In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import chart_studio.plotly as py
import plotly.express as px
```

Read data

- Read the load data file
- Read the column decription file

In [2]:

```
data = pd.read_csv('../data/loan.csv', low_memory=False)
meta_data = pd.read_excel('../data/Data_Dictionary.xlsx')
meta_data.columns = ['column_name', 'description']
```

Getting the info of loan data using `data.info()` function

In [3]:

```
data.info(
    max_cols = 111,
    show_counts = True
)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 111 columns):
 #    Column                          Non-Null Count  Dtype
---   ------                          --------------  -----
 0    id                              39717 non-null  int64
 1    member_id                       39717 non-null  int64
 2    loan_amnt                       39717 non-null  int64
 3    funded_amnt                     39717 non-null  int64
 4    funded_amnt_inv                 39717 non-null  float64
 5    term                            39717 non-null  object
 6    int_rate                        39717 non-null  object
 7    installment                     39717 non-null  float64
 8    grade                           39717 non-null  object
 9    sub_grade                       39717 non-null  object
 10   emp_title                       37258 non-null  object
 11   emp_length                      38642 non-null  object
 12   home_ownership                  39717 non-null  object
 13   annual_inc                      39717 non-null  float64
 14   verification_status             39717 non-null  object
```

View of loan data

Top 5 rows of load data

In [4]:

```
data.head()
```

Out[4]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90g_dpd_24m	nur
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	NaN	

5 rows × 111 columns

Column analysis

Create new data frame for colum analysis where computed following data for each column

- Total number of missing values
- Total % of missing value
- Merged with description data
- Data type of column
- A shample column which store
  - If column is of int or float type then min and max value
  - If column is of object type the few shample values
- New categorical column for missing value categories depends on % of missing values
  - If 0 then VL (very low)

- If b/w 1 - 10 then L(low)
- If b/w 11 - 30 then M(medium)
- If b/w 31 - 80 then H(high)
- If b/w 81 - 100 then VH(very high)

In [5]:
```
column_analysis = pd.DataFrame(
    data.isna().sum(),
    columns=['total_of_nan']
)

column_analysis['missing_value_percent'] = round(
    column_analysis['total_of_nan'] / len(data) * 100,
    2
)
column_analysis['column_name'] = column_analysis.index
```

In [6]:
```
column_analysis.set_index(pd.RangeIndex(1, len(column_analysis)+1), drop=True, inplace=True)
```

In [7]:
```
column_analysis = column_analysis.merge(meta_data, on=['column_name'], how='left')
```

In [8]:
```
def get_column_type(col_name):
    return data[col_name].dtype
```

In [9]:
```
column_analysis['dtype'] = column_analysis.column_name.apply(get_column_type)
```

In [10]:
```
def get_sample_value(col_name):
    col = column_analysis[column_analysis.column_name == col_name]
    dtype = col['dtype'].values[0]
    col_value = data[-data[col_name].isna()][col_name].values
    if (dtype == int or dtype == float) and col['missing_value_percent'].values[0] < 100:
        return F'{min(col_value)} | {max(col_value)}'
    elif dtype == object:
        _values = pd.Series(list(map(str,col_value))).unique()
        if(len(_values)/len(data) <= .15):
            return ' | '.join(_values)
        return ' | '.join(_values[:10])
    else:
        return None
```

In [11]:
```
column_analysis['sample_value'] = column_analysis.column_name.apply(get_sample_value)
```

In [12]:
```
def get_column_category(missins_percent):
    if missins_percent == 0:
        return 'Very Low (0% missing)'
    elif missins_percent > 0 and missins_percent <= 10:
        return 'Low (1-10% missing)'
    elif missins_percent > 11 and missins_percent <= 30:
        return 'Medium (11-30% missing)'
    elif missins_percent > 31 and missins_percent <= 80:
        return 'High (31-80% missing)'
    else:
        return 'Very High (80-100% missing)'
```

In [13]:
```
column_analysis['missing_category'] = column_analysis.missing_value_percent.apply(get_column_category)
```

Calculated % of column under each category

Grupedby type then computed % under each category

In [14]:
```
missing_category_percentage = pd.DataFrame(
    round(
        column_analysis.groupby(by = 'missing_category')
            .count()['total_of_nan']/len(column_analysis)*100,
        2
    ),
)
missing_category_percentage['missing_category'] = missing_category_percentage.index
missing_category_percentage.set_index(pd.RangeIndex(1, len(missing_category_percentage)+1), drop=True, inplac
missing_category_percentage.columns = ['column_percent','missing_category']
```

```
In [15]: missing_category_percentage
```

```
Out[15]:
```

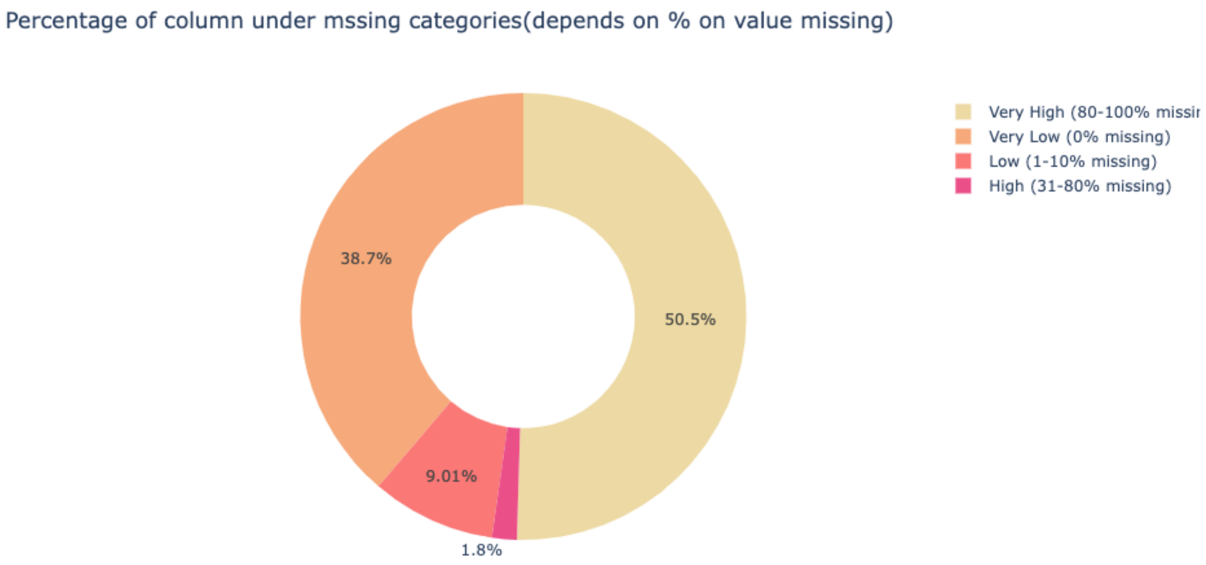
	column_percent	missing_category
1	1.80	High (31-80% missing)
2	9.01	Low (1-10% missing)
3	50.45	Very High (80-100% missing)
4	38.74	Very Low (0% missing)

Ploting missing % of columns values category wise

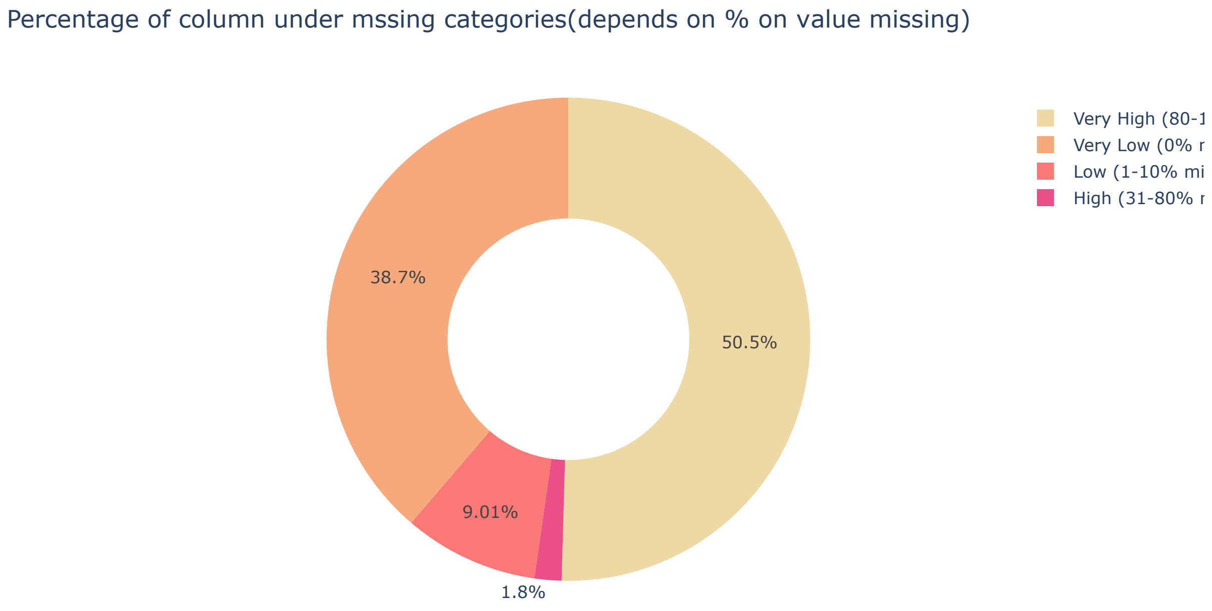
Some intresting insite found

- 50% of columns have 100% missing value
- 1.8% of columns have 31 - 80% missing values
- Only 38.7% of columns have 0% missing values

So decided to go with only those rows which have less then 10% missing value


Percentage of column under mssing categories(depends on % on value missing)

```
In [16]: fig = px.pie(
             missing_category_percentage,
             values='column_percent',
             names='missing_category',
             title='Percentage of column under mssing categories(depends on % on value missing)',
             hole=.5,
             color_discrete_sequence=px.colors.sequential.Agsunset_r,
             labels={'missing_category':'Category', 'column_percent': 'Column Percentage'},
         )
         fig.show()
```



```
In [17]: column_analysis.set_index('column_name', inplace=True)
```

Saving the column analysis data for future review

```
In [18]: column_analysis.to_excel('../data/explore_data.xlsx')
```