Τμήμα Εφαρμοσμένης Πληροφορικής

Μάθημα: Ανάλυση Δεδομένων μεγάλου Όγκου (Big Data)

Θέμα: Flight Delay Analytics με Apache Spark

Ονοματεπώνυμο: Βασίλειος Ραφαήλ Αβραμίδης

Αριθμός Μητρώου: ics23033

Διδάσκων: Ασημίνα Δημαρά

1. Σκοπός και Μεθοδολογία 3
1.1 Στόχος της Εργασίας 3
1.2 Dataset 3
1.3 Περιβάλλον Εκτέλεσης 4
2. Υλοποίηση 4
2.1 Ερώτημα RDD 4
2.2 Ερώτημα DataFrame 6
3. Οπτικοποίηση Δεδομένων 8
4. Σύγκριση RDD-DataFrame 10
4.1 Ευκολία Υλοποίησης και Εκφραστικότητα Κώδικα 10
4.2 Χρόνος Εκτέλεσης 10
4.3 Δυνατότητες Βελτιστοποίησης (Catalyst Optimizer) 12
4.4 Συμπέρασμα 13
5. Επέκταση Ανάλυσης 13
5.1 Μέση καθυστέρηση ανά ώρα αναχώρησης 14
5.2 Κατανομή καθυστερήσεων ανά αεροπορική εταιρεία 15
5.3 Κατανομή καθυστερήσεων ανά μήνα 15
6. Βιβλιογραφία 16

1. Σκοπός και Μεθοδολογία

1.1 Στόχος της Εργασίας

Η παρούσα εργασία έχει ως στόχο τη σύγκριση των Apache Spark RDDs και του Apache Spark DataFrame API στην ανάλυση μεγάλου όγκου δεδομένων. Η αξιολόγηση περιλαμβάνει τη μέτρηση των χρόνων εκτέλεσης και τη σύγκριση της ευκολίας υλοποίησης. Για την επίτευξη αυτού του σκοπού χρησιμοποιήθηκε ένα dataset που περιέχει δεδομένα πτήσεων.

1.2 Dataset

Το dataset περιλαμβάνει 2.000 εγγραφές πτήσεων σε αρχείο .csv και περιέχει τα βασικά πεδία που χρησιμοποιήθηκαν για την ανάλυση:

- **FL_DATE**: ημερομηνία εκτέλεσης της πτήσης.
- **AIRLINE**: κωδικός αεροπορικής εταιρείας.
- **FL_NUM**: αριθμός πτήσης.
- **ORIGIN_AIRPORT**: κωδικός αεροδρομίου αναχώρησης.
- **DEST_AIRPORT**: κωδικός αεροδρομίου προορισμού.
- **SCHED_DEP**: προγραμματισμένη ώρα αναχώρησης.
- **DEP_DELAY**: καθυστέρηση αναχώρησης σε λεπτά.
- **SCHED_ARR**: προγραμματισμένη ώρα άφιξης.
- **ARR_DELAY**: καθυστέρηση άφιξης σε λεπτά.
- **DIST_KM**: απόσταση διαδρομής σε χιλιόμετρα.
- **CANCELLED**: κατάσταση πτήσης (0 = εκτελέστηκε, 1 = ακυρώθηκε).

1.3 Περιβάλλον Εκτέλεσης

Για την υλοποίηση του στόχου χρησιμοποιήθηκε το περιβάλλον Google Colab το οποίο επιτρέπει την συγγραφή κώδικα Python σε περιβάλλον notebook, παρέχοντας άμεση εκτέλεση κώδικα και οπτικοποίηση αποτελεσμάτων.

Η ανάλυση έγινε με την χρήση της Python 3.12.12 και της βιβλιοθήκης PySpark 3.5.1. Το περιβάλλον εκτέλεσης του Google Colab χρησιμοποίησε CPU AMD EPYC 7B12 με 2 vCPUs και 12,7 GB RAM.

Για την διευκόλυνση του χρήστη, έχει δημιουργηθεί ένα GitHub repository που περιέχει το αρχείο .csv με τα δεδομένα των πτήσεων και αντλείται από εκεί αυτόματα.

2. Υλοποίηση

Στα πλαίσια της σύγκρισης των 2 API, δόθηκαν 2 διαφορετικά ερωτήματα, τα οποία επιλύθηκαν αντίστοιχα με RDD και DataFrame, ώστε να αξιολογηθεί η απόδοση και η ευκολία υλοποίησης κάθε προσέγγισης.

2.1 Ερώτημα RDD

Για την αξιολόγηση των RDDs, ζητήθηκε να υπολογιστεί ο μέσος χρόνος καθυστέρησης αναχώρησης ανά αεροδρόμιο και ύστερα να εμφανιστούν τα 10 αεροδρόμια με τον μεγαλύτερο μέσο χρόνο καθυστέρησης αναχώρησης.

Αρχικά, έπρεπε να αφαιρεθούν οι ακυρωμένες πτήσεις (CANCELLED == 1) και εγγραφές με κενές ή αρνητικές τιμές στο πεδίο καθυστέρησης αναχώρησης (DEP_DELAY), ώστε να διασφαλιστεί η ακρίβεια των υπολογισμών.

```
filtered_rdd = flight_rdd.filter(lambda line: line.split(",")[10] == '0' \
                                 and line.split(",")[8].isdigit() \
                                 and line.split(",")[6].isdigit() \
                                 and int(line.split(",")[6]) > 0)\
                                 .map(lambda line: line.split(","))
```

Ο παραπάνω κώδικας πραγματοποίησε την εν λόγω διαδικασία με την μέθοδο .filter() των RDDs.

Κρίθηκε αναγκαίο να γίνει ομαδοποίηση σε ζεύγη αεροδρομίων και καθυστερήσεων αναχώρησης.

```
average_rdd = filtered_rdd\
.map(lambda col: (col[3], (int(col[6]), 1)))
```

Ο παραπάνω κώδικας δημιουργεί ένα νέο RDD με ζεύγη, όπου το key είναι το αεροδρόμιο αναχώρησης (ORIGIN_AIRPORT) και το value είναι ένα ζεύγος (DEP_DELAY, 1). Το δεύτερο στοιχείο του value χρησιμοποιείται ως μετρητής για τον υπολογισμό του μέσου χρόνου καθυστέρησης ανά αεροδρόμιο.

Στον παρακάτω κώδικα, κάθε ζεύγος (key, value) συναθροίζεται με τη χρήση της μεθόδου .reduceByKey(). Για κάθε αεροδρόμιο αναχώρησης αθροίζονται ξεχωριστά οι τιμές των καθυστερήσεων αναχώρησης και οι αντίστοιχοι μετρητές. Στη συνέχεια, με τη μέθοδο .mapValues() δημιουργούνται ζεύγη όπου το key είναι το αεροδρόμιο αναχώρησης και το value ο μέσος όρος καθυστέρησης, ο οποίος

υπολογίζεται ως το πηλίκο του αθροίσματος των καθυστερήσεων προς τον μετρητή.

```
.reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))\
.mapValues(lambda b: round(b[0] / b[1], 2))
```

Μέσω της μεθόδου .takeOrdered() στον παραπάνω κώδικα μπορούμε να πάρουμε ταξινομημένα δεδομένα. Επειδή, όμως, η default διάταξη που επιστρέφει είναι αύξουσα, μέσω των lambda function αντιστρέφουμε το πρόσημο στα key και επιτυγχάνουμε φθίνουσα διάταξη.

```
top10 = average_rdd.takeOrdered(10, key=lambda x: -x[1])
```

Τέλος, τα αποτελέσματα αποθηκεύονται σε αρχείο .csv στον τοπικό υπολογιστή.

2.2 Ερώτημα DataFrame

Για την αξιολόγηση των DataFrames, ζητήθηκε να υπολογιστεί ο μέσος χρόνος καθυστέρησης αναχώρησης ανά δρομολόγιο και ύστερα να εμφανιστούν τα 10 δρομολόγια με τον μεγαλύτερο μέσο χρόνο καθυστέρησης αναχώρησης.

Παρομοίως με το προηγούμενο ερώτημα, έπρεπε να αφαιρεθούν οι ακυρωμένες πτήσεις (CANCELLED == 1) και οι εγγραφές με κενές ή αρνητικές τιμές στο πεδίο καθυστέρησης αναχώρησης.

Λόγω της εκφραστικότητας που παρέχει το Spark DataFrame API, η υλοποίηση είναι πιο απλή και ξεκάθαρη.

```
delayed_flights = (
    flights_df
    .na.drop(subset=["DEP_DELAY", "ARR_DELAY"]) #Filtering out NU
    .filter((col("DEP_DELAY") > 0) & (col("CANCELLED") == 0)) #Fi
)
```

Μέσω της μεθόδου .na.drop() αφαιρούνται οι κενές τιμές από τα πεδία καθυστέρησης αναχώρησης και άφιξης, ενώ μέσω της μεθόδου .filter() γίνεται το φιλτράρισμα έτσι ώστε να μείνουν εγγραφές με ενεργές πτήσεις που είχαν καθυστέρηση αναχώρησης.

```
grouped_routes = delayed_flights.groupBy("ORIGIN_AIRPORT", "DEST_AIRPORT") #Groupi

top_average_delay = (
    grouped_routes
    .avg("DEP_DELAY") #Calculating the average value of the 'DEP_DELAY' column.
    .orderBy("avg(DEP_DELAY)", ascending=False) #Ordering the DataFrame by the 'av
    .limit(10) #Keeping only the top-10 routes.
)
```

Ύστερα, μέσω της μεθόδου .groupBy() τα δεδομένα οργανώνονται σε ζεύγη δρομολογίων (ORIGIN_AIRPORT, DEST_AIRPORT). Για να υπολογιστεί ο μέσος όρος μιας στήλης στο Spark DataFrame API γίνεται εύκολα μέσω της μεθόδου .avg() χωρίς να είναι απαραίτητοι λοιποί μετασχηματισμοί. Τέλος, με τις μεθόδους .orderBy() και .limit(10) τα δεδομένα ταξινομούνται κατά φθίνουσα σειρά (ascending = False) και κρατώνται οι πρώτες 10 διαδρομές με την μεγαλύτερη μέση καθυστέρηση αναχώρησης.

Τα αποτελέσματα αποθηκεύονται σε αρχείο .csv στον τοπικό υπολογιστή.

3. Οπτικοποίηση Δεδομένων

Τα αποτελέσματα των 2 ερωτημάτων αποθηκεύονται στον τοπικό υπολογιστή ως αρχεία .csv.

Το αποτέλεσμα του ερωτήματος των Apache RDDs:

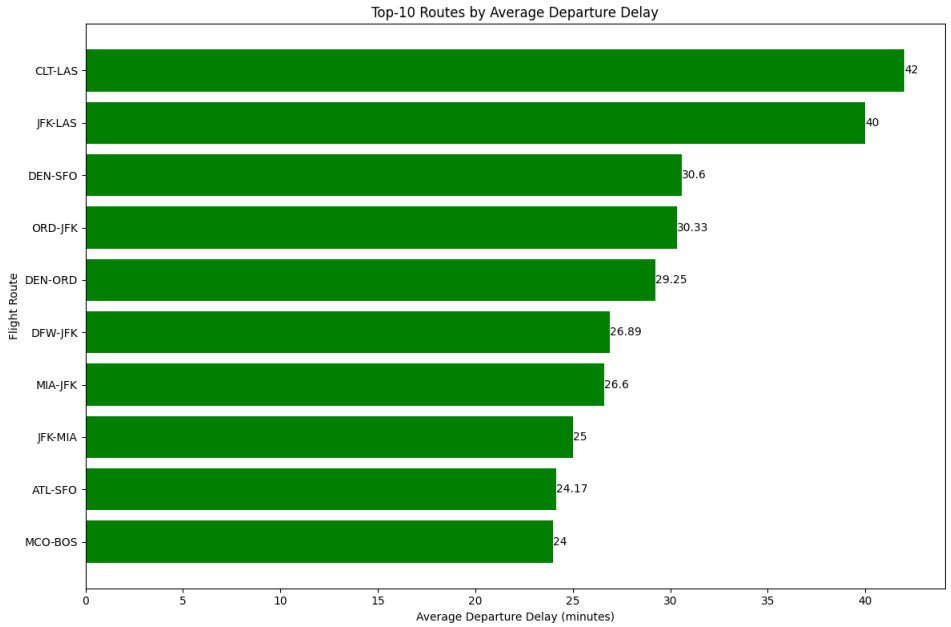
```
ORIGIN_AIRPORT,AVG_DEP_DELAY
DFW,18.9
SEA,17.28
ORD,16.84
JFK,16.14
LAS,15.92
DEN,15.69
PHX,15.59
CLT,15.4
MIA,15.09
BOS,14.78
```

Το αποτέλεσμα του ερωτήματος του Apache DataFrame API:

```
ORIGIN_AIRPORT,DEST_AIRPORT,AVG_DEP_DELAY
CLT,LAS,42.00
JFK,LAS,40.00
DEN,SFO,30.60
ORD,JFK,30.33
DEN,ORD,29.25
DFW,JFK,26.89
MIA,JFK,26.60
JFK,MIA,25.00
ATL,SFO,24.17
MCO,BOS,24.00
```

Όπως διακρίνουμε, δεν είναι ευανάγνωστο για έναν μέσο χρήστη το αποτέλεσμα για αυτό κρίθηκε ότι έπρεπε να δημιουργηθεί γράφημα με χρήση της βιβλιοθήκης Matplotlib της Python.

Το γράφημα για το ερώτημα των DataFrame:



Το παραπάνω γράφημα ράβδων απεικονίζει τα δρομολόγια ως προς την μέση καθυστέρηση αναχώρησης τους.

Παρατηρούμε ότι το δρομολόγιο από το αεροδρόμιο με κωδικό CLT προς το αεροδρόμιο με κωδικό LAS έχει την μεγαλύτερη μέση καθυστέρηση αναχώρησης με τιμή 42 λεπτά.

4. Σύγκριση RDD-DataFrame

Η σύγκριση μεταξύ RDDs και Spark DataFrame API έγινε με βάση τα εξής κριτήρια:

- **Ευκολία Υλοποίησης / Εκφραστικότητα Κώδικα** – πόσο εύκολα και καθαρά μπορεί να γραφεί και να συντηρηθεί ο κώδικας.
- **Χρόνος Εκτέλεσης** – απόδοση και ταχύτητα επεξεργασίας των δεδομένων.
- **Δυνατότητες Βελτιστοποίησης (Catalyst Optimizer)** – υποστήριξη αυτόματων βελτιστοποιήσεων και χρήση schema για βελτίωση της απόδοσης.

4.1 Ευκολία Υλοποίησης και Εκφραστικότητα Κώδικα

Τα RDDs δεν προσφέρουν υψηλού επιπέδου συναρτήσεις όπως το .groupBy() ή το .filter() που κάνουν τον κώδικα πιο σύντομο και κατανοητό. Αντιθέτως, πρέπει να σκέφτεσαι χειροκίνητα πώς θα κατανεμηθούν τα δεδομένα, να χειριστείς τύπους και δομές χωρίς schema και να γράψεις αρκετό χαμηλού επιπέδου κώδικα για μετασχηματισμούς και συγκεντρώσεις. Αυτό καθιστά την υλοποίηση πιο περίπλοκη και χρονοβόρα σε σύγκριση με το πιο δηλωτικό και αυτοματοποιημένο DataFrame API.

4.2 Χρόνος Εκτέλεσης

Για την αξιολόγηση της απόδοσης των RDDs και των DataFrames, το πρόγραμμα εκτελέστηκε συνολικά πέντε (5) φορές. Από τις εκτελέσεις αυτές, αφαιρέθηκαν η μεγαλύτερη και η μικρότερη τιμή χρόνου, ώστε να υπολογιστεί ο μέσος όρος των τριών ενδιάμεσων εκτελέσεων.

Καταγράφηκαν δύο διαφορετικοί χρόνοι για κάθε εκτέλεση:

- **Συνολικός χρόνος εκτέλεσης**: Ο χρόνος από την εκκίνηση του προγράμματος έως την παραγωγή του τελικού αποτελέσματος.
- **Χρόνος πρώτου action**: Η χρονική στιγμή κατά την οποία ο Spark εκτελεί πραγματικά τον υπολογισμό, λόγω της lazy evaluation. Στο Spark, οι μετασχηματισμοί (transformations) δεν εκτελούνται άμεσα, αλλά μόνο όταν κληθεί ένα action, όπως count(), collect() ή show().

Ακολουθούν οι μετρήσεις για τα RDD's:

Table5

RDD	Total Runtime
Test 1	1.378
Test 3	1.356
Test 2	1.342
Test 5	1.325
Test 4	1.281
Average	1.341

Table6

RDD	First Actior
Test 1	1.359
Test 3	1.333
Test 2	1.321
Test 5	1.300
Test 4	1.262
Average	1.318

Ακολουθούν οι μετρήσεις για το DataFrame API:

Table2

DataFrame	Total Runtime
Test 4	1.206
Test 1	0.874
Test 3	0.711
Test 2	0.699
Test 5	0.613
Average	0.761

Table3

DataFrame	First Actior
Test 4	0.985
Test 1	0.771
Test 3	0.592
Test 2	0.559
Test 5	0.485
Average	0.641

Παρατηρούμε ότι και στις 2 υλοποιήσεις τον περισσότερο χρόνο τον καταλαμβάνει το “First Action”, καθώς το Spark δεν εκτελεί άμεσα τους μετασχηματισμούς που γράφουμε στον κώδικα.

Οι μετασχηματισμοί όπως map(), filter(), flatMap() ή groupBy() δημιουργούν νέα RDDs ή DataFrames από τα υπάρχοντα δεδομένα, αλλά δεν εκτελούνται άμεσα λόγω της lazy evaluation. Η πραγματική εκτέλεση των μετασχηματισμών γίνεται μόνο όταν κληθεί κάποιο action, δηλαδή εντολές που επιστρέφουν αποτέλεσμα ή αποθηκεύουν δεδομένα σε εξωτερικό αποθηκευτικό μέσο.

Κλασικά παραδείγματα actions είναι:

- count() → επιστρέφει τον αριθμό των στοιχείων
- show() → εμφανίζει τα δεδομένα (σε DataFrames)
- write.csv() → αποθηκεύει τα δεδομένα σε αρχείο .csv.

4.3 Δυνατότητες Βελτιστοποίησης (Catalyst Optimizer)

Η ανάλυση των αποτελεσμάτων δείχνει, επίσης, ότι οι υλοποιήσεις με Apache Spark DataFrames εκτελούνται περίπου δύο φορές ταχύτερα σε σχέση με τα αντίστοιχα RDDs.

Το πλεονέκτημα αυτό οφείλεται στον Catalyst Optimizer, ο οποίος αναλύει την αλυσίδα των transformations και παράγει ένα βελτιστοποιημένο σχέδιο εκτέλεσης, μειώνοντας περιττούς υπολογισμούς και μεταφορές δεδομένων.

Αντίθετα, τα Apache Spark RDDs δεν επωφελούνται από τον Catalyst, με αποτέλεσμα ο προγραμματιστής να πρέπει να εφαρμόσει χειροκίνητα τις βελτιστοποιήσεις, γεγονός που αυξάνει τον χρόνο εκτέλεσης

4.4 Συμπέρασμα

Με βάση τα αποτελέσματα, η χρήση του Apache DataFrame API θεωρείται καταλληλότερη για μεγάλα σύνολα δεδομένων. Η εκτέλεση των DataFrames ήταν περίπου δύο φορές ταχύτερη σε σχέση με τα RDDs, χάρη στον Catalyst Optimizer. Επιπλέον, τα DataFrames προσφέρουν ευκολία υλοποίησης και πιο εκφραστικό κώδικα, καθιστώντας τα πιο προσβάσιμα και κατανοητά για νέους προγραμματιστές, σε αντίθεση με τα RDDs που απαιτούν περισσότερη χειρωνακτική βελτιστοποίηση και μεγαλύτερη εμπειρία για αποδοτική χρήση.

5. Επέκταση Ανάλυσης

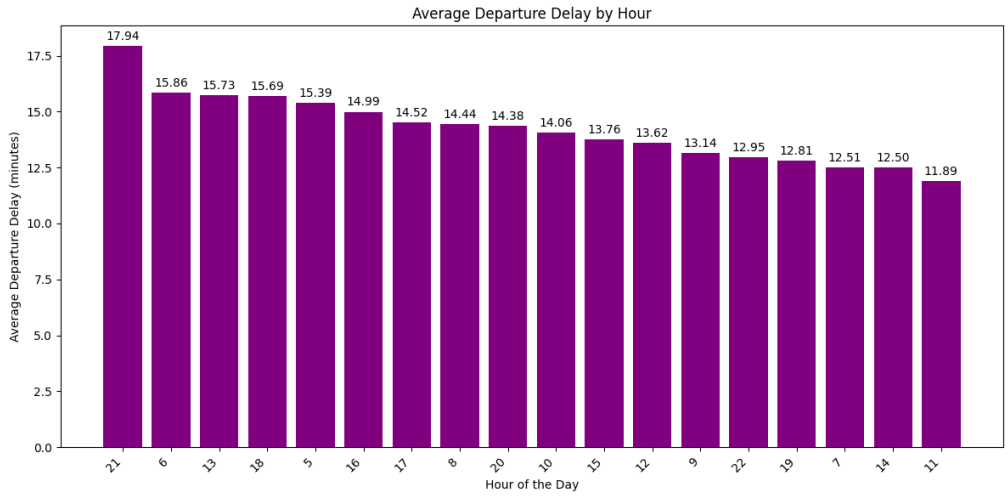
Η ανάλυση εμπλουτίστηκε με σκοπό να προσφέρει πιο αναλυτική εικόνα πτήσεων, αναδεικνύοντας μοτίβα και τάσεις που δεν ήταν ορατά από τα συνολικά στατιστικά στοιχεία.

5.1 Μέση καθυστέρηση ανά ώρα αναχώρησης

Υπολογίζεται η μέση καθυστέρηση για κάθε ώρα, αναδεικνύοντας τις ώρες της ημέρας που εμφανίζουν μεγαλύτερες ή μικρότερες καθυστερήσεις.

- Με αυτόν τον τρόπο μπορούμε να εντοπίσουμε τα χρονικά διαστήματα υψηλού ρίσκου, π.χ. ώρες αιχμής στα αεροδρόμια.

Ακολουθεί η οπτικοποίηση:



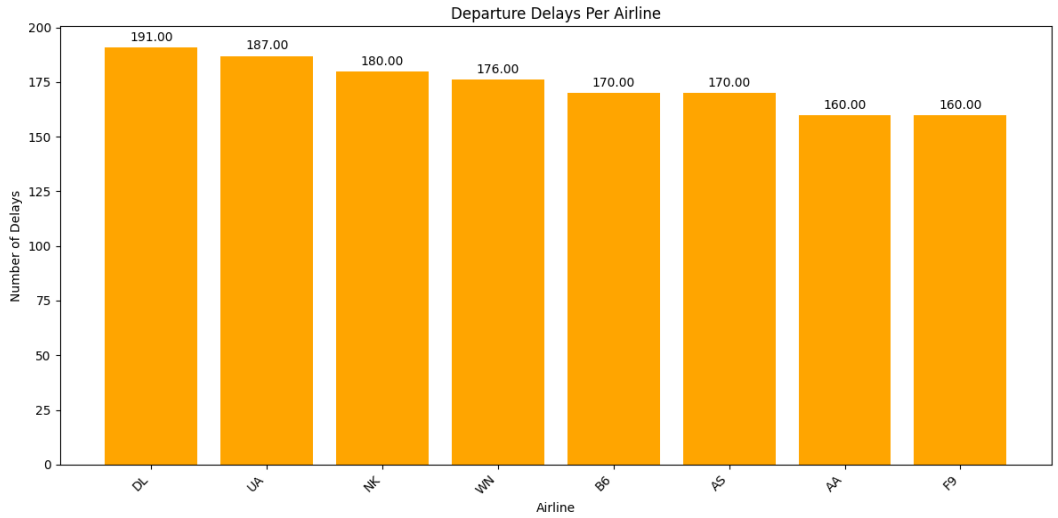
Παρατηρούμε ότι οι μέσες καθυστερήσεις αναχώρησης είναι μεγαλύτερες τις βραδινές ώρες, με τις χαμηλότερες καθυστερήσεις να παρατηρούνται νωρίτερα μέσα στην ημέρα.

5.2 Κατανομή καθυστερήσεων ανά αεροπορική εταιρεία

Υπολογίζεται ο αριθμός των καθυστερήσεων ανά αεροπορική εταιρεία, αναδεικνύοντας τις εταιρίες με τις περισσότερες ή τις λιγότερες καθυστερήσεις.

- Με αυτόν τον τρόπο μπορούμε να κατανοήσουμε του επιπέδου αξιοπιστίας των αεροπορικών εταιρειών.

Ακολουθεί η οπτικοποίηση:



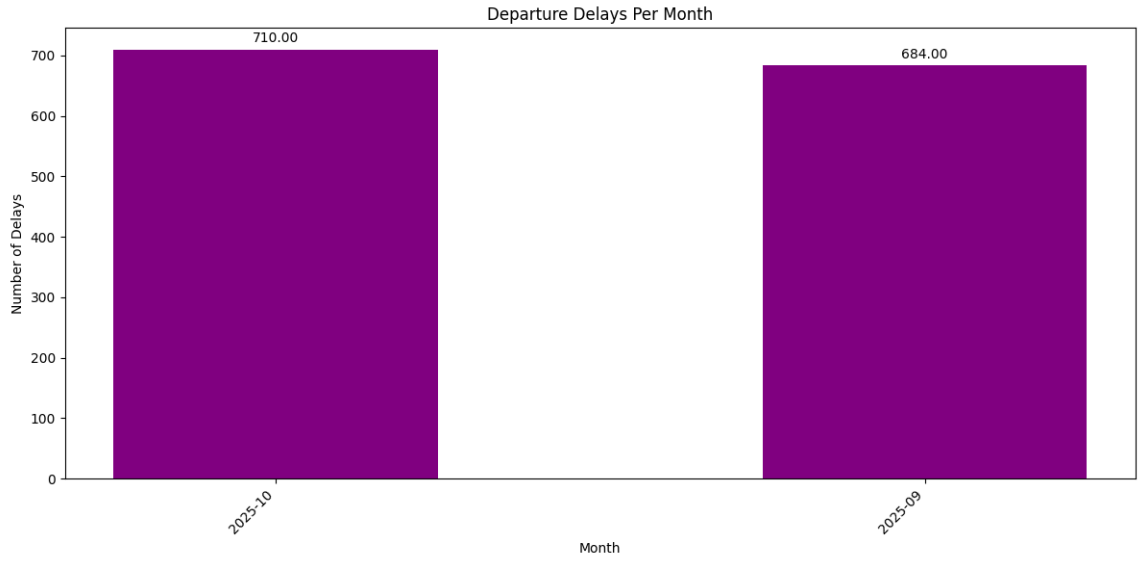
Παρατηρούμε ότι η εταιρεία με κωδικό DL παρουσίασε τις περισσότερες συνολικές καθυστερήσεις. Αντίθετα, η εταιρεία με κωδικό F9 εμφάνισε τις λιγότερες καθυστερήσεις, γεγονός που την καθιστά την πιο συνεπή στα χρονοδιαγράμματά της.

5.3 Κατανομή καθυστερήσεων ανά μήνα

Υπολογίζεται ο αριθμός των καθυστερήσεων ανά μήνα, αναδεικνύοντας τους μήνες με τις περισσότερες ή τις λιγότερες καθυστερήσεις.

- Με αυτόν τον τρόπο μπορούμε να καταλάβουμε καλύτερα πότε εμφανίζονται πιο συχνά προβλήματα στις πτήσεις.

Ακολουθεί η οπτικοποίηση:



Παρατηρούμε ότι ο μήνας Οκτώβριος παρουσίασε τις περισσότερες καθυστερήσεις, ενώ ο Σεπτέμβριος είχε τις λιγότερες. Αυτό δείχνει ότι τον Οκτώβριο σημειώθηκαν πιο συχνά προβλήματα στις πτήσεις σε σχέση με τον προηγούμενο μήνα.

6. Βιβλιογραφία

Τα παρακάτω εργαλεία τεχνητής νοημοσύνης χρησιμοποιήθηκαν στην επεξήγηση και ανάλυση του τρόπου λειτουργίας των μεθόδων του Apache Spark DataFrame API και των Apache Spark RDD's αντίστοιχα. Ενώ επιπλέον, το παρακάτω YouTube βίντεο και οι διαφάνειες του μαθήματος παρουσίασαν τις διαφορές που έχουν μεταξύ τους αυτές οι δύο εναλλακτικές.

- Google. (2025). Gemini 3 Pro [Generative AI model] https://gemini.google.com/
- OpenAI. (2025). ChatGPT (model GPT-5) [Large language model]. https://chatgpt.com/
- Perplexity AI. (2025). Perplexity (Large language model). https://www.perplexity.ai/
- BigDataELearning. (2021, July 8). RDD vs Dataframe vs Dataset [Video]. YouTube. https://www.youtube.com/watch?v=26zl50iNBp8