# Μεθοδολογία

## Ερευνητική διαδικασία (θα συμπληρωθεί μετά την ολοκλήρωση της μεθόδου RFM)

## Το σύνολο δεδομένων: περιγραφή, σχολιασμός, και καθαρισμός (παρατίθεται στις επόμενες σελίδες)

## Η μέθοδος RFM (θα συμπληρωθεί στο επόμενο βήμα, πριν την Ερευνητική διαδικασία)

# 2. Το σύνολο δεδομένων

## 2.1 Περιγραφή συνόλου δεδομένων

Για την παρούσα ανάλυση χρησιμοποιήθηκαν δεδομένα από τις συναλλαγές πελατών ενός ηλεκτρονικού καταστήματος με είδη δώρων. Το κατάστημα εδρεύει στο Ηνωμένο Βασίλειο και εξυπηρετεί πελάτες λιανικής αλλά και χονδρικής. Το εν λόγω σύνολο δεδομένων αντλήθηκε από την ιστοσελίδα Kaggle[[1]](#footnote-1), ωστόσο είναι επίσης διαθέσιμο και στην ιστοσελίδα UCI Machine Learning Repository[[2]](#footnote-2), από όπου αντλούμε πληροφορίες για τις μεταβλητές του. Τα δεδομένα βρίσκονται σε μορφή excel και είναι χωρισμένα σε δύο καρτέλες (*Year 2009-2010*, *Year 2010-2011*)σύμφωνα με το έτος πραγματοποίησης των συναλλαγών*.* Οι συναλλαγές χαρακτηρίζονται από οκτώ μεταβλητές, οι οποίες παρουσιάζονται παρακάτω:

1. **Invoice**: Αριθμός τιμολογίου. Είναι εξαψήφιος για την κάθε συναλλαγή. Αν ξεκινάει με το γράμμα “c” υποδηλώνει ακύρωση τιμολογίου (cancellation).
2. **StockCode**: Κωδικός προϊόντος. Είναι ένας πενταψήφιος αριθμός, μοναδικός για κάθε προϊόν.
3. **Description**: Η ονομασία του προϊόντος.
4. **Quantity**: Η ποσότητα προϊόντων που αντιστοιχεί σε κάθε συναλλαγή.
5. **InvoiceDate**: Η ημερομηνία και η ώρα έκδοσης του τιμολογίου.
6. **Price**: Η τιμή μιας μονάδας προϊόντος εκφρασμένη σε λίρες (£).
7. **Customer ID**: Ο μοναδικός αριθμός καταχώρισης πελάτη. Αποτελείται από 5 ψηφία.
8. **Country**: Το όνομα της χώρας στην οποία κατοικεί ο κάθε πελάτης.

Για τον καθαρισμό του συνόλου δεδομένων, καθώς και για την πραγματοποίηση της περιγραφικής και διερευνητικής ανάλυσης των δεδομένων (EDA), χρησιμοποιήθηκε το Jupyter Notebook και το εργαλείο οπτικοποίησης δεδομένων Microsoft Power BI.

A white background with text

Description automatically generatedΑρχικά ενοποιήθηκαν τα δύο excel sheets με την χρήση των εντολών pd.read\_excel() και pd.concat() της Python σε ένα σύνολο δεδομένων, το οποίο ονομάστηκε df\_raw, όπου με pd συμβολίζεται η βιβλιοθήκη pandas της Python.

A screenshot of a computer code

Description automatically generatedΠαρακάτω παρατίθενται πληροφορίες για το σύνολο των εγγραφών (1 067 371) και των μεταβλητών (8) του ενοποιημένου dataset, όπως προκύπτει από την εντολή df\_raw.info() της Python:

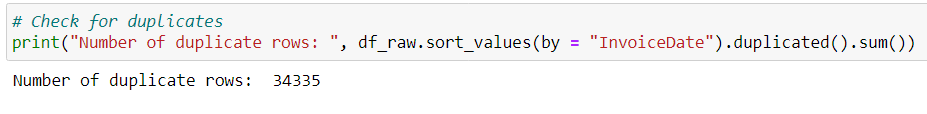
Με την εντολή df\_raw.head(10) βλέπουμε ενδεικτικά τις δέκα πρώτες συναλλαγές του dataset:

A screenshot of a computer screen

Description automatically generated

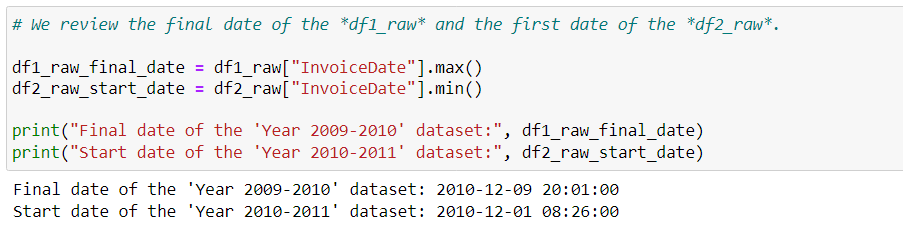
## 2.2 Διερευνητική ανάλυση δεδομένων και καθαρισμός δεδομένων

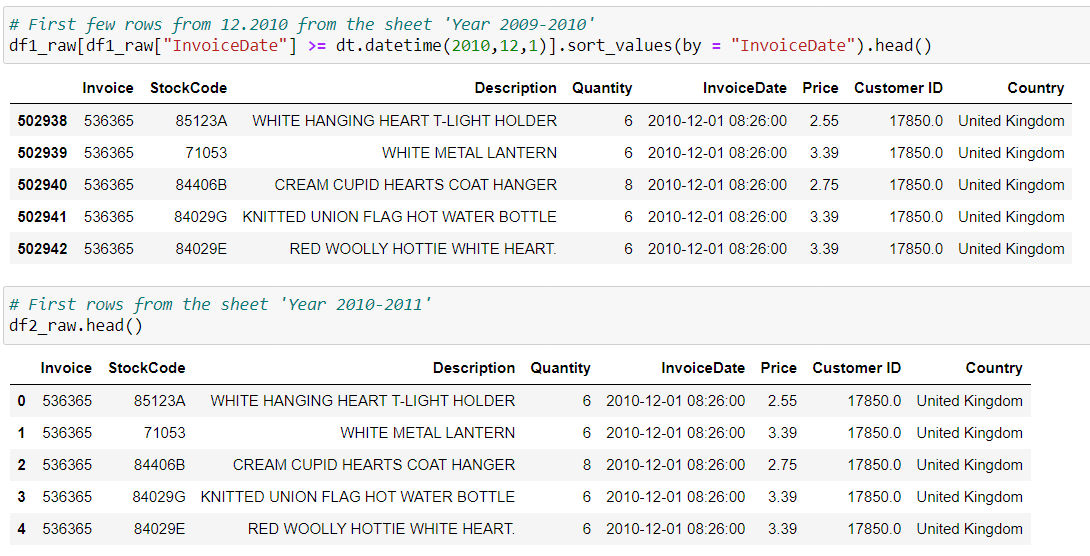
### 2.2.1 Αφαίρεση διπλών καταχωρίσεων

Το πρώτο βήμα είναι να ελεγχθεί αν υπάρχουν διπλές καταχωρίσεις. Με την εντολή df\_raw.sort\_values(by=“InvoiceDate”).duplicated().sum() παρατηρείται ότι ο αριθμός των διπλών καταχωρίσεων είναι 34 335.

A graph with blue bars

Description automatically generated with medium confidenceΣτο παρακάτω διάγραμμα απεικονίζεται η κατανομή των διπλών καταχωρίσεων όπου διαπιστώνεται ότι το μεγαλύτερο μέρος αυτών παρατηρείται κατά το μήνα Δεκέμβριο 2010. Το γεγονός αυτό πιθανότατα προκλήθηκε κατά την ενοποίηση των δύο excel sheets, τα οποία περιείχαν κοινές συναλλαγές που αφορούσαν στον συγκεκριμένο μήνα.

Για να εξακριβώσουμε την ισχύ της παραπάνω υπόθεσης, υπολογίζουμε την τελευταία ημερομηνία του df1\_raw (Year 2009-2010) και την πρώτη ημερομηνία του df2\_raw (Year 2010-2011). Όπως φαίνεται στην παρακάτω εικόνα, υπάρχουν εγγραφές που αφορούν το χρονικό διάστημα 01/12/2010 έως 09/12/2010 (9 ημέρες), οι οποίες συναντώνται και στους δύο πίνακες.

Ταυτόχρονα μπορούμε να διαπιστώσουμε το παραπάνω γεγονός και οπτικά, εκτυπώνοντας από τον πίνακα df1\_raw τις συναλλαγές που πραγματοποιήθηκαν κατά το συγκεκριμένο χρονικό διάστημα, και από τον πίνακα df2\_raw τις πρώτες εγγραφές του.

Μετά τις παραπάνω παρατηρήσεις, αφαιρούμε τις διπλές καταχωρίσεις μέσω της εντολής pd.concat(), η οποία έχει ως αποτέλεσμα τη δημιουργία του πίνακα **df**, ο οποίος περιέχει 1 044 848 συναλλαγές.

A screenshot of a computer code

Description automatically generated

Σε αυτό το σημείο ο αριθμός των διπλών καταχωρίσεων είναι 11 812, δηλαδή περίπου 66% λιγότερες σε σχέση με εκείνες του αρχικού dataset.

A graph with blue bars

Description automatically generated with medium confidence

Σε πραγματικές συνθήκες θα ζητούσαμε από τους υπεύθυνους συλλογής των δεδομένων του dataset να μας εξηγήσουν τί σημαίνουν οι διπλές αυτές καταχωρίσεις. Συγκεκριμένα, δεδομένου ότι η ποσότητα του κάθε προϊόντος σε κάθε μία από αυτές τις διπλές καταχωρίσεις είναι 1, μπορούμε να σχηματίσουμε την υπόθεση ότι για αυτές τις συναλλαγές το σύστημα αντί να αθροίσει τις ποσότητες του κάθε προϊόντος για κάθε παραγγελία, τις καταχώρισε ξεχωριστά σε ισόποσες εγγραφές. Εφόσον δεν υπάρχει κάποιο ανασταλτικό επιχείρημα που να καταρρίπτει αυτή την υπόθεση, θα συμπεριλάβουμε το υπόλοιπο 34% των διπλών καταχωρίσεων στην ανάλυση που ακολουθεί.

### 2.2.2 Ελλείπουσες τιμές

Χρησιμοποιώντας την εντολή df.isnull().sum() της Python παρατηρούμε τον αριθμό των ελλειπουσών τιμών για κάθε μεταβλητή. Οι μόνες μεταβλητές που έχουν missing values είναι η μεταβλητή Description (4 275) και η μεταβλητή Customer ID (235 287). Για να αποκτήσουμε εικόνα του σχετικού μεγέθους των ελλειπουσών τιμών της κάθε μεταβλητής χρησιμοποιούμε την εντολή missing/len(df) \* 100, όπου missing είναι το αποτέλεσμα της προηγούμενης εντολής που εκτελέσαμε.

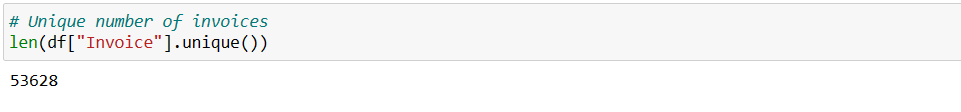
A screenshot of a computer

Description automatically generated

Προτού ξεκινήσουμε την διαμόρφωση του συνόλου δεδομένων για τις ανάγκες της παρούσας ανάλυσης θα πρέπει να διερευνηθούν οι μεταβλητές του και να αντληθούν χρήσιμες πληροφορίες. Στην επόμενη ενότητα ακολουθεί η διερευνητική ανάλυση του συνόλου δεδομένων.

### 2.2.3 Ανάλυση των μεταβλητών του συνόλου δεδομένων

#### *2.2.3.1 Στήλη Invoice*

Ο μοναδικός αριθμός των τιμολογίων είναι 53 628.

Μέσω της περιγραφής που έχει δώσει ο ίδιος ο εκδότης του dataset πληροφορούμαστε ότι ορισμένα από τα τιμολόγια αφορούν ακυρώσεις, και συγκεκριμένα όσα ξεκινούν με το γράμμα “C”. Θα επικεντρωθούμε αρχικά στις ακυρώσεις, και έπειτα στο πώς αυτές συνδέονται με τις μεταβλητές Quantity και Price. Για τον σκοπό αυτό δημιουργήσαμε ένα DataFrame[[3]](#footnote-3) με τις συναλλαγές που αφορούν ακυρώσεις τιμολογίων. Ο αριθμός τους είναι 19 165 και αντιστοιχεί A computer code with text

Description automatically generated with medium confidenceστο 1,8% του συνολικού αριθμού των συναλλαγών του dataset.



Από τις ακυρώσεις τιμολογίων μόνο μία συναλλαγή παρουσιάζει θετική τιμή στην στήλη Quantity, ενώ οι υπόλοιπες έχουν αρνητική τιμή. Η εν λόγω συναλλαγή έχει ελλείπουσα τιμή στην στήλη Customer ID, όπως φαίνεται και στην παρακάτω εικόνα.

A screenshot of a computer

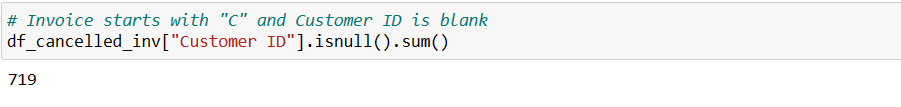
Description automatically generated

Εκτός από τα τιμολόγια που ξεκινούν από “C” υπάρχουν και κάποια που ξεκινούν από “A”. Τα τιμολόγια αυτά είναι μόνο 6 και έχουν missing value στη στήλη Customer ID. Τα 5 από αυτά έχουν αρνητική τιμή στην στήλη Price και μοναδιαία τιμή στην στήλη Quantity. Παρατηρώντας το περιεχόμενο της στήλης Description για κάθε ένα από αυτά τα 6 τιμολόγια (“Adjust bad debt”) ερμηνεύουμε πως πρόκειται για ενέργειες λογιστικής φύσεως και συνεπώς ότι δεν σχετίζονται με τις πωλήσεις του ηλεκτρονικού καταστήματος. Επιπλέον παρατηρούμε ότι υπάρχει μία συσχέτιση μεταξύ των τιμολογίων “A563185”, “A563186”, και “A563187”. Στην παρακάτω εικόνα παρατίθενται λεπτομέρειες για τα εν λόγω 6 τιμολόγια.

A screenshot of a computer

Description automatically generated

Από τα 19 165 ακυρωμένα τιμολόγια τα 719 (3%) έχουν missing value στην στήλη Customer ID.

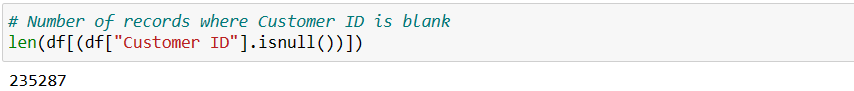


#### *2.2.3.2 Στήλη Quantity*

Η στήλη Quantity αναφέρεται είτε στην ποσότητα των προϊόντων που αγοράστηκαν σε κάθε συναλλαγή (Quantity > 0) είτε, όπως είδαμε και στην ανάλυση της στήλης Invoice, σε επιστροφές προϊόντων (Quantity < 0). Παρατηρούμε ότι δεν υπάρχουν ελλείπουσες τιμές στην στήλη Quantity και συνεπώς δεν χρειάζεται να προβούμε σε κάποια διαγραφή για την ανάλυση που θα ακολουθήσει.

Οι εγγραφές με αρνητική τιμή στην στήλη Quantity είναι συνολικά 22 557. Ο αριθμός αυτός ξεπερνάει τις εγγραφές των τιμολογίων που ξεκινούν από “C” και έχουν αρνητική τιμή (19164). Αυτό υποδηλώνει ότι η αρνητική τιμή στην ποσότητα δεν σημαίνει απαραίτητα μόνο την επιστροφή κάποιου προϊόντος, αλλά ενδέχεται να είναι συνδεδεμένη και με άλλα είδη συναλλαγών, όπως ακυρώσεις πωλήσεων ή ελαττωματικά προϊόντα.

#### *2.2.3.3 Στήλη Customer ID*

H στήλη Customer ID αναφέρεται στον μοναδικό αριθμό πελάτη (αναγνωριστικό πελάτη). Η στήλη αυτή περιέχει 235 287 missing values, δηλαδή περίπου το 22,5% των εγγραφών. Το ποσοστό είναι σημαντικά μεγάλο για να αγνοηθεί, και ανάλογα με την ανάλυση που θα πραγματοποιηθεί στα δεδομένα απαιτείται και διαφορετικός χειρισμός των missing values.

Σε πραγματικές συνθήκες θα ζητούσαμε από τους υπεύθυνους συλλογής των δεδομένων του dataset να μας εξηγήσουν γιατί οι συγκεκριμένες εγγραφές δεν έχουν Customer ID. Αφού κάτι τέτοιο δεν είναι εφικτό, μπορούμε να υποθέσουμε ότι οι συγκεκριμένοι πελάτες δεν πραγματοποίησαν ποτέ εγγραφή στο online κατάστημα και επομένως δεν τους αποδόθηκε αναγνωριστικό πελάτη.

Ο αριθμός των μοναδικών Customer ID του dataset που εξετάζουμε είναι5 943, στον οποίο αριθμό φτάνουμε με χρήση της παρακάτω εντολής της Python.

A close-up of a computer code

Description automatically generated

Ο αριθμός αυτός αντιπροσωπεύει 5 942 εγγεγραμμένους πελάτες συν όλους εκείνους οι οποίοι δεν πραγματοποίησαν ποτέ εγγραφή και των οποίων το Customer ID συμβολίζεται με τον κενό χαρακτήρα.

Οι ελλείπουσες τιμές της στήλης Customer ID θα αφαιρεθούν στην επόμενη ενότητα για τους σκοπούς της ανάλυσης RFM με στόχο την τμηματοποίηση των πελατών.

#### *2.2.3.4 Στήλη Country*

A screenshot of a computer

Description automatically generatedΟι συναλλαγές του dataset προέρχονται από 42 χώρες. Ενώ δεν παρατηρούνται missing values στην στήλη Country, υπάρχουν 756 εγγραφές στις οποίες η χώρα είναι “Unspecified”.

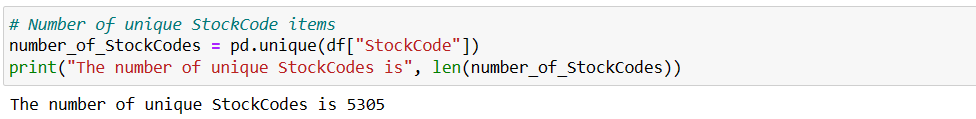
A graph with numbers and a bar

Description automatically generatedΤο μεγαλύτερο μερίδιο των συναλλαγών πραγματοποιήθηκε στο Ηνωμένο Βασίλειο (91.87%), ύστερα στην Ιρλανδία (1.69%) και στη Γερμανία (1.66%). Στο παρακάτω γράφημα παρουσιάζονται οι 10 χώρες με τον μεγαλύτερο αριθμό συναλλαγών.

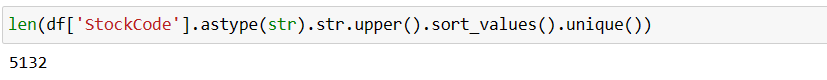
#### *2.2.3.5 Στήλη StockCode*

Η στήλη StockCode αναφέρεται στους κωδικούς προϊόντων που αγοράστηκαν. Η στήλη StockCode δεν περιέχει ελλείπουσες τιμές.

Κάτι το οποίο θα είχε αξία να ερευνηθεί είναι ποιοι κωδικοί έχουν τις περισσότερες πωλήσεις κατά το χρονικό διάστημα που εξετάζεται. Προς αυτόν τον σκοπό εκτελέστηκε η εντολή df.unique() της Python, σύμφωνα με την οποία οι μοναδικές τιμές της στήλης StockCode είναι 5 305.



Παράλληλα εκτελέσαμε την αντίστοιχη εντολή στο Power BI χρησιμοποιώντας την έκφραση DISTINCTCOUNTNOBLANKS της γλώσσας DAX, όπου διαπιστώσαμε ότι ο μοναδικός αριθμός προϊόντων διαφέρει (5 131 σε σχέση με 5 305). Είναι κρίσιμο να μάθουμε γιατί είναι διαφορετικά τα αποτελέσματα που προκύπτουν από τα δύο εργαλεία, και να φέρουμε τις τιμές σε συμφωνία, γιατί αλλιώς οι επερχόμενες αναλύσεις μέσω Python και Power BI θα αποκλίνουν και θα αποφέρουν πιθανώς διαφορετικά συμπεράσματα και συνεπώς αμφισημία.

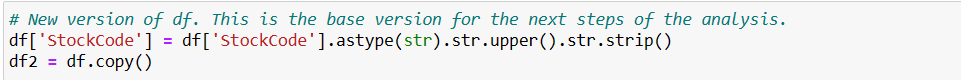
Δεδομένου ότι η Python αναφέρει περισσότερους μοναδικούς κωδικούς είναι λογικό να υποθέσουμε ότι η Python διάβασε κάποιες τιμές της στήλης ως μοναδικές, κάτι που δεν έκανε το Power BI, διαφορετικά δεν θα υπήρχε αυτή η ασυμφωνία. Με σκοπό να ερευνήσουμε το αν είναι σωστή αυτή η υποψία πρέπει αρχικά να φέρουμε τις τιμές της στήλης σε ένα κοινό επίπεδο ώστε να είναι συγκρίσιμες μεταξύ τους. Συγκεκριμένα, μέσω Python μετατρέπουμε το data type όλων των τιμών της στήλης StockCode σε τύπο string και έπειτα όλους τους χαρακτήρες των strings σε κεφαλαία γράμματα: με αυτό τον τρόπο όλες οι τιμές μπορούν να συγκριθούν μεταξύ τους. Σε αυτό το σημείο μετράμε εκ νέου τις μοναδικές τιμές της στήλης και το αποτέλεσμα που προκύπτει είναι 5 132, δηλαδή μία παραπάνω από αυτές του Power BI.

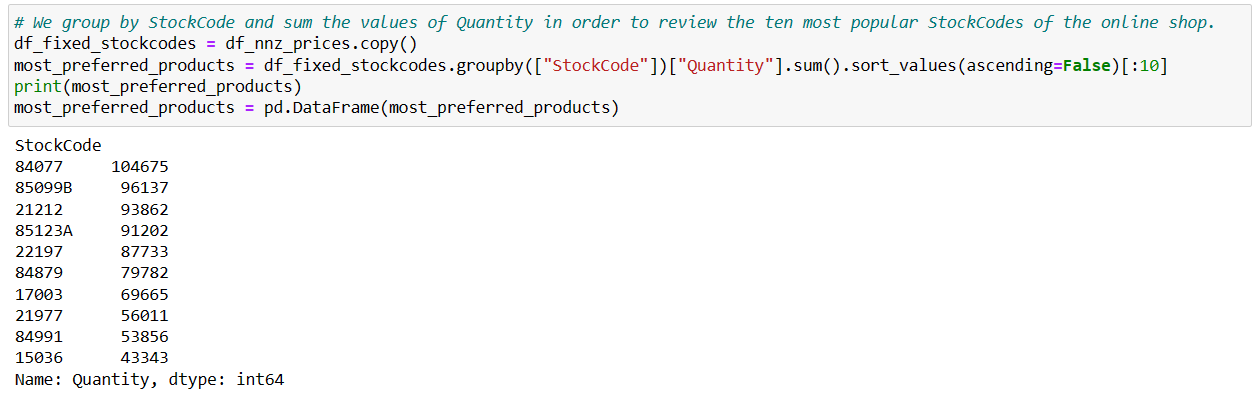
Για να εντοπίσουμε τη μία τιμή που εμφανίζεται ως διαφορά, παραθέσαμε τη λίστα των μοναδικών τιμών που προέκυψαν απότην Python και από το PowerBI σε δύο διαφορετικά αρχεία. Στη συνέχεια ταξινομήσαμε αλφαβητικά τις τιμές που περιέχονται στα δύο αρχεία και αντιπαραβάλλοντάς τες εντοπίσαμε ότι η εγγραφή "47503J" εμφανιζόταν με δύο τρόπους στα αποτελέσματα της Python: "47503J " και "47503J" (στην πρώτη περίπτωση μετά τον χαρακτήρα J υπάρχει ο κενός χαρακτήρας). Στη συνέχεια χρησιμοποιήσαμε την εντολή str.strip() για να αφαιρέσουμε το κενό που δημιουργήθηκε στην ονομασία και στο τέλος επιβεβαιώσαμε ότι οι μοναδικές τιμές της στήλης StockCode είναι 5 131 και για τα δύο εργαλεία.

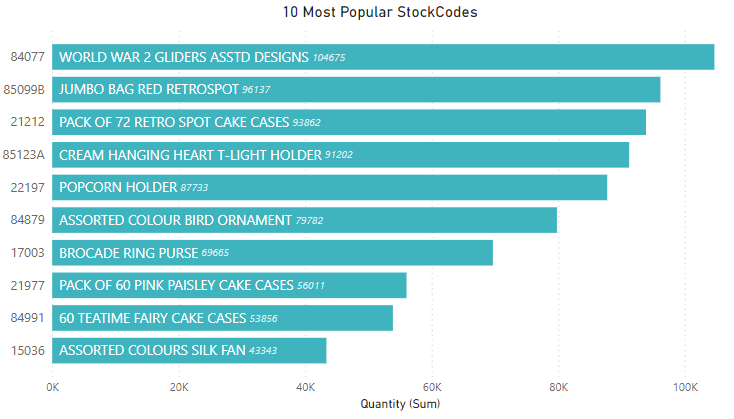
A screenshot of a computer code

Description automatically generated

Σημειώνουμε ότι δεν χρειάστηκε να κάνουμε όλα τα παραπάνω βήματα στο Power BI για να καταλήξουμε στον ίδιο αριθμό, καθώς η εντολή DISTINCTCOUNTNOBLANKS έκανε έναν αριθμό από υποθέσεις, οικεία βουλήσει, αγνοώντας για παράδειγμα τη διαφορά ανάμεσα στα πεζά και τα κεφαλαία γράμματα ή τους προπορευόμενους και λήγοντες κενούς χαρακτήρες στις ονομασίες των StockCode. Αν και στην συγκεκριμένη περίπτωση μπορεί να θεωρηθεί πλεονέκτημα, δεν σημαίνει ότι σε οποιαδήποτε ανάλυση θα καταλήγαμε στο ίδιο συμπέρασμα. Στο συγκεκριμένο dataset έχουμε να κάνουμε με μοναδικούς αριθμούς προϊόντων που είναι ξεκάθαροι και πεπερασμένοι, ενώ σε κάποιο άλλο dataset μια μικρή διαφορά στην ονομασία των προϊόντων θα μπορούσε να υποδηλώνει δύοαντικείμενα διακριτά. Κάθε dataset είναι ξεχωριστό και η εις βάθος κατανόηση των στοιχείων του είναι κρίσιμη προτού ο αναλυτής προβεί στον χειρισμό του.

Στην συνέχεια δημιουργήθηκε ένα νέο DataFrame, το df2, το οποίο περιλαμβάνει τις νέες τιμές της στήλης StockCode έπειτα από τις τροποποιήσεις που εφαρμόστηκαν πιο πάνω.

Κρίνουμε πως ο υπολογισμός των 10 πιο δημοφιλών προϊόντων του dataset θα είχε αξία για τον ιδιοκτήτη του καταστήματος και συνεπώς στην συνέχεια προβαίνουμε ακριβώς σε αυτή την ανάλυση. Αρχικά πρέπει να αποκλείσουμε τα προϊόντα που έχουν μηδενική τιμή στην στήλη Price, καθώς τα συγκεκριμένα είναι προϊόντα που δεν πωλήθηκαν αλλά προσφέρθηκαν ως δώρα ή προσφορές από το ηλεκτρονικό κατάστημα, και συνεπώς δεν υποδηλώνουν τις προτιμήσεις των πελατών. Το ίδιο θα κάναμε και στην περίπτωση που είχαμε ελλείπουσες τιμές στη στήλη Price, κάτι το οποίο δεν συμβαίνει στο παρόν dataset. Στην επόμενη εικόνα η συλλογή των προϊόντων με μη μηδενική τιμή συμβολίζεται με το DataFrame df\_nnz\_prices. Εκτελώντας τις υπόλοιπες εντολές της εικόνας φτάνουμε στη λίστα με τα δέκα πιο ευπώλητα προϊόντα, η οποία περιλαμβάνεται και αυτή στην ίδια εικόνα. Ακολουθώντας την ίδια ανάλυση μέσω Power BI καταλήγουμε στο ραβδόγραμμα της μεθεπόμενης εικόνας. Παρατηρούμε ότι οι δύο ξεχωριστές αναλύσεις συμπίπτουν.

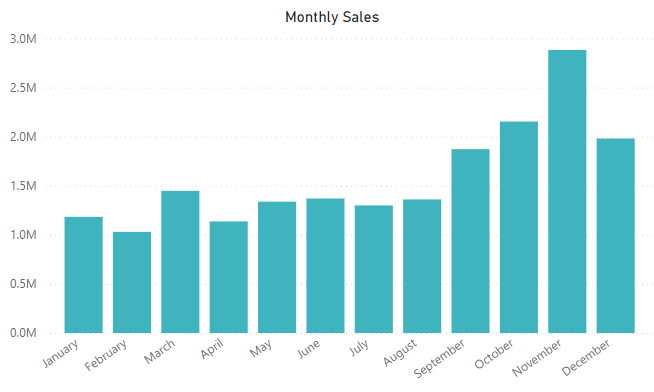


#### *2.2.3.6 Στήλη InvoiceDate*

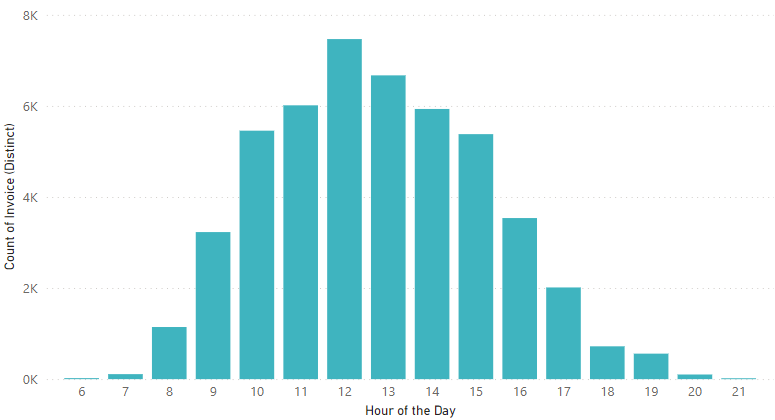
Η στήλη InvoiceDate αναφέρεται στην ημερομηνία και ώρα έκδοσης κάθε τιμολογίου. Όπως αναφέρθηκε και στην αρχική περιγραφή του dataset οι συναλλαγές εκτείνονται από 01/12/2009 έως 09/12/2011. Με σκοπό να ερευνηθεί ποιοι μήνες είχαν τις περισσότερες πωλήσεις δημιουργήθηκαν δύο νέες στήλες: η στήλη Month και η στήλη TotalAmount, οι οποίες προσαρτήθηκαν στο dataset. Η στήλη **Month** δημιουργήθηκε με την χρήση της βιβλιοθήκης datetime της Python, ενώ η στήλη **TotalAmount** προέκυψε από το γινόμενο των στηλών Price και Quantity.

A computer screen shot of text

Description automatically generated

Στο παρακάτω ραβδόγραμμα απεικονίζονται οι πωλήσεις ανά μήνα. Παρατηρήθηκε ότι οι μήνες Οκτώβριος, Νοέμβριος και Δεκέμβριος ήταν οι μήνες με τις περισσότερες πωλήσεις.

Επιπλέον δημιουργήθηκε η στήλη Hourώστενα ερευνηθεί ποιες ώρες κατά την διάρκεια της ημέρας έγιναν οι περισσότερες παραγγελίες. Σύμφωνα με το γράφημα που ακολουθεί ο μεγαλύτερος όγκος των παραγγελιών πραγματοποιείται ανάμεσα στις 10 π.μ. και στις 3 μ.μ.



Για τους σκοπούς της ανάλυσης που θα ακολουθήσει θα αφαιρέσουμε τα στοιχεία της ώρας από την στήλη InvoiceDate, της οποίας οι τιμές θα εμφανίζονται στο εξής στη μορφή «έτος-μήνας-ημέρα».

#### *2.2.3.7 Συνδυασμοί Customer ID και StockCode*

Όπως αναφέρθηκε στην περιγραφή του dataset, συχνά παρατηρείται ότιο ίδιος πελάτης αγόρασε ένα προϊόν σε διαφορετικές ποσότητες ενώ επίσης μπορεί να επέστρεψε έναν αριθμό τους. Θεωρητικά αναμένουμε πως ο αριθμός που προκύπτει από τη διαφορά των πωληθείσων ποσοτήτων και των επιστραμμένων είναι μεγαλύτερος ή ίσος του μηδενός (αφού κανείς δεν μπορεί να επιστρέψει μεγαλύτερο αριθμό από προϊόντα από όσα αγόρασε). Παρόλα αυτά στην πράξη γνωρίζουμε ότι αυτό είναι μια υπόθεση, της οποίας η αλήθεια θα πρέπει να εξεταστεί. Συνεπώς σκοπός μας είναι για κάθε πελάτη να συμψηφίσουμε τις ποσότητες που αγόρασε και τις ποσότητες που ακύρωσε και, εάν η τελική διαφορά ανά προϊόν βρεθεί με αρνητικό πρόσημο, να διαγραφούν από το dataset όλες οι εγγραφές που αναφέρονται στον συγκεκριμένο συνδυασμό πελάτη και κωδικού προϊόντος.

## 2.3 Ενέργειες μεταχείρισης αρχικού συνόλου δεδομένων εν συνόψει

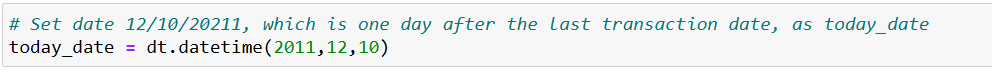
Στην παρακάτω λίστα παρατίθενται συνοπτικά οι αλλαγές που αναφέρθηκαν παραπάνω που είτε έχουν γίνει είτε θα γίνουν για τον καθαρισμό και την προετοιμασία ενός ορθού dataset για το επόμενο βήμα, το οποίο θα είναι η σύσταση του dataset που θα χρησιμοποιηθεί για την ανάλυση RFM:

* Αναφορικά με τα πρωταρχικά δύο υποσύνολα δεδομένων: διαγραφή των εγγραφών των τελευταίων εννέα ημερών από το πρώτο υποσύνολο, και έπειτα συγχώνευση των δύο υποσυνόλων σε ένα ενιαίο
* Μετατροπή των τιμών της στήλης Invoice σε τύπο string
* Μετατροπή των τιμών της στήλης StockCode σε τύπο string, κεφαλαία, και αφαίρεση των περιμετρικών κενών χαρακτήρων
* Διαγραφή ελλειπουσών τιμών από την στήλη Customer ID
* Η διαγραφή των ελλειπουσών τιμών από την στήλη Customer ID ταυτόχρονα συμπαρασύρει και όλες τις εγγραφές οι οποίες έχουν αρνητική τιμή στην στήλη Price, τις οποίες κατά συνέπεια δεν χρειάζεται να διαγράψουμε
* Για κάθε μοναδικό συνδυασμό Customer ID και StockCode άθροιση των ποσοτήτων των επιμέρους τιμολογίων στα οποία αναφέρεται ο κάθε κωδικός προϊόντος StockCode, και μετέπειτα διαγραφή όλων εκείνων των εγγραφών για τις οποίες το προκύπτον άθροισμα είναι αρνητικό.
* Αφαίρεση της ώρας από την στήλη InvoiceDate

Τα παραπάνω βήματα παρουσιάζονται αναλυτικά στο notebook με την ονομασία 01.cleandata[[4]](#footnote-4) το οποίο είναι διαθέσιμο στην ιστοσελίδα GitHub.

# Η μέθοδος RFM

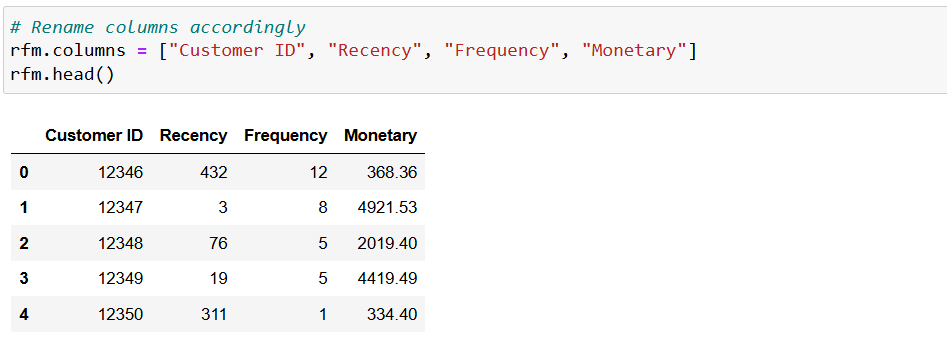
## 3.1 Υπολογισμός των μετρικών Recency, Frequency, Monetary

Για την τμηματοποίηση των πελατών σε ομάδες με όμοια χαρακτηριστικά, διαμορφώνεται το σύνολο δεδομένων RFM. Για τον υπολογισμό της μετρικής Recency θα πρέπει αρχικά να σημειωθεί πότε πραγματοποιήθηκε η τελευταία συναλλαγή του dataset (09/12/2011). Ως ημερομηνία αναφοράς ορίζεται η αμέσως επόμενη ημέρα, δηλαδή η ημερομηνία 10/12/2011. Η μεταβλητή Recency προκύπτει από τον αριθμό των ημερών που μεσολαβούν ανάμεσα στην ημερομηνία της πιο πρόσφατης αγοράς που πραγματοποίησε ο κάθε πελάτης και στην ημερομηνία αναφοράς.

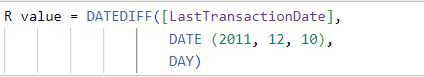
Η μετρική Frequency ορίζεται ως ο αριθμός των μοναδικών τιμών της στήλης Invoice για κάθε πελάτη. Επιπλέον, για τον υπολογισμό της μετρικής Monetary, είναι απαραίτητη η δημιουργία της στήλης TotalValue η οποία προκύπτει από το γινόμενο των στηλών Price και Quantity για κάθε συναλλαγή. Η μετρική Monetary προκύπτει από το άθροισμα των τιμών της στήλης TotalValue για κάθε πελάτη. Έχοντας ως βάση τις στήλες InvoiceDate, Invoice και TotalValue και με τον συνδυασμό των εντολών της Python groupby() και agg(), υπολογίζουμε τις μετρικές Recency, Frequency και Monetary για κάθε πελάτη.

A computer code with colorful text

Description automatically generated with medium confidence



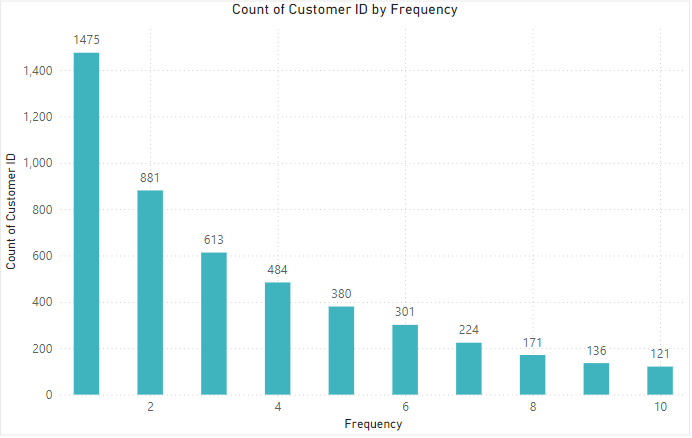
Τα παραπάνω βήματα υλοποιήθηκαν αντίστοιχα και στο Power BI όπου με την χρήση των εντολών DATEDIFF, DISTINCTCOUNT και SUM της γλώσσας DAX υπολογίστηκαν ως measures οι τρεις μετρικές για κάθε πελάτη.



## 3.2 Υπολογισμός των RFM scores

Ο υπολογισμός των παραπάνω μετρικών για κάθε πελάτη δίνει την δυνατότητα στην επιχείρηση να έχει πρόσβαση σε πολύ χρήσιμες πληροφορίες, ο όγκος των οποίων παρόλα αυτά τις καθιστά δύσκολα διαχειρίσιμες. Για να αντιμετωπιστεί αυτό το πρόβλημα, σύμφωνα με την RFM ανάλυση, για κάθε μία από τις μετρικές Recency, Frequency και Monetary, κάθε καταναλωτής λαμβάνει μία βαθμολογία (score) από τον αριθμό 1 έως τον αριθμό 5 (το 5 αντιπροσωπεύει την καλύτερη βαθμολογία). Για να γίνει αυτό, το σύνολο των καταναλωτών χωρίζεται σε 5 ισοπληθείς κατηγορίες. Το 20% των πελατών με μεγαλύτερη συχνότητα συναλλαγών λαμβάνει score 5 για την στήλη Frequency, το 20% των πελατών με τις αμέσως μεγαλύτερες συχνότητες λαμβάνει score 4, και η λογική αυτή επαναλαμβάνεται μέχρι το score 1. Η ίδια διαδικασία επαναλαμβάνεται για τη στήλη Monetary, ενώ για τη στήλη Recency μεγαλύτερα score λαμβάνουν μικρότερες τιμές εγγύτητας αγοράς προς την ημερομηνία αναφοράς. Στην συνέχεια για κάθε πελάτη οι βαθμολογίες που έχει λάβει σε κάθε κατηγορία συνενώνονται και δημιουργούν το τριψήφιο RFM score του πελάτη.

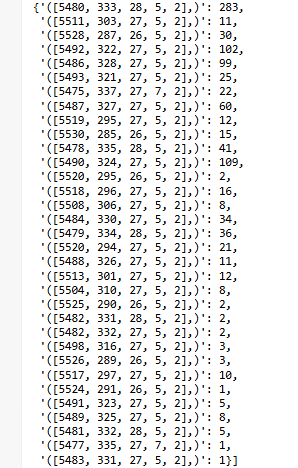
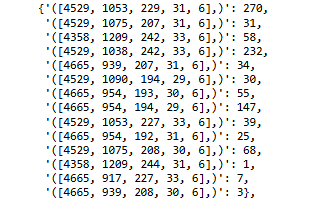
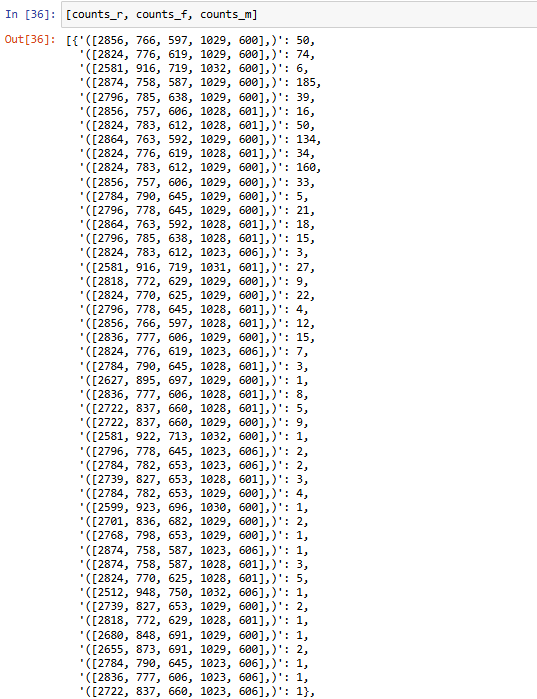
Ακολουθώντας τα παραπάνω βήματα για το παρόν dataset παρατηρήθηκε ότι για την μετρική Frequency ήταν αδύνατο να χωριστούν οι πελάτες σε 5 ισοπληθείς κατηγορίες: 1475 πελάτες έκαναν μόνο μία αγορά, ο αριθμός των οποίων αποτελεί το 25,22% των συνολικών πελατών (το οποίο είναι μεγαλύτερο όχι μόνο από το 20% που υποδεικνύεται από την ανάλυση αλλά μεγαλύτερο ακόμα και από το 25% αν η στήλη είχε χωριστεί σε 4 ισοπληθείς ομάδες). Στην προκειμένη περίπτωση θα μπορούσαμε να χωρίσουμε τη στήλη Frequency σε 3 ισοπληθείς ομάδες αλλά αυτό θα σήμαινε ότι θα έπρεπε να χωρίσουμε και τις άλλες δύο στήλες στα 3. Ο λόγος για τον οποίο θα έπρεπε να γίνει αυτό είναι ότι μετά την παραγωγή των RFM scores και της συνένωσής τους ακολουθεί το βήμα της συσταδοποίησης των πελατών στον τρισδιάστατο RFM χώρο, όπου απαιτείται οι 3 ορθογώνιοι άξονες να έχουν τις ίδιες μονάδες μέτρησης, καθώς οι περισσότεροι αλγόριθμοι συσταδοποίησης χρησιμοποιούν ως μετρική απόστασης την Ευκλείδεια απόσταση. Κρίνουμε ότι η διακριτοποίηση των πελατών με 33=27 ετικέτες δημιουργεί μια πιο αδρή εικόνα των καταναλωτικών συνηθειών των πελατών του καταστήματος σε σχέση με την διακριτοποίησή τους με 53=125 ετικέτες.

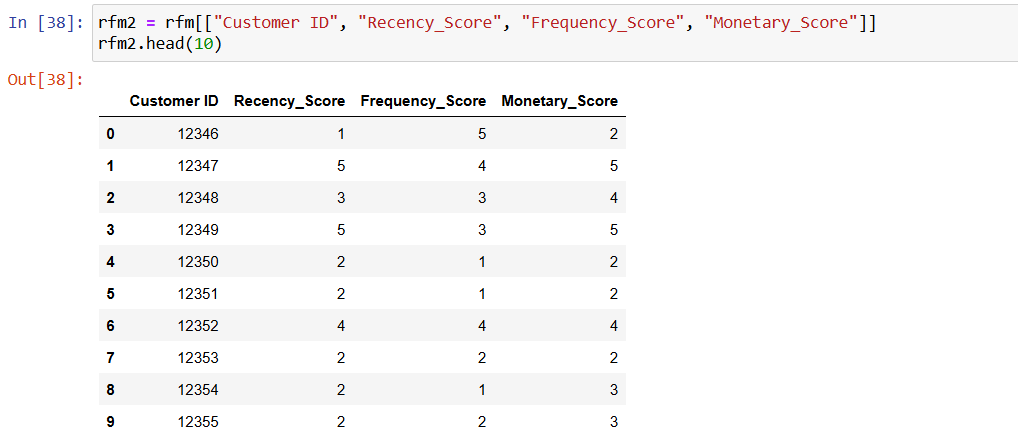


Δεδομένου λοιπόν ότι δεν μπορούμε να χρησιμοποιήσουμε την μέθοδο ομαδοποίησης που ακολουθείται συνήθως στην RFM ανάλυση κρίναμε κατάλληλη την χρήση του αλγορίθμου συσταδοποίησης K-means για την ομαδοποίηση κάθε μίας από τις τρεις μετρικές, Recency, Frequency και Monetary. Το αποτέλεσμα της συσταδοποίησης θα είναι πανομοιότυπο με το αποτέλεσμα της συνηθισμένης μεθόδου απόδοσης score όταν για k χρησιμοποιηθεί ο αριθμός 5.

To Power BI δεν δίνει την δυνατότητα εκτέλεσης συσταδοποίησης μέσω Κ-means ώστε να διεξάγουμε ανεξάρτητη ανάλυση, όμως δίνει την δυνατότητα εκτέλεσης κώδικα Python, στην οποία γλώσσα η συσταδοποίηση μέσω K-means εκφράζεται με δύο εντολές κώδικα. Δεδομένου τούτου θα μπορούσαμε να διεξάγουμε clustering ξεχωριστά στην Python και στο Power BI και να καταλήξουμε ανεξάρτητα στα RFM scores. Το πρόβλημα εδώ είναι ότι ο K-means είναι πιθανοτικός, και συνεπώς όσες φορές εκτελείται εξάγονται διαφορετικά αποτελέσματα, το οποίο σημαίνει ότι δεν είναι δυνατό να καταλήξουμε στο ίδιο αποτέλεσμα σε ένα βήμα και από τα δύο εργαλεία. Για αυτόν τον λόγο, και αφού το Power BI μπορεί να εκτελέσει κώδικα Python, μπορούμε να επιλέξουμε οποιοδήποτε από τα δυο εργαλεία για να εξάγουμε τα RFM scores, και στην συνέχεια να εισάγουμε το αποτέλεσμα του ενός εργαλείου προς περαιτέρω ανάλυση στο άλλο. Για αυτή την διαδικασία επιλέξαμε την Python με γνώμονα την ταχύτητα εξαγωγής των αποτελεσμάτων της.

Επιπρόσθετα, το αποτέλεσμα του clustering μέσω K-means είναι πιθανοτικό, το οποίο σημαίνει ότι δεν είναι απαραίτητο ότι κάθε εκτέλεσή του παράγει το ίδιο αποτέλεσμα. Για αυτό τον λόγο, διεξάγουμε clustering μέσω K-means 1000 φορές, ώστε να δεχθούμε ως ορθό αποτέλεσμα των RFM scores το πιο συχνό αποτέλεσμα συσταδοποίησης. Εισάγοντας αυτό το αποτέλεσμα στο Power BI τότε μπορούμε να προχωρήσουμε και στα δύο εργαλεία στην περαιτέρω ανάλυση με την λογική RFM, ήτοι στην ομαδοποίηση των πελατών στον τρισδιάστατο RFM χώρο. Παρατηρήστε ότι θα μπορούσαμε να κάνουμε ακριβώς την ίδια ανάλυση εκτελώντας κώδικα Python μέσω Power BI και να εισάγουμε το αποτέλεσμα πίσω στο jupyter notebook. Αυτός ο τρόπος όμως θα ήταν πιο χρονοβόρος και θα απαιτούσε από αυτόν που χρησιμοποιούσε αποκλειστικά το Power BI ως εργαλείο γνώσεις Python.





1. <https://www.kaggle.com/datasets/kabilan45/online-retail-ii-dataset> [↑](#footnote-ref-1)
2. <https://archive.ics.uci.edu/dataset/502/online+retail+ii> [↑](#footnote-ref-2)
3. Η δομή δεδομένων DataFrame είναι η βασική δομή δεδομένων της βιβλιοθήκης pandas. Η βιβλιοθήκη pandas είναι το βασικό μέσο επεξεργασίας δεδομένων σε μορφή πινάκων της Python. [↑](#footnote-ref-3)
4. <https://github.com/ArgyroMp/msc-thesis/blob/master/Notebooks/01.cleandata.ipynb> [↑](#footnote-ref-4)