



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

«Ανάλυση του καλαθιού αγορών»

Εξαμηνιαία Εργασία

στο μάθημα «**Βάσεις Δεδομένων**»

των φοιτητών

Αχλάτης Στέφανος-Σταμάτης, Α.Μ.: 03116149

Τζαβάρα Νεφέλη-Παναγιώτα, Α.Μ.:03117046

Χριστοφίδη Γεωργία, Α.Μ.:03117015

Σημαντικό Σχόλιο:

Ο Αχλάτης Στέφανος-Σταμάτης υπέβαλε την εργασία στο MS Teams

Διδάσκοντες: Β. Καντερέ, Ν. Κοζύρης

Βοηθός Διδασκαλίας: Μ. Κρομμύδα

Αθήνα, Ιούλιος 2020

Περίληψη

Στην παρούσα εργασία υλοποιήθηκε το σύστημα βάσης δεδομένων μιας αλυσίδας Σούπερ Μάρκετ. Μέσω κατάλληλα διαμορφωμένου user interface ο χρήστης μπορεί να δει διάφορες πληροφορίες που σχετίζονται με την αλυσίδα όπως οι αγορές που έγιναν σε συγκεκριμένα καταστήματα συγκεκριμένες ημερομηνίες, αλλά και να εξάγει στατιστικά χαρακτηριστικά για την αλυσίδα, όπως το πιο δημοφιλή ζεύγη προϊόντων. Για το interface χρησιμοποιήθηκε HTML για το user side της ιστοσελίδας και PHP για το server side. Για την βάση δεδομένων χρησιμοποιήσαμε MySQL.

Πίνακας περιεχομένων

1	Σχεδιασμός Βάσης	4
1.1	Επεξήγηση του ER	5
1.2	Σχεσιακό Διάγραμμα Βάσης	6
2	Κώδικας SQL	7
2.1	Περιορισμοί Ακεραιότητας	7
2.2	Αναφορική Ακεραιότητα	7
2.3	Ρητοί Περιορισμοί - Περιορισμοί Πεδίου Τιμών	10
2.4	Επιπλέον Ρητοί Περιορισμοί	13
2.5	Εισαγωγή Dummy Data	14
3	Indexes	14
4	Queries	15
5	Triggers	30
6	Δικά μας ερωτήματα	33
7	Σύστημα και γλώσσες προγραμματισμού	34
8	Πίνακας περιεχομένων video	36

Σχεδιασμός Βάσης

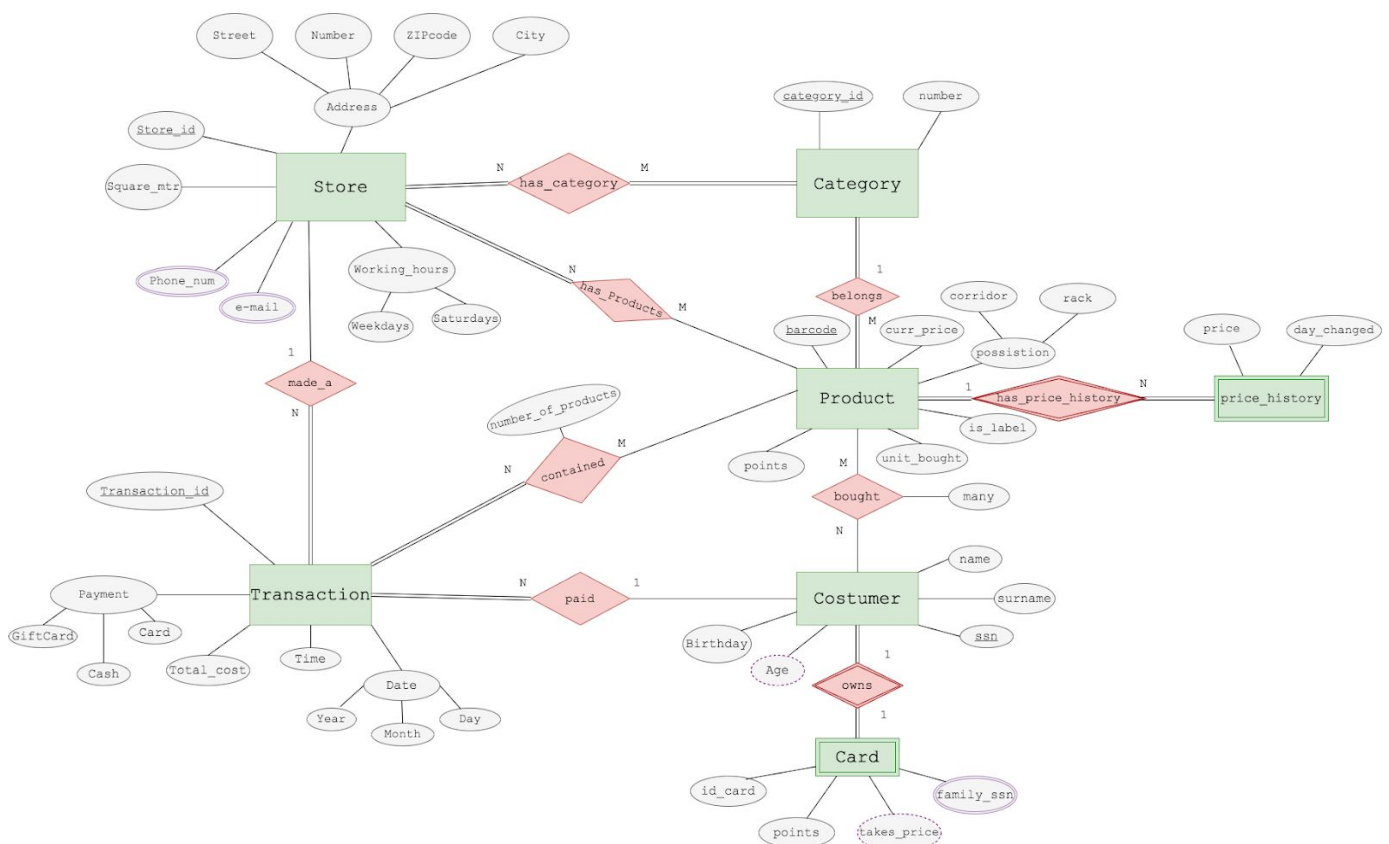
Το σύστημα πρέπει να αποθηκεύει πληροφορίες για τα καταστήματα της εταιρείας. Υπάρχουν συνολικά δέκα καταστήματα σε τρεις διαφορετικές πόλεις της Ελλάδας. Για τα καταστήματα αυτά μας ενδιαφέρουν οι ώρες λειτουργίας, η διεύθυνση, τα τετραγωνικά του καταστήματος και οι κατηγορίες προϊόντων που προσφέρονται. Προσφέρονται προϊόντα από έξι κατηγορίες: Φρέσκα προϊόντα, είδη ψυγείου, είδη κάβας, είδη προσωπικής περιποίησης, είδη σπιτιού και προϊόντα για κατοικίδια. Σε κάθε κατηγορία υπάρχουν από δέκα έως είκοσι προϊόντα.

Για κάθε προϊόν το σύστημα πρέπει να αποθηκεύει, την τρέχουσα τιμή του, τον διάδρομο του καταστήματος στο οποίο βρίσκεται, το ράφι στο οποίο βρίσκεται και μια ειδική σημείωση αν το προϊόν είναι ετικέτα του καταστήματος ή όχι. Το σύστημα πρέπει ακόμα να κρατάει ιστορικό με όλες τις αλλαγές τιμών των προϊόντων.

Κάθε πελάτης θα έχει την προσωπική του κάρτα με την οποία θα κάνει αγορές και θα κερδίζει επιβραβεύσεις, πιο συγκεκριμένα κάθε φορά που η καρτα συγκεντρώνει ένα συγκεκριμένο αριθμό πόντων κερδίζει μια δωροεπιταγή για μια επόμενη αγορά του. Το σύστημα θα πρέπει να αποθηκεύει προσωπικά και οικογενειακά στοιχεία των πελατών, καθώς και τον αριθμό της κάρτας που χρησιμοποιούν.

Για κάθε συναλλαγή του χρήστη θα πρέπει να καταγράφεται το συνολικό κόστος της αγοράς, το πλήθος και το είδος των προϊόντων που αγόρασε, η ώρα κατά την οποία έγινε η συναλλαγή, ο τρόπος πληρωμής.

Έτσι, το ER που περιγράφει το παραπάνω σύστημα βάσης δεδομένων είναι το εξής:



Επεξήγηση του ER

Αρχικά επιλέγουμε να κάνουμε οντότητα το Κατάστημα (Store), δεδομένου ότι σε πολλά queries στην συνέχεια θέλουμε να αναφερόμαστε στο εκάστοτε store.

Το store έχει ως primary key το attribute store_id με βάση το οποίο κάθε κατάστημα έχει ένα μοναδικό κωδικό με το οποίο μπορούμε μοναδικά να αναφερθούμε σε αυτό.

Το store έχει όλα τα ζητούμενα attributes και αξίζει να σημειωθεί ότι οι πληροφορίες σχετικά με την διεύθυνση του εκάστοτε Σούπερ Μάρκετ ομαδοποιήθηκαν σε ένα Composite Attribute με το όνομα Address, όπως αντίστοιχα το attribute με τις ώρες λειτουργίας στις καθημερινές και τα σαββατοκύριακα. Επίσης, τα attributes Phone_num και e-mail είναι multivalued attributes, γιατί μπορούν να λάβουν περισσότερα του ενός τιμές.

Τα καταστήματα έχουν οσοδήποτε πλήθος κατηγοριών προϊόντων και αυτό φαίνεται από τη σχέση has_category. Η κάθε κατηγορία έχει ένα primary key στο οποίο αναφερόμαστε και ένα attribute το οποίο περιέχει τον αριθμό των προϊόντων που ανήκουν σε αυτή την κατηγορία. Επιλέχθηκε το category να είναι οντότητα και όχι χαρακτηριστικό του product γιατί θέλουμε να βλέπουμε ποιου είδους προϊόντα έχει το κάθε κατάστημα άμεσα, και να έχουμε ομαδοποιημένα τα προϊόντα σε κατηγορίες.

Κάθε Προϊόν (Product) ανήκει σε μία Category και έχει ως primary key το μοναδικό barcode του. Επίσης για διευκόλυνση σε ορισμένα queries επιλέχθηκε το store και το product να συνδέονται μέσω της σχέσης has_product, που δηλώνει ότι το εκάστοτε κατάστημα έχει κάποια συγκεκριμένα προϊόντα.

Δεδομένου ότι το Προϊόν μπορεί να μην έχει ιστορικό αλλαγών τιμής, π.χ. το προϊόν να είναι νέο στο κατάστημα ή να έχει επιβληθεί διατίμηση, επιλέχθηκε η σχέση που συνδέει το προϊόν με το price_history του να είναι weak όπως και η οντότητα του price_history, που αποθηκεύει την τιμή και την ημερομηνία αλλαγής της.

Η οντότητα Customer έχει όλα τα απαραίτητα attributes με primary key το ssn του πελάτη και αξίζει να σημειωθεί ότι το attribute age είναι παραγόμενο από το attribute που περιέχει την ημερομηνία γεννήσεως του πελάτη.

Στην συνέχεια η οντότητα Customer συνδέεται μέσω της σχέσης bought με τα προϊόντα που έχει αγοράσει, και μέσω μιας αδύναμης σχέσης owns με την αδύναμη οντότητα card. Ωστόσο, στην συνέχεια, αναθεωρήσαμε και πλέον ορίζουμε ότι ο κάθε πελάτης πρέπει να έχει υποχρεωτικά κάρτα, επομένως στο ανανεωμένο ER τόσο η σχέση owns όσο και η οντότητα κάρτα θα είναι ισχυρές. Η κάρτα έχει όλα τα απαιτούμενα attributes και ένα multivalued attribute στο οποίο αποθηκεύονται όλοι οι ssn αριθμοί που σχετίζονται με αυτή την κάρτα.

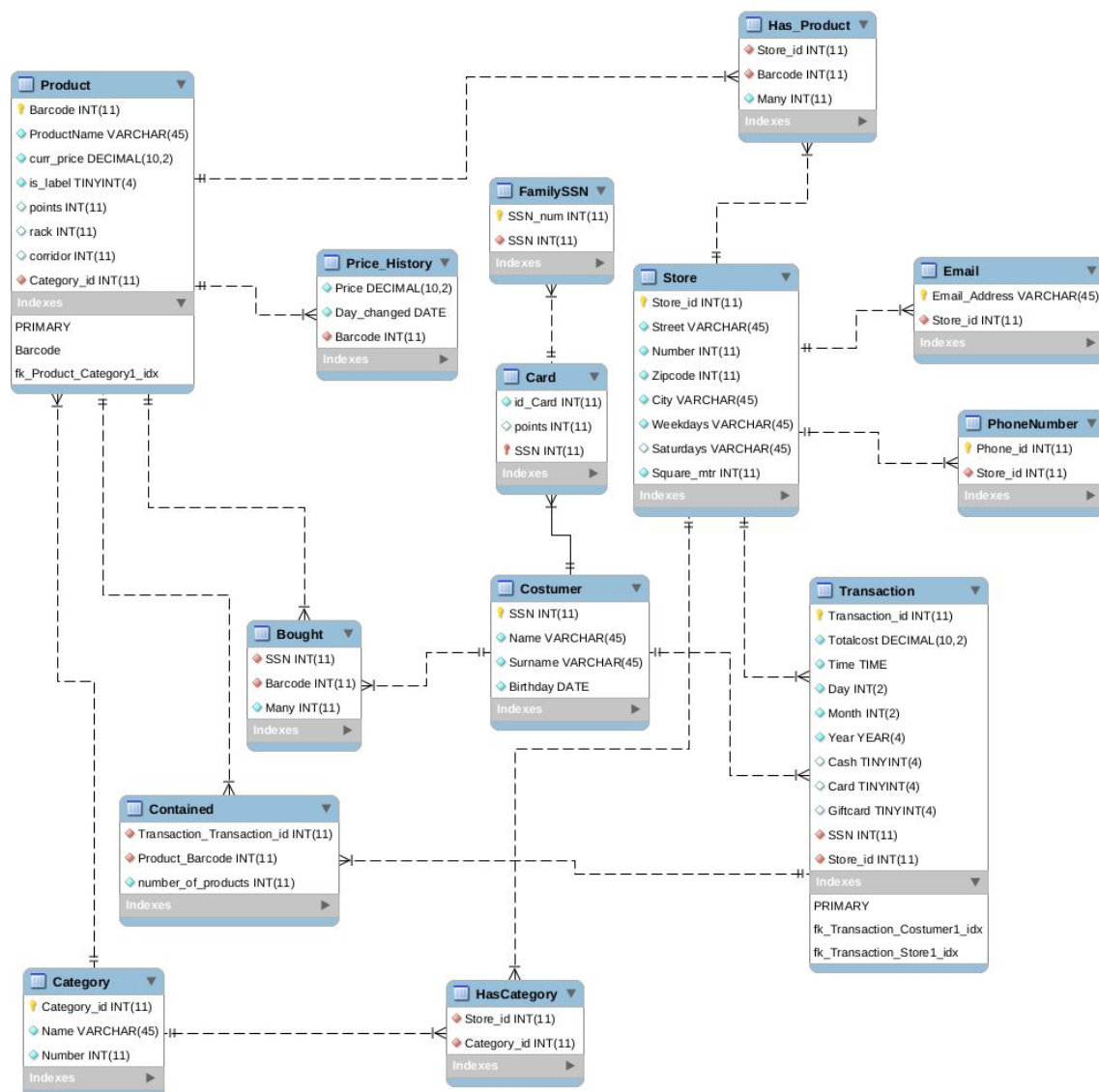
Ο κάθε πελάτης έχει πληρώσει (paid) πολλές συναλλαγές Transactions, που χαρακτηρίζονται από το transaction_id (primary key) που επομένως είναι μοναδικό, και τα υπόλοιπα attributes είναι όσα ζητούνται, ορισμένα από τα οποία είναι ιεραρχικά δομημένα. Έτσι, κάθε κατάστημα έκανε (made_a) οσοδήποτε πολλές συναλλαγές και η κάθε συναλλαγή περιείχε (contained) έναν αριθμό προϊόντων. Ο αριθμός προϊόντων που περιέχει η κάθε συναλλαγή παρουσιάζεται ως attribute στην σχέση contained.

Επομένως, ανακεφαλαιώνοντας, για ισχυρές οντότητες επιλέχθηκαν όσα ουσιαστικά υπάρχουν σίγουρα στο σύστημα και διευκολύνουν την περιγραφή της βάσης, ενώ ως ασθενείς όσα ουσιαστικά δεν υπάρχουν σίγουρα στο σύστημα και εξαρτώνται από άλλες, ισχυρές.

Σαν σχέσεις επιλέχθηκαν τα ρήματα που συνδέουν τις οντότητες μεταξύ τους και ως ασθενείς σχέσεις ορίστηκαν όσες συνδέουν μια οντότητα με μία αδύναμη οντότητα.

Σχεσιακό Διάγραμμα Βάσης

Με χρήση του MySQL Workbench σχεδιάστηκε το σχεσιακό διάγραμμα της βάσης, σύμφωνα με το E-R διάγραμμα το οποίο είχαμε υποβάλει, στο οποίο πραγματοποιήθηκαν ορισμένες τροποποιήσεις με σκοπό να εξυπηρετούνται καλύτερα οι σκοποί της βάσης. Τα ονόματα των tables προκύπτουν άμεσα από το ER, όπου πίνακες αποτελούν οι οντότητες, οι σχέσεις NxM, όπως και τα multivalued attributes.



Κώδικας SQL

Ο κώδικας SQL υλοποιεί τη ΒΔ. Κάθε table περιέχει όλα τα attributes βάσει του σχεσιακού, τα primary keys καθώς και τα foreign keys μαζί με τους τύπους δεδομένων τους και τους περιορισμούς τους.

```
-- -----  
-- Table `mydb`.`Category`  
-- -----  
  
• DROP TABLE IF EXISTS `mydb`.`Category` ;  
• CREATE TABLE `mydb`.`Category` (  
  `Category_id` INT(11) NOT NULL AUTO_INCREMENT,  
  `Name` VARCHAR(45) NOT NULL,  
  `Number` INT(11) NOT NULL,  
  PRIMARY KEY (`Category_id`))  
ENGINE = InnoDB;
```

Για το Category, παραδείγματος χάριν βλέπουμε ως INT που είχαμε ορίσει το id της κατηγορίας (1,2,3,4,5,6) , το αλφαριθμητικό για το Name και πάλι INT για τον αριθμό των προϊόντων που είναι σε κάθε κατηγορία. Το id της κατηγορίας προφανώς δεν μπορεί να είναι null και πρέπει να αυξάνεται αυτόματα με την εισαγωγή νέας κατηγορίας.

Αυτό, βέβαια, είναι ένα παράδειγμα και ακολουθούν πιο αναλυτικά οι περιορισμοί ακεραιότητας.

Περιορισμοί Ακεραιότητας

Αναφορική Ακεραιότητα

Για την διατήρηση της αναφορικής ακεραιότητας θέσαμε κάποιους περιορισμούς στα εξωτερικά κλειδιά. Δηλαδή, μετά από την αλλαγή ή διαγραφή εγγραφής από κάποιο πίνακα ανάλογα με τις ενέργειες που θέσαμε στα εξωτερικά κλειδιά καθορίζουμε αν θα επηρεαστούν και οι υπόλοιποι πίνακες.

πίνακας Product:

- **Category_id (fk_Product_Category1)**
 - ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός προϊόντος πρέπει να διαγράφεται και από την κατηγορία που ανήκει
 - ON UPDATE CASCADE, δηλαδή όταν αλλάξει η κατηγορία κάποιου προϊόντος πρέπει να ενημερώνεται ο πίνακας με το νέο Category_id

πίνακας Transaction:

- **SSN (fk_Transaction_Costumer1)**
 - ON DELETE NO ACTION, δηλαδή εάν ο πελάτης διαγραφεί από τη βάση, οι συναλλαγές με το SSN του θα παραμείνουν
 - ON UPDATE CASCADE, δηλαδή κατά την αλλαγή του SSN του πελάτη θα πρέπει να ενημερωθεί σε όλες του τις συναλλαγές
- **Store_id (fk_Transaction_Store1)**
 - ON DELETE SET NULL, δηλαδή κατά τη διαγραφή ενός καταστήματος από τη βάση θα έχουμε τις παλιές συναλλαγές αυτού του καταστήματος
 - ON UPDATE CASCADE, δηλαδή κατά την αλλαγή του id του καταστήματος θέλουμε σε κάθε συναλλαγή του να είναι αλλαγμένο.

πίνακας Card

- **SSN (fk_Card_Costumer1)**
 - ON DELETE CASCADE, δηλαδή εάν ο πελάτης διαγραφεί από τη βάση πρέπει να διαγραφεί το SSN από τη κάρτα του
 - ON UPDATE CASCADE, δηλαδή κατά την αλλαγή του SSN του πελάτη θα πρέπει να ενημερωθεί η κάρτα του

πίνακας PhoneNumber

- **Store_id (fk_PhoneNumber_Store1)**
 - ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός καταστήματος σταματάει να υφίσταται και ο τηλεφωνικός του αριθμός
 - ON UPDATE CASCADE, δηλαδή κατά την αλλαγή ενός καταστήματος το αναμενόμενο είναι να αλλάξει και το τηλέφωνό του π.χ. μετακίνηση οπότε πρέπει αυτή η αλλαγή να μεταφερθεί και στον πίνακα PhoneNumber

πίνακας Email

- **Store_id (fk_Email_Store)**
 - ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός καταστήματος σταματάει να υφίσταται και η διεύθυνση ηλ. ταχυδρομείου του
 - ON UPDATE CASCADE, δηλαδή κατά την αλλαγή του id του καταστήματος η νέα διεύθυνση ηλ.ταχυδρομείου πρέπει να ενημερωθεί .

πίνακας Price_History

- **Barcode (fk_Price_History_Product1)**
 - ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός Barcode σταματάει να υφίσταται το προϊόν τόσο στη βάση αλλά και στο ιστορικό τιμών που είχε
 - ON UPDATE CASCADE, δηλαδή κατά την αλλαγή του Barcode ενός προϊόντος πρέπει να αλλάξει και στο ιστορικό ώστε να μπορούμε να αναφερθούμε σε αυτό το προϊόν

πίνακας Contained

- **Transaction_id (fk_Contained_Transaction1)**
 - ON DELETE CASCADE, δηλαδή κατά την διαγραφή μιας συναλλαγής αυτή πρέπει να διαγράφεται και από τη σχέση που περιγράφει ποιά προϊόντα έχουν υπάρξει σε συναλλαγές
 - ON UPDATE CASCADE, δηλαδή κατά την αλλαγή του id μιάς συναλλαγής πρέπει αυτή να μεταφέρεται και στον πίνακα Contained
- **Barcode (fk_Contained_Product1)**
 - ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός προϊόντος διαγράφεται και από τη σχέση Contained
 - ON UPDATE CASCADE, δηλαδή κατά την ενημέρωση ενημερώνουμε και τον εν λόγω πίνακα αλλιώς δεν θα είχε νόημα το table της συσχέτισης για αυτό το προϊόν

πίνακας HasCategory

- **Store_id (fk_HasCategory_Store1)**
 - ON DELETE CASCADE, δηλαδή όταν διαγράφεται ένα κατάστημα θα διαγραφεί και από τη συσχέτιση με το ποιές κατηγορίες προϊόντων διαθέτει
 - ON UPDATE CASCADE, δηλαδή όταν αλλάζει το κατάστημα πρέπει να ενημερωθεί το id στην σχέση HasCategory για να μπορούμε να δούμε ποιες κατηγορίες προϊόντων έχει
- **Category_id (fk_HasCategory_Category1)**
 - ON DELETE CASCADE, δηλαδή όταν διαγραφεί μια κατηγορία αυτή πρέπει να διαγραφεί και από τη σχέση
 - ON UPDATE CASCADE, δηλαδή αντιστοίχως πρέπει να ενημερώσουμε τη σχέση κατά την ενημέρωση των κατηγοριών

πίνακας Bought

- **SSN (fk_Bought_Costumer1)**
 - ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός πελάτη από τη βάση, διαγράφουμε και τον ίδιο απο τη σχέση Bought, που περιγράφει τα προϊόντα που έχει αγοράσει αυτός ο πελάτης.
 - ON UPDATE CASCADE, δηλαδή όταν αλλάζει το SSN του πελάτη ενημερώνουμε και τον πίνακα με τις αγορές του
- **Barcode (fk_Bought_Product1)**
 - ON DELETE CASCADE
 - ON UPDATE CASCADE

Και τα δύο είναι σε πλήρη αντιστοιχία με το SSN απλώς αντί για Πελάτη διαγράφουμε/ενημερώνουμε το προϊόν

πίνακας Family SSN

- **SNN(fk_Bought_Product1)**
 - ➔ ON DELETE CASCADE, δηλαδή όταν διαγράφεται το SSN του πελάτη δεν υφίστανται και τα οικογενειακά του στοιχεία
 - ➔ ON UPDATE CASCADE, δηλαδή ενημερώνεται ο πίνακας με τους συγγενείς του πελάτη

πίνακας HasProduct

- **Store_id (fk_Has_Product_Store1)**
 - ➔ ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός καταστήματος από τη βάση διαγράφεται και η σχέση που περιγράφει τα προϊόντα που έχει
 - ➔ ON UPDATE CASCADE, δηλαδή ενημερώνεται ο πίνακας με τα προϊόντα που διαθέτει κάθε κατάστημα ύστερα από αλλαγή του καταστήματος
- **Barcode (fk_Has_Product_Product1)**
 - ➔ ON DELETE CASCADE, δηλαδή κατά τη διαγραφή ενός προϊόντος από τη βάση διαγράφεται και από τη σχέση αυτή
 - ➔ ON UPDATE CASCADE, δηλαδή ενημερώνεται ο πίνακας με τα προϊόντα που διαθέτει κάθε κατάστημα ύστερα από αλλαγή του προϊόντος

Ακεραιότητα Οντότητας

Για να εξασφαλίσουμε την ακεραιότητα οντότητας όλα τα πεδία που επιλέχθηκαν ως πρωτεύοντα κλειδιά δεν μπορούν να έχουν NULL τιμές.

Ρητοί Περιορισμοί-Περιορισμοί Πεδίου Τιμών

Πιο κάτω παρουσιάζονται οι πίνακες που υλοποιήσαμε όπου,

PK: Primary key

NN: Not Null

AI: Auto Increment

[illegible]

Store - Table																		
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges										
Column Name									Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Store_id									INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Street									VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Number									INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Zipcode									INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
City									VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Weekdays									VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Saturdays									VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		NULL
Square_mtr									INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
										<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Transaction - Table																		
Table	Columns	Indexes	Foreign Keys	Triggers	Partitioning	Options	Inserts	Privileges										
Column Name									Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Transaction_id									INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Totalcost									DECIMAL(10,2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Time									TIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Day									INT(2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Month									INT(2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Year									YEAR(4)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Cash									TINYINT(4)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		NULL
Card									TINYINT(4)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		NULL
Giftcard									TINYINT(4)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		NULL
SSN									INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Store_id									INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
										<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Οι περισσότερες τιμές των attributes έχουν το πεδίο Not Null πλήρες αφού στις περισσότερες σχέσεις θα ήταν απαγορευτικό να βρίσκουμε null στοιχεία. π.χ. το κάθε προϊόν θα έχει μια συγκεκριμένη τιμή και barcode, που δεν θα μπορούσε ποτέ να είναι απροσδιόριστη, ενώ ο διάδρομος και το ράφι θα μπορούσε να είναι, αφού θα μπορούσε να μην έχει τοποθετηθεί ακόμα/να βρίσκεται στην αποθήκη του καταστήματος.

Επιπλέον Ρητοί Περιορισμοί

Σε κάθε ερώτημα που γίνεται μέσω της ρηρ ελέγχουμε αν ο χρήστης έχει βάλει σημασιολογικά ορθή είσοδο, παραδείγματος χάριν.

Εδώ βλέπουμε τον σχετικό έλεγχο που κάνει η ρηρ:

```
$query = "SELECT *
          FROM Transaction
          WHERE Store_id = $search1 AND Day = $search2 AND Month = $search3 AND Year = $search4";

$result = mysqli_query($con,$query);
if($search1 < 0 || $search1 > 10 || $search2 < 0 || $search2 > 31 ||
    $search3 < 0 || $search3 > 12 || $search4 < 2015 || $search4 > 2020){
    echo "<div style ='color:white'> Λάθος είσοδος!</div>";
}

else if(!$result || mysqli_num_rows($result)==0 ) {
    echo "<div style ='color:white'>Δεν βρέθηκε καμία αγορά για την είσοδο που δώσατε!</div>";
}
```

Και στο UI αν δοθεί μια σημασιολογικά μη ορθή είσοδος όπως η εξής:

Αγορές

Καταστήματος Αγοράς(1-10):

Ημέρα της Αγοράς (1-31):

Μήνας της Αγοράς(1-12):

Χρονία της Αγοράς(2015-2020):

Το αποτέλεσμα θα είναι:

Αγορές

Λάθος είσοδος!

[Back](#)

Εισαγωγή Dummy Data

Τα δεδομένα που τοποθετήθηκαν ήταν ορθά συνδεδεμένα μεταξύ του, πχ ένας πελάτης που ψώνισε από ένα συγκεκριμένο κατάστημα θα μπορούσε να είχε αγοράσει μόνο προϊόντα που βρίσκονται στο κατάστημα, ή η συνολική τιμή των προϊόντων προκύπτει από το άθροισμα των επιμέρους γινομένων τον ποσοτήτων με τα current κόστη των προϊόντων που αγόρασε.

Επίσης τα δεδομένα ήταν αρκετά έτσι ώστε τα queries να επέστρεφαν μη κενές απαντήσεις και να αναδειχθεί μέσω των απαντήσεων και των δεδομένων η ορθότητα των queries.

```
LOCK TABLES `mydb`.`Transaction` WRITE;
INSERT INTO `mydb`.`Transaction` VALUES
(1,37.70,'10:45',7,2,2020,0,1,0,21314155,1),(2,63.6,'11:05',7,3,2019,0,1,0,21314151,1),(3,24.6,'11:05',7,3,2020,0,1,0,21314151,1),
(4,39.6,'13:15',5,2,2019,1,1,0,21314156,1),(5,17.9,'11:05',9,4,2019,1,0,0,21314155,1),(6,64,'13:05',10,4,2020,0,1,0,21314151,1),
(7,24.5,'11:45',9,2,2020,1,0,1,21314155,1),(8,70.7,'11:55',11,7,2019,1,0,1,21314155,1),
(9,20.2,'11:30',29,3,2020,0,1,0,21314154,2),(10,40,'10:30',30,4,2020,0,1,0,21314154,2),(11,71.5,'10:30',19,5,2020,1,0,0,21314151,2),
(12,26.9,'12:30',29,3,2020,0,1,0,21314151,2),(13,14.6,'12:38',13,2,2019,1,1,0,21314156,2),(14,45.5,'10:30',21,3,2019,0,1,0,21314153,2),
(15,48.1,'10:45',17,2,2020,0,1,0,21314152,3),(16,40.1,'11:05',27,3,2020,0,1,0,21314152,3),(17,35.7,'11:05',6,4,2020,0,1,0,21314152,3),
(18,69.0,'13:15',15,2,2020,1,1,0,21314152,3),(19,34.5,'11:05',10,4,2020,1,0,0,21314155,3),(20,17.8,'13:05',10,4,2020,0,1,0,21314152,3),
(21,27.5,'11:45',19,2,2020,1,0,1,21314156,3),(22,19.4,'11:55',10,7,2019,1,0,1,21314151,3),
(23,53.3,'11:05',5,5,2020,0,1,1,21314152,4),(24,13.4,'13:05',10,4,2020,0,1,0,21314153,4),
(25,4.5,'11:45',9,5,2020,1,0,1,21314151,4),(26,16.5,'11:55',11,7,2020,1,0,1,21314152,4),
(27,27.9,'16:05',9,4,2019,1,0,0,21314154,5),(28,16.8,'18:05',10,10,2020,0,1,0,21314154,5),
(29,5.3,'15:45',12,6,2020,1,0,1,21314155,5),(30,22,'20:55',11,8,2020,1,1,1,21314154,5),
(31,14.2,'17:45',11,8,2020,1,1,1,21314151,6),(32,27.0,'18:55',11,8,2020,1,1,1,21314152,6),
(33,22.4,'8:45',11,7,2020,1,0,1,21314155,7),(34,18.0,'9:55',11,8,2019,1,1,1,21314154,7),
(35,34.1,'8:15',12,6,2019,0,1,1,21314156,8),(36,32.3,'11:55',11,8,2020,1,1,1,21314153,8),
(37,7.0,'8:17',12,6,2020,0,1,1,21314155,8),(38,22.0,'12:55',11,8,2020,1,1,1,21314153,8),
(39,6.7,'11:45',7,1,2019,0,1,0,21314155,9),(40,54.2,'15:05',7,1,2020,0,1,1,21314151,9),(41,15.5,'11:05',9,3,2020,0,1,0,21314151,9),
(42,22.0,'13:15',5,1,2020,1,1,0,21314156,9),(43,38.0,'13:05',9,1,2019,1,0,1,21314155,9),(44,26.5,'13:05',11,4,2020,1,1,0,21314151,9),
(45,4.2,'12:45',5,1,2020,1,0,1,21314155,9),(46,10.6,'14:55',11,3,2020,1,0,1,21314155,9),
(47,69.0,'15:51',10,12,2019,1,0,0,21314153,10);
UNLOCK TABLES;
```

Ένα παράδειγμα είναι το προηγούμενο, όπου φαίνεται ότι στο σύστημα μας έγιναν 47 συναλλαγές και από δύο και πάνω συναλλαγές στο κάθε κατάστημα.

Μάλιστα, με μια πιο προσεκτική ματιά ευκολα φαίνεται ότι αυτά τα δεδομένα νοηματοδοτούν τα queries που δίνονται στην συνέχεια, παραδείγματος χάρι τις πιο δημοφιλείς ώρες για κάθε κατάστημα, τον καλύτερο πελάτη για κάθε κατάστημα κλπ.

Indexes

Κάθε φορά που σε μία οντότητα/σχέση ορίζουμε ένα primary key, η SQL δημιουργεί αυτόματα ένα αντίστοιχο index - ευρετήριο, ώστε να είναι δυνατή στη συνέχεια η αναζήτηση στο συγκεκριμένο πίνακα. Τα ευρετήρια (indexes) χρησιμοποιούνται για την βελτιστοποίηση της απόδοσης της ΒΔ και την μείωση των απαιτήσεων του web server. Ορίσαμε τα εξής indexes:

```
-- indexes for the forth Part
CREATE INDEX idx_transaction_ssn
ON `mydb`.`Transaction` (SSN);

CREATE INDEX idx_Contained_transaction_id
ON `mydb`.`Contained` (Transaction_Transaction_id);

CREATE INDEX idx_Contained_Product_Barcode
ON `mydb`.`Contained` (Product_Barcode);
-- end of indexes
```

Επιλέχθηκε να οριστεί ως index της transaction το ssn του πελάτη που κάνει την αγορά, επειδή είναι λογικό πέρα από το κωδικό της συναλλαγής να θέλουμε να μάθουμε τον πελάτη που έκανε αυτή την συναλλαγή. Επίσης, στην σχέση Contained ορίσαμε ως ευρετήρια το κωδικό της συναλλαγής και το barcode, επειδή είναι πιθανό να χρειάζεται να ανακαλούνται συχνά, οπότε θέλουμε γρήγορη επίδοση στην εύρεση τους.

Queries

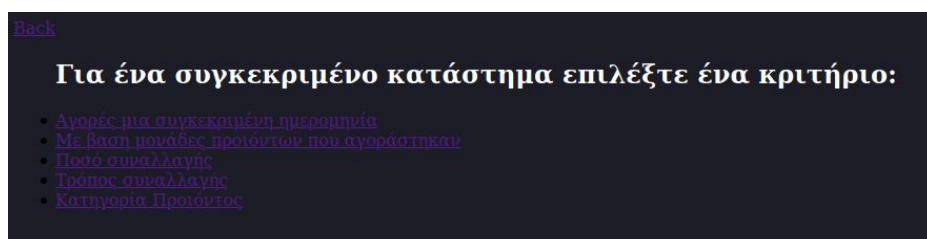
Σε αυτό το σημείο θα παρουσιαστούν όλα τα queries που χρησιμοποιούμε στο σύστημα μας. Επειδή τα περισσότερα queries είναι παραμετροποιήσιμα δηλαδή πρέπει ο χρήστης να βάλει τις ζητούμενες πληροφορίες για να κάνει το κατάλληλο ερώτημα, πχ να ρωτήσει για τα προϊόντα ενός συγκεκριμένου μαγαζιού, επιλέγουμε να παρουσιάσουμε τα ερωτήματα όπως γράφτηκαν στην php.

Ερώτημα 6α

Για τα ερωτήματα αυτής της ομάδας ερωτημάτων ο χρήστης από την αρχική σελίδα της εφαρμογής μας πρέπει να επιλέξει να επιλέξει το “Αγορές”



Και στην συνέχεια για κάθε ερώτημα επιλέγει με την σειρά τα εξής:



Σαν πρώτο ερώτημα ο χρήστης επιλέγει το πρώτο bullet και μετά πρέπει να επιλέξει το κατάστημα που τον ενδιαφέρει και την ημερομηνία που θέλει να δει τι αγορές έγιναν στο εν λόγω κατάστημα.

Το ερώτημα είναι το εξής:

```
$query = "SELECT *
FROM Transaction
WHERE Store_id = $search1 AND Day = $search2 AND Month = $search3 AND Year = $search4";
```

Έτσι, στο UI βλέπουμε ότι ο χρήστης πληκτρολογεί τις παραμέτρους του store_id και της ημερομηνίας της αγοράς.

Για την διευκόλυνση του ο χρήστης μπορεί να δει την αντιστοιχία του store_id με το κατάστημα που αναζητά, αυτό έγινε πάλι με χρήση query.

Αγορές

Καταστήματος Αγοράς(1-10):

Ημέρα της Αγοράς (1-31):

Μήνας της Αγοράς(1-12):

Χρονία της Αγοράς(2015-2020):

Mapping για ID Καταστημάτων Μασούτης:

Store ID	Street	Number	City
1	Vilara	44	Ioannina
2	Skoufa	42	Ioannina
3	Xarilaou Trikoupi	12	Ioannina
4	Traulantwni	12	Athina
5	Psaroudaki	13	Athina
6	Voukourestiou	17	Athina
7	Vlaxava	30	Thessaloniki
8	Saadi Levi	8	Thessaloniki
9	Notara	37	Thessaloniki
10	Elenis Zografou	6	Ioannina

Το αποτέλεσμα της αναζήτησης είναι το εξής:

Αγορές	
Transaction ID	Total Cost
47	69.00

Προφανώς αν ο χρήστης δώσει λάθος είσοδο υπάρχει κατάλληλο μήνυμα που θα δώσει η php και δεν θα γίνει κάποιο ερώτημα στη βάση, ενώ αν στην βάση δεν βρεθεί καμία

αγορά την συγκεκριμένη ημερομηνία στο συγκεκριμένο κατάστημα η ρηρ θα επιστρέψει μήνυμα που θα ενημερώνει τον χρήστη ότι δεν βρέθηκε τέτοια είσοδο.

Σαν επόμενο ερώτημα ο χρήστης πρέπει να επιλέξει το κατάστημα που τον ενδιαφέρει και οι μονάδες προϊόντων που αγοράστηκαν και περιμένει να δει όλες τις συναλλαγές που πληρούν αυτά τα κριτήρια.

Το ερώτημα είναι το εξής:

```
$query = "SELECT DISTINCT `Transaction_id`  
FROM Contained,Transaction  
WHERE  
Contained.number_of_products = $search2 AND  
Transaction.Store_id = $search1";
```

Με παράδειγμα εφαρμογής στο UI το εξής:

Αγορές

ID Καταστήματος Αγοράς(1-10):

Αριθμός προϊόντων:

Mapping για ID Καταστημάτων Μασσούτης:

Store ID	Street	Number	City
1	Vilara	44	Ioannina
2	Skoufa	42	Ioannina
3	Xarilaou Trikoupi	12	Ioannina
4	Traulantwni	12	Athina
5	Psaroudaki	13	Athina
6	Voukourestiou	17	Athina
7	Vlaxava	30	Thessaloniki
8	Saadi Levi	8	Thessaloniki
9	Notara	37	Thessaloniki
10	Elenis Zografou	6	Ioannina

Και με αποτέλεσμα το εξής πίνακα:

Αγορές

Transaction ID
1
2
3
4
5
6
7
8

Σαν επόμενο ερώτημα ο χρήστης πρέπει να επιλέξει το κατάστημα που τον ενδιαφέρει και το ποσό συναλλαγής και περιμένει να δει όλες τις συναλλαγές που πληρούν αυτά τα κριτήρια.

Το ερώτημα είναι το εξής:

```
$query = "SELECT DISTINCT Transaction_id
FROM Transaction
WHERE
Totalcost = $search2 AND
Store_id = $search1";
```

Με παράδειγμα εφαρμογής στο UI το εξής:

Αγορές

ID Καταστήματος Αγοράς(1-10):

Συνολικό κόστος συναλλαγής:

Mapping για ID Καταστημάτων Μασούτης:

Store ID	Street	Number	City
1	Vilara	44	Ioannina
2	Skoufa	42	Ioannina
3	Xarilaou Trikoupi	12	Ioannina
4	Traulantwni	12	Athina
5	Psaroudaki	13	Athina
6	Voukourestiou	17	Athina
7	Vlaxava	30	Thessaloniki
8	Saadi Levi	8	Thessaloniki
9	Notara	37	Thessaloniki
10	Elenis Zografou	6	Ioannina

Και με αποτέλεσμα το εξής πίνακα:

Αγορές

Transaction ID

47

Σαν επόμενο ερώτημα ο χρήστης πρέπει να επιλέξει το κατάστημα που τον ενδιαφέρει και τον τρόπο που έγινε η συναλλαγή πχ με μετρητά ή κατα και περιμένει να δει όλες τις συναλλαγές που πληρούν αυτά τα κριτήρια.

Αν ο χρήστης επιλέγει να κάνει την συναλλαγή με μετρητά η ρηρ καλεί το πρώτο ερώτημα ενώ αν επιλέξει να κάνει την συναλλαγή με κάρτα καλεί το δεύτερο:

```
$query0 = "SELECT DISTINCT Transaction_id
FROM Transaction
WHERE
Store_id = $search1 AND Cash = 1";
```

```
$query1 = "SELECT DISTINCT Transaction_id
FROM Transaction
WHERE
Store_id = $search1 AND Card = 1";
```

Με παράδειγμα εφαρμογής στο UI το εξής:

Αγορές

ID Καταστήματος Αγοράς(1-10):

Τρόπος πληρωμής:

Για τον τρόπο πληρωμής:
Πληκτρολογήστε 1 για Μετρητά
Πληκτρολογήστε 2 για Κάρτα

Mapping για ID Καταστημάτων Μασούτης:

Store ID	Street	Number	City
1	Vilara	44	Ioannina
2	Skoufa	42	Ioannina

Με αποτέλεσμα το εξής:

Αγορές

Transaction ID

25

26

Σαν επόμενο ερώτημα ο χρήστης πρέπει να επιλέξει το κατάστημα που τον ενδιαφέρει και την κατηγορία προϊόντων που περιλάμβανε η εκάστοτε αγορά και περιμένει να δει όλες τις συναλλαγές που πληρούν αυτά τα κριτήρια.
Το ερώτημα είναι το εξής:

```
$query = "SELECT DISTINCT Transaction_id
FROM Transaction, Product
WHERE
Transaction.Store_id = $search1 AND
Product.Category_id = $search2";
```

Με παράδειγμα εφαρμογής στο UI το εξής:

Αγορές

ID Καταστήματος Αγοράς (1-10):

ID Κατηγορίας (1-6):

Mapping για ID Καταστημάτων και Κατηγορίας Προϊόντων:

Store ID	Street	Number	City
1	Vilara	44	Ioannina
2	Skoufa	42	Ioannina
3	Xarilaou Trikoupi	12	Ioannina
4	Traulantwni	12	Athina
5	Psaroudaki	13	Athina
6	Voukourestiou	17	Athina
7	Vlaxava	30	Thessaloniki
8	Saadi Levi	8	Thessaloniki
9	Notara	37	Thessaloniki
10	Elenis Zografou	6	Ioannina

Category ID	Name
1	Fresh Products
2	Fridge Products
3	Liquor Products
4	Hygiene Products
5	Home Decor
6	Pet Products

και αποτέλεσμα το εξής:


Αγορές

Transaction ID
35
36
37
38

Ερώτημα 6b

Για τα ερωτήματα αυτής της ομάδας ο χρήστης από την αρχική σελίδα της εφαρμογής μας πρέπει να επιλέξει να επιλέξει το “Προφίλ Πελατών”

Σουπερμάρκετ Μασούτης



Τι θα θέλατε να δείτε;

- [Αγορές](#)
- [Προφίλ Πελατών](#)
- [Συμπροσάρματα Μαρκετινγκ](#)
- [Χρήση Όψων](#)
- [Αλλαγές Πληροφοριών](#)
- [Δικά μας Ερωτήματα](#)
- [Τα προϊόντα μας](#)

Σε αυτό το κομμάτι ο χρήστης πρέπει να επιλέξει έναν συγκεκριμένο χρήστη και το σύστημα πρέπει να βρει ορισμένα χαρακτηριστικά και πληροφορίες για τον χρήστη:

Προφίλ Πελατών

SSN Πελάτη (21314151-21314156):

Mapping για SSN πελατών:

SSN	Name	Surname
21314151	Selim	Selguk
21314152	Manwlis	Sarrhs
21314153	Sotiris	Kontizas
21314154	Theios	Leonidas
21314155	Panagiotis	Iliaras
21314156	Alexis	Barmperos

Και τότε γίνονται τα απαραίτητα ερωτήματα στην sql:

- Για τα δέκα πιο δημοφιλή προϊόντα που αγοράζει ο πελάτης που επιλέχθηκε:

```
$query6 = "SELECT DISTINCT `ProductName`, `Many`  
FROM `Product`, `Costumer`, `Bought`  
WHERE Costumer.SSN = Bought.SSN &&  
Bought.Barcode = Product.Barcode && Costumer.SSN = $search1  
ORDER BY Bought.`Many` DESC  
LIMIT 10" ;
```

- Για το πόσα καταστήματα επισκέπτεται:

```
$query1 = "SELECT COUNT(DISTINCT `Store_id`)  
FROM `Transaction`  
WHERE SSN = $search1";
```

- Για το ποιά καταστήματα επισκέπτεται:

```
$query2 = "SELECT DISTINCT `Store_id`  
FROM `Transaction`  
WHERE SSN = $search1";
```

- Για την κατασκευή ενός διαγράμματος με τις ώρες που επισκέπτεται συνήθως καταστήματα της αλυσίδας:

```
$query3 = "SELECT COUNT(`Time`), `Time`  
FROM `Transaction`  
WHERE SSN = $search1  
GROUP BY `Time`";
```

- Για τον μέσο όρο των συναλλαγών του ανά μήνα:

```
$query4 = "SELECT avg(count)  
FROM (  
    SELECT COUNT(`Transaction_ID`) AS count  
    FROM `Transaction`  
    WHERE SSN = $search1  
    GROUP BY `Month`  
) as counts";
```

- και για τον μέσο όρο των συναλλαγών του ανά εβδομάδα:

```
$query5 = "SELECT avg(count)
FROM (
SELECT COUNT(DISTINCT`Transaction_ID`) AS count
FROM `Transaction`
WHERE SSN = $search1
GROUP BY WEEKDAY(`Day`)
) as counts";
```

Και τα αποτελέσματα στο UI για την είσοδο που δόθηκε πριν είναι αντίστοιχα:

Top Ten Products	Πλήθος
Trapezi Kipou	19
Ksirafakia Gillette	10
Pensa	10
Krema Ksero Gw	9
Entera	6
Patsas	4
Afroloutro Dove	4
Rodakina	3
Kefali Arni	3
Skilotrofi	3

Store ID
1
2
3
4
6
9

Number of Visited Stores
6

Ωρα Επίσκεψης	Αριθμός Επισκέψεων
10:30:00	1
11:05:00	3
11:45:00	1
11:55:00	1
12:30:00	1
13:05:00	2
15:05:00	1
17:45:00	1

Μέσος Όρος Επισκέψεων ανά Μήνα
1.8333

Μέσος Όρος Επισκέψεων ανά Βδομάδα
11.0000

Μια εποπτική και συνολική ενοπτεία της απάντησης φαίνεται εδώ όπου για οικονομία χώρου δεν φαίνεται πλήρως η απάντηση αλλά αποτελείται από τους προηγούμενους πίνακες.

[Back](#)

Προφίλ Πελατών

Παρακάτω φαίνονται με την σειρά:

- * Ο αριθμός των διαφορετικών καταστημάτων που έχει πεισκεφτεί ο πελάτης και ποια είναι αυτά
- * Διάγραμμα με τις συνηθέστερες ώρες επίσκεψης
- * Μέσος ώρος επισκέψεων ανά μήνα και ανά βδομάδα
- * Top Ten προϊόντα και πόσες φορές τα έχει αγοράσει

Number of Visited Stores

6

Store ID
1
2
3
4
6
9

Ερώτημα 6c:

Για τα ερωτήματα αυτής της ομάδας ο χρήστης από την αρχική σελίδα της εφαρμογής μας πρέπει να επιλέξει να επιλέξει το “Συμπεράσματα Μαρκετινγκ”



Σε αυτό το κομμάτι θέλουμε να αναλύσουμε κάποια χαρακτηριστικά της αλυσίδας μας, επομένως θα εκτεθούν ερωτήματα τα οποία δεν χρειάζονται είσοδο από τον χρήστη. Για το πιο δημοφιλή ζεύγος προϊόντων:

```
$query5 = "SELECT 100*SUM(case when (`Costumer`.`Birthday` > '1920-01-01' and  
`Costumer`.`Birthday` < '1975-01-01' and `Bought`.`SSN` = `Costumer`.`SSN`) then 1 else 0 end)/  
SUM(case when (`Bought`.`SSN` = `Costumer`.`SSN`) then 1 else 0 end) as AGE_45_99,  
100*SUM(case when (`Costumer`.`Birthday` < '1994-01-01'  
and `Costumer`.`Birthday` > '1975-01-01' and `Bought`.`SSN` = `Costumer`.`SSN`) then 1 else 0 end)/  
SUM(case when (`Bought`.`SSN` = `Costumer`.`SSN`) then 1 else 0 end) as AGE_26_44,  
100*SUM(case when (`Costumer`.`Birthday` < '2020-01-01' and  
`Costumer`.`Birthday` > '1994-01-01' and `Bought`.`SSN` = `Costumer`.`SSN`) then 1 else 0 end)/  
SUM(case when (`Bought`.`SSN` = `Costumer`.`SSN`) then 1 else 0 end) as AGE_0_25  
FROM `Costumer`,`Bought`";
```

Για τις πιο δημοφιλείς θέσεις μέσα στο κατάστημα για την τοποθέτηση προϊόντων:

```
$query1 = "SELECT `rack`,`corridor`, COUNT(`Product_Barcode`) AS counter  
FROM `Product`,`Contained`  
WHERE `Product`.`Barcode` = `Contained`.`Product_Barcode`  
GROUP BY `rack`,`corridor`  
ORDER BY counter DESC";
```

Για το ποσοστό ανά κατηγορία προϊόντων που οι χρήστες εμπιστεύονται προϊόντα με ετικέτα του καταστήματος:

```
$query2 = "SELECT `Product`.`Category_id` as a, 100*SUM(case when `Product`.`is_label` = 1 and  
`Product`.`Barcode` = `Bought`.`Barcode` then 1 else 0 end)/  
SUM(case when `Product`.`Barcode` = `Bought`.`Barcode` then 1 else 0 end) as result  
FROM `Product`,`Bought`  
GROUP BY `Product`.`Category_id`";
```

Για τις ώρες που οι καταναλωτές ξοδεύουν περισσότερα λεφτά:

```
$query3 = "SELECT `Time`  
FROM `Transaction`  
GROUP BY `Time`  
ORDER BY SUM(`Totalcost`) DESC" ;
```

Για το ποσοστό των ηλικιακών ομάδων που επισκέπτονται το κατάστημα κάθε ώρα λειτουργίας

```
$query4 = "SELECT C1.`Product_Barcode` AS LA , C2.`Product_Barcode` AS LA2
FROM `Contained` AS C1 , `Contained` AS C2
WHERE C1.`Product_Barcode` <> C2.`Product_Barcode` AND
C1.`Transaction_Transaction_id` = C2.`Transaction_Transaction_id`
GROUP BY C1.`Product_Barcode` , C2.`Product_Barcode`
ORDER BY COUNT(*) DESC
LIMIT 25";
```

Το UI φαίνεται ως εξής:

Back

Συμπεράσματα Marketing

Βρέθηκαν τα εξείς αποτελέσματα:

Rack	Corridor
5	6
3	2
1	1
6	6
4	3
3	1
7	7
2	5

Category_id	Percentage
1	0.0000
2	75.0000
3	50.0000
4	0.0000
5	0.0000
6	0.0000

Famous Working Hours
11:05:00

Barcode Πρώτου Προϊόντος		Barcode Δεύτερου Προϊόντος
6660067		6660044
6660044		6660067
6660053		6660073
6660073		6660053
6660053		6660067
6660042		6660053
6660042		6660067
6660053		6660042
6660067		6660042
6660067		6660053
6660044		6660053
6660044		6660042
6660055		6660042
6660035		6660073
6660042		6660055
6660053		6660044
6660073		6660035
6660042		6660044
6660042		6660066
6660073		6660066
6660045		6660044
6660043		6660055
6660073		6660033
6660044		6660045
6660044		6660034

Ποσοστό Πελατών Ηλικίας 0-25	Ποσοστό Πελατών Ηλικίας 26-14	Ποσοστό Πελατών Ηλικίας 45-99
33.3333	50.0000	16.6667

Ερώτημα 6d:

Παρακάτω φαίνονται τα views που έχουμε χρησιμοποιήσει για το σύστημα μας:

```
-- views
create VIEW ourcostumers AS
select `Costumer`.`Name`, `Costumer`.`Surname`, `Costumer`.`Birthday`, `Costumer`.`SSN`
from `Costumer`;

CREATE VIEW transa AS
SELECT
`Transaction`.`Transaction_id`,
`Store`.`Store_id`,
`Category`.`Category_id`
FROM
`Category`
NATURAL JOIN `Transaction`
INNER JOIN `Store`
ON `Transaction`.`Store_id` = `Store`.`Store_id`;

-- end views
```

Όπου το πρώτη όψη είναι updatable ενώ η δεύτερη όχι.

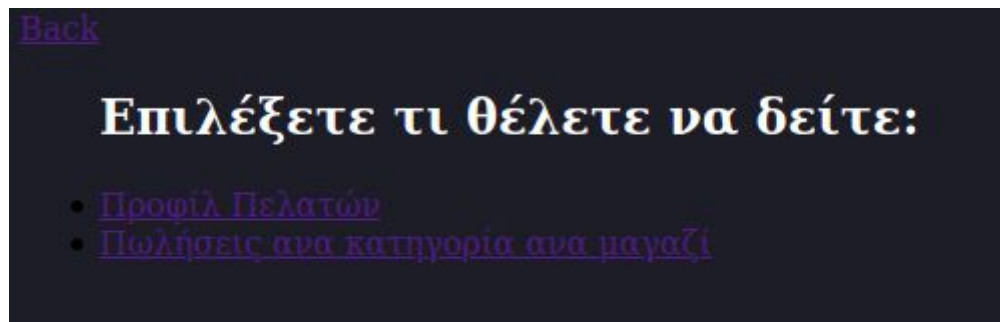
Αυτό φαίνεται και από εδώ:

```
mysql> UPDATE ourcustomers SET Surname = 'Selguk';  
Query OK, 5 rows affected (0.14 sec)  
Rows matched: 6 Changed: 5 Warnings: 0
```

Ο χρήστης αφότου επιλέξει να χρησιμοποιήσει όψεις από την αρχική σελίδα της εφαρμογής μας:



Επιλέγει ποια από τις δύο κατηγορίες όψεων θέλει να εφαρμόσει:



Για το προφίλ πελατών:

Εκμεταλλευόμενοι το view που μόλις ορίσαμε κάνουμε το εξής ερώτημα:

```
$sql = "SELECT * from ourcustomers";
```

και στο UI βλέπουμε το εξής αποτέλεσμα:

Προφίλ Πελατών			
Name	Surname	Birthday	SSN
Selim	Selguk	1975-10-15	21314151
Manwlis	Sarrhs	1985-12-01	21314152
Sotiris	Kontizas	1978-03-21	21314153
Theios	Leonidas	1962-08-11	21314154
Panagiotis	Iliaras	1995-11-25	21314155
Alexis	Barmperos	1995-12-15	21314156

Για τις πωλήσεις ανα κατηγορία ανα μαγαζί πρέπει ο χρήστης να ορίσει την κατηγορία και το μαγαζί που θέλει να μελετήσει:

[Back](#)

Πωλήσεις ανα κατηγορία ανα μαγαζί

ID Καταστήματος(1-10):

ID Κατηγορίας(1-7):

Mapping για ID Καταστημάτων και Κατηγορίας Προϊόντων:

Store ID	Street	Number	City
1	Vilara	44	Ioannina
2	Skoufa	42	Ioannina
3	Xarilaou Trikoupi	12	Ioannina
4	Traulantwni	12	Athina
5	Psaroudaki	13	Athina
6	Voukourestiou	17	Athina
7	Vlaxava	30	Thessaloniki
8	Saadi Levi	8	Thessaloniki
9	Notara	37	Thessaloniki
10	Elenis Zografou	6	Ioannina

Category ID	Name
1	Fresh Products
2	Fridge Products
3	Liquor Products
4	Hygiene Products
5	Home Decor
6	Pet Products

Εκμεταλλεούμενοι τη νέα όψη που φτιάξαμε κάνουμε το εξής ερωτήματα:

```
$query = "SELECT `Transaction_id`
FROM transa
WHERE
`Store_id` = $search1 AND
`Category_id` = $search2";
```

Και έχουμε την εξής απάντηση:

Πωλήσεις ανα κατηγορία ανα μαγαζί

Transaction ID
27
28
29
30

Triggers

1ο trigger

Όταν ο πελάτης χρησιμοποιήσει τη δωροκάρτα για τη συναλλαγή του και εξαργυρώσει τους πόντους του, αυτοί μηδενίζονται.

```
DELIMITER $$
CREATE TRIGGER giftcard
AFTER UPDATE ON `Transaction`
FOR EACH ROW
BEGIN
UPDATE `Card`
SET points=0
WHERE GiftCard=1 ;
END$$
DELIMITER ;
```

2ο trigger

Κάθε φορά που αλλάζουμε τιμή σε ένα προϊόν, ανανεώνουμε το Price History του και στην ημερομηνία αλλαγής βάζουμε την σημερινή ημέρα.


```

DELIMITER $$
CREATE TRIGGER newprice
AFTER UPDATE ON Product
FOR EACH ROW
BEGIN
UPDATE Price_History
SET Day_changed = curdate(), curr_price = new.curr_price
WHERE barcode=new.barcode;
|
#INSERT INTO Price_History(Price, Day_changed, Barcode)
#VALUES (new.curr_price, CURDATE(), new.Barcode)

END$$
DELIMITER ;

```

3ο trigger

Όταν οι πόντοι του χρήστη ξεπερνούν τους 100, τότε του δίνεται το δικαίωμα να εξαργυρώσει πόντους στην επόμενη συναλλαγή του, χρησιμοποιώντας δωροκάρτα για την πληρωμή.

```

DELIMITER $$
CREATE TRIGGER epivraveusi
AFTER UPDATE ON `Transaction`
FOR EACH ROW
BEGIN
UPDATE `Card`
SET giftcard = 1
WHERE points>100 ;
END$$|
DELIMITER ;

```

4ο trigger

Κατά την προσθήκη νέου προϊόντος, αυξάνεται ο αριθμός προϊόντων που περιέχει η κατηγορία στην οποία ανήκει.

```

DELIMITER $$
CREATE TRIGGER Category_New
AFTER INSERT ON Product
FOR EACH ROW
BEGIN
UPDATE Category, Product
SET `Category`.`Number` = `Category`.`Number` + 1
WHERE `Category`.`Category_id` = `Product`.`Category_id`;
END$$
DELIMITER ;

```

5ο trigger

Κατά την αφαίρεση ενός προϊόντος, μειώνεται κατά 1 ο αριθμός προϊόντων που περιέχει η κατηγορία στην οποία ανήκει.

```

DELIMITER $$
CREATE TRIGGER Category_DEL
AFTER DELETE ON Product
FOR EACH ROW
BEGIN
UPDATE Category, Product
SET `Category`.`Number` = `Category`.`Number` - 1
WHERE `Category`.`Category_id` = `Product`.`Category_id`;
END$$
DELIMITER ;

```

6ο trigger

Όταν ο πελάτης πραγματοποιεί νέα συναλλαγή, προστίθενται στην κάρτα του οι πόντοι των προϊόντων τα οποία αγόρασε.

```

DELIMITER $$
CREATE TRIGGER ADDPoints
AFTER INSERT ON Transaction
FOR EACH ROW
BEGIN
UPDATE `Card`
SET `Card`.points = `Card`.points + `Product`.points
WHERE `Customer`.SSN = `Transaction`.SSN;
END$$
DELIMITER ;

```

Με τα παραπάνω triggers εξασφαλίζεται η σταθερότητα και η ορθή λειτουργία της βάσης μας σε περίπτωση διαγραφής, προσθήκης ή επεξεργασίας των δεδομένων, οπότε ο χρήστης μπορεί να τα διαχειριστεί μέσω του κώδικα που επισυνάπτεται.

- `DELETE FROM Product WHERE Barcode='6666666';`

```
INSERT INTO Product (Barcode, ProductName, curr_price, is_label, points, rack, corridor, Category_id)
VALUES('6666666', 'kourtina mpaniou', 9.69, true, 10,2, 3,6);
```

Δικά μας ερωτήματα

Τα ερωτήματα ανάλυσης πληροφορίας που επιλέχθηκαν από εμάς είναι τα εξής:

- Εμφανίζουμε τον “καλύτερο πελάτη” της βάσης μας, δηλαδή εκείνον που έχει πραγματοποιήσει την συναλλαγή με το μεγαλύτερο κόστος σε κατάσταση της αλυσίδας μας. Αυτό το ερώτημα θα μπορούσε να χρησιμεύσει για κάποια επιβράβευση των καλύτερων πελατών μας.
- Εμφανίζουμε το ποσοστό των συναλλαγών στις οποίες γίνεται χρήση giftcard, δηλαδή εξαργυρώνονται πόντοι από την κάρτα του πελάτη, με αποτέλεσμα κάποια έκπτωση στη συναλλαγή του. Το ερώτημα αυτό μας βοηθά να αξιολογήσουμε κατά πόσο οι πελάτες μας εξαργυρώνουν τους πόντους τους και κατά συνέπεια πόσο ευχαριστημένοι είναι από το σύστημα συλλογής πόντων που χρησιμοποιείται και πόσο αυτό συμφέρει την αλυσίδα μας.

Ο χρήστης αφότου επιλέξει να χρησιμοποιήσει τα “Δικά μας Ερωτήματα” από την αρχική σελίδα της εφαρμογής μας:



Έτσι, για το πρώτο ερώτημα γίνεται η εξής ερώτηση:

```
$query1 = "SELECT `Costumer`.`Name` , `Costumer`.`Surname`, `Transaction`.`Totalcost`
FROM `Costumer`, `Transaction`
WHERE `Costumer`.`SSN` = `Transaction`.`SSN`
ORDER BY `Transaction`.`Totalcost` DESC
LIMIT 1";
```

Ενώ για το δεύτερο ερώτημα γίνεται η εξής ερώτηση:

```
$query2 = "SELECT
100*SUM(case when `Transaction`.`Giftcard` = 1 then 1 else 0 end)/
count(distinct `Transaction_id`) AS result
FROM `Transaction`";
```

και έχουμε τις εξής απαντήσεις:

[Back](#)

Δικά μας ερωτήματα!

Βρέθηκαν τα στοιχεία του πελάτη με την μεγαλύτερη συναλλαγή και το ποσοστό των αγορών που χρησιμοποιήθηκε η giftcard:

Όνομα	Επίθετο	Συνολικό Κόστος
Selim	Selguk	71.50

Αποτέλεσμα
44.6809

Σύστημα και γλώσσες προγραμματισμού

Για την υλοποίησή του σε GNU-Linux χρησιμοποιήθηκε το LAMP stick δηλαδή:

- GNU-Linux (διανομή ubuntu 18.04 ως λειτουργικό σύστημα)

Η εφαρμογή αναπτύχθηκε στο λειτουργικό σύστημα Linux, και συγκεκριμένα σε Ubuntu. Τα λειτουργικά συστήματα αυτά είναι της οικογένειας Unix, και επομένως αποτελούν δωρεάν και open source λογισμικό. Ως λειτουργικό σύστημα, χρησιμοποιείται πολύ σε εφαρμογές που χρησιμοποιούν server.

- Apache (Web Server)

Ο Apache είναι ένας web server, δηλαδή ένας εξυπηρετητής παγκόσμιου ιστού. Στην περίπτωση μας χρησιμοποιείται σαν διακομιστής για τη συνεργασία με το σύστημα της βάσης δεδομένων. Γενικότερα, ο Apache επικοινωνεί με τον εξυπηρετητή που παράγει τις σελίδες που ζητούν - επισκέπτονται οι χρήστες μέσω του browser με πρωτόκολλα HTTP, και τις αποστέλλει στον browser για προβολή.

- MySQL (DataBase Management System)

Το MySQL αποτελεί ένα σύστημα διαχείρισης βάσεων δεδομένων. Είναι δωρεάν και open source, ενώ έχει και μεγάλο community για υποστήριξη λόγω της ευρείας χρήσης του. Βασικό πλεονέκτημα είναι ότι μπορεί να λειτουργεί σε διαφορετικά λειτουργικά συστήματα, άρα είναι εύκολη η μεταφορά της βάσης από ένα μηχάνημα σε ένα άλλο. Γενικά χρησιμοποιείται για το σχεδιασμό και τη διαχείριση βάσεων μικρού έως και μεσαίου μεγέθους.

- PHP (Scripting Language)

Η PHP είναι μια γλώσσα προγραμματισμού η οποία, εκτός των άλλων, χρησιμοποιείται και για server-side scripting σε ό,τι αφορά την ανάπτυξη web εφαρμογών.

Συγκεκριμένα, προτιμήθηκε για την εφαρμογή μας διότι είναι φιλική προς την HTML, με την οποία δομούνται όλες τις σελίδες του front end. Συν τοις άλλοις, όπως και τα υπόλοιπα στοιχεία του LAMP μοντέλου, διατίθεται κι αυτή δωρεάν και είναι open source, ενώ διαθέτει και αυτή μεγάλο community για υποστήριξη.

Ως database (storage) engine επιλέχθηκε η InnoDB. Όλο το προαναφερθέν λογισμικό είναι ανοιχτού κώδικα (open source). Στο πρότζεκτ χρησιμοποιήθηκαν views και triggers. Τα triggers χρησιμοποιήθηκαν για την διατήρηση της ακεραιότητας των δεδομένων της βάσης.

Αναλυτικά βήματα εγκατάστασης

Προαπαιτούμενα:

Για την εγκατάσταση σε Ubuntu Based System εγκαθιστούμε τα ακόλουθα με την εξής σειρά:

-Apache2 HTTP server

-mysql-server mysql-client

-php libapache2-mod-php php-mysql php-common php-pear php-mbstring
(Συγκεκριμένα για την εργασία χρησιμοποιήσαμε PHP 7.2)

-php7.2-cgi

Την τελευταία έκδοση(λογοι συμβατότητας με php 7.2) του phpmyadmin ή mysql-workbench για την διαχείριση της βάσης και την δημιουργία των δικαιωμάτων χρηστών.

Πίνακας περιεχομένων video

Ερώτημα	Αρχή	Τέλος
1	00:00	00:58
2	00:59	2:58
3	2:59	3:46
4	3:46	4:49
5	4:49	5:49
6	5:49	8:40
7	8:40	10:11

8	10:11	10:30
9	10:30	11:20
10	11:20	11:35
11	11:35	11:48
12	11:48	12:13
13	12:13	13:51
14	13:51	14:42
15	14:42	16:40