Motion_ DA Take Home Assignment_Zixuan (Gia) Gao

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import re
```

In [2]:

```python
df = pd.read_csv("Motion - Dataset - DA Take Home Assignment.csv")
```

1. data cleaning

In [3]:

```python
duplicate_rows = df[df.duplicated()]
print(duplicate_rows)

missing_values = df.isna()
```

```
Empty DataFrame
Columns: [Registration in Ireland, Make, Model, CC, Litres, Fuel, Bodytype, Transmission, Doors, Power, Year of First Registration, Taxclass, Price, Imported, Cou
nty, No. of Vehicles]
Index: []
```

In [4]:

```python
# Data type conversion
df['Registration in Ireland'] = pd.to_datetime(df['Registration in Ireland'], format='%d/%m/%Y', errors='coerce')

cols_to_str = ['Make', 'Model', 'Fuel', 'Bodytype', 'Transmission', 'Imported', 'County', 'Power']
df[cols_to_str] = df[cols_to_str].astype(str)

cols_to_numeric = ['CC', 'Litres', 'Doors', 'Year of First Registration', 'Price', 'No. of Vehicles']
df[cols_to_numeric] = df[cols_to_numeric].apply(pd.to_numeric, errors='coerce')
```

In [5]:

```python
# Fill the 'Model' column based on specific regex matching conditions
df['Model'] = df['Model'].where(~df['Model'].str.contains(r'\d{1,2}-[A-Za-z]{3}', na=False),
                                 df.groupby(['Make', 'Bodytype'])['Model'].transform(lambda x: x.mode().iloc[0] if not x.mode().empty else None))

# Fill 'CC', 'Litres', and 'Doors' columns
update_cols = ['CC', 'Litres', 'Doors']
conditions = (df['CC'] == 0) & (df['Fuel'] == 'DIESEL')
df.update(df[conditions].groupby(['Make', 'Model', 'Fuel', 'Bodytype'])[update_cols].transform(lambda x: x.mode().iloc[0]))

# Fill 'Litres' column (non-electric cars)
litres_condition = (df['Litres'] == 0) & (df['Fuel'] != 'ELECTRIC')
```

```
df['Litres'] = df['Litres'].where(~litres_condition,
                                  df.groupby(['Make', 'Model', 'Fuel', 'Bodytype'])['Litres'].transform(lambda x: x.mode().iloc[0] if not x.mode().empty else None

# Fill 'Doors' column
df['Doors'] = df['Doors'].where(df['Doors'] != 0,
                                df.groupby(['Make', 'Model', 'Fuel', 'Bodytype'])['Doors'].transform(lambda x: x.mode().iloc[0] if not x.mode().empty else None))

# Standardize the 'Bodytype' column, convert lowercase letters to uppercase
df['Bodytype'] = df['Bodytype'].apply(lambda x: re.sub(r'[a-z]', lambda m: m.group().upper(), x))

# Fill 'Price' column using mean value
price_condition = df['Price'] == -1
df['Price'] = df['Price'].where(~price_condition,
                                df.groupby(['Make', 'Model', 'Fuel', 'Bodytype'])['Price'].transform(lambda x: x[x != -1].mean() if not x.empty else None))
```

In [6]:

```
print(df.dtypes)
```

```
Registration in Ireland       datetime64[ns]
Make                                  object
Model                                 object
CC                                   float64
Litres                               float64
Fuel                                  object
Bodytype                              object
Transmission                          object
Doors                                float64
Power                                 object
Year of First Registration             int64
Taxclass                              object
Price                                float64
Imported                              object
County                                object
No. of Vehicles                        int64
dtype: object
```

In [8]:

```
df.to_csv("updated_motion_data.csv", index=False)
```

2. Analysis

Note: The data for 2024 is up to April, which means it represents 1/3 of the year. In the following analysis, the lower values for some variables in 2024 do not represent the full year's sales, but rather the sales up to April.

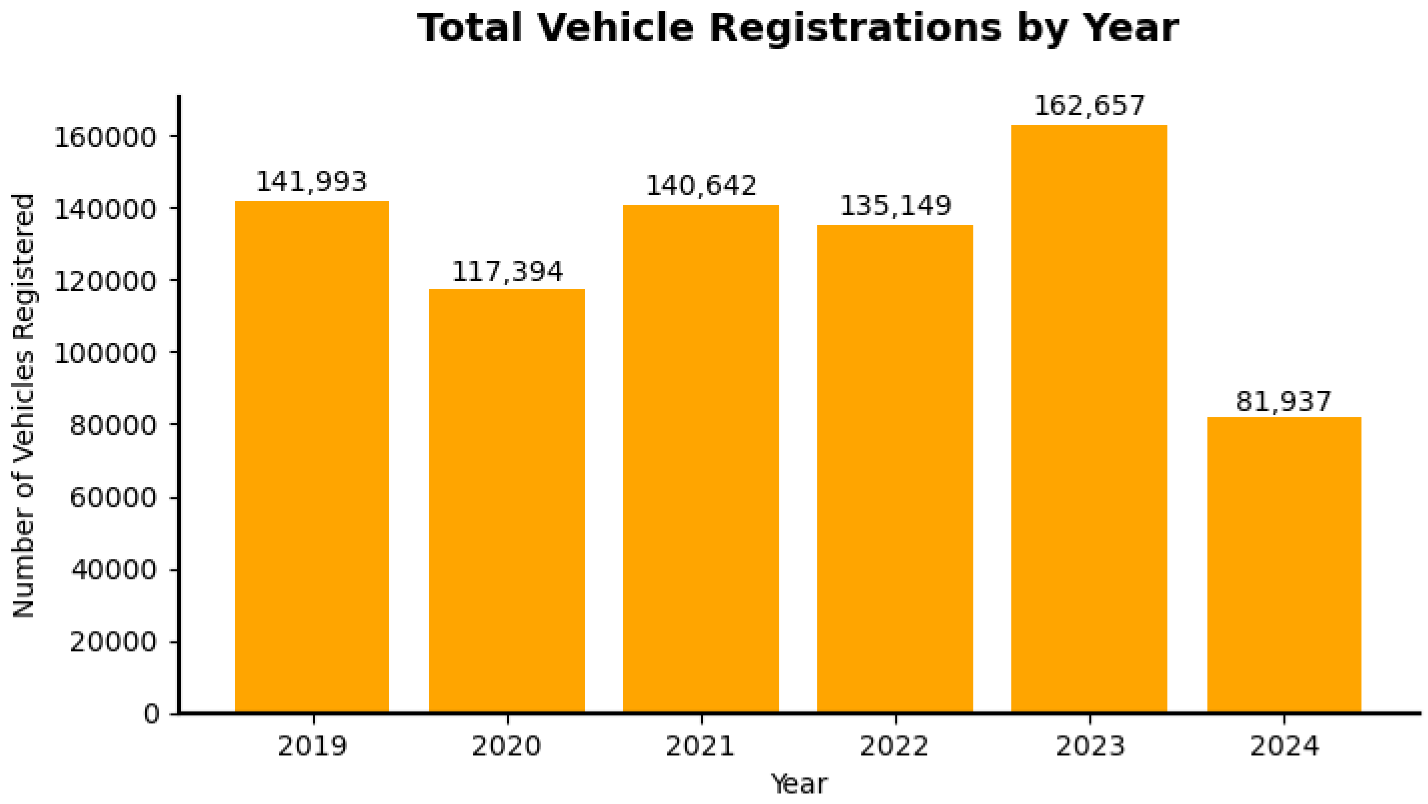
Q1) Do a topline analysis of the Irish Market.

In [7]:

```
def get_custom_colors():
    return ['#FF6F00',  # Deep orange
            '#FF8C00',  # Slightly deeper orange
            '#FFA500',  # Ginger yellow
            '#FFB347',  # Slightly lighter orange
            '#FFBF00',  # Bright orange
            '#FFD700',  # Golden yellow
            '#FFE066',  # Bright light yellow
            '#FFF0B3',  # Lightest yellow
            '#F0A202',  # Golden brown
            '#E67E22',  # Carrot color
            '#FF9933',  # Bright ginger yellow
            '#FFCC66',  # Light orange
            '#FFC300',  # Lemon orange
            '#FFDD99',  # Light orange yellow
            '#FFBB33',  # Bright yellow
            '#FFD34E',  # Bright golden yellow
            '#FFDA75',  # Beige
            '#FFE8A1',  # Milky yellow
            '#FFF1C1',  # Light milky yellow
            '#FFF5E1']  # Lightest milky yellow
```

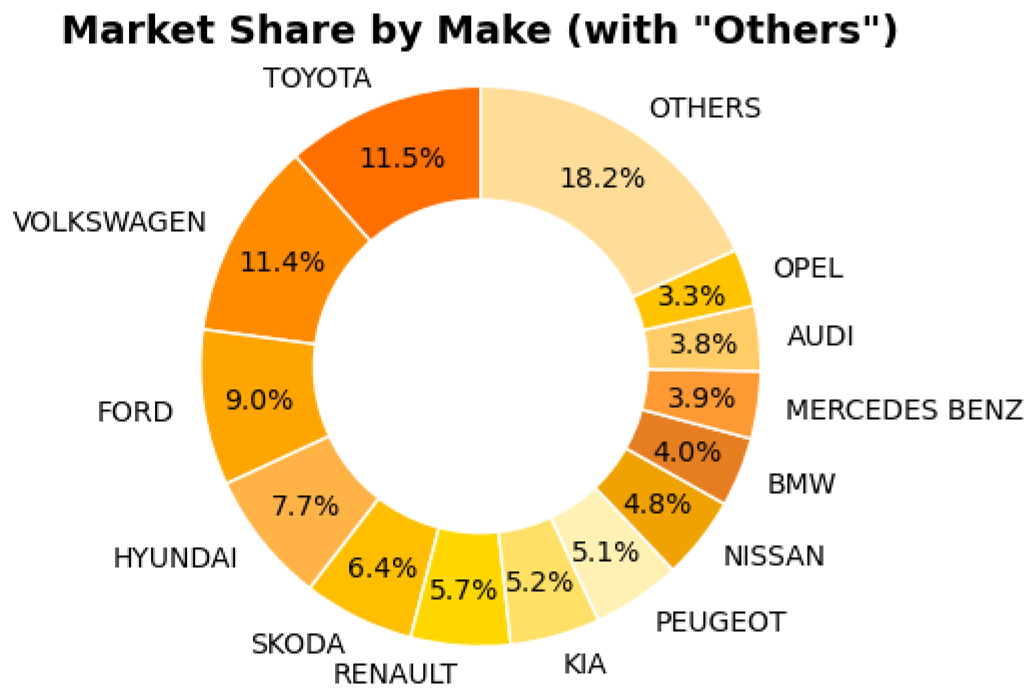
In [8]:

```
# 1. total number of vehicle registrations by year(2019-2024/4) with bar chart

# Extract the year and sum the number of vehicle registrations
df['Year'] = df['Registration in Ireland'].dt.year
yearly_registration = df.groupby('Year')['No. of Vehicles'].sum().reset_index()

plt.figure(figsize=(8, 4))
bars = plt.bar(yearly_registration['Year'], yearly_registration['No. of Vehicles'], color='#FFA500')

for bar in bars:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() * 1.02, f'{bar.get_height():,}', ha='center', fontsize=10)

plt.title('Total Vehicle Registrations by Year', fontsize=14, fontweight='bold', pad=20)
plt.xlabel('Year', fontsize=10)
plt.ylabel('Number of Vehicles Registered', fontsize=10)

plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['bottom'].set_linewidth(1.5)
plt.gca().spines['left'].set_linewidth(1.5)

plt.xticks(yearly_registration['Year'])
plt.show()
```



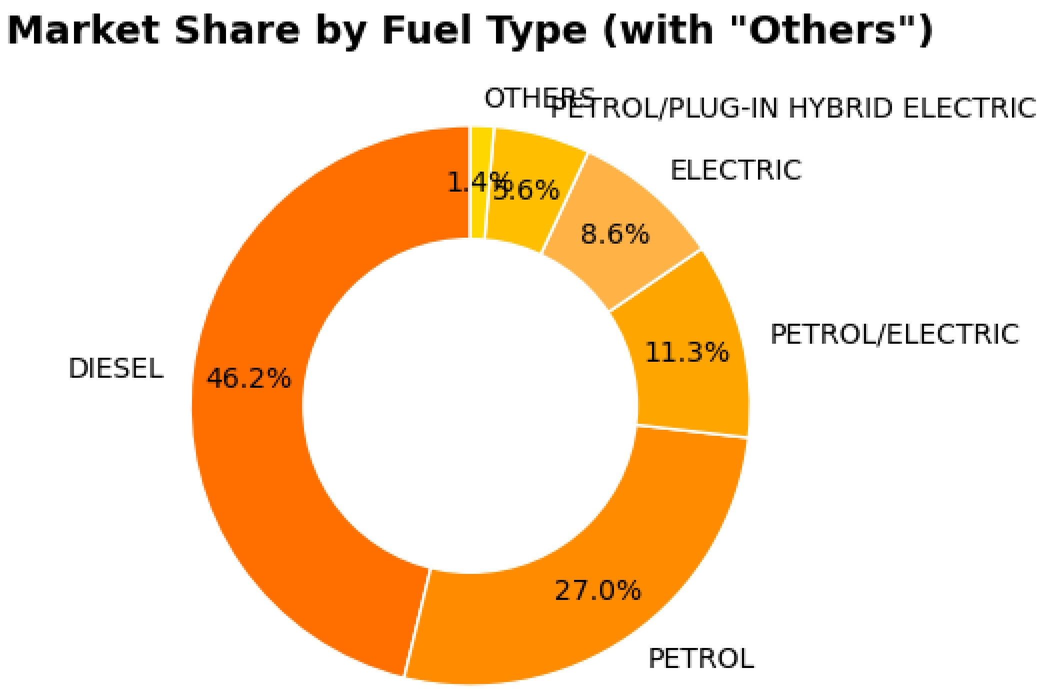
In [9]:

```
# 2. Market share of different make with Donut Chart

make_market_share = df.groupby('Make')['No. of Vehicles'].sum().sort_values(ascending=False)

# Set a threshold to classify brands with a share of less than 3% as "Others"
threshold_make = 0.03 * make_market_share.sum()
othersmake_share = make_market_share[make_market_share < threshold_make].sum()
make_market_share = make_market_share[make_market_share >= threshold_make]
make_market_share['OTHERS'] = othersmake_share

plt.figure(figsize=(5, 4))
plt.pie(make_market_share, labels=make_market_share.index, autopct='%1.1f%%', startangle=90,
        colors=get_custom_colors(), wedgeprops={'edgecolor': 'white'}, pctdistance=0.8)

plt.gca().add_artist(plt.Circle((0, 0), 0.6, fc='white'))

plt.title('Market Share by Make (with "Others")', fontsize=14, fontweight='bold')
plt.axis('equal')
plt.show()
```



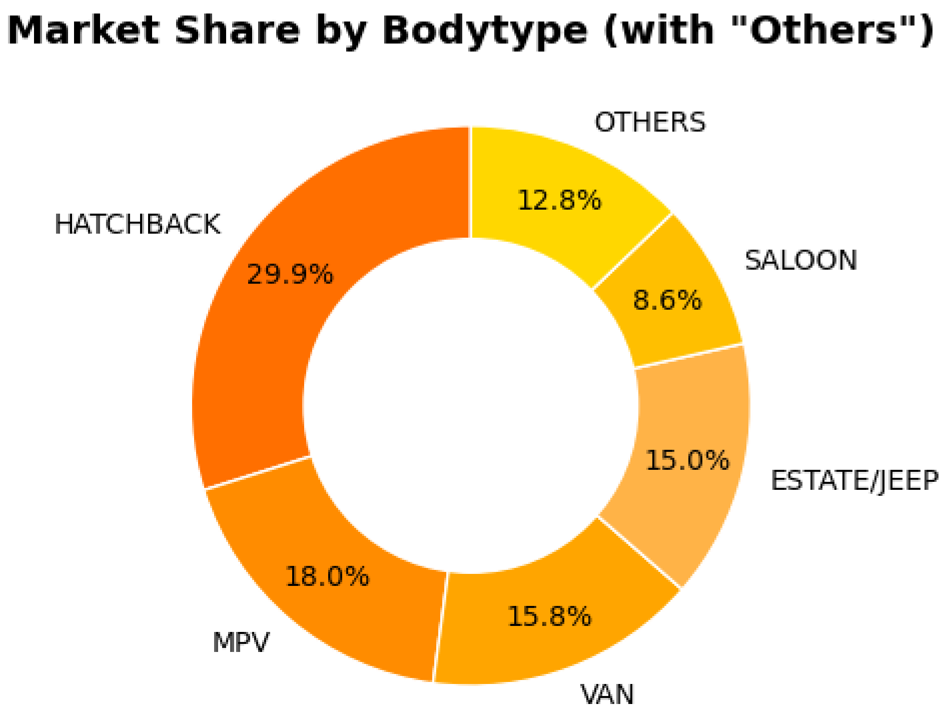
In [10…

```
# 3. Market share of different Fuel type with Donut Chart

fuel_market_share = df.groupby('Fuel')['No. of Vehicles'].sum().sort_values(ascending=False)

# Set a threshold to classify brands with a share of less than 5% as "Others"
threshold_fuel = 0.05 * fuel_market_share.sum()
othersfuel_share = fuel_market_share[fuel_market_share < threshold_fuel].sum()
othersfuel_labels = fuel_market_share[fuel_market_share < threshold_fuel].index.tolist()
fuel_market_share = fuel_market_share[fuel_market_share >= threshold_fuel]
fuel_market_share['OTHERS'] = othersfuel_share
df['Fuel'] = df['Fuel'].apply(lambda x: 'OTHERS' if x in othersfuel_labels else x)

plt.figure(figsize=(5, 4))
plt.pie(fuel_market_share, labels=fuel_market_share.index, autopct='%1.1f%%', startangle=90,
        colors=get_custom_colors(), wedgeprops={'edgecolor': 'white'}, pctdistance=0.80)

plt.gca().add_artist(plt.Circle((0, 0), 0.60, fc='white'))

plt.title('Market Share by Fuel Type (with "Others")', fontsize=14, fontweight='bold', pad=20)
plt.axis('equal')
plt.show()
```



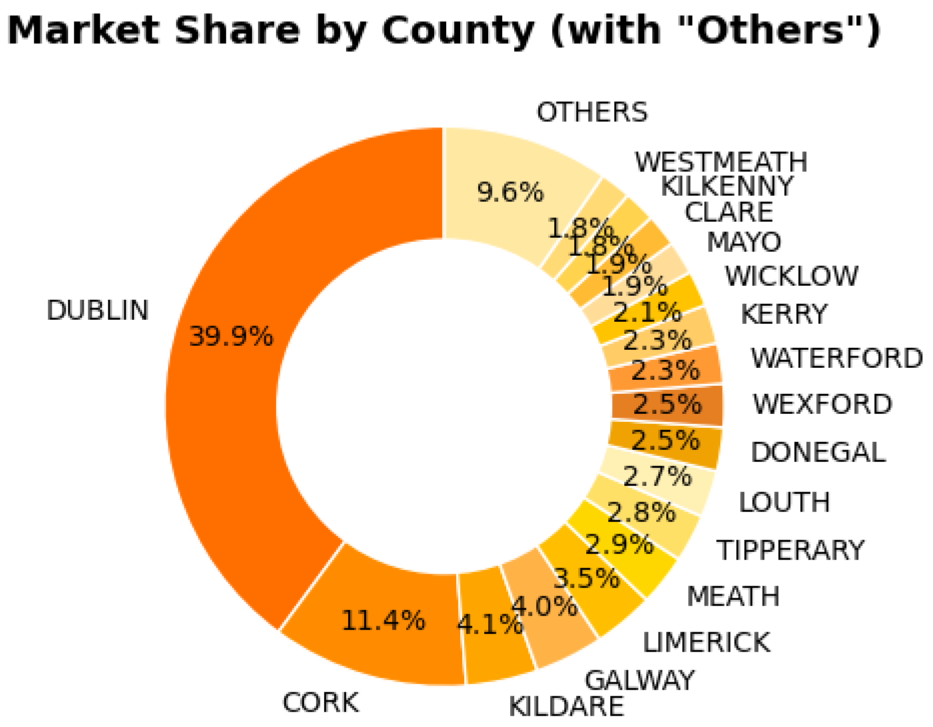
In [11…

```
# 4. Market share of different Bodytype with Donut Chart

bodytype_market_share = df.groupby('Bodytype')['No. of Vehicles'].sum().sort_values(ascending=False)

# Set a threshold to classify brands with a share of less than 8% as "Others"
threshold_bodytype = 0.08 * bodytype_market_share.sum()
othersbodytype_share = bodytype_market_share[bodytype_market_share < threshold_bodytype].sum()
othersbodytype_labels = bodytype_market_share[bodytype_market_share < threshold_bodytype].index.tolist()
bodytype_market_share = bodytype_market_share[bodytype_market_share >= threshold_bodytype]
bodytype_market_share['OTHERS'] = othersbodytype_share
df['Bodytype'] = df['Bodytype'].apply(lambda x: 'OTHERS' if x in othersfuel_labels else x)

plt.figure(figsize=(5, 4))
plt.pie(bodytype_market_share, labels=bodytype_market_share.index, autopct='%1.1f%%', startangle=90,
        colors=get_custom_colors(), wedgeprops={'edgecolor': 'white'}, pctdistance=0.80)

plt.gca().add_artist(plt.Circle((0, 0), 0.60, fc='white'))

plt.title('Market Share by Bodytype (with "Others")', fontsize=14, fontweight='bold', pad=20)
plt.axis('equal')
plt.show()
```



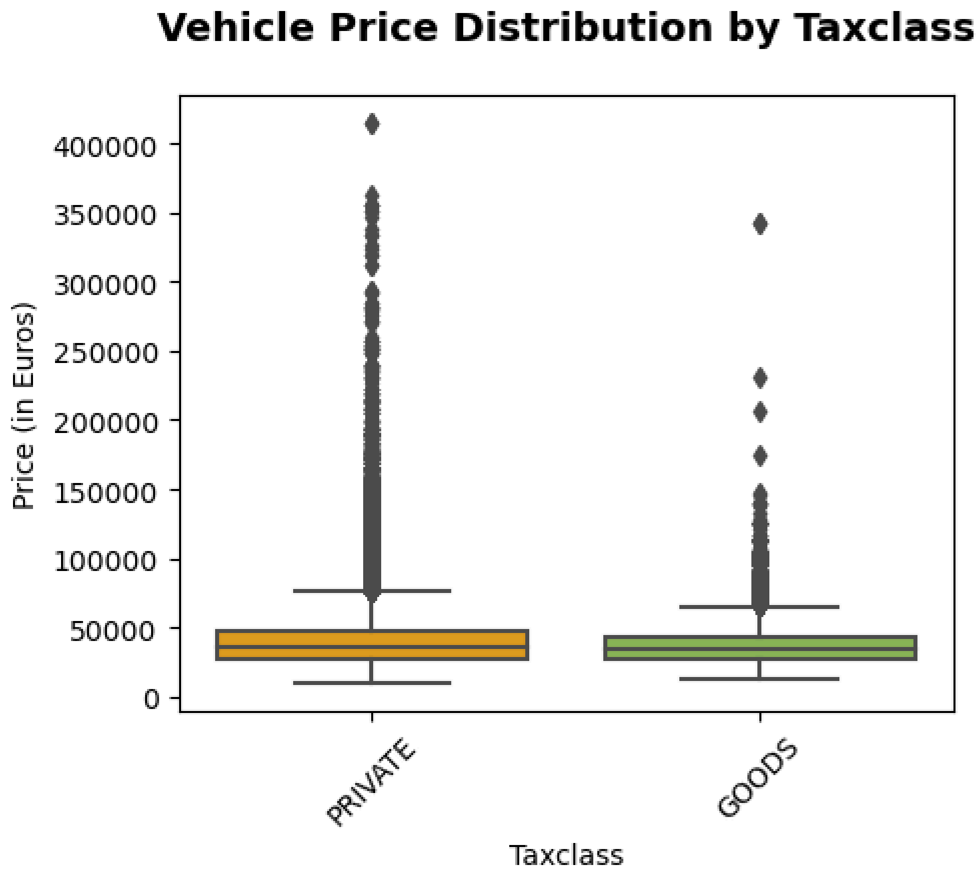
In [12…

```
# 5. Market share of different County with Donut Chart

county_market_share = df.groupby('County')['No. of Vehicles'].sum().sort_values(ascending=False)

# Set a threshold to classify brands with a share of less than 5% as "Others"
threshold_county = 0.015 * county_market_share.sum()
otherscounty_share = county_market_share[county_market_share < threshold_county].sum()
otherscounty_labels = county_market_share[county_market_share < threshold_county].index.tolist()
county_market_share = county_market_share[county_market_share >= threshold_county]
county_market_share['OTHERS'] = otherscounty_share
df['County'] = df['County'].apply(lambda x: 'OTHERS' if x in othersfuel_labels else x)

plt.figure(figsize=(5,4))
plt.pie(county_market_share, labels=county_market_share.index, autopct='%1.1f%%', startangle=90,
        colors=get_custom_colors(), wedgeprops={'edgecolor': 'white'}, pctdistance=0.80)

plt.gca().add_artist(plt.Circle((0, 0), 0.60, fc='white'))

plt.title('Market Share by County (with "Others")', fontsize=14, fontweight='bold', pad=20)
plt.axis('equal')
plt.show()
```



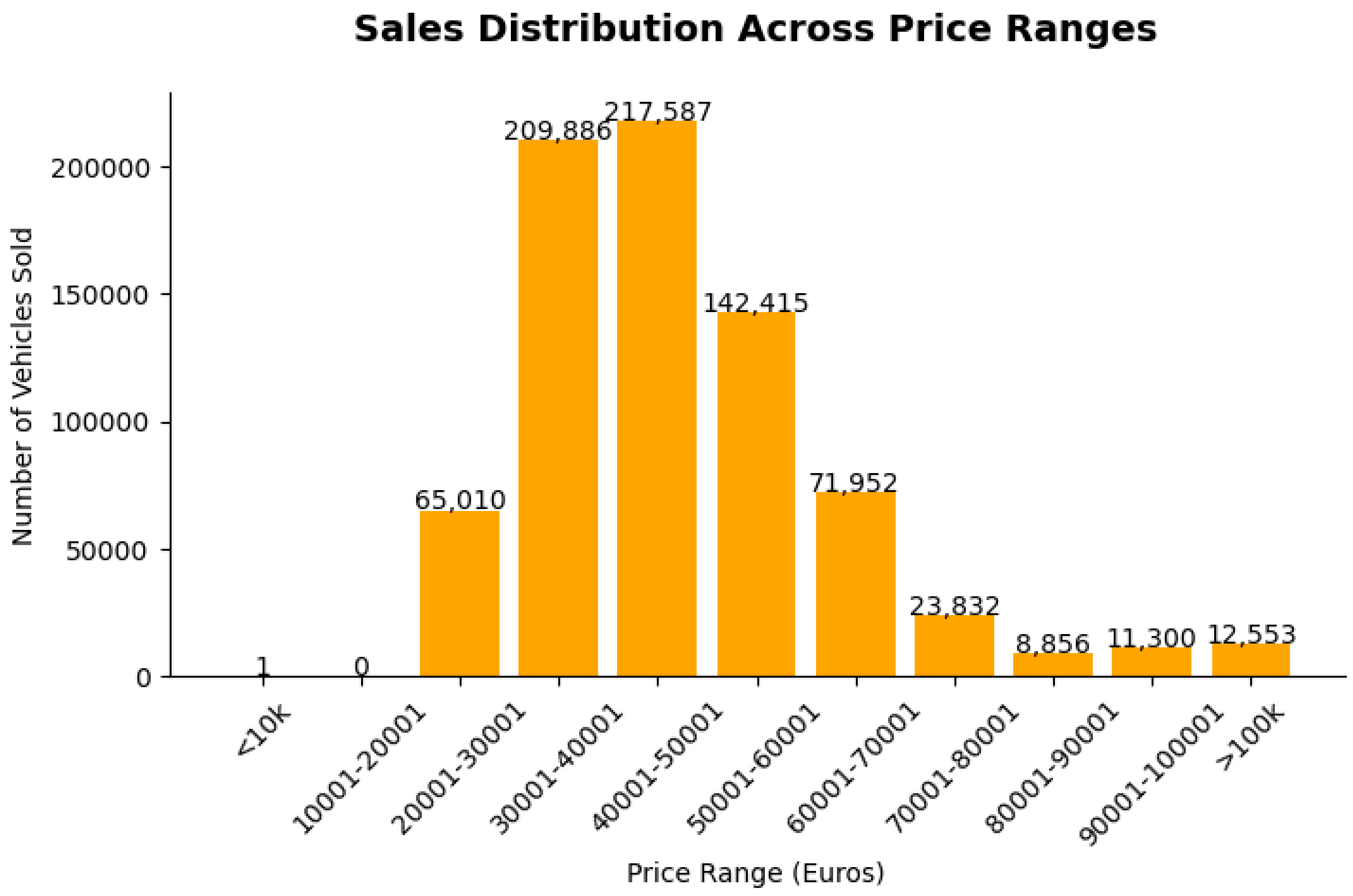
In [13…

```
# 6. Vehicle Price Distribution by Taxclass with boxplot

colors = ['#FFA500', '#8BC34A']

plt.figure(figsize=(5, 4))
sns.boxplot(x='Taxclass', y='Price', data=df, palette=colors)

plt.title('Vehicle Price Distribution by Taxclass', fontsize=14, fontweight='bold', pad=20)
plt.xlabel('Taxclass', fontsize=10)
plt.ylabel('Price (in Euros)', fontsize=10)
plt.xticks(rotation=45)

plt.show()
```



In [14…

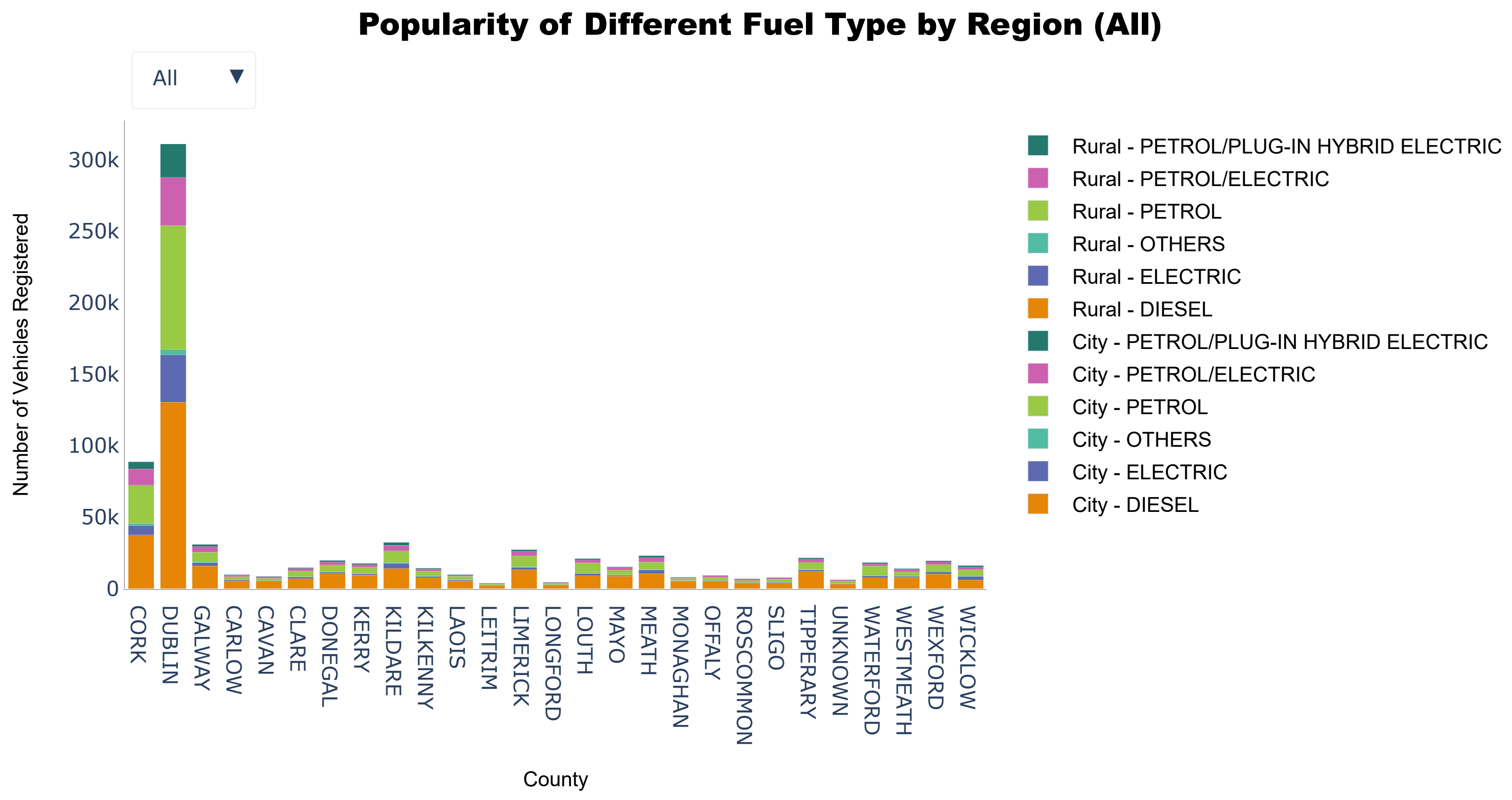
```
# 7. Accepted Price Range with bar chart

# Define bins: less than 10k, 10k to 100k in 10k increments, and greater than 100k
bins = [0, 10000] + list(range(10001, 100001, 10000)) + [df['Price'].max()]
labels = ['<10k'] + [f'{i}-{i+10000}' for i in range(10001, 100000, 10000)] + ['>100k']

df['Price Range'] = pd.cut(df['Price'], bins=bins, labels=labels, include_lowest=True)
price_range_sales = df.groupby('Price Range')['No. of Vehicles'].sum()

plt.figure(figsize=(8, 4))
bars = plt.bar(price_range_sales.index, price_range_sales, color='#FFA500')

for bar in bars:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 500, f'{bar.get_height():,}', ha='center', fontsize=10)

plt.title('Sales Distribution Across Price Ranges', fontsize=14, fontweight='bold', pad=20)
plt.xlabel('Price Range (Euros)', fontsize=10)
plt.ylabel('Number of Vehicles Sold', fontsize=10)
plt.xticks(rotation=45)

plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

plt.show()
```



In [15…

```
# 8. Popularity of Different Fuel Type by city and rural county with interactive stacked bar chart

# Create city and rural classification
city_list = ['DUBLIN', 'CORK', 'GALWAY']
df['Region'] = df['County'].apply(lambda x: 'City' if x in city_list else 'Rural')

# Group by 'Region', 'County', and 'Fuel' to aggregate vehicle counts
region_fuel_distribution = df.groupby(['Region', 'County', 'Fuel'])['No. of Vehicles'].sum().reset_index()

fig = go.Figure()

# Add data for city and rural regions
def add_region_data(region_data, region_name, visible=False):
    for fuel_type in region_data['Fuel'].unique():
        filtered_data = region_data[region_data['Fuel'] == fuel_type]
        fig.add_trace(go.Bar(
            x=filtered_data['County'],
            y=filtered_data['No. of Vehicles'],
            name=f'{region_name} - {fuel_type}',
            visible=visible,
            marker=dict(color=px.colors.qualitative.Vivid[region_data['Fuel'].unique().tolist().index(fuel_type) % len(px.colors.qualitative.Vivid)])
        ))
```

```
# Add city and rural data
add_region_data(region_fuel_distribution[region_fuel_distribution['Region'] == 'City'], 'City', visible=True)
add_region_data(region_fuel_distribution[region_fuel_distribution['Region'] == 'Rural'], 'Rural')

# Add a dropdown selector to choose between city or rural data
fig.update_layout(
    updatemenus=[
        {
            'buttons': [
                {
                    'label': 'All',
                    'method': 'update',
                    'args': [
                        {'visible': [True] * len(fig.data)},
                        {'title': 'Popularity of Different Fuel Type by Region (All)'}
                    ]
                },
                {
                    'label': 'City',
                    'method': 'update',
                    'args': [
                        {'visible': [trace.name.startswith('City') for trace in fig.data]},
                        {'title': 'Popularity of Different Fuel Type by Region (City)'}
                    ]
                },
                {
                    'label': 'Rural',
                    'method': 'update',
                    'args': [
                        {'visible': [trace.name.startswith('Rural') for trace in fig.data]},
                        {'title': 'Popularity of Different Fuel Type by Region (Rural)'}
                    ]
                }
            ],
            'direction': 'down',
            'showactive': True,
            'x': 0.15,
            'y': 1.15
        }
    ],
    xaxis_title="County",
    yaxis_title="Number of Vehicles Registered",
    title=dict(font=dict(family="Arial Black", size=18, color='black'), x=0.5, xanchor='center'),
    xaxis=dict(
        title=dict(font=dict(family="Arial", size=12, color='black')),
        showline=True,
        linecolor='black',
        showgrid=False,
    ),
    yaxis=dict(
        title=dict(font=dict(family="Arial", size=12, color='black')),
        showline=True,
        linecolor='black',
        showgrid=False,
    ),
    legend=dict(font=dict(family="Arial", size=12, color='black')),
    barmode='stack',
    plot_bgcolor='white',
    height=500,
    width=900
)

fig.show()
fig.write_html("Popularity of Different Fuel Type by city and rural county.html")
```

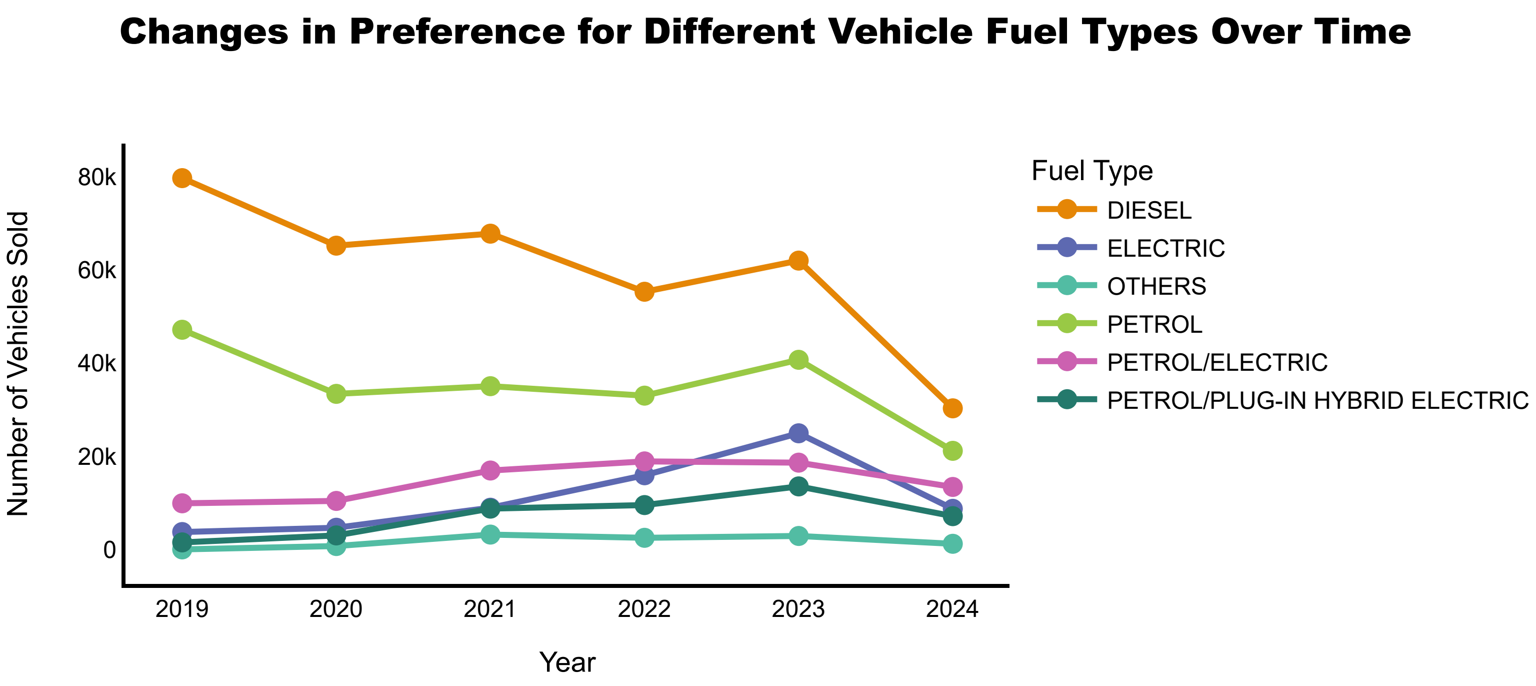


Q2) How have buyer preferences evolved over time?

In [16…

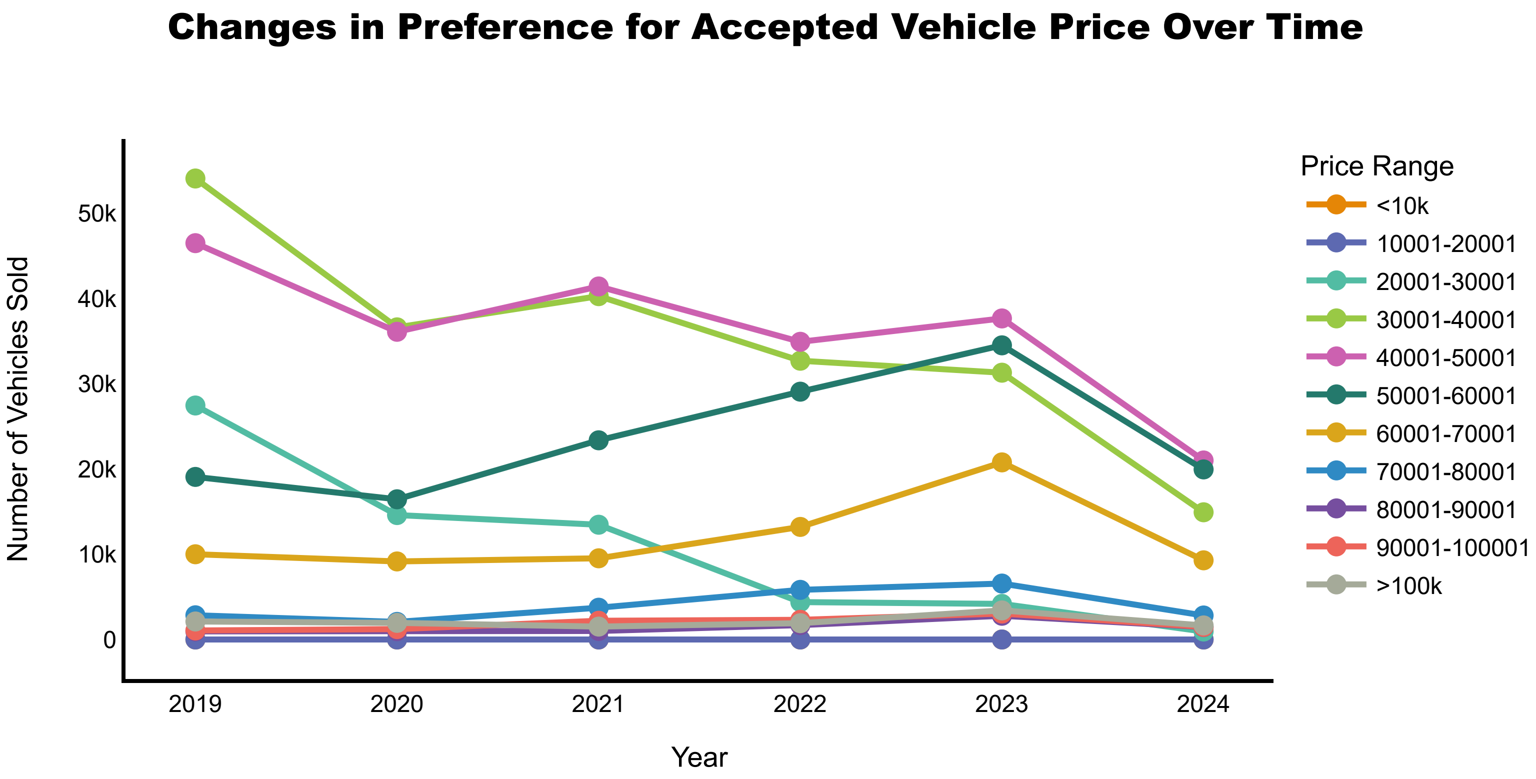
```
# 9. Changes in Preference for Different Vehicle Fuel Types Over Time with interactive line chart

# Number of vehicles registered in Ireland by year, month and fuel type
```

```
df['Year'] = df['Registration in Ireland'].dt.year
fuel_trend = df.groupby(['Year', 'Fuel'])['No. of Vehicles'].sum().reset_index()

fig = px.line(fuel_trend, x='Year', y='No. of Vehicles', color='Fuel',
              title='Changes in Preference for Different Vehicle Fuel Types Over Time',
              markers=True,
              color_discrete_sequence=px.colors.qualitative.Vivid)

fig.update_traces(marker_size=10, line_width=3)

fig.update_layout(
    width=800, height=400,
    title=dict(text='Changes in Preference for Different Vehicle Fuel Types Over Time', font=dict(family="Arial Black", size=18, color='black'),
               x=0.5, xanchor='center'),
    xaxis_title="Year",
    yaxis_title="Number of Vehicles Sold",

    xaxis=dict(showline=True, showgrid=False, linecolor='black', linewidth=2),
    yaxis=dict(showline=True, showgrid=False, linecolor='black', linewidth=2),

    legend_title="Fuel Type",
    plot_bgcolor='white',
    font=dict(family="Arial", size=12, color='black')
)

fig.show()
fig.write_html("Changes in Preference for Different Vehicle Fuel Types Over Time.html")
```



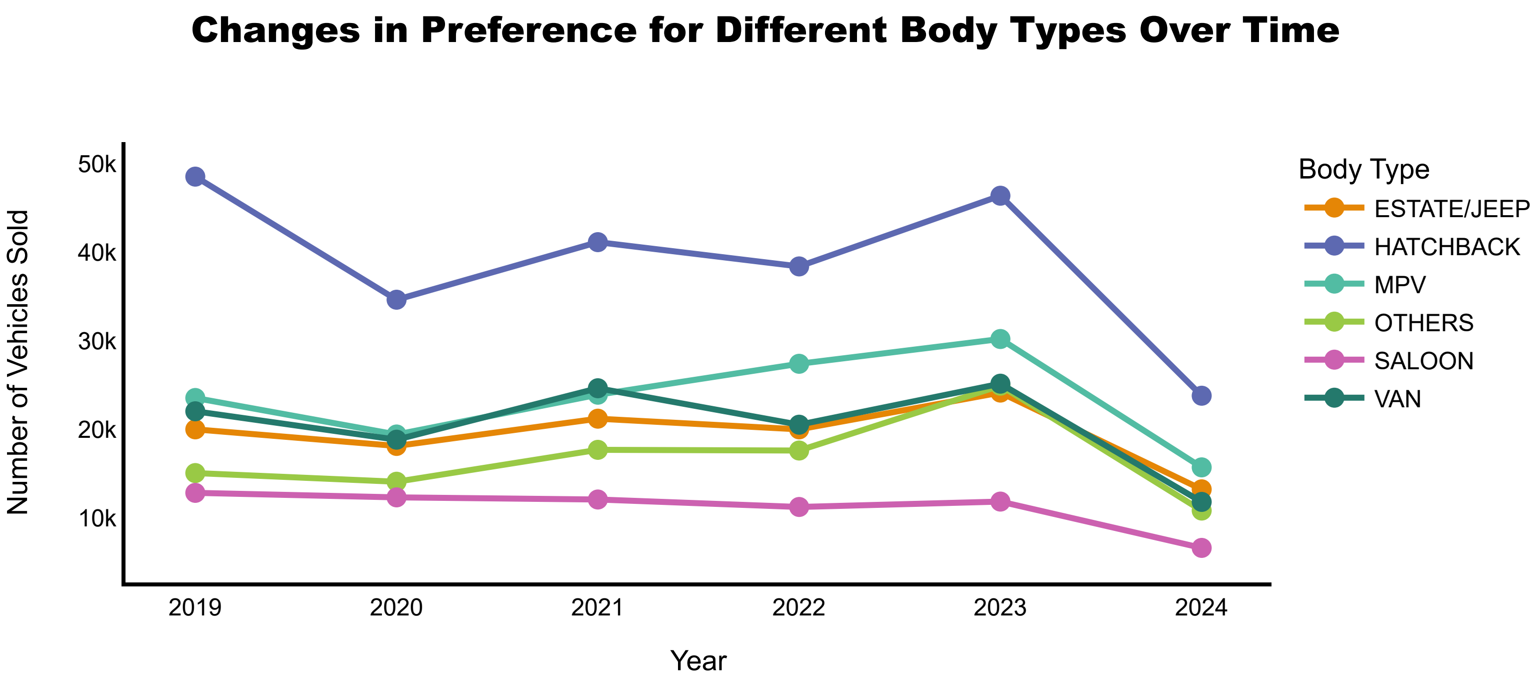
In [17…

```
# 9. Changes in Preference for Accepted Vehicle Price Over Time with interactive line chart

# Summarize the number of vehicles per year and price range
price_demand_by_year = df.groupby(['Year of First Registration', 'Price Range'])['No. of Vehicles'].sum().reset_index()

# Create the line chart for vehicle demand by price range and year
fig = px.line(price_demand_by_year,
              x='Year of First Registration',
              y='No. of Vehicles',
              color='Price Range',
              labels={'No. of Vehicles': 'Number of Vehicles Sold'},
              title='Changes in Preference for Accepted Vehicle Price Over Time',
              markers=True,
              color_discrete_sequence=px.colors.qualitative.Vivid)

fig.update_traces(marker_size=10, line_width=3)

fig.update_layout(
    width=800, height=450,
    title=dict(text='Changes in Preference for Accepted Vehicle Price Over Time', x=0.5, xanchor='center', font=dict(family="Arial Black", size=18, color='black')
    xaxis_title="Year",
    yaxis_title="Number of Vehicles Sold",

    xaxis=dict(showline=True, linecolor='black', linewidth=2, showgrid=False),
    yaxis=dict(showline=True, linecolor='black', linewidth=2, showgrid=False),

    plot_bgcolor='white',
    font=dict(family="Arial", size=12, color='black'),
    legend_title="Price Range"
)

fig.show()
fig.write_html("Changes in Preference for Accepted Vehicle Price Over Time.html")
```



In [18…

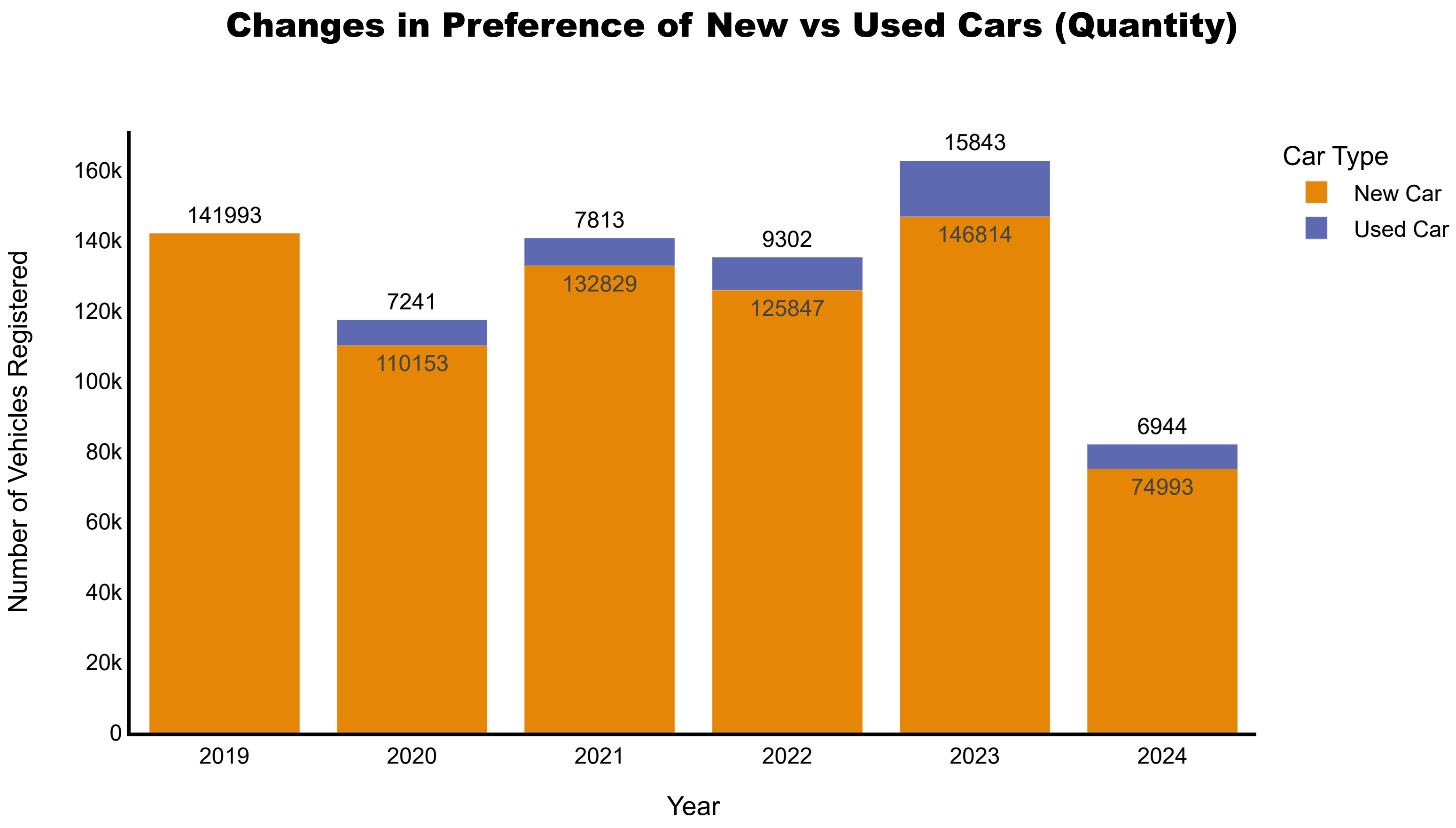
```
# 10. Changes in Preference for Different Body Types Over Time with interactive line chart

df['Bodytype'] = df['Bodytype'].apply(lambda x: 'OTHERS' if x in othersbodytype_labels else x)
body_type_trend = df.groupby(['Year', 'Bodytype'])['No. of Vehicles'].sum().reset_index()

fig = px.line(body_type_trend, x='Year', y='No. of Vehicles', color='Bodytype',
              labels={'No. of Vehicles': 'Number of Vehicles Sold', 'Year': 'Year'},
              title='Changes in Preference for Different Body Types Over Time',
              markers=True,
              color_discrete_sequence=px.colors.qualitative.Vivid)

fig.update_traces(marker=dict(size=10, symbol='circle'),
                  line=dict(width=3))

fig.update_layout(
    width=800,
    height=400,
    title=dict(font=dict(family="Arial Black", size=18, color='black'), x=0.5, xanchor='center'),
    xaxis=dict(
        showgrid=False,
        zeroline=True,
        linecolor='black',
        linewidth=2,
        title='Year'
    ),
    yaxis=dict(
        showgrid=False,
        zeroline=True,
        linecolor='black',
        linewidth=2,
        title='Number of Vehicles Sold'
    ),
    legend=dict(title="Body Type", font=dict(family="Arial", size=12, color='black'), bordercolor='gray', borderwidth=0),
    plot_bgcolor='white',
    font=dict(family="Arial", size=12, color='black')
)

fig.show()
fig.write_html("Changes in Preference for Different Body Types Over Time.html")
```

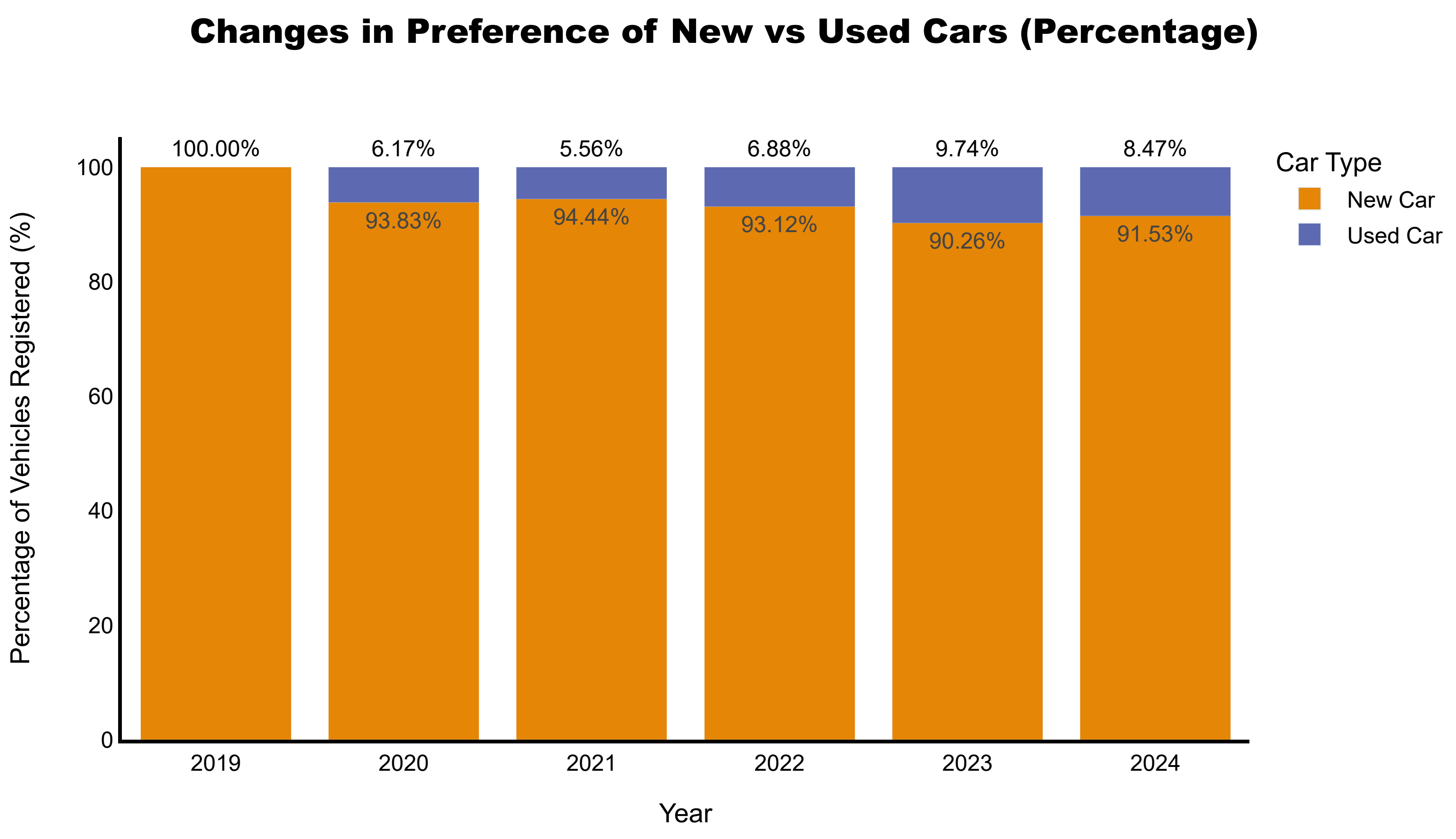


In [19…

```
# 11. Changes in Preference for new vs used cars over time with interactive stacked bar chart

df['Year of First Registration'] = pd.to_numeric(df['Year of First Registration'], errors='coerce')
df['Year in Ireland'] = df['Registration in Ireland'].dt.year

# Label new and used cars: if 'Year of First Registration' matches 'Year in Ireland', it is a new car, otherwise it is a used car
df['Car Type'] = df.apply(lambda row: 'New Car' if row['Year of First Registration'] == row['Year in Ireland'] else 'Used Car', axis=1)

# Aggregate the number of vehicles by year and car type
car_registration_trend = df.groupby(['Year in Ireland', 'Car Type'])['No. of Vehicles'].sum().reset_index()

# Calculate the percentage for each car type by year
car_registration_trend['Percentage'] = car_registration_trend.groupby('Year in Ireland')['No. of Vehicles'].transform(lambda x: x / x.sum() * 100)
```

```
def create_bar_chart(df, y_value, y_label, title, text_format):
    fig = px.bar(df, x='Year in Ireland', y=y_value, color='Car Type',
                 labels={y_value: y_label, 'Year in Ireland': 'Year'},
                 title=title, text=y_value,
                 color_discrete_sequence=px.colors.qualitative.Vivid,
                 hover_data=['No. of Vehicles', 'Percentage'])

    fig.update_traces(texttemplate=text_format, textposition='outside')

    fig.update_layout(
        width=800,
        height=500,
        title=dict(font=dict(family="Arial Black", size=18, color='black'), x=0.5, xanchor='center'),

        xaxis=dict(showline=True, linecolor='black', linewidth=2, showgrid=False),
        yaxis=dict(showline=True, linecolor='black', linewidth=2, showgrid=False, title=y_label),

        barmode='stack',
        legend=dict(title="Car Type", font=dict(family="Arial", size=12, color='black'), bordercolor='gray', borderwidth=0),
        plot_bgcolor='white',
        font=dict(family="Arial", size=12, color='black')
    )
    return fig

fig_quantity = create_bar_chart(car_registration_trend, 'No. of Vehicles', 'Number of Vehicles Registered', 'Changes in Preference of New vs Used Cars (Quantity)'
fig_percentage = create_bar_chart(car_registration_trend, 'Percentage', 'Percentage of Vehicles Registered (%)', 'Changes in Preference of New vs Used Cars (Percen

fig_quantity.show()
fig_percentage.show()
fig.write_html("Changes in Preference of New vs Used Cars (Quantity).html")
fig.write_html("Changes in Preference of New vs Used Cars (Percentage).html")
```


Changes in Preference of New vs Used Cars (Quantity)
Number of Vehicles Registered
0
20k
40k
60k
80k
100k
120k
140k
160k
141993
7241
110153
7813
132829
9302
125847
15843
146814
6944
74993
2019
2020
2021
2022
2023
2024
Year
Car Type
New Car
Used Car


Changes in Preference of New vs Used Cars (Percentage)

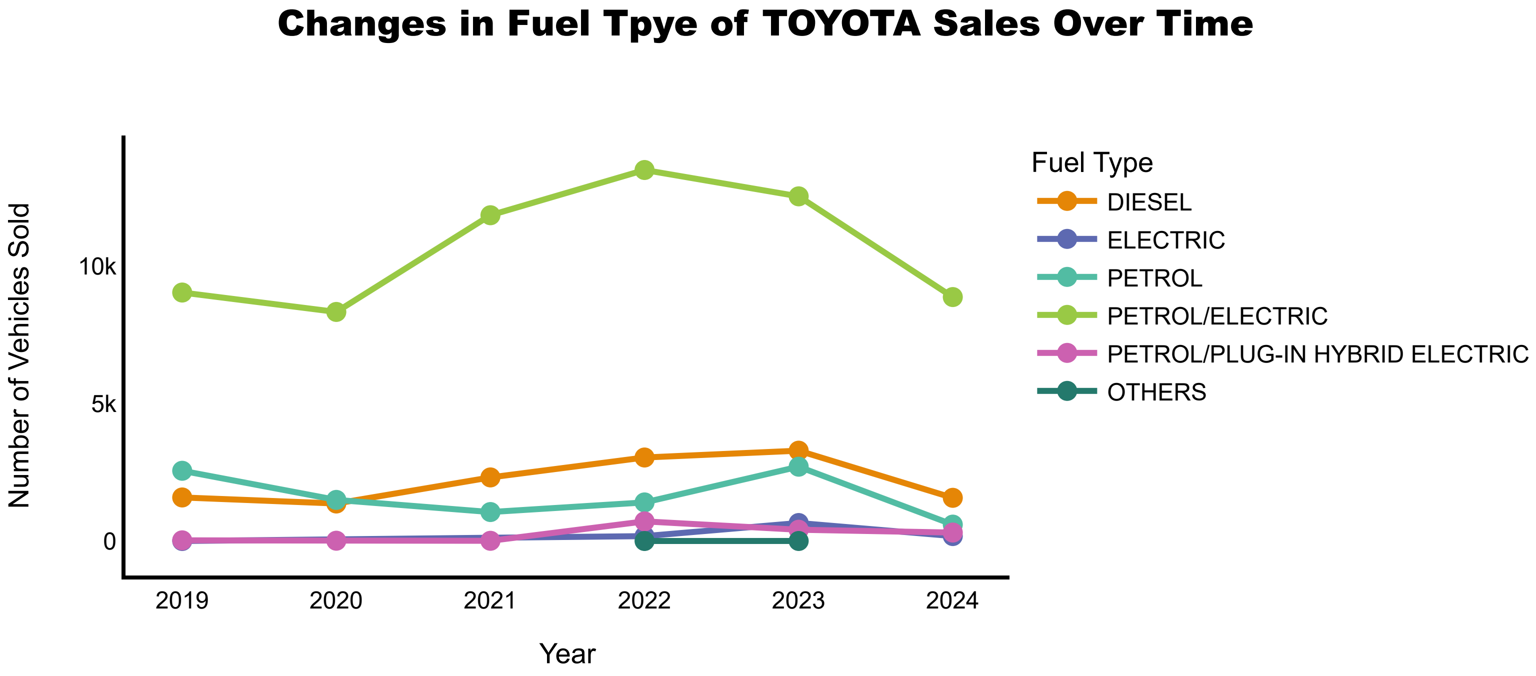
Q3) What are the possible reasons that Toyota has seen an increase in sales over the years?

In [20…

```
toyota_data = df[df['Make'] == 'TOYOTA']
```

In [21…

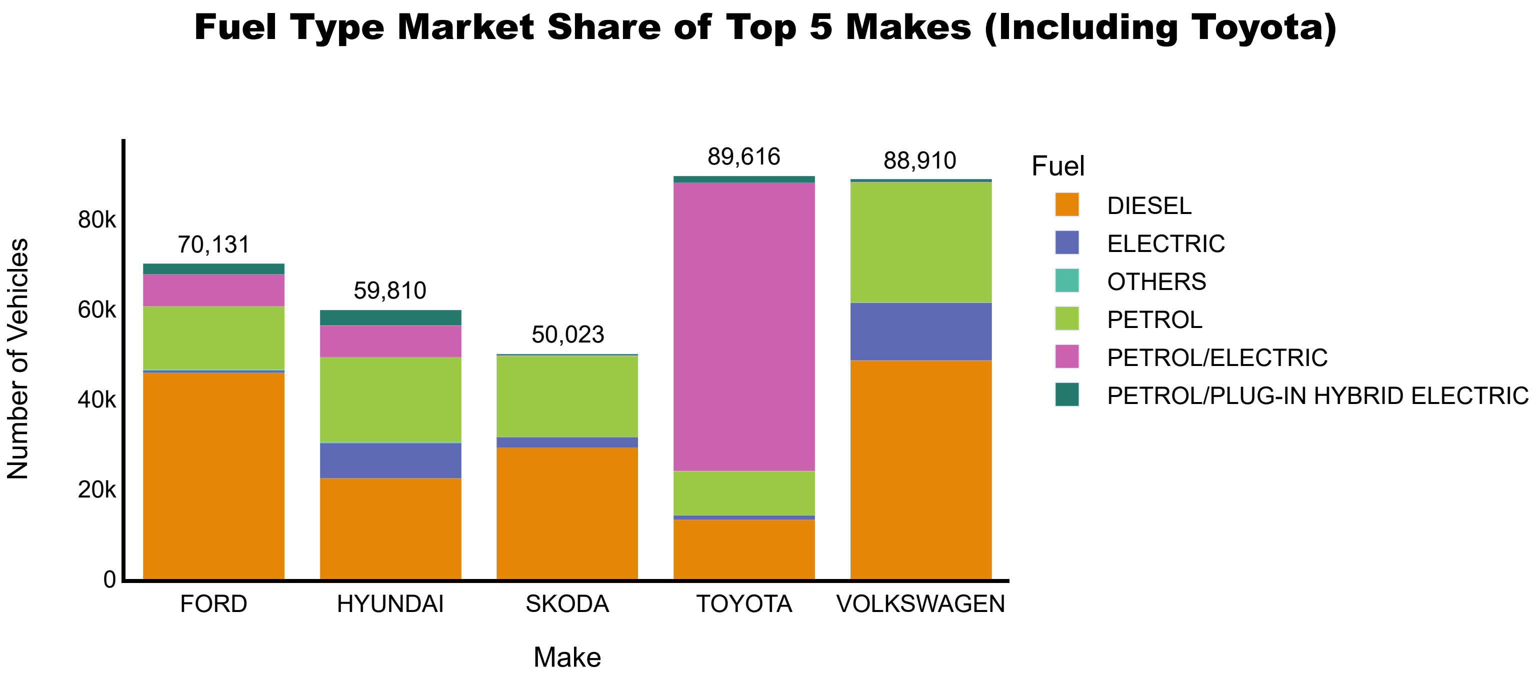
```
# 12. Changes in Fuel Tpye of TOYOTA Sales Over Time with interactive line chart

# Toyota vehicles by year and fuel type
toyota_fuel_trend = toyota_data.groupby(['Year', 'Fuel'])['No. of Vehicles'].sum().reset_index()

fig = px.line(toyota_fuel_trend,
              x='Year',
              y='No. of Vehicles',
              color='Fuel',
              title='Toyota Sales Trend by Fuel Type',
              labels={'No. of Vehicles': 'Number of Vehicles Sold', 'Year': 'Year'},
              markers=True,
              color_discrete_sequence=px.colors.qualitative.Vivid)

fig.update_traces(marker_size=10, line_width=3)

fig.update_layout(
    width=800,
    height=400,
    title=dict(text='Changes in Fuel Tpye of TOYOTA Sales Over Time', font=dict(family="Arial Black", size=18, color='black'), x=0.5, xanchor='center'),
    xaxis_title="Year",
    yaxis_title="Number of Vehicles Sold",

    xaxis=dict(showline=True, linecolor='black', linewidth=2, showgrid=False),
    yaxis=dict(showline=True, linecolor='black', linewidth=2, showgrid=False),

    plot_bgcolor='white',
    font=dict(family="Arial", size=12, color='black'),
    legend_title="Fuel Type"
)

fig.show()
fig.write_html("Changes in Fuel Tpye of TOYOTA Sales Over Time.html")
```



In [22…

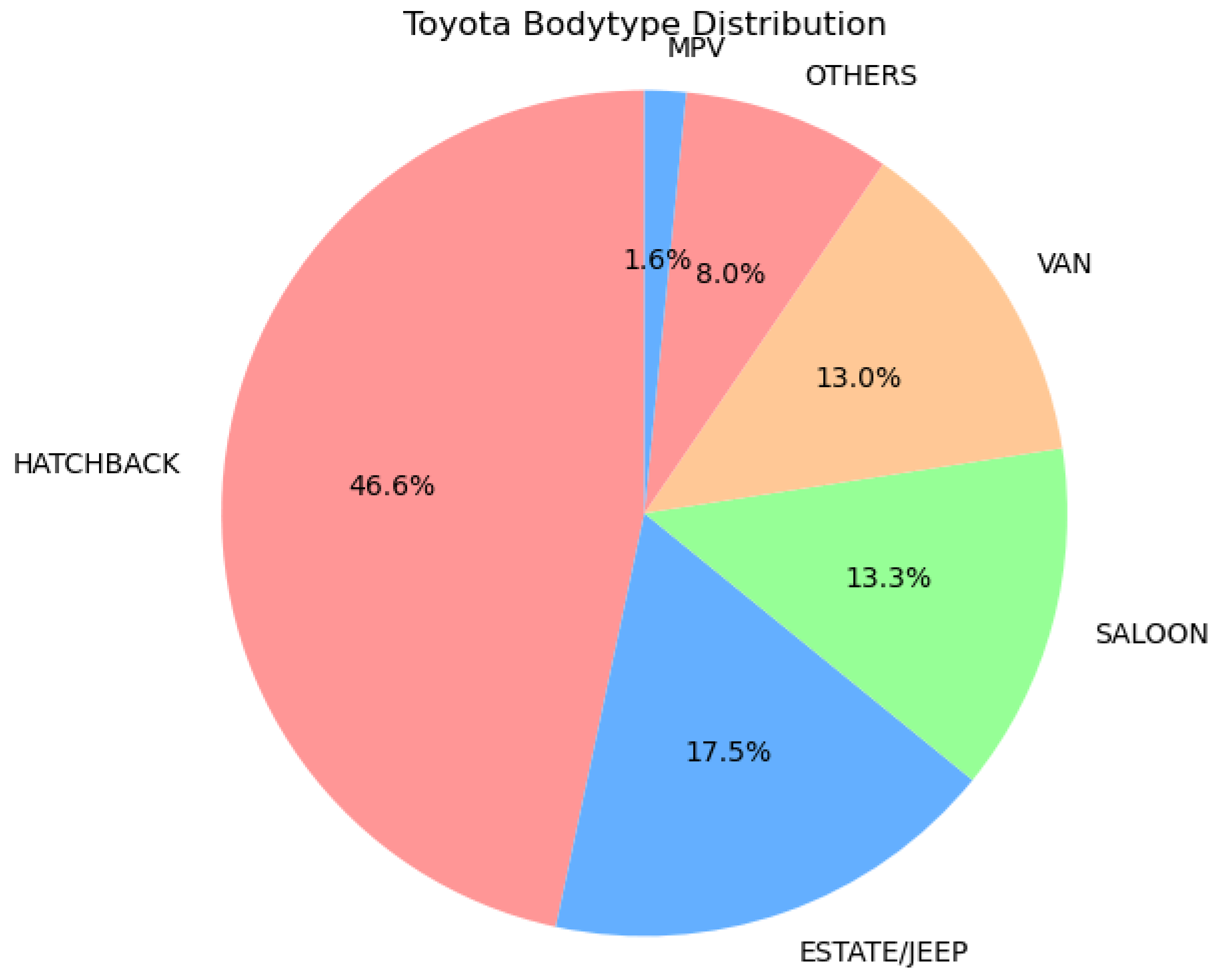
```
# 13. Comparison of Different Fuel Types Among the Top 5 Makes with interactive stacked bar chart

# Get the top five most popular makes
top_make = df['Make'].value_counts().nlargest(5).index

# Filter data for the top five most popular makes
top_make_data = df[df['Make'].isin(top_make)]

# Group by make and fuel type to get the number of vehicles
fuel_share_by_make = top_make_data.groupby(['Make', 'Fuel'])['No. of Vehicles'].sum().reset_index()
```

```
# Calculate total number of vehicles for each 'Make'
make_totals = fuel_share_by_make.groupby('Make')['No. of Vehicles'].sum().reset_index()

# Create an interactive stacked bar chart for the fuel type market share of top 5 makes
fig = px.bar(fuel_share_by_make,
             x='Make',
             y='No. of Vehicles',
             color='Fuel',
             title='Fuel Type Market Share of Top 5 Makes (Including Toyota)',
             labels={'No. of Vehicles': 'Number of Vehicles', 'Make': 'Make'},
             color_discrete_sequence=px.colors.qualitative.Vivid)

fig.update_layout(
    barmode='stack',
    width=800,
    height=400,
    title=dict(font=dict(family="Arial Black", size=18, color='black'), x=0.5, xanchor='center'),

    xaxis=dict(title="Make", showline=True, linecolor='black', linewidth=2, showgrid=False),
    yaxis=dict(title="Number of Vehicles", showline=True, linecolor='black', linewidth=2, showgrid=False),

    plot_bgcolor='white',
    font=dict(family="Arial", size=12, color='black')
)

for i, row in make_totals.iterrows():
    fig.add_annotation(
        x=row['Make'],
        y=row['No. of Vehicles'],
        text=f"{row['No. of Vehicles']:,}",
        showarrow=False,
        font=dict(size=12, color='black'),
        yshift=10
    )

fig.show()
fig.write_html("Fuel Type Market Share of Top 5 Makes (Including Toyota).html")
```



In [23…

```
# Counting the occurrences of each bodytype
bodytype_counts = toyota_data['Bodytype'].value_counts()

# Plotting the pie chart
plt.figure(figsize=(6, 6))
plt.pie(bodytype_counts, labels=bodytype_counts.index, autopct='%1.1f%%', startangle=90, colors=['#ff9999','#66b3ff','#99ff99','#ffcc99'])
plt.title('Toyota Bodytype Distribution')
plt.axis('equal')  # Equal aspect ratio ensures that pie chart is drawn as a circle.

# Show the plot
plt.show()
```


Toyota Bodytype Distribution
MPV
OTHERS
VAN
HATCHBACK
SALOON
ESTATE/JEEP
1.6%
8.0%
13.0%
46.6%
13.3%
17.5%

In [ ]: