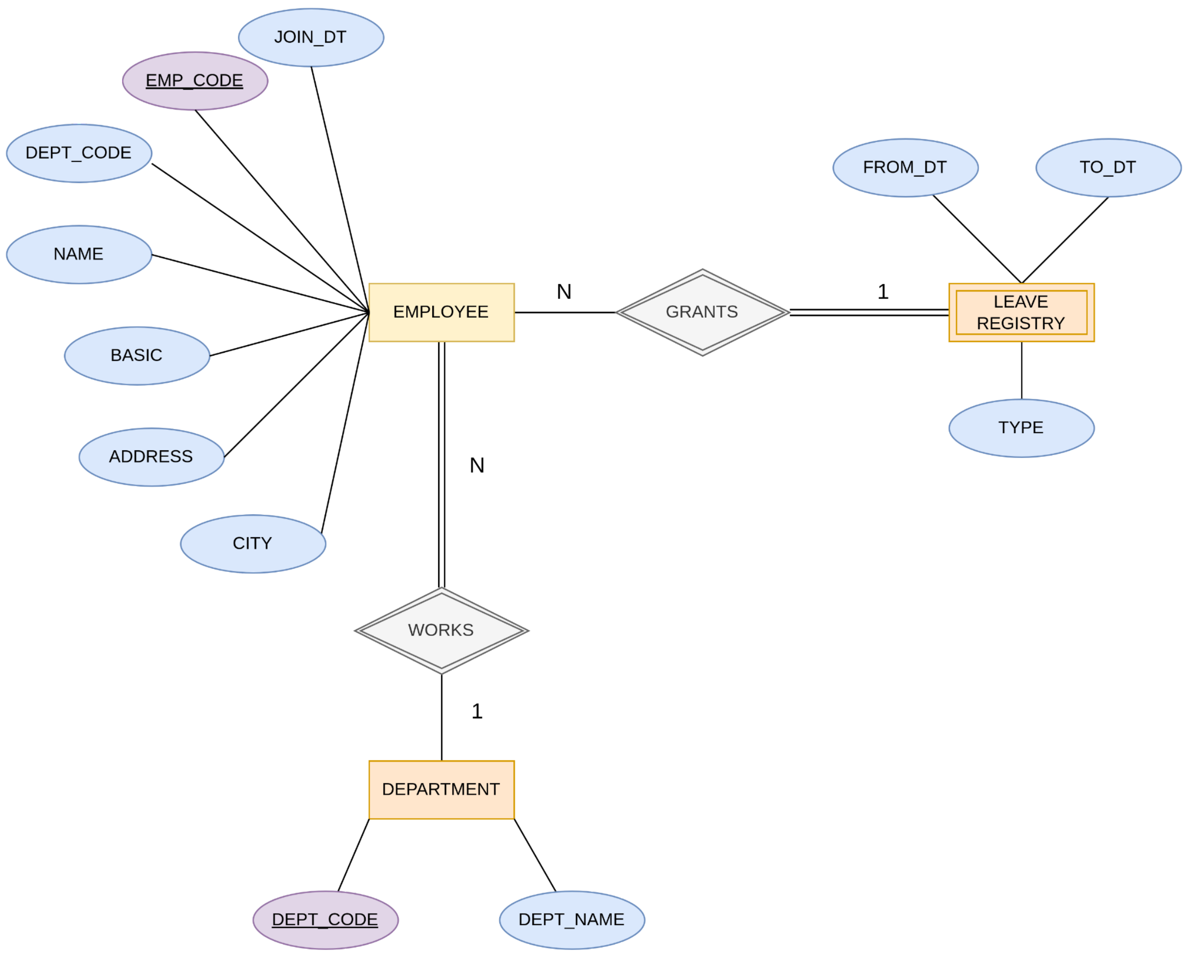
Name : Tanmoy Sarkar
Roll No : 002010501020
Assignment No : 3
—---

1. In an organisation, a number of departments exist. Each department has a name & unique code. Number of employees work in each department. Each employee has a unique employee code. Detailed information like name, address, city, basic, date of join are also stored. In a leave register for each employee leave records are kept showing leave type (CL/EL/ML etc.), from-date, to-date. When an employee retires or resigns then all the leave information pertaining to him are also deleted. Basic salary must be within Rs.5000 to Rs.9000. A department can not be deleted if any employee record refers to it. Valid grades are A/B/C. Employee name must be in uppercase only. Default value for joining date is system date. Design & implement the tables with necessary constraints to support the scenario depicted above.

ER Diagram



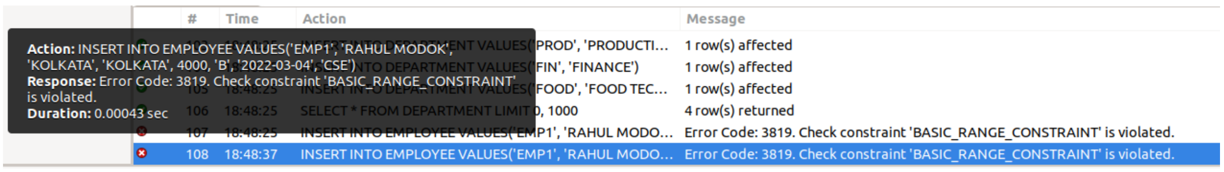
SQL For Create Table

```
CREATE TABLE DEPARTMENT(
  DEPTCODE VARCHAR(10) PRIMARY KEY,
  NAME VARCHAR(15) NOT NULL
);

CREATE TABLE EMPLOYEE(
  NAME VARCHAR(25) NOT NULL ,
  EMPCODE CHAR(5) PRIMARY KEY,
  ADDRESS VARCHAR(50) NOT NULL,
  CITY VARCHAR(20) NOT NULL,
  BASIC INTEGER NOT NULL ,
  GRADE CHAR(1) NOT NULL ,
  JN_DATE DATE DEFAULT(CURRENT_DATE),
  DEPTCODE VARCHAR(10) NOT NULL,
  FOREIGN KEY(DEPTCODE) REFERENCES DEPARTMENT(DEPTCODE) ON DELETE RESTRICT
  CONSTRAINT NAME_UPPERCASE_CONSTRAINT CHECK(NAME = UPPER(NAME)),
  CONSTRAINT BASIC_RANGE_CONSTRAINT CHECK(BASIC>=5000 AND BASIC<=9000),
  CONSTRAINT GRADE_CONSTRAINT CHECK(GRADE IN ('A','B','C'))
);

CREATE TABLE LEAVE_REGISTER(
  EMPCODE CHAR(5) NOT NULL,
  LEAVE_TYPE CHAR(5) NOT NULL,
  FROM_DATE DATE NOT NULL,
  TO_DATE DATE NOT NULL,
  PRIMARY KEY(FROM_DATE,EMPCODE),
  FOREIGN KEY(EMPCODE) REFERENCES EMPLOYEE(EMPCODE) ON DELETE CASCADE,
  CONSTRAINT TYPE_CONSTRAINT CHECK(LEAVE_TYPE IN ('CL', 'EL', 'ML'))
);
```

2. Try to violate the constraints that you have implemented in the table & note, what happens. [Try with suitable INSERT/UPDATE/DELETE instruction]

a. Tried to violate **BASIC_RANGE_CONSTRAINT** by providing less than 5000 rupees.

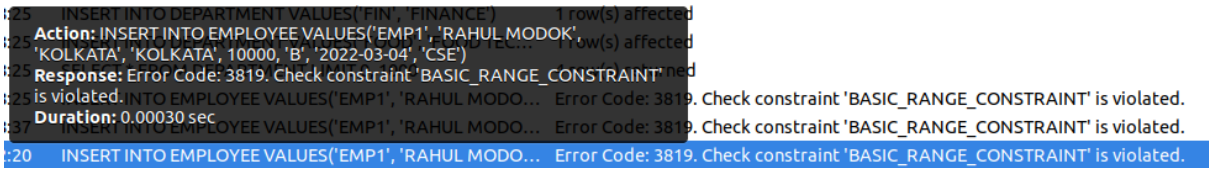
```
INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODOK', 'KOLKATA', 'KOLKATA',
4000, 'B', '2022-03-04', 'CSE');
```



```
Action: INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODOK',
'KOLKATA', 'KOLKATA', 4000, 'B', '2022-03-04', 'CSE')
Response: Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT'
is violated.
Duration: 0.00043 sec

#    Time      Action                                             Message
               ...ENT VALUES('PROD', 'PRODUCTI...                 1 row(s) affected
               ...NTO DEPARTMENT VALUES('FIN', 'FINANCE')         1 row(s) affected
105  18:48:25  INSERT INTO DEPARTMENT VALUES('FOOD', 'FOOD TEC... 1 row(s) affected
106  18:48:25  SELECT * FROM DEPARTMENT LIMIT 0, 1000             4 row(s) returned
107  18:48:25  INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO... Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.
108  18:48:37  INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO... Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.
```

b. Tried to violate **BASIC_RANGE_CONSTRAINT** by providing greater than 9000 rupees.

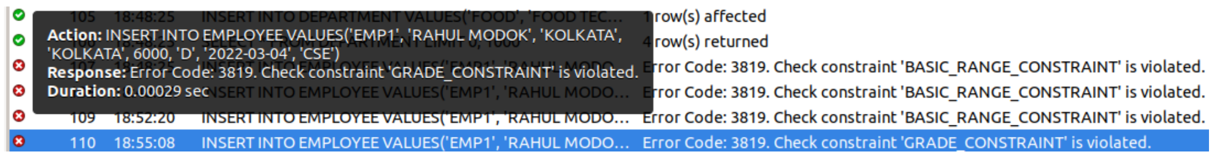
```
INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODOK', 'KOLKATA', 'KOLKATA',
10000, 'B', '2022-03-04', 'CSE');
```



```
:25   INSERT INTO DEPARTMENT VALUES('FIN', 'FINANCE')          1 row(s) affected
:25                                                            1 row(s) affected
:25
:25   INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO...   Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.
:37   INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO...   Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.
:20   INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO...   Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.

Action: INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODOK',
'KOLKATA', 'KOLKATA', 10000, 'B', '2022-03-04', 'CSE')
Response: Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT'
is violated.
Duration: 0.00030 sec
```

c. Tried to violate **GRADE_CONSTRAINT** by providing ‘D’ as grade.

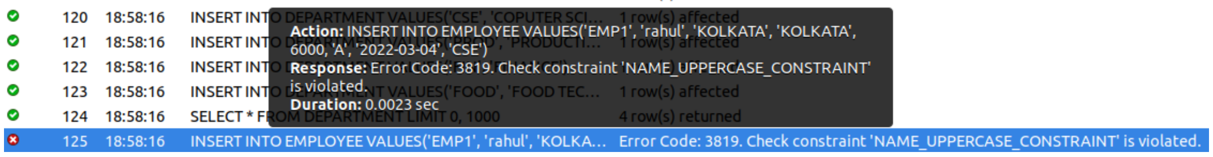
```
INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODOK', 'KOLKATA', 'KOLKATA',
6000, 'D', '2022-03-04', 'CSE');
```



```
105   18:48:25   INSERT INTO DEPARTMENT VALUES('FOOD', 'FOOD TEC...   1 row(s) affected
                                                                       4 row(s) returned
                                                                       Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.
                 INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO...    Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.
109   18:52:20   INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO...    Error Code: 3819. Check constraint 'BASIC_RANGE_CONSTRAINT' is violated.
110   18:55:08   INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODO...    Error Code: 3819. Check constraint 'GRADE_CONSTRAINT' is violated.

Action: INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODOK', 'KOLKATA',
'KOLKATA', 6000, 'D', '2022-03-04', 'CSE')
Response: Error Code: 3819. Check constraint 'GRADE_CONSTRAINT' is violated.
Duration: 0.00029 sec
```

d. Tried to violate **NAME_UPPERCASE_CONSTRAINT** by providing a lowercase name.

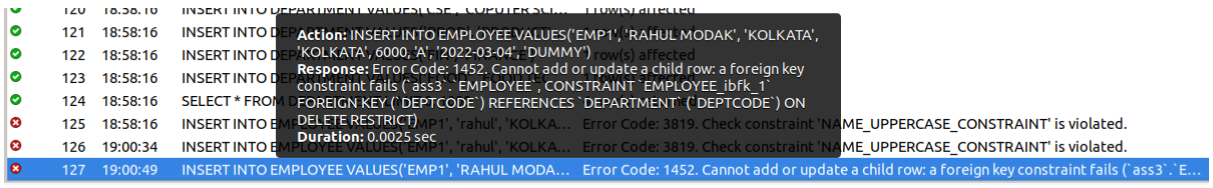
```
INSERT INTO EMPLOYEE VALUES('EMP1', 'rahul, 'KOLKATA', 'KOLKATA', 6000,
'A', '2022-03-04', 'CSE');
```



```
120   18:58:16   INSERT INTO DEPARTMENT VALUES('CSE', 'COPUTER SCI...   1 row(s) affected
121   18:58:16   INSERT INTO DEPARTMENT VALUES('PROD', 'PRODUCTI...     1 row(s) affected
122   18:58:16   INSERT INTO
123   18:58:16   INSERT INTO DEPARTMENT VALUES('FOOD', 'FOOD TEC...     1 row(s) affected
124   18:58:16   SELECT * FROM DEPARTMENT LIMIT 0, 1000                  4 row(s) returned
125   18:58:16   INSERT INTO EMPLOYEE VALUES('EMP1', 'rahul', 'KOLKA...   Error Code: 3819. Check constraint 'NAME_UPPERCASE_CONSTRAINT' is violated.

Action: INSERT INTO EMPLOYEE VALUES('EMP1', 'rahul', 'KOLKATA', 'KOLKATA',
6000, 'A', '2022-03-04', 'CSE')
Response: Error Code: 3819. Check constraint 'NAME_UPPERCASE_CONSTRAINT'
is violated.
Duration: 0.0023 sec
```

e. Create EMPLOYEE with non-existent department

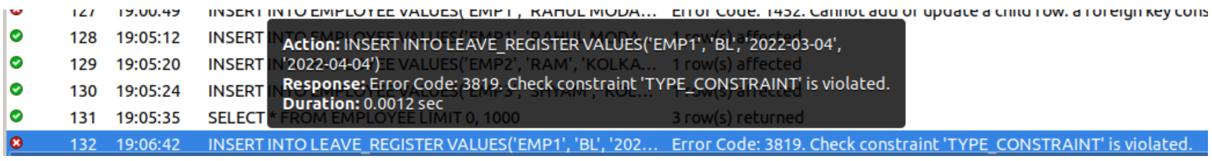
```
INSERT INTO EMPLOYEE VALUES('EMP1', ‘RAHUL MODAK’, 'KOLKATA', 'KOLKATA',
6000, 'A', '2022-03-04', 'DUMMY');
```



```
120   18:58:16   INSERT INTO DEPARTMENT VALUES('CSE', 'COPUTER SCI...   1 row(s) affected
121   18:58:16   INSERT INTO DEPARTMENT
122   18:58:16   INSERT INTO DEPARTMENT                                  row(s) affected
123   18:58:16   INSERT INTO DEPARTMENT
124   18:58:16   SELECT * FROM D
125   18:58:16   INSERT INTO EMPLOYEE VALUES('EMP1', 'rahul', 'KOLKA...   Error Code: 3819. Check constraint 'NAME_UPPERCASE_CONSTRAINT' is violated.
126   19:00:34   INSERT INTO EMPLOYEE VALUES('EMP1', 'rahul', 'KOLKA...   Error Code: 3819. Check constraint 'NAME_UPPERCASE_CONSTRAINT' is violated.
127   19:00:49   INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODA...   Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`ass3`.`E...

Action: INSERT INTO EMPLOYEE VALUES('EMP1', 'RAHUL MODAK', 'KOLKATA',
'KOLKATA', 6000, 'A', '2022-03-04', 'DUMMY')
Response: Error Code: 1452. Cannot add or update a child row: a foreign key
constraint fails (`ass3`.`EMPLOYEE`, CONSTRAINT `EMPLOYEE_ibfk_1`
FOREIGN KEY (`DEPTCODE`) REFERENCES `DEPARTMENT` (`DEPTCODE`) ON
DELETE RESTRICT)
Duration: 0.0025 sec
```

f. Create Record in **LEAVE REGISTER** with invalid **TYPE**

```
INSERT INTO LEAVE_REGISTER VALUES('EMP1', 'BL', '2022-03-04',
'2022-04-04');
```

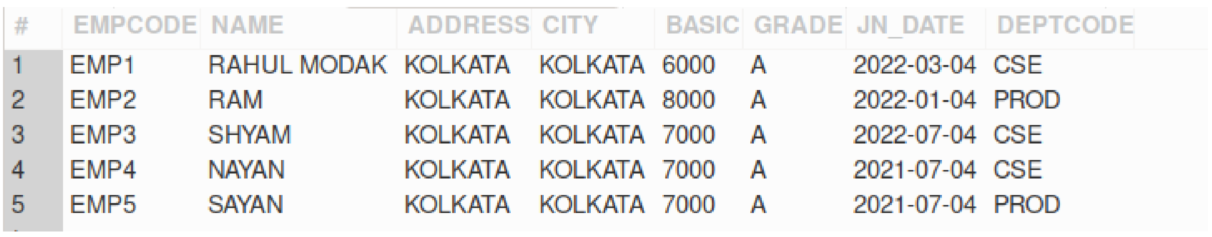


```
128  19:05:12  INSERT
129  19:05:20  INSERT                    1 row(s) affected
130  19:05:24  INSERT
131  19:05:35  SELECT
132  19:06:42  INSERT INTO LEAVE_REGISTER VALUES('EMP1', 'BL', '202...   Error Code: 3819. Check constraint 'TYPE_CONSTRAINT' is violated.

Action: INSERT INTO LEAVE_REGISTER VALUES('EMP1', 'BL', '2022-03-04',
'2022-04-04')
Response: Error Code: 3819. Check constraint 'TYPE_CONSTRAINT' is violated.
Duration: 0.0012 sec
```

3. a) Create a table having empcode , Name, deptname, & basic From the existing tables along with the records of the employee who are in a particular department (say, d1) and with a basic Rs. 7000/-

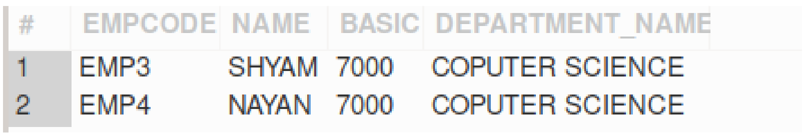
```
CREATE TABLE NEW_EMPLOYEE AS
SELECT EMPLOYEE.EMPCODE, EMPLOYEE.NAME, EMPLOYEE.BASIC, DEPARTMENT.NAME
AS DEPARTMENT_NAME
FROM EMPLOYEE
INNER JOIN DEPARTMENT ON DEPARTMENT.DEPTCODE = EMPLOYEE.DEPTCODE
WHERE EMPLOYEE.BASIC = 7000 AND EMPLOYEE.DEPTCODE = 'CSE';
```

EMPLOYEE TABLE -



#	EMPCODE	NAME	ADDRESS	CITY	BASIC	GRADE	JN_DATE	DEPTCODE
1	EMP1	RAHUL MODAK	KOLKATA	KOLKATA	6000	A	2022-03-04	CSE
2	EMP2	RAM	KOLKATA	KOLKATA	8000	A	2022-01-04	PROD
3	EMP3	SHYAM	KOLKATA	KOLKATA	7000	A	2022-07-04	CSE
4	EMP4	NAYAN	KOLKATA	KOLKATA	7000	A	2021-07-04	CSE
5	EMP5	SAYAN	KOLKATA	KOLKATA	7000	A	2021-07-04	PROD

OUTPUT [NEW EMPLOYEE TABLE] -

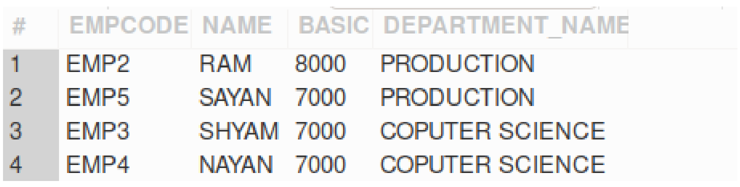


#	EMPCODE	NAME	BASIC	DEPARTMENT_NAME
1	EMP3	SHYAM	7000	COPUTER SCIENCE
2	EMP4	NAYAN	7000	COPUTER SCIENCE

3. b) From the existing table, add the employees with the basic salary greater than or equal to 7000/-

```
INSERT INTO NEW_EMPLOYEE (
      SELECT EMPLOYEE.EMPCODE, EMPLOYEE.NAME, EMPLOYEE.BASIC,
DEPARTMENT.NAME AS DEPARTMENT_NAME FROM EMPLOYEE
    INNER JOIN DEPARTMENT ON DEPARTMENT.DEPTCODE = EMPLOYEE.DEPTCODE
    WHERE EMPLOYEE.BASIC >= 7000
);
```

OUTPUT [NEW EMPLOYEE TABLE] -



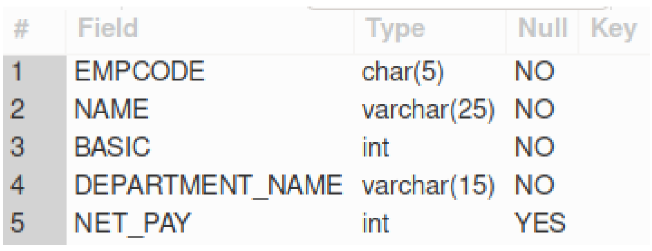
#	EMPCODE	NAME	BASIC	DEPARTMENT_NAME
1	EMP2	RAM	8000	PRODUCTION
2	EMP5	SAYAN	7000	PRODUCTION
3	EMP3	SHYAM	7000	COPUTER SCIENCE
4	EMP4	NAYAN	7000	COPUTER SCIENCE

3. c) Alter the table to add a net pay column.

Add new column

```
ALTER TABLE NEW_EMPLOYEE ADD NET_PAY INT;
DESCRIBE NEW_EMPLOYEE;
```

OUTPUT -

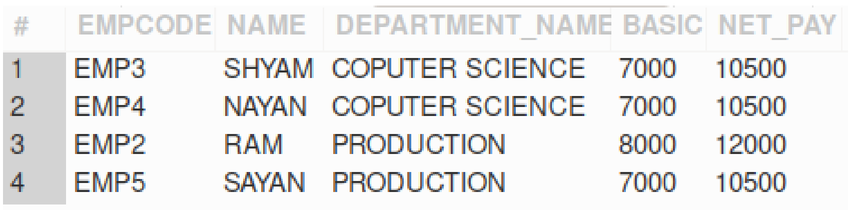


#	Field	Type	Null	Key
1	EMPCODE	char(5)	NO	
2	NAME	varchar(25)	NO	
3	BASIC	int	NO	
4	DEPARTMENT_NAME	varchar(15)	NO	
5	NET_PAY	int	YES	

3. d) Replace NET_PAY with 1.5* Basic.

```
UPDATE NEW_EMPLOYEE SET NET_PAY = 1.5*BASIC;
```

OUTPUT -

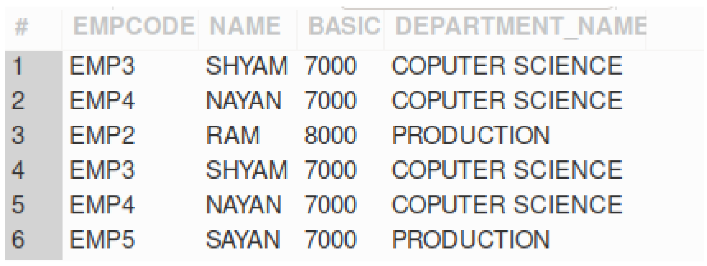


#	EMPCODE	NAME	DEPARTMENT_NAME	BASIC	NET_PAY
1	EMP3	SHYAM	COPUTER SCIENCE	7000	10500
2	EMP4	NAYAN	COPUTER SCIENCE	7000	10500
3	EMP2	RAM	PRODUCTION	8000	12000
4	EMP5	SAYAN	PRODUCTION	7000	10500

3. e) Try to remove the net net pay column.

```
ALTER TABLE NEW_EMPLOYEE DROP COLUMN NET_PAY;
```

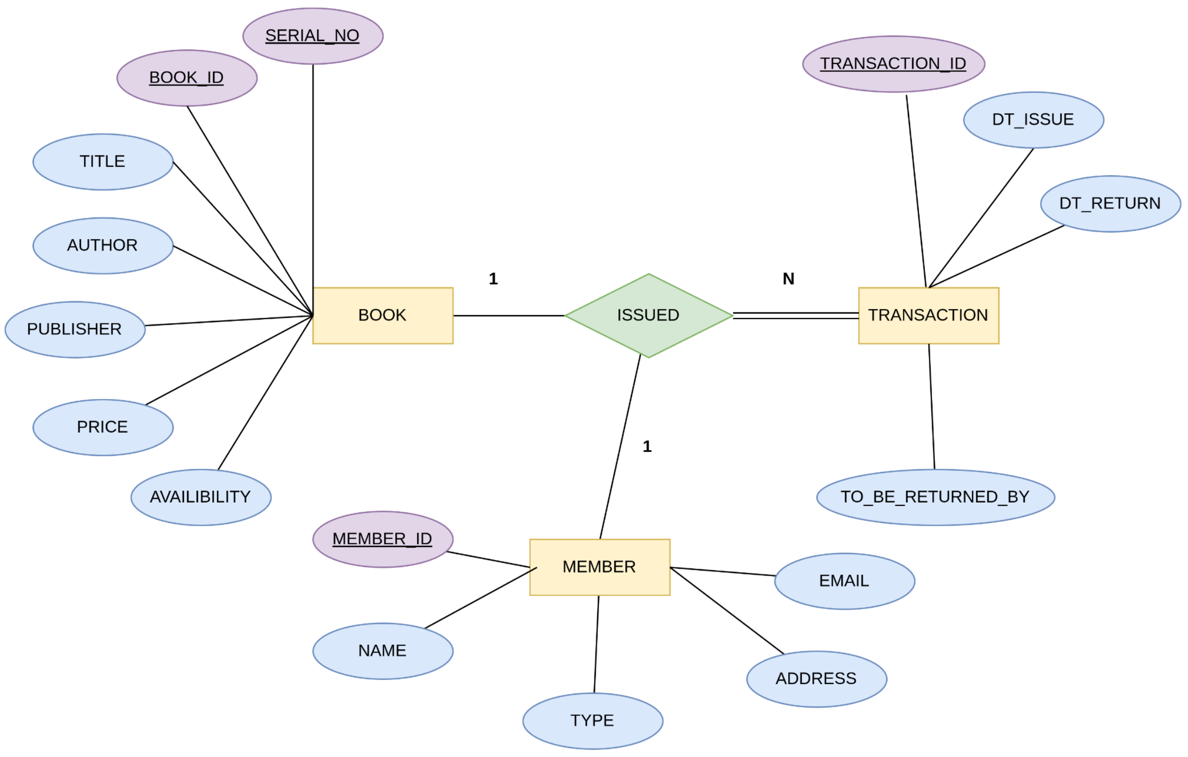
OUTPUT -



#	EMPCODE	NAME	BASIC	DEPARTMENT_NAME
1	EMP3	SHYAM	7000	COPUTER SCIENCE
2	EMP4	NAYAN	7000	COPUTER SCIENCE
3	EMP2	RAM	8000	PRODUCTION
4	EMP3	SHYAM	7000	COPUTER SCIENCE
5	EMP4	NAYAN	7000	COPUTER SCIENCE
6	EMP5	SAYAN	7000	PRODUCTION

5. In a library, for each book book-id, serial number (denotes copy number of a book), title, author, publisher and price are stored. Book-id and serial number together will be a unique identifier for a book. Members are either students or faculty. Each member has a unique member-id. Name, e-mail, address are also to be stored. Maximum number of books that a member can retain depends on the member type. There may be other such parameters that depend on member type. Design should be flexible. For any transaction (book issue or return), members are supposed to place transactions slip. Each Transaction will have a unique id. Users will submit member-id, book-id, and serial number (only for book return). Design and create the tables to store the book, member and transaction information. When a book is issued to a member of a field like, To_Be_Returned_By has to be set as DT_Issue + 7 days. At the time of book return, DT_Return will store the actual return date. While a new book arrives, the serial number will be the last serial number for the Book-id +1. System should also keep track of the status of each physical book -- whether issued or available.

ER Diagram :



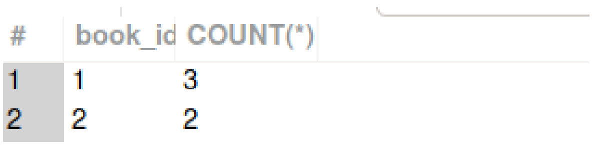
CREATE TABLE

```
-- Create tables
CREATE TABLE book(
    book_id INT NOT NULL,
    serial_num INT NOT NULL,
    title VARCHAR(100),
    author VARCHAR(50),
    publisher VARCHAR(60),
    price INT,
    available BOOLEAN DEFAULT true,
    PRIMARY KEY (book_id, serial_num)
);


CREATE TABLE member(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(25),
    email VARCHAR(100),
    member_type CHAR(7),
    max_books INT,
    CONSTRAINT check_type CHECK(member_type IN ('faculty', 'student')),
    CONSTRAINT check_maxbooks CHECK((member_type = 'faculty' AND
max_books = 10) OR (member_type = 'student' AND max_books = 4))
);

CREATE TABLE transaction_slip(
    id INT PRIMARY KEY AUTO_INCREMENT,
    member_id INT,
    book_id INT,
    book_serial INT,
    issue_date DATE DEFAULT(CURRENT_DATE),
    return_date DATE,
    to_be_returned_by DATE,
    FOREIGN KEY (member_id) REFERENCES member(id) ON DELETE RESTRICT,
    FOREIGN KEY (book_id, book_serial) REFERENCES book(book_id,
serial_num) ON DELETE RESTRICT
);

-- Create trigger to verify all condition of a transaction
delimiter //
CREATE TRIGGER transaction_trigger
BEFORE INSERT ON transaction_slip
FOR EACH ROW
BEGIN
      DECLARE msg VARCHAR(32) DEFAULT "";
    -- checks for returning books
    IF (NEW.issue_date IS NULL AND NEW.return_date IS NOT NULL) THEN
        IF NOT EXISTS (SELECT * FROM book  WHERE book_id = NEW.book_id
AND serial_num = NEW.book_serial AND available = false) THEN
            set msg = concat('Book is already returned !');
            signal sqlstate '45000' set message_text = msg;
        ELSE
            UPDATE book SET available = NOT available WHERE book_id =
NEW.book_id AND serial_num = NEW.book_serial;
            UPDATE transaction_slip SET return_date = NEW.return_date
            WHERE member_id = NEW.member_id AND book_id = NEW.book_id
AND book_serial = NEW.book_serial AND return_date IS NULL;
        END IF;
      -- checks for issuing books
    ELSE
            -- if book not available
        IF NOT EXISTS ( SELECT * FROM book WHERE book_id = NEW.book_id
AND serial_num = NEW.book_serial AND available = true) THEN
            set msg = concat('Book Not Available !');
            signal sqlstate '45000' set message_text = msg;
        ELSE
        -- if available, update
            IF (SELECT COUNT(*) FROM transaction_slip t WHERE
t.ISSUE_DATE is not NULL and t.RETURN_DATE is NULL and t.member_id =
NEW.member_id) >= (SELECT m.max_books FROM member m WHERE m.id =
NEW.member_id) THEN
                  set msg = concat('Member has issued max books !');
                  signal sqlstate '45000' set message_text = msg;
            ELSE
                  UPDATE book SET available = NOT available WHERE
book_id = NEW.book_id AND serial_num = NEW.book_serial;
                  SET NEW.to_be_returned_by =
CAST(DATE_ADD(NEW.issue_date, INTERVAL 7 DAY) as DATE);
            END IF;
        END IF;
    END IF;
END;
//

-- Create trigger to increase serial_no automatically
delimiter //
CREATE TRIGGER before_insert_update_serial_no
BEFORE INSERT ON book
FOR EACH ROW
BEGIN
      SET NEW.serial_num = COALESCE((SELECT MAX(serial_num) FROM book
WHERE book_id = NEW.book_id), 0) + 1;
END
//
```

a) Display total number of copies (irrespective of issued or not) for each book in the library and number of such copies issued

```
SELECT book_id, COUNT(*) FROM book
GROUP BY book_id;
```

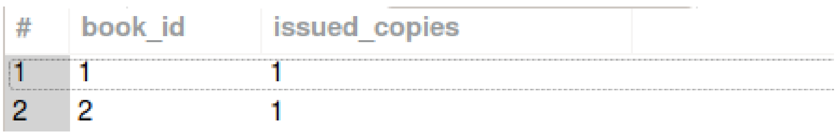
OUTPUT -



#	book_id	COUNT(*)
1	1	3
2	2	2

```
SELECT book_id, COUNT(*) AS issued_copies FROM book
GROUP BY book_id, available HAVING available = false;
```

OUTPUT -

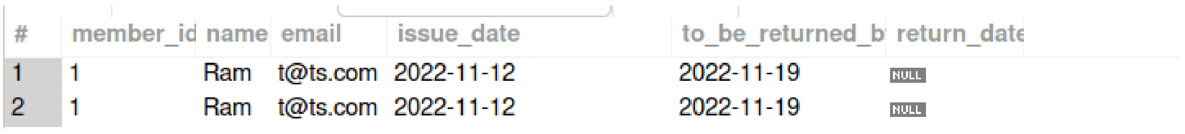


#	book_id	issued_copies
1	1	1
2	2	1

b) Find the members holding the books even after due date

```
SELECT t1.member_id, member.name, member.email, t1.issue_date,
t1.to_be_returned_by, t1.return_date FROM
transaction_slip AS t1
JOIN member ON member.id = t1.member_id
WHERE t1.to_be_returned_by < CURRENT_DATE AND t1.return_date IS NULL;
```

OUTPUT -

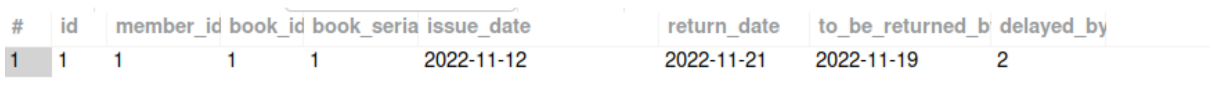


#	member_id	name	email	issue_date	to_be_returned_b	return_date
1	1	Ram	t@ts.com	2022-11-12	2022-11-19	NULL
2	1	Ram	t@ts.com	2022-11-12	2022-11-19	NULL

c) Find the transaction details for delayed book returns and delay in terms of number of days.

```
SELECT t1.member_id, member.name, member.email, t1.issue_date,
t1.to_be_returned_by, t1.return_date FROM
transaction_slip AS t1
JOIN member ON member.id = t1.member_id
WHERE t1.to_be_returned_by < CURRENT_DATE AND t1.return_date IS NULL;
```

OUTPUT -



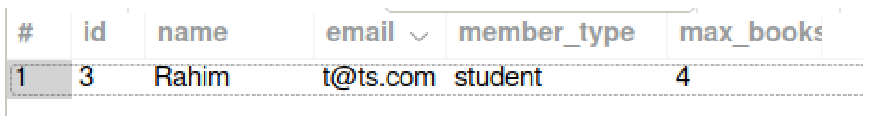
#	id	member_id	book_id	book_seria	issue_date	return_date	to_be_returned_b	delayed_by
1	1	1	1	1	2022-11-12	2022-11-21	2022-11-19	2

d) Find the student members not making any transaction and do the same for faculty members.

Students not making any transactions

```
SELECT member.* FROM member
LEFT JOIN transaction_slip AS t ON t.member_id = member.id
WHERE t.member_id IS NULL and member.member_type = 'student';
```

OUTPUT -

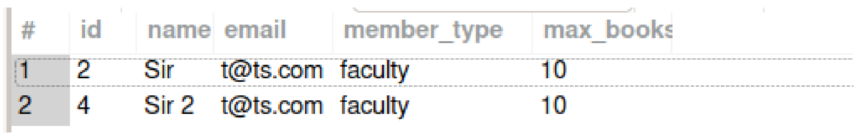


#	id	name	email	member_type	max_books
1	3	Rahim	t@ts.com	student	4

Faculties not making any transactions

```
SELECT member.* FROM member
LEFT JOIN transaction_slip AS t ON t.member_id = member.id
WHERE t.member_id IS NULL and member.member_type = 'faculty';
```

OUTPUT -

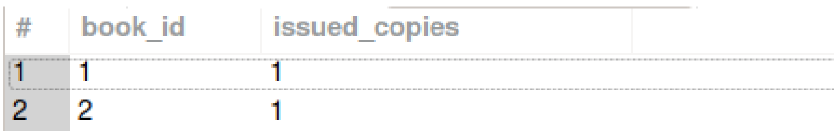


#	id	name	email	member_type	max_books
1	2	Sir	t@ts.com	faculty	10
2	4	Sir 2	t@ts.com	faculty	10

e) Find the count of issue for each book (not the specific copy).

```
SELECT book_id, COUNT(*) AS issued_copies FROM book
GROUP BY book_id, available HAVING available = false;
```

OUTPUT -



#	book_id	issued_copies
1	1	1
2	2	1