

**CST FACULTY**

**Advanced Database**

**Practical Project |** Aulon Morina

**Perpilimi I orarit te ligjeratave**

|  |  |  |  |
| --- | --- | --- | --- |
| Group X | | | |
| **Nr** | **SID** | **Name** | **Surname** |
| 1 | 160306020 | Aulon | Morina |
| 2 |  |  |  |
| 3 |  |  |  |

**Database Systems**

DB-2020

**Project Template 1.2[[1]](#endnote-1)**

# Perpilimi I Orarit te ligjeratave ne FSHK

Bache

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Group X | | | |  | Points | | | |  |
| **Nr** | **SID** | **Name** | **Surname** |  | Part 1 | Part 2 | Part 3 | Add. | **Total** |
| 1 | 160306020 | Aulon | Morina |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |

Note:Bachelor Students | max 3 in group, Online & PT students| Individual, Master Students| max 2 in group

**PART I**

# Detailed description of the project

Describe your problem. About what it is?

Why you have chosen this project?

What you want to solve, and in which way you will solve it?

Background research. Description about any existing system for that purpose. (You can use images for illustration).

Problemi ne kete projekt ka te beje me perpilimin e orarit te ligjeratave ne FSHK.   
Projekti shfrytzohet nga tre perdoures te ndryshem dhe me funksione te ndryshme ne sistem si Administratori, Profesori dhe Studenti.

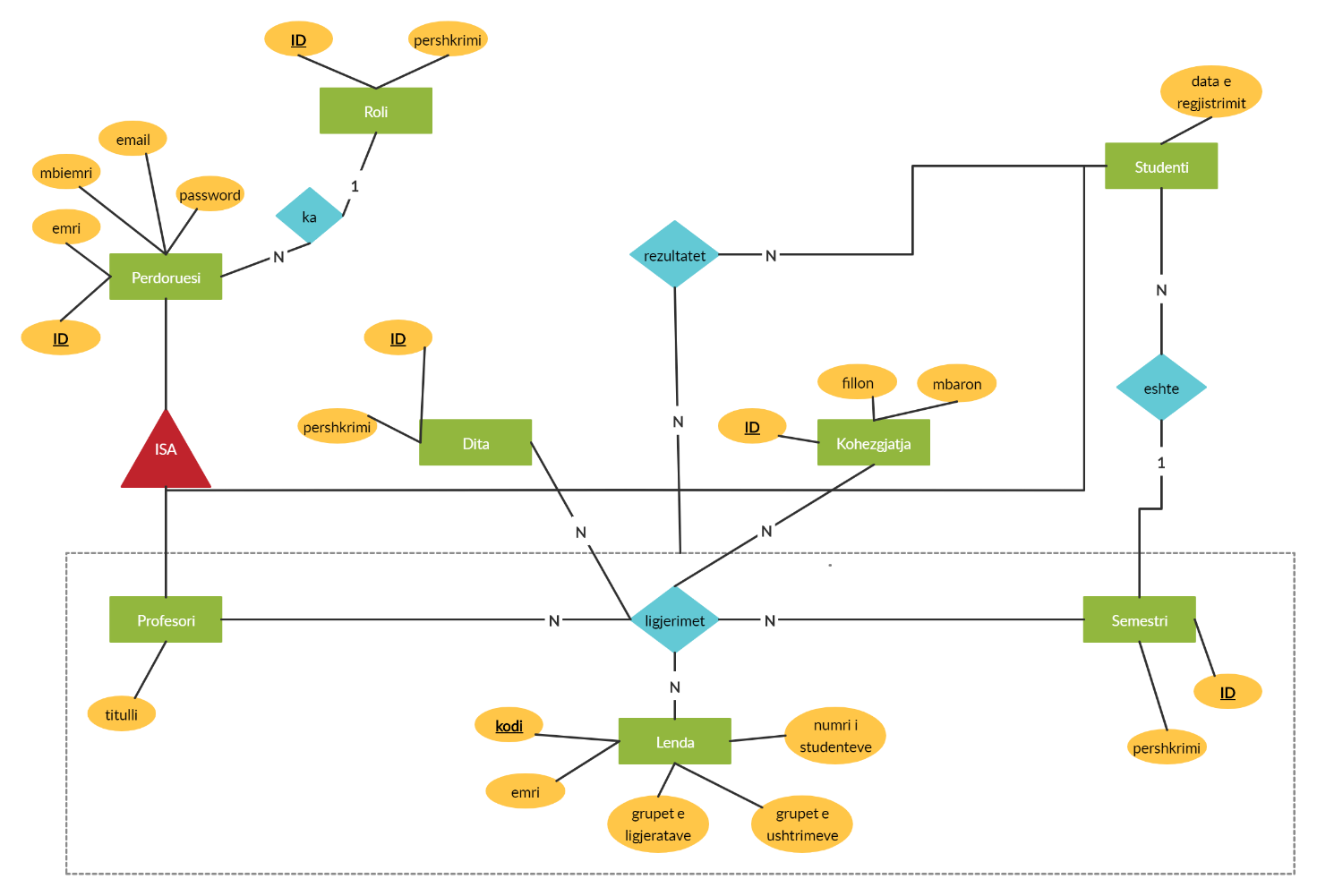
Administratori ka te drejte te manipuloje me te dhenat( fshirje, editim apo shtm) te tyre.   
  
Profesoret mund ti shohin lendet e tyre perkatese dhe orarin e ligjeratave.

Studentet mund ti regjistrojne lendet edhe ta shohin orarin per semestrin akual ne te cilin ndodhen.

Sistem I ngjashem eshte edhe SMU ( Sistemi Menaxhues I Universitetit) pra ne menyre elektronike por ajo eshte shume me teper komplekse per dallim nga ky mini-projekt.

# E-R model

Per kete projekt (database) kam dizajnuar nje E-R diagram ku shfaqet logjika e funksionit te saj ne teresi.



# Explain all the constrains

Te gjitha relacionet shprehin lidhjen mes tabelave permes te cilave kryhen “operacione” te ndryshme rreth databazes, shtimit, editimit apo fshirjes se N-sheve..

Secila tabele permban nje qeles primar permes se ciles identifikohen kolonat tjera te tabelave perkatese dhe keshtu eliminohet redudanca.

Shembull te nje relacioni specific ne databaze kemi lidhjen IS-A ndermjet tre aktereve kryesore te databazes te cilet jane Perdoruesit, por keta Perdorues mund te kene role varesisht se si jane regjistruar paraprakisht, pra tri perdoruesit mund te jene **Profesor**, **Student** apo **Administrator**.

Administratori nuk eshte paraqit ne kete E-R diagram pasiqe aty nuk mund te insertohen permes UI, regjistrimi I tyre behet direkt nga databaza ndersa Profesoret dhe Studentet jane te paraqitur.

Lidhje tjeter e participimit te disa celesve primar edhe tek lidhja aggregate ndermjet rezultatet dhe ligjerimeve sepse logjikisht vlen si lidhje dhe participim per arsye se rezultatet paraqesin tabelen finale kur nje student regjistron nje apo disa lende.

Ne te gjitha tabelat participuese qeles primar kane nga nje **ID.**

# E-R to relational model

Modeli relacionar per kete databaze eshte si ne vijim:

**Roli** (ID (pk), pershkrimi )

**Perdoruesi** (ID (pk), emri, mbiemri, email, password, roli (fk) )

**Studenti** (ID (pk), viti I regjistrimit, semestri (fk))

**Profesori** (ID (pk), titulli)

**Semestri** (ID (pk), pershkrimi )

**Lenda** (kodi (pk), emri, grupet e ligjeratave, grupet e ushtrimeve, numri i studenteve )

**Dita** (ID (pk), pershkrimi )

**Kohezgjatja** (ID (pk), fillon, mbaron )

**rezultatet** ((lenda (pk + fk), profesori (pk + fk) ),semestri (pk + fk), studenti (pk + fk), )

**ligjerimet**(dita (pk + fk),lenda (pk + fk), profesori (pk + fk), semestri (pk + fk), kohezgjatja(pk + fk) )

**PART II**

# Implementing the project

1. Implement the relational model in any DBMS. For this step you can use MS SQL Server or Oracle.

CREATE TABLE roli(

id INT NOT NULL,

pershkrimi CHAR(20),

PRIMARY KEY(id)

)

CREATE TABLE perdoruesi (

id INT NOT NULL AUTO\_INCREMENT UNIQUE,

emri CHAR(20),

mbiemri CHAR(20),

email CHAR(40),

fjalekalimi CHAR(32),

roli INT NOT NULL,

PRIMARY KEY(id),

FOREIGN KEY(roli) REFERENCES roli(id)

)

CREATE TABLE semestri (

id INT NOT NULL,

pershkrimi CHAR(20),

PRIMARY KEY(id)

)

CREATE TABLE studenti (

id INT NOT NULL AUTO\_INCREMENT UNIQUE,

vitiRegjistrimit DATE,

semestri INT,

PRIMARY KEY(id),

FOREIGN KEY(id) REFERENCES perdoruesi(id),

FOREIGN KEY(semestri) REFERENCES semestri(id)

)

CREATE TABLE profesori (

id INT NOT NULL AUTO\_INCREMENT UNIQUE,

titulliAkademik CHAR(30),

PRIMARY KEY(id),

FOREIGN KEY(id) REFERENCES perdoruesi(id)

)

CREATE TABLE lenda (

kodi INT NOT NULL AUTO\_INCREMENT UNIQUE,

emri CHAR(30),

grupetEligjeratave INT,

grupetEushtrimeve INT,

numriStudenteve INT,

PRIMARY KEY(kodi)

)

CREATE TABLE dita(

id INT NOT NULL,

pershkrimi CHAR(20),

PRIMARY KEY(id)

)

CREATE TABLE kohezgjatja(

id INT NOT NULL AUTO\_INCREMENT,

fillimiOres TIME,

mbarimiOres TIME,

PRIMARY KEY(id)

)

CREATE TABLE ligjerimet (

dita INT,

lenda INT,

profesori INT,

kohezgjatja INT,

semestri INT,

PRIMARY KEY(dita,lenda, profesori, semestri, kohezgjatja),

FOREIGN KEY(lenda) REFERENCES lenda(kodi),

FOREIGN KEY(profesori) REFERENCES profesori(id),

FOREIGN KEY(semestri) REFERENCES semestri(id),

FOREIGN KEY(dita) REFERENCES dita(id),

FOREIGN KEY(kohezgjatja) REFERENCES kohezgjatja (id) )

CREATE TABLE rezultatet (

lenda INT,

profesori INT,

semestri INT,

studenti INT,

regjistruar CHAR,

PRIMARY KEY(lenda, profesori, semestri, studenti),

FOREIGN KEY(lenda) REFERENCES lenda(kodi),

FOREIGN KEY(profesori) REFERENCES profesori(id),

FOREIGN KEY(semestri) REFERENCES semestri(id),

FOREIGN KEY(studenti) REFERENCES studenti(id)

)

1. Define indexes.

**INDEXAT** jane te paradefinuara ne baze te celesave primar dhe te huaj varesisht relacionit.

Keto indexe jane te klasteruara, sepse nuk kemi krijuar ndonje INDEX speficik per kerkim, permes SELECT Querive koha e kerkimit eshte me e madhe, prandaj nese deshirojm te krijojm kerkim me te shpejte varesisht per tabela te cilat tentojme te nxjerrim te dhena me se shpeshti duhet te krijojm index jo te klasteruar per kolonat specifike te ndonje tabele.

Shembull ne tabelen **perdoruesi:**

Te gjindet perdoruesi me emrin Zirije Hasani.

(Metoda e zakonshme e klasteruar)

**SELECT** id, emri **FROM** perdoruesi **WHERE** emri = “Zirije” **AND** mbiemri “Hasani”

**-------------------------------------------------------------------------**

(Metoda e indexit jo te klasterurar)

**CREATE INDEX** emri\_perdoruesit

**ON** perdoruesi (emri, mbiemri)

Disa index ate krijuara meposhte:

**CREATE INDEX** lendaINDEX

**ON** lenda (kodi, emri)

**CREATE INDEX** semestriINDEX

**ON** semestri (pershkrimi)

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

**CREATE INDEX** profesoriINDEX

**ON** profesori(id, titulliAkademik)

1. Given an example where **Unique** key is applied.

Celesi **UNIQUE** eshte perdor tek tri tabela: Studenti, Profesori dhe Lenda.

UNIQUE Constraint nuk lejon duplifikimin e te dhenave me vlere te njejte keshtu qe me kete rast ne shmangim p.sh dy student te kene ID te njejte apo dy profesore apo edhe dy lende. Logjikisht vlen jashtzakonisht.

CREATE TABLE lenda (

kodi INT NOT NULL AUTO\_INCREMENT UNIQUE,

emri CHAR(30),

grupetEligjeratave INT,

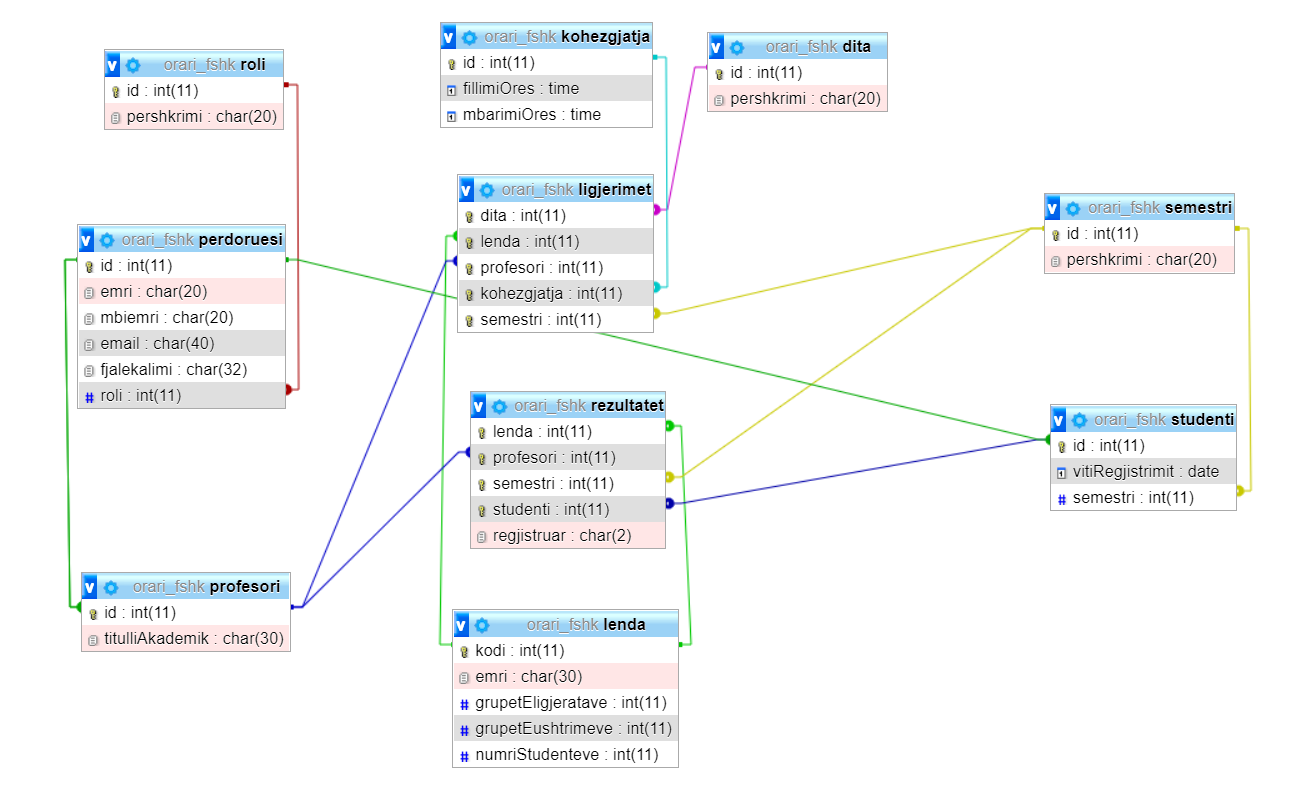
grupetEushtrimeve INT,

numriStudenteve INT,

PRIMARY KEY(kodi)

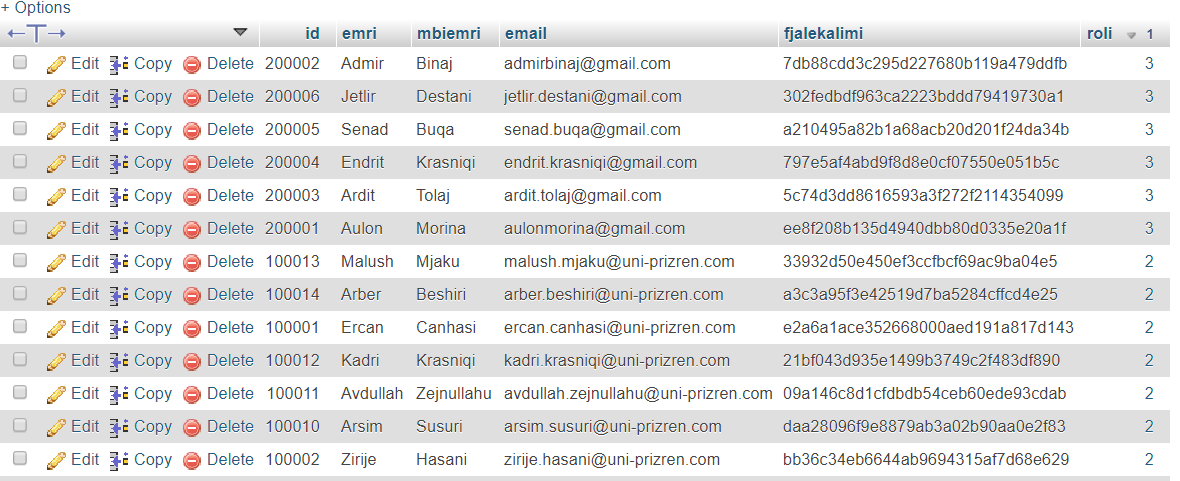
# Relational Diagram implemented in DBMS

Put here a print screen from your overall diagram.



Insert at least 10 records at each table. (Show small print-screens here)

**Tabela Perdoruesi**

****

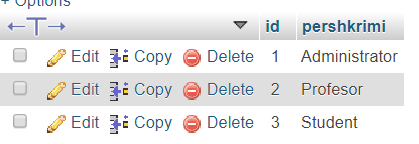
**Tabela Lenda**

****

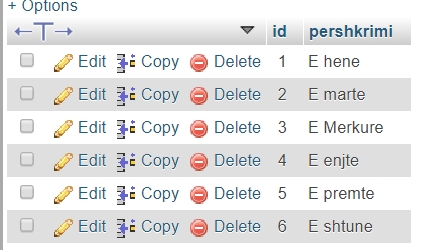
**Tabela ligjerimet**

****

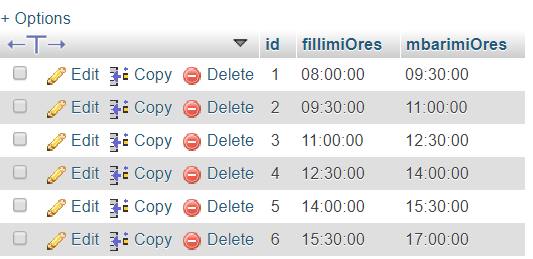
**Tabela roli**

****

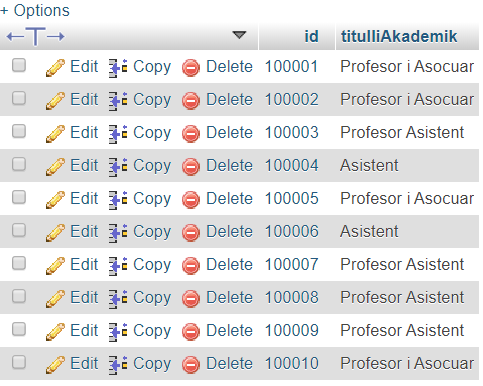
**Tabela dita**

****

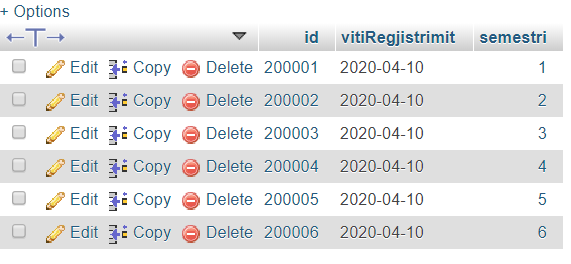
**Tabela kohezgjatja**

****

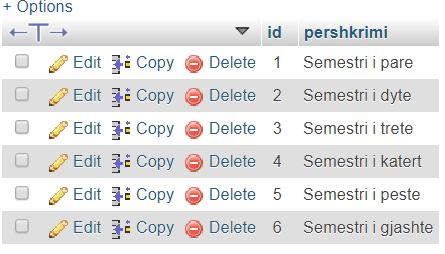
**Tabela profesori**

****

**Tabela studenti**

****

**Tabela semestri**

****

**Tabela rezultatet**

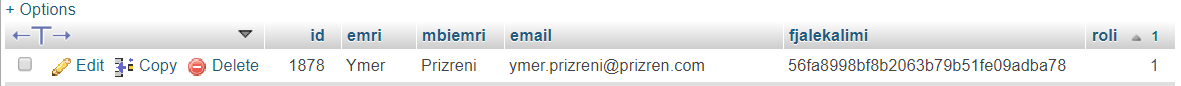
****

Give one example for each DML (INSERT, UPDATE and DELETE)

1. **INSERTIMI**

INSERT INTO perdoruesi (id, emri, mbiemri, email, fjalekalimi, roli)

VALUES ('1878', 'Ymer', 'Prizreni', 'ymer.prizreni@prizren.com', MD5('lidhja1878'), '1');

****

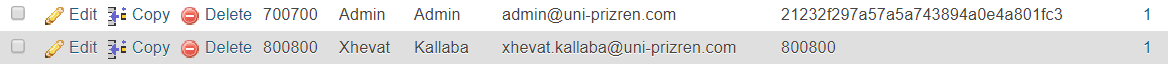
**2. UPDATE**

UPDATE rezultatet SET regjistruar = ‘”Po”

WHERE rezultatet.dita = 1 AND rezultatet.lenda = 101 AND rezultatet.profesori = 100001 AND rezultatet.semestri = 1 AND rezultatet.kohezgjatja = 1 AND rezultatet.studenti = 200001



**3. DELETE**

****

 DELETE FROM perdoruesi WHERE perdoruesi.id = 700700



# Integrity Constrains

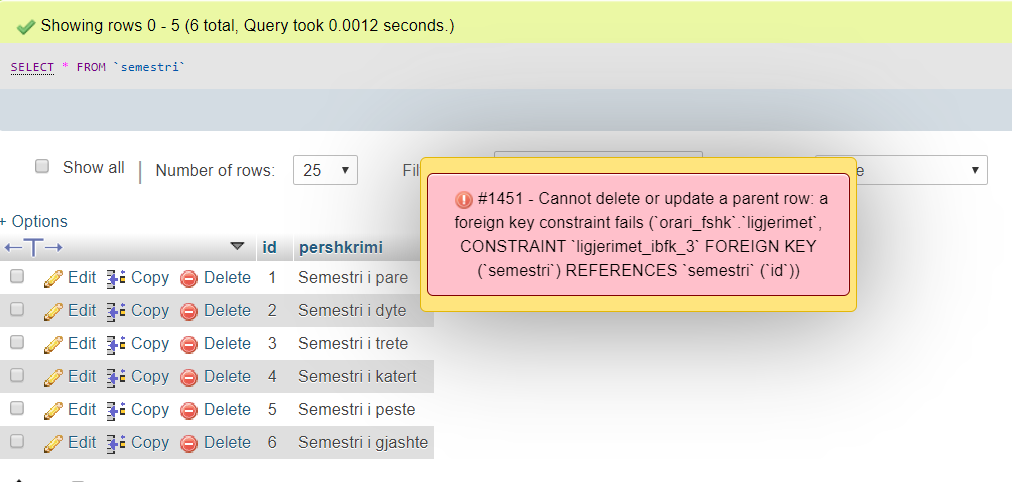
1. Explain the integrity constrains, relational integrity. Example, if you delete the primary key in a table, what will be happened with the records where the value of this key is used as a foreign key.

Tek tabelat Kohezgjatja, Lenda, Perdoruesi, kam perdor **AUTO\_INCREMENT** ne ID e tyre perkatese per arsye qe sa here te Insertohet nga nje rresht I ri, me automatizem I shtohet nje ID varesisht logjikes se eshte ajo ID e definuar. P.sh nese ID eshte e tipit INT dhe fillon prej 1, supozojme se kemi gjashte rreshta, me ID (1, 2, 3, 4, 5, 6) dhe tentojme te insertojme nje rresht pa e definuar ID, atehere rreshtit te shtate I vendoset ID 7. D.m.th rritet per nje.

Nese tentojme te shlyejme Celesat Primar ne disa tabela e te cilat jane po ashtu si Celes te huaj ne disa tabela tjera, nuk e lejon per shkak te integritetit dhe sigurise se te dhenave, mund te realizohet vetem nese shlyhen dy tabelat. Shembull

Celesat primare te tabelave nuk guxojne te mbeten te zbrazta (pa plotsuar) keshtu qe ato kane edhe **NOT NULL.**

Nese tentojme ta shlyjme ndonje qeles primar te nje tabele e qe eshte foreign key ne tabelen tjeter atehere kemi nje shembull konkret, p.sh: Tentojme ta shlyejme tek **tabela Semestri** njeren nga semestret, semestri eshte foreign key tek **tabela ligjerimet:**

****

**PART III**

# Queries(Views)

You need to have at least 5 simple queries and 10 complex queries, using different operators, nested queries, etc.

For each query you need to show.

1. The query requirement?
2. The Answer of the query in relational Algebra
3. The answer of the query in SQL
4. Print screen from the results generated by SQL

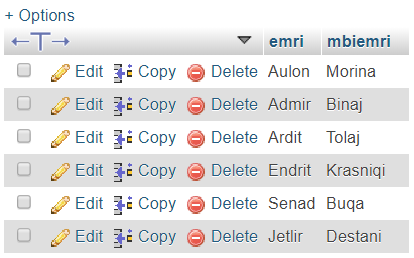
-------------------------------------------------------------------------------------------------------------------------

**1. Gjej perdoruesit te cilet kane rolin e studentit.**

a. **π** emri, mbiemri σ roli = 3 (Perdoruesi)

b. SELECT emri, mbiemri FROM perdoruesi WHERE roli = 3

c.



**2. Gjej lendet qe e permbajne fjalen “Hyrje”.**

a. σ emri LIKE “Hyrje%”(Lenda)

b. SELECT emri FROM lenda WHERE emri LIKE 'Hyrje%'

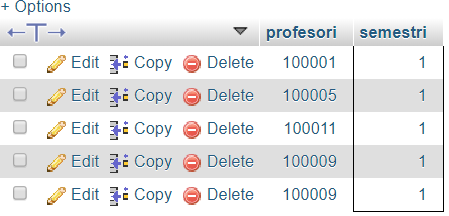
c.



**3. Gjej profesoret qe ligjerojne ne semestrin e pare.** a. **π** profesori,semestri .σ semestri = 1(Ligjerimet)

b. SELECT profesori, semestri FROM ligjerimet WHERE semestri = 1

c.

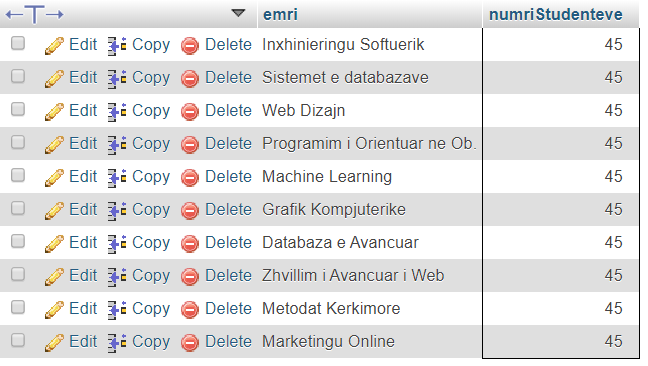
****

**4.** **Gjej te gjitha lendet, te cilat kane nje numer te studenteve aktiv 45.**

a. **π** profesori, numriStudenteve σ numriStudenteve = 45 (Lenda)

b. SELECT profesori, numriStudenteve FROM lenda WHERE numriStudenteve = 45

c.



**5. Gjej administratoret e sistemit.**

a. **π** emri, mbiemri, email σroli = 1 (Perdoruesi)

b. SELECT emri, mbiemri, email FROM perdoruesi WHERE roli = 1

c.



**6. Selekto te gjitha lendet qe ligjerohen ne diten e hene.**

a. **π** l.emri AS Lenda, d.pershkrimi AS Dita, semestri **σ**d.pershkrimi = ‘E hene’

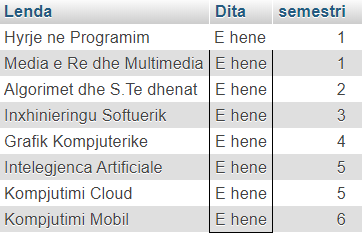
(**ρ**l(lenda) ⨝ **ρ** lgj(ligjerimet) ⨝ **ρ**d(Dita))

b. SELECT l.emri AS Lenda, d.pershkrimi AS Dita, semestri

FROM lenda l, ligjerimet lgj, dita d

WHERE l.kodi = lgj.lenda AND d.id = lgj.dita AND d.pershkrimi = "E hene"

c.

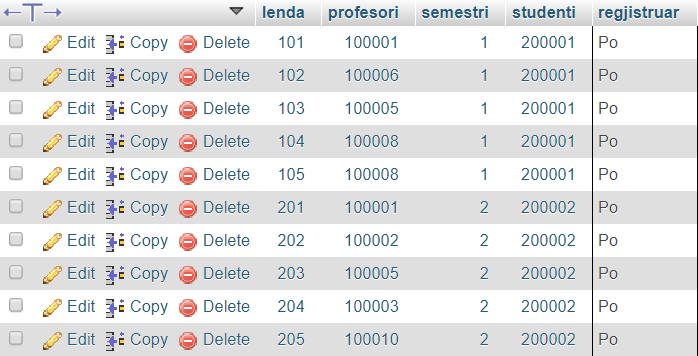
****

**7.Selekto studentet te cilet kane regjistruar ndonjeren nga lendet.**

a. **σ** regjistruar = “Po” (rezultatet)

b. SELECT \* FROM rezultatet WHERE regjistruar = "Po"

c.



**8. Gjej emrat e studenteve te cilet jane te regjistruar ne vitin 2019 dhe 2017.**

a. **π** CONCAT(pr.emri, ' ', pr.mbiemri) as Studenti ,s.vitiregjistrimit

**σ**s.vitiregjistrimit ≥ '2016-01-01' ∧ s.vitiregjistrimit ≤ '2020-01-01'

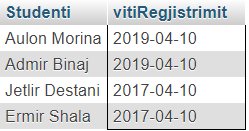
(**ρ**s(studenti) ⨝ **ρ** p(perdoruesi))

b. SELECT CONCAT(pr.emri, ' ', pr.mbiemri) as Studenti, vitiRegjistrimit

FROM studenti s, perdoruesi pr

WHERE s.vitiRegjistrimit >= '2016-01-01' AND s.vitiRegjistrimit <= '2020-01-01' AND s.id = pr.id

c.



**9. Gjej lendet te cilat ligjerohen ne orarin prej 08:00 deri ne ora 09:30.**

a. **π** l.emri, CONCAT(k.fillimiOres, k.mbarimiOres) AS Kohezgjatja **σ** kohezgjatja = 1

(**ρ**l(lenda) ⨝ **ρ** lgj(ligjerimet) ⨝ **ρ**k(kohezgjatja)) **γ** Lenda

b. SELECT l.emri AS Lenda, CONCAT(k.fillimiOres, ' - ', k.mbarimiOres) AS Kohezgjatja

FROM lenda l, ligjerimet lgj, kohezgjatja k

WHERE l.kodi = lgj.lenda AND kohezgjatja = 1 GROUP BY Lenda

c.



**10. Gjej mesataren e studenteve te regjistruar per te gjitha lendet semestrale.**

a. **π** AVG(numriStudenteve) as “mestarja e studenteve te regjistruar”(Lenda)

b. SELECT AVG(numriStudenteve) AS "Mesatarja e Studenteve Te Regjistruar" FROM lenda

c.



**11. Gjej lendet qe ligjerohen nga profesoresha Zirije.**

a. **π**l.emri AS Lenda, lgj.semestri as Semestri **σ** pr.emri = “Zirije”

(**ρ**l(lenda) ⨝ **ρ** lgj(ligjerimet) ⨝ **ρ**pr(perdoruesi)) ⨝ **ρ**p(profesori))

b. SELECT l.emri AS Lenda, lgj.semestri as Semestri

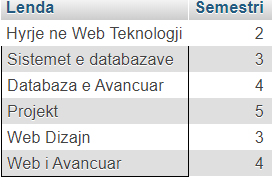
FROM lenda l, ligjerimet lgj

WHERE l.kodi = lgj.lenda AND lgj.profesori IN ( SELECT pr.id

FROM profesori p, perdoruesi pr

WHERE pr.id = p.id AND pr.emri = "Zirije");

c.



**12. Selekto profesoret qe kane titull akademik Profesor i Asocuar.**

a. . **π** CONCAT(pr.emri, ‘ ’ pr.mbiemri) AS Profesor, titulliAkademik

**σ** titulliAkademik = “Profesor i Asocuar”

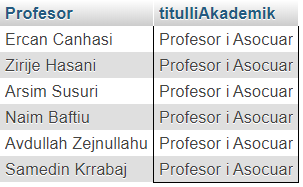
(**ρ**p(profesori) ⨝ **ρ** pr(perdoruesi))

b. SELECT CONCAT(pr.emri,' ', pr.mbiemri) AS Profesor, titulliAkademik

FROM profesori p, perdoruesi pr

WHERE p.id = pr.id AND titulliAkademik = "Profesor i Asocuar"

c.



**13. Gjej numrin e studenteve te cilet kane regjistruar lendet.**

a. **π** COUNT(\*)AS numriTotaliRegjistrimeve (Rezultatet)

b. SELECT COUNT(\*) AS NumriTotalIRegjistrimeve FROM rezultatet

c.



**14. Gjej lendet qe kane mbi 2 grupe te ushtrimeve.**

a. **π** l.kodi, l.emri, l.grupetEushtrimeve **σ** l.grupetEushtrimeve > 2 (Lenda)

b. SELECT l.kodi, l.emri, l.grupetEushtrimeve FROM lenda l WHERE l.grupetEushtrimeve > 2

c.



**15. Selekto studentet, emri I te cileve u fillon me shkronjen A.**

a. **π** pr.emri, pr.mbiemri, s.emestri **σ** pr.emri LIKE ‘A%’

(**ρ**pr(perdoruesi) ⨝ **ρ**s(studenti))

b. SELECT pr.emri, pr.mbiemri, s.semestri FROM perdoruesi pr, studenti s

WHERE pr.id = s.id AND pr.emri LIKE 'A%'

c.



# Normalization (Schema refinement)

**1NF -** Ne formen e pare normale cdo atribut duhet te kete vlere atomike pra vlera e saj nuk mund te dekompozohet. Shembull konkret ne kete database e kemi tek atributi Kohezgjatja mund te permbaje te dhena si fillimin e ores dhe mbarimin e ores, per ate arsye edhe e kam kriju tabelen e vecante **Kohezgjatja** e cila I permban keto atribute ID, FilimiOres, MbarimiOres dhe keshtu krijohet nje logjik specifike sa I perket kohes se kur mbahen ligjeratat e caktuara.

**2NF –** Ne formen e dyte normale cdo atribut jo celes eshte ne varshmeri te plote funksionale   
 prej celesit primar. Ne databazen qe e kam krijuar, secila tabele permban nga nje celes primar, pra automatikisht tabela jone e ploteson kushtin e Formes se dyte normale.

tblRoli ( **ID ->** pershkrimi )

tblPerdoruesi ( **ID, roli ->** emri, mbiemri, email, password)

tblStudenti ( **ID, semestri ->** viti I regjistrimit)

tblProfesori ( **ID ->** titulli)

tblSemestri ( **ID ->** pershkrimi )

tblLenda ( **kodi ->** emri, grupet e ligjeratave, grupet e ushtrimeve, numri i studenteve )

tblDita (**ID ->** pershkrimi )

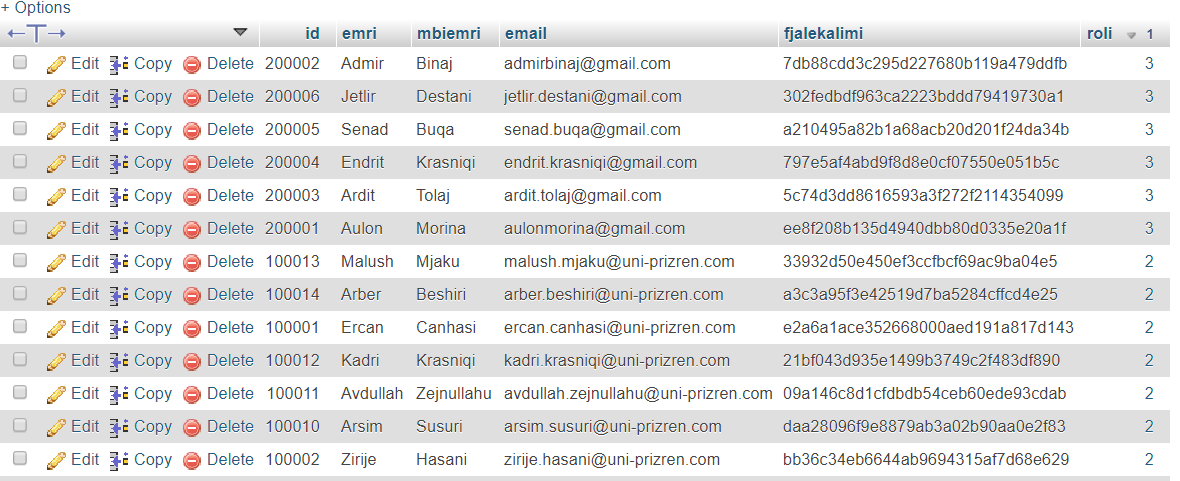
tblKohezgjatja ( **ID ->** fillimiOres, mbarimiOres )

tblrezultatet ((lenda (pk + fk), profesori (pk + fk) ), semestri (pk + fk), studenti (pk + fk), )

tblligjerimet(dita (pk + fk), lenda (pk + fk), profesori (pk + fk), semestri (pk + fk), kohezgjatja(pk + fk) )

**3NF –** Forma e trete normale nuk lejon perseritjen e disa atributeve, pra nuk lejohet redudanca. Shembull ku e kam menaxhuar kete situate ngjashem e kemi tek keto dy tabela ne vijim:

tabela **Perdoruesi**



**⭡ roli** si Foreign Key tek tabela e Perdoruesit

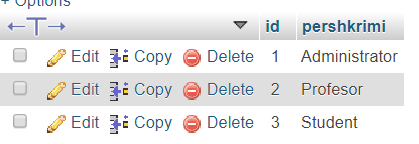


tabela **Perdoruesi**

# Programmability

Create at least two triggers (one FOR and one INSTEAD OF trigger)

Create stored procedure for Insert, Update and Delete.

Create at least one function

**TRIGGERS**

(Roli I ketij trigeri eshte qe pas insertimit te nje **perdoruesi,** ID e tij dhe te dhenat tjera specifike per ate perdorues te vendosen tek **tabela Studenti.**

NOW() paraqet nje funksion te gatshem qe merr daten aktuale(real-time) te regjistrimt.

DELIMITER //

CREATE TRIGGER shtoStudentin

AFTER INSERT ON `perdoruesi`

FOR EACH ROW

BEGIN

INSERT INTO `studenti` (id, vitiRegjistrimit, semestri)

VALUES (new.id, NOW(), '1');

END; //

DELIMITER ;

(Ky triger ka per qellim fshirjen e N-shes perpos tabeles Perdoruesi edhe tek tabela Studenti, dhe kjo nevojitet nese ID e njeres tabela eshte FOREIGN KEY tek tabela tjeter.

Pa kete trigger nuk mundet te ndodh diqka e tille, shfaqet errori I Constraint, per shkak te nivelit te larte te integritet qe vete sistemi I databazes e ka)

DELIMITER //

CREATE TRIGGER shlyejStudentin

AFTER DELETE ON `perdoruesi`

FOR EACH ROW

BEGIN

DELETE FROM studenti where id = OLD.id;

END; //

DELIMITER ;

**FUNCTION**

(Ky funksion na mundeson te shikojm vitin e studimeve te nje student te caktuar, duke selektuar vetem semestrin akual qe ai ndodhet)

DELIMITER //

CREATE FUNCTION vitiStudimeve(idSemestri INT)

RETURNS CHAR(20)

DETERMINISTIC

BEGIN

DECLARE vitiStudimeve CHAR(20);

IF idSemestri = 1 OR idSemestri = 2 THEN

SET vitiStudimeve = 'Viti i pare';

ELSEIF (idSemestri = 3 OR idSemestri = 4) THEN

SET vitiStudimeve = 'Viti i dyte';

ELSEIF (idSemestri = 5 OR idSemestri = 6) THEN

SET vitiStudimeve = 'Viti i katert';

END IF;

RETURN (vitiStudimeve);

END; //

DELIMITER ;

Shembull i perdorimit te ketij funksioni:

SELECT CONCAT(pr.emri,' ', pr.mbiemri) AS Studenti, vitiStudimeve(s.semestri) AS VitiStudimeve

FROM perdoruesi pr, studenti s

WHERE s.id = pr.id AND s.semestri = 1

Rezultati:



**STORE PROCEDURE**

**1. INSERT PROCEDURE**

(Kjo store procedure e inserton nje perdorues me te dhenat qe i jipen kurr thirret procedura)

DELIMITER //

CREATE PROCEDURE insertoPerdorues(IN p\_id INT, IN p\_emri CHAR(20), IN p\_mbiemri CHAR(20), IN p\_email CHAR(40), IN p\_fjalekalimi CHAR(32), IN p\_roli INT)

BEGIN

INSERT INTO `perdoruesi`(id, emri, mbiemri, email, fjalekalimi, roli)

VALUES (p\_id, p\_emri, p\_mbiemri, p\_email, MD5(p\_fjalekalimi), p\_roli);

END //

DELIMITER ;

CALL insertoPerdorues('123123', 'Lirdina', 'Kukaj', 'lina.bill@hotmail.com', MD5('123123'), '3');

**2. UPDATE PROCEDURE**

(Kjo store procedure ben update-im te te dhenave ne tabelen perdoruesi)

DELIMITER //

CREATE PROCEDURE ndryshoPerdorues(IN p\_id INT, IN p\_emri CHAR(20), IN p\_mbiemri CHAR(20), IN p\_email CHAR(40), IN p\_fjalekalimi CHAR(32))

BEGIN

UPDATE `perdoruesi`

SET emri = p\_emri, mbiemri = p\_mbiemri, email = p\_email, fjalekalimi = MD5(p\_fjalekalimi)

WHERE id = p\_id;

END //

DELIMITER ;

CALL ndryshoPerdorues('123123', 'Fati', 'Hasi', 'fatihasi@hotmail.com', MD5('123123'));

**3. DELETE PROCEDURE**

(Kjo store procedure e fshin N-shen nga tabela perdoruesi)

DELIMITER //

CREATE PROCEDURE fshijPerdorues(IN p\_id INT)

BEGIN

DELETE FROM `perdoruesi`

WHERE id = p\_id;

END //

DELIMITER ;

CALL fshijPerdorues(‘123123’);

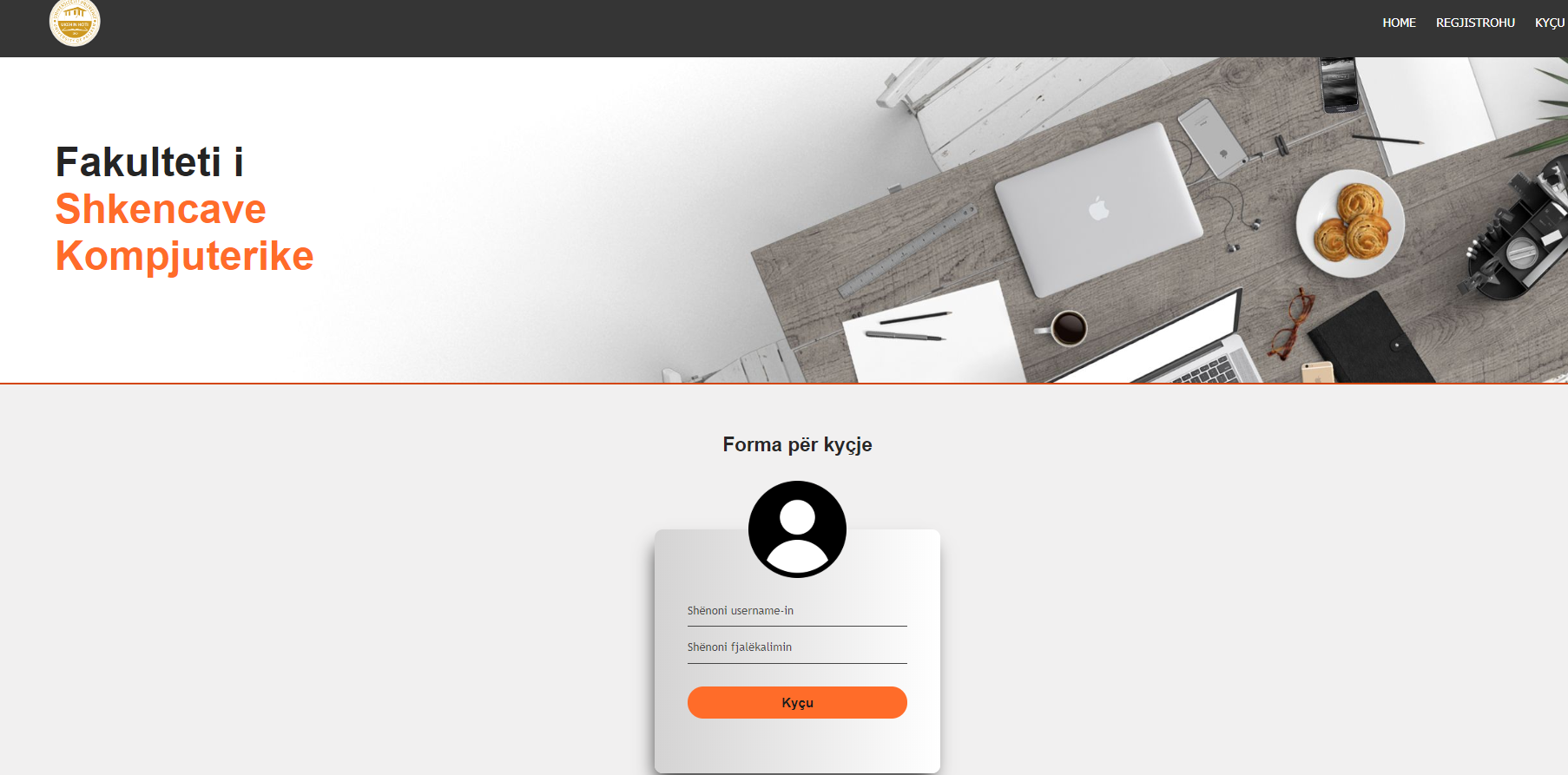
# GUI – Extra

You can design a GUI for performing any task in your application. I encourage you to use PHP.

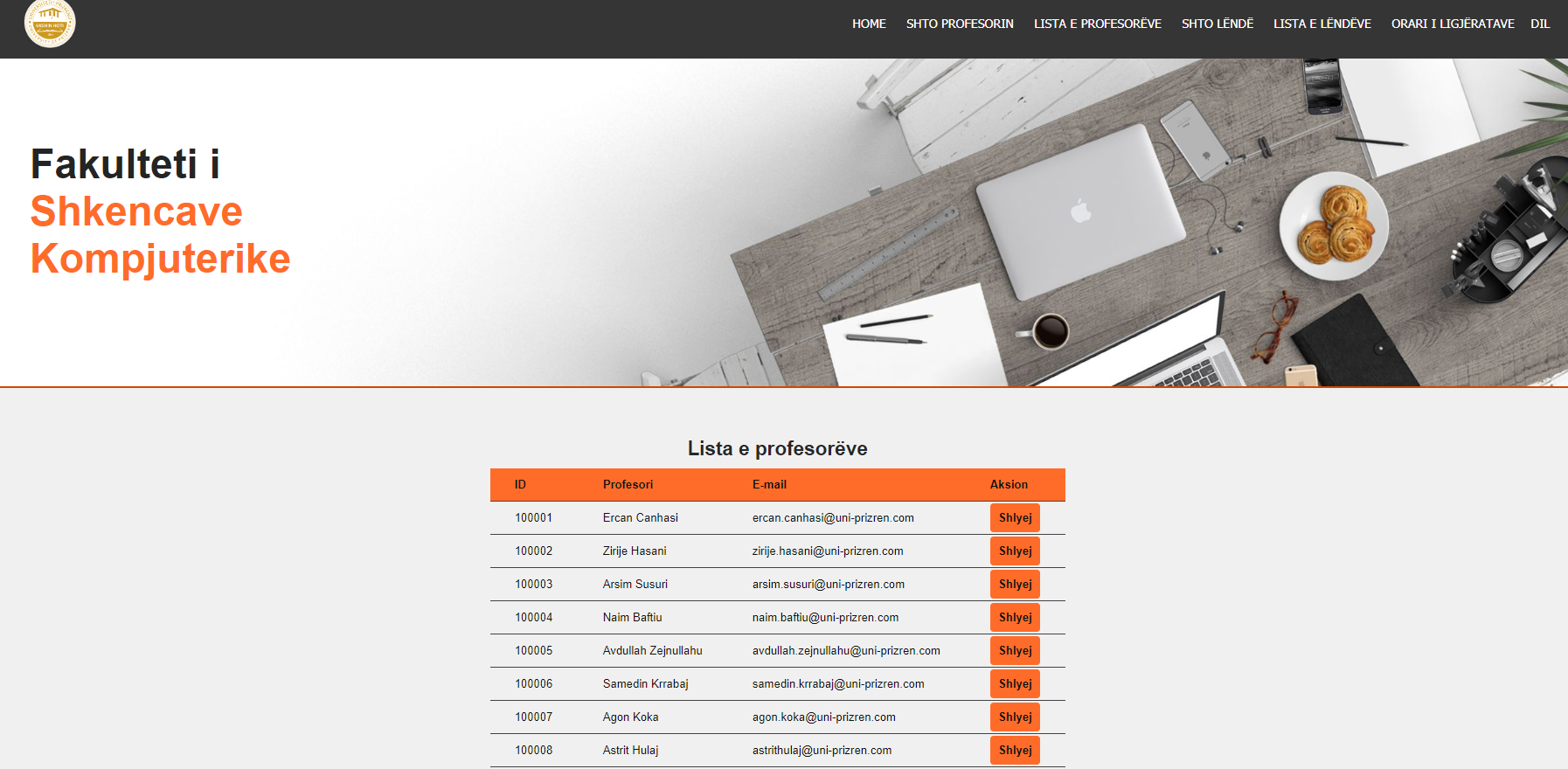
1. Disa nga Screenshots te meposhtme paraqesin interface-in e Web Aplikacionit te zhvilluar me

   **HTML, CSS, PHP.**

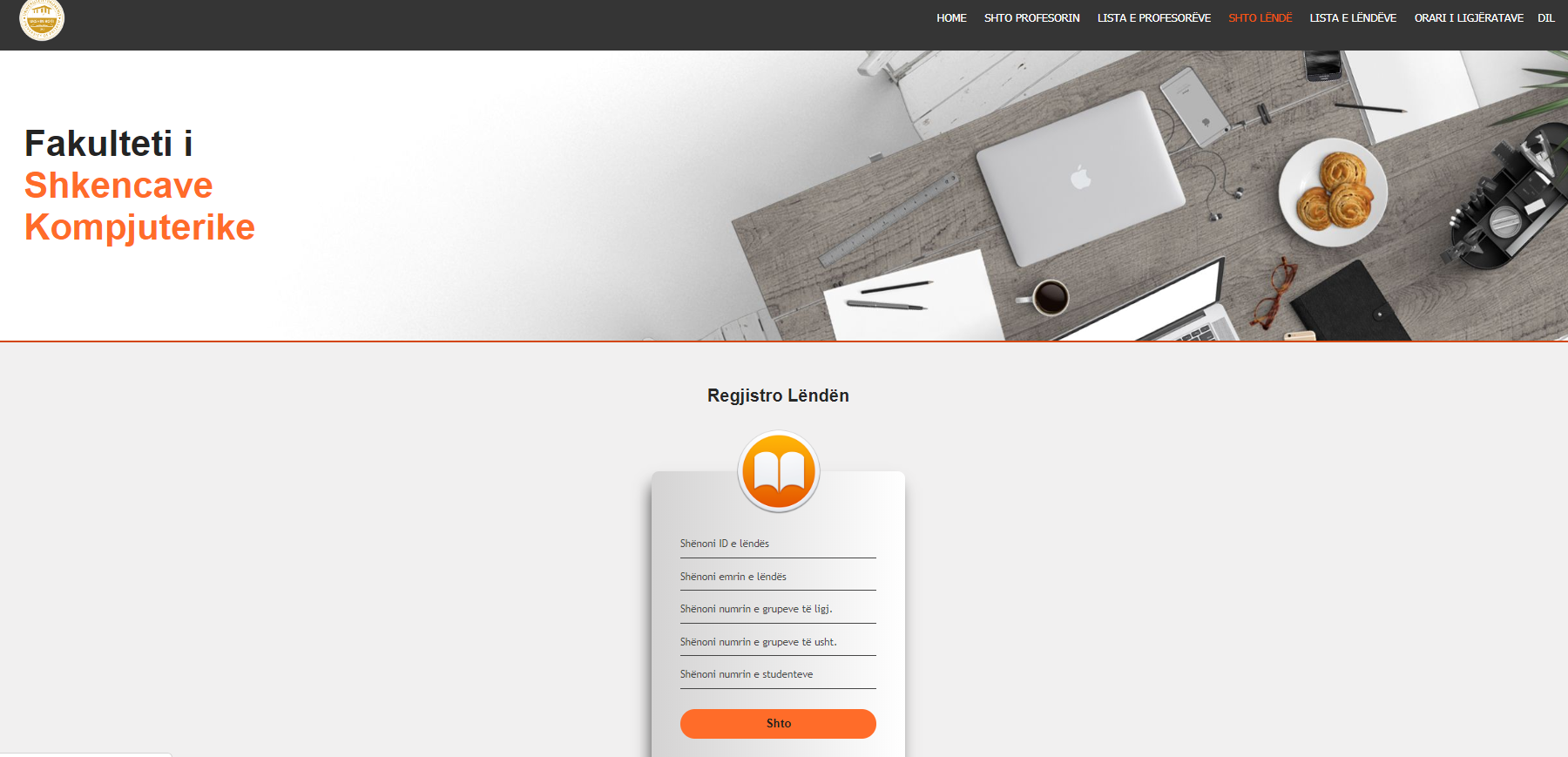
   **1. PJESA E LOG-IN**

   ****

   **2. LISTEN E PROFESOREVE ( ADMIN )**

   ****

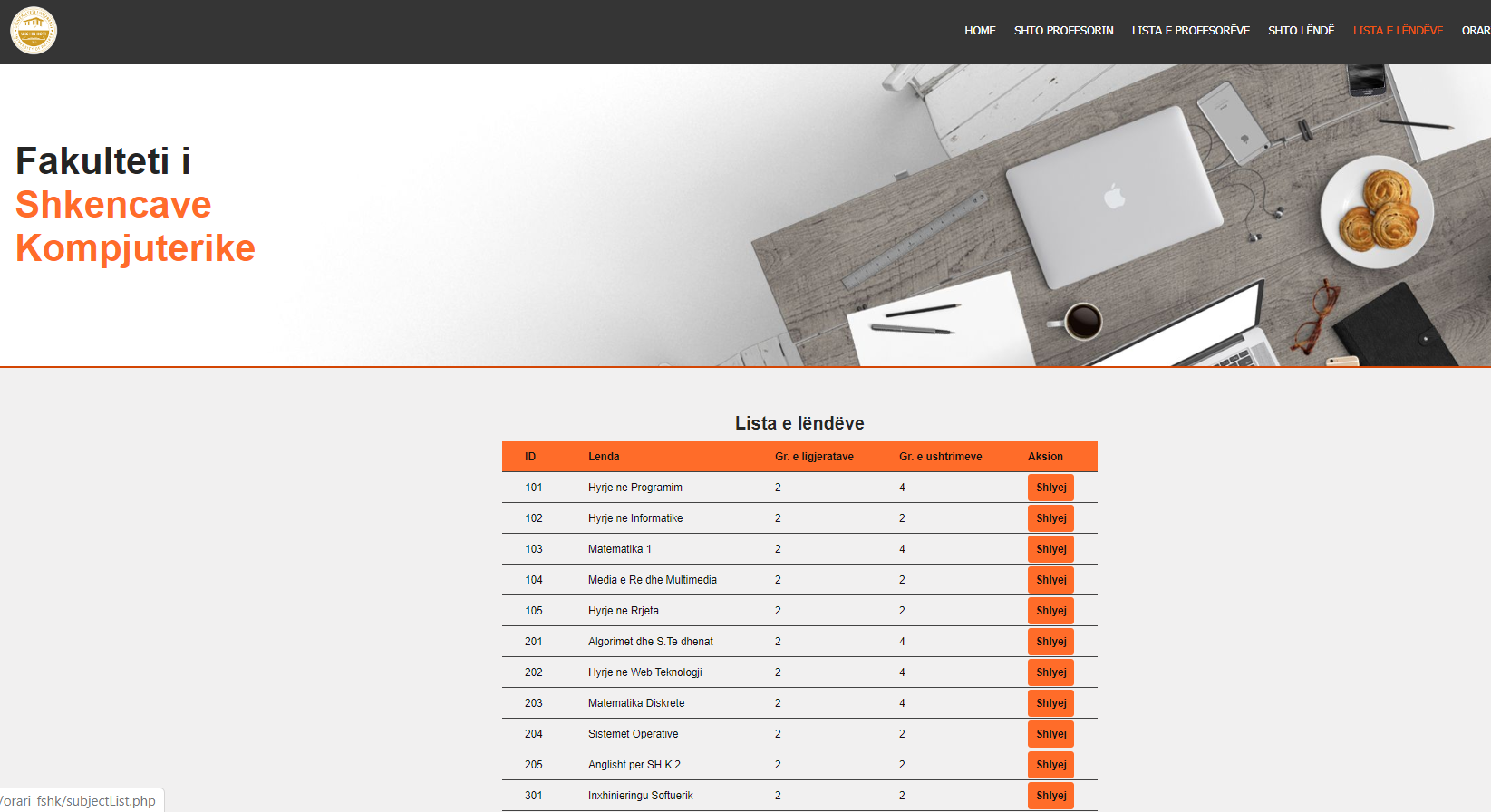
   **3. FORMA E REGJISTRIMT TE LENDEVE ( ADMIN )**

   ****

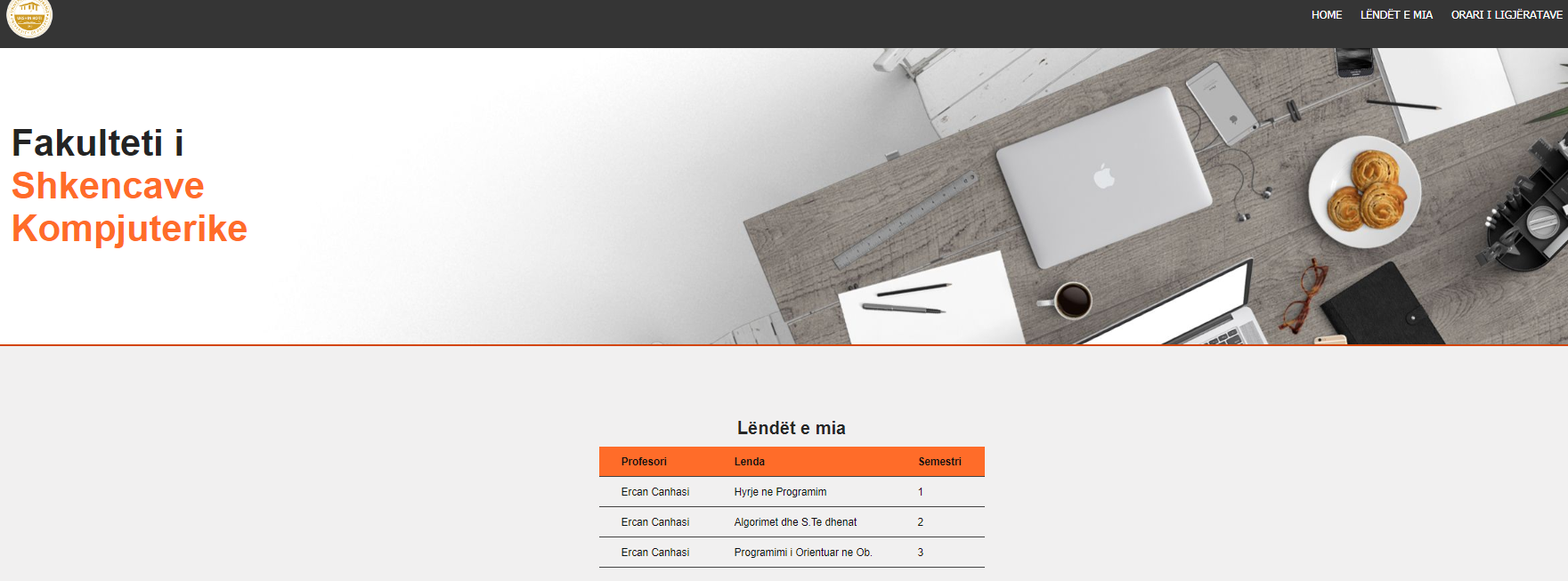
   **4. PAMJA E ORARIT TE LIGJERATAVE ( ADMIN DHE PROFESOR)**

   ****

   **5. PJESA E LENDEVE (ADMIN)**

   ****

   **6. LENDET E PROFESORIT (PROF)**

   **** [↑](#endnote-ref-1)