

Department of Computer Science &  
Engineering

University of Dhaka

Database System Lab Assignment

Course code – 2211

2<sup>nd</sup> year 2<sup>nd</sup> semester, 2016

Submitted to: Mr. Abu Ahmed Ferdaus,  
Associate Professor, Department of Computer  
Science and Engineering

Submitted by: H.M. Jameul Haque Sarker

Roll. 30

Batch-21

## Task to do:

Consider a relational database model of your own and go for the following tasks.

1. Consider the database of at least 4 schemas/tables.
2. Plan for attributes that cover all general datatypes.
3. Plan for constraints of different types (primary keys, foreign keys, unique keys, check, not null etc.) with appropriate **constraint names**.
4. Create appropriate data for the above schemas.
5. Implement the database in Oracle 10g Express Edition creating a user.
6. Plan queries and find the answers (at least 10).
7. Formulate everything in a report which includes:
  - a. Brief description of the system that you are going to implement in the database
  - b. Mention schemas with attributes
  - c. Expected **functional dependencies** on your schemas.
  - d. Proof of good database design so that the schemas are in **good form** (BCNF or 3NF)
  - e. **Schema diagram** of the database
  - f. **E-R diagram** of the database
  - g. Snapshots of **SQL DDL** of all the schemas/tables
  - h. Snapshots of the **instances** (data of the populated tables)
  - I. **Query statements in English language, SQL statements** and snapshots of the **outputs**. SQL statements **should** contain the following:
    - i. **natural join, cross product, outer join, join with using, on**
    - ii. nested sub-queries with clauses (**some, all, any, exists, unique** etc.)
    - iii. **order by, group by, having** clauses
    - iv. Use of **with** clause
    - v. Use of Functions (**string/text, numeric**) , set operations (**union, intersect, difference/minus**)
    - vi. **Update, delete** operations
  - j. Conclusion of the work

## Overview Of the Database:

The project is about to make a database of our own and apply different queries on that database. I decided to make a database of a House renting Agency.

In that database system, I made 7 tables to implement the system. Among 7 tables 3 of them are independent. The tables are – ‘Branch’, ‘Client’, ‘Owner’, ‘Registration’, ‘Staff’, ‘House\_for\_rent’, ‘Visiting’. ‘Branch’, ‘Client’, and ‘Owner’ are independent tables. Information of the client, branch and owner are kept in those 3 tables respectively. And another four tables depend on this three. ‘Staff’ table depends on ‘Branch’ table because a staff can’t be created without a branch. So to relate that relation staff must be depended on the ‘branch’ table. Again the ‘House\_for\_rent’ table is dependent on ‘Owner’ and ‘Branch’ table because this table gives the information of - which house belongs with which owner and which branch. ‘Registration’ table is depends on ‘Client’ and ‘Branch’ tables because these table holds the information where a client is registered. ‘Visiting’ table is depends on ‘Client’, ‘Staff’ and ‘House\_for\_rent’ table because this table gives us the information about which client by which staff visited which house.

Thus I tried to implement a database system of a House renting agency.

## Schemas with attributes:

As I mentioned earlier here I made 7 schemas which are –

1. Branch
2. Client
3. Owner

4. Staff
5. Registration
6. House\_for\_rent
7. Visiting

Branch table has 6 attributes to keep the branch information – Branch\_id, Zila, Upazila, Thana, Location, Postcode.

The schema for 'Branch' relation is-

Branch(branch\_id, Zila, Upazila, Thana, Location, Postcode)

Client relation has 5 attributes. The schema of 'Client' relation is-

Client (client\_id, name, phone, house\_type, max\_rent)

To keep owner information I have made 'Owner' relation. Owner relation has 4 attributes. The schema of 'Owner' relation is-

Owner (Owner\_id, Owner\_name, address, mobile)

Staff relation has 7 attributes. Every staff works under a branch. The schema of 'Staff' relation is-

Staff (Staff\_id, Branch\_id, name, post, gender, dob, salary)

Registration relation has 4 attributes. Every client is registered under a branch. The schema of 'Registration' relation is-

Registration (client\_id, branch\_id, staff\_id, date\_joined)

House\_for\_rent relation has 9 attributes. Every house has an owner and some qualities. The schema of 'House\_for\_rent' relation is-

House\_for\_rent (house\_id, owner\_id, branch\_id, type, living\_rooms, bath\_rooms, kitchen, dining\_rooms, rent)

Visiting relation has 7 attributes. The schema of 'Visiting' relation is-

Visiting (client\_id, House\_id, staff\_id, visiting\_date, status, comments, confirmation)

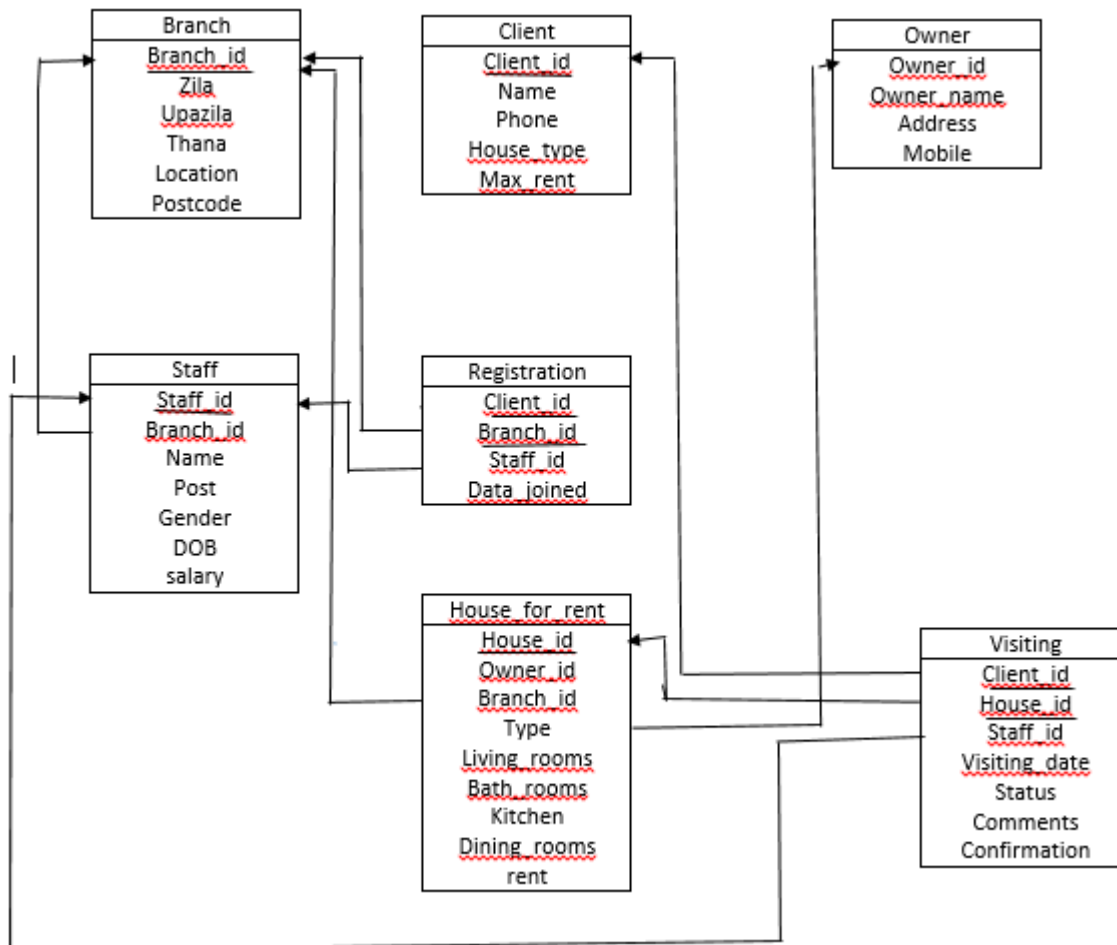
So I use following schemas in this database system –

- Branch (branch\_id, Zila, Upazila, Thana, Location, Postcode)
- Client (client\_id, name, phone, house\_type, max\_rent)
- Owner (Owner\_id, Owner\_name, address, mobile)
- Staff (Staff\_id, Branch\_id, name, post, gender, dob, salary)
- Registration (client\_id, branch\_id, staff\_id, date\_joined)
- House\_for\_rent (house\_id, owner\_id, branch\_id, type, living\_rooms, bath\_rooms, kitchen, dining\_rooms, rent)
- Visiting (client\_id, House\_id, staff\_id, visiting\_date, status, comments, confirmation)

Schema Diagram:

A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams. Figure shows the schema diagram for the House renting agency. Each relation appears as a box, with the relation name at the top, and the attributes listed inside the box. Primary key attributes are shown underlined. Foreign key dependencies appear as

arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation



Snapshot of SQL DDL:

We define an SQL relation by using the create table command. The following command creates a relation branch in the database.

## Branch relation:

```
CREATE TABLE "BRANCH"  
(  
    "BRANCH_ID" VARCHAR2(5),  
    "ZILA" VARCHAR2(15),  
    "UPAZILA" VARCHAR2(25),  
    "THANA" VARCHAR2(25),  
    "LOCATION" VARCHAR2(35),  
    "POSTCODE" VARCHAR2(10),  
    PRIMARY KEY ("BRANCH_ID") ENABLE,  
    CONSTRAINT "BRANCH_CON" CHECK ( "BRANCH_ID" like 'B-%') ENABLE  
)
```

## Client relation:

```
CREATE TABLE "CLIENT"  
(  
    "CLIENT_ID" VARCHAR2(5),  
    "NAME" VARCHAR2(20),  
    "PHONE" VARCHAR2(12),  
    "HOUSE_TYPE" VARCHAR2(10),  
    "MAX_RENT" NUMBER(10,2),  
    PRIMARY KEY ("CLIENT_ID") ENABLE,  
    CONSTRAINT "CLIENT_CON" CHECK ( "CLIENT_ID" like 'C-%') ENABLE  
)
```

## Owner relation:

```
CREATE TABLE "OWNER"  
(  
    "OWNER_ID" VARCHAR2(5),  
    "OWNER_NAME" VARCHAR2(15),  
    "ADDRESS" VARCHAR2(40),  
    "MOBILE" VARCHAR2(12),  
    PRIMARY KEY ("OWNER_ID") ENABLE,  
    CONSTRAINT "OWNER_CON" CHECK ( "OWNER_ID" like 'O-%') ENABLE  
)
```

## Staff relation:

```
CREATE TABLE "STAFF"  
(  
    "STAFF_ID" VARCHAR2(5),  
    "BRANCH_ID" VARCHAR2(5),  
    "NAME" VARCHAR2(15),  
    "POST" VARCHAR2(15),  
    "GENDER" VARCHAR2(5),  
    "DOB" DATE,  
    "SALARY" NUMBER(10,2),  
    PRIMARY KEY ("STAFF_ID") ENABLE,  
    CONSTRAINT "STAFF_CON" CHECK ( "STAFF_ID" like 'S-%') ENABLE,  
    FOREIGN KEY ("BRANCH_ID")  
        REFERENCES "BRANCH" ("BRANCH_ID") ON DELETE CASCADE ENABLE  
)
```

## Registration relation:

```
CREATE TABLE "REGISTRATION"  
(  
    "CLIENT_ID" VARCHAR2(5),  
    "BRANCH_ID" VARCHAR2(5),  
    "STAFF_ID" VARCHAR2(5),  
    "DATE_JOINED" DATE,  
    PRIMARY KEY ("CLIENT_ID", "BRANCH_ID") ENABLE,  
    FOREIGN KEY ("CLIENT_ID")  
        REFERENCES "CLIENT" ("CLIENT_ID") ON DELETE CASCADE ENABLE,  
    FOREIGN KEY ("BRANCH_ID")  
        REFERENCES "BRANCH" ("BRANCH_ID") ON DELETE CASCADE ENABLE,  
    FOREIGN KEY ("STAFF_ID")  
        REFERENCES "STAFF" ("STAFF_ID") ON DELETE CASCADE ENABLE  
)
```



## House\_for\_rent relation:

```
CREATE TABLE "HOUSE_FOR_RENT"
(
  "HOUSE_ID" VARCHAR2(5),
  "OWNER_ID" VARCHAR2(5),
  "BRANCH_ID" VARCHAR2(5),
  "TYPE" VARCHAR2(10),
  "LIVING_ROOMS" NUMBER(4,0),
  "BATH_ROOMS" NUMBER(2,0),
  "KITCHEN" NUMBER(2,0),
  "DINING_ROOMS" NUMBER(2,0),
  "RENT" NUMBER(10,0),
  PRIMARY KEY ("HOUSE_ID") ENABLE,
  CONSTRAINT "HOUSE_FOR_RENT_CON" CHECK ( "RENT" is not null) ENABLE,
  CONSTRAINT "HOUSE_FOR_RENT_CON1" CHECK ( "HOUSE_ID" like 'H-%') ENABLE,
  FOREIGN KEY ("OWNER_ID")
    REFERENCES "OWNER" ("OWNER_ID") ON DELETE CASCADE ENABLE,
  FOREIGN KEY ("BRANCH_ID")
    REFERENCES "BRANCH" ("BRANCH_ID") ON DELETE CASCADE ENABLE
)
```









## Visiting relation:

```
CREATE TABLE "VISITING"
(
  "CLIENT_ID" VARCHAR2(5),
  "HOUSE_ID" VARCHAR2(5),
  "STAFF_ID" VARCHAR2(5),
  "VISITING_DATE" DATE,
  "STATUS" VARCHAR2(10),
  "COMMENTS" VARCHAR2(20),
  "CONFIRMATION" VARCHAR2(4),
  PRIMARY KEY ("CLIENT_ID", "HOUSE_ID") ENABLE,
  CONSTRAINT "VISITING_CON" CHECK ( "CONFIRMATION" in ('Yes','No')) ENABLE,
  FOREIGN KEY ("CLIENT_ID")
    REFERENCES "CLIENT" ("CLIENT_ID") ON DELETE CASCADE ENABLE,
  FOREIGN KEY ("HOUSE_ID")
    REFERENCES "HOUSE_FOR_RENT" ("HOUSE_ID") ON DELETE CASCADE ENABLE,
  FOREIGN KEY ("STAFF_ID")
    REFERENCES "STAFF" ("STAFF_ID") ON DELETE CASCADE ENABLE
)
```

## 1. Snapshot of instances:

Here I will show the table with data.

Branch:

EDIT	BRANCH_ID	ZILA	UPAZILA	THANA	LOCATION	POSTCODE
	B-001	Gazipur	Gazipur Sadar	Gazipur Sadar Thana	Majeda Complex,Joydebpur	1700
	B-002	Dhaka	Dhanmondi	Dhanmondi Thana	Gausia Market Level-3	1209
	B-003	Dhaka	Gulshan	Gulshan Thana	Future Tower,Level-5	1212
	B-004	Dhaka	Mohammadpur	Mohammadpur Thana	Rose bela complex Level-1	1207
	B-008	Dhaka	Lalbag	Lalbag Thana	modhopara,lalbag	1234
	B-005	Jamalpur	Bakshiganj	Bakshiganj Thana	Jamalpur new market,Level-5	2000
	B-006	Dhaka	Uttara	Uttara Thana	Raju compex, Level-9	1230
	B-007	Dhaka	Savar	Asulia Thana	Bridgemore,savar	1340
row(s) 1 - 8 of 8						

Client:

EDIT	CLIENT_ID	NAME	PHONE	HOUSE_TYPE	MAX_RENT
	C-001	Md.Al-helel	01718067627	Flat	12000
	C-002	Md.Imran Arafat	01911439089	Flat	9000
	C-003	Md.Tariq Hasan	01678678790	House	5000
	C-004	Md.Rashid Ahmed	01836221219	House	5500
	C-005	Md.Soikot	01521254432	Flat	10000
	C-006	Nur Alam	01937890765	Flat	9500
	C-007	Jotirmoy Nug	01750785843	House	4500
	C-008	Tutul	01543217890	Flat	15000

Owner:

EDIT	OWNER_ID	OWNER_NAME	ADDRESS	MOBILE
	O-001	Jishan	Choker goli,Joydebpur,Gazipur	1718076756
	O-002	Sakib	Munsipara,Dhanmondi,Dhaka	1911435678
	O-003	Dipto	Choudhri coloni, lane no.4,Gulsion,Dhaka	1836456789
	O-004	Samim	Town hall,Lane no.2,Mohmmadpur,Dhaka	1678789056
	O-005	Samin	Bardhankuti,Bakshiganj,Jamalpur	1521254637
	O-006	Imjamam	New Model Town,Ultra,Dhaka	1734216708
	O-007	Faruk	Baharipara,Mohmmadpur,Dhaka	1521678954
row(s) 1 - 7 of 7				









Staff:

EDIT	STAFF_ID	BRANCH_ID	NAME	POST	GENDER	DOB	SALARY
	S-M21	B-001	Riddho	Manager	M	01-OCT-90	30000
	S-A09	B-005	Saklian Jamal	Assistant	M	02-FEB-91	12000
	S-M05	B-002	Suvo ahmed	Manager	M	03-NOV-89	24000
	S-S14	B-001	Taquir Ahmed	Supervisor	M	04-JUN-91	24000
	S-A07	B-003	Noboni	Assistant	F	13-FEB-93	18000
	S-M22	B-004	Nahinan Asraf	Manager	F	26-MAR-91	28000
	S-A10	B-006	Imran Arafat	Assistant	M	25-NOV-91	18000
	S-A13	B-002	Tahsin Al-Sayed	Assistant	M	13-JUN-88	16000
	S-A11	B-007	Sohan	Assistant	M	13-NOV-90	16000
row(s) 1 - 9 of 9							







## Registration:

EDIT	CLIENT_ID	BRANCH_ID	STAFF_ID	DATE_JOINED
	C-001	B-001	S-M21	04-FEB-15
	C-002	B-002	S-A13	02-NOV-15
	C-003	B-003	S-A07	03-JUN-16
	C-004	B-004	S-M22	01-OCT-16
	C-005	B-005	S-A09	01-NOV-16
	C-006	B-006	S-A10	02-NOV-16
	C-007	B-006	S-A10	03-OCT-16
	C-008	B-007	S-A11	28-NOV-16
row(s) 1 - 8 of 8				

## House\_for\_rent:

EDIT	HOUSE_ID	OWNER_ID	BRANCH_ID	TYPE	LIVING_ROOMS	BATH_ROOMS	KITCHEN	DINING_ROOMS	RENT
	H-001	O-001	B-001	Flat	3	3	2	2	11500
	H-002	O-002	B-002	Flat	3	2	1	1	8500
	H-003	O-003	B-003	House	4	1	-	-	5000
	H-004	O-004	B-004	House	4	1	1	-	5500
	H-005	O-005	B-005	Flat	3	2	2	1	10000
	H-006	O-006	B-006	Flat	4	2	1	1	9500
	H-007	O-001	B-001	Flat	3	3	2	2	12000
	H-008	O-007	B-006	House	2	1	1	-	4500

Visiting:

EDIT	CLIENT_ID	HOUSE_ID	STAFF_ID	VISITING_DATE	STATUS	COMMENTS	CONFIRMATION
	C-002	H-002	S-A13	25-NOV-16	Visited	Good	Yes
	C-003	H-003	S-A07	30-NOV-16	Pending	-	-
	C-004	H-004	S-M22	30-OCT-16	Visited	Too remote	No
	C-005	H-005	S-A09	31-OCT-16	Visited	Nice	Yes
	C-006	H-006	S-A10	29-NOV-16	Pending	-	-
	C-001	H-007	S-S14	30-NOV-16	Pending	-	-
row(s) 1 - 6 of 6							

## 2. Query and SQL Statements

Natural join:

Find the registration information in branch 'B-006'?

SQL: select branch\_id,client\_id,staff\_id,date\_joined

from branch natural join registration

Where branch\_id='B-006'

RA:  $\Pi_{branch\_id, client\_id, staff\_id, date\_joined} (\sigma_{branch\_id='B-006'} (branch \bowtie registration))$

BRANCH_ID	CLIENT_ID	STAFF_ID	DATE_JOINED
B-006	C-006	S-A10	02-NOV-16
B-006	C-007	S-A10	03-OCT-16



Left outer join: Find every branch where a client may be registered or not.

SQL: select branch\_id,client\_id,staff\_id,date\_joined  
from branch natural left outer join registration

RA:  $\Pi_{\text{branch\_id,client\_id,staff\_id,date\_joined}}(\text{branch} \bowtie \text{registration})$

BRANCH_ID	CLIENT_ID	STAFF_ID	DATE_JOINED
B-001	C-001	S-M21	04-FEB-15
B-002	C-002	S-A13	02-NOV-15
B-003	C-003	S-A07	03-JUN-16
B-004	C-004	S-M22	01-OCT-16
B-005	C-005	S-A09	01-NOV-16
B-006	C-006	S-A10	02-NOV-16
B-006	C-007	S-A10	03-OCT-16
B-007	C-008	S-A11	28-NOV-16
B-008	-	-	-

Right outer join: Find those branches where minimum a client is registered.

SQL: select branch\_id,client\_id,staff\_id,date\_joined  
from branch natural right outer join registration

RA:  $\Pi_{\text{branch\_id,client\_id,staff\_id,date\_joined}}(\text{branch} \bowtie \text{registration})$

BRANCH_ID	CLIENT_ID	STAFF_ID	DATE_JOINED
B-001	C-001	S-M21	04-FEB-15
B-002	C-002	S-A13	02-NOV-15
B-003	C-003	S-A07	03-JUN-16
B-004	C-004	S-M22	01-OCT-16
B-005	C-005	S-A09	01-NOV-16
B-006	C-006	S-A10	02-NOV-16
B-006	C-007	S-A10	03-OCT-16
B-007	C-008	S-A11	28-NOV-16

Full outer join: Find every branch and client information.

SQL: select \*

from branch natural full outer join registration

RA:  $\Pi(\text{branch} \bowtie \text{registration})$

BRANCH_ID	ZILA	UPAZILA	THANA	LOCATION	POSTCODE	CLIENT_ID	STAFF_ID	DATE_JOINED
B-001	Gazipur	Gazipur Sadar	Gazipur Sadar Thana	Majeda Complex,Joydebpur	1700	C-001	S-M21	04-FEB-15
B-002	Dhaka	Dhanmondi	Dhanmondi Thana	Gausia Market Level-3	1209	C-002	S-A13	02-NOV-15
B-003	Dhaka	Gulshan	Gulshan Thana	Future Tower,Level-5.	1212	C-003	S-A07	03-JUN-16
B-004	Dhaka	Mohammadpur	Mohammadpur Thana	Rose bela complex Level-1	1207	C-004	S-M22	01-OCT-16
B-005	Jamalpur	Bakshiganj	Bakshiganj Thana	Jamalpur new market,Level-5	2000	C-005	S-A09	01-NOV-16
B-006	Dhaka	Uttara	Uttara Thana	Raju compex, Level-9	1230	C-006	S-A10	02-NOV-16
B-006	Dhaka	Uttara	Uttara Thana	Raju compex, Level-9	1230	C-007	S-A10	03-OCT-16
B-007	Dhaka	Savar	Asulia Thana	Bridgemore,savar	1340	C-008	S-A11	28-NOV-16
B-008	Dhaka	Lalbag	Lalbag Thana	modhopara,lalbag	1234	-	-	-

Nested-subquery with clauses:

All clause:

Find those clients, whose visiting date is not submitted yet.

SQL:       select client\_id  
               from Client  
               where client\_id != all(select client\_id  
                                       from visiting)

RA:

$t_1 \leftarrow \Pi_{\text{client\_id}}(\text{client})$   
 $t_2 \leftarrow \Pi_{\text{client\_id}}(\text{visiting})$   
 $t_3 \leftarrow \Pi_{\text{client\_id}}(t_1 \cdot t_2)$   
 $\Pi_{\text{client\_id}}(t_3)$

CLIENT_ID
C-007
C-008

Any clause: Find those clients, whose visiting date at least submitted.

SQL:

```
select client_id
from Client
where client_id = any(select client_id
from visiting)
```

RA:

$t_1 \leftarrow \Pi_{\text{client\_id}}(\text{client})$   
 $t_2 \leftarrow \Pi_{\text{client\_id}}(\text{visiting})$   
 $t_3 \leftarrow \Pi_{\text{client\_id}}(t_1 \cap t_2)$   
 $\Pi_{\text{client\_id}}(t_3)$

CLIENT_ID
C-001
C-002
C-003
C-004
C-005
C-006



Not in clause: Find the information of those clients having no house for rent.

SQL:

```
select*  
  
from client  
  
where client_id not in (select client_id from visiting  
  
where confirmation='Yes')
```

RA:

```
t1←Π client_id(client)  
t2←Π client_id(σ confirmation="Yes" (visiting))  
t3←Π client_id(t1 - t2)  
Π (σ client.client_id= t3.client_id(client × t3))
```

CLIENT_ID	NAME	PHONE	HOUSE_TYPE	MAX_RENT
C-001	Md.Al-helel	01718067627	Flat	12000
C-003	Md.Tariq Hasan	01678678790	House	5000
C-004	Md.Rashid Ahmed	01836221219	House	5500
C-006	Nur Alam	01937890765	Flat	9500
C-007	Jotirmoy Nug	01750785843	House	4500
C-008	Tutul	01543217890	Flat	15000

Group by and having clauses:

Find the details of those branches which has most employees

SQL:

```

select *
from branch
where branch_id in (select branch_id
from staff group by branch_id
having count(staff_id) >= all( select count(staff_id)
from staff group by branch_id))

```

RA:

```

t1 ← branch_id G count(staff_id) (Staff)
t2 ← G max (count(staff_id)) as max_num (t1)
t3 ← Π branch_id (σ count(staff_id) = max_num(t1 × t2))
Π branch_id,zila,upazila,thana,location,postcode(σ branch.branch_id=
t3.branch_id(branch × t3))

```

BRANCH_ID	ZILA	UPAZILA	THANA	LOCATION	POSTCODE
B-001	Gazipur	Gazipur Sadar	Gazipur Sadar Thana	Majeda Complex,Joydebpur	1700
B-002	Dhaka	Dhanmondi	Dhanmondi Thana	Gausia Market Level-3	1209

Order by clause: Find the employee information order by gender (desc) and name (asc).

SQL:

```

select *
from staff
order by gender desc,name

```

RA:

$\psi$  = Order by

$\Pi (\psi_{\text{gender}}(\text{desc}), \text{name}(\text{staff}))$

STAFF_ID	BRANCH_ID	NAME	POST	GENDER	DOB	SALARY
S-A10	B-006	Imran Arafat	Assistant	M	25-NOV-91	18000
S-M21	B-001	Riddho	Manager	M	01-OCT-90	30000
S-A09	B-005	Saklian Jamal	Assistant	M	02-FEB-91	12000
S-A11	B-007	Sohan	Assistant	M	13-NOV-90	16000
S-M05	B-002	Suvo ahmed	Manager	M	03-NOV-89	24000
S-A13	B-002	Tahsin Al-Sayed	Assistant	M	13-JUN-88	16000
S-S14	B-001	Taquir Ahmed	Supervisor	M	04-JUN-91	24000
S-M22	B-004	Nahinan Asraf	Manager	F	26-MAR-91	28000
S-A07	B-003	Noboni	Assistant	F	13-FEB-93	18000

String operation: Find the information of those client having grammenphone cell number.

SQL:

```
select *
from client
where phone like '017%'
```

RA:

$\Pi(\sigma_{\text{phone LIKE ("017\%"})}(\text{client}))$

CLIENT_ID	NAME	PHONE	HOUSE_TYPE	MAX_RENT
C-001	Md.Al-helel	01718067627	Flat	12000
C-007	Jotirmoy Nug	01750785843	House	4500

Set operation:

Intersect:

SQL: Find those clients, whose visiting date at least submitted.

```

select client_id
from Client
intersect
(select client_id
from visiting)

```

RA:

```

 $t_1 \leftarrow \Pi_{client\_id}(client)$ 
 $t_2 \leftarrow \Pi_{client\_id}(visiting)$ 
 $t_3 \leftarrow \Pi_{client\_id}(t_1 \cap t_2)$ 
 $\Pi_{client\_id}(t_3)$ 

```

CLIENT_ID
C-001
C-002
C-003
C-004
C-005
C-006

Union: Find the information of employee whose branch id is 'B-001' or 'B-002'

SQL:

```

select *
from staff
where branch_id='B-001'
union
(select *
from staff

```

where branch\_id='B-002')

RA:

$t_1 \leftarrow \Pi (\sigma_{\text{branch\_id}='B-001'}(\text{Staff}))$

$t_2 \leftarrow \Pi (\sigma_{\text{branch\_id}='B-002'}(\text{Staff}))$

$\Pi (t_1 \cup t_2)$

STAFF_ID	BRANCH_ID	NAME	POST	GENDER	DOB	SALARY
S-A13	B-002	Tahsin Al-Sayed	Assistant	M	13-JUN-88	16000
S-M05	B-002	Suvo ahmed	Manager	M	03-NOV-89	24000
S-M21	B-001	Riddho	Manager	M	01-OCT-90	30000
S-S14	B-001	Taquir Ahmed	Supervisor	M	04-JUN-91	24000

Minus: Find those clients, whose visiting date is not submitted yet.

SQL:

select client\_id

from client

minus

(select client\_id

from visiting)

RA:

$t_1 \leftarrow \Pi_{\text{client\_id}}(\text{client})$

$t_2 \leftarrow \Pi_{\text{client\_id}}(\text{visiting})$

$\Pi_{\text{client\_id}}(t_1 - t_2)$

CLIENT_ID
C-007
C-008

Update: Give all managers in this a 10 percent salary rise.

SQL:

```
update staff set salary = salary * 1.1 where staff_id in (  
    select staff_id from staff where post='Manager')
```

RA:

```
t1 ←  $\Pi$  ( $\sigma_{\text{post}=\text{'Manager'}}$  (Staff))  
t2 ←  $\Pi_{\text{staff\_id, branch\_id, name, post, gender, dob, salary} * 1.1}$  (t1)  
Staff ← (Staff - t1)  $\cup$  t2  
Result: 3 row(s) updated.
```

Before update:

STAFF_ID	BRANCH_ID	NAME	POST	GENDER	DOB	SALARY
S-M21	B-001	Riddho	Manager	M	01-OCT-90	30000
S-A09	B-005	Saklian Jamal	Assistant	M	02-FEB-91	12000
S-M05	B-002	Suvo ahmed	Manager	M	03-NOV-89	24000
S-S14	B-001	Taquir Ahmed	Supervisor	M	04-JUN-91	24000
S-A07	B-003	Noboni	Assistant	F	13-FEB-93	18000
S-M22	B-004	Nahinan Asraf	Manager	F	26-MAR-91	28000
S-A10	B-006	Imran Arafat	Assistant	M	25-NOV-91	18000
S-A13	B-002	Tahsin Al-Sayed	Assistant	M	13-JUN-88	16000
S-A11	B-007	Sohan	Assistant	M	13-NOV-90	16000

After update:

STAFF_ID	BRANCH_ID	NAME	POST	GENDER	DOB	SALARY
S-M21	B-001	Riddho	Manager	M	01-OCT-90	36300
S-A09	B-005	Saklian Jamal	Assistant	M	02-FEB-91	12000
S-M05	B-002	Suvo ahmed	Manager	M	03-NOV-89	29040
S-S14	B-001	Taquir Ahmed	Supervisor	M	04-JUN-91	24000
S-A07	B-003	Noboni	Assistant	F	13-FEB-93	18000
S-M22	B-004	Nahinan Asraf	Manager	F	26-MAR-91	33880
S-A10	B-006	Imran Arafat	Assistant	M	25-NOV-91	18000
S-A13	B-002	Tahsin Al-Sayed	Assistant	M	13-JUN-88	16000
S-A11	B-007	Sohan	Assistant	M	13-NOV-90	16000

Delete: Delete all employees work in branch “B-002”.

SQL:

```
delete from staff
where Branch_id='B-002'
```

RA:

```
staff ← staff – (σbranch_id = "B-002" (staff))
```

Before delete:

STAFF_ID	BRANCH_ID	NAME	POST	GENDER	DOB	SALARY
S-M21	B-001	Riddho	Manager	M	01-OCT-90	30000
S-A09	B-005	Saklian Jamal	Assistant	M	02-FEB-91	12000
S-M05	B-002	Suvo ahmed	Manager	M	03-NOV-89	24000
S-S14	B-001	Taqir Ahmed	Supervisor	M	04-JUN-91	24000
S-A07	B-003	Noboni	Assistant	F	13-FEB-93	18000
S-M22	B-004	Nahinan Asraf	Manager	F	26-MAR-91	28000
S-A10	B-006	Imran Arafat	Assistant	M	25-NOV-91	18000
S-A13	B-002	Tahsin Al-Sayed	Assistant	M	13-JUN-88	16000
S-A11	B-007	Sohan	Assistant	M	13-NOV-90	16000

After delete:

STAFF_ID	BRANCH_ID	NAME	POST	GENDER	DOB	SALARY
S-M21	B-001	Riddho	Manager	M	01-OCT-90	30000
S-A09	B-005	Saklian Jamal	Assistant	M	02-FEB-91	12000
S-S14	B-001	Taqir Ahmed	Supervisor	M	04-JUN-91	24000
S-A07	B-003	Noboni	Assistant	F	13-FEB-93	18000
S-M22	B-004	Nahinan Asraf	Manager	F	26-MAR-91	28000
S-A10	B-006	Imran Arafat	Assistant	M	25-NOV-91	18000
S-A11	B-007	Sohan	Assistant	M	13-NOV-90	16000

Result: 2 rows deleted.

Aggregate function:

Max: Find the employee name who earn more than every employee of the agency?

SQL:

```
select name,salary
from staff
where staff.salary=(select max(salary) max_salary from
staff)
```

RA:

$$t \leftarrow G_{\text{max}(\text{salary}) \text{ as max\_salary}((\text{Staff}))}$$
$$\Pi_{\text{name,salary}} (\sigma_{\text{salary} = \text{max\_salary}} (\text{Staff} \times t))$$

NAME	SALARY
Riddho	30000

Min: Find the employee name who earn less than every employee of the agency?

SQL:

```
select name,salary
from staff
where staff.salary=(select min(salary) min_salary
from staff)
```

RA:

$$t \leftarrow G_{\text{min}(\text{salary}) \text{ as min\_salary}((\text{Staff}))}$$
$$\Pi_{\text{name,salary}} (\sigma_{\text{salary} = \text{min\_salary}} (\text{Staff} \times t))$$





Proof: As in canonical cover one functional dependencies exists, this schema is in our desired form.

Schema: Owner (Owner\_id, Owner\_name, address, mobile)

FD: Owner\_id  $\rightarrow$  Owner\_name, address, mobile

Proof: As in canonical cover one functional dependencies exists, this schema is in our desired form.

Schema: Staff (Staff\_id, Branch\_id, name, post, gender, dob, salary)FD: Staff\_id  $\rightarrow$  Branch\_id, name, post, gender, dob, salary

Proof: As in canonical cover one functional dependencies exists, this schema is in our desired form.

Schema: Registration (client\_id, branch\_id, staff\_id, date\_joined)

FD: client\_id,branch\_id  $\rightarrow$  staff\_id, date\_joined

Proof: As in canonical cover one functional dependencies exists, this schema is in our desired form.

Schema: House\_for\_rent (house\_id, owner\_id, branch\_id, type, living\_rooms,bath\_rooms, kitchen, dining\_rooms, rent)

FD: house\_id  $\rightarrow$  owner\_id, branch\_id, type, living\_rooms,bath\_rooms, kitchen, dining\_rooms, rent

Proof: As in canonical cover one functional dependencies exists, this schema is in our desired form.

Schema: Visiting (client\_id, House\_id, staff\_id, visiting\_date, status, comments, confirmation)

FD: client\_id,house\_id→ staff\_id, visiting\_date, status, comments, confirmation

Proof: As in canonical cover one functional dependencies exists, this schema is in our desired form.

Discussions :

By doing this project by my own I get to learn many things about database system. I have faced several problems while doing this project. I get to know about queries of multiple relations using relational algebra.

I have learned about functional dependencies, different types of clauses and different string operations. I apply various set operation as a query and get the desired results.

But desired result does not come first time. But I tried and get success.

This project helps me to get a clear idea of database system.

