Γεώργιος Παπουτσάκης 8200137

9/1/2025

Περιεχόμενα

Εισαγωγή

Το συγκεκριμένο αρχείο περιλαμβάνει την αναλυτική περιγραφή της διαδικασίας που ακολουθήθηκε για την υλοποίηση της εργασίας στο μάθημα της Επιχειρηματικής Ευφυΐας και Ανάλυσης Μεγάλων Δεδομένων. Η εργασία αφορά την ανάλυση ενός μεγάλου Dataset, την δημιουργία μίας Αποθήκης Δεδομένων, την Οπτικοποίηση τους και δύο μοντέλα Data mining.

Επιλογή Dataset

Για την εύρεση του dataset χρησιμοποιήθηκε η ιστοσελίδα https://www.kaggle.com/ και επιλέχθηκε το παρακάτω dataset:

https://www.kaggle.com/datasets/ahmedabbas757/coffee-sales

Αναφέρεται σε συναλλαγές που πραγματοποιήθηκαν σε 3 καταστήματα καφέ την περίοδο 1/1/2023 – 30/6/2023. Αποτελείται από 149.116 γραμμές και 11 στήλες ('transaction_id', 'transaction_date', 'transaction_time', 'transaction_qty', 'store_id', 'store_location', 'product_id', 'unit_price', 'product_category', 'product_type', 'product_detail').

Dataset Cleaning

Για τον καθαρισμό του dataset χρησιμοποιήθηκε python σε jupyter notebook. Πρώτα από όλα ελέγχθηκε αν υπήρχαν διπλές εγγραφές και Null τιμές. Δεν υπήρξαν τέτοιες περιπτώσεις στο dataset.

```
[4]: df[df.duplicated()]
```

[4]:

transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	product_type	product_detail

```
[5]: df.isna().sum()
```

```
[5]: transaction_id      0
     transaction_date    0
     transaction_time    0
     transaction_qty     0
     store_id            0
     store_location      0
     product_id          0
     unit_price          0
     product_category    0
     product_type        0
     product_detail      0
     dtype: int64
```

Το επόμενο βήμα του καθαρισμού αφορούσε τη διόρθωση των transaction_id. Παρατηρήθηκε ότι κάθε γραμμή του πίνακα είχε διαφορετικό transaction_id, ακόμη και όταν αφορούσε την ίδια συναλλαγή. Επομένως, αντικαταστάθηκαν τα transaction_id για τις συναλλαγές που πραγματοποιήθηκαν την ίδια ημέρα και ώρα στο ίδιο κατάστημα, αποδίδοντάς τους ένα νέο, κοινό transaction_id.

- If two or more transactions happen at the exact same **time**, on the same **day**, in the same **store**, then they are considered part of a bigger transaction that includes different kinds of products.

```
[8]: df['transaction_datetime'] = df.apply(
         lambda row: pd.Timestamp.combine(row['transaction_date'], row['transaction_time']), axis=1
     )

[9]: df['actual_transaction_id'] = df.groupby(['transaction_datetime', 'store_id']).ngroup()

[10]: df['transaction_id'] = df['actual_transaction_id']
      df.drop(columns=['actual_transaction_id'], inplace=True)

[11]: df.head(6)
```

[11]:

	transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	product_type	product_detail	t
0	0	2023-01-01	07:06:11	2	5	Lower Manhattan	32	3.0	Coffee	Gourmet brewed coffee	Ethiopia Rg	
1	1	2023-01-01	07:08:56	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea	Spicy Eye Opener Chai Lg	
2	2	2023-01-01	07:14:04	2	5	Lower Manhattan	59	4.5	Drinking Chocolate	Hot chocolate	Dark chocolate Lg	
3	3	2023-01-01	07:20:24	1	5	Lower Manhattan	22	2.0	Coffee	Drip coffee	Our Old Time Diner Blend Sm	
4	4	2023-01-01	07:22:41	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea	Spicy Eye Opener Chai Lg	
5	4	2023-01-01	07:22:41	1	5	Lower Manhattan	77	3.0	Bakery	Scone	Oatmeal Scone	

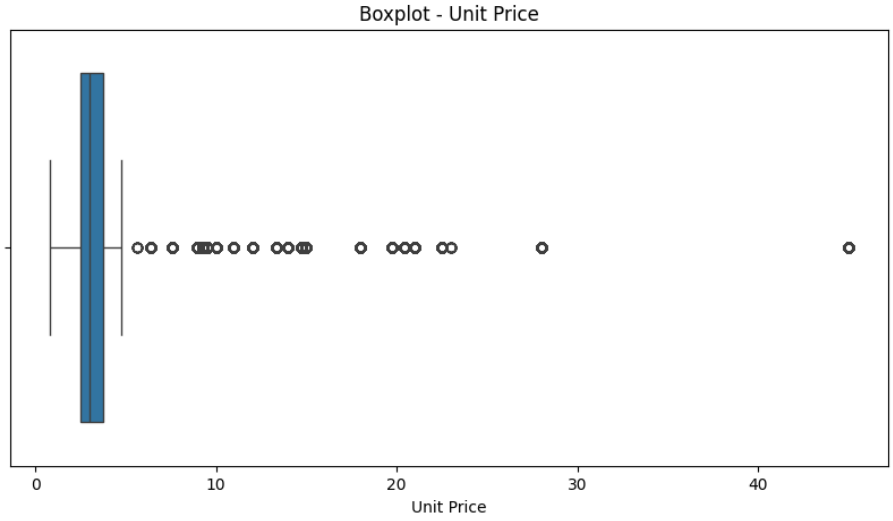
Επίσης, δημιουργήθηκε μία νέα στήλη index ώστε να υπάρχει ένα μοναδικό στοιχείο για κάθε εγγραφή του αρχείου. Δεν είναι απαραίτητο για την python αλλά θα χρειαστεί ως πρωτεύον κλειδί όταν φτιαχτούν οι πίνακες της βάσης.

```
[34]: df = df.reset_index()
      df
```

[34]:

	index	transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	...	day_name	isW
0	0	0	2023-01-01	07:06:11	2	5	Lower Manhattan	32	3.00	Coffee	...	Sunday	
1	1	1	2023-01-01	07:08:56	2	5	Lower Manhattan	57	3.10	Tea	...	Sunday	
2	2	2	2023-01-01	07:14:04	2	5	Lower Manhattan	59	4.50	Drinking Chocolate	...	Sunday	
3	3	3	2023-01-01	07:20:24	1	5	Lower Manhattan	22	2.00	Coffee	...	Sunday	
4	4	4	2023-01-01	07:22:41	2	5	Lower Manhattan	57	3.10	Tea	...	Sunday	
...	...	...	...	...	...	...	...	...	...	...	...	...	

Ο επόμενος έλεγχος αφορούσε τα outliers. Αρχικά, φάνηκε ότι υπάρχουν κάποιες ακραίες τιμές στις συναλλαγές. Ωστόσο, εξετάζοντας πιο προσεκτικά αυτές τις συναλλαγές, παρατηρήθηκε ότι δεν αφορούν συνηθισμένα προϊόντα καφέ, αλλά κάποια premium. Επομένως, αποφασίστηκε να διατηρηθούν χωρίς τροποποίηση, καθώς οι τιμές τους κρίθηκαν λογικές για τα συγκεκριμένα προϊόντα.



[24]:
```
df[df['unit_price']>30].head()
```

[24]:

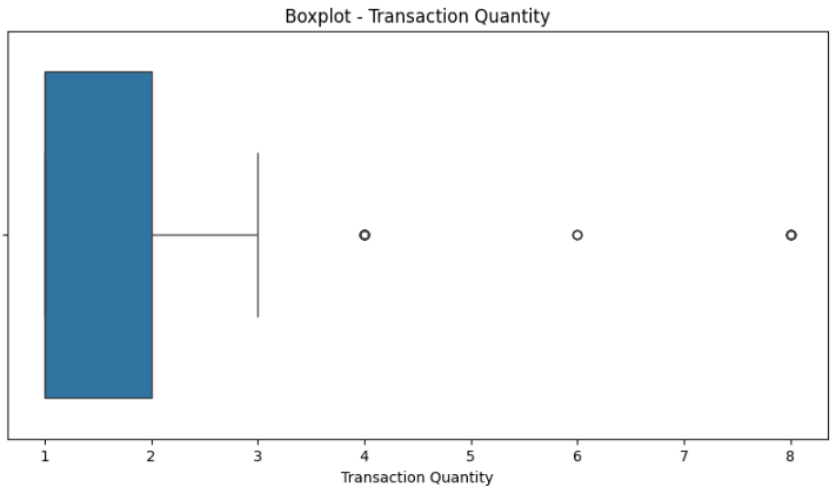
	transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	product_type	product_detail
5182	4274	2023-01-10	09:20:43	1	5	Lower Manhattan	8	45.0	Coffee beans	Premium Beans	Civet Cat
5874	4791	2023-01-11	11:24:31	1	8	Hell's Kitchen	8	45.0	Coffee beans	Premium Beans	Civet Cat
5939	4843	2023-01-11	13:53:09	1	5	Lower Manhattan	8	45.0	Coffee beans	Premium Beans	Civet Cat
7616	6070	2023-01-14	10:57:38	1	8	Hell's Kitchen	8	45.0	Coffee beans	Premium Beans	Civet Cat
8443	6668	2023-01-15	18:18:51	1	3	Astoria	8	45.0	Coffee beans	Premium Beans	Civet Cat

[25]:
```
df[df['unit_price']==28].head()
```

[25]:

	transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	product_type	product_detail
3296	2832	2023-01-07	07:45:15	1	5	Lower Manhattan	9	28.0	Coffee beans	Organic Beans	Organic Decaf Blend
3302	2836	2023-01-07	07:50:42	1	5	Lower Manhattan	9	28.0	Coffee beans	Organic Beans	Organic Decaf Blend

Σχετικά με την ποσότητα, παρατηρήθηκαν ορισμένες τιμές που ξεχωρίζουν, όμως δεν αποκλίνουν από ρεαλιστικά ανθρώπινα δεδομένα. Επομένως, αποφασίστηκε να διατηρηθούν χωρίς τροποποίηση.



Δεν υπήρξαν συναλλαγές με μηδενικές ή αρνητικές τιμές στην τιμή ή την ποσότητα. Γενικότερα το dataset ήταν πολύ καθαρό εξαιρεθούν οι τιμές των transaction_id.

Στη συνέχεια, υπολογίστηκαν νέες τιμές και προστέθηκαν ως επιπλέον στήλες, κυρίως με βάση την ώρα και την ημέρα της συναλλαγής, καθώς και τις γεωγραφικές συντεταγμένες των καταστημάτων, αφού αρχικά ήταν γνωστή μόνο η περιοχή.

Οι νέες στήλες που προστέθηκαν είναι: 'index', 'time_of_day', 'month_name', 'day_name', 'isWeekend', 'day', 'month', 'year', 'hour', 'minutes', 'seconds', 'latitude' και 'longitude'.

Ακολουθεί ενδεικτικά ο τρόπος υπολογισμού ορισμένων από αυτές.

```
[26]: df['month_name'] = df['transaction_date'].dt.month_name()

[27]: df['day_name'] = df['transaction_date'].dt.day_name()

[28]: df['isWeekend'] = df['day_name'].isin(['Saturday', 'Sunday'])

[31]: df['day'] = df['transaction_datetime'].dt.day
      df['month'] = df['transaction_datetime'].dt.month
      df['year'] = df['transaction_datetime'].dt.year

      df['hour'] = df['transaction_datetime'].dt.hour
      df['minutes'] = df['transaction_datetime'].dt.minute
      df['seconds'] = df['transaction_datetime'].dt.second

[75]: coordinates = {
          'Hell\'s Kitchen': {'latitude': 40.76417, 'longitude': -73.99222},
          'Astoria': {'latitude': 40.764357, 'longitude': -73.923462},
          'Lower Manhattan': {'latitude': 40.7128, 'longitude': -74.0060}
      }

      def get_latitude(location):
          return coordinates.get(location, {}).get('latitude', None)

      def get_longitude(location):
          return coordinates.get(location, {}).get('longitude', None)

      df['latitude'] = df['store_location'].apply(get_latitude)
      df['longitude'] = df['store_location'].apply(get_longitude)
```

Τελευταίο βήμα στην python ήταν η αποθήκευση του ανανεωμένου dataframe ως νέο CSV αρχείο.

```
[76]: df.to_csv('Cleaned Coffee Shop Sales.csv', index=False)
```

Για τα επόμενα βήματα της εργασίας χρειάστηκε να εγκατασταθούν και χρησιμοποιηθούν τα εργαλεία:

- SQL Server
- SQL Server Management Studio
- SQL Server Integration Services
- SQL Server Analysis Services
- Visual Studio
- Power Bi Desktop

Δημιουργία Βάσης

Για τη δημιουργία της βάσης δεδομένων χρησιμοποιήθηκε τοπικά το SQL Server Management Studio, με σκοπό τη διαμόρφωση ενός Star Schema για τα δεδομένα. Ο Fact Table θα αφορά τις συναλλαγές, ενώ οι Πίνακες Διάστασης θα σχετίζονται με το κατάστημα, το προϊόν, την ημερομηνία και την ώρα.

Αφού καθορίστηκαν οι απαραίτητοι πίνακες για τη δομή του Star Schema, προχώρησε η δημιουργία τους μέσω SQL queries στο Management Studio. Σε κάθε πίνακα προστέθηκε

ένας επιπλέον index, ο οποίος αποτρέπει το πρόγραμμα από το να σταματά λόγω σφάλματος σε περίπτωση εισαγωγής μιας εγγραφής με ήδη υπάρχον πρωτεύον κλειδί. Φυσικά όμως δεν θα προστίθεται η εγγραφή, απλά θα αποφεύγετε το crash (θα φανεί χρήσιμο εισαγωγή των δεδομένων-ETL).

Create Dimension T...UF1FHLP\User (58))

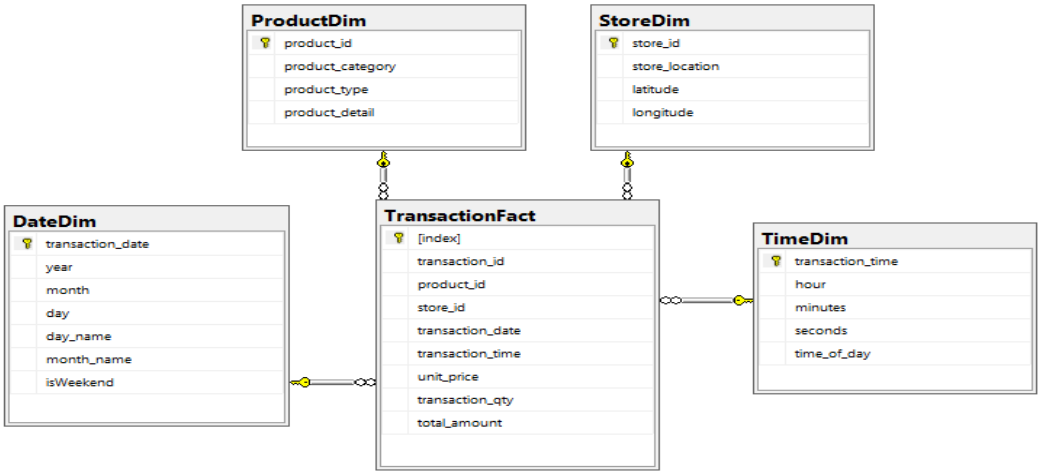
```sql
-- Product Dimension
CREATE TABLE dbo.ProductDim (
    product_id INT PRIMARY KEY,
    product_category NVARCHAR(100),
    product_type NVARCHAR(100),
    product_detail NVARCHAR(200),
);

CREATE UNIQUE INDEX idx_product_id
ON dbo.ProductDim(product_id)
WITH (IGNORE_DUP_KEY = ON);


-- Store Dimension
CREATE TABLE dbo.StoreDim (
    store_id INT PRIMARY KEY,
    store_location NVARCHAR(200),
    latitude FLOAT,
    longitude FLOAT
);

CREATE UNIQUE INDEX idx_store_id
ON dbo.StoreDim(store_id)
WITH (IGNORE_DUP_KEY = ON);

-- Date Dimension
CREATE TABLE dbo.DateDim (
    transaction_date DATE PRIMARY KEY,
```

Στην συνέχεια δημιουργείται και ο Fact Table.

Create and insert t...-UF1FHLP\User (53))

```sql
CREATE TABLE dbo.TransactionFact (
    [index] INT PRIMARY KEY,
    transaction_id INT,
    product_id INT NOT NULL,
    store_id INT NOT NULL,
    transaction_date DATE NOT NULL,
    transaction_time TIME NOT NULL,
    unit_price FLOAT,
    transaction_qty INT,
    total_amount FLOAT,
    CONSTRAINT FK_Product FOREIGN KEY (product_id) REFERENCES dbo.ProductDim (product_id),
    CONSTRAINT FK_Store FOREIGN KEY (store_id) REFERENCES dbo.StoreDim (store_id),
    CONSTRAINT FK_Date FOREIGN KEY (transaction_date) REFERENCES dbo.DateDim (transaction_date),
    CONSTRAINT FK_Time FOREIGN KEY (transaction_time) REFERENCES dbo.TimeDim (transaction_time)
);
```

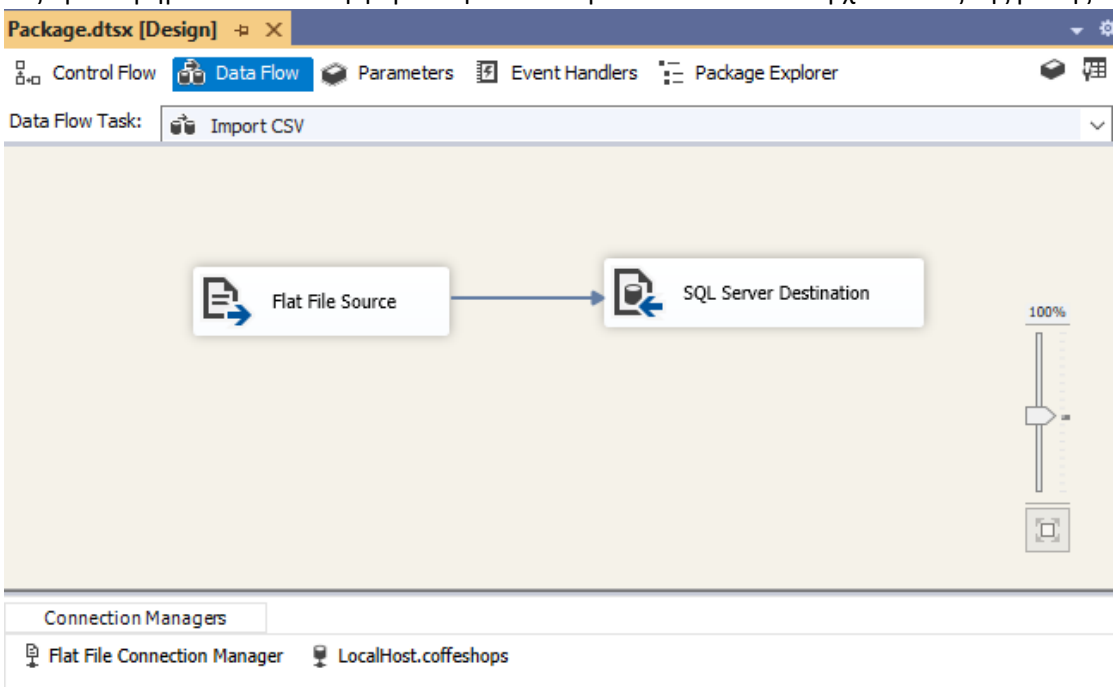
Η τιμή του προϊόντος βρίσκεται εντός του Fact Table γιατί όπως παρατηρήθηκε ένα προϊόν μπορεί να έχει διαφορετική τιμή ανάλογα την ημερομηνία που πωλήθηκε.



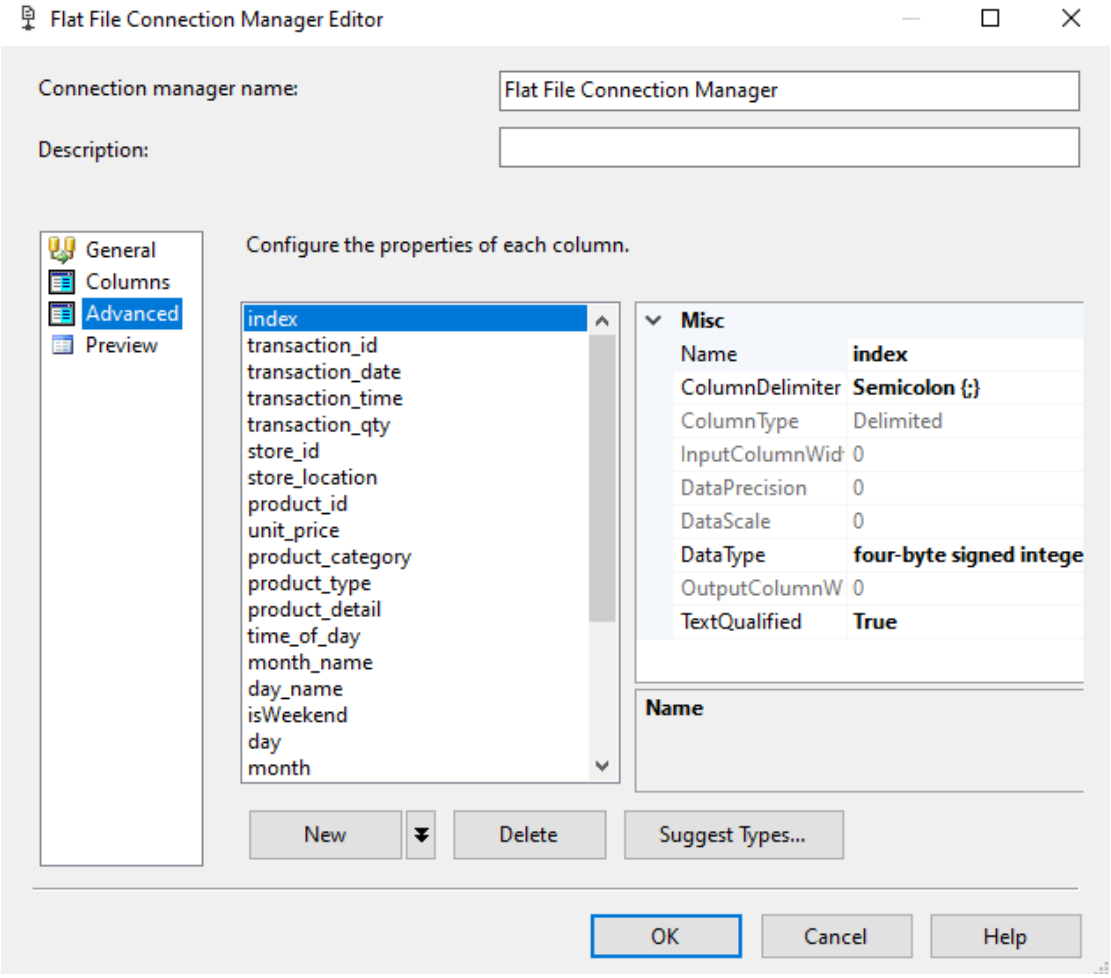
ETL

Για το ETL χρησιμοποιήθηκε το SQL Server Integration Services (SSIS) μέσω του Visual Studio.

Ως πρώτο βήμα απαιτείται η φόρτωση των δεδομένων από το CSV αρχείο εντός της βάσης.

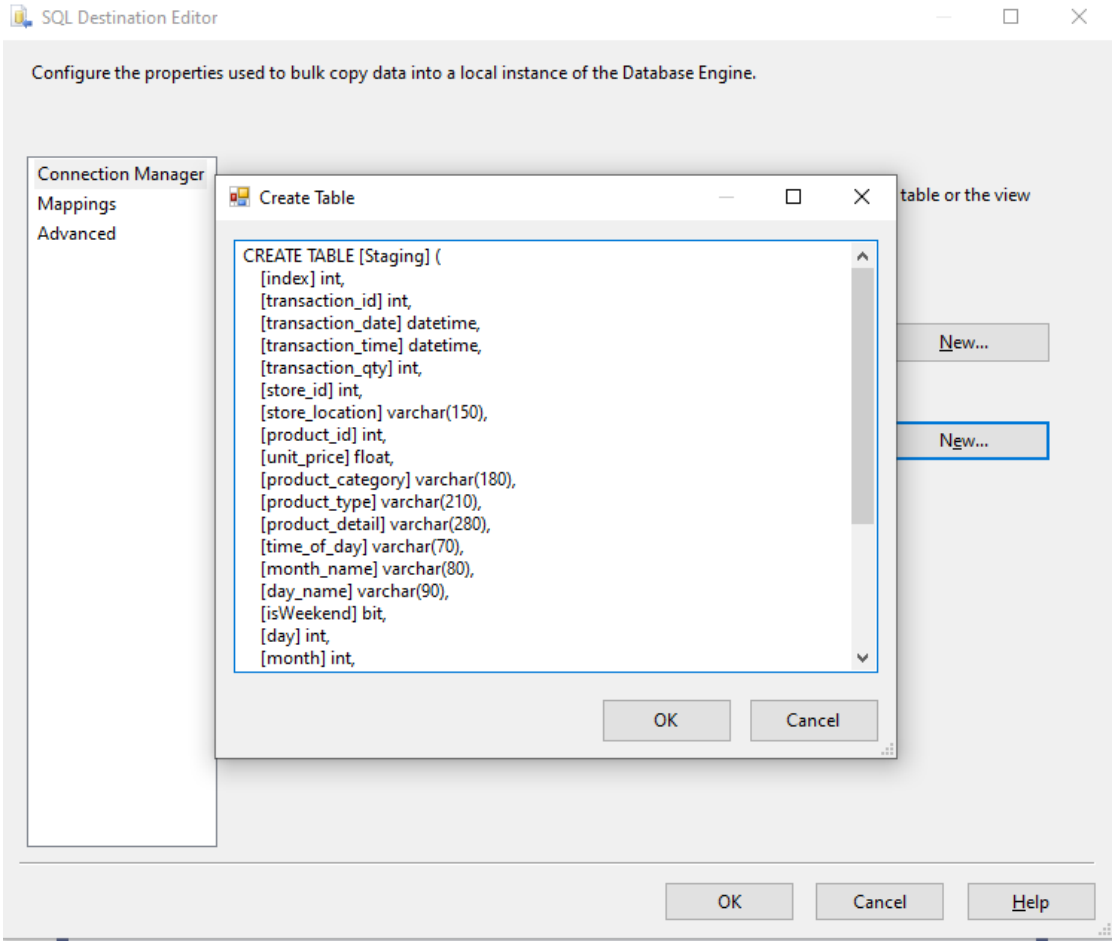


Στο κουτάκι του Flat File Source ρυθμίζεται ο κατάλληλος Connection Manager(τοποθεσία αρχείου και τύποι δεδομένων).

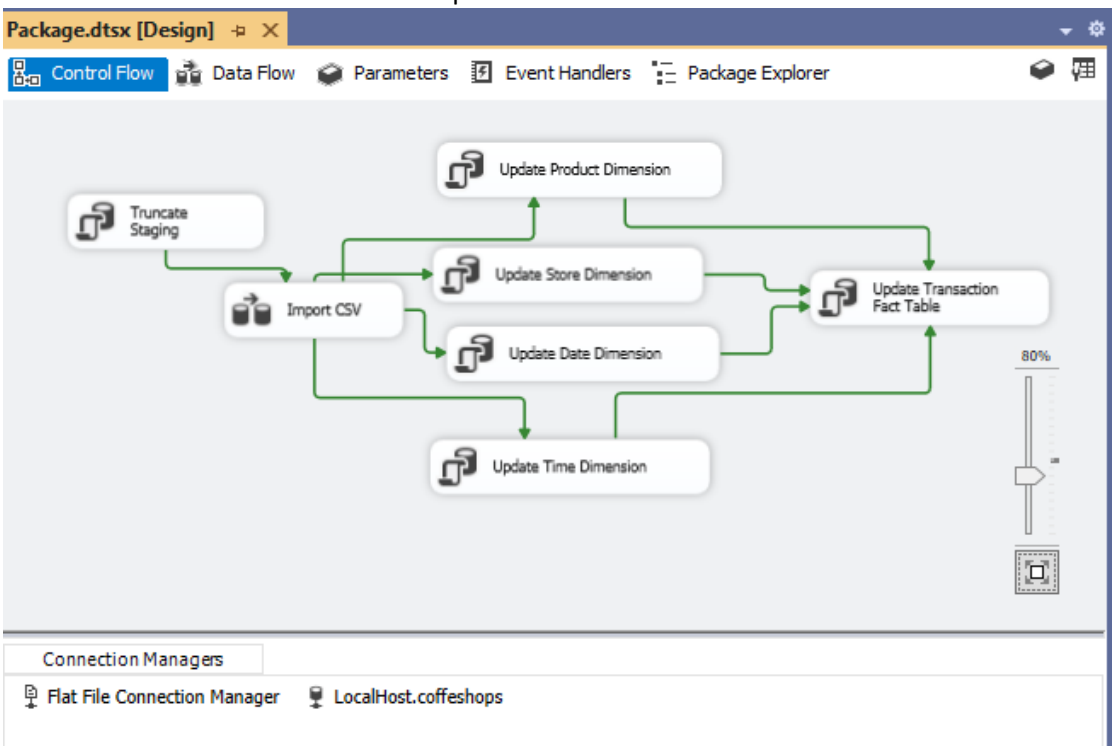


Στο κουτάκι του SQL Server Destination ρυθμίζεται ο κατάλληλος Connection Manager (στοιχεία βάσης, δημιουργία βοηθητικού πίνακα, map των στηλών του CSV με τις στήλες του βοηθητικού πίνακα). Ο βοηθητικός πίνακας θα ονομάζεται Staging και έχει ως σκοπό την προσωρινή φιλοξενία των εγγραφών στην βάση μέχρι να μεταφερθούν στου κατάλληλους

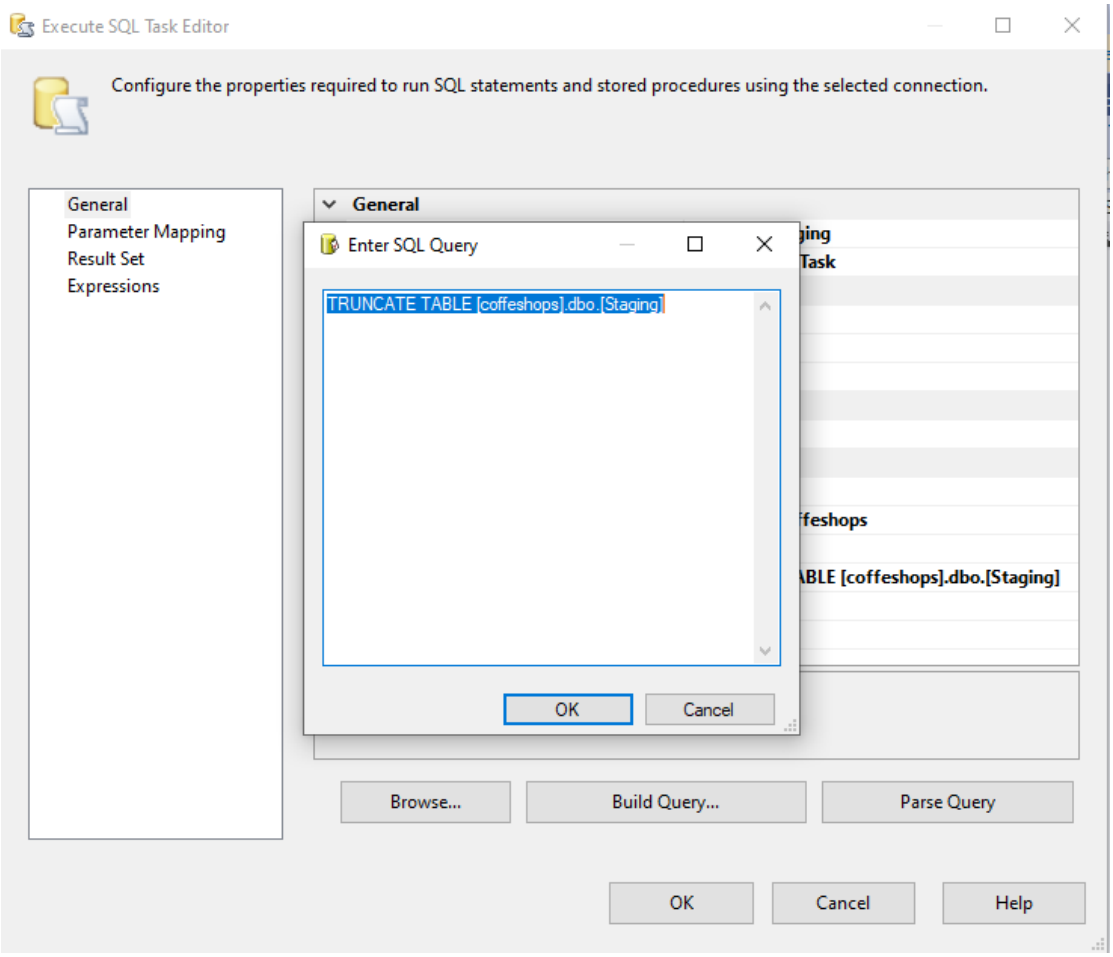
πίνακες.



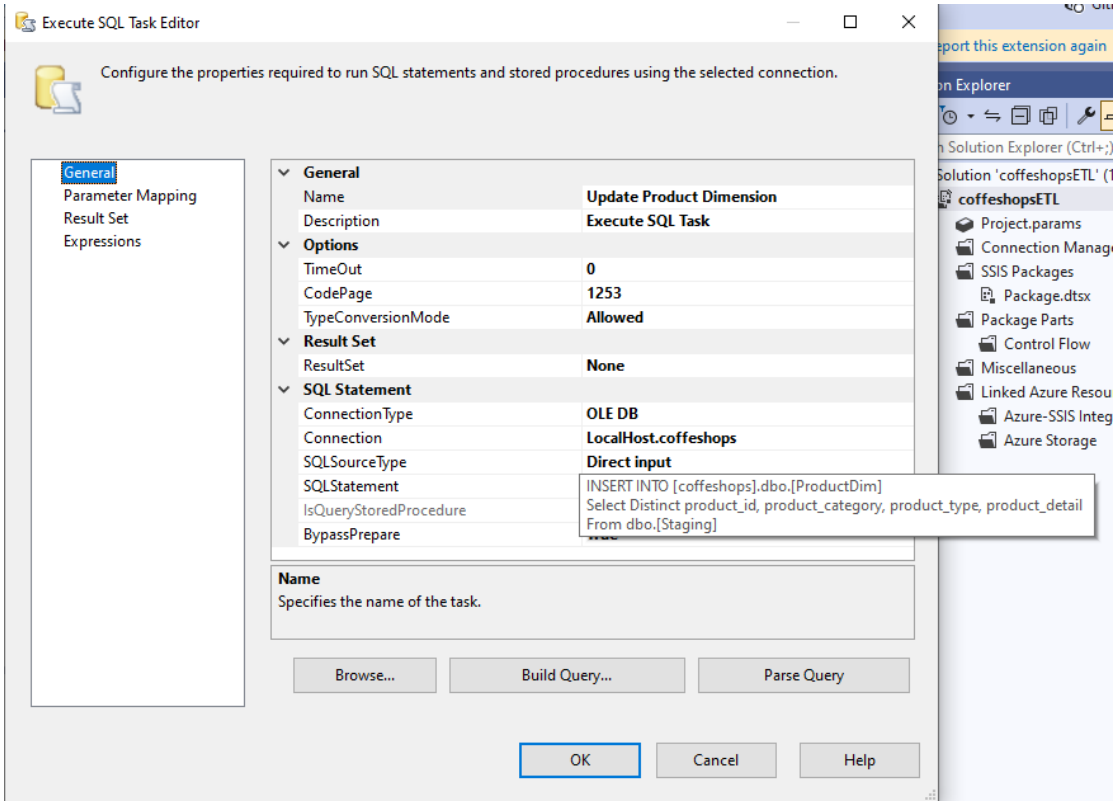
Το τελικό Control Flow θα είναι το παρακάτω



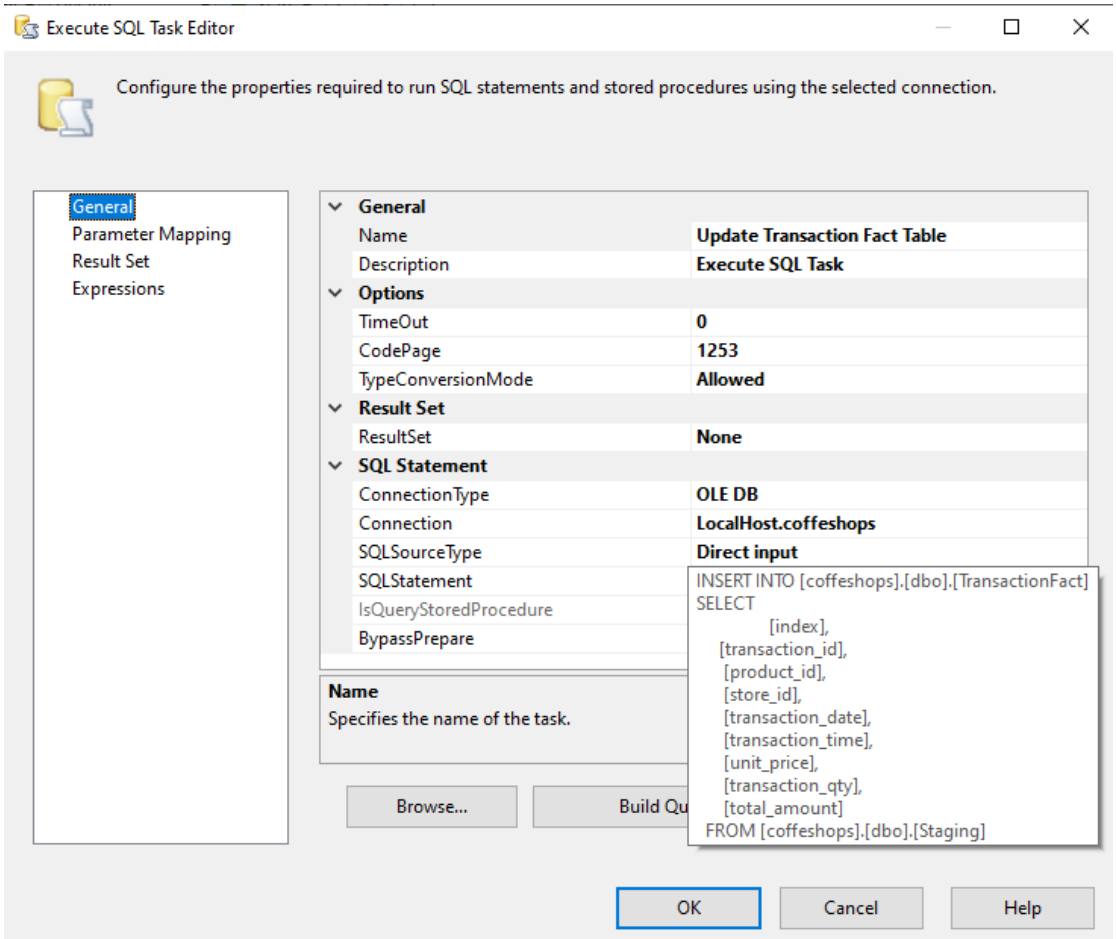
Αρχικά, ο πίνακας Staging εκκαθαρίζεται, ώστε να διασφαλιστεί ότι, σε περίπτωση επανάληψης της διαδικασίας (κάτι που συνέβη πολλές φορές στην πράξη), οι ίδιες γραμμές δεν θα εισάγονται πολλαπλές φορές στους κύριους πίνακες. Αυτό αποτρέπει τη δημιουργία διπλότυπων εγγραφών και διατηρεί τη συνέπεια των δεδομένων.



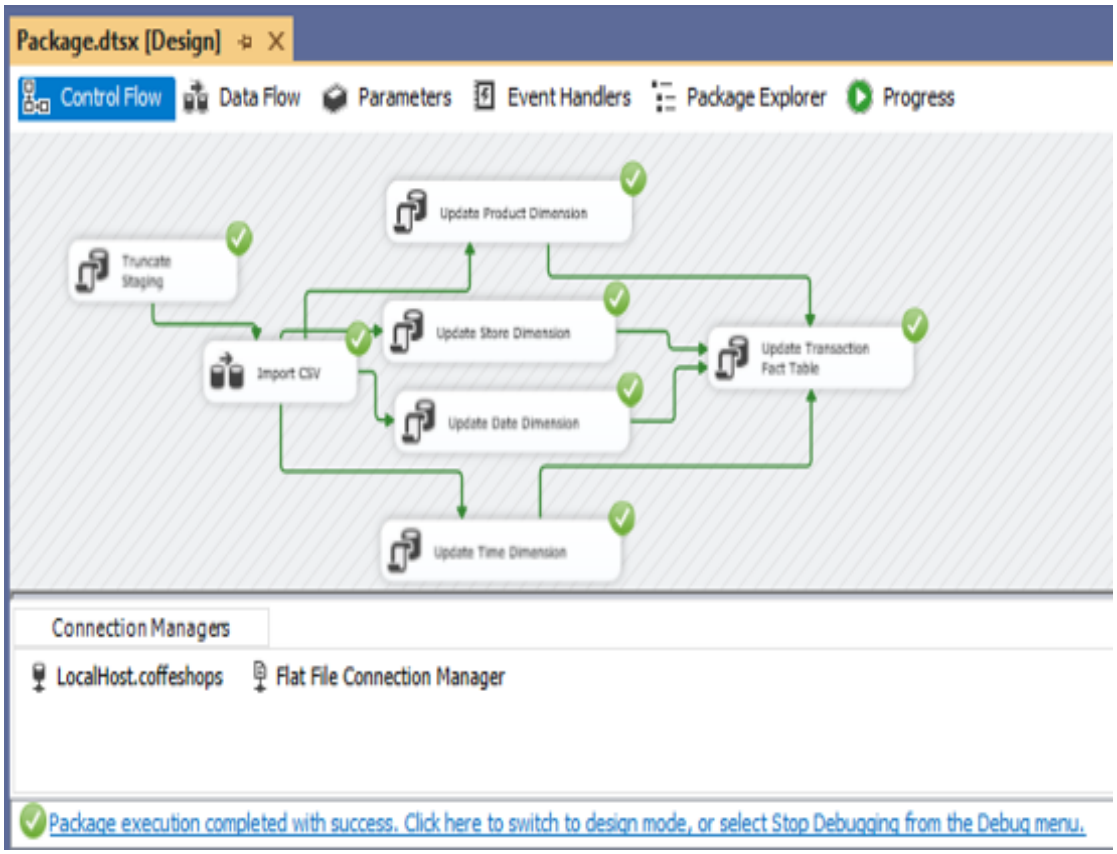
Στη συνέχεια, προχωρούμε στην ανανέωση των πινάκων διάστασης, εισάγοντας τις κατάλληλες εγγραφές που περιέχονται στον πίνακα Staging. Αυτό επιτυγχάνεται μέσω του ακόλουθου SQL query για τον πίνακα διάστασης του Προϊόντος (με αντίστοιχο τρόπο εφαρμόζεται και στους υπόλοιπους πίνακες διάστασης). Σε αυτό το σημείο, ο index που προστέθηκε κατά τη δημιουργία των πινάκων παίζει σημαντικό ρόλο, καθώς αποτρέπει την εισαγωγή διπλότυπων εγγραφών, διασφαλίζοντας έτσι τη συνέπεια των δεδομένων και την ομαλή εκτέλεση της διαδικασίας.



Τελευταίο βήμα είναι η ενημέρωση του Fact Table το οποίο θα γίνει με το ακόλουθο query.



Πλέον, εάν τρέξουμε το Flow, εκτελείται με επιτυχία.

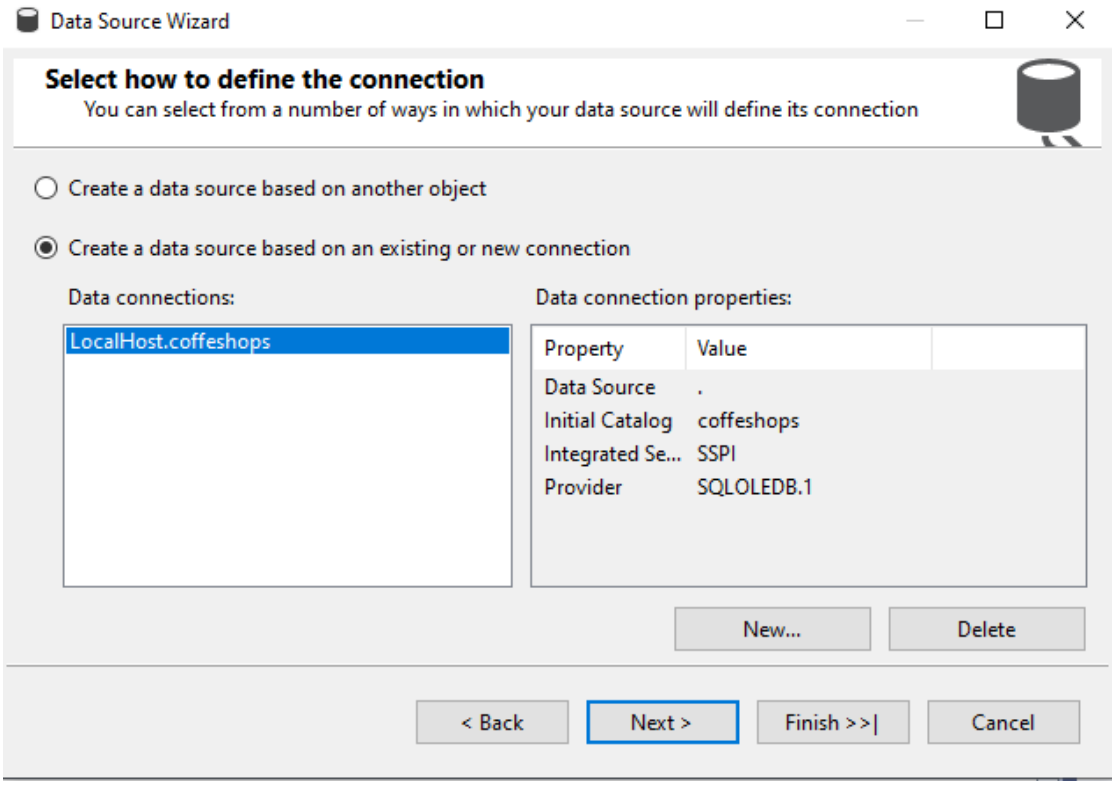


Με αυτόν τον τρόπο, μπορούμε αυτόματα να φορτώνουμε τα δεδομένα από οποιοδήποτε CSV αρχείο της ίδιας μορφής προς την τοπική βάση του SQL Server.

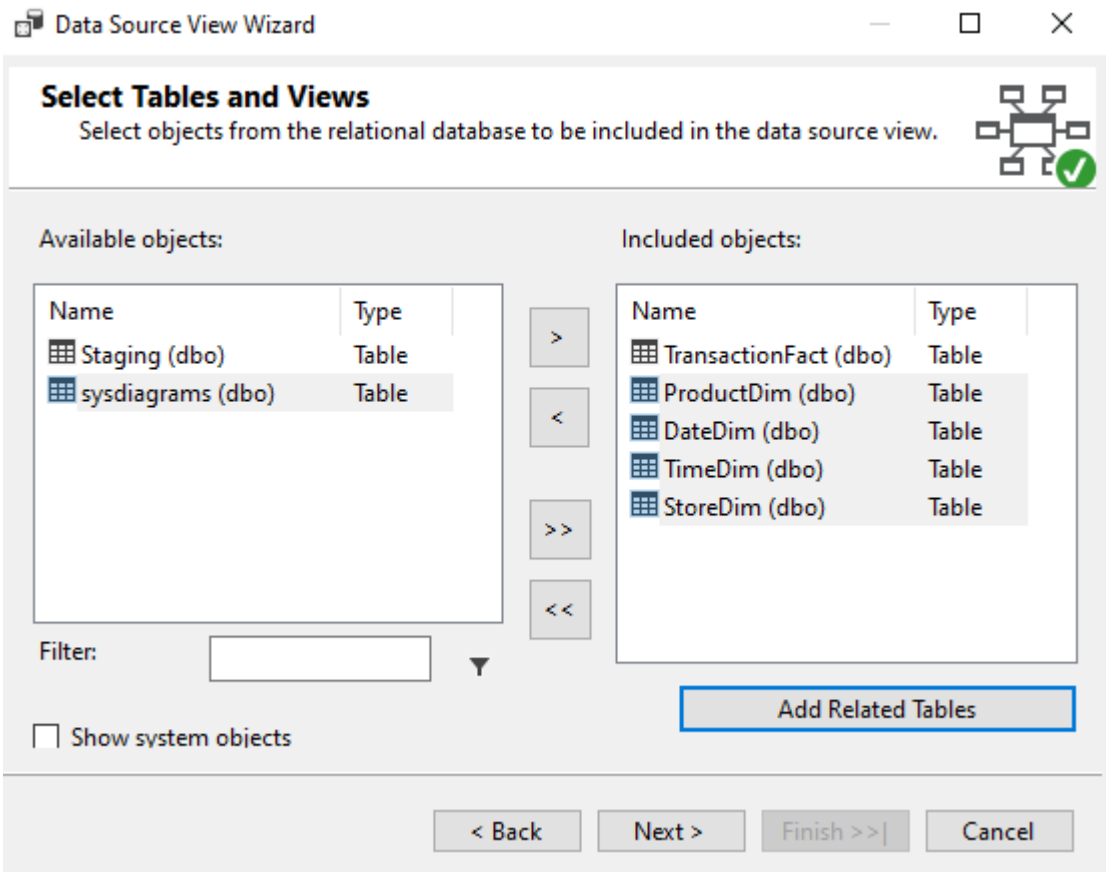
Δημιουργία Κύβου

Για την δημιουργία του κύβου χρησιμοποιήθηκε το SQL Server Analysis Services (SSAS) μέσω του Visual Studio.

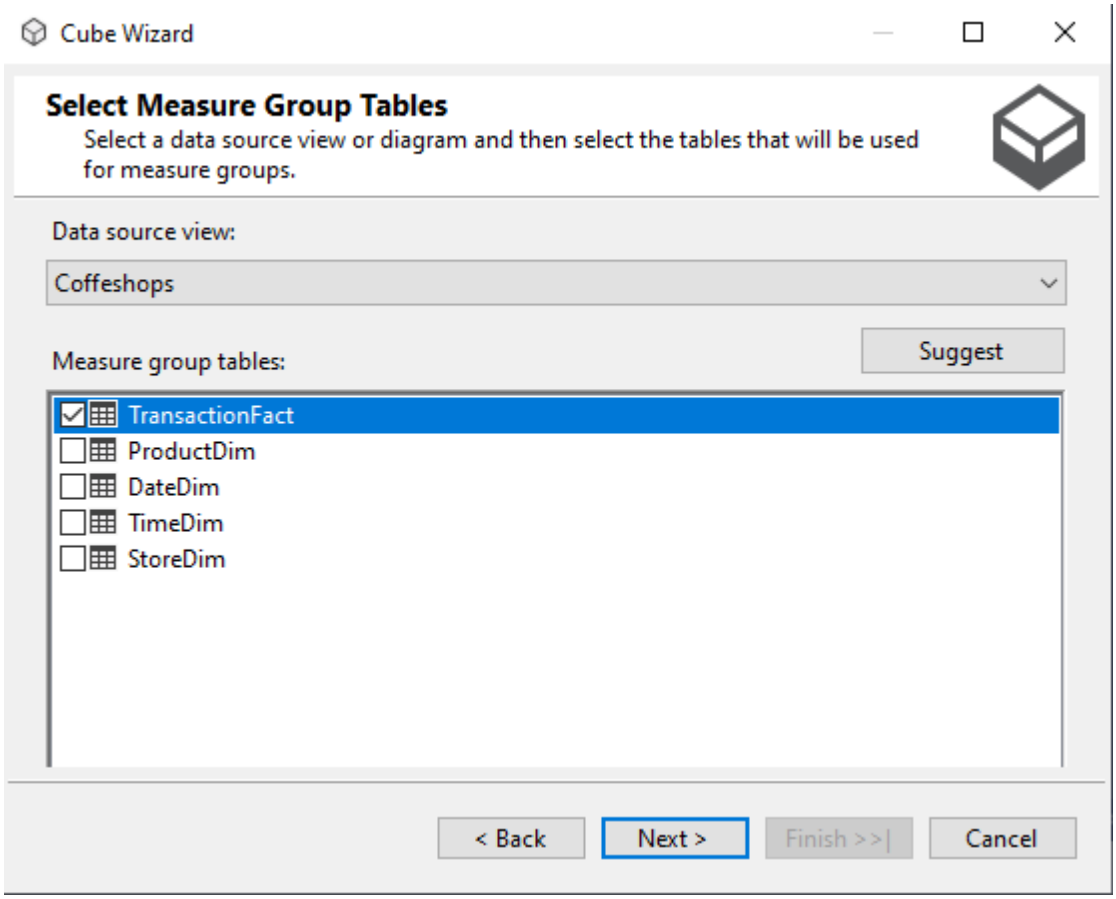
Αρχικά, δημιουργείται το Data Source

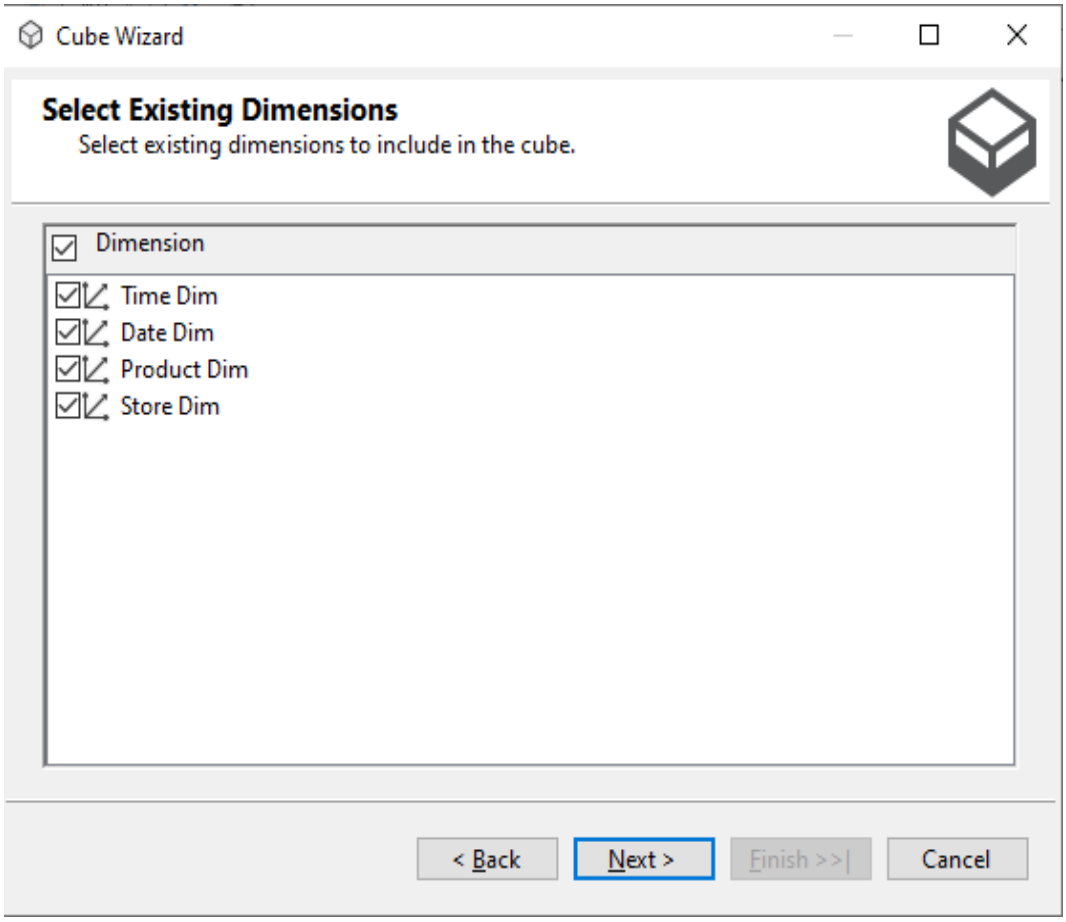


Έπειτα δημιουργείται το Source View επιλέγοντας μόνο πίνακες της βάσης που ανήκουν στο Star Schema.

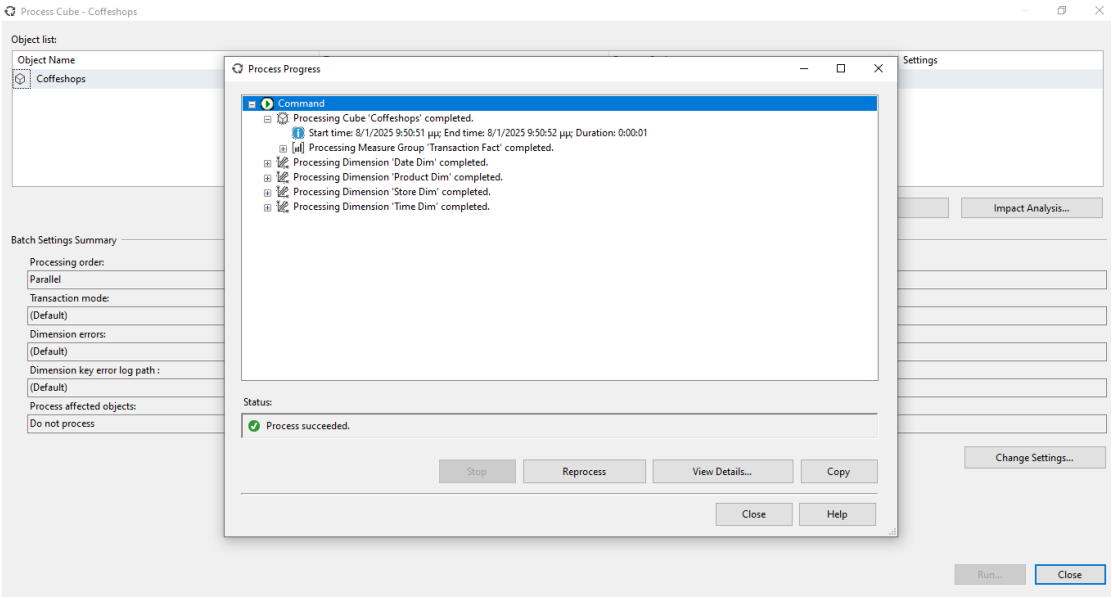


Δημιουργείται ο κύβος επιλέγοντας μόνο τον Fact Table ως measure group table και τις διαστάσεις του θα έχουν προεπιλεγεί από τους πίνακες Διάστασης.

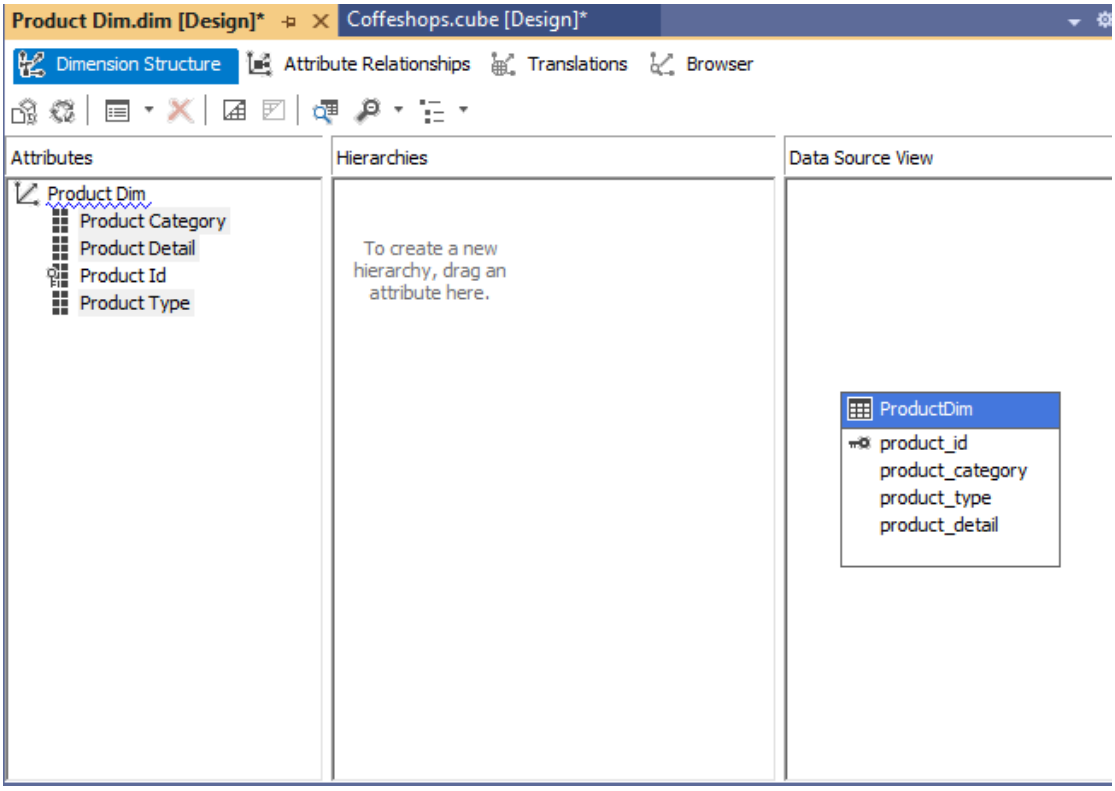

Cube Wizard
Select Measure Group Tables
Select a data source view or diagram and then select the tables that will be used for measure groups.
Data source view:
Coffeshops
Measure group tables:
Suggest
TransactionFact
ProductDim
DateDim
TimeDim
StoreDim
< Back
Next >
Finish >>|
Cancel


Cube Wizard
Select Existing Dimensions
Select existing dimensions to include in the cube.
Dimension
Time Dim
Date Dim
Product Dim
Store Dim
< Back
Next >
Finish >>|
Cancel

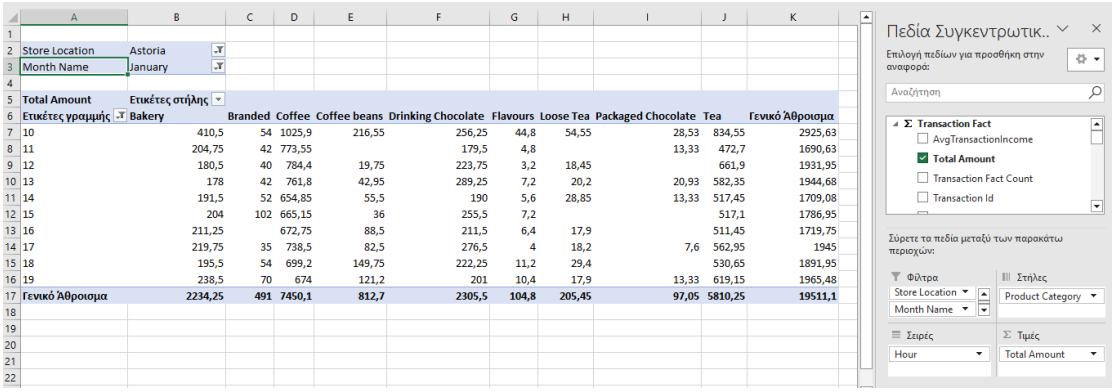
Γίνεται process ο κύβος.



Προστίθενται τα attributes σε όλους τους πίνακες διάστασης για να μπορούν να χρησιμοποιούνται στις μετρικές εάν χρειαστεί και ξαναγίνεται process τον κύβο.

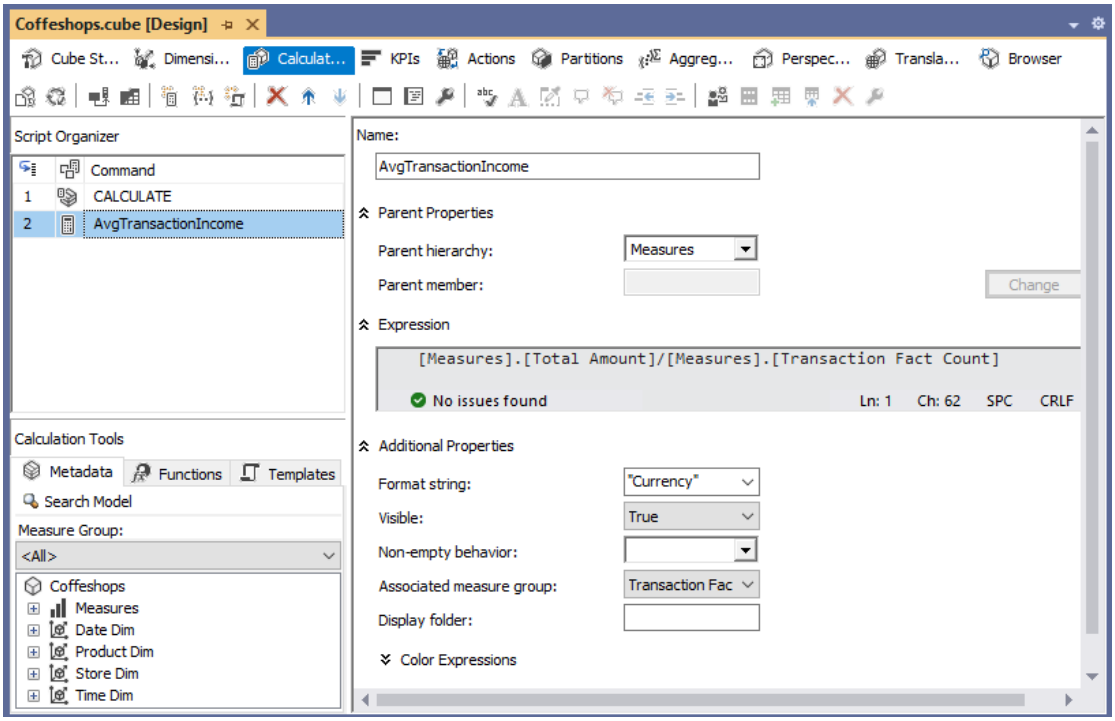


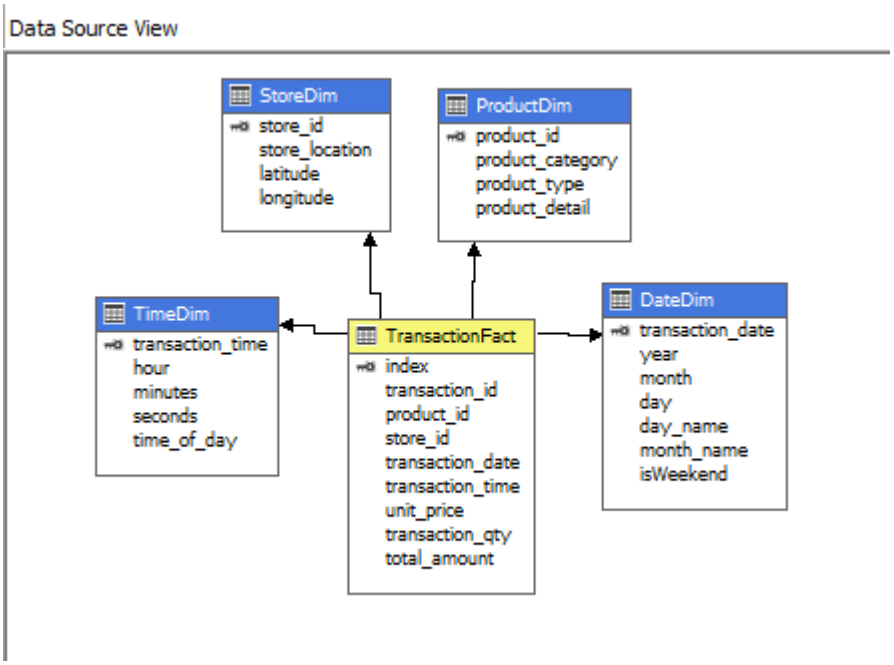
Μέσω του browser, μπορούμε να περιηγηθούμε στον κύβο και να εκτελέσουμε διάφορους περίπλοκους συνδυασμούς των μετρικών και των διαστάσεων. Για παράδειγμα, μπορούμε να υπολογίσουμε τα έσοδα για τις κατηγορίες των προϊόντων τον μήνα Ιανουάριο στο κατάστημα της Astoria, τις ώρες 12:00 με 19:59.



Store Location	Astoria									
Month Name	January									
Total Amount	Ετικέτες στήλης									
Ετικέτες γραμμής	Bakery	Branded	Coffee	Coffee beans	Drinking Chocolate	Flavours	Loose Tea	Packaged Chocolate	Tea	Γενικό Άθροισμα
10	410,5	54	1025,9	216,55	256,25	44,8	54,55	28,53	834,55	2925,63
11	204,75	42	773,55		179,5	4,8		13,33	472,7	1690,63
12	180,5	40	784,4	19,75	223,75	3,2	18,45		661,9	1931,95
13	178	42	761,8	42,95	289,25	7,2	20,2	20,93	582,35	1944,68
14	191,5	52	654,85	55,5	190	5,6	28,85	13,33	517,45	1709,08
15	204	102	665,15	36	255,5	7,2			517,1	1786,95
16	211,25		672,75	88,5	211,5	6,4	17,9		511,45	1719,75
17	219,75	35	738,5	82,5	276,5	4	18,2	7,6	562,95	1945
18	195,5	54	699,2	149,75	222,25	11,2	29,4		530,65	1891,95
19	238,5	70	674	121,2	201	10,4	17,9	13,33	619,15	1965,48
Γενικό Άθροισμα	2234,25	491	7450,1	812,7	2305,5	104,8	205,45	97,05	5810,25	19511,1

Τέλος μπορούμε να δημιουργήσουμε και κάποιες μετρικές όπως την μέση τιμή συναλλαγής.





Visualizations

Για τις οπτικοποιήσεις χρησιμοποιήθηκε το Power Bi Desktop. Η σύνδεση έγινε απευθείας με την τοπική βάση του SQL Server.

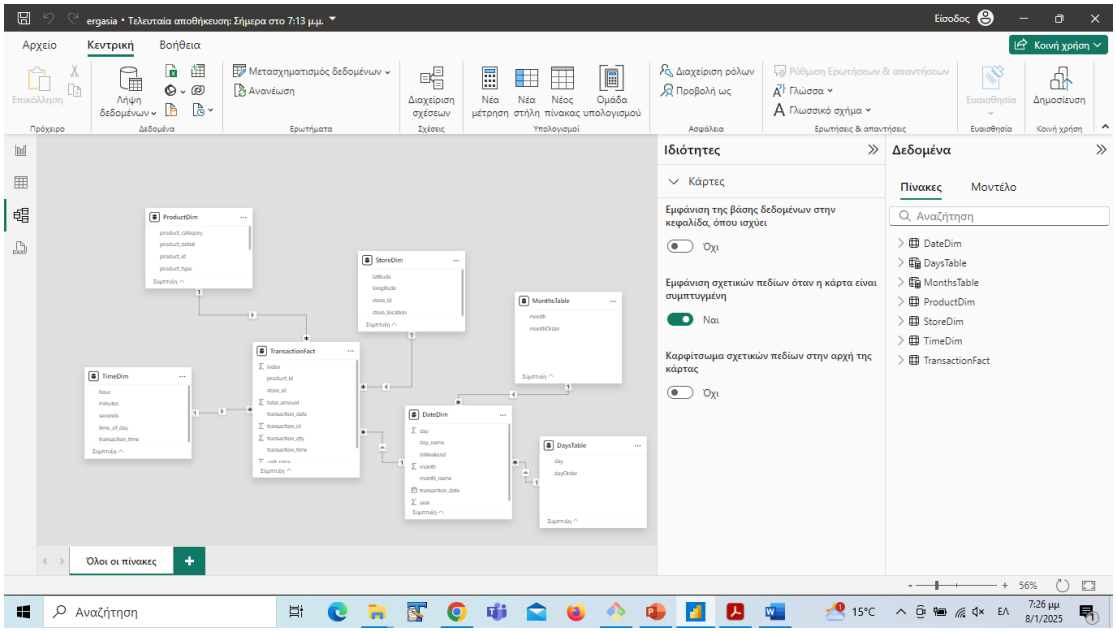
Δημιουργήθηκαν συνολικά 6 διαγράμματα και 3 φίλτρα. Φυσικά, έγιναν πολλές αλλαγές στο panel των ρυθμίσεων για κάθε διάγραμμα. Τα διαγράμματα είναι:

- Column Chart για κάθε κατάστημα το οποίο περιλαμβάνει τα συνολικά του έσοδα ανά μήνα και ανά μέρα του μήνα (υποστηρίζει δυνατότητα drill down).
- Line Chart για κάθε κατάστημα το οποίο περιλαμβάνει τον συνολικό αριθμό συναλλαγών ανά μήνα και ανά μέρα του μήνα (υποστηρίζει δυνατότητα drill down).
- Χάρτης που περιέχει τις τοποθεσίες των καταστημάτων μέσω των συντεταγμένων τους και το μέγεθος της κάθε φυσαλίδας αντιστοιχεί στα συνολικά έσοδα του καταστήματος.
- Pie chart με τον αριθμό των συναλλαγών που έγιναν ανάλογα την ώρα της μέρας.
- Bar Chart με τα έσοδα ανά προϊόν (κατηγορία ή τύπος) για όλα τα καταστήματα (υποστηρίζει δυνατότητα drill down).
- Matrix με τις ποσότητες ανά προϊόν που πωλήθηκαν (κατηγορία ή τύπος) για όλα τα καταστήματα (υποστηρίζει δυνατότητα drill down).
- Φίλτρο 1: Επιλογή Καταστήματος
- Φίλτρο 2: Επιλογή Μήνα
- Φίλτρο 3: Επιλογή Ημέρας της εβδομάδας

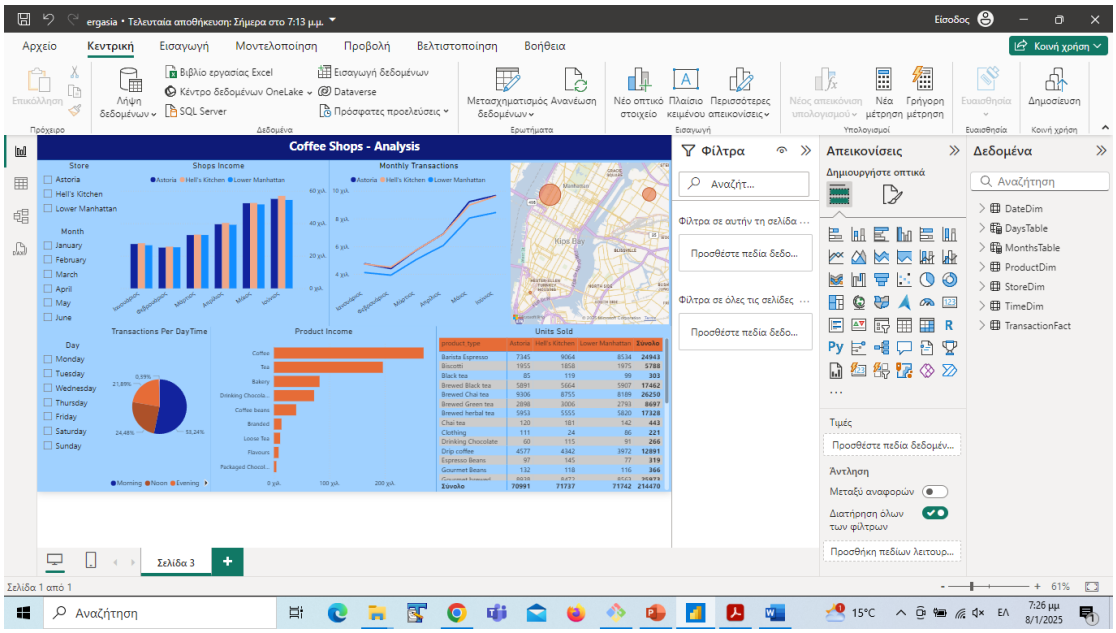
Για την σωστή - ταξινομημένη προβολή των ημερών και μηνών στα φίλτρα δημιουργήθηκαν 2 νέοι πίνακες στο power bi οι οποίοι συνδέθηκαν με τον πίνακα DateDim που περιείχε την αταξινόμητη σειρά.

```
DaysTable = DATATABLE(
    "day", STRING,
    "dayOrder", INTEGER,
    {
        {"Monday", 1},
        {"Tuesday", 2},
        {"Wednesday", 3},
        {"Thursday", 4},
        {"Friday", 5},
        {"Saturday", 6},
        {"Sunday", 7}
    })
```

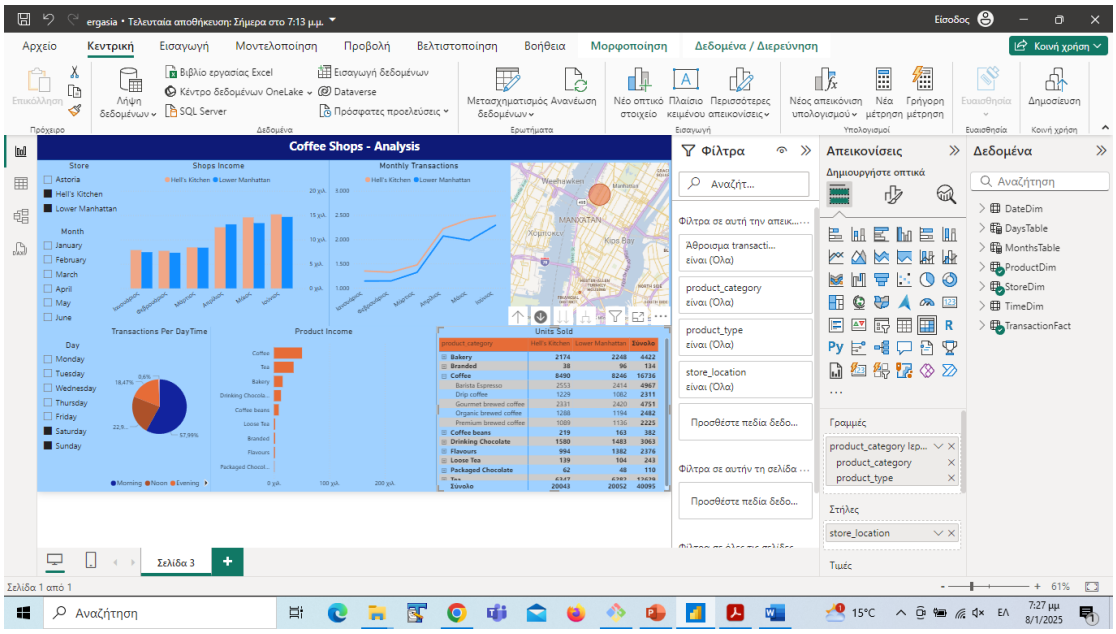
```
MonthsTable = DATATABLE(
    "month", STRING,
    "monthOrder", INTEGER,
    {
        {"January", 1},
        {"February", 2},
        {"March", 3},
        {"April", 4},
        {"May", 5},
        {"June", 6}
    })
```



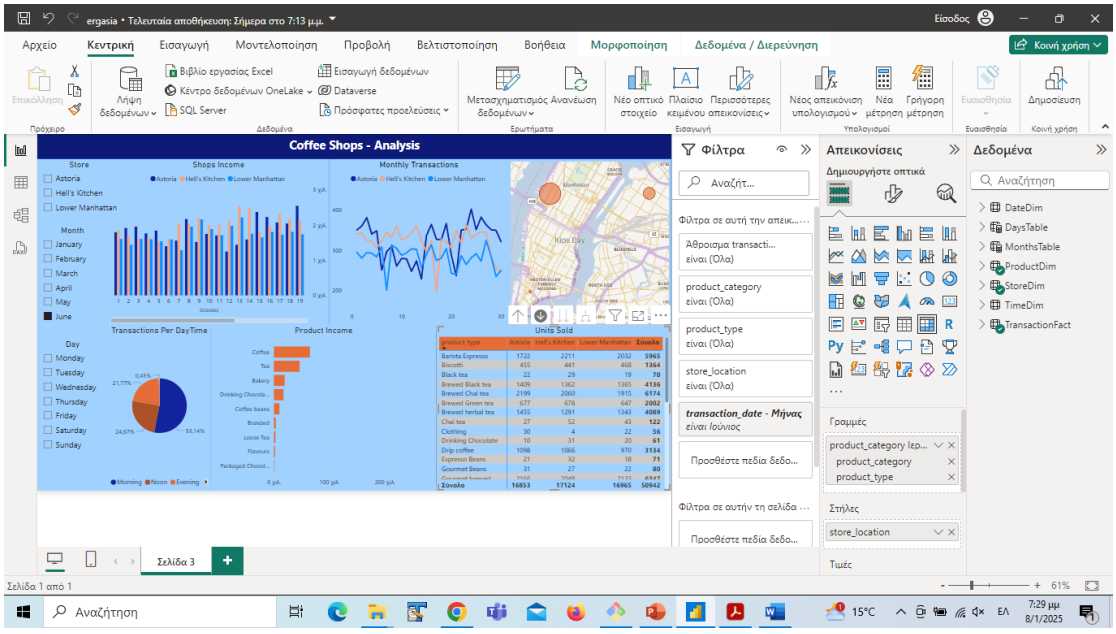
Στην συνέχεια βλέπουμε το τελικό αποτέλεσμα της οπτικοποίησης και μερικά παραδείγματα με τη χρήση των φίλτρων.



Παράδειγμα 1: Επιλογή δύο καταστημάτων και ημέρες Σάββατο και Κυριακή. Επίσης ανοίγουμε την κατηγορία του coffee στο matrix.



Παράδειγμα 2: Χρήση drill down για τον μήνα Ιούνιο, επιλογή morning από το Pie Chart και χρήση της ιεραρχίας από το matrix για όλους τους τύπους των προϊόντων.



Data mining

Τα 2 μοντέλα data mining που χρησιμοποιήθηκαν ήταν clustering στις συναλλαγές εστιάζοντας στα προϊόντα που περιλαμβάνει η κάθε συναλλαγή και association rules σχετικά με το πια ζευγάρια προϊόντων αγοράζονται συχνότερα μαζί. Και για τις 2 μεθόδους χρησιμοποιήθηκε python σε jupyter notebook.

Για το clustering αρχικά χρειάζεται ένας πίνακας ο οποίος να έχει στις γραμμές τις συναλλαγές και στις στήλες τα προϊόντα. Οι τιμές των κελιών θα είναι 1 εάν υπάρχει το αντίστοιχο προϊόν στην συναλλαγή, ανεξάρτητα από το πόσες φορές υπάρχει, και 0 εάν δεν υπάρχει το προϊόν. Επίσης πρέπει να οριστεί ποια κατηγορία προϊόντων θα χρησιμοποιηθεί ανάμεσα σε product category, type, description. Επέλεξα την type καθώς είναι η μεσαία στην ιεραρχία και δοκιμαστικά έδωσε τα καλύτερα αποτελέσματα.

[49]:
```
transaction_products_type = pd.crosstab(df['transaction_id'], df['product_type'])
transaction_products_type.head()
```

[49]:

product_type	Barista Espresso	Biscotti	Black tea	Brewed Black tea	Brewed Chai tea	Brewed Green tea	Brewed herbal tea	Chai tea	Clothing	Drinking Chocolate	...	Housewares	Organic Beans	Organic Chocolate	Organic brewed coffee	Pastry	Premium Beans
transaction_id																	
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0

5 rows × 29 columns

[50]:
```
transaction_products_type_binary = transaction_products_type.clip(upper=1)

print(transaction_products_type_binary.values.ravel().max())
print(transaction_products_type_binary.values.ravel().min())
```

```
1
0
```

Στην συνέχεια εφαρμόζεται k-means στον παραπάνω πίνακα. Με δοκιμές, ο καλύτερος αριθμός για τα clusters ήταν 5. Τέλος υπολογίζονται διάφορες μετρικές για τα clusters.

[51]:
```
kmeans = KMeans(n_clusters=5, random_state=123)
clusters = kmeans.fit_predict(transaction_products_type_binary)

transaction_products_type_binary['cluster'] = clusters
```

[52]:
```
cluster_analysis_product_type = transaction_products_type_binary.groupby('cluster').mean()
cluster_analysis_product_type
```

[53]:
```
# Εμφάνιση top 4 προϊόντων για κάθε cluster
for cluster_id, row in cluster_analysis_product_type.iterrows():
    print(f"Cluster {cluster_id}:")
    top4 = row.nlargest(4)
    for product, mean_value in top4.items():
        print(f"  {product}: {mean_value:.2f}")
    print()
```

```
Cluster 0:
  Hot chocolate: 0.20
  Brewed Black tea: 0.19
  Brewed herbal tea: 0.19
  Organic brewed coffee: 0.15

Cluster 1:
  Gourmet brewed coffee: 1.00
  Scone: 0.08
  Biscotti: 0.05
  Pastry: 0.04

Cluster 2:
  Barista Espresso: 1.00
  Regular syrup: 0.30
  Scone: 0.12
  Sugar free syrup: 0.11

Cluster 3:
  Drip coffee: 1.00
  Scone: 0.07
  Pastry: 0.05
  Biscotti: 0.04

Cluster 4:
  Brewed Chai tea: 1.00
  Scone: 0.08
  Pastry: 0.05
  Biscotti: 0.05
```

[66]:
```
transaction_products_type_binary['cluster'].value_counts()
```

[66]:
```
cluster
0    58115
4    17107
1    16889
2    16236
3     8443
Name: count, dtype: int64
```

[57]:
```
# Mean quantity and total_amount
cluster_means = transactions.groupby('cluster').agg(
    mean_quantity=('total_quantity', 'mean'),
    mean_total_amount=('total_total_amount', 'mean')
).reset_index()

print(cluster_means)
```

```
   cluster  mean_quantity  mean_total_amount
0        0       1.726043           5.925638
1        1       1.749896           5.309445
2        2       2.458795           7.786758
3        3       1.738482           4.906099
4        4       1.754136           5.665791
```

Τα τελικά clusters ονομάζονται

- Cluster 0: "Warm Beverage Enjoyers"
- Cluster 1: "Gourmet Coffee Lovers"
- Cluster 2: "Espresso Fans"
- Cluster 3: "Clasic Coffee Lovers"
- Cluster 4: "Chai Tea Enthusiasts"

Ξεκινώντας τα association rules πρέπει αρχικά θα φιλτραριστούν οι συναλλαγές που περιλάμβαναν τουλάχιστον 2 διαφορετικά προϊόντα. Στη συνέχεια θα χρησιμοποιηθεί ο πίνακας με τα μηδενικά και άσσους από το clustering.

[70]:
```
# Filter transcations με quantity >= 2
filtered_transaction_ids = transactions[transactions['different_products'] >= 2]['transaction_id']
filtered_transaction_products = transaction_products_type_binary.loc[
    transaction_products_type_binary.index.isin(filtered_transaction_ids)]
```

[71]:
```
filtered_transaction_products_no_cluster = filtered_transaction_products.drop(columns=['cluster'])
```

[72]:
```
filtered_transaction_products.head()
```

[72]:

product_type	Barista Espresso	Biscotti	Black tea	Brewed Black tea	Brewed Chai tea	Brewed Green tea	Brewed herbal tea	Chai tea	Clothing	Drinking Chocolate	...	Organic Beans	Organic Chocolate	Organic brewed coffee	Pastry	Premium Beans	Premium brewed coffee
transaction_id																	
4	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0
14	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0
19	0	1	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0
22	0	1	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0
23	0	0	0	0	0	0	1	0	0	0	...	0	0	0	1	0	0

5 rows × 30 columns

Για να βρεθούν οι κανόνες συσχέτισης εφαρμόζεται ο αλγόριθμος apriori. Οι προϋποθέσεις για να οριστεί ο κανόνας είναι να υπάρχει το ζευγάρι των προϊόντων σε τουλάχιστον 4% των συναλλαγών και το lift (πόσο πιο συχνά εμφανίζονται τα προϊόντα μαζί από ότι θα αναμένονταν ανεξάρτητα) να είναι πάνω από 1.

Επιλέγονται οι 5 κανόνες με το μεγαλύτερο confidence (πιθανότητα να αγοραστεί το 2ο προϊόν εφόσον αγοραστεί το 1ο).

[75]:
```
from mlxtend.frequent_patterns import apriori, association_rules

frequent_itemsets = apriori(filtered_transaction_products_no_cluster, min_support=0.04, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1, num_itemsets=2)

rules_sorted = rules.sort_values(by='confidence', ascending=False)
rules_sorted.head()
```

```
C:\Users\User\AppData\Local\Programs\Python\Python312\Lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:161: DeprecationWarning: DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

[75]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty	kulczy
0	(Regular syrup)	(Barista Espresso)	0.171332	0.311911	0.170676	0.996172	3.193771	1.0	0.117236	179.772282	0.828909	0.546047	0.994437	0.77
2	(Sugar free syrup)	(Barista Espresso)	0.062472	0.311911	0.062092	0.993923	3.186558	1.0	0.042607	113.221903	0.731905	0.198828	0.991168	0.59
1	(Barista Espresso)	(Regular syrup)	0.311911	0.171332	0.170676	0.547195	3.193771	1.0	0.117236	1.830077	0.998258	0.546047	0.453575	0.77
7	(Gourmet brewed coffee)	(Scone)	0.115832	0.331861	0.044697	0.385876	1.162765	1.0	0.006257	1.087955	0.158320	0.110911	0.080844	0.26
5	(Brewed Chai tea)	(Scone)	0.125220	0.331861	0.046940	0.374862	1.129577	1.0	0.005385	1.068787	0.131133	0.114449	0.064360	0.25