

INDEX

1. Normalisation.....
2. SQL Query to create Table.....
3. Tables.....
4. Simple Queries.....
5. Complex Queries.....
6. Views.....
7. Procedures.....
8. Triggers.....
9. Functions.....

1 . PROBLEM STATEMENT:

Every Year more than 1500+ student take admission at the National Institute of Technology, Karnataka, Surathkal (Herein referred to as NITK in the report) for various Undergraduate, Postgraduate, Management and Doctoral studies. Often times, these students come from various parts of India, or even different countries to study. Such students are classified into the category of 'Hostelers' and are allocated a hostel room in either of the several hostel blocks, upon payment of certain amount of fees. They can also avail mess food from either one of the eight-mess spread across the southern part of the campus, depending on their preference and on first come-first serve basis. Please do note that there are separate hostel blocks, as well as mess for boys and girls.

The responsibility for allocating hostel blocks and mess to the newly admitted student is carried out by the Mess and Hostel Allocation Office (Located in Sahyadri Block 7) for both boys and girls. This office comes under the office of Dean- Student Welfare.

However, due to the ever-increasing number of students, the procedure has become increasingly more hectic and gruesome. Students have to wait in hour long Queues for completing basic Fee payment for Mess and Hostel allocation and then wait their turns to site their hostel block and mess preferences, only after which they are handed over the keys to their room and the mess cards to their respective mess. With the process being mostly carried out with the archaic skill of book-keeping, leads to an increase in error due to human fault.

The project under the course IT252 titles 'IRIS Hostel and Mess Management System' aims to :-

- To integrate the Hostel and Mess Allotment process with Integrated Resource and Information Sharing (IRIS), NITK's digital platform for various purposes like fee payment, online resourcing of academics, etc.
- To semi-automate the process of Hostel and Mess allotment, so as to reduce human errors to maximum extent
- To make the Fee payment procedure for both hostel and Mess available via Online Platform.
- Most importantly, to maintain a database of the students with respectively allocated mess and hostel rooms, which can be easily accessed by the actors (mentioned later on in the report)

2 . ACTORS:

People who interact with the database are:-

- Students availing hostel, aka Hostellers
- Employees of The Mess and Hostel Allocation Office
- Dean Student Welfare's Office
- IRIS management

3 . SOME SAMPLE QUERIES:

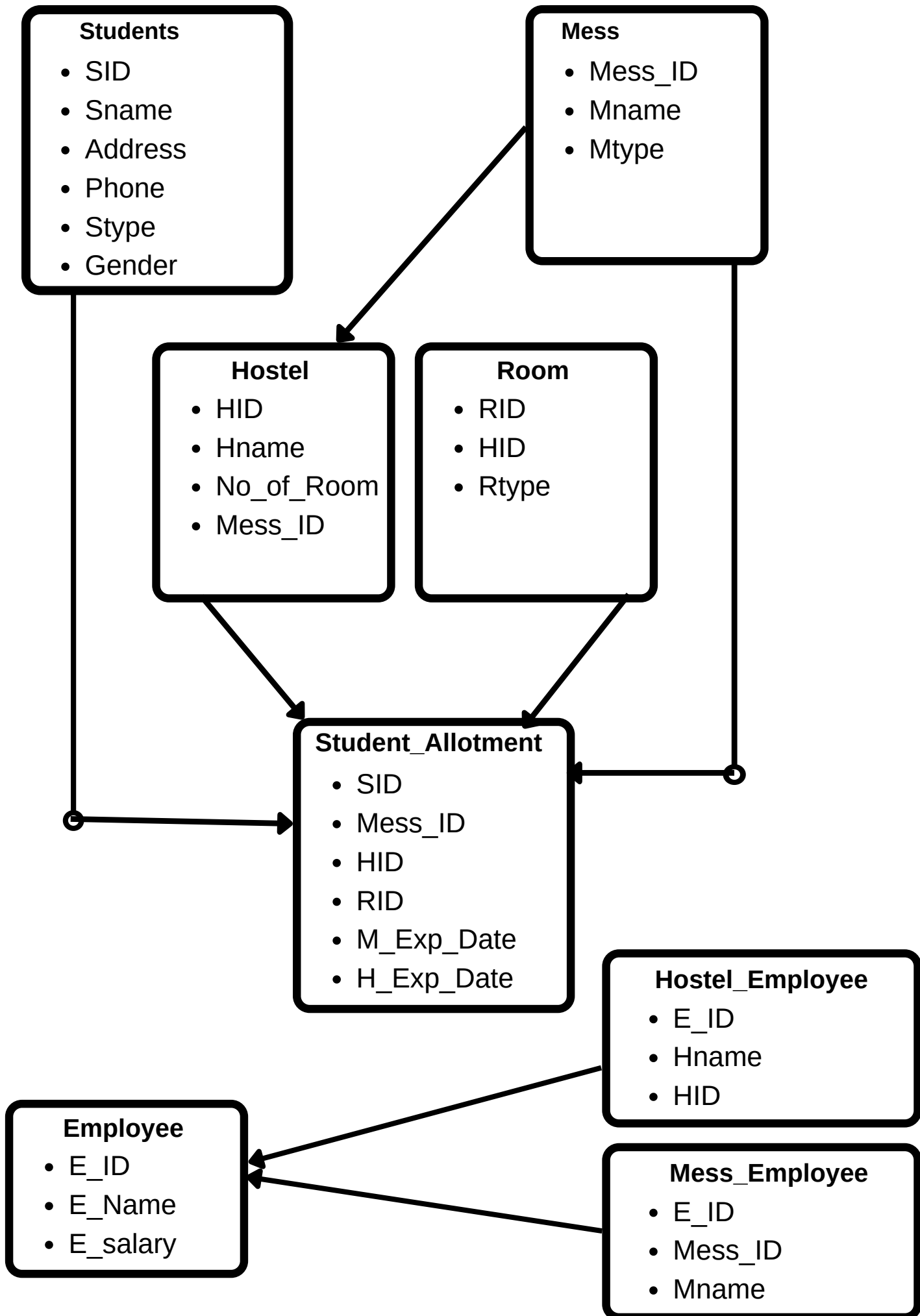
A student can

- Check the list of total mess in NITK.
- Check list of hostel blocks in NITK.
- Check List of Veg and Non-Veg Mess in NITK..
- Get information about their allotment status.

Mess or Hostel authority can

- Check No of students staying in a hostel.
- Check No of students in a mess.
- Check the expiry status of the mess of a particular student.
- Verify the enrollment process.

4. EER MODEL (CONCEPTUAL MODEL) {BEFORE NOMALIZATION}



5. GLOBAL CONCEPTUAL SCHEMA (BEFORE NOMALIZATION)

Students

SID	Sname	Address	Phone	Stype
Gender				

Mess

Mess_ID	Mname	Mtype
---------	-------	-------

Hostel

HID	Hname	No_of_Room	MID
-----	-------	------------	-----

Room

RID	HID	Rtype
-----	-----	-------

Student_Allotment

SID	MID	HID	RID	M_Exp_Date
-----	-----	-----	-----	------------

H_Exp_Date

Employee

E_ID	E_Name	E_salary
------	--------	----------

Hostel_Employee

E_ID	Hname	HID
------	-------	-----

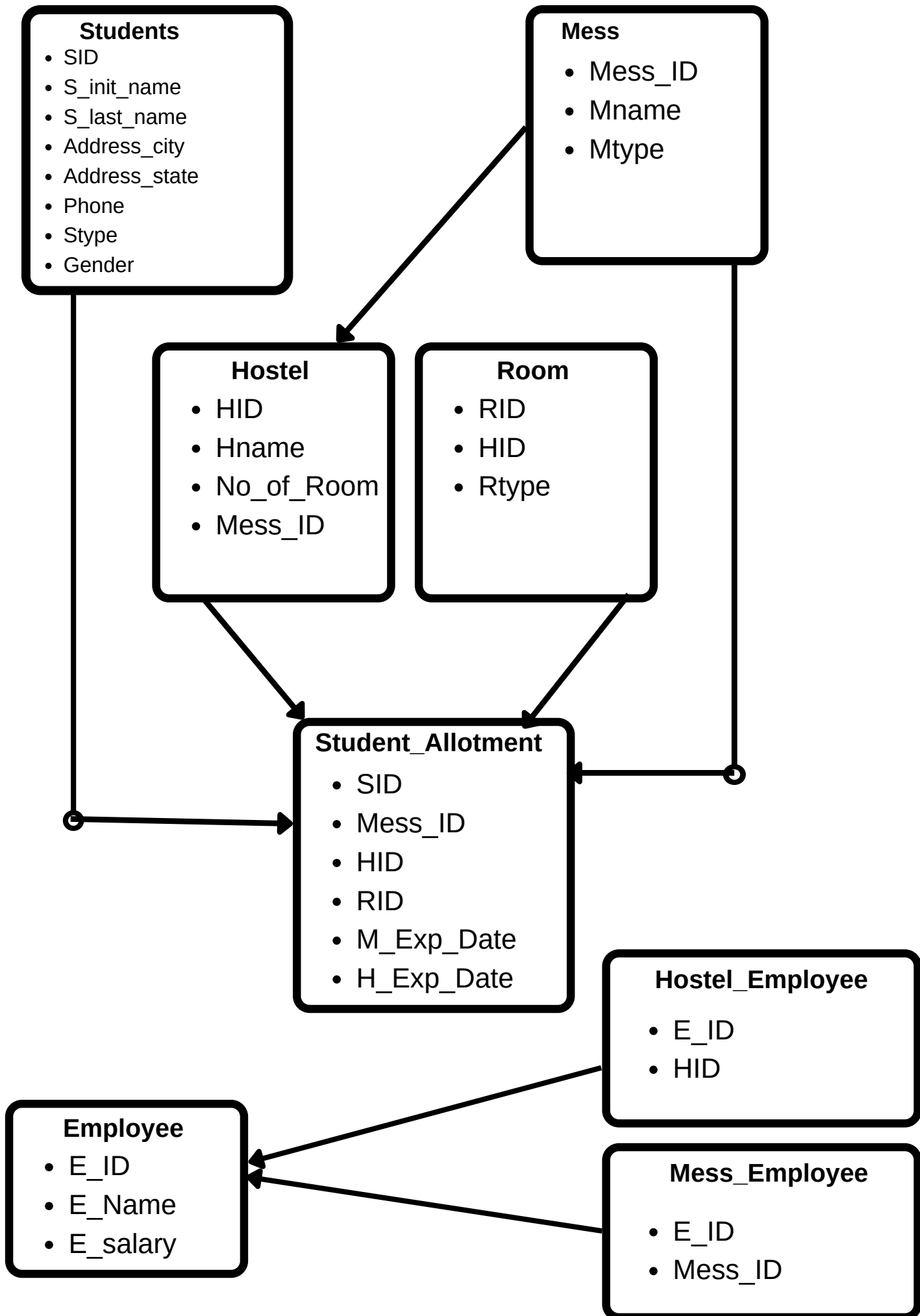
Mess_Employee

Mess_ID	Mname	E_ID
---------	-------	------

Wardens

WID	HID	Wname	Phone
-----	-----	-------	-------

4. EER MODEL (CONCEPTUAL MODEL) {AFTER NOMALIZATION}



Normalisation:

One of the biggest issues database schema designers have faced happens to be data redundancy, which is also one of the biggest factors which gave rise to relational schema which provided an extremely intuitive & sophisticated method of removing redundancy from in our databases. Data redundancy refers to the state of the database in which the same piece of data occupies more than storage space in more than 2 separate locations. This leads to inefficient queries & wastage of a lot of storage space. The concept of Normalization was introduced to overcome the issue of data redundancy.

Anomalies are caused when there is too much redundancy in the database's information. Anomalies can often be caused when the tables that make up the database suffer from poor construction. There are three types of Data Anomalies: Update Anomalies, Insertion Anomalies, and Deletion Anomalies.

- **Update Anomalies** happen when the person charged with the task of keeping all the records current and accurate, is asked, for example, to change an employee's title due to a promotion. If the data is stored redundantly in the same table, and the person misses any of them, then there will be multiple titles associated with the employee. The end user has no way of knowing which is the correct title.
- **Insertion Anomalies** happen when inserting vital data into the database is not possible because other data is not already there. For example, if a system is designed to require that a

customer be on file before a sale can be made to that customer, but you cannot add a customer until they have bought something, then you have an insert anomaly. It is the classic "catch-22" situation.

- **Deletion Anomalies** happen when the deletion of unwanted information causes desired information to be deleted as well. For example, if a single database record contains information about a particular product along with information about a salesperson for the company and the salesperson quits, then information about the product is deleted along with salesperson information.

Normalisation is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation usually causes insertion, deletion and updation anomalies as stated previously. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

Relational schema is a derivative of set theory in mathematics & hence it is extremely easy to analyze relational schemas by considering relations as sets & applying the concepts of set theory over them. In a set, we map a key to a particular value, such that $t[X] = y$ & for each X there exists a y . We express this function symbolically as $X \rightarrow y$, where (\rightarrow stands for "determines"). This relation is known as functional dependency in DBMS, where a key is used to determine the rest of the attributes of a tuple of R .

Closure of a Set - The set of all those attributes which can be functionally determined from an attribute set is called a closure of that attribute set.

Super Key - Super Key is an attribute (or set of attributes) that is used to uniquely identify all attributes in a relation.

Candidate Key - Candidate key is a set of attributes (or attribute) which uniquely identify the tuples in a relation or table.

Primary Key - Minimal super key which is used to uniquely identify a tuple in a relation.

Every normal form should be lossless, and FD preserved.

There are three(in real, there are 5) different types of normal forms, each being a more irreducible form of the same data, such that we reach a 0% redundancy state at the end. BCNF is considered to be a 0% redundancy state & our goal is to reach BCNF of our relational schema.

1st Normal Form: A relation is said to be in first normal form then it should satisfy the following

- No multi-valued attribute
- No composite attribute
- Identify primary key

Here, the relationship is converted to either relation or foreign key or merging relations.

2nd Normal Form:

- Repeating column values are taken out and maintained in a separate table. So that change can be done only once in the new table rather than all entries in the first table. Rule is foreign key must be on the N side else again multi-value in a column will occur.
- Identify the prime **attribute** (part of the candidate key that determines anything else), it is also called **partial dependency**, and eliminate it. Because, 2nd NF is based on **Full Functional dependency** (key should determine all other attributes in a table)
- Use foreign keys on many sides.

3rd Normal Form:

- Only columns with direct dependency of the primary key shall be in the entity.
- No transitive dependencies: non-prime attributes transitively depending on the key.
Example: $A \rightarrow B \rightarrow C \implies A \rightarrow C$. Example: $A \rightarrow B \rightarrow C \implies A \rightarrow C$.
 $A \rightarrow B$ is a non-key attribute here, $B \rightarrow C$ suddenly becomes a key attribute here. Because of this, we will get repeated values in a column. Therefore, it should be eliminated.
- 3rd NF should hold the condition that: if $X \rightarrow Y$ then
Either X is a super key Or Y is a prime attribute
Following this condition will never allow transitive dependency.

5. GLOBAL CONCEPTUAL SCHEMA (AFTER NOMALIZATION)

Students

SID	S_init_name	S_last_name	Gender	Address_City
Address_state	Phone	Stype		

Mess

MID	Mname	Mtype
-----	-------	-------

Hostel

HID	Hname	No_of_Room	MID
-----	-------	------------	-----

Room

RID	HID	Rtype
-----	-----	-------

Student_Allotment

SID	MID	HID	RID	M_Exp_Date
				H_Exp_Date

Employee

E_ID	E_Name	E_salary
------	--------	----------

Hostel_Employee

E_ID	HID
------	-----

Mess_Employee

Mess_ID	E_ID
---------	------

Wardens

WID	HID	Wname	Phone
-----	-----	-------	-------

SQL Query to Create Table

I. Students

```
CREATE TABLE Students (  
    SID int PRIMARY KEY,  
    S_init_name varchar(125) NOT NULL,  
    S_last_name varchar(125) NOT NULL,  
    Gender char NOT NULL,  
    Address_City varchar(150) NOT NULL,  
    Address_state varchar(150) NOT NULL,  
    Phone bigint NOT NULL  
);
```

II. Mess

```
CREATE TABLE Mess (  
    Mess_ID int PRIMARY KEY,  
    Mname varchar(25) NOT NULL,  
    Mtype varchar(25) NOT NULL  
);
```

III. Hostel

```
CREATE TABLE Hostel (  
    HID int PRIMARY KEY,  
    Hname varchar(25) NOT NULL,  
    Mess_ID int NOT NULL ,  
    FOREIGN KEY (Mess_ID) REFERENCES Mess (Mess_ID)  
);
```

IV. Hostel Employee

```
create table Hostel_Employee(  
    HID int ,  
    Hname varchar(140) NOT NULL,  
    E_ID int NOT NULL,  
    foreign key (HID) references Hostel(HID),  
    FOREIGN key (E_ID) references Employee(E_ID)  
);  
  
create table Employee(  
    E_Name varchar(100) NOT NULL ,  
    E_ID int primary key ,  
    E_salary int NOT NULL,  
);
```

V. Mess Employee

```
create table Mess_Employee(  
  
    Mess_ID int ,  
    Mname varchar(140) NOT NULL,  
    E_ID int NOT NULL,  
    FOREIGN key (Mess_ID) references Mess(Mess_ID),  
    FOREIGN key (E_ID) references Employee(E_ID)  
  
);
```

VI. Room

```
CREATE TABLE Room (  
    RID varchar(25) PRIMARY KEY,  
    HID int NOT NULL,  
    Rtpye varchar(25) NOT NULL,
```

```
FOREIGN KEY (HID) REFERENCES Hostel(HID)
);
```

VII. Student Allotment

```
CREATE TABLE Student_Allotment (
    SID int PRIMARY KEY,
    Mess_ID int NOT NULL,
    HID int NOT NULL,
    RID varchar(25) NOT NULL,

    FOREIGN KEY (HID) REFERENCES Hostel(HID),
    FOREIGN KEY (Mess_ID) REFERENCES Mess(Mess_ID),
    FOREIGN KEY (RID) REFERENCES Room(RID)
);
```

Tables

I. Student

```
mysql> select * from students;
```

SID	S_init_name	S_last_name	stype	Gender	Address_City	Address_state	Phone
1	Spandan	Patkar	0	M	Karjat	Maharashtra	8308955566
2	Rushit	Kausal	1	M	Vadodra	Uttrakhand	8308332316
3	Jatin	Majumdar	0	M	Rishikesh	Gujrat	8308337786
4	Rohit	Bhojwani	0	M	Kolkata	Chattisgarh	8308333523
5	Avinash	Taparia	1	M	Raigarh	WestBengal	8308334234
6	Ayush	Taparia	0	M	Siliguri	WestBengal	8308334643
7	Jatin	Taparia	1	M	Rishikesh	Bihar	8308334636
8	Avinash	Rauniyar	1	M	Raigarh	Up	8308345234

```
8 rows in set (0.00 sec)
```

II. Mess

```
mysql> select * from mess;
```

Mess_ID	Mname	Mtype
1	Vindhya	NVG
2	Satpura	NVG
3	Arawali	NVG
4	Karawali	VG
5	Sahydari	VG
6	MegaMess	VG

```
6 rows in set (0.00 sec)
```

III. Hostel

```
mysql> select * from Hostel;
```

HID	Hname	Mess_ID	N_Rooms
1	Vindhya	1	20
2	Satpura	2	20
3	Arawali	3	20
4	Karawali	4	20
5	Sahydari	5	20
6	Himalya	6	20
7	Trishul	6	20
8	Everest	6	20

```
8 rows in set (0.00 sec)
```

IV. Hostel Employee

```
mysql> select * from Hostel_Employee;
```

HID	E_ID
1	1231
2	1238
3	1237
4	1233
5	1234
6	1237
7	1232
8	1236
1	1239

```
9 rows in set (0.00 sec)
```


V. Employee

```
mysql> select * from employee;
+-----+-----+-----+
| E_Name | E_ID | E_salary |
+-----+-----+-----+
| Sai    | 1231 | 10000    |
| Shyam  | 1232 | 20000    |
| Adarsh | 1233 | 15000    |
| Raman  | 1234 | 18000    |
| Rohit  | 1235 | 40000    |
| Dhoni  | 1236 | 30000    |
| Onkar  | 1237 | 25000    |
| Samosa | 1238 | 45000    |
| Chutney| 1239 | 20000    |
+-----+-----+-----+
9 rows in set (0.00 sec)
```

VI. Mess Employee

```
mysql> select * from Mess_employee;
+-----+-----+
| Mess_ID | E_ID |
+-----+-----+
| 1       | 1231 |
| 2       | 1232 |
| 3       | 1233 |
| 4       | 1234 |
| 5       | 1235 |
| 6       | 1236 |
| 2       | 1237 |
| 3       | 1238 |
+-----+-----+
8 rows in set (0.00 sec)
```

VII. Room

```
mysql> select * from Room;
+-----+-----+-----+
| RID   | HID   | Rtpye  |
+-----+-----+-----+
| A-001 | 6     | Single |
| A-002 | 6     | Single |
| A-003 | 6     | Single |
| A-401 | 1     | Double |
| A-402 | 1     | Double |
| A-403 | 1     | Double |
| B-101 | 8     | Single |
| B-102 | 8     | Single |
| B-103 | 8     | Single |
| B-501 | 5     | Triple |
| B-502 | 5     | Triple |
| B-503 | 5     | Triple |
| C-201 | 7     | Triple |
| C-202 | 7     | Triple |
| C-203 | 7     | Triple |
| C-601 | 6     | Single |
| C-602 | 6     | Single |
| C-603 | 6     | Single |
| D-301 | 2     | Double |
| D-302 | 2     | Double |
| D-303 | 2     | Double |
+-----+-----+-----+
21 rows in set (0.00 sec)
```

VIII. Student Allotment

```
mysql> select * from Student_allotment;
```

SID	Mess_ID	HID	RID	M_exp_date	H_exp_date
1	1	1	A-001	2022-05-17	2022-05-17
2	2	2	B-101	2022-05-17	2022-05-17
3	3	3	C-201	2022-05-17	2022-05-17
4	4	4	D-301	2022-05-17	2022-05-17
5	5	5	A-401	2022-05-17	2022-05-17
6	6	6	B-501	2022-05-17	2022-05-17
7	3	7	C-601	2022-05-17	2022-05-17
8	4	8	A-403	2022-05-17	2022-05-17

```
8 rows in set (0.00 sec)
```

SIMPLE QUERIES:

I. Query to obtain average salary of Employees.

Query: select avg(E_Salary) from Employee;

```
mysql> select avg(E_Salary) from Employee;
+-----+
| avg(E_Salary) |
+-----+
|      24777.7778 |
+-----+
1 row in set (0.05 sec)
```

II. Query for Finding Details of mess veg or non veg

Query: Select Mname , Mtype from Mess where Mess_ID = 2;

```
mysql> Select Mname , Mtype from Mess where Mess_ID = 2;
+-----+-----+
| Mname  | Mtype |
+-----+-----+
| Satpura | NVG   |
+-----+-----+
1 row in set (0.00 sec)
```

III. Finding all hostels having triple sharing

Query: SELECT * from room WHERE Rtype = 'Triple';

```
mysql> SELECT * from room WHERE Rtpye = 'Triple';
```

RID	HID	Rtpye
B-501	5	Triple
B-502	5	Triple
B-503	5	Triple
C-201	7	Triple
C-202	7	Triple
C-203	7	Triple

```
6 rows in set (0.01 sec)
```

IV. Find count of day scholars i.e. stype=0

Query: SELECT * from Students where stype=0;

```
mysql> SELECT * from Students where stype=0;
```

SID	S_init_name	S_last_name	stype	Gender	Address_City	Address_state	Phone
1	Spandan	Patkar	0	M	Karjat	Maharashtra	8308955566
3	Jatin	Majumdar	0	M	Rishikesh	Gujrat	8308337786
4	Rohit	Bhojwani	0	M	Kolkata	Chattisgarh	8308333523
6	Ayush	Taparia	0	M	Siliguri	WestBengal	8308334643

```
4 rows in set (0.14 sec)
```

V. Find room no.,hostel and mess allotted to student with specific SID (say SID=6)

Query: SELECT * from student_allotment WHERE SID = 6;

```
mysql> SELECT * from student_allotment WHERE SID = 6;
```

SID	Mess_ID	HID	RID
6	6	6	B-501

```
1 row in set (0.00 sec)
```

COMPLEX QUERIES:

I. Find Hostel, Mess expiry dates and Last name of student with a given first name (say first name = 'Spandan')

Query: select sa.M_exp_date,sa.H_exp_date from Student_Allotment sa join Students s on (s.SID = sa.SID and s.S_init_name = "Spandan");

```
mysql> select s.S_last_name,sa.M_exp_date,sa.H_exp_date from Student_Allotment sa join
Students s on (s.SID = sa.SID and s.S_init_name = "Spandan");
+-----+-----+-----+
| S_last_name | M_exp_date | H_exp_date |
+-----+-----+-----+
| Patkar      | 2022-05-17 | 2022-05-17 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

II. Get details of mess with maximum number of students

Query: select sa.Mess_ID,m.Mname, Count(sa.Mess_ID) as Max_Student_Count from Student_Allotment sa join Mess m on (sa.Mess_ID = m.Mess_ID) group by sa.Mess_ID having Count(sa.Mess_ID) = (select max(a.mbsa) from (select Mess_ID, Count(Mess_ID) as mbsa from Student_Allotment group by Mess_ID) a);

```
mysql> select sa.Mess_ID,m.Mname, Count(sa.Mess_ID) as Max_Student_Count from S
tudent_Allotment sa join Mess m on (sa.Mess_ID = m.Mess_ID) group by sa.Mess_ID
having Count(Mess_ID) = (select max(a.mbsa)
from (select Mess_ID, Count(Mess_ID) as mbsa from Student_Allotment group by Me
ss_ID) a);
ERROR 1052 (23000): Column 'Mess_ID' in having clause is ambiguous
mysql> select sa.Mess_ID,m.Mname, Count(sa.Mess_ID) as Max_Student_Count from S
tudent_Allotment sa join Mess m on (sa.Mess_ID = m.Mess_ID) group by sa.Mess_ID
having Count(sa.Mess_ID) = (select max(a.mbs
a) from (select Mess_ID, Count(Mess_ID) as mbsa from Student_Allotment group by
Mess_ID) a);
+-----+-----+-----+
| Mess_ID | Mname      | Max_Student_Count |
+-----+-----+-----+
| 3       | Arawali    | 2                 |
| 4       | Karawali   | 2                 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

III. Delete the Employee whose salary is more than 20,000 and belong to a particular given hostel (say block 2)

Query: delete from Employee where E_ID in (select h.E_ID from Hostel_Employee h where HID = 2) and E_salary >20000;

```
mysql> delete from Employee where E_ID in (select h.E_ID from Hostel_Employee h where HID = 2) and E_salary >20000;
Query OK, 1 row affected (0.01 sec)

mysql> select * from Employee ;
+-----+-----+-----+
| E_Name | E_ID | E_salary |
+-----+-----+-----+
| Sai    | 1231 | 10000    |
| Shyam  | 1232 | 20000    |
| Adarsh | 1233 | 15000    |
| Raman  | 1234 | 18000    |
| Rohit  | 1235 | 40000    |
| Dhoni  | 1236 | 30000    |
| Onkar  | 1237 | 25000    |
| Chutney| 1239 | 20000    |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

IV. Find the Mess name of student with a given name (say Avinash Rauniyar)

Query: select m.Mname from Mess m join Student_Allotment sa on (sa.Mess_ID = m.Mess_ID) join Students s on (sa.SID = s.SID and s.S_init_name = "Avinash" and s.S_last_name = "Rauniyar");

```
mysql> select m.Mname from Mess m join Student_Allotment sa on (sa.Mess_ID = m.Mess_ID) join Students s on (sa.SID = s.SID and s.S_init_name = "Avinash" and s.S_last_name = "Rauniyar");
+-----+
| Mname |
+-----+
| Karawali |
+-----+
1 row in set (0.06 sec)
```

V. Find the count of triple shared rooms in the hostel 'Sahydri'

Query:

select count(*) from Room r join Hostel h on(h.HID = r.HID and h.Hname = "Sahydari");

```
mysql> select count(*) from Room r join Hostel h on(h.HID = r.HID and h.Hname = "Sahydari");
-> //
+-----+
| count(*) |
+-----+
|      3 |
+-----+
1 row in set (0.00 sec)
```


VIEWS:

- I. Query to create a view to see which student has been allotted which room.

Query: create view Room_Allotment as select
sa.SID,s.S_init_name,s.S_last_name,sa.RID from Student_Allotment sa join
Students s on (sa.SID = s.SID);

```
mysql> create view Room_Allotment as select sa.SID,s.S_init_name,s.S_last_name,sa.RID from Student_Allotment sa join Students s on (sa.SID = s.SID);
Query OK, 0 rows affected (0.08 sec)

mysql> select * from Room_Allotment;
+-----+-----+-----+-----+
| SID | S_init_name | S_last_name | RID |
+-----+-----+-----+-----+
| 1 | Spandan | Patkar | A-001 |
| 5 | Avinash | Taparia | A-401 |
| 8 | Avinash | Rauniyar | A-403 |
| 2 | Rushit | Kausal | B-101 |
| 6 | Ayush | Taparia | B-501 |
| 3 | Jatin | Majumdar | C-201 |
| 7 | Jatin | Taparia | C-601 |
| 4 | Rohit | Bhojwani | D-301 |
+-----+-----+-----+-----+
8 rows in set (0.03 sec)
```

- II. Query to create a view to see which student has been allotted which mess.

Query: create view Mess_Allotment as select
sa.SID,s.S_init_name,s.S_last_name,sa.Mess_ID from Student_Allotment sa
join Students s on (sa.SID = s.SID);

```
mysql> create view Mess_Allotment as select sa.SID,s.S_init_name,s.S_last_name,sa.Mess_ID from Student_Allotment sa join Students s on (sa.SID = s.SID);
Query OK, 0 rows affected (0.03 sec)

mysql> select * from Mess_Allotment;
+-----+-----+-----+-----+
| SID | S_init_name | S_last_name | Mess_ID |
+-----+-----+-----+-----+
| 1 | Spandan | Patkar | 1 |
| 2 | Rushit | Kausal | 2 |
| 3 | Jatin | Majumdar | 3 |
| 7 | Jatin | Taparia | 3 |
| 4 | Rohit | Bhojwani | 4 |
| 8 | Avinash | Rauniyar | 4 |
| 5 | Avinash | Taparia | 5 |
| 6 | Ayush | Taparia | 6 |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

III. Query to create a view to show the name of the mess inside the hostel.

Query: create view Mess_Allotment as select sa.SID,s.S_init_name,s.S_last_name,sa.Mess_ID from Student_Allotment sa join Students s on (sa.SID = s.SID);

```
mysql> create view Mess_Hostel as select m.Mname,h.Hname from Mess m join Hostel h on (h.Mess_ID = m.Mess_ID);
Query OK, 0 rows affected (0.04 sec)

mysql> select * from Mess_Hostel;
+-----+-----+
| Mname | Hname |
+-----+-----+
| Vindhya | Vindhya |
| Satpura | Satpura |
| Arawali | Arawali |
| Karawali | Karawali |
| Sahydari | Sahydari |
| MegaMess | Himalya |
| MegaMess | Trishul |
| MegaMess | Everest |
+-----+-----+
8 rows in set (0.00 sec)
```

IV. Query to Create views for showing all the hostel employees

Query: create view Hostel_Emp as select h.HID,e.E_Name from Hostel_Employee h join Employee e on (e.E_ID = h.E_ID);

```
mysql> create view Hostel_Emp as select h.HID,e.E_Name from Hostel_Employee h join
Employee e on (e.E_ID = h.E_ID);
Query OK, 0 rows affected (0.41 sec)

mysql> select * from Hostel_Emp;
+-----+-----+
| HID | E_Name |
+-----+-----+
| 1 | Sai |
| 2 | Samosa |
| 3 | Onkar |
| 4 | Adarsh |
| 5 | Raman |
| 6 | Onkar |
| 7 | Shyam |
| 8 | Dhoni |
| 1 | Chutney |
+-----+-----+
9 rows in set (0.00 sec)
```

V. Query to Create views for showing all the Mess employees

Query: create view Mess_Emp as select m.Mess_ID,e.E_Name from Mess_Employee m join Employee e on (e.E_ID = m.E_ID);

```
mysql> create view Mess_Emp as select m.Mess_ID,e.E_Name from Mess_Employee m join
Employee e on (e.E_ID = m.E_ID);
Query OK, 0 rows affected (0.01 sec)

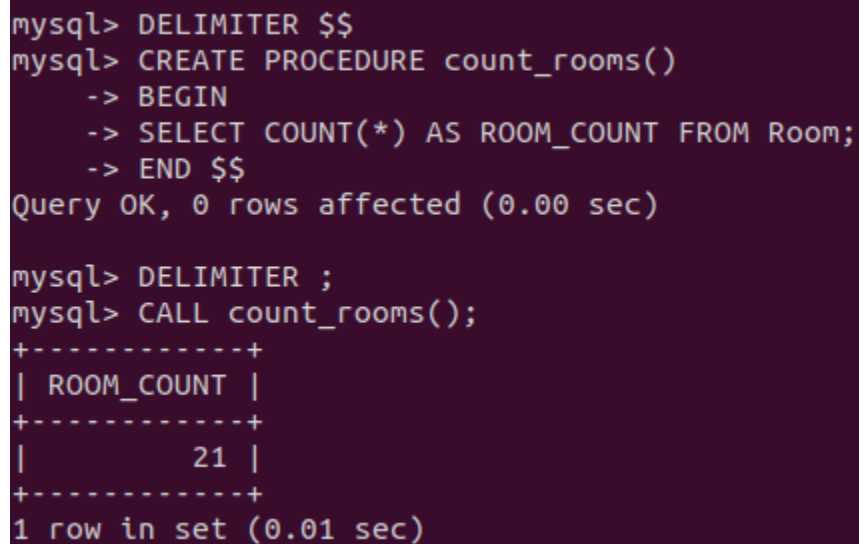
mysql> select * from Mess_Emp;
+-----+-----+
| Mess_ID | E_Name |
+-----+-----+
| 1 | Sai |
| 2 | Shyam |
| 3 | Adarsh |
| 4 | Raman |
| 5 | ROhit |
| 6 | Dhoni |
| 2 | Onkar |
| 3 | Samosa |
+-----+-----+
8 rows in set (0.00 sec)
```


Procedures :

I. Query to count the total number of rooms in all the hostels:-

Query:

```
DELIMITER $$
CREATE PROCEDURE count_rooms()
BEGIN
SELECT COUNT(*) AS ROOM_COUNT FROM Room;
END $$
DELIMITER ;
CALL count_rooms();
```



```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE count_rooms()
-> BEGIN
-> SELECT COUNT(*) AS ROOM_COUNT FROM Room;
-> END $$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> CALL count_rooms();
+-----+
| ROOM_COUNT |
+-----+
|          21 |
+-----+
1 row in set (0.01 sec)
```

II. Query to count the average salary of employees.

Query:

```
DELIMITER $$
CREATE PROCEDURE avg_salary()
BEGIN
SELECT AVG(E_salary) as Average_Salary from `Employee`; END $$
DELIMITER ;
Call avg_salary();
```

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE avg_salary()
    -> BEGIN
    -> SELECT AVG(E_salary) as Average_Salary from `Employee`; END $$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> Call avg_salary();
+-----+
| Average_Salary |
+-----+
|      28477.7778 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

III. Query to get the details of the student along with their hostel and room number.

Query:

```
DELIMITER $$
CREATE PROCEDURE stud_room_details()
BEGIN
SELECT sa.SID,s.S_init_name,s.S_last_name ,h.Hname,r.RID from Student_Allotm
ent sa join Students s on (sa.SID = s.SID) join Hostel h on (sa.HID = h.HID) join R
oom r on (sa.RID = r.RID);
END $$
```

```
mysql> SELECT sa.SID,s.S_init_name,s.S_last_name ,h.Hname,r.RID from Student_Allotm
ent sa join Students s on (sa.SID = s.SID) join Hostel h on (sa.HID = h.HID) join R
oom r on (sa.RID = r.RID);
+-----+
| SID | S_init_name | S_last_name | Hname   | RID   |
+-----+
| 1 | Spandan    | Patkar      | Vindhya | A-001 |
| 2 | Rushit     | Kausal      | Satpura | B-101 |
| 3 | Jatin      | Majumdar    | Arawali | C-201 |
| 4 | Rohit      | Bhojwani    | Karawali | D-301 |
| 5 | Avinash    | Taparia     | Sahydari | A-401 |
| 6 | Ayush      | Taparia     | Himalya | B-501 |
| 7 | Jatin      | Taparia     | Trishul | C-601 |
| 8 | Avinash    | Rauniyar    | Everest | A-403 |
+-----+
8 rows in set (0.00 sec)
```

IV. Query to get the number of students in a particular mess using Mess ID.

Query:

```
DELIMITER $$
CREATE PROCEDURE stud_count_mess(IN MSID int)
BEGIN
SELECT COUNT(*) FROM Student_Allotment WHERE Mess_ID = MSID;
END $$
```

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE stud_count_mess(IN MSID int)
-> BEGIN
-> SELECT COUNT(*) FROM Student_Allotment WHERE Mess_ID = MSID;
-> END$$
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> call stud_count_mess(4);
+-----+
| COUNT(*) |
+-----+
|          2 |
+-----+
1 row in set (0.00 sec)
```

V. Query to get the number of students in a particular mess using Hostel ID.

Query:

```
DELIMITER $$
CREATE PROCEDURE stud_count_hostel(IN HSID int)
BEGIN
SELECT COUNT(*) FROM Student_Allotment WHERE HID = HSID;
END $$
```

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE stud_count_hostel(IN HSID int)
-> BEGIN
-> SELECT COUNT(*) FROM Student_Allotment WHERE HID = HSID;
-> END $$
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> call stud_count_hostel(5);
+-----+
| COUNT(*) |
+-----+
|          1 |
+-----+
1 row in set (0.00 sec)
```

Triggers:

- I. **Create a trigger which will add employees to a new table named 'Trusted Employees' if they receive a salary increment (provided by the user)**

Query:

```
CREATE TABLE Trusted_Employee( name varchar(30),E_ID int);
DELIMITER //
CREATE TRIGGER UpdateTrustedEmployee AFTER UPDATE
ON Employee FOR EACH ROW
BEGIN
DECLARE rem INT;
SET rem = NEW.E_salary - OLD.E_salary;
IF rem>0 THEN
INSERT INTO Trusted_Employee VALUES(OLD.E_Name,OLD.E_ID); END IF;
END//
DELIMITER;
```

```
UPDATE Employee set E_salary = E_salary*1.1;
```



```
mysql> update Employee set E_salary = E_salary*1.1;
-> //
Query OK, 9 rows affected (0.01 sec)
Rows matched: 9  Changed: 9  Warnings: 0

mysql> select * from Trusted_Employee;
-> //
+-----+-----+
| E_Name | E_ID |
+-----+-----+
| Sai    | 1231 |
| Shyam  | 1232 |
| Adarsh | 1233 |
| Raman  | 1234 |
| Rohit  | 1235 |
| Dhoni  | 1236 |
| Onkar  | 1237 |
| Samosa | 1238 |
| Chutney | 1239 |
+-----+-----+
9 rows in set (0.00 sec)
```

- II. Create a table which will add the members removed from the table 'Students_Allotment' to a new table named 'alumini'

```
mysql> select * from Student_Allotment;
+-----+-----+-----+-----+-----+-----+
| SID | Mess_ID | HID | RID | M_exp_date | H_exp_date |
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | A-001 | 2022-05-17 | 2022-05-17 |
| 2 | 2 | 2 | B-101 | 2022-05-17 | 2022-05-17 |
| 3 | 3 | 3 | C-201 | 2022-05-17 | 2022-05-17 |
| 4 | 4 | 4 | D-301 | 2022-05-17 | 2022-05-17 |
| 5 | 5 | 5 | A-401 | 2022-05-17 | 2022-05-17 |
| 6 | 6 | 6 | B-501 | 2022-05-17 | 2022-05-17 |
| 7 | 3 | 7 | C-601 | 2022-05-17 | 2022-05-17 |
| 8 | 4 | 8 | A-403 | 2022-05-17 | 2022-05-17 |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
mysql> delete from Student_Allotment where SID=1;
Query OK, 1 row affected (0.01 sec)
```

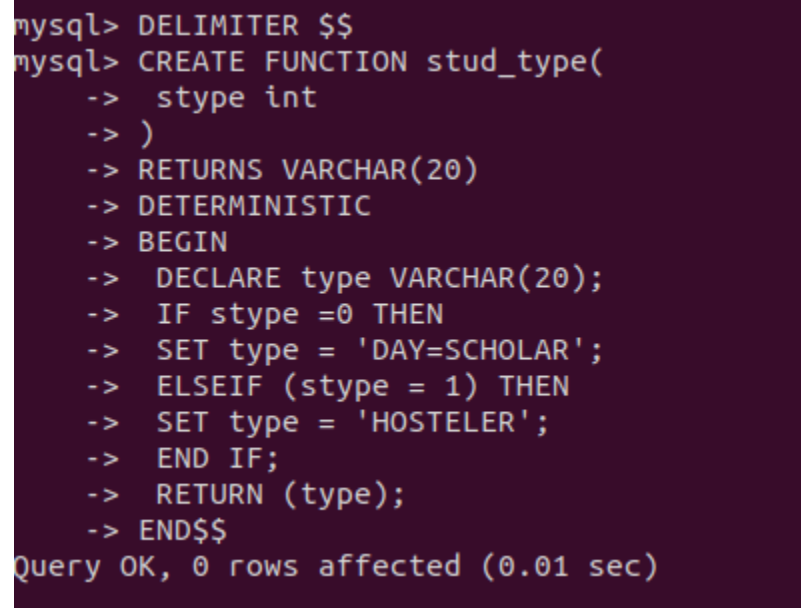
```
mysql> select * from Alumini;
+-----+-----+-----+-----+
| SID | Mess_ID | HID | RID |
+-----+-----+-----+-----+
| 1 | 1 | 1 | A-001 |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Functions:

- I. Create a function to get the Student Type (Day-Scholars or Hosteler) using the appropriate attribute of table Students.

Query:-

```
DELIMITER $$
CREATE FUNCTION stud_type(
  stype int
)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
  DECLARE type VARCHAR(20);
  IF stype =0 THEN
    SET type = 'DAY=SCHOLAR';
  ELSEIF (stype = 1) THEN
    SET type = 'HOSTELER';
  END IF;
  RETURN (type);
END$$
DELIMITER ;
```



```
mysql> DELIMITER $$
mysql> CREATE FUNCTION stud_type(
->   stype int
-> )
-> RETURNS VARCHAR(20)
-> DETERMINISTIC
-> BEGIN
->   DECLARE type VARCHAR(20);
->   IF stype =0 THEN
->     SET type = 'DAY=SCHOLAR';
->   ELSEIF (stype = 1) THEN
->     SET type = 'HOSTELER';
->   END IF;
->   RETURN (type);
-> END$$
Query OK, 0 rows affected (0.01 sec)
```

Testing:-

```
mysql> select SID,S_init_name,stype,stud_type(stype) from Students;
+-----+-----+-----+-----+
| SID | S_init_name | stype | stud_type(stype) |
+-----+-----+-----+-----+
| 1 | Spandan | 0 | DAY=SCHOLAR |
| 2 | Rushit | 1 | HOSTELER |
| 3 | Jatin | 0 | DAY=SCHOLAR |
| 4 | Rohit | 0 | DAY=SCHOLAR |
| 5 | Avinash | 1 | HOSTELER |
| 6 | Ayush | 0 | DAY=SCHOLAR |
| 7 | Jatin | 1 | HOSTELER |
| 8 | Avinash | 1 | HOSTELER |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

- II. Create a function to classify Salary of Employee as low(<=15000),Intermediate(>15000 & <=30000) & High(>30000):-

Query:-

```
DELIMITER $$
CREATE FUNCTION salary_classification(
    salary int
)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE remark VARCHAR(20);
    IF salary<=15000 THEN
        SET remark = 'LOW';
    ELSEIF (salary>15000 AND salary <=30000) THEN
        SET remark = 'INTERMEDIATE';
    ELSEIF (salary >30000) THEN
        SET remark = 'HIGH';
    END IF;
    RETURN (remark);
END$$
DELIMITER ;
```

```
mysql> DELIMITER $$
mysql> CREATE FUNCTION salary_classification(
-> salary int
-> )
-> RETURNS VARCHAR(20)
-> DETERMINISTIC
-> BEGIN
-> DECLARE remark VARCHAR(20);
-> IF salary<=15000 THEN
-> SET remark = 'low';
-> ELSEIF (salary>15000 AND salary <=30000) THEN
-> SET remark = 'intermediate';
-> ELSEIF (salary >30000) THEN
-> SET remark = 'high';
-> END IF;
-> RETURN (remark);
-> END$$
Query OK, 0 rows affected (0.01 sec)
```

Testing:-

```
mysql> select E_salary,salary_classification(E_salary) from Employee;
+-----+-----+
| E_salary | salary_classification(E_salary) |
+-----+-----+
| 11000 | low |
| 33000 | high |
| 16500 | intermediate |
| 19800 | intermediate |
| 44000 | high |
| 33000 | high |
| 27500 | intermediate |
| 49500 | high |
| 22000 | intermediate |
+-----+-----+
```

III. Create a function to rate a particular mess out of 10 using the Mess_ID:-

Query:-

```
DELIMITER $$
CREATE FUNCTION mess_rating(
meid int
)
```

```
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE rating INT;
IF meid = 1 THEN
SET rating = 5;
ELSEIF (meid = 2) THEN
SET rating = 6;
ELSEIF (meid = 3) THEN
SET rating = 3;
ELSEIF (meid = 4) THEN
SET rating = 8;
ELSEIF (meid = 5) THEN
SET rating = 9;
ELSEIF (meid = 6) THEN
SET rating = 1;
END IF;
RETURN (rating);
END$$
DELIMITER ;
```

```
mysql> DELIMITER $$
mysql> CREATE FUNCTION mess_rating(
-> meid int
-> )
-> RETURNS INT
-> DETERMINISTIC
-> BEGIN
-> DECLARE rating INT;
-> IF meid = 1 THEN
-> SET rating = 5;
-> ELSEIF (meid = 2) THEN
-> SET rating = 6;
-> ELSEIF (meid = 3) THEN
-> SET rating = 3;
-> ELSEIF (meid = 4) THEN
-> SET rating = 8;
-> ELSEIF (meid = 5) THEN
-> SET rating = 9;
-> ELSEIF (meid = 6) THEN
-> SET rating = 1;
-> END IF;
-> RETURN (rating);
-> END$$
Query OK, 0 rows affected (0.00 sec)
```

Testing:-

```
mysql> select Mess_ID,Mname,mess_rating(Mess_ID) from Mess;
```

```
+-----+-----+-----+
| Mess_ID | Mname   | mess_rating(Mess_ID) |
+-----+-----+-----+
|      1  | Vindhya |          5           |
|      2  | Satpura |          6           |
|      3  | Arawali |          3           |
|      4  | Karawali|          8           |
|      5  | Sahydari|          9           |
|      6  | MegaMess|          1           |
+-----+-----+-----+
6 rows in set (0.00 sec)
```