H

**Teja Sri Paluri**

**G18 PowerBI and SQL batch**

HOSTEL ACCOUNTING

USING MY SQL

Sure-trust

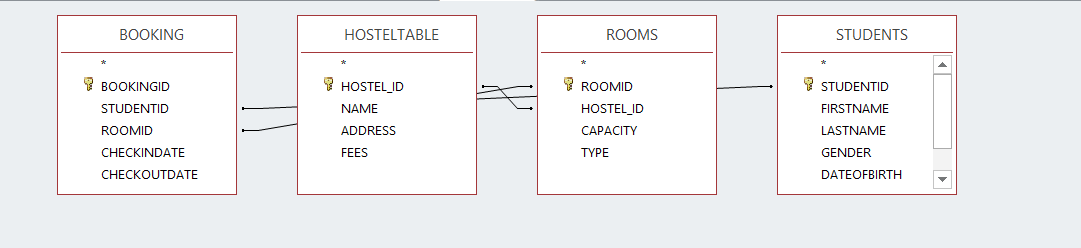
HOSTEL ACCOUNTING

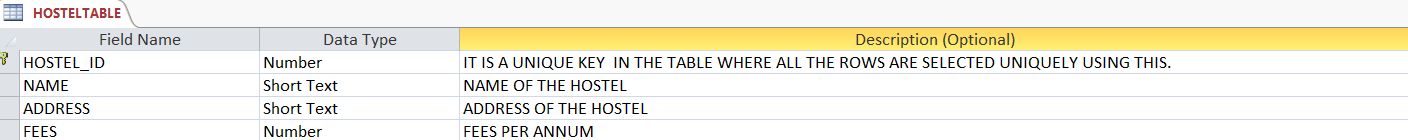
Designing a schema for Hostel Management involves defining the database structure and relationships between different entities. Below is a comprehensive schema for such a system focusing on key components and their interrelationships.

Features:

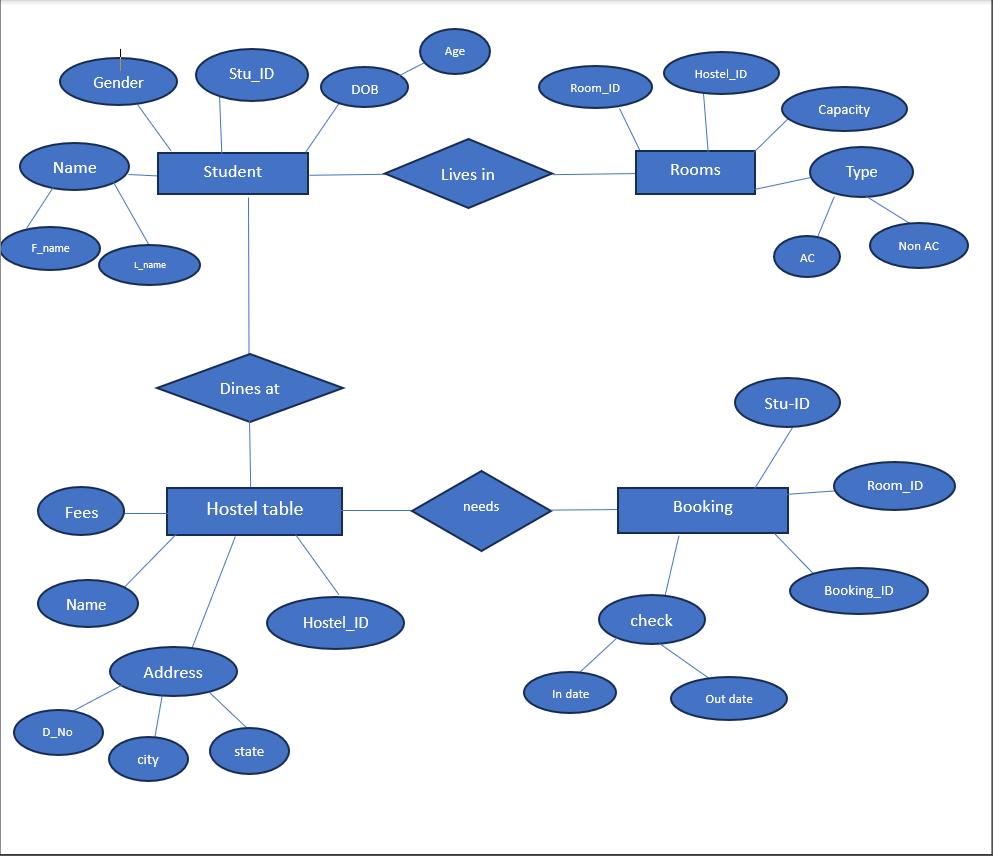
* Hostel: Information about Hostel Name, Address and their Fees.
* Rooms: Information about Capacity and Type of Rooms.
* Students: Information about Students First Name, Last Name, Gender, DOB, Phone number and Email.
* Booking: Information about Booking ID, Room ID, Student ID, Check-in Date and Check-out Date

Defining schemas for Application:





ER Diagram



Creating a Database using DDL and DML Commands:

CREATE TABLE HOSTEL (

HOSTEL\_ID INT PRIMARY KEY,

NAME CHAR(50),

ADDRESS CHAR(50),

FEES INT

);

INSERT INTO HOSTEL VALUES (1, 'DWARAKAMAI HOSTEL', 'KOMMADI, MADURAWADA', 60000);

INSERT INTO HOSTEL VALUES (2, 'ORANGE HOSTEL', 'KOMMADI, MADURAWADA', 70000);

INSERT INTO HOSTEL VALUES (3, 'SEA BREEZE HOSTEL', 'BEACH ROAD, VISAKHAPATNAM', 40000);

INSERT INTO HOSTEL VALUES (4, 'SUNRISE HOSTEL', 'DABAGARDENS, VISAKHAPATNAM', 50000);

INSERT INTO HOSTEL VALUES (5, 'Hilltop Hostel', 'Kailasagiri, Visakhapatnam', 45000);

INSERT INTO HOSTEL VALUES (6, 'City Center Hostel', 'Dwaraka Nagar, Visakhapatnam', 60000);

INSERT INTO HOSTEL VALUES (7, 'Lighthouse Hostel', 'Yarada Beach, Visakhapatnam', 50000);

INSERT INTO HOSTEL VALUES (8, 'Forest Edge Hostel', 'Araku Valley, Visakhapatnam', 80000);

SELECT \* FROM HOSTEL;

Output:

+-----------+--------------------+------------------------------+-------+

| HOSTEL\_ID | NAME | ADDRESS | FEES |

+-----------+--------------------+------------------------------+-------+

| 1 | DWARAKAMAI HOSTEL | KOMMADI, MADURAWADA | 60000 |

| 2 | ORANGE HOSTEL | KOMMADI, MADURAWADA | 70000 |

| 3 | SEA BREEZE HOSTEL | BEACH ROAD, VISAKHAPATNAM | 40000 |

| 4 | SUNRISE HOSTEL | DABAGARDENS, VISAKHAPATNAM | 50000 |

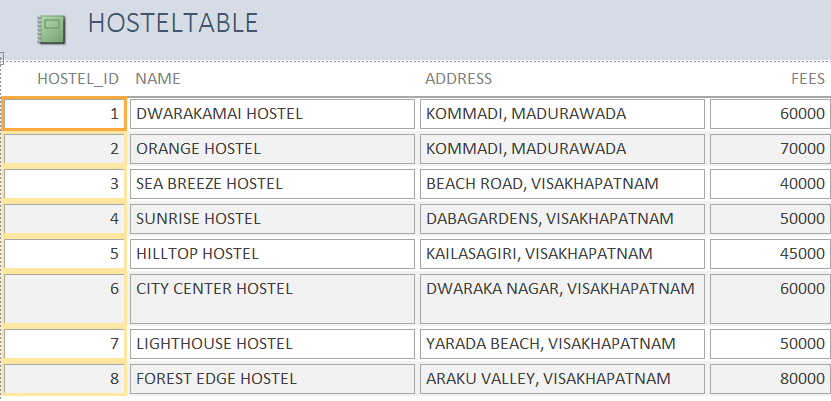
| 5 | Hilltop Hostel | Kailasagiri, Visakhapatnam | 45000 |

| 6 | City Center Hostel | Dwaraka Nagar, Visakhapatnam | 60000 |

| 7 | Lighthouse Hostel | Yarada Beach, Visakhapatnam | 50000 |

| 8 | Forest Edge Hostel | Araku Valley, Visakhapatnam | 80000 |

+-----------+--------------------+------------------------------+-------+



CREATE TABLE ROOMS (

ROOMID INT NOT NULL,

HOSTEL\_ID INT,

CAPACITY INT,

TYPE CHAR(20),

FOREIGN KEY (HOSTEL\_ID) REFERENCES HOSTEL(HOSTEL\_ID)

);

INSERT INTO ROOMS VALUES (1, 1, 4, 'Standard');

INSERT INTO ROOMS VALUES (2, 1, 4, 'Deluxe');

INSERT INTO ROOMS VALUES (4, 2, 3, 'Standard');

INSERT INTO ROOMS VALUES (5, 2, 3, 'Standard');

INSERT INTO ROOMS VALUES (6, 3, 2, 'Standard');

INSERT INTO ROOMS VALUES (7, 4, 2, 'Standard');

INSERT INTO ROOMS VALUES (8, 7, 3, 'Deluxe');

INSERT INTO ROOMS VALUES (9, 5, 1, 'Single');

SELECT \* FROM ROOMS;

+--------+-----------+----------+----------+

| ROOMID | HOSTEL\_ID | CAPACITY | TYPE |

+--------+-----------+----------+----------+

| 1 | 1 | 4 | Standard |

| 2 | 1 | 4 | Deluxe |

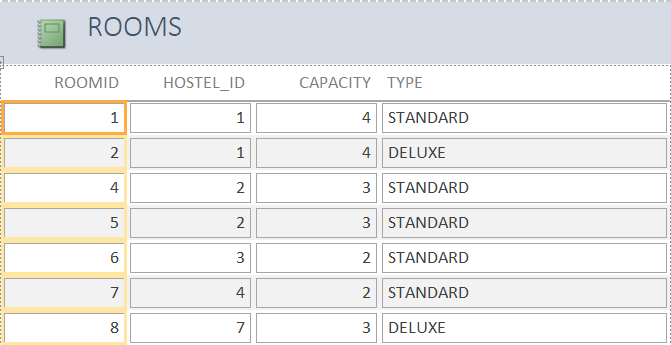
| 4 | 2 | 3 | Standard |

| 5 | 2 | 3 | Standard |

| 6 | 3 | 2 | Standard |

| 7 | 4 | 2 | Standard |

| 8 | 7 | 3 | Deluxe |



CREATE TABLE STUDENTS (

STUDENTID INT PRIMARY KEY,

FIRSTNAME CHAR(20),

LASTNAME CHAR(20),

GENDER CHAR(30) CHECK (GENDER IN ('male', 'female')),

DATEOFBIRTH CHAR(30),

PHONENUMBER BIGINT,

EMAIL CHAR(50)

);

INSERT INTO STUDENTS VALUES (1, 'neeraj', 'chopra', 'male', '30-10-2004', 8900234567, 'neerajchopra@gmail.com');

INSERT INTO STUDENTS VALUES (2, 'sanjeev', 'kumar', 'male', '21-04-2001', 7902345671, 'sanjeev@gmail.com');

INSERT INTO STUDENTS VALUES (3, 'nalini', 'padmanaban', 'female', '22-04-2003', 7902345311, 'nalini@gmail.com');

INSERT INTO STUDENTS VALUES (4, 'saisri', 'chigilipalli', 'female', '20-06-2004', 8932285927, 'saisri@gmail.com');

INSERT INTO STUDENTS VALUES (5, 'goutham', 'nanda', 'male', '20-06-2004', 8930085927, 'nanda@gmail.com');

INSERT INTO STUDENTS VALUES (6, 'sowjanya', 'kintali', 'female', '01-02-2004', 8930085927, 'sowji@gmail.com');

INSERT INTO STUDENTS VALUES (7, 'meenakshi', 'desai', 'female', '01-02-2003', 9032245671, 'meena@gmail.com');

INSERT INTO STUDENTS VALUES (8, 'arjun', 'suravar', 'male', '02-11-2000', 9823456711, 'arjun@gmail.com');

SELECT \* FROM STUDENTS;

Output:

+-----------+-----------+--------------+--------+-------------+--------

| STUDENTID | FIRSTNAME | LASTNAME | GENDER | DATEOFBIRTH PHONENUMBER | EMAIL

+-----------+-----------+--------------+--------+-------------+--------

| 1 | neeraj | chopra | male | 30-10-2004 | 8900234567 | neerajchopra@gmail.com |

| 2 | sanjeev | kumar | male | 21-04-2001 | 7902345671 | sanjeev@gmail.com |

| 3 | nalini | padmanaban | female | 22-04-2003 | 7902345311 | nalini@gmail.com |

| 4 | saisri | chigilipalli | female | 20-06-2004 | 8932285927 | saisri@gmail.com |

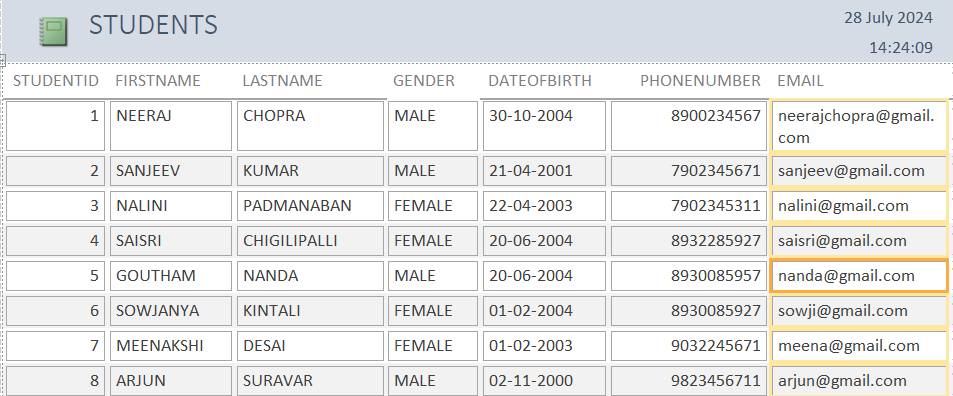
| 5 | goutham | nanda | male | 20-06-2004 | 8930085927 | nanda@gmail.com |

| 6 | sowjanya | kintali | female | 01-02-2004 | 8930085927 | sowji@gmail.com |

| 7 | meenakshi | desai | female | 01-02-2003 | 9032245671 | meena@gmail.com |

| 8 | arjun | suravar | male | 02-11-2000 | 9823456711 | arjun@gmail.com |

+-----------+-----------+--------------+--------+-------------+--------



CREATE TABLE BOOKING (

BOOKINGID INT,

STUDENTID INT REFERENCES STUDENTS(STUDENTID),

ROOMID INT REFERENCES ROOMS(ROOMID),

CHECKINDATE CHAR(30),

CHECKOUTDATE CHAR(30)

);

INSERT INTO BOOKING VALUES (1, 1, 1, '2024-06-01', '2024-06-30');

INSERT INTO BOOKING VALUES (2, 2, 2, '2024-06-05', '2024-06-25');

INSERT INTO BOOKING VALUES (3, 3, 3, '2024-06-10', '2024-06-20');

INSERT INTO BOOKING VALUES (4, 4, 4, '2024-06-15', '2024-06-22');

INSERT INTO BOOKING VALUES (5, 6, 5, '2024-07-01', '2024-07-10');

INSERT INTO BOOKING VALUES (6, 8, 6, '2024-07-01', '2024-07-10');

SELECT \* FROM BOOKING;

+-----------+-----------+--------+-------------+--------------+

| BOOKINGID | STUDENTID | ROOMID | CHECKINDATE | CHECKOUTDATE |

+-----------+-----------+--------+-------------+--------------+

| 1 | 1 | 1 | 2024-06-01 | 2024-06-30 |

| 2 | 2 | 2 | 2024-06-05 | 2024-06-25 |

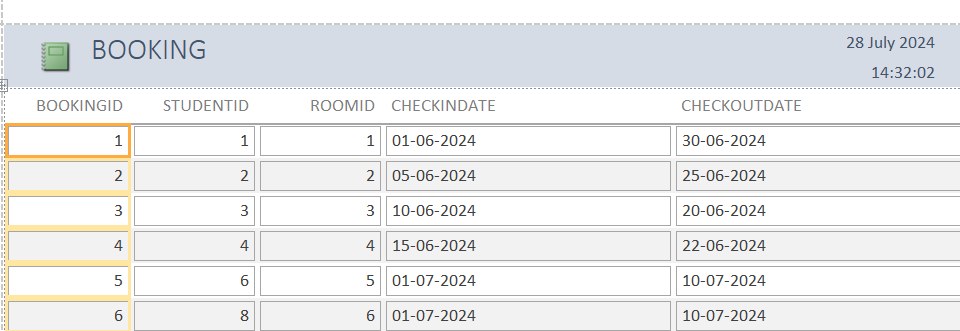
| 3 | 3 | 3 | 2024-06-10 | 2024-06-20 |

| 4 | 4 | 4 | 2024-06-15 | 2024-06-22 |

| 5 | 6 | 5 | 2024-07-01 | 2024-07-10 |

| 6 | 8 | 6 | 2024-07-01 | 2024-07-10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |



Constraints :

Unique constraint:

null values can be inserted in unique constraint.but It doesnot allows duplicate values

SQL> alter table booking add constraint w unique(bookingid);

Table altered.

SQL> insert into booking values(4, 4, 4, '2024-06-15', '2024-06-22');

insert into booking values(4, 4, 4, '2024-06-15', '2024-06-22')

\*

ERROR at line 1:

ORA-00001: unique constraint (SYSTEM.W) violated

SQL> insert into booking values(null, 4, 4, '2024-06-15', '2024-06-22');

1 row created.

SQL> select \* from booking;

BOOKINGID STUDENTID ROOMID CHECKINDATE CHECKOUTDATE

---------- ---------- ---------- ------------------------------ ------------------------------

1 1 1 2024-06-01 2024-06-30

2 2 2 2024-06-05 2024-06-25

3 3 3 2024-06-10 2024-06-20

4 4 4 2024-06-15 2024-06-22

4 4 2024-06-15 2024-06-22

Operations(enable,disable,drop):

SQL> alter table booking disable constraint w;

Table altered.

SQL> insert into booking values(4, 4, 4, '2024-06-15', '2024-06-22');

1 row created.

SQL> select \* from booking;

BOOKINGID STUDENTID ROOMID CHECKINDATE CHECKOUTDATE

---------- ---------- ---------- ------------------------------ ------------------------------

1 1 1 2024-06-01 2024-06-30

2 2 2 2024-06-05 2024-06-25

3 3 3 2024-06-10 2024-06-20

4 4 4 2024-06-15 2024-06-22

4 4 2024-06-15 2024-06-22

4 4 4 2024-06-15 2024-06-22

6 rows selected.

SQL> alter table booking enable constraint w;

alter table booking enable constraint w

\*

ERROR at line 1:

ORA-02299: cannot validate (SYSTEM.W) - duplicate keys found

SQL> delete from booking where bookingid=4;

2 rows deleted.

SQL> alter table booking enable constraint w;

Table altered.

SQL> insert into booking values(4, 4, 4, '2024-06-15', '2024-06-22');

1 row created.

SQL> insert into booking values(4, 3, 2, '2024-06-15', '2024-06-22');

insert into booking values(4, 3, 2, '2024-06-15', '2024-06-22')

\*

ERROR at line 1:

ORA-00001: unique constraint (SYSTEM.W) violated

SQL> alter table booking drop constraint w;

Table altered.

Primary key constraint :

It cannot accept duplicate values as well as null values

SQL> alter table booking add primary key(bookingid);

Table altered.

SQL> insert into booking values(4, 3, 2, '2024-06-15', '2024-06-22');

1 row created.

SQL> insert into booking values(4, 3, 2, '2024-06-15', '2024-06-22');

insert into booking values(4, 3, 2, '2024-06-15', '2024-06-22')

\*

ERROR at line 1:

ORA-00001: unique constraint (SYSTEM.SYS\_C004111) violated

SQL> insert into booking values(null, 3, 2, '2024-06-15', '2024-06-22');

insert into booking values(null, 3, 2, '2024-06-15', '2024-06-22')

\*

ERROR at line 1:

ORA-01400: cannot insert NULL into ("SYSTEM"."BOOKING"."BOOKINGID")

Operations :

SQL> alter table booking disable constraint k;

Table altered.

SQL> insert into booking values(null, 3, 2, '2024-06-15', '2024-06-22');

1 row created.

SQL> select \* from booking;

BOOKINGID STUDENTID ROOMID CHECKINDATE CHECKOUTDATE

---------- ---------- ---------- ------------------------------ ------------------------------

1 1 1 2024-06-01 2024-06-30

2 2 2 2024-06-05 2024-06-25

3 3 3 2024-06-10 2024-06-20

4 3 2 2024-06-15 2024-06-22

3 2 2024-06-15 2024-06-22

SQL> alter table booking enable constraint k;

Table altered.

SQL> insert into booking values(null, 3, 2, '2024-06-15', '2024-06-22');

insert into booking values(null, 3, 2, '2024-06-15', '2024-06-22')

\*

ERROR at line 1:

ORA-01400: cannot insert NULL into ("SYSTEM"."BOOKING"."BOOKINGID")

SQL> insert into booking values(null, 3, 2, '2024-06-15', '2024-06-22');

insert into booking values(3, 3, 2, '2024-06-15', '2024-06-22')

SQL> insert into booking values(4, 3, 2, '2024-06-15', '2024-06-22');

insert into booking values(4, 3, 2, '2024-06-15', '2024-06-22')

\*

ERROR at line 1:

ORA-00001: unique constraint (SYSTEM.SYS\_C004111) violated

SQL> alter table booking drop constraint k;

Table altered.

Foreign key:

SQL> select \* from hostel;

//parent table hostel\_id primary key

HOSTEL\_ID NAME ADDRESS

---------- ------------------------------ ------------------------------

1 dwarakamai Hostel kommali, madhuravada

2 orange Hostel kommali, madhuravada

3 Sea Breeze Hostel Beach Road, Visakhapatnam

4 Sunrise Hostel Dabagardens, Visakhapatnam

5 Hilltop Hostel Kailasagiri, Visakhapatnam

6 City Center Hostel Dwaraka Nagar, Visakhapatnam

7 Lighthouse Hostel Yarada Beach, Visakhapatnam

8 Forest Edge Hostel Araku Valley, Visakhapatnam

8 rows selected.

SQL> select \* from rooms;

//child table hostel id foreign key references hostel\_id from hostel table

ROOMID HOSTELID CAPACITY TYPE

---------- ---------- ---------- --------------------

1 1 4 Standard

2 1 4 deluxe

3 2 4 Suite

4 2 3 standard

5 2 3 standard

6 3 2 Standard

9 5 1 Single

7 4 2 Standard

8 7 3 Deluxe

10 4 3 Deluxe

11 4 3 single

11 rows selected.

SQL> insert into rooms values(12,9,5,'suite');

insert into rooms values(12,9,5,'suite')

\*

ERROR at line 1:

ORA-02291: integrity constraint (SYSTEM.SYS\_C004101) violated - parent key not

found

//we cannot delete id from parent table without deleting in child table

SQL> delete from hostel where hostel\_id=4;

delete from hostel where hostel\_id=4

\*

ERROR at line 1:

ORA-02292: integrity constraint (SYSTEM.SYS\_C004101) violated - child record

Found

SQL> select \* from rooms;

ROOMID HOSTELID CAPACITY TYPE

---------- ---------- ---------- -----------------------------

1 1 4 Standard

2 1 4 deluxe

3 2 4 Suite

4 2 3 standard

5 2 3 standard

6 3 2 Standard

9 5 1 Single

7 4 2 Standard

10 4 3 Deluxe

11 4 3 single

10 rows selected.

SQL> select \* from hostel;

HOSTEL\_ID NAME ADDRESS

---------- ------------------------------ ------------------------------

1 dwarakamai Hostel kommadi, madhuravada

2 orange Hostel kommadi, madhuravada

3 Sea Breeze Hostel Beach Road, Visakhapatnam

4 Sunrise Hostel Dabagardens, Visakhapatnam

5 Hilltop Hostel Kailasagiri, Visakhapatnam

6 City Center Hostel Dwaraka Nagar, Visakhapatnam

7 Lighthouse Hostel Yarada Beach, Visakhapatnam

8 Forest Edge Hostel Araku Valley, Visakhapatnam

8 rows selected.

//changes in child table doesnot reflect in the parent table

Aggregate Functions :

AGGREGATE FUNCTIONS:

SQL> select max(TYPE) from Rooms;

MAX(TYPE)

--------------------

standard

SQL>select min(TYPE) from Rooms;

MIN(TYPE)

--------------------

Deluxe

SQL>select count(TYPE) from Rooms;

COUNT(TYPE)

-----------

8

SQL> select sum(CAPACITY) from Rooms;

SUM(CAPACITY)

-------------

22

SQL>select avg(CAPACITY) from Rooms;

AVG(CAPACITY)

-------------

2.75

SQL> select distinct(TYPE) from Rooms;

TYPE

--------------------

Single

Deluxe

standard

Standard

deluxe

SQL>select stddev(CAPACITY) from Rooms;

STDDEV(CAPACITY)

----------------

1.03509834

SQL>select HOSTELID , sum(CAPACITY) from Rooms Group By HOSTELID;

HOSTELID SUM(CAPACITY)

---------- -------------

1 8

2 6

5 1

4 2

3 2

7 3

6 rows selected.

SQL>select HOSTELID, sum(CAPACITY) from Rooms Group By HOSTELID having sum(CAPACITY)&gt;2;

HOSTELID SUM(CAPACITY)

---------- -------------------------------

1 8

2 6

7 3

**VIEWS:**

SIMPLE VIEW:WITH ONLY ONE TABLE

QUERY AND EXAMPLE:

CREATE VIEW CAKESUPPLY AS SELECT STUDENTS.DATEOFBIRTH,STUDENTS.FIRSTNAME,STUDENTS.PHONENUMBER FROM STUDENTS;

SELECT\* FROM CAKESUPPLY;

OUTPUT: +-------------+-----------+-------------+

| DATEOFBIRTH | FIRSTNAME | PHONENUMBER |

+-------------+-----------+-------------+

| 30-10-2004 | neeraj | 8900234567 |

| 21-04-2001 | sanjeev | 7902345671 |

| 22-04-2003 | nalini | 7902345311 |

| 20-06-2004 | saisri | 8932285927 |

| 20-06-2004 | goutham | 8930085927 |

| 01-02-2004 | sowjanya | 8930085927 |

| 01-02-2003 | meenakshi | 9032245671 |

| 02-11-2000 | arjun | 9823456711 |

+-------------+-----------+-------------+

VIEW WITH MULTIPLE TABLES:

SELECT \* FROM BOOKING;

CREATE VIEW VACATE AS

SELECT STUDENTS.studentid, BOOKING.roomid

FROM STUDENTS

JOIN BOOKING ON STUDENTS.studentid = BOOKING.studentid;

OUTPUT:

+-----------+--------+

| studentid | roomid |

+-----------+--------+

| 1 | 1 |

| 2 | 2 |

| 3 | 3 |

| 4 | 4 |

| 6 | 5 |

| 8 | 6 |

+-------------+------------+

DELETE THE ROW FROM VIEW:

DELETE FROM CAKESUPPLY WHERE FIRSTNAME='NEERAJ';

SELECT\* FROM CAKESUPPLY;

+-------------+-----------+-------------+----------------------+

| DATEOFBIRTH | FIRSTNAME | PHONENUMBER |

+-------------+-----------+-------------+----------------------+

| 21-04-2001 | sanjeev | 7902345671 |

| 22-04-2003 | nalini | 7902345311 |

| 20-06-2004 | saisri | 8932285927 |

| 20-06-2004 | goutham | 8930085927 |

| 01-02-2004 | sowjanya | 8930085927 |

| 01-02-2003 | meenakshi | 9032245671 |

| 02-11-2000 | arjun | 9823456711 |

+-------------+-----------+-------------+------------------------+

PARENT TABLE ASLO DIDN’T CONTAIN NEERAJ ROW :

SELECT\* FROM STUDENTS;

+-----------+-----------+--------------+--------+-------------+-------------+-------------------+

| STUDENTID | FIRSTNAME | LASTNAME | GENDER | DATEOFBIRTH | PHONENUMBER|

EMAIL |

+-----------+-----------+--------------+--------+-------------+-------------+-------------------+

| 2 | sanjeev | kumar | male | 21-04-2001 | 7902345671 | sanjeev@gmail.com |

| 3 | nalini | padmanaban | female | 22-04-2003 | 7902345311 | nalini@gmail.com |

| 4 | saisri | chigilipalli | female | 20-06-2004 | 8932285927 | saisri@gmail.com |

| 5 | goutham | nanda | male | 20-06-2004 | 8930085927 | nanda@gmail.com |

| 6 | sowjanya | kintali | female | 01-02-2004 | 8930085927 | sowji@gmail.com |

| 7 | meenakshi | desai | female | 01-02-2003 | 9032245671 | meena@gmail.com |

| 8 | arjun | suravar | male | 02-11-2000 | 9823456711 | arjun@gmail.com |

+-----------+-----------+--------------+--------+-------------+-------------+-------------------+

REPLACE OF VIEW TABLE (CAKESUPPLY):

CREATE OR REPLACE VIEW CAKESUPPLY AS SELECT STUDENTID,DATEOFBIRTH,FIRSTNAME,PHONENUMBER FROM STUDENTS WHERE GENDER='FEMALE';

SELECT\* FROM CAKESUPPLY;

+-----------+-------------+-----------+-------------+---------------------------+

| STUDENTID | DATEOFBIRTH | FIRSTNAME | PHONENUMBER |

+-----------+-------------+-----------+-------------+----------------------------+

| 3 | 22-04-2003 | nalini | 7902345311 |

| 4 | 20-06-2004 | saisri | 8932285927 |

| 6 | 01-02-2004 | sowjanya | 8930085927 |

| 7 | 01-02-2003 | meenakshi | 9032245671 |

+-----------+-------------+-----------+-------------+----------------------------+

UPDATE THE CAKESUPPLY :

UPDATE CAKESUPPLY SET FIRSTNAME='VISHALI' WHERE STUDENTID=3;

SELECT\* FROM CAKESUPPLY;

OUTPUT:

+-----------+-------------+-----------+-------------+----------------------------+

| STUDENTID | DATEOFBIRTH | FIRSTNAME | PHONENUMBER |

+-----------+-------------+-----------+-------------+-----------------------------+

| 3 | 22-04-2003 | VISHALI | 7902345311 |

| 4 | 20-06-2004 | saisri | 8932285927 |

| 6 | 01-02-2004 | sowjanya | 8930085927 |

| 7 | 01-02-2003 | meenakshi | 9032245671 |

+-----------+-------------+-----------+--------------+----------------------------+

VIEW CREATED WITH MULTIPLE TABLE USING JOIN. SO IT CANNOT DELETE A ROW FROM VIEW TABLE:

EXAMPLE:

CREATE VIEW VACATE AS

SELECT STUDENTS.studentid, BOOKING.roomid

FROM STUDENTS

JOIN BOOKING ON STUDENTS.studentid = BOOKING.studentid;

OUTPUT:

+-----------+--------------+

| studentid | roomid |

+-----------+--------+

| 2 | 2 |

| 3 | 3 |

| 4 | 4 |

| 6 | 5 |

| 8 | 6 |

+-----------+--------------+

DELETE FROM VACATE WHERE STUDENTID=1;

SELECT\* FROM VACATE;

RROR 1395 (HY000) at line 85: Can not delete from join view 'sandbox\_db.vacate'

Queries & nested queries:

SQL> select \* from students where studentid=(select bookingid from booking where booking.bookingid=students.studentid);

STUDENTID FNAME LNAME GENDER DOB MOBILENO

EMAIL

---------- -------------------- -------------------- ------------------------------ ---------------------

------------

1 neeraj chopra male 30-10-2004 8900234567 neerajchopra@gmail.com

2 sanjeev kumar male 21-04-2001 7902345671 sanjeev@gmail.com

3 nalini padmanaban female 22-04-2003 7902345311 nalini@gmail.com

SQL> select fname,email from students s,booking b where b.bookingid=s.id;

FNAME EMAIL

-------------------- --------------------------------------------------

neeraj neerajchopra@gmail.com

nalini nalini@gmail.com

SQL> select fname from students s,hostel h,rooms r where r.roomid=h.hostel\_id and r.roomid=s.roomid and h.name='dwarakamai Hostel';

FNAME

--------------------

gouthami

Functions :

SQL> create or replace function no\_of\_students return number is

2 total number(2):=0;

3 begin

4 select count(id) into total from students;

5 return total;

6 end;

7 /

SQL> declare

2 c number(2);

3 begin

4 c:=no\_of\_students();

5 dbms\_output.put\_line('total no.of students'|| c);

6 end;

7 /

total no.of students8

Triggers:

1.creating a trigger on rooms table which give error when the capacity of room is more than 5

SQL> create or replace trigger t1 before insert on rooms

2 for each row

3 declare

4 begin

5 if((:new.capacity>5)) then

6 raise\_application\_error(-20110,'room size must be lessthan or equal to 5');

7 end if;

8 end;

9 /

Trigger created.

SQL> insert into rooms values(12,6,6,'standard');

insert into rooms values(12,6,6,'standard')

\*

ERROR at line 1:

ORA-20110: room size must be lessthan or equal to 5

ORA-06512: at "SYSTEM.T1", line 4

ORA-04088: error during execution of trigger 'SYSTEM.T1'

1. Create trigger on students where updated names should be stored in another table having name,id and corresponding roomid of the student

SQL> create table updatingstudents(id int,name char(20),roomid int);

Table created.

SQL> create or replace trigger t2

2 before update of fname on students

3 for each row

4 begin

5 insert into updatingstudents values(:new.id,:new.fname,:new.roomid);

6 end;

7 /

Trigger created.

SQL> update students set fname='gouthami' where id=5;

1 row updated.

SQL> select \* from updatingstudents;

ID NAME ROOMID

---------- -------------------- ----------

5 gouthami 1

SQL> select \* from students;

ID FNAME GENDER DOB MOBILENO EMAIL ROOMID

---------- -------------------- ------------------------------ ------------------------------ ---------- -------------------------------------------------- ----------

1 neeraj male 30-10-2004 8900234567 neerajchopra@gmail.com 2

2 sanjeev male 21-04-2001 7902345671 sanjeev@gmail.com 5

3 nalini female 22-04-2003 7902345311 nalini@gmail.com 3

4 saisri female 20-06-2004 8932285927 saisri@gmail.com 3

5 gouthami male 20-06-2004 8930085927 nanda@gmail.com 1

6 sowjanya female 01-02-2004 8930085927 sowji@gmail.com 4

7 meenakshi female 01-02-2003 9032245671 meena@gmail.com 4

8 arjun male 02-11-2000 9823456711 arjun@gmail.com 6

8 rows selected.

SQL> update students set fname='anand' where email=' arjun@gmail.com';

0 rows updated.

SQL> update students set fname='anand' where email='arjun@gmail.com';

1 row updated.

SQL> select \* from updatingstudents;

ID NAME ROOMID

---------- -------------------- ----------

5 gouthami 1

8 anand 6

Normalization up to Third Normal Form:

Original table:

+-----------+--------------------+------------------------------+-------+

| HOSTEL\_ID | NAME | ADDRESS | FEES |

+-----------+--------------------+------------------------------+-------+

| 1 | DWARAKAMAI HOSTEL | KOMMADI, MADURAWADA | 60000 |

| 2 | ORANGE HOSTEL | KOMMADI, MADURAWADA | 70000 |

| 3 | SEA BREEZE HOSTEL | BEACH ROAD, VISAKHAPATNAM | 40000 |

| 4 | SUNRISE HOSTEL | DABAGARDENS, VISAKHAPATNAM | 50000 |

| 5 | Hilltop Hostel | Kailasagiri, Visakhapatnam | 45000 |

| 6 | City Center Hostel | Dwaraka Nagar, Visakhapatnam | 60000 |

| 7 | Lighthouse Hostel | Yarada Beach, Visakhapatnam | 50000 |

| 8 | Forest Edge Hostel | Araku Valley, Visakhapatnam | 80000 |

+-----------+--------------------+------------------------------+-------+

To convert this table into Normal Forms (1NF, 2NF, and 3NF), we need to ensure that the database design is organized and structured according to normalization rules:

### First Normal Form (1NF):

1NF requires that each table column must contain atomic (indivisible) values and that each entry in a column must be of the same data type.

Given table is already in 1NF because:

* Each cell contains a single value.
* Each column contains values of a single type.

### Second Normal Form (2NF):

2NF requires that the table is in 1NF and all non-key attributes are fully functionally dependent on the primary key.

To convert the table to 2NF, we need to identify any partial dependencies and eliminate them. Since the table has a single-column primary key (HOSTEL\_ID), all non-key attributes (NAME, ADDRESS, FEES) are fully functionally dependent on HOSTEL\_ID. Thus, it is already in 2NF.

### Third Normal Form (3NF):

3NF requires that the table is in 2NF and all the attributes are functionally dependent only on the primary key (i.e., there should be no transitive dependency).

In this table, the ADDRESS attribute contains multiple pieces of information (location, city). We can further decompose this to ensure that there are no transitive dependencies.

**After removing transitive dependencies to ensure it is in 3NF:**

+-----------+--------------------+-------+-------------------+

| HOSTEL\_ID| NAME | FEES | LOCATION\_ID |

+-----------+--------------------+-------+-------------------+

| 1 | DWARAKAMAI HOSTEL | 60000 | 1 |

| 2 | ORANGE HOSTEL | 70000 | 1 |

| 3 | SEA BREEZE HOSTEL | 40000 | 2 |

| 4 | SUNRISE HOSTEL | 50000 | 3 |

| 5 | Hilltop Hostel | 45000 | 4 |

| 6 | City Center Hostel | 60000 | 5 |

| 7 | Lighthouse Hostel | 50000 | 6 |

| 8 | Forest Edge Hostel | 80000 | 7 |

+-----------+--------------------+-------+-------------------+

+-----------+-------------------+--------------------+

| LOCATION\_ID | AREA | CITY |

+-------------+-----------------+--------------------+

| 1 | KOMMADI | MADURAWADA |

| 2 | BEACH ROAD | VISAKHAPATNAM |

| 3 | DABAGARDENS | VISAKHAPATNAM |

| 4 | Kailasagiri | Visakhapatnam |

| 5 | Dwaraka Nagar| Visakhapatnam |

| 6 | Yarada Beach | Visakhapatnam |

| 7 | Araku Valley| Visakhapatnam |

+-------------+-----------------+--------------------+

By decomposing the original table into the Hostel and Location tables, we have removed any transitive dependencies, ensuring that the schema is in 3NF.

Boyce-Codd Normal Form (BCNF):

* BCNF is the advance version of 3NF. It is stricter than 3NF.
* A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
* For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

HOSTEL TABLE:

+-----------+--------------------+-------+-----------------+

| HOSTEL\_ID | NAME | FEES | LOCATION\_ID |

+-----------+--------------------+-------+----------------+

| 1 | DWARAKAMAI HOSTEL | 60000 | 1 |

| 2 | ORANGE HOSTEL | 70000 | 1 |

| 3 | SEA BREEZE HOSTEL | 40000 | 2 |

| 4 | SUNRISE HOSTEL | 50000 | 3 |

| 5 | Hilltop Hostel | 45000 | 4 |

| 6 | City Center Hostel| 60000 | 5 |

| 7 | Lighthouse Hostel | 50000 | 6 |

| 8 | Forest Edge Hostel| 80000 | 7 |

+-----------+--------------------+-------+-------------+

n the Hostel table:

* HOSTEL\_ID is the primary key.
* HOSTEL\_ID -> NAME, FEES, LOCATION\_ID.

Since HOSTEL\_ID is a super key and determines all other attributes, this table is in BCNF.

LOCATION TABLE:

+-------------+-------------------+-----------------+

| LOCATION\_ID | AREA | CITY |

+-------------+-------------------+-----------------+

| 1 | KOMMADI | MADURAWADA |

| 2 | BEACH ROAD | VISAKHAPATNAM |

| 3 | DABAGARDENS | VISAKHAPATNAM |

| 4 | Kailasagiri | Visakhapatnam |

| 5 | Dwaraka Nagar | Visakhapatnam |

| 6 | Yarada Beach | Visakhapatnam |

| 7 | Araku Valley | Visakhapatnam |

+-------------+-------------------+-----------------+

In the Location table:

* LOCATION\_ID is the primary key.
* LOCATION\_ID -> AREA, CITY.

Since LOCATION\_ID is a super key and determines all other attributes, this table is also in BCNF.

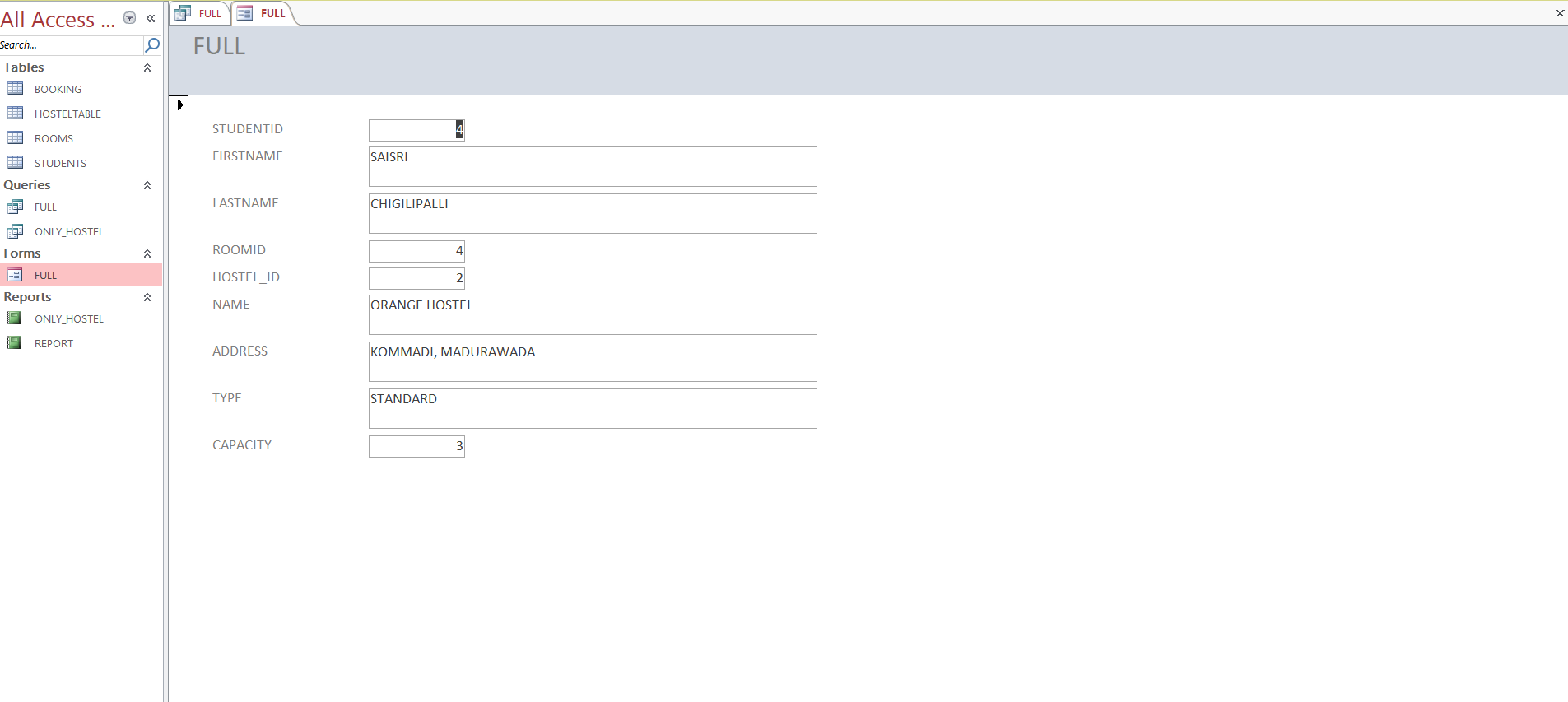
Both tables, Hostel and Location, comply with BCNF as all non-trivial functional dependencies have a super key on the left-hand side. Thus, the schema provided is in BCNF.

FORMS AND REPORTS :

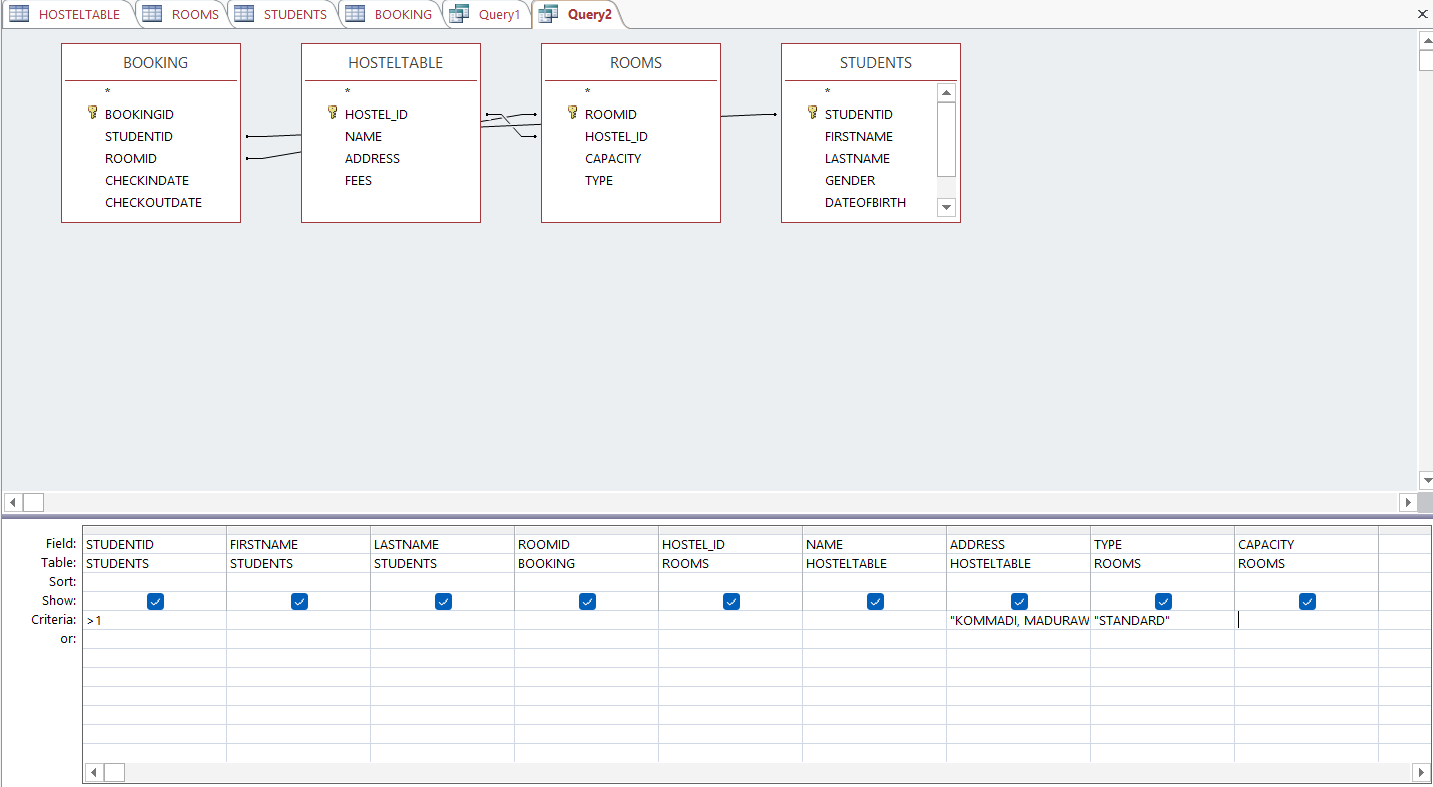
In Microsoft Access, forms and reports are essential tools for data management and presentation. Forms facilitate data entry and user interaction by providing a user-friendly interface to input and edit data in tables. They can include controls like text boxes, buttons, and dropdowns for efficient data handling. Reports, on the other hand, are designed to format, summarize, and present data in a readable and printable format. They can include grouping, sorting, and filtering to highlight key information. Both forms and reports can be customized with design and layout features to meet specific user requirements, improving data accuracy and decision-making processes.

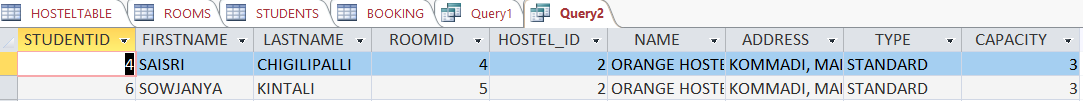
1.DESIGN A FORM ON STUDENTS WHO STAY IN THE KOMMADI PG HOSTELS WITH THE HOSTEL ADDRESS , ROOM ID , HOSTEL NAME AND TYPE OF THE ROOM AND THE ROOM SHOULD BE STANDARD.

FORMS:

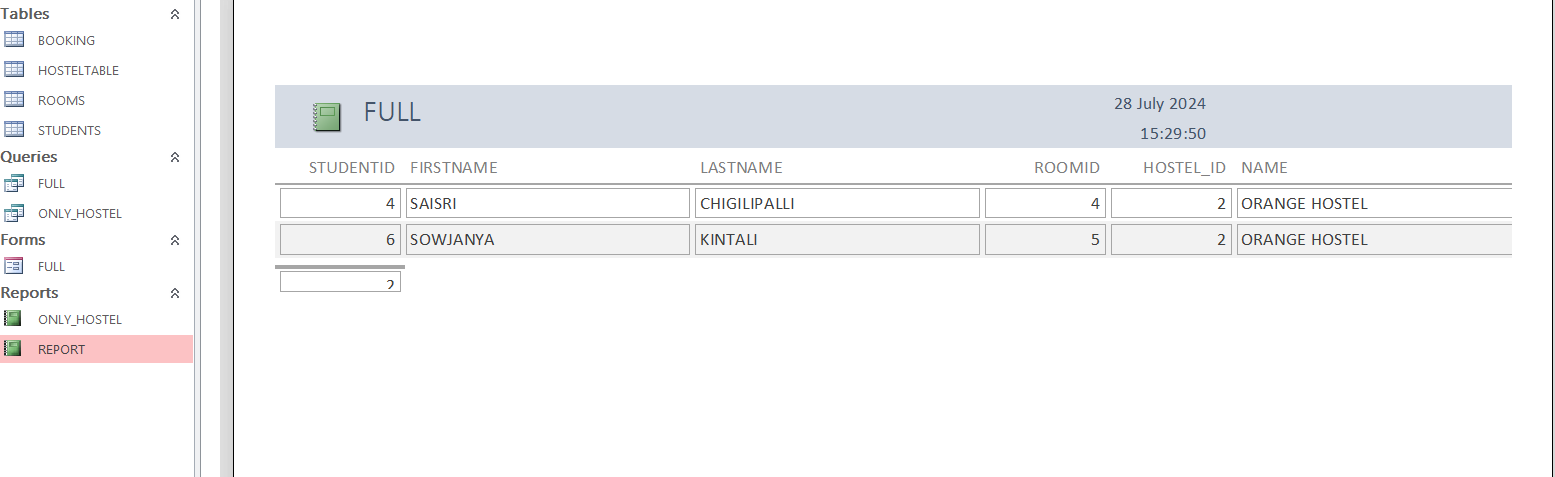


QUERY:

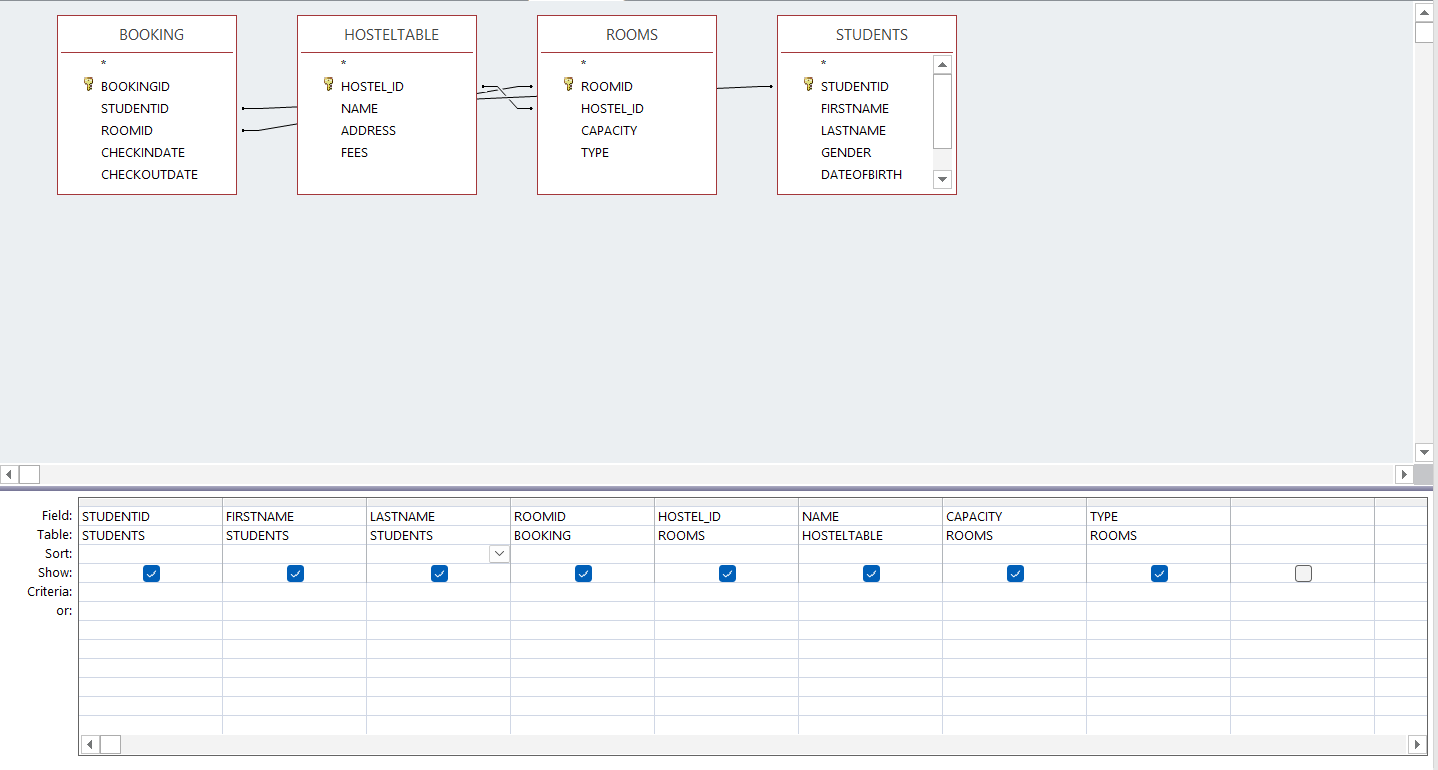


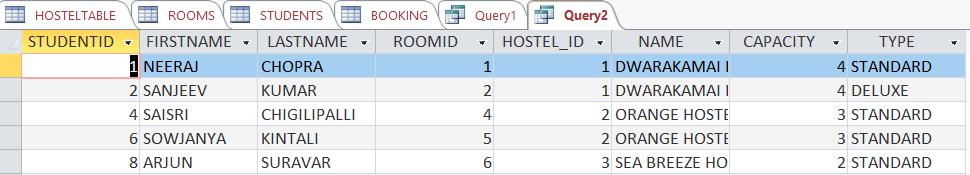


REPORTS:



DESIGN A REPORTS ON STUDENTS WHO ARE STAYING IN VISAKHAPATNAM HOSTELS?

x



Conclusion:

The Hostel Accounting System using MySQL effectively manages hostel data through well-structured schemas, constraints, normalization, and SQL operations. By implementing tables for hostels, rooms, students, and bookings along with functions, triggers, views, and reports, the project ensures data accuracy, consistency, and ease of access. Overall, it demonstrates how MySQL can be used to design a robust, scalable, and practical database solution for hostel management.