```
from pyspark.sql import SparkSession
from pyspark.sql.functions import when, col, count
import time

# Measure the full execution time
start_time = time.time()

# Create a SparkSession
spark = SparkSession.builder \
    .appName("Query 1") \
    .getOrCreate()

# Read both CSV files into DataFrames
crime_data_2010_2019 = "s3://initial-notebook-data-bucket-
dblab-905418150721/CrimeData/
Crime_Data_from_2010_to_2019_20241101.csv"
crime_data_2020_present = "s3://initial-notebook-data-bucket-
dblab-905418150721/CrimeData/
Crime_Data_from_2020_to_Present_20241101.csv"

# Read both CSV files into DataFrames
crime_df_2010_2019 = spark.read.csv(crime_data_2010_2019,
header=True, inferSchema=True)
crime_df_2020_present =
spark.read.csv(crime_data_2020_present, header=True,
inferSchema=True)

# Combine the two DataFrames and remove duplicates
crime_data =
crime_df_2010_2019.union(crime_df_2020_present).dropDuplicates
()

# Filter records for "AGGRAVATED ASSAULT" in the column "Crm
Cd Desc"
filtered_data = crime_data.filter(col("Crm Cd
Desc").contains("AGGRAVATED ASSAULT"))

# Add a new column for age groups based on "Vict Age"
filtered_data = filtered_data.withColumn(
    "Age Group",
    when(col("Vict Age") < 18, "Children (<18)") \
    .when((col("Vict Age") >= 18) & (col("Vict Age") <= 24),
"Young adults (18-24)") \
    .when((col("Vict Age") >= 25) & (col("Vict Age") <= 64),
"Adults (25-64)") \
    .when(col("Vict Age") > 64, "Elderly (>64)") \
    .otherwise("Not Known")
)

# Group by "Age Group" and count occurrences, then sort by
```

```
count in descending order
age_group_counts = filtered_data.groupBy("Age 
Group").agg(count("*").alias("Count")).orderBy(col("Count").de
sc())

# Trigger Spark execution and display results
age_group_counts.show()

# Measure total execution time
end_time = time.time()
print(f"DataFrame API took: {end_time - start_time:.2f} 
seconds")
```

```
+--------------------+------+
|           Age Group| Count|
+--------------------+------+
|      Adults (25-64)|121093|
|Young adults (18-24)| 33605|
|      Children (<18)| 15928|
|       Elderly (>64)|  5985|
+--------------------+------+

DataFrame API took: 27.76 seconds
```

```
from pyspark.sql import SparkSession
import time

# Measure the full execution time
start_time = time.time()

# Create a SparkSession
spark = SparkSession.builder \
    .appName("Query 1 - RDD") \
    .getOrCreate()

# S3 paths for the CSV files
crime_data_2010_2019_path = "s3://initial-notebook-data-
bucket-dblab-905418150721/CrimeData/
Crime_Data_from_2010_to_2019_20241101.csv"
crime_data_2020_present_path = "s3://initial-notebook-data-
bucket-dblab-905418150721/CrimeData/
Crime_Data_from_2020_to_Present_20241101.csv"

# Read both CSV files into RDDs
crime_data_2010_2019_rdd = 
spark.read.csv(crime_data_2010_2019_path, header=True, 
inferSchema=True).rdd
crime_data_2020_present_rdd = 
spark.read.csv(crime_data_2020_present_path, header=True, 
inferSchema=True).rdd
```

```python
# Combine the two RDDs and remove duplicates
crime_data_rdd = crime_data_2010_2019_rdd.union(crime_data_2020_present_rdd).distinct()

# Filter records for "AGGRAVATED ASSAULT" in the column "Crm Cd Desc"
filtered_rdd = crime_data_rdd.filter(lambda row: "AGGRAVATED ASSAULT" in str(row["Crm Cd Desc"]))

# Define age group categorization
def categorize_age(row):
    age = row["Vict Age"]
    if age is None:
        return "Not Known"
    elif age < 18:
        return "Children (<18)"
    elif 18 <= age <= 24:
        return "Young adults (18-24)"
    elif 25 <= age <= 64:
        return "Adults (25-64)"
    elif age > 64:
        return "Elderly (>64)"
    else:
        return "Not Known"

# Map rows to age groups and count occurrences
age_groups_rdd = filtered_rdd.map(lambda row: (categorize_age(row), 1))
age_group_counts_rdd = age_groups_rdd.reduceByKey(lambda a, b: a + b).sortBy(lambda x: x[1], ascending=False)

# Perform the action and measure time
for age_group, count in age_group_counts_rdd.collect():
    print(f"{age_group}: {count}")

# Calculate total execution time
end_time = time.time()
print(f"RDD API took: {end_time - start_time:.2f} seconds")
```

```
Adults (25-64): 121093
Young adults (18-24): 33605
Children (<18): 15928
Elderly (>64): 5985
RDD API took: 14.30 seconds
```