AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF CS

INTRODUCTION TO DATABASE

Section: x

Group: x

Project Report:

Apartment Management System

Supervised By

XXXX XXXXXX XXXXX

Submitted By

Name	ID	Contribution
1.		
2.		
3.		
4.		
5.		

Date of Submission: **Expired**

TABLE OF CONTENTS

1. Introduction

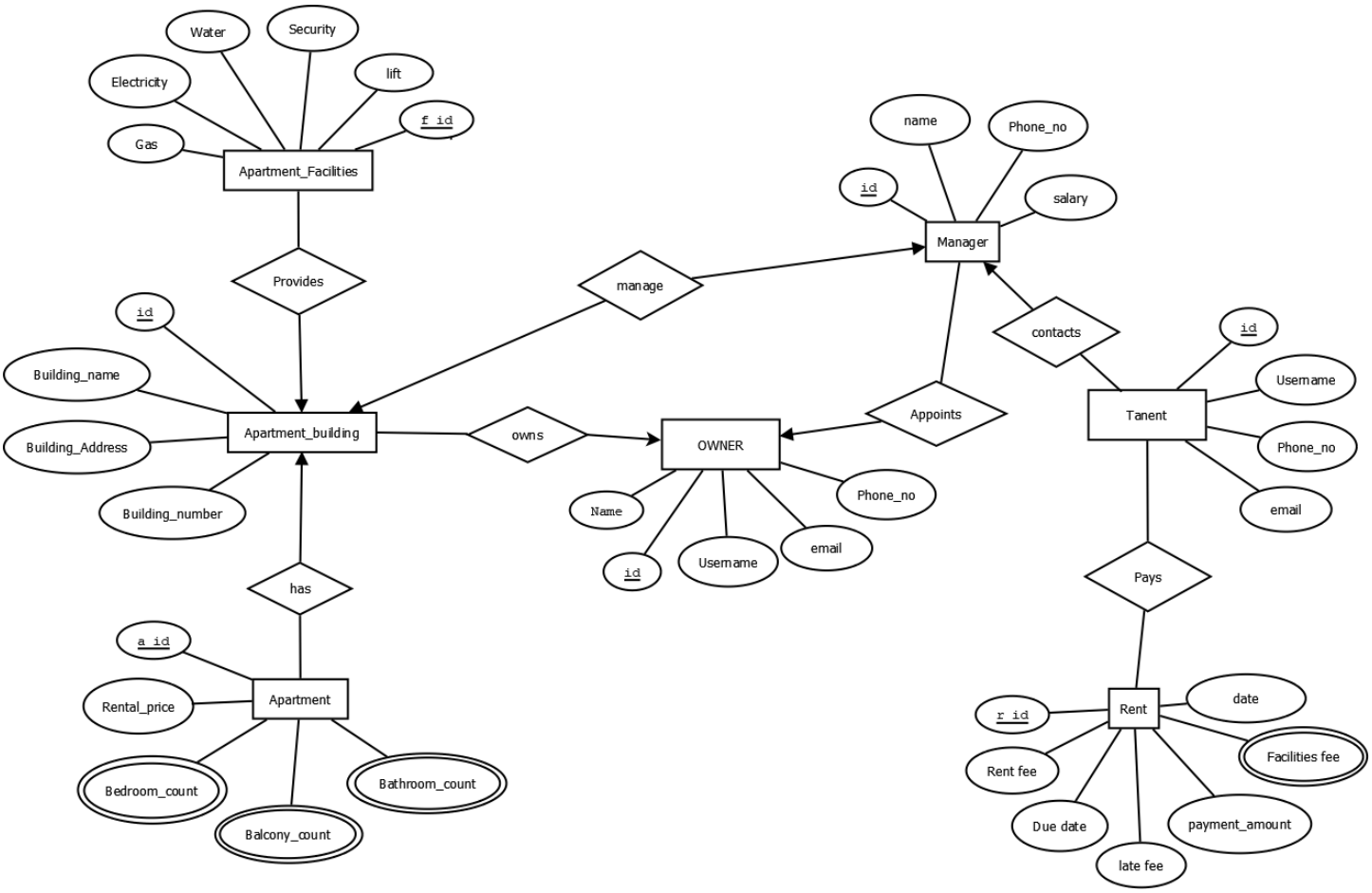
It is safe to say that most activities such as apartment allocation carried out in most cities in Bangladesh are done manually. Therefore, there is much strain on the individuals running the apartment. An Apartment Management system is a database developed for managing most activities in the apartment with the help of the apartment administrator.

In our project, we will build a complete management system for an apartment, creating an information-sharing bond between manager and tenant.
This research work aims to provide a solution to the problem of apartment management by designing a user-friendly computerized system that will be compatible with the existing manual systems. The database to be developed will solve apartment management's problem, thus helping to reduce issues associated with the manual apartment management system.

2. Scenario Description

In an "Apartment Management System," an owner might appoint a manager. One owner might appoint one manager: the system stores owner id, username, name, email, phone number. And an owner can contact many managers. A manager manages the apartment. One manager manages exactly one apartment, but one manager can manage multiple apartments. This system stores the manager's id, name, phone, and salary. A tenant can rent only one apartment in the same building. This system also records a multiple value facilities fee and rent details such as rent-id, due date, rent fee, late fee, payment date, and payment amount the tenants pay. An id identifies every tenant. The system also stores tenant username, username, phone, and email. A manager can contact many tenants. But a tenant can contact only one manager. An owner can own many apartments buildings, but one apartment has only one owner. A unique id identifies an apartment building. Also, it has the building's name, buildings address, and buildings number in an apartment management system. An Apartment building has many apartments, but one apartment exists only in one apartment building. A unique id identifies apartments. The system also stores rental price as well as multiple info like bedroom count and bathroom count, balcony count. The system also holds a list of apartment facilities such as facilities-id, gas, electricity, water, security, lift. An apartment management system might record all these things.

3. ER Diagram


Water
Security
Electricity
lift
f_id
Gas
Apartment_Facilities
Provides
id
Building_name
Building_Address
Building_number
Apartment_building
has
a_id
Rental_price
Apartment
Bedroom_count
Balcony_count
Bathroom_count
manage
owns
OWNER
Name
id
Username
email
Phone_no
Appoints
name
Phone_no
id
salary
Manager
contacts
id
Username
Phone_no
email
Tanent
Pays
r_id
Rent
date
Facilities fee
Rent fee
Due date
payment_amount
late fee

4. **Normalization:**

Appoint:

UNF

appoint (owner_name, owner_id, owner_username, owner_email, owner_phone_no, manager_name, manager_id, manager_phone_no, manager_salary)

1NF

There is no multivalued attribute in 1NF.

1. owner_name, owner_id, owner_username, owner_email, owner_phone_no, manager_name, manager_id, manager_phone_no, manager_salary

2NF

1. owner_id, owner_name, owner_username, owner_email, owner_phone_no
2. manager_id, manager_name, manager_phone_no, manager_salary

3NF

There is no transitive dependency

1. owner_id, owner_name, owner_username, owner_email, owner_phone_no
2. manager_id, manager_name, manager_phone_no, manager_salary

Table Creation:

1. owner_id, owner_name, owner_username, owner_email, owner_phone_no
2. manager_id, manager_name, manager_phone_no, manager_salary, **owner_id**

Contacts:

UNF

contacts (tenant_id, tenant_username, tenant_email, tenant_phone_no, manager_id, manager_name, manager_phone_no, manager_salary)

1NF

There is no multivalued attribute in 1NF.

1. tenant_id, tenant_username, tenant_email, tenant_phone_no, manager_id, manager_name, manager_ phone_no, manager_salary

2NF

1. tenant_id, tenant_username, tenant_email, tenant_phone_no
2. manager_id, manager_name, manager_ phone_no, manager_salary

3NF

There is no transitive dependency

1. tenant_id, tenant_username, tenant_email, tenant_phone_no
2. manager_id, manager_name, manager_ phone_no, manager_salary

Table Creation:

1. tenant_id, tenant_username, tenant_email, tenant_phone_no, **manager_id**
2. manager_id, manager_name, manager_phone_no, manager_salary

Manage:

UNF

manage (manager_id, manager_name, manager_phone_no, manager_salary, apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address)

1NF

There is no multivalued attribute in 1NF

1. manager_id, manager_name, manager_phone_no, manager_salary, apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address

2NF

1. manager_id, manager_name, manager_phone_no, manager_salary
2. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address.

3NF

There is no transitive dependency

1. manager_id, manager_name, manager_phone_no, manager_salary
2. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address.

Table creation:

1. manager_id, manager_name, manager_phone_no, manager_salary
2. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, **manager_id, owner_id**

Provides:

UNF

provides (apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, facilities_id, gas, electrecity, water, security, lift)

1NF

There is no multivalued attribute in 1NF

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, facilities_id, gas, electrecity, water, security, lift

2NF

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address
2. facilities_id, gas, electrecity, water, security, lift

3NF

There is no transitive dependency

1. apartment_building_id, apartment_building_name, apartment_building_number, apartment_building_address
2. facilities_id, gas, electrecity, water, security, lift

Table creation

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address
2. facilities_id, gas, electrecity, water, security, lift, **apartment_building_id**

Owns:

UNF

owns (apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, owner_id, owner_name, owner_username, owner_email, owner_phone_no)

1NF

There is no multi valued attribute in 1NF

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, owner_id, owner_name, owner_username, owner_email, owner_phone_no

2NF

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address
2. owner_id, owner_name, owner_username, owner_email, owner_phone_no

3NF

There is no transitive dependency

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address
2. owner_id, owner_name, owner_username, owner_email, owner_phone_no

Table creation

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, **owner_id**
2. owner_id, owner_name, owner_username, owner_email, owner_phone_no

Has:

UNF

has (apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, apartment_id, apartment_rental_price, bedroom_count, bathroom_count, balcony_count)

1NF

There is three (bedroom_count, bathroom_count, balcony_count) multi valued attribute.

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, apartment_id, apartment_rental_price, bedroom_count, bathroom_count, balcony_count

2NF

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address
2. apartment_id, apartment_rental_price, bedroom_count, bathroom_count, balcony_count

3NF

There is no transitive dependency.

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address
2. apartment_id, apartment_rental_price, bedroom_count, bathroom_count, balcony_count

Table creation

1. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address
2. apartment_id, apartment_rental_price, bedroom_count, bathroom_count, balcony_count, **apartment_building_id**

Pays:

UNF

pay (rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, date, tenant_id, tenant_username, tenant_phone_no, tenant_email).

1NF

There is one (facilities_fee) multivalued attribute.

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, date, tenant_id, tenant_username, tenant_phone_no, tenant_email

2NF

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, date
2. tenant_id, tenant_username, tenant_phone_no, tenant_email

3NF

There is no transitive dependency.

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, date
2. tenant_id, tenant_username, tenant phone_no, tenant_email

Table creation

1. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, date
2. tenant_id, tenant_username, tenant_phone_no, tenant_email
3. **rent_id, tenant_id**

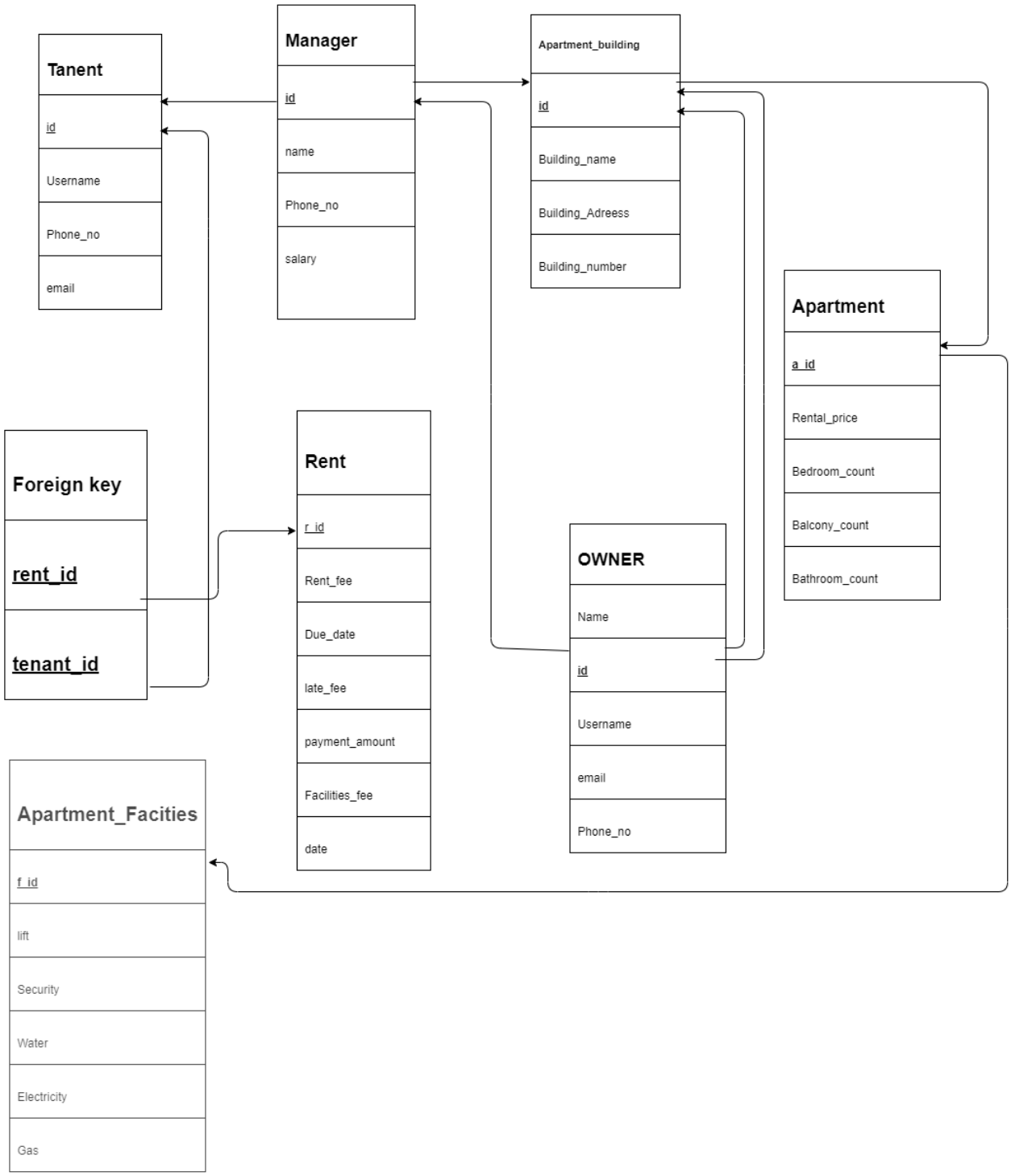
Temporary table

1. owner_id, owner_name, owner_username, owner_email, owner_phone_no
2. manager_id manager_name, manager_phone_no, manager_salary, **owner_id**
3. tenant_id, tenant_username, tenant_email, tenant_phone_no, **manager_id**
4. ~~manager_id, manager_name, manager_phone_no, manager_salary~~
5. ~~manager_id, manager_name, manager_phone_no, manager_salary~~
6. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, **manager_id, owner_id**
7. ~~apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address~~
8. facilities_id, gas, electrecity, water, security, lift, **apartment_building_id**
9. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, **owner_id**
10. ~~owner_id owner_name, owner_username, owner_email, owner_phone_no~~
11. ~~apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address~~
12. apartment_id, apartment_rental_price, bedroom_count, bathroom_count, balcony_count, **apartment_building_id**
13. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee, date
14. ~~tenant_id, tenant_username, tenant_phone_no, tenant_email~~
15. **rent_id, tenant_id**

Final Table

1. owner_id, owner_name, owner_username, owner_email, owner_phone_no
2. manager_id, manager_name, manager_phone_no, manager_salary, **owner_id**
3. tenant_id, tenant_username, tenant_email, tenant_phone_no, **manager_id**
4. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, **manager_id, owner_id**
5. facilities_id, gas, electrecity, water, security, lift, **apartment_building_id**
6. apartment_building_id, apartment_building_name, apt_building_no, apartment_building_address, **owner_id**
7. apartment_id, apartment_rental_price, bedroom_count1, bedroom_count2, bedroom_count3,bathroom_count1, bathroom_count2, bathroom_count3, balcony_count1, balcony_count2, balcony_count3, **apartment_building_id**
8. rent_id, late_fee, rent_fee, due_date, payment_amount, facilities_fee1, facilities_fee2, facilities_fee3,date
9. **rent_id, tenant_id**

5. Schema Diagram



6. Table Creation

User creation:

```
CREATE USER apt IDENTIFIED BY house101;
GRANT connect, resource, unlimited tablespace TO apt;
GRANT ALL PRIVILEGES TO apt;
```

Owner table:

CREATE TABLE Owner(owner_id number(10)PRIMARY KEY,owner_name varchar2(20),owner_username varchar2(12),email varchar2(30),phone number);

CREATE SEQUENCE ower_id_seq INCREMENT BY 1 START WITH 101 MAXVALUE 500 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	-	-	-	✓	-	-
	OWNER_USERNAME	Varchar2	12	-	-	-	✓	-	-
	EMAIL	Varchar2	30	-	-	-	✓	-	-
	PHONE	Number	-	-	-	-	✓	-	-

1 - 5

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

Tenant table:

CREATE TABLE Tenant(tanant_id number(10)PRIMARY KEY,tanant_username varchar2(20),email varchar2(30),phone number,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 101 START WITH 5 MAXVALUE 500 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_USERNAME	Varchar2	20	-	-	-	✓	-	-
	EMAIL	Varchar2	30	-	-	-	✓	-	-
	PHONE	Number	-	-	-	-	✓	-	-
	MANAGER_ID	Number	-	10	0	-	✓	-	-

1 - 5

Apartment Building table:

CREATE TABLE Apartment_Building(building_id number(10)PRIMARY KEY,building_number number(10),building_name varchar2(20),address varchar2(20),manager_id number(10),owner_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 101 START WITH 5 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_ID	Number	-	10	0	1	-	-	-
	BUILDING_NUMBER	Number	-	10	0	-	✓	-	-
	BUILDING_NAME	Varchar2	20	-	-	-	✓	-	-
	ADDRESS	Varchar2	20	-	-	-	✓	-	-
	MANAGER_ID	Number	-	10	0	-	✓	-	-
	OWNER_ID	Number	-	10	0	-	✓	-	-

1 - 6

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 101 START WITH 5 MAXVALUE 500 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

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_id);

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	Number	-	10	0	-	✓	-	-
	BATHROOM	Number	-	10	0	-	✓	-	-
	BALCONY	Number	-	10	0	-	✓	-	-
	APT_BUILDING_NO	Number	-	10	0	-	✓	-	-

1 - 6

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

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_id);

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

7. Data Insertion

Owner table:

```
INSERT INTO Owner VALUES(ower_id_seq.nextval,'Sergio Marquina','Professor','elpro@gmail.com',01712);
INSERT INTO Owner VALUES(ower_id_seq.nextval,'Thomas Shelby','Thomas','t.shelby@gmail.com',01713);
INSERT INTO Owner VALUES(ower_id_seq.nextval,'Jon Snow','Snow','knowsnothing@gmail.com',01714);
INSERT INTO Owner VALUES(ower_id_seq.nextval,'Walter White','WalWhite','walterbhai@gmail.com',01715);
INSERT INTO Owner VALUES(ower_id_seq.nextval, 'Ragnar Lothbrok','Vikings','ragnar.viki@gmail.com',01716);
select * from Owner;
```

Results Explain Describe Saved SQL History

OWNER_ID	OWNER_NAME	OWNER_USERNAME	EMAIL	PHONE
1	Sergio Marquina	Professor	elpro@gmail.com	1712
2	Thomas Shelby	Tommy	t.shelby@gmail.com	1713
3	Jon Snow	Snow	knowsnothing@gmail.com	1714
4	Walter White	WalWhite	walterbhai@gmail.com	1715
5	Ragnar Lothbrok	Vikings	ragnar.viki@gmail.com	1716

Manager table:

```
INSERT INTO Manager VALUES(manager_id_seq.nextval,'Christian Grey',20000,234564,1);
INSERT INTO Manager VALUES(manager_id_seq.nextval,'Anastasia Steele',15000,234566,2);
INSERT INTO Manager VALUES(manager_id_seq.nextval,'Massimo Torricelli',20000,234567,3);
INSERT INTO Manager VALUES(manager_id_seq.nextval,'Hardin Scott',10000,234568,4);
INSERT INTO Manager VALUES(manager_id_seq.nextval,'Tessa Young',50000,234569,5);
select * from Manager;
```

Results Explain Describe Saved SQL History

MANAGER_ID	MANAGER_NAME	SALARY	PHONE	OWNER_ID
1	Christian Grey	20000	234564	1
2	Anastasia Steele	15000	234566	2
3	Massimo Torricelli	20000	234567	3
4	Hardin Scott	10000	234568	4
5	Tessa Young	50000	234569	5

5 rows returned in 0.02 seconds CSV Export

Tenant table:

```
INSERT INTO Tenant VALUES(tanant_id_seq.nextval,'Jonas','jonas@gmail.com',354564,1);
INSERT INTO Tenant VALUES(tanant_id_seq.nextval,'Martha','martha@gmail.com',236864,2);
INSERT INTO Tenant VALUES(tanant_id_seq.nextval,'Otis','otis@gmail.com',446564,3);
INSERT INTO Tenant VALUES(tanant_id_seq.nextval,'Meave','meave@gmail.com',887664,4);
INSERT INTO Tenant VALUES(tanant_id_seq.nextval,'Carla','carla@gmail.com',345454,5);
select * from tenant;
```

Results Explain Describe Saved SQL History

TANANT_ID	TANANT_USERNAME	EMAIL	PHONE	MANAGER_ID
1	Jonas	jonas@gmail.com	354564	1
2	Martha	martha@gmail.com	236864	2
3	Otis	otis@gmail.com	446564	3
4	Meave	meave@gmail.com	887664	4
5	Carla	carla@gmail.com	345454	5

Apartment Building table:

```
INSERT INTO Apartment_Building VALUES(101,4353,'Shelby Home','Birmingham',1,1);
INSERT INTO Apartment_Building VALUES(102,3244,'House Stark','North-Winterfell',2,2);
INSERT INTO Apartment_Building VALUES(103,4545,'House Lannister','Casterly Rock',3,3);
INSERT INTO Apartment_Building VALUES(104,6536,'The Palace','Kings Landing',4,4);
INSERT INTO Apartment_Building VALUES(105,7857,'Valhalla','Asgard',5,5);

select * from Apartment_Building;
```

Results Explain Describe Saved SQL History

BUILDING_ID	BUILDING_NUMBER	BUILDING_NAME	ADDRESS	MANAGER_ID	OWNER_ID
101	4353	Shelby Home	Birmingham	1	1
102	3244	House Stark	North-Winterfell	2	2
103	4545	House Lannister	Casterly Rock	3	3
104	6536	The Palace	Kings Landing	4	4
105	7857	Valhalla	Asgard	5	5

Rent table:

```
INSERT INTO Rent VALUES(1,5000,200,4000,2500,500,TO_DATE('12-1-2021','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('02-08-2021','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	200	4000	2500	500	12-JAN-21
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-21

5 rows returned in 0.00 seconds CSV Export

Apartment table:

```
INSERT INTO Apartment VALUES(1,5000,3,3,2,101);
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	BATHROOM	BALCONY	APT_BUILDING_NO
1	5000	3	3	2	101
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 0.00 seconds CSV Export

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
5	5
4	4

5 rows returned in 0.00 seconds CSV Export

Facility table:

```
INSERT INTO Facility VALUES(1,'available','available','available','secured','not available',101);
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	not available	101
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

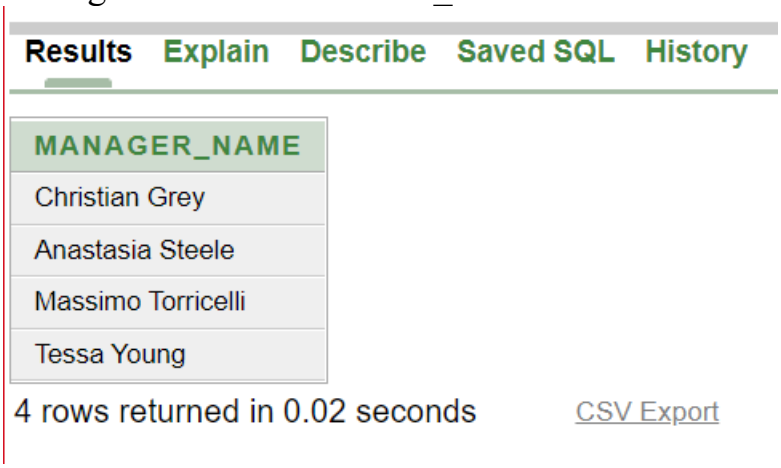
5 rows returned in 0.00 seconds CSV Export

8. Query Writing

SUB-QUERY:

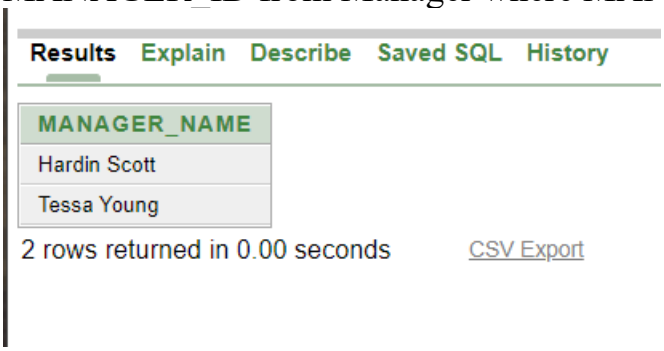
1. Display the manager who earns more than Hardin Scott.

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



2. Display the manager name who join after Massimo Torricelli.

➢ select MANAGER_NAME from Manager where MANAGER_ID>(select MANAGER_ID from Manager where MANAGER_NAME='Massimo Torricelli');



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;

Results Explain Describe Saved SQL History

TANANT_USERNAME	BUILDING_NAME
Jonas	Shelby Home
Martha	House Stark
Otis	House Lannister
Meave	The Palace
Carla	Valhalla

5 rows returned in 0.01 seconds CSV Export

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

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

Results Explain Describe Saved SQL History

RENT_PRICE	BEDROOM	GAS	LIFT
5000	3	available	not available
7000	4	available	available
2000	1	available	not available
5000	3	available	not available
8000	4	available	available

5 rows returned in 0.00 seconds CSV Export

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	2500	500
7000	0	400
2000	2200	200
5000	3400	100
8000	4400	300

5 rows returned in 0.00 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,BATHROOM,APT_BUILDING_NO from Apartment;
Select * from APARTMENTVIEW;

Results Explain Describe Saved SQL History

RENT_PRICE	BEDROOM	BATHROOM	APT_BUILDING_NO
5000	3	3	101
7000	4	4	102
2000	1	2	103
5000	3	3	104
8000	4	5	105

9. Relational Algebra:

1. Find the manager_name where m_salary is greater than 2000.

   ➢ πmanager_name(σ manager_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” $^{(Manager)}$) {Table: MANAGER}

3. Find the apt_building_no where rental_price is greater than 7000.

   ➢ π apt_building_no (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}

10. Conclusion:

This project is relatively simple to understand and implement. It fulfills all the current requirements of a 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 six relationships among all the entities with cardinality.
In our project, we have mentioned the queries and inserted screenshots of the tables we created using those queries. The normalization process makes this project simpler.
This project overall covers the following fields:

a. **Apartment details:** This AMS stores all the information, records, and data related to the apartment. These data include the total number of flats and rooms, types of rooms.
b. **Personal Information:** Personal information of every manager & member is also stored in this system.
c. **Facilities Provided:** Whether an apartment provides facilities (water, gas, electricity, security) or not is also mentioned in this system.
d. **Rent Collection:** This system also gives solutions to rent collection issues.

Difficulties & Problems

o **Leak of personal information:** Personal information of managers and members may leak when someone searches for an available apartment.

o **Less Security:** Due to lack of proper security, this system can be easily hacked.

o **Parking Slot Confusion:** This system doesn't give any solution to mark parking slots for each member individually.

11. Future Development:

Increasing the system's security will eliminate the risk of data leakage and hacking. The parking spot issue will be resolved if each unit is assigned a unique id. The addition of GPS tracking will enhance this system with an intelligent touch. People will be able to navigate online for vacant apartment buildings. Furthermore, the database must gather a large amount of data for one or more units in a building, making system upgrades and data storage extremely expensive. We will focus on making pricing easier in the future. If these issues are resolved, it is expected that this system will become more efficient, user-friendly, and convenient in the future. We may add, modify, update, or delete something at any time.

Finally, in response to user demands, the system may be updated with additional functionality. The project is very flexible in that sense.

Thank You

Best Regards

Avro aka Alen
avroalen@gmail.com
t.me/a4vro