Diamonds and Dollars: Story, Stats, Stacked, Sense

(https://im.rediff.com/money/2018/feb/23diamond1.jpg?w=670&h=900)

What's This Notebook About?

Its a story of guy in search for his perfect diamond for his Dday, and we are attempting to help him find it with a Good visualization, Statistics, and Good machine learning model.

Anything Interesting here?

Yes. Whole content can be dived into four parts. Few things about Data, Statistics, Visualization, and Modeling.

- Visualization of Statistics of The Data.
- How much does people pay for a diamond for it weight, color, flaws and cuts
- Fun Experiments with **Default feature importances**
- Implementation of Permutation Feature Importance With a **Very Basic Idea and ELI5 Comparission.**
- Finally, Interpretation of model predictions with **SHAP Values**.

1. Introduction

Wikipedia says lot about diamonds. Cutting all those jargons and things that are out of scope of this notebook. Lets say, **Diamond is a chunk of carbon**, which been through hell on earth at extreme pressures and temperatures. Diamonds are rare, fancy, desiarable, and worth keeping. This a data story of a hopeless guy in search for his perfect diamond.

~~An Analysis~~ A Story of A Perfect Stone

Lets say there is 🤵 with zero knowledge on 💎 and went to purchase a beautiful 💍 for his fiancee 👰. He just had a chat with 👴,and jeweller keeps on going with details, specs and all the boring stuff with price tags of shock. Poor guy just 😵 to the core and looking for desperate help in making a fine choice with trade off between 💲💰 tag and 🤩 of the diamond. Luckily our guy is a 📈📊📉 guy, so he fired up his 💻 and got a kaggle dataset of diamonds, started making a data dance to help him find this perfect diamond.

Our guy is likely to answer few of following questions to himself by end of his analysis

- How much should one pay 💰 for a 💎?
- There are range of diamonds out there which deserve all his savings👆, Which one the worth spending fortune💸💸?
- Have to many technical and physical parameters 🚧 of diamonds, which things have a best correlation📈📊 to Price tag?
- Ultimately can he predict a price 💲 for his ideal diamond based 🎯📊✓?

Okay he got his questions ✅ and knows whats he is 🔍 for. As this is an ❓area for him, He just want to know few details of about diamonds specs, so that he can make few assumptions and make a better model. When he googled 🔎 for requried information,he came to know about **4C's of** 💎, **and Diamond components** are key parameters that decide the 💲tag.

1.1 Diamond 101: 4C's Diamond - Carat, Color, Clarity, Cut

Our guy understood that diamond quality and price are gauged by four important parameters called 4 C's, which are Carat, Clartiy, Color, Cut. Infographic gives what and what about 4 C's in a fun way. Lets see the elobrate these 4C's.

1. **Carat:** Carat is the weight of the diamond. Bigger the diamond, rare to find them.
2. **Clarity:** Clarity of the diamond is intrusions in the diamond, which could be internal flaws, external defects.
3. **Color:** Color of the diamond is self explinary, and on scale of AtoZ, A is best and colorless, where as Z is worst and yellowish color. The colorless diamonds are rare and costly.

4. **Cut:** Cut of the diamond is about physical dimensions of the diamond. It determine the light amount enter into the diamond. Anotomy of the diamond is an important parameter.

1.2 Diamond Components and Cuts: Price Proportional Parameters

There are several things to consider to see in a diamond. Few such is table%, depth% and cuts. Diamond anotomy is given it the cut of the diamond infographic in the above image.

- All the basic information about the diamond cut are givn above.
- Side view of the diamond show the terminology and meaning of his features.
- Thus table length, table width, diamond depth, depth%, and table% are very important features to determine the price of a diamond.

Hmm, thanks to google, 👨 got all the information about 💎, and basic terminology of the 💎. Now, what? As part of routine, he want to 📊📈 diamonds to see which feature are 🔥 and which are 🚨! So, next **Let the Data Tell A Story** 💭🖼 for itself about 💵💵.

2. Libraries And Utilities

In [31]:
```
#importing modules

import warnings
warnings.filterwarnings('ignore')
import time
t = time.time()

print('Importing startred...')

# basic moduele
import os
import numpy as np
import pandas as pd
import re
from scipy import stats
from random import randint

# visualization moduels
import matplotlib.pyplot as plt
import matplotlib
%matplotlib inline
import seaborn as sns
#import missingno as msno


# preprocessing modules
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

from sklearn.linear_model import LinearRegression
from sklearn. linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
#from xgboost import XGBRegressor

from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics



print('Done,All the required modules are imported. Time elapsed: {}sec'.format(time.time()
```

```
Importing startred...
Done,All the required modules are imported. Time elapsed: 0.001995563507080078sec
```

2.1 Dataloading, Colorpalette for visulaization

In [2]:
```
df = pd.read_csv('E:\\machine learning\\projects\\diamonds.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  53940 non-null  int64  
 1   carat       53940 non-null  float64
 2   cut         53940 non-null  object 
 3   color       53940 non-null  object 
 4   clarity     53940 non-null  object 
 5   depth       53940 non-null  float64
 6   table       53940 non-null  float64
 7   price       53940 non-null  int64  
 8   x           53940 non-null  float64
 9   y           53940 non-null  float64
 10  z           53940 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

In [4]:
```
df.head(10)
```

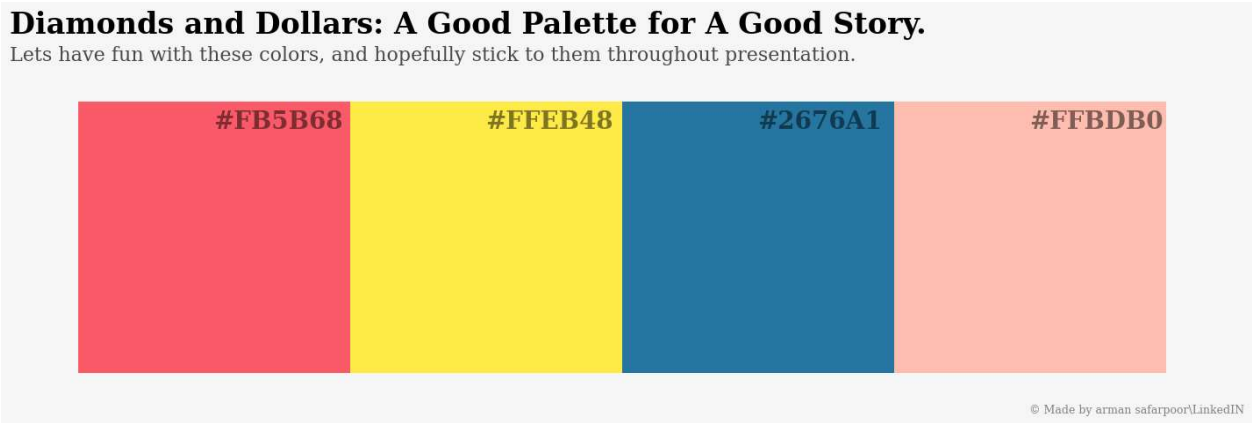
Out[4]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

In [5]:
```
df.shape
```

Out[5]: (53940, 11)

In [6]:
```
# color palette

#black  '#1F0C07',
#Light pink '#FA74BF'

colors = ['#FB5B68','#FFEB48','#2676A1','#FFBDB0',]
colormap = matplotlib.colors.LinearSegmentedColormap.from_list("",colors)


sns.palplot(colors, size = 4)
plt.text(-0.75,-0.75, 'Diamonds and Dollars: A Good Palette for A Good Story.',
         {'font':'serif', 'size':24, 'weight':'bold'})
plt.text(-0.75,-0.65, 'Lets have fun with these colors, and hopefully stick to them through
         {'font':'serif', 'size':16},alpha = 0.7)
for idx,values in enumerate(colors):
    plt.text(idx,-0.40, colors[idx],{'font':'serif', 'size':20, 'weight':'bold'}, alpha = (
plt.gcf().set_facecolor('#f5f6f6')
plt.box(None)
plt.axis('off')
plt.text(3,0.65,'© Made by arman safarpoor\LinkedIN',{'font':'serif', 'size':9,  'color':'
plt.show()
```



3. Statistical Insights about Dataset

Histories were made,and kingdoms fell for claiming precious things since the stone age. One such precious thing for us is Diamonds. As per one survey of luxury goods, Diamonds are most desirable things for 90 out of 100 people, why u ask? People say because of it's rarity and stunning beauty. Though, diamonds are diamonds, each have its own unique characteristics to consider for gauging its value. Coming to this dataset, we are given few such features along with there market price value in USD. Lets see these stats from statistical stand point and get some insights about diamonds.

📣📣📢, lets continue our story. At the end of the day our 👨 need to find his perfect diamond. To Meet make it happen fast, guy have decided to analyize data and remove outliers💣 from dataset. He decided to approched guy friend of him named 👨‍🏫(lol statistics) to help him his perfect data to make a 🎯 model to get his perfect 💎.
👨‍🏫 idea is simple, see basic stats and determine which paremeters are odd from statistical standpoint and ✂✂ to off.

3.1 Data Summary: Descriptive Statistics & Datatypes

In [7]:
```
## information and descriptive statistical dataframes

df.drop(columns = ['Unnamed: 0'], inplace = True) ### unnecessary coloumn, repeated index

info_df = (pd.DataFrame({'Features':df.columns,'Non Null Count':df.count(),
                         'Null Count': df.isnull().sum(), 'Datatypes':df.dtypes,})
        .sort_values(by='Datatypes').reset_index(drop = True))
stats = df.describe(include = 'all').T.reset_index()



tota = pd.merge( info_df,stats,
                right_on = 'index',left_on = 'Features',how = 'inner').drop(columns = ['in
tota = tota.fillna(0)



colors = ['#FB5B68','#FFEB48','#2676A1','#FFBDB0',]
tota.style.bar(subset = ['mean', 'std', 'min', '25%','50%','75%', 'max'],axis = 1 ,color =
    .format({'mean':"{:20,.0f}",
             'std':"{:20,.0f}",
             'min':"{:20,.0f}",
             '25%':"{:20,.0f}",
             '50%':"{:20,.0f}",
             '75%':"{:20,.0f}",
             'max':"{:20,.0f}"})\
    .format({"Features": lambda x:  x.upper()},
           )\
    .set_properties(**{'background-color': '#f9f9f9',
                                   'color': 'black',
                                   'border-color': 'white'})\
    .hide_index()
```

Out[7]:

Features	Non Null Count	Null Count	Datatypes	unique	top	freq	mean	std	min	25%	
PRICE	53940	0	int64	0	0	0	3932.799722	3989.439738	326.000000	950.000000	24
CARAT	53940	0	float64	0	0	0	0.797940	0.474011	0.200000	0.400000	
DEPTH	53940	0	float64	0	0	0	61.749405	1.432621	43.000000	61.000000	
TABLE	53940	0	float64	0	0	0	57.457184	2.234491	43.000000	56.000000	
X	53940	0	float64	0	0	0	5.731157	1.121761	0.000000	4.710000	
Y	53940	0	float64	0	0	0	5.734526	1.142135	0.000000	4.720000	
Z	53940	0	float64	0	0	0	3.538734	0.705699	0.000000	2.910000	
CUT	53940	0	object	5	Ideal	21551	0.000000	0.000000	0.000000	0.000000	
COLOR	53940	0	object	7	G	11292	0.000000	0.000000	0.000000	0.000000	
CLARITY	53940	0	object	8	SI1	13065	0.000000	0.000000	0.000000	0.000000	

Mr.statistics, 👨‍🎓 found that data have 3️⃣ categorical features, which Cut, Clarity, Color, along with 4️⃣ Numerical features are depth%, table%, x,y,and z. 🎯 is 💲. Now 👨 told his friend 👨‍🎓 about 4C's he learned from his research. So, 👨‍🎓 decided that they are clearly dealing with Ordinal categorical features, So, it is Safe to assume they have some kind of ordered ⚖️⚖️⚖️ on 💲.
With initial understandig of stats, we can make these observations,

- There are no **Explicit Null Values** Present in the dataset.

- Price, depth,y, and z have many outliers , which can be be infered for percentailes and standad deviation.
- Data normailzation could be needed for price.
- They have three kinds of datatypes, int, float, and object.

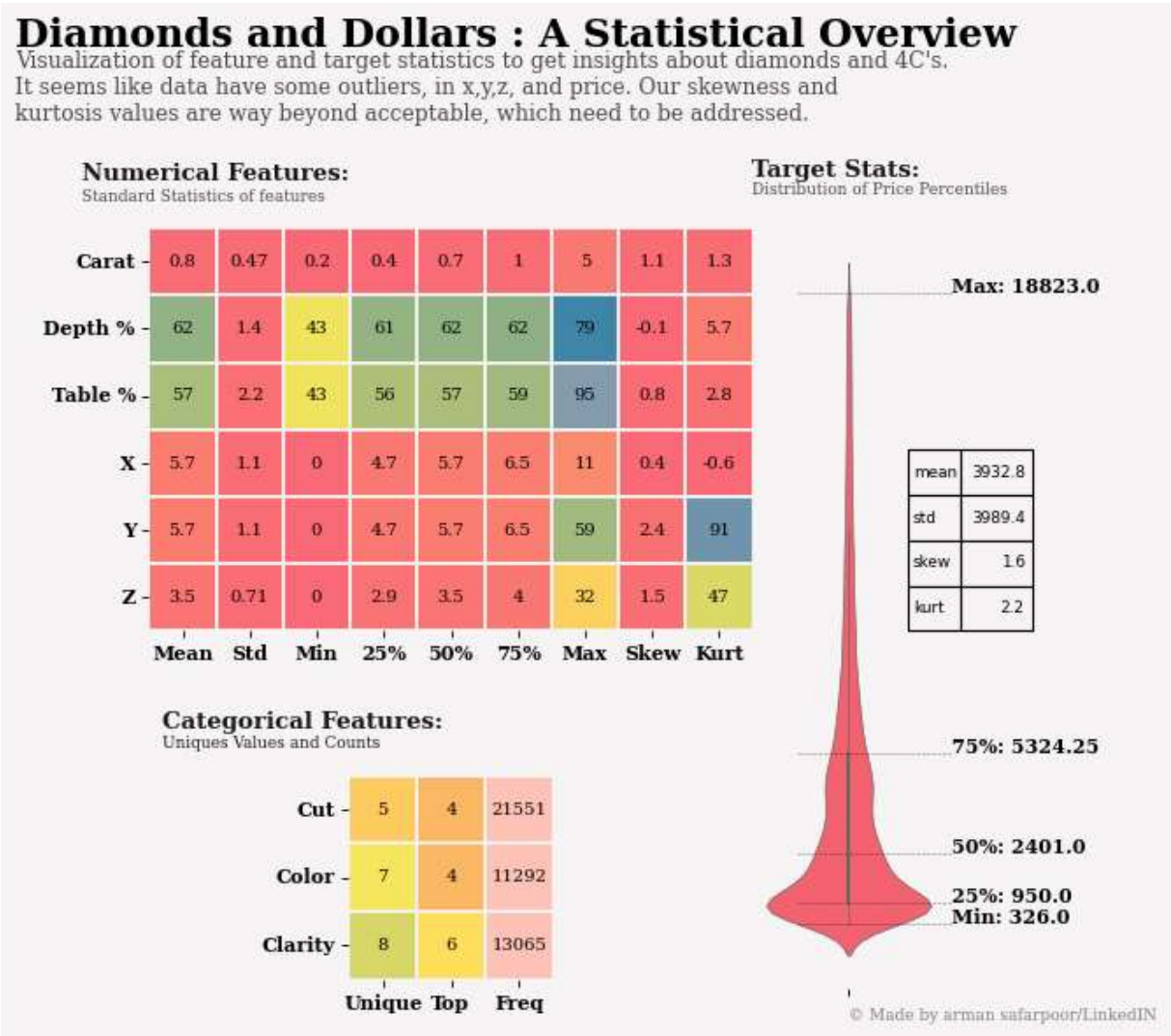
3.2 Better way to look at Statistics: Visualization

In [8]:
```
### Statistics dataframe creation

des_stats = df.describe(exclude = ['object']).drop(index = ['count'], axis = 0).T
skew = []
kurt = []
num_cols = df.select_dtypes(exclude = ['object']).columns
for col in num_cols:
    skew.append(df[col].skew().round(1))
    kurt.append(df[col].kurt().round(1))

stats = pd.DataFrame({'skew':skew,'kurt':kurt}, index = num_cols)

all_stats = pd.merge(left = des_stats,right = stats, left_index = True, right_index = True
```

In [9]:

```
fig = plt.figure(figsize =(15,12), dpi = 80)
fig.patch.set_facecolor('#f6f5f5')

gs = fig.add_gridspec(8,8)
gs.update(wspace = 0, hspace = 3)

back_ground = '#f6f5f5'
ax1 = fig.add_subplot(gs[0:3,0:3])
ax2 = fig.add_subplot(gs[0:5,3])
ax3 = fig.add_subplot(gs[3:5,1:2])
ax4 = fig.add_subplot(gs[0:3,3])

axes = [ax1,ax2, ax3]

for ax in axes:
    for loc in ['left','right','top','bottom']:
        ax.spines[loc].set_visible(False)
    ax.set_facecolor('#f6f5f5')


#colormap for visualization
colormap  = matplotlib.colors.LinearSegmentedColormap.from_list("", colors)


#### statistics of numerical features
sns.heatmap(all_stats.drop(index = ['price']),
            annot = True, ax = ax1,
            vmin = -0, vmax = 120,
            square = True, linewidths = 0.09,linecolor = '#f6f5f5',
            cbar = False,
            cmap =colormap, annot_kws={'font':'serif', 'size':9, 'weight':'normal', 'color

#### statistics of target
#ax4 = ax2.twinx()
sns.violinplot(y = df['price'], ax = ax2, palette= [colors[0]],inner = 'box', saturation =


## labeling stats in violin plot
pr = df['price'].describe().drop(index = ['count', 'mean','std'], axis = 0)

for idx,value in pr.items():
    ax2.plot([-0.25,0.5], [value,value], **{'linewidth': 0.5, 'linestyle': '--','color':'bl
    if idx == 'max':
        idx = 'Max'
    elif idx == 'min':
        idx = 'Min'
    else:
        idx = idx
    ax2.text(0.5,value,'{}: {}'.format(idx,value) ,{'font':'serif', 'size':10, 'weight':'bo


ax2.axes.get_yaxis().set_visible(False)

### price stats table
price_stats = all_stats.T['price'].to_frame().round(1)
price_stats = (price_stats.T[['mean', 'std', 'skew', 'kurt']]).T
bbox = [1, 0, 0.35, 0.45]
ax4.table(cellText = price_stats.values, rowLabels = price_stats.index,
          bbox=bbox,cellColours = np.array(['#f6f5f5','#f6f5f5','#f6f5f5','#f6f5f5']).resha
          rowColours = ['#f6f5f5','#f6f5f5','#f6f5f5','#f6f5f5'])

ax4.axis('off')
```

```
#### statistics of categorical features
sx = df.describe(exclude = ['int64', 'float64','float']).drop(index = ['count'], axis = 0)
sx['top']= ['4', '4', '6']
sx = sx.astype(int)

sns.heatmap(sx,annot = True,fmt = '1.0f',vmin = -0, vmax = 20,ax = ax3,
            square = True, linewidths = 0.09,cbar = False,linecolor = '#f6f5f5',
            cmap =colormap, annot_kws={'font':'serif', 'size':9, 'weight':'normal', 'color

#### axis and labeling
ax1.set_yticklabels(labels = ['Carat', 'Depth %', 'Table %','X', 'Y', 'Z'],rotation = 0,**
ax1.set_xticklabels(labels = ['Mean','Std','Min','25%','50%','75%', 'Max','Skew','Kurt'],r

ax3.set_yticklabels(labels = ['Cut', 'Color', 'Clarity'],rotation = 0,**{'font':'serif', '
ax3.set_xticklabels(labels = ['Unique', 'Top', 'Freq'],rotation = 0,**{'font':'serif', 'si

### titles
ax1.text(-1., -0.7 , 'Numerical Features:',{'font':'serif', 'size':12, 'weight':'bold','co
ax1.text(-1., -0.4, 'Standard Statistics of features',{'font':'serif', 'size':8, 'color':'

ax2.text(-0.47, 22250 , 'Target Stats: ',{'font':'serif', 'size':12, 'weight':'bold','colo
ax2.text(-0.47, 21750 , 'Distribution of Price Percentiles',{'font':'serif', 'size':8,'col

ax3.text(-2.75, -0.7 , 'Categorical Features:',{'font':'serif', 'size':12, 'weight':'bold'
ax3.text(-2.75, -0.4 , 'Uniques Values and Counts',{'font':'serif', 'size':8,  'color':'bl

ax1.text(-2,-2.7,'Diamonds and Dollars : A Statistical Overview',{'font':'serif', 'size':2
ax1.text(-2,-1.6,"Visualization of feature and target statistics to get insights about dia

ax2.text(0,-2500,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':8,  'color':

fig.show()
```



From Stastical overview deduce following insights

1. **Cut:** Ideal cut is common with highest frequency. Cut determines ✂✂ of the jeweal as it determines the amount of light refract within the diamond and brings out that shining beauty, so, assuming this one goes hand in hand with 💲.

Cut Categories: Fair, Good, Very Good, Premium, Ideal (Ordered from worst to best)

2. **Clarity:**SI1 grade is common with highest frequency. Clarity of the diamond is nothing but free of any intrutions or flaws. Who does not love flawless things? atleast me :) So, lets assume this is super important.

Clarity Categories: I1 , SI2, SI1, VS2, VS1, VVS2, VVS1, IF (Ordered from worst to best)

3. **Color:** G colorscale is common kind of diamond. Surpricingly diamond with no color are extremely rare and regard as highly valuable.

Color Categories: J, I, H, G, F, E, D (Ordered from colored to less color, same is order of rarity)

4. **Carat:** 1 carat diamonds are common, and beyond 1 carat are rare. As caret is a ⚖ parameter, finding a dinmond so heavy is very difficult and rare. It makes some sence. More the carat value, rare the diamond.

5. **Depth%:** It a ratio of the total depth (height of the diamond) to the table of the diamond, so, it is an empirical parmaeter, and it determinds the light prenetrations in to the diamond. As per wikipedia, as the depth increases diamond looks dull and it's value reduces. So, need furthur study. Based on cut shapes, ideal depth % should be in between (55-70)%
6. **Table%:** It is a ratio of the truncated table on top of the diamond to whole diameter avereage. This measure the size of the diamond, does deals with the light beam penetrations as well. Based on cut shapes, ideal Table % should be in between (52-60)%
7. **X,Y,Z :** These are dimensions in mm, Interestingly we have values of 0 as minimum, as these are table and depth parameters, its not practically possible. Based on these three dimensions Depth and Table perentages could be caclulated. There are severe outlier in these 3 columns.
8. **Price :** Most of the dimonds are under 5k USD and outliers are there upto 18k USD. This is target, which is a continous distribution, and lots of paremeters are effecting this price value.

📢 Missing values alert!!!!! Data have missing values hidden in X,Y,Z features as zero values, if anyvalue is zero, then physically diamond don't exit. Looks like he found invincible diamonds in our data. 🛸

After though 📈 of statistics of the data, our 👨 found there are hidden 💎s in the data, and relatively high skewed **(acceptable in range of |3|)** datapoints along with high kurotisis values (acceptable in range of |10|, as noramal distribution has 3) for y,z which indicated fat tail or high outliers . Our 👨 decided to drop 🗑 Those missing values and **apply 3 Standard Deviation** to ✂✂ outliers.

3.3 Setting up Stage for Data Viz: Missing Diamonds, Duplicate Diamonds, And Odd diamonds

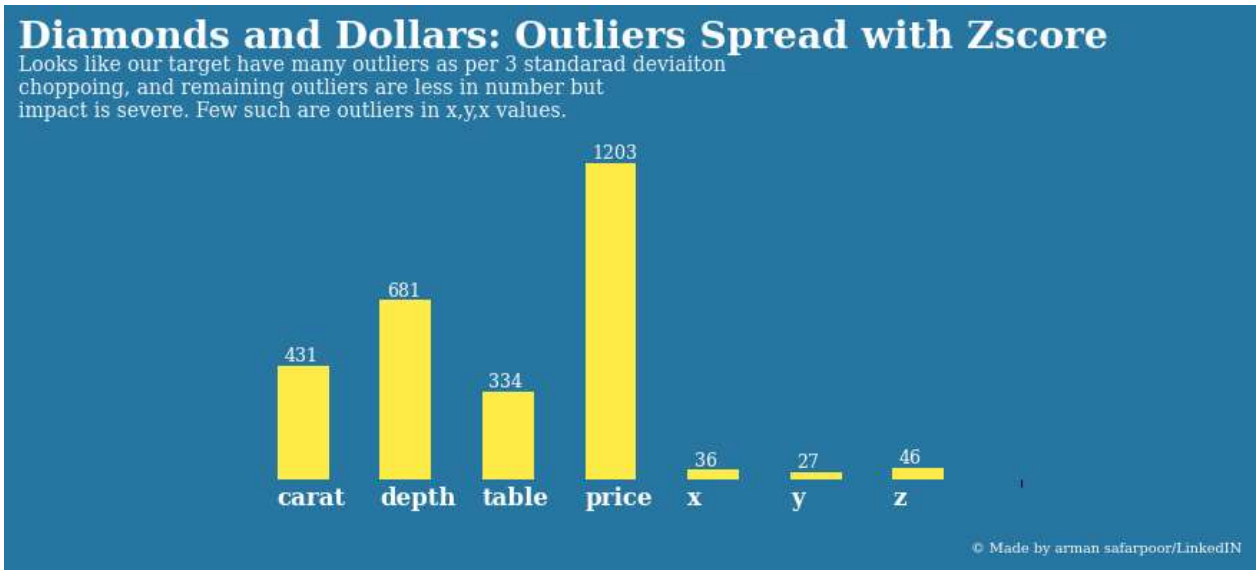
In [10]:
```
### dataframe free of null values

print("Cleaning data....")
print('Shape of Dataset after without any processing: {}'.format(df.shape))
df = df[(df['x'] != 0) | (df['y'] != 0) | (df['z'] != 0)]


print('Shape of Dataset after droping null values: {}'.format(df.shape))

## checking for duplicate values
if df.duplicated().any:
    df = df.drop_duplicates()
    temp3 = df.shape[0]
    print('Duplicates exist,and Shape of Dataset after dropping duplicates: {}'.format(df.s
df.reset_index(drop = True, inplace = True)

orig_df = df.copy() ## lets keep it for futur comparision
```

```
Cleaning data....
Shape of Dataset after without any processing: (53940, 10)
Shape of Dataset after droping null values: (53933, 10)
Duplicates exist,and Shape of Dataset after dropping duplicates: (53788, 10)
```

In [11]:
```
#### zscore dataframe

zscore_df = df.copy().select_dtypes(exclude = 'object') # this is not yet zscore just a cop
obj_col = zscore_df.columns

for col in obj_col:
    zscore_df[col] = np.abs((df[col] - df[col].mean())/df[col].std())


## outliers
std = 3

outliers = df[(zscore_df['x'] > std)| (zscore_df['y'] > std)| (zscore_df['z'] > std)|
              (zscore_df['price'] > std)| (zscore_df['depth'] > std)| (zscore_df['table'] >
df = df.drop(index = outliers.index).reset_index(drop = True)



print('Outliers removing....')
print('Number of Outliers in the dataset are: {}'.format(outliers.shape[0]))
print('Shape of Dataset after removing Outliers: {}'.format(df.shape[0]))
```

```
Outliers removing....
Number of Outliers in the dataset are: 2169
Shape of Dataset after removing Outliers: 51619
```

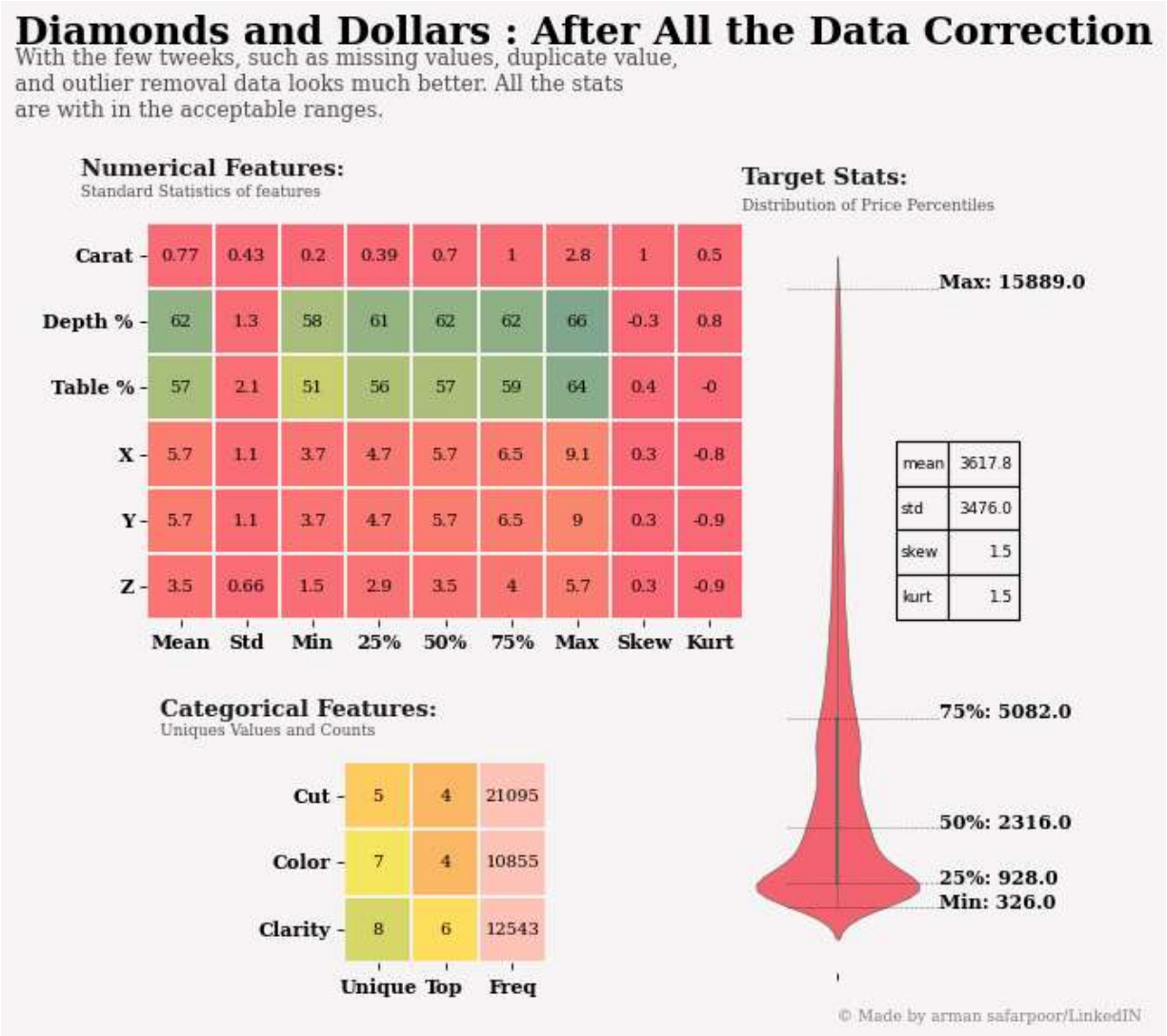
In [12]:
```
temp = zscore_df.copy()

for col in temp.columns:
    temp[col] = temp[col].apply( lambda x: np.nan if x > 3  else x )

nullvalues = temp.isnull().sum()

fig, ax = plt.subplots(figsize = (7,3), dpi = 100)
fig.patch.set_facecolor(colors[2])
ax.set_facecolor(colors[2])

for loc in  ['left','right', 'bottom', 'top']:
    ax.spines[loc].set_visible(False)

ax.axes.get_yaxis().set_visible(False)
#ax.axes.get_xaxis().set_visible(False)

ax.bar(x = nullvalues.index, height = nullvalues.values, width = 0.5, color = colors[1])

for idx,pa in enumerate(ax.patches):
    ax.text(pa.get_x() + pa.get_width()/8, pa.get_height() + 15,
            pa.get_height(),{'font':'serif', 'size':9, 'weight':'normal', 'color':'white'}
    ax.text(pa.get_x() , -100, nullvalues.index[idx],{'font':'serif', 'size':12, 'weight':


fig.text(-0.1,1.11,'Diamonds and Dollars: Outliers Spread with Zscore ',
         {'font':'serif', 'size':19, 'weight':'bold','color':'white'})
fig.text(-0.1,0.95,'Looks like our target have many outliers as per 3 standarad deviaiton

ax.set_xticks(ticks = '')


fig.text(0.85,-0.06,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':7,  'colo
fig.show()
```



In [13]:
```
### Statistics dataframe creation after statistical insights

des_stats = df.describe(exclude = ['object']).drop(index = ['count'], axis = 0).T
skew = []
kurt = []
num_cols = df.select_dtypes(exclude = ['object']).columns
for col in num_cols:
    skew.append(df[col].skew().round(1))
    kurt.append(df[col].kurt().round(1))

stats = pd.DataFrame({'skew':skew,'kurt':kurt}, index = num_cols)

all_stats = pd.merge(left = des_stats,right = stats, left_index = True, right_index = True
```

In [14]:
```
### Stats viualization after statistical insights

fig = plt.figure(figsize =(15,12), dpi = 80)
fig.patch.set_facecolor('#f6f5f5')

gs = fig.add_gridspec(8,8)
gs.update(wspace = 0, hspace = 3)

back_ground = '#f6f5f5'
ax1 = fig.add_subplot(gs[0:3,0:3])
ax2 = fig.add_subplot(gs[0:5,3])
ax3 = fig.add_subplot(gs[3:5,1:2])
ax4 = fig.add_subplot(gs[0:3,3])

axes = [ax1,ax2, ax3]

for ax in axes:
    for loc in ['left','right','top','bottom']:
        ax.spines[loc].set_visible(False)
    ax.set_facecolor('#f6f5f5')


#colormap for visualization
colormap  = matplotlib.colors.LinearSegmentedColormap.from_list("", colors)


#### statistics of numerical features
sns.heatmap(all_stats.drop(index = ['price']),
            annot = True, ax = ax1,
            vmin = -0, vmax = 120,
            square = True, linewidths = 0.09,linecolor = '#f6f5f5',
            cbar = False,
            cmap =colormap, annot_kws={'font':'serif', 'size':9, 'weight':'normal', 'color

#### statistics of target
#ax4 = ax2.twinx()
sns.violinplot(y = df['price'], ax = ax2, palette= [colors[0]],inner = 'box', saturation =
               **{'linecolor' :'black' })


## labeling stats in violin plot
pr = df['price'].describe().drop(index = ['count', 'mean','std'], axis = 0)

for idx,value in pr.items():
    ax2.plot([-0.25,0.5], [value,value], **{'linewidth': 0.5, 'linestyle': '--','color':'bl
    if idx == 'max':
        idx = 'Max'
    elif idx == 'min':
        idx = 'Min'
    else:
        idx = idx
    ax2.text(0.5,value,'{}: {}'.format(idx,value) ,{'font':'serif', 'size':10, 'weight':'bo


ax2.axes.get_yaxis().set_visible(False)

### price stats table
price_stats = all_stats.T['price'].to_frame().round(1)
price_stats = (price_stats.T[['mean', 'std', 'skew', 'kurt']]).T
bbox = [1, 0, 0.35, 0.45]
ax4.table(cellText = price_stats.values, rowLabels = price_stats.index,
          bbox=bbox,cellColours = np.array(['#f6f5f5','#f6f5f5','#f6f5f5','#f6f5f5']).resha
          rowColours = ['#f6f5f5','#f6f5f5','#f6f5f5','#f6f5f5'])
```

```
ax4.axis('off')

#### statistics of categorical features
sx = df.describe(exclude = ['int64', 'float64','float']).drop(index = ['count'], axis = 0)
sx['top']= ['4', '4', '6']
sx = sx.astype(int)

sns.heatmap(sx,annot = True,fmt = '1.0f',vmin = -0, vmax = 20,ax = ax3,
            square = True, linewidths = 0.09,cbar = False,linecolor = '#f6f5f5',
            cmap =colormap, annot_kws={'font':'serif', 'size':9, 'weight':'normal', 'color

#### axis and labeling
ax1.set_yticklabels(labels = ['Carat', 'Depth %', 'Table %','X', 'Y', 'Z'],rotation = 0,**

ax1.set_xticklabels(labels = ['Mean','Std','Min','25%','50%','75%', 'Max','Skew','Kurt'],
                    rotation = 0,**{'font':'serif', 'size':10, 'weight':'bold', 'color':'b

ax3.set_yticklabels(labels = ['Cut', 'Color', 'Clarity'],rotation = 0,**{'font':'serif', '
                                                                         'color':'black'})
ax3.set_xticklabels(labels = ['Unique', 'Top', 'Freq'],rotation = 0,**{'font':'serif', 'si
                                                                       'color':'black'})

### titles
ax1.text(-1., -0.7 , 'Numerical Features:',{'font':'serif', 'size':12, 'weight':'bold','co
ax1.text(-1., -0.4, 'Standard Statistics of features',{'font':'serif', 'size':8, 'color':'

ax2.text(-0.47, 18500 , 'Target Stats: ',{'font':'serif', 'size':12, 'weight':'bold','colo
ax2.text(-0.47, 17850 , 'Distribution of Price Percentiles',{'font':'serif', 'size':8,'col

ax3.text(-2.75, -0.7 , 'Categorical Features:',{'font':'serif', 'size':12, 'weight':'bold'
ax3.text(-2.75, -0.4 , 'Uniques Values and Counts',{'font':'serif', 'size':8,  'color':'bl


ax1.text(-2,-2.7,'Diamonds and Dollars : After All the Data Correction',
         {'font':'serif', 'size':20, 'weight':'bold', 'color':'black'})
ax1.text(-2,-1.6,
         "With the few tweeks, such as missing values, duplicate value, \nand outlier remov
         {'font':'serif', 'size':11,  'color':'black'}, alpha = 0.7)

ax2.text(0,-2500,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':8,  'color':

fig.show()
```



4. Finding a Story in Data: Pattrens and Plots

All the data tweeks are completed from statistical stand poit and lets construct few features for visualization purpose. For binning of the diamonds following articles were refered.

- how to read diamond grading report (https://4cs.gia.edu/files/GIA-how-to-read-a-diamond-grading-report-EN.pdf)
- What makes a diamond rare? (https://www.jewelry-secrets.com/Blog/diamond-rarity-is-a-factor-in-price/)

In [15]:
```
### binnining and feature engineering



df_backup = df.copy()


####################### Rarity of the diamond
c1 = [(df['carat'] > 1.5)] ## rare
c2 = [(df['carat'] < 1) | ((df['cut'] == 'Ideal') | (df['cut'] == 'Premium') |
                           (df['cut'] == 'Very Good') | (df['cut'] == 'Good')) ] ## common
c5 = [(df['carat'] < 0.5)] ## very common

## rare
c3 = [(df['carat'] > 0.50) & (df['carat'] <1.0) & ((df['clarity'] == 'VVSI') |
                                                   (df['clarity'] == 'IF')) & ((df['color']

## ultra rare
c4 = [(df['carat'] > 1) & ( (df['clarity'] == 'VVS1')|(df['clarity'] == 'IF')) &
      ((df['color'] == 'D') | (df['color'] == 'E')) & ((df['cut'] == 'Ideal')) ]

df.loc[c2[0], 'rarity'] = 'Common'
df.loc[c5[0], 'rarity'] = 'Very Common'
df.loc[c1[0], 'rarity'] = 'Rare'
df.loc[c3[0], 'rarity'] = 'Rare'
df.loc[c4[0], 'rarity'] = 'Ultra Rare'

df = df.fillna('Common')


############## size of the diamond
df['diamond_size'] = pd.cut(df['carat'], bins = [0,0.5,1,2,3], labels = ['Tiny','Small','N
df['diamond_size'] = df['diamond_size'].astype('object')

########shape of the diamond

df['shape'] = (np.abs(df['x'] - df['y'])).apply(lambda x: 'Regualr' if x <=0.03 else 'Fancy


######## Volume of the diamond

df['Volume'] = df['x']*df['y']*df['z']
```

4.1 Univariate Numerical Feature Distributions

In [16]:
```
### univariate ananalysis


fig =  plt.figure(figsize = (14,8.5), dpi = 85)
fig.patch.set_facecolor('#f5f6f6')



                                                    ##### Note to reader I could have used
                                                           #spec for more controlover plots
gs = fig.add_gridspec(3,3)
gs.update(wspace =0.2,hspace = 0.2)


ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])
ax2 = fig.add_subplot(gs[0,2])

ax3 = fig.add_subplot(gs[1,0])
ax4 = fig.add_subplot(gs[1,1])
ax5 = fig.add_subplot(gs[1,2])


ax6 = fig.add_subplot(gs[2,0])
ax7 = fig.add_subplot(gs[2,1])
ax8 = fig.add_subplot(gs[2,2])



axes = [ax0,ax1,ax2,ax3, ax4,ax5,ax6,ax7,ax8]


for ax in axes:
    ax.set_facecolor('#f5f6f6')
    ax.tick_params(axis='x',
                   labelsize = 12, which = 'major',
                   direction = 'out',pad = 2,
                   length = 1.5)
    ax.tick_params(axis='y', colors= 'black')
    ax.axes.get_yaxis().set_visible(False)

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

cols = df.select_dtypes(exclude = 'object').columns

### ax0- carat

sns.kdeplot(x = df[cols[0]],fill = True, color = colors[0], alpha = 1, ax = ax0)
sns.kdeplot(x = df[cols[0]],color = colors[1], alpha = 1, ax = ax0)
ax0.set_xlabel(xlabel = '')
ax0.text((df[cols[0]].max() ), 1.4,'Carat', **{'font':'serif', 'size':18,'weight':'bold'},




### ax1 - depth
sns.kdeplot(x = df[cols[1]], color = colors[0], alpha = 1, ax = ax1)
sns.kdeplot(x = df[cols[1]], fill = True, color = colors[1], alpha = 1, ax = ax1)
ax1.set_xlabel(xlabel = '')
ax1.text((df[cols[1]].max()-2 ), 0.35,'Depth %', **{'font':'serif', 'size':18,'weight':'bo
```

```
### ax2 -- table
sns.kdeplot(x = df[cols[2]],color = colors[0], alpha = 1, ax = ax2)
sns.kdeplot(x = df[cols[2]], fill = True, color = colors[1], alpha = 1, ax = ax2)

ax2.set_xlabel(xlabel = '')
ax2.text((df[cols[2]].max()), 0.25,'Table %', **{'font':'serif', 'size':18,'weight':'bold'}
```

```
### ax3 --- price
sns.kdeplot(x = df[cols[3]],  color = colors[0], alpha = 1, ax = ax3)
sns.kdeplot(x = df[cols[3]], fill = True, color = colors[1], alpha = 1, ax = ax3)
ax3.set_xlabel(xlabel = '')
ax3.text((df[cols[3]].max()), 0.00025,'Price', **{'font':'serif', 'size':18,'weight':'bold
```

```
### ax4 --- free space

ax4.axes.get_xaxis().set_visible(False)
ax4.spines['bottom'].set_visible(False)
ax4.text(-0.01,0.25,
         'X,y,z are physical dimensions of diamond, \nwith carat as weight prameter. Volume
          **{'font':'serif', 'size':11.5,'weight':'bold'}, alpha = 0.9 )
```

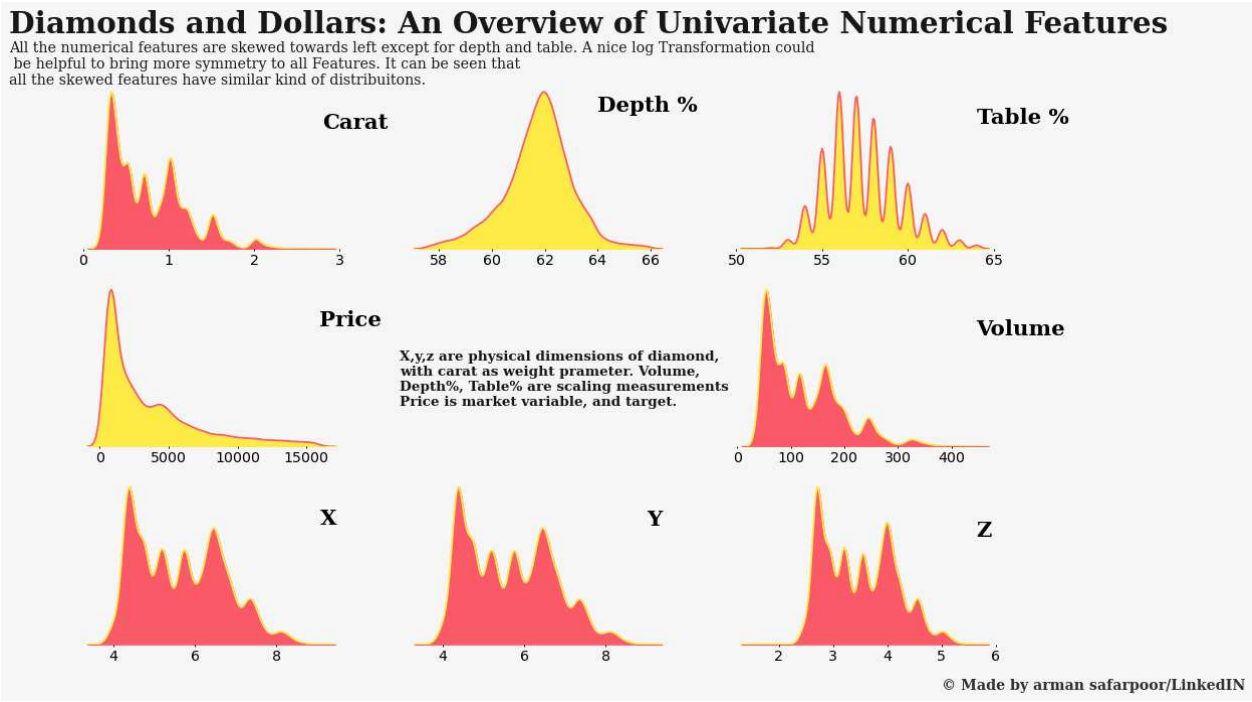
```
### ax5 --- Volume
sns.kdeplot(x = df[cols[7]], fill = True, color = colors[0], alpha = 1, ax = ax5)
sns.kdeplot(x = df[cols[7]],  color = colors[1], alpha = 1, ax = ax5)
ax5.set_xlabel(xlabel = '')
ax5.text((df[cols[7]].max()), 0.008,'Volume', **{'font':'serif', 'size':18,'weight':'bold'}
```

```
### ax6 ---- X
sns.kdeplot(x = df[cols[4]], fill = True, color = colors[0], alpha = 1, ax = ax6)
sns.kdeplot(x = df[cols[4]], color = colors[1], alpha = 1, ax = ax6)
ax6.set_xlabel(xlabel = '')
ax6.text((df[cols[4]].max()), 0.4,'X', **{'font':'serif', 'size':18,'weight':'bold'}, alpha
```

```
### ax7 --- Y
sns.kdeplot(x = df[cols[5]], fill = True, color = colors[0], alpha = 1, ax = ax7)
sns.kdeplot(x = df[cols[5]], color = colors[1], alpha = 1, ax = ax7)
ax7.set_xlabel(xlabel = '')
ax7.text((df[cols[5]].max() ), 0.4,'Y', **{'font':'serif', 'size':18,'weight':'bold'}, alpl
```

```
### ax8 ---- z
sns.kdeplot(x = df[cols[6]], fill = True, color = colors[0], alpha = 1, ax = ax8)
sns.kdeplot(x = df[cols[6]],  color = colors[1], alpha = 1, ax = ax8)
ax8.set_xlabel(xlabel = '')
ax8.text((df[cols[6]].max()), 0.6,'Z', **{'font':'serif', 'size':18,'weight':'bold'}, alpha

ax8.text(5.,-0.25,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':12, 'weight
```

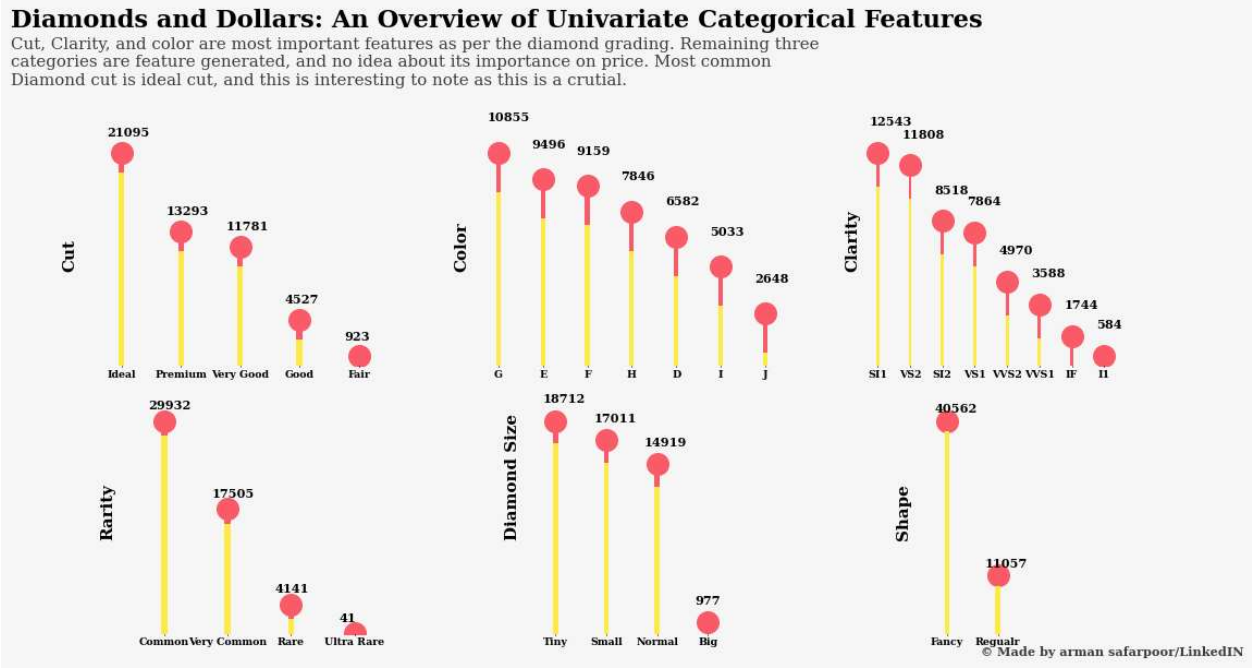
```
fig.text(0.07,0.95,'Diamonds and Dollars: An Overview of Univariate Numerical Features' ,
         **{'font':'serif', 'size':25,'weight':'bold',}, alpha = 0.9)

fig.text(0.07,0.88,'All the numerical features are skewed towards left except for depth and

fig.show()
```



4.2 Univariate Categorical Feature Distributions

In [17]:
```
### univariate ananalysis


fig =  plt.figure(figsize = (15,6.5), dpi = 90)
fig.patch.set_facecolor('#f5f6f6')



                                                ##### Note to reader I could have used
                                                      #spec for more controlover plots
gs = fig.add_gridspec(2,3)
gs.update(wspace =0.1,hspace = 0.2)


ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])
ax2 = fig.add_subplot(gs[0,2])

ax3 = fig.add_subplot(gs[1,0])
ax4 = fig.add_subplot(gs[1,1])
ax5 = fig.add_subplot(gs[1,2])




axes = [ax0,ax1,ax2,ax3, ax4,ax5,]


for ax in axes:
    ax.set_facecolor('#f5f6f6')
    ax.tick_params(axis='x',
                   labelsize = 1, which = 'major',
                   direction = 'out',pad = 2,
                   length = 1)
    ax.tick_params(axis='y', colors= 'black')
    ax.axes.get_yaxis().set_visible(False)
    ax.axes.get_xaxis().set_visible(True)

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

cols = df.select_dtypes(exclude = ['int64','float64','float']).columns
labels = ['Cut', 'Color', 'Clarity', 'Rarity', 'Diamond Size', 'Shape']



### ax0- carat
s = 250

for col,ax,label in zip(cols,axes,labels):
    ax.bar(x = df[col].value_counts().index, height = df[col].value_counts().values, width
    ax.scatter(x = df[col].value_counts().index, y = df[col].value_counts().values, s = s,
    ht = df[col].value_counts().values.max()
    ax.text(-1,ht/2.2 ,label,**{'font':'serif', 'size':12, 'weight':'bold', 'rotation' : '
    ax.set_xticklabels(df[col].value_counts().index , rotation = 0,**{'font':'serif', 'size

    for pa in ax.patches:
        ax.text((pa.get_x() - 2*pa.get_width() ), pa.get_height()+1600, pa.get_height(),
                **{'font':'serif', 'size':9, 'weight':'bold',}, alpha = 1)
    height = [ val - 2000 if (val - 2000) > 0 else 0 for val in df[col].value_counts().valu
    ax.bar(x = df[col].value_counts().index, height = height , width = 0.1, color = colors
```

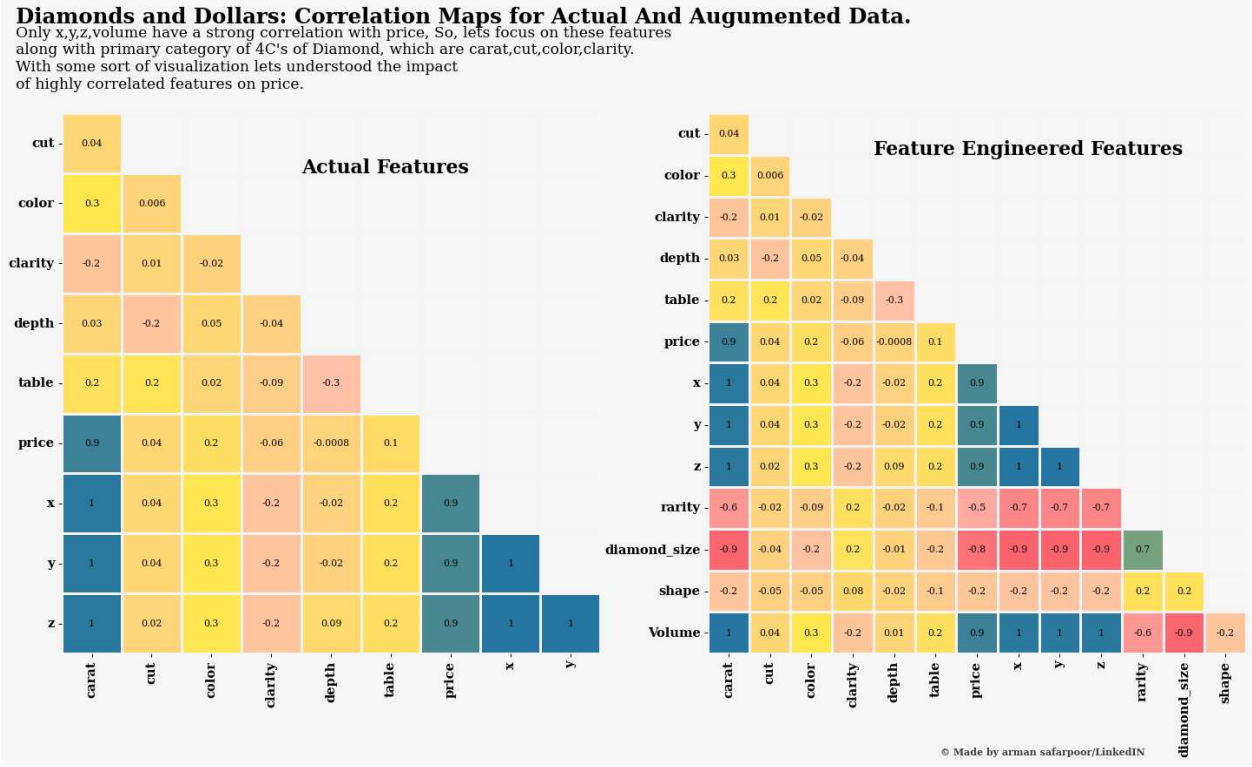
```
fig.text(0.09,1.06,'Diamonds and Dollars: An Overview of Univariate Categorical Features'
         **{'font':'serif', 'size':18,'weight':'bold', }, alpha = 1)

fig.text(0.09,0.97,'''Cut, Clarity, and color are most important features as per the diamor
categories are feature generated, and no idea about its importance on price. Most common
Diamond cut is ideal cut, and this is interesting to note as this is a crutial.''' ,
         **{'font':'serif', 'size':12,'weight':'normal',}, alpha = 0.75)


ax0.set_xlim(-1,5)
ax1.set_xlim(-1,7)
ax2.set_xlim(-1,10)
ax3.set_xlim(-1.6,4)
ax4.set_xlim(-2,5)
ax5.set_xlim(-2,5)


fig.text(0.75,0.075,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':9, 'weigh

fig.show()
```



4.3 Overall Correlationship Maps For Features

In [18]:
```
actualdf = df_backup.copy()

featdf = df.copy()

for col in actualdf.select_dtypes(exclude = ['int','float','float64','int64']).columns:
    le = LabelEncoder()
    actualdf[col] = le.fit_transform(actualdf[col])

for col in featdf.select_dtypes(exclude = ['int','float','float64','int64']).columns:
    le = LabelEncoder()
    featdf[col] = le.fit_transform(featdf[col])




## actual correlations
act_corr = actualdf.corr()
mask1 = np.triu(np.ones_like(act_corr, dtype=np.bool))
mask1 = mask1[1:, :-1]
act_corr = act_corr.iloc[1:,:-1].copy()

## featured correlations
fea_corr = featdf.corr()
mask2 = np.triu(np.ones_like(fea_corr, dtype=np.bool))
mask2 = mask2[1:, :-1]
fea_corr = fea_corr.iloc[1:,:-1].copy()

cust_colors = ['#FB5B68','#FFBDB0','#FFEB48','#2676A1',]
colormap = matplotlib.colors.LinearSegmentedColormap.from_list("",cust_colors)

fig, ax = plt.subplots(1,2, figsize = (20,10), dpi = 88)
fig.patch.set_facecolor('#f5f6f6')

axes = ax.ravel()

for ax in axes:
    ax.set_facecolor('#f5f6f6')


### actual heatmap
sns.heatmap(data = act_corr, ax = axes[0], vmin= -1,vmax = 1 ,
            annot = True, fmt = '1.0g', annot_kws={'font':'serif', 'size':9, 'weight':'nor
            square = True, linewidth = 1, linecolor = '#f6f5f5',
             cmap = colormap, mask = mask1, cbar = False,alpha = 1)


### featured heatmap
sns.heatmap(data = fea_corr, ax = axes[1], vmin = -1, vmax = 1,
            annot = True, fmt = '1.0g',annot_kws={'font':'serif', 'size':9, 'weight':'norma
            square = True, linewidth = 1, linecolor = '#f6f5f5',
             cmap = colormap, mask = mask2, cbar = False, alpha = 1)


for ax in axes:
    ax.set_xticklabels(ax.get_xticklabels(), rotation =90,**{'font':'serif', 'size':12, 'w
    ax.set_yticklabels(ax.get_yticklabels(), rotation =0,**{'font':'serif', 'size':12, 'we


axes[0].text(4, 1, 'Actual Features',{'font':'serif', 'size':18, 'weight':'bold', 'color':
axes[1].text(4,1, 'Feature Engineered Features',{'font':'serif', 'size':18, 'weight':'bold


fig.text(0.095,0.965, 'Diamonds and Dollars: Correlation Maps for Actual And Augumented Dat
```

```
        {'font':'serif', 'size':20, 'weight':'bold', 'color':'black'})
fig.text(0.095,0.88, '''Only x,y,z,volume have a strong correlation with price, So, lets fo
along with primary category of 4C's of Diamond, which are carat,cut,color,clarity.
With some sort of visualization lets understood the impact
of highly correlated features on price.''',{'font':'serif', 'size':14, 'weight':'normal',


fig.text(0.7,0.01,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':9, 'weight'


fig.show()
```



4.4 4C's and Dimensions of Diamond: Price Comparions

In [19]:
```
def custom_scatter(x,y,value, size = None,figsize = None, ax = None, marker = None, color =

    x_labels = [val for val in sorted(x.unique())]
    y_labels  = [val for val in sorted(y.unique())]

    x_to_num = {val:idx for idx,val in enumerate(x_labels)}
    y_to_num = {val:idx for idx,val in enumerate(y_labels)}


    x_values = x.map(x_to_num)
    y_values = y.map(y_to_num)


    if size == None:
        size = 1
    if ax == None:
        fig,ax = plt.subplots(figsize = figsize)
    if marker == None:
        marker = None
    else:
        marker = marker

    ax.scatter(x = x_values,y = y_values, s = value * size, c = color, marker = marker)


    ax.set_xticks(ticks = x_values.unique())
    ax.set_yticks(ticks = y_values.unique())

    ax.set_xticklabels(labels = x_labels,**{'font':'serif', 'size':12,'weight':'bold'}, al
    ax.set_yticklabels(labels = y_labels,**{'font':'serif', 'size':12,'weight':'bold'}, al

    ax.set_xlim(x_values.min()-0.5, x_values.max()+0.5)
    ax.set_ylim(y_values.min()-0.5, y_values.max()+0.5)



    ax.grid(False, 'major')

    ax.grid(False, 'minor', )
    ax.set_xticks([t + 0.5 for t in ax.get_xticks()], minor=True)
    ax.set_yticks([t + 0.5 for t in ax.get_yticks()], minor=True)


    #ax.set_xlim([-0.5, max([v for v in x_to_num.values()]) + 0.5])
    #ax.set_ylim([-0.5, max([v for v in y_to_num.values()]) + 0.5])

    return None
```

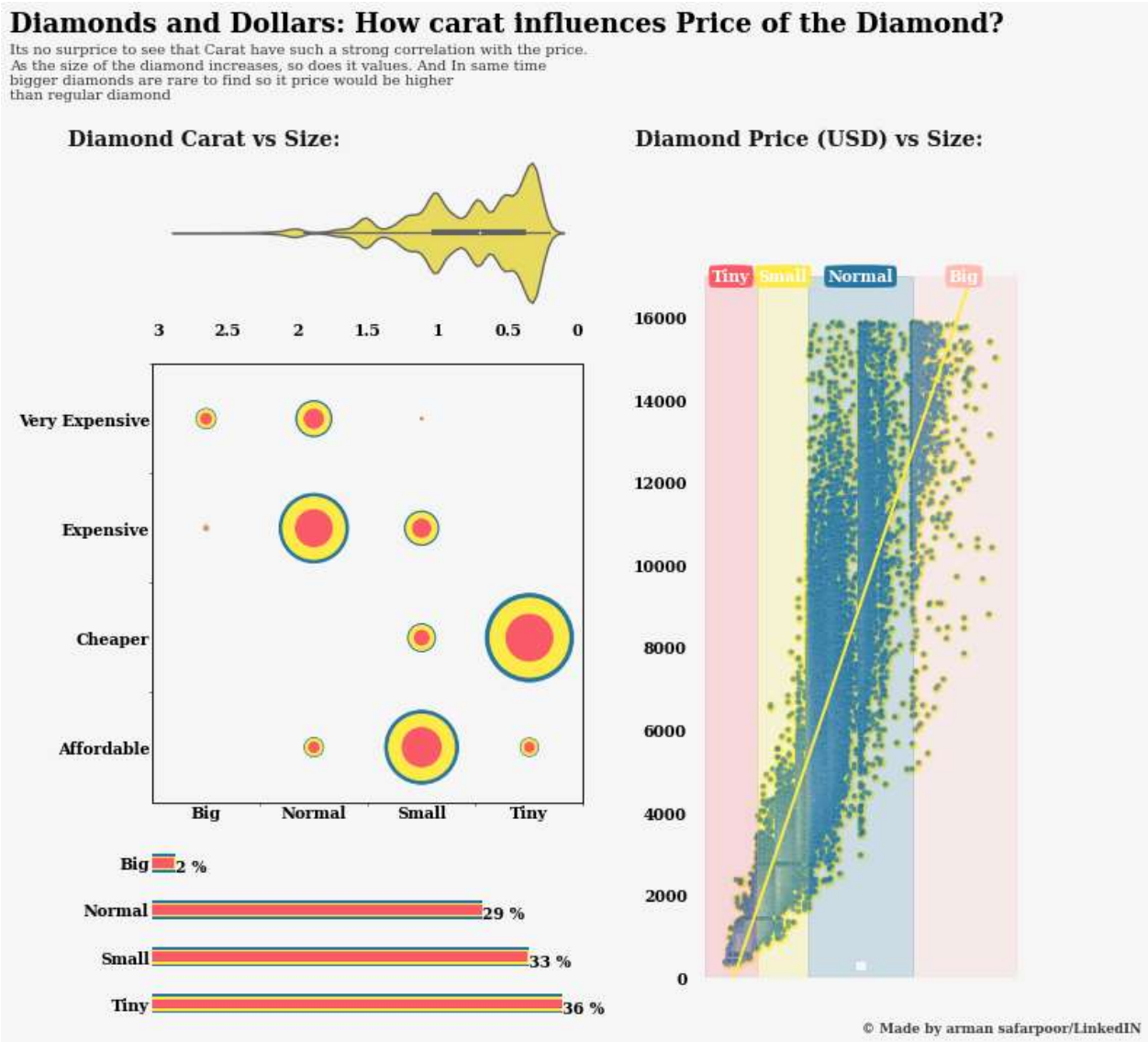
1. Carat of The Diamond

In [20]:

```
tem = df.copy()
tem['price_cat'] = pd.cut(df['price'], bins = [0,1500,4000,10000,100000], labels = ['Cheap
tem['price_cat'] = tem['price_cat'].astype('object')


tab = pd.crosstab(tem['diamond_size'],tem['price_cat'])
tab_val = pd.melt(tab.reset_index(),id_vars = 'diamond_size')
tab_val.columns = ['x','y','value']


fig = plt.figure(figsize = (12,12), dpi = 65)
fig.patch.set_facecolor('#f5f6f6')
gs = fig.add_gridspec(20,20)
gs.update(wspace = 0.6, hspace = 0)

ax0 = fig.add_subplot(gs[0:,0:])
ax0.set_facecolor('#f5f6f6')
ax0.axes.get_xaxis().set_visible(False)
ax0.axes.get_yaxis().set_visible(False)
for loc in ['left', 'right', 'top', 'bottom']:
        ax0.spines[loc].set_visible(False)



ax1 = fig.add_subplot(gs[0:4,0:10])

ax2 = fig.add_subplot(gs[3:19,12:20])

ax3 = fig.add_subplot(gs[5:15,0:10])

ax4 = fig.add_subplot(gs[16:20,0:10])

axes = [ax1,ax2,ax3,ax4]

for ax in axes:
    ax.set_facecolor('#f5f6f6')
    ax.tick_params(axis='both',
                   labelsize = 12, which = 'major',
                   direction = 'out',pad = 2,
                   length = 0.001)

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_linewidth(1)


################################################violin plot
sns.violinplot(x = -df['carat'],  ax = ax1, color = colors[1], alpha = 1, linecolor = 'bla


## violin
ticks = [3.5,3,2.5,2,1.5,1,0.5,0]
ax1.set_xticklabels(labels = ticks, **{'font':'serif','size':12, 'weight':'bold'},alpha = 1
ax1.set_xlabel('')
for loc in ['left', 'right', 'top', 'bottom']:
    ax1.spines[loc].set_visible(False)


################################################ scatter plot and regplot
sns.regplot(x = 'carat', y = 'price',data = tem, ax = ax2, color = colors[1],)
sns.scatterplot(x = 'carat', y = 'price',data = tem, ax = ax2, alpha = 0.7,size = 0.5, col

ax2.legend(labels = [])
```

```
ax2.set_yticklabels(np.arange(0,18000,2000),**{'font':'serif','size':12, 'weight':'bold'},a
ax2.set_xticklabels('')
ax2.set_ylabel('')
ax2.set_xlabel('')
for loc in ['left', 'right', 'top', 'bottom']:
    ax2.spines[loc].set_visible(False)


## generating spans for diamond sizes
## www.kaggle.com/subinium/simple-matplotlib-visualization-tips/  --- #thanks to @subinium

bin_labels = ['Tiny','Small','Normal','Big']
size_bins = [[0, 0.5], [0.5,  1], [1, 2], [2, 3]]

for idx, label in enumerate(bin_labels):
        ax2.annotate(label,
                    xy=(sum(size_bins[idx])/2 ,17000),
                    xytext=(0,0), textcoords='offset points',
                    va="center", ha="center",
                    **{'font':'serif','size':12, 'weight':'bold','color':'white'},
                    bbox=dict(boxstyle='round4', pad=0.2, color=colors[idx], alpha=1))
        ## adding span over region
        ax2.axvspan(size_bins[idx][0],size_bins[idx][1], ymax = 1, color=colors[idx], alpha

ax2.set_ylim(0,17000)

########################################################## scattermap
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'], size = 0.25, a
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'], size = 0.2, ax
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'], size = 0.07, a
ax3.set_facecolor('#f5f6f6')


######################################################### final ratio plot
dia_size = round(tem.diamond_size.value_counts(normalize = True) * 100,0).astype(int)

ax4.barh(y = dia_size.index, width = dia_size.values, height = 0.4, color = colors[2])



for pa in ax4.patches:
        ax4.text((pa.get_width()), pa.get_y(),'{} %'.format(pa.get_width()), **{'font':'se

ax4.barh(y = dia_size.index, width = dia_size.values, height = 0.3, color = colors[1])
ax4.set_yticklabels(labels = dia_size.index,**{'font':'serif', 'size':12,'weight':'bold'},



ax4.barh(y = dia_size.index, width = dia_size.values -00.1, height = 0.2, color = colors[0
for loc in ['left', 'right', 'top', 'bottom']:
    ax4.spines[loc].set_visible(False)
ax4.axes.get_xaxis().set_visible(False)




### titles and descriptions

fig.text(0,0.98, 'Diamonds and Dollars: How carat influences Price of the Diamond?',{'font


fig.text(0,0.92,'''Its no surprice to see that Carat have such a strong correlation with th
```

```
As the size of the diamond increases, so does it values. And In same time
bigger diamonds are rare to find so it price would be higher
than regular diamond''',{'font':'serif', 'size':11,'weight':'normal'}, alpha = 0.8)


fig.text(0.05, 0.88, 'Diamond Carat vs Size:',{'font':'serif', 'size':16,'weight':'bold'},


fig.text(0.55, 0.88, 'Diamond Price (USD) vs Size:',{'font':'serif', 'size':16,'weight':'b

fig.text(0.75,0.1,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':10, 'weight
fig.show()
```

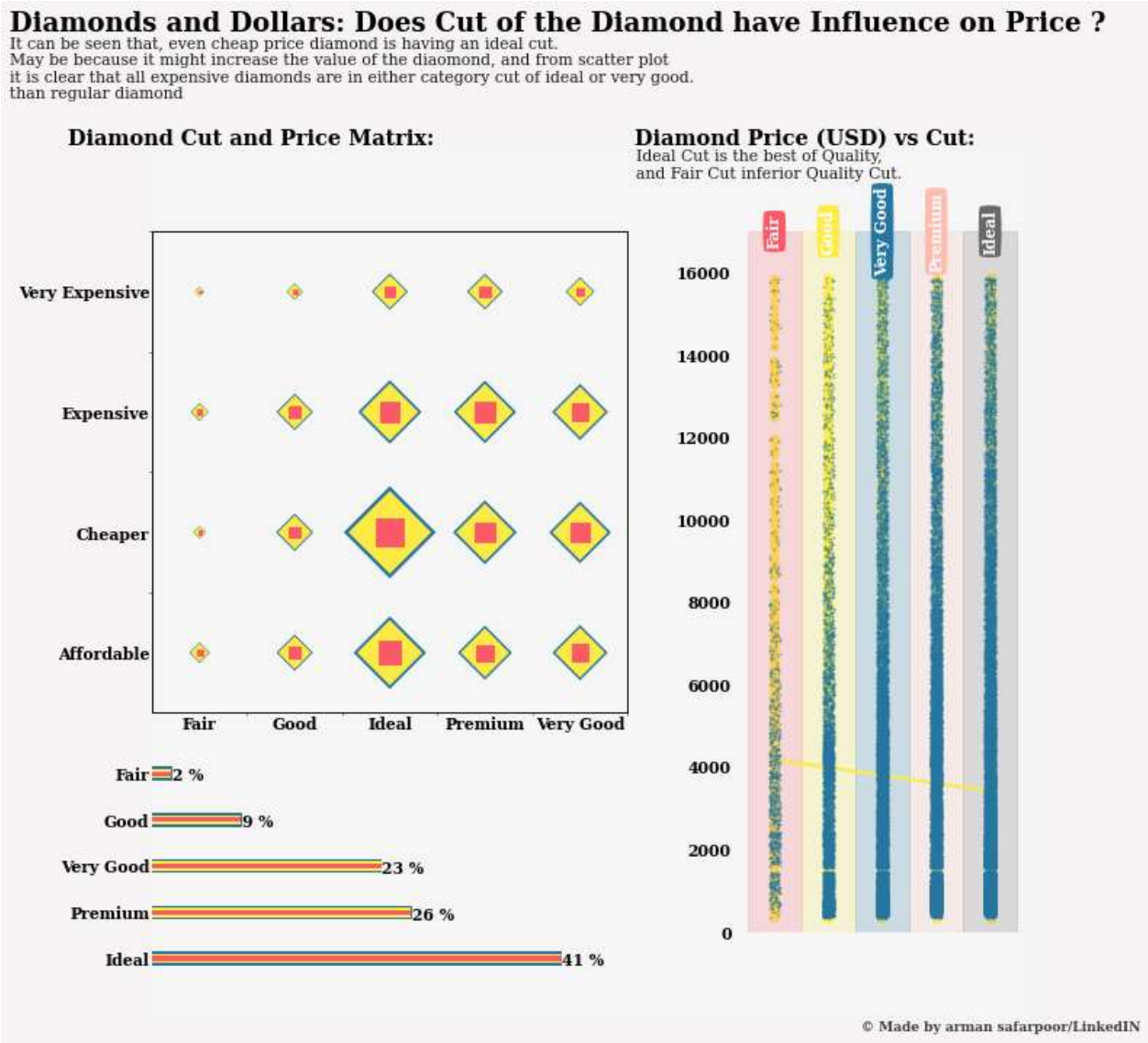


2. Cut of The Diamond

In [21]:
```
tem = df.copy()
tem['price_cat'] = pd.cut(df['price'], bins = [0,1500,4000,10000,100000],
                          labels = ['Cheaper','Affordable', 'Expensive', 'Very Expensive'])
tem['price_cat'] = tem['price_cat'].astype('object')


tab = pd.crosstab(tem['cut'],tem['price_cat'])
tab_val = pd.melt(tab.reset_index(),id_vars = 'cut')
tab_val.columns = ['x','y','value']

cut_labels = ['Fair', 'Good', 'Very Good', 'Premium', 'Ideal']
cut_to_num = { val : idx for idx,val in enumerate(cut_labels)}

tem['cut'] = tem['cut'].map(cut_to_num)


fig = plt.figure(figsize = (12,12), dpi = 65)
fig.patch.set_facecolor('#f6f5f5')
gs = fig.add_gridspec(20,20)
gs.update(wspace = 0.6, hspace = 0)

ax0 = fig.add_subplot(gs[0:,0:])
ax0.set_facecolor('#f5f6f6')
ax0.axes.get_xaxis().set_visible(False)
ax0.axes.get_yaxis().set_visible(False)
for loc in ['left', 'right', 'top', 'bottom']:
        ax0.spines[loc].set_visible(False)


#ax1 = fig.add_subplot(gs[0:4,0:10])

ax2 = fig.add_subplot(gs[2:18,13:20])

ax3 = fig.add_subplot(gs[2:13,0:11])

ax4 = fig.add_subplot(gs[14:19,0:10])

axes = [ax1,ax2,ax3,ax4]

for ax in axes:
    ax.set_facecolor('#f5f6f6')
    ax.tick_params(axis='both',
                   labelsize = 12, which = 'major',
                   direction = 'out',pad = 2,
                   length = 0.001)

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_linewidth(1)




################################################ scatter plot and regplot
sns.regplot(x = 'cut', y = 'price',data = tem, ax = ax2, color = colors[1])
sns.stripplot(x = 'cut', y = 'price',data = tem, ax = ax2, alpha = 0.5, color = colors[2],
ax2.set_yticklabels(np.arange(0,18000,2000),**{'font':'serif','size':12, 'weight':'bold'},
ax2.set_xticklabels('')
ax2.set_ylabel('')
ax2.set_xlabel('')
for loc in ['left', 'right', 'top', 'bottom']:
    ax2.spines[loc].set_visible(False)
```

```
bin_labels = ['Fair', 'Good', 'Very Good', 'Premium', 'Ideal']
size_bins = [[-0.5,0.5],[0.5, 1.5], [1.5,  2.5], [2.5, 3.5], [3.5, 4.5]]

col_here = ['#FB5B68','#FFEB48','#2676A1','#FFBDB0','dimgrey']
for idx, label in enumerate(bin_labels):
        ax2.annotate(label,
                     xy=(sum(size_bins[idx])/2 ,17000),
                     xytext=(0,0), textcoords='offset points',
                     va="center", ha="center",rotation = 90,
                     **{'font':'serif','size':12, 'weight':'bold','color':'white'},
                     bbox=dict(boxstyle='round4', pad=0.2, color=col_here[idx], alpha=1))
        ## adding span over region
        ax2.axvspan(size_bins[idx][0],size_bins[idx][1], ymax = 1, color=col_here[idx], al

ax2.set_ylim(0,17000)


########################################################## scattermap

custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'], marker = 'D',
               size = 0.25, ax = ax3, color = colors[2])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'], marker = 'D',
               size = 0.2, ax = ax3, color = colors[1])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'], marker = 's',
               size = 0.05,ax = ax3, color = colors[0])
ax3.set_facecolor('#f5f6f6')


######################################################### final ratio plot

cut = round(df.cut.value_counts(normalize = True) * 100,0).astype(int)

ax4.barh(y = cut.index, width = cut.values, height = 0.3, color = colors[2])



for pa in ax4.patches:
        ax4.text((pa.get_width()), pa.get_y(),'{} %'.format(pa.get_width()),
                 **{'font':'serif', 'size':12, 'weight':'bold'}, alpha = 1)

ax4.set_yticklabels(labels = cut.index,**{'font':'serif', 'size':12,'weight':'bold'}, alpha


ax4.barh(y = cut.index, width = cut.values -00.1, height = 0.2, color = colors[1])
ax4.barh(y = cut.index, width = cut.values -00.1, height = 0.1, color = colors[0])
for loc in ['left', 'right', 'top', 'bottom']:
    ax4.spines[loc].set_visible(False)
ax4.axes.get_xaxis().set_visible(False)




### titles and descriptions

fig.text(0,0.98, 'Diamonds and Dollars: Does Cut of the Diamond have Influence on Price ?'
         {'font':'serif', 'size':20.,'weight':'bold'}, alpha = 1)

fig.text(0,0.92,'''It can be seen that, even cheap price diamond is having an ideal cut.
May be because it might increase the value of the diaomond, and from scatter plot
it is clear that all expensive diamonds are in either category cut of ideal or very good.
than regular diamond''',{'font':'serif', 'size':12,'weight':'normal'}, alpha = 0.9)
```

```
fig.text(0.05, 0.88, 'Diamond Cut and Price Matrix:',{'font':'serif', 'size':16,'weight':'

fig.text(0.55, 0.88, 'Diamond Price (USD) vs Cut:',{'font':'serif', 'size':16,'weight':'bo

fig.text(0.552, 0.85, 'Ideal Cut is the best of Quality, \nand Fair Cut inferior Quality Cu
        {'font':'serif', 'size':12,'weight':'normal'}, alpha = 0.9)

fig.text(0.75,0.1,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':10, 'weight
fig.show()
```

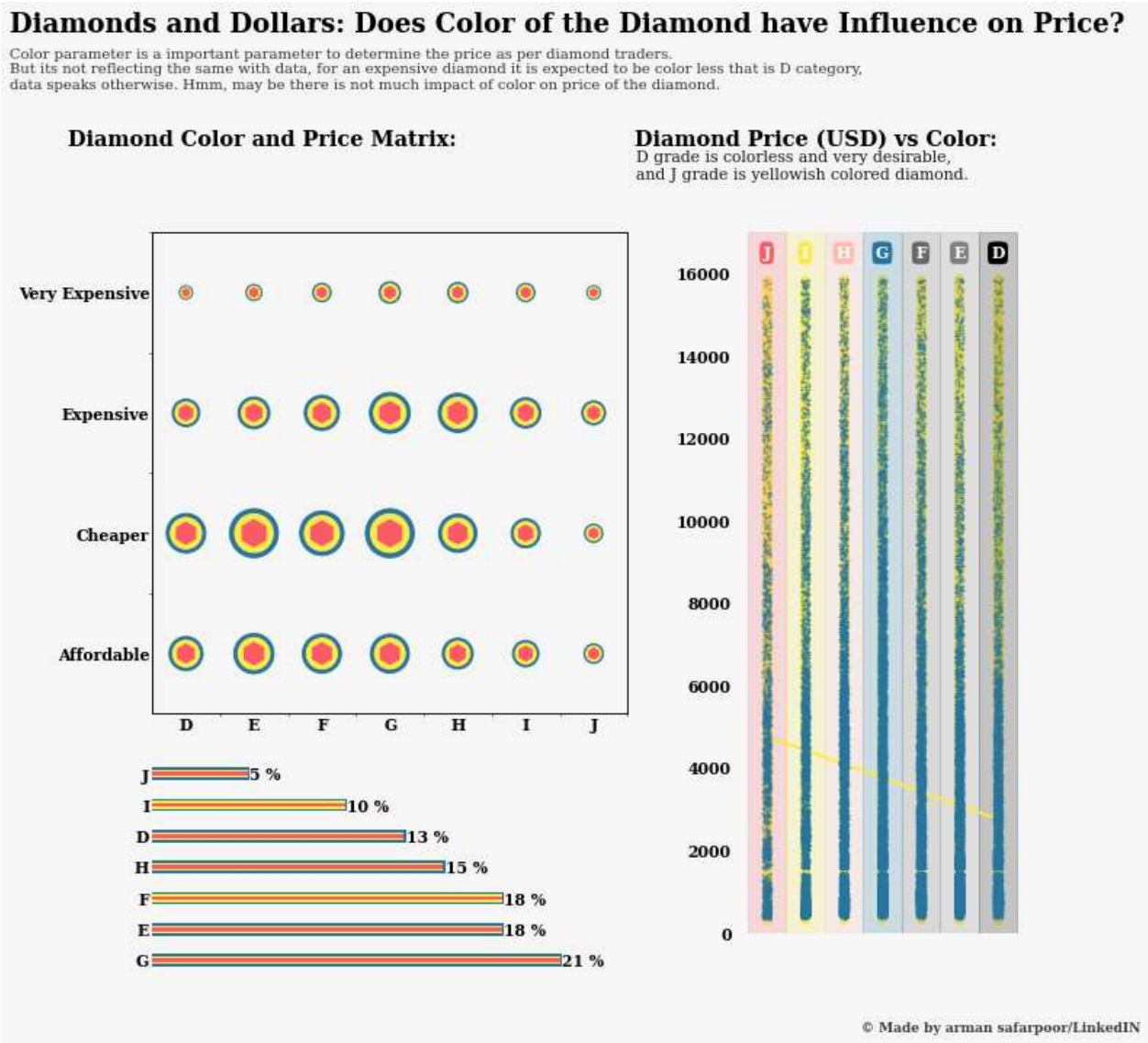


3. Color of The Diamond

In [22]:
```
tem = df.copy()
tem['price_cat'] = pd.cut(df['price'], bins = [0,1500,4000,10000,100000],
                          labels = ['Cheaper','Affordable', 'Expensive', 'Very Expensive'])
tem['price_cat'] = tem['price_cat'].astype('object')


tab = pd.crosstab(tem['color'],tem['price_cat'])
tab_val = pd.melt(tab.reset_index(),id_vars = 'color')
tab_val.columns = ['x','y','value']

color_labels = [ 'J', 'I', 'H', 'G', 'F', 'E', 'D']
color_to_num = { val : idx for idx,val in enumerate(color_labels)}

tem['color'] = tem['color'].map(color_to_num)


fig = plt.figure(figsize = (12,12), dpi = 65)
fig.patch.set_facecolor('#f5f6f6')
gs = fig.add_gridspec(20,20)
gs.update(wspace = 0.6, hspace = 0)


ax0 = fig.add_subplot(gs[0:,0:])
ax0.set_facecolor('#f5f6f6')
ax0.axes.get_xaxis().set_visible(False)
ax0.axes.get_yaxis().set_visible(False)
for loc in ['left', 'right', 'top', 'bottom']:
        ax0.spines[loc].set_visible(False)


#ax1 = fig.add_subplot(gs[0:4,0:10])

ax2 = fig.add_subplot(gs[2:18,13:20])

ax3 = fig.add_subplot(gs[2:13,0:11])

ax4 = fig.add_subplot(gs[14:19,0:10])

axes = [ax1,ax2,ax3,ax4]

for ax in axes:
    ax.set_facecolor('#f5f6f6')
    ax.tick_params(axis='both',
                   labelsize = 12, which = 'major',
                   direction = 'out',pad = 2,
                   length = 0.001)

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_linewidth(1)




################################################ scatter plot and regplot
sns.regplot(x = 'color', y = 'price',data = tem, ax = ax2, color = colors[1])
sns.stripplot(x = 'color', y = 'price',data = tem, ax = ax2,  color = colors[2], size = 2,
ax2.set_yticklabels(np.arange(0,18000,2000),**{'font':'serif','size':12, 'weight':'bold'},
ax2.set_xticklabels('')
ax2.set_ylabel('')
ax2.set_xlabel('')
for loc in ['left', 'right', 'top', 'bottom']:
    ax2.spines[loc].set_visible(False)
```

```
bin_labels = ['J', 'I', 'H', 'G', 'F', 'E', 'D']
size_bins = [[-0.5,0.5],[0.5, 1.5], [1.5,  2.5], [2.5, 3.5], [3.5, 4.5],[4.5,5.5],[5.5,6.5

color = ['#FB5B68','#FFEB48','#FFBDB0','#2676A1','dimgrey', 'grey','black']
for idx, label in enumerate(bin_labels):
        ax2.annotate(label,
                     xy=(sum(size_bins[idx])/2 ,16500),
                     xytext=(0,0), textcoords='offset points',
                     va="center", ha="center",rotation = 0,
                     **{'font':'serif','size':12, 'weight':'bold','color':'white'},
                     bbox=dict(boxstyle='round4', pad=0.2, color=color[idx], alpha=1))
        ## adding span over region
        ax2.axvspan(size_bins[idx][0],size_bins[idx][1], ymax = 1, color=color[idx], alpha=

ax2.set_ylim(0,17000)


########################################################## scattermap

custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               marker = 'o',size = 0.32, ax = ax3, color = colors[2])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               marker = 'o',size = 0.2, ax = ax3, color = colors[1])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               size = 0.1,marker = 'h', ax = ax3, color = colors[0])
ax3.set_facecolor('#f5f6f6')


######################################################## final ratio plot

color = round(df.color.value_counts(normalize = True) * 100,0).astype(int)

ax4.barh(y = color.index, width = color.values, height = 0.4, color = colors[2])



for pa in ax4.patches:
        ax4.text((pa.get_width()), pa.get_y(),'{} %'.format(pa.get_width()),
                 **{'font':'serif', 'size':12, 'weight':'bold'}, alpha = 1)

ax4.set_yticklabels(labels = color.index,**{'font':'serif', 'size':12,'weight':'bold'}, al


ax4.barh(y = color.index, width = color.values -00.1, height = 0.25, color = colors[1])
ax4.barh(y = color.index, width = color.values -00.1, height = 0.12, color = colors[0])
for loc in ['left', 'right', 'top', 'bottom']:
    ax4.spines[loc].set_visible(False)
ax4.axes.get_xaxis().set_visible(False)




### titles and descriptions

fig.text(0,0.98, 'Diamonds and Dollars: Does Color of the Diamond have Influence on Price?
         {'font':'serif', 'size':20.,'weight':'bold'}, alpha = 1)


fig.text(0,0.93,'''Color parameter is a important parameter to determine the price as per c
But its not reflecting the same with data, for an expensive diamond it is expected to be co
data speaks otherwise. Hmm, may be there is not much impact of color on price of the diamon
         {'font':'serif', 'size':11}, alpha = 0.8)
```

```
fig.text(0.05, 0.88, 'Diamond Color and Price Matrix:',{'font':'serif', 'size':16,'weight'
fig.text(0.55, 0.88, 'Diamond Price (USD) vs Color:',{'font':'serif', 'size':16,'weight':'

fig.text(0.552, 0.85, 'D grade is colorless and very desirable,\nand J grade is yellowish
        {'font':'serif', 'size':12}, alpha = 0.9)

fig.text(0.75,0.1,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':10, 'weight
fig.show()
```

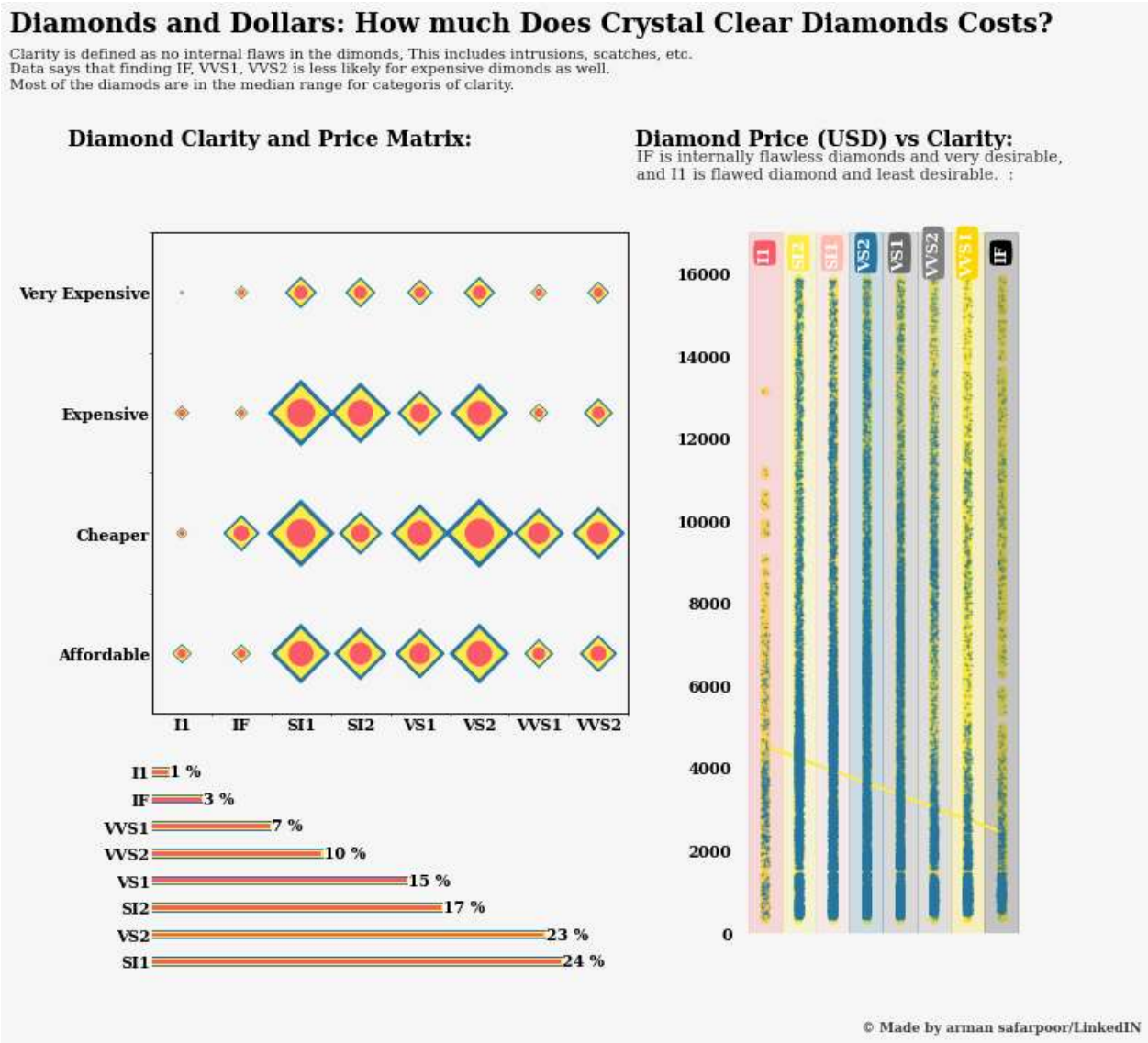


4. Clarity of The Diamond

In [23]:
```
tem = df.copy()
tem['price_cat'] = pd.cut(df['price'], bins = [0,1500,4000,10000,100000],
                          labels = ['Cheaper','Affordable', 'Expensive', 'Very Expensive']
tem['price_cat'] = tem['price_cat'].astype('object')


tab = pd.crosstab(tem['clarity'],tem['price_cat'])
tab_val = pd.melt(tab.reset_index(),id_vars = 'clarity')
tab_val.columns = ['x','y','value']



clarity_labels = [ 'I1' , 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2', 'VVS1', 'IF' ]
clarity_to_num = { val : idx for idx,val in enumerate(clarity_labels)}

tem['clarity'] = tem['clarity'].map(clarity_to_num)


fig = plt.figure(figsize = (12,12), dpi = 65)
fig.patch.set_facecolor('#f5f6f6')
gs = fig.add_gridspec(20,20)
gs.update(wspace = 0.6, hspace = 0)


ax0 = fig.add_subplot(gs[0:,0:])
ax0.set_facecolor('#f5f6f6')
ax0.axes.get_xaxis().set_visible(False)
ax0.axes.get_yaxis().set_visible(False)
for loc in ['left', 'right', 'top', 'bottom']:
        ax0.spines[loc].set_visible(False)

#ax1 = fig.add_subplot(gs[0:4,0:10])

ax2 = fig.add_subplot(gs[2:18,13:20])

ax3 = fig.add_subplot(gs[2:13,0:11])

ax4 = fig.add_subplot(gs[14:19,0:10])

axes = [ax1,ax2,ax3,ax4]

for ax in axes:
    ax.set_facecolor('#f5f6f6')
    ax.tick_params(axis='both',
                   labelsize = 12, which = 'major',
                   direction = 'out',pad = 2,
                   length = 0.001)

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_linewidth(1)




############################################## scatter plot and regplot
sns.regplot(x = 'clarity', y = 'price',data = tem, ax = ax2, color = colors[1])
sns.stripplot(x = 'clarity', y = 'price',data = tem, ax = ax2, alpha = 0.5, color = colors
ax2.set_yticklabels(np.arange(0,18000,2000),**{'font':'serif','size':12, 'weight':'bold'},
ax2.set_xticklabels('')
ax2.set_ylabel('')
ax2.set_xlabel('')
for loc in ['left', 'right', 'top', 'bottom']:
    ax2.spines[loc].set_visible(False)
```

```
bin_labels = [ 'I1' , 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2', 'VVS1', 'IF' ]
size_bins = [[-0.5,0.5],[0.5, 1.5], [1.5,  2.5], [2.5, 3.5], [3.5, 4.5],[4.5,5.5],[5.5,6.5

color = ['#FB5B68','#FFEB48','#FFBDB0','#2676A1','dimgrey', 'grey','gold','black']
for idx, label in enumerate(bin_labels):
        ax2.annotate(label,
                     xy=(sum(size_bins[idx])/2 ,16500),
                     xytext=(0,0), textcoords='offset points',
                     va="center", ha="center",rotation = 90,
                     **{'font':'serif','size':12, 'weight':'bold','color':'white'},
                     bbox=dict(boxstyle='round4', pad=0.2, color=color[idx], alpha=1))
        ## adding span over region
        ax2.axvspan(size_bins[idx][0],size_bins[idx][1], ymax = 1, color=color[idx], alpha

ax2.set_ylim(0,17000)


########################################################## scattermap

custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               marker = 'D',size = 0.3, ax = ax3, color = colors[2])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               marker = 'D',size = 0.2, ax = ax3, color = colors[1])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               size = 0.1,marker = 'o', ax = ax3, color = colors[0])
ax3.set_facecolor('#f5f6f6')


######################################################## final ratio plot

clarity = round(df.clarity.value_counts(normalize = True) * 100,0).astype(int)

ax4.barh(y = clarity.index, width = clarity.values, height = 0.35, color = colors[2])



for pa in ax4.patches:
        ax4.text((pa.get_width()), pa.get_y(),'{} %'.format(pa.get_width()),
                 **{'font':'serif', 'size':12, 'weight':'bold'}, alpha = 1)

ax4.set_yticklabels(labels = clarity.index,
                    **{'font':'serif', 'size':12,'weight':'bold'}, alpha = 1)


ax4.barh(y = clarity.index, width = clarity.values, height = 0.25, color = colors[1])
ax4.barh(y = clarity.index, width = clarity.values -00.1, height = 0.15, color = colors[0]
for loc in ['left', 'right', 'top', 'bottom']:
    ax4.spines[loc].set_visible(False)
ax4.axes.get_xaxis().set_visible(False)




### titles and descriptions

fig.text(0,0.98, 'Diamonds and Dollars: How much Does Crystal Clear Diamonds Costs?',
         {'font':'serif', 'size':20.,'weight':'bold'}, alpha = 1)


fig.text(0,0.93,'''Clarity is defined as no internal flaws in the dimonds, This includes ir
Data says that finding IF, VVS1, VVS2 is less likely for expensive dimonds as well.
```

```
Most of the diamods are in the median range for categoris of clarity.''',{'font':'serif',


fig.text(0.05, 0.88, 'Diamond Clarity and Price Matrix:',{'font':'serif', 'size':16,'weigh

fig.text(0.55, 0.88, 'Diamond Price (USD) vs Clarity:',{'font':'serif', 'size':16,'weight'

fig.text(0.552, 0.85,
         'IF is internally flawless diamonds and very desirable,\nand I1 is flawed diamond
         {'font':'serif', 'size':12}, alpha = 0.8)

fig.text(0.75,0.1,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':10, 'weight
fig.show()
```



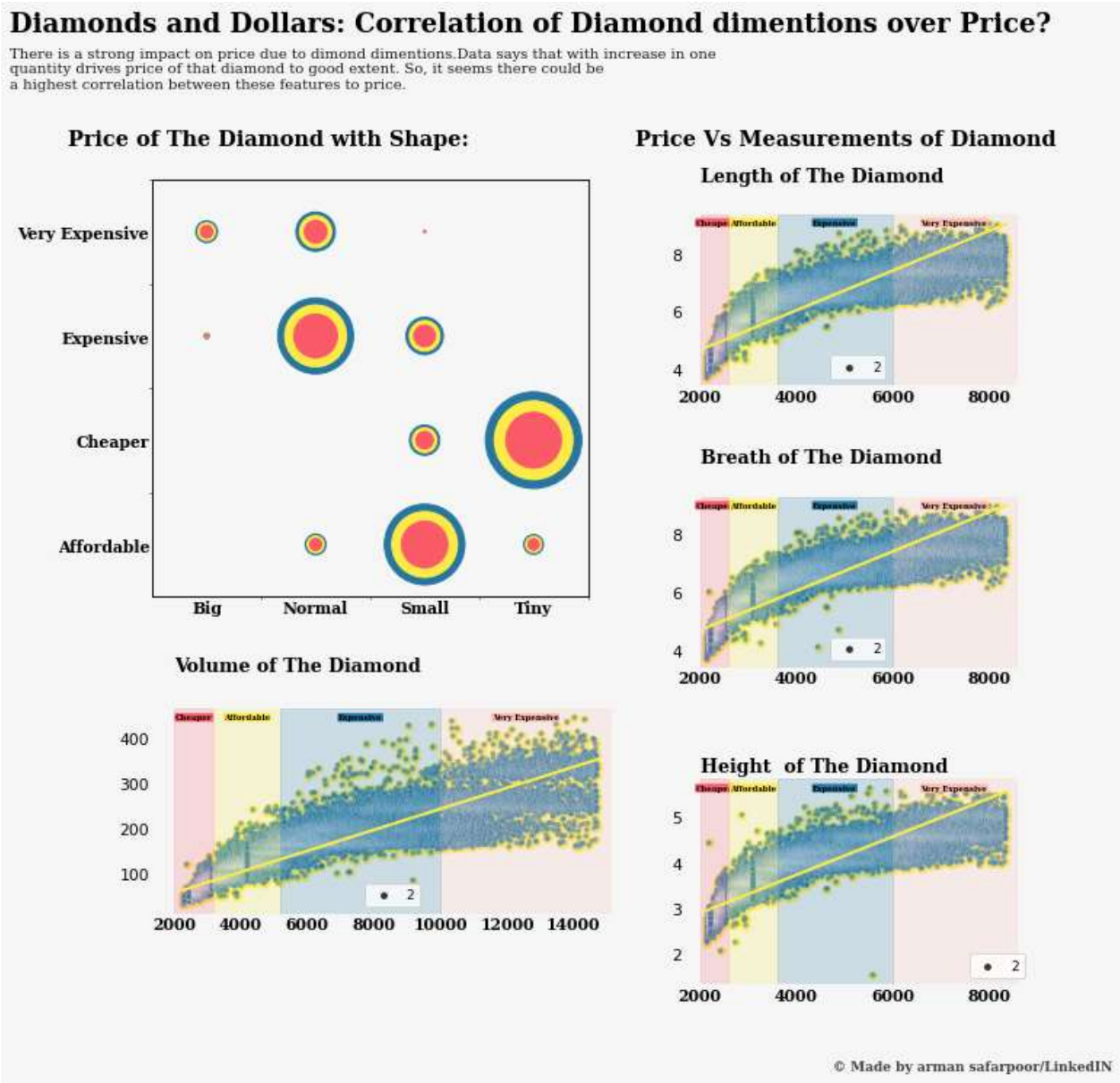
5. Dimensions of The Diamond

In [24]:

```
tem = df.copy()
tem['price_cat'] = pd.cut(df['price'], bins = [0,1500,4000,10000,100000],
                          labels = ['Cheaper','Affordable', 'Expensive', 'Very Expensive'])
tem['price_cat'] = tem['price_cat'].astype('object')


tab = pd.crosstab(tem['diamond_size'],tem['price_cat'])
tab_val = pd.melt(tab.reset_index(),id_vars = 'diamond_size')
tab_val.columns = ['x','y','value']




shape_labels = [ 'Regular','Fancy']
shape_to_num = { val : idx for idx,val in enumerate(clarity_labels)}

tem['shape'] = tem['shape'].map(shape_to_num)

fig = plt.figure(figsize = (12,12), dpi = 65)
fig.patch.set_facecolor('#f5f6f6')
gs = fig.add_gridspec(25,20)
gs.update(wspace = 0.2, hspace = 0.2)

ax0 = fig.add_subplot(gs[0:,0:])
ax0.set_facecolor('#f5f6f6')
ax0.axes.get_xaxis().set_visible(False)
ax0.axes.get_yaxis().set_visible(False)
for loc in ['left', 'right', 'top', 'bottom']:
        ax0.spines[loc].set_visible(False)

ax1 = fig.add_subplot(gs[2:7,12:])

ax2 = fig.add_subplot(gs[10:15,12:])

ax3 = fig.add_subplot(gs[18:24,12:])


ax4 = fig.add_subplot(gs[16:22,0:11])

ax5 = fig.add_subplot(gs[1:13,0:10])


axes = [ax0,ax1, ax2, ax3, ax4]

for ax in axes:
    ax.set_facecolor('#f5f6f6')
    ax.tick_params(axis='both',
                   labelsize = 12, which = 'major',
                   direction = 'out',pad = 2,
                   length = 0.02)

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)


axes = [ax1, ax2, ax3,ax4]

cols = ['x','y','z','Volume']

for ax,col in zip(axes,cols):

    sns.regplot(y = tem[col], x = 'price',data = tem, ax = ax, color = colors[1])
```

```
    sns.scatterplot(y = tem[col], x = 'price',data = tem, ax = ax, alpha = 0.6,color = col
    ax.set_xticklabels(np.arange(0,18000,2000),**{'font':'serif','size':12, 'weight':'bold
    #ax.set_yticklabels('')
    ax.set_ylabel('')
    ax.set_xlabel('')


    bin_labels = ['Cheaper','Affordable', 'Expensive', 'Very Expensive']
    [0,1500,4000,10000,100000]
    size_bins = [[0,1500],[1500, 4000], [4000, 10000], [10000,tem.price.max()+500]]

    color = ['#FB5B68','#FFEB48','#2676A1','#FFBDB0']
    for idx, label in enumerate(bin_labels):
        ax.annotate(label,
                    xy=(sum(size_bins[idx])/2 ,tem[col].max()),
                    xytext=(0,0), textcoords='offset points',
                    va="center", ha="center",rotation = 0,
                    **{'font':'serif','size':6, 'weight':'bold','color':'black'},
                    bbox=dict(boxstyle='round4', pad=0., color=color[idx], alpha= 1))
        ## adding span over region
        ax.axvspan(size_bins[idx][0],size_bins[idx][1], ymax = 1, color=color[idx], alpha=(

        #ax.set_ylim(0,17000)


########################################################## scattermap

custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               marker = 'o',size = 0.3, ax = ax5, color = colors[2])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               marker = 'o',size = 0.2, ax = ax5, color = colors[1])
custom_scatter(x = tab_val['x'], y = tab_val['y'], value = tab_val['value'],
               size = 0.1,marker = 'o', ax = ax5, color = colors[0])
ax5.set_facecolor('#f5f6f6')


ax5.tick_params(axis='both',
                  labelsize = 12, which = 'major',
                  direction = 'out',pad = 2,
                  length = 0.02)


for loc in ['left', 'right', 'top', 'bottom']:
    ax5.spines[loc].set_linewidth(1)



### titles and descriptions

fig.text(0,0.98, 'Diamonds and Dollars: Correlation of Diamond dimentions over Price?',
         {'font':'serif', 'size':20.,'weight':'bold'}, alpha = 1)


fig.text(0,0.93,'''There is a strong impact on price due to dimond dimentions.Data says tha
quantity drives price of that diamond to good extent. So, it seems there could be
a highest correlation between these features to price.''',{'font':'serif', 'size':11}, alpl


fig.text(0.05, 0.88, 'Price of The Diamond with Shape:',{'font':'serif', 'size':16,'weight

fig.text(0.55, 0.88, 'Price Vs Measurements of Diamond',{'font':'serif', 'size':16,'weight

ax1.text(0,10.5,'Length of The Diamond',{'font':'serif', 'size':14,'weight':'bold'}, alpha
```

```
ax2.text(0,10.5,'Breath of The Diamond',{'font':'serif', 'size':14,'weight':'bold'}, alpha
ax3.text(0,6.,'Height  of The Diamond',{'font':'serif', 'size':14,'weight':'bold'}, alpha =
ax4.text(0,550,'Volume of The Diamond',{'font':'serif', 'size':14,'weight':'bold'}, alpha =


fig.text(0.725,0.065,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':10, 'wei
fig.show()
```

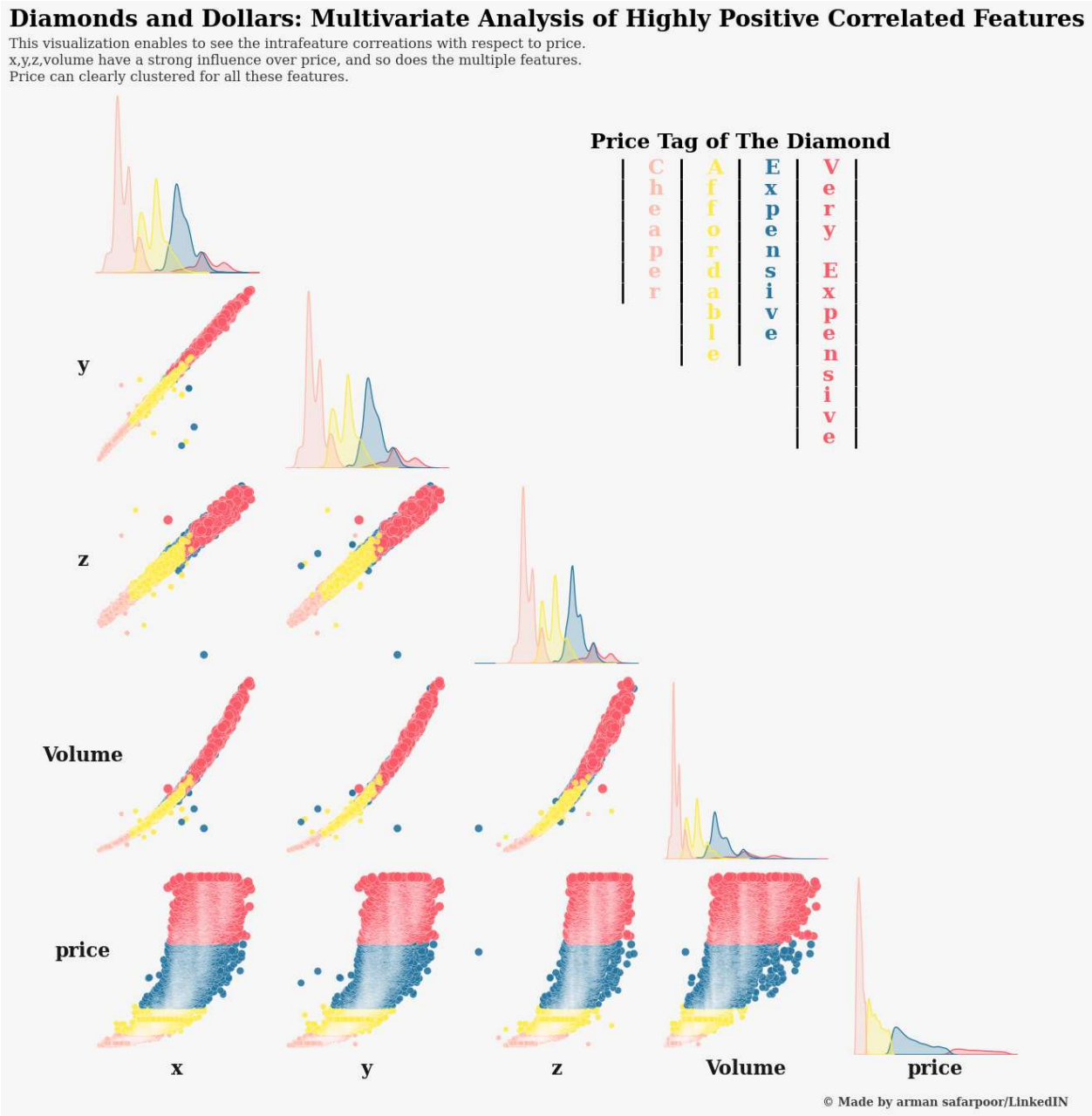


4.5 Multi-variate Analysis of Diamond

In [25]:
```
### two variable comparisions

tem = featdf.copy()


tem['Price Category'] = pd.cut(df['price'], bins = [0,1500,4000,10000,100000],
                               labels = ['Cheaper','Affordable', 'Expensive', 'Very Expens
tem['Price Category'] = tem['Price Category'].astype('object')

cols_here = ['#FFBDB0', '#FFEB48', '#2676A1', '#FB5B68']
order = ['Cheaper','Affordable', 'Expensive', 'Very Expensive']

fig = plt.gcf();
fig.set_size_inches(10,10)

fig.set_dpi(50)


g = sns.pairplot(data = tem, vars = ['x','y','z','Volume','price'],
                 hue= 'Price Category', hue_order = order,
             corner = True, diag_kind='kde', palette = cols_here,
                plot_kws = {'alpha':0.90, 'size' : tem['price']*2, 'linewidth' : 0.2})
g._legend.remove()

for ax in plt.gcf().axes:
    ax.set_facecolor('#f5f6f6')
    for loc in ['left','right','top','bottom']:
        ax.spines[loc].set_visible(False)
    ax.set_xticks(ticks = [])
    ax.set_yticks(ticks = [])

    ax.set_xlabel(xlabel = ax.get_xlabel(), **{'font':'serif', 'size':17,'weight':'bold'},
    ax.set_ylabel(ylabel = ax.get_ylabel(), **{'font':'serif', 'size':17,'weight':'bold'},

plt.gcf().patch.set_facecolor('#f5f6f6')

### titles and descriptions

plt.gcf().text(0.,1.0425, 'Diamonds and Dollars: Multivariate Analysis of Highly Positive (
               {'font':'serif', 'size':20.,'weight':'bold'}, alpha = 1)


plt.gcf().text(0,0.99,'''This visualization enables to see the intrafeature correations wit
x,y,z,volume have a strong influence over price, and so does the multiple features.
Price can clearly clustered for all these features.''',{'font':'serif', 'size':12}, alpha =


## legend



plt.gcf().text(0.55 - 0.05 , 0.9 + 0.025, 'Price Tag of The Diamond',
               {'font':'serif', 'size':18,'weight':'bold', 'color':'black'})
i = 0 ## xcontrol
for word,color in zip(order,cols_here):
    j = 0 ##ycontrol

    for char in list(word):

        plt.gcf().text(0.55 - 0.025 +i, 0.9  - j , '|', {'font':'serif', 'size':18,'weight
        plt.gcf().text(0.55 + i, 0.9 - j , char , {'font':'serif', 'size':18,'weight':'bol
        plt.gcf().text(0.55 +0.025 +i, 0.9  - j , '|', {'font':'serif', 'size':18,'weight'
        j += 0.02
```

```
    i += 0.05

plt.gcf().text(0.7,0.,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':10, 'we

plt.show()
```

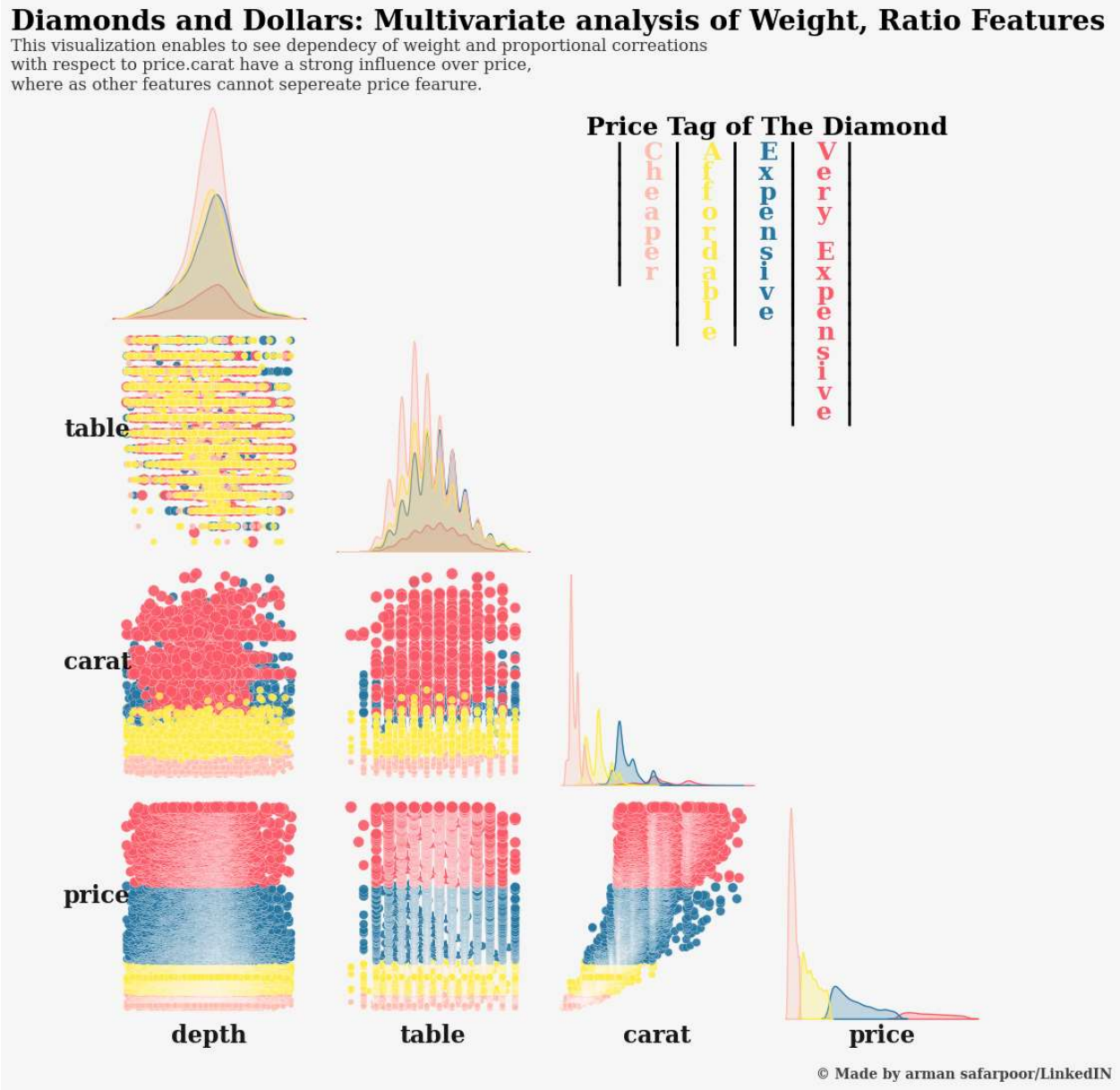
```
<Figure size 500x500 with 0 Axes>
```



In [26]:
```
### two variable comparisions

tem = featdf.copy()


tem['Price Category'] = pd.cut(df['price'], bins = [0,1500,4000,10000,100000],
                               labels = ['Cheaper','Affordable', 'Expensive', 'Very Expens
tem['Price Category'] = tem['Price Category'].astype('object')

cols_here = ['#FFBDB0', '#FFEB48', '#2676A1', '#FB5B68']
order = ['Cheaper','Affordable', 'Expensive', 'Very Expensive']

fig = plt.gcf();
fig.set_size_inches(12,12)

fig.set_dpi(70)


g = sns.pairplot(data = tem, vars = ['depth','table','carat','price'],
                 hue= 'Price Category', hue_order = order,
             corner = True, diag_kind='kde', palette = cols_here,
                plot_kws = {'alpha':0.90, 'size' : tem['price']*2, 'linewidth' : 0.2})
g._legend.remove()

for ax in plt.gcf().axes:
    ax.set_facecolor('#f5f6f6')
    for loc in ['left','right','top','bottom']:
        ax.spines[loc].set_visible(False)
    ax.set_xticks(ticks = [])
    ax.set_yticks(ticks = [])

    ax.set_xlabel(xlabel = ax.get_xlabel(), **{'font':'serif', 'size':17,'weight':'bold'},
    ax.set_ylabel(ylabel = ax.get_ylabel(), **{'font':'serif', 'size':17,'weight':'bold'},

plt.gcf().patch.set_facecolor('#f5f6f6')

### titles and descriptions

plt.gcf().text(0.,1.05, 'Diamonds and Dollars: Multivariate analysis of Weight, Ratio Featu
               {'font':'serif', 'size':20.,'weight':'bold'}, alpha = 1)


plt.gcf().text(0,0.99,'''This visualization enables to see dependecy of weight and proport
with respect to price.carat have a strong influence over price,
where as other features cannot sepereate price fearure.''',{'font':'serif', 'size':12}, al


## legend



plt.gcf().text(0.55 - 0.05 , 0.92 + 0.025, 'Price Tag of The Diamond',
               {'font':'serif', 'size':18,'weight':'bold', 'color':'black'})
i = 0 ## xcontrol
for word,color in zip(order,cols_here):
    j = 0 ##ycontrol

    for char in list(word):

        plt.gcf().text(0.55 - 0.025 +i, 0.92  - j , '|', {'font':'serif', 'size':18,'weigh
        plt.gcf().text(0.55 + i, 0.92 - j , char , {'font':'serif', 'size':18,'weight':'bo
        plt.gcf().text(0.55 +0.025 +i, 0.92  - j , '|', {'font':'serif', 'size':18,'weight
        j += 0.02
```

```
    i += 0.05

plt.gcf().text(0.7,0.,'© Made by arman safarpoor/LinkedIN',{'font':'serif', 'size':10, 'we

plt.show()
```

```
<Figure size 840x840 with 0 Axes>
```



5. Modeling and Results

With the visualization part our guy, understood that price, has strong correaltionss with the dimensions of the diamond and 4C's of the diamond.

Now, lets build a model to predict accurately as required for our guy.

Here method is simple, find featrue importance, select features, build some base models, and stack them high to make a final predictions. In the end lets see how can we interpert our model to our guy and help him find his ring.

lets do few last minute tweeks to data before jump into modeling.

In [27]:
```
print('final preprocessing of variables......')
#log transform to all the variables to make less skewed
featdf["price"] = np.log1p(featdf["price"])

#Log transform skewed numeric features:
numeric_feats = featdf.dtypes[featdf.dtypes != "object"].index

skewed_feats = featdf[numeric_feats].dropna().skew() #compute skewness
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index

featdf[skewed_feats] = np.log1p(featdf[skewed_feats])

# split data into X and y
X = featdf.drop(columns = ['price'])
y = featdf['price']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 2021)

#X_train,X_val,y_train,y_val = train_test_split(X_train,y_train,test_size = 0.1, random_sta

print('All set! Good to go!')
```

```
final preprocessing of variables......
All set! Good to go!
```

In [32]:
```
# Building pipelins of standard scaler and model for varios regressors.
from sklearn.pipeline import Pipeline

pipeline_lr=Pipeline([("scalar1",StandardScaler()),
                      ("lr_classifier",LinearRegression())])

pipeline_dt=Pipeline([("scalar2",StandardScaler()),
                      ("dt_classifier",DecisionTreeRegressor())])

pipeline_rf=Pipeline([("scalar3",StandardScaler()),
                      ("rf_classifier",RandomForestRegressor())])


pipeline_kn=Pipeline([("scalar4",StandardScaler()),
                      ("rf_classifier",KNeighborsRegressor())])


#pipeline_xgb=Pipeline([("scalar5",StandardScaler()),
                      #("rf_classifier",XGBRegressor())])

# List of all the pipelines
pipelines = [pipeline_lr, pipeline_dt, pipeline_rf, pipeline_kn] #pipeline_xgb]

# Dictionary of pipelines and model types for ease of reference
pipe_dict = {0: "LinearRegression", 1: "DecisionTree", 2: "RandomForest",3: "KNeighbors"} #

# Fit the pipelines
for pipe in pipelines:
    pipe.fit(X_train, y_train)
```

In [33]:
```
cv_results_rms = []
for i, model in enumerate(pipelines):
    cv_score = cross_val_score(model, X_train,y_train,scoring="neg_root_mean_squared_error'
    cv_results_rms.append(cv_score)
    print("%s: %f " % (pipe_dict[i], cv_score.mean()))
```

```
LinearRegression: -0.202770
DecisionTree: -0.120994
RandomForest: -0.088393
KNeighbors: -0.151566
```

In [34]:
```
# Model prediction on test data
pred = pipeline_rf.predict(X_test)
```

In [35]:
```
# Model Evaluation
print("R^2:",metrics.r2_score(y_test, pred))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, pred))*(len(y_test)-1)/(len(y_test)-X
print("MAE:",metrics.mean_absolute_error(y_test, pred))
print("MSE:",metrics.mean_squared_error(y_test, pred))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
R^2: 0.9919941163885106
Adjusted R^2: 0.9919840216759065
MAE: 0.06213174947337511
MSE: 0.007697144449797319
RMSE: 0.08773337135775257
```

6. Summary and Conclusions

With all the analysis, visualizaitons, Modeling and interpretation of the model, now I am sure this model will give the best diamond for our guy.
Well its time to help our guy, with our little project, he got his prefect diamond, and lived happily ever after.

Lets revist What have we done so far?

- First we have exposed ourself to dataset terminology, and dataset.
- Viewed data from statistical stand point.
- Find few patterns from data
- Fitted data to models
- Made sure our mode is trust worthy.

What else can we do in futur? may be tunning base models and traing the stacked regressor with those base models and intrepreting stacked model if possible.

Thanks for reading all the way here.....Please Up vote if you find my work useful.

In [ ]: