Data visulaization on sales data - M00977896

In [9]:
```
#importing the librabries pandas, altair and datetime
import pandas as pd
import altair as alt
from datetime import datetime

alt.data_transformers.disable_max_rows() # maximum number of rows restriction
#Loding the sales dataset in dataframe using pandas
df_s_eda = pd.read_csv("sale_eda.csv")
```

In [10]:
```
df_s_eda.isnull().sum()
```

Out[10]:
```
title                0
latitude             0
longitude            0
Locality             0
Region               0
Country              0
price                0
type                 0
address              0
beds                 0
baths                0
completion_status    0
furnishing           0
post_date            0
area                 0
agency_name          0
purpose              0
year                 0
month                0
day                  0
quarter              0
building_name        0
price_per_sq_unit    0
price_category       0
dtype: int64
```

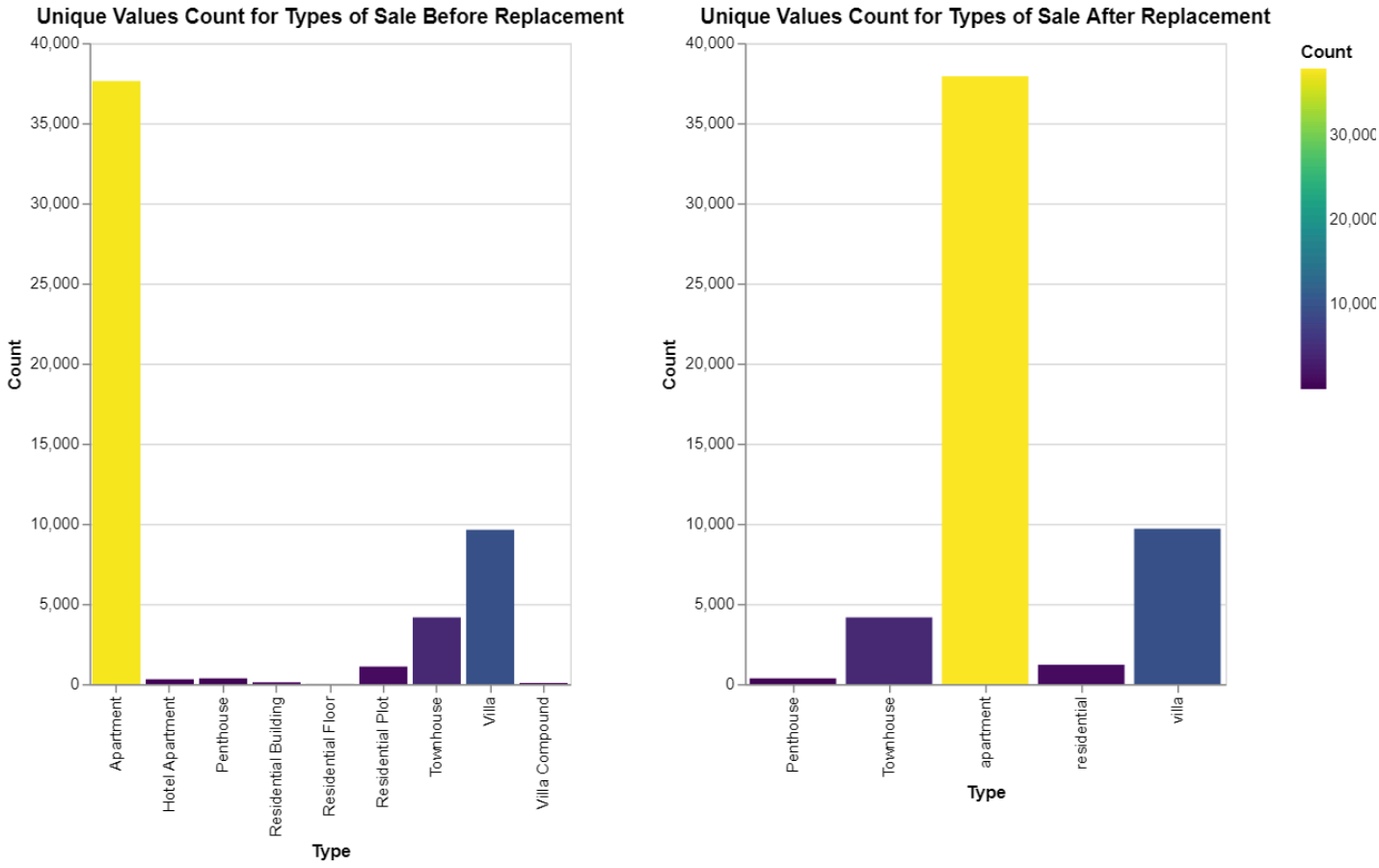
In [11]:
```
#checking data-types
df_s_eda.dtypes
```

Out[11]:
```
title                 object
latitude             float64
longitude            float64
Locality              object
Region                object
Country               object
price                float64
type                  object
address               object
beds                   int64
baths                  int64
completion_status     object
furnishing            object
post_date             object
area                 float64
agency_name           object
purpose               object
year                   int64
month                  int64
day                    int64
quarter                int64
building_name         object
price_per_sq_unit    float64
price_category        object
dtype: object
```

In [12]:

```
# br chart of count of types for rent before replacement of sale data
value_counts_s_before = df_s_eda['type'].value_counts()
value_counts_df_s_before = value_counts_s_before.reset_index()
value_counts_df_s_before.columns = ['type', 'Count']

beds_bar_chart_s_before = alt.Chart(value_counts_df_s_before).mark_bar().encode(
    x=alt.X('type:O', title='Type'),
    y=alt.Y('Count:Q', title='Count'),
    color=alt.Color('Count:Q', scale=alt.Scale(scheme='viridis'), title='Count')
).properties(
    title='Unique Values Count for Types of Sale Before Replacement',
    width=300,
    height=400
)

# grouping the data
replacement_mapping = {
    'Apartment': 'apartment',
    'Hotel Apartment': 'apartment',
    'Penthouse': 'Penthouse',
    'Residential Building': 'residential',
    'Residential Plot': 'residential',
    'Residential Floor': 'residential',
    'Townhouse': 'Townhouse',
    'Villa': 'villa',
    'Villa Compound': 'villa'
}
df_s_eda['type'] = df_s_eda['type'].replace(replacement_mapping)

# bar chart of count of types for sale after replacement
value_counts_s_after = df_s_eda['type'].value_counts()
value_counts_df_s_after = value_counts_s_after.reset_index()
value_counts_df_s_after.columns = ['type', 'Count']

beds_bar_chart_s_after = alt.Chart(value_counts_df_s_after).mark_bar().encode(
    x=alt.X('type:O', title='Type'),
    y=alt.Y('Count:Q', title='Count'),
    color=alt.Color('Count:Q', scale=alt.Scale(scheme='viridis'), title='Count')
).properties(
    title='Unique Values Count for Types of Sale After Replacement',
    width=300,
    height=400
)

#horizontally displaying both charts using hconcat
combined_beds_bar_chart = alt.hconcat(beds_bar_chart_s_before, beds_bar_chart_s_after)
combined_beds_bar_chart
```

Out[12]:



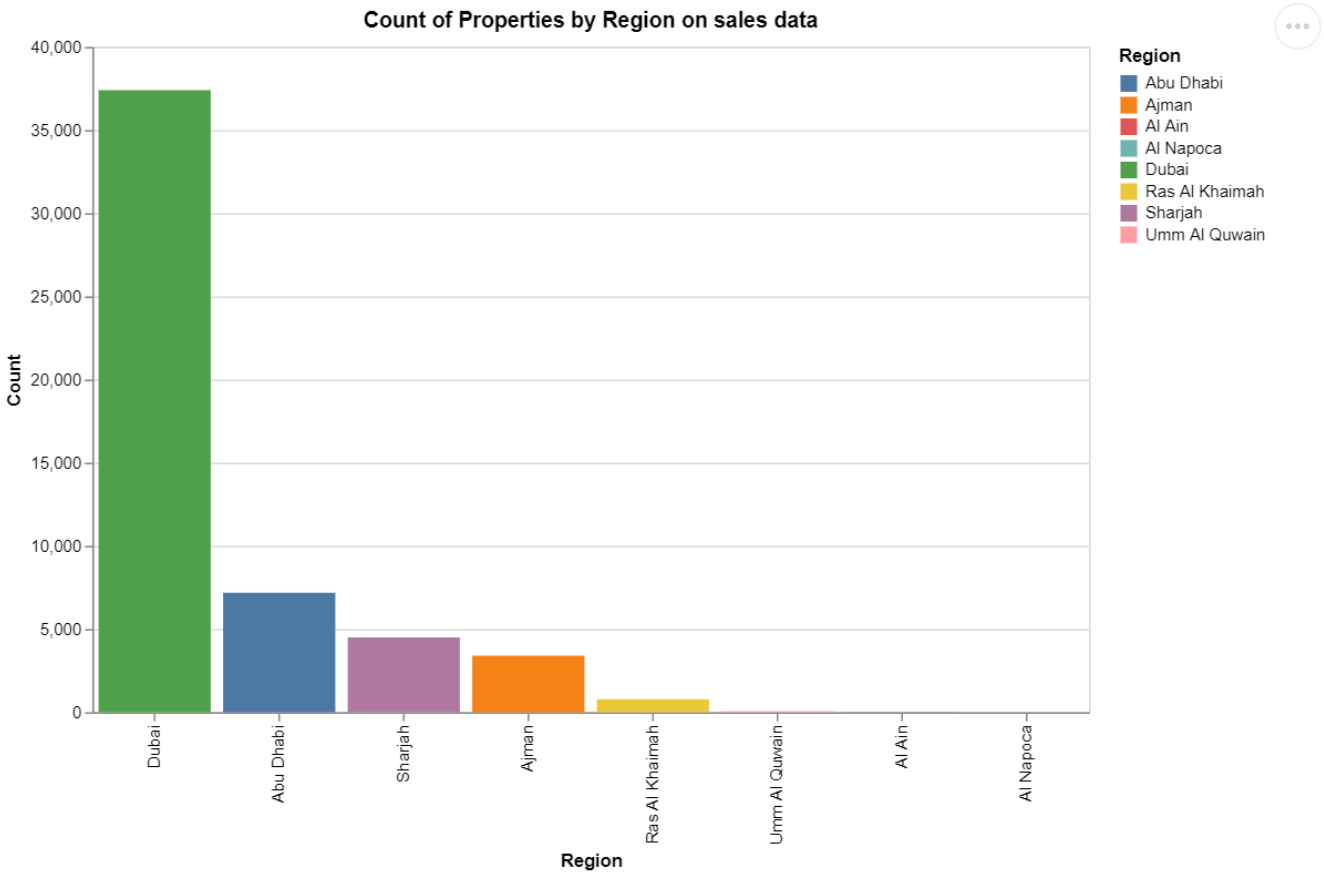
The x-axis encodes the "Type" of property (e.g., Apartment, Villa), and the y-axis encodes the "Count" of each type. The use of color encoding represents the count's intensity, helping to visually distinguish the frequency of each category. Created two bar charts: The image contains two side-by-side bar charts, each representing the count of different property types before and after grouping into 5 categories by using hconcat for horizontal visual. Encode data: The x-axis of each chart is encoded with the Type variable, representing the property types. The y-axis is encoded with the Count variable, representing the count of each property type. The distribution of property types in the dataset was significantly impacted by the replacement procedure, as the charts show. It caused a shift in the relative frequency of certain categories and a consolidation of kinds. The precise modifications will probably vary depending on the replacement criteria and the kinds of properties that are involved.

Checking Categorical values

In [13]:

```
#create a bar chart to visualize the count of properties by region of sales data
region_count_chart = alt.Chart(df_s_eda).mark_bar().encode(
    x=alt.X('Region:N', title='Region', sort='-y'),
    y=alt.Y('count()', title='Count'),
    tooltip=['Region', 'count()'],
    color='Region:N'
).properties(
    width=600,
    height=400,
    title='Count of Properties by Region on sales data'
)

region_count_chart
```

Out[13]:

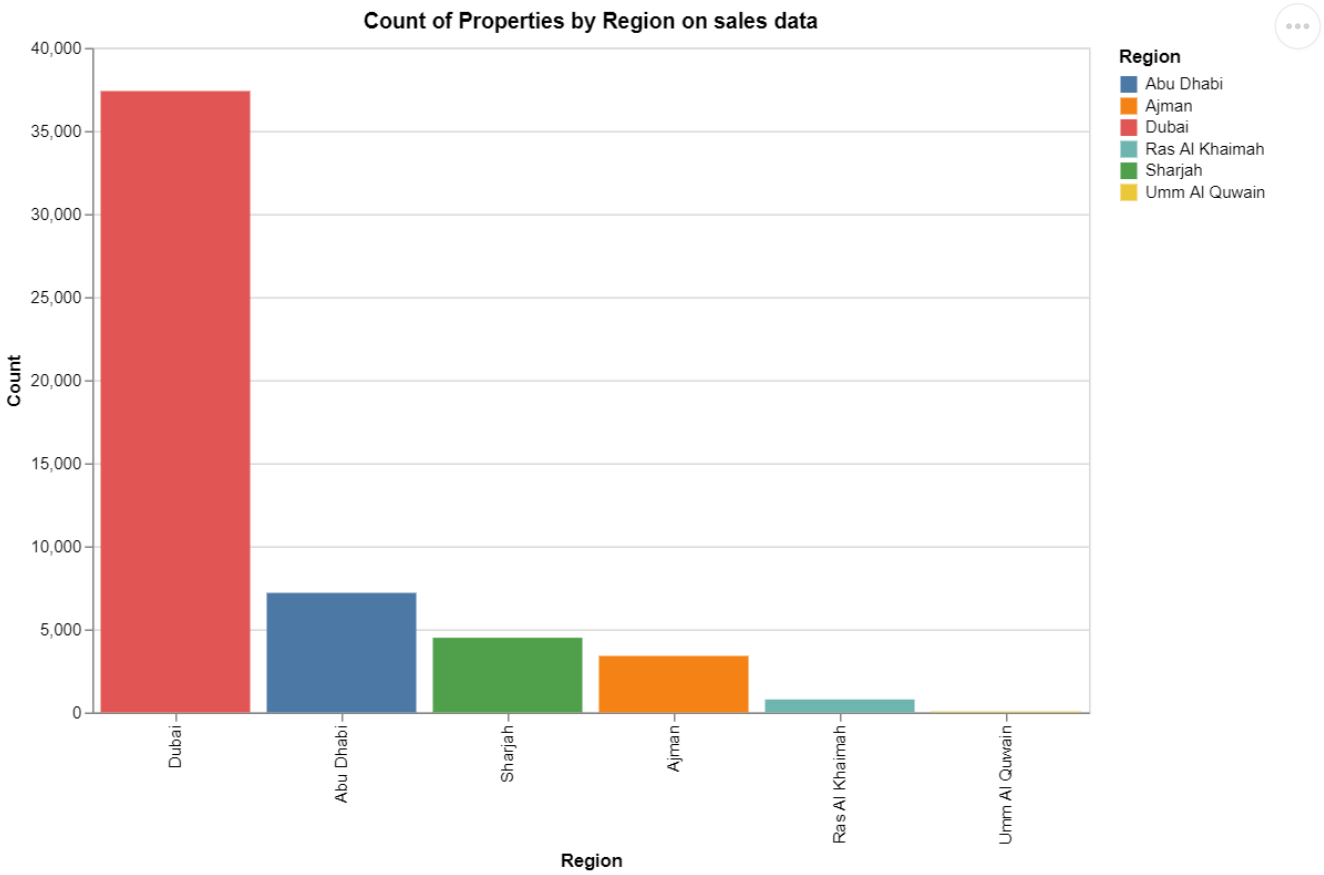


The seven emirates are: Abu Dhabi, Dubai, Sharjah, Ajman, Umm Al Quwain, Ras Al Khaimah and Fujairah.City "AL NAPOCA" comes under Ajman emirates and Al Ain comes under "Abu dhabi". There no data for Fujairah emirate

In [14]:

```
# So replace al napoca with ajman and alain to abu dhabi
# initially ajman consit of "3373" records after,  below code the count came to "3374".
#intially abudhabi consist of "7183" records after , below code gave the value as "7195"
df_s_eda['Region'] = df_s_eda['Region'].replace('Al Napoca', 'Ajman')
df_s_eda['Region'] = df_s_eda['Region'].replace('Al Ain', 'Abu Dhabi')

# Create a bar chart to visualize the count of properties by region
region_count_chart = alt.Chart(df_s_eda).mark_bar().encode(
    x=alt.X('Region:N', title='Region', sort='-y'),
    y=alt.Y('count()', title='Count'),
    tooltip=['Region', 'count()'],
    color='Region:N'
).properties(
    width=600,
    height=400,
    title='Count of Properties by Region on sales data'
)

region_count_chart
```

Out[14]:



Among the different emirates, it is clear that Dubai has the most properties on sales data.As for the number of properties, Dubai ranks first before Abu Dhabi. Sharjah and Ajman on the other hand have a fair share of properties. Properties in Ras Al Khaimah, Umm Al Quwain and Al Ain are two quite considerably few in number.

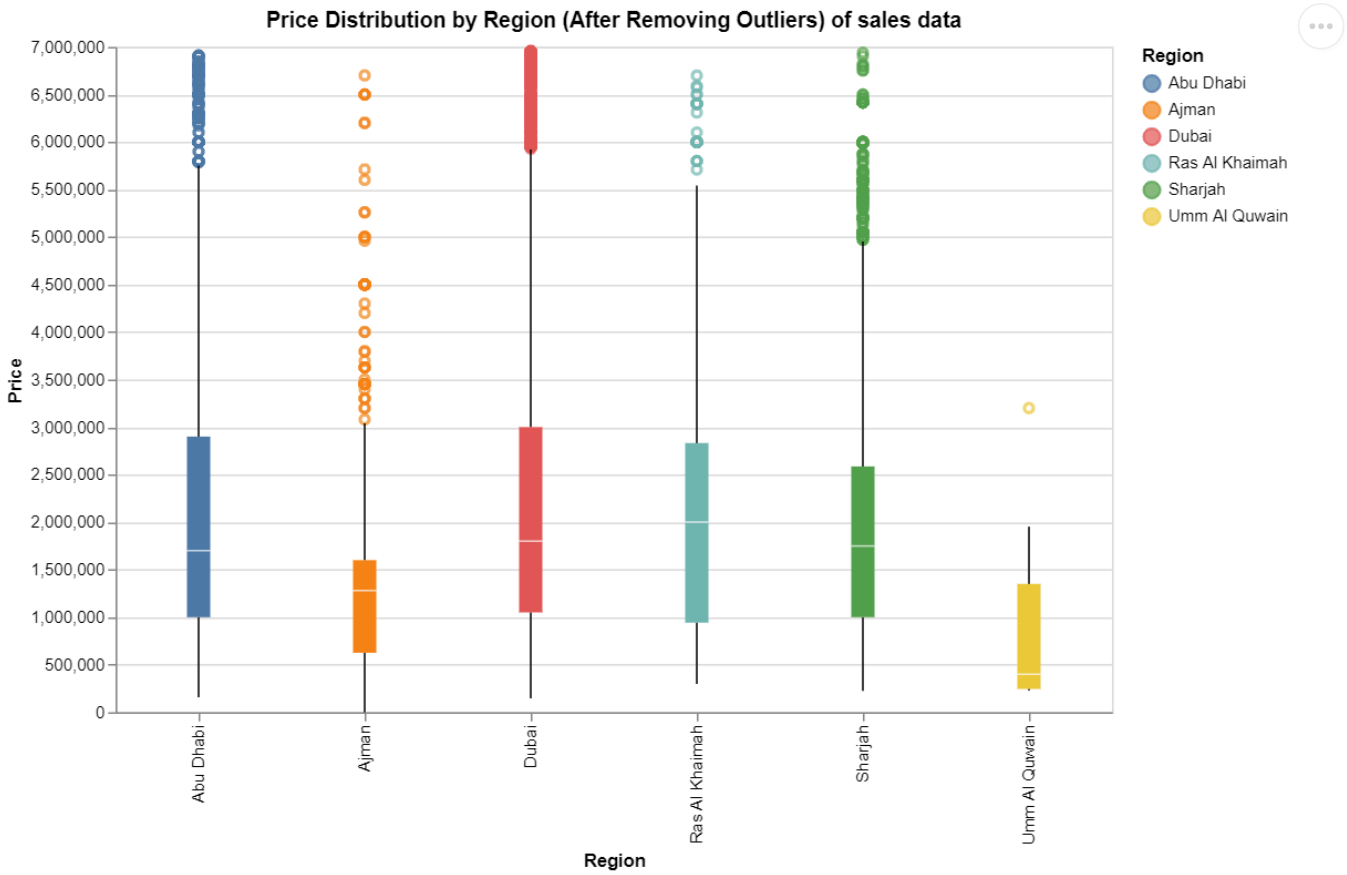
Analysis on outliers of sales data

In [15]:

```
boxplot_before = alt.Chart(df_s_eda).mark_boxplot().encode(
    y=alt.Y('price:Q', title='Price'),
    x=alt.X('Region:N', title='Region'),
    color='Region:N'
).properties(
    title='Price Distribution by Region (Before Removing Outliers) of sales data',
    width=600,
    height=400
)

boxplot_before.show()
```

The plot indicates that no outliers were removed. Some of the data points consist of extreme values which could be described as outliers in Dubai and Abu Dhabi.The particular case of Dubai is especially outstanding, with extreme price values over 200 million and a few near 100 million and one in 400 million. Ajman and Al Ain are far thinner in price spectrum compared to Abu Dhabi and Dubai. Umm Al Quwain shows very few data points, which could be an indication of fewer sales or fewer outliers for that region. For sales data properties are near to 400 million.

In [16]:
```
# using IQR for 'price' column
Q1 = df_s_eda['price'].quantile(0.25)
Q3 = df_s_eda['price'].quantile(0.75)
IQR = Q3 - Q1

#define lower and upper bounds for sales data
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

#filter out outliers
df_s_eda = df_s_eda[(df_s_eda['price'] >= lower_bound) & (df_s_eda['price'] <= upper_bound)]

#create a boxplot to visualize outliers after removal
boxplot_after = alt.Chart(df_s_eda).mark_boxplot().encode(
    y=alt.Y('price:Q', title='Price'),
    x=alt.X('Region:N', title='Region'),
    color='Region:N'
).properties(
    title='Price Distribution by Region (After Removing Outliers) of sales data',
    width=600,
    height=400
)
boxplot_after.show()
```



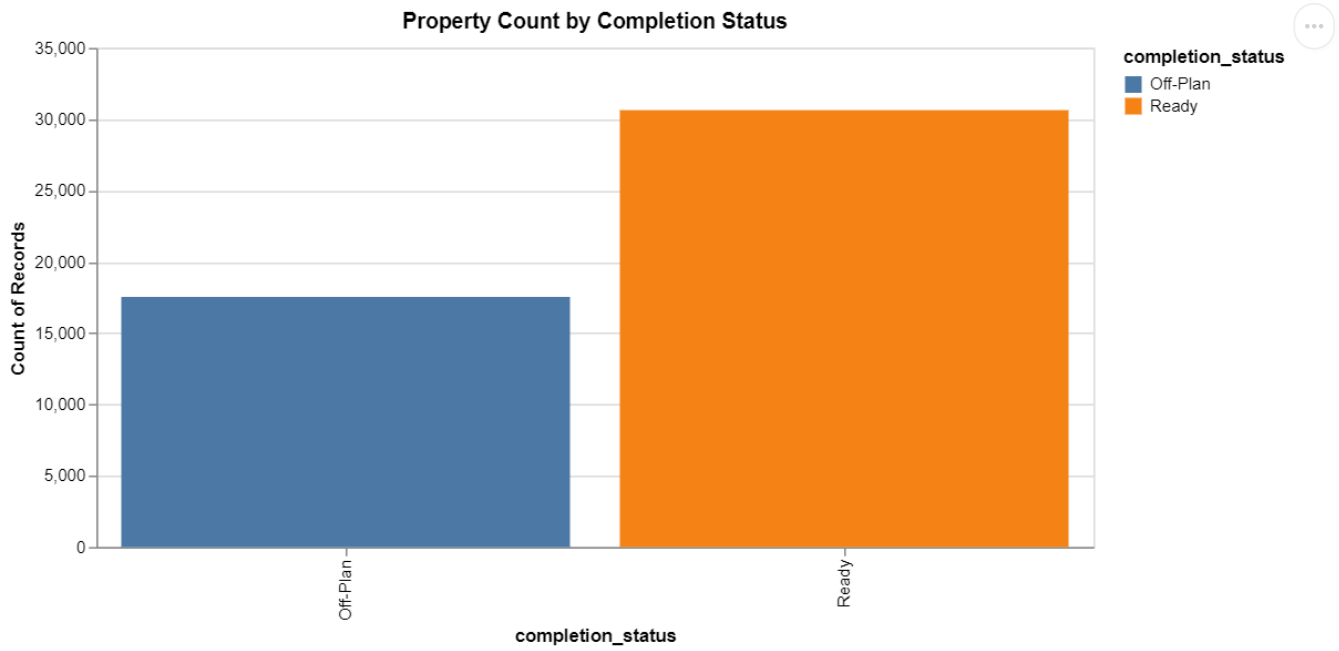
Price spreads are larger in Abu Dhabi, Dubai, Ras Al Khaimah, and Sharjah compared to Ajman, Al Ain, and Umm Al Quwain. In the box plots, boxes show the middle 50% of each within a region from the first quartile to the third quartile, and whiskers indicate data dispersion up to 1.5 times the IQR from the quartiles. Ajman has a noticeable number of outliers above the upper whisker, which may suggest that there is a higher concentration of higher-priced properties outside the general spread. Abu Dhabi and Dubai still show wider ranges of prices within the IQR, but now the extreme outliers from the earlier graph have been removed.

Checking Categories

In [17]:
```
# Property Count by Completion Status in sales
completion_status = alt.binding_radio(options=df_s_eda['completion_status'].unique(), name='Completion Status:')
completion_select_sale = alt.selection_single(fields=['completion_status'], bind=completion_status)

#Counting of Properties by Completion Status and creating bar chart
bar_chart = alt.Chart(df_s_eda).mark_bar().encode(
    x='completion_status:N',
    y='count()',
    color='completion_status:N',
    tooltip=['count()']
).add_selection(
    completion_select_sale
).transform_filter(
    completion_select_sale
).properties(
    title='Property Count by Completion Status',
    width=600,
    height=300,

)
bar_chart
```

```
<ipython-input-17-6246e72bd300>:3: AltairDeprecationWarning: Deprecated in `altair=5.0.0`. Use selection_point instead.
  completion_select_sale = alt.selection_single(fields=['completion_status'], bind=completion_status)
<ipython-input-17-6246e72bd300>:6: AltairDeprecationWarning: Deprecated in `altair=5.0.0`. Use add_params instead.
  bar_chart = alt.Chart(df_s_eda).mark_bar().encode(
```

Out[17]:

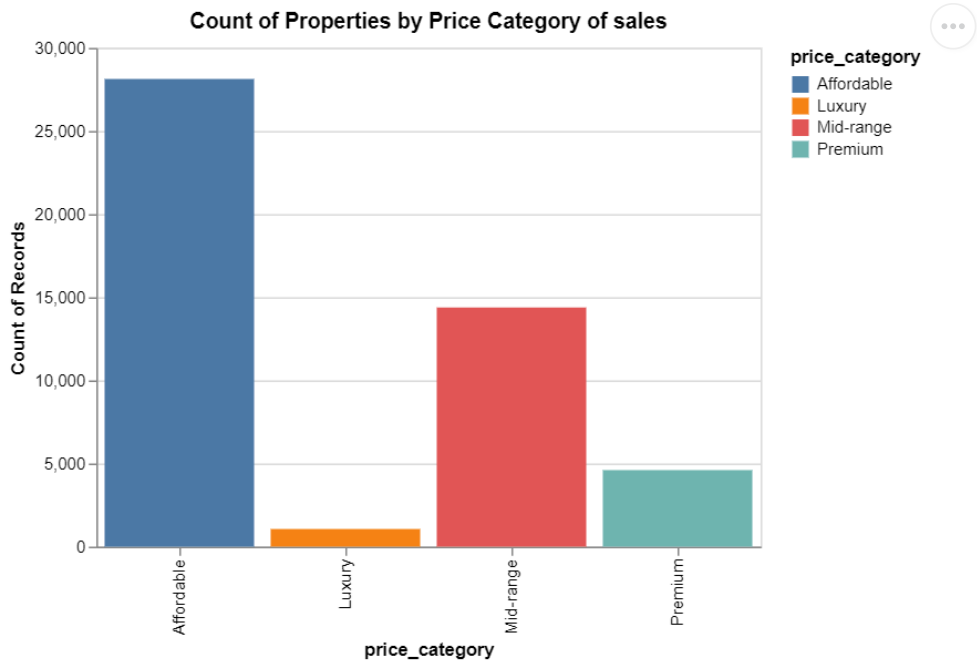


Completion Status: ○ Off-Plan ○ Ready

Off-Plan Properties: Approximately 18,000 properties are in the "Off-Plan" status. This could represent a significant portion of the overall property market.Ready Properties: The count of "Ready" properties is approximately 31,000. This suggests that there is a larger supply of completed properties available.

In [18]:
```
# Bar chart with count of price category

price_category_count = alt.Chart(df_s_eda).mark_bar().encode(
    x='price_category:N',
    y='count():Q',
    color='price_category:N'
).properties(
    width=400,
    height=300,
    title='Count of Properties by Price Category of sales'
)

price_category_count
```

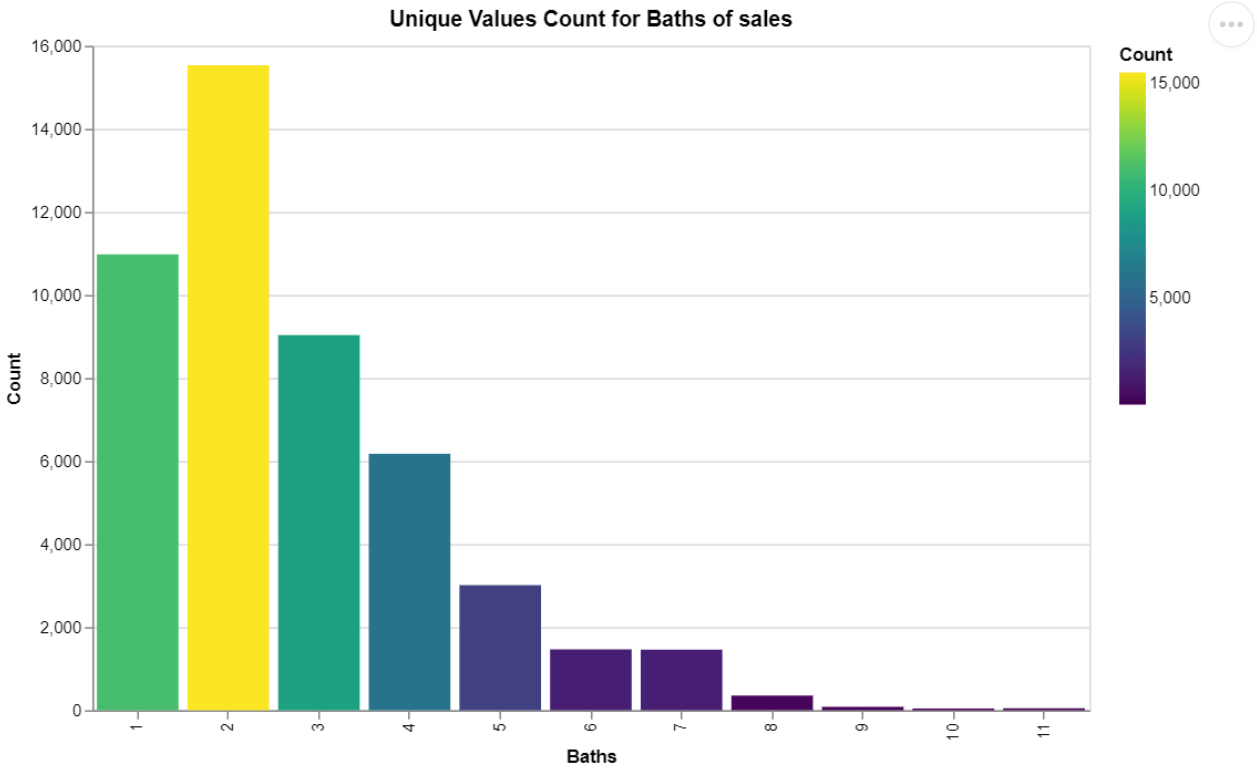
Out[18]:



Affordable Properties: With about 28,000 listings, affordable properties lead the pack, showing a strong demand for budget-friendly options. Luxury Properties: In contrast, luxury properties are much rarer, with only around 2,000 available, reflecting a niche market. Mid-Range Properties: Mid-range options sit in the middle with about 15,000 listings, appealing to buyers looking for a balance between affordability and luxury. Premium Properties: Premium properties are the least common, totaling around 5,000, catering to a select group of buyers seeking high-end features.

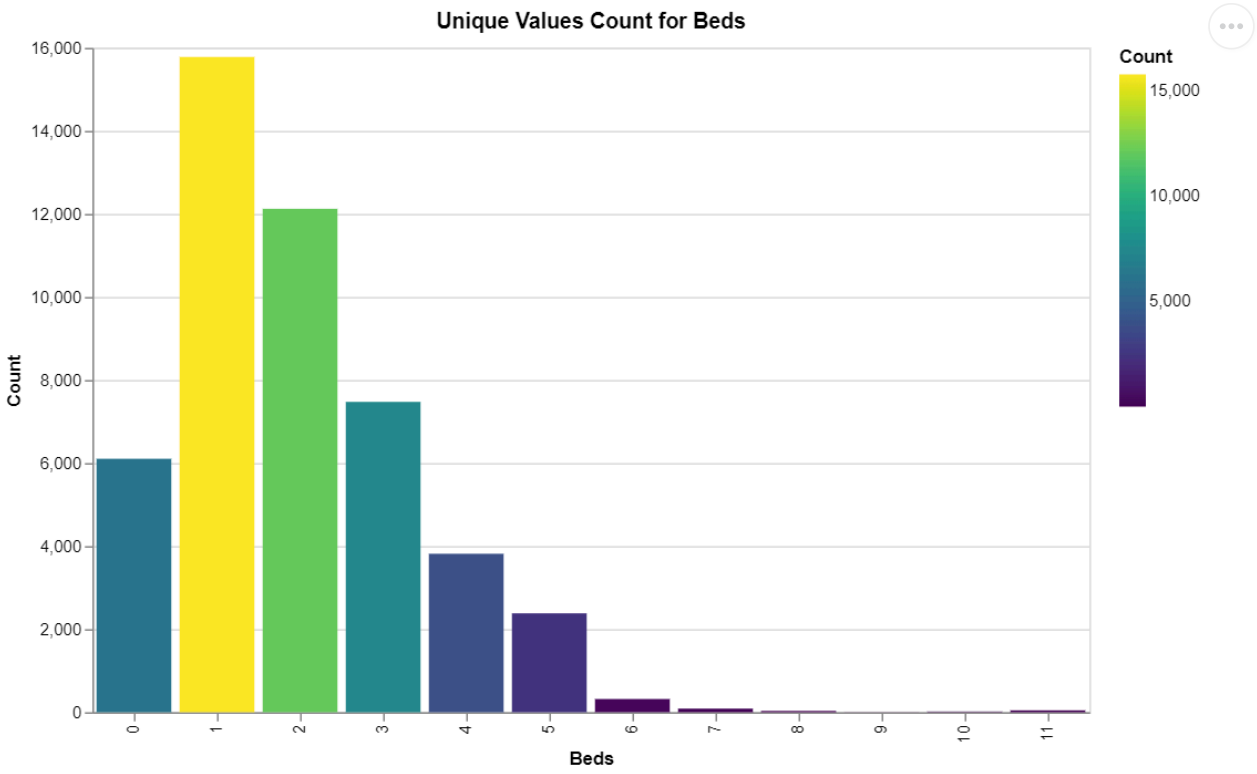
In [19]:
```
#bar chart of count of baths in sales
value_counts = df_s_eda['baths'].value_counts()
```

```
value_counts_df = value_counts.reset_index()
value_counts_df.columns = ['baths', 'Count']

beds_bar_chart = alt.Chart(value_counts_df).mark_bar().encode(
    x=alt.X('baths:O', title= 'Baths'),
    y=alt.Y('Count:Q', title='Count'),
    color=alt.Color('Count:Q', scale=alt.Scale(scheme='viridis'), title='Count')
).properties(
    title='Unique Values Count for Baths of sales',
    width=600,
    height=400
)
beds_bar_chart.display()
```



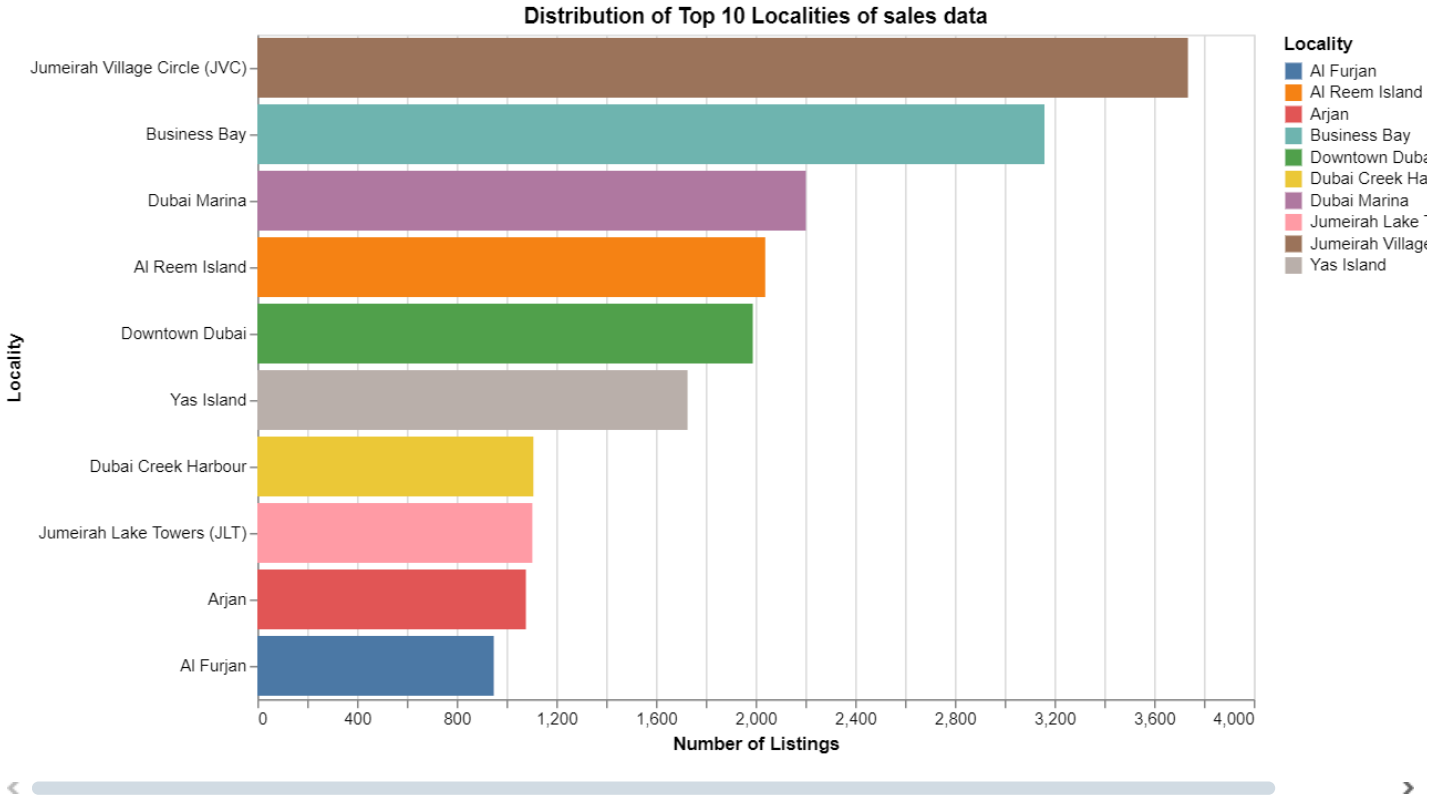
From the above graph, count of baths is 16k where 2- bathrooms are the highest with 15000+ baths and from 6 bathroom to 11 bathroom the count decrases.

In [20]:
```
# Bar chart of count of beds in sales

value_counts = df_s_eda['beds'].value_counts()
value_counts_df = value_counts.reset_index()
value_counts_df.columns = ['beds', 'Count']

beds_bar_chart = alt.Chart(value_counts_df).mark_bar().encode(
    x=alt.X('beds:O', title='Beds'),
    y=alt.Y('Count:Q', title='Count'),
    color=alt.Color('Count:Q', scale=alt.Scale(scheme='viridis'), title='Count')
).properties(
    title='Unique Values Count for Beds',
    width=600,
    height=400
)
beds_bar_chart.display()
```



In [21]:
```python
# Calculate the top 10 count of listings per locality
locality_distribution = df_s_eda['Locality'].value_counts().reset_index()
locality_distribution.columns = ['Locality', 'count']

# Limit to top 10 localities
top_localities = locality_distribution.head(10)

# Horizontal bar chart for distribution of top localities
bar_chart = alt.Chart(top_localities).mark_bar().encode(
    y=alt.Y('Locality:N', title='Locality', sort='-x'),
    x=alt.X('count:Q', title='Number of Listings'),
    color=alt.Color('Locality:N', title='Locality'),
    tooltip=['Locality', 'count']
).properties(
    width=600,
    height=400,
    title='Distribution of Top 10 Localities of sales data'
)

bar_chart
```

Out[21]:

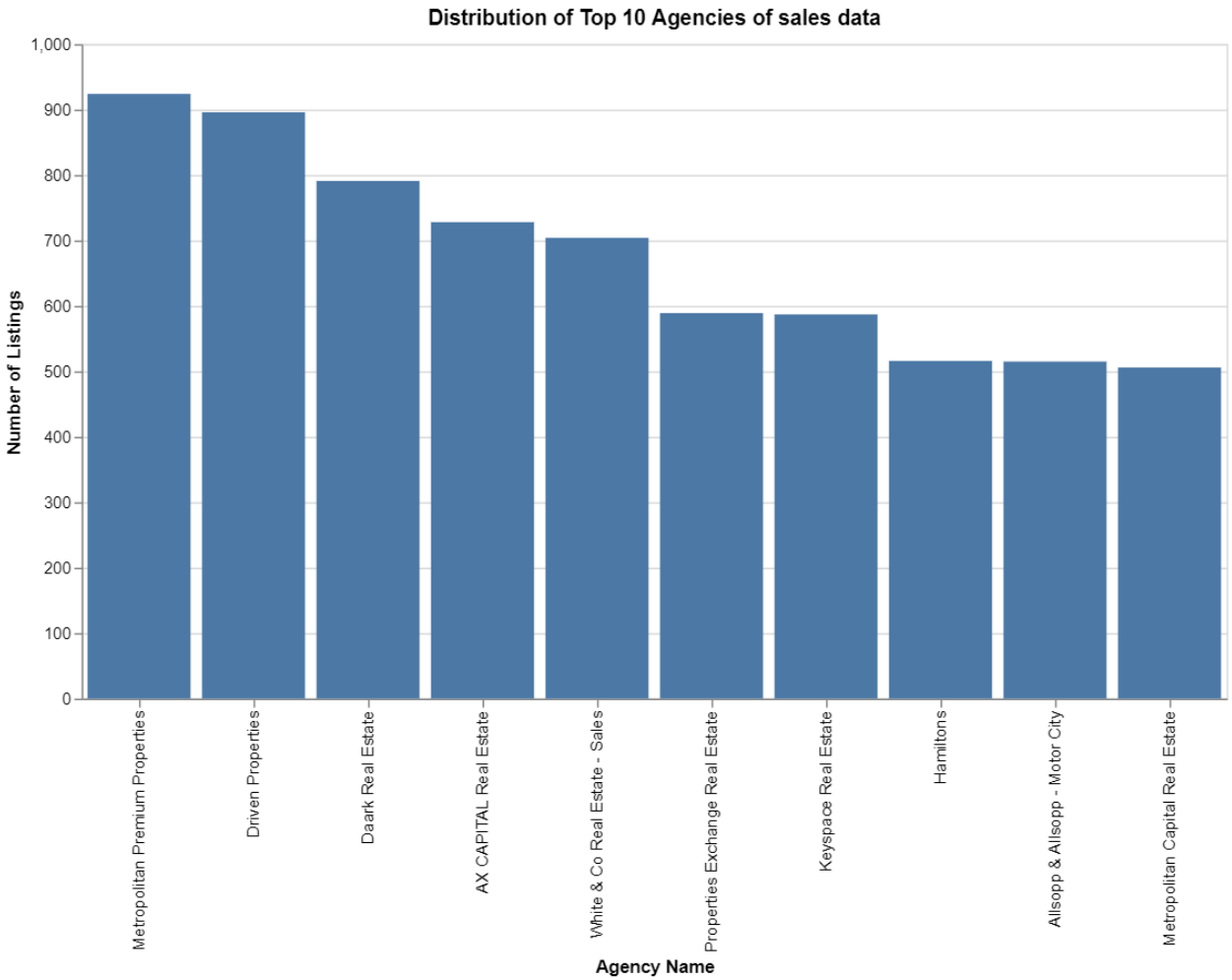


Jumeirah Village Circle has the highest listings at 3,600 sales, hence being the most active locality for property sales. Business Bay comes in second with almost 2,800 sales, reflecting that it is another major hub for property listings. Dubai Marina slots in third with around 2,000 listings, stating its popularity within the market. Between 1,500 and 2,000 listings are in each of Al Reem Island, Downtown Dubai and Yas Island. These are also very major centers with regard to UAE property market, if a look is being taken at the regions of Dubai and Abu Dhabi. Downtown Dubai is considered as a high-end real estate hub with about 1,800 listings. JVC, Business Bay, and Dubai Marina lead in the front line when it comes to sales listings of property. Mid-tier locations include Downtown Dubai, Al Reem Island, and Yas Island, which equally show high activity. Smaller communities include Al Furjan and Arjan, which still comprise a large component of the overall listings.

In [22]:

```
# Calculate the top 10 count of listings per agency
agency_distribution = df_s_eda['agency_name'].value_counts().reset_index()
agency_distribution.columns = ['Agency Name', 'count']

# Limit to top 10 agencies
top_agencies = agency_distribution.head(10)

# Bar chart for distribution of top agencies
bar_chart_agencies = alt.Chart(top_agencies).mark_bar().encode(
    x=alt.X('Agency Name:N', title='Agency Name', sort='-y'),
    y=alt.Y('count:Q', title='Number of Listings'),
    tooltip=['Agency Name', 'count']
).properties(
    width=700,
    height=400,
    title='Distribution of Top 10 Agencies of sales data'
).interactive()

bar_chart_agencies
```

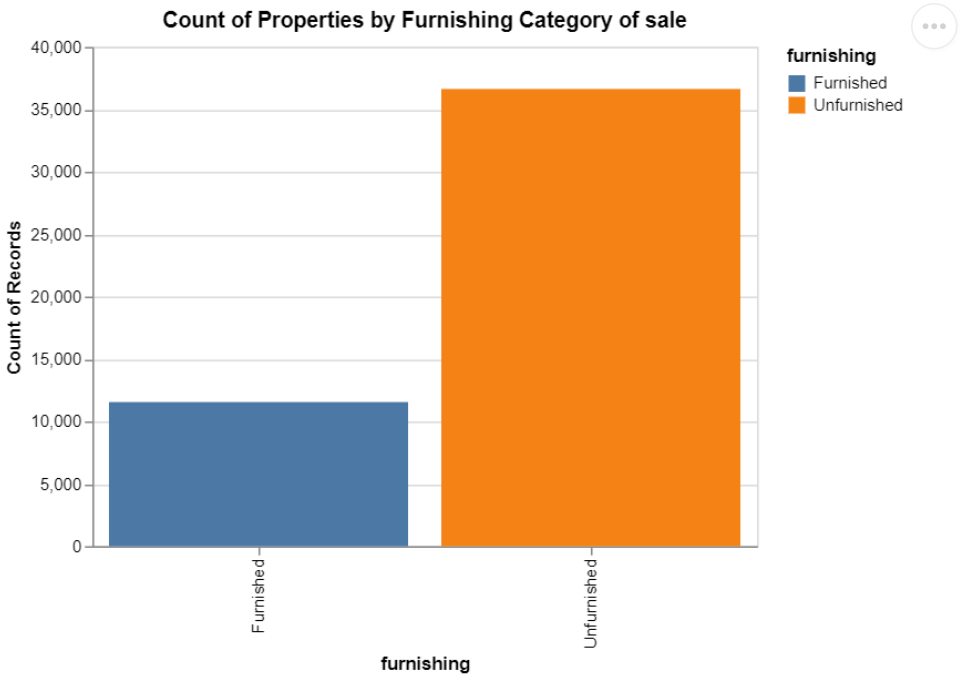
Out[22]:



Leading Agencies: form the graph with over 900 listings each, Driven Properties and Metropolitan Premium Properties are at the top of the list. The two agencies in question have a high market presence as they tower over all other agencies in this dataset. Mid-Level Organizations: Daria Real Estate, AX CAPITAL Real Estate, White & Co Real Estate - Sales, and Properties Exchange Real Estate are in the next level. Each of these agencies has between 650 and 750 listings. Lower Tier Agencies: 600+ listings each, Keyspace Real Estate, Hamiltons, Allsopp & Allsopp - Motor City, and Metropolitan Capital Real Estate narrow the difference. Naturally, they place in the top 10, but they are still far behind.

In [23]:
```
#the below bar chart with count of furnishing category
furnishing_count = alt.Chart(df_s_eda).mark_bar().encode(
    x='furnishing:N',
    y='count():Q',
    color='furnishing:N'
).properties(
    width=400,
    height=300,
    title='Count of Properties by Furnishing Category of sale'
)

furnishing_count
```

Out[23]:



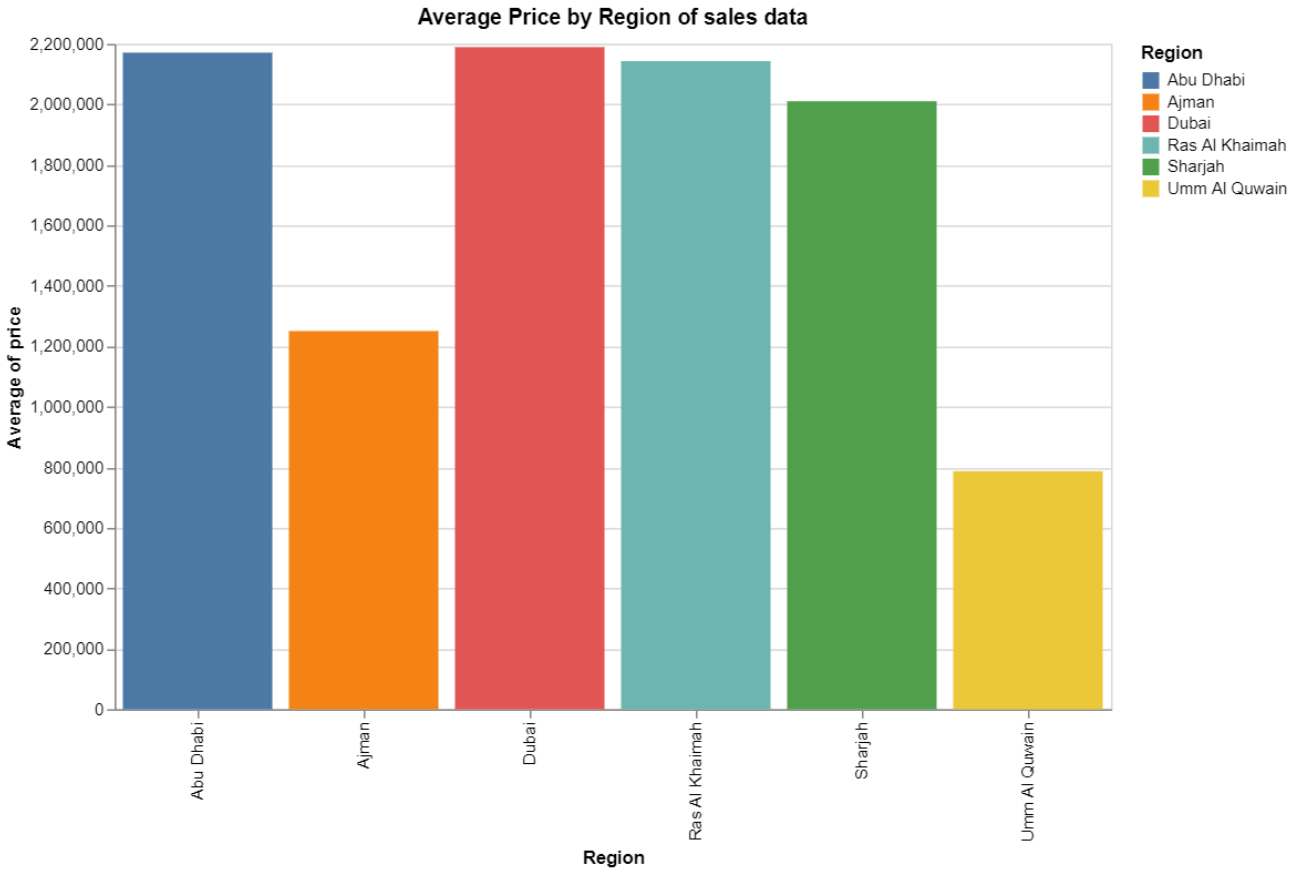
More furnished properties are available than the number of unfurnished properties on offer; this is explained by the fact that the orange bar is taller. This means a large number of properties for sale are furnished, while few properties are not furnished. Large Quantity Difference: If some concrete numerical values were given on the y-axis Count of Records, it would give a better understanding of the quantity difference between the furnished and the unfurnished properties.

Data Analysis on price

In [24]:

```
#calculating Avg price by regions
type_options = df_s_eda['type'].unique().tolist()
type_widget = alt.binding_radio(options=type_options, name='Property Type:')
type_selector = alt.selection_point(fields=['type'], bind=type_widget)

bar_chart_region = alt.Chart(df_s_eda).mark_bar().encode(
    x='Region:N',
    y='average(price):Q',
    color='Region:N',
    tooltip=['Region', 'average(price):Q']
).properties(
    width=600,
    height=400,
    title='Average Price by Region of sales data'
).add_params(
    type_selector
).transform_filter(
    type_selector
)

bar_chart_region
```

Out[24]:



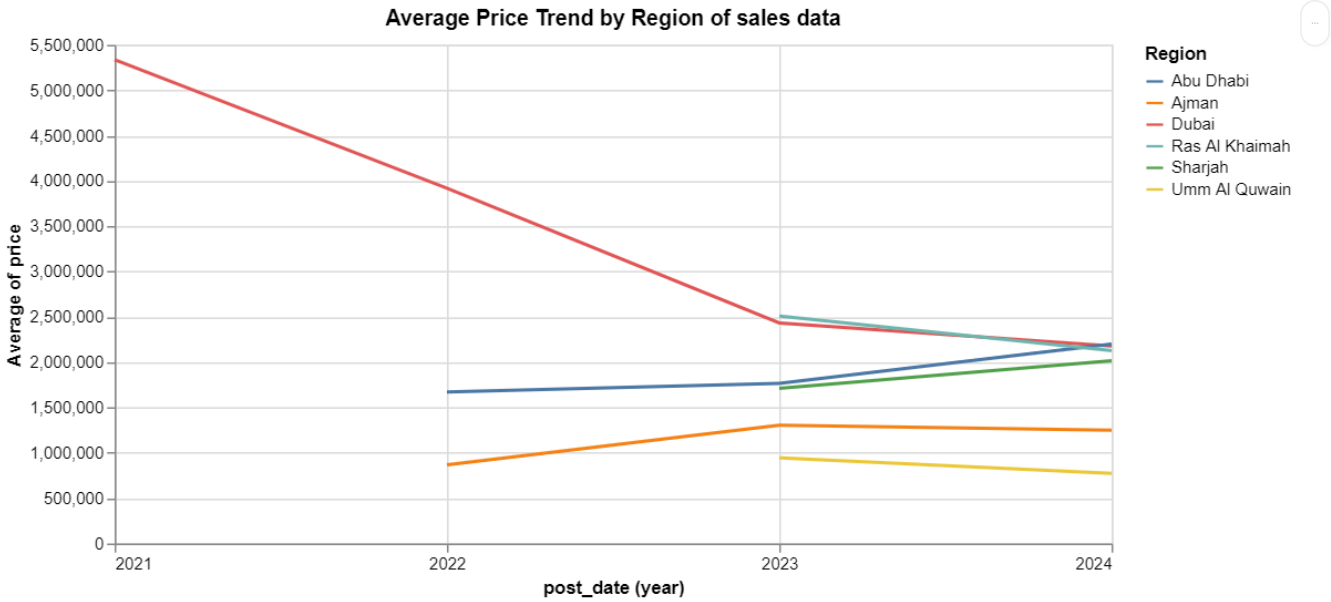
Property Type: ○ apartment ○ Townhouse ○ villa ○ residential ○ Penthouse

The average price of properties varies significantly across regions, indicating that location is a major factor influencing property values. Create a bar chart: The chart type is a bar chart, suitable for comparing categorical data (regions) with a quantitative value (average price). Encode data: The x-axis is encoded with the Region variable, and the y-axis is encoded with the Average Price variable. Add color: The bars may be colored based on the Region variable to differentiate between regions. Used widgt for property type as a radio button. Al ain have the highest average price, while Lowest is umm al quwain. Other can be seen on the same line with some fluctuation.

In [25]:

```
#create location options and labels for the radio buttons of emirates
location_options = df_s_eda['Region'].unique().tolist()
widget = alt.binding_radio(options=location_options + [None], name="Region")
selector = alt.selection_point(fields=['Region'], bind=widget)

#created a line chart showing price trends over time for selected Region
price_trend_chart = alt.Chart(df_s_eda).mark_line().encode(
    x='year(post_date):T',
    y='average(price):Q',
    color='Region:N'
).properties(
    width=600,
    height=300,
    title='Average Price Trend by Region of sales data'
).add_params(
    selector
).transform_filter(
    selector
)

price_trend_chart
```

Out[25]:

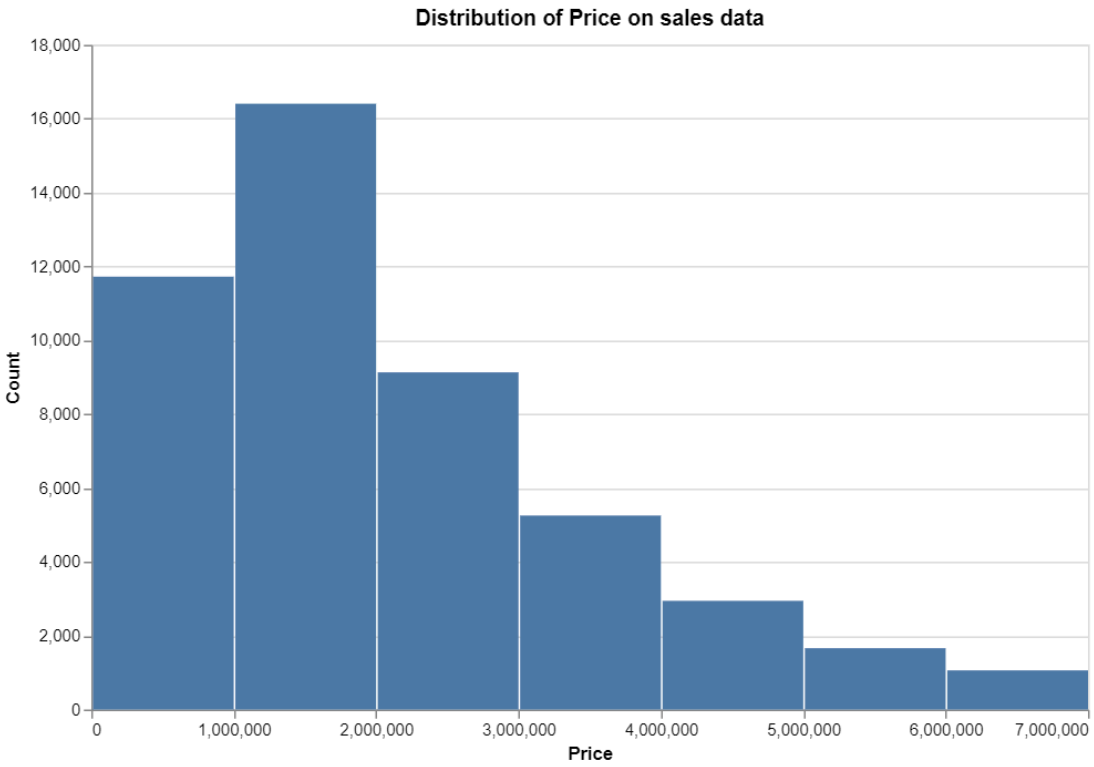


Region ○ Dubai ○ Abu Dhabi ○ Sharjah ○ Ras Al Khaimah ○ Ajman ○ Umm Al Quwain ◉ null

This graph shows the trend of the average price in locality across different regions in the UAE from 2021 to 2024. From this graph, it seems that prices have changed over time, with big regional differences. Dubai, while having the highest average prices in 2021, faced a steep drop by 2024. Other regions show moderate changes in prices, such as Abu Dhabi, Ajman, and Ras Al Khaimah. At the same time, some regions, such as Umm Al Quwain and Al Ain, did not exhibit wild fluctuations in price levels.

In [26]:
```
#create a histogram for the 'price' column
hist_price = alt.Chart(df_s_eda).mark_bar().encode(
    alt.X('price:Q', bin=True, title='Price'),
    alt.Y('count()', title='Count'),
    tooltip=['count()']
).properties(
    width=600,
    height=400,
    title='Distribution of Price on sales data'
)
hist_price
```

Out[26]:



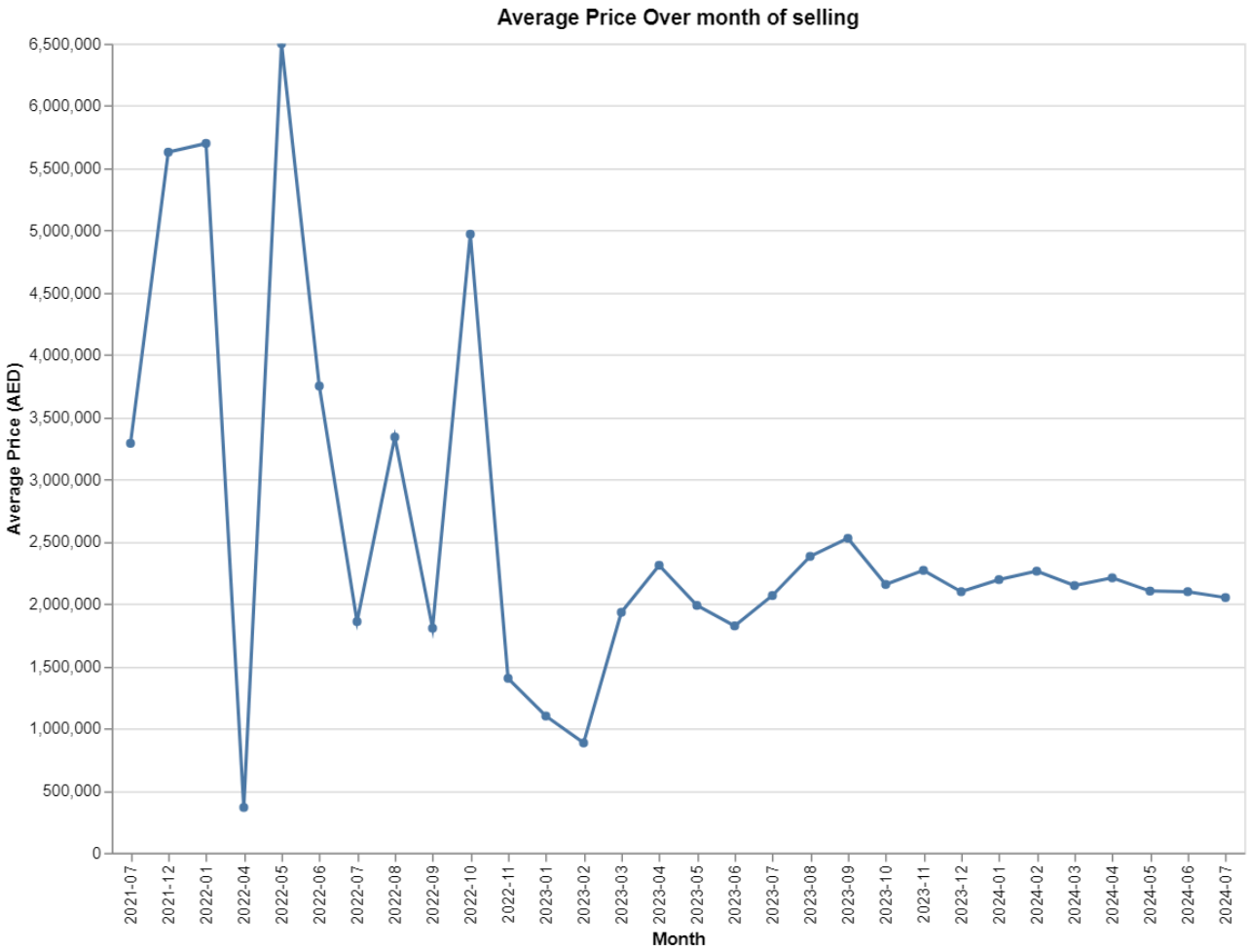
Create a histogram: The chart type is a histogram, suitable for visualizing the distribution of a quantitative variable (price). Encode data: The x-axis is encoded with the price variable, which is binned using the bin=True parameter. The y-axis is encoded with the count() function, which calculates the count of properties within each bin. Add tooltip: The tooltip parameter is used to add a tooltip that displays the count of properties for each bar. Customize the chart: The chart's title, axis labels, width, and height can be customized using Vega-Lite's options. Trend Skewness: A few expensive properties push the average price higher, as seen by the distribution's rightward skew. Concentration: A substantial percentage of the homes are priced in the lower tiers, indicating that most houses are reasonably priced.

Time series Analayis on price

In [27]:
```
# analysisng aveg price by month
df_s_eda['post_date'] = pd.to_datetime(df_s_eda['post_date'])

# grouping year and month converted to string format
df_s_eda['year_month'] = df_s_eda['post_date'].dt.to_period('M').astype(str)

# avg price per month
monthly_avg_price = df_s_eda.groupby('year_month')['price'].mean().reset_index()

#creating line chart
line_avg_price = alt.Chart(monthly_avg_price).mark_line(point=True).encode(
    x=alt.X('year_month:O', title='Month'),
    y=alt.Y('price:Q', title='Average Price (AED)'),
    tooltip=['year_month', 'price']
).properties(
    width=700,
    height=500,
    title='Average Price Over month of selling'
).interactive()

line_avg_price
```

```
<ipython-input-27-d2dda511b31a>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_s_eda['post_date'] = pd.to_datetime(df_s_eda['post_date'])
<ipython-input-27-d2dda511b31a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_s_eda['year_month'] = df_s_eda['post_date'].dt.to_period('M').astype(str)
```

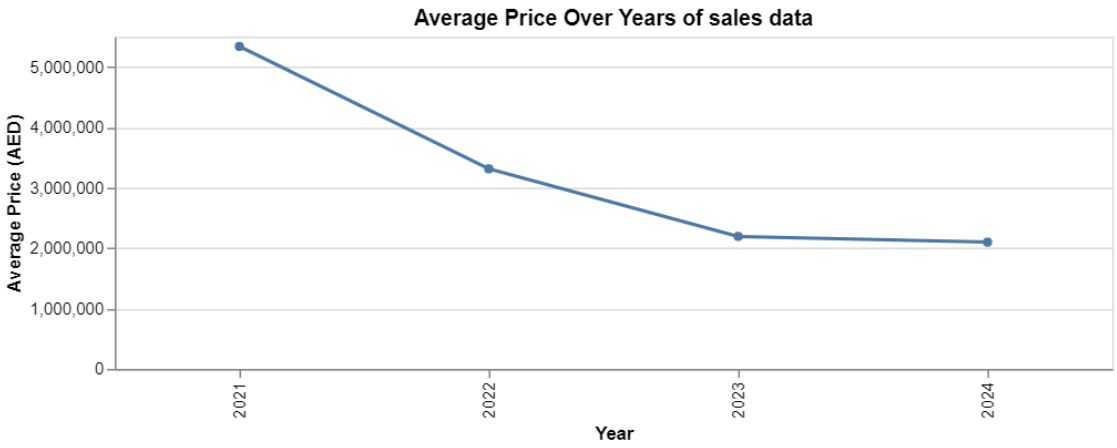
Out[27]:



Create a line chart: The chart type is a line chart, suitable for visualizing trends over time. Encode data: The x-axis is encoded with the Month of Selling variable, and the y-axis is encoded with the Average Price (AED) variable. Add markers: The line chart may have markers (points) at each data point to enhance visualization. Customize the chart: The chart's title, axis labels, grid lines, and overall appearance can be customized using Vega-Lite's options. Volatility: Over the course of the time period, there are notable swings in the average price. There are bursts of sudden spikes and drops, suggesting that the market is unstable. Seasonal Patterns: In certain years, there may be faint seasonal tendencies, however they are not particularly noticeable. For instance, prices may rise or fall throughout different months. Overall Increase: The average price appears to be generally rising despite some volatility, indicating that the market has been growing over time.

In [28]:
```
# Group data by year and calculate average price
yearly_data = df_s_eda.groupby('year', as_index=False)['price'].mean()

# Create a line chart for average price over years
line_yearly_price = alt.Chart(yearly_data).mark_line(point=True).encode(
    x=alt.X('year:O', title='Year'),
    y=alt.Y('price:Q', title='Average Price (AED)'),
    tooltip=['year', 'price']
).properties(
    width=600,
    height=200,
    title='Average Price Over Years of sales data'
).interactive()

line_yearly_price
```

Out[28]:



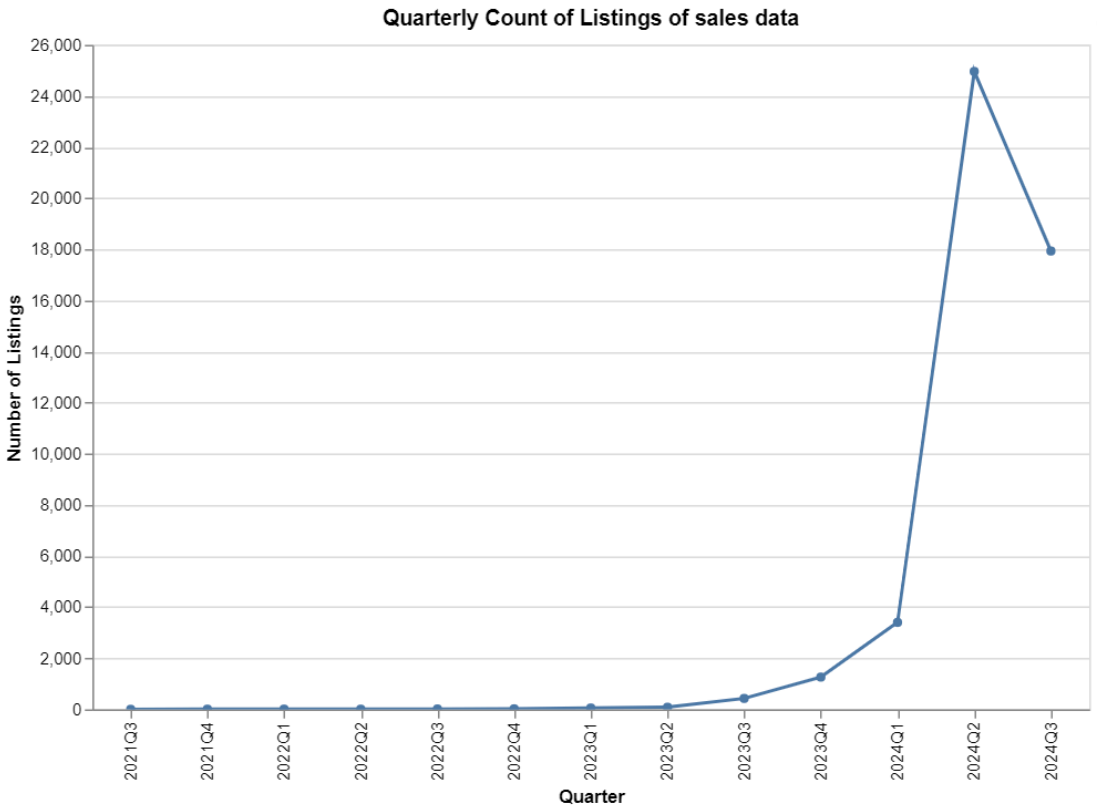
The sales line chart shows that, from 2021 and 2024, average house prices showed a discernible decline. The biggest reduction in prices was in 2022–2021, at which time the market saw fast transition. Prices dropped sharply during this time. This steep decline may be related to larger market disruptions, demand variations, or changes in the economy. After that, the pattern from 2022 to 2024 indicates a more moderate decline, indicating that although prices decreased, the pace of reduction reduced. This sustained decline in real estate values may be the consequence of continuing modifications to market dynamics,which may be impacted by variables including the state of the economy, consumer inclinations, or government actions.

In [29]:

```
# Extract year and quarter for grouping and convert to string format
df_s_eda['year_quarter'] = df_s_eda['post_date'].dt.to_period('Q').astype(str)

# Calculate count of listings per quarter
quarterly_listing_count = df_s_eda.groupby('year_quarter')['Locality'].count().reset_index()
quarterly_listing_count.rename(columns={'Locality': 'count'}, inplace=True)

# Line chart for quarterly count of listings
line_quarterly_count = alt.Chart(quarterly_listing_count).mark_line(point=True).encode(
    x=alt.X('year_quarter:O', title='Quarter'),
    y=alt.Y('count:Q', title='Number of Listings'),
    tooltip=['year_quarter', 'count']
).properties(
    width=600,
    height=400,
    title='Quarterly Count of Listings of sales data'
).interactive()

line_quarterly_count
```

```
<ipython-input-29-b8809edd7350>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_s_eda['year_quarter'] = df_s_eda['post_date'].dt.to_period('Q').astype(str)
```

Out[29]:



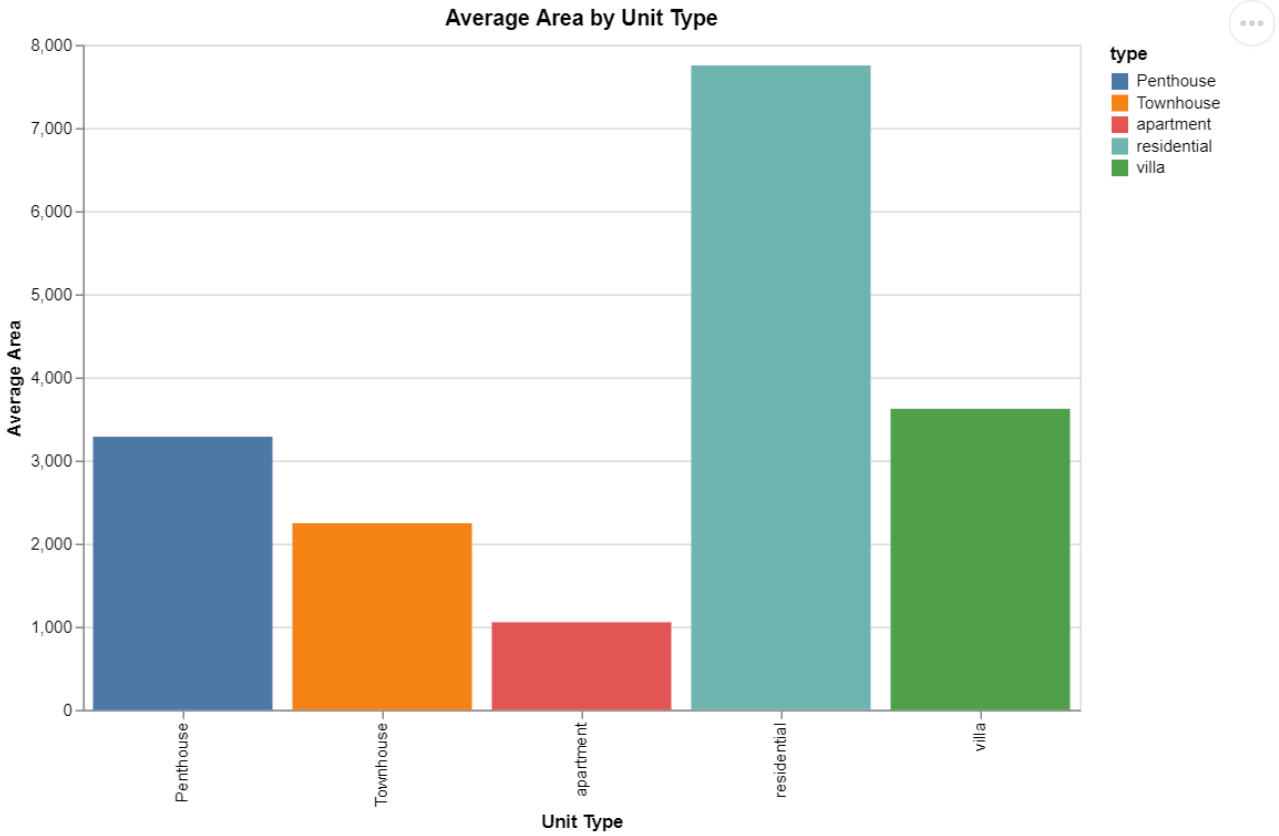
It is a line chart that represents the number of property listings on a quarterly basis. The time ranges from Q3 2021 until Q3 2024. First, there were low and stable numbers of listings that remained almost unchanged up to Q3 2023. As data was collected from this period. At that point, listings aggressively started to grow up to Q2 2024, where it reached more than 24,000. During Q3 2024, the number decreased from that of the past quarter, though higher in comparison with earlier periods. Comparison: Sharp Growth: The price has drastically gone up since Q4 2023, reaching its peak in Q2 2024. This may signal the boom period in the real estate market, or sudden growth of new property initiatives or price correction. Slowing Down Slightly: 2024 Q2-2024 Q3-In the post-peak period, listings have slightly fallen, which may be due to market saturation or supply reduction and change in demand. It means that this would be a dynamic trend in the real estate market-a growth with a majority growth period followed by stabilization or a slight correction.

Analysis on area

In [30]:

```
# avg area by types
avg_area_by_type = df_s_eda.groupby('type')['area'].mean().reset_index(name='avg_area')

avg_area_type_chart = alt.Chart(avg_area_by_type).mark_bar().encode(
    x=alt.X('type:N', title='Unit Type'),
    y=alt.Y('avg_area:Q', title='Average Area'),
    color='type:N',
    tooltip=['type', 'avg_area']
).properties(
    width=600,
    height=400,
    title='Average Area by Unit Type'
)

avg_area_type_chart
```

Out[30]:



The bar chart shows the average area by unit type for each of the selected property types. Of all the unit types, the average area for a residential unit is the highest, followed by villas and penthouses. Townhouses show an average area, while the lowest can be seen in apartments. It gives a reflection of space depending on the type of property, as viewed from apartments being compact to residential units that provide the most space. For encoding used colors for types.

Thank you