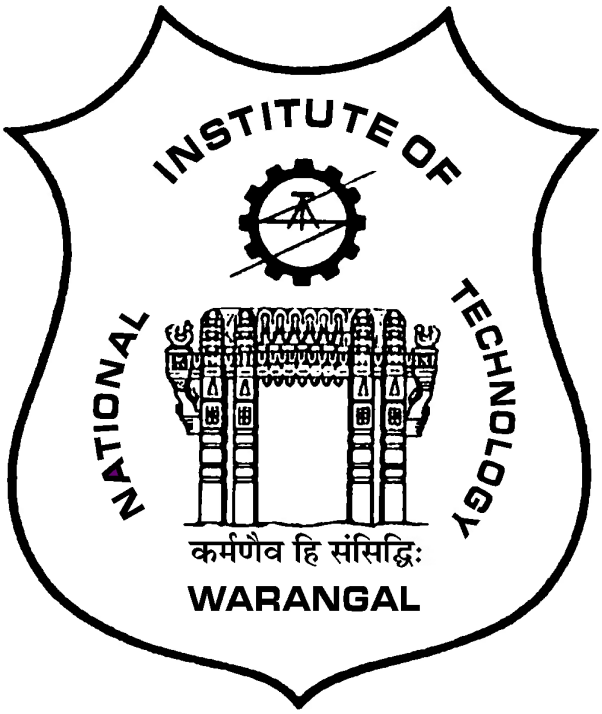
National Institute of Technology
Warangal



Department of Computer Science and Engineering
B.Tech - CSE

TRAINING AND PLACEMENT
DATABASE MANAGEMENT SYSTEM

SAGAR CHANDRA RAKHSHIT (21CSB0A72)

FACULTY:
DR.RAMAKRISHNUDU
Prof. CSE Dept.

DBMS Project Problem Statement

A training and placement database management system is a software system designed to manage data related to the training and placement of students or job seekers. The system should be able to store and retrieve information about job openings, candidates, employers, training programs, and placements. The database should be able to handle large amounts of data, and the system should have robust security measures in place to ensure that the information is protected.

The main problem statement for a training and placement database management system is the need to efficiently manage and track the placement and training processes. The system should be able to:

1.Streamline the placement process by providing a platform for employers to post job openings, and job seekers to apply for those openings.

2.Manage and track the progress of each candidate through the various stages of the recruitment process, from application to interview to offer.

3.Provide data analysis and reporting to help identify areas where improvements can be made to the recruitment process, and to monitor the success of training programs.

4.Ensure the security and privacy of the data, with appropriate access controls and encryption measures in place.

5.Provide an easy-to-use interface for both employers and job seekers, with intuitive navigation and clear instructions.

6.Allow for customisation to meet the specific needs of different organisations, including the ability to add or remove fields, create custom reports, and integrate with other systems.

Overall, the goal of a training and placement database management system is to streamline the recruitment and training process, and to provide actionable insights to improve outcomes for both employers and job seekers.

ER MODEL ASSUMPTION:

Here are some of the assumptions for a training and placement database management system:

Entities: The main entities in a training and placement database management system are likely to be job seekers, employers, job openings, training programs, and placements. Each of these entities can have different attributes, such as job seeker name, job opening title, employer name, training program duration, and placement date.

Attributes: Attributes are characteristics or properties of an entity. In a training and placement database management system, attributes can include things like job seeker skills, job opening location, employer industry, training program type, and placement salary.

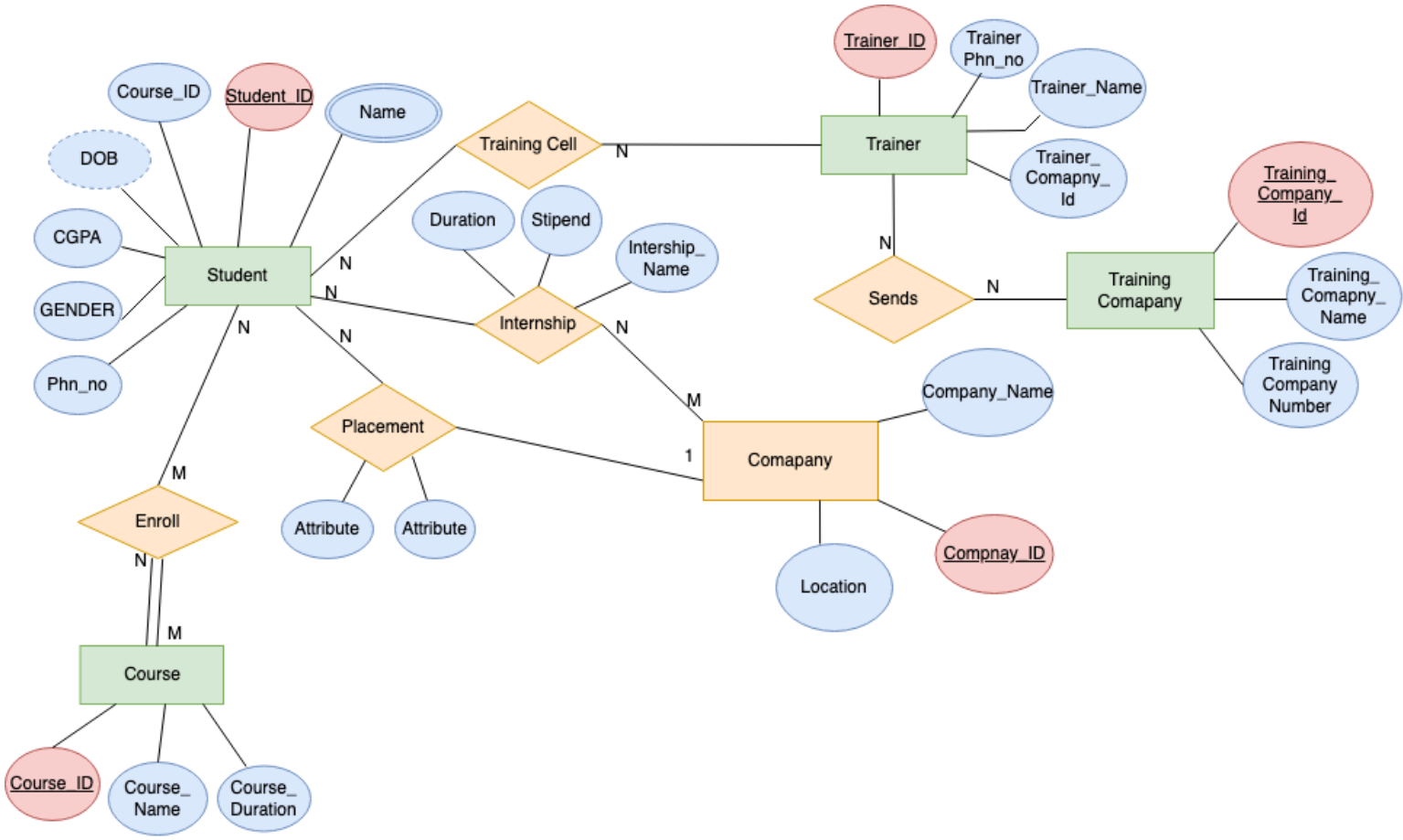
Relationships: Relationships represent the connections between entities. In a training and placement database management system, relationships can include things like a job seeker applying for a job opening, an employer posting a job opening, a job seeker attending a training program, and a job seeker being placed in a job by an employer.

Cardinality: Cardinality refers to the number of occurrences of an entity that can be related to the occurrences of another entity. In a training and placement database management system, the cardinality can vary between entities. For example, a job seeker can apply for many job openings, but a job opening can only have one employer.

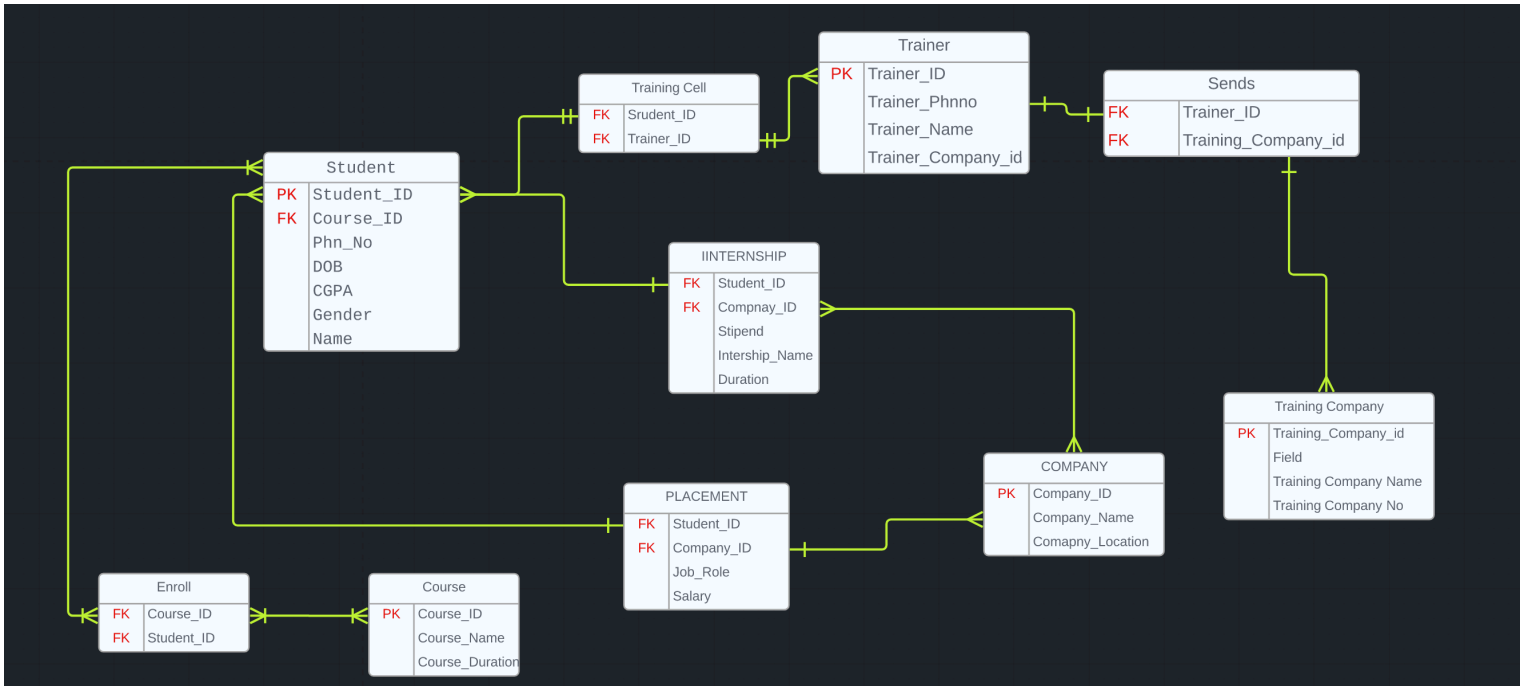
Primary keys: Primary keys are unique identifiers for entities. In a training and placement database management system, primary keys can be used to uniquely identify job seekers, employers, job openings, training programs, and placements.

Foreign keys: Foreign keys are attributes in an entity that refer to the primary key of another entity. In a training and placement database management system, foreign keys can be used to establish relationships between entities, such as a job seeker applying for a job opening or attending a training program.

ER MODEL:


Trainer_ID
Trainer Phn_no
Trainer_Name
Trainer_ Comapny_ Id
Trainer
Training Cell
N
Course_ID
Student_ID
Name
DOB
CGPA
GENDER
Phn_no
Student
N
N
N
N
Duration
Stipend
Intership_ Name
Internship
N
M
N
Sends
N
Training_ Company_ Id
Training Comapany
Training_ Comapny_ Name
Training Company Number
Company_Name
Placement
1
Comapany
Compnay_ID
Location
Attribute
Attribute
M
Enroll
N
M
Course
Course_ID
Course_ Name
Course_ Duration

RELATIONAL MODEL:



RELATIONAL SCHEMA:

Student

Attribute	Data Type	Contraints
Student_ID	Int	Primary Key
Name	Varchar(25)	Not Null
Phn_No	Varchar(25)	Not Null
DOB	Date	Not Null
CGPA	Decimal	Not Null
Course_ID	Int	Foreign Key

Trainer

Attribute	Data Type	Contraints
Trainer_ID	Varchar	PK
Trianer_Phnno	decimal	Not null
Trainer_Name	varchar(25)	Not null

Training Company

Attribute	Data Type	Contraints
Training_Comapny_ID	Varchar	PK
Training_Comapn_Name	varchar(25)	Not null
Training_Comapany No	varchar(10)	Not null

Company

Attribute	Data Type	Contraints
Company_ID	Int	PK
Company_Name	varchar(15)	Not null
Company_Location	varchar(15)	

INTERNSHIP

Attribute	Data Type	Contraints
STIPEND	DECIMAL	
DURATION	INT	
INTERNSHIP_NAME	varchar(15)	
STUDENT_ID,COMAPNT_ID	INT	PRIMARY KEY

SQL QUERIES(RELATIONAL SCHEMAS)

STUDENT TABLE:

```
CREATE TABLE STUDENT(
    STUDENT_ID int PRIMARY KEY,
    COURSE_ID INT,
    NAME VARCHAR(20),
    DOB DATE,
    CGPA DECIMAL(10,2),
    GENDER VARCHAR(5),
    PHN_NO DECIMAL(10)
);
```

INSERTION:

```
insert into student
values(1,101,'SAGAR',"2020-01-01",9.82,"MALE",8193193010),
(2,102,'SOHAM',"2003-08-08",7.86,"MALE",7193193010),
(3,110,'ANURAG',"2007-07-01",8.95,"MALE",6441193010),
(4,109,'SAM',"2006-01-01",5.65,"MALE",9193958010),
(5,201,'SHASHANK',"2004-09-01",7.83,"MALE",7003567212);
```

STUDENT

STUDENT_ID	COURSE_ID	NAME	DOB	CGPA	GENDER	PHN_NO
1	101	SAGAR	2020-01-01	9.82	MALE	8193193010
2	102	SOHAM	2003-08-08	7.86	MALE	7193193010
3	110	ANURAG	2007-07-01	8.95	MALE	6441193010
4	109	SAM	2006-01-01	5.65	MALE	9193958010
5	201	SHASHANK	2004-09-01	7.83	MALE	7003567212

COURSE TABLE

```
CREATE TABLE COURSE(
  COURSE_ID INT PRIMARY KEY,
  COURSE_NAME VARCHAR(10),
  COURSE_DURATION INT,
  STUDENT_ID INT,
  FOREIGN KEY(STUDENT_ID) REFERENCES
STUDENT(STUDENT_ID)
  );
```

INSERTION:

```
  INSERT INTO COURSE
  VALUES(101,"CS101",5,1),
  (109,"BT543",4,4),
  (102,"ECE131",3,2),
  (110,"CS103",4,3),
  (201,"CS890",4,5);
```

COURSE

COURSE_ID	COURSE_NAME	COURSE_DURATION	STUDENT_ID
101	CS101	5	1
109	BT543	4	4
102	ECE131	3	2
110	CS103	4	3
201	CS890	4	5

ENROLL TABLE (MANY-MANY RELATIONSHIP B/W COURSE & STUDENT)

```
CREATE TABLE ENROLL(
  STUDENT_ID INT,
  COURSE_ID INT,
  PRIMARY KEY(STUDENT_ID,COURSE_ID)
  );
```

INSERTION:

```
INSERT INTO ENROLL
VALUES(1,101),
(2,102),
(3,110),
(4,109),
(5,201);
```

ENROLL

STUDENT_ID	COURSE_ID
1	101
2	102
3	110
4	109
5	201

TRAINER TABLE

```
CREATE TABLE TRAINER(
  TRAINER_ID VARCHAR(20),
  TRAINER_NAME  VARCHAR(255),
  TRAINER_PHONE_NO DECIMAL(10),
  PRIMARY KEY(TRAINER_ID),
  STUDENT_ID INT ,
  FOREIGN KEY (STUDENT_ID) REFERENCES STUDENT(STUDENT_ID)
);
```

INSERTION:

```
insert into TRAINER
VALUES('TS101','JOHN',924919412,1),
('CS101','MICHEAL',864919412,2),
('EC101','SHWAN',704919412,3),
('TY401','STEPHAN',784919412,4),
('IC501','TRISTAN',644919412,5);
```

ABC trainer_id	ABC trainer_name	123 trainer_phone_no	123 student_id
TS101	JOHN	924,919,412	1
CS101	MICHEAL	864,919,412	2
EC101	SHWAN	704,919,412	3
TY401	STEPHAN	784,919,412	4
IC501	TRISTAN	644,919,412	5

TRIANING COMAPNY TABLE

```
CREATE TABLE TRAINING_COMPANY(
  TRAINING_COMPANY_ID VARCHAR(20) NOT NULL,
  TRAINING_COMPANY_NAME  VARCHAR(255),
  TRAINING_COMPANY_NO DECIMAL(10),
  PRIMARY KEY(TRAINING_COMPANY_ID),
  TRAINER_ID VARCHAR(20),
  FOREIGN KEY (TRAINER_ID) REFERENCES TRAINER(TRAINER_ID)
  );
```

INSERTION:

```
insert into training_company
VALUES('SAM101','SAMSUNG','7003756921','TS101'),
('AP203','APPLE','8017436511','IC501'),
('OPPO901','OPPO','900356054','TY401'),
('BT301','BOAT','6003756054','CS101'),
('CR501','CORSAIR','8600470037','TY401');
```

training_company_id	training_company_name	training_company_no	trainer_id
SAM101	SAMSUNG	7,003,756,921	TS101
AP203	APPLE	8,017,436,511	IC501
OPPO901	OPPO	900,356,054	TY401
BT301	BOAT	6,003,756,054	CS101
CR501	CORSAIR	8,600,470,037	TY401

COMPANY TABLE

```
create table COMPANY(
COMAPANY_ID INT primary key,
COMPANY_NAME VARCHAR(20),
LOCATIION VARCHAR(50),
STUDENT_ID INT,
foreign KEY(STUDENT_ID) references STUDENT(STUDENT_ID)
);
```

INSERTION:

```
insert into company
VALUES(6001,'ORACLE','HYDERABAD',1),
(76001,'WEBEX','DELHI',3),
(8901,'GOOGLE','DELHI',4),
(34001,'INFOSYS','KOLKATA',5),
(5401,'ASUS','BANGALORE',2);
```

```
select * from COMPANY   Enter a SQL expression to filter results (use Ctrl+Space)
```

	123 comapany_id	ABC company_name	ABC locatiion	123 student_id
1	6,001	ORACLE	HYDERABAD	1
2	76,001	WEBEX	DELHI	3
3	8,901	GOOGLE	DELHI	4
4	34,001	INFOSYS	KOLKATA	5
5	5,401	ASUS	BANGALORE	2

INTERNSHIP(M-M RELATIONSHIP B/W COMPANY AND STUDENT)

```
create table INTERNSHIP(
STUDENT_ID INT,
COMPANY_ID INT,
DURATION INT,
STIPEND DECIMAL(10),
INTERNSHIP_NAME VARCHAR(30),
primary key(STUDENT_ID,COMPANY_ID)
);
```

INSERTION:

```
insert into internship
VALUES(1,6001,6,45000,'FRONTEND'),
(2,34001,3,90000,'SDE'),
(3,76001,3,145000,'GUI'),
(4,8901,6,60000,'FULLSTACK'),
(5,5401,3,140000,'BACKEND');
```

```
elect * from INTERNSHIP  Enter a SQL expression to filter results (use Ctrl+Space)
```

	123 student_id	123 company_id	123 duration	123 stipend	ABC internship_name
1	1	6,001	6	45,000	FRONTEND
2	2	34,001	3	90,000	SDE
3	3	76,001	3	145,000	GUI
4	4	8,901	6	60,000	FULLSTACK
5	5	5,401	3	140,000	BACKEND

NORMALIZATION

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

To normalize the table into 1NF, 2NF, 3NF and BCNF, we need to follow the following steps:

1NF: Make sure that each column of the table contains atomic (indivisible) values.

2NF: Remove partial dependencies by putting them in a separate table.

3NF: Remove transitive dependencies by putting them in a separate table.

BCNF: Remove candidate keys that are not necessary for the functional dependencies.

IN STUDENT TABLE :

```
CREATE TABLE STUDENT(
    STUDENT_ID int PRIMARY KEY,
    COURSE_ID INT,
    NAME VARCHAR(20),
    DOB DATE,
    CGPA DECIMAL(10,2),
    GENDER VARCHAR(5),
    PHN_NO DECIMAL(10)
);
```

1NF:

The given table is already in 1NF as it satisfies the atomicity property of the First Normal Form.

2NF:

To achieve 2NF, we need to identify the partial dependencies in the given table. Here, the functional dependencies are as follows:
Student_ID -> Name, DOB, CGPA, Gender, Phone Number
Course_ID -> No partial dependency

3NF:
To achieve 3NF, we need to identify the transitive dependencies in the table. Here, there is no transitive dependency in the table.

BCNF:
To achieve BCNF, we need to identify the functional dependencies and eliminate any overlapping functional dependencies. In this table, there is no overlapping functional dependency. Therefore, the given table is already in BCNF.

IN COURSE TABLE

```
CREATE TABLE COURSE(
  COURSE_ID INT PRIMARY KEY,
  COURSE_NAME VARCHAR(10),
  COURSE_DURATION INT,
  STUDENT_ID INT,
  FOREIGN KEY(STUDENT_ID) REFERENCES STUDENT(STUDENT_ID)
  );
```

1NF:
The given table is already in 1NF as it satisfies the atomicity property of the First Normal Form.

2NF:
As there is no partial dependency, the table is already in 2NF.

3NF:
To achieve 3NF, we need to identify the transitive dependencies in the table. Here, there is no transitive dependency in the table.

BCNF:

As there is only one functional dependency, COURSE_ID -> COURSE_NAME, COURSE_DURATION, the table is already in BCNF.

IN ENROLL TABLE:

```
CREATE TABLE ENROLL(
  STUDENT_ID INT,
  COURSE_ID INT,
  PRIMARY KEY(STUDENT_ID,COURSE_ID)
```

```
);
```

.

The given table has already been normalized and is already in 1NF, 2NF, 3NF, and BCNF. This is because it has only two columns, both of which form the primary key, and there are no partial dependencies or transitive dependencies. Therefore, no further normalization is required.

IN TRAINER TABLE:

1NF:

TRAINER TABLE:

TRAINER_ID	TRAINER_NAME	TRAINER_PHONE_NO	STUDENT_ID
TS101	JOHN	924919412	1
CS101	MICHEAL	864919412	2
EC101	SHWAN	704919412	3
TY401	STEPHAN	784919412	4
IC501	TRISTAN	644919412	5

2NF:

TRAINER STUDENT TABLE:

TRAINER_STUDENT table:

TRAINER_ID	STUDENT_ID
TS101	1
CS101	2
EC101	3
TY401	4
IC501	5

TRIANER TABLE IN 2NF

TRAINER_ID	TRAINER_NAME	TRAINER_PHONE_NO
TS101	JOHN	924919412
CS101	MICHEAL	864919412
EC101	SHWAN	704919412
TY401	STEPHAN	784919412
IC501	TRISTAN	644919412

3NF:

IN 3NF THIS TABLES ARE CREATED (AS THERE IS TRANSITIVE DEPENDENCIES)

TRAINER_ID
TS101
CS101
EC101
TY401
IC501

TRAINER_ID	TRAINER_PHONE_NO
TS101	924919412
CS101	864919412
EC101	704919412
TY401	784919412
IC501	644919412

TRAINER_ID	TRAINER_NAME	TRAINER_PHONE_NO
TS101	JOHN	924919412
CS101	MICHEAL	864919412
EC101	SHWAN	704919412
TY401	STEPHAN	784919412
IC501	TRISTAN	644919412

BCNF:

As there is only one functional dependency, TRAINER_ID -> TRAINER_NAME, TRAINER_PHONE_NO, STUDENT_ID, the table is already in BCNF.

IN TRAINING COMPANY

```
CREATE TABLE TRAINING_COMPANY(
  TRAINING_COMPANY_ID VARCHAR(20) NOT NULL,
  TRAINING_COMPANY_NAME  VARCHAR(255),
  TRAINING_COMPANY_NO DECIMAL(10),
  PRIMARY KEY(TRAINING_COMPANY_ID),
  TRAINER_ID VARCHAR(20),
  FOREIGN KEY (TRAINER_ID) REFERENCES TRAINER(TRAINER_ID)
  );
```

1NF:
TRAINING_COMPANY(
TRAINING_COMPANY_ID,
TRAINING_COMPANY_NAME,
TRAINING_COMPANY_NO,
TRAINER_ID
)

2NF:
TRAINING_COMPANY(
TRAINING_COMPANY_ID,
TRAINING_COMPANY_NAME,
TRAINING_COMPANY_NO
)

TRAINER_TRAINING_COMPANY(
TRAINER_ID,
TRAINING_COMPANY_ID
)

3NF:
TRAINING_COMPANY(
TRAINING_COMPANY_ID,
TRAINING_COMPANY_NAME,
TRAINING_COMPANY_NO
)

TRAINER(
TRAINER_ID,
TRAINER_NAME,
TRAINER_PHONE_NO,
STUDENT_ID
)

TRAINER_TRAINING_COMPANY(
TRAINER_ID,
TRAINING_COMPANY_ID
)

BCNF:
TRAINING_COMPANY(
TRAINING_COMPANY_ID,
TRAINING_COMPANY_NAME,
TRAINING_COMPANY_NO
)

TRAINER(
TRAINER_ID,
TRAINER_NAME,
TRAINER_PHONE_NO
)

TRAINER_TRAINING_COMPANY(
TRAINER_ID,
TRAINING_COMPANY_ID
)

IN COMPANY TABLE:

```
create table COMPANY(
COMAPANY_ID INT primary key,
COMPANY_NAME VARCHAR(20),
LOCATIION VARCHAR(50),
STUDENT_ID INT,
foreign KEY(STUDENT_ID) references STUDENT(STUDENT_ID)
);
```

1NF:
CREATE TABLE COMPANY(

```
COMPANY_ID INT PRIMARY KEY,
COMPANY_NAME VARCHAR(20),
LOCATION VARCHAR(50),
STUDENT_ID INT,
FOREIGN KEY(STUDENT_ID) REFERENCES STUDENT(STUDENT_ID)
);

INSERT INTO COMPANY
VALUES(6001,'ORACLE','HYDERABAD',1),
(76001,'WEBEX','DELHI',3),
(8901,'GOOGLE','DELHI',4),
(34001,'INFOSYS','KOLKATA',5),
(5401,'ASUS','BANGALORE',2);
```

2NF:
There are no partial dependencies, so the table is already in 2NF.

3NF:
There are no transitive dependencies, so the table is already in 3NF.

BCNF:
The table is already in BCNF, because there is only one candidate key (COMPANY_ID) and there are no non-trivial functional dependencies.

IN INTERNSHIP TABLE:

```
create table INTERNSHIP(
STUDENT_ID INT,
COMPANY_ID INT,
DURATION INT,
STIPEND DECIMAL(10),
INTERNSHIP_NAME VARCHAR(30),
primary key(STUDENT_ID,COMPANY_ID)
);
```

1NF:
INTERNSHIP table is already in 1NF as there are no repeating groups or arrays.

2NF:

The table already has a composite primary key, so we need to check for partial dependencies. There are no partial dependencies as all non-key attributes are fully dependent on the primary key.

3NF:

There are no transitive dependencies in the table.

BCNF:

The table is already in BCNF as there are no non-trivial functional dependencies other than the candidate keys.

Therefore, the INTERNSHIP table is already in 1NF, 2NF, 3NF, and BCNF. No further normalization is required.