Payal Patel PySpark Assignment 1/27/2020

In [1]:
```
spark.version
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	
0	application_1580143975927_0001	pyspark	idle	Link (http://ip63.ec2.internal:20888/proxy/application_1580143975

```
SparkSession available as 'spark'.
```

```
'2.4.4'
```

In [2]:
```
# I - load loans data into pyspark dataframe
loans_df = spark.read.csv('s3a://spark01272020/loan_200k.csv', header=True)
```

In [3]:
```
# II - show first 10 rows
loans_df.show(10)
```

```
+---------+---------+--------------+-------------+-------+------+-------+----
--------------+----------+
|member_id|loan_amnt|term_in_months|interest_rate|payment|income|default|
purpose|addr_state|
+---------+---------+--------------+-------------+-------+------+-------+----
--------------+----------+
|   123685|     1800|            36|        17.22|  64.38|  1896|      0|debt
_consolidation|        AZ|
|   139937|      500|            36|         9.01|  15.91|  2000|      1|
other|        AR|
|   288338|      500|            36|            8|  15.67|  3300|      1|
educational|        GA|
|   228911|     1600|            36|         7.43|  49.72|  3500|      1|
other|        CT|
|   252052|     1525|            36|        10.71|  49.72|  3600|      1|
moving|        NY|
|   518840|     2000|            36|        13.22|  67.61|  4000|      0|
educational|        FL|
|    93312|     1200|            36|        10.28|  38.88|  4000|      1|
educational|        FL|
|    91067|     5350|            36|        13.12| 180.57|  4000|      0|
car|        NJ|
|   678265|     1400|            36|        11.86|  46.41|  4080|      1|debt
_consolidation|        CA|
|   647711|     1200|            36|        12.73|  40.28|  4200|      0|
educational|        VA|
+---------+---------+--------------+-------------+-------+------+-------+----
--------------+----------+
only showing top 10 rows
```

In [4]:
```
# III - number of records in dataframe
loans_df.count()
```

```
200000
```

In [5]:
```
# IV - descriptive stats of loans dataframe
loans_df.describe().show()
```

```
+-------+-----------------+--------------+-----------------+-----------------
--+-----------------+------------------+-------------------+-------+---------
-+
|summary|        member_id|     loan_amnt|   term_in_months|     interest_ra
te|          payment|            income|            default|purpose|addr_stat
e|
+-------+-----------------+--------------+-----------------+-----------------
--+-----------------+------------------+-------------------+-------+---------
-+
|  count|           200000|        200000|           200000|            2000
00|           200000|            200000|             200000| 200000|    20000
0|
|   mean|    5498547.85784|    13839.48175|         41.94852|13.9820232500010
42|423.0070926999864| 72181.61598585005|           0.133875|   null|      nul
l|
| stddev|3596746.684002879|8088.7917552115|10.36243826086473|4.38397373818412
55|240.9242967957215|55869.731296036894|0.34051881613255913|   null|      nul
l|
|    min|          1000004|          1000|               36|
10|              100|             10000|                  0|    car|        A
K|
|    max|           999984|          9975|               60|              9.
99|           999.68|             99999|                  1|wedding|        W
Y|
+-------+-----------------+--------------+-----------------+-----------------
--+-----------------+------------------+-------------------+-------+---------
-+
```

In [8]:
```
# V - Show member_id and 0 or 1 depending on loan_amnt > 5000.
from pyspark.sql import functions as F
loans_df.select('member_id', F.when(loans_df.loan_amnt >5000,1))
```

```
DataFrame[member_id: string, CASE WHEN (loan_amnt > 5000) THEN 1 END: int]
```

In [10]:
```
# VI - create new column, 'new_amnt'
from pyspark.sql.functions import floor
loans_df = loans_df.withColumn("new_amnt", floor("payment"))
```

In [11]:
```
# VII - load loans into temporary table
loans_df.createOrReplaceTempView("loans")
```

In [13]:
```
# VIII - for each of the purposes of the loans, show/display what is the avera
ge income?
# Order your results by descending average income
sqlDF = spark.sql("SELECT purpose, avg(income) as avg_income FROM loans group
 by purpose order by avg_income desc").show()
```

```
+------------------+------------------+
|           purpose|        avg_income|
+------------------+------------------+
|  home_improvement| 88717.95904243464|
|    small_business| 83345.74022429906|
|  renewable_energy| 82872.55910526316|
|             house| 81054.22032840723|
|    major_purchase| 72035.47574585634|
|           medical| 71842.33159580053|
|       credit_card|  71551.8185169462|
|debt_consolidation| 71251.33441512073|
|           wedding| 69392.10282977558|
|             other| 66981.70704792837|
|            moving| 65727.52776353275|
|               car|   64650.543676823|
|          vacation| 64296.40992673993|
|       educational|51232.925545023696|
+------------------+------------------+
```

In [14]:
```
#IX - show/display how many records (observations) defaulted, and how many did
NOT defaulted?
sqlDF = spark.sql("SELECT default, count(*) FROM loans group by default").show
()
```

```
+-------+--------+
|default|count(1)|
+-------+--------+
|      0|  173225|
|      1|   26775|
+-------+--------+
```

In [18]:
```
#X -for each of the following states: NC, CA, MA, TX, FL, show/display what is
the maximum interest?
sqlDF = spark.sql("SELECT addr_state, max(interest_rate) FROM loans where addr
_state in ('NC', 'CA', 'MA', 'TX', 'FL') group by addr_state").show()
```

```
+----------+------------------+
|addr_state|max(interest_rate)|
+----------+------------------+
|        CA|             9.99|
|        NC|             9.99|
|        TX|             9.99|
|        MA|             9.99|
|        FL|             9.99|
+----------+------------------+
```