CREDIT CARD TRANSACTION ANALYSIS

Team Members

Abhishek Yadav
Anurag Yadav
Himanshu Dhiman
Kaviya R
Nikita Nannaware
Shreyash Gupta

CONTENTS

1 Architecture Diagram

2 Offline Analysis

3 Cloud Setup

4 Cloud Analysis

5 Visualizations

TEAM CLUSTER

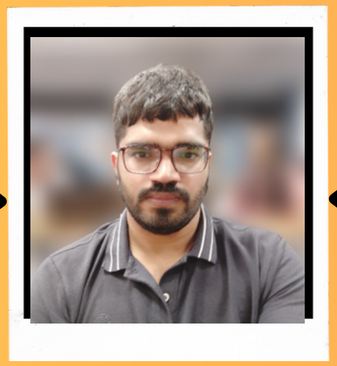
Cluster Manager

Executor 1

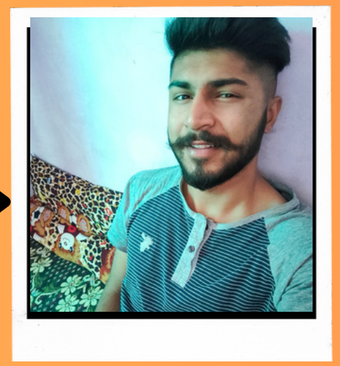
Executor 2

Executor 3

Executor 4



Executor 5



Executor 6

ARCHITECTURE DIAGRAM

SOURCE

Azure SQL

DATA INGESTION

Azure Data Factory

APACHE SQOOP

ANALYSIS

Azure Synapse Analytics

databricks

Azure Data Lake Storage Gen2

Apache Spark

Spark SQL

hadoop HDFS

VISUALIZATION

DATA DESCRIPTION

CREDIT CARD DEFAULTED. CSV

- **NO OF ROWS**: 1002
- **SIZE**: 92.2KB

- **CUSTID**: UNIQUE CUSTOMER ID
- **LIMIT_BAL**: MAXIMUM SPENDING LIMIT FOR THE CUSTOMER
- **SEX**: SEX OF THE CUSTOMER. 1 (MALE) AND 2 (FEMALE)
- **EDUCATION**: EDUCATION LEVEL OF THE CUSTOMER. **1** (GRADUATE), **2** (UNIVERSITY), **3** (HIGH SCHOOL) AND **4** (OTHERS)
- **MARRIAGE**: MARITAL STATUS OF THE CUSTOMER. **1** (SINGLE), **2** (MARRIED) AND **3** (OTHERS)
- **AGE**: AGE OF CUSTOMER
- **PAY_1 TO PAY_6**: PAYMENT STATUS FOR THE LAST 6 MONTHS
- **BILL_AMT1 TO BILL_AMT6**: THE BILLED AMOUNT FOR CREDIT CARD FOR EACH OF THE LAST 6 MONTHS.
- **PAY_AMT1 TO PAY_AMT6**: THE ACTUAL AMOUNT THE CUSTOMER PAID FOR EACH OF THE LAST 6 MONTHS
- **DEFAULTED**: WHETHER THE CUSTOMER DEFAULTED OR NOT ON THE 7TH MONTH. THE VALUES ARE 0 (DID NOT DEFAULT) AND 1 (DEFAULTED)

CREDIT CARD. CSV

- **NO OF ROWS**: 284808
- **SIZE**: 144MB

- **TIME**: TIME DIFFERENCE BETWEEN TRANSACTION TIME AND FIRST ENTRY IN DATABASE
- **V1 TO V28**: RESULT OF PCA TRANSFORMATION OF CUSTOMER INFORMATION
- **AMOUNT**: AMOUNT PAID BY CUSTOMER
- **CLASS**: WEATHER CUSTOMER IS DEFAULTER OR NOT

DATA IMOPRT USING SQOOP

```
SQOOP IMPORT
--CONNECT JDBC:SQLSERVER://ABHI23426.DATABASE.WINDOWS.NET:1433/CREDIT
--USERNAME=ABHI23426
--PASSWORD= *WHY_SO_SERIOUS_ABOUT_PASSWORDS*
--TABLE TRANSACTION_DATA
--TARGET-DIR /USER/CLOUDERA/CREDIT_CARD
```

SPARK ANALYSIS

```
import findspark
findspark.init()
from pyspark import SparkContext
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder.appName('credit card').getOrCreate()
```

```
sc = spark.sparkContextl
```

```
rdd = sc.textFile('creditcard_csv.csv')
```

```
rdd.take(2)
```

```
['Time,V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,V12,V13,V14,V15,V16,V17,V18,V19,V20,V21,V22,V23,V24,V25,V26,V27,V28,Amount,Class',
 "0,-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427443,-0.338320769942518,0.462387777762292,0.2395985540612
57,0.0986979012610507,0.363786969611213,0.0907941719789316,-0.551599533260813,-0.617800855762348,-0.991389847235408,-0.31116935
3699879,1.46817697209427,-0.470400525259478,0.207971241929242,0.0257905801985591,0.403992960255733,0.251412098239705,-0.0183067
77944153,0.277837575558899,-0.110473910188767,0.0669280749146731,0.128539358273528,-0.189114843888824,0.133558376740387,-0.0210
530534538215,149.62,'0'"]
```

Importing required Modules and packages for Analysis.
Also Reading File as text file and creating rdd out of it

Splitting data,converting time into proper datetime format and selecting only required columns

```
rdd2 = credit.map(lambda x: x.split(','))
```

```
import datetime as dt
```

```
# as we don't have any info about columnn v1 to v28 we are not selecting them

ccard = rdd2.map(lambda x: [str(dt.datetime.strptime('01-01-2023 00:00:00', '%d-%m-%Y %H:%M:%S')+\
                                dt.timedelta(seconds = float(x[0]))),\
                            float(x[-2]), int(x[-1][1])])
```

```
ccard.take(10)
```

```
[['2023-01-01 00:00:00', 149.62, 0],
 ['2023-01-01 00:00:00', 2.69, 0],
 ['2023-01-01 00:00:01', 378.66, 0],
 ['2023-01-01 00:00:01', 123.5, 0],
 ['2023-01-01 00:00:02', 69.99, 0],
 ['2023-01-01 00:00:02', 3.67, 0],
 ['2023-01-01 00:00:04', 4.99, 0],
 ['2023-01-01 00:00:07', 40.8, 0],
 ['2023-01-01 00:00:07', 93.2, 0],
 ['2023-01-01 00:00:09', 3.68, 0]]
```

1. WHAT IS THE AVERAGE AMOUNT BY CLASS ?

```
[1]: classgroup = ccard.map(lambda x: [x[2], (x[1],1)])

[1]: classgroup.take(5)

[[0, (149.62, 1)],
 [0, (2.69, 1)],
 [0, (378.66, 1)],
 [0, (123.5, 1)],
 [0, (69.99, 1)]]

[1]: class_amt = classgroup.reduceByKey(lambda a,b: [a[0]+b[0], a[1]+b[1]])

[1]: class_amt.take(5)

[(0, [25102462.040004306, 284315]), (1, [60127.97, 492])]

[1]: class_avg_amt = class_amt.mapValues(lambda x: x[0]/x[1])

[1]: class_avg_amt.take(3)

[(0, 88.29102242232842), (1, 122.21132113821139)]
```

2. WHAT IS THE HIGHEST AMOUNT BY CLASS AND TIME ?

```
# grouping based on class and hour
classtimegroup = ccard.map(lambda x: [(x[2], x[0][11:13]), x[1]] )
```

```
classtimegroup.keys().distinct().take(6)
```

```
[(1, '00'), (0, '01'), (1, '03'), (1, '04'), (1, '07'), (1, '08')]
```

```
high_amt = classtimegroup.reduceByKey(max).sortBy(lambda x: x[1], ascending = False)
```

```
high_amt.take(5)
```

```
[((0, '22'), 25691.16),
 ((0, '13'), 19656.53),
 ((0, '02'), 18910.0),
 ((0, '11'), 12910.93),
 ((0, '12'), 11898.09)]
```

3. WHAT IS THE LOWEST AMOUNT BY TIME?

```
timegroup = ccard.map(lambda x: [ x[0][11:13], x[1]])
```

```
low_amt = timegroup.reduceByKey(min).sortBy(lambda x: x[1])
```

```
low_amt.take(5)
```

```
[('00', 0.0), ('02', 0.0), ('10', 0.0), ('01', 0.0), ('06', 0.0)]
```

4. WHAT IS THE TOTAL AMOUNT BY CLASS ?

```
1  class_amt.take(5)
```

```
[(0, [25102462.040004306, 284315]), (1, [60127.97, 492])]
```

```
1  class_total_amt = class_amt.map(lambda x: (x[0], x[1][0]))
```

```
1  class_amt.take(5)
```

```
[(0, [25102462.040004306, 284315]), (1, [60127.97, 492])]
```

```
1  class_total_amt.take(5)
```

```
[(0, 25102462.040004306), (1, 60127.97)]
```

5. WHAT IS THE AVERAGE AMOUNT BY CLASS AND TIME ?

```
1 classtimegroup.take(5)
```

```
[[(0, '00'), 149.62],
 [(0, '00'), 2.69],
 [(0, '00'), 378.66],
 [(0, '00'), 123.5],
 [(0, '00'), 69.99]]
```

```
1 classtimemapgroup = classtimegroup.map(lambda x: [x[0], (x[1],1)])
```

```
1 classtimemapgroup.take(5)
```

```
[[(0, '00'), (149.62, 1)],
 [(0, '00'), (2.69, 1)],
 [(0, '00'), (378.66, 1)],
 [(0, '00'), (123.5, 1)],
 [(0, '00'), (69.99, 1)]]
```

```
1 classtimemid = classtimemapgroup.reduceByKey(lambda a,b: (a[0]+b[0], a[1]+b[1]))
```

```
1 classtimemid.take(5)
```

```
[((1, '00'), (1820.07, 6)),
 ((0, '01'), (263046.13000000035, 4210)),
 ((1, '03'), (233.91000000000003, 17)),
 ((1, '04'), (1548.1999999999998, 23)),
 ((1, '07'), (2757.779999999999, 23))]
```

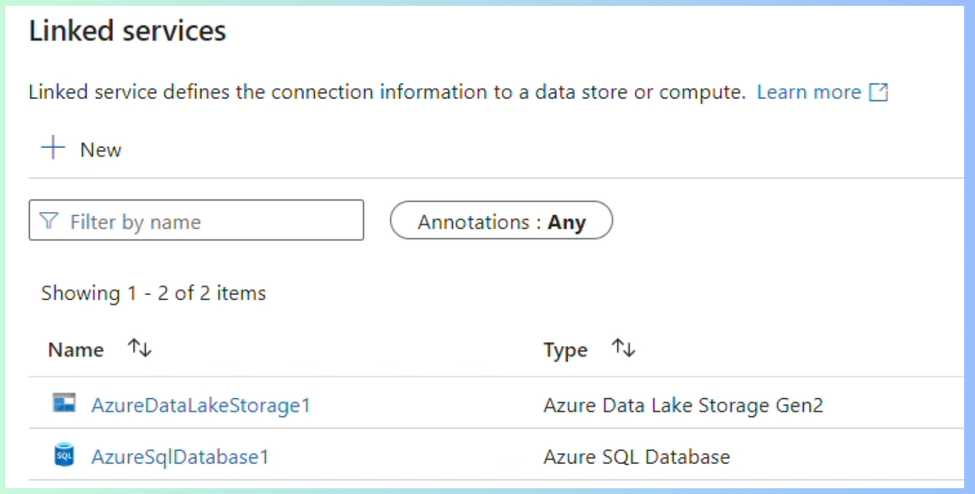
```
1 class_time_avg = classtimemid.mapValues(lambda x: x[0]/x[1])
```

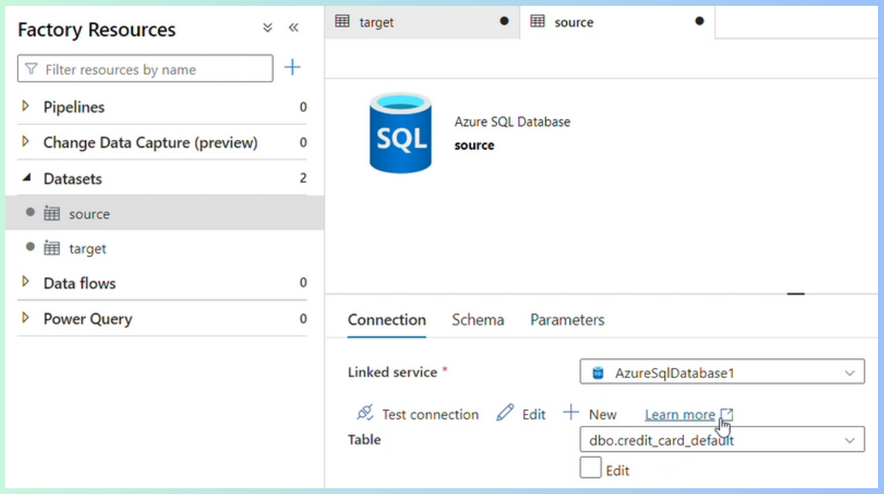
```
1 class_time_avg.take(5)
```

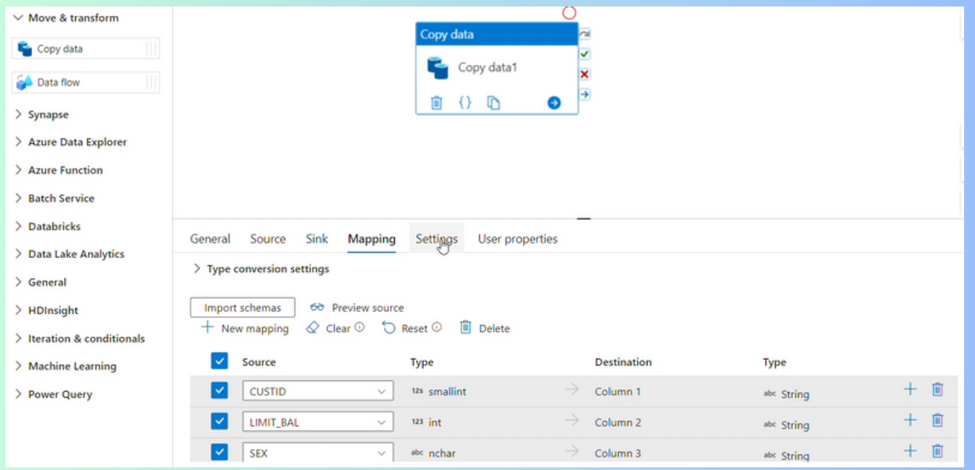
```
[((1, '00'), 303.34499999999997),
 ((0, '01'), 62.48126603325424),
 ((1, '03'), 13.759411764705884),
 ((1, '04'), 67.31304347826087),
 ((1, '07'), 119.90347826086952)]
```

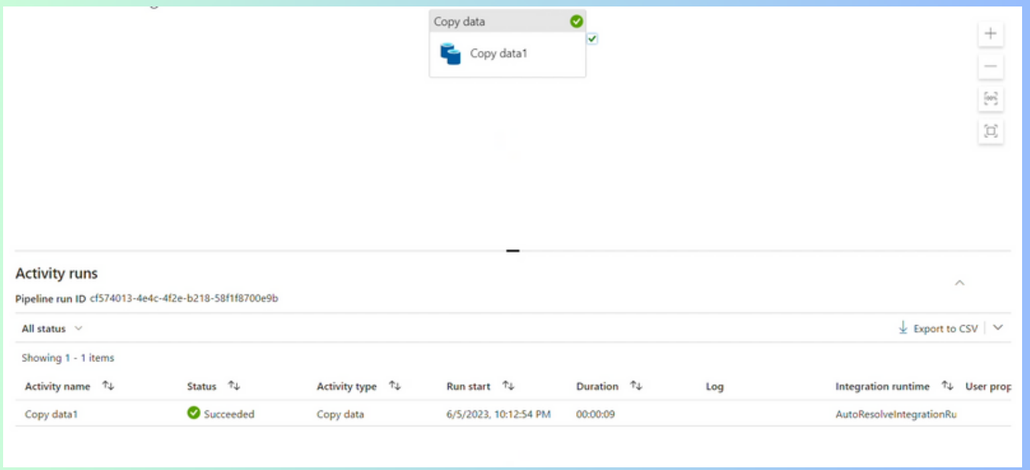
Believe Us. You're halfway there. Be with Us to the finish line.

DATA INGESTION USING AZURE DATA FACTORY









MOUNTING ADLS TO DATABRICKS

Cmd 2

```
dbutils.fs.mount(
  source= "wasbs://capston@myfirstacc18.blob.core.windows.net",
  mount_point= "/mnt/mount_point",
  extra_configs= {"fs.azure.account.key.myfirstacc18.blob.core.windows.net":"4Me+xeEDr8OnffclCH1mBHM0e6qrhHVr/E7ENo/MEBKGIYL7UZGAemftTdrM28abm27rWk5bYGME+AStx6lOrw=="}
)
```

Out[8]: True

Command took 11.34 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:05:00 PM on Himanshu Dhiman's Cluster

Cmd 3

```
dbutils.fs.ls('/mnt/mount_point')
```

Out[10]: [FileInfo(path='dbfs:/mnt/mount_point/credit-card-default-1000.csv', name='credit-card-default-1000.csv', size=94292, modificationTime=1686553962000)]

Command took 0.27 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:05:45 PM on Himanshu Dhiman's Cluster

Cmd 4

```
data = sc.textFile("dbfs:/mnt/mount_point/credit-card-default-1000.csv")
```

Command took 0.11 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:06:35 PM on Himanshu Dhiman's Cluster

Cmd 5

```
data.take(5)
```

▸ (1) Spark Jobs

Out[14]: ['CUSTID,LIMIT_BAL,SEX,EDUCATION,MARRIAGE,AGE,PAY_1,PAY_2,PAY_3,PAY_4,PAY_5,PAY_6,BILL_AMT1,BILL_AMT2,BILL_AMT3,BILL_AMT4,BILL_AMT5,BILL_AMT6,PAY_AMT1,PAY_AMT2,PAY_AMT3,PAY_AMT4,PAY_AMT5,PAY_AMT6,DEFAULTED',
 '530,20000,2,2,2,21,-1,-1,2,2,-2,-2,0,0,0,0,0,0,0,0,0,0,162000,0,0',
 '38,60000,2,2,2,22,0,0,0,0,-2,-2,0,0,0,0,0,0,0,0,0,0,1576,0',
 '43,10000,1,2,2,22,0,0,0,0,-2,-2,0,0,0,0,0,0,0,0,0,0,1500,0',
 '47,20000,2,1,2,22,0,0,2,-1,0,-1,1131,291,582,291,0,291,291,582,0,0,130291,651,0']

Command took 1.23 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:06:45 PM on Himanshu Dhiman's Cluster

DATA CLEANING

Cmd 7

```
header = data.first()
```

▸ (1) Spark Jobs

Command took 0.29 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:09:00 PM on Himanshu Dhiman's Cluster

Cmd 8

```
data = data.filter(lambda x:x!=header)
```

Command took 0.11 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:09:08 PM on Himanshu Dhiman's Cluster

Cmd 9

```
data.take(5)
```

▸ (1) Spark Jobs

```
Out[17]: ['530,20000,2,2,2,21,-1,-1,2,2,-2,-2,0,0,0,0,0,0,0,0,0,0,162000,0,0',
 '38,60000,2,2,2,22,0,0,0,0,-2,-2,0,0,0,0,0,0,0,0,0,0,1576,0',
 '43,10000,1,2,2,22,0,0,0,0,-2,-2,0,0,0,0,0,0,0,0,0,0,1500,0',
 '47,20000,2,1,2,22,0,0,0,2,-1,0,-1,1131,291,582,291,0,291,291,582,0,0,130291,651,0',
 '70,20000,1,4,2,22,2,0,0,0,-1,-1,1692,13250,433,1831,0,2891,13250,433,1831,0,2891,153504,0']
```

Command took 0.33 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:09:46 PM on Himanshu Dhiman's Cluster

Cmd 10

```
data.count()
```

▸ (1) Spark Jobs

```
Out[18]: 1000
```

Command took 0.32 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:09:53 PM on Himanshu Dhiman's Cluster

Cmd 11

```
### 3.Cleanup data. Remove lines that are not "CSV"
```

Command took 0.11 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:10:38 PM on Himanshu Dhiman's Cluster

Cmd 12

```
data = data.map(lambda x:x.split(','))
```

Command took 0.06 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:09:57 PM on Himanshu Dhiman's Cluster

Cmd 13

```
data = data.filter(lambda x:all(x))
```

Command took 0.07 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:10:53 PM on Himanshu Dhiman's Cluster

Cmd 14

```
data.count()
```

▸ (1) Spark Jobs

```
Out[23]: 1000
```

Command took 0.33 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:10:57 PM on Himanshu Dhiman's Cluster

Cmd 15

```
data.take(1)
```

▸ (1) Spark Jobs

```
Out[24]: [['530',
  '20000',
  '2',
  '2',
  '2',
  '21',
  '-1',
  '-1',
  '2',
  '2',
  '-2',
  '-2',
  '0',
  '0',
  '0',
  '0',
  '0',
  '0',
  '0',
  '0',
  '0',
```

Command took 0.30 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 1:11:03 PM on Himanshu Dhiman's Cluster

ROUNDING OF AGE TO RANGE OF 10S

Cmd 23

```
### rounding of age to range of 10s.
```

Command took 0.06 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:17:28 PM on Himanshu Dhiman's Cluster

Cmd 24

```
from pyspark.sql.functions import *
```

Command took 0.03 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:17:52 PM on Himanshu Dhiman's Cluster

Cmd 25

```
def func(x):
    x = floor(x/10)
    x = x*10
#     return (x//10)*10
    return x


df = df.withColumn('range_of_10s', func(col('AGE')) )
```

▸ df: pyspark.sql.dataframe.DataFrame = [CUSTID: long, LIMIT_BAL: long ... 24 more fields]

Command took 0.13 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:17:57 PM on Himanshu Dhiman's Cluster

Cmd 26

```
df.show(5)
```

▸ (1) Spark Jobs

CUSTID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	DEFAULTED	range_of_10s
530	20000	2	2	2	21	-1	-1	2	2	-2	-2	0	0	0	0	0	0	0	0	0	0	162000	0	0	20
38	60000	2	2	2	22	0	0	0	0	-2	-2	0	0	0	0	0	0	0	0	0	0	0	1576	0	20
43	10000	1	2	2	22	0	0	0	0	-2	-2	0	0	0	0	0	0	0	0	0	0	0	1500	0	20
47	20000	2	1	2	22	0	0	2	-1	0	-1	1131	291	582	291	0	291	291	582	0	0	130291	651	0	20
70	20000	1	4	2	22	2	0	0	0	-1	-1	1692	13250	433	1831	0	2891	13250	433	1831	0	2891	153504	0	20

only showing top 5 rows

Command took 0.56 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:18:16 PM on Himanshu Dhiman's Cluster

NORMALIZE SEX TO ONLY 1 AND 2

Cmd 27

```
### Normalize sex to only 1 and 2
```

Command took 0.03 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:18:27 PM on Himanshu Dhiman's Cluster

Cmd 28

```
df.withColumn("SEX",regexp_replace("SEX",'F','2')).show(10)
```

▸ (1) Spark Jobs

CUSTID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	DEFAULTED	range_of_10s
538	20000	2	2	2	21	-1	-1	2	2	-2	-2	0	0	0	0	0	0	0	0	0	0	162000	0	0	20
38	60000	2	2	2	22	0	0	0	0	-2	-2	0	0	0	0	0	0	0	0	0	0	0	1576	0	20
43	10000	1	2	2	22	0	0	0	0	-2	-2	0	0	0	0	0	0	0	0	0	0	0	1500	0	20
47	20000	2	1	2	22	0	0	2	-1	0	-1	1131	291	582	291	0	291	291	582	0	0	130293	651	0	20
78	20000	1	4	2	22	2	0	0	0	-1	-1	1692	13250	433	1831	0	0	2891	13250	433	1831	0	2891	153504	0	20
79	30000	2	2	2	22	0	0	0	0	-2	-2	1544	1648	1284	1868	892	-304	1300	1000	1000	1000	1000	304	39544	0	20
99	50000	2	3	1	22	0	0	0	0	-1	-1	0	0	0	0	166	541	0	0	0	166	543	4268	0	20
104	50000	2	3	2	22	0	0	0	0	-1	-1	780	0	780	390	390	500	0	780	0	390	500	18300	0	20
135	30000	2	2	2	22	0	0	0	0	-2	-2	92	92	92	0	92	0	92	92	0	92	0	1883	0	20
178	50000	2	2	2	22	0	0	0	0	2	-1	-190	-9850	-9850	10311	10161	7319	0	0	20161	0	7319	13899	0	20

only showing top 10 rows

Command took 0.53 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:18:33 PM on Himanshu Dhiman's Cluster

Cmd 29

```
df = df.withColumn("SEX",when(df['SEX'] == 'M','1').when(df['SEX'] == 'F','2').otherwise(df['SEX']))
```

▸ df: pyspark.sql.dataframe.DataFrame = [CUSTID: long, LIMIT_BAL: long ... 24 more fields]

Command took 0.15 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:18:38 PM on Himanshu Dhiman's Cluster

Cmd 30

```
df1= df.replace(['M','F'],['1','2'],'AGE')
```

▸ df1: pyspark.sql.dataframe.DataFrame = [CUSTID: long, LIMIT_BAL: long ... 24 more fields]

Command took 0.10 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:18:43 PM on Himanshu Dhiman's Cluster

AVERAGE BILLED AMOUNT

Cmd 31

```
### 7. Average billed amount
```

Command took 0.04 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:19:01 PM on Himanshu Dhiman's Cluster

Cmd 32

```
df.select(avg(col("BILL_AMT1")+col("BILL_AMT2")+col("BILL_AMT3")+col("BILL_AMT4")+col("BILL_AMT5")+col("BILL_AMT6")).alias("avg_bill_amount")).show()
```

▸ (2) Spark Jobs

```
+-----------------+
|avg_bill_amount|
+-----------------+
|      259790.697|
+-----------------+
```

Command took 1.53 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:19:05 PM on Himanshu Dhiman's Cluster

Cmd 33

AVERAGE PAY DURATION

Cmd 34

```
1  df.select(avg(col("PAY_1")+col("PAY_2")+col("PAY_3")+col("PAY_4")+col("PAY_5")+col("PAY_6")).alias("avg_pay_amount")).show()
```

▸ (2) Spark Jobs

```
+---------------+
|avg_pay_amount|
+---------------+
|         -1.185|
+---------------+
```

Command took 0.58 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:19:18 PM on Himanshu Dhiman's Cluster

ADD SEXNAME TO THE DATA USING JOINS

Cmd 38

```
### 10. Add SEX Name to the data using SQL joins
```

Command took 0.10 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:20:16 PM on Himanshu Dhiman's Cluster

Cmd 39

```
sex_name = sc.parallelize([['1','Male'], ['2', "Female"]])
```

Command took 0.07 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:20:20 PM on Himanshu Dhiman's Cluster

Cmd 40

```
sex_name_df = sex_name.toDF(["Id","SEX_NAME"])
```

▸ (2) Spark Jobs

▸ sex_name_df: pyspark.sql.dataframe.DataFrame = [Id: string, SEX_NAME: string]

Command took 0.60 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:20:28 PM on Himanshu Dhiman's Cluster

Cmd 41

```
df1 = df.join(sex_name_df,df.SEX == sex_name_df.Id ,'inner').select("*")
```

▸ df1: pyspark.sql.dataframe.DataFrame = [CUSTID: long, LIMIT_BAL: long ... 27 more fields]

Command took 0.16 seconds -- by himanshudhiman.dhiman@gmail.com at 6/12/2023, 2:20:36 PM on Himanshu Dhiman's Cluster

CREATING TEMPORARY VIEW TO DO SQL QUERIES

PERCENTAGE OF DEFAULTERES BY MARRIAGE AND EDUCATION STATUS

Cmd 58

```
spark.sql("SELECT MARRIAGE, EDUCATION, count(*) as Total, SUM(DEFAULTED) as Defaults, \
          ROUND(SUM(DEFAULTED) * 100 / count(*)) as PER_DEFAULT FROM CCDATA GROUP BY MARRIAGE,EDUCATION ORDER BY 1,2").show()
```

▸ (7) Spark Jobs

MARRIAGE	EDUCATION	Total	Defaults	PER_DEFAULT
1	1	123	71	58.0
1	2	198	105	53.0
1	3	87	52	60.0
1	4	3	2	67.0
2	1	268	69	26.0
2	2	243	65	27.0
2	3	55	24	44.0
2	4	4	2	50.0
3	1	4	4	100.0
3	2	7	3	43.0
3	3	8	6	75.0

Command took 1.74 seconds -- by himanshudhiman.dhiman@gmail.com at 6/13/2023, 10:17:21 AM on Himanshu Dhiman's Cluster

PERCENTAGE OF DEFAULTERES BY GENDER

Cmd 47

```
spark.sql("SELECT SEX_NAME,count(*) as Total,SUM(DEFAULTED) as Defaults, \
          ROUND(SUM(DEFAULTED)*100/count(*)) as PER_DEFAULT FROM CCDATA GROUP BY SEX_NAME").show()
```

▸ (7) Spark Jobs

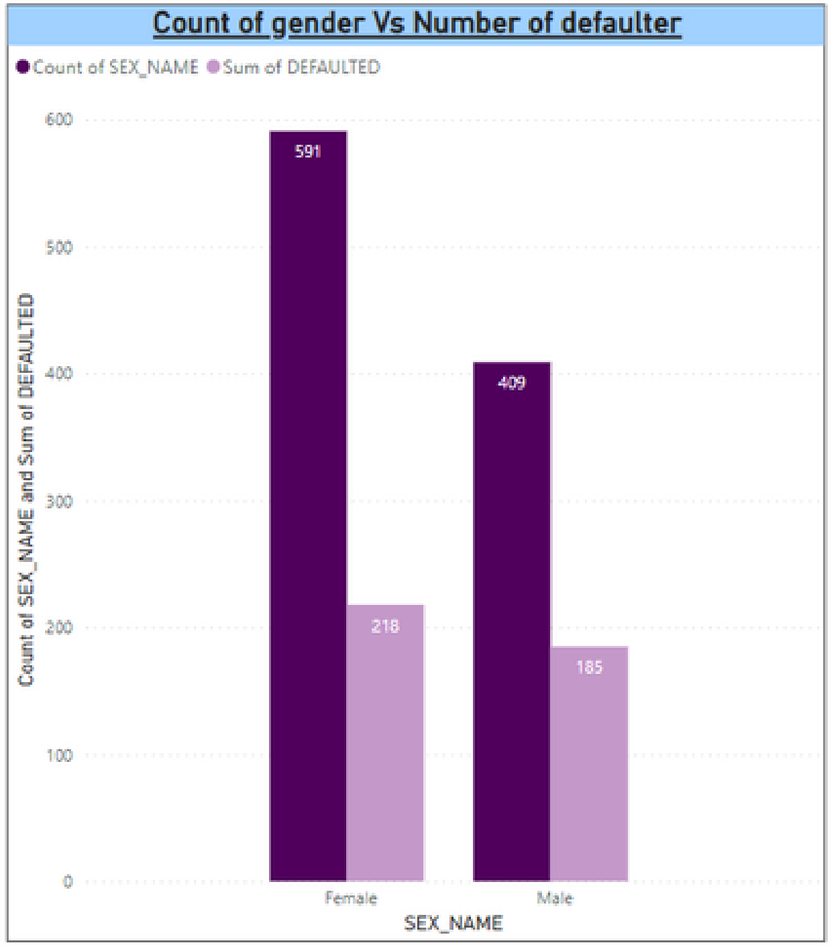
```
+--------+-----+--------+-----------+
|SEX_NAME|Total|Defaults|PER_DEFAULT|
+--------+-----+--------+-----------+
|  Female|  591|     218|       37.0|
|    Male|  409|     185|       45.0|
+--------+-----+--------+-----------+
```

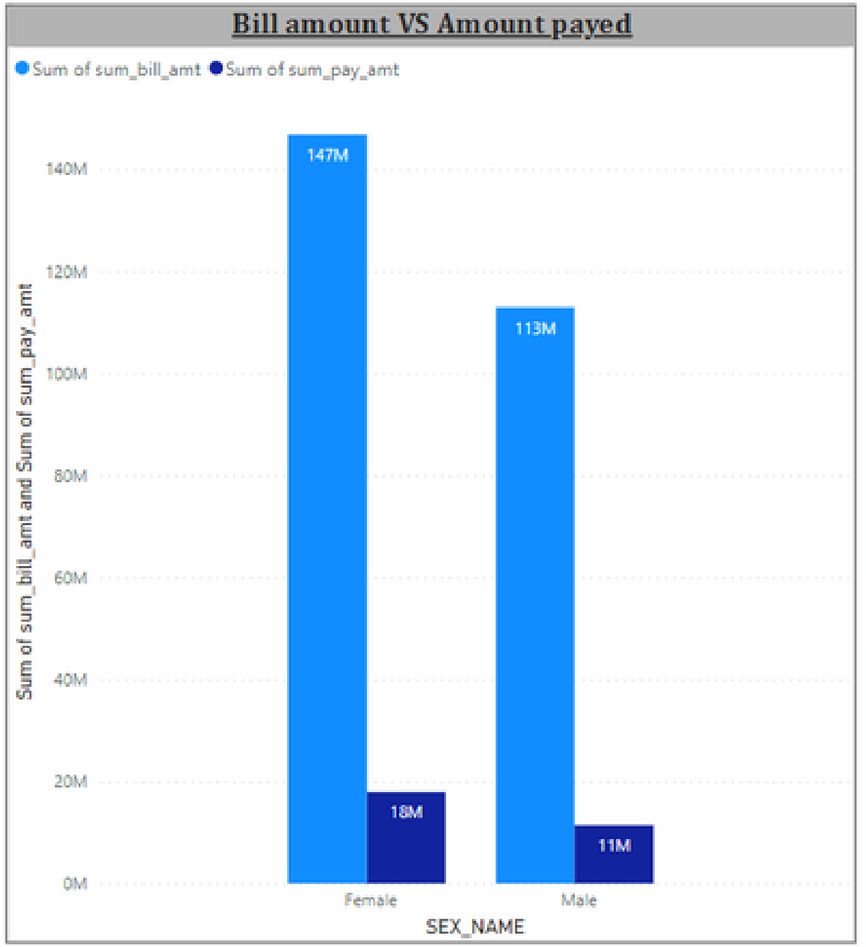
Command took 2.01 seconds -- by himanshudhiman.dhiman@gmail.com at 6/13/2023, 10:17:19 AM on Himanshu Dhiman's Cluster

Cmd 48

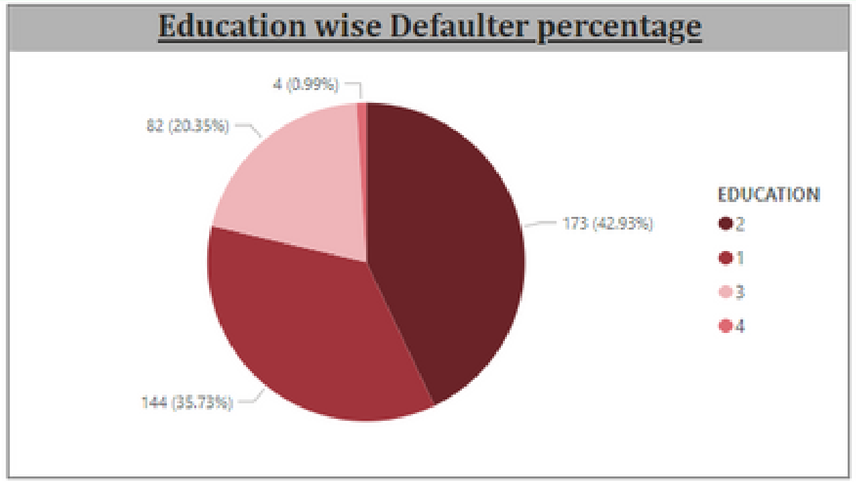
VISUALIZATION REPORTS

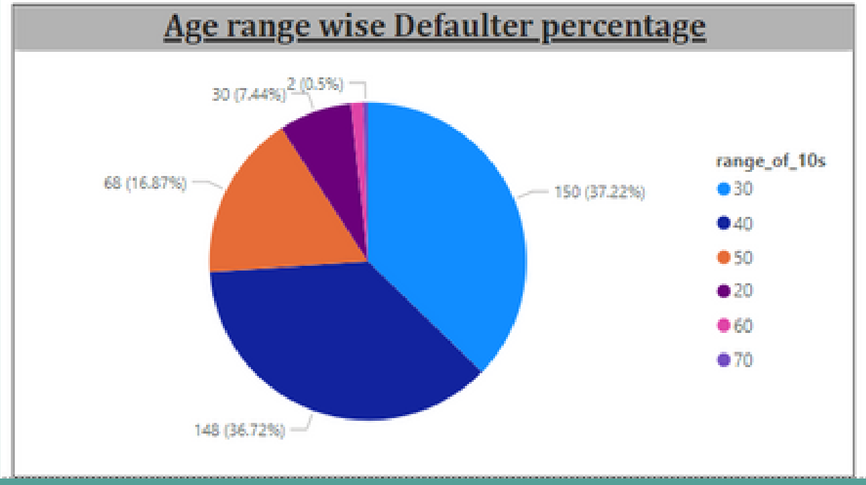
VISUALIZATION

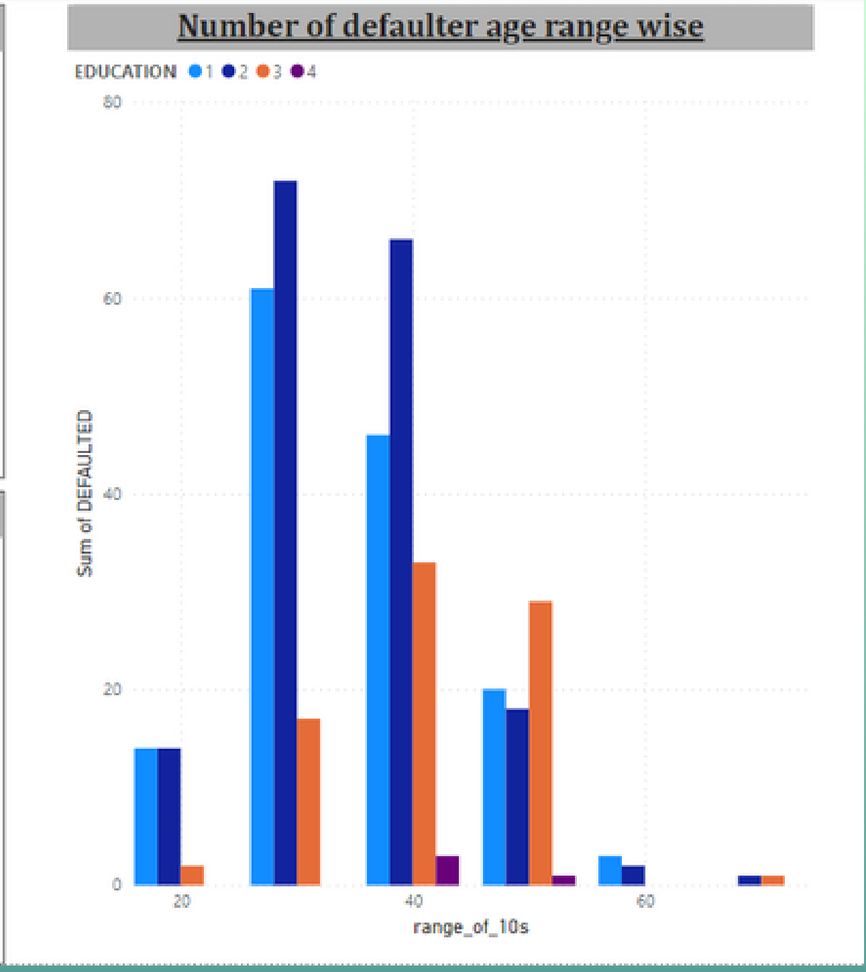




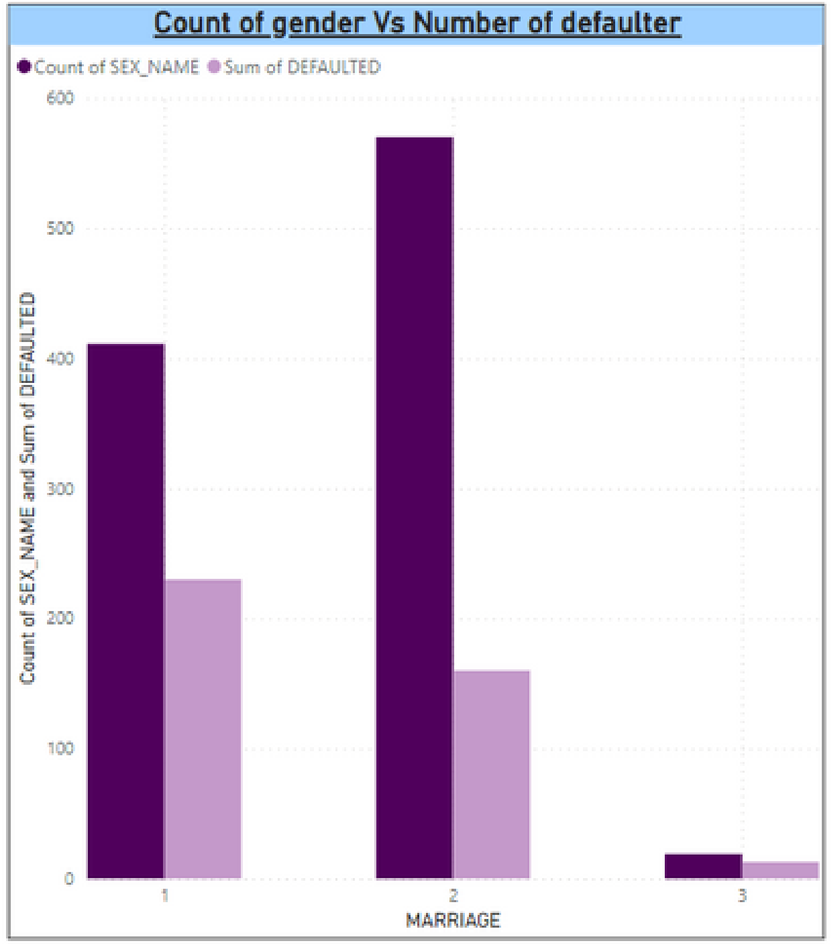
VISUALIZATION

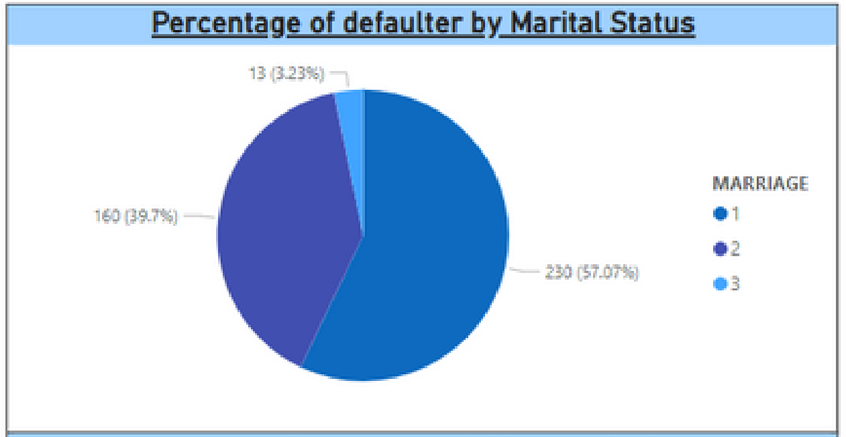


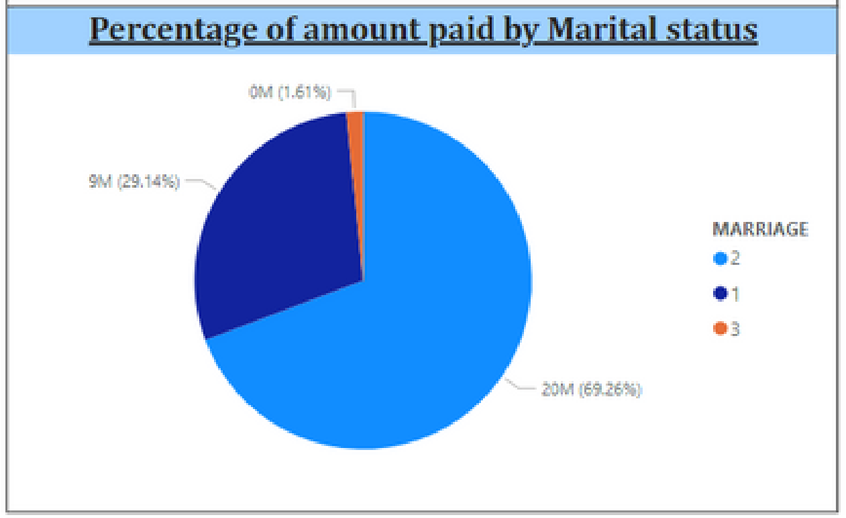




VISUALIZATION







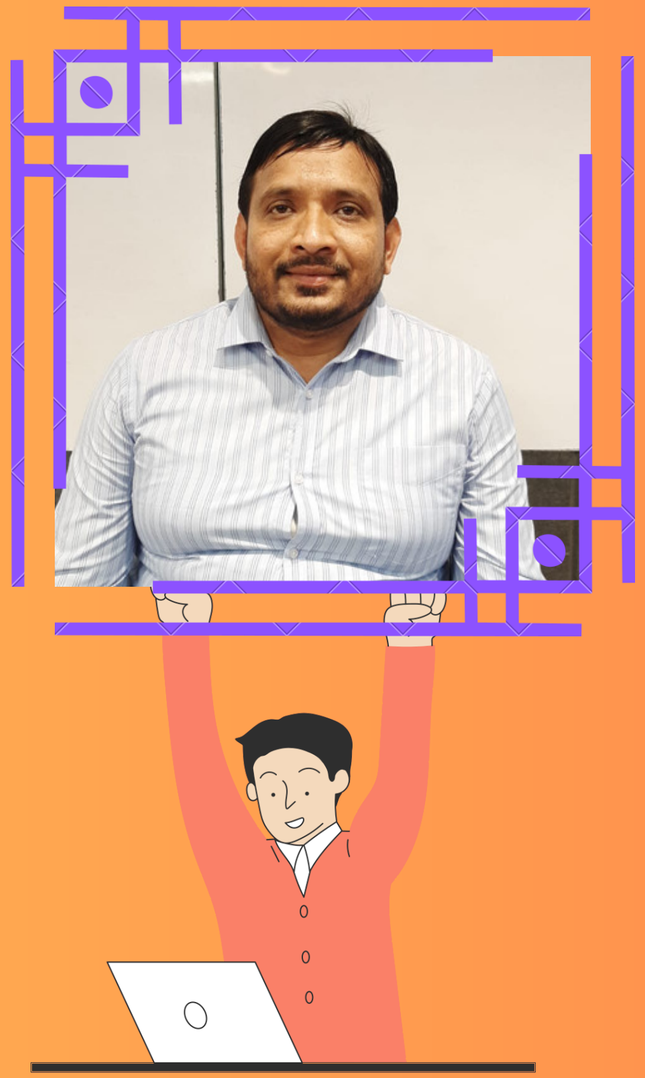
6

CONCLUSION

- PAYMENT HISTORY
- CREDIT SCORE
- DEBT-TO-INCOME RATIO
- EMPLOYMENT AND INCOME STABILITY
- PREVIOUS DEFAULT HISTORY

Let's Take a moment to thank our mentor Venkat sir.

Thank You so much sir for giving us wonderful training. Being as a good mentor, thank you for giving industrial knowledge as well as sharing your experience, not only technical stuff but also real time team work experience.
We whole heartedly thank your sir for your irreplaceable and immeasurable work that created an impact on our growth and success



Thank You..!

Futurense

THANK YOU!

Have a great day ahead.

6

ANY QUESTIONS ?