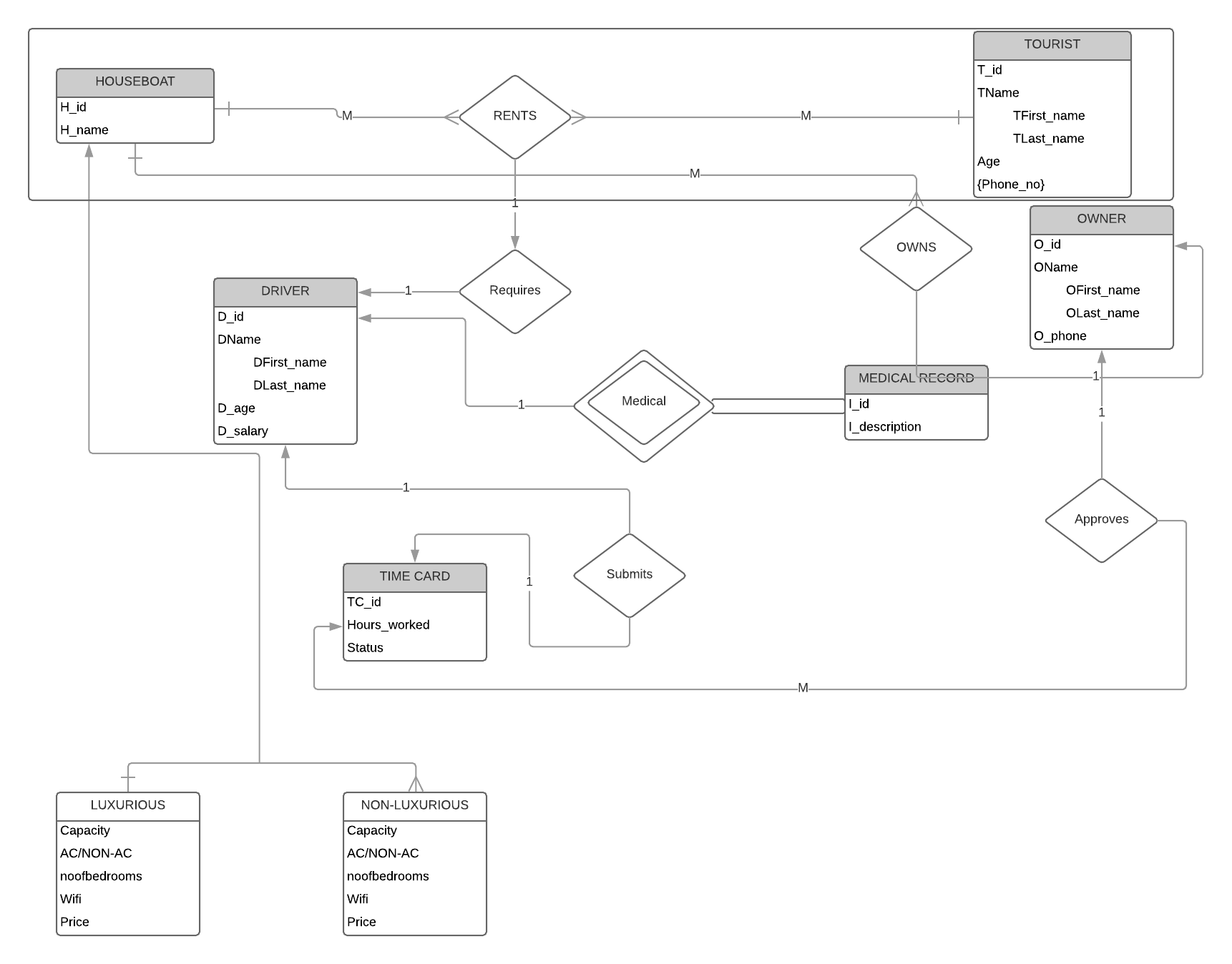
HOUSEBOAT MANAGEMENT

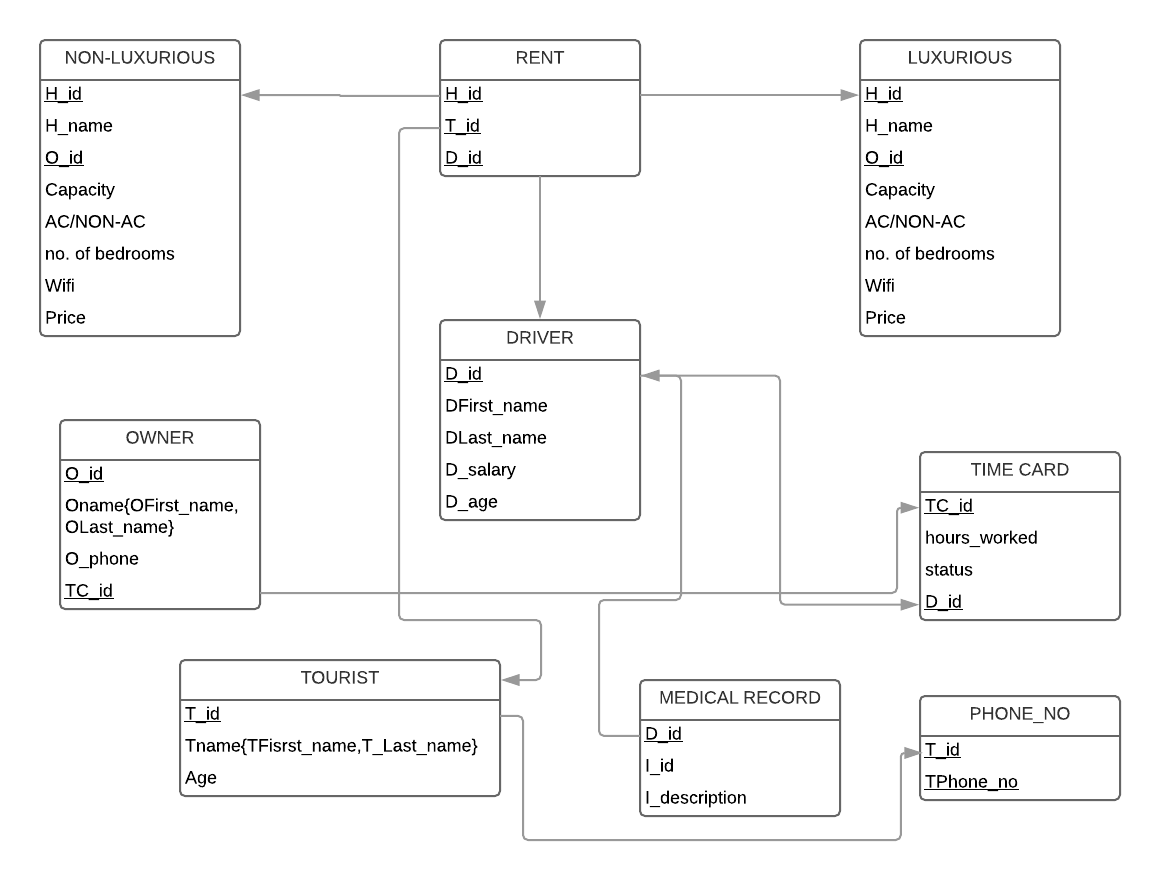
**Problem Statement:**

An agency keeps track of house boats, its owners and the customers who rented it. An owner identified by owner number, name and city can have many house boats and a house boat is owned by only 1 person. A house boat is identified by an id, name and capacity (number of people it can hold). A customer can rent many house boats and a house boat can be rented by many customers on different dates. Design a database system for the Agency to maintain all the data conveniently and efficiently.

**ER Diagram:**

****

**Schema Diagram:**

****

**Relational Schema:** Primary Keys: \_\_\_\_\_\_\_

Foreign Keys: grey

Tourist(T\_id, TFirst\_name, TLast\_name, Age)

Rent(H\_id, T\_id, D\_id)

Driver(D\_id, DFirst\_name, DLast\_name, D\_salary, D\_age)

Medical\_Record(**D\_id**, I\_id, I\_description)

Time Card(TC\_id, Hours\_worked, Status, D\_id)

Owner(O\_id, OFirst\_name, OLast\_name, O\_phone, TC\_id)

Luxurious(H\_id, H\_name, O\_id, Capacity, AC/non AC, Noofbedrooms,Wifi, Price)

Non- Luxurious(H\_id, H\_name, O\_id, Capacity, AC/non AC, Noofbedrooms,Wifi, Price)

Phone\_No(T\_id, TPhone\_no)

**Summary:**

* The table tourist contains the details of the tourist - name, age and phone number.
* The entities houseboat and tourist forms an aggregate which results in the formation of relation rents, which will be having the primary keys of both tourist and houseboat entities. If driver is unavailable, tourist wont be able to rent a houseboat.
* The driver entity contains details of the driver such as his id and age. It also contains his medical records, which is a separate entity and is treated as a weak enitity. The primary key of the driver entity is included in the medical records entity. All the attributes in the medical records will belong to only one driver(Total Participation).
* The houseboat entity can be differentiated on the basis of type as luxurious and non-luxurious, which can be taken as two separate entities with attributes of their own. Since these two entities are disjoint, we can neglect the houseboat entitiy by adding its attributes to the luxurious and non-luxurious entities.
* The time card entity will have details related to the driver, such as his id and the number of hours worked by him. Each driver will have his time crad which has to be approved by the respective owner.

The relation between entities:

* One to one relationship: Time card and Driver, Driver and Medical records.
* One to many relationship: Luxurious and owner, Non-Luxurious and Owner, Time card and owner.
* Many to many relationship: Tourist and Luxurious, Tourist and Non-Luxurious.

**Out of Scope:**

* We used aggregation between Houseboat and Tourist entities.
* Total participation is included through the Medical records entity.
* Added new entities such as Medical records, Time card and Luxurious/ Non Luxurious entities.
* Medical record is a weak entity.
* Specialization is achieved by two entities: Luxurious and Non-Luxurious.

**Normalization:**

The Normalization process can be shown as follows:

A = Tid

B = TFirst\_name

C = TLast\_name

D = Age

E = Phone

F = Country

G = H\_id

H = H\_name

I = Capacity

J = AC/NON-AC

K = Bedroom

L = Wifi

M = Price

N = Rent\_date

O = D\_id

P = DFirst\_name

Q = DLast\_name

R = D\_salary

S = D\_age

T = I\_id

U = I\_description

V = O\_id

W = OFirst\_name,

X = OLast\_name,

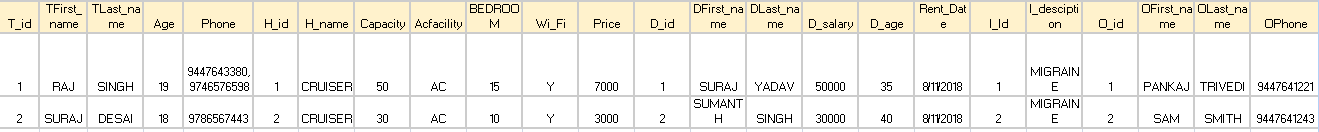
Y = OPhone.

0NF Form:

In the 0NF Form, we list all the attributes as follows:

T\_id, TFirst\_name, TLast\_name, Age, Phone, Country, H\_id, H\_name, Capacity, Acfacility, Bedroom, Wifi, Price, Rent\_date, D\_id, DFirst\_name, DLast\_name, D\_salary, D\_age, I\_id, I\_description, O\_id, OFirst\_name, OLast\_name, OPhone.

We combined all of the attributes into a single table. The table is shown here as follows, with a few values added in the table:



Functional dependencies are:

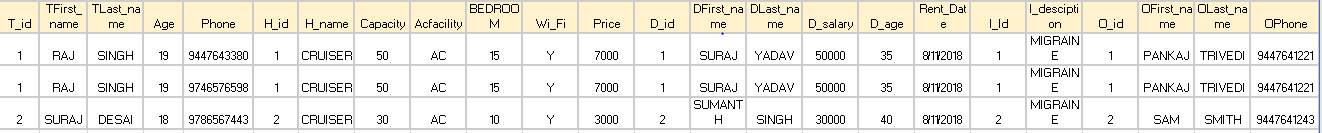
AGNOV -> ABCDEFGHIJKLMNOPQRSTUVWXY

Primary Key:

AGNOV

1NF Form:

In the 1NF Form, we split the rows that had the multivalued attributes, so that each column can have unique values.



Functional Dependencies:

AGNOV -> ABCDEFGHIJKLMNOPQRSTUVWXY

2NF Form:

In the 2NF Form, the partial dependencies are removed, and the tables can be split as follows:

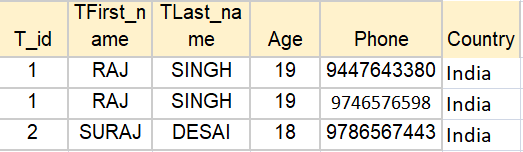
Tourist( T\_id, TFirst\_name, TLast\_name, Age, Phone, Country)

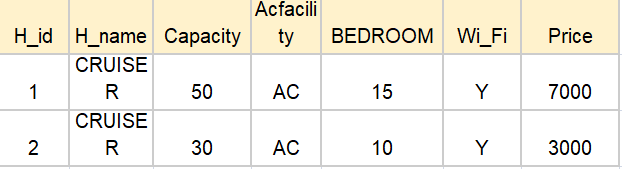
Houseboat(H\_id, H\_name, Capacity, Acfacility, Bedroom, Wifi, Price)

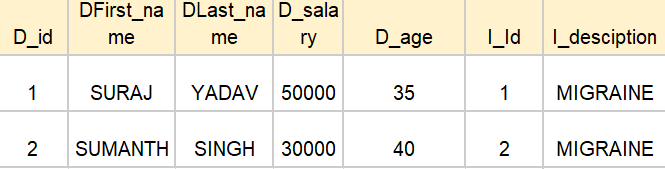
Driver(D\_id, DFirst\_name, DLast\_name, D\_salary, D\_age, I\_id, I\_description)

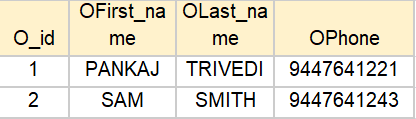
Owner(O\_id, OFirst\_name, OLast\_name, OPhone)

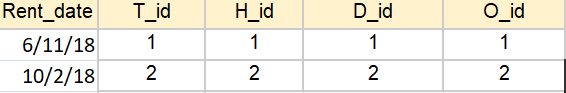
Management(Rent\_date, T\_id, H\_id, D\_id, O\_id)











The Tourist attributes have partial dependencies only on T\_id.

The Houseboat attributes have partial dependencies only on H\_id.

The Driver attributes have partial dependencies only on D\_id.

The Medical\_Record attributes have partial dependencies only on I\_id.

The Owner attributes have partial dependencies only on O\_id.

The Management entitiy contains the primary keys of all the other entities.

Functional Dependencies are:

A -> BCDEF

G -> HIJKLM

O -> PQRSTU

T -> U

V ->WXY

3NF Form:

In the 3NF Form, the tables can be further divided as follows:

Tourist( T\_id, TFirst\_name, TLast\_name, Age, Phone, Country)

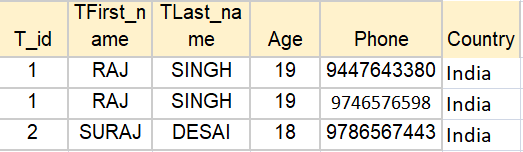
Houseboat(H\_id, H\_name, Capacity, AC/NON-AC, Bedroom, Wifi, Price)

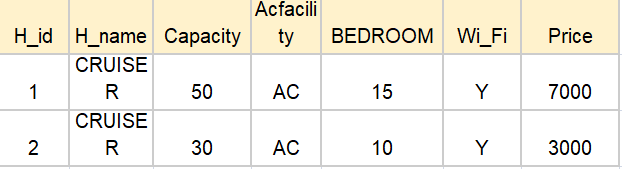
Driver(D\_id, DFirst\_name, DLast\_name, D\_salary, D\_age, I\_id)

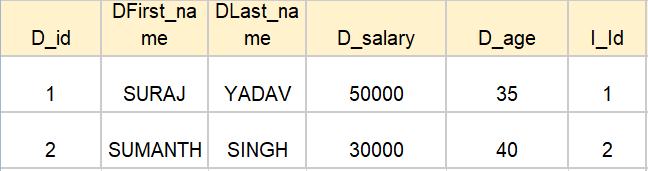
Medical\_Record(I\_id, I\_description)

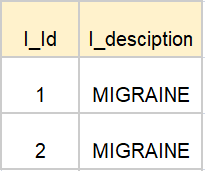
Owner(O\_id, OFirst\_name, OLast\_name, OPhone)

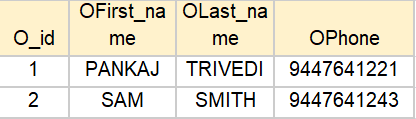
Management(Rent\_date, T\_id, H\_id, D\_id, O\_id)

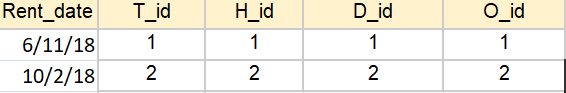












In the 3NF form, we remove the transitive dependencies. The transitive dependency here is found in the Driver entity, where the I\_description depends only on I\_id.

Functional Dependencies are:

A -> BCDEF

G -> HIJKLM

O -> PQRST

T -> U

V ->WXY

QUERIES

DDL STATEMENTS:

create table tourist(t\_id varchar(10) primary key, tfirst\_name varchar(20), tlast\_name varchar(20), age int, phone varchar(13), country varchar(20));

create table houseboat(h\_id varchar(10) primary key, h\_name varchar(20), capacity int, acfacility varchar(6), noofbedrooms int, wifi varchar(6), price int);

create table medicalrecords(i\_id varchar(10) primary key, idescription varchar(80));

create table driver(d\_id varchar(10) primary key, dfirst\_name varchar(20), dlast\_name varchar(20), dsalary int, dage int, i\_id varchar(10), foreign key(i\_id) references medicalrecords on update cascade on delete cascade);

create table boatowner(o\_id varchar(10) primary key, ofirst\_name varchar(20),olast\_name varchar(20), ophone varchar(13));

create table management(rent\_date date, t\_id varchar(10), h\_id varchar(10), d\_id varchar(10), o\_id varchar(10), foreign key(t\_id) references tourist on update cascade on delete cascade, foreign key(d\_id) references driver on update cascade on delete cascade, foreign key(h\_id) references houseboat on update cascade on delete cascade, foreign key(o\_id) references boatowner on update cascade on delete cascade, primary key(rent\_date, t\_id));

insert into tourist values('T1', 'Sowmya', 'Khanna', 29, '+919446252773', 'India');

insert into tourist values('T2', 'Rajesh', 'Khanna', 34, '+919497822113', 'India');

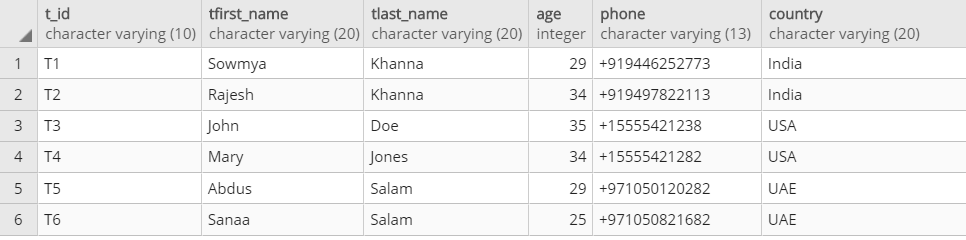
insert into tourist values('T3', 'John', 'Doe', 35, '+15555421238', 'USA');

insert into tourist values('T4', 'Mary', 'Jones', 34, '+15555421282', 'USA');

insert into tourist values('T5', 'Abdus', 'Salam', 29, '+971050120282', 'UAE');

insert into tourist values('T6', 'Sanaa', 'Salam', 25, '+971050821682', 'UAE');

select \* from tourist;



insert into houseboat values('H123', 'Aqua Castle', 6, 'AC', 2, 'Y', 33810);

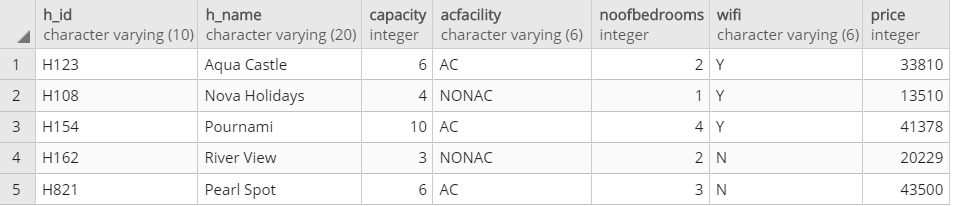
insert into houseboat values('H108', 'Nova Holidays', 4, 'NONAC', 1, 'Y', 13510);

insert into houseboat values('H154', 'Pournami', 10, 'AC', 4, 'Y', 41378);

insert into houseboat values('H162', 'River View', 3, 'NONAC', 2, 'N', 20229);

insert into houseboat values('H821', 'Pearl Spot', 6, 'AC', 3, 'N', 43500);

select \* from houseboat;



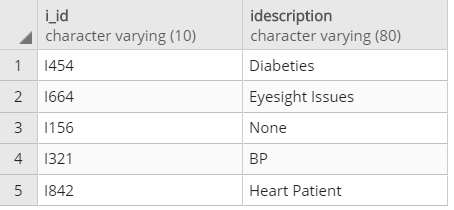
insert into medicalrecords values('I454','Diabeties');

insert into medicalrecords values('I156','None');

insert into medicalrecords values('I664','Eyesight Issues');

insert into medicalrecords values('I321','BP');

insert into medicalrecords values('I842','Heart Patient');

select \* from medicalrecords; 

insert into driver values('D454', 'Suresh', 'M. N', 12000, 40, 'I454');

insert into driver values('D156', 'Venu', 'S. K', 8000, 35, 'I156');

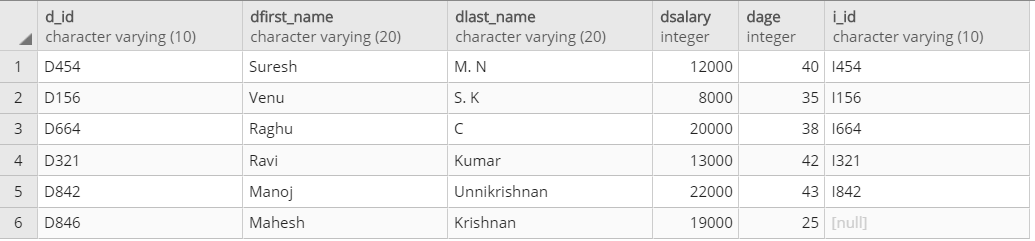
insert into driver values('D664', 'Raghu', 'C', 20000, 38, 'I664');

insert into driver values('D321', 'Ravi', 'Kumar', 13000, 42, 'I321');

insert into driver values('D842', 'Manoj', 'Unnikrishnan', 22000, 43, 'I842');

insert into driver values('D846', 'Mahesh', 'Krishnan', 19000, 25, NULL);

select \* from driver;



insert into boatowner values('O123','Aravindan', 'Menon', '+919746576958');

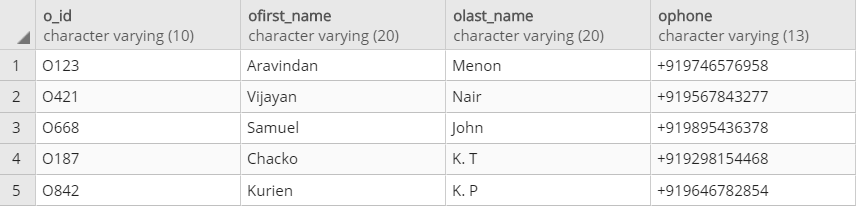
insert into boatowner values('O421','Vijayan', 'Nair', '+919567843277');

insert into boatowner values('O668','Samuel', 'John', '+919895436378');

insert into boatowner values('O187','Chacko', 'K. T', '+919298154468');

insert into boatowner values('O842','Kurien', 'K. P', '+919646782854');

select \* from boatowner;



insert into management values('06-11-18', 'T1', 'H108', 'D156', 'O187');

insert into management values('06-11-18', 'T2', 'H108', 'D156', 'O187');

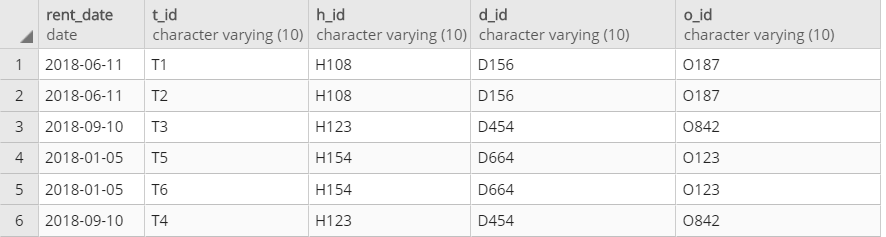
insert into management values('09-10-18', 'T3', 'H123', 'D454', 'O842');

insert into management values('09-10-18', 'T4', 'H123', 'D454', 'O842');

insert into management values('01-05-18', 'T5', 'H154', 'D664', 'O123');

insert into management values('01-05-18', 'T6', 'H154', 'D664', 'O123');

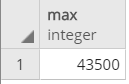
select \* from management;



Q1. Aggregate functions, group by....having:

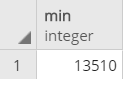
Find the maximum price a houseboat can have.

select max(price) from houseboat;



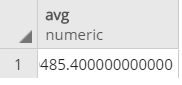
Find the minimum price a houseboat can have.

select min(price) from houseboat;



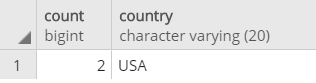
Find the average price of a housboat.

select avg(price) from houseboat;



Display the number of tourists whose age is greater than 30 country – wise if the count is greater than 1.

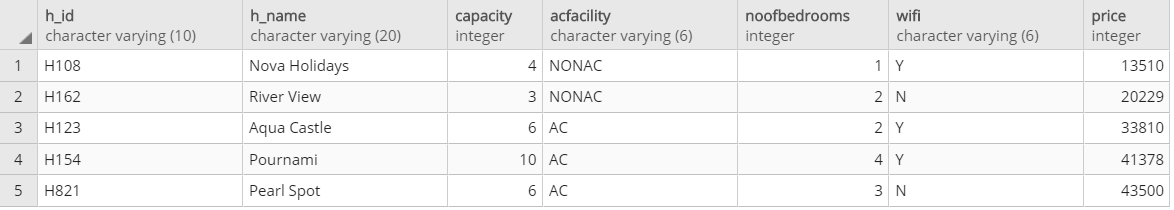
select count(t\_id), country from tourist where age>30 group by country having count(t\_id)>1;



Q2. Order by:

Display the details of each housboat in the increasing order of price.

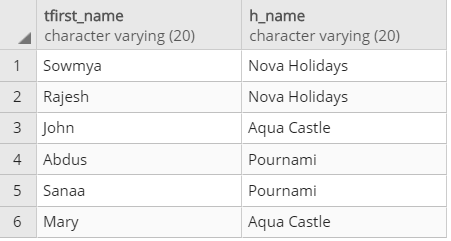
select \* from houseboat order by price;



Q3. Join, Outer Join

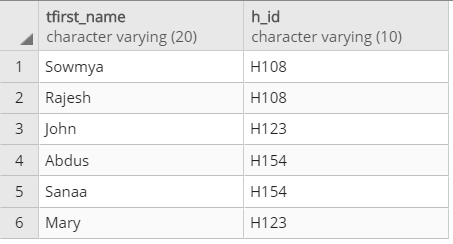
Display first name of each tourist and the name of the houseboat that they have rented.

select Tourist.TFirst\_name, Houseboat.H\_name from ((Management INNER JOIN Tourist ON Management.T\_id = Tourist.T\_id) INNER JOIN Houseboat ON Management.H\_id = Houseboat.H\_id);



Display the first name of the tourist along with the id of the houseboat that they have booked.

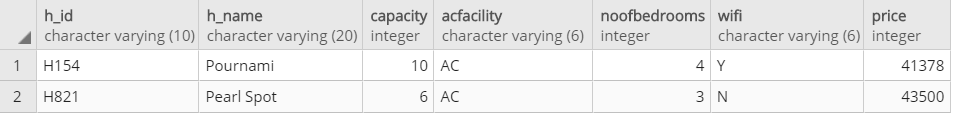
select Tourist.tfirst\_name, Management.h\_id from Tourist FULL OUTER JOIN Management ON Tourist.t\_id = Management.t\_id;



Q4. Query having Boolean Operators

Display the details of each housboat whose price is greater than 40000.

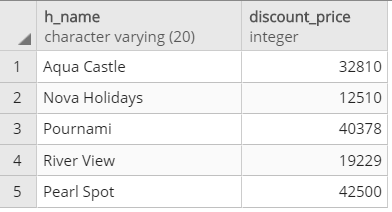
select \* from houseboat where price>40000;



Q5. Query using Arithmetic Operators

Display the name and the discount price of each houseboat, where the discount price = price – 1000.

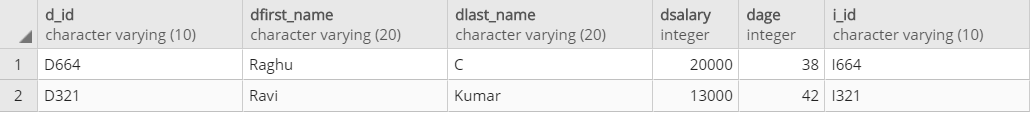
select h\_name, price-1000 as discount\_price from houseboat;



Q6. A search query using string operators

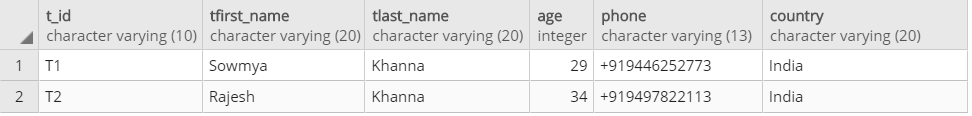
Select details of drivers whose first name starts with R.

select \* from driver where dfirst\_name like 'R%';



Select details of tourist whose last name has a h in it.

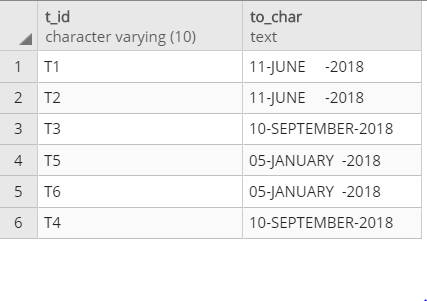
select \* from tourist where tlast\_name like '%h%';



Q7. Usage of to\_char, extract

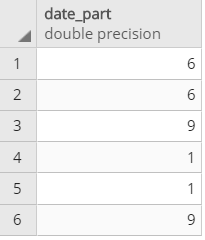
Display tourist id, and the corresponding rent date in the format DD-MONTH-YYYY

select t\_id, rent\_date, to\_char(rent\_date, 'DD-MONTH-YYYY') from management;



Display the months at which each houseboat was rented.

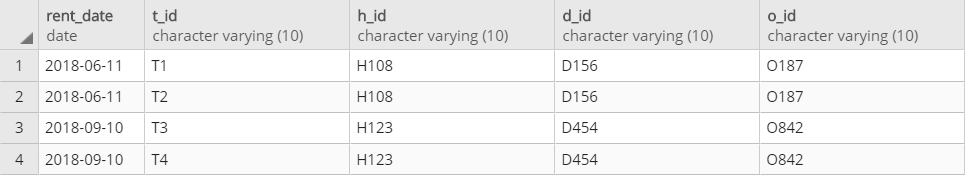
select extract(month from rent\_date) from Management;



Q8. Between, IN, Not Between, NOT IN

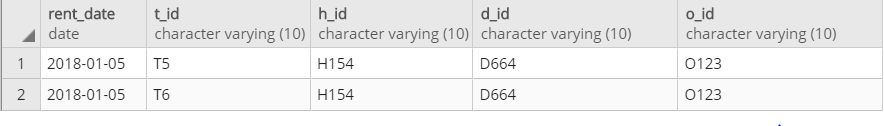
Display details of records where rent dates is between '06-10-18' and '09-11-18'

select \* from Management where Rent\_date BETWEEN '06-10-18' AND '09-11-18';



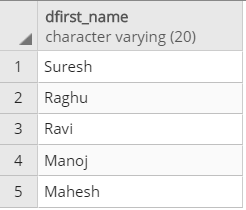
Display details of records where rent dates is not between '06-10-18' and '09-11-18'

select \* from Management where Rent\_date NOT BETWEEN '06-10-18' AND '09-11-18';



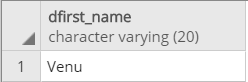
Display first name from driver whose salary is greater than 9000,

select dfirst\_name from driver where dsalary in (select dsalary from driver where dsalary>9000);



Display first name of driver whose salary is not greater than 9000,

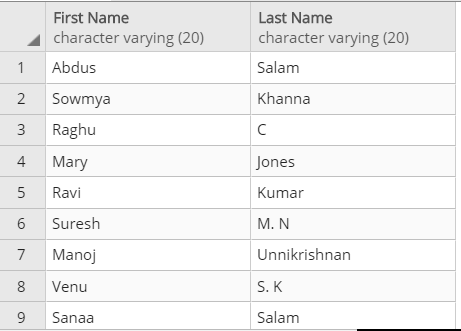
select dfirst\_name from driver where dsalary not in (select dsalary from driver where dsalary>9000);



Q9. Set operations

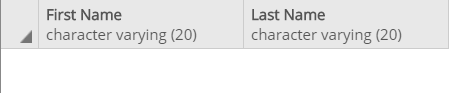
Display first name and last name of all tourists and drivers,

select tfirst\_name "First Name", tlast\_name "Last Name" from tourist union select dfirst\_name, dlast\_name from driver;



Display first name and last name of tourists and drivers

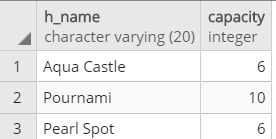
select tfirst\_name "First Name", tlast\_name "Last Name" from tourist intersect select dfirst\_name, dlast\_name from driver;



Q10. Subqueries

Display houseboat name and capacity of houseboats that have ac facility

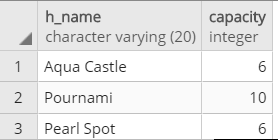
select h\_name, capacity from houseboat where h\_id in(select h\_id from houseboat where acfacility like 'AC');



Q11. Subquery using EXISTS/NOT EXISTS, ANY/ALL

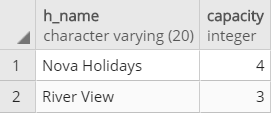
Display houseboat name and capacity of houseboats that have capacities greater than or equal to the capacity of Pearl Spot

select h\_name, capacity from houseboat where capacity >=ANY(select capacity from houseboat where h\_name like 'Pearl Spot');



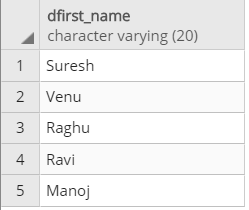
Select housboat name and capacity of houseboats that have capacity less than the capacity of Pearl Spot

select h\_name, capacity from houseboat where capacity < ALL(select capacity from houseboat where h\_name like 'Pearl Spot');



Display first names of drivers who have medical records

select d.dfirst\_name from driver d where EXISTS(SELECT i\_id from medicalrecords m where d.i\_id!=m.i\_id);



Display first names of drivers who do not have medical records

select d.dfirst\_name from driver d where NOT EXISTS(SELECT i\_id from medicalrecords m where d.i\_id!=m.i\_id);

