1. **Αναφέρετε τις τεχνολογίες που χρησιμοποιήσατε για την ανάπτυξη της εφαρμογής.**
2. **Δείξτε το σχεσιακό σχήμα που χρησιμοποιήσατε για την βάση δεδομένων, τονίστε διαφορές από το ER diagram που σας δόθηκε.**

Κατά κύριο λόγο έχουμε ακολουθήσει το ER που μας δόθηκε ως λύση, με κάποιες αλλαγές που έγιναν προκειμένου να διευκολυνθούμε στην ανάπτυξη. Συγκεκριμένα,

* Προσθέσαμε στο Transaction το FK Store\_id references Store(Store\_id) ώστε να συνδέσουμε τις συναλλαγες με το κατάστημα στο οποίο γίνονται, και τα attributes Total\_amount και Total\_pieces
* Όλα τα derived attributes (όπως το end\_date της Older\_prices) τα έχουμε συμπεριλάβει στους αντίστοιχους πίνακές μας.
* Επίσης, έχουμε θεωρήσει ότι σε όλα τα καταστήματα της αλυσίδας, διατίθενται τα ίδια προϊόντα και στις ίδιες τιμές. (Δε μπορεί να διαφοροποιηθεί το ένα κατάστημα από τα υπόλοιπα ως προς τα προϊόντα που προσφέρει).

1. **Σχολιάστε τους περιορισμούς ακεραιότητας που έχετε επιλέξει. Τι θα συμβεί στην βάση αν προ-σπαθήσετε να διαγράψετε ένα προϊόν και γιατί; Δείξτε το κομμάτι του κώδικα στην SQL που θα καθο-ρίσει τη συμπεριφορά. Θα μπορούσατε να έχετε επιλέξει κάποια άλλη συμπεριφορά; Τι αλλαγές θα χρειαζόταν ο κώδικας;**

Έχουμε επιβάλει τον περιορισμό ακεραιότητας on delete cascade σε όσες περιπτώσεις κάποιο attribute κάνει reference ως foreign key στο primary key του Product, το Barcode. Συγκεκριμένα έχουμε

foreign key (Product\_barcode) references Product(Barcode) on delete cascade on update cascade στα tables Older\_prices, Offers,

foreign key (Product\_barcode) references Product(Barcode) on delete cascade on update cascade στο table Contains

Ο περιορισμός ακεραιότητας on delete cascade εξασφαλίζει ότι όταν διαγραφεί το προϊόν, θα διαγραφούν και τα entries του Older\_prices και του Offers και του Contains που έχουν ως foreign key το συγκεκριμένο barcode προϊόντος. Η mySQL δίνει τη δυνατότητα αντί για cascade να επιλέξουμε ένα από τα ακόλουθα: RESTRICT – NO ACTION που απορρίπτει τη διαγραφή, SET NULL, SET DEFAULT. Το να απορρίψουμε τη διαγραφή δεν έχει νόημα.

Επίσης **δε** θα μπορούσαμε αντί για on delete cascade, που οδηγεί στη διαγραφή που αναφέραμε πριν, να έχουμε κάνει την επιλογή SET NULL / DEFAULT με κάποιο default barcode όπως 00..000 . SET NULL δε θα μπορούσαμε να κάνουμε, καθώς θέλουμε το barcode να είναι not null (αποτελεί PK του Product), και μας δίνει error αν προσπαθήσουμε. Επίσης καθώς το Product\_barcode αποτελεί μέρος του primary key για τα Offers, Contains, Older\_prices δε θα μπορούσαμε να το κάνουμε on delete set default διότι τότε θα βρισκόμασταν με 2 ή περισσότερα entries με ίδιο ΡΚ: (Card\_number, DateTime, Store\_id, Product\_barcode) στην Contains, ή (Product\_barcode, Store\_id) στην Offers, πράγμα το οποίο δε γίνεται.

1. **Δείξτε στον κώδικα της SQL τα ευρετήρια που έχετε χρησιμοποιήσει, δικαιολογήστε τις επιλογές σας.**

CREATE UNIQUE INDEX transaction\_pk ON Transaction(DateTime, Store\_id, Card\_number);

CREATE UNIQUE INDEX category\_pk ON Product(Barcode);

CREATE UNIQUE INDEX contains\_pk ON Contains(Product\_barcode, Card\_number, DateTime, Store\_id);

Έχουμε ορίσει ευρετήρια στους πίνακες Transaction, Product, Contains βάσει των primary keys τους, γιατί τα περισσότερα queries μας κάνουν χρήση των τριών αυτών πινάκων. Χρησιμοποιώντας το explain της mySQL, μπροστά από ένα query, μπορούμε να δούμε αν χρησιμοποιεί κάποιο από τα indexes που έχουμε ορίσει. Πράγματι, διαπιστώνουμε ότι τα ευρετήρια αυτά χρησιμοποιούνται και επομένως επιταχύνεται η αναζήτηση.

1. **Δείξτε τα δεδομένα που είναι διαθέσιμα στην βάση δεδομένων σας.**
2. **Χρησιμοποιείστε την διεπαφή χρήστη για να δείξετε την αναζήτηση στις αγορές που έχουν γίνει με βάση πολλαπλά κριτήρια. Ξεκινήστε επιλέγοντας ένα κατάστημα, στην συνέχεια περιορίστε τις αγορές με βάση την ημερομηνία. Εμφανίστε μόνο τις αγορές που έγιναν με μετρητά σε αυτό το διάστημα. Στην συνέχεια αλλάξτε το χρονικό διάστημα που εξετάζετε. Τέλος, περιορίστε ακόμα περισσότερο τις αγορές που εμφανίζετε δείχνοντας αγορές με περισσότερα από 5 προϊόντα. Εξηγήστε πως διαμορ-φώνεται το SQL ερώτημα με βάση τις επιλογές κριτηρίων.**
3. **Στην συνέχεια επιλέξτε έναν πελάτη και δείξτε για αυτόν τα δέκα πιο δημοφιλή προϊόντα που αγοράζει, πόσα και ποια καταστήματα επισκέπτεται, το διάγραμμα με τις ώρες που επισκέπτεται ένα από αυτά τα καταστήματα και τον μέσο όρο των συναλλαγών του την τελευταία εβδομάδα και τον τελευταίο μήνα. Δείξτε και εξηγήστε το SQL ερώτημα που υπολογίζει τον μέσο όρο. (6b)**

SELECT MONTH(T.DateTime), SUM(T.Total\_amount), COUNT(\*), SUM(Total\_amount)/COUNT(\*) as Montlhy\_average

FROM Transaction as T

WHERE T.Card\_number = 1

GROUP BY MONTH(T.DateTime)

ORDER BY MONTH(T.DateTime) ;

Ως μέσο όρο συναλλαγών ανά μήνα, εννοούμε τη μέση αξία μίας συναλλαγής που πραγματοποιήθηκε εκείνον το μήνα. Για τον υπολογισμό αυτού του μέσου όρου, κάνουμε group by month τις συναλλαγές του πελάτη, αθροίζουμε τη συνολική αξία τους, και διαιρουμε με το πλήθος συναλλαγών που πραγματοποίησε ο πελάτης μέσα στο συγκεκριμένο μήνα. Η αθροιστική συνάρτηση SUM(Total\_amount) υπολογίζει τη συνολική αξία των αγορών-συναλλαγών, και η συνάρτηση count μετρά το πλήθος των συναλλαγών. Μετά ο υπολογισμός του Μ.Ο είναι μια απλή διαίρεση.

Για το μέσο όρο ανά εβδομάδα το query είναι εντελώς ίδιο, απλώς αντί για group by month κάνουμε group by week.

1. **Δείξτε τα πιο δημοφιλή ζεύγη προϊόντων (6c)**

Θεωρούμε ότι δύο προϊόντα εμφανίζονται μαζί σε μια συναλλαγή όταν έχει αγοραστεί στην ίδια συναλλαγή τουλάχιστον ένα τεμαχιο από το καθένα.

Το query δε ζητείται να δειχθεί, είναι πάντως το εξής:

SELECT T.DateTime, T.Card\_number, T.Store\_id, P1.Barcode, P2.Barcode, COUNT(\*)

FROM Product AS P1, Product AS P2, Transaction as T, Contains as C1, Contains as C2

WHERE

# T.Card\_number = 1 AND

T.Card\_number = C1.Card\_number

AND T.DateTime = C1.DateTime

AND T.Store\_id = C1.Store\_id

AND T.Card\_number = C2.Card\_number

AND T.DateTime = C2.DateTime

AND T.Store\_id = C2.Store\_id

AND P1.Barcode < P2.Barcode

AND P1.Barcode = C1.Product\_barcode

AND P2.Barcode = C2.Product\_barcode

GROUP BY P1.Barcode, P2.Barcode

ORDER BY COUNT(\*) DESC;

1. **Δείξτε τις πιο δημοφιλείς θέσεις μέσα στο κατάστημα για την τοποθέτηση προϊόντων καθώς και το SQL ερώτημα που τις υπολογίζει. (6c)**

Σε κάθε κατάστημα έχουμε διαφορετική τοποθέτηση των προϊόντων, καθώς ο κώδικας που γράψαμε όταν κάναμε populate τη βάση, κάνει generate διαφορετικές τοποθετήσεις για κάθε κατάστημα, οπότε στο query επιλέγουμε συγκεκριμένο κατάστημα κάθε φορά (π.χ. WHERE T.Store\_id = 3 για να δούμε για το κατάστημα με id = 3).

Ως θέση του προϊόντος μέσα στο κατάστημα θεωρούμε το συνδυασμό (Alley\_number, Shelf\_number). Οι δημοφιλέστερες θέσεις επομένως είναι αυτές στις οποίες έχουν τοποθετηθεί τα δημοφιλέστερα προϊόντα, αυτά που εμφανίζονται συχνότερα σε συναλλαγές.

SELECT O.Alley\_number,O.Shelf\_number, COUNT(\*)

FROM Transaction as T, Contains as C, Product as P, Offers as O

WHERE

T.DateTime = C.DateTime

AND T.Card\_number = C.Card\_number

AND T.Store\_id = C.Store\_id

AND C.Product\_barcode = P.Barcode

AND O.Product\_barcode = P.Barcode

AND T.Store\_id = O.Store\_id

AND T.Store\_id = 3

**GROUP BY O.Alley\_number, O.Shelf\_number**

**ORDER BY COUNT(\*) DESC;**

1. **Δείξτε το ποσοστό ανά κατηγορία προϊόντων που οι χρήστες εμπιστεύονται προϊόντα με ετικέτα του καταστήματος. (6c)**

Το ποσοστό αυτό υπολογίζεται ως ο λόγος των αγορών που περιέχουν προϊόν που ανήκει στην κατηγορία και φέρει label του καταστήματος, προς το πλήθος των αγορών που περιέχουν προϊόν που ανήκει στην κατηγορία ανεξάρτητα από το αν φέρει label του καταστήματος ή όχι.

select \*, bn\_count/total\_count\*100 as percentage from

(select Category\_id, count(\*) as bn\_count

from

(select \* from

Transaction natural join Contains as C inner join Product as P on Product\_barcode = Barcode) as q1

where Brand\_name = True

group by Category\_id) as q1

natural join

(

SELECT Category\_id, COUNT(\*) as total\_count

from

(select \* from

Transaction natural join Contains as C inner join Product as P on Product\_barcode = Barcode) as q1

group by Category\_id) as q2

order by category\_id;

1. **Δείξτε τις ώρες που οι καταναλωτές ξοδεύουν περισσότερα λεφτά καθώς και το SQL ερώτημα που τις υπολογίζει. (6c)**

SELECT T.Store\_id, HOUR(T.DateTime), SUM(T.Total\_amount) , COUNT(\*)

FROM Transaction as T

GROUP BY HOUR(DateTime), T.Store\_id

ORDER BY T.Store\_id, SUM(T.Total\_amount) DESC;

1. **Δείξτε το ποσοστό των ηλικιακών ομάδων που επισκέπτονται το κατάστημα κάθε ώρα λειτουργί- ας. (6c)**
2. Δείξτε τις δύο όψεις που έχετε φτιάξει, δείτε τον SQL κώδικα που τις υπολογίζει. Σχολιάστε αν εί- ναι ενημερώσιμες και γιατί. (6d)

create view customer\_info as

select \* from customer;

Η όψη customer\_info είναι ενημερώσιμη καθώς δημιουργείται με ένα απλό select all από το table customer και κανένα άλλο.

create view **sales\_per\_product\_category** as

select T.DateTime, T.Card\_number, T.Store\_id, P.Barcode, Ctg.Name

FROM Transaction as T, Contains as Cn, Category as Ctg, Product as P

WHERE

T.Card\_number = Cn.Card\_number

AND T.DateTime = Cn.DateTime

AND T.Store\_id = Cn.Store\_id

AND Cn.Product\_barcode = P.Barcode

# AND Ctg.Category\_id = 1

ORDER BY T.DateTime, T.Card\_number, T.Store\_id, Ctg.Name;

Η όψη sales\_per\_product\_category είναι ενημερώσιμη, όπως βλέπουμε κάνοντας select table\_name, is\_updatable from information\_schema.views. Παρ’όλα αυτά βλέπουμε πως δεν μπορούμε να κάνουμε update σε αυτήν καθώς το order by δημιουργεί πρόβλημα, insert καθώς αυτό θα κάνει αλλαγές σε περισσότερα από ένα βασικά tables, και delete δεν μπορούμε να κάνουμε καθώς παίρνουμε το error message “cannot delete from join view”. Οπότε επί της ουσίας δεν είναι updatable το view αυτό γιατί δημιουργείται από το join διαφορετικών tables και αυτό περιπλέκει τα πράγματα όταν θέλουμε να κάνουμε updates - modifications.

1. **Δείξτε το ιστορικό τιμών για δύο προϊόντα. Σχολιάστε ποιες επιλογές σε περιορισμούς στην βάση δεδομένων που επιτρέπουν να δείχνετε το ιστορικό (6f).**

Έχουμε δημιουργήσει ένα trigger, το οποίο ενεργοποιείται όποτε γίνεται update στο πεδίο Price του table “Product”. Συγκεκριμένα, αφού γίνει η αλλαγή της τιμής, η αλλαγή αυτή «πυροδοτεί» αλλαγή στον πίνακα Older\_prices, προσθέτοντας μια εγγραφή για τη νέα τιμή του εν λόγω προϊόντος. Εφόσον επιτρέπεται να αλλάξουμε την τιμή του attribute Price του προϊόντος, είναι δυνατόν να κάνουμε τέτοιες αλλαγές και να βλέπουμε το ιστορικό τους.

1. **Δείξτε τα δύο ερωτήματα που έχετε σχεδιάσει εσείς. Σχολιάστε την επιλογή σας και δείξτε τον αντίστοιχο SQL κώδικα. (6g)**

* Ποιο κατάστημα κάνει τις περισσότερες πωλήσεις, δηλαδή δέχεται την περισσότερη κίνηση

SELECT COUNT(\*)

FROM Transaction as T

GROUP BY T.Store\_id

ORDER BY COUNT(\*) DESC;

* Πώς μεταβάλλεται ο λόγος πωλήσεις προϊόντος ανά συναλλαγή πριν και μετά την αλλαγή της τιμής του. Συγκεκριμένα κοιτάζουμε το λόγο τεμάχια του προϊόντος που πουλήθηκαν ανά συναλλαγή, στο διάστημα από την προτελευταία αλλαγή τιμής μέχρι την πιο πρόσφατη, και στο διάστημα μετά την αλλαγή τιμής.

SELECT Start\_date,End\_date into @sdate, @edate from

Older\_prices

where Product\_barcode = '1234567891'

order by Start\_date desc

limit 1,1;

select before\_ratio, after\_ratio

from

(

select ntrans\_before, npieces\_before, npieces\_before/ntrans\_before as before\_ratio

from

(SELECT COUNT(\*) as ntrans\_before

FROM Transaction as T

WHERE

T.DateTime <= @edate and T.DateTime > @sdate) as q1

join

(

SELECT sum(Cn.Pieces) as npieces\_before

FROM Transaction as T, Contains as Cn, Product as P

WHERE

T.Card\_number = Cn.Card\_number

AND T.DateTime = Cn.DateTime

AND T.Store\_id = Cn.Store\_id

AND Cn.Product\_barcode = P.Barcode

AND P.Barcode = '1234567891'

AND T.DateTime <= @edate and T.DateTime > @sdate)

as q2)

as q1

join

(

select ntrans\_after, npieces\_after, npieces\_after/ntrans\_after as after\_ratio

from

(SELECT COUNT(\*) as ntrans\_after

FROM Transaction as T

WHERE

T.DateTime > @edate) as q1

join

(

SELECT sum(Cn.Pieces) as npieces\_after

FROM Transaction as T, Contains as Cn, Product as P

WHERE

T.Card\_number = Cn.Card\_number

AND T.DateTime = Cn.DateTime

AND T.Store\_id = Cn.Store\_id

AND Cn.Product\_barcode = P.Barcode

AND P.Barcode = '1234567891'

AND T.DateTime > @edate) as q2

) as q2;