
# *Αναφορά Εξαμηνιαίου Project στις Βάσεις Δεδομένων 2020-2021*

***Βικέντιος Βιτάλης – Συμεών Ποργιώτης – Στέφανος Τσώλος***

***el18803 - el18053 - el18050***

***Team CS***

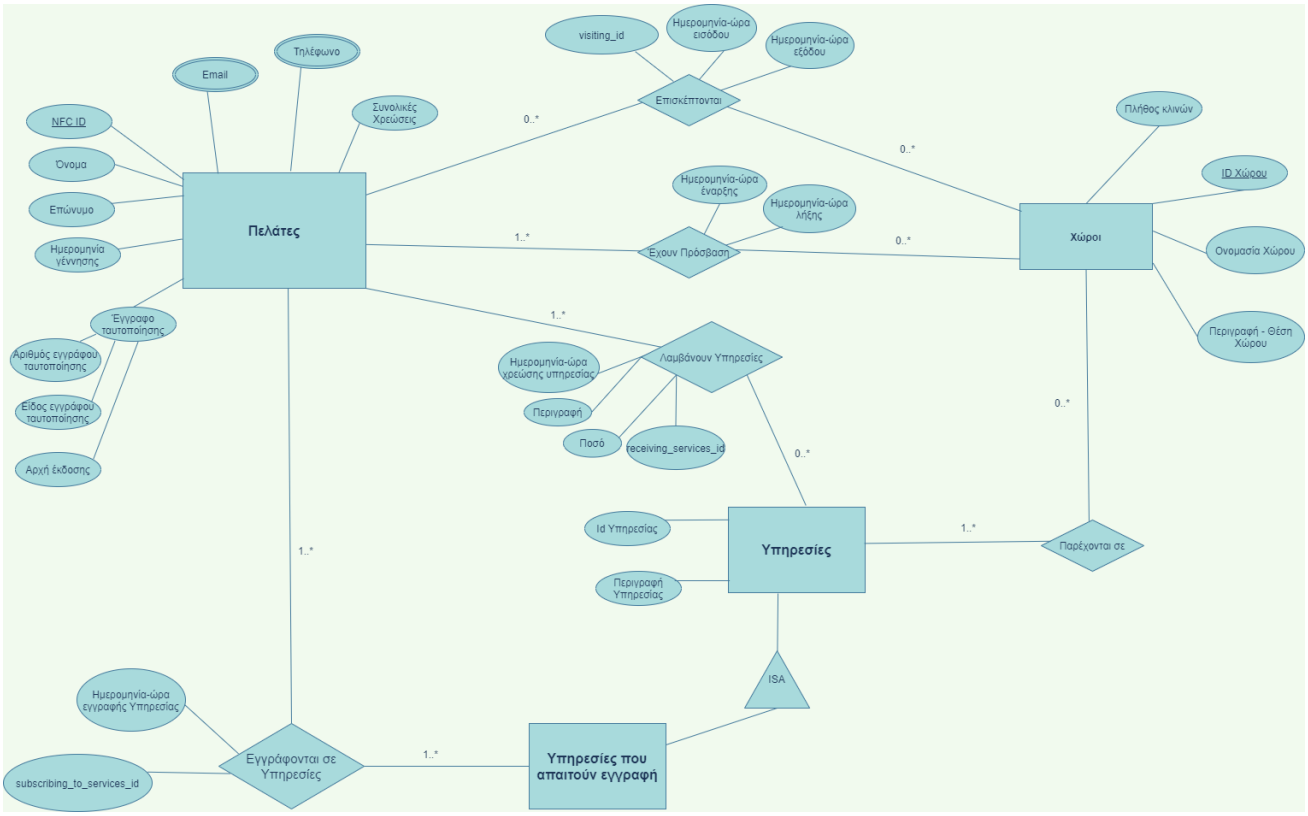
## Εισαγωγή:

Αρχικά θα θέλαμε να επισημάνουμε ότι αλλάξαμε το αρχικό ER που παραδώσαμε στην τελική μορφή που παρουσιάζεται στο πρώτο σχήμα, παίρνοντας βοήθεια κι από την προτεινόμενη λύση που μας δόθηκε, καθώς στην πορεία φάνηκε πως:

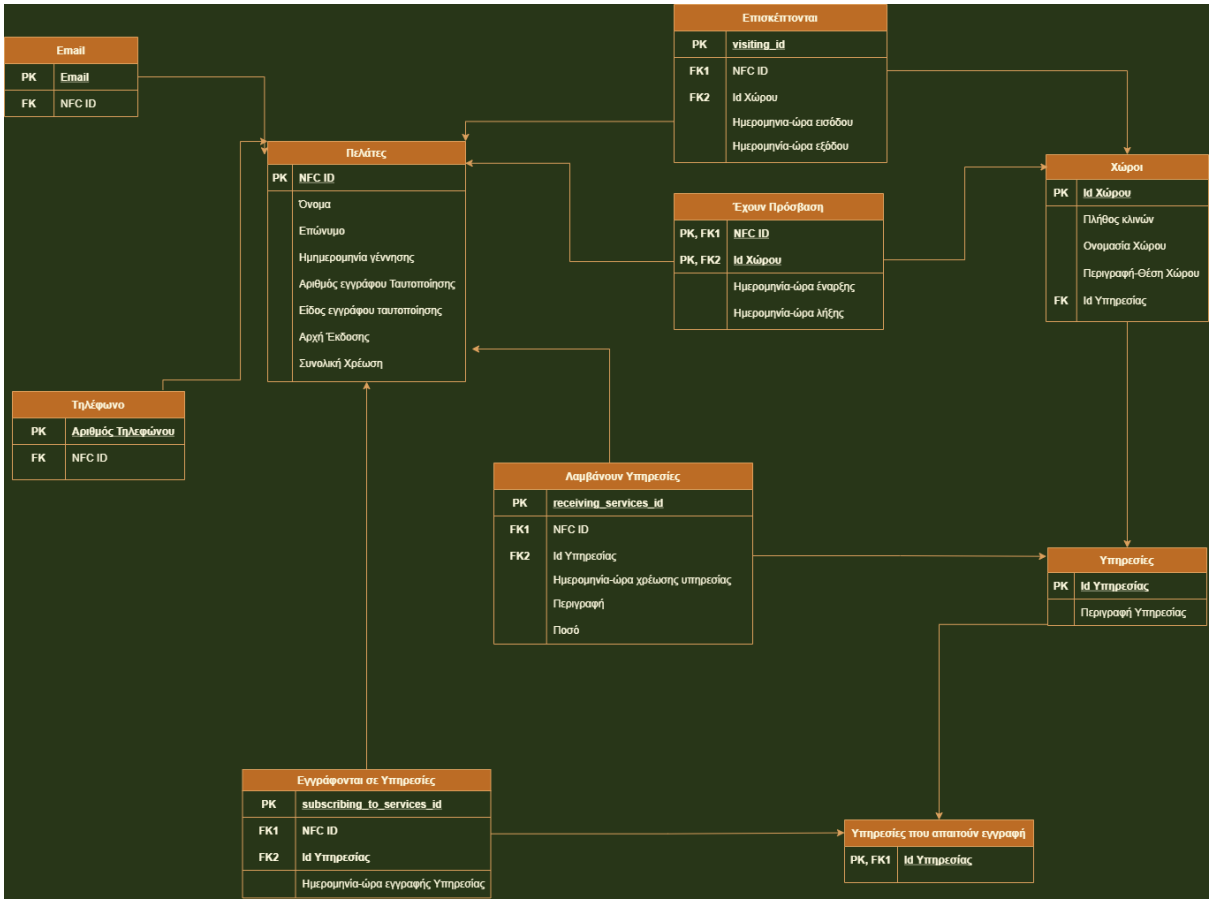
1. Χρειάστηκε να γίνει ενοποίηση του πίνακα “χρέωσης υπηρεσίας” με τον πίνακα “λαμβάνουν υπηρεσίες”.
2. Έχουμε προσθέσει 3 primary keys στους εξής πίνακες:
   - Εγγράφονται σε υπηρεσίες
   - Λαμβάνουν υπηρεσίες
   - Επισκέπτονται

Ο κύριος λόγος είναι πως στην διαδικασία update/delete βοήθησαν στην βέλτιστη υλοποίησή τους. Επίσης, κατά την εισαγωγή δεδομένων αντιμετωπίσαμε την δυσκολία των duplicate entries, η οποία λύθηκε με την παραπάνω προσθήκη.

Τελικό ER Diagram, το οποίο βασίζεται στο ‘ER - Απλή Λύση’ που δόθηκε από τους διδάσκοντες, και περιέχει τις τροποποιήσεις που αναφέρθηκαν παραπάνω.



**1.** Το σχεσιακό διάγραμμα που προκύπτει με βάση το τελικό ER είναι το ακόλουθο:




**a.** Για τους περιορισμούς μπορούμε να πούμε:
• Τα κλειδιά παρουσιάζονται στο παραπάνω σχεσιακό διάγραμμα. καθώς και στο αρχείο All Tables για τον ορισμό των πινάκων. Με τα primary keys μια σχέση θα έχει σε αυτό το πεδίο μοναδική τιμή για κάθε εγγραφή της, η οποία δεν είναι null. Με τα foreign keys μια σχέση θα πρέπει να έχει σε αυτό το πεδίο μία τιμή, η οποία υπάρχει σε κάποια εγγραφή της σχέσης αναφοράς (της σχέσης από την οποία προέρχεται το κλειδί) και χρησιμοποιήθηκαν σε σχέσεις ένα προς πολλά για την ικανοποίηση περιορισμών ακεραιότητας.
• Οι περιορισμοί πεδίου τιμών παρουσιάζονται στο αρχείο All Tables που ορίζουν τους πίνακες παρακάτω.
• Για τα triggers έχουμε:

/* Εξασφαλίζει ότι, μόλις διαγραφεί ένας πελάτης, ανανεώνεται -παίρνοντας την τιμή null στην μεταβλητή nfc_id- το nfc_id στο receiving_services που αφορά τον πελάτη. Επιλέξαμε να είναι επιτρεπτή η τιμή null στο nfc_id του πίνακα receiving_services, διότι η πληροφορία κάθε receiving_services είναι απαραίτητη για τα ερωτήματα εύρεσης επαφών πιθανού κρούσματος. Όπως επίσης και για την εξαγωγή στατιστικών για τον τελευταίο μήνα και χρόνο. Σε πλήρη αντιστοιχία με τα παραπάνω ενημερώνουμε και τον πίνακα visiting. */

```
delimiter $$

CREATE TRIGGER delete_customer BEFORE DELETE ON customer
        FOR EACH ROW
    BEGIN
        UPDATE visiting SET nfc_id = NULL WHERE nfc_id = OLD.nfc_id;
        UPDATE receiving_services SET nfc_id = NULL WHERE nfc_id = OLD.nfc_id;
    END$$
delimiter ;

/* Εξασφαλίζει ότι μετά την εισαγωγή στον πίνακα λαμβάνει υπηρεσίες θα γίνει
σύγχρονη προσθήκη των χρεώσεων στον πίνακα του πελάτη. */

delimiter $$

CREATE TRIGGER get_new_charges AFTER INSERT ON receiving_services
        FOR EACH ROW
    BEGIN
    UPDATE customer SET total_charges = total_charges + NEW.service_fee WHERE nfc_id
= NEW.nfc_id;
    END$$
delimiter;

/*Εξασφαλίζουμε πως ακόμα και αν διαγραφεί ένας πελάτης από την βάση μας ,
επειδή έχει αποχωρήσει από το ξενοδοχείο και δεν θέλουμε να κρατάμε τα στοιχεία
του , ότι δεν θα χαθούν τα records από τα tables visiting και receiving_services
καθώς αυτό θα επέφερε λάθος αποτελέσματα στα ερωτήματα με τις όψεις . */
```

**b.** Ευρετήρια

```
-- 3 Main indexes of our database
-- nfc_id / facility_id / service_id
CREATE INDEX nfc_id_idx ON receiving_services(nfc_id);
CREATE INDEX facility_id_idx ON having_access(facility_id);
CREATE INDEX service_id_idx  ON receiving_services(service_id);
```

Δεδομένου ότι τα primary keys λειτουργούν ήδη ως indexes, διαλέξαμε τα παραπάνω τρία indexes. Αποτελούν τρία συχνά χρησιμοποιούμενα κλειδιά, στα tables receiving_services και having_access που παρατηρούμε ότι καλούνται συχνά στα επόμενα ερωτήματα.

**c.** Χρησιμοποιήθηκε SQL είναι το MySQL Workbench 8.0 CE για την υλοποίηση της βάσης δεδομένων, Apache Web Server για την επικοινωνία server-client, PHP και μικρή χρήση JavaScript για το server side της εφαρμογής, HTML και μικρή χρήση CSS για το client side της εφαρμογής. Οι JavaScript και CSS χρησιμοποιήθηκαν μέσω της βιβλιοθήκης Bootstrap. Για την PHP κάναμε χρήση του editor, Visual Studio Code.

**d.** Για να εγκατασταθεί η εφαρμογή μας θα πρέπει να εγκατασταθούν στον υπολογιστή σας τα εξής:

- MySQL Workbench 8.0 CE
- Apache Web Server 2.4

• Php 8.0
• https://www.mockaroo.com/ (χρησιμοποιήθηκε για την παραγωγή δεδομένων της βάσης μας)

Όταν έχουν εγκατασταθεί αυτά, στον φάκελο **C:\Apache24\htdocs** θα πρέπει να μεταφέρετε τα περιεχόμενα του φακέλου μας project που περιέχει όλα τα απαραίτητα αρχεία.
Ανοίγοντας cmd στην τοποθεσία **C:\Apache24\bin**, πληκτρολογούμε την εντολή **httpd -k start**. Έπειτα, στο MySQL Workbench τρέχουμε τα αρχεία με την εξής σειρά:

- Database Creation.sql - για να δημιουργήσουμε τη βάση
- All Tables.sql - ολόκληρο για να δημιουργήσουμε τα tables
- Views.sql - (Γραμμές 19-τέλος) για να δημιουργήσουμε τα views
- Triggers.sql - ολόκληρο για να δημιουργήσουμε τα triggers
- Insert.sql - ολόκληρο για να κάνουμε insert δεδομένα

Τώρα μπορείτε να τρέξει η εφαρμογή μέσω του browser σας γράφοντας στην μπάρα διεύθυνσης localhost (ή 127.0.0.1).

**2.** /*Αρχικά φτιάξαμε ένα αρχείο "Database Creation", προκειμένου να δημιουργούμε τη βάση δεδομένων μας στην περίπτωση που δεν υπάρχει.*/

```
CREATE DATABASE IF NOT EXISTS project2021;
USE project2021;

show databases;

DROP DATABASE project2021
```

/* Εν συνεχεία, έχουμε στο παραδοτέο .zip file ένα αρχείο με όλους τους πίνακες οι οποίοι χρειαζονται για την βάση μας. Κάνουμε χρήση της εντολής use project2021 σε όλα τα αρχεία, προκειμένου να γλυτώνουμε χρήση ονομασίας τύπου project2021.customer. */

```
USE project2021;
```

/* Πίνακας για τους πελάτες. Μεταβλητή nfc_id τύπου INT η οποία είναι το primary key, αλλά και auto_increment. Δηλαδή, δεν απαιτεί εισαγωγή δεδομένων και από μόνη της αυξάνεται κατά 1, αρχίζοντας απο το 1. Επώνυμο, όνομα, τύπος εγγράφου ταυτοποίησης κι αρχή έκδοσης ταυτοποίησης, σε μορφή varchar με μέγιστο αριθμό χαρακτήρων 20 bits. Η μεταβλητή ημερομηνία γέννησης (date_of_birth) σε μορφή date ('YYYY-MM-DD') η οποία δεν πρέπει να λαμβάνει μηδενική τιμή. Τέλος, έχουμε την μεταβλητή αριθμός εγγράφου ταυτοποίησης τύπου ακεραίου και την μεταβλητή συνολικές χρεώσεις που θα είναι κι αυτή ακέραιας μορφής, αλλά και μη μηδενική. */

```
CREATE TABLE customer (
   nfc_id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(20),
   last_name VARCHAR(20),
   date_of_birth DATE NOT NULL,
   id_number INT,
   id_type VARCHAR(20),
   publish_dept_id VARCHAR(20),
```

```
    total_charges int not null
);
```

/* Για τον πίνακα email, εισάγουμε μια μεταβλητή τύπου varchar με μέγιστο αριθμό αποδεκτών bits ίσο με 100, την οποία και ορίζουμε ως πρωτεύον κλειδί. Εν συνεχεία, την μεταβλητή nfc_id ακέραιας μορφής, η οποία αντιπροσωπεύει τον μοναδικό αριθμό κάθε πελάτη που αντιστοιχεί με το bracelet διαμονής στο ξενοδοχείο. Επίσης, θέτουμε σαν εξωτερικό κλειδί του πίνακα email, το nfc_id που αναφέρεται στον πίνακα πελάτες και του αναθέτουμε την λειτουργία αλληλουχίας εντολής στην περίπτωση update/delete. */

```
CREATE TABLE email (
        email VARCHAR(100) ,
        nfc_id INT ,
        PRIMARY KEY (email),
        FOREIGN KEY (nfc_id) REFERENCES customer(nfc_id) ON UPDATE CASCADE ON DELETE
CASCADE
);
```

/* Για τον πίνακα phone_number εισάγουμε μια μεταβλητή τύπου varchar (phone_num) με μέγιστο αριθμό αποδεκτών bits ίσο με 15 την οποία κι ορίζουμε ως πρωτεύον κλειδί. Εν συνεχεία, την nfc_id ακέραιας μορφής, η οποία αντιπροσωπεύει τον μοναδικό αριθμό κάθε πελάτη που αντιστοιχεί με το bracelet διαμονής στο ξενοδοχείο. Επίσης, θέτουμε σαν εξωτερικό κλειδί του πίνακα phone_number, το nfc_id που αναφέρεται στον πίνακα πελάτες και του αναθέτουμε την λειτουργία αλληλουχίας εντολής στην περίπτωση update/delete.*/

```
CREATE TABLE phone_number(
   phone_num VARCHAR(15) NOT NULL UNIQUE,
   nfc_id INT ,
   PRIMARY KEY (phone_num),
   FOREIGN KEY (nfc_id) REFERENCES customer(nfc_id) ON UPDATE CASCADE ON DELETE
CASCADE
);
```

/* Για τον πίνακα υπηρεσίες θέτουμε την μεταβλητή service_id ακέραιου τύπου και πρωτεύον κλειδί. Επίσης, εισάγουμε την μεταβλητή περιγραφή υπηρεσίας ως varchar(τύπος δεδομένων ενός πεδίου σε ένα σύστημα διαχείρισης βάσης δεδομένων που μπορεί να περιέχει γράμματα και αριθμούς) με μέγιστο αριθμό αποδεκτών bits 50. */

```
CREATE TABLE services(
   service_id INT PRIMARY KEY,
   service_description VARCHAR(50)
);
```

/* Για τον πίνακα facilities θέτουμε τη μεταβλητή facility_id ακέραιου τύπου και πρωτεύον κλειδί. Με παρόμοια λογική και σε πλήρη αντιστοιχία με το παραδοτέο σχεσιακό διάγραμμα μας δημιουργήσαμε τον πίνακα εγκαταστάσεις/facilities. Αξίζει να αναφέρουμε την λειτουργία αλληλουχίας εντολής στην περίπτωση update/delete στο εξωτερικό κλειδί service_id. Κάτι τέτοιο γίνεται προκειμένου να γίνεται ταυτόχρονη ανανέωση δεδομένων. Με άλλα λόγια, είναι μια μορφή ενδοεπικοινωνίας των πινάκων μας, η οποία κρατάει ενήμερα τα απαραίτητα κοινά στοιχεία στην βάση δεδομένων μας.*/

```
CREATE TABLE facilities(
   facility_id INT PRIMARY KEY,
   number_of_bedrooms INT NOT NULL,
   facility_name VARCHAR(50),
   facility_description varchar(50),
   service_id INT NOT NULL,
   FOREIGN KEY (service_id) REFERENCES services(service_id) ON UPDATE CASCADE ON DELETE
CASCADE);
```

/* Για τον πίνακα having_access (έχουν πρόσβαση) θέτουμε την μεταβλητή nfc_id ακέραιου τύπου και πρωτεύον κλειδί. Η μεταβλητή facility_id είναι ακέραιου τύπου. Οι μεταβλητές ημερομηνία-ώρα έναρξης (start_datetime) και ώρα ημερομηνία-ώρα λήξης (finish_datetime) σε μορφή date ('YYYY-MM-DD') η οποία δεν πρέπει να λαμβάνει μηδενική τιμή. Επίσης, θέτουμε ως εξωτερικά κλειδιά του πίνακα having_access, το nfc_id που αναφέρεται στον πίνακα πελάτες και το facility_id που αναφέρεται στον πίνακα facilities. Τους αναθέτουμε την λειτουργία αλληλουχίας εντολής στην περίπτωση update/delete.*/

```
CREATE TABLE having_access (
   nfc_id INT ,
   facility_id INT ,
   start_datetime DATETIME NOT NULL ,
   finish_datetime DATETIME NOT NULL ,
   PRIMARY KEY (nfc_id, facility_id) ,
   FOREIGN KEY (nfc_id) REFERENCES customer(nfc_id) ON UPDATE CASCADE ON DELETE
CASCADE ,
   FOREIGN KEY (facility_id) REFERENCES facilities(facility_id) ON UPDATE CASCADE ON DELETE
CASCADE);
```

/* Για τον πίνακα sub_compulsory_service (υπηρεσίες που απαιτούν εγγραφή στην καρτέλα πελάτη προκειμένου να γίνει χρήση τους) έχουμε θέσει σαν primary key και ακέραιου τύπου την μεταβλητή service_id. Αλλά την έχουμε θέσει ταυτόχρονα και ως εξωτερικό κλειδί το οποίο αναφέρεται στις υπηρεσίες και σε περίπτωση update/delete ενημερώνεται στον πίνακα services. */

```
CREATE TABLE sub_compulsory_service (
   service_id INT PRIMARY KEY ,
   FOREIGN KEY (service_id) REFERENCES services(service_id) ON UPDATE CASCADE ON DELETE
CASCADE);
```

/* Για τον πίνακα subscribing_to_services (εγγράφονται σε υπηρεσίες) θέτουμε τη μεταβλητή subscribing_to_services_id ακέραιου τύπου και πρωτεύον κλειδί.
Η μεταβλητή sub_date είναι της μορφής datetime και παίρνει τιμή μέσω του current timestamp. Αξίζει να αναφέρουμε την λειτουργία αλληλουχίας εντολής στην περίπτωση update/delete στα εξωτερικά κλειδιά nfc_id και service_id. Κάτι τέτοιο γίνεται προκειμένου να γίνεται ταυτόχρονη ανανέωση δεδομένων. Με άλλα λόγια, είναι μια μορφή ενδοεπικοινωνίας των πινάκων μας, η οποία κρατάει ενήμερα τα απαραίτητα κοινά στοιχεία στην βάση δεδομένων μας. */

```
CREATE TABLE subscribing_to_services (
   subscribing_to_services_id INT PRIMARY KEY AUTO_INCREMENT ,
   service_id INT ,
   nfc_id INT ,
   sub_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
   FOREIGN KEY (nfc_id) REFERENCES customer(nfc_id) ON UPDATE CASCADE ON DELETE
CASCADE,
   FOREIGN KEY (service_id) REFERENCES sub_compulsory_service(service_id) ON UPDATE
CASCADE ON DELETE CASCADE);
```

/* Ο πίνακα receiving_services (λαμβάνουν υπηρεσίες) αρχικά ήταν weak entity που είχε ως PK τα δυο FKs του. Ωστόσο, είχαμε ανάγκη να μπορούμε να προσθέτουμε στοιχεία στο πίνακα που θα είχαν ίδια FKs καθώς δυο άτομα, δηλαδή δύο διαφορετικά nfc_id, μπορει να λαμβάνουν την ίδια υπηρεσία, δηλαδή το ίδιο service_id. Επομένως, αναθεωρήσαμε τον πίνακα αυτόν ως strong entity που θα έχει το δικό του primary key (receiving_services). Με αυτόν τον πίνακα μπορούμε να κρατάμε πληροφορία σχετικά με το ποιές υπηρεσίες έλαβε ο κάθε πελάτης, το πότε τις έλαβε, τι ακριβώς υπηρεσία έλαβε και το τι κόστος είχε. */

```
CREATE TABLE receiving_services (
   receiving_services_id INT PRIMARY KEY AUTO_INCREMENT ,
   service_id INT ,
   nfc_id INT ,
   charge_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   charge_description VARCHAR(30),
   service_fee INT NOT NULL,
   FOREIGN KEY (nfc_id) REFERENCES customer(nfc_id) ON UPDATE CASCADE ON DELETE
CASCADE,
   FOREIGN KEY (service_id) REFERENCES services(service_id) ON UPDATE CASCADE ON DELETE
CASCADE);
```

/* Ο πίνακας visiting (χώροι που έχει επισκεφθεί ο πελάτης) μας δίνει τις πληροφορίες για το ποιοι χώροι έχουν επισκεφθεί από ποιους πελάτες και πότε. Είναι αντίστοιχος του receiving services καθώς και αυτός αρχικά ήταν weak entity όμως αναγκαστικά να του δώσουμε ενα PK και να τον κάνουμε strong entity για ακριβώς την ίδια αιτία , δηλαδή δύο άτομα να επισκεφθούν τον ίδιο χώρο ή το ίδιο άτομο να επισκεφθεί τον ίδιο χώρο πολλές φορές .
*/

```
CREATE TABLE visiting (
   visiting_id INT PRIMARY KEY AUTO_INCREMENT ,
   nfc_id INT ,
   facility_id INT ,
   entrance_datetime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ,
   exit_datetime DATETIME NOT NULL ,
   FOREIGN KEY (nfc_id) REFERENCES customer(nfc_id) ON UPDATE CASCADE ON DELETE
CASCADE ,
   FOREIGN KEY (facility_id) REFERENCES facilities(facility_id) ON UPDATE CASCADE ON DELETE
CASCADE
);
```

## Insert:

Στο υποβληθέν zip έχουμε τοποθετήσει όλα τα απαραίτητα δεδομένα εισαγωγής στη βάση σε ένα αρχείο με όνομα Insert.sql. Προκειμένου να έχουμε αρκετά δεδομένα που εισάγουμε στην βάση μας κάναμε χρήση της ιστοσελίδας mockaroo.com, με την βοήθεια της οποίας παράξαμε αρκετά data ώστε να είναι όλα τα ερωτήματα επαρκώς καλυμμένα, ενώ σε πολλές περιπτώσεις τροποποιήσαμε “χειροκίνητα” τα δεδομένα προκειμένου να είναι πιο ρεαλιστικά.

## Update – Delete:

/* Σε αυτό το αρχείο έχουμε δοκιμάσει την ορθή λειτουργία ενημέρωσης και διαγραφής δεδομένων σε όλους τους πίνακες της βάσης μας. Αρχικά, επιλέγουμε τον πίνακα επιθυμητής ενημέρωσης και την μεταβλητή αλλαγής (συμβατά δεδομένα με τον τύπο μεταβλητών). Έχοντας όμως σαν σημείο αναφοράς το κλειδί που ακολουθεί μετά τον τελεστή where. Με την ίδια λογική έχουν δομηθεί όλες οι παρακάτω συναρτήσεις. */

```
USE project2021;

-- UPDATE Function customer
 UPDATE customer SET
            first_name = 'Eldor'
    WHERE nfc_id = 50;

-- SELECT * FROM customer;
DELETE FROM customer WHERE nfc_id = 18;

-- ------------------------------------

-- UPDATE Function services
 UPDATE services SET
            service_description = 'Beverage'
    WHERE service_id = 1;

-- SELECT * FROM services;
DELETE FROM services WHERE service_description = 'Drink';

-- ------------------------------------

-- UPDATE Function facilities
 UPDATE facilities SET
            number_of_bedrooms = 4
    WHERE facility_id = 50;

-- SELECT * FROM facilities;
DELETE FROM facilities WHERE facility_description = 'West';

-- ------------------------------------

-- UPDATE Function email
 UPDATE email SET
            email = 'test50@ntua.gr',
            nfc_id = 50 WHERE email = 'vrisley1d@mashable.com';

-- SELECT * FROM email;
DELETE FROM email WHERE nfc_id = 50;

 -- ------------------------------------

-- UPDATE Function phone_number
UPDATE phone_number SET
            phone_num = '6144950249',
    nfc_id = 50 WHERE phone_num = '7144950243';

-- SELECT * FROM phone_number;
 DELETE FROM phone_number WHERE nfc_id = 50;
```

```
-- ----------------------------------------

-- UPDATE Function having_access
 UPDATE having_access SET
            finish_datetime = '2021-07-21 15:15:05'
    WHERE nfc_id = 50;

-- SELECT * FROM having_access;
DELETE FROM having_access WHERE nfc_id = 50;

-- ------------------------------------

-- SELECT * FROM sub_compulsory_service;
DELETE FROM sub_compulsory_service WHERE service_id = 7;

-- ------------------------------------------------------------------------------------------------------

-- UPDATE Function subscribing_to_services
UPDATE subscribing_to_services SET
            service_id = 5,
    sub_date = '2021-12-12' WHERE nfc_id = 50;
-- SELECT * FROM subscribing_to_services;
 DELETE FROM subscribing_to_services WHERE nfc_id = 50;

-- ----------------------------------------------------------

-- UPDATE Function receiving_services
UPDATE receiving_services SET
            service_id = 5,
    charge_date = '2021-12-12 16:54:30' WHERE nfc_id = 50;

-- SELECT * FROM receiving_services;
 DELETE FROM receiving_services WHERE nfc_id = 50;

-- ------------------------------------------------------

-- UPDATE Function visiting
UPDATE visiting SET
    exit_datetime = '2021-12-12 16:54:30'
    WHERE nfc_id = 50;

-- SELECT * FROM visiting;
 DELETE FROM visiting WHERE nfc_id = 50;

-- ------------------------------------------------------
```

## Views:

```
USE project2021;

-- Drop views
drop view Available_Services;
drop view Yphresies;
drop view sales;
drop view thecustomers;
drop view visits;
drop view danger;
drop view in_danger;
drop view min;
```

```
drop view custo_age_group;
drop view time_diff;
drop view most_visited;
drop view time_diff_2;
drop view most_serviced;
drop view serviced_most;

USE project2021;

-- Ερωτημα 7 ------------------------------------------------------------------------------------
-- View for the all the available services of our hotel
create view Available_Services as
select * from services;

select * from Available_Services;

-- Service fee and charge date for each used service
create view Yphresies as
select receiving_services.service_id, receiving_services.nfc_id, services.service_description,
receiving_services.charge_description, receiving_services.service_fee,
receiving_services.charge_date FROM receiving_services
left join services
on receiving_services.service_id = services.service_id;

select * from Yphresies
order by nfc_id;

-- Ερωτημα 8 ------------------------------------------------------------------------------------
```

/* Για το view sales χρησιμοποιήσαμε την πληροφορία από δύο tables τα receiving_services και services . Στο table receiving_services κρατάμε την πληροφορία του ποιος πελάτης αγόρασε ποιά υπηρεσία και με τι κόστος . Για να βρούμε ποιά είναι η υπηρεσία θα πρέπει να κάνουμε LEFT JOIN με το services με βάση το service_id */

```
-- View for sales
create view sales as
select receiving_services.service_id,
receiving_services.charge_description,services.service_description,sum(receiving_services.service_f
ee) as total_fees, count(receiving_services.nfc_id) AS num_of_sales FROM receiving_services
left join services
on receiving_services.service_id = services.service_id
group by service_id
order by service_id;

select * from sales;
```

/* Για το view thecustomers η πληροφορία που μας ζητάει το υποερώτημα είναι όλα τα στοιχεία των πελατών. Επομένως, χρειάζεται να συννενώσουμε την πληροφορία που υπάρχει στα tables customer, email και phone_num. Για τον σκοπό αυτό έχουμε κάνει LEFT JOIN στον customer και στο email με βάση το nfc_id ώστε να πάρουμε το email του κάθε nfc_id και ακριβώς το ίδιο με το phone_num ώστε να πάρουμε τον αριθμό τηλεφώνου που έχει καταχωρηθεί στη βάση. */

```
-- Views for all the necessary data of each customer
create view thecustomers as
```

```
select customer.nfc_id, customer.first_name, customer.last_name, customer.date_of_birth,
customer.id_number,
customer.id_type, customer.publish_dept_id , email.email, phone_number.phone_num,
customer.total_charges from customer
left join email
on customer.nfc_id = email.nfc_id
left join phone_number
on customer.nfc_id = phone_number.nfc_id;

SELECT * FROM  thecustomers;

-- Ερωτημα 9 ------------------------------------------------------------------------------------------
-- View in case of new COVID-19 patient which tracks entrance and exit datetimes from each service
create view visits as
select visiting.nfc_id, visiting.facility_id, facilities.facility_name, facilities.service_id,
facilities.facility_description,
visiting.entrance_datetime, visiting.exit_datetime from visiting
left join facilities
on facilities.facility_id = visiting.facility_id;

select * from visits;

-- View for a specific positive customer
create view danger as
select * from visits;
-- where nfc_id = 2;
select * from danger;
```

/* Για το view που αφορά ποιους χώρους έχει χρησιμοποιήσει ο πελάτης που βρέθηκε θετικός στον κορονοϊό έχουμε δημιουργήσει το view visits που κρατάει όλες τις επισκέψεις που έχουν γίνει από όλους τους πελάτες σε όλους τους χώρους όπως και το πότε εισήλθαν και το πότε αναχώρησαν από τον κάθε χώρο. Για τον σκοπό αυτό έχουμε κάνει LEFT JOIN ανάμεσα στα tables visiting και facilities για να βρούμε το όνομα του χώρου που χρησιμοποιήθηκε από τον κάθε πελάτη με βάση το facility_id του χώρου. Μέσω του UI ο χρήστης θα έχει την δυνατότητα να μπορεί να επιλέξει τον πελάτη που τον ενδιαφέρει δηλαδή το nfc_id του και από εκεί να βρούμε το ζητούμενο . */

```
-- Ερωτημα 10 ------------------------------------------------------------------------------------------
-- Customers who are in danger
create view in_danger as
select visits.exit_datetime as sick_person_left, visiting.entrance_datetime, visiting.nfc_id, visits.nfc_id
as CoV2_positive from visits,visiting
where visiting.nfc_id <> visits.nfc_id
order by nfc_id,Cov2_positive;

select * from in_danger;

create view min as
select CoV2_positive,nfc_id,entrance_datetime,sick_person_left,Minutes
from (
    select CoV2_positive, nfc_id, entrance_datetime, sick_person_left,
    TIMESTAMPDIFF(minute, entrance_datetime, sick_person_left) AS 'Minutes'
    from in_danger
) as joined
where Minutes <= 60 and Minutes >= -60
order by nfc_id;
```

```
select * from min;
```

/* Για να βρούμε ποιοί πελάτες βρίσκονται σε κίνδυνο υπάρχουν δύο στάδια. Το πρώτο είναι να φτιάξουμε ένα view στο οποίο θα κρατάμε όλους τους χώρους που έχει επισκεφθεί ο κάθε πελάτης, όπως και το πότε αποχώρησε από τον χώρο αυτό. Επιπλέον, χρειάζεται να κρατήσουμε την χρονική στιγμή που εισήλθε στον ίδιο χώρο ένας άλλος πελάτης (στο σημείο αυτό κάνουμε select ανάμεσα στα δύο tables). Στην ουσία το sick_person_left αφορά όλους τους πελάτες αφού ακόμα δεν γνωρίζουμε ποιός είναι αυτός που όντως είναι θετικός στον κορονοϊό. Το δεύτερο στάδιο είναι να δημιουργήσουμε ένα view στο οποίο θα κρατάμε μόνο τις γραμμές του view in_danger στις οποίες η διαφορά χρόνου μεταξύ εξόδου ύποπτου ατόμου και είσοδου επόμενου πελάτη είναι 60 λεπτά. Στην ουσία το δεύτερο view υπάρχει για να δείξουμε την πρόθεσή μας να κρατήσουμε μόνο αυτές τις γραμμές. Τέλος μέσω του UI μπορούμε να υποδηλώσουμε ποιο είναι το άτομο που είναι θετικός στον κορονοϊό και άρα και ποια άλλα άτομα κινδυνεύουν . */

```
-- Ερωτημα 11 ------------------------------------------------------------------------------------
-- All Customer Ages-----------------------------------------------------------------------------
create view custo_age_group as
select nfc_id,
CASE
WHEN age between 19 AND 40 THEN '20-40'
WHEN age between 41 AND 60 THEN '41-60'
ELSE '61+'
END AS Team
from (select nfc_id, TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) as age, null as Team from
customer) as xx;

select * from custo_age_group;

-- 11A Last Year Visits--------------------------------------------------------------------------
create view time_diff as
select nfc_id,facility_id,months
from (
        select visiting.nfc_id, visiting.facility_id, month('2021-12-30') - month(visiting.exit_datetime) -
(DATE_FORMAT('2021-12-30', '%m%d') <  DATE_FORMAT(visiting.exit_datetime, '%m%d')) as
months,
   year('2021-12-30') - year(visiting.exit_datetime) - (DATE_FORMAT('2021-12-30', '%m%d')
<  DATE_FORMAT(visiting.exit_datetime, '%m%d')) as years
   from visiting
) sub
where years = 0
order by years desc;

select * from time_diff;

-- 11B Most Visited Spaces ----------------------------------------------------------------------
create view most_visited as
select joined.nfc_id, joined.facility_id, facilities.facility_name, Team, months
from (
   select time_diff.nfc_id, custo_age_group.Team, time_diff.facility_id, time_diff.months
   from time_diff
   left join custo_age_group
   on time_diff.nfc_id = custo_age_group.nfc_id
) as joined
```

```
left join facilities
on joined.facility_id = facilities.facility_id;

select * from most_visited;
```

/*Παραθέτουμε το αντίστοιχο query που χρησιμοποιήσαμε στην php για να πάρουμε τα σωστά δεδομένα όπου για παράδειγμα έχουμε πάρει τον τελευταίο ένα χρόνο και την ηλικιακή ομάδα 20 έως 40 */

```
select facility_id, facility_name, visits from (
select facility_id, facility_name, count(nfc_id) as visits, Team from most_visited where months<=12
group by Team, facility_id
) as x
where Team = '20-40';

-- 11C Most Serviced Select + View
create view time_diff_2 as
select nfc_id,service_id, months
from (
        select receiving_services.nfc_id, receiving_services.service_id, month('2021-12-30') -
month(receiving_services.charge_date) - (DATE_FORMAT('2021-12-30', '%m%d')
<  DATE_FORMAT(receiving_services.charge_date, '%m%d')) as months,
   year('2021-12-30') - year(receiving_services.charge_date) - (DATE_FORMAT('2021-12-30',
'%m%d') <  DATE_FORMAT(receiving_services.charge_date, '%m%d')) as years
   from receiving_services
) sub
where years = 0
order by years desc;

select * from time_diff_2;

create view most_serviced as
select joined.nfc_id, joined.service_id, Team, months, services.service_description
from (
   select time_diff_2.nfc_id, custo_age_group.Team, time_diff_2.service_id, time_diff_2.months
   from time_diff_2
   left join custo_age_group
   on time_diff_2.nfc_id = custo_age_group.nfc_id
) as joined
left join services
on joined.service_id = services.service_id;

select * from most_serviced;
```

/*Παραθέτουμε το αντίστοιχο query που χρησιμοποιήσαμε στην php για να πάρουμε τα σωστά δεδομένα όπου για παράδειγμα έχουμε πάρει τον τελευταίο ένα χρόνο και την ηλικιακή ομάδα 20 έως 40 */

```
select service_id, service_description, visits from (
select service_id, service_description, count(nfc_id) as visits, Team from most_serviced where
months<=12 group by Team, service_id
) as x
where Team = '20-40';

-- Serviced most
create view serviced_most as
select distinct(joined.nfc_id), joined.service_id, Team, months, services.service_description
```

```
from (
   select time_diff_2.nfc_id, custo_age_group.Team, time_diff_2.service_id, time_diff_2.months
   from time_diff_2
   left join custo_age_group
   on time_diff_2.nfc_id = custo_age_group.nfc_id
) as joined
left join services
on joined.service_id = services.service_id;

select * from serviced_most;
```

/*Παραθέτουμε το αντίστοιχο query που χρησιμοποιήσαμε στην php για να πάρουμε τα σωστά δεδομένα όπου για παράδειγμα έχουμε πάρει τον τελευταίο ένα χρόνο και την ηλικιακή ομάδα 20 έως 40 . Η διαφορά με το προηγούμενο ερώτημα είναι πως θέλουμε να μετράμε την υπηρεσία που έχει χρησιμοποιηθεί από τον ίδιο πελάτη πολλές φορές μόνο μια και για αυτό στο query χρησιμοποιούμε την εντολή count(**distinct** nfc_id) */

```
select service_id, service_description, customers from (
select service_id, service_description, count(nfc_id) as customers, Team from serviced_most where
months<=12 group by Team, service_id
) as x
where Team = '20-40';
```

**3.** Στον φάκελο **C:\Apache24\htdocs** θα πρέπει να μεταφερθούν όλα τα απαραίτητα αρχεία. Τα ζητούμενα αρχεία για την δημιουργία της βάσης στην mysql αλλά και όλοι οι κώδικες της php για το UI περιέχονται στο zip υποβολής.

Table 1 Περιεχόμενα video:

| **Ερώτημα** | **Αρχή** | **Τέλος** |
|---|---|---|
| (a) | 00:09 | 00:31 |
| (b) | 00:32 | 01:02 |
| (c) | 01:03 | 01:50 |
| (d) | 01:57 | 02:34 |
| (e) | 02:35 | 03:17 |
| (f).7 | 03:24 | 04:02 |
| (f).9 | 04:03 | 04:40 |
| (f).10 | 04:41 | 05:25 |
| (f).11 | 05:27 | 08:36 |
| (g) | 08:39 | 09:24 |