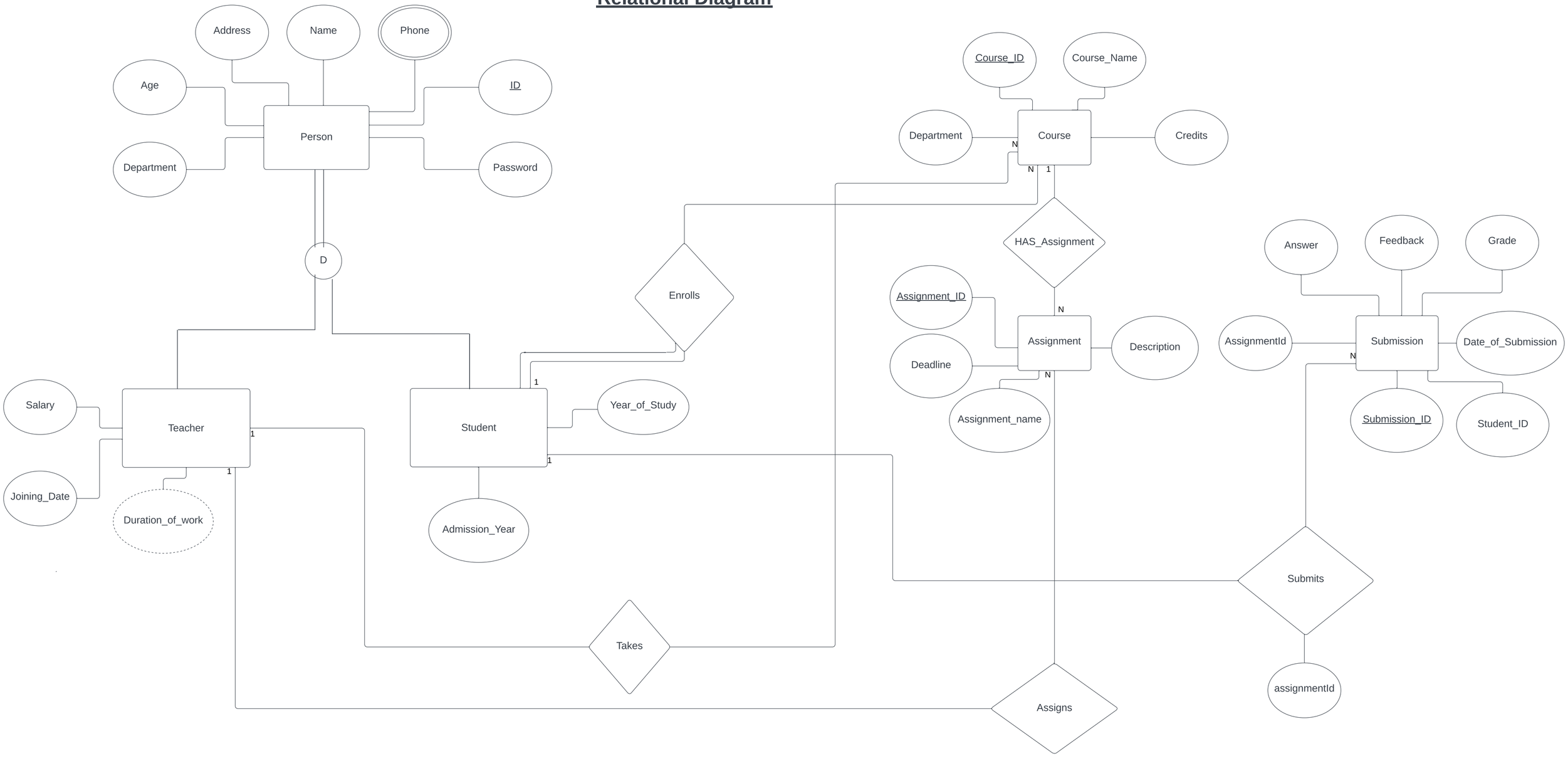
Student Assignment Management System

Relational Diagram



Before Normalisation

Person

name	id	password	department	address	age	Phone
Varchar(50)	Varchar(10) Primary Key	Varchar(50) NOT NULL	enum(CS,ECE,EEE,,ENI, Mech,Civil,Chemical) NOT NULL	Varchar(50) NOT NULL	INT NOT NULL	BigInt Multivalued NOT NULL

Teacher

id	joinDate
Varchar(10) Primary Key Foreign Key - Person(ID)	DATE NOT NULL

Student

id	admissionYear	yearOfStudy
Varchar(10) Primary Key Foreign Key - Person(ID)	DATE NOT NULL	INT NOT NULL CHECK(yearOfStudy>=1 AND yearOfStudy <=5)

HAS_Assignment

assignemntId	courseId
Varchar(10) Primary Key Foreign Key - Assignment(assignmentId)	Varchar(10) Primary Key Foreign Key - Course(courseId)

Course

courseId	courseName	credits	department
Varchar(10) Primary Key	Varchar(20) NOT NULL	INT NOT NULL CHECK(credits>0 and credits <=5)	Varchar(20) NOT NULL

Assignment

assignmentId	description	assignmentName	deadline
Varchar(10) Primary Key	Varchar(50) NOT NULL	Varchar(50) NOT NULL	DATETIME NOT NULL

Enrolls

studentId	courseId
Varchar(10) Primary Key Foreign Key - Student(ID)	Varchar(10) Primary Key

Takes

teacherId	courseId
Varchar(10) Primary Key Foreign Key - Teacher(ID)	Varchar(10) Primary Key

Assigns

assignmentId	teacherId
Varchar(10) Primary Key	Varchar(10) Primary key Foreign Key- Person

Submits

submissionId	assignmentId	studentId
Varchar(10) Primary Key	Varchar(10) Primary Key	Varchar(10) PrimaryKey Foreign Key Student(ID)

Submission

submissionId	assignmentId	dateOfSubmission	answer	Feedback	Grade
Varchar(10) Primary Key	Varchar(10) Primary Key	DATETIME NOT NULL	Varchar(200) NOT NULL	Varchar(50)	enum('A','B','C,'D','E','NC')

First Normal Form

Person

name	id	password	department	address	age
Varchar(50)	Varchar(10) Primary Key	Varchar(50) NOT NULL	enum(CS,ECE,EEE,,ENI, Mech,Civil,Chemical) NOT NULL	Varchar(50) NOT NULL	INT NOT NULL

Phone (Multivalued in Person Table)

id	phone
Varchar(10) Primary Key Foreign Key - Person(ID)	Varchar(10) primary key

Teacher

id	joinDate
Varchar(10) Primary Key Foreign Key - Person(ID)	DATE NOT NULL

Student

id	admissionYear	yearOfStudy
Varchar(10) Primary Key Foreign Key - Person(ID)	DATE NOT NULL	INT NOT NULL CHECK(yearOfStudy>=1 AND yearOfStudy <=5)

HAS_Assignment

assignemntId	courseId
Varchar(10) Primary Key Foreign Key - Assignment(assignmentId)	Varchar(10) Primary Key Foreign Key - Course(courseId)

Course

courseId	courseName	credits	department
Varchar(10) Primary Key	Varchar(20) NOT NULL	INT NOT NULL CHECK(credits>0 and credits <=5)	Varchar(20) NOT NULL

Assignment

assignmentId	description	assignmentName	deadline
Varchar(10) Primary Key	Varchar(50) NOT NULL	Varchar(50) NOT NULL	DATETIME NOT NULL

Enrolls

studentId	courseId
Varchar(10) Primary Key Foreign Key - Student(ID)	Varchar(10) Primary Key

Takes

teacherId	courseId
Varchar(10) Primary Key Foreign Key - Teacher(ID)	Varchar(10) Primary Key

Assigns

assignmentId	teacherId
Varchar(10) Primary Key	Varchar(10) Primary key Foreign Key- Person

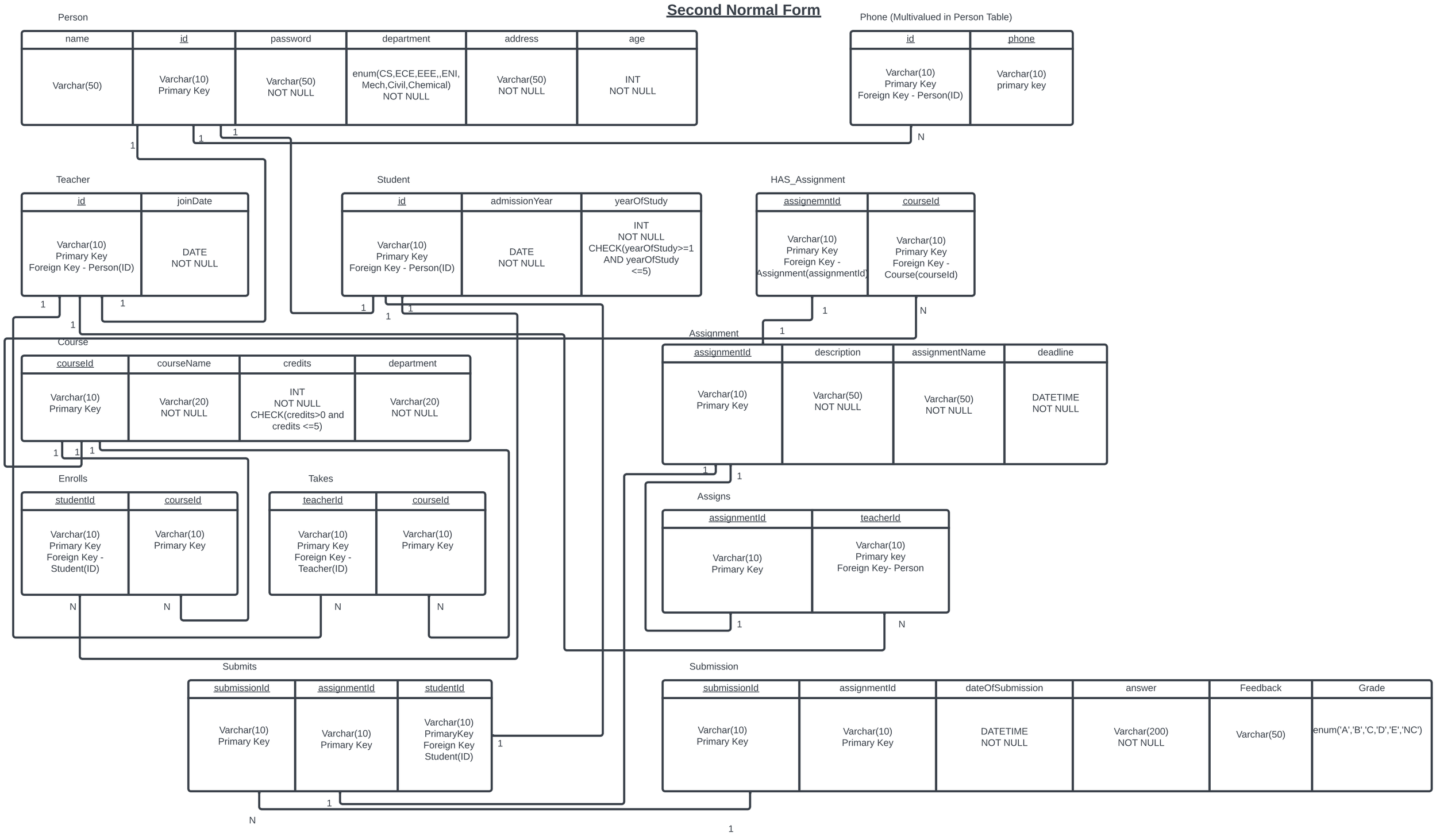
Submits

submissionId	assignmentId	studentId
Varchar(10) Primary Key	Varchar(10) Primary Key	Varchar(10) PrimaryKey Foreign Key Student(ID)

Submission

submissionId	assignmentId	dateOfSubmission	answer	Feedback	Grade
Varchar(10) Primary Key	Varchar(10) Primary Key	DATETIME NOT NULL	Varchar(200) NOT NULL	Varchar(50)	enum('A','B','C,'D','E','NC')

Second Normal Form



Third Normal Form

Person

name	id	password	department	address	age
Varchar(50)	Varchar(10) Primary Key	Varchar(50) NOT NULL	enum(CS,ECE,EEE,,ENI, Mech,Civil,Chemical) NOT NULL	Varchar(50) NOT NULL	INT NOT NULL

Phone (Multivalued in Person Table)

id	phone
Varchar(10) Primary Key Foreign Key - Person(ID)	Varchar(10) primary key

Teacher

id	joinDate
Varchar(10) Primary Key Foreign Key - Person(ID)	DATE NOT NULL

Student

id	admissionYear	yearOfStudy
Varchar(10) Primary Key Foreign Key - Person(ID)	DATE NOT NULL	INT NOT NULL CHECK(yearOfStudy>=1 AND yearOfStudy <=5)

HAS_Assignment

assignemntId	courseId
Varchar(10) Primary Key Foreign Key - Assignment(assignmentId)	Varchar(10) Primary Key Foreign Key - Course(courseId)

Course

courseId	courseName	credits	department
Varchar(10) Primary Key	Varchar(20) NOT NULL	INT NOT NULL CHECK(credits>0 and credits <=5)	Varchar(20) NOT NULL

Assignment

assignmentId	description	assignmentName	deadline
Varchar(10) Primary Key	Varchar(50) NOT NULL	Varchar(50) NOT NULL	DATETIME NOT NULL

Enrolls

studentId	courseId
Varchar(10) Primary Key Foreign Key - Student(ID)	Varchar(10) Primary Key

Takes

teacherId	courseId
Varchar(10) Primary Key Foreign Key - Teacher(ID)	Varchar(10) Primary Key

Assigns

assignmentId	teacherId
Varchar(10) Primary Key	Varchar(10) Primary key Foreign Key- Person

Submits

submissionId	assignmentId	studentId
Varchar(10) Primary Key	Varchar(10) Primary Key	Varchar(10) PrimaryKey Foreign Key Student(ID)

Submission

submissionId	assignmentId	dateOfSubmission	answer	Feedback	Grade
Varchar(10) Primary Key	Varchar(10) Primary Key	DATETIME NOT NULL	Varchar(200) NOT NULL	Varchar(50)	enum('A','B','C,'D','E','NC')

Functional Dependencies of Different Tables

Table: **Person**
ID -> Department, Age, Address, Name, Phone, Password

Table: **Teacher**
ID -> joinDate

Table: **Student**
ID -> admissionYear, yearOfStudy

Table: **Course**
courseId -> courseName, credits, department

Table: **Submission**
submissionId ->assignmentId, dateOfSubmission, answer, feedback, grade

Table: **Assignment**
assignmentId ->assignmentName, description, deadline

Table: **Enrolls**
No Functional Dependences as a combination of all attributes of the table form the Primary Key

Table: **Takes**
No Functional Dependences as a combination of all attributes of the table form the Primary Key

Table: **HasAssignment**
No Functional Dependences as a combination of all attributes of the table form the Primary Key

Table: **Assigns**
No Functional Dependences as a combination of all attributes of the table form the Primary Key

Table: **Submits**
No Functional Dependences as a combination of all attributes of the table form the Primary Key

Table: **Phone**
No Functional Dependences as a combination of all attributes of the table form the Primary Key

Documentation for ER Diagram

The ER Diagram shown is a Diagram which has all the Entities along with their attributes.

The Person-
Teacher - Student Entities have common attributes and the common attributes are stored in the Person Entity with the Teacher and Student Entity being a disjoint specialisation of Person. The Teacher and Student entities have attributes specific to each specialisation and in our implementation of the code, no teacher can be a student and vice versa.
Teacher has attributes such as Joining Date and Salary.
Student has attributes such as Admission Year and Year of Study.

The Course Entity has details specific to a course - Course ID, Course Name, credits and Department.

The Assignment Entity has details such as Assignment ID, Assignment Name, Description and Deadline.

Similarly, Submission has Submission ID, Assignment ID, Student ID to figure out which student is submitting the assignment, Date of Submission and Answer.

The Takes relation helps us understand the teacher who is taking a particular course.
The Enrolls relation helps us understand the student who has enrolled in a particular course.
A Teacher can assign an assignment to the students enrolled in a particular course and the Assigns Relation keeps track of this.
Each course will have assignments which are recorded in the hasAssignment Relation.
The Submits Relation is the link between a Student and his/her submission for a particular Assignment.

Conversion from ER Diagram to Relational Model.

All Entities become a Table in the Relational Model and the attributes in the ER Diagram will be the fields of the Table in the Relational Model.
The Field which can uniquely identify the tuple will become the Primary Key and the weak entities will have a primary key which is formed by the combination of a field from that table and the primary key from the parent table.
Foreign Key References are to be specified in the Relational Model.

Normalization

Once the Relational Model is created, the tables need to be normalized to Reduce Redundancies and eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies. Normalization reduces larger tables into smaller ones and links them using relations.

The **First Normal** form states that there **should not have any multivalued attributes** and all fields should have only atomic values. This leads to the creation of the Phone table which has a combination of the ID and Phone as a primary key and gets rid of the multivalued attribute in the Person Table.

The **Second Normal** form needs all the attributes to be **Fully Functionally dependent on the entire primary key**. On observations, this is already followed in the First Normal Form for our Relational Model and hence, it is in Second Normal Form as well.

The **Third Normal** form states that there **should not be any transitive dependencies** should exist and this is also followed in the Second Normal form of our relational model hence, we have the relational model in the Third Normal form as well.

The following documentation is for the procedures available in the sql file. Comments have been added in the sql file as well., it would be better to go through with the respective functions in the sql file.

Procedures:-

1) get_id(OUT id BIGINT)

this procedure is used to generate unique id for identification where it needs to be generated like in **identification of assignment, submission through its IDs**. while teacher and student IDs are assumed to be not generated and given through institution. It basically increments its count each time its called , and starts from 1000000000, which is the required format for IDs in this database. It returns to the id variable. Course id is also determined by institution and shouldn't be generated using this procedure.

Screenshot:-

```
call get_id(@id_variable);
select @id_variable;
```

Result Grid | Filter Rows: | Export:

@id_variable
1000000024

2) registerStudent(IN nam varchar(50), IN id_tmp BIGINT,IN pwd varchar(50),in dept Varchar(50), in adrs varchar(50),in ag integer,in qstn varchar(50), in answr varchar(50),in admn_yr year, in YOS int):-

allows student to get registered through their details:-
name, id, password, department, adress, age, security question, recovery answer, admission year and year of study.

Screenshot:

```
call registerstudent('Charmander',1234567111,'password','CS','BITS PILANI-5TH STREET',3,'WHAT IS YOUR PET\'S NAME','PET','2021',4);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

success
id registered successfully

3) registerTeacher(IN nam varchar(50), IN id_tmp BIGINT,IN pwd varchar(50),in dept Varchar(50), in adrs varchar(50),in ag integer, in qstn varchar(50), in answr varchar(50),in jn_date date)

allows student to get registered through their details:-
name, id, password, department, adress, age, security question, recovery answer, join date
Screenshot:

```
call registerteacher('Charizard',1234567112,'password','CS','BITS PILANI-5TH STREET',43,'WHAT IS YOUR FAVOURITE SHOW\'S NAME','POKEMON','2021-01-01');
select * from teacher where id=1234567112;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	joinDate
1234567112	2021-01-01

4) setpassword(in uname bigint,in newpass varchar(50), in oldpass varchar(50))
checks if username and old password matches, then updates to newpassword
Screenshot:

```
call setpassword(1234567880,'123#','password');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

success
password changed successfully

5) forgotpassword(in uname bigint,in newpass varchar(50), in answr varchar(50))
changes password with recovery question\'s answer and changes if it matches with username

Screenshot:

```
call forgotpassword(1234567880,'newpass','PET');

select * from person where id=1234567880;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

name	id	password	department	address	age	question	answer_recovery
Charmander	1234567880	newpass	CS	BITS PILANI-5TH STREET	3	WHAT IS YOUR PET'S NAME	PET
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6) login(IN tmp_id bigint, In pwd varchar(50))

checks for loginid and matching password. then if true, populates @id_logged and @pwd_logged variables with the logged in id and password for future authentication.

Screenshot:

```
call logout();

call login(1234567880,'newpass');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

success
logged in successfully

7) logout()

when called, makes the **@id_logged** and **@pwd_logged** as **NULL**

Screenshot:

```
call logout();
```

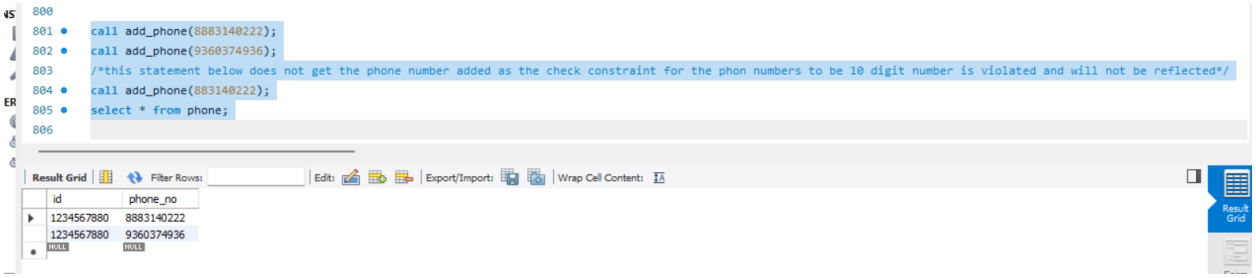
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

success
logged out successfully

8) add_phone(in p_no bigint)

checks if the user has already provided the phone number and then inserts it if they haven't. **The user must be logged in to use this function.**

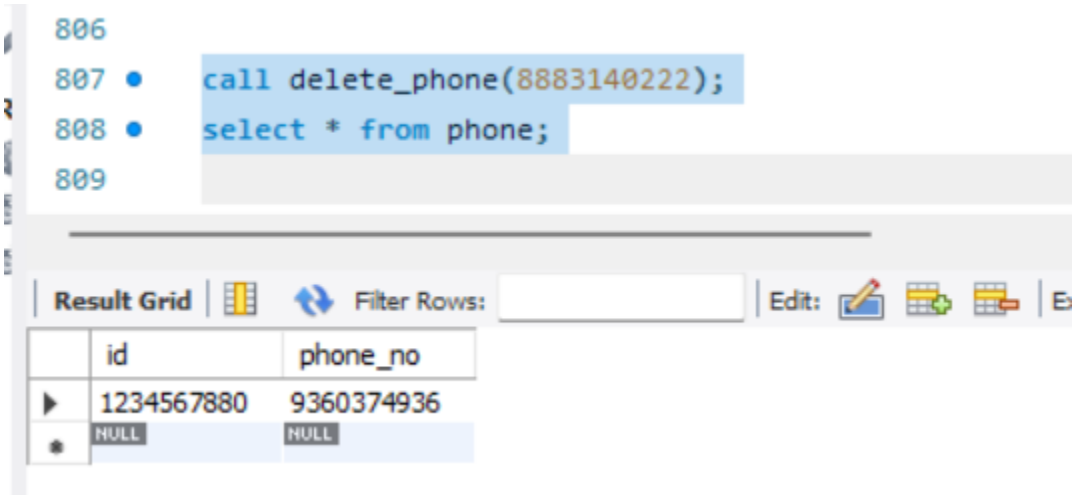
Screenshot:



9) delete_phone(in p_no bigint)

deletes the phone number if it is present under their id. **The user must be logged in to use this function.**

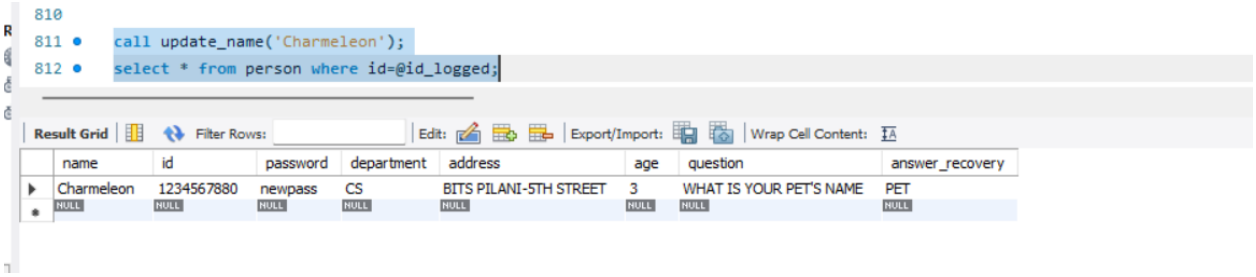
Screenshot:



10) update_name(in tmp_name varchar(50))

check if username and password matches from the logged in variables and then changes name accordingly. **The user must be logged in to use this function**.

Screenshot:



11) AddAssigns(IN aid BIGINT, IN tid bigint)

lets the teacher assign any other teacher to view and grade the assignments,
it first **checks if the user logged in is a teacher** and then checks if the assignment has been **assigned by the current teacher**,if it satisfies it adds in the new teacher for the assignment. **The user must be logged in to use this function**.

Screenshot:

```
call AddAssigns(1000000002,1234567892);
call AddAssigns(1000000002,1234567894);
select * from assigns where assignmentid=1000000002;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

assignmentId	teacherId
1000000002	1234567890
1000000002	1234567892
1000000002	1234567894
NULL	NULL

12) ViewAssignment()

checks for the id and password matching from the logged in variables, then **shows the assignments assigned by the teacher** from the table **assignment natural join assigns** which has each row with teacher id and assignment id. The user must be logged in to use this function.

Screenshot:

```
call ViewAssignment();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentId	assignmentName	description	Deadline
1000000006	Backend app	Do an backend app	2023-09-05 19:00:00
1000000007	Backend app-2	Do an backend app-2	2023-09-05 19:00:00
1000000002	Sql app	Do an sql app	2023-09-05 09:00:00
1000000003	Sql app-2	Do an sql app	2023-09-05 09:00:00
1000000004	Sql app-3	Do an sql app	2023-09-05 09:00:00
1000000005	Sql app-4	Do an sql app	2023-09-05 09:00:00

13) AddAssignment(IN assignmentName varchar(50), IN description varchar(100), IN Deadline DATETIME, IN cid BIGINT)

lets the teacher add an assignment through **assignment name, description, deadline and courseID**. The teacher must be logged in to use this feature. Also , the unique id from assignment is automatically generated and used, can viewed via viewAssignment().

Screenshot:

```
call AddAssignment('Create the ultimate app','It is a challenge','2023-09-01 19:00',1234567890);
call AddAssignment('Create the ultimate app-2','It is a challenge','2023-09-01 19:00',1234567890);
call AddAssignment('Create the ultimate app-3','It is a challenge','2023-09-01 19:00',1234567890);
call Addassignment('Create the ultimate app-4','It is a challenge','2023-09-01 19:00',1234567890);
call AddAssignment('Some assignment','some desc','2023-09-01 23:59',1234567892);
call ViewAssignment();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentId	assignmentName	description	Deadline
1000000012	Some assignment	some desc	2023-09-01 23:59:00
1000000008	Create the ultimate app	It is a challenge	2023-09-01 19:00:00
1000000009	Create the ultimate app-2	It is a challenge	2023-09-01 19:00:00
1000000010	Create the ultimate app-3	It is a challenge	2023-09-01 19:00:00
1000000011	Create the ultimate app-4	It is a challenge	2023-09-01 19:00:00

14) deleteAssignment(IN assgn_id bigint)

lets the teacher delete an assignment **and all its submission by assignment ID**. The teacher must be logged in to use this feature.

Screenshot:

```
call deleteassignment(1000000011);
call ViewAssignment();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentId	assignmentName	description	Deadline
1000000012	Some assignment	some desc	2023-09-01 23:59:00
1000000008	Create the ultimate app	It is a challenge	2023-09-01 19:00:00
1000000009	Create the ultimate app-2	It is a challenge	2023-09-01 19:00:00
1000000010	Create the ultimate app-3	It is a challenge	2023-09-01 19:00:00
1000000011	Create the ultimate app-4	It is a challenge	2023-09-01 19:00:00

15) Enroll(IN c_id BIGINT)

lets the student enroll in a specific course via course id. The student must be logged in to use this feature.

Screenshot:

```
call login(1234567881,'password');
call Enroll(1234567890);
call enroll(1234567812);
call enroll(1234167890);
select * from enrolls where studentid=@id_logged;
```

studentId	courseId
1234567881	1234167890
1234567881	1234567812
1234567881	1234567890
NULL	NULL

16) TeacherTakes(IN courseId BIGINT)

lets the teacher take a particular course via course id. **Checks if the logged variables are matching and are in teacher table**, and the courseid is in course table and then inserts the values into takes table matching the teacher and the course they take.

Screenshot:

```
call login(1234567894,'password');
call TeacherTakes(1234567891);
call TeacherTakes(1234567892);
call TeacherTakes(1234167890);
call TeacherTakes(1234567812);
select * from takes where teacherid=1234567894;

call logout();
```

teacherId	courseId
1234567894	1234167890
1234567894	1234567812
1234567894	1234567891
1234567894	1234567892
NULL	NULL

17) addcollab(IN sub_id BIGINT, IN sid BIGINT)

lets the student **add in other student collaborators on the assignment via submission id and student id of the other collaborator**. The student must be logged in and should be a collaborator of the submission in the first place.

```
call addcollab(1000000016,1234567880);
select * from submits where submissionid=1000000016;

call assgn_searchbyid(1000000002);
call assgn_searchbyname('Backend app');
call assgn_searchbydeadline('2023-09-05 19:00:00');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

submissionId	studentId	assignmentId
1000000016	1234567872	1000000002
1000000016	1234567880	1000000002
NULL	NULL	NULL

18) MakeSubmission(IN asgn_id BIGINT, IN answer varchar(50))

lets the student **make a submission via assignment id and answer** for the submission. The student **must be logged in and should be a enrolled in the course** in which the assignment is there. **Submission id is automatically generated via get_id procedure**

```
call viewsubmission();
call MakeSubmission(1000000002,'ans.pdf');
call viewsubmission();
select * from course natural join student;
select * from submits;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentId	submissionId	dateOfSubmission	answer	grade	feedback	studentId
1000000002	1000000015	2023-04-11 00:53:22	ans.pdf	A	NULL	1234567811

19) viewongoing()

lets the student view ongoing assignments or assignment with future deadline. The user must be logged in. returns the particular assignments **where deadline is greater than now()** after validating student or teacher. Also it **returns only the assignment where the student or teacher has been referenced in submits or assigns table respectively.**

Screenshot:

```
call viewongoing();

select * from course natural join student;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentName	description	deadline	courseid	coursename	department
Sql app	Do an sql app	2023-09-05 09:00:00	1234567890	DBMS	CS
Sql app-2	Do an sql app	2023-09-05 09:00:00	1234567890	DBMS	CS
Sql app-3	Do an sql app	2023-09-05 09:00:00	1234567890	DBMS	CS
Sql app-4	Do an sql app	2023-09-05 09:00:00	1234567890	DBMS	CS
Backend app	Do an backend app	2023-09-05 19:00:00	1234567890	DBMS	CS
Backend app-2	Do an backend app-2	2023-09-05 19:00:00	1234567890	DBMS	CS
Create the ultimate app	It is a challenge	2023-09-01 19:00:00	1234567890	DBMS	CS
Create the ultimate app-2	It is a challenge	2023-09-01 19:00:00	1234567890	DBMS	CS

20) viewfinished()

lets the student or teacher **view finished assignments or assignments past deadline**. The user must be logged in. returns the particular assignments **where deadline is lesser than now()** after validating student or teacher. Also it returns only the assignment where the student or teacher has been referenced in submits or assigns table respectively

Screenshot:

```
call viewfinished();

select * from course natural join student;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentName	description	deadline	courseid	coursename	department
Past assignment	No one can submit	2022-12-12 09:00:00	1234567890	DBMS	CS

21) assgn_searchbyid(in tmp_id bigint)

lets the student or teacher search assignment by id. It returns the assignments which are assigned to or by a student or teacher respectively. otherwise it doesnt return. The user must be logged in.

```
call assgn_searchbyid(1000000002);
call assgn_searchbyname('Backend app');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentName	description	deadline	courseid	coursename	department
Sql app	Do an sql app	2023-09-05 09:00:00	1234567890	DBMS	CS

22) assgn_searchbyname(in tmp_name varchar(50))

lets the student or teacher **search assignment by name**. It returns the assignments which are assigned to or by a student or teacher respectively. otherwise it doesnt return. The user must be logged in.

Screenshot:

```
call assgn_searchbyid(1000000002);
call assgn_searchbyname('Backend app');

call assgn_searchbydeadline('2023-09-05 19:00:00');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentName	description	deadline	courseid	coursename	department
Backend app	Do an backend app	2023-09-05 19:00:00	1234567890	DBMS	CS

23) assgn_searchbydeadline(in tmp_date datetime)

lets the student or teacher **search assignment by deadline**. It returns the assignments which are assigned to or by a student or teacher respectively. otherwise it doesnt return. The user must be logged in.

Screenshot:

```
call assgn_searchbydeadline('2023-09-05 19:00:00');

call updateanswer('App.sql',1000000016);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentName	description	deadline	courseid	coursename	department
Backend app	Do an backend app	2023-09-05 19:00:00	1234567890	DBMS	CS
Backend app-2	Do an backend app-2	2023-09-05 19:00:00	1234567890	DBMS	CS

24) viewsubmission()

lets the student or teacher view their submission or submissions of their assignment respectively. **returns the assignments relevant to them by matching the studentid or teacherid**. The user must be logged in.

Screenshot:

```
call viewsubmission();

call logout();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentId	submissionId	dateOfSubmission	answer	grade	feedback	studentId
1000000002	1000000016	2023-04-11 00:53:22	App.sql	B	Subpar	1234567872
1000000003	1000000017	2023-04-11 00:53:22	answer.pdf	NULL	NULL	1234567872
1000000005	1000000019	2023-04-11 00:53:22	answer.pdf	NULL	NULL	1234567872
1000000006	1000000020	2023-04-11 00:53:22	this_deserves_b.pdf	NULL	NULL	1234567872
1000000007	1000000021	2023-04-11 00:53:22	answer.pdf	NULL	NULL	1234567872
1000000008	1000000022	2023-04-11 00:53:22	answer.pdf	NULL	NULL	1234567872
1000000009	1000000023	2023-04-11 00:53:22	answer.pdf	NULL	NULL	1234567872

25) assgn_searchby_submitted_percent(in start_perc float, in end_perc float)

lets the teacher search their assignments by the percentage of submitted students via **start percentage (start_perc)** and **end percentage (end_perc)**. returns the values between the start and end percentages only **with their assignmentid and assignmentname**. Also this searches in **only the assignments asssigned by the teacher. The teacher must be logged in**.

Screenshot:

```
call GiveFeedbackGrade(1000000015,NULL,'A');
call assgn_searchby_submitted_percent(10,20);
call viewsubmission();
call viewongoing();
/* adding manually into assignment table*/
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

percent	a_id	assignmentname	description	deadline
20.0000	1000000003	Sql app-2	Do an sql app	2023-09-05 09:00:00
20.0000	1000000005	Sql app-4	Do an sql app	2023-09-05 09:00:00
20.0000	1000000006	Backend app	Do an backend app	2023-09-05 19:00:00
20.0000	1000000007	Backend app-2	Do an backend app-2	2023-09-05 19:00:00

26) GetAssignmentCompletionRate(IN aid BIGINT)

returns the assignment completion rate for a particular assignment for a teacher. it returns even if the assignment is not assigned by a particular teacher. The teacher must be logged in.

Screenshot:

```
call login(1234567890,'password');
call GetAssignmentCompletionRate(1000000002);
call viewsubmission();
call GiveFeedbackGrade(1000000015,NULL,'A');
call assgn_searchby_submitted_percent(10,20);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

completition_rate
60.0000

27) GetAssignmentGradeStat(IN aid BIGINT)

returns the **Grade statistics for a particular assignment** for a teacher **via assignment ID**.it returns even if the assignment is not assigned by a particular teacher. Teacher must be logged in.

```
call viewsubmission();
call GetAssignmentGradeStat(1000000002);
call GetAssignmentCompletionRate(1000000002);
call logout();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

grade	count
A	1
B	2

28) GiveFeedbackGrade(IN sId BIGINT,IN fb varchar(100), IN tmp_grade varchar(2))
lets the **teacher give feedback(fb) and grade(tmp_grade)** on a submission via **submission id (sid)**. inserts into the submission table in the grade and feedback attributes for their assignments assigned by the logged in teacher only.
Screenshot:

```
call viewsubmission();
call GiveFeedbackGrade(1000000015,NULL,'A');
select * from submission where submissionid=1000000015;
call assgn_searchby_submitted_percent(10,20);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

assignmentId	submissionId	dateOfSubmission	answer	grade	feedback
1000000002	1000000015	2023-04-11 00:53:22	ans.pdf	A	NULL
NULL	NULL	NULL	NULL	NULL	NULL

29) updatedeadline(IN tmp_deadline datetime,IN aId BIGINT)
lets the teacher **update deadline(tmp_deadline) on an assignment via assignment id (aid)**. It updates **only if the logged teacher has assigned the assignment**.
Screenshot:

```
call GetAssignmentCompletionRate(1000000002);
call updatedeadline('2023-09-11 00:00:00',1000000002);
select * from assignment where deadline='2023-09-11 00:00:00';
call logout();
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

assignmentId	assignmentName	description	Deadline
1000000002	Sql app	Do an sql app	2023-09-11 00:00:00
NULL	NULL	NULL	NULL

30) deletesubmission(IN sub_id BIGINT)
lets the **student delete a submission via submission id(sub_id)**. The student must be logged in and should be one of the collaborators on the submission.
Screenshot:

```
call viewsubmission();
call deletesubmission(1000000024);
call viewsubmission();
call deletesubmission(1000000013);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentId	submissionId	dateOfSubmission	answer	grade	feedback	studentId
1000000002	1000000015	2023-04-11 00:53:22	ans.pdf	A	NULL	1234567811

31) updateanswer(IN answer_tmp varchar(50), in sub_id bigint)

lets the student **update answer(answer_tmp)** on their submission via **submission id (sub_id)**. validates the student logged in credentials and **check if the matching submission id and student id record exists**, then update the answer atribute from submission if deadline is greater than now().

Screenshot:

```
call assgn_searchbydeadline('2023-09-05 19:00:00');
call updateanswer('App.sql',1000000015);
call viewsubmission();
call deletesubmission(1000000018);
call viewsubmission();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

assignmentId	submissionId	dateOfSubmission	answer	grade	feedback	studentId
1000000002	1000000015	2023-04-11 00:53:22	App.sql	A	NULL	1234567811

32) forgotpass_view(in tmp_id bigint)

lets to view security question when password is forgotten through user id(tmp_id).

Screenshot:

```
call setpassword(1234567880,'123#','password');

call forgotpass_view(1234567880);

call forgotpassword(1234567880,'newpass','PET');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

question
WHAT IS YOUR PET'S NAME

33) TRIGGER Check_previous_submits after INSERT ON submits

this trigger doesn't allow to submits if student is making multiple submissions for an assignment. This trigger makes the insertion into submits table failed which can make the procedure makesubmission to fail which has a transaction that does not commit if error occurs