Βάσεις Δεδομένων

Εργασία 2018



Παναγιώτης Ιωαννίδης (**Π16036**)

Διονύσης Νίκας (**Π16097**)

Αθανάσιος Παραβάντης (**Π16112**)

Περιεχόμενα

[1. Εισαγωγή 4](#_Toc515547628)

[2. Ανάλυση Πινάκων 5](#_Toc515547629)

[2.1. Car Makes 5](#_Toc515547630)

[2.2. Car Models 6](#_Toc515547631)

[2.3. Car Warehouse 7](#_Toc515547632)

[2.4. Customers 8](#_Toc515547633)

[2.5. Employee 9](#_Toc515547634)

[2.6. Roles 10](#_Toc515547635)

[2.7. Sales History 11](#_Toc515547636)

[2.8. Service History 12](#_Toc515547637)

[3. Ανάλυση Δομών 13](#_Toc515547638)

[3.1. Plate Number 13](#_Toc515547639)

[3.2. Car Condition 13](#_Toc515547640)

[3.3. Sales Action 13](#_Toc515547641)

[4. Θεωρία Κανονικοποίησης 14](#_Toc515547642)

[5. Επεξήγηση των Queries 14](#_Toc515547643)

[5.1. Μοντέλα αυτοκινήτων με το μέγιστο πλήθος ζημιών 14](#_Toc515547644)

[5.2. Μέσο κέδρος της εταιρίας από επισκευές ανά μήνα 15](#_Toc515547645)

[5.3. Ο πωλητής με το μέγιστο «τζίρο» 16](#_Toc515547646)

[5.4. Επισκευές που βρίσκονται σε εκκρεμότητα 18](#_Toc515547647)

[5.5. Εργασίες του τεχνικού ‘Χ’ τον τελευταίο μήνα 19](#_Toc515547648)

[5.6. Αυτοκίνητα που ήρθαν για επισκευή πάνω από 1 φορά τον τελευταίο χρόνο 20](#_Toc515547649)

[6. Επεξήγηση Trigger & Cursor 22](#_Toc515547650)

[6.1. Trigger 22](#_Toc515547651)

[6.2. Cursor 25](#_Toc515547652)

[7. Παράδειγμα σύνδεσης με JDBC 26](#_Toc515547653)

[7.1. Δομή 26](#_Toc515547654)

[7.2. Εκτέλεση 27](#_Toc515547655)

# Εισαγωγή

Η φετινή εργασία έχει θέμα τη δημιουργία μιας βάσης δεδομένων για εταιρία εμπορίας αυτοκινήτων. Για τις ανάγκες της επιχείρησης, ζητείται η αποθήκευση αρκετών στοιχείων που καθορίζουν σημαντικές πληροφορίες και υποχρεώσεις που πρέπει να εκπληρώσει. Όσο για το τεχνικό κομμάτι, χρησιμοποιήσαμε τη **PostgreSQL** για την δημιουργία της ΒΔ, καθώς και τη γλώσσα προγραμματισμού **Java** ώστε να υλοποιήσουμε μια παραδειγματική εφαρμογή που συνδέεται με τη βάση και αλληλεπιδρά με τα αποθηκευμένα δεδομένα. Η δημιουργία εικονικών καταχωρήσεων έγινε με το εργαλείο **Mockaroo**.

Σύμφωνα με την εκφώνηση της εργασίας, μας ζητούνται δομές και πίνακες που θα υποστηρίζουν την αποθήκευση των παρακάτω στοιχείων της επιχείρησης:

* Στοιχεία πωλητών και τεχνικών
* Ιστορικό επισκευών και σχετικές λεπτομέρειες
* Ιστορικό αγορών και σχετικές λεπτομέρειες
* Αυτοκίνητα που διαχειρίζεται η εταιρία
* Ιδιώτες που συναλλάσσονται με την εταιρία

Στην αμέσως επόμενη ενότητα θα αναλύσουμε τη σχεδίαση της βάσης δεδομένων για την υποστήριξη των παραπάνω πληροφοριών.

# Ανάλυση Πινάκων

Όλες οι εντολές SQL για τη δημιουργία των παρακάτω πινάκων βρίσκονται στο αρχείο **part1/all.sql** και ξεχωριστά η καθεμία στο φάκελο **part1**. Χρησιμοποιώντας το query tool του pgAdmin, εκτελώντας ότι βρίσκεται στο part1/all.sql θα δημιουργηθεί επιτυχώς η βάση δεδομένων της εργασίας.

## Car Makes

Ο πίνακας car\_makes αποθηκεύει κωδικούς εταιριών κατασκευής αυτοκινήτων και τις αντίστοιχες ονομασίες τους.

**Χαρακτηριστικά:**

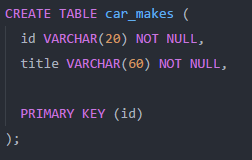
* **id** (20 characters)
* **title** (20 characters)

**Πρωτεύον Κλειδί:**

* **id**: Κάθε εταιρία αντιπροσωπεύεται από έναν μοναδικό κωδικό.

**Ξένα Κλειδιά:**

* Κανένα



## Car Models

Ο πίνακας car\_models αποθηκεύει κωδικούς μοντέλων αυτοκινήτων, την εταιρία κατασκευαστή και την ονομασία του μοντέλου.

**Χαρακτηριστικά**

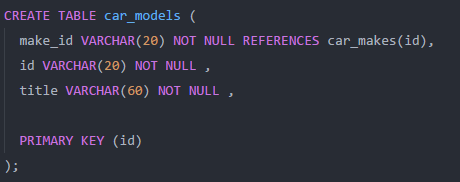
* **make\_id** (20 characters)
* **id** (20 charaters)
* **title** (60 charaters)

**Πρωτεύον Κλειδί**

* **id**: Κάθε μοντέλο αμαξιού αντιπροσωπεύεται από έναν μοναδικό κωδικό.

**Ξένα Κλειδιά**

* **make\_id**: αναφορά στο **id** του πίνακα **car\_makes**.



## Car Warehouse

Ο πίνακας car\_warehouse αποθηκεύει τα αυτοκίνητα που διαχειρίζεται η εταιρία.

**Χαρακτηριστικά:**

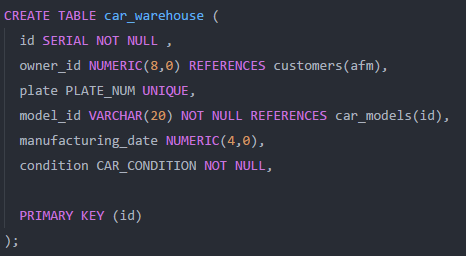
* **id** (serial)
* **owner\_id** (numeric 8, 0)
* **plate** (plate\_num, unique)
* **model\_id** (20 characters)
* **manufacturing\_date** (numeric 4, 0)
* **condition** (car\_condition)

**Πρωτεύον Κλειδί:**

* **id**: Κάθε εγγραφή χαρακτηρίζεται από έναν αύξον αριθμό.

**Ξένα Κλειδιά:**

* **owner\_id** αναφορά στο **afm** του πίνακα **customers**.
* **model\_id** αναφορά στο **id** του πίνακα **car\_models**.



## Customers

Ο πίνακας customers αποθηκεύει τα στοιχεία πελατών που συναλλάσσονται με την εταιρία.

Χαρακτηριστικά:

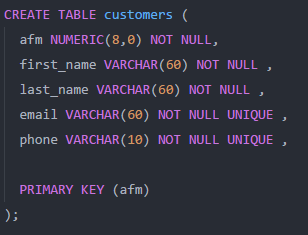
* **afm** (numeric 8, 0)
* **first\_name** (60 characters)
* **last\_name** (60 characters)
* **email** (60 characters)
* **phone** (60 characters)

**Πρωτεύον Κλειδί:**

* **afm**: Κάθε πελάτης έχει ένα μοναδικό αριθμό ΑΦΜ.

**Ξένα Κλειδιά:**

* Κανένα



## Employee

Ο πίνακας employee αποθηκεύει τους εργαζόμενους της εταιρίας που χωρίζονται σε δυο κατηγορίες: τεχνικοί και πωλητές.

Χαρακτηριστικά:

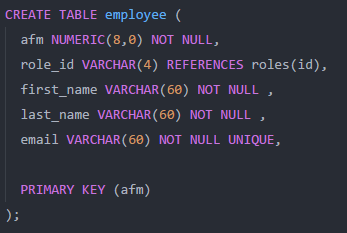
* **afm** (numeric 8, 0)
* **role\_id** (4 characters)
* **first\_name** (60 characters)
* **last\_name** (60 characters)
* **email** (60 characters)

Πρωτεύον Κλειδί:

* **afm**: Κάθε εργαζόμενος έχει ένα μοναδικό αριθμό ΑΦΜ.

Ξένα Κλειδιά:

* **role\_id** αναφέρεται στο **id** του πίνακα **roles**.



## Roles

Ο πίνακας roles αποθηκεύει όλους τους πιθανούς ρόλους των εργαζομένων της εταιρίας.

**Χαρακτηριστικά:**

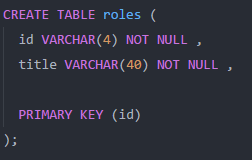
* **id** (4 characters)
* **title** (40 characters)

**Πρωτεύον Κλειδί:**

* **id**: Κάθε τίτλος έχει έναν μοναδικό κωδικό id.

**Ξένα Κλειδιά:**

* Κανένα



## Sales History

Ο πίνακας sales\_history αποθηκεύει το ιστορικό αγορών και πωλήσεων της εταιρίας.

**Χαρακτηριστικά:**

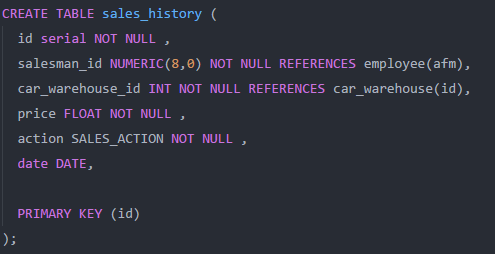
* **id** (serial)
* **salesman**\_id (numeric 8, 0)
* **car\_warehouse\_id** (integer)
* **price** (float)
* **action** (sales\_action)
* **date** (date)

**Πρωτεύον Κλειδί:**

* **id**: Κάθε εγγραφή χαρακτηρίζεται από έναν αύξον αριθμό.

**Ξένα Κλειδιά:**

* **salesman\_id**: αναφέρεται στο **afm** του πίνακα **employees**.
* **car\_warehouse\_id**: αναφέρεται στο **id** του πίνακα **car\_warehouse**.



## Service History

Ο πίνακας service\_history αποθηκεύει το ιστορικό του service της εταιρίας.

**Χαρακτηριστικά:**

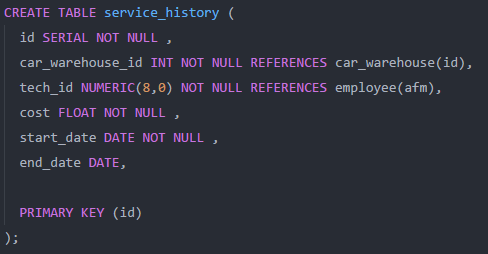
* **id** (serial)
* **car\_warehouse\_id** (integer)
* **tech\_id** (numeric 8, 0)
* **cost** (float)
* **start\_date** (date)
* **end\_date** (date)

**Πρωτεύον Κλειδί:**

* **id**: Κάθε εγγραφή χαρακτηρίζεται από έναν αύξον αριθμό.

**Ξένα Κλειδιά:**

* **tech\_id**: αναφορά στο **afm** του πίνακα **employee**.
* **car\_warehouse\_id**: αναφορά στο **id** του πίνακα **car\_warehouse.**



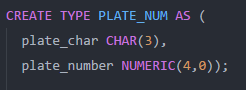
# Ανάλυση Δομών

Όλες οι εντολές SQL για τη δημιουργία των παρακάτω δομών βρίσκονται στο αρχείο **part1/all.sql** και ξεχωριστά η καθεμία στο φάκελο **part1**.

## Plate Number

Χρησιμοποιείται για την αναπαράσταση πινάκων κυκλοφορίας.

* **plate\_char** (characters)
* **plate\_number** (numeric)



## Car Condition

Χρησιμοποιείται για την κατάσταση του αυτοκινήτου.

* Enum: new, used



## Sales Action

Χρησιμοποιείται για τον διαχωρισμό των αγορών από τις πωλήσεις.

* Enum: buy, sale



# Θεωρία Κανονικοποίησης

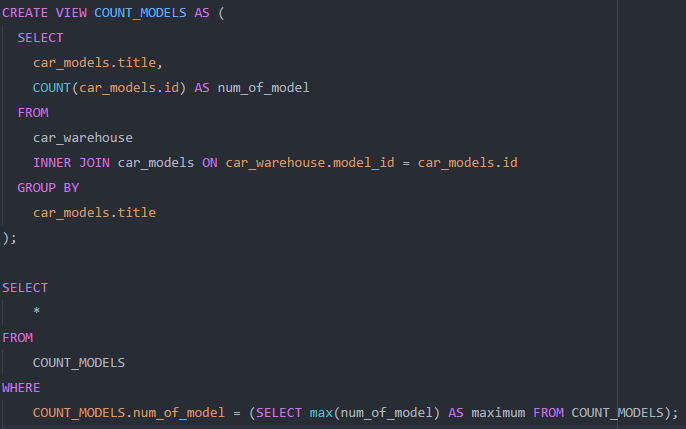
Η εφαρμογής της θεωρίας κανονικοποίησης πάνω στο σχεσιακό σχήμα της ΒΔ βρίσκεται στο αρχείο **bcnf\_3nf.txt**.

# Επεξήγηση των Queries

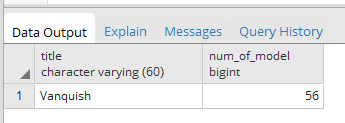
Όλα τα queries για τα ερωτήματα που ζητούνται στην εργασία βρίσκονται στο αρχείο **part2/all.sql** και ξεχωριστά το καθένα στο φάκελο **part2**.

## Μοντέλα αυτοκινήτων με το μέγιστο πλήθος ζημιών

Για την απλούστευση του ερωτήματος, δημιουργούμε ένα view το οποίο περιέχει όλα τα μοντέλα αμαξιών και πόσες φορές έχει έρθει το καθένα για service, ανεξάρτητα αν πολλοί διαφορετικοί πελάτες έχουν το ίδιο αμάξι.

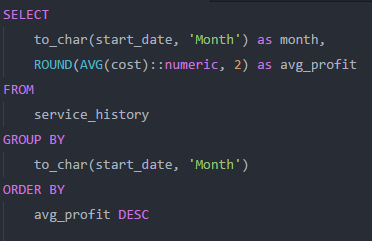


Ενδεικτικά αποτελέσματα του query:

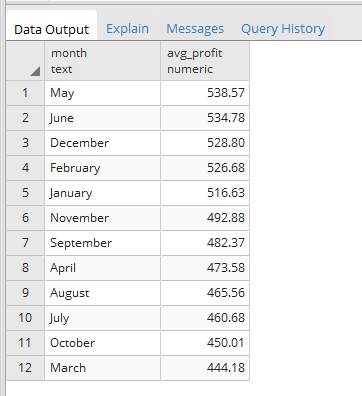


## Μέσο κέδρος της εταιρίας από επισκευές ανά μήνα

Το συγκεκριμένο ερώτημα είναι αρκετά απλό και δεν χρειάζεται χρήση view για την απλούστευση του. Γίνεται στρογγυλοποίηση του κέρδους για τη προβολή του.



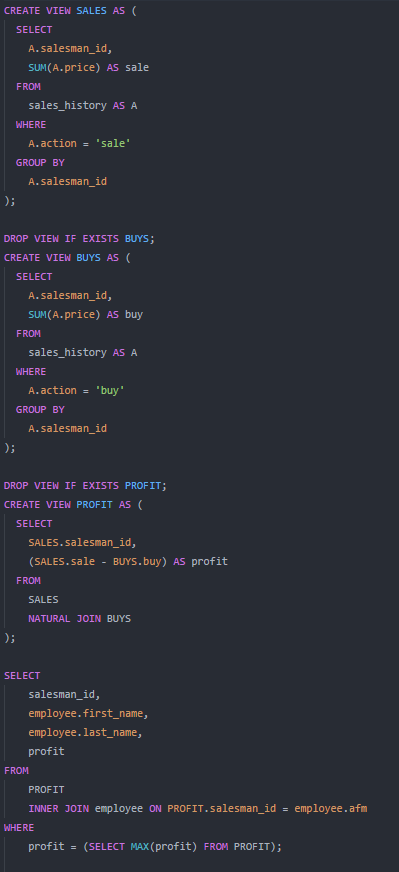
Ενδεικτικά αποτελέσματα του query:



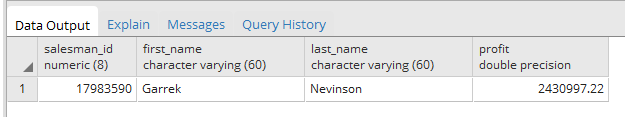
## Ο πωλητής με το μέγιστο «τζίρο»

Αυτό το ερώτημα γίνεται αρκετά απλούστερο με τη χρήση views όπως βλέπουμε παρακάτω.

Δημιουργήσαμε τρία views για τις πωλήσεις, τις αγορές και το κέρδος:

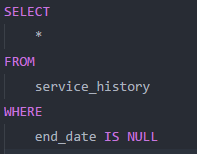


Ενδεικτικά αποτελέσματα του query:

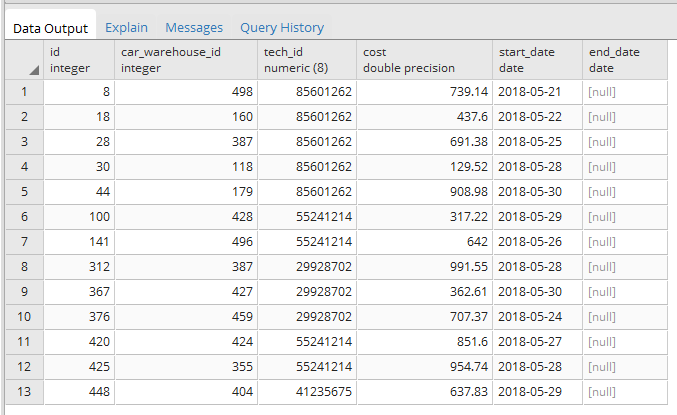


## Επισκευές που βρίσκονται σε εκκρεμότητα

Πολύ απλό ερώτημα στο οποίο ο μόνος έλεγχος είναι στην ύπαρξη της τελικής ημερομηνίας.

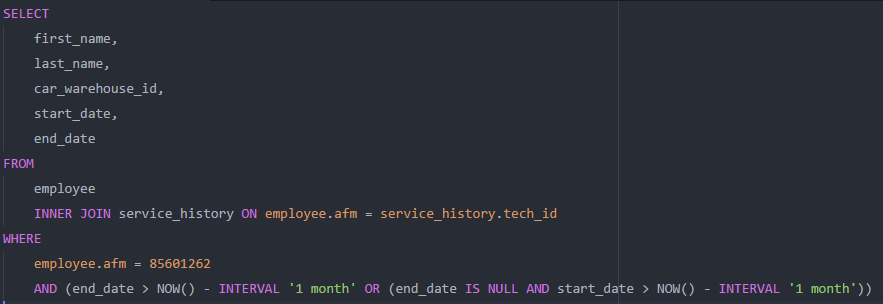


Ενδεικτικά αποτελέσματα του query:

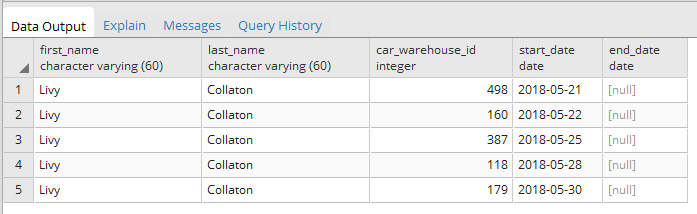


## Εργασίες του τεχνικού ‘Χ’ τον τελευταίο μήνα

Για τις ανάγκες του ερωτήματος επιλέξαμε τον τεχνικό με ΑΦΜ 85601262.

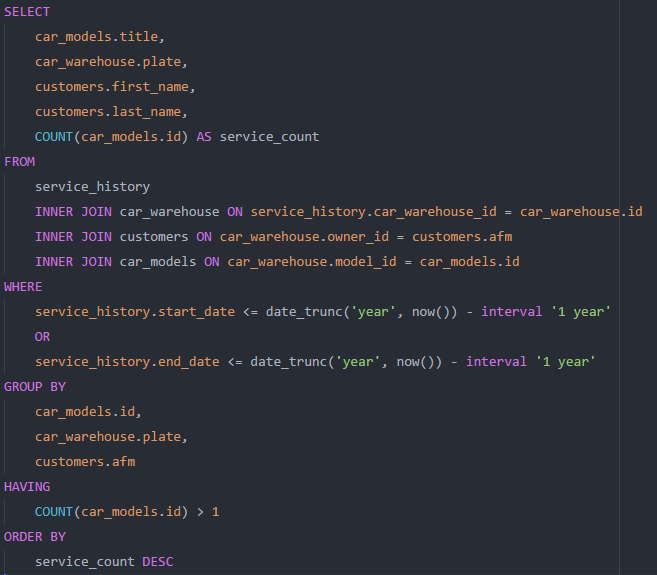


Ενδεικτικά αποτελέσματα του query:

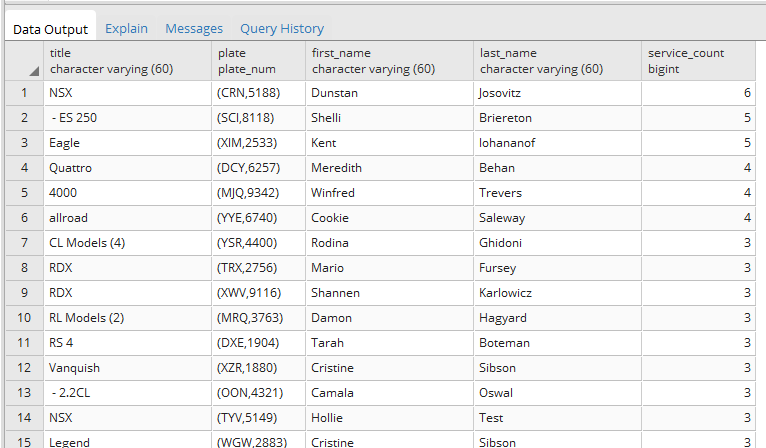


## Αυτοκίνητα που ήρθαν για επισκευή πάνω από 1 φορά τον τελευταίο χρόνο

Σε αυτό το ερώτημα έχουμε κάνει αρκετά join, groups και ordering για τη σωστή απάντηση με τα κατάλληλα στοιχεία. Γίνεται έλεγχος και για την ημερομηνία αρχής και τέλους των επισκευών.



Ενδεικτικά αποτελέσματα του query:



# Επεξήγηση Trigger & Cursor

Ο trigger και cursor που ζητούνται στην εργασία βρίσκονται στο αρχείο **part3/all.sql** και ξεχωριστά το καθένα στο φάκελο **part3**.

## Trigger

Σε αυτό ερώτημα κατασκευάσαμε έναν trigger ο οποίος υλοποιεί την αυτόματη ενημέρωση του ιστορικού των αυτοκίνητων που διατηρείται από την εταιρεία.

Η υλοποίηση που επιλέξαμε να πραγματοποιήσουμε είναι όσο το δυνατόν πιο κοντά στη λειτουργεία μια πραγματικής εφαρμογής και καλύπτει τις, πιθανές, δυο περιπτώσεις:

1. Να υπάρχει ήδη στο ιστορικό των αυτοκίνητων (car\_warehouse) το αυτοκίνητο που θα εισαχθεί στο service\_history.
2. Να μην υπάρχει στο ιστορικό των αυτοκίνητων (car\_warehouse) το αυτοκίνητο που θα εισαχθεί στο service\_history. Εκεί θα πρέπει πρώτα να δημιουργήσουμε την εγγραφή με τα στοιχεία του νέου πελάτη (σχέση customers), δεύτερον θα δημιουργήσουμε την εγγραφή με τα στοιχεία του αυτοκίνητου (σχέση car\_warehouse) και τέλος τα απαραίτητα στοιχεία του αυτοκίνητου που ήρθε στον πίνακα των επισκευών (σχέση service\_history).

Για να πετύχουμε την αλυσιδωτή υλοποίηση που έχει περιγραφεί στην δεύτερη περίπτωση, δημιουργούμε ένα προσωρινό πίνακα με όνομα temp\_car\_service οπού περιέχει τα καταλληλά χαρακτηριστικά.

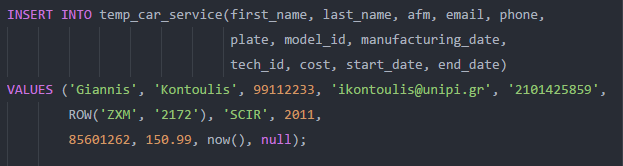
Στη συνέχεια, δημιουργούμε τη συνάρτηση που θα εκτελείται από τον trigger. Αρχικά γίνεται έλεγχος για τον αν υπάρχει ο αριθμός κυκλοφορίας στον πίνακα με τα στοιχεία των αυτοκίνητων (car\_warehouse). Αν υπάρχει τότε εκτελείται η πρώτη περίπτωση, δηλαδή η απευθείας εισαγωγή των στοιχείων στον πίνακα service\_history.

Αν δεν υπάρχει ο αριθμός κυκλοφορίας τότε:

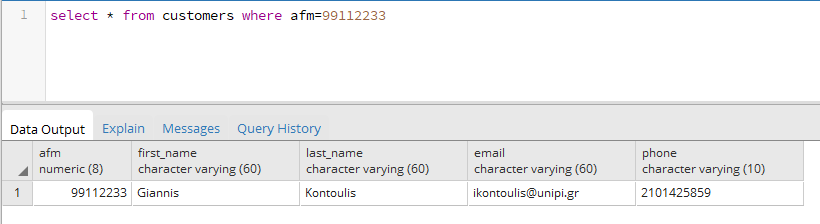
1. Γίνεται η εισαγωγή των στοιχείων του πελάτη στον πίνακα customers.
2. Εισάγονται τα στοιχεία του αυτοκίνητου στον πίνακα car\_warehouse.
3. Τέλος, εισάγονται τα στοιχεία στον πίνακα service history.

Αμέσως μετά δημιουργούμε τον trigger πάνω στον προσωρινό πίνακα temp\_car\_service. Ορίζουμε ότι ο trigger θα πρέπει να ενεργοποιείται πριν εισαγωγή των στοιχείων σε αυτόν τον πίνακα.

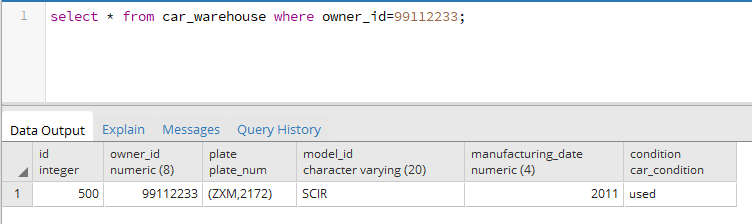
Ας δούμε ένα παράδειγμα στο οποίο εισάγουμε τα εξής στον πίνακα temp\_car\_service:



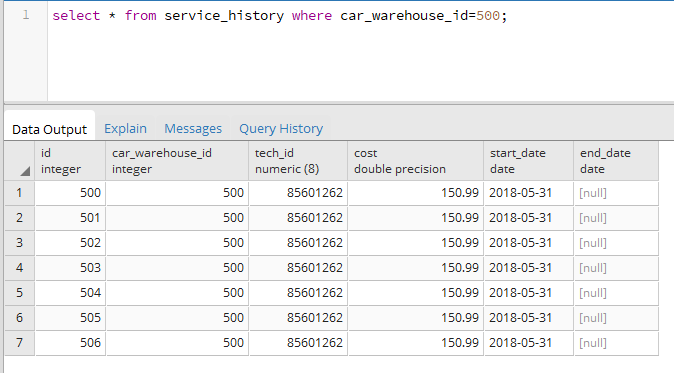
Έχοντας δημιουργήσει το αντίστοιχο function και trigger, παρατηρούμε ότι ο πελάτης έχει καταχωρηθεί στους customers:



Το αμάξι έχει καταχωρηθεί στο car\_warehouse:



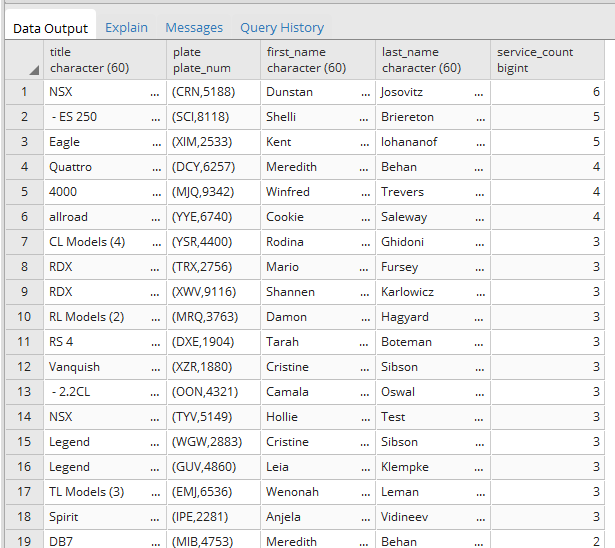
Και τέλος, εάν ο πελάτης έφερε το αμάξι αρκετές φορές, αυτό φαίνεται και στο service\_history:



## Cursor

Για τον cursor που μας ζητείται, υλοποιήσαμε το 6ο query της εργασίας. Όπως και στη προηγούμενη ενότητα για την επεξήγησή του, εμφανίζονται τα ίδια αποτελέσματα, δηλαδή αυτοκίνητα που ήρθαν για επισκευή πάνω από 1 φορά τον τελευταίο χρόνο.

Ουσιαστικά, φτιάχνουμε ένα temp\_type που θα χρησιμοποιηθεί στον cursor για να επιστρέψει ένα setof temp\_type. Με αυτό το τρόπο μπορούμε εύκολα να επιστρέψουμε τα ίδια αποτελέσματα όπως στο 6ο query, με τη χρήση του cursor.



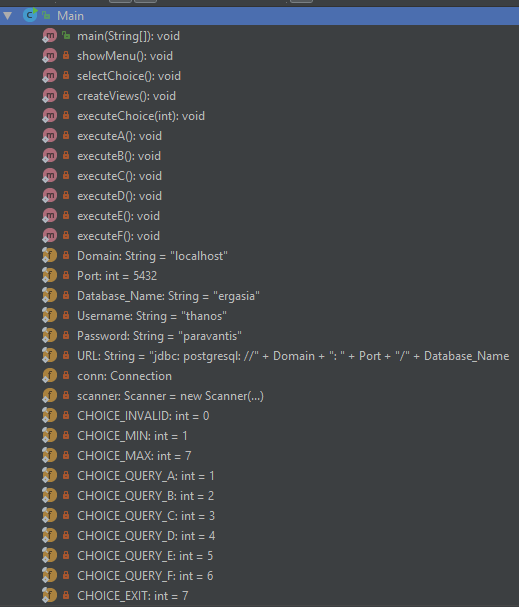
# Παράδειγμα σύνδεσης με JDBC

Το project βρίσκεται στον φάκελο **JDBC\_Example**.

## Δομή

Σύμφωνα με την εκφώνηση της άσκησης, υλοποιήσαμε σε Java ένα πρόγραμμα που χρησιμοποιεί το JDBC ώστε να συνδεθεί με τη βάση δεδομένων και να εκτελέσει ερωτήματα.

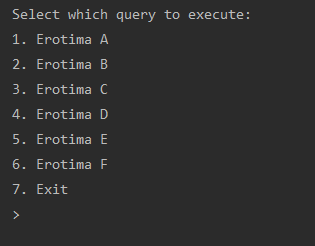
Αποτελείται από μόνο ένα αρχείο Main.java όπου έχει τις εξής μεθόδους και ιδιότητες:



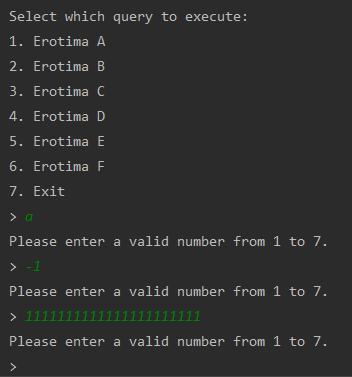
Η εφαρμογή εκκινείτε με τη κλήση της main() και στη συνέχεια η μέθοδος showMenu() καλείται για να εμφανίσει ένα μενού επιλογών. Κάθε φορά που γίνεται μια επιλογή από το χρήστη μέσω της κονσόλας, εκτελείται η κατάλληλη μέθοδος για να καλύψει το ερώτημα.

## Εκτέλεση

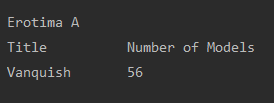
Με την εκτέλεση του προγράμματος εμφανίζεται το μενού επιλογών:



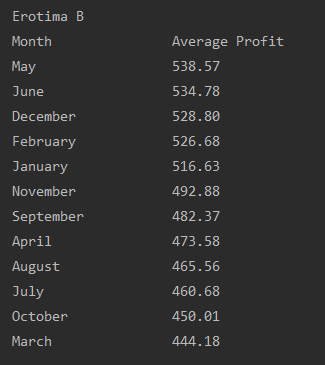
Ο χρήστης πρέπει να εισάγει ένα νούμερο από το 1 μέχρι το 7 για να εκτελεστεί η αντίστοιχη λειτουργία. Αφού εμφανιστούν τα αντίστοιχα αποτελέσματα, το μενού ξαναεμφανίζεται. Σημειώνεται ότι αν γίνει λάθος εισαγωγή αριθμού, εμφανίζονται αντίστοιχα μηνύματα λάθους.



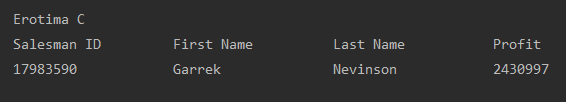
Αποτελέσματα εκτέλεση του ερωτήματος Α:



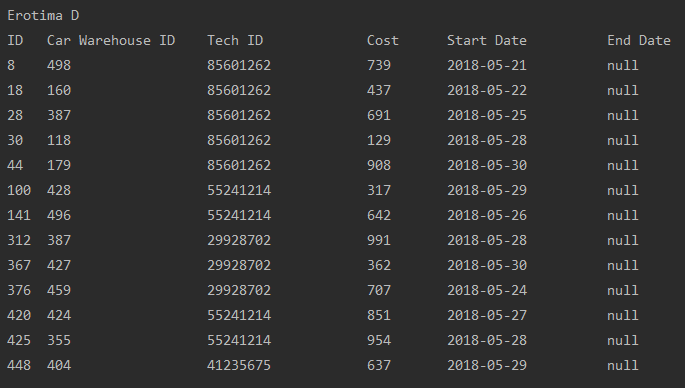
Αποτελέσματα εκτέλεσης του ερωτήματος Β:



Αποτελέσματα εκτέλεσης του ερωτήματος C:

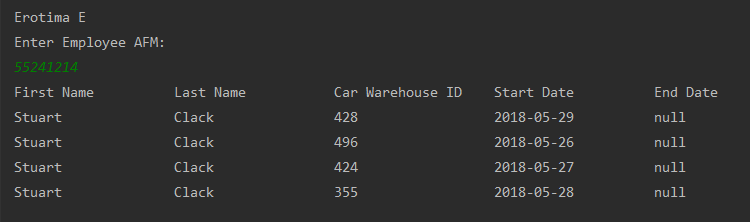


Αποτελέσματα εκτέλεσης του ερωτήματος D:



Στο ερώτημα Ε απαραίτητη προϋπόθεση είναι η εισαγωγή του ΑΦΜ για την εμφάνιση σχετικών εργασιών του τεχνικού. Εάν ο χρήστης πληκτρολογήσει χαρακτήρες τότε το ΑΦΜ 85601262 θα χρησιμοποιηθεί αυτόματα. Διαφορετικά, αν δεν υπάρχει τεχνικός με το επιλεγμένο ΑΦΜ τότε δεν εμφανίζονται αποτελέσματα.

Αποτελέσματα εκτέλεσης του ερωτήματος E:



Αποτελέσματα εκτέλεσης του ερωτήματος F:

