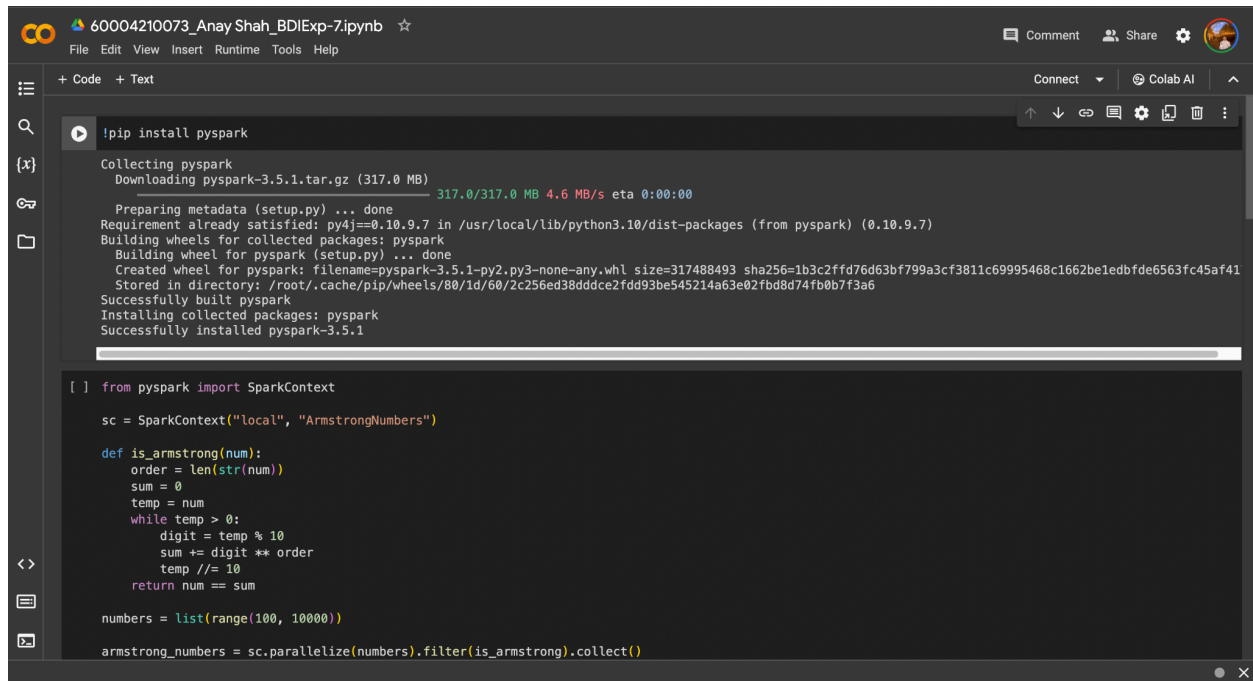


Output:

Colab Link:

https://colab.research.google.com/drive/1SQidp8U4OVT9kgNUqX24_QQH5SHaRdiJ?usp=sharing



```
!pip install pyspark

Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
    317.0/317.0 MB 4.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488493 sha256=1b3c2ffd76d63bf799a3cf3811c69995468c1662be1edbfde6563fc45af41
  Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38ddce2fdd93be545214a63e02fbd8d74fb0b7f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1

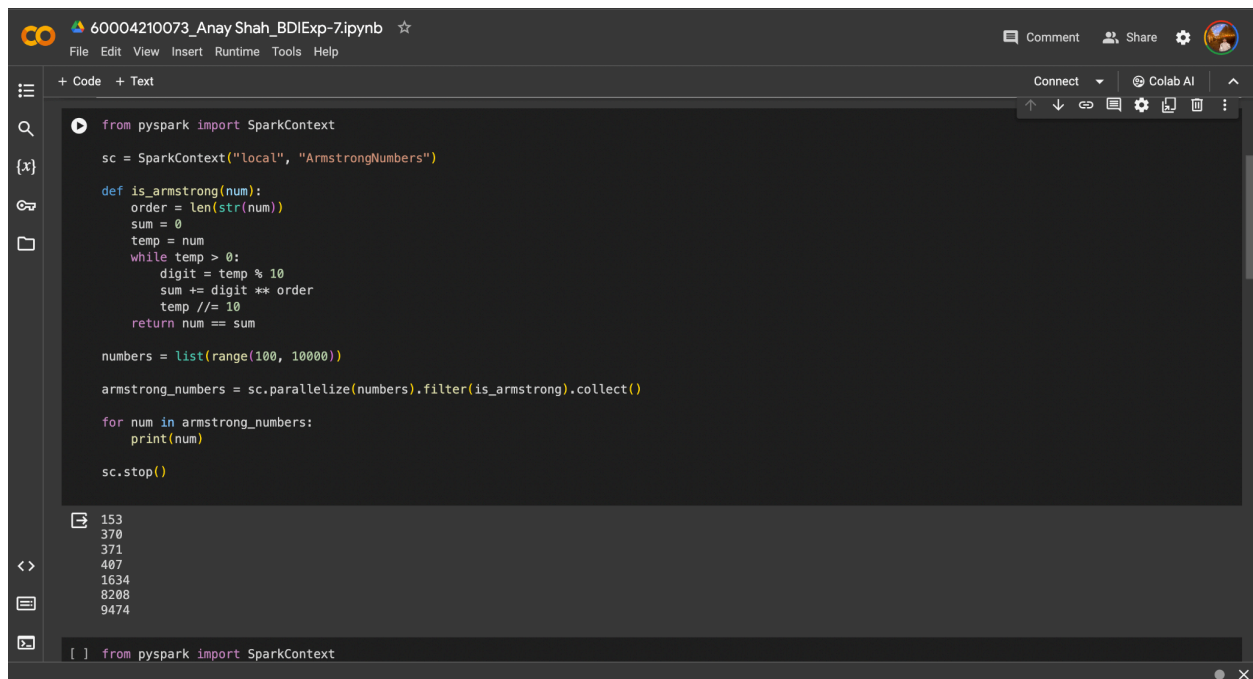
[ ] from pyspark import SparkContext

sc = SparkContext("local", "ArmstrongNumbers")

def is_armstrong(num):
    order = len(str(num))
    sum = 0
    temp = num
    while temp > 0:
        digit = temp % 10
        sum += digit ** order
        temp //= 10
    return num == sum

numbers = list(range(100, 10000))

armstrong_numbers = sc.parallelize(numbers).filter(is_armstrong).collect()
```



```
from pyspark import SparkContext

sc = SparkContext("local", "ArmstrongNumbers")

def is_armstrong(num):
    order = len(str(num))
    sum = 0
    temp = num
    while temp > 0:
        digit = temp % 10
        sum += digit ** order
        temp //= 10
    return num == sum

numbers = list(range(100, 10000))

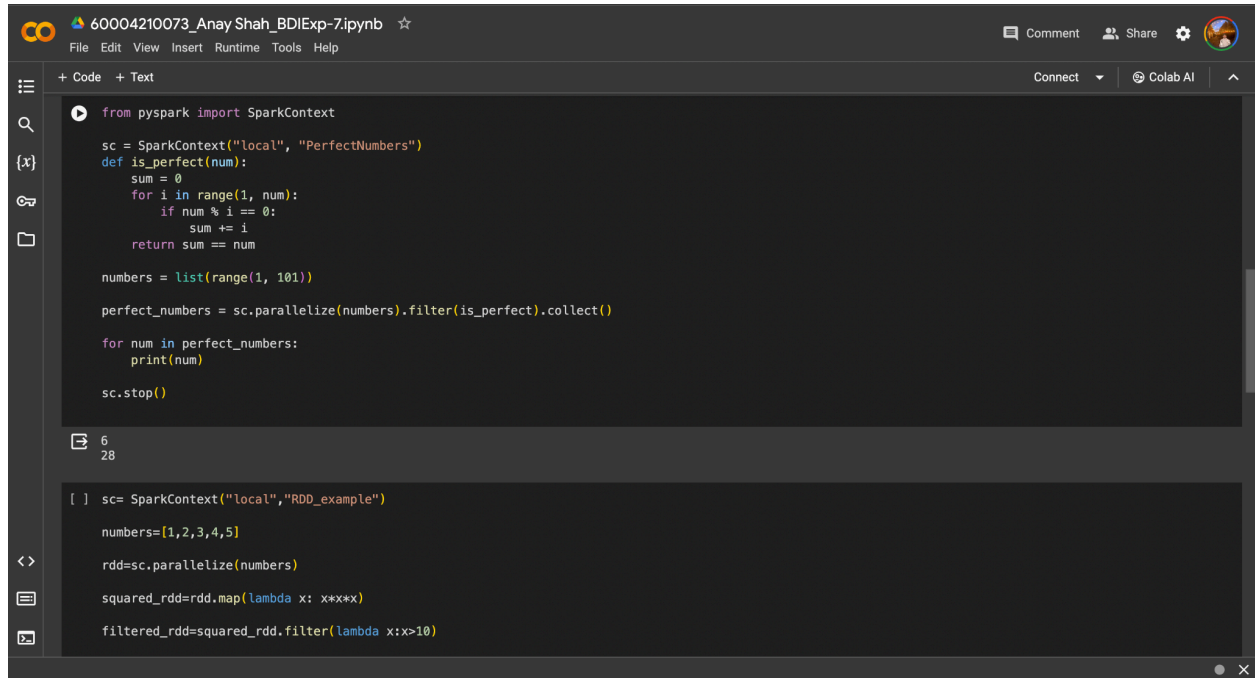
armstrong_numbers = sc.parallelize(numbers).filter(is_armstrong).collect()

for num in armstrong_numbers:
    print(num)

sc.stop()

153
370
371
407
1634
8208
9474

[ ] from pyspark import SparkContext
```



```
from pyspark import SparkContext

sc = SparkContext("local", "PerfectNumbers")
def is_perfect(num):
    sum = 0
    for i in range(1, num):
        if num % i == 0:
            sum += i
    return sum == num

numbers = list(range(1, 101))

perfect_numbers = sc.parallelize(numbers).filter(is_perfect).collect()

for num in perfect_numbers:
    print(num)

sc.stop()
```

6
28

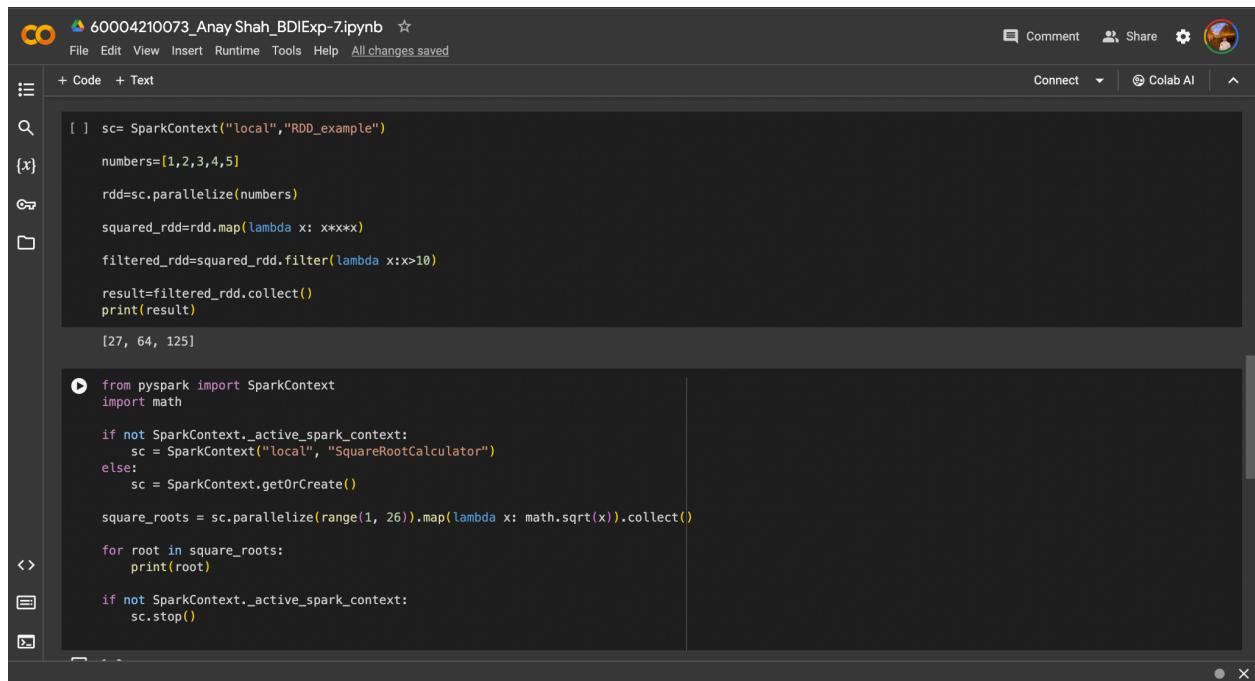
```
[ ] sc= SparkContext("local","RDD_example")

numbers=[1,2,3,4,5]

rdd=sc.parallelize(numbers)

squared_rdd=rdd.map(lambda x: x*x*x)

filtered_rdd=squared_rdd.filter(lambda x:x>10)
```



```
[ ] sc= SparkContext("local","RDD_example")

numbers=[1,2,3,4,5]

rdd=sc.parallelize(numbers)

squared_rdd=rdd.map(lambda x: x*x*x)

filtered_rdd=squared_rdd.filter(lambda x:x>10)

result=filtered_rdd.collect()
print(result)

[27, 64, 125]
```

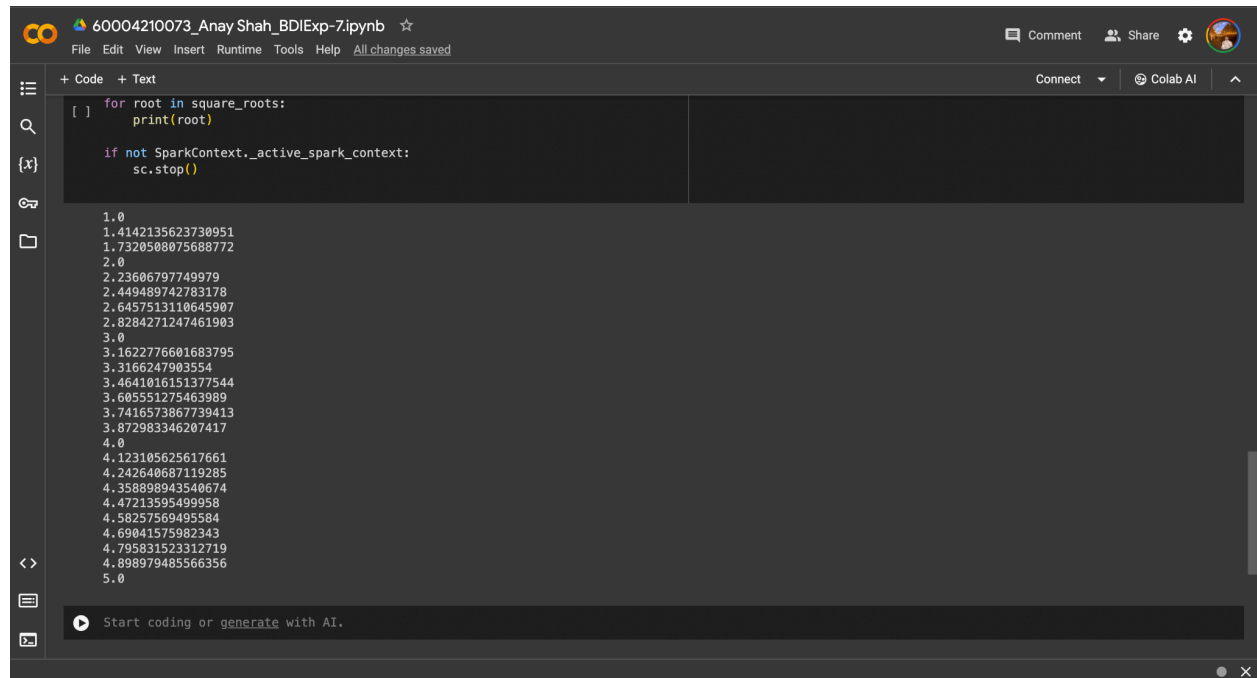
```
from pyspark import SparkContext
import math

if not SparkContext._active_spark_context:
    sc = SparkContext("local", "SquareRootCalculator")
else:
    sc = SparkContext.getOrCreate()

square_roots = sc.parallelize(range(1, 26)).map(lambda x: math.sqrt(x)).collect()

for root in square_roots:
    print(root)

if not SparkContext._active_spark_context:
    sc.stop()
```



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes the Jupyter logo, the notebook name "60004210073_Anay Shah_BDIExp-7.ipynb", and a star icon. Below the top bar is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a link for "All changes saved". On the right side of the top bar are buttons for "Comment", "Share", "Settings", and a user profile icon. The left sidebar contains icons for a list view, search, code editor, output view, and file explorer. The main area is divided into two sections: a code editor and an output area. The code editor shows a Python code cell with the following code:

```
[ ] for root in square_roots:
    print(root)

if not SparkContext._active_spark_context:
    sc.stop()
```

The output area displays the results of the code execution, which are the square roots of the numbers in the "square_roots" list, printed one per line:

```
1.0
1.4142135623730951
1.7320508075688772
2.0
2.23606797749979
2.449489742783178
2.6457513110645907
2.8284271247461903
3.0
3.1622776601683795
3.3166247903554
3.4641016151377544
3.605551275463989
3.7416573867739413
3.872983346207417
4.0
4.123105625617661
4.242640687119285
4.358898943540674
4.47213595499958
4.58257569495584
4.69041575982343
4.795831523312719
4.898979485566356
5.0
```

At the bottom of the output area, there is a button that says "Start coding or generate with AI."