# Financial Hedging Data Analysis and Visualization

The concept of hedging strategies.

Monthly Mortgage & Owner Costs: The sum of mortgage payments, home equity loans, utilities, and property taxes.

Monthly Owner Costs: The sum of utilities and property taxes.

Gross Rent: Contract rent plus the estimated average monthly cost of utilities.

Household Income: The sum of the householder and all other individuals aged 15 years and older who reside in the household.

Family Income: The sum of the incomes of all members aged 15 years and older who are related to the householder.

In [19]:
```
#Load the librarys
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams

import plotly.tools as tls
import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot, plot
import plotly.graph_objs as go
init_notebook_mode(connected=True)
import warnings

from scipy import stats

%matplotlib inline

# figure size in inches
rcParams['figure.figsize'] = 12,6
```

## Read The Dataset

In [20]:
```
df_features = pd.read_csv("../input/real_estate_db.csv", encoding='ISO-8859-1' )

del df_features['BLOCKID']
del df_features['UID']
```

In [21]:
```
def resumetable(df):
    print(f"Dataset Shape: {df.shape}")
    summary = pd.DataFrame(df.dtypes,columns=['dtypes'])
    summary = summary.reset_index()
    summary['Name'] = summary['index']
    summary = summary[['Name','dtypes']]
    summary['Missing'] = df.isnull().sum().values
    summary['Uniques'] = df.nunique().values
    summary['First Value'] = df.loc[0].values
    summary['Second Value'] = df.loc[1].values
    summary['Third Value'] = df.loc[2].values

    for name in summary['Name'].value_counts().index:
        summary.loc[summary['Name'] == name, 'Entropy'] = round(stats.entropy(df[name].value_counts(normalize=True), base=2),2)

    return summary
```

# Resume the data

## First part of features

In [22]:
```
#Looking the shape of data
resumetable(df_features)[:43]
```

```
Dataset Shape: (39030, 78)
```

Out[22]:

| | Name | dtypes | Missing | Uniques | First Value | Second Value | Third Value | Entropy |
|---|---|---|---|---|---|---|---|---|
| **0** | SUMLEVEL | int64 | 0 | 1 | 140 | 140 | 140 | 0.00 |
| **1** | COUNTYID | int64 | 0 | 310 | 16 | 20 | 20 | 6.58 |
| **2** | STATEID | int64 | 0 | 52 | 2 | 2 | 2 | 5.10 |
| **3** | state | object | 0 | 52 | Alaska | Alaska | Alaska | 5.10 |
| **4** | state_ab | object | 0 | 52 | AK | AK | AK | 5.10 |
| **5** | city | object | 0 | 8172 | Unalaska | Eagle River | Jber | 11.50 |
| **6** | place | object | 0 | 11856 | Unalaska City | Anchorage | Anchorage | 12.53 |
| **7** | type | object | 0 | 6 | City | City | City | 1.89 |
| **8** | primary | object | 0 | 1 | tract | tract | tract | 0.00 |
| **9** | zip_code | int64 | 0 | 15098 | 99685 | 99577 | 99505 | 13.47 |
| **10** | area_code | int64 | 0 | 275 | 907 | 907 | 907 | 7.83 |
| **11** | lat | float64 | 0 | 38710 | 53.6211 | 61.1743 | 61.2847 | 15.23 |
| **12** | lng | float64 | 0 | 38713 | -166.771 | -149.284 | -149.654 | 15.23 |
| **13** | ALand | int64 | 0 | 38665 | 2823180154 | 509234898 | 270593047 | 15.23 |
| **14** | AWater | int64 | 0 | 23212 | 3101986247 | 1859309 | 66534601 | 9.86 |
| **15** | pop | int64 | 0 | 8319 | 4619 | 3727 | 8736 | 12.63 |
| **16** | male_pop | int64 | 0 | 4886 | 2725 | 1780 | 5166 | 11.74 |
| **17** | female_pop | int64 | 0 | 4986 | 1894 | 1947 | 3570 | 11.78 |
| **18** | rent_mean | float64 | 462 | 38220 | 1366.25 | 2347.69 | 2071.31 | 15.21 |
| **19** | rent_median | float64 | 462 | 2517 | 1405 | 2351 | 2089 | 10.48 |
| **20** | rent_stdev | float64 | 462 | 38205 | 650.164 | 382.736 | 442.891 | 15.20 |
| **21** | rent_sample_weight | float64 | 462 | 37997 | 131.51 | 4.32064 | 195.078 | 15.20 |
| **22** | rent_samples | float64 | 462 | 2333 | 372 | 44 | 1749 | 10.41 |
| **23** | rent_gt_10 | float64 | 463 | 9972 | 0.85676 | 0.79545 | 0.99469 | 9.25 |
| **24** | rent_gt_15 | float64 | 463 | 17136 | 0.65676 | 0.56818 | 0.97403 | 12.91 |
| **25** | rent_gt_20 | float64 | 463 | 21017 | 0.47838 | 0.56818 | 0.9268 | 13.77 |
| **26** | rent_gt_25 | float64 | 463 | 22548 | 0.35405 | 0.45455 | 0.8902 | 13.99 |
| **27** | rent_gt_30 | float64 | 463 | 22954 | 0.28108 | 0.20455 | 0.73022 | 14.01 |
| **28** | rent_gt_35 | float64 | 463 | 22704 | 0.21081 | 0.20455 | 0.62574 | 13.97 |
| **29** | rent_gt_40 | float64 | 463 | 22271 | 0.15135 | 0.20455 | 0.54368 | 13.90 |
| **30** | rent_gt_50 | float64 | 463 | 20971 | 0.12432 | 0 | 0.32999 | 13.67 |
| **31** | universe_samples | int64 | 0 | 2368 | 661 | 50 | 1933 | 10.43 |
| **32** | used_samples | int64 | 0 | 2276 | 370 | 44 | 1694 | 10.35 |
| **33** | hi_mean | float64 | 390 | 38415 | 107395 | 136547 | 69361.2 | 15.22 |
| **34** | hi_median | float64 | 390 | 30964 | 92807 | 119141 | 57976 | 14.81 |
| **35** | hi_stdev | float64 | 390 | 38410 | 70691.1 | 84268.8 | 45054.4 | 15.22 |
| **36** | hi_sample_weight | float64 | 390 | 38424 | 329.854 | 288.409 | 1104.23 | 15.23 |
| **37** | hi_samples | float64 | 390 | 3740 | 874 | 1103 | 1955 | 11.35 |

| | Name | dtypes | Missing | Uniques | First Value | Second Value | Third Value | Entropy |
|---|---|---|---|---|---|---|---|---|
| **38** | family_mean | float64 | 434 | 38387 | 114330 | 148642 | 67678.5 | 15.22 |
| **39** | family_median | float64 | 434 | 31867 | 101229 | 143026 | 58248 | 14.86 |
| **40** | family_stdev | float64 | 434 | 38384 | 63955.8 | 69628.7 | 38155.8 | 15.22 |
| **41** | family_sample_weight | float64 | 434 | 38385 | 161.152 | 159.209 | 1023.98 | 15.22 |
| **42** | family_samples | float64 | 434 | 2881 | 519 | 836 | 1858 | 10.91 |

## Second part of features

In [23]:
```
#looking the shape of data
resumetable(df_features)[43:]
```

```
Dataset Shape: (39030, 78)
```

Out[23]:

| | Name | dtypes | Missing | Uniques | First Value | Second Value | Third Value | Entropy |
|---|---|---|---|---|---|---|---|---|
| **43** | hc_mortgage_mean | float64 | 841 | 37939 | 2266.23 | 2485.11 | NaN | 15.19 |
| **44** | hc_mortgage_median | float64 | 841 | 3266 | 2283 | 2306 | NaN | 11.03 |
| **45** | hc_mortgage_stdev | float64 | 841 | 37926 | 768.535 | 919.762 | NaN | 15.19 |
| **46** | hc_mortgage_sample_weight | float64 | 841 | 37851 | 41.6564 | 180.929 | NaN | 15.20 |
| **47** | hc_mortgage_samples | float64 | 841 | 2343 | 155 | 797 | NaN | 10.58 |
| **48** | hc_mean | float64 | 890 | 37793 | 840.672 | 712.331 | 525.891 | 15.18 |
| **49** | hc_median | float64 | 890 | 1487 | 776 | 742 | 810 | 9.52 |
| **50** | hc_stdev | float64 | 890 | 37782 | 341.856 | 336.988 | 392.272 | 15.17 |
| **51** | hc_samples | float64 | 890 | 1365 | 58 | 256 | 22 | 9.74 |
| **52** | hc_sample_weight | float64 | 890 | 33170 | 29.7437 | 159.323 | 10.8344 | 14.91 |
| **53** | home_equity_second_mortgage | float64 | 677 | 8133 | 0.00469 | 0.03609 | 0 | 10.49 |
| **54** | second_mortgage | float64 | 677 | 8856 | 0.01408 | 0.06078 | 0 | 10.97 |
| **55** | home_equity | float64 | 677 | 17375 | 0.02817 | 0.07407 | 0 | 13.33 |
| **56** | debt | float64 | 677 | 25932 | 0.7277 | 0.75689 | 0 | 14.40 |
| **57** | second_mortgage_cdf | float64 | 677 | 17055 | 0.50216 | 0.1552 | 1 | 11.81 |
| **58** | home_equity_cdf | float64 | 677 | 23341 | 0.77143 | 0.56228 | 1 | 13.76 |
| **59** | debt_cdf | float64 | 677 | 27680 | 0.30304 | 0.23925 | 1 | 14.51 |
| **60** | hs_degree | float64 | 275 | 22289 | 0.82841 | 0.9409 | 0.99097 | 14.20 |
| **61** | hs_degree_male | float64 | 289 | 22592 | 0.82784 | 0.97253 | 0.99661 | 14.13 |
| **62** | hs_degree_female | float64 | 328 | 21533 | 0.8294 | 0.91503 | 0.98408 | 14.06 |
| **63** | male_age_mean | float64 | 273 | 38090 | 38.4584 | 37.2622 | 21.9629 | 15.21 |
| **64** | male_age_median | float64 | 273 | 717 | 39.25 | 39.3333 | 22.25 | 8.55 |
| **65** | male_age_stdev | float64 | 273 | 37342 | 17.6545 | 19.6676 | 11.0966 | 15.17 |
| **66** | male_age_sample_weight | float64 | 273 | 38513 | 709.063 | 503.834 | 1734.06 | 15.23 |
| **67** | male_age_samples | float64 | 273 | 4885 | 2725 | 1780 | 5166 | 11.77 |
| **68** | female_age_mean | float64 | 302 | 38060 | 32.7818 | 38.9796 | 22.2043 | 15.21 |
| **69** | female_age_median | float64 | 302 | 694 | 31.9167 | 39.6667 | 23.1667 | 8.59 |
| **70** | female_age_stdev | float64 | 302 | 37312 | 19.3188 | 20.0551 | 13.8658 | 15.17 |
| **71** | female_age_sample_weight | float64 | 302 | 38488 | 440.464 | 466.655 | 887.678 | 15.23 |
| **72** | female_age_samples | float64 | 302 | 4985 | 1894 | 1947 | 3570 | 11.81 |
| **73** | pct_own | float64 | 390 | 29602 | 0.25053 | 0.94989 | 0.00759 | 14.68 |
| **74** | married | float64 | 275 | 26133 | 0.47388 | 0.52381 | 0.50459 | 14.49 |
| **75** | married_snp | float64 | 275 | 11637 | 0.30134 | 0.01777 | 0.06676 | 12.96 |
| **76** | separated | float64 | 275 | 6893 | 0.03443 | 0.00782 | 0.01 | 10.64 |
| **77** | divorced | float64 | 275 | 15838 | 0.09802 | 0.13575 | 0.01838 | 13.65 |

## Look head of the data

In [24]:
```
df_features.head()
```

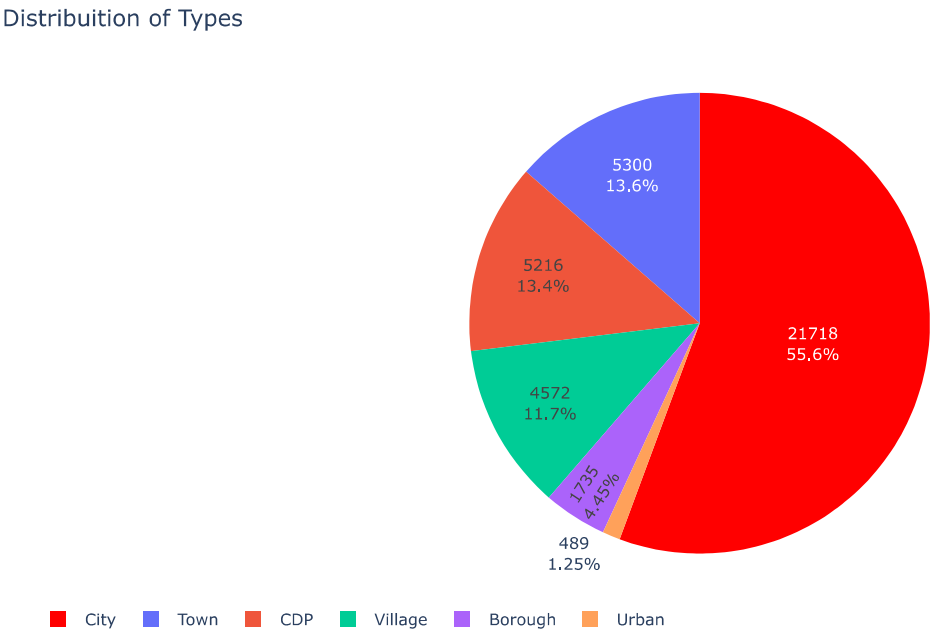
Out[24]:

| | SUMLEVEL | COUNTYID | STATEID | state | state_ab | city | place | type | primary | zip_code | ... | female_age_mean | female_age_median | female_age_stdev | female_age_sample_weight | female_age_samples | pct_own | r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 140 | 16 | 2 | Alaska | AK | Unalaska | Unalaska City | City | tract | 99685 | ... | 32.78177 | 31.91667 | 19.31875 | 440.46429 | 1894.0 | 0.25053 | |
| **1** | 140 | 20 | 2 | Alaska | AK | Eagle River | Anchorage | City | tract | 99577 | ... | 38.97956 | 39.66667 | 20.05513 | 466.65478 | 1947.0 | 0.94989 | |
| **2** | 140 | 20 | 2 | Alaska | AK | Jber | Anchorage | City | tract | 99505 | ... | 22.20427 | 23.16667 | 13.86575 | 887.67805 | 3570.0 | 0.00759 | |
| **3** | 140 | 20 | 2 | Alaska | AK | Anchorage | Point Mackenzie | City | tract | 99501 | ... | 37.00750 | 34.00000 | 22.06347 | 281.49420 | 1049.0 | 0.20247 | |
| **4** | 140 | 20 | 2 | Alaska | AK | Anchorage | Anchorage | City | tract | 99504 | ... | 34.96611 | 31.75000 | 20.49887 | 655.98066 | 2905.0 | 0.56936 | |

5 rows × 78 columns

# Type Feature

In [25]:
```
percentual_types = round(df_features["type"].value_counts(), 2)

types = round(df_features["type"].value_counts() / len(df_features["type"]) * 100,2)

labels = list(types.index)
values = list(types.values)

trace1 = go.Pie(labels=labels, values=values, marker=dict(colors=['red']), text = percentual_types.values,)

layout = go.Layout(title='Distribuition of Types', legend=dict(orientation="h"));

fig = go.Figure(data=[trace1], layout=layout)
iplot(fig)
```



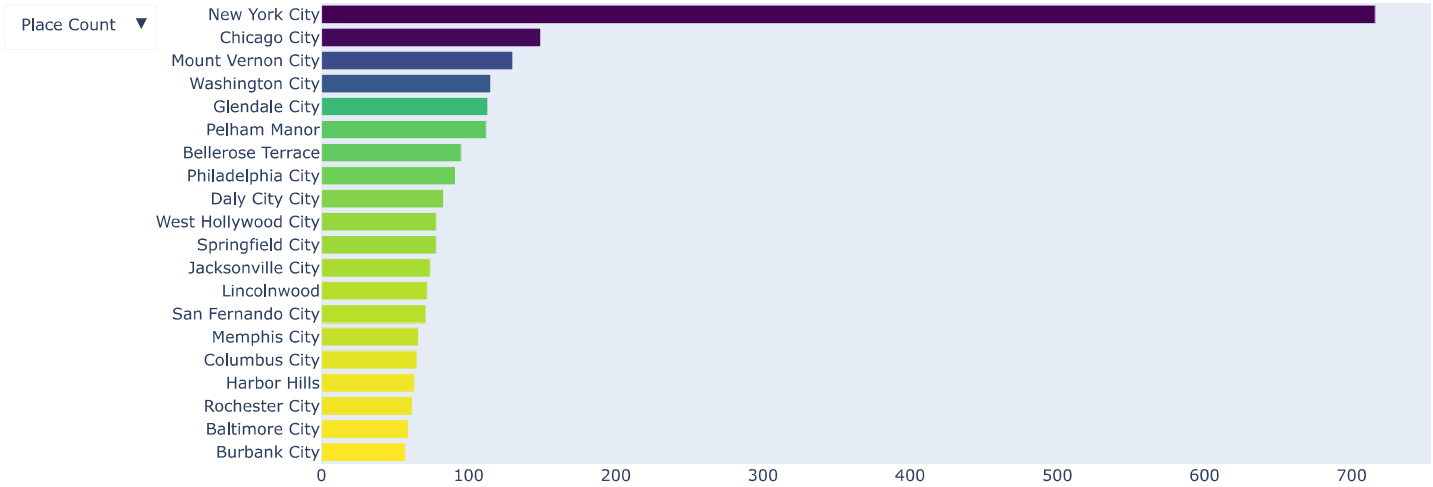

# Interactive Plot

In [26]:
```
state_count = df_features["state"].value_counts()
city_count = df_features.city.value_counts()
place_count = df_features.place.value_counts()
primary_count = df_features.primary.value_counts()
```

In [27]:
```
trace1 = go.Bar(x=state_count[:20].values[::-1],
                y=state_count[:20].index[::-1],
                orientation='h', visible=True,
                      name='Top 20 States',
                      marker=dict(
                          color=city_count[:20].values[::-1],
                          colorscale = 'Viridis',
                          reversescale = True
                      ))

trace2 = go.Bar(x=city_count[:20].values[::-1],
                      y=city_count[:20].index[::-1],
                      orientation = 'h', visible=False,
                      name='TOP 20 Citys',
                      marker=dict(
                          color=city_count[:20].values[::-1],
                          colorscale = 'Viridis',
                          reversescale = True
                      ))

trace3 = go.Histogram(y=sorted(df_features['type'], reverse=True), histnorm='percent', orientation='h', visible=False,
                      name='Type Count')

trace4 = go.Bar(x=place_count[:20].values[::-1],
                y=place_count[:20].index[::-1],
```

```
                orientation='h', visible=False,
                name='Top 20 Place',
                marker=dict(
                    color=city_count[:20].values[::-1],
                    colorscale = 'Viridis',
                    reversescale = True
                      ))

data = [trace1, trace2, trace3, trace4]

updatemenus = list([
    dict(active=-1,
         x=-0.15,
         buttons=list([
            dict(
                label = 'State Count',
                 method = 'update',
                 args = [{'visible': [True, False, False, False]},
                         {'title': 'TOP 20 State Count'}]),

             dict(
                  label = 'City Count',
                 visible=True,
                 method = 'update',
                 args = [{'visible': [False, True, False, False]},
                     {'title': 'TOP 20 City Count'}]),

            dict(
                 label = 'Type Count',
                 method = 'update',
                 args = [{'visible': [False, False, True, False]},
                     {'title': 'Type Counts'}]),

            dict(
                 label = 'Place Count',
                 method = 'update',
                 args = [{'visible': [False, False, False, True]},
                     {'title': ' Top 20 Place Count'}])
        ]),
    )
])


layout = dict(title='The count of the principal Categorical Features <br>(Select from Dropdown)',
              showlegend=False,
              updatemenus=updatemenus)

fig = dict(data=data, layout=layout)

iplot(fig)
```

Top 20 Place Count



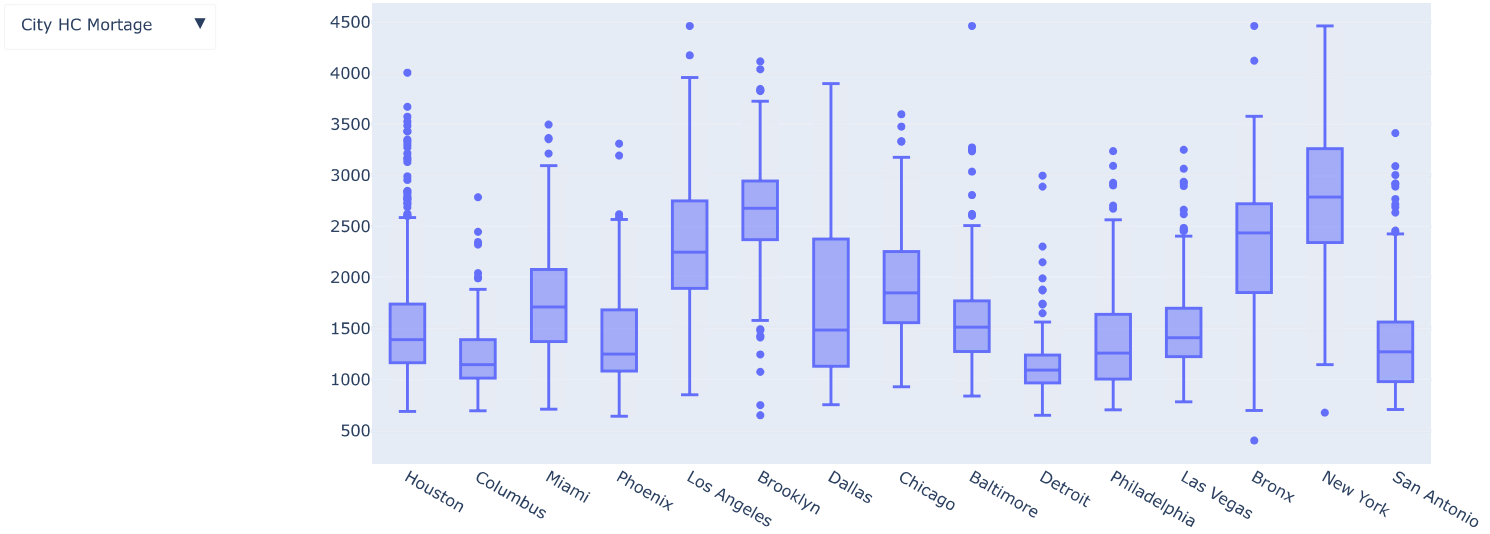

In [28]:
```
df_features['ALand_div_1M'] = np.log(df_features['ALand'] / 1000000)
```

## Some Boxplots of City's

In [29]:
```
trace1  = go.Box(
    x=df_features[df_features.city.isin(city_count[:15].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:15].index.values)]['rent_median'],
    showlegend=False, visible=True
)

trace2  = go.Box(
    x=df_features[df_features.city.isin(city_count[:15].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:15].index.values)]['family_median'],
    showlegend=False, visible=False
)

trace3 = go.Box(
    x=df_features[df_features.city.isin(city_count[:15].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:15].index.values)]['hi_median'],
    showlegend=False, visible=False
)

trace4 = go.Box(
    x=df_features[df_features.city.isin(city_count[:15].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:15].index.values)]['hc_mortgage_mean'],
    showlegend=False, visible=False
)

data = [trace1, trace2, trace3, trace4]

updatemenus = list([
    dict(active=-1,
```

```
        x=-0.15,
        buttons=list([

           dict(
               label = 'City Rent Boxplot',
                method = 'update',
                args = [{'visible': [True, False, False, False]},
                    {'title': 'TOP 15 Citys - Rent Median'}]),

            dict(
                 label = 'City Family Boxplot',
                method = 'update',
                args = [{'visible': [False, True, False, False]},
                    {'title': 'TOP 15 Citys - Family Income Median'}]),

           dict(
                label = 'City House Inc',
                method = 'update',
                args = [{'visible': [False, False, True, False]},
                    {'title': 'TOP 15 Citys - House income Median'}]),

           dict(
                label =  'City HC Mortage',
                method = 'update',
                args = [{'visible': [False, False, False, True]},
                    {'title': 'TOP 15 Citys - Home Cost Mortage'}])
       ]),
    )
])

layout = dict(title='Citys BoxPlots of Medians <br>(Select metrics from Dropdown)',
              showlegend=False,
              updatemenus=updatemenus)

fig = dict(data=data, layout=layout)

iplot(fig, filename='dropdown')
```

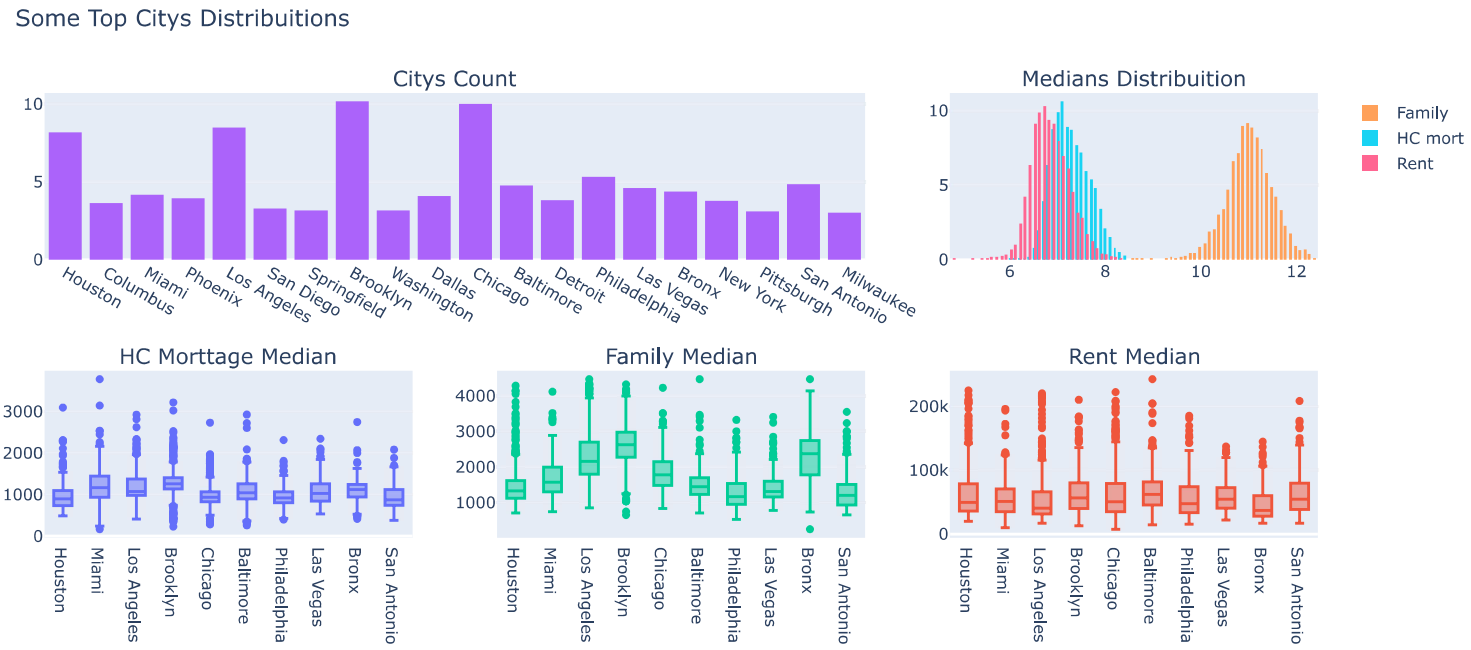
TOP 15 Citys - Home Cost Mortage




## Data Visualization

In [30]:
```
city_count = df_features.city.value_counts()

#First plot
trace0 = go.Box(
    x=df_features[df_features.city.isin(city_count[:10].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:10].index.values)]['rent_median'],
    showlegend=False
)

#Second plot
trace1 = go.Box(
    x=df_features[df_features.city.isin(city_count[:10].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:10].index.values)]['family_median'],
    showlegend=False
)

#Second plot
trace2 = go.Box(
    x=df_features[df_features.city.isin(city_count[:10].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:10].index.values)]['hc_mortgage_median'],
    showlegend=False
)

#Third plot
trace3 = go.Histogram(
    x=df_features[df_features.city.isin(city_count[:20].index.values)]['city'], histnorm='percent',
    showlegend=False
)
#Third plot
trace4 = go.Histogram(
    x=np.log(df_features['family_median']).sample(5000), histnorm='percent', autobinx=True,
```

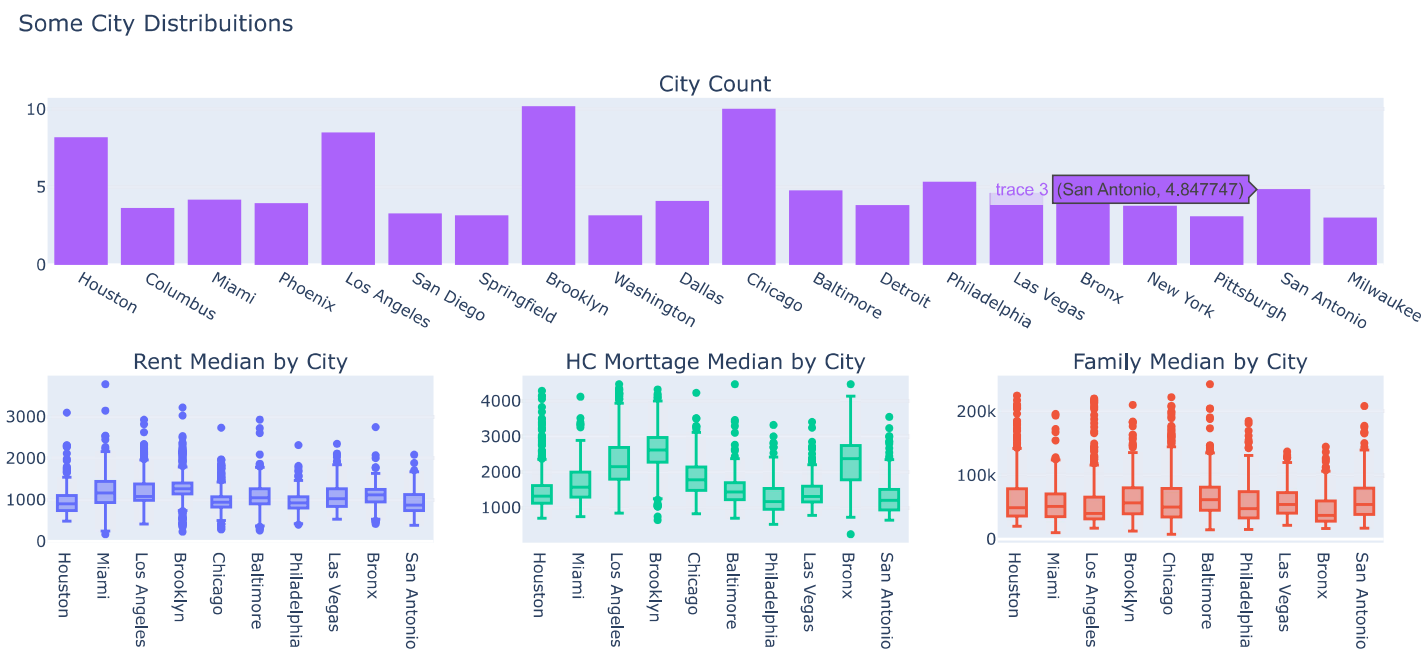
```
    showlegend=True, name='Family'
)

#Third plot
trace5 = go.Histogram(
    x=np.log(df_features['hc_mortgage_median']).sample(5000), histnorm='percent', autobinx=True,
    showlegend=True, name='HC mort'
)

#Third plot
trace6 = go.Histogram(
    x=np.log(df_features['rent_median']).sample(5000), histnorm='percent', autobinx=True,
    showlegend=True, name="Rent"
)

#Creating the grid
fig = tls.make_subplots(rows=2, cols=3, specs=[[{'colspan': 2}, None, {}], [{}, {}, {}]],
                          subplot_titles=("Citys Count",
                                          "Medians Distribuition",
                                          "HC Morttage Median",
                                          "Family Median",
                                          "Rent Median"))

#setting the figs
fig.append_trace(trace0, 2, 1)
fig.append_trace(trace1, 2, 3)
fig.append_trace(trace2, 2, 2)
fig.append_trace(trace3, 1, 1)
fig.append_trace(trace4, 1, 3)
fig.append_trace(trace5, 1, 3)
fig.append_trace(trace6, 1, 3)

fig['layout'].update(showlegend=True, title="Some Top Citys Distribuitions")

iplot(fig)
```




# Boxplots of some City values

In [31]:
```
#First plot
trace0 = go.Box(
    x=df_features[df_features.city.isin(city_count[:10].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:10].index.values)]['rent_median'],
    showlegend=False
)

#Second plot
trace1 = go.Box(
    x=df_features[df_features.city.isin(city_count[:10].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:10].index.values)]['family_median'],
    showlegend=False
)

#Second plot
trace2 = go.Box(
    x=df_features[df_features.city.isin(city_count[:10].index.values)]['city'],
    y=df_features[df_features.city.isin(city_count[:10].index.values)]['hc_mortgage_median'],
    showlegend=False
)

#Third plot
trace3 = go.Histogram(
    x=df_features[df_features.city.isin(city_count[:20].index.values)]['city'], histnorm='percent',
    showlegend=False
)

#Creating the grid
fig = tls.make_subplots(rows=2, cols=3, specs=[[{'colspan': 3}, None, None], [{}, {}, {}]],
                          subplot_titles=("City Count",
```

```
                                    "Rent Median by City",
                                    "HC Morttage Median by City",
                                    "Family Median by City"
                                    ))
#setting the figs
fig.append_trace(trace0, 2, 1)
fig.append_trace(trace1, 2, 3)
fig.append_trace(trace2, 2, 2)
fig.append_trace(trace3, 1, 1)

fig['layout'].update(showlegend=True, title="Some City Distribuitions")
iplot(fig)
```

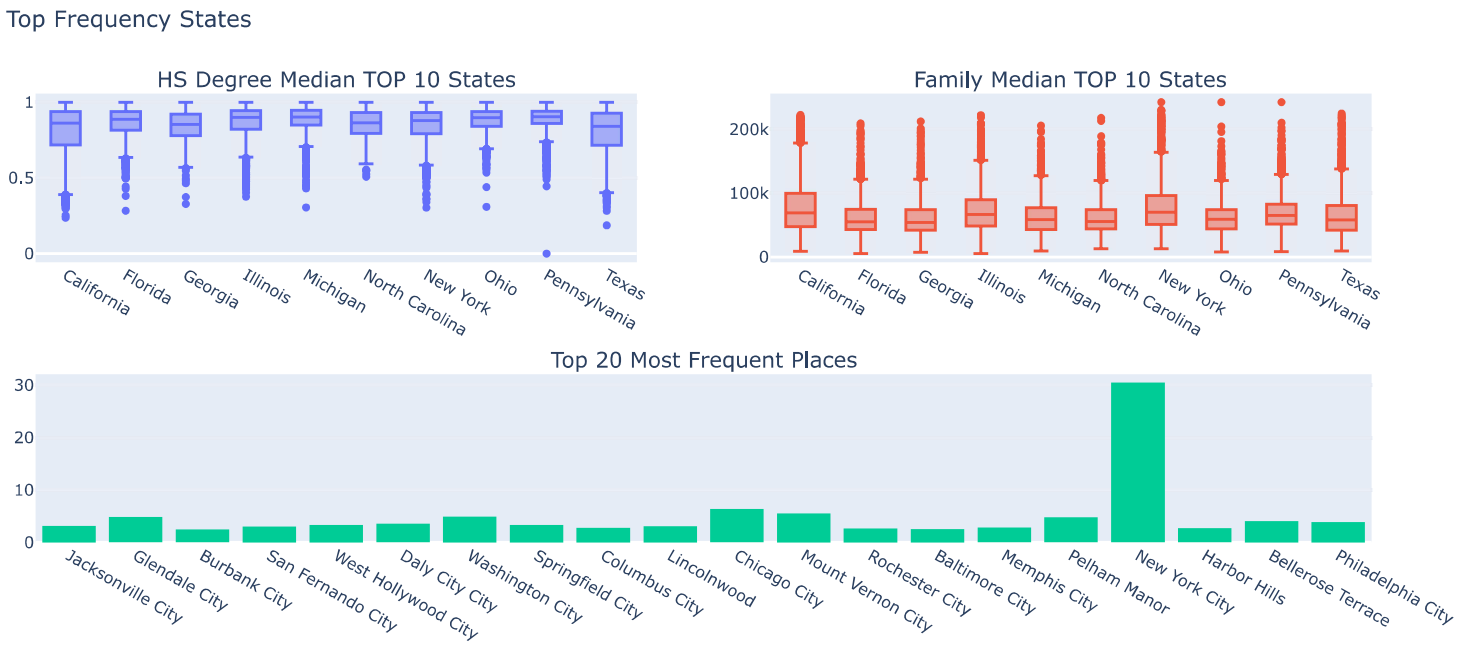



## State by some numerical features

In [32]:

```
#First plot
trace0 = go.Box(
    x=df_features[df_features.state.isin(state_count[:10].index.values)]['state'],
    y=df_features[df_features.state.isin(state_count[:10].index.values)]['hs_degree'],
    name="Top 10 States", showlegend=False
)

#Second plot
trace1 = go.Box(
    x=df_features[df_features.state.isin(state_count[:10].index.values)]['state'],
    y=df_features[df_features.state.isin(state_count[:10].index.values)]['family_median'],
    name="Top 15 Sucessful", showlegend=False
)

#Third plot
trace2 = go.Histogram(
    x=df_features[df_features.place.isin(place_count[:20].index.values)]['place'],
    histnorm='percent', name="Top 20 Place's", showlegend=False
```

```
)

#Creating the grid
fig = tls.make_subplots(rows=2, cols=2, specs=[[{}, {}], [{'colspan': 2}, None]],
                          subplot_titles=('HS Degree Median TOP 10 States',
                                          'Family Median TOP 10 States',
                                          "Top 20 Most Frequent Places"))

#setting the figs
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 2)
fig.append_trace(trace2, 2, 1)

fig['layout'].update(showlegend=True, title="Top Frequency States")

iplot(fig)
```




## Data Visualization

In [33]:
```
#First plot
trace0 = go.Box(
    x=df_features['type'],
    y=df_features['rent_median'],
    showlegend=False
)

#Second plot
trace1 = go.Box(
    x=df_features['type'],
    y=df_features['family_median'],
    showlegend=False
)
```

```
#Second plot
trace2 = go.Histogram(
    x=df_features['type'], histnorm="percent",
    showlegend=False
)

trace3 = go.Scatter(
    x=df_features['rent_median'],
    y=df_features['family_median'],
    showlegend=False,
    mode = 'markers'
)

#Creating the grid
fig = tls.make_subplots(rows=2, cols=3, specs=[[{}, {}, {}], [{'colspan': 3}, None, None]],
                          subplot_titles=("Rent Median by Type",
                                          "Type Count",
                                          "Family Median by Type",
                                          "Rent Median x Family Median"))

#setting the figs
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 3)
fig.append_trace(trace2, 1, 2)
fig.append_trace(trace3, 2, 1)

fig['layout'].update(showlegend=True,
                     title="Some Type Distribuitions")

iplot(fig)
```