Determining What Kind of Car to Import

Importing Datas

Car Sales Data ¶

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

car = pd.read_csv('Car.csv')
car
```

Out[1]:

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	M
0	Hyundai	Accent 2dr hatch	Sedan	Asia	Front	$10,539	$10,107	1.6	4.0	103.0	
1	Toyota	Echo 2dr manual	Sedan	Asia	Front	$10,760	$10,144	1.5	4.0	108.0	
2	Saturn	Ion1 4dr	Sedan	USA	Front	$10,995	$10,319	2.2	4.0	140.0	
3	Toyota	Echo 4dr	Sedan	Asia	Front	$11,290	$10,642	1.5	4.0	108.0	
4	Kia	Rio 4dr auto	Sedan	Asia	Front	$11,155	$10,705	1.6	4.0	104.0	
...	...	...	...	...	...	...	...	...	...	...	
426	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
427	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
428	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
429	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
430	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

431 rows × 15 columns

Ownership Trend Data

In [2]:
```
trend_car = pd.read_csv('cartrend.csv')
trend_car
```

Out[2]:

	year	make	number
0	2005	ALFA ROMEO	914
1	2005	ALPINA	0
2	2005	ASTON MARTIN	23
3	2005	AUDI	2025
4	2005	AUSTIN	137
...	...	...	...
1069	2017	VOLKSWAGEN	27644
1070	2017	VOLVO	10539
1071	2017	WULING	41
1072	2017	ZOTYE	43
1073	2017	OTHERS	81

1074 rows × 3 columns

Data Cleaning

In [3]:
```
#Dropping all null values from data set

car.isnull().sum()
car_drop = car.dropna(inplace = True)
car_clean = car
car_clean
```

Out[3]:

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepov
0	Hyundai	Accent 2dr hatch	Sedan	Asia	Front	$10,539	$10,107	1.6	4.0	10
1	Toyota	Echo 2dr manual	Sedan	Asia	Front	$10,760	$10,144	1.5	4.0	10
2	Saturn	Ion1 4dr	Sedan	USA	Front	$10,995	$10,319	2.2	4.0	14
3	Toyota	Echo 4dr	Sedan	Asia	Front	$11,290	$10,642	1.5	4.0	10
4	Kia	Rio 4dr auto	Sedan	Asia	Front	$11,155	$10,705	1.6	4.0	10
...	...	...	...	...	...	...	...	...	...	
418	Jaguar	XKR convertible 2dr	Sports	Europe	Rear	$86,995	$79,226	4.2	8.0	39
419	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6.0	29
420	Mercedes-Benz	S500 4dr	Sedan	Europe	All	$86,970	$80,939	5.0	8.0	30
421	Mercedes-Benz	SL500 convertible 2dr	Sports	Europe	Rear	$90,520	$84,325	5.0	8.0	30
422	Mercedes-Benz	CL500 2dr	Sedan	Europe	Rear	$94,820	$88,324	5.0	8.0	30

421 rows × 15 columns

In [4]:
```
#Converting MSRP and Invoice into strings containing only numbers
msrp = car_clean['MSRP']
car_clean['MSRP_Value'] = msrp.str[1:3] + msrp.str[4:]
invoice = car_clean['Invoice']
car_clean['Invoice_Value'] = invoice.str[1:3] + invoice.str[4:]
```

In [5]:
```
#Converting into integers
car_clean['MSRP_Value'] = car_clean['MSRP_Value'].astype(int)
car_clean['Invoice_Value'] = car_clean['Invoice_Value'].astype(int)

car_clean
```

Out[5]:

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepov
0	Hyundai	Accent 2dr hatch	Sedan	Asia	Front	$10,539	$10,107	1.6	4.0	10
1	Toyota	Echo 2dr manual	Sedan	Asia	Front	$10,760	$10,144	1.5	4.0	10
2	Saturn	Ion1 4dr	Sedan	USA	Front	$10,995	$10,319	2.2	4.0	14
3	Toyota	Echo 4dr	Sedan	Asia	Front	$11,290	$10,642	1.5	4.0	10
4	Kia	Rio 4dr auto	Sedan	Asia	Front	$11,155	$10,705	1.6	4.0	10
...	...	...	...	...	...	...	...	...	...	
418	Jaguar	XKR convertible 2dr	Sports	Europe	Rear	$86,995	$79,226	4.2	8.0	39
419	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6.0	29
420	Mercedes-Benz	S500 4dr	Sedan	Europe	All	$86,970	$80,939	5.0	8.0	30
421	Mercedes-Benz	SL500 convertible 2dr	Sports	Europe	Rear	$90,520	$84,325	5.0	8.0	30
422	Mercedes-Benz	CL500 2dr	Sedan	Europe	Rear	$94,820	$88,324	5.0	8.0	30

421 rows × 17 columns

In [6]:
```python
#Calculating the price deviations
car_clean['Price Deviation'] = car_clean['Invoice_Value'] - car_clean['MSRP_Value']
car_clean
```

Out[6]:

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepov
0	Hyundai	Accent 2dr hatch	Sedan	Asia	Front	$10,539	$10,107	1.6	4.0	10
1	Toyota	Echo 2dr manual	Sedan	Asia	Front	$10,760	$10,144	1.5	4.0	10
2	Saturn	Ion1 4dr	Sedan	USA	Front	$10,995	$10,319	2.2	4.0	14
3	Toyota	Echo 4dr	Sedan	Asia	Front	$11,290	$10,642	1.5	4.0	10
4	Kia	Rio 4dr auto	Sedan	Asia	Front	$11,155	$10,705	1.6	4.0	10
...	...	...	...	...	...	...	...	...	...	
418	Jaguar	XKR convertible 2dr	Sports	Europe	Rear	$86,995	$79,226	4.2	8.0	39
419	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6.0	29
420	Mercedes-Benz	S500 4dr	Sedan	Europe	All	$86,970	$80,939	5.0	8.0	30
421	Mercedes-Benz	SL500 convertible 2dr	Sports	Europe	Rear	$90,520	$84,325	5.0	8.0	30
422	Mercedes-Benz	CL500 2dr	Sedan	Europe	Rear	$94,820	$88,324	5.0	8.0	30

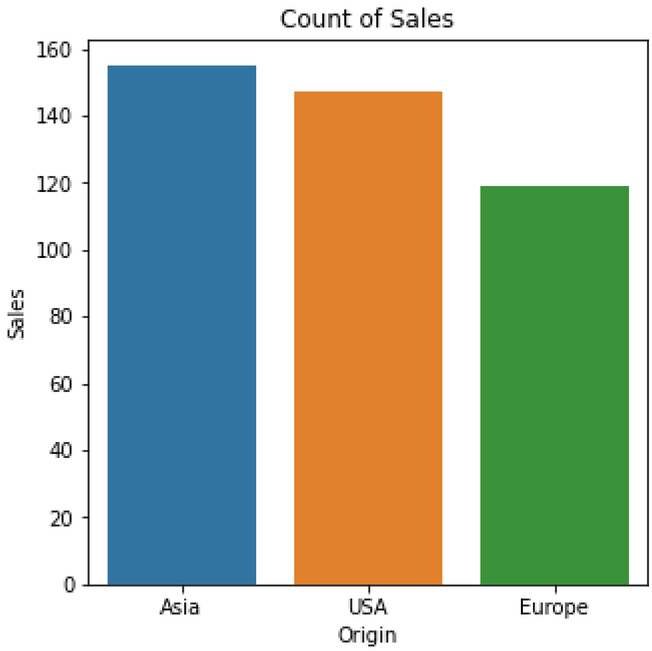
421 rows × 18 columns

Separation into Origins

In [7]:
```python
#Finding the count of cars from each origin
all_origins = car_clean['Origin'].unique()

sales = car_clean['Origin'].value_counts()
sales = sales[all_origins]
print(sales)
```

```
Asia      155
USA       147
Europe    119
Name: Origin, dtype: int64
```

In [8]:
```
#Plotting the number of cars for each origin of car

sns.countplot(car_clean['Origin'])
sns.countplot(car_clean['Origin']).set_xticklabels

fig = plt.gcf()
fig.set_size_inches(5,5)
plt.xlabel('Origin')
plt.ylabel('Sales')
plt.title('Count of Sales')
plt.show()
```



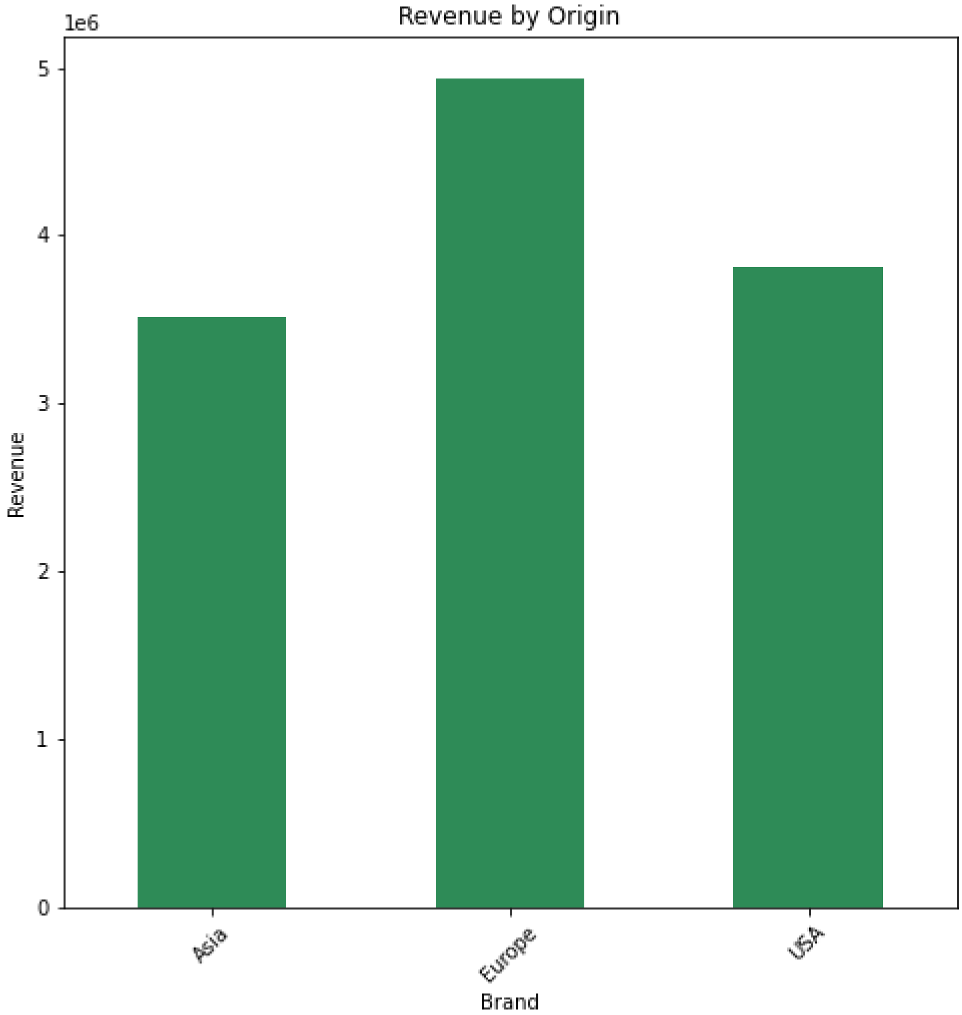
In [9]:
```
#Separating data based on origins
asia = car_clean.loc[car_clean['Origin'] == 'Asia']
usa = car_clean.loc[car_clean['Origin'] == 'USA']
europe = car_clean.loc[car_clean['Origin'] == 'Europe']

print(asia)
print(usa)
print(europe)
```

```
      Make                  Model    Type Origin DriveTrain     MSRP  \
0    Hyundai        Accent 2dr hatch   Sedan   Asia      Front  $10,539
1     Toyota         Echo 2dr manual   Sedan   Asia      Front  $10,760
3     Toyota                Echo 4dr   Sedan   Asia      Front  $11,290
4        Kia            Rio 4dr auto   Sedan   Asia      Front  $11,155
5     Toyota           Echo 2dr auto   Sedan   Asia      Front  $11,560
..       ...                     ...     ...    ...        ...      ...
383   Toyota            Land Cruiser     SUV   Asia        All  $54,765
387    Lexus              LS 430 4dr   Sedan   Asia       Rear  $55,750
395    Lexus  SC 430 convertible 2dr  Sports   Asia       Rear  $63,200
396    Lexus                  LX 470     SUV   Asia        All  $64,800
419    Acura  NSX coupe 2dr manual S  Sports   Asia       Rear  $89,765

     Invoice  EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  \
0    $10,107         1.6        4.0       103.0      29.0         33.0
1    $10,144         1.5        4.0       108.0      35.0         43.0
3    $10,642         1.5        4.0       108.0      35.0         43.0
4    $10,705         1.6        4.0       104.0      25.0         32.0
5    $10,896         1.5        4.0       108.0      33.0         39.0
..       ...         ...        ...         ...       ...          ...
383  $47,986         4.7        8.0       325.0      13.0         17.0
387  $48,583         4.3        8.0       290.0      18.0         25.0
395  $55,063         4.3        8.0       300.0      18.0         23.0
396  $56,455         4.7        8.0       235.0      13.0         17.0
419  $79,978         3.2        6.0       290.0      17.0         24.0

     Weight  Wheelbase  Length  MSRP_Value  Invoice_Value  Price Deviation
0    2255.0       96.0   167.0       10539          10107             -432
1    2035.0       93.0   163.0       10760          10144             -616
3    2055.0       93.0   163.0       11290          10642             -648
4    2458.0       95.0   167.0       11155          10705             -450
5    2085.0       93.0   163.0       11560          10896             -664
..      ...        ...     ...         ...            ...              ...
383  5390.0      112.0   193.0       54765          47986            -6779
387  3990.0      115.0   197.0       55750          48583            -7167
395  3840.0      103.0   178.0       63200          55063            -8137
396  5590.0      112.0   193.0       64800          56455            -8345
419  3153.0      100.0   174.0       89765          79978            -9787

[155 rows x 18 columns]
         Make                      Model    Type Origin DriveTrain  \
2      Saturn                   Ion1 4dr   Sedan    USA      Front
6    Chevrolet                  Aveo 4dr   Sedan    USA      Front
11   Chevrolet        Aveo LS 4dr hatch   Sedan    USA      Front
17        Ford       Focus ZX3 2dr hatch   Sedan    USA      Front
21       Dodge               Neon SE 4dr   Sedan    USA      Front
..        ...                        ...     ...    ...        ...
379   Cadillac          Deville DTS 4dr   Sedan    USA      Front
384   Cadillac                 Escalade     SUV    USA      Front
386   Cadillac             Escalade EXT   Truck    USA        All
412   Cadillac       XLR convertible 2dr  Sports    USA       Rear
415      Dodge  Viper SRT-10 convertible 2dr  Sports    USA       Rear

        MSRP  Invoice  EngineSize  Cylinders  Horsepower  MPG_City  \
2    $10,995  $10,319         2.2        4.0       140.0      26.0
6    $11,690  $10,965         1.6        4.0       103.0      28.0
11   $12,585  $11,802         1.6        4.0       103.0      28.0
17   $13,270  $12,482         2.0        4.0       130.0      26.0
21   $13,670  $12,849         2.0        4.0       132.0      29.0
..       ...      ...         ...        ...         ...       ...
379  $50,595  $46,362         4.6        8.0       300.0      18.0
384  $52,795  $48,377         5.3        8.0       295.0      14.0
386  $52,975  $48,541         6.0        8.0       345.0      13.0
412  $76,200  $70,546         4.6        8.0       320.0      17.0
415  $81,795  $74,451         8.3       10.0       500.0      12.0
```

```
     MPG_Highway  Weight  Wheelbase  Length  MSRP_Value  Invoice_Value  \
2           35.0  2692.0      103.0   185.0       10995          10319
6           34.0  2370.0       98.0   167.0       11690          10965
11          34.0  2348.0       98.0   153.0       12585          11802
17          33.0  2612.0      103.0   168.0       13270          12482
21          36.0  2581.0      105.0   174.0       13670          12849
..           ...     ...        ...     ...         ...            ...
379         26.0  4044.0      115.0   207.0       50595          46362
384         18.0  5367.0      116.0   199.0       52795          48377
386         17.0  5879.0      130.0   221.0       52975          48541
412         25.0  3647.0      106.0   178.0       76200          70546
415         20.0  3410.0       99.0   176.0       81795          74451

     Price Deviation
2               -676
6               -725
11              -783
17              -788
21              -821
..               ...
379            -4233
384            -4418
386            -4434
412            -5654
415            -7344

[147 rows x 18 columns]
              Make                  Model    Type  Origin DriveTrain     MSRP  \
54            MINI                 Cooper   Sedan  Europe      Front  $16,999
74      Volkswagen               Jetta GL   Wagon  Europe      Front  $19,005
76      Volkswagen           Golf GLS 4dr   Sedan  Europe      Front  $18,715
92      Volkswagen     GTI 1.8T 2dr hatch   Sedan  Europe      Front  $19,825
93            MINI               Cooper S   Sedan  Europe      Front  $19,999
..             ...                    ...     ...     ...        ...      ...
417           Audi               RS 6 4dr  Sports  Europe      Front  $84,600
418         Jaguar    XKR convertible 2dr  Sports  Europe       Rear  $86,995
420  Mercedes-Benz               S500 4dr   Sedan  Europe        All  $86,970
421  Mercedes-Benz  SL500 convertible 2dr  Sports  Europe       Rear  $90,520
422  Mercedes-Benz             CL500 2dr   Sedan  Europe       Rear  $94,820

     Invoice  EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  \
54   $15,437         1.6        4.0       115.0      28.0         37.0
74   $17,427         2.0        4.0       115.0      24.0         30.0
76   $17,478         2.0        4.0       115.0      24.0         31.0
92   $18,109         1.8        4.0       180.0      24.0         31.0
93   $18,137         1.6        4.0       163.0      25.0         34.0
..       ...         ...        ...         ...       ...          ...
417  $76,417         4.2        8.0       450.0      15.0         22.0
418  $79,226         4.2        8.0       390.0      16.0         23.0
420  $80,939         5.0        8.0       302.0      16.0         24.0
421  $84,325         5.0        8.0       302.0      16.0         23.0
422  $88,324         5.0        8.0       302.0      16.0         24.0

     Weight  Wheelbase  Length  MSRP_Value  Invoice_Value  Price Deviation
54   2524.0       97.0   143.0       16999          15437            -1562
74   3034.0       99.0   174.0       19005          17427            -1578
76   2897.0       99.0   165.0       18715          17478            -1237
92   2934.0       99.0   168.0       19825          18109            -1716
93   2678.0       97.0   144.0       19999          18137            -1862
..      ...        ...     ...         ...            ...              ...
417  4024.0      109.0   191.0       84600          76417            -8183
418  4042.0      102.0   187.0       86995          79226            -7769
420  4390.0      122.0   203.0       86970          80939            -6031
421  4065.0      101.0   179.0       90520          84325            -6195
422  4085.0      114.0   196.0       94820          88324            -6496
```

```
[119 rows x 18 columns]
```

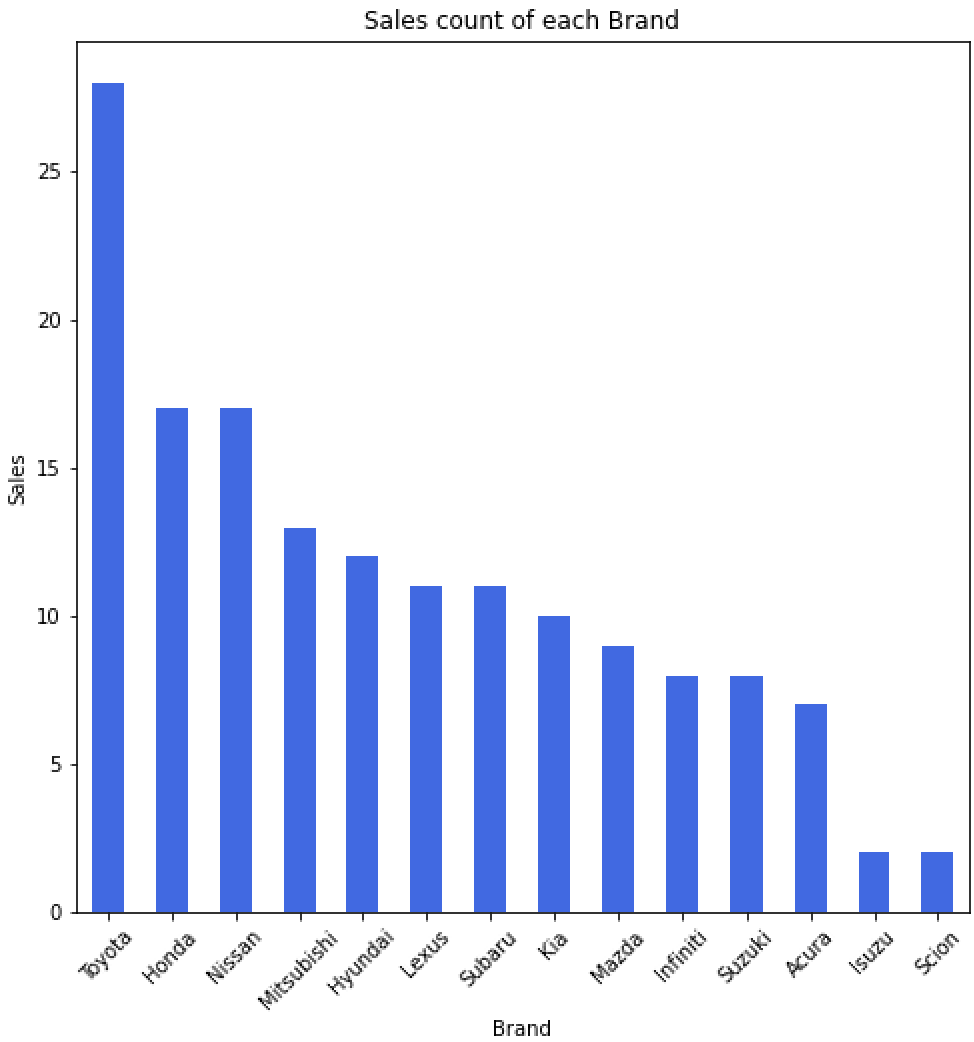
In [10]:
```
rev_by_origin = car_clean.groupby('Origin')['Invoice_Value'].sum()

#Plotting total revenue from each origin
plt.figure(figsize=(8,8))
rev_by_origin.plot.bar(color='seagreen')
plt.title('Revenue by Origin')
plt.xlabel('Brand')
plt.xticks(rotation=45)
plt.ylabel('Revenue')
plt.show()

print(rev_by_origin)
```




```
Origin
Asia      3512296
Europe    4936193
USA       3814553
Name: Invoice_Value, dtype: int64
```

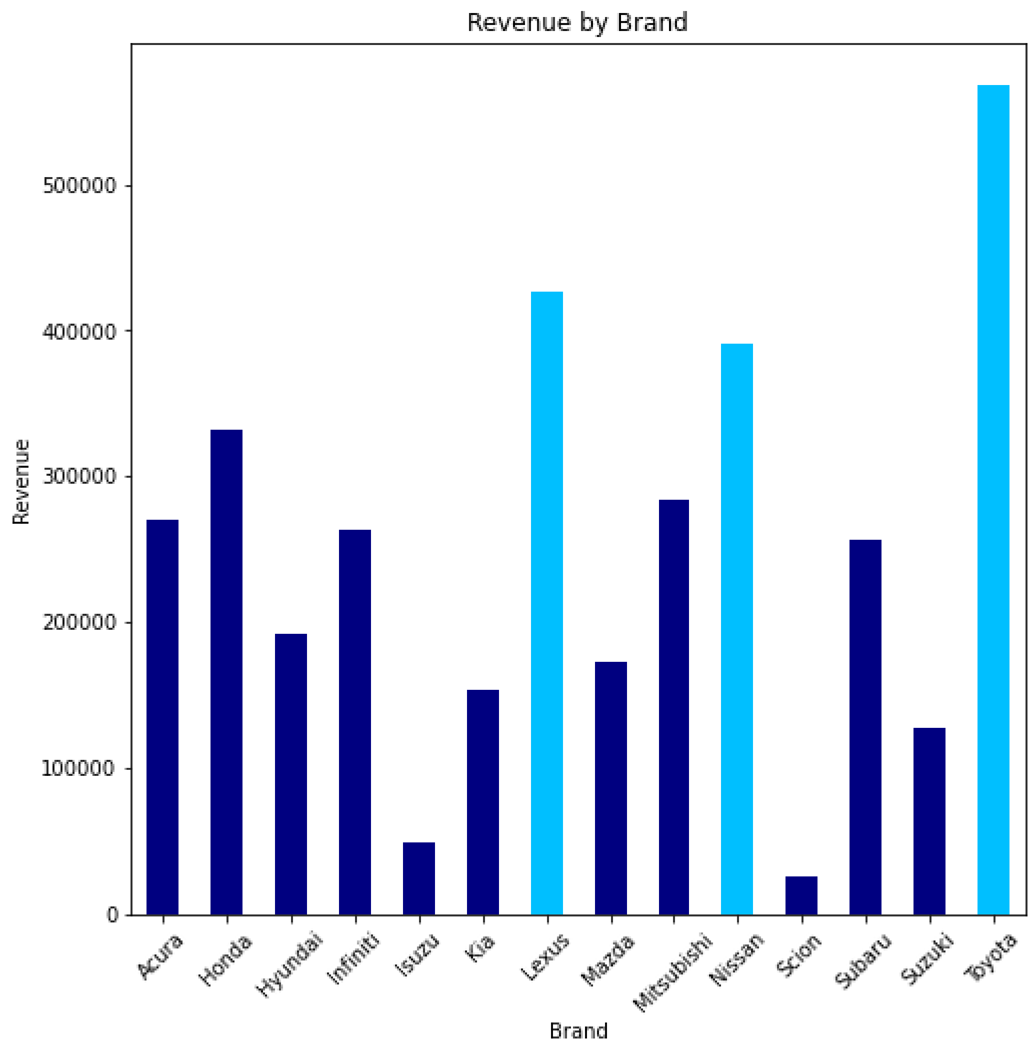
Analysis for Asian Cars

In [11]:
```
#Counts for each make in Asian Cars
count_make = asia['Make'].value_counts().sort_values(ascending=False)

plt.figure(figsize=(8,8))
count_make.plot.bar(color='royalblue')
plt.xlabel('Brand')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.title('Sales count of each Brand')
plt.show()
```



In [12]:
```
#Summing the revenue gotten by the sales of each make of car
rev_by_make = asia.groupby('Make')['Invoice_Value'].sum()

print(rev_by_make)

#Plotting the Revenue of each Make of car
plt.figure(figsize=(8,8))
ax=rev_by_make.plot.bar()

#Countries to highlight
Top_3 = ['Toyota','Lexus','Nissan']

for ticks in ax.xaxis.get_major_ticks():
    if ticks.label1.get_text() in Top_3:
        ax.patches[rev_by_make.index.get_indexer([ticks.label1.get_text()])[0]].set_f
acecolor('deepskyblue')

    else:
        ax.patches[rev_by_make.index.get_indexer([ticks.label1.get_text()])[0]].set_f
acecolor('navy')


plt.title('Revenue by Brand')
plt.xlabel('Brand')
plt.xticks(rotation=45)
plt.ylabel('Revenue')
plt.show()
```

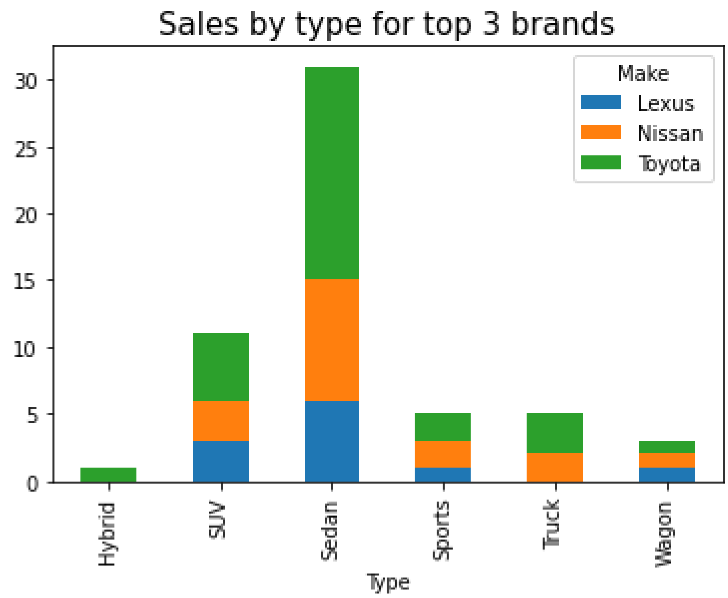
```
Make
Acura          270136
Honda          331717
Hyundai        192424
Infiniti       263040
Isuzu           49238
Kia            153919
Lexus          426360
Mazda          173144
Mitsubishi     283852
Nissan         390957
Scion           25820
Subaru         256273
Suzuki         127130
Toyota         568286
Name: Invoice_Value, dtype: int64
```



In [13]:

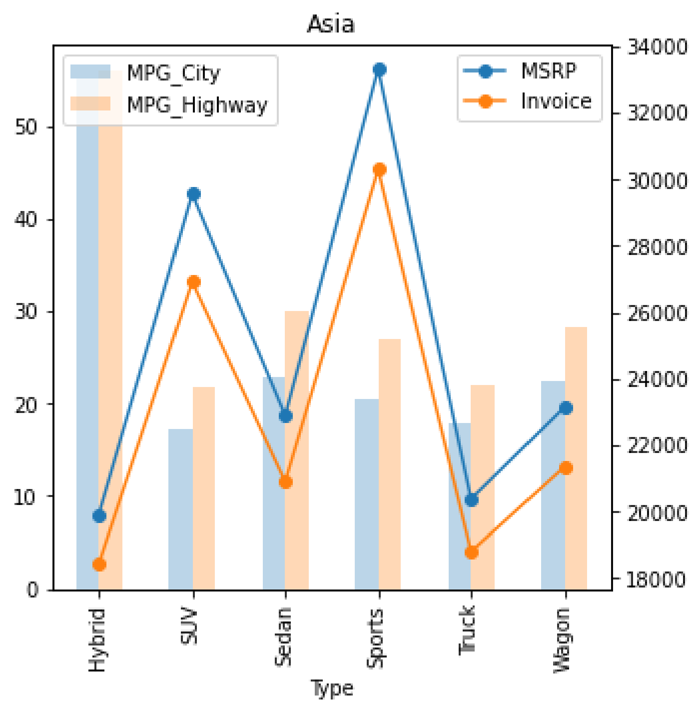
```
#Determining the demand for each type for top 3 brands
is_topa = asia['Make'].isin(['Toyota','Lexus','Nissan'])
topa = asia[is_topa]

plt.figure()
topa_bytype = topa.groupby(['Make','Type'])['Type'].count().unstack('Make')
topa_bytype.plot.bar(stacked=True)
plt.title('Sales by type for top 3 brands', size=15);
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



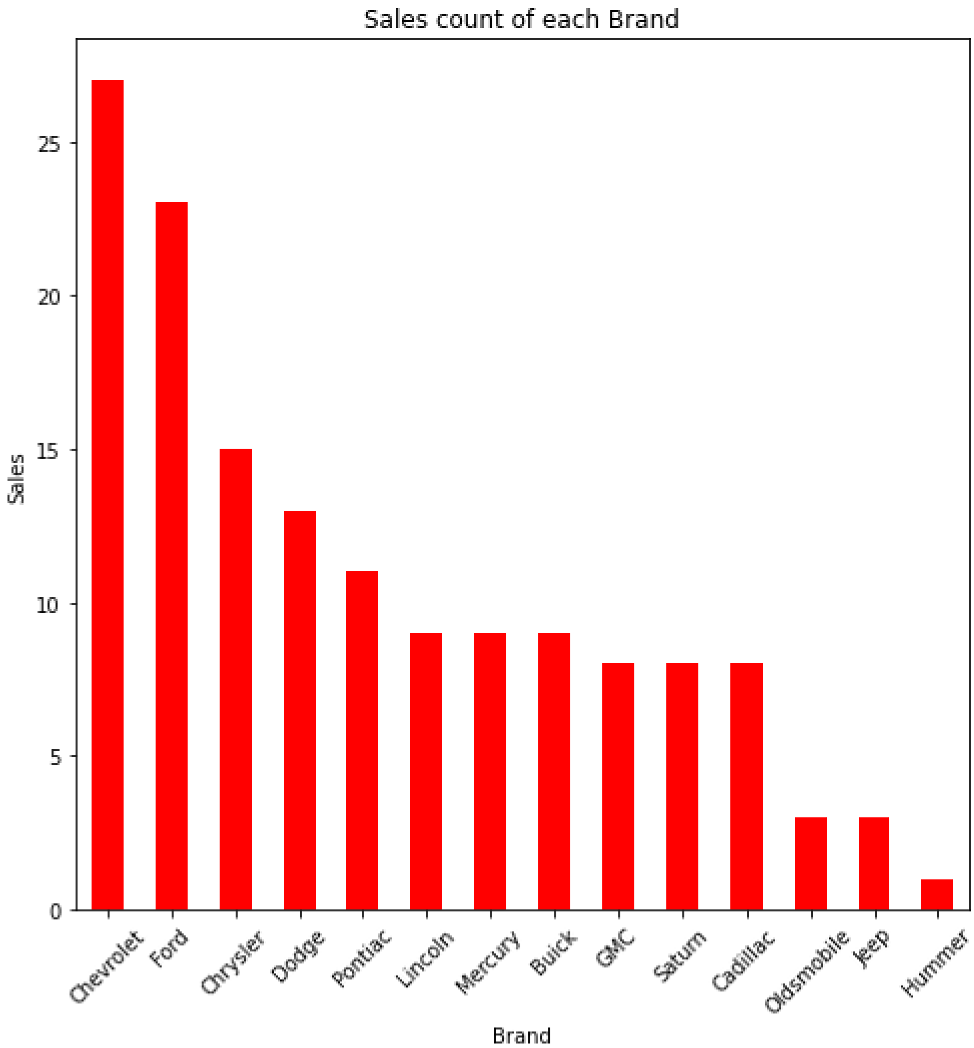
In [14]:

```
#Plotting Invoice and MSRP to find the price deviation for each type
ax = asia.groupby('Type')[['MPG_City', 'MPG_Highway']].mean().plot.bar(alpha=0.3, fig
size=(5,5))
plt.legend(loc='upper left')
ax.twinx().plot(asia.groupby('Type')[['MSRP_Value', 'Invoice_Value']].mean(), label=
'Invoice', marker='o')
plt.legend(['MSRP', 'Invoice'], loc='best');
plt.title('Asia')
plt.show()
```



Analysis for USA Cars

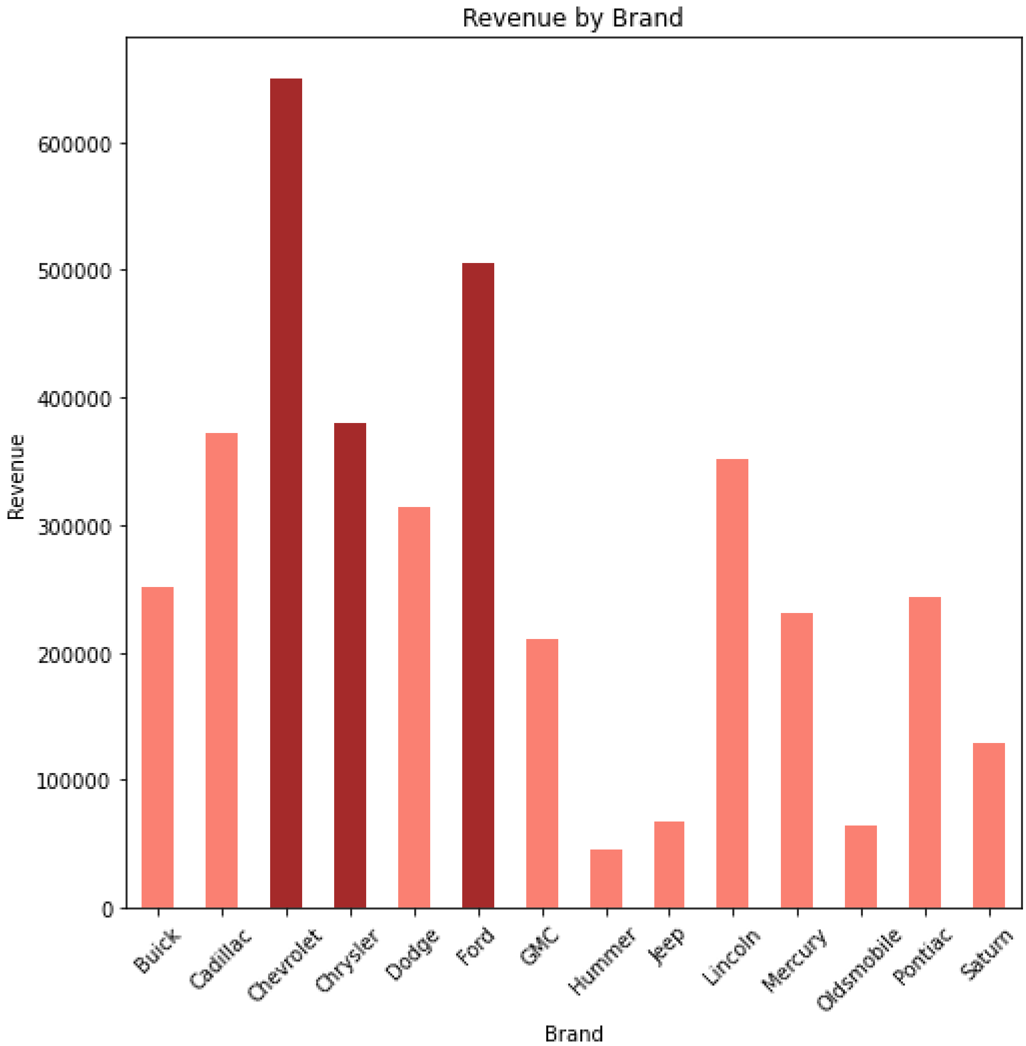
In [15]:
```
#Counts for each make in USA Cars
count_make = usa['Make'].value_counts().sort_values(ascending=False)

plt.figure(figsize=(8,8))
count_make.plot.bar(color='r')
plt.xlabel('Brand')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.title('Sales count of each Brand')
plt.show()
```



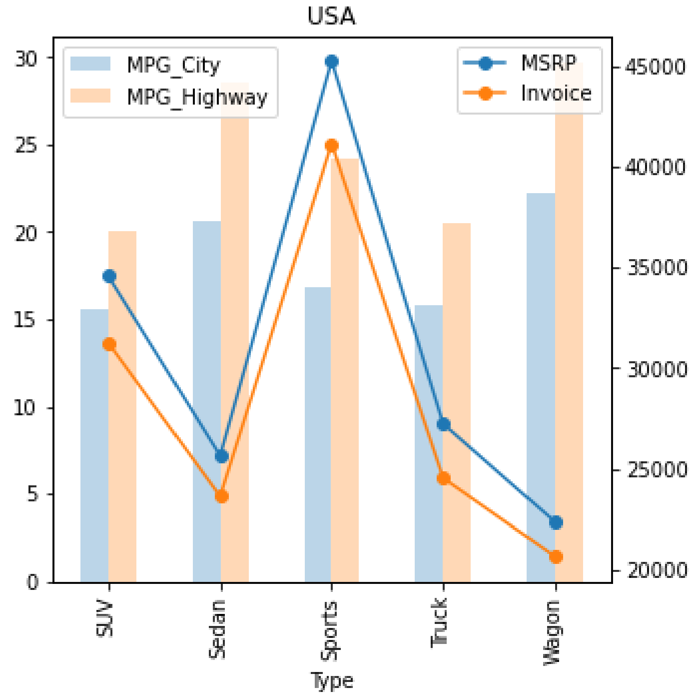
In [16]:

```
#Summing the revenue gotten by the sales of each make of car
rev_by_make = usa.groupby('Make')['Invoice_Value'].sum()

print(rev_by_make)

#Plotting the Revenue of each Make of car
plt.figure(figsize=(8,8))
ax=rev_by_make.plot.bar()

#Countries to highlight
Top_3 = ['Chevrolet','Ford','Chrysler']

for ticks in ax.xaxis.get_major_ticks():
    if ticks.label1.get_text() in Top_3:
        ax.patches[rev_by_make.index.get_indexer([ticks.label1.get_text()])[0]].set_f
acecolor('brown')

    else:
        ax.patches[rev_by_make.index.get_indexer([ticks.label1.get_text()])[0]].set_f
acecolor('salmon')


plt.title('Revenue by Brand')
plt.xlabel('Brand')
plt.xticks(rotation=45)
plt.ylabel('Revenue')
plt.show()
```

```
Make
Buick          250694
Cadillac       371415
Chevrolet      649642
Chrysler       379051
Dodge          314081
Ford           504919
GMC            210315
Hummer          45815
Jeep            67934
Lincoln        352222
Mercury        230918
Oldsmobile      65247
Pontiac        243756
Saturn         128544
Name: Invoice_Value, dtype: int64
```

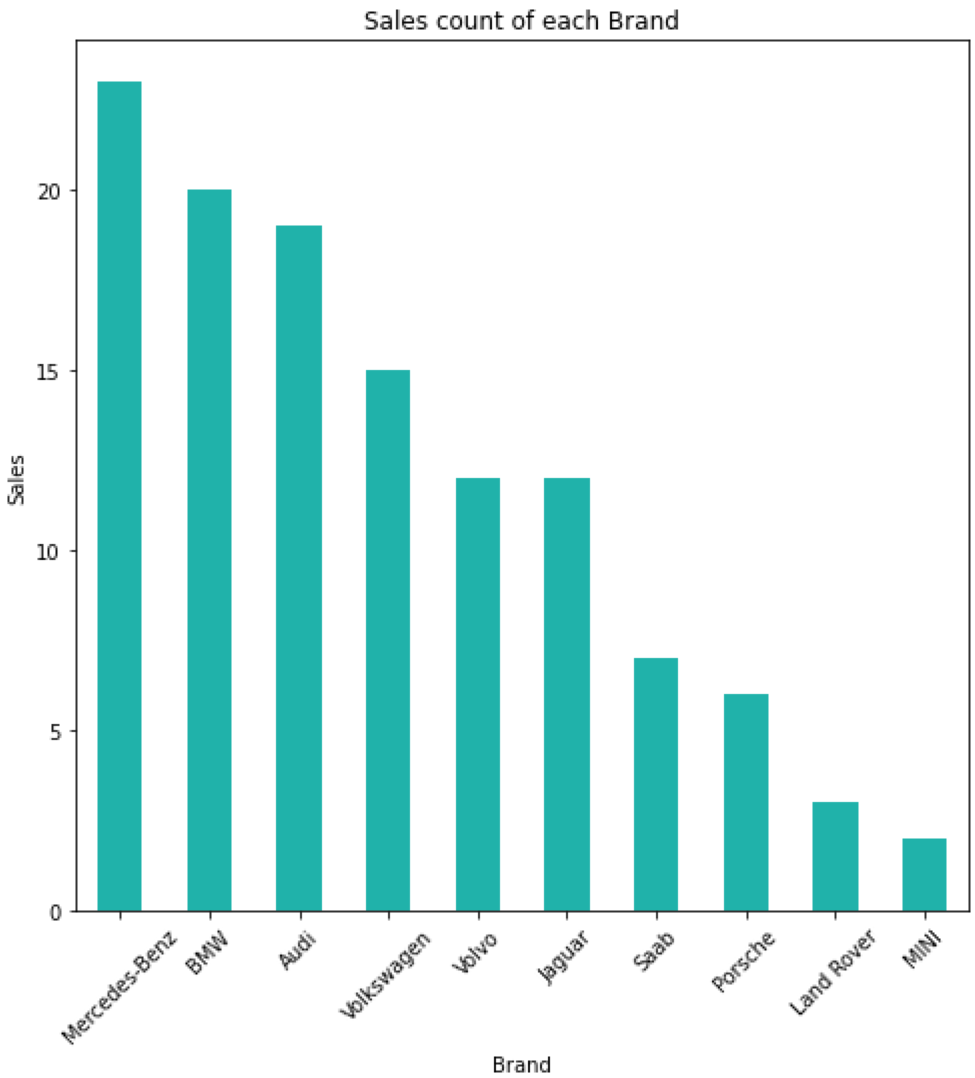


In [17]:
```
#Plotting Invoice and MSRP to find the price deviation for each type
ax = usa.groupby('Type')[['MPG_City', 'MPG_Highway']].mean().plot.bar(alpha=0.3, figs
ize=(5,5))
plt.legend(loc='upper left')
ax.twinx().plot(usa.groupby('Type')[['MSRP_Value', 'Invoice_Value']].mean(), label='I
nvoice', marker='o')
plt.legend(['MSRP', 'Invoice'], loc='best');
plt.title('USA')
plt.show()
```



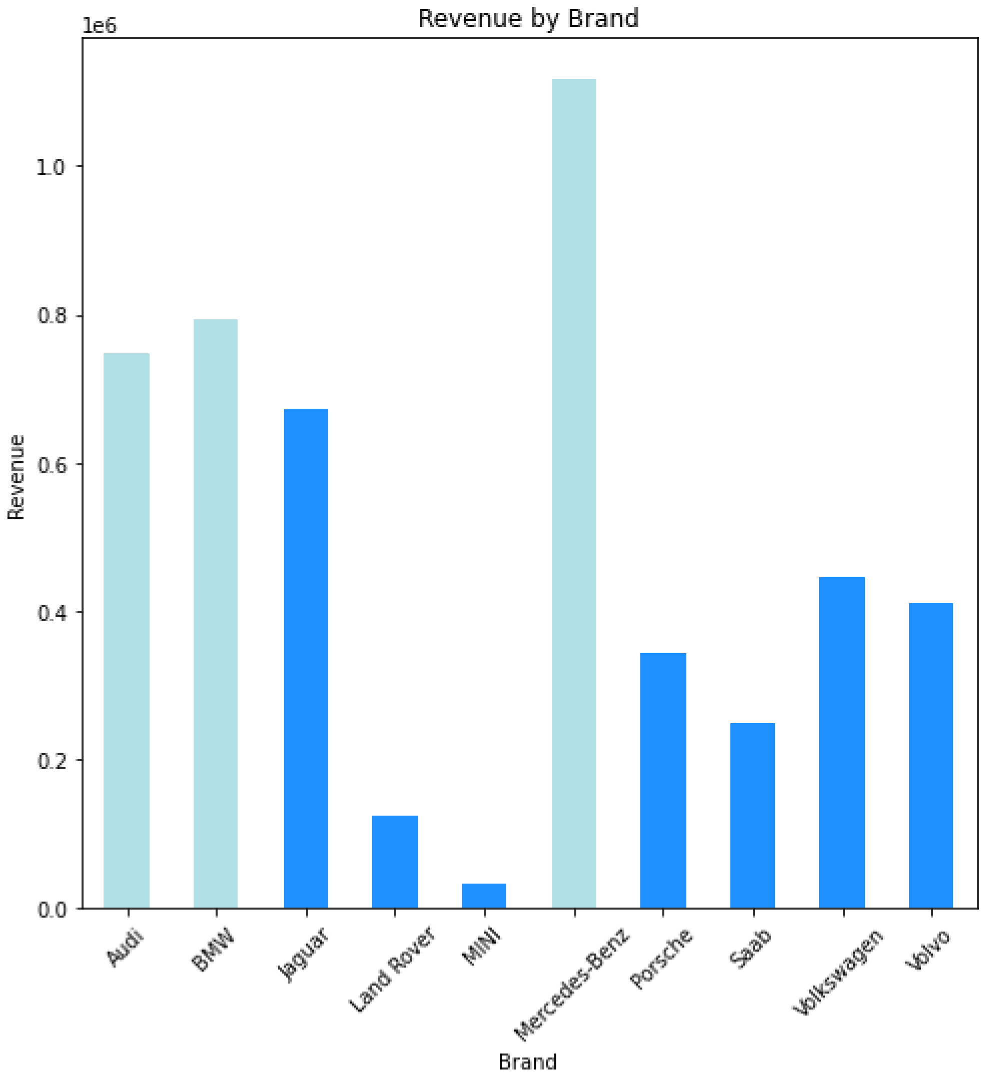
Analysis for Europe Cars

In [18]:
```
#Counts for each make in Europe Cars
count_make = europe['Make'].value_counts()

plt.figure(figsize=(8,8))
count_make.plot.bar(color='lightseagreen')
plt.xlabel('Brand')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.title('Sales count of each Brand')
plt.show()
```

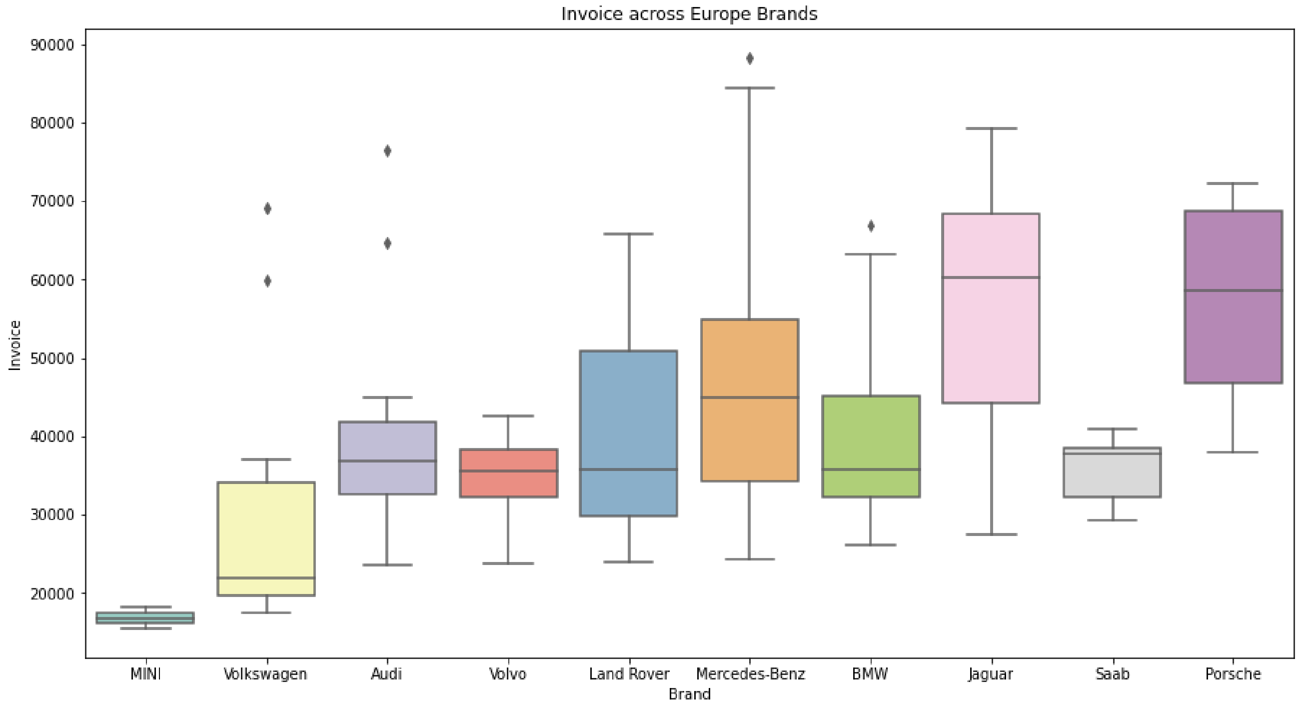


In [19]:
```
#Summing the revenue gotten by the sales of each make of car
rev_by_make = europe.groupby('Make')['Invoice_Value'].sum()

print(rev_by_make)

#Plotting the Revenue of each Make of car
plt.figure(figsize=(8,8))
ax=rev_by_make.plot.bar()

#Countries to highlight
Top_3 = ['Mercedes-Benz','BMW','Audi']

for ticks in ax.xaxis.get_major_ticks():
    if ticks.label1.get_text() in Top_3:
        ax.patches[rev_by_make.index.get_indexer([ticks.label1.get_text()])[0]].set_f
acecolor('powderblue')

    else:
        ax.patches[rev_by_make.index.get_indexer([ticks.label1.get_text()])[0]].set_f
acecolor('dodgerblue')


plt.title('Revenue by Brand')
plt.xlabel('Brand')
plt.xticks(rotation=45)
plt.ylabel('Revenue')
plt.show()
```

```
Make
Audi              747272
BMW               792413
Jaguar            673181
Land Rover        125553
MINI               33574
Mercedes-Benz    1116944
Porsche           342080
Saab              249342
Volkswagen        445240
Volvo             410594
Name: Invoice_Value, dtype: int64
```



In [20]:
```
#Comparing invoice values across brands
plt.figure(figsize=(15,8))
ax = sns.boxplot(europe['Make'], europe['Invoice_Value'], palette='Set3')
plt.xlabel('Brand')
plt.ylabel('Invoice')
plt.title('Invoice across Europe Brands')
plt.show()
```



Analysis of Consumer Ownership

In [21]:
```
# trend of top brands

trend_car[['year']] = trend_car[['year']].astype(int)
trend_car
```

Out[21]:

	year	make	number
0	2005	ALFA ROMEO	914
1	2005	ALPINA	0
2	2005	ASTON MARTIN	23
3	2005	AUDI	2025
4	2005	AUSTIN	137
...	...	...	...
1069	2017	VOLKSWAGEN	27644
1070	2017	VOLVO	10539
1071	2017	WULING	41
1072	2017	ZOTYE	43
1073	2017	OTHERS	81

1074 rows × 3 columns

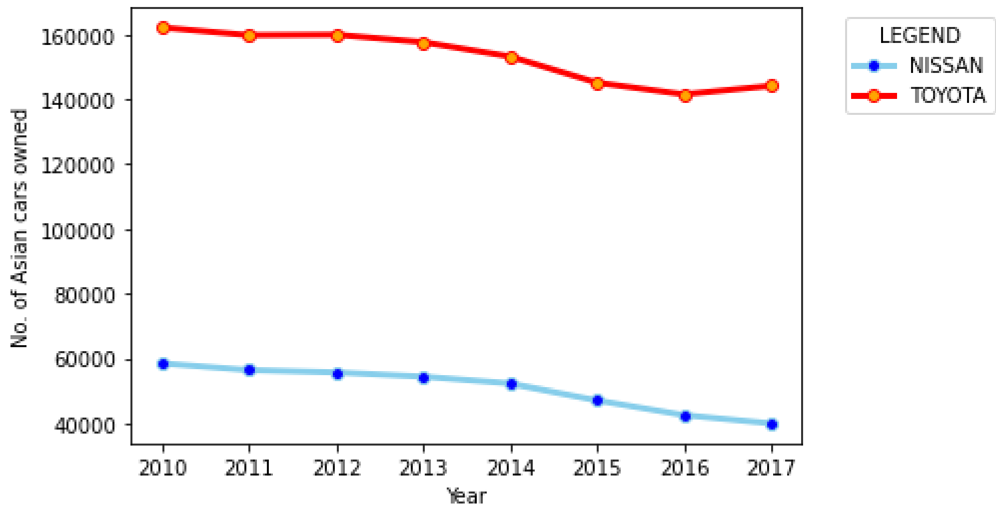
In [22]:
```
trend_car_2010 = trend_car['year'] >= 2010
cars = trend_car.loc[trend_car_2010]
cars
```

Out[22]:

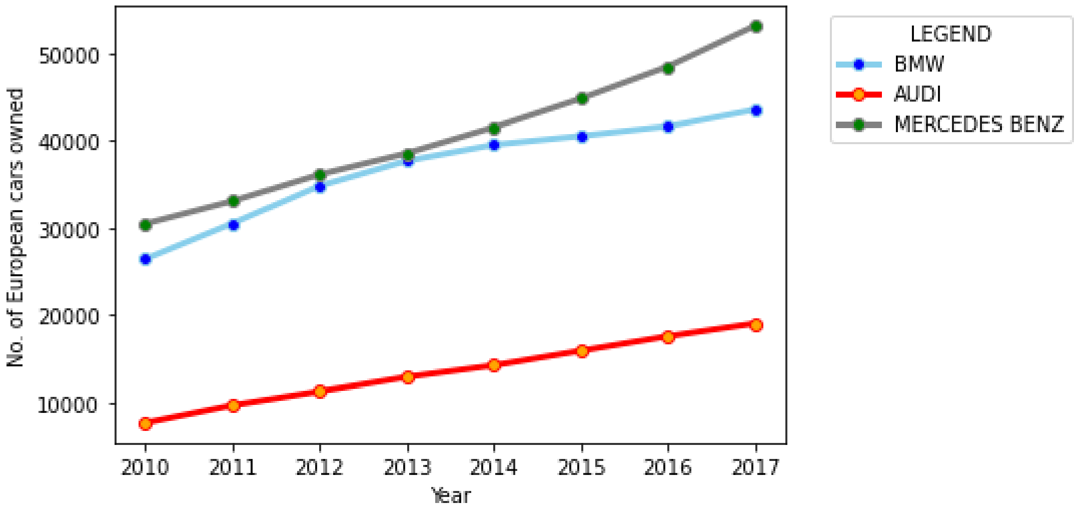
	year	make	number
415	2010	ALFA ROMEO	924
416	2010	ALPINA	0
417	2010	ASTON MARTIN	134
418	2010	AUDI	7645
419	2010	AUSTIN	152
...	...	...	...
1069	2017	VOLKSWAGEN	27644
1070	2017	VOLVO	10539
1071	2017	WULING	41
1072	2017	ZOTYE	43
1073	2017	OTHERS	81

659 rows × 3 columns

In [23]:
```
#asian cars trend: Nissan, Toyota and Toyota(Lexus)

nissan_top_trend = cars['make'] == 'NISSAN'
toyota_top_trend = cars['make'] == 'TOYOTA'
nissan_trend = cars.loc[nissan_top_trend]
toyota_trend = cars.loc[toyota_top_trend]

plt.plot(nissan_trend['year'], nissan_trend['number'],marker='.', markerfacecolor='bl
ue', markersize=12, color='skyblue', linewidth=3, label = 'NISSAN')
plt.plot(toyota_trend['year'], toyota_trend['number'],marker='.', markerfacecolor='or
ange', markersize=12, color='red', linewidth=3, label = 'TOYOTA')
plt.xlabel('Year')
plt.ylabel('No. of Asian cars owned')
plt.legend(title='LEGEND', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [24]:
```
#continental cars trend: BMW, Mercedes Benz, Audi

bmw_top_trend = cars['make'] == 'B.M.W.'
audi_top_trend = cars['make'] == 'AUDI'
mbenz_top_trend = cars['make'] == 'MERCEDES BENZ'
bmw_trend = cars.loc[bmw_top_trend]
audi_trend = cars.loc[audi_top_trend]
mbenz_trend = cars.loc[mbenz_top_trend]


plt.plot(bmw_trend['year'], bmw_trend['number'], marker='.', markerfacecolor='blue',
markersize=12, color='skyblue', linewidth=3, label = 'BMW')
plt.plot(audi_trend['year'], audi_trend['number'],marker='.', markerfacecolor='orang
e', markersize=12, color='red', linewidth=3, label = 'AUDI')
plt.plot(mbenz_trend['year'], mbenz_trend['number'],marker='.', markerfacecolor='gree
n', markersize=12, color='grey', linewidth=3, label = 'MERCEDES BENZ')
plt.xlabel('Year')
plt.ylabel('No. of European cars owned')
plt.legend(title='LEGEND', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [25]:
```
#American cars trend: Chevrolet, Ford and Chrysler

chevy_top_trend = cars['make'] == 'CHEVROLET'
ford_top_trend = cars['make'] == 'FORD'
chrysler_top_trend = cars['make'] == 'CHRYSLER'
chevy_trend = cars.loc[chevy_top_trend]
ford_trend = cars.loc[ford_top_trend]
chrysler_trend = cars.loc[chrysler_top_trend]


plt.plot(chevy_trend['year'], chevy_trend['number'],marker='.', markerfacecolor='blu
e', markersize=12, color='skyblue', linewidth=3, label = 'CHEVROLET')
plt.plot(ford_trend['year'], ford_trend['number'],marker='.', markerfacecolor='orang
e', markersize=12, color='red', linewidth=3, label = 'FORD')
plt.plot(chrysler_trend['year'], chrysler_trend['number'],marker='.', markerfacecolor
='green', markersize=12, color='grey', linewidth=3, label = 'CHRYSLER')
plt.xlabel('Year')
plt.ylabel('No. of American cars owned')
plt.legend(title='LEGEND', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```