American International University - Bangladesh

Apartment Management System

Submitted to: Juena Ahmed Noshin

Course: Introduction to Database

Section : B

NAME	ID	CONTRIBUTION
Riazul Zannah	22-47218-1	
Arafat, Safuan ul Haque	19-39945-1	
MD Raquibur Rahman	22-47220-1	
Prithanjoly Biswas Pew	20-43126-1	

CONTENTS:

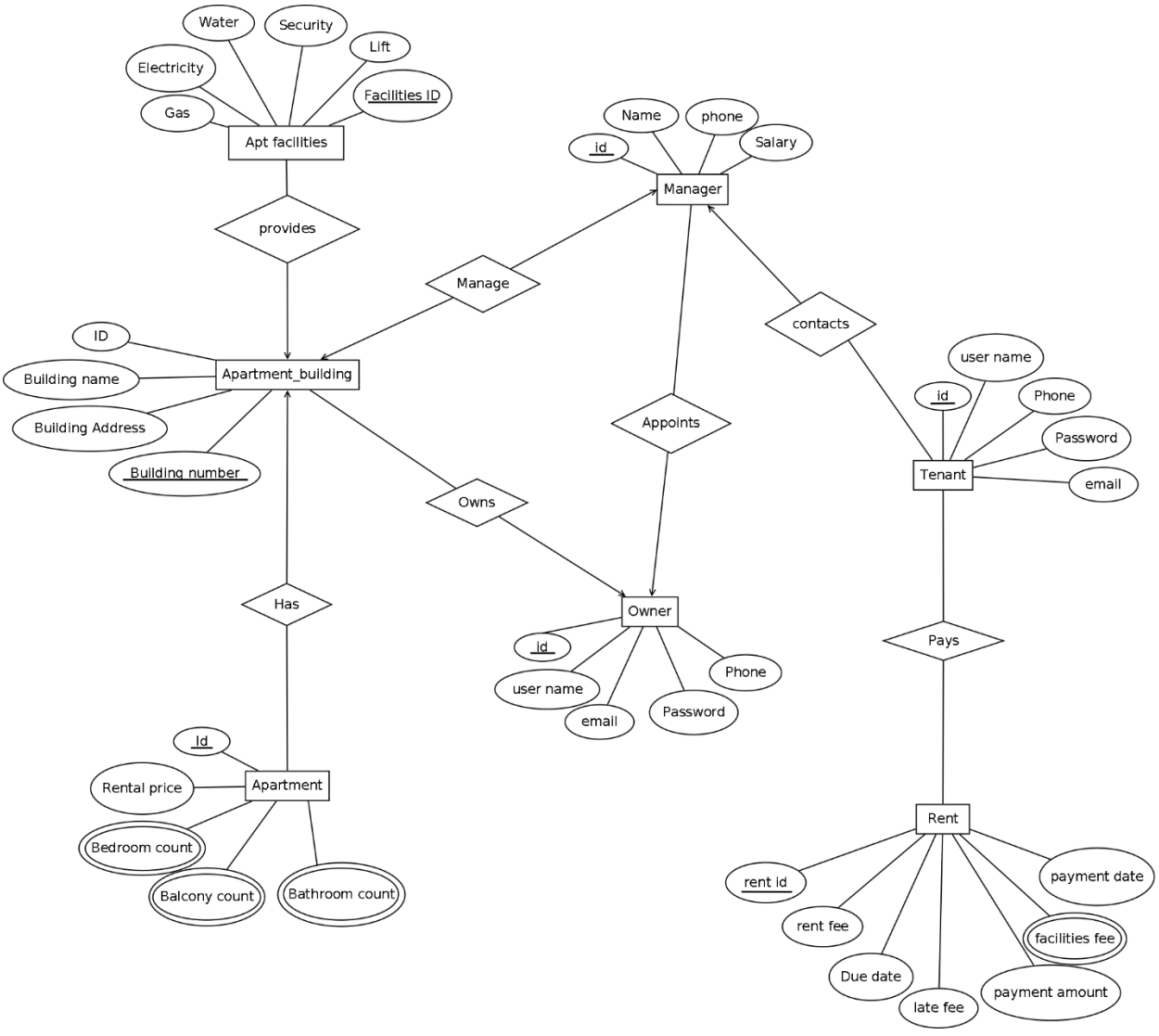
Introduction:

In our project, we will build a complete management system which will contain all information about apartment. It will contain information about owner, manager, tenant, apartment's information and relation between owner, manager and tenant. The system will help a tenant to know about all facilities and also rental information of apartment. This system keeps store the record of every contact details of apartment manager and apartment tenant. This system stores all details about buildings, apartment facilities etc.

Scenario Description:

In an “Apartment Management System” an owner can appoints many managers. The system stores owner id, name, email, pass phone number. And an owner can contact with many managers. Apartment is managed by a manager. This system stores manager id, name, phone and salary. A tenant can rent only one apartment in the same building. This system also keeps a record of rent details (rent-id, due date, rent fee, facilities fee, late fee, payment date, payment amount) paid by the tenants. Every tenant is identified by a user name. The system also stores tenant id, user name, phone, password and email. A manager can contact many tenants. But a tenant can contact only one manager. A manager can only manage one apartment building. An owner can own many apartments buildings. In an apartment management system, apartment building has a unique id, building’s name, buildings address, buildings number. An Apartment building has many apartments. In an apartment management system details of apartments’ such as id, rental price, bedroom count and bathroom count, balcony count are also stored. The system also stores a list of apartment facilities (facilities-id, gas, electricity, water, security, lift).Everything is recorded in this system.

ER Diagram:


Water
Security
Lift
Electricity
Facilities ID
Gas
Apt facilities
provides
Name
phone
id
Salary
Manager
Manage
contacts
ID
Building name
Apartment_building
Building Address
Building number
Appoints
user name
id
Phone
Password
Tenant
email
Owns
Has
Owner
Pays
id
user name
email
Password
Phone
id
Rental price
Apartment
Bedroom count
Balcony count
Bathroom count
Rent
rent id
rent fee
Due date
late fee
payment amount
facilities fee
payment date

Normalization:

Appoint :

UNF

Appoint (owner_name, owner_id, owner_pass, owner_email, owner_phone number, manager_name, manager_id, manager_phone number, manager_salary)

1NF

there is no multi valued attribute,relation already in 1UNF.

1. owner_name, owner_id, owner_pass, owner_email, owner_phone number, manager_name, manager_id, manager_phone number, manager_salary

2NF

1. owner_id, owner_name, owner_pass, owner_email, owner_phone number.
2. manager_id, manager_name, manager_phone number, manager_salary .

3NF

there is no transitive dependency

1. owner_id, owner_name, owner_pass, owner_email, owner_phone number.
2. manager_id, manager_name, manager_phone number, manager_salary ,

Table creation

1. owner_id, owner_name, owner_pass, owner_email, owner_phone number.

2. manager_id, manager_name, manager_phone number, manager_salary **,owner-id.**

Contact :

UNF

Contact (tenant_id , tenant_name, tenant_email, tenant_phone numb, tenant_pass, manager_id , manager_name, manager_phone number, manager_salary)

1NF

there is no multi valued attribute,relation already in 1UNF.

1.tenant_id , tenant_name, tenant_email, tenant_phone numb,

tenant_pass, manager_id , manager_name, manager_phone number,manager_salary

2NF

1. tenant_id , tenant_name, tenant_email, tenant_phone numb,

tenant_pass .

2. manager_id , manager_name, manager_phone number,

manager_salary

3NF

there is no transitive dependency

1. tenant_id , tenant_name, tenant_email, tenant_phone numb,

tenant_pass .

2. manager_id , manager_name, manager_phone number,

manager_salary .

Table creation

1. tenant_id , tenant_name, tenant_email, tenant_phone numb,

tenant_pass**,manager_id .**

2. manager_id , manager_name, manager_phone number,

manager_salary , .

Manage :

UNF

Manage (manager_id , manager_name, manager_phone apartment building_number,number, manager_salary , apartment building_name , apartment building_number,apartment building_address)

1NF

there is no multi valued attribute,relation already in 1UNF.

1. manager_id , manager_name, manager_phone number, manager_salary , apartment building_name , apartment building_address ,apartment building_name .

2NF

1.manager_id , manager_name, manager_phone number, manager_salary .

2.apartment building_name , apartment building_number, apartmentbuilding_address .

3NF

there is no transitive dependency

1. manager_id , manager_name, manager_phone number,

manager_salary .

2. apartment building_name , apartment building_number, apartment building_address .

Table creation

1. manager_id , manager_name, manager_phone number,

manager_salary .

2. apartment building_name , apartment building_number, apartment building_address **,manager-id.**

Provides :

UNF

Provides (apartment building_name , apartment building_number,

apartment building_address , facility_id, gas, electrecity, water,

security, lift)

1NF

there is no multi valued attribute,relation already in 1UNF.

1. apartment building_name , apartment building_number, apartment building_address , facility_id ,gas, electrecity, water, security, lift .

2NF

1. apartment building_name , apartment building_number, apartment building_address .

2. facility_id ,gas, electrecity, water, security, lift .

3NF

there is no transitive dependency

1. apartment building_name , apartment building_number, apartment building_address .

2. facility_id ,gas, electrecity, water, security, lift.

Table creation

1. apartment building_name , apartment building_number, apartment building_address .

2. facility_id , gas, electrecity, water, security, lift ,**partment building-number.**

owns:

UNF

owns (apartment building_name , apartment building_number,

apartment building_address, owner_name, owner_id, owner_email,

owner_phnone number,owner_pass)

1NF

there is no multi valued attribute,relation already in 1UNF.

1.apartment building_name , apartment building_number, apartment building_address, owner_name, owner_id, owner_email, owner_phnone number,owner_pass .

2NF

1.apartment building_name , apartment building_number, apartment building_address.

2. owner_name, owner_id, owner_email, owner_phnone number,owner_pass .

3NF

there is no transitive dependency.

1.apartment building_name , apartment building_number, apartment building_address.

2. owner_name, owner_id, owner_email, owner_phnone

number,owner_pass .

Table creation

1.apartment building_name , apartment building_number, apartment building_address,**owner_id.**

2. owner_name, owner_id, owner_email, owner_phnone

number,owner_pass ,.

Has

UNF

Has (apartment building_name , apartment building_number,

apartment building_address, apartment_id, apartment_rent price,

bedroom_count, washroom_count, belcony_count)

1NF

there is three (bedroom_count, washroom_count, belcony_count) multi

valued attribute, relation already in 1UNF.

1. apartment building_name , apartment building_number, apartment building_address, apartment_id, apartment_rent price, bedroom_count, washroom_count, belcony_count .

2NF

1. apartment building_name , apartment building_number, apartment building_address

2. apartment_id, apartment_rent price, bedroom_count,

washroom_count, belcony_count .

3NF

there is no transitive dependency.

1. apartment building_name , apartment building_number, apartment building_address

2. apartment_id, apartment_rent price, bedroom_count,

washroom_count, belcony_count .

Table creation

1. apartment building_name , apartment building_number, apartment building_address

2. apartment_id, apartment_rent price, bedroom_count,

washroom_count, belcony_count,**apartment building_number .**

Pays :

UNF

Pay (rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, payment_late,tenant_id, tenant_name, tenant phone_number,tenant_password, tenant_email) .

1NF

there is no multi valued attribute, relation already in 1UNF.

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee,payment_late,tenant_id, tenant_name, tenant phone_number,tenant_password, tenant_email

2NF

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, payment_late.

2.tenant_id, tenant_name, tenant phone_number,tenant_password, tenant_email.

3NF

there is no transitive dependency.

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, payment_late.

2.tenant_id, tenant_name, tenant phone_number,tenant_password, tenant_email.

Table creation

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee,payment_late.

2.tenant_id, tenant_name, tenant phone_number,tenant_password, tenant_email.

3.rent_id ,tenant_id .

Temporary tables :

1. owner_id, owner_name, owner_pass, owner_email, owner_phone number.

2. manager_id, manager_name, manager_phone number, manager_salary ,**owner-id**.

3. tenant_id , tenant_name, tenant_email, tenant_phone numb, tenant_pass,**manager_id** .

4. ~~manager_id , manager_name, manager_phone number, manager_salary~~ .

5. ~~manager_id , manager_name, manager_phone number, manager_salary .~~

6. apartment building_name , apartment building_number, apartment building_address ,**manager-id**.

7. ~~apartment building_name , apartment building_number, apartment building_address .~~

8. facility_id , gas, electrecity, water, security, lift ,**apartment building-number.**

9.apartment building_name , apartment building_number, apartment building_address,**owner_id.**

10. ~~owner_name, owner_id, owner_email, owner_phnone number,owner_pass ,.~~

11~~. apartment building_name , apartment building_number, apartment building_address.~~

12. apartment_id, apartment_rent price, bedroom_count, washroom_count, belcony_count,**apartment building_number** .

13. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, payment_late.

14.~~tenant_id, tenant_name, tenant phone_number,tenant_password, tenant_email.~~

15.**rent_id ,tenant_id** .

Final tables :

1. owner_id, owner_name, owner_pass, owner_email, owner_phone

number.

2. manager_id, manager_name, manager_phone number,

manager_salary ,**owner-id**.

3. tenant_id , tenant_name, tenant_email, tenant_phone numb,

tenant_pass,**manager_id** .

4. apartment building_name , apartment building_number, apartment building_address ,**manager-id.**

5. facility_id , gas, electrecity, water, security, lift ,**apartment building-number.**

6.apartment building_name , apartment building_number, apartment building_address,**owner_id.**

7. apartment_id, apartment_rent price, bedroom_count,

washroom_count, belcony_count,**apartment building_number** .

8. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, payment_late.

9.**rent_id ,tenant_id** .

Schema Diagram :

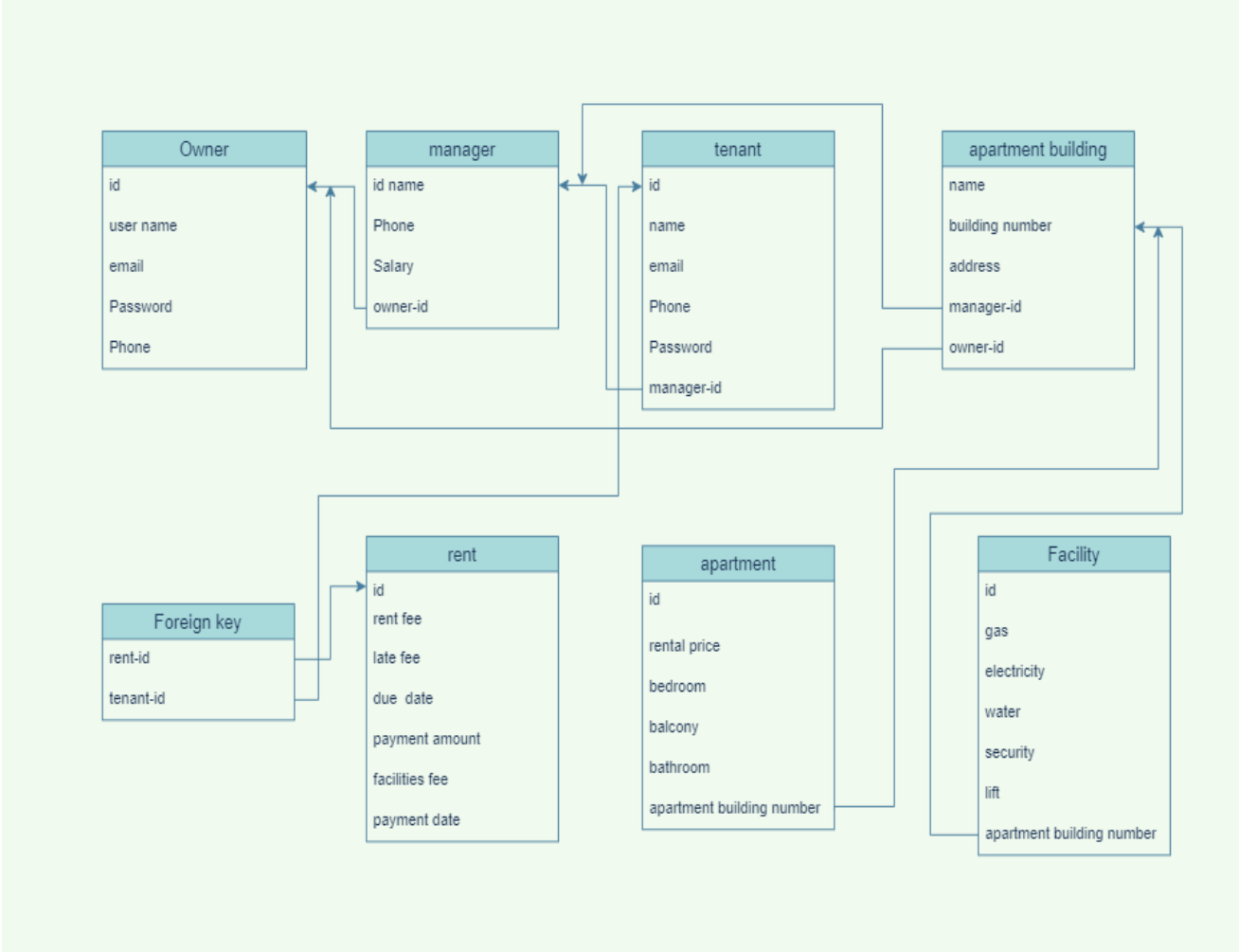


Table Creation:

CREATE USER Apartment IDENTIFIED BY house;

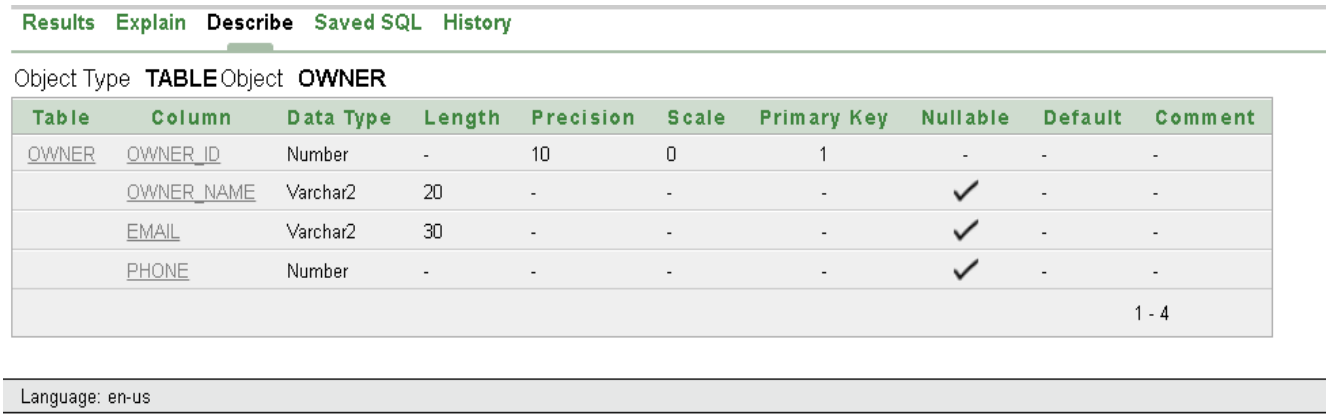
GRANT connect, resource, unlimited tablespace TO Apartment;

Owner table:

CREATE TABLE Owner(owner_id number(10)PRIMARY KEY,owner_name varchar2(20),email varchar2(30),phone number);

CREATE SEQUENCE ower_id_seq INCREMENT BY 1 START WITH 5 MAXVALUE 50 NOCACHE NOCYCLE;

describe Owner;



Results Explain Describe Saved SQL History

Object Type TABLE Object OWNER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
OWNER	OWNER_ID	Number	-	10	0	1	-	-	-
	OWNER_NAME	Varchar2	20	-	-	-	✓	-	-
	EMAIL	Varchar2	30	-	-	-	✓	-	-
	PHONE	Number	-	-	-	-	✓	-	-

1 - 4

Language: en-us

Manager table:

CREATE TABLE Manager(manager_id number(10)PRIMARY KEY,manager_name varchar2(20),salary number(10),phone number,owner_id number(10));

ALTER TABLE manager ADD CONSTRAINT qq1 FOREIGN KEY(owner_id) REFERENCES Owner(owner_id);

CREATE SEQUENCE manager_id_seq INCREMENT BY 1 START WITH 5 MAXVALUE 50 NOCACHE NOCYCLE;

Describe Manager;

Results Explain Describe Saved SQL History

Object Type TABLE Object MANAGER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGER	MANAGER_ID	Number	-	10	0	1	-	-	-
	MANAGER_NAME	Varchar2	20	-	-	-	✓	-	-
	SALARY	Number	-	10	0	-	✓	-	-
	PHONE	Number	-	-	-	-	✓	-	-
	OWNER_ID	Number	-	10	0	-	✓	-	-

1 - 5

Language: en-us

Tenant table:

CREATE TABLE Tenant(tanant_id number(10)PRIMARY KEY,tanant_name varchar2(20),email varchar2(30),phone number,password varchar2(30),manager_id number(10));

ALTER TABLE tenant ADD CONSTRAINT qq2 FOREIGN KEY(manager_id) REFERENCES manager(manager_id);

CREATE SEQUENCE tanant_id_seq INCREMENT BY 1 START WITH 5 MAXVALUE 50 NOCACHE NOCYCLE;

Describe Tenant;

Results Explain Describe Saved SQL History

Object Type TABLE Object TENANT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TENANT	TANANT_ID	Number	-	10	0	1	-	-	-
	TANANT_NAME	Varchar2	20	-	-	-	✓	-	-
	EMAIL	Varchar2	30	-	-	-	✓	-	-
	PHONE	Number	-	-	-	-	✓	-	-
	PASSWORD	Varchar2	30	-	-	-	✓	-	-
	MANAGER_ID	Number	-	10	0	-	✓	-	-

1 - 6

Language: en-us

Apartment_Building table:

CREATE TABLE Apartment_Building(building_number number(10)PRIMARY KEY,building_name varchar2(20),address varchar2(20),owner_id number(10),manager_id number(10));

ALTER TABLE Apartment_Building ADD CONSTRAINT qq3 FOREIGN KEY(manager_id) REFERENCES manager(manager_id);

ALTER TABLE Apartment_Building ADD CONSTRAINT qq4 FOREIGN KEY(owner_id) REFERENCES Owner(owner_id);

CREATE SEQUENCE building_number_seq INCREMENT BY 1 START WITH 105 MAXVALUE 500 NOCACHE NOCYCLE;

Describe Apartment_Building;

Results Explain Describe Saved SQL History

Object Type TABLE Object APARTMENT_BUILDING

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
APARTMENT_BUILDING	BUILDING_NUMBER	Number	-	10	0	1	-	-	-
	BUILDING_NAME	Varchar2	20	-	-	-	✓	-	-
	ADDRESS	Varchar2	20	-	-	-	✓	-	-
	OWNER_ID	Number	-	10	0	-	✓	-	-
	MANAGER_ID	Number	-	10	0	-	✓	-	-

1 - 5

Language: en-us Copy

Rent table:

CREATE TABLE Rent(ID number(10)PRIMARY KEY,rent_fee number(10),late_fee number(10),due_fee number(10),payment_amount number(10),facilities_fee number(10),payment_date date);

CREATE SEQUENCE rent_id_seq INCREMENT BY 1 START WITH 5 MAXVALUE 50 NOCACHE NOCYCLE;

Describe Rent;

Results Explain Describe Saved SQL History

Object Type TABLE Object RENT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RENT	ID	Number	-	10	0	1	-	-	-
	RENT_FEE	Number	-	10	0	-	✓	-	-
	LATE_FEE	Number	-	10	0	-	✓	-	-
	DUE_FEE	Number	-	10	0	-	✓	-	-
	PAYMENT_AMOUNT	Number	-	10	0	-	✓	-	-
	FACILITIES_FEE	Number	-	10	0	-	✓	-	-
	PAYMENT_DATE	Date	7	-	-	-	✓	-	-

1 - 7

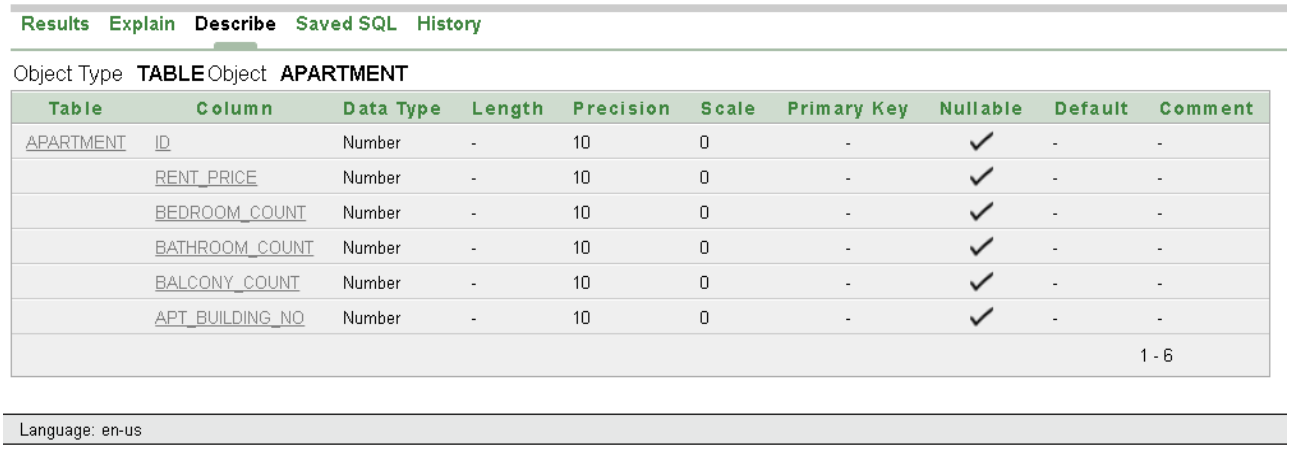
Language: en-us

Apartment table:

CREATE TABLE Apartment(id number(10),rent_price number(10),bedroom number(10),bathroom number(10),balcony number(10),apt_building_no number(10));

ALTER TABLE Apartment ADD CONSTRAINT qq5 FOREIGN KEY(apt_building_no) REFERENCES Apartment_Building(building_number);

Describe Apartment;



Results Explain Describe Saved SQL History

Object Type TABLE Object APARTMENT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
APARTMENT	ID	Number	-	10	0	-	✓	-	-
	RENT_PRICE	Number	-	10	0	-	✓	-	-
	BEDROOM_COUNT	Number	-	10	0	-	✓	-	-
	BATHROOM_COUNT	Number	-	10	0	-	✓	-	-
	BALCONY_COUNT	Number	-	10	0	-	✓	-	-
	APT_BUILDING_NO	Number	-	10	0	-	✓	-	-

1 - 6

Language: en-us

Foreign_key table:

CREATE TABLE Foreign_key(rent_id number(10), tenant_id number(10));

ALTER TABLE Foreign_key ADD CONSTRAINT qq6 FOREIGN KEY(rent_id) REFERENCES Rent(ID);

ALTER TABLE Foreign_key ADD CONSTRAINT qq7 FOREIGN KEY(tenant_id) REFERENCES Tenant(tanant_id);

Describe Foreign_key;

Results Explain Describe Saved SQL History

Object Type TABLE Object FOREIGN_KEY

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FOREIGN_KEY	RENT_ID	Number	-	10	0	-	✓	-	-
	TENANT_ID	Number	-	10	0	-	✓	-	-

1 - 2

Language: en-us

Facility table:

CREATE TABLE Facility(id number(10),gas varchar2(20),electricity varchar2(20),water varchar2(20),secuirity varchar2(20), lift varchar2(20),apt_building_no number(10));

ALTER TABLE Facility ADD CONSTRAINT qq8 FOREIGN KEY(apt_building_no) REFERENCES Apartment_Building(building_number);

Describe Facility;

Results Explain Describe Saved SQL History

Object Type TABLE Object FACILITY

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FACILITY	ID	Number	-	10	0	-	✓	-	-
	GAS	Varchar2	20	-	-	-	✓	-	-
	ELECTRICITY	Varchar2	20	-	-	-	✓	-	-
	WATER	Varchar2	20	-	-	-	✓	-	-
	SECUIRITY	Varchar2	20	-	-	-	✓	-	-
	LIFT	Varchar2	20	-	-	-	✓	-	-
	APT_BUILDING_NO	Number	-	10	0	-	✓	-	-

1 - 7

Language: en-us

Data insert

Owner table:

```
INSERT INTO Owner VALUES(1,'Naruto','naruto@gmail.com',122344);
INSERT INTO Owner VALUES(2,'Sasuke','sasuke@gmail.com',125234);
INSERT INTO Owner VALUES(3,'sakura','sakura@gmail.com',144344);
INSERT INTO Owner VALUES(4,'hinata','hinata@gmail.com',122847);
INSERT INTO Owner VALUES(5,'neji','neji@gmail.com',124544);
select * from owner;
```

Results Explain Describe Saved SQL History

OWNER_ID	OWNER_NAME	EMAIL	PHONE
1	Naruto	naruto@gmail.com	122344
2	Sasuke	sasuke@gmail.com	125234
3	sakura	sakura@gmail.com	144344
4	hinata	hinata@gmail.com	122847
5	neji	neji@gmail.com	124544

5 rows returned in 0.01 seconds CSV Export

Language: en-us

Manager table:

INSERT INTO Manager VALUES(1,'natsu',20000,234564,1);

INSERT INTO Manager VALUES(2,'lucy',20000,234564,2);

INSERT INTO Manager VALUES(3,'gray',20000,234564,3);

INSERT INTO Manager VALUES(4,'juvia',10000,234564,4);

INSERT INTO Manager VALUES(5,'erza',50000,234564,5);

select * from Manager;

Results Explain Describe Saved SQL History

MANAGER_ID	MANAGER_NAME	SALARY	PHONE	OWNER_ID
1	natsu	20000	234564	1
2	lucy	20000	234564	2
3	gray	20000	234564	3
4	juvia	10000	234564	4
5	erza	50000	234564	5

5 rows returned in 0.14 seconds CSV Export

Language: en-us

Tenant table:

INSERT INTO Tenant
VALUES(1,'violet','violet@gmail.com',354564,'xxyr',1);

INSERT INTO Tenant
VALUES(2,'gilbert','gilbert@gmail.com',236864,'rrfyr',2);

INSERT INTO Tenant
VALUES(3,'erica','erica@gmail.com',446564,'dgvyr',3);

INSERT INTO Tenant
VALUES(4,'rose','rose@gmail.com',887664,'eeyr',4);

INSERT INTO Tenant
VALUES(5,'edward','edward@gmail.com',345454,'ttrr',5);

select * from tenant;

Results Explain Describe Saved SQL History

TANANT_ID	TANANT_NAME	EMAIL	PHONE	PASSWORD	MANAGER_ID
1	violet	violet@gmail.com	354564	xxyr	1
2	gilbert	gilbert@gmail.com	236864	rrfyr	2
3	erica	erica@gmail.com	446564	dgvyr	3
4	rose	rose@gmail.com	887664	eeyr	4
5	edward	edward@gmail.com	345454	ttrr	5

5 rows returned in 0.00 seconds CSV Export

Language: en-us

Apartment_Building table:

INSERT INTO Apartment_Building VALUES(100,'A','uttara',1,1);

INSERT INTO Apartment_Building VALUES(102,'B','gulshan',2,2);

INSERT INTO Apartment_Building VALUES(103,'C','baridhara',3,3);

INSERT INTO Apartment_Building VALUES(104,'D','bashundhara',4,4);

INSERT INTO Apartment_Building VALUES(105,'E','tongi',5,5);

select * from Apartment_Building ;

Results Explain Describe Saved SQL History

BUILDING_NUMBER	BUILDING_NAME	ADDRESS	OWNER_ID	MANAGER_ID
100	A	uttara	1	1
102	B	gulshan	2	2
103	C	baridhara	3	3
104	D	bashundhara	4	4
105	E	tongi	5	5

5 rows returned in 0.14 seconds CSV Export

Language: en-us

Rent table:

INSERT INTO Rent VALUES(1,5000,100,3000,2400,300,TO_DATE('10-1-2020','DD-MM-YYYY'));

INSERT INTO Rent VALUES(2,7000,100,7000,0,400,TO_DATE('01-03-2021','DD-MM-YYYY'));

INSERT INTO Rent VALUES(3,2000,0,0,2200,200,TO_DATE('02-04-2021','DD-MM-YYYY'));

INSERT INTO Rent VALUES(4,5000,200,2000,3400,100,TO_DATE('01-01-2021','DD-MM-YYYY'));

INSERT INTO Rent VALUES(5,8000,100,4000,4400,300,TO_DATE('02-08-2020','DD-MM-YYYY'));

select * from Rent ;

Results Explain Describe Saved SQL History

ID	RENT_FEE	LATE_FEE	DUE_FEE	PAYMENT_AMOUNT	FACILITIES_FEE	PAYMENT_DATE
1	5000	100	3000	2400	300	10-JAN-20
2	7000	100	7000	0	400	01-MAR-21
3	2000	0	0	2200	200	02-APR-21
4	5000	200	2000	3400	100	01-JAN-21
5	8000	100	4000	4400	300	02-AUG-20

5 rows returned in 0.17 seconds CSV Export

Language: en-us

Apartment table:

INSERT INTO Apartment VALUES(1,5000,3,3,2,100);

INSERT INTO Apartment VALUES(2,7000,4,4,3,102);

INSERT INTO Apartment VALUES(3,2000,1,2,1,103);

INSERT INTO Apartment VALUES(4,5000,3,3,2,104);

INSERT INTO Apartment VALUES(5,8000,4,5,4,105);

select * from Apartment ;

Results Explain Describe Saved SQL History

ID	RENT_PRICE	BEDROOM_COUNT	BATHROOM_COUNT	BALCONY_COUNT	APT_BUILDING_NO
1	5000	3	3	2	100
2	7000	4	4	3	102
3	2000	1	2	1	103
4	5000	3	3	2	104
5	8000	4	5	4	105

5 rows returned in 0.25 seconds CSV Export

Language: en-us

Foreign_key table:

```
INSERT INTO Foreign_key VALUES(1,1);
INSERT INTO Foreign_key VALUES(2,2);
INSERT INTO Foreign_key VALUES(3,3);
INSERT INTO Foreign_key VALUES(5,5);
INSERT INTO Foreign_key VALUES(4,4);
select * from Foreign_key ;
```

Results | Explain | Describe | Saved SQL | History

RENT_ID	TENANT_ID
1	1
2	2
3	3
4	4
5	5

5 rows returned in 0.22 seconds CSV Export

Language: en-us

Facility table:

```
INSERT INTO Facility
VALUES(1,'available','available','available','secured','not available',100);

INSERT INTO Facility
VALUES(2,'available','available','available','secured','available',102);

INSERT INTO Facility VALUES(3,'available','available','available','not
secured','not available',103);

INSERT INTO Facility
VALUES(4,'available','available','available','secured','not available',104);

INSERT INTO Facility
VALUES(5,'available','available','available','secured','available',105);

select * from Facility ;
```

Results Explain Describe Saved SQL History

ID	GAS	ELECTRICITY	WATER	SECUIRITY	LIFT	APT_BUILDING_NO
1	available	available	available	secured	available	100
2	available	available	available	secured	available	102
3	available	available	available	not secured	not available	103
4	available	available	available	secured	not available	104
5	available	available	available	secured	available	105

5 rows returned in 0.47 seconds CSV Export

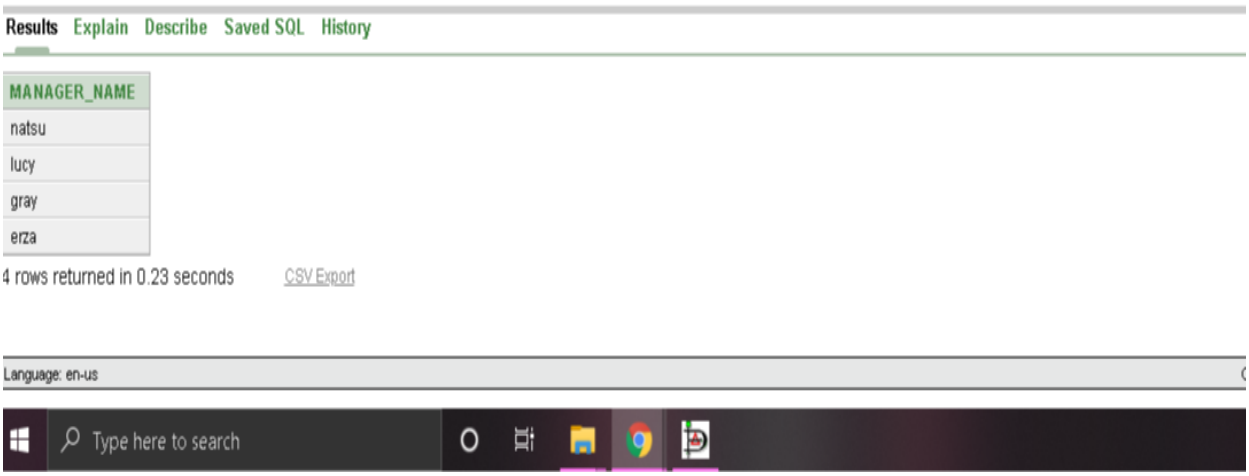
Language: en-us

Query Writing:

SUB-QUERY:

1. **Display the manager who earns more than juvia.**

➢ select MANAGER_NAME from Manager where SALARY>(select SALARY from Manager where MANAGER_NAME='juvia');



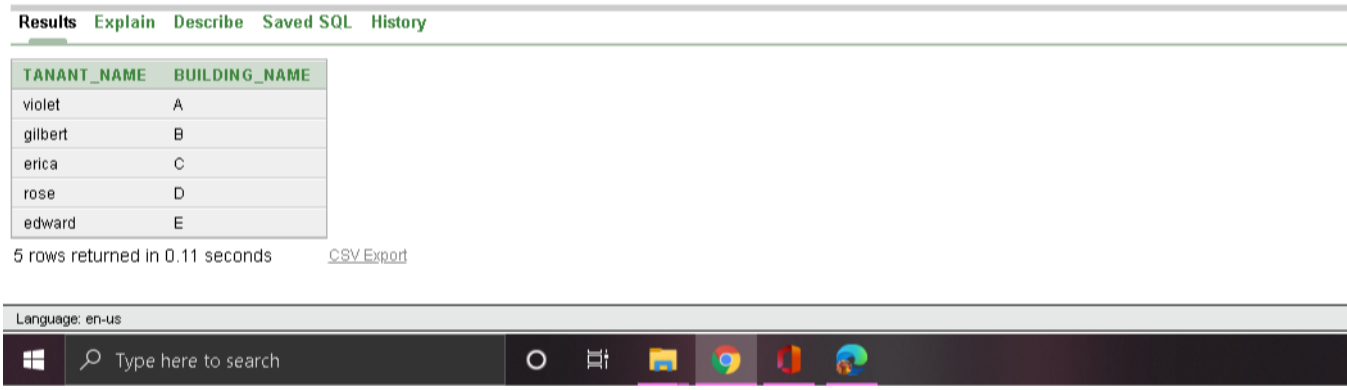
2. Display the manager name who join after gray.

- select MANAGER_NAME from Manager where MANAGER_ID>(select MANAGER_ID from Manager where MANAGER_NAME='gray');

JOINING:

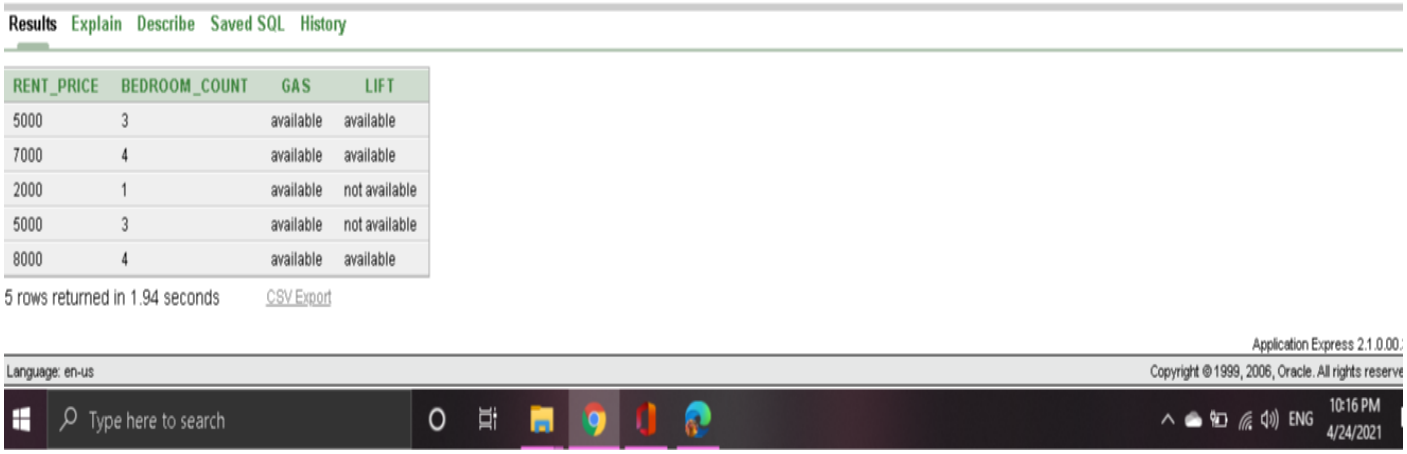
1. **Write a query to display tenant_name,apartment_building from the table tenant,apartment building.**

➢ Select Tenant.TANANT_NAME,Apartment_Building.BUILDING_NAME from Tenant,Apartment_Building where Tenant.MANAGER_ID=Apartment_Building.MANAGER_ID;



2. **Write a query to display RENT_PRICE, BEDROOM_COUNT, GAS, LIFT from Apartment, Facility**.

➢ Select Apartment.RENT_PRICE,Apartment.BEDROOM_COUNT,FACILITY.GAS,FACILITY.LIFT from Apartment,Facility where Apartment.APT_BUILDING_NO=FACILITY.APT_BUILDING_NO;



VIEW:

1. **Create a view called RENTVIEW based on the RENT_FEE, PAYMENT_AMOUNT AND FACILITES_FEE from the RENT table.**

➢ Create view RENTVIEW as select RENT_FEE, PAYMENT_AMOUNT, FACILITIES_FEE from RENT;
Select * from RENTVIEW

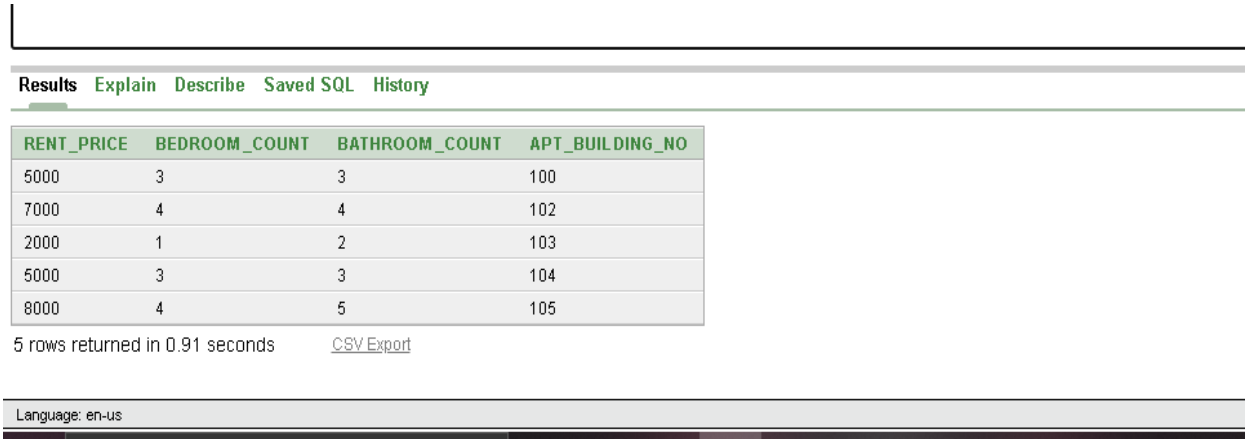
Results Explain Describe Saved SQL History

RENT_FEE	PAYMENT_AMOUNT	FACILITIES_FEE
5000	2400	300
7000	0	400
2000	2200	200
5000	3400	100
8000	4400	300

5 rows returned in 1.03 seconds CSV Export

2. **Create a view called APARTMENTVIEW based on the RENT_PRICE,BEDROOM_COUNT,BATHROOM_COUNT,APT_BUILDING_NO from the APARTMENT table.**

- Create view APARTMENTVIEW as select RENT_PRICE, BEDROOM_COUNT,BATHROOM_COUNT,APT_BUILDING_NO from Apartment;
  Select * from APARTMENTVIEW



Results Explain Describe Saved SQL History

RENT_PRICE	BEDROOM_COUNT	BATHROOM_COUNT	APT_BUILDING_NO
5000	3	3	100
7000	4	4	102
2000	1	2	103
5000	3	3	104
8000	4	5	105

5 rows returned in 0.91 seconds CSV Export

Language: en-us

Relational Algebra:

1. Find the manager_name where m_salary is greater than 2000.
   - πmanager_name(σ m_salary>2000$^{(manager)}$) {Table: MANAGER}
2. Find the user_name and email where manager _id is 5.
   - π user_name,email(σ manager_id =”5” $^{(Member)}$) {Table:MEMBER}
3. Find the apt_id where rental_price is greater than 7000.
   - π Apt_id(σ rental_price>7000$^{(Apertment)}$) {Table:APARTMENT_TABLE}
4. Find the F_id where building_id is 3.
   - π F_id(σ building_id=’3’$^{(Apartment\_facilities)}$) {Table:APARTMENT_FACILITIES }
5. Find the rent_id where payment_amount is 3400.
   - π rent_id(σ payment_amount=”3400’’$^{(Rent)}$) {Table: RENT}

Conclusion:

This project is relatively simple to understand and implement. It fulfills all the current requirements of local building management company. The system is very user-friendly; a person with basic computer skills can easily use this system. Queries of this project “Apartment Management System” are run in 'Oracle 10g'.Here we made 6 relationship among all the entities with cardinality. In our project, we have mentioned the queries as well as inserted screenshots of the tables we created using those queries. Normalization process makes this project simpler. This project overall covers the following fields:

1. Apartment details: This AMS stores all the information, records and data related to apartment. These data include total number of flats and rooms, types of rooms.

2. Personal Information: Personal information of every manager & tenants is also stored in this system.

3. Facilities Provided: Whether an apartment provides facilities (water, gas, electricity , security) or not is also mentioned in this system.

4. Rent Collection: This system also gives solution to rent collection issues. Difficulties & Problems Leak of personal information: Personal information of manager and members may leak when someone search for an available apartment. Less Security: Due to lack of proper security this system can be easily hacked. Parking Slot Confusion: This system doesn’t give any solution to mark parking slot for each member individually .

Future Development:

Adding more security to this system will solve the possibility of information leakage and hacking. If parking slots are given unique id for per apartment, that will solve the parking slot confusion. Including GPS tracking in this system will give the system a smart touch. People will be able to find out vacant apartment building through online searching. Moreover, for one or more apartments in a building, the database has to collect a large amount of data, which makes the system very expensive to upgrade and store data. In future, we will try to make it more convenient to reduce the costing. If these issues are addressed, it is expected that in future this system will be more convenient and user friendly. In the future If you want you can add, edit, update or delete something. Lastly, new features can be added into the system as per user requirement. The project is very flexible in that aspect.