```
#check that java is installed
!java -version
```

```
openjdk version "11.0.28" 2025-07-15
OpenJDK Runtime Environment (build 11.0.28+6-post-Ubuntu-1ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 11.0.28+6-post-Ubuntu-1ubuntu122.04.1, mixed mode, sharing)
```

```
#install pyspark
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
```

```
# ============================
# STEP 1: Install & Import
# ============================
!pip install pyspark --quiet

from pyspark.sql import SparkSession
from pyspark.sql.functions import lit, rand, floor, concat_ws, explode, array, col, count
import time

# Create Spark Session
spark = SparkSession.builder \
    .appName("SaltingExample") \
    .config("spark.sql.shuffle.partitions", 8) \
    .getOrCreate()
```

```
# ============================
# STEP 2: Create Skewed Sales Data
# ============================
# Fact table: sales (skew on customer_id=1)
num_rows = 5_000_000
skew_ratio = 0.8  # 80% rows belong to customer_id=1

# Create skewed data
data_skewed = []
for i in range(num_rows):
    if i < num_rows * skew_ratio:
        data_skewed.append((1, i % 100, float(i % 500)))  # customer_id=1
    else:
        cust_id = i % 1000 + 2  # other customers
        data_skewed.append((cust_id, i % 100, float(i % 500)))

sales_df = spark.createDataFrame(data_skewed, ["customer_id", "product_id", "amount"])

# Dimension table: customers
customers = [(1, "VIP Customer")] + [(i, f"Customer_{i}") for i in range(2, 1002)]
customers_df = spark.createDataFrame(customers, ["customer_id", "customer_name"])
```

```
# ============================
# STEP 3: Show Skew Distribution
# ============================
print("📊 Customer Distribution (Skewed):")
sales_df.groupBy("customer_id").agg(count("*").alias("cnt")) \
    .orderBy(col("cnt").desc()) \
    .show(5)
```

```
📊 Customer Distribution (Skewed):
+-----------+-------+
|customer_id|    cnt|
+-----------+-------+
|          1|4000000|
|          2|   1000|
|         12|   1000|
|         26|   1000|
|         28|   1000|
+-----------+-------+
only showing top 5 rows
```

```
# ============================
# STEP 4: Skewed Join (No Salting)
# ============================
start = time.time()
joined_skewed = sales_df.join(customers_df, "customer_id")
joined_skewed.count()  # Force execution
end = time.time()

print(f"⏳ Join Time (Skewed, No Salting): {round(end - start, 2)} sec")
```

```
⏳ Join Time (Skewed, No Salting): 10.88 sec
```

```
# Partition size distribution before salting
partition_sizes_skewed = joined_skewed.rdd.mapPartitions(lambda it: [sum(1 for _ in it)]).collect()
print("\n Partition Sizes (Skewed):", partition_sizes_skewed)
```

```
 Partition Sizes (Skewed): [2499584, 2500416]
```

```
# ============================
# STEP 5: Salting the Join
# ============================
salt_size = 10  # Break heavy key into 10 parts

# Add salt to customers table
customers_salted = customers_df.withColumn("salt", floor(rand() * salt_size)) \
    .withColumn("join_key", concat_ws("_", col("customer_id"), col("salt")))

# Add all salt values for sales table
sales_salted = sales_df.withColumn("salt", explode(array([lit(i) for i in range(salt_size)]))) \
    .withColumn("join_key", concat_ws("_", col("customer_id"), col("salt")))

# Join on salted key
start = time.time()
joined_salted = sales_salted.join(customers_salted, "join_key") \
    .drop("join_key", "salt")
joined_salted.count()  # Force execution
end = time.time()

print(f"⚡ Join Time (With Salting): {round(end - start, 2)} sec")
```

```
⚡ Join Time (With Salting): 47.13 sec
```

Double-click (or enter) to edit

```
# ============================
# STEP 6: Cleanup
# ============================
spark.stop()
```

```
Start coding or generate with AI.
```