Ερώτημα 1:
Γενικά όσον αφορά τα Dataframe API και Spark RDD API έχουμε:

- **RDD API**: Παρέχει χαμηλού επιπέδου αφαίρεση για τα κατανεμημένα δεδομένα. Ο χρήστης πρέπει να ορίζει ρητά τους μετασχηματισμούς και τις ενέργειες, ενώ διαχειρίζεται και τη σειριοποίηση και βελτιστοποίηση μόνος του. Έτσι, προσφέρει μεγαλύτερη ευελιξία, αλλά απαιτεί περισσότερη προσπάθεια και γραφή κώδικα. Όσον αφορά την απόδοση, Δεν εκμεταλλεύεται το query optimization και κάθε ενέργεια εκτελείται όπως ορίζεται, χωρίς προσαρμογές, κάτι που μπορεί να οδηγήσει σε λιγότερο αποδοτική εκτέλεση. Γενικά, η χρήση του ενδείκνυται για αρκετά πολύπλοκες λειτουργίες που απαιτούν λεπτομερή έλεγχο των δεδομένων ή όταν χρησιμοποιούνται προσαρμοσμένες μορφές δεδομένων, ωστόσο έχει μεγαλύτερες απαιτήσεις σε μνήμη και CPU.
- **DataFrame API**: Είναι πιο φιλικό προς το χρήστη, ειδικά αν έχει εξοικείωση με SQL, καθώς επιτρέπει ερωτήματα παρόμοια με αυτά των βάσεων δεδομένων. Εκμεταλλεύεται τον **Catalyst Optimizer** και την **Tungsten Execution Engine**, που: βελτιστοποιούν το πλάνο εκτέλεσης των ερωτημάτων, χρησιμοποιούν πιο αποδοτικούς αλγορίθμους για φιλτράρισμα, συνενώσεις (joins) και ομαδοποιήσεις (aggregations) και μπορούν να αποθηκεύσουν δεδομένα σε μορφή στηλών (columnar storage) για καλύτερη απόδοση. έτσι καθίσταται ταχύτερο και πιο αποδοτικό για μεγάλους όγκους δεδομένων.

  Οι παρατηρήσεις μας αυτές επιβεβαιώνονται και από τους χρόνους που παίρνουμε για τις 2 διαφορετικές εκτελέσεις:

```
+------------+--------------+
|   Age Group|Incident Count|
+------------+--------------+
|      Adults|        121052|
|Young Adults|         33588|
|    Children|         10825|
|     Elderly|          5985|
+------------+--------------+

DataFrame API execution time: 29.42403221130371 seconds
```

```
Adults: 121052 incidents
Young Adults: 33588 incidents
Children: 10825 incidents
Elderly: 5985 incidents
RDD API execution time: 38.487404346466064 seconds
```

Ερώτημα 2:

1)

- **DataFrame API:** Κάθε μετασχηματισμός προστίθεται στο logical query plan, το οποίο στη συνέχεια βελτιστοποιείται από το Spark. Απαιτεί περισσότερα βήματα, όπως withColumn, filter, groupBy και join, για να φτάσει στο επιθυμητό αποτέλεσμα.
- **SQL API:** Όλη η λογική δηλώνεται εκ των προτέρων σε ένα ερώτημα ενώ μειώνεται ο επιπλέον φόρτος από πολλαπλούς μετασχηματισμούς όταν αντιμετωπίζουμε σύνθετες διαδικασίες.

  Και τα δύο APIs καταλήγουν σε παρόμοια πλάνα εκτέλεσης, καθώς μεταφράζονται στο εσωτερικό λογικό μοντέλο του Spark. Οι διαφορές που ενδέχεται να προκύψουν οφείλονται στα ενδιάμεσα στάδια που δημιουργούνται από τα πολλά βήματα του DataFrame API. Παρατηρούμε στο ερώτημά μας ότι η διαφορά μεταξύ των χρόνων δεν είναι μεγάλη.

  Dataframe API:

```
Execution Time: 2.2361557483673096 seconds
+----+-----------+--------------------+----+
|Year|AREA NAME  |ClosedCasePercentage|Rank|
+----+-----------+--------------------+----+
|2010|Rampart    |32.85090742017      |1   |
|2010|Olympic    |31.515289821999087  |2   |
|2010|Harbor     |29.36028339237341   |3   |
|2011|Olympic    |35.03192688118192   |1   |
|2011|Rampart    |32.500296103280824  |2   |
|2011|Harbor     |28.516260162601625  |3   |
|2012|Olympic    |34.295435879385195  |1   |
|2012|Rampart    |32.461037450569904  |2   |
|2012|Harbor     |29.534834324553948  |3   |
|2013|Olympic    |33.58217940999398   |1   |
|2013|Rampart    |32.1060382916053    |2   |
```

  SQL API:

```
|2022|West Valley|26.536367172306498  |1   |
|2022|Harbor     |26.337538060026098  |2   |
|2022|Topanga    |26.236786469344608  |3   |
|2023|Foothill   |26.750524109014673  |1   |
|2023|Topanga    |26.538022616453986  |2   |
|2023|Mission    |25.662731120516817  |3   |
|2024|Foothill   |18.667786174083403  |1   |
|2024|77th Street|17.6147382029735    |2   |
|2024|Mission    |17.187827911857294  |3   |
+----+-----------+--------------------+----+

Query execution time: 2.17 seconds
```

  2) Επιλέξαμε την υλοποίηση του Dataframe API για να πειραματιστούμε με το .parquet τύπο αρχείων και έχουμε:

  - **CSV**: Τα αρχεία CSV είναι λιγότερο αποδοτικά σε ταχύτητα ανάγνωσης/εγγραφής λόγω της απουσίας βελτιστοποιημένης μορφοποίησης.

Η ανάγνωση δεδομένων από CSV απαιτεί χρόνο για την αναγνώριση τύπων δεδομένων και τη διαχείριση του text parsing.

- **Parquet:** Το Parquet είναι μία column-based μορφή αρχείου που συμπιέζει δεδομένα και βελτιστοποιεί την ανάγνωση συγκεκριμένων στηλών. Η εγγραφή σε Parquet είναι αργή λόγω της συμπίεσης, αλλά η ανάγνωση είναι σημαντικά ταχύτερη.

Οι παρατηρήσεις μας επιβεβαιώνονται από τα αποτελέσματα που πήραμε, όπου και βλέπουμε σημαντική μείωση του χρόνου (1 τάξη μεγέθους!).

```
Execution Time (Parquet): 0.36 seconds
+----+-----------+--------------------+----+
|Year|AREA NAME  |ClosedCasePercentage|Rank|
+----+-----------+--------------------+----+
|2010|Rampart    |32.85090742017      |1   |
|2010|Olympic    |31.515289821999087  |2   |
|2010|Harbor     |29.36028339237341   |3   |
|2011|Olympic    |35.03192688118192   |1   |
|2011|Rampart    |32.500296103280824  |2   |
|2011|Harbor     |28.516260162601625  |3   |
|2012|Olympic    |34.295435879385195  |1   |
|2012|Rampart    |32.461037450569904  |2   |
```