ΜΑΘΗΜΑ: ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Αναφορά εξαμηνιαίας εργασίας: Ανάλυση του καλαθιού αγορών

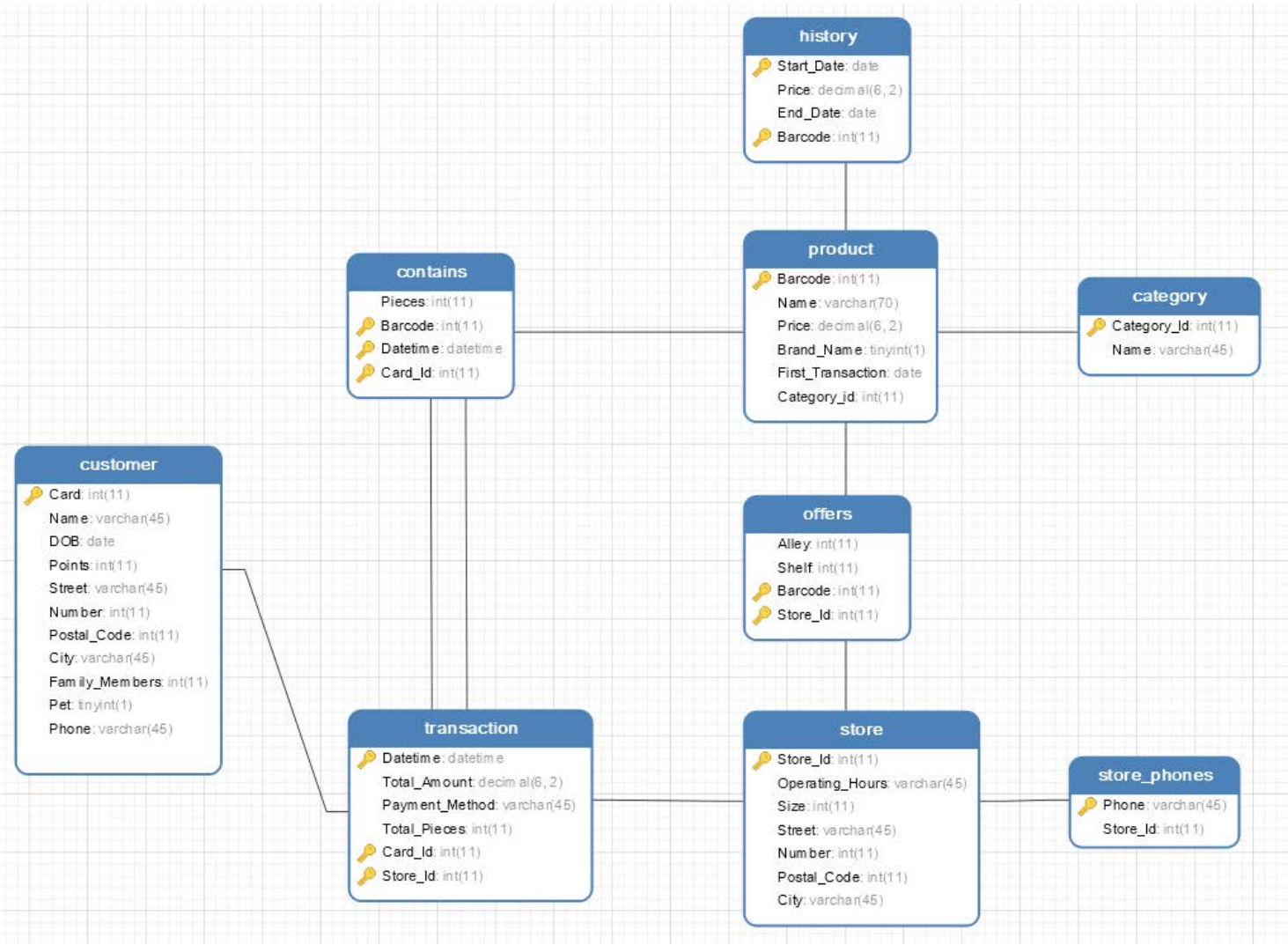
ΟΜΑΔΑ 7
ΑΝΤΩΝΙΟΥ ΓΙΩΡΓΟΣ 03117715
ΚΥΡΙΑΚΟΥ ΔΗΜΗΤΡΗΣ 03117601
ΧΑΤΖΗΧΡΙΣΤΟΦΗ ΧΡΙΣΤΟΣ 03117711
εξάμηνο: 6^{ο}

Η εργασία βρίσκεται στην ακόλουθη διεύθυνση:

http://160.20.145.194:8080/Team7/Project/index.jsp

περιεχόμενα αναφοράς:

1. Το σχεσιακό διάγραμμα της ΒΔ



a. Περιορισμοί

Προτιμήθηκε η υλοποίηση της ΒΔ με τη χρήση του ER diagram που δόθηκε από τις διδάσκουσες (με κάποιες διαφοροποιήσεις) αν και σε μεγάλο βαθμό συμβάδιζε με το διάγραμμα που δημιουργήθηκε σε πρώτη φάση από την ομάδα μας

Σχετικά με την υλοποίηση της ΒΔ:

- Το attribute age που υπήρχε στο αρχικό ER diagram δεν υλοποιήθηκε στην τελική ΒΔ. Αντ' αυτού, στα ερωτήματα που χρειάστηκε η ηλικία των πελατών, υπολογίστηκε με βάση την ημερομηνία γέννησης των πελατών (attribute DOB) και την τρέχουσα ημέρα που εκτελείται το εκάστοτε ερώτημα.

- Ο τύπος δεδομένων που χρησιμοποιήθηκε για κάθε attribute φαίνεται στο πιο πάνω διάγραμμα.

- Η σχέση provides του αρχικού ER diagram δεν υλοποιήθηκε και οι λειτουργίες της συγχωνεύτηκαν με αυτές της σχέσης offers.

- Για λόγους απλότητας τα attributes Total_Amount, Total_Pieces δεν υλοποιήθηκαν ως derived attributes αφού η ΒΔ δεν υποστηρίζει την προσθήκη νέων αγορών και οι ήδη υπάρχουσες θεωρούνται δεδομένες και χρησιμοποιούνται μόνο για τα ερωτήματα ανάλυσης καλαθιού αγορών.

- Στη μηχανή αναζήτησης αγορών χρησιμοποιήθηκε start και end date αντί συγκεκριμένης ημερομηνίας για να είναι πιο ευέλικτη η αναζήτηση.

- Για λόγους απλότητας θεωρήθηκε πως όλα τα καταστήματα της αλυσίδας υπεραγορών έχουν κοινό ωράριο λειτουργίας 9:00 – 21:00 όλες τις ημέρες της εβδομάδας.

- Θεωρήθηκε ότι τα Φρέσκα προϊόντα και τα είδη κάβας δεν μπορούν να αποτελούν προϊόντα ετικέτας του καταστήματος ενώ για τις υπόλοιπες κατηγορίες ο χαρακτηρισμός των προϊόντων ως προϊόντα ετικέτας του καταστήματος έγινε τυχαία.

- Οι πελάτες του καταστήματος είναι ενήλικοι για να μην υπάρχουν περιορισμοί στα είδη των προϊόντων που αγοράζουν, τον τρόπο πληρωμής με κάρτα και να μην θίγονται θέματα πνευματικών δικαιωμάτων ανηλίκων.

- Τα attributes store_Id, Customer_Id, Barcode προκύπτουν αυτόματα από το σύστημα και δεν μπορούν να αλλαχτούν για να αποκλείεται η πιθανότητα ύπαρξης δύο ίδιων τιμών

- Αποφεύχθηκε η χρήση της πρότασης with της SQL διότι δεν υποστηρίζεται από παλαιότερες εκδόσεις MySQL όπως και αυτήν που χρησιμοποιήσαμε. Χρησιμοποιήθηκε η έκδοση MySQL 5.6 για να είναι συμβατή η εφαρμογή και για χρήστες με παλιές εκδόσεις MySQL

Σχετικά με τα εικονικά δεδομένα:

- Τα εικονικά δεδομένα που εισήχθησαν στη βάση θεωρούνται έγκυρα αν και μπορεί κάποια από αυτά να φαίνονται παράλογα (για παράδειγμα τα προϊόντα που αγοράζονται σε ζεύγη μπορεί να μην έχουν λογική συνάφεια). Αυτό οφείλεται στην τυχαία δημιουργία τους με τη χρήση προγραμμάτων δημιουργίας εικονικών δεδομένων από εμάς τους ίδιους.

- Όλες οι εικονικές αγορές της ΒΔ έχουν πραγματοποιηθεί μετά την 1^{η} Ιανουαρίου 2020 ενώ οι αλλαγές στις τιμές των προϊόντων οι οποίες βρίσκονται ήδη στον πίνακα history έχουν γίνει πριν από την πιο πάνω ημερομηνία. Αυτή η παραδοχή έγινε για να είναι πιο εύκολη η δημιουργία των εικονικών δεδομένων για τις αγορές στον πίνακα transactions (συγκεκριμένα για το συνολικό κόστος). Παρόλα αυτά, τυχόν αλλαγή της τιμής ενός προϊόντος ενημερώνει άμεσα και τον πίνακα history με την τρέχουσα ημερομηνία.

- Τα υπάρχοντα καταστήματα της αλυσίδας υπεραγορών χωρίζονται σε 3 κατηγορίες ανάλογα με το μέγεθος τους. Τα μεγάλα καταστήματα (>750 τμ) προσφέρουν όλα τα προϊόντα που είναι καταχωρημένα στη ΒΔ, τα μεσαία καταστήματα ([460,750] τμ) προσφέρουν τα μισά προϊόντα και τα μικρά καταστήματα (<460 τμ) περιέχουν το 1/3 των προϊόντων. Παρόλα αυτά μπορούν να προστεθούν ή να αφαιρεθούν προϊόντα χειροκίνητα σε κάθε κατάστημα.

- Οι πόντοι των πελατών θεωρούνται ανεξάρτητοι του είδους των προϊόντων που αγοράζει ο πελάτης καθώς και των συνολικών αγορών του. Παρουσιάζουν τυχαίες θετικές ακέραιες τιμές με τη λογική ότι ο πελάτης μπορεί να έχει εξαργυρώσει κάποιους από αυτούς.

- Τα εικονικά δεδομένα της ΒΔ έχουν ως άξονα την Κυπριακή πραγματικότητα (πχ πόλεις, αριθμοί τηλεφώνων) ένεκα της καταγωγής των μελών της ομάδας μας.

- Ως ονόματα πελατών χρησιμοποιήθηκαν τα ονόματα ποδοσφαιριστών που αγωνίστηκαν στα παγκόσμια κύπελλα ποδοσφαίρου αντρών και γυναικών το 2018 και 2019 αντίστοιχα. Παράλληλα, ως προϊόντα του καταστήματος χρησιμοποιήθηκαν προϊόντα της αλυσίδας ΕΛΛΗΝΙΚΕΣ ΥΠΕΡΑΓΟΡΕΣ ΣΚΛΑΒΕΝΙΤΗΣ Α.Ε.Ε. μέσω της online υπηρεσίας delivery efood. Οι πηγές για τα πιο πάνω είναι οι ακόλουθες:

  https://en.wikipedia.org/wiki/2018_FIFA_World_Cup_squads
  https://en.wikipedia.org/wiki/2019_FIFA_Women%27s_World_Cup_squads
  https://www.e-food.gr/delivery/menu/sklavenitis

b. Ευρετήρια (indexes)

- Ευρετήριο για τα attributes Card_Id και Datetime του πίνακα contains:

```
CREATE INDEX contains_CardId_Datetime_index
ON contains (Card_Id,Datetime)
```

Το συγκεκριμένο ευρετήριο επιλέχτηκε διότι αποτελεί το primary key του πίνακα contains ο οποίος έχει περισσότερες εγγραφές από οποιοδήποτε άλλο πίνακα της βάσης δεδομένων. Επίσης, χρησιμοποιείται σε πολλά ερωτήματα, για παράδειγμα στο ερώτημα για τα πιο δημοφιλή ζεύγη προϊόντων και το ερώτημα για τις πιο δημοφιλείς θέσεις για τοποθέτηση προϊόντος μέσα στο κατάστημα.

- Ευρετήριο για το attribute Total_Pieces του πίνακα transaction:

```
CREATE INDEX transaction_totPieces_idx
ON transaction (Total_Pieces)
```

Το συγκεκριμένο ευρετήριο επιλέχτηκε διότι κατά την αναζήτηση των αγορών υπάρχει συγκεκριμένο κριτήριο που αναφέρεται στο συνολικό αριθμό προϊόντων. Με αυτό τον τρόπο εξασφαλίζεται η ταχύτερη διεκπεραίωση της αναζήτησης

- Ευρετήριο για το attribute DOB του πίνακα customer:

```
CREATE INDEX customer_DOB_idx
ON customer (DOB)
```

Το συγκεκριμένο ευρετήριο επιλέχτηκε για να εκτελούνται ταχύτερα ερωτήματα τα οποία έχουν να κάνουν με την ανάλυση των αγορών με βάση την ηλικιακή ομάδα.

c. Σύστημα και γλώσσες προγραμματισμού που χρησιμοποιήθηκαν

Αρχικά χρησιμοποιήσαμε το MySQL Workbench ώστε να υλοποιήσουμε την βάση δεδομένων (όλους τους πίνακες, τα indexes, τους περιορισμούς). Έπειτα εγκαταστάθηκε σε host τον οποίο ενοικιάσαμε ο webserver (Apache Tomcat 9.0) και η MySQL (Version 5.6) και ξεκίνησε η υλοποίηση του front-end μέσω του Bootstrap Studio. Στη συνέχεια, ανεβάσαμε τα αρχεία του front-end στον host και ξεκίνησε η υλοποίηση του back-end μέσω της Java. Στα back-end αρχεία της Java βρίσκονται μέσα και τα Queries τα οποία έπρεπε να υλοποιηθούν για την εφαρμογή. Σημείωση: Για να μπορούμε να χειριστούμε τον host χρησιμοποιήθηκε το WinSCP, το PuTTY ώστε να μπορούμε να μεταφέρουμε τα αρχεία στον Host. Επιπλέον στον host ανέβηκε και η βάση δεδομένων που υλοποιήσαμε στην αρχή. Για να μπορούμε να βλέπουμε την βάση δεδομένων χρησιμοποιήθηκε το Navicat.

d. Αναλυτικά βήματα για την εγκατάσταση της εφαρμογής

Εαν κάποιος θελήσει να εγκαταστήσει την εφαρμογή την οποία δημιουργήσαμε, καλό θα ήταν να ενοικιάσει ένα host, να εγκαταστήσει πάνω MySQL(5.6+) και Apache Tomcat 9.0. Τέλος να ανεβάσει το zip file και την βάση δεδομένων (sql file) τα οποία παραδόθηκαν με την αναφορά και το site θα είναι σε πλήρη λειτουργία.

2. Λίστα DDL βάσης

```
-- ----------------------------
-- Table structure for `category`
-- ----------------------------
DROP TABLE IF EXISTS `category`;
CREATE TABLE `category` (
  `Category_Id` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(45) NOT NULL,
  PRIMARY KEY (`Category_Id`),
  UNIQUE KEY `Category_Id_UNIQUE` (`Category_Id`),
  UNIQUE KEY `Name_UNIQUE` (`Name`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;


-- ----------------------------
-- Table structure for `contains`
-- ----------------------------
DROP TABLE IF EXISTS `contains`;
CREATE TABLE `contains` (
  `Pieces` int(11) NOT NULL,
  `Barcode` int(11) NOT NULL,
  `Datetime` datetime NOT NULL,
  `Card_Id` int(11) NOT NULL,
  PRIMARY KEY (`Barcode`,`Datetime`,`Card_Id`),
  KEY `fk_barcode_idx` (`Barcode`),
  KEY `fk_datetime_idx` (`Datetime`),
  KEY `contains_CardId_Datetime_index` (`Card_Id`,`Datetime`),
  CONSTRAINT `fk_barcode_contains` FOREIGN KEY (`Barcode`) REFERENCES
`product` (`Barcode`) ON UPDATE CASCADE,
  CONSTRAINT `fk_cardid_contains` FOREIGN KEY (`Card_Id`) REFERENCES
`transaction` (`Card_Id`),
  CONSTRAINT `fk_datetime_contains` FOREIGN KEY (`Datetime`) REFERENCES
`transaction` (`Datetime`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- ----------------------------
-- Table structure for `customer`
-- ----------------------------
DROP TABLE IF EXISTS `customer`;
CREATE TABLE `customer` (
  `Card` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(45) NOT NULL,
  `DOB` date NOT NULL,
  `Points` int(11) NOT NULL,
  `Street` varchar(45) NOT NULL,
  `Number` int(11) NOT NULL,
  `Postal_Code` int(11) NOT NULL,
  `City` varchar(45) NOT NULL,
  `Family_Members` int(11) NOT NULL,
  `Pet` tinyint(1) NOT NULL,
  `Phone` varchar(45) NOT NULL,
  PRIMARY KEY (`Card`),
  UNIQUE KEY `Card_UNIQUE` (`Card`),
  KEY `customer_DOB_idx` (`DOB`)
) ENGINE=InnoDB AUTO_INCREMENT=231 DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for `history`
-- ----------------------------
DROP TABLE IF EXISTS `history`;
CREATE TABLE `history` (
  `Start_Date` date NOT NULL,
  `Price` decimal(6,2) NOT NULL,
  `End_Date` date NOT NULL,
  `Barcode` int(11) NOT NULL,
  PRIMARY KEY (`Start_Date`,`Barcode`),
  KEY `fk_barcode_idx` (`Barcode`),
  CONSTRAINT `fk_barcode_history` FOREIGN KEY (`Barcode`) REFERENCES
`product` (`Barcode`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- ----------------------------
-- Table structure for `offers`
-- ----------------------------
DROP TABLE IF EXISTS `offers`;
CREATE TABLE `offers` (
  `Alley` int(11) NOT NULL,
  `Shelf` int(11) NOT NULL,
  `Barcode` int(11) NOT NULL,
  `Store_Id` int(11) NOT NULL,
  PRIMARY KEY (`Barcode`,`Store_Id`),
  KEY `fk_barcode_offers_idx` (`Barcode`),
  KEY `fk_storeId_offers_idx` (`Store_Id`),
  CONSTRAINT `fk_barcode_offers` FOREIGN KEY (`Barcode`) REFERENCES `product`
(`Barcode`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_storeId_offers` FOREIGN KEY (`Store_Id`) REFERENCES `store`
(`Store_Id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- ----------------------------
-- Table structure for `product`
-- ----------------------------
DROP TABLE IF EXISTS `product`;
CREATE TABLE `product` (
  `Barcode` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(70) NOT NULL,
  `Price` decimal(6,2) NOT NULL,
  `Brand_Name` tinyint(1) NOT NULL,
  `First_Transaction` date NOT NULL,
  `Category_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`Barcode`),
  UNIQUE KEY `Barcode_UNIQUE` (`Barcode`),
  KEY `fk_categoryId_product_idx` (`Category_id`),
  CONSTRAINT `fk_categoryId_product` FOREIGN KEY (`Category_id`) REFERENCES
`category` (`Category_Id`) ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=126 DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for `store`
-- ----------------------------
DROP TABLE IF EXISTS `store`;
CREATE TABLE `store` (
  `Store_Id` int(11) NOT NULL AUTO_INCREMENT,
  `Operating_Hours` varchar(45) NOT NULL,
  `Size` int(11) NOT NULL,
  `Street` varchar(45) NOT NULL,
  `Number` int(11) NOT NULL,
  `Postal_Code` int(11) NOT NULL,
  `City` varchar(45) NOT NULL,
  PRIMARY KEY (`Store_Id`),
  UNIQUE KEY `Store_Id_UNIQUE` (`Store_Id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8;


-- ----------------------------
-- Table structure for `store_phones`
-- ----------------------------
DROP TABLE IF EXISTS `store_phones`;
CREATE TABLE `store_phones` (
  `Phone` varchar(45) NOT NULL,
  `Store_Id` int(11) NOT NULL,
  PRIMARY KEY (`Phone`),
  KEY `fk_storeId_storePhones_idx` (`Store_Id`),
  CONSTRAINT `fk_storeId_storePhones` FOREIGN KEY (`Store_Id`) REFERENCES
`store` (`Store_Id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- ----------------------------
-- Table structure for `transaction`
-- ----------------------------
DROP TABLE IF EXISTS `transaction`;
CREATE TABLE `transaction` (
  `Datetime` datetime NOT NULL,
  `Total_Amount` decimal(6,2) DEFAULT NULL,
  `Payment_Method` varchar(45) CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci NOT NULL,
  `Total_Pieces` int(11) DEFAULT NULL,
  `Card_Id` int(11) NOT NULL DEFAULT '0',
  `Store_Id` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`Datetime`,`Card_Id`,`Store_Id`),
  KEY `fk_cardId_transaction_idx` (`Card_Id`),
  KEY `fk_storeId_transaction_idx` (`Store_Id`),
  KEY `Datetime` (`Datetime`),
  KEY `transaction_totPieces_idx` (`Total_Pieces`),
  CONSTRAINT `fk_cardId_transaction` FOREIGN KEY (`Card_Id`) REFERENCES
`customer` (`Card`) ON UPDATE CASCADE,
  CONSTRAINT `fk_storeId_transaction` FOREIGN KEY (`Store_Id`) REFERENCES
`store` (`Store_Id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- View structure for `CustomerInfo`
-- ----------------------------
DROP VIEW IF EXISTS `CustomerInfo`;
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER
VIEW `CustomerInfo` AS (select `customer`.`Card` AS `Card`,`customer`.`Name`
AS `Name`,`customer`.`Points` AS `Points` from `customer`);



-- ----------------------------
-- View structure for `SalesPerStorePerCategory`
-- ----------------------------
DROP VIEW IF EXISTS `SalesPerStorePerCategory`;
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER
VIEW `SalesPerStorePerCategory` AS (select `transaction`.`Store_Id` AS
`Store_Id`,
`product`.`Category_id` AS `Category_id`,`category`.`Name` AS
`Category`,sum(`product`.`Price`)
AS `TotalAmount` from (((`transaction` join `product`) join `contains`) join
`category`)
where ((`contains`.`Barcode` = `product`.`Barcode`) and
(`category`.`Category_Id` = `product`.`Category_id`)
and (`transaction`.`Card_Id` = `contains`.`Card_Id`) and
(`transaction`.`Datetime` = `contains`.`Datetime`))
group by `transaction`.`Store_Id`,`product`.`Category_id` order by
`transaction`.`Store_Id`,`product`.`Category_id`);
```

3. Κώδικας SQL

Ευρετήριο:

QUERIES ΕΡΩΤΗΜΑΤΟΣ 6c

ΠΙΟ ΔΗΜΟΦΙΛΗ ΖΕΥΓΗ ΠΡΟΪΟΝΤΩΝ

ΠΙΟ ΔΗΜΟΦΙΛΕΙΣ ΘΕΣΕΙΣ ΜΕΣΑ ΣΤΟ ΚΑΤΑΣΤΗΜΑ ΓΙΑ ΤΗΝ ΤΟΠΟΘΕΤΗΣΗ ΠΡΟΪΟΝΤΩΝ

ΠΟΣΟΣΤΟ ΑΝΑ ΚΑΤΗΓΟΡΙΑ ΠΡΟΪΟΝΤΩΝ ΠΟΥ ΟΙ ΧΡΗΣΤΕΣ ΕΜΠΙΣΤΕΥΟΝΤΑΙ ΠΡΟΪΟΝΤΑ ΜΕ ΕΤΙΚΕΤΑ ΤΟΥ ΚΑΤΑΣΤΗΜΑΤΟΣ

ΩΡΕΣ ΠΟΥ ΟΙ ΚΑΤΑΝΑΛΩΤΕΣ ΞΟΔΕΥΟΥΝ ΠΕΡΙΣΣΟΤΕΡΑ ΛΕΦΤΑ

ΠΟΣΟΣΤΟ ΤΩΝ ΗΛΙΚΙΑΚΩΝ ΟΜΑΔΩΝ ΠΟΥ ΕΠΙΣΚΕΠΤΟΝΤΑΙ ΤΟ ΚΑΤΑΣΤΗΜΑ ΚΑΘΕ ΩΡΑ ΛΕΙΤΟΥΡΓΙΑΣ

ΥΛΟΠΟΙΗΣΕΙΣ ΑΝΑΖΗΤΗΣΕΩΝ

ΑΝΑΖΗΤΗΣΗ ΑΓΟΡΩΝ

ΑΝΑΖΗΤΗΣΗ ΠΕΛΑΤΩΝ

ΤΑ ΔΥΟ ΔΙΚΑ ΜΑΣ QUERIES ΓΙΑ ΤΟ ΕΡΩΤΗΜΑ 6g

ΜΕΣΟ ΠΟΣΟ ΠΟΥ ΞΟΔΕΥΕΙ Ο ΚΑΘΕ ΚΑΤΑΝΑΛΩΤΗΣ, ΑΝΑΛΟΓΑ ΜΕ ΤΑ ΜΕΛΗ ΤΗΣ ΟΙΚΟΓΕΝΕΙΑΣ ΤΟΥ, ΚΑΙ ΑΝ ΕΧΕΙ Ή ΟΧΙ ΚΑΤΟΙΚΙΔΙΟ

ΣΥΝΟΛΙΚΟ ΠΟΣΟ ΠΟΥ ΞΟΔΕΨΕ ΚΑΘΕ ΗΛΙΚΙΑΚΗ ΟΜΑΔΑ ΣΕ ΑΛΚΟΟΛ

QUERIES ΕΡΩΤΗΜΑΤΟΣ 6c

ΠΙΟ ΔΗΜΟΦΙΛΗ ΖΕΥΓΗ ΠΡΟΪΟΝΤΩΝ:

```
SELECT first_prod.Name as n1,second_prod.Name as n2, count(*) AS freq
FROM
(SELECT DISTINCT contains.Barcode, Datetime,Card_Id,Name
      FROM contains
      join product
where contains.Barcode = product.Barcode) AS first_prod
join
(SELECT DISTINCT contains.Barcode, Datetime,Card_Id,Name
      FROM contains
      join product
where contains.Barcode = product.Barcode) AS second_prod
WHERE(
      first_prod.Datetime = second_prod.Datetime AND
      first_prod.Card_Id = second_prod.Card_Id AND
      first_prod.Barcode != second_prod.Barcode AND
      first_prod.Barcode < second_prod.Barcode
)
GROUP BY first_prod.Barcode,second_prod.Barcode
ORDER BY freq DESC
LIMIT 10;
```

ΠΙΟ ΔΗΜΟΦΙΛΕΙΣ ΘΕΣΕΙΣ ΜΕΣΑ ΣΤΟ ΚΑΤΑΣΤΗΜΑ ΓΙΑ ΤΗΝ ΤΟΠΟΘΕΤΗΣΗ ΠΡΟΪΟΝΤΩΝ:

```
SELECT maxFreqPerShop2.Store_Id as sid, Alley,Shelf
FROM(
      SELECT Store_Id,MAX(counter) AS m
      FROM(
            SELECT Alley,Shelf,O.Store_Id, COUNT(*) AS counter
            FROM offers AS O, contains AS C, transaction AS T
            WHERE C.Card_Id = T.Card_Id  and C.Datetime = T.Datetime and
O.Store_Id = T.Store_Id  and O.Barcode = C.Barcode
      GROUP BY Alley,Shelf,O.Store_Id) AS timesShelfVisited
GROUP BY timesShelfVisited.Store_Id) AS maxFreqPerShop,
(SELECT Alley,Shelf,O.Store_Id, COUNT(*) AS counter
      FROM offers AS O, contains AS C, transaction AS T
      WHERE C.Card_Id = T.Card_Id  and C.Datetime = T.Datetime and O.Store_Id
= T.Store_Id  and O.Barcode = C.Barcode
GROUP BY Alley,Shelf,O.Store_Id) AS maxFreqPerShop2
WHERE maxFreqPerShop.Store_Id = maxFreqPerShop2.Store_Id and maxFreqPerShop.m
= maxFreqPerShop2.counter
GROUP BY maxFreqPerShop2.Store_Id;
```

ΠΟΣΟΣΤΟ ΑΝΑ ΚΑΤΗΓΟΡΙΑ ΠΡΟΪΟΝΤΩΝ ΠΟΥ ΟΙ ΧΡΗΣΤΕΣ ΕΜΠΙΣΤΕΥΟΝΤΑΙ ΠΡΟΪΟΝΤΑ ΜΕ ΕΤΙΚΕΤΑ ΤΟΥ ΚΑΤΑΣΤΗΜΑΤΟΣ:

```
SELECT allproducts.Category_id as cid ,allproducts.categoryName as cname
,(Brand_products_p_cat/Total_products_p_cat)*100 as percentage
from
(
      SELECT product.Category_id,count(contains.Barcode) as
Total_products_p_cat,category.Name as categoryName
      FROM product,contains,category
      WHERE product.Barcode = contains.Barcode
      and category.Category_Id = product.Category_id
      GROUP BY product.Category_id
      ORDER BY product.Category_id
) as allproducts
LEFT JOIN
(
      SELECT product.Category_id,count(contains.Barcode) as
Brand_products_p_cat,category.Name
      FROM product,contains,category
      WHERE product.Barcode = contains.Barcode
      and category.Category_Id = product.Category_id
      and Brand_Name=1
      GROUP BY product.Category_id
      ORDER BY product.Category_id
) as brandproducts
on allproducts.Category_id = brandproducts.Category_id;
```

ΩΡΕΣ ΠΟΥ ΟΙ ΚΑΤΑΝΑΛΩΤΕΣ ΞΟΔΕΥΟΥΝ ΠΕΡΙΣΣΟΤΕΡΑ ΛΕΦΤΑ:

```
SELECT hours,SUM(Total_Amount) AS Tot_amount_per_hour
FROM(SELECT hour(Datetime) AS hours,Total_Amount
      FROM transaction
ORDER BY hours asc) AS pinakas
GROUP BY hours
ORDER BY hours ASC;
```

ΠΟΣΟΣΤΟ ΤΩΝ ΗΛΙΚΙΑΚΩΝ ΟΜΑΔΩΝ ΠΟΥ ΕΠΙΣΚΕΠΤΟΝΤΑΙ ΤΟ ΚΑΤΑΣΤΗΜΑ ΚΑΘΕ ΩΡΑ ΛΕΙΤΟΥΡΓΙΑΣ:

```
SELECT allCust.hours,
youngCust.TotNumberOfCust/allCust.TotNumberOfCust *100 AS custYoungerThan35,
midCust.TotNumberOfCust/allCust.TotNumberOfCust *100 AS custBetween35and65,
seniorCust.TotNumberOfCust/allCust.TotNumberOfCust *100 AS custOlderThan65
FROM(SELECT hours,count(*) AS TotNumberOfCust
      FROM(SELECT hour(Datetime) AS hours, DOB
            FROM transaction,customer
            WHERE transaction.Card_Id = customer.Card
      ORDER BY hours asc) AS pinakas
GROUP BY hours) AS allCust
LEFT JOIN
(SELECT hours,count(*) AS TotNumberOfCust
      FROM(SELECT hour(Datetime) AS hours, DOB
            FROM transaction,customer
            WHERE transaction.Card_Id = customer.Card
            and (year(CURDATE())- YEAR(DOB)) < 35
      ORDER BY hours asc) AS pinakas
GROUP BY hours) AS youngCust
on allCust.hours = youngCust.hours
LEFT JOIN
(SELECT hours,count(*) AS TotNumberOfCust
      FROM(SELECT hour(Datetime) AS hours, DOB
            FROM transaction,customer
            WHERE transaction.Card_Id = customer.Card
            and (year(CURDATE())- YEAR(DOB)) <= 65
            and (year(CURDATE())- YEAR(DOB)) >= 35
      ORDER BY hours asc) AS pinakas
GROUP BY hours) AS midCust
on allCust.hours = midCust.hours
LEFT JOIN
(SELECT hours,count(*) AS TotNumberOfCust
      FROM(SELECT hour(Datetime) AS hours, DOB
            FROM transaction,customer
            WHERE transaction.Card_Id = customer.Card
            and (year(CURDATE())- YEAR(DOB)) > 65
      ORDER BY hours asc) AS pinakas
GROUP BY hours) AS seniorCust
on allCust.hours = seniorCust.hours;
```

ΥΛΟΠΟΙΗΣΕΙΣ ΑΝΑΖΗΤΗΣΕΩΝ

ΑΝΑΖΗΤΗΣΗ ΑΓΟΡΩΝ: (Συνδυασμός SQL και Java, ούτως ώστε να μπορεί να γίνει αναζήτηση χωρίς να γεμίσουμε όλα τα πεδία)

```
SELECT DISTINCT
Datetime,Store_Id,Card_Id,Payment_Method,Total_Pieces,Total_Amount
FROM( SELECT
T.Store_Id,T.Card_Id,T.Datetime,T.Total_Pieces,T.Payment_Method,T.Total_Amoun
t,P.Category_Id
      FROM product as P,transaction as T,contains as C
WHERE C.Barcode = P.Barcode AND T.Datetime = C.Datetime AND T.Card_Id =
C.Card_Id) as A
WHERE 1=1 ;
if(!.equals(shopname)) {
sqltemp +=  AND Store_Id =  + shopname;
}
if(!.equals(datestart)) {
sqltemp +=  AND Datetime >=  + ' + datestart + '  ;
}
if(!.equals(dateend)) {
sqltemp +=  AND Datetime <  + ' + dateend + '  ;
}
if(!.equals(tpieces)) {
sqltemp +=  AND Total_Pieces =  + tpieces;
}
if(!.equals(paymethod)) {
sqltemp +=  AND Payment_Method =  + ' +paymethod + ';
}
if(!.equals(tcost)) {
sqltemp +=  AND Total_Amount =  + tcost;
}
if(!.equals(cat)) {
sqltemp +=  AND  + cat +  = all (SELECT P.Category_Id
      FROM contains as C, product as P
WHERE C.Barcode = P.Barcode AND C.Datetime = A.Datetime AND C.Card_Id =
A.Card_Id)
GROUP BY Datetime;
```

ΑΝΑΖΗΤΗΣΗ ΠΕΛΑΤΩΝ:

Αριθμός καταστημάτων που επισκέφτηκε:

```
SELECT count(DISTINCT Store_Id) AS total FROM transaction WHERE Card_Id = ?;
```

Κατάλογος καταστημάτων που επισκέφτηκε:

```
SELECT DISTINCT Store_Id FROM transaction WHERE Card_Id =? ORDER BY(Store_Id)
Asc;
```

Πιο δημοφιλή προϊόντα:

```
SELECT P.Name
FROM contains as C, product as P
WHERE Card_Id = ? AND P.Barcode = C.Barcode
GROUP BY P.Barcode
ORDER BY SUM(C.Pieces) Desc
LIMIT 10;
```

Συνήθεις ώρες που ο πελάτης επισκέπτεται κάθε κατάστημα

```
SELECT DISTINCT M.Store_Id, hours
FROM (
      SELECT MAX(count_hours) as max_value, Store_Id
      FROM (  SELECT count(hour(Datetime)) as count_hours,Store_Id
            FROM transaction
            WHERE Card_Id = ?
            GROUP BY hour(Datetime) ,Store_Id
      ORDER BY Store_Id) as counter_of_visits_per_hour_per_store_id
GROUP BY Store_Id) as M,
(SELECT COUNT(hours) as count_hours,Store_Id,hours
      FROM (  SELECT hour(Datetime) as hours ,Store_Id
            FROM transaction
            WHERE Card_Id = ?
      ORDER BY Store_Id Desc) as hours_visit_per_store_id
      GROUP BY hours,Store_Id
ORDER BY Store_Id) as D
WHERE M.max_value = D.count_hours AND M.Store_Id = D.Store_Id
GROUP BY M.Store_Id
ORDER BY M.Store_Id;
```

Μέσος αριθμός αγορών ανά μήνα και ανά εβδομάδα

```
SELECT
30.5 * count( * ) / (
      (( YEAR ( CURDATE( ) ) - 2020 ) * 365.25 + ( MONTH ( CURDATE( ) ) - 1 )
* 30.5 + DAY ( CURDATE( ) - 1 ) ) +
      - (( YEAR ( Datetime ) - 2020 ) * 365.25 + ( MONTH ( Datetime ) - 1 ) *
30.5 + DAY ( Datetime ) - 1 )  +
) AS AVGpMonth,
7 * count( * ) / ( +
      (( YEAR ( CURDATE( ) ) - 2020 ) * 365.25 + ( MONTH ( CURDATE( ) ) - 1 )
* 30.5 + DAY ( CURDATE( ) - 1 ) ) +
      - (( YEAR ( Datetime ) - 2020 ) * 365.25 + ( MONTH ( Datetime ) - 1 ) *
30.5 + DAY ( Datetime ) - 1 ) +
) AS AVGpWeek
FROM transaction
WHERE Card_Id = ?
ORDER BY Datetime ASC
LIMIT 1;
```

Σημείωση: **Υπάρχουν επίσης queries σε όλα τα insert, delete, update των product, store, customer και στα insert, delete των offer, store_phone.**

ΤΑ ΔΥΟ ΔΙΚΑ ΜΑΣ QUERIES ΓΙΑ ΤΟ ΕΡΩΤΗΜΑ 6g:

ΜΕΣΟ ΠΟΣΟ ΠΟΥ ΞΟΔΕΥΕΙ Ο ΚΑΘΕ ΚΑΤΑΝΑΛΩΤΗΣ, ΑΝΑΛΟΓΑ ΜΕ ΤΑ ΜΕΛΗ ΤΗΣ ΟΙΚΟΓΕΝΕΙΑΣ ΤΟΥ, ΚΑΙ ΑΝ ΕΧΕΙ Ή ΟΧΙ ΚΑΤΟΙΚΙΔΙΟ:

```
SELECT Pets.Family_Members as fmembers ,noPets.avgWithoutPet,Pets.avgWithPet
FROM
(SELECT Family_Members,avg(Total_Amount) AS avgWithoutPet
      FROM transaction
      left join customer
      on transaction.Card_Id = customer.Card
      WHERE Pet=0
      GROUP BY Family_Members
ORDER BY Family_Members) AS noPets
join
(SELECT Family_Members,avg(Total_Amount) AS avgWithPet
      FROM transaction
      left join customer
      on transaction.Card_Id = customer.Card
      WHERE Pet=1
      GROUP BY Family_Members
ORDER BY Family_Members) AS Pets
on noPets.Family_Members = Pets.Family_Members;
```

ΣΥΝΟΛΙΚΟ ΠΟΣΟ ΠΟΥ ΞΟΔΕΨΕ ΚΑΘΕ ΗΛΙΚΙΑΚΗ ΟΜΑΔΑ ΣΕ ΑΛΚΟΟΛ:

```
SELECT
case
      when year(CURDATE())-year(DOB) <30 then '18-29'
      when year(CURDATE())-year(DOB) between 30 and 49 then '30-49'
      when year(CURDATE())-year(DOB) between 50 and 65 then '50-65'
      when year(CURDATE())-year(DOB) >65 then '65'
END AS age_range,sum(Price) AS TotalAmountSpentOnAlcohol
FROM contains,customer,product
WHERE
contains.Card_Id = customer.Card
and contains.Barcode = product.Barcode
and product.Category_id = 3
GROUP BY age_range
ORDER BY age_range asc;
```

4. Απαιτούμενος κώδικας για την εγκατάσταση της εφαρμογής

Ο κώδικας που απαιτείται για την εγκατάσταση της εφαρμογής επισυνάπτεται στο φάκελο μαζί με την παρούσα αναφορά και ένα README αρχείο με οδηγίες για εγκατάστασή