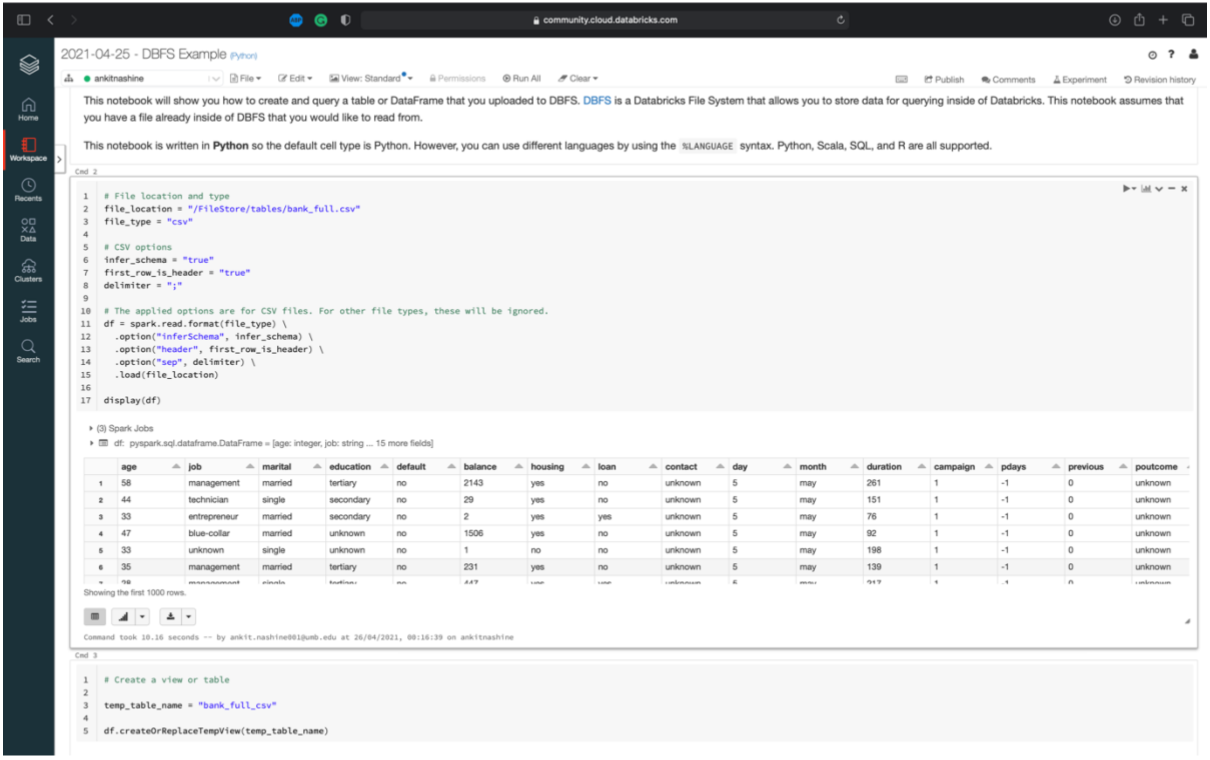
Programming Assignment – 1
ANKIT NASHINE
Data – Bank-full.csv

Importing Bank-full.csv and converting it into table.

Loading bank-full.csv into databricks and writing codes to convert it to table and viewing it.



This notebook will show you how to create and query a table or DataFrame that you uploaded to DBFS. DBFS is a Databricks File System that allows you to store data for querying inside of Databricks. This notebook assumes that you have a file already inside of DBFS that you would like to read from.

This notebook is written in **Python** so the default cell type is Python. However, you can use different languages by using the `%LANGUAGE` syntax. Python, Scala, SQL, and R are all supported.

Cmd 2

```
# File location and type
file_location = "/FileStore/tables/bank_full.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ";"

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [age: integer, job: string ... 15 more fields]

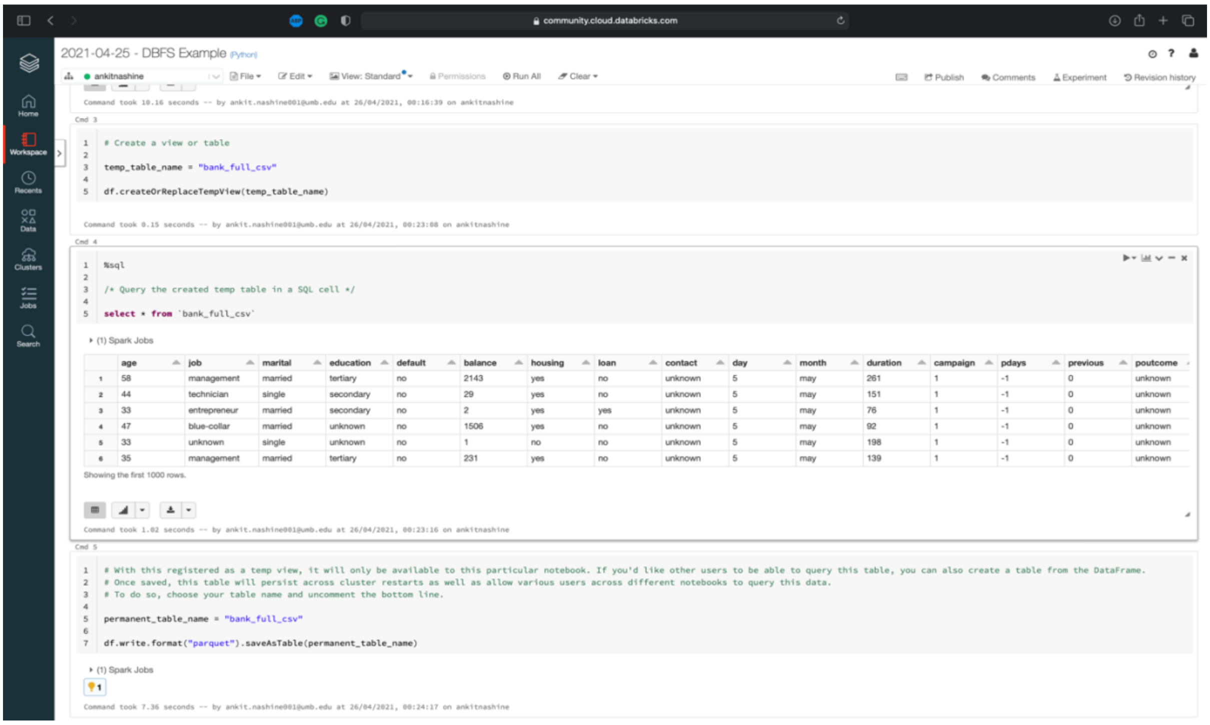
	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
1	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
2	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
3	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown
4	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown
5	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown
6	35	management	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	unknown
7	28	management	single	tertiary	no	447	yes	yes	unknown	5	may	217	1	-1	0	unknown

Showing the first 1000 rows.

```
Command took 10.16 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 00:16:39 on ankitnashine
```

Cmd 3

```
# Create a view or table

temp_table_name = "bank_full_csv"

df.createOrReplaceTempView(temp_table_name)
```



Command took 10.16 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 00:16:39 on ankitnashine

Cmd 3

Create a view or table

temp_table_name = "bank_full_csv"

df.createOrReplaceTempView(temp_table_name)

```
Command took 0.15 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 00:23:08 on ankitnashine
```

Cmd 4

```
%sql

/* Query the created temp table in a SQL cell */

select * from `bank_full_csv`
```

(1) Spark Jobs

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
1	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
2	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
3	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown
4	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown
5	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown
6	35	management	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	unknown

Showing the first 1000 rows.

```
Command took 1.02 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 00:23:16 on ankitnashine
```

Cmd 5

```
# With this registered as a temp view, it will only be available to this particular notebook. If you'd like other users to be able to query this table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow various users across different notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "bank_full_csv"

df.write.format("parquet").saveAsTable(permanent_table_name)
```

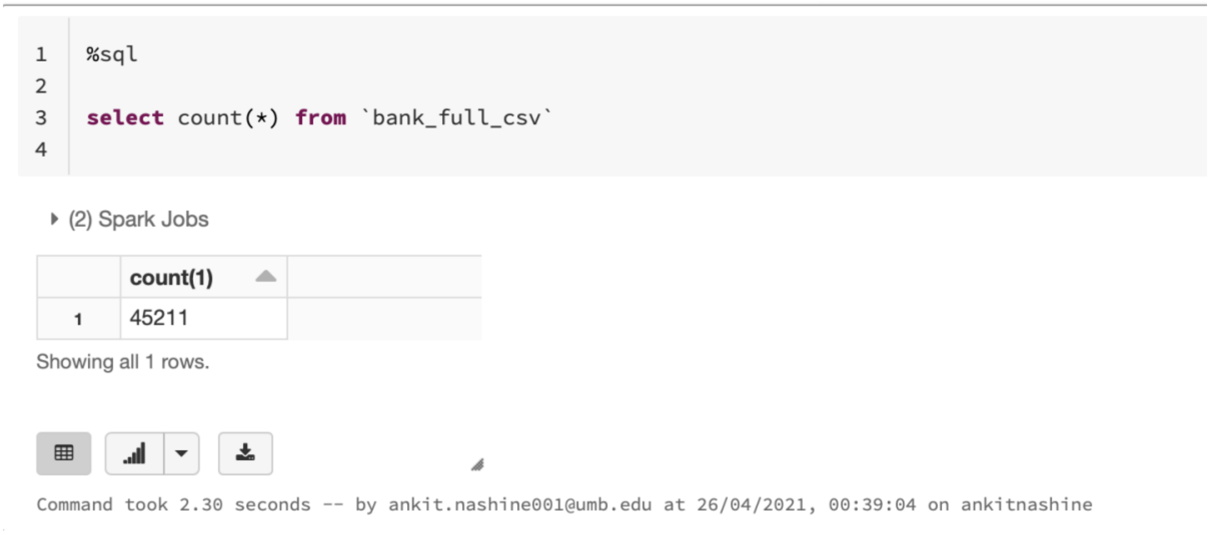
(1) Spark Jobs

1

```
Command took 7.36 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 00:24:17 on ankitnashine
```

HYPOTHESIS

1.1 Total Count of bank Clients



```
%sql

select count(*) from `bank_full_csv`

```

▸ (2) Spark Jobs

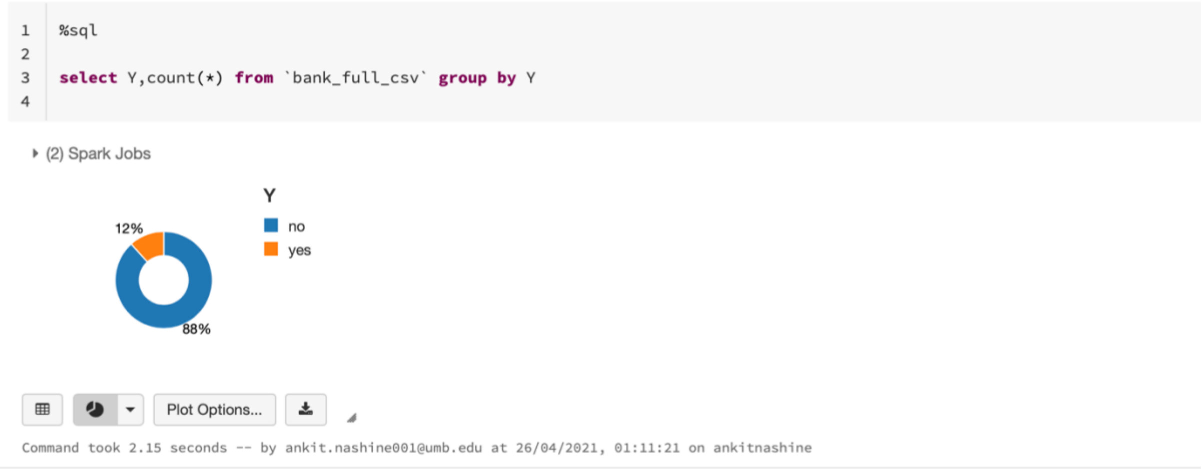
	count(1)
1	45211

Showing all 1 rows.

```
Command took 2.30 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 00:39:04 on ankitnashine
```

There are total 45211 clients. These clients will be used for further analysis.

1.2 Subscribers and non-subscribers (clients) in a term deposit.



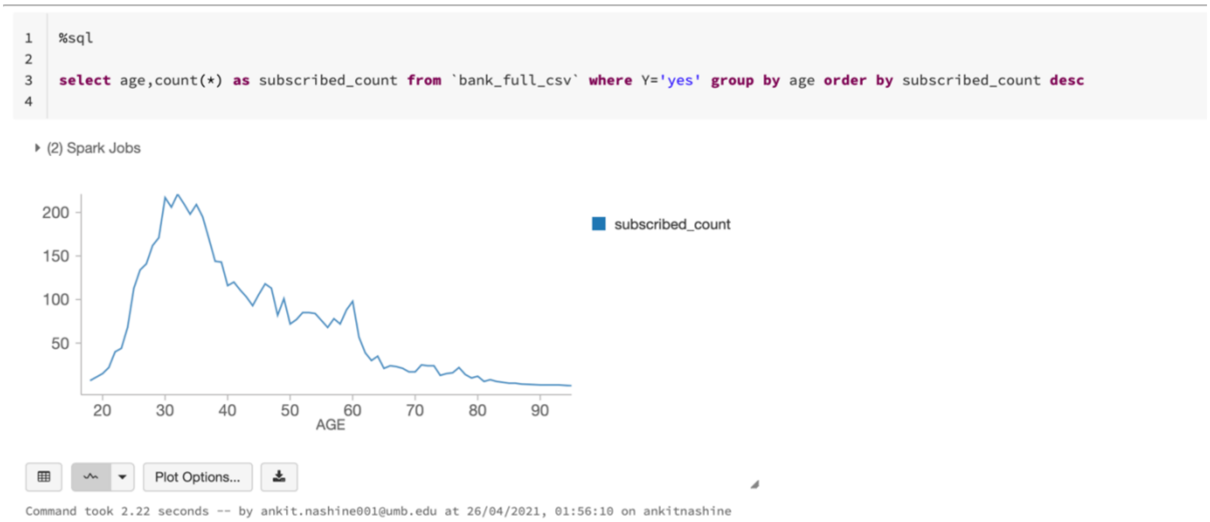
```
%sql

select Y,count(*) from `bank_full_csv` group by Y

```

▸ (2) Spark Jobs

```
Command took 2.15 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 01:11:21 on ankitnashine
```

Out of total clients, only 12% have subscribed for a term deposit. So, to understand subscribers, further analysis will be done on 12% population (i.e. subscribers).

2. Subscriber’s age group analysis.

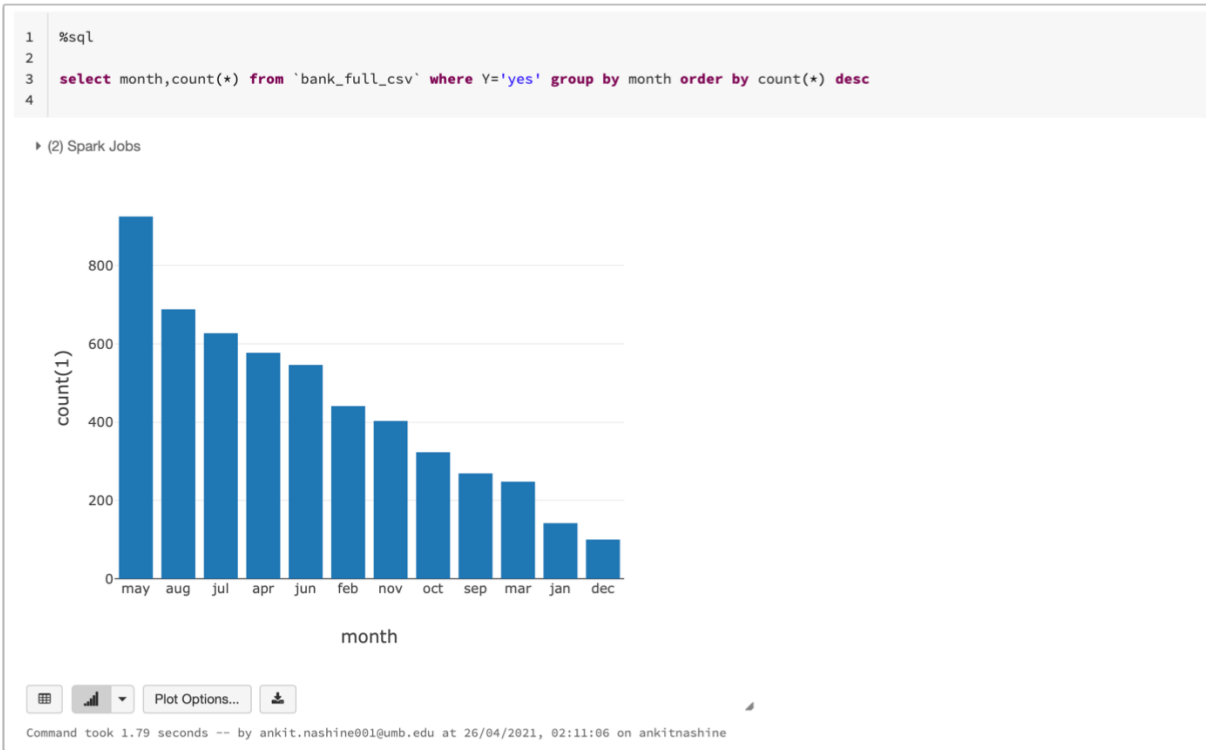


```
%sql

select age,count(*) as subscribed_count from `bank_full_csv` where Y='yes' group by age order by subscribed_count desc

```

```
Command took 2.22 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 01:56:10 on ankitnashine
```

It can be inferred from the chart above that most of the subscribers fall in the age range of 25 – 45.

3. Contact month of subscribers.

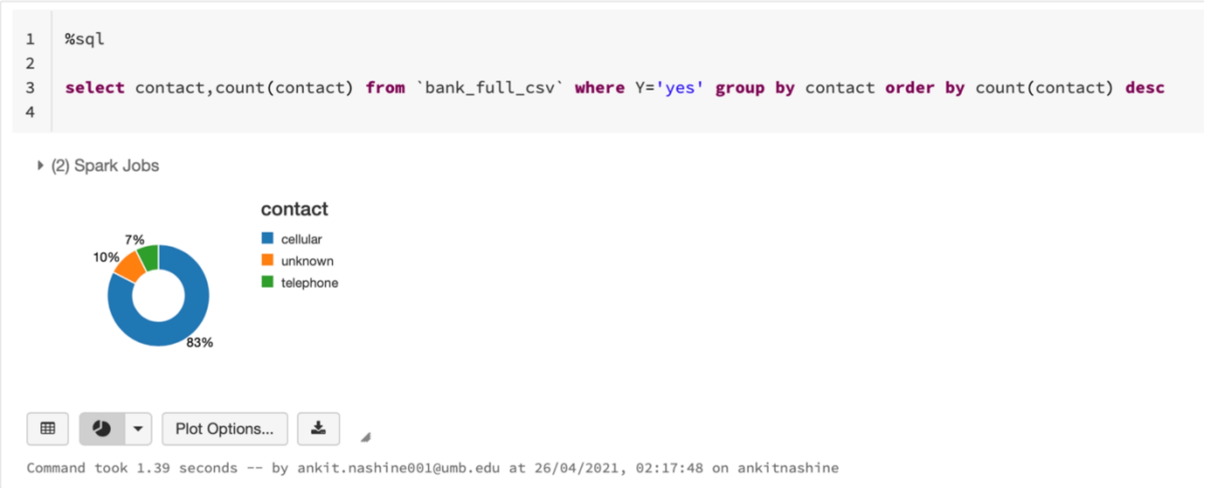


```
%sql

select month,count(*) from `bank_full_csv` where Y='yes' group by month order by count(*) desc

```

```
Command took 1.79 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 02:11:06 on ankitnashine
```

It can be inferred from the above chart that the best month to contact clients for making them subscribe is May. Should avoid contacting in Jan, Dec to be more feasible.

4. Best contact mode clients.

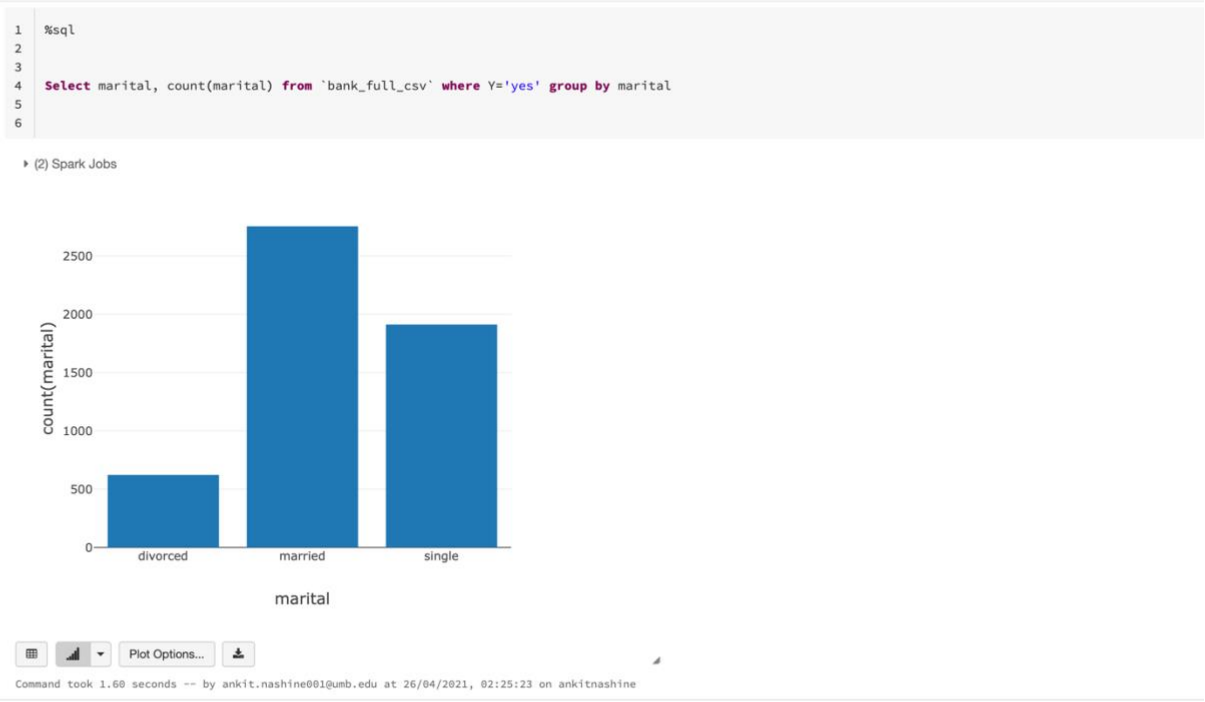


```
%sql

select contact,count(contact) from `bank_full_csv` where Y='yes' group by contact order by count(contact) desc

```

```
Command took 1.39 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 02:17:48 on ankitnashine
```

It can be inferred that clients with cellular mode of contact are more likely to subscribe.

5. Marital Status

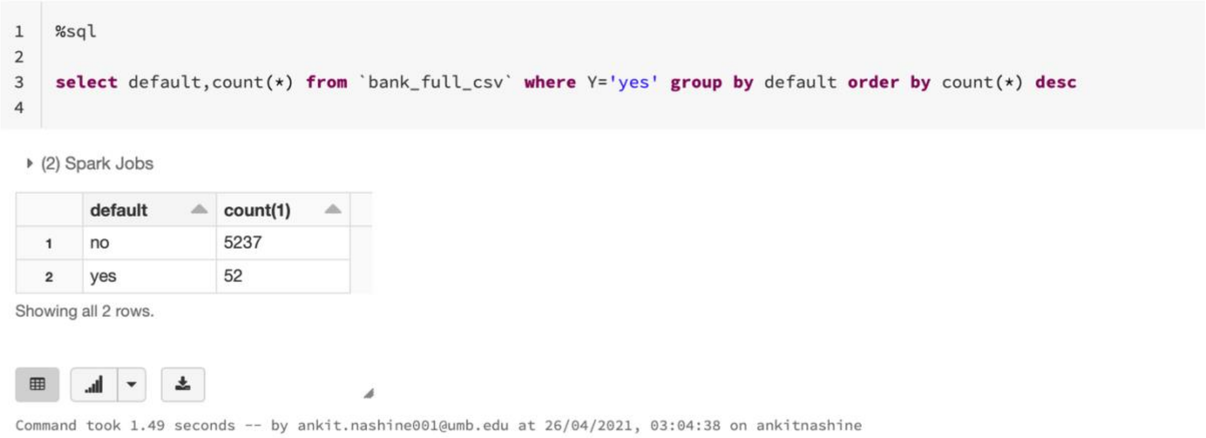


```
%sql


Select marital, count(marital) from `bank_full_csv` where Y='yes' group by marital


```

```
Command took 1.60 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 02:25:23 on ankitnashine
```

Married and single people are more likely to subscribe.

6. Credit History Analysis



```
%sql

select default,count(*) from `bank_full_csv` where Y='yes' group by default order by count(*) desc
```

▸ (2) Spark Jobs

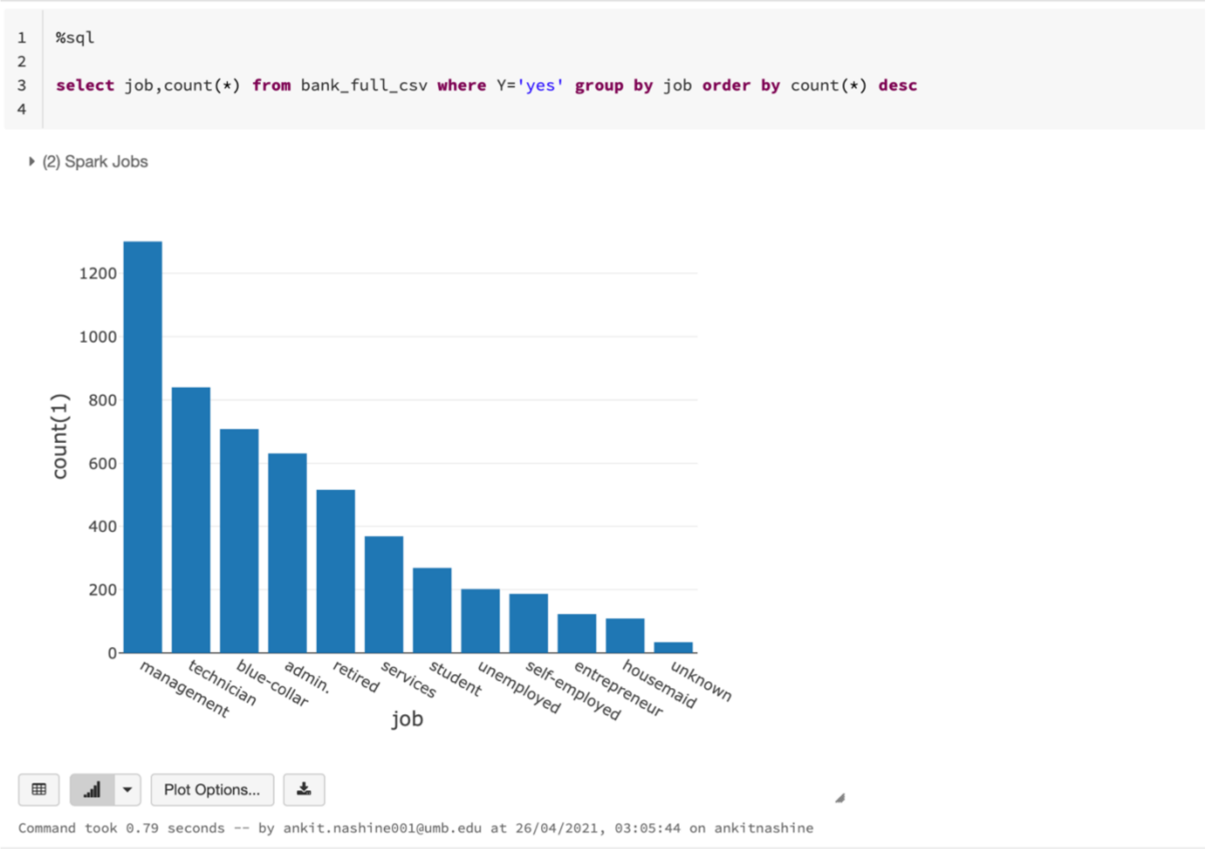
	default	count(1)
1	no	5237
2	yes	52

Showing all 2 rows.

```
Command took 1.49 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 03:04:38 on ankitnashine
```

Clients with credit history not default are potential subscribers.

7. Subscriber’s Job Analysis



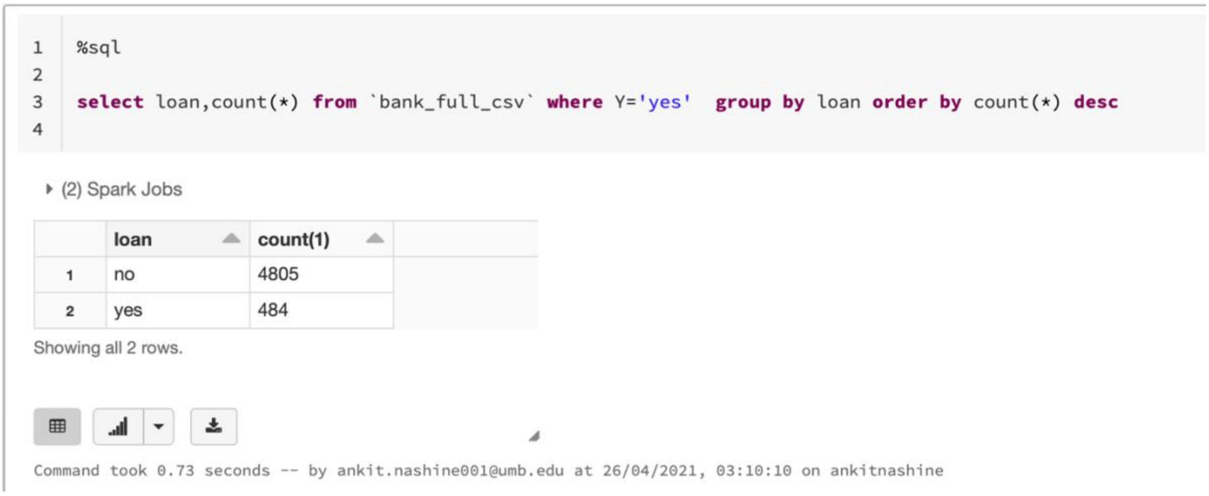
```
%sql

select job,count(*) from bank_full_csv where Y='yes' group by job order by count(*) desc
```

▸ (2) Spark Jobs

```
Command took 0.79 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 03:05:44 on ankitnashine
```

Management job people are most likely to subscribe.
Best is to contact people falls under first half of jobs and ignore second half such as housemaid, unemployed, etc.

8. Analyzing subscribers based on Loan Status



```
%sql

select loan,count(*) from `bank_full_csv` where Y='yes'  group by loan order by count(*) desc

```

(2) Spark Jobs

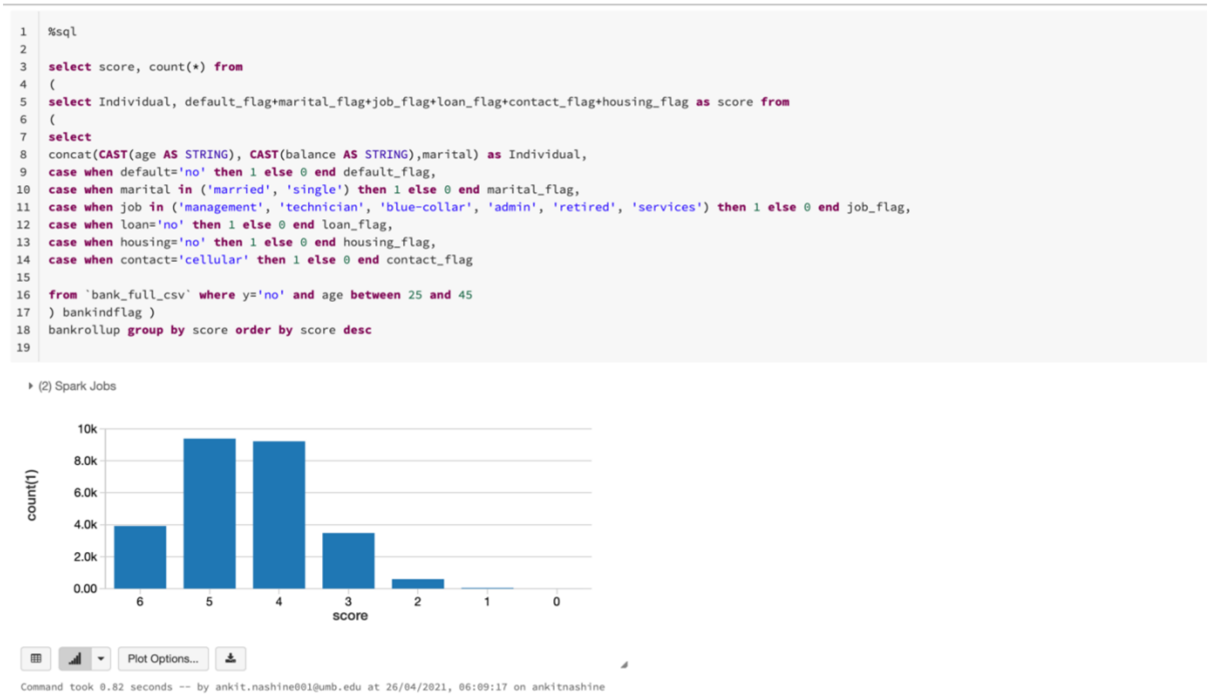
	loan	count(1)
1	no	4805
2	yes	484

Showing all 2 rows.

```
Command took 0.73 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 03:10:10 on ankitnashine
```

Based on the above results, clients with no loans are potential subscribers.

9. Final model to predict Potential Subscribers



```
%sql

select score, count(*) from
(
select Individual, default_flag+marital_flag+job_flag+loan_flag+contact_flag+housing_flag as score from
(
select
concat(CAST(age AS STRING), CAST(balance AS STRING),marital) as Individual,
case when default='no' then 1 else 0 end default_flag,
case when marital in ('married', 'single') then 1 else 0 end marital_flag,
case when job in ('management', 'technician', 'blue-collar', 'admin', 'retired', 'services') then 1 else 0 end job_flag,
case when loan='no' then 1 else 0 end loan_flag,
case when housing='no' then 1 else 0 end housing_flag,
case when contact='cellular' then 1 else 0 end contact_flag

from `bank_full_csv` where y='no' and age between 25 and 45
) bankindflag )
bankrollup group by score order by score desc

```

(2) Spark Jobs

```
Command took 0.82 seconds -- by ankit.nashine001@umb.edu at 26/04/2021, 06:09:17 on ankitnashine
```

Based on previous analysis, predicting clients (non-subscribers) which are most likely to subscribe and contacting then to offer term deposit.
Should contact clients with score 6 and 5.