"Lending Club Analysis and Data Cleaning using PySpark"

Description:

1. In this project I have used the lending club dataset. Lending club is a peer to peer lending company. They will be providing the loans to companies, peoples. Sometimes peoples will not have good credit history but they want immediately want loan will come to lending club and ask for loan.
2. There will be set of peoples who will be seeing the people's profile
    a. If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company
    b. If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company
3. In this the interest rate are higher than the other banks.
4. Above is the overview of the lending club and I have used the the 2018 Q4 dataset.

Dataset:

The size of the dataset is 70mb which contains 128K records & 125 columns.

LoanStats_2018Q4.csv

Problem:

The values present in the dataset is uncleaned(Some columns like term contains the datatypes as string('36 months') but in the business use we need to convert it into int(36).
If we downstream this data to machine learning models we have to make it to numeric.

Data Cleaning overview:

1. Since it contains 145 coulmns need to limit it into another df.

2.

```
df_sel.describe("loan_amnt","emp_length","dti","delinq_2yrs","revol_util","total_acc").show()
```

```
+-------+-----------------+----------+------------------+------------------+----------+------------------+
|summary|        loan_amnt|emp_length|               dti|       delinq_2yrs|revol_util|         total_acc|
+-------+-----------------+----------+------------------+------------------+----------+------------------+
|  count|           128412|    128412|            128175|            128412|    128256|            128412|
|   mean| 15971.32102139987|      null|19.933177530719778|0.22783696227766875|     null|22.677413325857398|
| stddev|10150.384232741915|      null|20.143542243475476| 0.7337929617806072|     null|12.129215673024733|
|    min|             1000|    1 year|               0.0|                 0|        0%|                 2|
|    max|            40000|       n/a|             999.0|                24|    99.90%|               160|
+-------+-----------------+----------+------------------+------------------+----------+------------------+
```

From the above image we can see in the 'emp_length' for the min it is showing as '1 year', mean & stddev it has treated as string but there is an integer value. Also we are having a good outliers for 'dti' for mean & stddev. We need to clean the above highlighed so that we can get the proper statistics.

3.

```
#Checking employee length varible
spark.sql("select distinct emp_length from loanstats limit 50").show()
```

```
+----------+
|emp_length|
+----------+
|   5 years|
|   9 years|
|    1 year|
|       n/a|
|   2 years|
|   7 years|
|   8 years|
|   4 years|
|   6 years|
|   3 years|
| 10+ years|
|  < 1 year|
+----------+
```

From the 'emp_length' we could see 4 different types of values present in it, we need only the values in **numeric** type.

4. Clean the 'term' column
5. Clean the 'revol_util' column.
6. Clean the 'dti' column -> Null values
7. Clean the 'loan_status'

Data Aggregation:

1.Schema of the dataset.

```
df.printSchema()

root
 |-- id: string (nullable = true)
 |-- member_id: string (nullable = true)
 |-- loan_amnt: integer (nullable = true)
 |-- funded_amnt: integer (nullable = true)
 |-- funded_amnt_inv: double (nullable = true)
 |-- term: string (nullable = true)
 |-- int_rate: string (nullable = true)
 |-- installment: double (nullable = true)
 |-- grade: string (nullable = true)
 |-- sub_grade: string (nullable = true)
 |-- emp_title: string (nullable = true)
 |-- emp_length: string (nullable = true)
 |-- home_ownership: string (nullable = true)
 |-- annual_inc: double (nullable = true)
 |-- verification_status: string (nullable = true)
 |-- issue_d: string (nullable = true)
 |-- loan_status: string (nullable = true)
 |-- pymnt_plan: string (nullable = true)
 |-- url: string (nullable = true)
 |-- desc: string (nullable = true)
 |-- purpose: string (nullable = true)
 |-- title: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- addr_state: string (nullable = true)
 |-- dti: double (nullable = true)
 |-- delinq_2yrs: integer (nullable = true)
 |-- earliest_cr_line: string (nullable = true)
 |-- inq_last_6mths: integer (nullable = true)
 |-- mths_since_last_delinq: integer (nullable = true)
```

2. Displaying the records

```
df.show()
```

```
+----+---------+---------+-----------+---------------+----------+--------+-----------+-----+---------+--------------------+----------
|  id|member_id|loan_amnt|funded_amnt|funded_amnt_inv|      term|int_rate|installment|grade|sub_grade|           emp_title|emp_lengt
+----+---------+---------+-----------+---------------+----------+--------+-----------+-----+---------+--------------------+----------
|null|     null|    10000|      10000|        10000.0| 36 months|  10.33%|     324.23|    B|       B1|                null| < 1 yea
|null|     null|     2500|       2500|         2500.0| 36 months|  13.56%|      84.92|    C|       C1|                Chef| 10+ year
|null|     null|    12000|      12000|        12000.0| 60 months|  13.56%|     276.49|    C|       C1|                null| < 1 yea
|null|     null|    15000|      15000|        14975.0| 60 months|  14.47%|     352.69|    C|       C2|                null|       n/
|null|     null|    16000|      16000|        16000.0| 60 months|  17.97%|     406.04|    D|       D1|Instructional Coo...|   5 year
|null|     null|     9600|       9600|         9600.0| 36 months|  23.40%|     373.62|    E|       E1| driver coordinator |   9 year
|null|     null|     4000|       4000|         4000.0| 36 months|  23.40%|     155.68|    E|       E1|            Security|   3 year
|null|     null|     3500|       3500|         3500.0| 36 months|  20.89%|     131.67|    D|       D4|       gas attendant| 10+ year
|null|     null|     9600|       9600|         9600.0| 36 months|  12.98%|     323.37|    B|       B5|                null|       n/
|null|     null|     8000|       8000|         8000.0| 36 months|  23.40%|     311.35|    E|       E1|             Manager| 10+ year
|null|     null|    16000|      16000|        16000.0| 60 months|  26.31%|     481.99|    E|       E4|Financial Relatio...| < 1 yea
|null|     null|    30000|      30000|        30000.0| 60 months|  18.94%|     777.23|    D|       D2|         Postmaster | 10+ year
|null|     null|    10000|      10000|        10000.0| 60 months|  19.92%|      264.5|    D|       D3|    Material Handler| 10+ year
|null|     null|     5000|       5000|         5000.0| 36 months|  17.97%|     180.69|    D|       D1|      Administrative|   6 year
|null|     null|    23000|      23000|        23000.0| 60 months|  20.89%|     620.81|    D|       D4|            Operator|   5 year
|null|     null|    32075|      32075|        32075.0| 60 months|  11.80%|     710.26|    B|       B4|  Nursing Supervisor| 10+ year
|null|     null|    13000|      13000|        13000.0| 36 months|  23.40%|     505.95|    E|       E1| Sale Representative|   2 year
|null|     null|    35000|      35000|        35000.0| 60 months|  15.02%|     833.02|    C|       C3|    Sr Sales Manager| 10+ year
|null|     null|    40000|      40000|        40000.0| 60 months|  11.31%|      875.9|    B|       B3|                null| < 1 yea
|null|     null|     8000|       8000|         8000.0| 36 months|  10.33%|     259.38|    B|       B1|            Optician|   6 year
+----+---------+---------+-----------+---------------+----------+--------+-----------+-----+---------+--------------------+----------
only showing top 20 rows
```

3.How many row's are in the dataset.

```
#check how many rows are there in the dataset
df.count()
```

```
128412
```

4.Creating a temporary variable to run the sql queries

```
#creating a temporary table to run the SQL queries.
df.createOrReplaceTempView("loansatats")
```

5.Since there are more than 145 column limiting only the selected column

```
#selecting only the required column which will be only usable.
df_sel = df.select("term","home_ownership","grade","purpose",
                   "int_rate","addr_state","loan_status","application_type",
                   "loan_amnt","emp_length","annual_inc","dti","delinq_2yrs",
                   "revol_util","total_acc","num_tl_90g_dpd_24m","dti_joint")
```

6.When displaying the output we can see the below error

```
df_sel.describe("loan_amnt","emp_length","dti","delinq_2yrs","revol_util","total_acc").show()

+-------+-----------------+----------+-----------------+------------------+----------+-----------------+
|summary|        loan_amnt|emp_length|              dti|       delinq_2yrs|revol_util|        total_acc|
+-------+-----------------+----------+-----------------+------------------+----------+-----------------+
|  count|           128412|    128412|           128175|            128412|    128256|           128412|
|   mean| 15971.32102139987|      null|19.933177530719778|0.22783696227766875|      null|22.677413325857398|
| stddev|10150.384232741915|      null|20.143542243475476| 0.7337929617806072|      null|12.129215673024733|
|    min|             1000|    1 year|              0.0|                 0|        0%|                2|
|    max|            40000|       n/a|            999.0|                24|    99.90%|              160|
+-------+-----------------+----------+-----------------+------------------+----------+-----------------+
```

7. Checking the 'emp_length'

```
#Checking employee length varible
spark.sql("select distinct emp_length from loanstats limit 50").show()

+----------+
|emp_length|
+----------+
|   5 years|
|   9 years|
|    1 year|
|       n/a|
|   2 years|
|   7 years|
|   8 years|
|   4 years|
|   6 years|
|   3 years|
| 10+ years|
|  < 1 year|
+----------+
```

It contains 4 different types of values present in it but we need only the numeric values.

Data after cleaned:

```
from pyspark.sql.functions import regexp_replace, regexp_extract
from pyspark.sql.functions import col

regexp_string='years|year|\\+|\\<'  #before + we nneed to prefix \\ or else it will take the + symbol as an operator.
df_sel.select(regexp_replace(col('emp_length'),regexp_string,"").alias("emplenght_cleaned"),col("emp_length")).show()
```

```
+-----------------+---------+
|emplenght_cleaned|emp_length|
+-----------------+---------+
|                1 |  < 1 year|
|               10 | 10+ years|
|                1 |  < 1 year|
|               n/a|       n/a|
|                5 |   5 years|
|                9 |   9 years|
|                3 |   3 years|
|               10 | 10+ years|
|               n/a|       n/a|
|               10 | 10+ years|
|                1 |  < 1 year|
|               10 | 10+ years|
|               10 | 10+ years|
|                6 |   6 years|
|                5 |   5 years|
|               10 | 10+ years|
|                2 |   2 years|
|               10 | 10+ years|
|                1 |  < 1 year|
|                6 |   6 years|
```

Cleaning the 'term' coumn from '36 months' to 36

```
+------------+----------+
|term_cleaned|      term|
+------------+----------+
|          36| 36 months|
|          36| 36 months|
|          60| 60 months|
|          60| 60 months|
|          60| 60 months|
|          36| 36 months|
|          36| 36 months|
|          36| 36 months|
|          36| 36 months|
|          36| 36 months|
|          60| 60 months|
|          60| 60 months|
|          60| 60 months|
|          36| 36 months|
|          60| 60 months|
|          60| 60 months|
|          36| 36 months|
|          60| 60 months|
|          60| 60 months|
|          36| 36 months|
+------------+----------+
only showing top 20 rows
```

0s completed at 12:58 PM

In the above one still it is not assigned to df. After assigning it to df,

```
df_sel.select("term","term_cleaned","emp_length","emplen_cleaned").show()
```

```
+----------+------------+----------+--------------+
|      term|term_cleaned|emp_length|emplen_cleaned|
+----------+------------+----------+--------------+
| 36 months|          36 |  < 1 year|             1|
| 36 months|          36 | 10+ years|            10|
| 60 months|          60 |  < 1 year|             1|
| 60 months|          60 |       n/a|              |
| 60 months|          60 |   5 years|             5|
| 36 months|          36 |   9 years|             9|
| 36 months|          36 |   3 years|             3|
| 36 months|          36 | 10+ years|            10|
| 36 months|          36 |       n/a|              |
| 36 months|          36 | 10+ years|            10|
| 60 months|          60 |  < 1 year|             1|
| 60 months|          60 | 10+ years|            10|
| 60 months|          60 | 10+ years|            10|
| 36 months|          36 |   6 years|             6|
| 60 months|          60 |   5 years|             5|
| 60 months|          60 | 10+ years|            10|
| 36 months|          36 |   2 years|             2|
| 60 months|          60 | 10+ years|            10|
| 60 months|          60 |  < 1 year|             1|
| 36 months|          36 |   6 years|             6|
+----------+------------+----------+--------------+
only showing top 20 rows
```

8.Print the schema

```
df_sel.printSchema()
```

```
root
 |-- term: string (nullable = true)
 |-- home_ownership: string (nullable = true)
 |-- grade: string (nullable = true)
 |-- purpose: string (nullable = true)
 |-- int_rate: string (nullable = true)
 |-- addr_state: string (nullable = true)
 |-- loan_status: string (nullable = true)
 |-- application_type: string (nullable = true)
 |-- loan_amnt: integer (nullable = true)
 |-- emp_length: string (nullable = true)
 |-- annual_inc: double (nullable = true)
 |-- dti: double (nullable = true)
 |-- delinq_2yrs: integer (nullable = true)
 |-- revol_util: string (nullable = true)
 |-- total_acc: integer (nullable = true)
 |-- num_tl_90g_dpd_24m: integer (nullable = true)
 |-- dti_joint: double (nullable = true)
 |-- term_cleaned: string (nullable = true)
 |-- emplen_cleaned: string (nullable = true)
```

Even though the the column is cleaned and it has only the numerical values spark takes this as '**string**' to convert the datatype into '**int**'we need to do it manually(Redefine the schema).

Correlation Matrix:

1.Checking for Correlation.

```
'''df_sel.stat.corr('annual_inc','loan_amnt')'''  #stat-refers to statistic function,corr-corelation between two column
                                                  #In here we are checking if the two columns are correlated to eacbh other
                                                  #-1- Negatively correlated
                                                  #+1 - Positively correlated
                                                  #0 - No Correlation

                                                  #OR
spark.sql("select corr(annual_inc,loan_amnt) from loanstatus_sel").show() #in sql

+-------------------------+
|corr(annual_inc, loan_amnt)|
+-------------------------+
|       0.20103225337914624|
+-------------------------+
```

```
#from above one we can see it is not so much correlated.
```

```
[27] spark.sql("select corr(loan_amnt,term_cleaned) from loanstatus_sel").show()

+---------------------------+
|corr(loan_amnt, term_cleaned)|
+---------------------------+
|          0.3925941141844244|
+---------------------------+
```

```
#still not a strong correlation
```

2.Cross Tab

```
#Crosstab - displays the frequency distribution of values for
df_sel.stat.crosstab('loan_status','grade').show()

+-----------------+-----+-----+-----+-----+----+---+---+
| loan_status_grade|    A|    B|    C|    D|   E|  F|  G|
+-----------------+-----+-----+-----+-----+----+---+---+
|       Fully Paid| 1188| 1333| 1175|  807| 360| 36|  9|
|  In Grace Period|   74|  112|  146|  122|  54|  4|  1|
|      Charged Off|   16|   35|   38|   19|   8|  3|  3|
|Late (31-120 days)|   68|  164|  234|  220| 142| 14|  7|
|          Current|36639|34139|29323|15823|5352|321| 79|
| Late (16-30 days)|   26|   78|  102|   81|  46|  9|  2|
+-----------------+-----+-----+-----+-----+----+---+---+
```

3.Frequency

```
freq=df_sel.stat.freqItems(['purpose','grade'],0.3)
freq.collect()

[Row(purpose_freqItems=['debt_consolidation', 'credit_card', 'other'], grade_freqItems=['A', 'B', 'C'])]
```

Aggregate Functions:

1.Checking how many null values are present in the column

```
#Checking how many null values are present in the column
from pyspark.sql.functions import isnan,when,count,col
df_sel.select([count(when(isnan(c)|col(c).isNull(),c)).alias(c) for c in df_sel.columns]).show()
```

```
+----+--------------+-----+-------+--------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+------------------+---------+
|term|home_ownership|grade|purpose|int_rate|addr_state|loan_status|application_type|loan_amnt|emp_length|annual_inc|dti|delinq_2yrs|revol_util|total_acc|num_tl_90g_dpd_24m|dti_joint|
+----+--------------+-----+-------+--------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+------------------+---------+
|   0|             0|    0|      0|       0|         0|          0|               0|        0|         0|         0|237|          0|       156|        0|                 0|   111630|
+----+--------------+-----+-------+--------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+------------------+---------+
```

Cleaning the 'revol_utill' column

```
#DTI Column
df_sel.describe("dti","revol_util").show()
```

```
+-------+-----------------+----------+
|summary|              dti|revol_util|
+-------+-----------------+----------+
|  count|           128175|    128256|
|   mean|19.933177530719778|      null|
| stddev|20.143542243475476|      null|
|    min|              0.0|        0%|
|    max|            999.0|    99.90%|
+-------+-----------------+----------+
```

Analysis:

```
#DTI Column
df_sel.describe("dti","revol_util").show()
```

```
+-------+------------------+----------+
|summary|               dti|revol_util|
+-------+------------------+----------+
|  count|            128175|    128256|
|   mean|19.933177530719778|      null|
| stddev|20.143542243475476|      null|
|    min|               0.0|        0%|
|    max|             999.0|    99.90%|
+-------+------------------+----------+
```

In the above image it shows the max value as 99 but in depth when analyzed,

```
from pyspark.sql.functions import regexp_replace, regexp_extract
#spark.sql("select ceil(regexp_replace(revol_util,'\%',"")), count(*) from loanstatus_sel group by ceil(regexp_replace(revol_util,'\%',""))")
spark.sql("select ceil(regexp_replace(revol_util, '\%', '')), count(*) from loanstatus_sel group by ceil(regexp_replace(revol_util, '\%', ''))").show()
```

```
+---------------------------------------+--------+
|CEIL(regexp_replace(revol_util, \%, , 1))|count(1)|
+---------------------------------------+--------+
|                                     29|    1824|
|                                     26|    1776|
|                                     65|    1297|
|                                     54|    1582|
|                                     19|    1537|
|                                      0|    1132|
|                                    112|       2|
|                                     22|    1665|
|                                      7|     944|
|                                     77|     964|
|                                     34|    1909|
|                                    184|       1|
|                                    126|       1|
|                                     94|     536|
|                                     50|    1697|
|                                    110|       3|
|                                     57|    1586|
|                                     32|    1766|
|                                     43|    1729|
|                                     84|     794|
+---------------------------------------+--------+
only showing top 20 rows
```

It shows the max value as 184 but we can get only as 99. The reason behind this is it takes this as a **String[So that 9>1].** Now we need to convert it into **int**.

After cleaning we got the below output

```
[26] df_sel=df_sel.withColumn("revolutil_cleaned",regexp_extract(col("revol_util"),"\\d+",0))
```

```
df_sel.describe('revol_util','revolutil_cleaned').show()

+-------+----------+-----------------+
|summary|revol_util| revolutil_cleaned|
+-------+----------+-----------------+
|  count|    128256|           128256|
|   mean|      null| 43.76206961077844|
| stddev|      null|24.801849528207015|
|    min|        0%|                0|
|    max|    99.90%|               99|
+-------+----------+-----------------+
```

'revol_util' is the column before cleaned. Once cleaned we can see the 'mean&stddev' values in it.

But still the max value is 99 since we haven't converted it into 'double'/'int'.

Filling the null vales with average values present in it.

```
#Defining a function since the columns may conatin a lot of null values in it
def fill_avg(df,colname):
  return df.select(colname).agg(avg(colname))
```

```
rev_avg=fill_avg(df_sel,'revolutil_cleaned')
```

Using Coalesce function.

```
#coalesce
'''
  It will take two variables
    -if the first is not null it will take it.
    -if the first is null it will take the second variable
'''
from pyspark.sql.functions import coalesce
df_sel=df_sel.withColumn('revolutil_cleaned',coalesce(col('revolutil_cleaned'),col('rev_avg')))
'''
In the above one['revolutil_cleaned'] it will take all the not null values  and whereever there is null it will use the 'rev_avg'.
So where ever there is a null it will substitute with the average(will convert the two columns into single columns)
'''
```

Converting the 'revolutil_cleaned' column to 'double' and displaying the describe function.

```
[38] #converting the datatype into double for the column 'revolutil_cleaned'
     df_sel=df_sel.withColumn('revolutil_cleaned',df_sel['revolutil_cleaned'].cast('double'))
```

```
df_sel.describe('revol_util','revolutil_cleaned').show()

+-------+----------+-----------------+
|summary|revol_util| revolutil_cleaned|
+-------+----------+-----------------+
|  count|    128256|           128412|
|   mean|      null|43.76206961077852 5|
| stddev|      null|24.786779696453955|
|    min|        0%|              0.0|
|    max|    99.90%|            183.0|
+-------+----------+-----------------+
```

From the above picture we could able to see now **max** is showing the correct value 183.0 where the old value is 99.90%.

DTI Column:

```
#Checking how many null values are present in the column
from pyspark.sql.functions import isnan,when,count,col
df_sel.select([count(when(isnan(c)|col(c).isNull(),c)).alias(c) for c in df_sel.columns]).show()

+----+--------------+-----+-------+--------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+------------------+---------+-
|term|home_ownership|grade|purpose|int_rate|addr_state|loan_status|application_type|loan_amnt|emp_length|annual_inc|dti|delinq_2yrs|revol_util|total_acc|num_tl_90g_dpd_24m|dti_joint|r
+----+--------------+-----+-------+--------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+------------------+---------+-
|   0|             0|    0|      0|       0|         0|          0|               0|        0|         0|         0|237|          0|       156|        0|                 0|   111630|
+----+--------------+-----+-------+--------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+------------------+---------+-
```

Still DTI column is showing it has 237 null values, Lets check which values are null.

```
#checking which values are null in the 'dti' column
spark.sql("select * from loanstatus_sel where dti is null").show()

+---------+--------------+-----+------------------+--------+----------+-----------+----------------+---------+----------+----------+----+-----------+----------+---------+----------
|     term|home_ownership|grade|           purpose|int_rate|addr_state|loan_status|application_type|loan_amnt|emp_length|annual_inc| dti|delinq_2yrs|revol_util|total_acc|num_tl_90g
+---------+--------------+-----+------------------+--------+----------+-----------+----------------+---------+----------+----------+----+-----------+----------+---------+----------
| 60 months|      MORTGAGE|    B|    major_purchase|  10.72%|        AZ|    Current|       Joint App|    13000|       n/a|       0.0|null|          0|       26%|       47|
| 60 months|          RENT|    C|debt_consolidation|  16.91%|        CA|    Current|       Joint App|    18000|       n/a|       0.0|null|          0|    35.20%|       12|
| 60 months|      MORTGAGE|    C|debt_consolidation|  16.91%|        NY| Fully Paid|       Joint App|    35000|       n/a|       0.0|null|          0|    90.10%|       39|
| 36 months|          RENT|    C|             other|  13.56%|        CA|    Current|       Joint App|     5500|       n/a|       0.0|null|          0|       13%|       17|
| 36 months|      MORTGAGE|    B|debt_consolidation|  10.33%|        TX|    Current|       Joint App|     4700|       n/a|       0.0|null|          0|     4.40%|       15|
| 36 months|          RENT|    A|debt_consolidation|   7.56%|        UT|    Current|       Joint App|    10800|       n/a|       0.0|null|          0|    77.80%|       20|
| 36 months|      MORTGAGE|    B|debt_consolidation|  10.72%|        CA|    Current|       Joint App|     6000| 10+ years|       0.0|null|          1|       39%|       22|
| 36 months|      MORTGAGE|    C|debt_consolidation|  14.47%|        GA|    Current|       Joint App|     5000|       n/a|       0.0|null|          1|    86.20%|       15|
| 36 months|          RENT|    E|debt_consolidation|  23.40%|        NY|    Current|       Joint App|    40000|       n/a|       0.0|null|          0|    93.90%|       27|
| 36 months|          RENT|    D|       credit_card|  18.94%|        CA|    Current|       Joint App|    35000|       n/a|       0.0|null|          0|    60.50%|       15|
| 36 months|      MORTGAGE|    A|       credit_card|   7.56%|        NJ|    Current|       Joint App|    20000|       n/a|       0.0|null|          0|    65.20%|       19|
| 36 months|      MORTGAGE|    B|debt_consolidation|  11.31%|        TN|    Current|       Joint App|    30000|       n/a|       0.0|null|          0|    75.80%|       19|
| 60 months|      MORTGAGE|    D|debt_consolidation|  20.89%|        AZ|    Current|       Joint App|    28800|       n/a|       0.0|null|          0|        0%|       30|
| 36 months|      MORTGAGE|    E|debt_consolidation|  25.34%|        KS|    Current|       Joint App|    40000|       n/a|       0.0|null|          0|    97.50%|        9|
| 60 months|      MORTGAGE|    D|debt_consolidation|  17.97%|        CO|    Current|       Joint App|    10000|       n/a|       0.0|null|          3|       76%|       26|
| 36 months|          RENT|    B|debt_consolidation|  11.80%|        CA|    Current|       Joint App|    12000|       n/a|       0.0|null|          0|   100.40%|        9|
| 36 months|      MORTGAGE|    A|             other|   7.56%|        CA|    Current|       Joint App|     7000|       n/a|       0.0|null|          0|     3.70%|       16|
| 60 months|      MORTGAGE|    C|debt_consolidation|  15.02%|        FL|    Current|       Joint App|    10000|       n/a|       0.0|null|          0|    98.30%|       10|
| 60 months|      MORTGAGE|    B|       credit_card|  10.72%|        SC|    Current|       Joint App|    36000|       n/a|       0.0|null|          0|    13.70%|        9|
| 60 months|      MORTGAGE|    D|debt_consolidation|  18.94%|        NY|    Current|       Joint App|    16000|       n/a|       0.0|null|          0|    49.50%|       54|
+---------+--------------+-----+------------------+--------+----------+-----------+----------------+---------+----------+----------+----+-----------+----------+---------+----------
```

When check we can see wherever the **application_type='Joint App'** the dti value is null.

Looking for Insights:

```
spark.sql("select application_type,dti,dti_joint from loanstatus_sel where dti is null").show()
```

```
+----------------+----+---------+
|application_type| dti|dti_joint|
+----------------+----+---------+
|       Joint App|null|    33.06|
|       Joint App|null|    17.67|
|       Joint App|null|     27.3|
|       Joint App|null|     8.74|
|       Joint App|null|    11.68|
|       Joint App|null|    28.01|
|       Joint App|null|    15.06|
|       Joint App|null|    12.79|
|       Joint App|null|     8.05|
|       Joint App|null|    24.22|
|       Joint App|null|    20.04|
|       Joint App|null|    12.75|
|       Joint App|null|    24.89|
|       Joint App|null|    22.61|
|       Joint App|null|    19.01|
|       Joint App|null|    15.34|
|       Joint App|null|      0.6|
|       Joint App|null|    23.69|
|       Joint App|null|    17.29|
|       Joint App|null|    29.25|
+----------------+----+---------+
only showing top 20 rows
```

From the above insights,
we are going to create a new column name 'dti_cleaned' in that with the help of coalesce function, 'dti' records null values will be filled with the values of 'dti_joint'.

```
#from the above insights ,
'''
we are going to create a new column name 'dti_cleaned' in that with the help of coalesce function, 'dti' records null values will be filled with the vlaues of 'dti_joint'
'''
df_sel=df_sel.withColumn('dti_cleaned',coalesce(col("dti"),col("dti_joint")))
#checking the result
from pyspark.sql.functions import isnan,when,count,col
df_sel.select([count(when(isnan(c)|col(c).isNull(),c)).alias(c) for c in df_sel.columns]).show()
```

```
------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+----------------+---------+----------------+-------+-----------+
nt_rate|addr_state|loan_status|application_type|loan_amnt|emp_length|annual_inc|dti|delinq_2yrs|revol_util|total_acc|num_tl_90g_dpd_24m|dti_joint|revolutil_cleaned|rev_avg|dti_cleaned|
------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+----------------+---------+----------------+-------+-----------+
      0|         0|          0|               0|        0|         0|         0|237|          0|       156|        0|               0|   111630|               0|      0|          0|
------+----------+-----------+----------------+---------+----------+----------+---+-----------+----------+---------+----------------+---------+----------------+-------+-----------+
```

Cleaning 'loan_status'

```
df_sel.groupby('loan_status').count().show()
```

```
+------------------+------+
|       loan_status| count|
+------------------+------+
|        Fully Paid|  4908|
|   In Grace Period|   513|
|       Charged Off|   122|
|Late (31-120 days)|   849|
|           Current|121676|
| Late (16-30 days)|   344|
+------------------+------+
```

From this we can understand only 'Fully Paid' alone shows that it has full paid, other values are showing like the payment is delayed then **Bad loan.**

```
#adding a new column as bad loan
from pyspark.sql import SparkSession
from pyspark.sql.functions import when
df_sel=df_sel.withColumn("bad loan",when(df_sel.loan_status.isin(["Late (31-120 days)", "Charged Off","In Grace Period","Late (16-30 days)"]),'Yes').otherwise("No"))
```

```
df_sel.groupby('bad loan').count().show()
```

```
+--------+------+
|bad loan| count|
+--------+------+
|      No|126584|
|     Yes|  1828|
+--------+------+
```

From this we can see majority of loan is good loan.
When checking on the bad loan we can see the intrest rate are pretty high.

```
df_sel.filter(df_sel.bad_loan == 'Yes').show()
```

```
+---------+--------------+-----+------------------+--------+----------+------------------+----------------+---------+----------+----------+-----+-----------+----------+---------+-----
|     term|home_ownership|grade|           purpose|int_rate|addr_state|       loan_status|application_type|loan_amnt|emp_length|annual_inc|  dti|delinq_2yrs|revol_util|total_acc|num_t
+---------+--------------+-----+------------------+--------+----------+------------------+----------------+---------+----------+----------+-----+-----------+----------+---------+-----
| 36 months|          RENT|    B|debt_consolidation|  10.33%|        PA|Late (31-120 days)|      Individual|    16000|   3 years|   71000.0|16.58|          0|    11.80%|       28|
| 60 months|      MORTGAGE|    D|debt_consolidation|  22.35%|        OH|Late (31-120 days)|       Joint App|    17500|   9 years|   50000.0|36.92|          0|    35.40%|       37|
| 60 months|          RENT|    C|debt_consolidation|  15.02%|        NY|Late (31-120 days)|      Individual|    30000|   2 years|   90000.0|14.28|          0|    14.90%|       10|
| 36 months|      MORTGAGE|    A|debt_consolidation|   8.19%|        PA|Late (31-120 days)|      Individual|    20975|   2 years|  165000.0|24.27|          0|    16.60%|       21|
| 60 months|          RENT|    A|debt_consolidation|   8.19%|        CA|Late (31-120 days)|      Individual|    14000|  < 1 year|   65000.0|13.48|          0|    39.10%|       18|
| 36 months|          RENT|    A|debt_consolidation|   8.19%|        WA|Late (31-120 days)|      Individual|    14000|    1 year|   36000.0|25.83|          0|    73.60%|       11|
| 36 months|          RENT|    B|          vacation|  11.80%|        NC|Late (31-120 days)|      Individual|    20000|   3 years|   50000.0| 9.18|          0|    13.90%|       26|
| 36 months|          RENT|    B|debt_consolidation|  11.80%|        NJ|Late (31-120 days)|      Individual|    32000|  < 1 year|   50000.0|  6.1|          0|    50.30%|       28|
| 36 months|           OWN|    E|             other|  26.31%|        IN|Late (31-120 days)|      Individual|     9100|   3 years|   62000.0|13.38|          0|    12.10%|       23|
| 36 months|          RENT|    C|       credit_card|  13.56%|        OR|Late (31-120 days)|      Individual|     5000|    1 year|   50000.0|16.77|          0|    30.50%|        9|
| 60 months|          RENT|    C|    major_purchase|  14.47%|        CA|Late (31-120 days)|      Individual|    36000|   9 years|   93600.0| 4.14|          0|     0.30%|       29|
| 36 months|          RENT|    E|       credit_card|  24.37%|        NY|Late (31-120 days)|      Individual|    19500| 10+ years|   56000.0|29.83|          0|       66%|       33|
| 36 months|          RENT|    E|          vacation|  25.34%|        NY|Late (31-120 days)|      Individual|     1500|  < 1 year|   20000.0| 9.24|          0|    83.20%|        3|
| 36 months|      MORTGAGE|    E|debt_consolidation|  25.34%|        GA|Late (31-120 days)|      Individual|     3525| 10+ years|  150000.0|14.25|          0|    31.10%|       15|
| 60 months|      MORTGAGE|    C|debt_consolidation|  16.91%|        CA|Late (31-120 days)|      Individual|    11000| 10+ years|   97000.0|28.79|          0|    31.70%|       32|
| 36 months|          RENT|    C|debt_consolidation|  13.56%|        MT|Late (31-120 days)|      Individual|    11500|   8 years|   33000.0| 2.55|          0|       20%|       12|
| 36 months|      MORTGAGE|    A|debt_consolidation|   8.81%|        CA|Late (31-120 days)|      Individual|    22000|   5 years|  165000.0|15.59|          0|    46.40%|       46|
| 60 months|           OWN|    D|debt_consolidation|  17.97%|        NV|Late (31-120 days)|      Individual|    19200|   9 years|   85000.0|39.11|          4|    28.30%|       53|
| 60 months|          RENT|    B|debt_consolidation|  11.80%|        IN|Late (31-120 days)|      Individual|    40000|   5 years|  130000.0|17.88|          0|    78.30%|       16|
| 36 months|      MORTGAGE|    A|       credit_card|   6.46%|        WA|       Charged Off|      Individual|     8000| 10+ years|   55000.0|17.83|          0|    29.10%|       49|
+---------+--------------+-----+------------------+--------+----------+------------------+----------------+---------+----------+----------+-----+-----------+----------+---------+-----
only showing top 20 rows
```

Dropping the column:

```
[19] #Dropping the unnecessary columns
     df_sel_final=df_sel.drop('revol_util','dti','dti_joint','bad loan')
```

```
df_sel_final.printSchema()
```

```
root
 |-- term: string (nullable = true)
 |-- home_ownership: string (nullable = true)
 |-- grade: string (nullable = true)
 |-- purpose: string (nullable = true)
 |-- int_rate: string (nullable = true)
 |-- addr_state: string (nullable = true)
 |-- loan_status: string (nullable = true)
 |-- application_type: string (nullable = true)
 |-- loan_amnt: integer (nullable = true)
 |-- emp_length: string (nullable = true)
 |-- annual_inc: double (nullable = true)
 |-- delinq_2yrs: integer (nullable = true)
 |-- total_acc: integer (nullable = true)
 |-- num_tl_90g_dpd_24m: integer (nullable = true)
 |-- bad_loan: string (nullable = false)
```

Storing data in a permanent table

```
#permanent table
#saving table in the parquet foramt
permanent_table='lc_loan_data'
df_sel.write.format("parquet").saveAsTable(permanent_table)
```

Notes:

1. Data frames in spark works in a distributed format like they will divide into chunks of parts and they assign it to worker node.
2. Data frames in Pandas is mostly dependent on Memory.
3. Data frames in spark has more function and used well for analyzing than SQL.
4. Spark is an lazy evaluation framework creates a DAG and then an action command comes and executes the DAG.So that it optimizes the DAG in the distributed way.
5. Cache()-Stores the df in the memory , subsequent operation can be performed faster without having to recompute the dataframe from source. If we cache too many dataframes will lead to excessive memory consumption and cause memory related issue.