```
/**********************
***FOR LOCAL BD*****
*********************/

CREATE USER 'anonimo'@'localhost' IDENTIFIED BY '12345';
GRANT INSERT, DELETE, SELECT, UPDATE, EXECUTE ON *.* TO
'anonimo'@'localhost';
FLUSH PRIVILEGES;


/**********************
**********************
*********************/



CREATE TABLE Course(id_course int AUTO_INCREMENT, NRC varchar(15), period
varchar(25), name varchar(75), PRIMARY KEY (id_course));
CREATE TABLE Person(id_person int AUTO_INCREMENT, name varchar(75),
phoneNumber varchar(15), email varchar(65), id_course int, activity_state
enum('Active', 'Inactive'), PRIMARY KEY(id_person), FOREIGN KEY(id_course)
REFERENCES Course(id_course));
CREATE TABLE Coordinator(id_person int, cubicle int, staff_number varchar(25),
PRIMARY KEY(id_person), FOREIGN KEY(id_person) REFERENCES
Person(id_person) ON DELETE CASCADE);
CREATE TABLE Professor(id_person int, cubicle int, staff_number varchar(25),
PRIMARY KEY(id_person), FOREIGN KEY(id_person) REFERENCES
Person(id_person));
CREATE TABLE Company(id_company int AUTO_INCREMENT, name varchar(75),
address varchar(75), email varchar(65), state varchar(45), phoneNumber
varchar(15),direct_users int, indirect_users int, sector
enum('Primary','Secondary','Tertiary'), city varchar(65), id_coordinator int,
id_course int, PRIMARY KEY(id_company), FOREIGN KEY(id_coordinator)
REFERENCES Coordinator(id_person), FOREIGN KEY(id_course) REFERENCES
Course(id_course));
CREATE TABLE Project(id_project int AUTO_INCREMENT, name
varchar(75),duration float, schedule varchar(75), general_purpose text,
general_description text, id_company int, charge_Responsable varchar(35),
name_Responsable varchar(75), email_Responsable varchar(65),PRIMARY
KEY(id_project), FOREIGN KEY(id_company) REFERENCES
```

```sql
Company(id_company));
CREATE TABLE Project_Mediate_Objetives(objetive varchar(65), id_project int,
FOREIGN KEY(id_project) REFERENCES Project(id_project) ON DELETE
CASCADE);
CREATE TABLE Project_Methodologies(methodology varchar(65), id_project int,
FOREIGN KEY(id_project) REFERENCES Project(id_project) ON DELETE
CASCADE);
CREATE TABLE Project_Resources(resource varchar(65), id_project int, FOREIGN
KEY(id_project) REFERENCES Project(id_project) ON DELETE CASCADE);
CREATE TABLE Project_Responsabilities(responsability varchar(65), id_project int,
FOREIGN KEY(id_project) REFERENCES Project(id_project) ON DELETE
CASCADE);
CREATE TABLE Project_Activities(activity varchar(65), id_project int, FOREIGN
KEY(id_project) REFERENCES Project(id_project) ON DELETE CASCADE);
CREATE TABLE Project_Immediate_Objetives(objetive varchar(65), id_project int,
FOREIGN KEY(id_project) REFERENCES Project(id_project) ON DELETE
CASCADE);
CREATE TABLE Practitioner(id_person int, enrollment varchar(35), id_project int,
id_professor int, PRIMARY KEY(id_person), FOREIGN KEY(id_person) REFERENCES
Person(id_person),FOREIGN KEY(id_project) REFERENCES Project(id_project) ,
FOREIGN KEY(id_professor) REFERENCES Professor(id_person));
CREATE TABLE  Practitioner_Selected_Projects(selected_project int ,id_person int,
FOREIGN KEY(selected_project) REFERENCES Project(id_project), FOREIGN
KEY(id_person) REFERENCES Practitioner(id_person));
CREATE TABLE Activity(id_activity int AUTO_INCREMENT, name varchar(75),
description text, deadline datetime, id_professor int, PRIMARY KEY(id_activity),
FOREIGN KEY(id_professor) REFERENCES Professor(id_person));
CREATE TABLE Delivery(id_delivery int AUTO_INCREMENT, id_activity int ,
id_practitioner int, observation text ,score float, file_path varchar(230), PRIMARY
KEY(id_delivery), FOREIGN KEY(id_activity) REFERENCES Activity(id_activity),
FOREIGN KEY(id_practitioner) REFERENCES Practitioner(id_person) ON DELETE
CASCADE);
CREATE TABLE AccessAccount(id_user INT, email varchar(75), password
varchar(75), password_recovery_code varchar(15), PRIMARY KEY(id_user),
FOREIGN KEY(id_user) REFERENCES Person(id_person) ON DELETE CASCADE);
CREATE TABLE Host(mac_address varchar(75), attempts INT);



CREATE TABLE Administrator(id_person INT, , FOREIGN KEY(id_person)
REFERENCES Person(id_person));
```

```
DELIMITER $$
CREATE PROCEDURE sendAddress(dir varchar(64))
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
DECLARE CONTINUE HANDLER FOR 1329
BEGIN
  set @idMac = NULL;
END;
START TRANSACTION;
SELECT SUM(attempts) FROM Host WHERE mac_address = dir INTO
@numAttempts;
IF @numAttempts > 4 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Attempts limit reached!';
ELSE
SELECT mac_address FROM Host WHERE mac_address = dir LIMIT 1 INTO
@idMac;
IF @idMac IS NOT NULL THEN
        UPDATE Host set attempts = attempts + 1 WHERE mac_address = dir;
ELSE
        INSERT INTO Host(mac_address, attempts) VALUES(dir, 1);
END IF;
END IF;
COMMIT;
END $$
DELIMITER ;


DELIMITER $$
```

```
CREATE PROCEDURE addProfessor(name_p varchar(75), phone_p varchar(15),
email_p varchar(65), id_cs INT, cubic_c INT, staff_number_c varchar(25))
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
START TRANSACTION;
INSERT INTO Person(name, phoneNumber, email, id_course, activity_state)
VALUES(name_p, phone_p, email_p, id_cs, 1);
SELECT LAST_INSERT_ID() INTO @id_p;
INSERT INTO Professor(id_person, cubicle, staff_number) VALUES(@id_p, cubic_c,
staff_number_c);
INSERT INTO AccessAccount(id_user, email, password) VALUES(@id_p, email_p,
MD5(RAND()));
COMMIT;
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE addCoordinator(name_p varchar(75), phone_p varchar(15),
email_p varchar(65), id_cs INT, cubic_c INT, staff_number_c varchar(25))
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
START TRANSACTION;
INSERT INTO Person(name, phoneNumber, email, id_course, activity_state)
VALUES(name_p, phone_p, email_p, id_cs, 1);
SELECT LAST_INSERT_ID() INTO @id_p;
INSERT INTO Coordinator(id_person, cubicle, staff_number) VALUES(@id_p,
cubic_c, staff_number_c);
INSERT INTO AccessAccount(id_user, email, password) VALUES(@id_p, email_p,
MD5(RAND()));
COMMIT;
END $$
DELIMITER ;



DELIMITER $$
CREATE PROCEDURE addPractitioner(name_p varchar(75), phone_p varchar(15),
email_p varchar(65), id_cs INT, enrollment_p varchar(35))
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
START TRANSACTION;
INSERT INTO Person(name, phoneNumber, email, id_course, activity_state)
VALUES(name_p, phone_p, email_p, id_cs, 1);
SELECT LAST_INSERT_ID() INTO @id_p;
INSERT INTO Practitioner(id_person, enrollment, id_project, id_professor)
VALUES(@id_p, enrollment_p, null, null);
INSERT INTO AccessAccount(id_user, email, password) VALUES(@id_p, email_p,
MD5(RAND()));
COMMIT;
END $$
DELIMITER ;
DELIMITER $$
CREATE PROCEDURE assignProject(person INT, project INT, OUT idProject INT)
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
START TRANSACTION;
SELECT COUNT(id_project) INTO @count FROM Practitioner WHERE id_project =
project;
IF @count < 3 THEN
        UPDATE Practitioner SET id_project = project WHERE id_person = person;
        SET idProject = project;
ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Table size limit reached';
END IF;
COMMIT;
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE addDelivery(activity int, practitioner int, filePath_to
varchar(230), OUT last_id INT)
BEGIN
DECLARE name_act INT;
SELECT NOW() INTO @now;
SELECT deadline INTO @deadline_activity FROM Activity WHERE id_activity =
activity;
IF @now < @deadline_activity THEN
        SELECT id_delivery INTO @already_delivered FROM Delivery WHERE
```

```
id_practitioner = practitioner;
ELSE
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = 'Activitys deadline is over';
END IF;

IF @already_delivered IS NULL THEN
	INSERT INTO Delivery(id_activity, id_practitioner, observation, score,
file_path) VALUES(activity, practitioner, null, null, filePath_to);
	SELECT LAST_INSERT_ID() INTO @last_delivery;
	SET last_id = @last_delivery;
ELSE
	UPDATE Delivery set file_path = filePath_to, score = null, observation = null
WHERE id_practitioner = practitioner;
	SET last_id = @already_delivered;
END IF;
END $$
DELIMITER ;



DELIMITER $$
CREATE PROCEDURE selectProject(person INT, project INT)
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
START TRANSACTION;
SELECT COUNT(id_person) INTO @countSelected FROM
Practitioner_Selected_Projects WHERE id_person = person;
IF @countSelected < 3 THEN
	INSERT INTO Practitioner_Selected_Projects(selected_project, id_person)
VALUES(project, person);
ELSE
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Table size limit reached';
END IF;
COMMIT;
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE removeProject(id_remove INT, OUT idRemove INT)
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
START TRANSACTION;
UPDATE Practitioner SET id_project = NULL WHERE id_project = id_remove;
DELETE FROM Project WHERE id_project = id_remove;
SET idRemove = id_remove;
COMMIT;
END $$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE removeMultivaluedAttributesProject(id INT)
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
 SHOW ERRORS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
 DECLARE EXIT HANDLER FOR SQLWARNING
 BEGIN
 SHOW WARNINGS LIMIT 1;
RESIGNAL;
 ROLLBACK;
 END;
START TRANSACTION;
DELETE FROM Project_Activities WHERE id_project = id;
DELETE FROM Project_Responsabilities WHERE id_project = id;
DELETE FROM Project_Mediate_Objetives WHERE id_project = id;
DELETE FROM Project_Methodologies WHERE id_project = id;
DELETE FROM Project_Resources WHERE id_project = id;
DELETE FROM Project_Immediate_Objetives WHERE id_project = id;
COMMIT;
END $$
DELIMITER ;


CREATE TRIGGER director_update_on_insert AFTER INSERT ON Course FOR EACH
ROW UPDATE Person set id_course = (SELECT max(id_course) FROM Course)
WHERE id_person = 1;

CREATE TRIGGER director_update_on_delete BEFORE DELETE ON Course FOR
EACH ROW UPDATE Person set id_course = (SELECT max(id_course) FROM
Course) WHERE id_person = 1;


SET GLOBAL event_scheduler = ON;

DELIMITER //
CREATE EVENT reset_attempts
ON SCHEDULE EVERY 10 MINUTE
ON COMPLETION PRESERVE
DO
BEGIN
UPDATE Host SET attempts = 0;
UPDATE AccessAccount SET password_recovery_code = null;
END //
DELIMITER ;
```