06
10	2022	3	1	1688	1.967780e+06

Testing the accuracy of the Model for Test Data for Houses with uniform data

In [60]:

```
prediction_houses_unif.to_csv('houses-predictions-2022-uniform-data.csv')                        # write the prediction fil
predicted_houses_result_unif = pd.read_csv('houses-predictions-2022-uniform-data.csv')           # read the new file
predicted_houses_result_unif.head()
```

Out[60]:

	Unnamed: 0	Year	Bed	Bath	Sqft	price
0	0	2022	2	1	692	9.559632e+05
1	1	2022	2	2	848	1.259148e+06
2	2	2022	2	2	994	1.449231e+06
3	3	2022	2	3	1258	1.893026e+06
4	4	2022	2	2	813	1.213581e+06

In [61]:

```
temp1 = pd.read_csv('houses-predictions-2022-uniform-data.csv',usecols=["Year", "Bed", "Bath", "Sqft"])          # read the
x_test = [list(row) for row in temp1.values]                                       # store the temp1 values as list of lists
temp2 = list(predicted_houses_result_unif['price'])                                    # read the target variable in temp2
y_test = [[i] for i in temp2]                                                      # store the temp2 values as list of lists
```

In [62]:

```
# Test the accuracy of the model. The model is accurate since it is getting 100% accuracy
model_houses_unif.score(x_test,y_test)
```

Out[62]:

```
1.0
```

In [63]:

```
# Check for the residual sum square error. Since model is accurate so no error
print('Residual sum of squares: %.2f' % np.mean((model_houses_unif.predict(x_test) - y_test)) ** 2)
```

```
Residual sum of squares: 0.00
```

Plotting graph for apartments

In [64]:

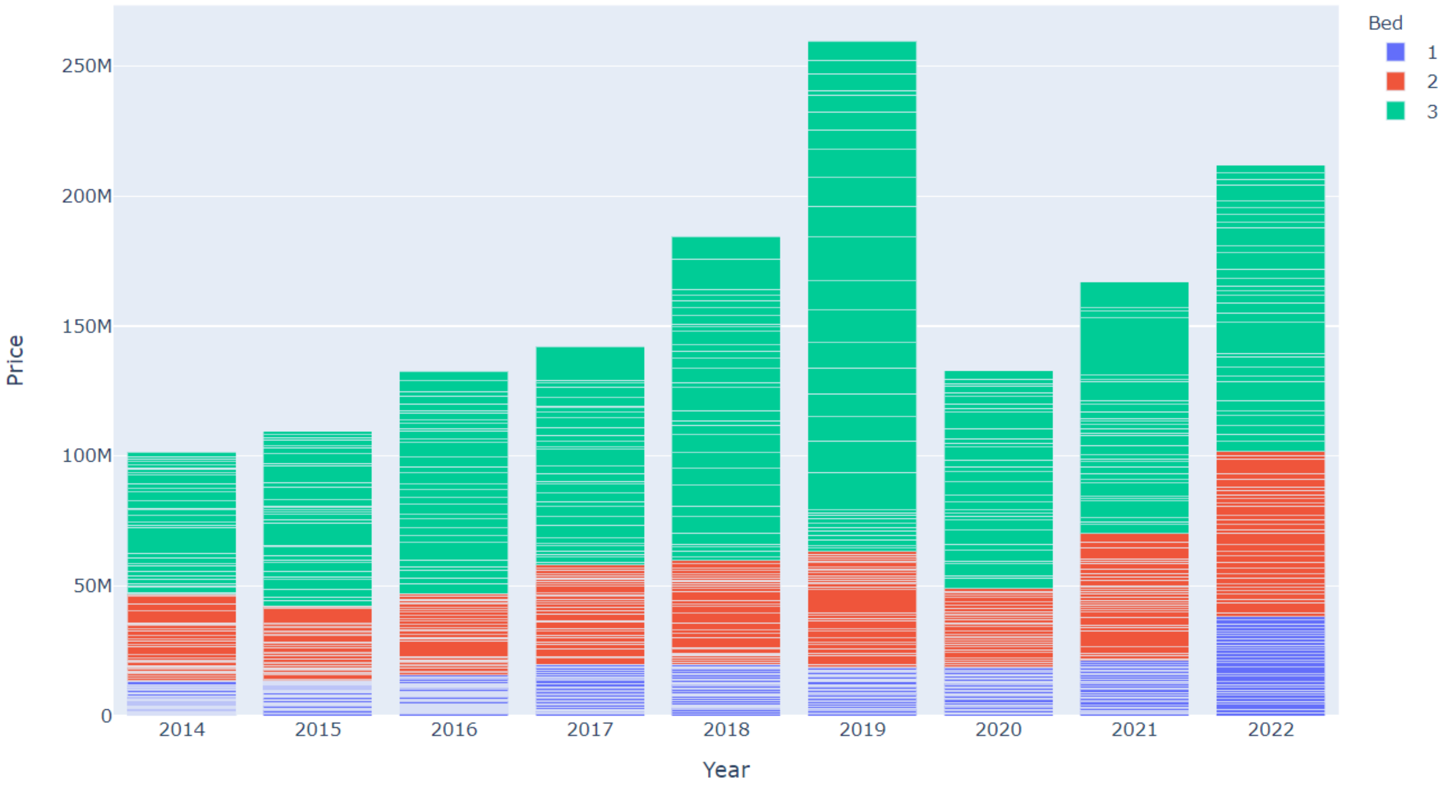
```
# Combined data for years 2014 to 2022 (30 1-bedroom apartments, 30 2-bedroom apartments, 30 3-bedroom apartments for each ye
ap = pd.read_csv('E:/Vancouver_Apartments_2014-2022.csv')
ap.head()
```

Out[64]:

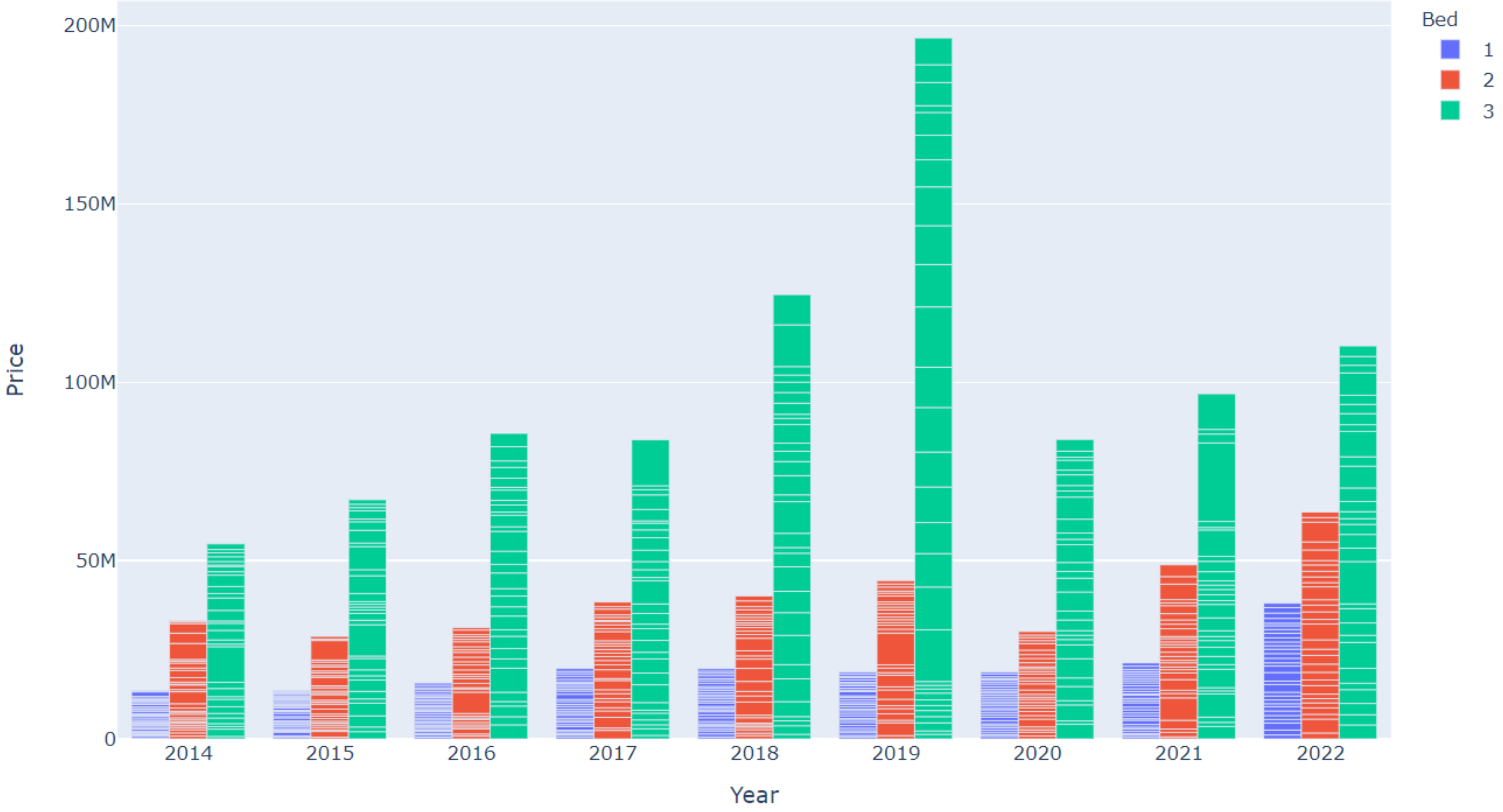
	Year	Bed	Bath	Sqft	price
0	2014	1	1	625	249900
1	2014	1	2	938	525000
2	2014	1	1	718	384900
3	2014	1	1	719	389000
4	2014	1	1	645	309000

In [65]:

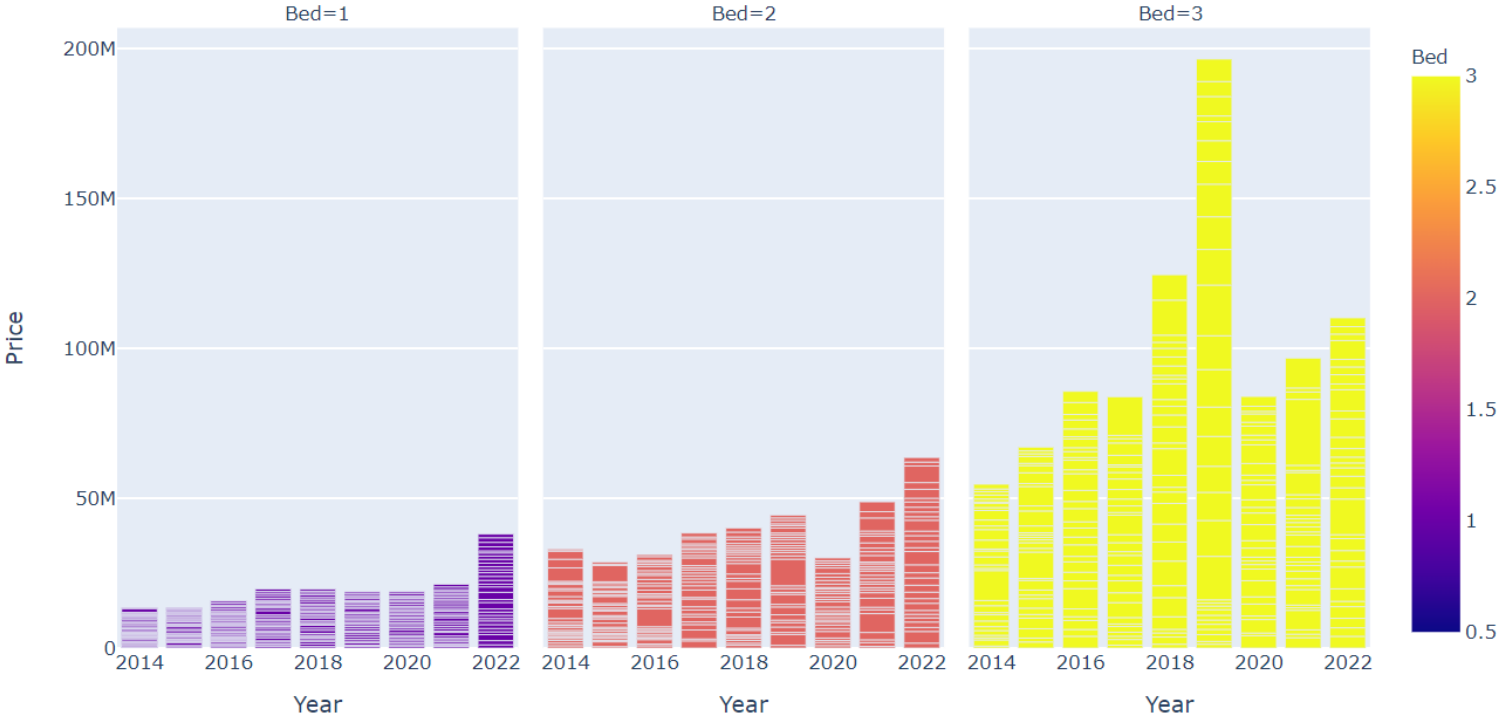
```
# Total price of all 1 bedroom, 2 bedroom, 3 bedroom apartments in a given year as stack
import plotly.express as px
data = pd.DataFrame()
data['Bed'] = ap["Bed"].astype(str)
data['Price'] = ap['price']
data['Year'] = ap["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bed',barmode = "stack", height=600)
fig.show()
```



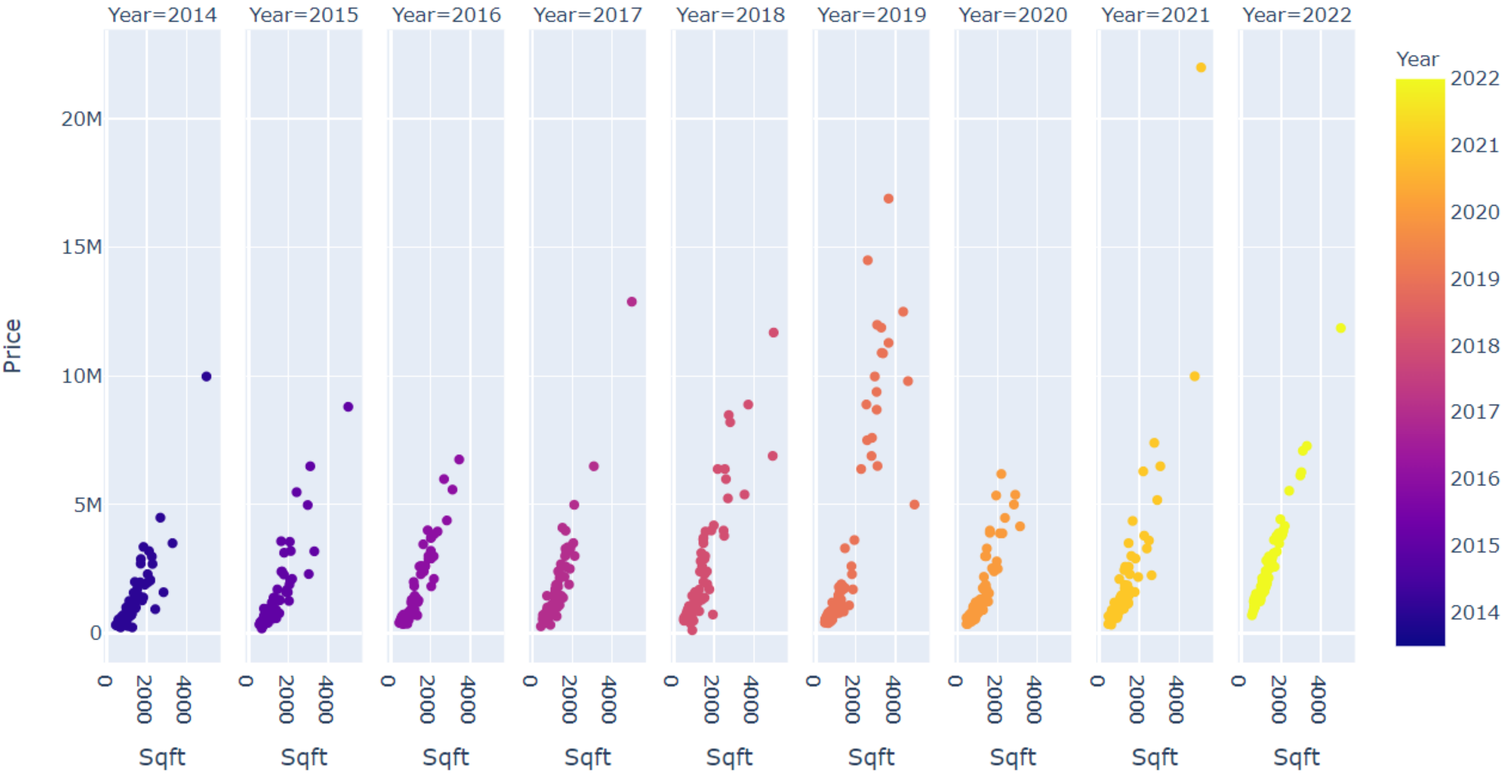
```
In [66]:  # Total price of all 1 bedroom, 2 bedroom, 3 bedroom apartments in a given year categorized on the basis of year
          import plotly.express as px
          data = pd.DataFrame()
          data['Bed'] = ap["Bed"].astype(str)
          data['Price'] = ap['price']
          data['Year'] = ap["Year"]
          fig = px.bar(data, x='Year', y='Price', color='Bed',barmode = "group", height=600)
          fig.show()
```



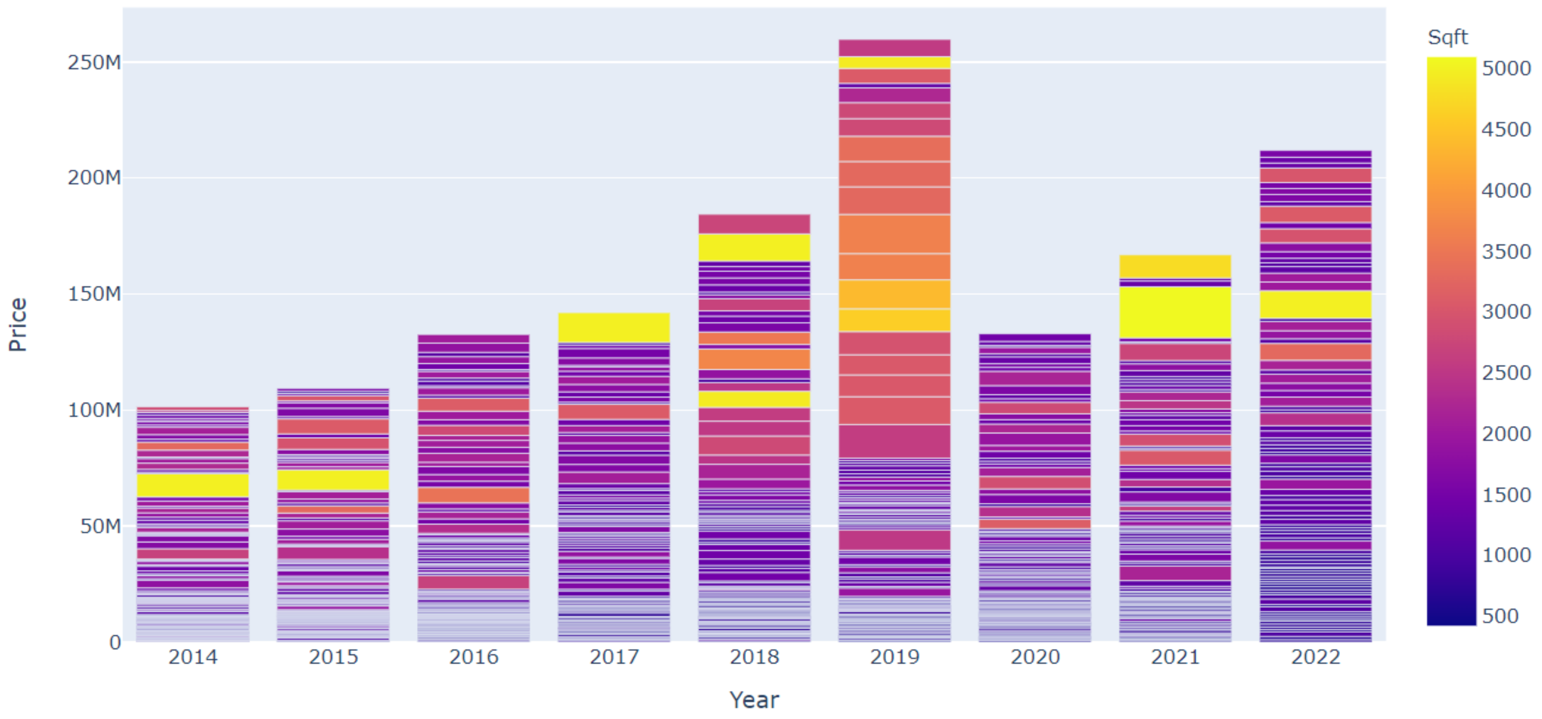
```
In [67]:  # Total price of all 1 bedroom, 2 bedroom, 3 bedroom apartments in a given year categorized on basis of beds
          data = pd.DataFrame()
          data['Bed'] = ap["Bed"].apply(pd.to_numeric)
          data['Price'] = ap["price"].apply(pd.to_numeric)
          data['Year'] = ap["Year"].apply(pd.to_numeric)
          fig = px.bar(data, x='Year', y='Price', color='Bed', facet_col = 'Bed')
          fig.show()
```



In [68]:
```
# Price of all the apartments based on sqft
data = pd.DataFrame()
data['Sqft'] = ap["Sqft"]
data['Price'] = ap["price"]
data['Year'] = ap["Year"]
fig = px.scatter(data, x='Sqft', y='Price', color='Year',
                 facet_col='Year', width=950)
fig.show()
```

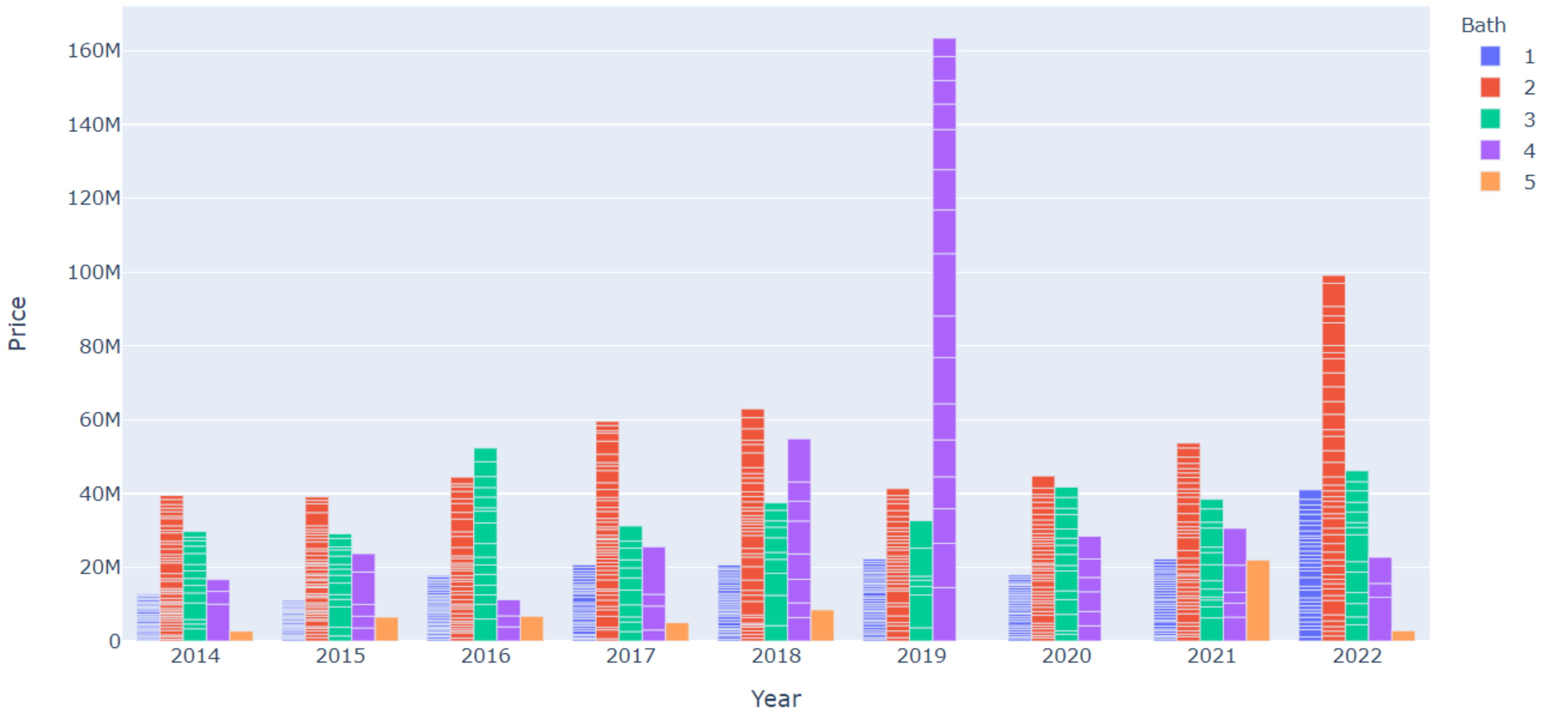


In [69]:
```
# Price of all the apartments based on sqft as stack
data = pd.DataFrame()
data['Sqft'] = ap["Sqft"]
data['Price'] = ap["price"]
data['Year'] = ap["Year"]
fig = px.bar(data, x='Year', y='Price',color='Sqft')
fig.show()
```



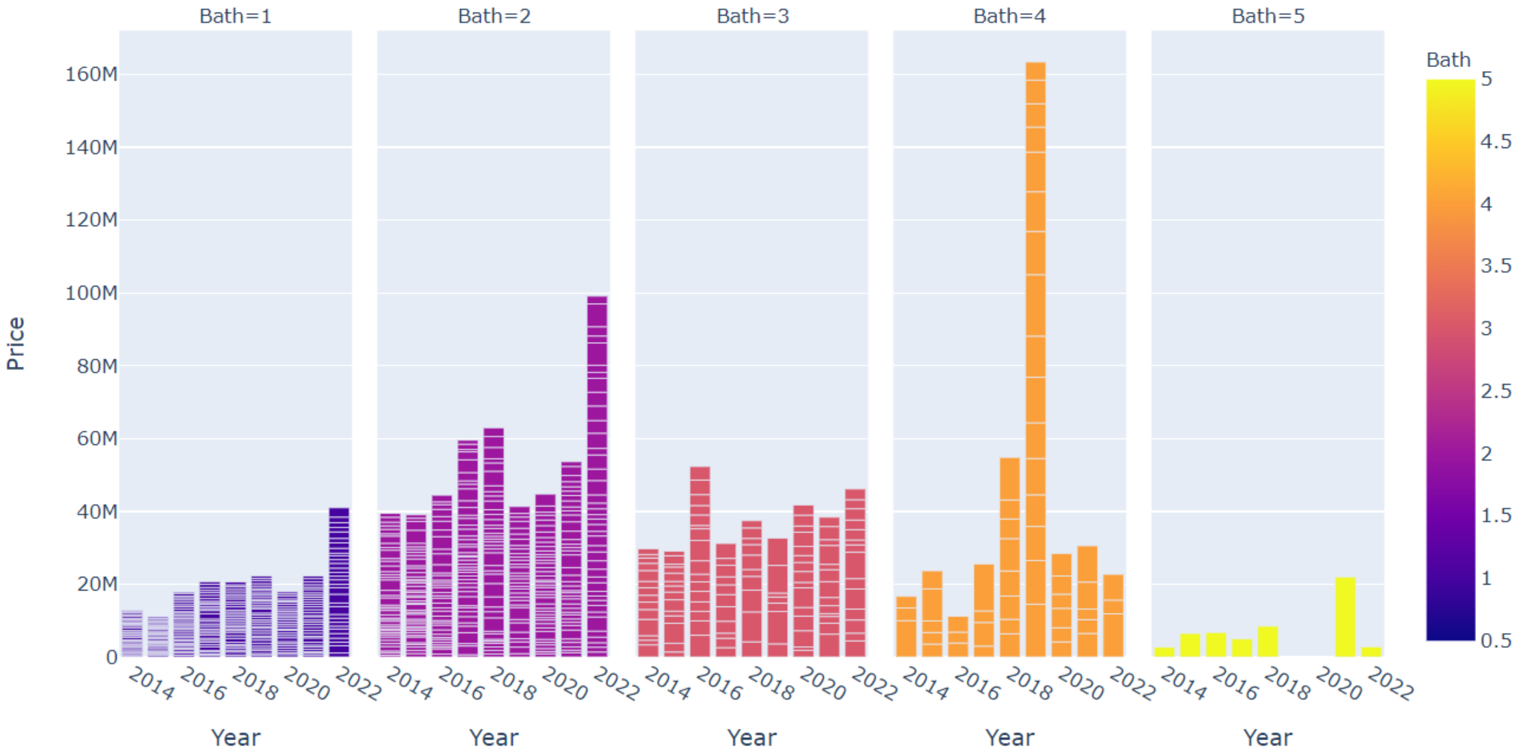
In [70]:

```
# Total price of all apartments based on number of baths in a given year categorized on the basis of year
data = pd.DataFrame()
data['Bath'] = ap["Bath"].astype(str)
data['Price'] = ap["price"]
data['Year'] = ap["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bath', barmode = 'group')
fig.show()
```

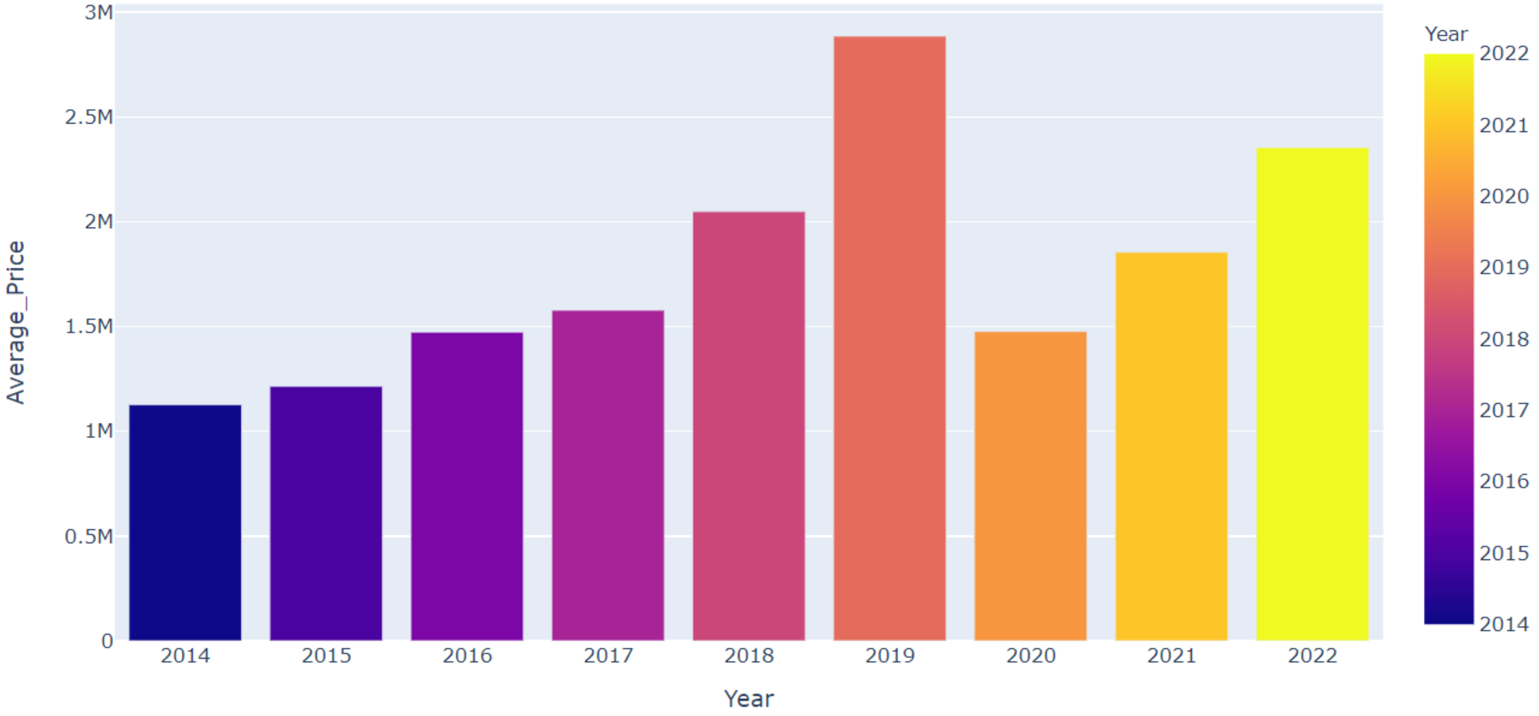


In [71]:

```
# Total price of all apartments based on number of baths in a given year categorized on the basis of number of baths
data = pd.DataFrame()
data['Bath'] = ap["Bath"]
data['Price'] = ap["price"]
data['Year'] = ap["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bath', facet_col = "Bath")
fig.show()
```



In [72]:
```python
# Average price of apartments for each year
ap_avg = pd.read_csv('E:/Vancouver_Apartments_Avg_Price.csv')
fig = px.bar(ap_avg, x='Year', y='Average_Price', color='Year')
fig.show()
```



Plotting graph for houses - non-uniform data

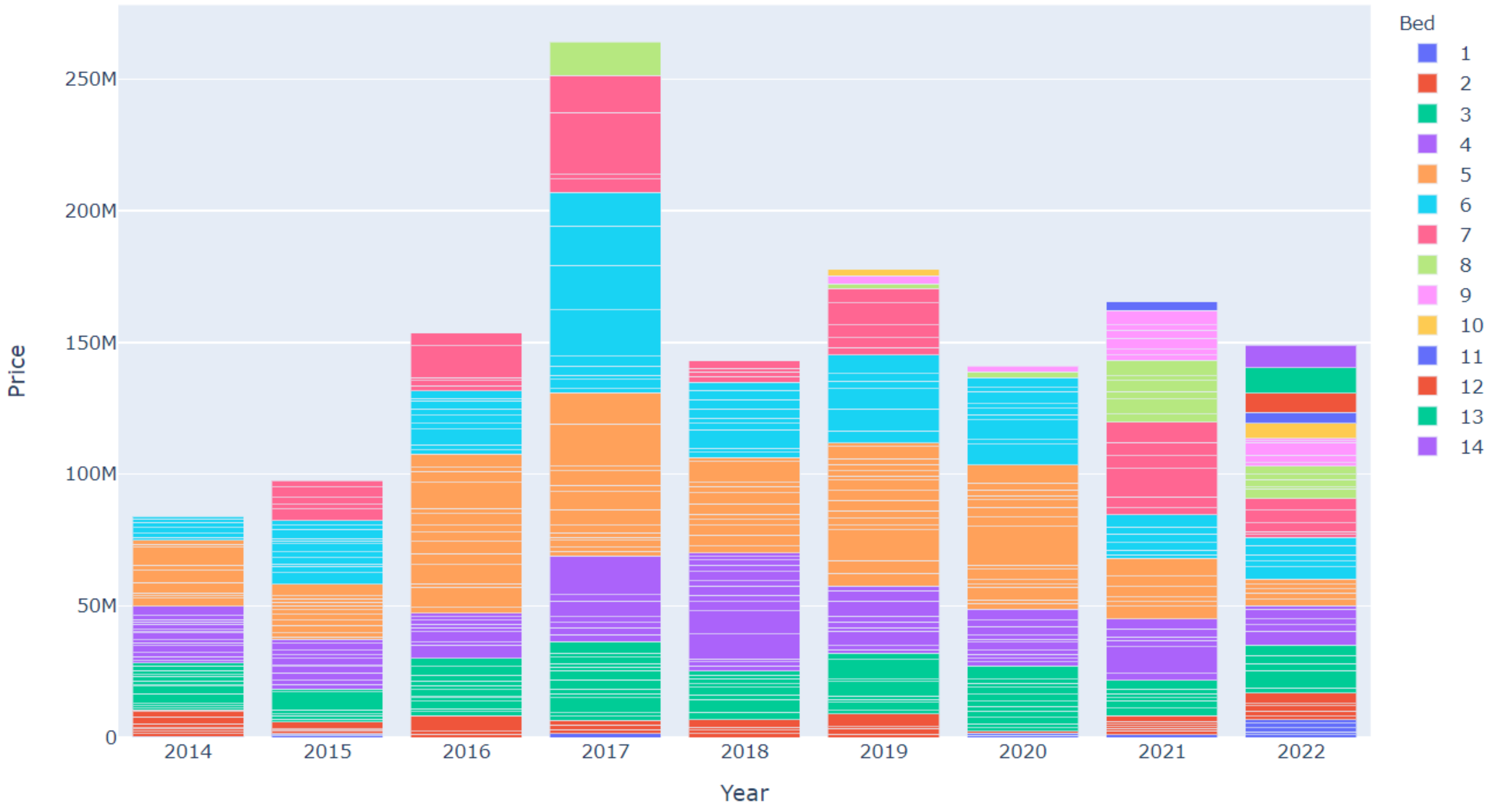
In [73]:
```python
# Combined data for years 2014 to 2022 (50 houses for each year)
hs = pd.read_csv('E:/Vancouver_Houses_2014-2022.csv')
hs.head()
```

Out[73]:

	Year	Bed	Bath	Sqft	price
0	2014	1	1	603	379000
1	2014	2	3	1476	1088000
2	2014	2	2	1055	769000
3	2014	2	3	1070	988000
4	2014	2	2	1548	419900

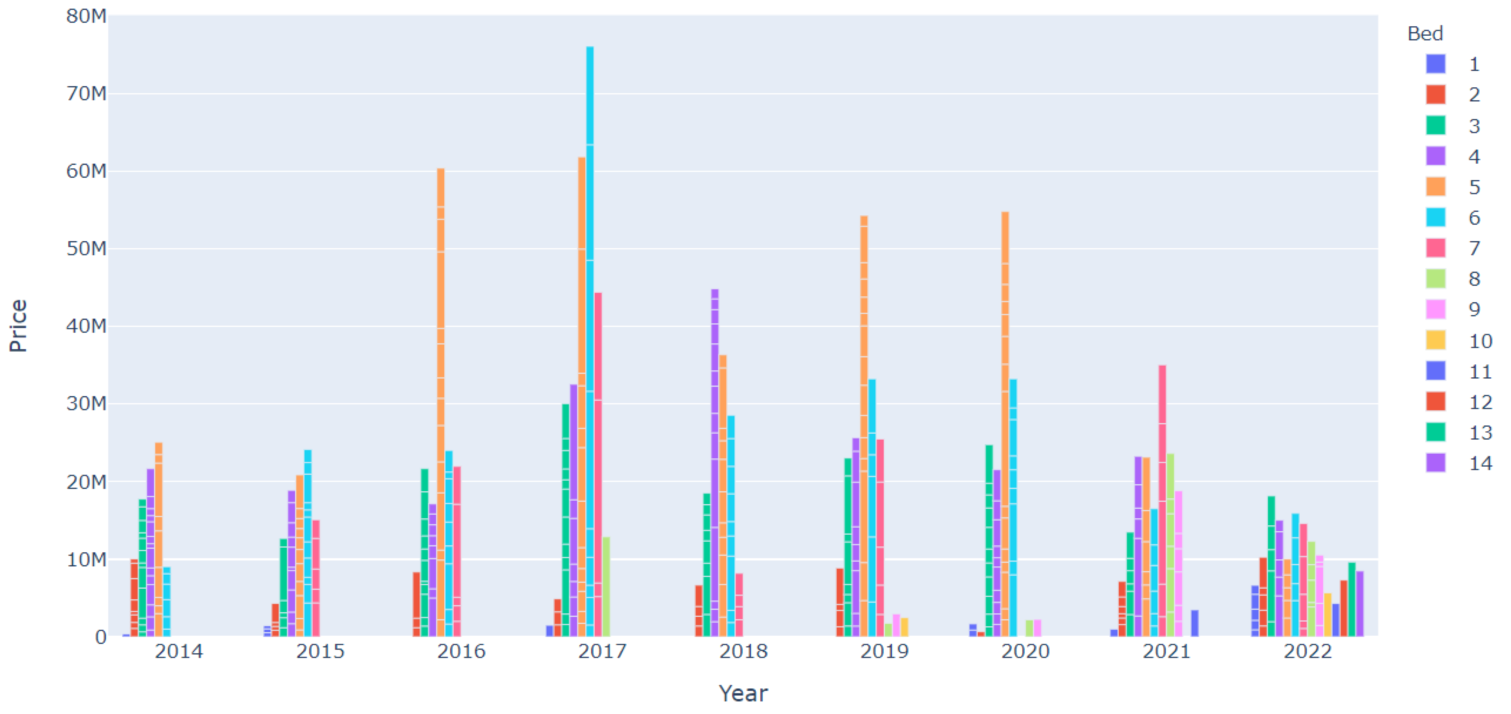
In [74]:

```
# Total price of all 1 bedroom - 14 bedroom houses in a given year as stack
import plotly.express as px
data = pd.DataFrame()
data['Bed'] = hs["Bed"].astype(str)
data['Price'] = hs['price']
data['Year'] = hs["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bed',barmode = "stack", height=600)
fig.show()
```



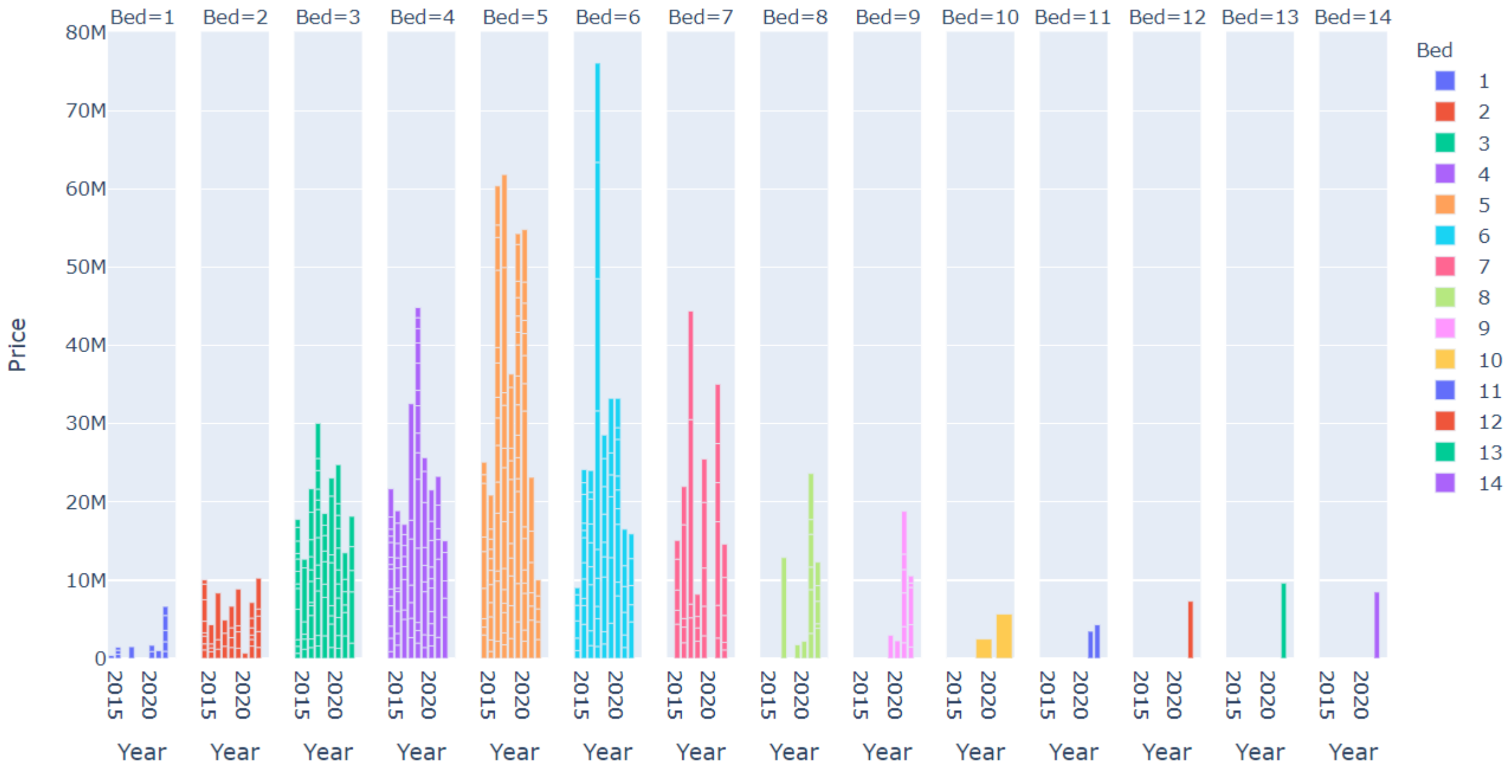
In [75]:

```
# Total price of all 1 bedroom - 14 bedroom houses in a given year categorized on the basis of year
import plotly.express as px
data = pd.DataFrame()
data['Bed'] = hs["Bed"].astype(str)
data['Price'] = hs['price']
data['Year'] = hs["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bed',barmode = "group")
fig.show()
```



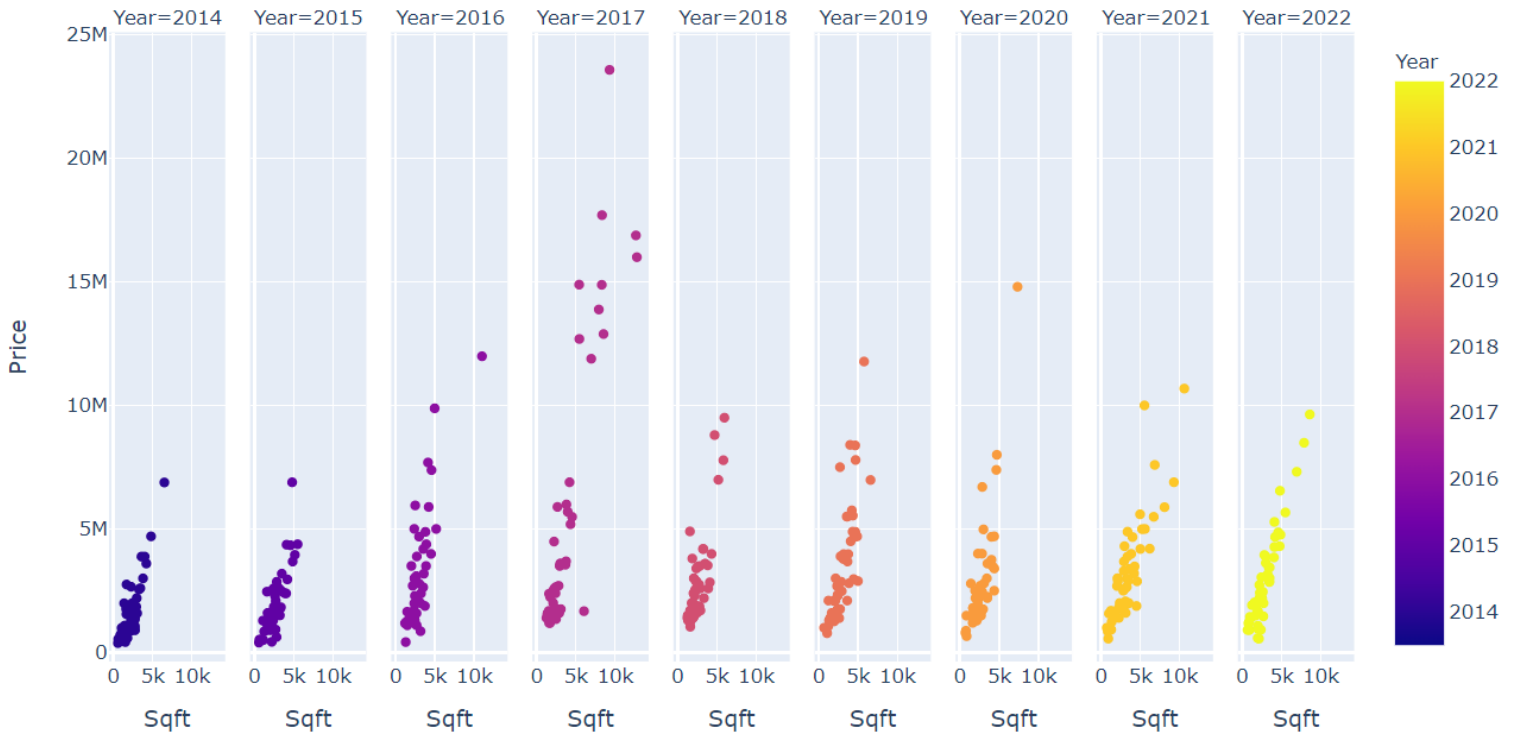
In [76]:

```
# Total price of all 1 bedroom - 14 bedroom houses in a given year categorized on basis of beds
import plotly.express as px
data = pd.DataFrame()
data['Bed'] = hs["Bed"].astype(str)
data['Price'] = hs['price']
data['Year'] = hs["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bed',facet_col = "Bed")
fig.show()
```



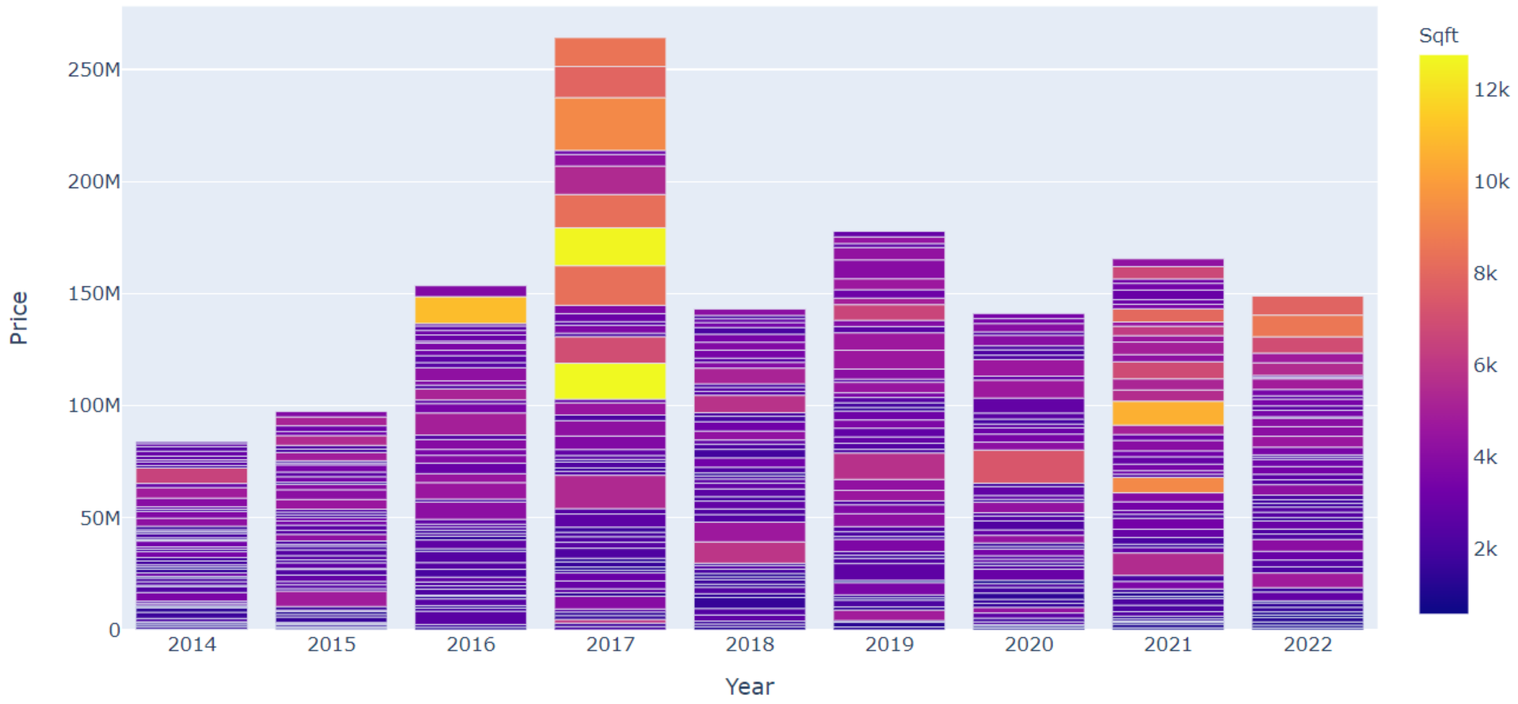
In [77]:

```
# Price of all the houses based on sqft
data = pd.DataFrame()
data['Sqft'] = hs["Sqft"]
data['Price'] = hs["price"]
data['Year'] = hs["Year"]
fig = px.scatter(data, x='Sqft', y='Price', color='Year',
                 facet_col='Year', width=950)
fig.show()
```



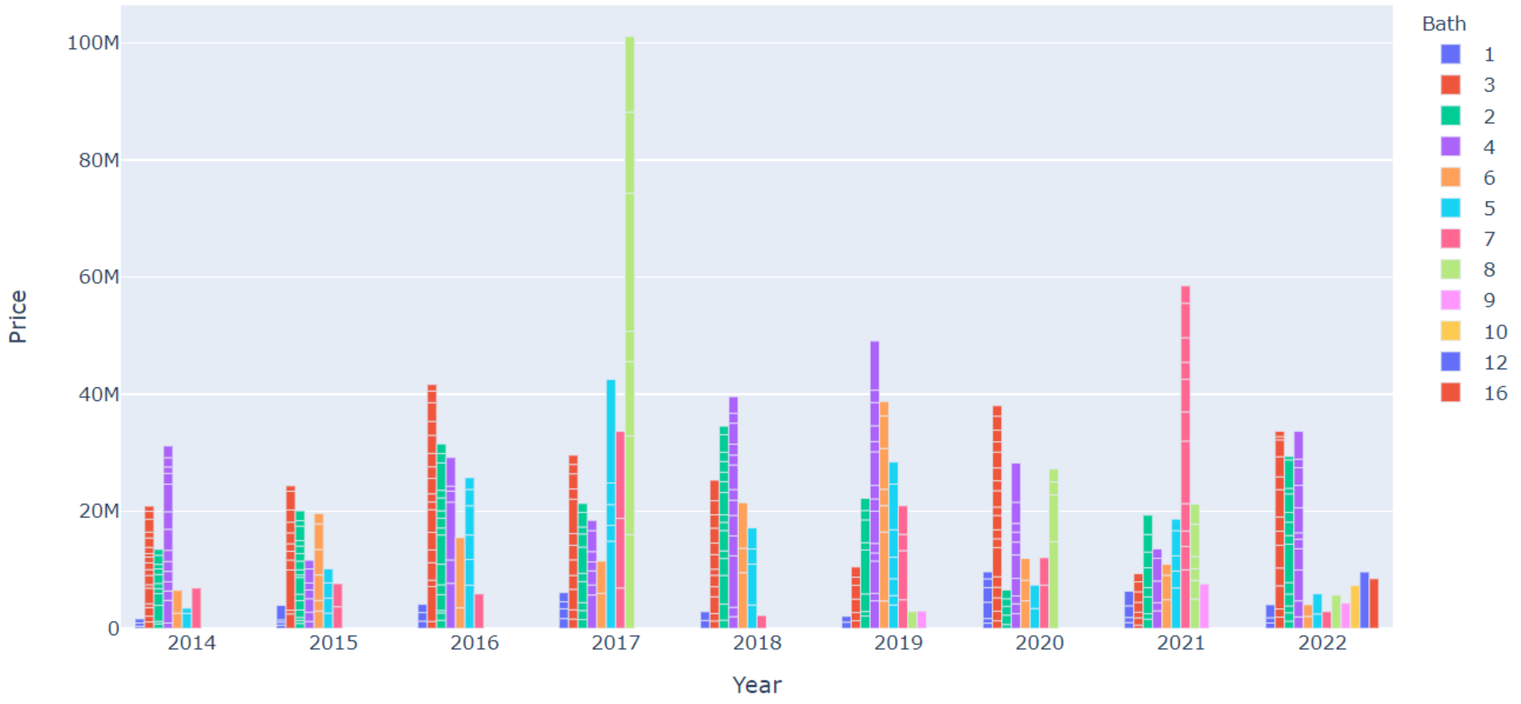
In [78]:

```python
# Price of all the houses based on sqft as stack
data = pd.DataFrame()
data['Sqft'] = hs["Sqft"]
data['Price'] = hs["price"]
data['Year'] = hs["Year"]
fig = px.bar(data, x='Year', y='Price',color='Sqft')
fig.show()
```



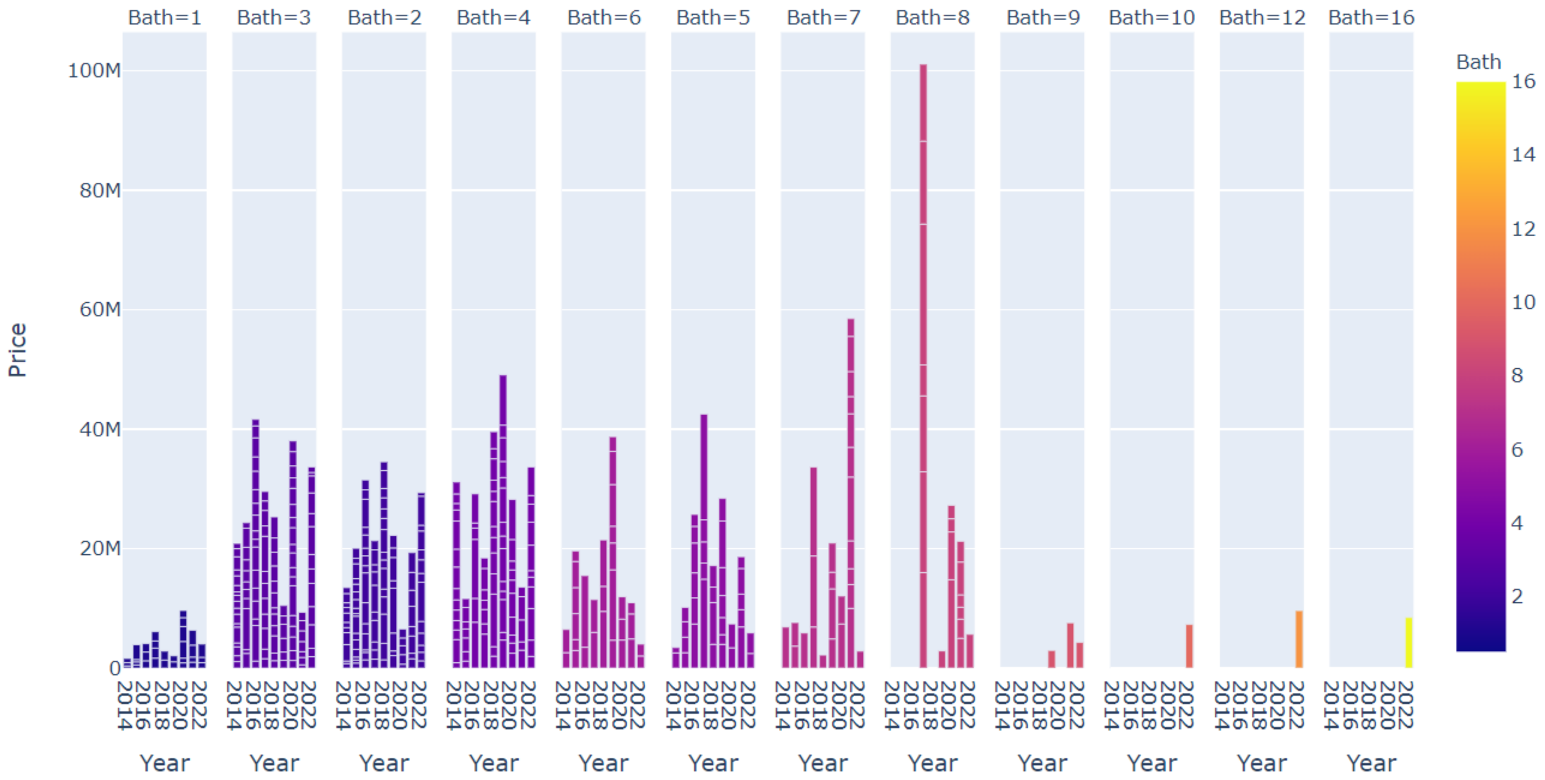
In [79]:

```python
# Total price of all houses based on number of baths in a given year categorized on the basis of year
data = pd.DataFrame()
data['Bath'] = hs["Bath"].astype(str)
data['Price'] = hs["price"]
data['Year'] = hs["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bath', barmode = 'group')
fig.show()
```



In [80]:

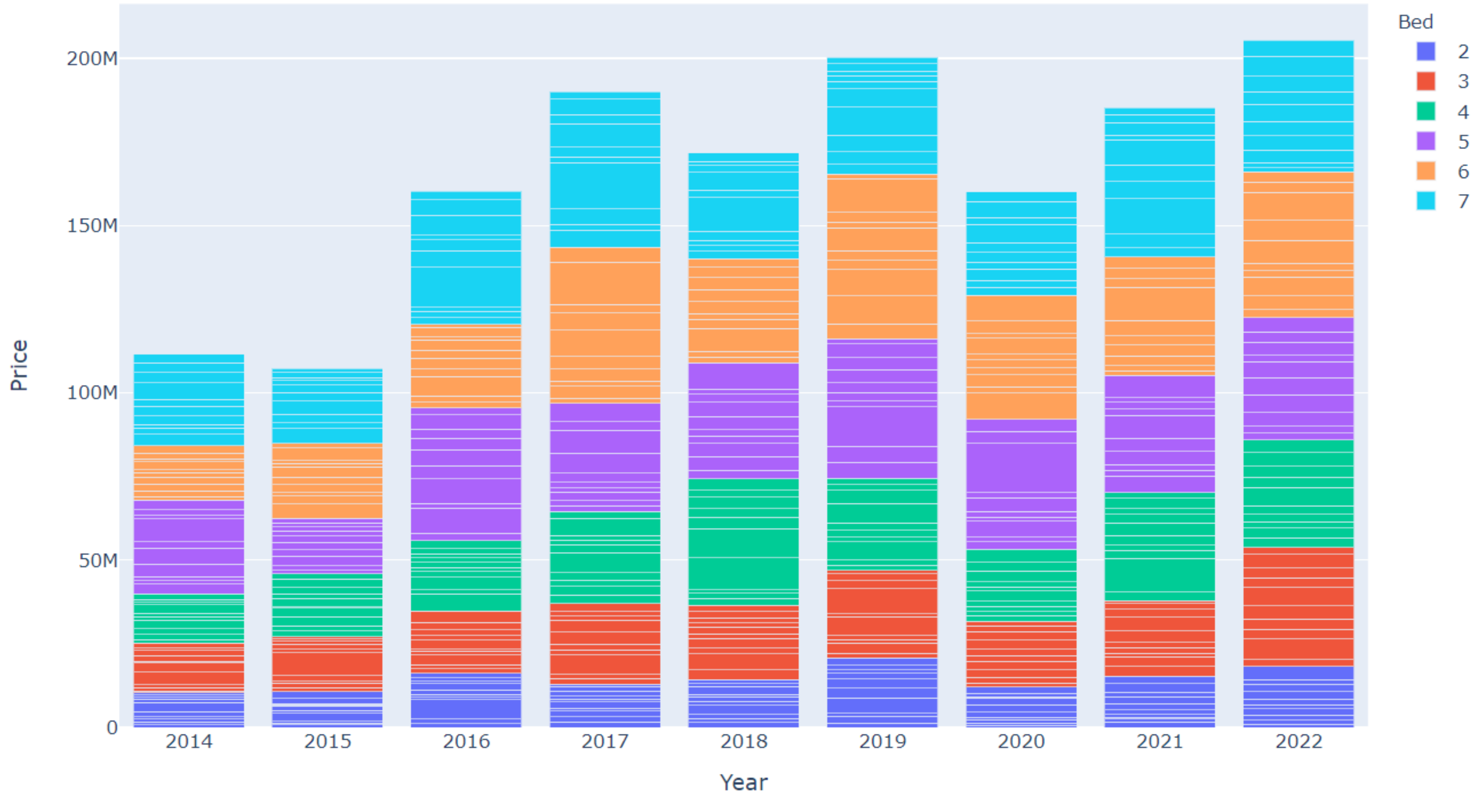
```python
# Total price of all houses based on number of baths in a given year categorized on the basis of number of baths
data = pd.DataFrame()
data['Bath'] = hs["Bath"]
data['Price'] = hs["price"]
data['Year'] = hs["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bath', facet_col = "Bath")
fig.show()
```



Plotting graph for houses - uniform data

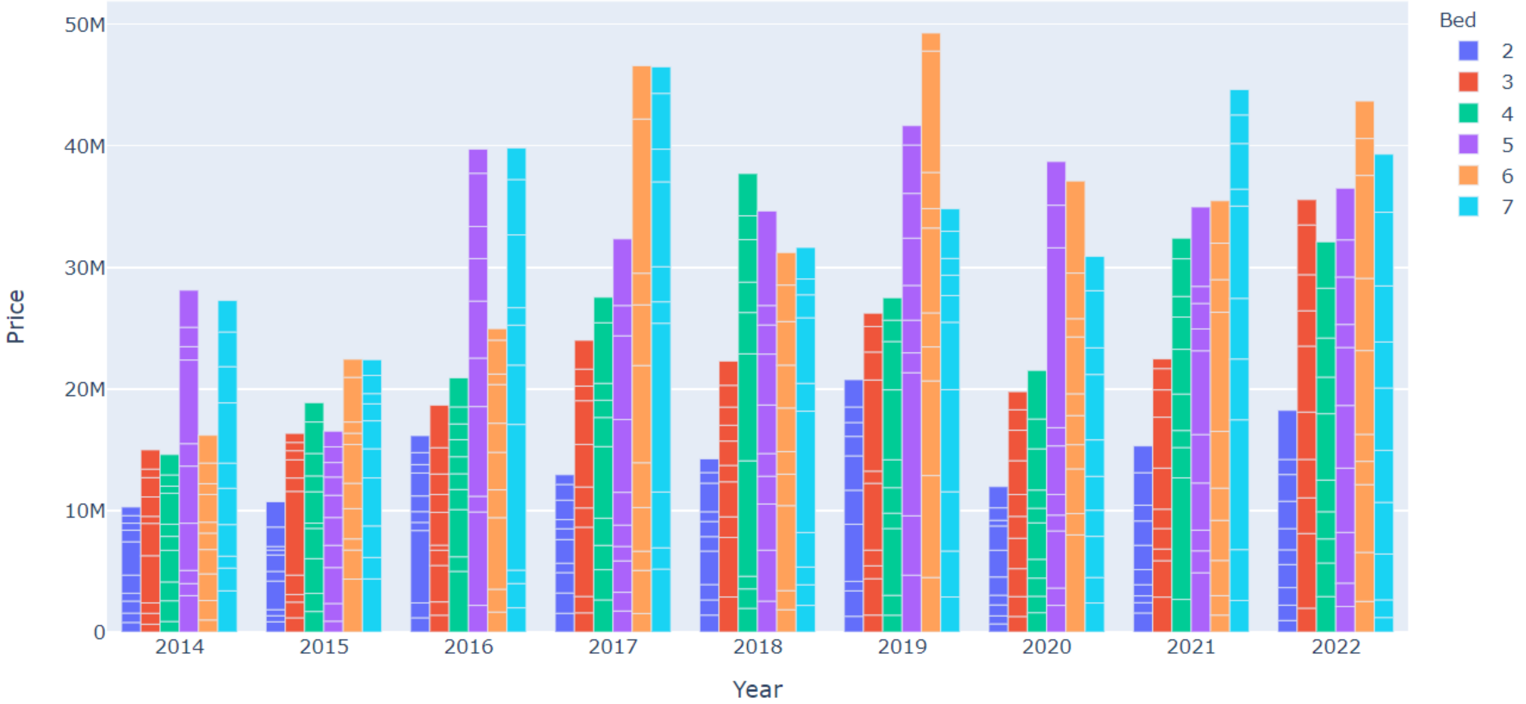
In [81]:

```python
# Total price of all 2 bedroom, 3 bedroom, 4 bedroom, 5 bedroom, 6 bedroom, 7 bedroom houses in a given year as stack
hx = pd.read_csv('E:/Vancouver_Houses_2014-2022 - Uniform data.csv')
import plotly.express as px
data = pd.DataFrame()
data['Bed'] = hx["Bed"].astype(str)
data['Price'] = hx['price']
data['Year'] = hx["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bed',barmode = "stack", height=600)
fig.show()
```



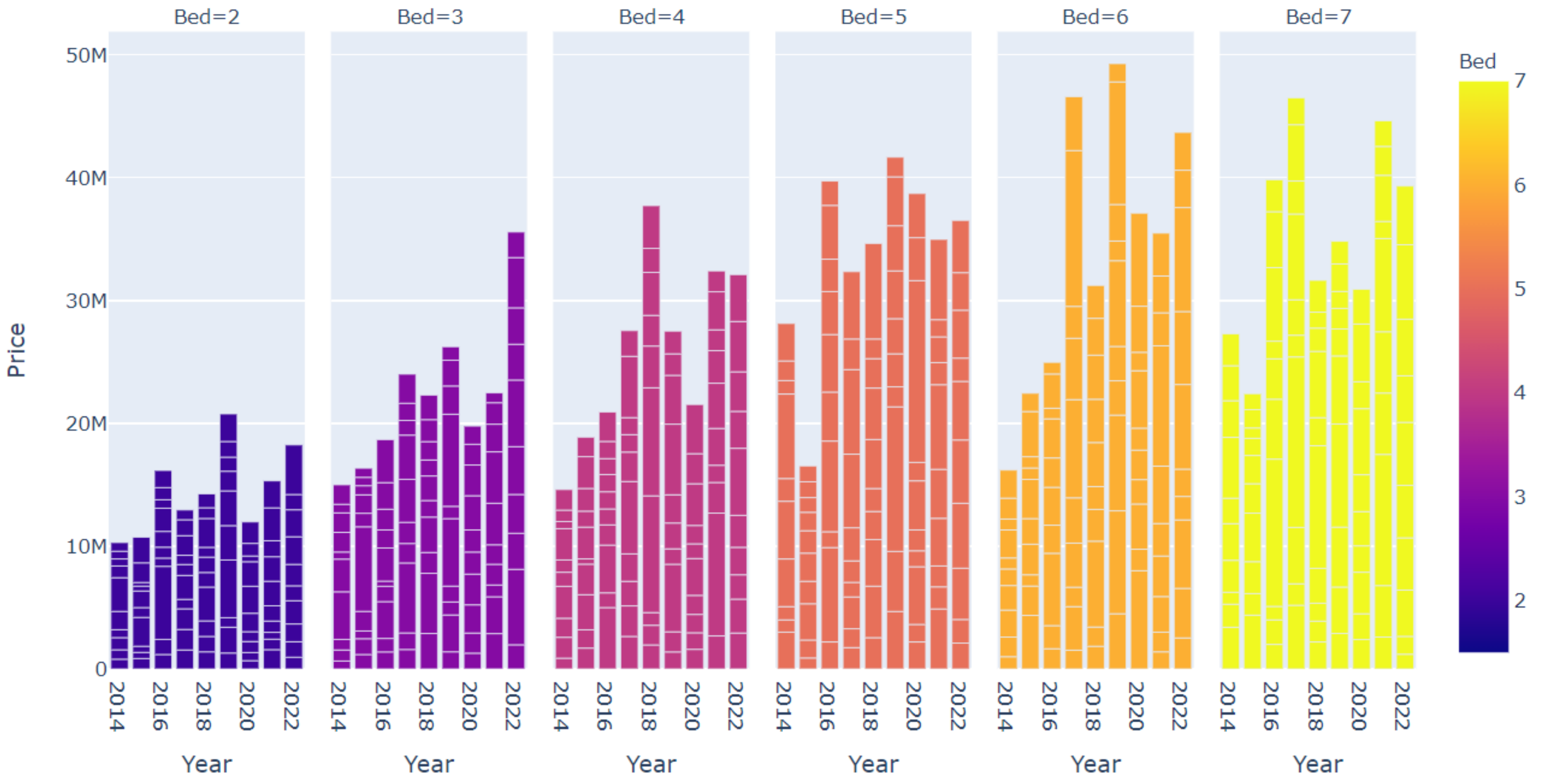
In [82]:

```
# Total price of all 2 bedroom - 7 bedroom houses in a given year categorized on the basis of year
data = pd.DataFrame()
data['Bed'] = hx["Bed"].astype(str)
data['Price'] = hx['price']
data['Year'] = hx["Year"]
fig = px.bar(data, x='Year', y='Price', color='Bed',barmode = "group")
fig.show()
```



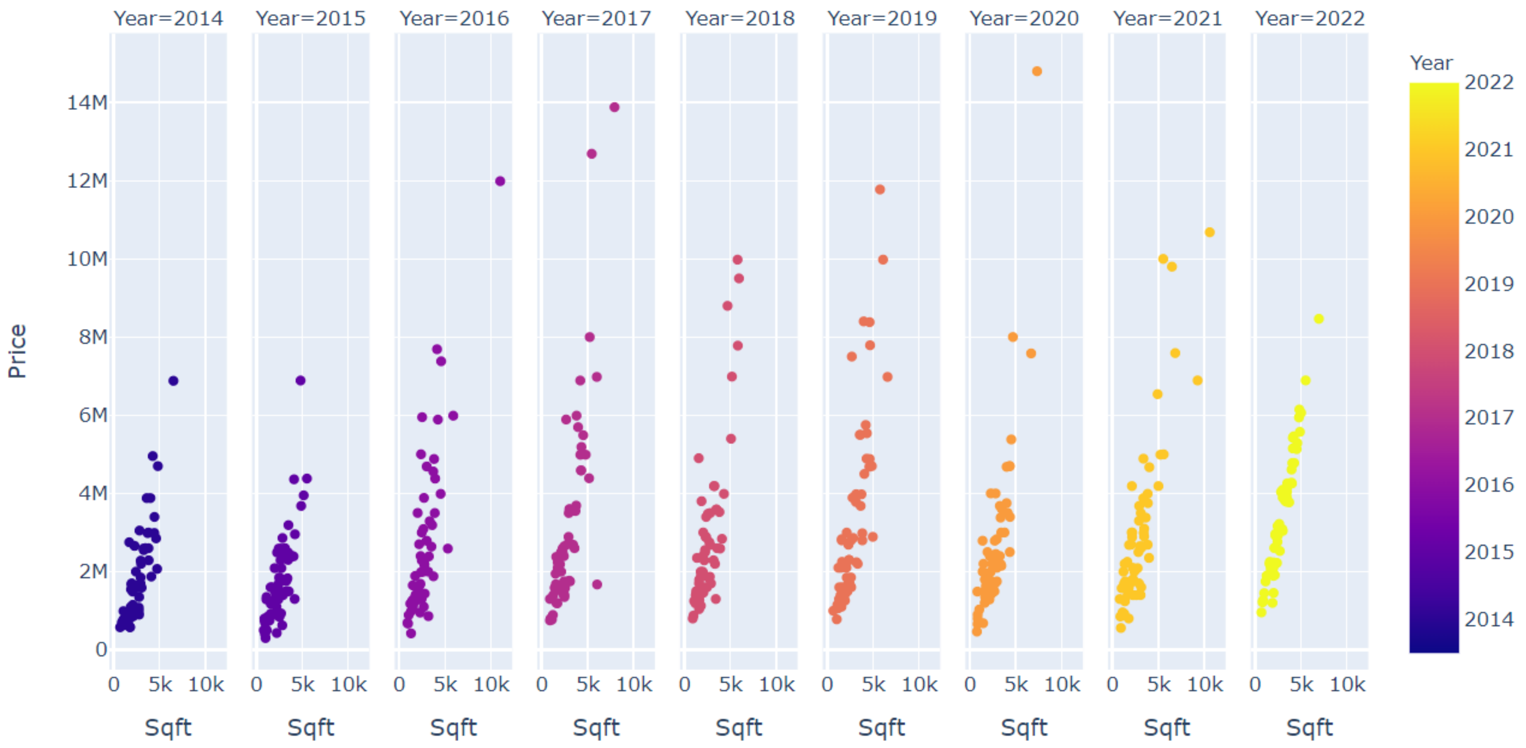
In [83]:

```
# Total price of all 2 bedroom - 7 bedroom houses in a given year categorized on the basis of number of beds
data = pd.DataFrame()
data['Bed'] = hx["Bed"].apply(pd.to_numeric)
data['Price'] = hx["price"].apply(pd.to_numeric)
data['Year'] = hx["Year"].apply(pd.to_numeric)
fig = px.bar(data, x='Year', y='Price', color='Bed', facet_col = 'Bed')
fig.show()
```



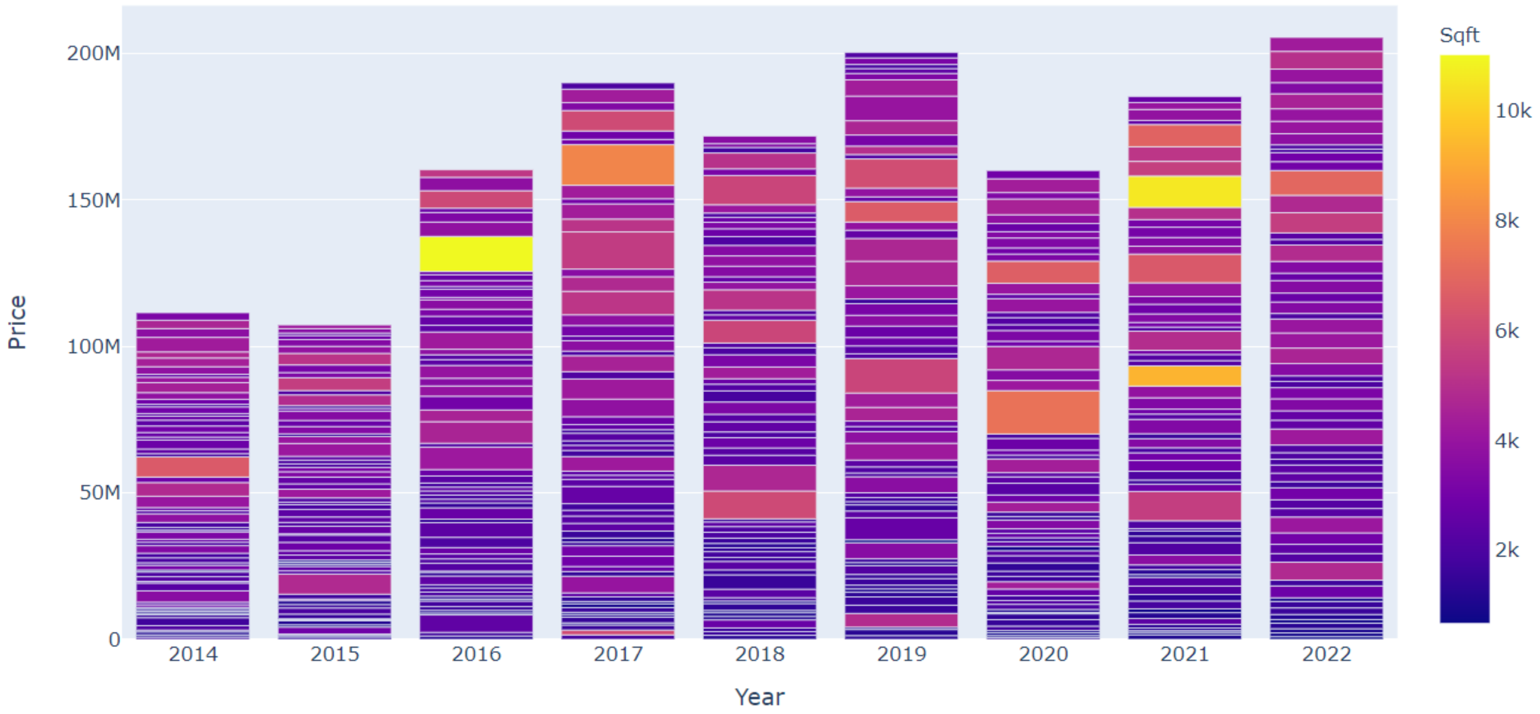
In [84]:

```python
# Price of 2 bedroom - 7 bedroom houses based on sqft
data = pd.DataFrame()
data['Sqft'] = hx["Sqft"]
data['Price'] = hx["price"]
data['Year'] = hx["Year"]
fig = px.scatter(data, x='Sqft', y='Price', color='Year',
                 facet_col='Year', width=950)
fig.show()
```

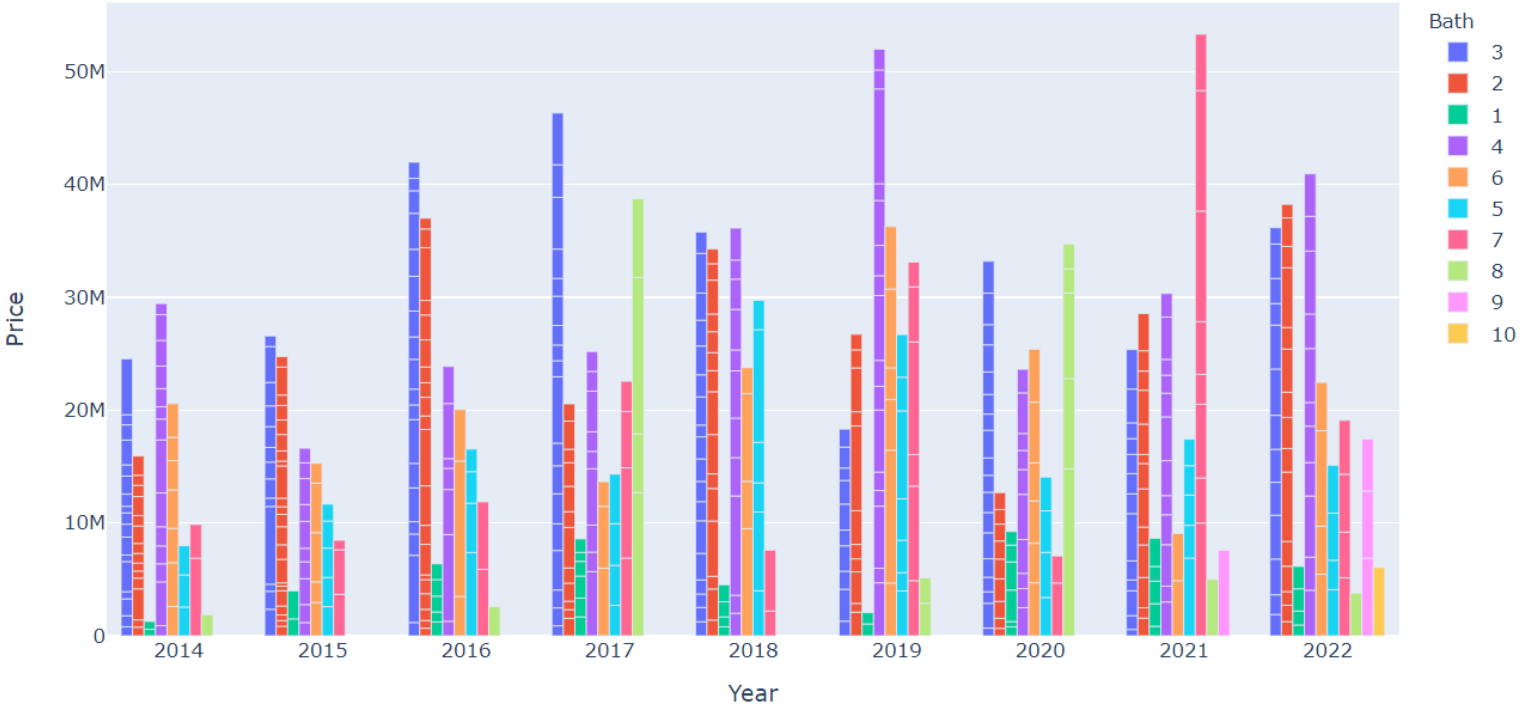


In [85]:

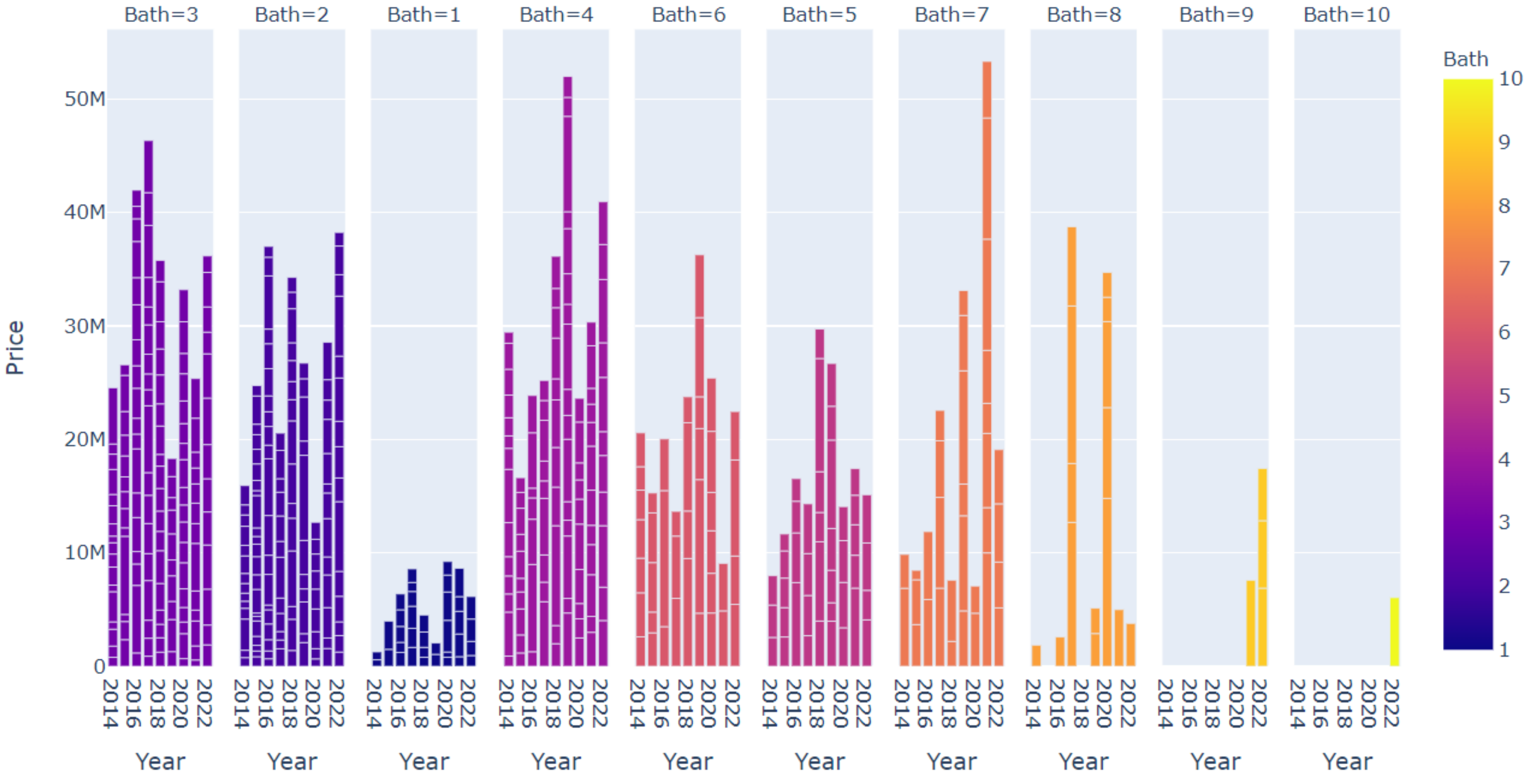
```python
# Price of all the houses based on sqft as stack
data = pd.DataFrame()
data['Sqft'] = hx["Sqft"]
data['Price'] = hx["price"]
data['Year'] = hx["Year"]
fig = px.bar(data, x='Year', y='Price',color='Sqft')
fig.show()
```



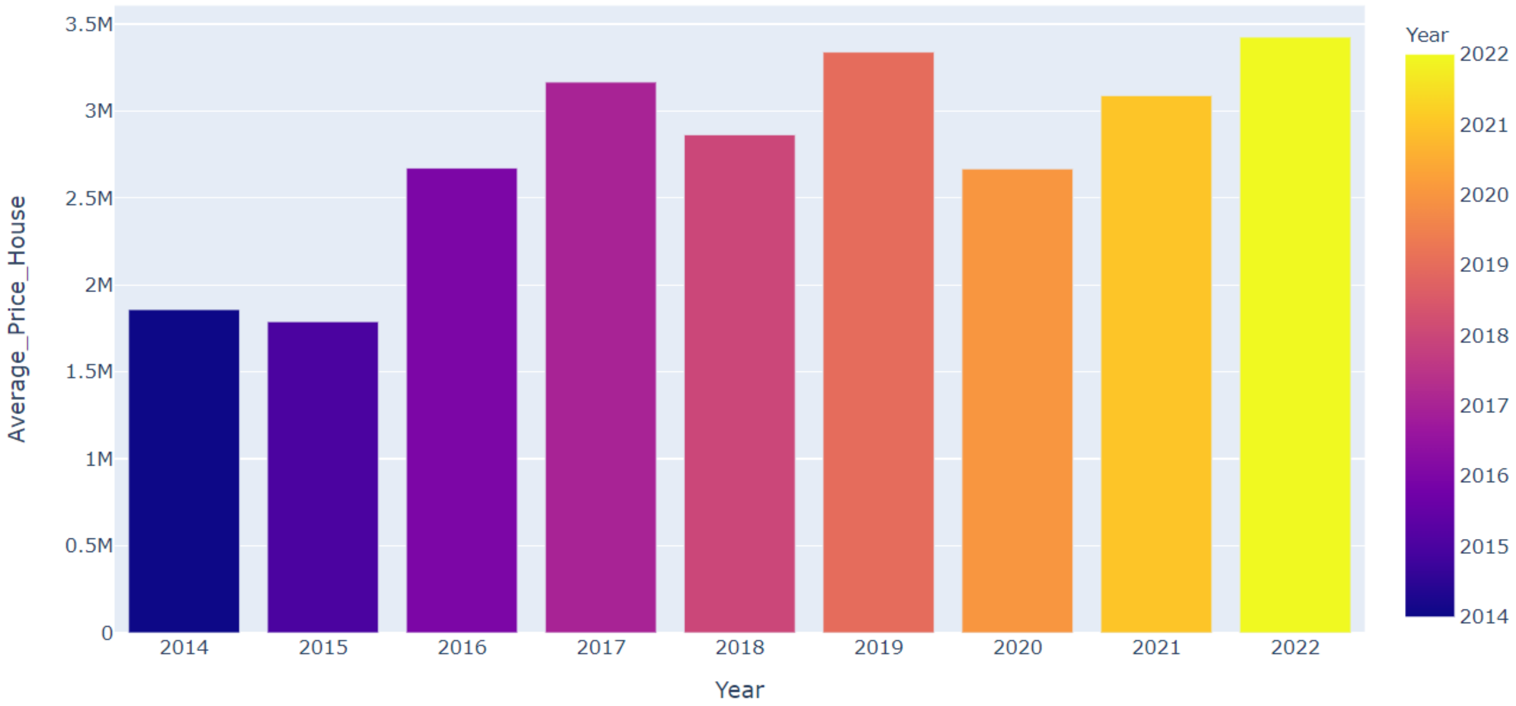
```
In [86]:  # Total price of all houses based on number of baths in a given year categorized on the basis of year
          data = pd.DataFrame()
          data['Bath'] = hx["Bath"].astype(str)
          data['Price'] = hx["price"]
          data['Year'] = hx["Year"]
          fig = px.bar(data, x='Year', y='Price', color='Bath', barmode = 'group')
          fig.show()
```



```
In [87]:  # Total price of all houses based on number of baths in a given year categorized on the basis of number of baths
          data = pd.DataFrame()
          data['Bath'] = hx["Bath"]
          data['Price'] = hx["price"]
          data['Year'] = hx["Year"]
          fig = px.bar(data, x='Year', y='Price', color='Bath', facet_col = "Bath")
          fig.show()
```



```
In [88]:   # Average price of houses(uniform data) for each year
           hsx_avg = pd.read_csv('E:/Vancouver_Houses_Avg_Price - from Uniform data.csv')
           fig = px.bar(hsx_avg, x='Year', y='Average_Price_House', color='Year')
           fig.show()
```



Thank You!