Name: Rahul Purushottam Gaonkar

HOMEWORK-3

Problem 1:

Consider a database modeling an online restaurant booking system, where a customer can search restaurants and book them. It is given by the following highly simplified database schema:

CUSTOMER (cid, cname, phone)
RESTAURANT (rid, rname, raddress, description, capacity)
BOOKING (bid, cid, rid, btime, quantity)

Each customer has a unique cid, along with a cname and a phone number. Each restaurant has a unique rid, along with a rname, raddress, and a short description such as “Chinese Food” or “Chicken, Sandwiches, Wings”. The capacity indicates how many seats in this restaurant. (We assume that parties can share tables as needed, or seats can be moved between tables, so we only keep track of seats, not tables.) For the booking table, it has a unique bid, and btime is the start time for the booking. You should assume that every booking starts on the hour and lasts for one hour (e.g., 2-3pm, 3-4pm, etc.). The quantity is the number of seats requested.

In this problem, you are going to design a simple web front end with PHP which allows people to access some of the information using a web browser. Your web application should support the following:

(i) On the start page, there is a box where users can type a customer name, a keyword, and the number of people. Also, the customer can enter or select the date and time they want to book for.

(ii) After the button is pressed, your PHP code should provide a list of all available restaurants which have the keyword in their descriptions or names, along with detailed information of each restaurant. If no keyword is entered, all available restaurants should be displayed.
Note: To simplify this problem, we assume every reservation is for one hour. Thus, to check if a restaurant is available, you can simply use the capacity minus the sum of the quantity already booked within that hour.

(iii) Next, the user should be able to click a button to book a table. After that, your code should add a record into the booking table and turn to another page which shows all previous bookings of this customer.

Test your application, using the data provided on the course page to populate your database. You will meet with the grader to give a quick demo. (Details on scheduling demos will be announced later, and demos will be a few days after the deadline for the other problems.)

Problem 2:

Consider a relational schema R = (A, B, C, D, E, F, G) satisfying the functional dependencies
F = {BF -> C, CF -> D, G -> A, G->E, FG -> AD}

a. Derive all candidate keys for this schema.

Ans:
BFG is the candidate key

Explanation:
$(BFG)^+$ i.e. Attribute Set Closure
Result = BFG

Result = BFGC (BF->C and BF $\subseteq$ BFG)
Result = BFGCD (CF->D and CF $\subseteq$ BFGCD)
Result = BFGCDA (G->A and G $\subseteq$ BFGCDA)
Result = BFGCDAE (G->E and G $\subseteq$ BFGCDAE)
Therefore, BFG is the candidate key as we can determine all the attributes mentioned in the relation R by it.

b. Derive a canonical cover of the functional dependencies in F
Ans:
1. Combining G->A and G->E into G->AE
Set is now {BF -> C, CF -> D, G -> AE, FG -> AD}
2. A is extraneous in FG -> AD
Set is now {BF -> C, CF -> D, G -> AE, FG ->D}

Canonical Cover: {BF -> C, CF -> D, G -> AE, FG ->D}

c. Is the above schema in BCNF? Prove or disprove. If it is not in BCNF, convert it into BCNF.
Ans:
R is not in BCNF (BF->C and BF is not the super key)
On decomposition we get
R1 = (B, F, C) and R2= (A, B, D, E, F, G)
R2 is not in BCNF (G->A, G->E and G is not the super key)
On decomposition we get
R1 = (B, F, C), R2 = (G, A, E) and R3 = (B, D, F, G)
R3 is not in BCNF (FG -> D and FG is not the super key)
On decomposition we get
R1 = (B, F, C), R2 = (G, A, E), R3 = (B, F, G) and R4 = (F, G, D)
Therefore, we have converted relation R in BCNF by decomposing it into relations R1, R2 and R3 and R4.

d. In the BCNF schema from c) dependency-preserving? Prove or disprove. If not, convert it into 3NF.
Ans:
The above BCNF schema is not dependency-preserving as the functional dependencies CF -> D is not held by the schema.

Steps to convert R into 3NF:

1. Finding out the canonical cover of F
Canonical Cover: {BF -> C, CF -> D, G -> AE, FG ->D}

2. Generate the following 3NF schema from the canonical cover
R1 = (B, F, C), R2 = (C, F, D), R3 = (G, A, E), R4 = (F, G, D)

3. If any of the above schema doesn't contain the candidate key i.e. BFG in this case create a new relation containing it.
R1 = (B, F, C), R2 = (C, F, D), R3 = (G, A, E), R4 = (F, G, D), R5 = (B, F, G)

4. Since there are no redundant schemas, no schema will be deleted.
R1 = (B, F, C), R2 = (C, F, D), R3 = (G, A, E), R4 = (F, G, D), R5 = (B, F, G)
Therefore, we have converted relation R in 3NF by decomposing it into relations R1, R2, R3, R4, R5.

Problem 3:

Consider the following single-table database schema modeling a patient-treatment database at a hospital:

PatientTreatment (pid, pname, page, psate, docid, docname, docage, doclevel, docsalary, did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount)

So, this table has information about a patient (pid, pname, page, pstate), a doctor (docid, docname, docage, doclevel, docsalary) treating that patient, a disease (did, dname, dtype, description), and the room where the treatment (or procedure) is performed (rid, rname, rcapacity). Thus, each record contains all the information for one procedure (say, doing an examination, giving an injection or performing a minor surgery) for one patient, where we also store the ttime, tcost and the insurance discount of the treatment that was done. (The insurance discount is the amount paid by the insurance, so the customer has to pay the rest.) In addition, we assume that the following conditions hold:

(i) For each room, it can be used for only one particular type of disease.
(ii) All doctors at the same level (e.g., resident, junior, senior) must have the same salary.

Functional Dependencies:
pid -> pname, page, pstate
docid -> docname, docage, doclevel, docsalary
did -> dname, dtype, description
rid -> rname, rcapacity
rid -> dtype
doclevel -> docsalary
pid, ttime -> tcost, insurancediscount

a) Explain why the above is not a good relational design. Name several reasons!
Ans: The above relational design is not a good relational design for the below reasons:

- Current schema does not reflect a lot of functional dependencies. For instance, as per assumptions in the question, {pid -> pname, page, pstate}, {docid -> docname, docage, doclevel, docsalary} and other functional dependencies exist but since {pid} and {docid} are not candidate key, we will end up repeating this information in several tuples.
- Inserting values for the separate entities in one table like Patient, Doctor, Disease or Room, will lead to storing NULL and duplicate values at several places. This will increase the storage and our data will be inconsistent.
- Unique IDs: pid, docid, did are also redundant because same patient will not be able to get a treatment from the same doctor for the same disease if they are unique. For example, a patient with pid=10 got a treatment from doctor with docid = 11 for disease with did = 101 at ttime =’ 2016-11-1 10:00:00’ i.e. (10, 11, 101, 2016-11-1 10:00:00) then the scenario (10, 11, 101, 2016-11-1 10:30:00) is not possible as pid, docid, did have to be unique.
- We must access this one table for querying even minimal data, this will be time-consuming as everything is under this table.
- Lastly, this schema does not adhere to the normalization rules. As everything is under one table, it will make maintenance and querying from the database difficult and it will be hard to see relation between each attribute. The relational database system works better with several small tables than a big table.

We can avoid this by converting the above relation into BCNF and 3NF depending on our requirement.

b) Derive all candidate keys for this table.
Ans: (pid, docid, did, rid, ttime) is the candidate key.
Explanation:
(pid, docid, did, rid, ttime)+ i.e. Attribute Set Closure
Result = pid, docid, did, rid, ttime
Result = pid, docid, did, rid, ttime, pname, page, pstate (pid -> pname, page, pstate and pid $\subseteq$ pid, docid, did, rid, ttime)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary (docid -> docname, docage, doclevel, docsalary and docid $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity (rid -> rname, rcapacity and rid $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype (rid -> dtype and rid $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype, tcost, insurancediscount (pid, ttime -> tcost, insurancediscount and pid, ttime $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype, tcost, insurancediscount, dname, description (did -> dname, dtype, description and did $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary,rname, rcapacity,dtype, tcost, insurancediscount)

Therefore, (pid, docid, did, rid, ttime) is the candidate key as we can determine all the attributes mentioned in the relation PatientTreatment by it.

c) Identify the set F of non-trivial functional dependencies for this schema. (It is enough to identify a subset E such that the closures of E and F are the same.)
Ans:
F = {pid -> pname, page, pstate
docid -> docname, docage, doclevel, docsalary
did -> dname, dtype, description
rid -> rname, rcapacity
rid -> dtype
doclevel -> docsalary
pid, ttime -> tcost, insurancediscount}

d) Derive a canonical cover of the functional dependencies in F.
Ans:
1. Combining rid -> dtype and rid -> rname, rcapacity into rid -> dtype, rname, rcapacity
Set now is
{pid -> pname, page, pstate,
docid -> docname, docage, doclevel, docsalary,
did -> dname, dtype, description,
rid -> rname, rcapacity, dtype,
doclevel -> docsalary,
pid, ttime -> tcost, insurancediscount}
2. docsalary is an extraneous attribute in docid -> docname, docage, doclevel, docsalary
Set now is

{pid -> pname, page, pstate,
docid -> docname, docage, doclevel,
did -> dname, dtype, description,
rid -> rname, rcapacity,dtype,
doclevel -> docsalary,
pid, ttime -> tcost, insurancediscount}

Canonical Cover:
{pid -> pname, page, pstate,
docid -> docname, docage, doclevel,
did -> dname, dtype, description,
rid -> rname, rcapacity,dtype,
doclevel -> docsalary,
pid, ttime -> tcost, insurancediscount}

e) Is the above schema in BCNF? Prove or disprove. If it is not in BCNF, convert it into BCNF.
Ans:
1.PatientTreatment relation is not in BCNF (pid -> pname, page, pstate and pid is not the superkey)
On decomposition we get
Patient (pid, pname, page, pstate) and PatientTreatment-1 (pid, docid, docname, docage, doclevel, docsalary, did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount)

2.PatientTreatment-1 is not in BCNF (docid -> docname, docage, doclevel, docsalary and docid is not a superkey)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel, docsalary) and PatientTreatment-2 (pid, docid, did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount)

3. PatientTreatment-2 and Doctor are not in BCNF (did -> dname, dtype, description and did is not a superkey in PatientTreatment-2; doclevel -> docsalary and doclevel is not a superkey in Doctor)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description) and PatientTreatment-3 (pid, docid, did, rid, rname, rcapacity, ttime, tcost, insurancediscount)
4. PatientTreatment-3 is not in BCNF (rid -> rname, rcapacity and rid is not the super key)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description), Room (rid, rname, rcapacity) and PatientTreatment-4 (pid, docid, did, rid, ttime, tcost, insurancediscount)
5. PatientTreatment-4 is not in BCNF (pid, ttime -> tcost, insurancediscount and pid, ttime is not the superkey)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description), Room (rid, rname, rcapacity), Treatment (pid, ttime, tcost, insurancediscount) and PatientTreatment-5 (pid, docid, did, rid, ttime)

Therefore, we have converted relation PatientTreatment in BCNF by decomposing it into relations Patient, Doctor, DoctorSalary, Disease, Room, Treatment and PatientTreatment-5.

f) Is the BCNF schema from e) dependency-preserving? Prove or disprove. If not, convert it into 3NF.
Ans: The above BCNF schema is not dependency-preserving as the dependency rid -> dtype in the set F is not preserved in the schema.
Steps to convert PatientTreatment into 3NF:

1. Finding out the canonical cover of F

Canonical Cover {pid -> pname, page, pstate,
docid -> docname, docage, doclevel,
did -> dname, dtype, description,
rid -> rname, rcapacity,dtype,
doclevel -> docsalary,
pid, ttime -> tcost, insurancediscount}

2. Generate the following 3NF schema from the canonical cover

Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description), Room (rid, rname, rcapacity, dtype), Treatment (pid, ttime, tcost, insurancediscount)

3. If any of the above schema doesn't contain the candidate key i.e. (pid, docid, did, rid, ttime) in this case create a new relation containing it.

Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description), Room (rid, rname, rcapacity, dtype), Treatment (pid, ttime, tcost, insurancediscount) and PatientTreatment-5 (pid, docid, did, rid, ttime)

4. Since there are no redundant schemas, no schema will be deleted.

Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description), Room (rid, rname, rcapacity, dtype), Treatment (pid, ttime, tcost, insurancediscount) and PatientTreatment-5 (pid, docid, did, rid, ttime)

Therefore, we have converted relation PatientTreatment in 3NF by decomposing it into relations Patient, Doctor, DoctorSalary, Disease, Room, Treatment and PatientTreatment-5.

g) Suppose we add an additional constraint that for any two patients who live in the same state and spend the same tcost for a treatment (or procedure), the insurance discount should also be the same. How would this change your answers to parts (b) to (f)?
Ans:
By adding the additional constraint, the set of functional dependencies is as follows:
F = {pid -> pname, page, pstate
docid -> docname, docage, doclevel, docsalary
did -> dname, dtype, description
rid -> rname, rcapacity
rid -> dtype
doclevel -> docsalary
pid, ttime -> tcost, insurancediscount
pstate, tcost -> insurancediscount}

b.
Candidate Key will remain the same i.e. (pid, docid, did, rid, ttime).
Explanation:
(pid, docid, did, rid, ttime)$^+$ i.e. Attribute Set Closure
Result = pid, docid, did, rid, ttime

Result = pid, docid, did, rid, ttime, pname, page, pstate (pid -> pname, page, pstate and pid $\subseteq$ pid, docid, did, rid, ttime)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary (docid -> docname, docage, doclevel, docsalary and docid $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity (rid -> rname, rcapacity and rid $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype (rid -> dtype and rid $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype, tcost, insurancediscount (pid, ttime -> tcost, insurancediscount and pid, ttime $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype)
Result = pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary, rname, rcapacity,dtype, tcost, insurancediscount, dname, description (did -> dname, dtype, description and did $\subseteq$ pid, docid, did, rid, ttime, pname, page, pstate, docname, docage, doclevel, docsalary,rname, rcapacity,dtype, tcost, insurancediscount)

Therefore, (pid, docid, did, rid, ttime) is the candidate key as we can determine all the attributes mentioned in the relation PatientTreatment by it.

c.
F = {pid -> pname, page, pstate
docid -> docname, docage, doclevel, docsalary
did -> dname, dtype, description
rid -> rname, rcapacity
rid -> dtype
doclevel -> docsalary
pid, ttime -> tcost, insurancediscount
pstate, tcost -> insurancediscount}

d.
Canonical Form:
1. Combining rid -> dtype and rid -> rname, rcapacity into rid -> dtype, rname, rcapacity
Set now is
{pid -> pname, page, pstate,
docid -> docname, docage, doclevel, docsalary,
did -> dname, dtype, description,
rid -> rname, rcapacity, dtype,
doclevel -> docsalary,
pid, ttime -> tcost, insurancediscount
pstate, tcost -> insurancediscount}
2. docsalary is an extraneous attribute in docid -> docname, docage, doclevel, docsalary
Set now is
{pid -> pname, page, pstate,
docid -> docname, docage, doclevel,
did -> dname, dtype, description,
rid -> rname, rcapacity,dtype,
doclevel -> docsalary,
pid, ttime -> tcost, insurancediscount

pstate, tcost -> insurancediscount}

Canonical Cover:
{pid -> pname, page, pstate,
docid -> docname, docage, doclevel,
did -> dname, dtype, description,
rid -> rname, rcapacity,dtype,
doclevel -> docsalary,
pid, ttime -> tcost, insurancediscount
pstate, tcost -> insurancediscount}

e.
BCNF:

1.PatientTreatment relation is not in BCNF (pid -> pname, page, pstate and pid is not the superkey)
On decomposition we get
Patient (pid, pname, page, pstate) and PatientTreatment-1 (pid, docid, docname, docage, doclevel, docsalary, did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount)

2.PatientTreatment-1 is not in BCNF (docid -> docname, docage, doclevel, docsalary and docid is not a superkey)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel, docsalary) and PatientTreatment-2 (pid, docid, did, dname, dtype, description, rid, rname, rcapacity, ttime, tcost, insurancediscount)

3. PatientTreatment-2 and Doctor are not in BCNF (did -> dname, dtype, description and did is not a superkey in PatientTreatment-2; doclevel -> docsalary and doclevel is not a superkey in Doctor)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description) and PatientTreatment-3 (pid, docid, did, rid, rname, rcapacity, ttime, tcost, insurancediscount)
4. PatientTreatment-3 is not in BCNF (rid -> rname, rcapacity and rid is not the super key)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description), Room (rid, rname, rcapacity) and PatientTreatment-4 (pid, docid, did, rid, ttime, tcost, insurancediscount)
5. PatientTreatment-4 is not in BCNF (pid, ttime -> tcost, insurancediscount and pid, ttime is not the superkey)
On decomposition we get
Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel), DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description), Room (rid, rname, rcapacity), Treatment (pid, ttime, tcost, insurancediscount) and PatientTreatment-5 (pid, docid, did, rid, ttime)

Therefore, we have converted relation PatientTreatment in BCNF by decomposing it into relations Patient, Doctor, DoctorSalary, Disease, Room, Treatment and PatientTreatment-5.

f.

3NF

The above BCNF schema is not dependency-preserving as the functional dependencies rid -> dtype and pstate, tcost -> insurancediscount are not held by the schema.

Steps to convert PatientTreatment relation into 3NF:

1. Finding out the canonical cover of F
   Canonical Cover:
   {pid -> pname, page, pstate,
   docid -> docname, docage, doclevel,
   did -> dname, dtype, description,
   rid -> rname, rcapacity,dtype,
   doclevel -> docsalary,
   pid, ttime -> tcost, insurancediscount
   pstate, tcost -> insurancediscount}

2. Generate the following 3NF schema from the canonical cover
   Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel),
   DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description),
   Room (rid, rname, rcapacity, dtype), Treatment (pid, ttime, tcost, insurancediscount)
   and Insurance (pstate, tcost, insurancediscount)

3. If any of the above schema doesn't contain the candidate key i.e. (pid, docid, did, rid, ttime) in this case create a new relation containing it.
   Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel),
   DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description),
   Room (rid, rname, rcapacity, dtype), Treatment (pid, ttime, tcost, insurancediscount)
   and Insurance (pstate, tcost, insurancediscount) and PatientTreatment-6(pid, docid, did, rid, ttime)

4. Since there are no redundant schemas, no schema will be deleted.
   Patient (pid, pname, page, pstate), Doctor (docid, docname, docage, doclevel),
   DoctorSalary (doclevel, docsalary), Disease (did, dname, dtype, description),
   Room (rid, rname, rcapacity, dtype), Treatment (pid, ttime, tcost, insurancediscount)
   and Insurance (pstate, tcost, insurancediscount) and PatientTreatment-6(pid, docid, did, rid, ttime)

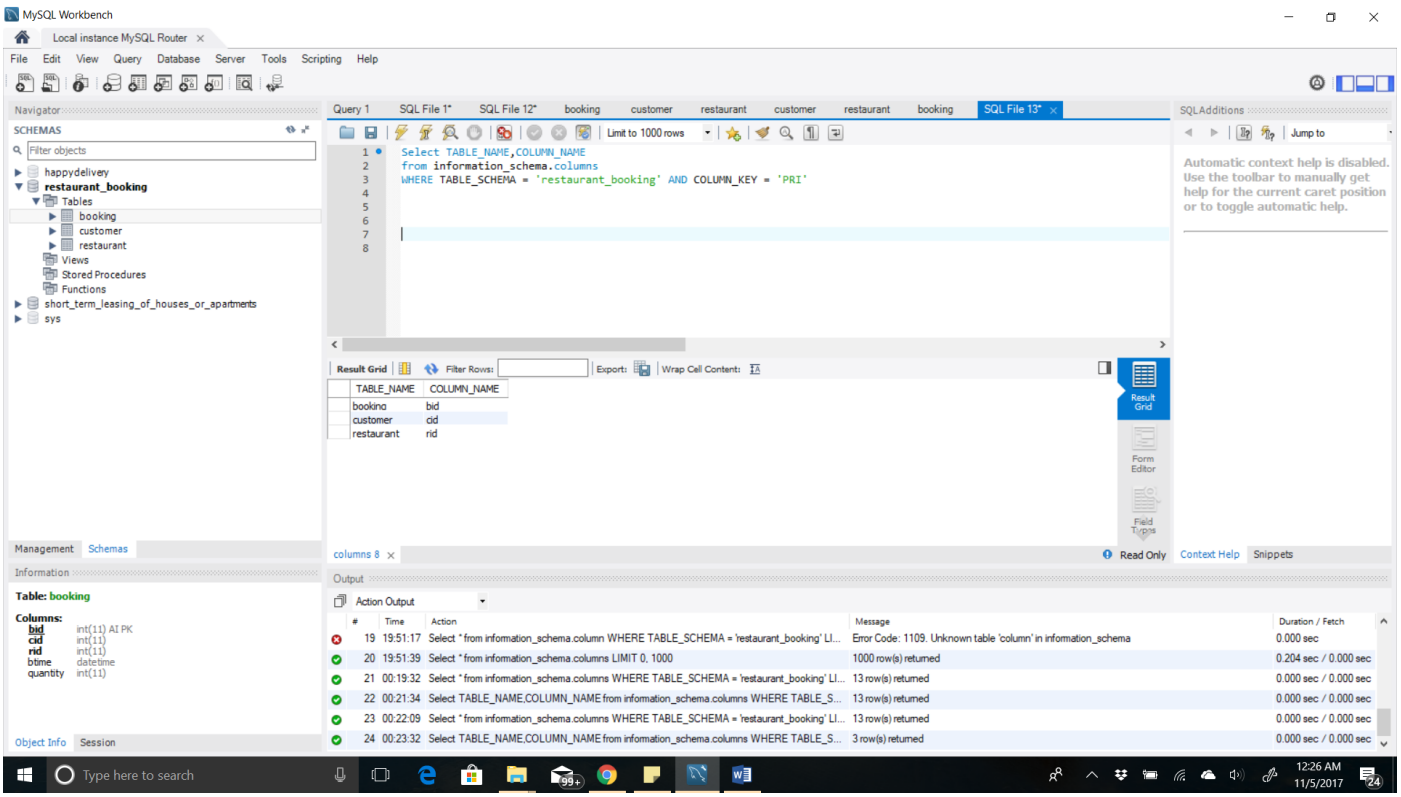
Therefore, we have converted relation PatientTreatment in 3NF by decomposing it into relations Patient, Doctor, DoctorSalary, Disease, Room, Treatment, Insurance and PatientTreatment-6

Problem 4:

In this problem, you are asked to explore the metadata querying facilities in your database system. Thus, the answers may depend on which system you have installed. So, remember to state which system you are using! Try to write the following queries using the restaurant booking database from Problem 1:

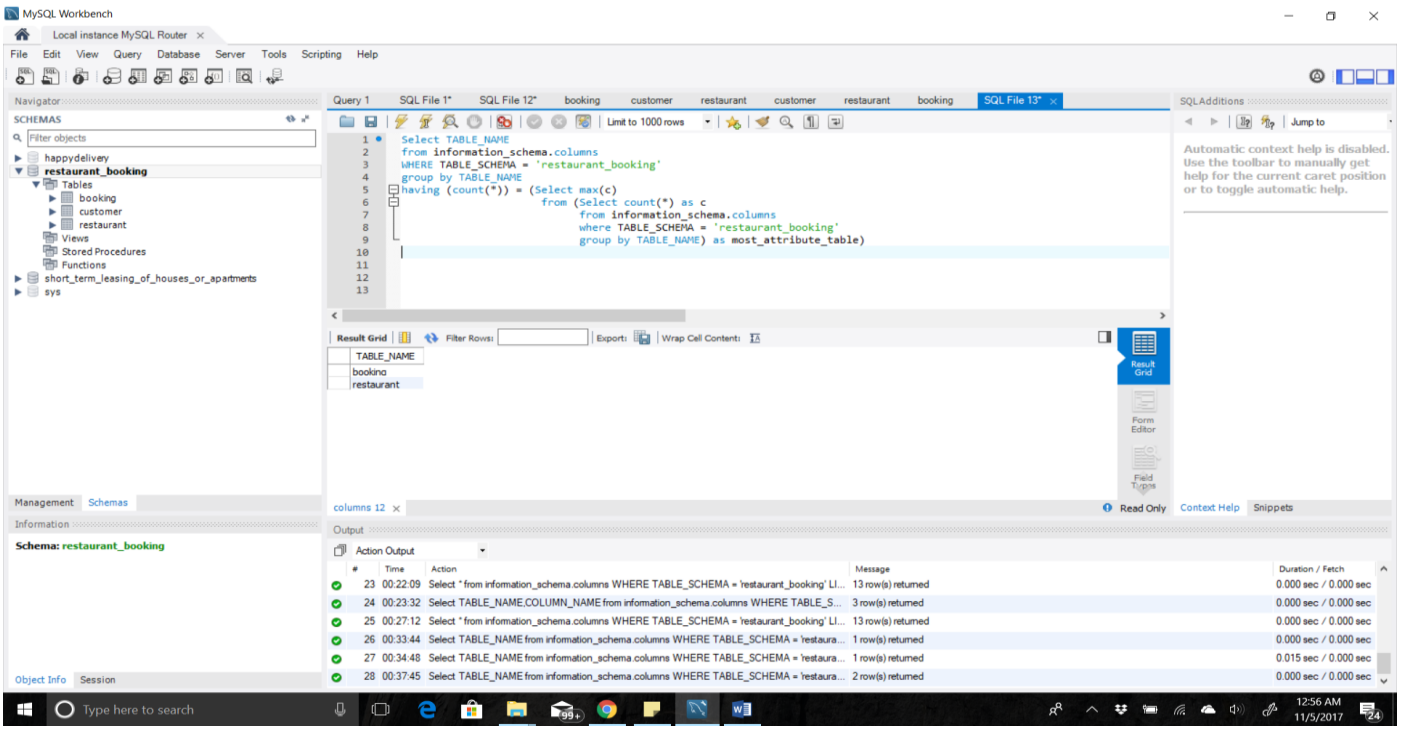
a. List all tables in the schema, and the attribute names of their primary keys.

Ans:
```
Select TABLE_NAME, COLUMN_NAME
from information_schema.columns
WHERE TABLE_SCHEMA = 'restaurant_booking' AND COLUMN_KEY = 'PRI'
```



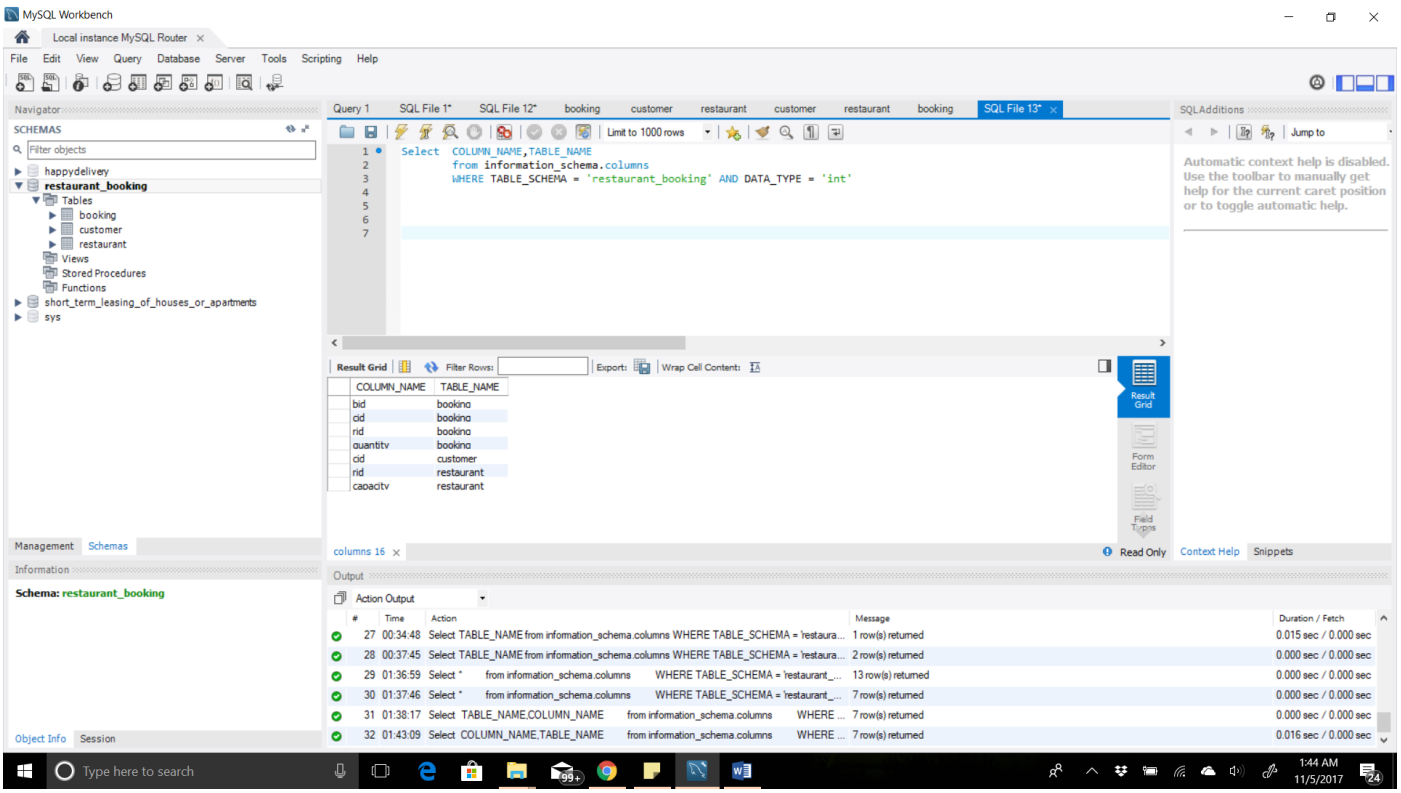
b. List the table with the most attributes.

Ans:
```
Select TABLE_NAME
    from information_schema.columns
    WHERE TABLE_SCHEMA = 'restaurant_booking'
    group by TABLE_NAME
    having (count(*)) = (Select max(c)
                from (Select count(*) as c
                    from information_schema.columns
                    where TABLE_SCHEMA = 'restaurant_booking'
                    group by TABLE_NAME) as most_attribute_table)
```



c. List the attribute name and table name of any attribute of type int.

Ans: Select COLUMN_NAME,TABLE_NAME
from information_schema.columns
WHERE TABLE_SCHEMA = 'restaurant_booking' AND DATA_TYPE = 'int'



d. List all pairs of tables that have an attribute with the same name and same data type.
Ans: Select A.TABLE_NAME,B.TABLE_NAME
from information_schema.COLUMNS A,information_schema.COLUMNS B
where A.COLUMN_NAME = B.COLUMN_NAME AND A.DATA_TYPE = B.DATA_TYPE AND A.TABLE_NAME < B.TABLE_NAME AND A.TABLE_SCHEMA = 'restaurant_booking' AND B.TABLE_SCHEMA = 'restaurant_booking'