

Προχωρημένα Θέματα Βάσεων Δεδομένων

Μαρία Κοιλαλού | Μυρτώ Ορφανάκου
AM:03119211 | AM:

15 Ιανουαρίου 2024

Spark Session

```
1 from pyspark.sql import SparkSession
2
3 def create_spark_session(num_executors):
4
5     spark = SparkSession.builder \
6         .appName("ADV DATABASES") \
7         .config("spark.executor.instances", str(num_executors)) \
8         .config("spark.executor.cores", "1") \
9         .config("spark.executor.memory", "1g") \
10        .config("spark.driver.memory", "4g") \
11        .config("spark.cleaner.periodicGC.interval", "1min") \
12        .config("spark.memory.offHeap.enabled", "true") \
13        .config("spark.memory.offHeap.size", "1g") \
14        .config("spark.default.parallelism", "4") \
15        .config("spark.serializer", "org.apache.spark.serializer.KryoSerializer") \
16        .config("spark.dynamicAllocation.enabled", "true") \
17        .config("spark.dynamicAllocation.minExecutors", "2") \
18        .config("spark.dynamicAllocation.maxExecutors", "4") \
19        .getOrCreate()
20
21 return spark
```

Your File Caption

```
1 spark = create_spark_session(4)
2 print("spark session created")
```

Your File Caption

Create DataFrame

```
1 schema1 = "`DR_NO` STRING, \
2             `Date Rptd` STRING, \
3             `DATE OCC` STRING, \
4             `TIME OCC` INTEGER, \
5             `AREA` INTEGER, \
6             `AREA NAME` STRING, \
7             `Rpt Dist No` INTEGER, \
8             `Part 1-2` INTEGER, \
9             `Crm Cd` INTEGER, \
10            `Crm Cd Desc` STRING, \
11            `Mocodes` STRING, \
12            `Vict Age` INTEGER, \
```

```

13 `Vict Sex` STRING, \
14 `Vict Descent` STRING, \
15 `Premis Cd` INTEGER, \
16 `Premis Desc` STRING, \
17 `Weapon Used Cd` INTEGER, \
18 `Weapon Desc` STRING, \
19 `Status` STRING, \
20 `Status Desc` STRING, \
21 `Crm Cd 1` INTEGER, \
22 `Crm Cd 2` INTEGER, \
23 `Crm Cd 3` INTEGER, \
24 `Crm Cd 4` INTEGER, \
25 `LOCATION` STRING, \
26 `Cross Street` STRING, \
27 `LAT` DOUBLE, \
28 `LON` DOUBLE"

```

```

30 data1 = spark.read.csv("/user/ubuntu/ta/advanced-db/data/crime_data_2010.csv", header=True, schema=schema1)
31 data2 = spark.read.csv("/user/ubuntu/ta/advanced-db/data/crime_data_2020.csv", header=True, schema=schema1)
32
33 df = data1.union(data2).distinct()
34
35 df = df.withColumn("Date Rptd", to_date(col("Date Rptd"), "MM/dd/yyyy hh:mm:ss a")) \
36 .withColumn("DATE OCC", to_date(col("DATE OCC"), "MM/dd/yyyy hh:mm:ss a"))
37
38 df.count()
39 print(f"Total number of rows: {df.count()}")
40
41 df.printSchema()
42

```

Your File Caption

Total number of rows: 2913595

root

```

|-- DR_NO: string (nullable = true)
|-- Date Rptd: date (nullable = true)
|-- DATE OCC: date (nullable = true)
|-- TIME OCC: integer (nullable = true)
|-- AREA: integer (nullable = true)
|-- AREA NAME: string (nullable = true)
|-- Rpt Dist No: integer (nullable = true)
|-- Part 1-2: integer (nullable = true)
|-- Crm Cd: integer (nullable = true)
|-- Crm Cd Desc: string (nullable = true)
|-- Mocodes: string (nullable = true)
|-- Vict Age: integer (nullable = true)
|-- Vict Sex: string (nullable = true)
|-- Vict Descent: string (nullable = true)
|-- Premis Cd: integer (nullable = true)
|-- Premis Desc: string (nullable = true)
|-- Weapon Used Cd: integer (nullable = true)
|-- Weapon Desc: string (nullable = true)
|-- Status: string (nullable = true)
|-- Status Desc: string (nullable = true)
|-- Crm Cd 1: integer (nullable = true)
|-- Crm Cd 2: integer (nullable = true)
|-- Crm Cd 3: integer (nullable = true)
|-- Crm Cd 4: integer (nullable = true)
|-- LOCATION: string (nullable = true)
|-- Cross Street: string (nullable = true)
|-- LAT: double (nullable = true)
|-- LON: double (nullable = true)

```

Query 1

Dataframe API

```
1 def query1_df(df):
2     crime_date = df.withColumn("Year", year("DATE OCC")).withColumn("Month", month("DATE OCC"))
3
4     count = crime_date.groupBy("Year", "Month").count()
5
6     window_spec = Window.partitionBy("Year").orderBy(desc("count"))
7     top_months = count.withColumn("rank", dense_rank().over(window_spec)).filter(col("rank") <= 3)
8
9     top_months = top_months.orderBy("Year", "rank")
10
11     return top_months
```

Your File Caption

SQL API

```
1 def query1_sql(df):
2     crime_date = df.withColumn("Year", year("DATE OCC")).withColumn("Month", month("DATE OCC"))
3
4     # Δημιουργία προσωρινής προβολής
5     crime_date.createOrReplaceTempView("crimes")
6
7     # SQL ερώτημα για την εύρεση των τριών μηνών με τον υψηλότερο αριθμό εγκλημάτων ανά έτος
8     query1 = """
9     SELECT Year, Month, count, rank
10    FROM (
11        SELECT Year, Month, count(*) AS count,
12               DENSE_RANK() OVER (PARTITION BY Year ORDER BY count(*) DESC) AS rank
13        FROM crimes
14        GROUP BY Year, Month
15    )
16    WHERE rank <= 3
17    ORDER BY Year, rank
18    """
19
20     top_months = crime_date.sparkSession.sql(query1)
21
22     return top_months
```

Your File Caption

| Year | Month | count | rank |
|------|-------|-------|------|
| 2010 | 1 | 19515 | 1 |
| 2010 | 3 | 18131 | 2 |
| 2010 | 7 | 17856 | 3 |
| 2011 | 1 | 18134 | 1 |
| 2011 | 7 | 17283 | 2 |
| 2011 | 10 | 17034 | 3 |
| 2012 | 1 | 17943 | 1 |
| 2012 | 8 | 17661 | 2 |
| 2012 | 5 | 17502 | 3 |
| 2013 | 8 | 17440 | 1 |
| 2013 | 1 | 16820 | 2 |
| 2013 | 7 | 16644 | 3 |
| 2014 | 7 | 12196 | 1 |
| 2014 | 10 | 12133 | 2 |
| 2014 | 8 | 12028 | 3 |
| 2015 | 10 | 19219 | 1 |

```
|2015| 8|19011| 2|
|2015| 7|18709| 3|
|2016| 10|19659| 1|
|2016| 8|19490| 2|
+----+-----+-----+----+
only showing top 20 rows
```

Q1 Dataframe time: 0.3153994083404541 seconds.

```
+----+-----+-----+----+
|Year|Month|count|rank|
+----+-----+-----+----+
|2010| 1|19515| 1|
|2010| 3|18131| 2|
|2010| 7|17856| 3|
|2011| 1|18134| 1|
|2011| 7|17283| 2|
|2011| 10|17034| 3|
|2012| 1|17943| 1|
|2012| 8|17661| 2|
|2012| 5|17502| 3|
|2013| 8|17440| 1|
|2013| 1|16820| 2|
|2013| 7|16644| 3|
|2014| 7|12196| 1|
|2014| 10|12133| 2|
|2014| 8|12028| 3|
|2015| 10|19219| 1|
|2015| 8|19011| 2|
|2015| 7|18709| 3|
|2016| 10|19659| 1|
|2016| 8|19490| 2|
+----+-----+-----+----+
only showing top 20 rows
```

Q1 SQL time: 0.6481153964996338 seconds.

Query 2

DataFrame\SQL API

```
1  def query2_df(df):
2
3
4  def day_part(hour):
5      if 500 <= hour < 1200:
6          return "Πρωί"
7      elif 1200 <= hour < 1700:
8          return "Απόγευμα"
9      elif 1700 <= hour < 2100:
10         return "Βράδυ"
11     else:
12         return "Νύχτα"
13
14  day_part_udf = udf(day_part, StringType())
15
16  df_day_part = df.withColumn("DayPart", day_part_udf(col("TIME OCC")))
17
18  df_street_crimes = df_day_part.filter(col("Premis Desc") == "STREET").groupBy("DayPart").count().orderBy(col("count").
19  ↪ desc())
20
21  return df_street_crimes
```

Your File Caption

RDD API

```
1 def query2_rdd(df):
2
3 def day_part(hour):
4     if 500 <= hour < 1200:
5         return "Πρωί"
6     elif 1200 <= hour < 1700:
7         return "Απόγευμα"
8     elif 1700 <= hour < 2100:
9         return "Βράδυ"
10    else:
11        return "Νύχτα"
12
13 rdd = df.rdd.filter(lambda row: row['Premis Desc'] == 'STREET')
14
15 def map_day_part(record):
16     hour = int(record["TIME OCC"])
17     part = day_part(hour)
18     return (part, 1)
19
20 rdd_mapped = rdd.map(map_day_part)
21 rdd_reduced = rdd_mapped.reduceByKey(lambda a, b: a + b)
22
23 rdd_street_crimes = rdd_reduced.sortBy(lambda x: x[1], ascending=False)
24
25 return rdd_street_crimes
```

Your File Caption

```
+-----+-----+
| DayPart| count|
+-----+-----+
| Νύχτα|231546|
| Βράδυ|182141|
| Απόγευμα|143974|
| Πρωί|120358|
+-----+-----+
```

Q2 Dataframe time: 0.31096601486206055 seconds.

[(Νύχτα, 231546), (Βράδυ, 182141), (Απόγευμα, 143974), (Πρωί, 120358)]

Q2 RDD time: 39.22880935668945 seconds.

Query 3

```
1 data3 = spark.read.csv("/user/ubuntu/ta/advanced-db/data/LA_income_2015.csv", header=True, schema=schema2)
2 data4 = spark.read.csv("/user/ubuntu/ta/advanced-db/data/revgecoding.csv", header=True, schema=schema3)
3
4 schema3 = "`LAT` DOUBLE, \
5           `LON` DOUBLE, \
6           `ZIPcode` INTEGER"
7
8 schema2 = "`Zip Code` INTEGER, \
9           `Community` STRING, \
10          `Estimated Median Income` STRING"
11
12 descent_mapping = {
13     'A': 'Other Asian',
14     'B': 'Black',
15     'C': 'Chinese',
16     'D': 'Cambodian',
17     'F': 'Filipino',
18     'G': 'Guamanian',
19     'H': 'Hispanic/Latin/Mexican',
20     'I': 'American Indian/Alaskan Native',
```

```

21 'J': 'Japanese',
22 'K': 'Korean',
23 'L': 'Laotian',
24 'O': 'Other',
25 'P': 'Pacific Islander',
26 'S': 'Samoan',
27 'U': 'Hawaiian',
28 'V': 'Vietnamese',
29 'W': 'White',
30 'X': 'Unknown',
31 'Z': 'Asian Indian'
32 }
33
34 data3 = data3.withColumn("Estimated Median Income", regexp_replace(col("Estimated Median Income"), "\$", ""))
35 data3 = data3.withColumn("Estimated Median Income", regexp_replace(col("Estimated Median Income"), ",", "").cast("
↪ float"))
36
37 crime_year = df.withColumn("Year", year("DATE OCC"))
38
39 crime_2015 = crime_year.filter(
40     (col("Year") == 2015) &
41     (col("Vict Descent").isNotNull()))
42
43 def map_descent(code):
44     return descent_mapping.get(code, "Unknown") # Default to "Unknown" if code not found
45
46 map_descent_udf = udf(map_descent, StringType())
47
48 crime_2015 = crime_2015.withColumn("Vict Descent", map_descent_udf(crime_2015["Vict Descent"]))
49
50 revgecoding = data4.dropDuplicates(['LAT', 'LON'])

```

Your File Caption

```

1 def query3 (crime_2015, data3, revgecoding):
2
3     crime_zip = crime_2015.join(revgecoding, ["LAT", "LON"], "left")
4
5     best3_zip = data3.orderBy("Estimated Median Income", ascending=False).limit(3)
6     worst3_zip = data3.orderBy("Estimated Median Income", ascending=True).limit(3)
7
8     best3_zip_list = [row['Zip Code'] for row in best3_zip.collect()]
9     worst3_zip_list = [row['Zip Code'] for row in worst3_zip.collect()]
10
11     crimes = crime_zip.filter(
12         (col("ZIPcode").isin(best3_zip_list)) |
13         (col("ZIPcode").isin(worst3_zip_list))
14     )
15
16     vict_descent_count = crimes.groupBy("Vict Descent").count().orderBy("count", ascending=False)
17
18
19     return vict_descent_count

```

Your File Caption

2 Executors

```

+-----+-----+
|      Vict Descent|count|
+-----+-----+
|Hispanic/Latin/Me...| 1053|
|          White|   610|
|          Black|   349|
|          Other|   272|
|          Unknown|   71|
|      Other Asian|   46|
|          Korean|    4|

```

| | Chinese | 1 |
|----------------------|---------|---|
| American Indian/A... | 1 | |

Number of Executors: 2
Q3 time: 10.069442987442017 seconds.

3 Executors

| | Vict Descent | count |
|----------------------|--------------|-------|
| Hispanic/Latin/Me... | 1053 | |
| White | 610 | |
| Black | 349 | |
| Other | 272 | |
| Unknown | 71 | |
| Other Asian | 46 | |
| Korean | 4 | |
| American Indian/A... | 1 | |
| Chinese | 1 | |

Number of Executors: 3
Q3 time: 5.2063148021698 seconds.

4 Executors

| | Vict Descent | count |
|----------------------|--------------|-------|
| Hispanic/Latin/Me... | 1053 | |
| White | 610 | |
| Black | 349 | |
| Other | 272 | |
| Unknown | 71 | |
| Other Asian | 46 | |
| Korean | 4 | |
| American Indian/A... | 1 | |
| Chinese | 1 | |

Number of Executors: 3
Q3 time: 5.2063148021698 seconds.

Query 4

```

1 schema4 = "`X` DOUBLE, \
2           `Y` DOUBLE, \
3           `FID` INTEGER, \
4           `DIVISION` STRING, \
5           `LOCATION` STRING, \
6           `PREC` INTEGER"
7
8
9 data5 = spark.read.csv("/user/ubuntu/ta/advanced-db/data/LAPD_Police_Stations.csv", header=True, schema=schema4)

```

Your File Caption

```

1 def query4(df, data5):
2
3     def haversine(lat1, lon1, lat2, lon2):
4         # Radius of the Earth in kilometers
5         R = 6371.0
6
7         lat1_rad = math.radians(lat1)
8         lon1_rad = math.radians(lon1)
9         lat2_rad = math.radians(lat2)
10        lon2_rad = math.radians(lon2)
11
12        dlat = lat2_rad - lat1_rad
13        dlon = lon2_rad - lon1_rad
14
15        a = math.sin(dlat / 2)**2 + math.cos(lat1_rad) * math.cos(lat2_rad) * math.sin(dlon / 2)**2
16        c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
17
18        distance = R * c
19        return distance
20
21    def get_distance(lat1, long1, lat2, long2):
22
23        def is_valid_coordinate(lat, lon):
24            return -90 <= lat <= 90 and -180 <= lon <= 180
25
26
27        if not is_valid_coordinate(lat1, long1) or not is_valid_coordinate(lat2, long2):
28            # Print the invalid rows
29            print(f"Invalid row: lat1={lat1}, long1={long1}, lat2={lat2}, long2={long2}")
30            return -1
31
32        try:
33            return haversine(lat1, long1, lat2, long2)
34        except ValueError:
35            return -1
36
37    df_4a = df.filter(
38        (df["AREA NAME"] != "Null Island") &
39        (df["Weapon Used Cd"].substr(1, 1) == "1")
40    )
41
42    df_4b = df.filter(
43        (df["AREA NAME"] != "Null Island") &
44        (df["Weapon Used Cd"].isNull())
45    )
46
47    joined_df_4a = df_4a.join(data5, df_4a["AREA"] == data5["PREC"])
48    joined_df_4b = df_4b.join(data5, df_4b["AREA"] == data5["PREC"])
49
50    distance_udf = udf(get_distance)
51
52    distance_df_4a = joined_df_4a.withColumn(
53        "DISTANCE",
54        distance_udf(
55            F.col("LAT"), F.col("LON"),
56            F.col("Y"), F.col("X")
57        ).cast("double")
58    )
59
60    distance_df_4b = joined_df_4b.withColumn(
61        "DISTANCE",
62        distance_udf(
63            F.col("LAT"), F.col("LON"),
64            F.col("Y"), F.col("X")
65        ).cast("double")
66    )
67
68    query_4_1a = distance_df_4a.groupBy("Year").agg(
69        F.count("*").alias("num_crimes"),
70        F.avg("DISTANCE").alias("average_distance")

```



```

71 ).orderBy("Year")
72
73 query_4_1b = distance_df_4b.groupBy("DIVISION").agg(
74     F.count("*").alias("num_crimes"),
75     F.avg("DISTANCE").alias("average_distance")
76 ).orderBy(F.desc("num_crimes"))
77
78 printΑπόσταση(" από το αστυνομικό τμήμα που ανέλαβε την έρευνα για το περιστατικό:")
79 print("(a)")
80 query_4_1a.show()
81 print("(b)")
82 query_4_1b.show()
83
84 cross_joined_df = df.crossJoin(data5.withColumnRenamed("LAT", "Y").withColumnRenamed("LON", "X"))
85
86 cross_joined_df = cross_joined_df.withColumn(
87     "DISTANCE",
88     distance_udf(col("LAT"), col("LON"), col("Y"), col("X")).cast("double")
89 )
90
91 windowSpec = Window.partitionBy("DR_NO").orderBy("DISTANCE")
92
93 nearest_station_df = cross_joined_df.withColumn(
94     "row_num",
95     F.row_number().over(windowSpec)
96 ).filter(col("row_num") == 1).drop("row_num")
97
98 cross_df_4a = df_4a.join(nearest_station_df.drop("Year"), "DR_NO")
99 cross_df_4b = df_4b.join(nearest_station_df.drop("Year"), "DR_NO")
100
101 query_4_2a = cross_df_4a.groupBy("Year").agg(
102     F.count("*").alias("num_crimes"),
103     F.avg("DISTANCE").alias("average_distance")
104 ).orderBy("Year")
105
106 query_4_2b = cross_df_4b.groupBy("DIVISION").agg(
107     F.count("*").alias("num_crimes"),
108     F.avg("DISTANCE").alias("average_distance")
109 ).orderBy(F.desc("num_crimes"))
110
111
112 printΑπόσταση(" από το πλησιέστερο αστυνομικό τμήμα:")
113 print("(a)")
114 query_4_2a.show()
115 print("(b)")
116 query_4_2b.show()

```

Your File Caption

Απόσταση από το αστυνομικό τμήμα που ανέλαβε την έρευνα για το περιστατικό:

(a)

```

+---+-----+-----+
|Year|num_crimes| average_distance|
+---+-----+-----+
|2010| 8213| 4.315547525861609|
|2011| 7232|2.7931783031826134|
|2012| 6550|37.401521647671025|
|2013| 5838| 2.826412721201962|
|2014| 4230|11.631025289489838|
|2015| 6763| 2.70609799276239|
|2016| 8100|2.7176445421299724|
|2017| 7788| 5.955847913803834|
|2018| 7413| 2.732823649229879|
|2019| 7129|2.7399419721721476|
|2020| 8491| 8.614767812336167|
|2021| 9767| 30.97834129556094|
|2022| 10025| 2.60865618645079|

```

|2023| 8741|2.5551410574543145|

+-----+

(b)

+-----+

| DIVISION|num_crimes| average_distance|

+-----+

| 77TH STREET| 94474|13.162079052889169|

| SOUTHEAST| 72832|14.527525557922345|

| SOUTHWEST| 72461| 9.898850561769162|

| CENTRAL| 63264|23.466578376496436|

| NEWTON| 61160|13.979683650325562|

| RAMPART| 55611|19.847575514305305|

| HOLLYWOOD| 50958|27.846180453246344|

| OLYMPIC| 48886| 17.19463769893097|

| PACIFIC| 42760|25.072161839219607|

| HOLLENBECK| 41393|19.600667247278746|

| MISSION| 40880|21.330305780965872|

| HARBOR| 40637|14.160243637196546|

|NORTH HOLLYWOOD| 39542|17.184089891996653|

| WILSHIRE| 37712|16.080369127182834|

| NORTHEAST| 37101|12.799215742840445|

| VAN NUYS| 36080| 19.92610084395629|

| WEST VALLEY| 33694|15.344662228609367|

| TOPANGA| 32340| 7.019388768312766|

| FOOTHILL| 32337|16.293601573548496|

| DEVONSHIRE| 28673| 16.71129045491336|

+-----+

only showing top 20 rows

Απόσταση από το πλησιέστερο αστυνομικό τμήμα:

(a)

+-----+

|Year|num_crimes| average_distance|

+-----+

|2010| 8213| 3.965480506097993|

|2011| 7232| 2.46181888566459|

|2012| 6550| 37.04806556244542|

|2013| 5838|2.4561803379459084|

|2014| 4230|11.240705060052028|

|2015| 6763| 2.38790278176303|

|2016| 8100|2.4291509215379303|

|2017| 7788| 5.620278866952371|

|2018| 7413| 2.409083506096955|

|2019| 7129|2.4301661049761214|

|2020| 8491| 8.305664894299344|

|2021| 9767|30.666116941658924|

|2022| 10025|2.3129679282459743|

|2023| 8741|2.2716948056968675|

+-----+

(b)

+-----+

| DIVISION|num_crimes| average_distance|

+-----+

| 77TH STREET| 78830|1.6735955739672674|

| SOUTHWEST| 78068| 2.161146839862122|

| HOLLYWOOD| 70652|1.9193991303646156|

| SOUTHEAST| 66697| 2.22223281229674|

| OLYMPIC| 60553|1.6657781632089452|

| CENTRAL| 59420| 0.866802366302343|

| WILSHIRE| 58032|2.4783030061160027|

```

|      RAMPART|    56297| 1.361662592838379|
|      VAN NUYS|   55252|2.8073800912108116|
|      NEWTON|    45398|1.5998998286047477|
|    HOLLENBECK|   43128| 326.0868010778892|
|      PACIFIC|   40356| 3.845258334030655|
| NORTH HOLLYWOOD|  40174|2.5926392412493717|
|      HARBOR|    39433| 3.686751365873389|
|    FOOTHILL|    38327| 3.977261444227064|
|    WEST VALLEY|   34804| 2.850125414289692|
|      TOPANGA|   32648| 3.045398368787074|
|    NORTHEAST|   27283| 3.765925137599394|
|      MISSION|   27016|3.7840372209112974|
|WEST LOS ANGELES|  22376| 2.712234758681144|
+-----+-----+-----+
only showing top 20 rows

```

hint & explain

Query 3

```

1 join_strategies = ["broadcast", "merge", "shuffle_hash", "shuffle_replicate_nl"]
2
3 for strategy in join_strategies:
4     print(f"Executing query with {strategy} join strategy")
5     query3_hne_results = query3_hne.query3(crime_2015, data3, revgecoding, strategy)
6     query3_hne_results.show()

```

Your File Caption

```

1 def query3(crime_2015, data3, revgecoding, join_strategy):
2
3     crime_zip = crime_2015.join(revgecoding.hint(join_strategy), ["LAT", "LON"], "left")
4
5     best3_zip = data3.orderBy("Estimated Median Income", ascending=False).limit(3)
6     worst3_zip = data3.orderBy("Estimated Median Income", ascending=True).limit(3)
7
8     best3_zip_list = [row['Zip Code'] for row in best3_zip.collect()]
9     worst3_zip_list = [row['Zip Code'] for row in worst3_zip.collect()]
10
11     crimes = crime_zip.filter(
12         (col("ZIPcode").isin(best3_zip_list)) |
13         (col("ZIPcode").isin(worst3_zip_list))
14     )
15
16     vict_descent_count = crimes.groupBy("Vict Descent").count().orderBy("count", ascending=False)
17
18     vict_descent_count.explain()
19
20     return vict_descent_count

```

Your File Caption

```

1 Executing query with broadcast join strategy
2 == Physical Plan ==
3 AdaptiveSparkPlan (53)
4 +- == Final Plan ==
5   TakeOrderedAndProject (31)
6   +- * HashAggregate (30)
7     +- AQEShuffleRead (29)
8       +- ShuffleQueryStage (28), Statistics(sizeInBytes=1016.0 B, rowCount=28)
9         +- Exchange (27)
10          +- * HashAggregate (26)
11            +- * Project (25)
12              +- * BroadcastHashJoin Inner BuildRight (24)

```

```

13      :- * Project (12)
14      :   +- BatchEvalPython (11)
15      :     +- * HashAggregate (10)
16      :       +- AQEShuffleRead (9)
17      :         +- ShuffleQueryStage (8), Statistics(sizeInBytes=88.4 MiB, rowCount=1.96E+5)
18      :           +- Exchange (7)
19      :             +- * HashAggregate (6)
20      :               +- Union (5)
21      :                 :- * Filter (2)
22      :                   :   +- Scan csv (1)
23      :                   :   +- * Filter (4)
24      :                   :     +- Scan csv (3)
25      +- BroadcastQueryStage (23), Statistics(sizeInBytes=8.0 MiB, rowCount=682)
26      +- BroadcastExchange (22)
27      +- * Project (21)
28      +- * Filter (20)
29      +- * HashAggregate (19)
30      +- AQEShuffleRead (18)
31      +- ShuffleQueryStage (17), Statistics(sizeInBytes=1475.8 KiB, rowCount=3.78E+4)
32      +- Exchange (16)
33      +- * HashAggregate (15)
34      +- * Filter (14)
35      +- Scan csv (13)
36 +- == Initial Plan ==
37   TakeOrderedAndProject (52)
38   +- HashAggregate (51)
39   +- Exchange (50)
40   +- HashAggregate (49)
41   +- Project (48)
42   +- BroadcastHashJoin Inner BuildRight (47)
43     :- Project (39)
44     :   +- BatchEvalPython (38)
45     :     +- HashAggregate (37)
46     :       +- Exchange (36)
47     :         +- HashAggregate (35)
48     :           +- Union (34)
49     :             :- Filter (32)
50     :               :   +- Scan csv (1)
51     :               :   +- Filter (33)
52     :               :     +- Scan csv (3)
53   +- BroadcastExchange (46)
54   +- Project (45)
55   +- Filter (44)
56   +- HashAggregate (43)
57   +- Exchange (42)
58   +- HashAggregate (41)
59   +- Filter (40)
60   +- Scan csv (13)
61
62 +-----+-----+
63 | Vict Descent|count|
64 +-----+-----+
65 | Hispanic/Latin/Me...| 1053|
66 | White| 610|
67 | Black| 349|
68 | Other| 272|
69 | Unknown| 71|
70 | Other Asian| 46|
71 | Korean| 4|
72 | Chinese| 1|
73 | American Indian/A...| 1|
74 +-----+-----+
75
76 Executing query with merge join strategy
77 == Physical Plan ==
78 AdaptiveSparkPlan (62)
79 +- == Final Plan ==
80   TakeOrderedAndProject (37)
81   +- * HashAggregate (36)
82   +- AQEShuffleRead (35)
83   +- ShuffleQueryStage (34), Statistics(sizeInBytes=584.0 B, rowCount=16)

```

```

84 +- Exchange (33)
85 +- * HashAggregate (32)
86 +- * Project (31)
87 +- * SortMergeJoin Inner (30)
88 :- * Sort (16)
89 : +- AQEShuffleRead (15)
90 : +- ShuffleQueryStage (14), Statistics(sizeInBytes=8.7 MiB, rowCount=1.96E+5)
91 : +- Exchange (13)
92 : +- * Project (12)
93 : +- BatchEvalPython (11)
94 : +- * HashAggregate (10)
95 : +- AQEShuffleRead (9)
96 : +- ShuffleQueryStage (8), Statistics(sizeInBytes=88.4 MiB, rowCount=1.96E
↪ +5)
97 : +- Exchange (7)
98 : +- * HashAggregate (6)
99 : +- Union (5)
100 : :- * Filter (2)
101 : : +- Scan csv (1)
102 : +- * Filter (4)
103 : +- Scan csv (3)
104 +- * Sort (29)
105 +- AQEShuffleRead (28)
106 +- ShuffleQueryStage (27), Statistics(sizeInBytes=16.0 KiB, rowCount=682)
107 +- Exchange (26)
108 +- * Project (25)
109 +- * Filter (24)
110 +- * HashAggregate (23)
111 +- AQEShuffleRead (22)
112 +- ShuffleQueryStage (21), Statistics(sizeInBytes=1475.8 KiB, rowCount
↪ =3.78E+4)
113 +- Exchange (20)
114 +- * HashAggregate (19)
115 +- * Filter (18)
116 +- Scan csv (17)
117 +- == Initial Plan ==
118 TakeOrderedAndProject (61)
119 +- HashAggregate (60)
120 +- Exchange (59)
121 +- HashAggregate (58)
122 +- Project (57)
123 +- SortMergeJoin Inner (56)
124 :- Sort (47)
125 : +- Exchange (46)
126 : +- Project (45)
127 : +- BatchEvalPython (44)
128 : +- HashAggregate (43)
129 : +- Exchange (42)
130 : +- HashAggregate (41)
131 : +- Union (40)
132 : :- Filter (38)
133 : : +- Scan csv (1)
134 : +- Filter (39)
135 : +- Scan csv (3)
136 +- Sort (55)
137 +- Exchange (54)
138 +- Project (53)
139 +- Filter (52)
140 +- HashAggregate (51)
141 +- Exchange (50)
142 +- HashAggregate (49)
143 +- Filter (48)
144 +- Scan csv (17)
145
146 +-----+-----+
147 | Vict Descent|count|
148 +-----+-----+
149 | Hispanic/Latin/Me...| 1053|
150 | White| 610|
151 | Black| 349|
152 | Other| 272|

```

```

153 |           Unknown |    71 |
154 |       Other Asian |   46 |
155 |           Korean |    4 |
156 | American Indian/A... |  1 |
157 |           Chinese |    1 |
158 +-----+-----+
159
160 Executing query with shuffle_hash join strategy
161 == Physical Plan ==
162 AdaptiveSparkPlan (58)
163 +- == Final Plan ==
164   TakeOrderedAndProject (35)
165   +- * HashAggregate (34)
166     +- AQEShuffleRead (33)
167       +- ShuffleQueryStage (32), Statistics(sizeInBytes=584.0 B, rowCount=16)
168         +- Exchange (31)
169           +- * HashAggregate (30)
170             +- * Project (29)
171               +- * ShuffledHashJoin Inner BuildRight (28)
172                 :- AQEShuffleRead (15)
173                   : +- ShuffleQueryStage (14), Statistics(sizeInBytes=8.7 MiB, rowCount=1.96E+5)
174                     +- Exchange (13)
175                       +- * Project (12)
176                         +- BatchEvalPython (11)
177                           +- * HashAggregate (10)
178                             +- AQEShuffleRead (9)
179                               +- ShuffleQueryStage (8), Statistics(sizeInBytes=88.4 MiB, rowCount=1.96E+5)
180                                 +- Exchange (7)
181                                   +- * HashAggregate (6)
182                                     +- Union (5)
183                                       :- * Filter (2)
184                                         : +- Scan csv (1)
185                                           +- * Filter (4)
186                                             +- Scan csv (3)
187   +- AQEShuffleRead (27)
188     +- ShuffleQueryStage (26), Statistics(sizeInBytes=16.0 KiB, rowCount=682)
189       +- Exchange (25)
190         +- * Project (24)
191           +- * Filter (23)
192             +- * HashAggregate (22)
193               +- AQEShuffleRead (21)
194                 +- ShuffleQueryStage (20), Statistics(sizeInBytes=1475.8 KiB, rowCount=3.78E
195                   +- Exchange (19)
196                     +- * HashAggregate (18)
197                       +- * Filter (17)
198                         +- Scan csv (16)
199 +- == Initial Plan ==
200   TakeOrderedAndProject (57)
201   +- HashAggregate (56)
202     +- Exchange (55)
203       +- HashAggregate (54)
204         +- Project (53)
205           +- ShuffledHashJoin Inner BuildRight (52)
206             :- Exchange (44)
207               : +- Project (43)
208                 +- BatchEvalPython (42)
209                   +- HashAggregate (41)
210                     +- Exchange (40)
211                       +- HashAggregate (39)
212                         +- Union (38)
213                           :- Filter (36)
214                             : +- Scan csv (1)
215                               +- Filter (37)
216                                 +- Scan csv (3)
217   +- Exchange (51)
218     +- Project (50)
219       +- Filter (49)
220         +- HashAggregate (48)
221           +- Exchange (47)
222             +- HashAggregate (46)

```

```

223 +- Filter (45)
224 +- Scan csv (16)
225
226 +-----+
227 | Vict Descent|count|
228 +-----+
229 |Hispanic/Latin/Me...| 1053|
230 |           White|    610|
231 |           Black|    349|
232 |           Other|    272|
233 |           Unknown|    71|
234 |           Other Asian|    46|
235 |           Korean|     4|
236 |           Chinese|     1|
237 |American Indian/A...|     1|
238 +-----+
239
240 Executing query with shuffle_replicate_n1 join strategy
241 == Physical Plan ==
242 AdaptiveSparkPlan (50)
243 +- == Final Plan ==
244   TakeOrderedAndProject (29)
245   +- * HashAggregate (28)
246     +- AQEShuffleRead (27)
247       +- ShuffleQueryStage (26), Statistics(sizeInBytes=3.5 KiB, rowCount=100)
248         +- Exchange (25)
249           +- * HashAggregate (24)
250             +- * Project (23)
251               +- CartesianProduct Inner (22)
252                 :- * Project (12)
253                   : +- BatchEvalPython (11)
254                     : +- * HashAggregate (10)
255                       : +- AQEShuffleRead (9)
256                         +- ShuffleQueryStage (8), Statistics(sizeInBytes=88.4 MiB, rowCount=1.96E+5)
257                           +- Exchange (7)
258                             +- * HashAggregate (6)
259                               +- Union (5)
260                                 :- * Filter (2)
261                                   : +- Scan csv (1)
262                                   +- * Filter (4)
263                                     +- Scan csv (3)
264               +- * Project (21)
265                 +- * Filter (20)
266                   +- * HashAggregate (19)
267                     +- AQEShuffleRead (18)
268                       +- ShuffleQueryStage (17), Statistics(sizeInBytes=1475.8 KiB, rowCount=3.78E+4)
269                         +- Exchange (16)
270                           +- * HashAggregate (15)
271                             +- * Filter (14)
272                               +- Scan csv (13)
273 +- == Initial Plan ==
274   TakeOrderedAndProject (49)
275   +- HashAggregate (48)
276     +- Exchange (47)
277       +- HashAggregate (46)
278         +- Project (45)
279           +- CartesianProduct Inner (44)
280             :- Project (37)
281               : +- BatchEvalPython (36)
282                 : +- HashAggregate (35)
283                   : +- Exchange (34)
284                     : +- HashAggregate (33)
285                       : +- Union (32)
286                         :- Filter (30)
287                           : +- Scan csv (1)
288                           +- Filter (31)
289                             +- Scan csv (3)
290           +- Project (43)
291             +- Filter (42)
292               +- HashAggregate (41)
293                 +- Exchange (40)

```

```

294 +- HashAggregate (39)
295 +- Filter (38)
296 +- Scan csv (13)
297

```

| | Vict | Descent | count |
|----------------------|---------|---------|-------|
| Hispanic/Latin/Me... | | | 1053 |
| | White | | 610 |
| | Black | | 349 |
| | Other | | 272 |
| | Unknown | | 71 |
| Other Asian | | | 46 |
| | Korean | | 4 |
| | Chinese | | 1 |
| American Indian/A... | | | 1 |

Your File Caption

Query 4

```

1 def query4(df, data5, join_strategy):
2
3     def haversine(lat1, lon1, lat2, lon2):
4         # Radius of the Earth in kilometers
5         R = 6371.0
6
7         lat1_rad = math.radians(lat1)
8         lon1_rad = math.radians(lon1)
9         lat2_rad = math.radians(lat2)
10        lon2_rad = math.radians(lon2)
11
12        dlat = lat2_rad - lat1_rad
13        dlon = lon2_rad - lon1_rad
14
15        a = math.sin(dlat / 2)**2 + math.cos(lat1_rad) * math.cos(lat2_rad) * math.sin(dlon / 2)**2
16        c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
17
18        distance = R * c
19        return distance
20
21    def get_distance(lat1, long1, lat2, long2):
22
23        def is_valid_coordinate(lat, lon):
24            return -90 <= lat <= 90 and -180 <= lon <= 180
25
26
27        if not is_valid_coordinate(lat1, long1) or not is_valid_coordinate(lat2, long2):
28            # Print the invalid rows
29            print(f"Invalid row: lat1={lat1}, long1={long1}, lat2={lat2}, long2={long2}")
30            return -1
31
32        try:
33            return haversine(lat1, long1, lat2, long2)
34        except ValueError:
35            return -1
36
37    df_4a = df.filter(
38        (df["AREA NAME"] != "Null Island") &
39        (df["Weapon Used Cd"].substr(1, 1) == "1")
40    )
41
42    df_4b = df.filter(
43        (df["AREA NAME"] != "Null Island") &
44        (df["Weapon Used Cd"].isNotNull())
45    )
46
47    joined_df_4a = df_4a.join(data5.hint(join_strategy), df_4a["AREA"] == data5["PREC"])

```



```

48 joined_df_4b = df_4b.join(data5.hint(join_strategy), df_4b["AREA"] == data5["PREC"])
49
50 joined_df_4a.explain()
51 joined_df_4b.explain()
52
53 distance_udf = udf(get_distance)
54
55 distance_df_4a = joined_df_4a.withColumn(
56     "DISTANCE",
57     distance_udf(
58         F.col("LAT"), F.col("LON"),
59         F.col("Y"), F.col("X")
60     ).cast("double")
61 )
62
63 distance_df_4b = joined_df_4b.withColumn(
64     "DISTANCE",
65     distance_udf(
66         F.col("LAT"), F.col("LON"),
67         F.col("Y"), F.col("X")
68     ).cast("double")
69 )
70
71 query_4_1a = distance_df_4a.groupBy("Year").agg(
72     F.count("*").alias("num_crimes"),
73     F.avg("DISTANCE").alias("average_distance")
74 ).orderBy("Year")
75
76 query_4_1b = distance_df_4b.groupBy("DIVISION").agg(
77     F.count("*").alias("num_crimes"),
78     F.avg("DISTANCE").alias("average_distance")
79 ).orderBy(F.desc("num_crimes"))
80
81 query_4_1a.explain()
82 query_4_1b.explain()
83
84 print("Απόσταση από το αστυνομικό τμήμα που ανέλαβε την έρευνα για το περιστατικό:")
85 print("(a)")
86 query_4_1a.show()
87 print("(b)")
88 query_4_1b.show()
89
90 cross_joined_df = df.crossJoin(data5.withColumnRenamed("LAT", "Y").withColumnRenamed("LON", "X"))
91
92
93 cross_joined_df = cross_joined_df.withColumn(
94     "DISTANCE",
95     distance_udf(col("LAT"), col("LON"), col("Y"), col("X")).cast("double")
96 )
97
98 windowSpec = Window.partitionBy("DR_NO").orderBy("DISTANCE")
99
100 nearest_station_df = cross_joined_df.withColumn(
101     "row_num",
102     F.row_number().over(windowSpec)
103 ).filter(col("row_num") == 1).drop("row_num")
104
105 cross_df_4a = df_4a.join(nearest_station_df.drop("Year"), "DR_NO")
106 cross_df_4b = df_4b.join(nearest_station_df.drop("Year"), "DR_NO")
107
108 query_4_2a = cross_df_4a.groupBy("Year").agg(
109     F.count("*").alias("num_crimes"),
110     F.avg("DISTANCE").alias("average_distance")
111 ).orderBy("Year")
112
113 query_4_2b = cross_df_4b.groupBy("DIVISION").agg(
114     F.count("*").alias("num_crimes"),
115     F.avg("DISTANCE").alias("average_distance")
116 ).orderBy(F.desc("num_crimes"))
117
118 print("Απόσταση από το πλησιέστερο αστυνομικό τμήμα:")

```

```
119 print("(a)")
120 query_4_2a.show()
121 print("(b)")
122 query_4_2b.show()
```

Your File Caption

Your File Caption

Your File Caption

Your File Caption

Your File Caption