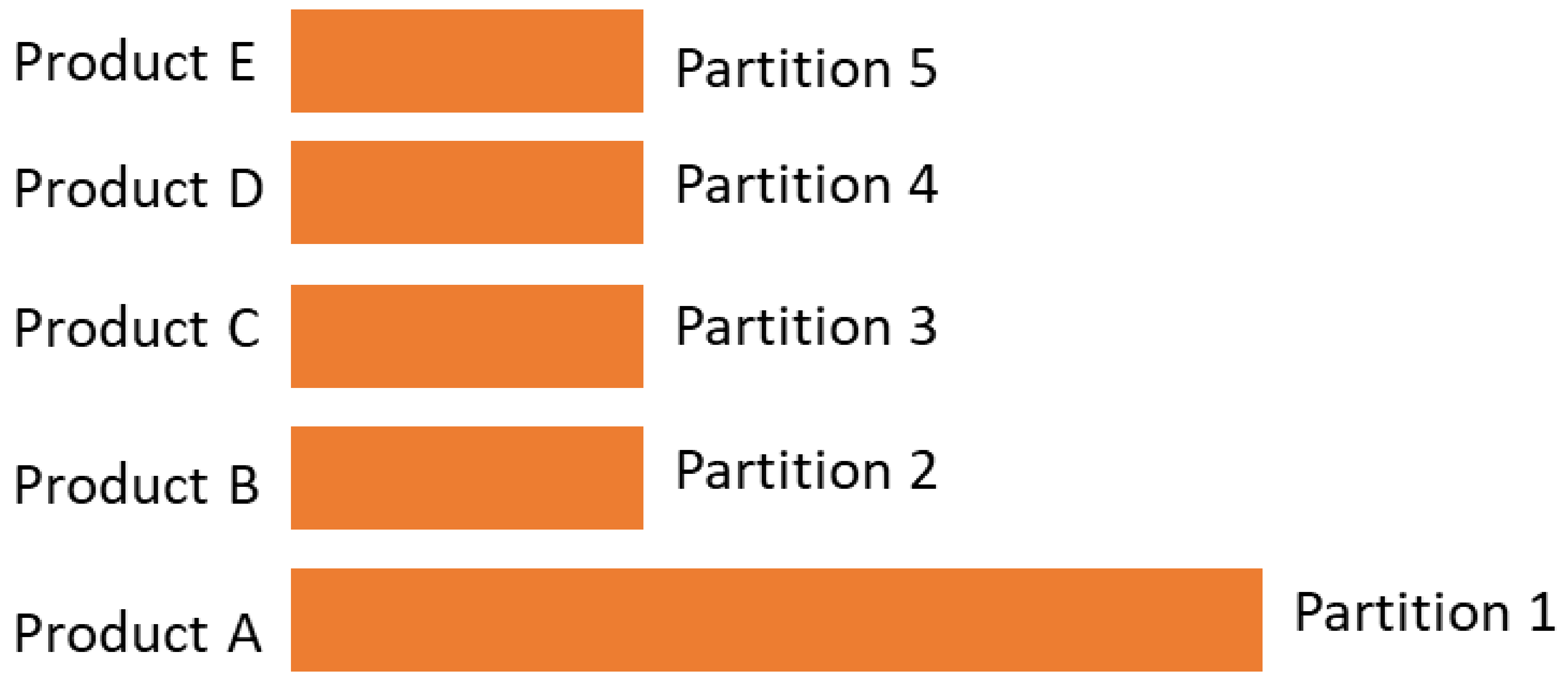
Data Skew in Apache Spark

Suppose we have an ecommerce sales dataset which is partitioned on product



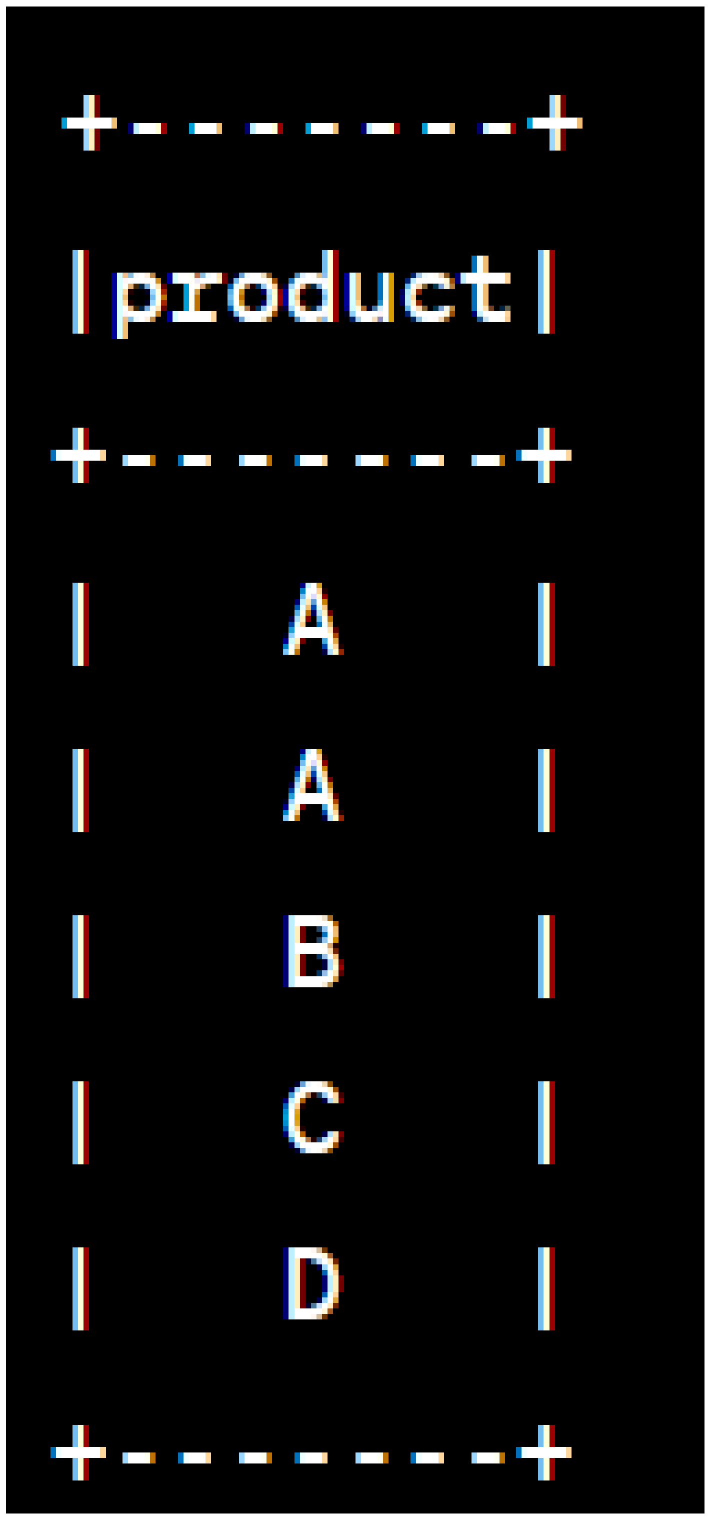
Product Sales dataframe after partitioned on Product

We can see since the data volume for Product A is way higher than others, partition size for it is way larger

Data Skew!!

@shuvajithazra

Mitigation Plan 1: Salting



```
+-------+
|product|
+-------+
|   A   |
|   A   |
|   B   |
|   C   |
|   D   |
+-------+
```

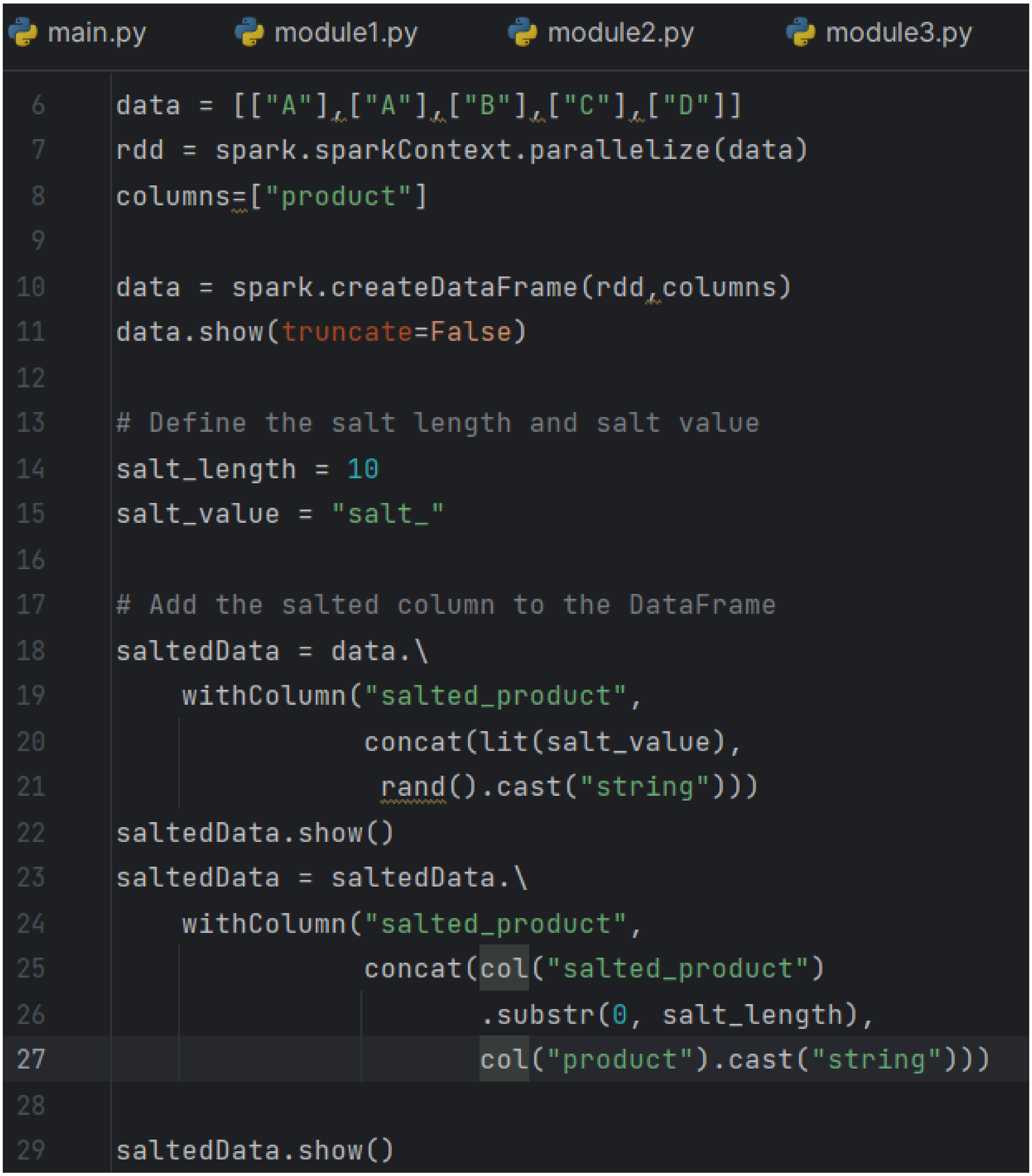
In the input dataset, we can observe data skew, as the key "A" appears more frequently compared to other keys. This uneven distribution of data can lead to data skew and adversely affect the performance of our data processing tasks.

@shuvajithazra

@shuvajithazra

Mitigation Plan 1: Salting

To mitigate data skew using the salting technique, we can apply the provided code snippet in PySpark.

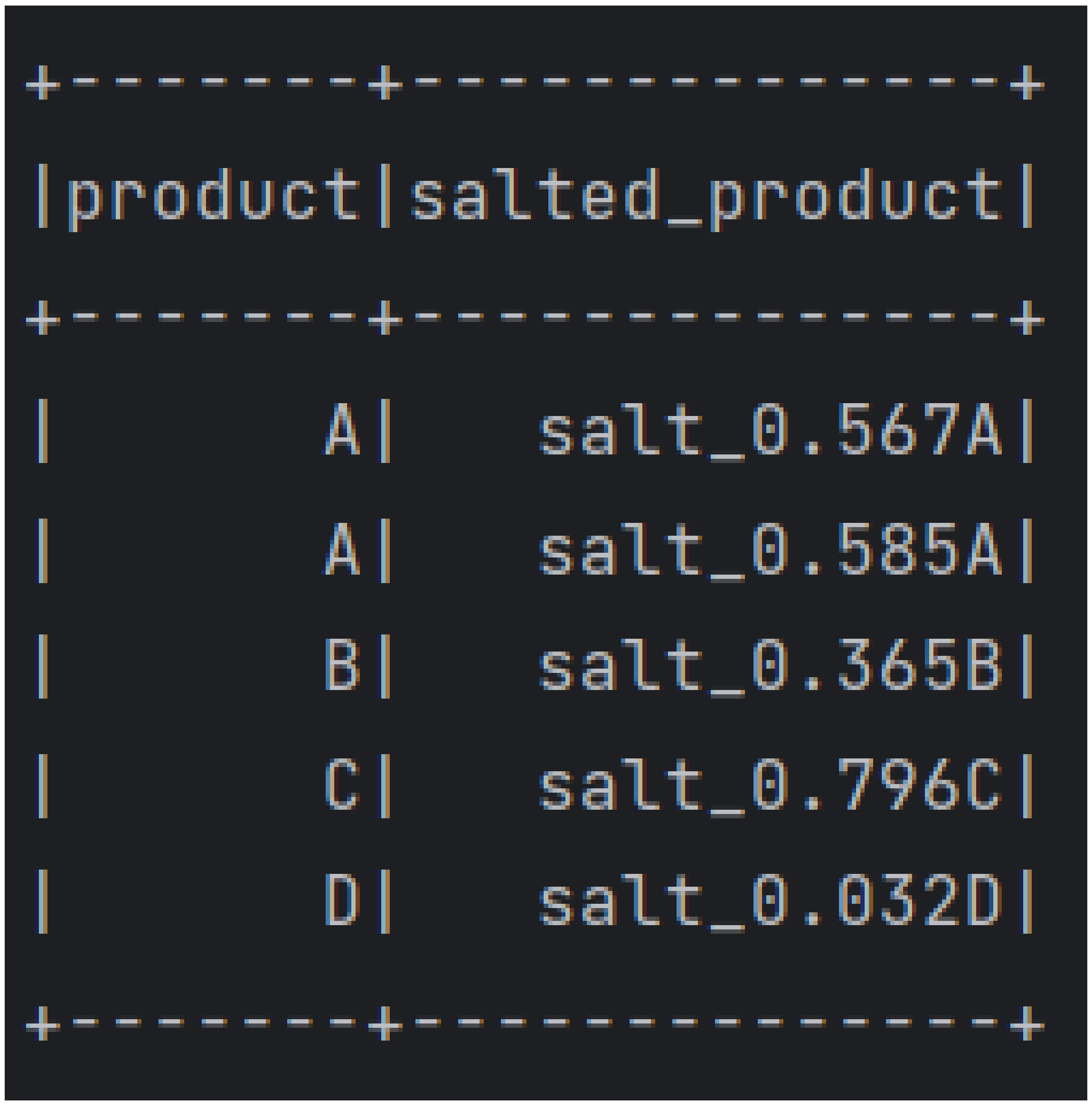


main.py module1.py module2.py module3.py

```python
data = [["A"],["A"],["B"],["C"],["D"]]
rdd = spark.sparkContext.parallelize(data)
columns=["product"]

data = spark.createDataFrame(rdd,columns)
data.show(truncate=False)

# Define the salt length and salt value
salt_length = 10
salt_value = "salt_"

# Add the salted column to the DataFrame
saltedData = data.\
    withColumn("salted_product",
               concat(lit(salt_value),
                rand().cast("string")))
saltedData.show()
saltedData = saltedData.\
    withColumn("salted_product",
               concat(col("salted_product")
                      .substr(0, salt_length),
                      col("product").cast("string")))

saltedData.show()
```

Mitigation Plan 1: Salting

After applying the salting technique, the output DataFrame saltedData would look like this:



```
+-------+---------------+
|product|salted_product|
+-------+---------------+
|      A|   salt_0.567A|
|      A|   salt_0.585A|
|      B|   salt_0.365B|
|      C|   salt_0.796C|
|      D|   salt_0.032D|
+-------+---------------+
```

The salting technique adds a random value (salt) to each product key, resulting in new salted keys. These salted keys introduce randomness and ensure that the skewed keys ("A" in this case) are distributed more evenly across partitions.

Problem: Additional Computational Overhead for Salting

@shuvajithazra

Mitigation Plan 2: AQL

Adaptive Query Language (AQL) is a powerful feature introduced in Apache Spark to combat data skew. It leverages dynamic adaptation techniques during query execution to identify and mitigate data skew effectively.

How?

1 Dynamic Query Optimization: AQL analyzes the query execution plan and detects potential data skew. It intelligently adapts the plan to redistribute the workload more evenly across the available resources, minimizing the impact of skew on performance.

2 Range Partitioning: AQL can leverage range partitioning, dividing skewed data into smaller ranges and distributing them across multiple partitions. By doing so, it ensures that the processing of skewed keys is distributed among different worker nodes, enabling better parallelism.

@shuvajithazra

Mitigation Plan 2: AQL

3 **Sampling**: AQL uses sampling techniques to estimate the distribution of data across partitions. By analyzing a sample of the data, it can make informed decisions about how to best redistribute the skewed data for optimal load balancing.

4 **Dynamic Repartitioning**: AQL dynamically determines when to repartition data during query execution. It intelligently redistributes data to achieve a more balanced distribution, optimizing resource utilization and reducing the impact of data skew.

Mitigation Plan 3: Skew Hint

- This can be applied only if you are using databricks
- Employ Databricks specific Skew Hint

Link for Databricks Documentation

@shuvajithazra