DBMS Project (UCS310)

Submitted By-

Vasudhara Mahajan	102103032
Iyasha Goyal	102103034
Kunal Garg	102103038
Sanjana Sinha	102103040

Group: 1 COE2

Submitted to-

Nitigya Sambyal Mam

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTIUTE OF ENGINEERING AND TECHNOLOGY,

(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB, INDIA

JAN-MAY 2023

INDEX-

Introduction:

The purpose of this apartment management database system is to manage the various activities of an apartment complex. This database system will help apartment managers to keep track of residents, apartment units, rent payments, maintenance requests, and other activities.

Requirement Analysis:

To create an effective database system for apartment management, we need to identify the requirements of the system. Here are some of the key requirements for this project:

1. Resident Management:

The system should allow the apartment manager to add new residents and update their details. The system should also provide functionality to delete residents who have moved out. Additionally, the system should store information about the resident's lease, including the start and end dates.

2. Apartment Unit Management:

The system should allow the apartment manager to add new apartment units and update their details. The system should also provide functionality to delete apartment units that are no longer available. Additionally, the system should store information about the apartment unit's size, number of rooms, and rental rate.

3. Rent Payment Management:

The system should allow residents to make rent payments. The system should also store information about rent payments, including the date and amount paid. The system should also provide functionality to generate reports on rent payments for a given period.

4. Maintenance Request Management:

The system should allow residents to submit maintenance requests, and the apartment manager should be able to track the progress of those requests. The system should store information about the maintenance request, including the date it was submitted, the type of request, and the status of the request.

5. Reporting:

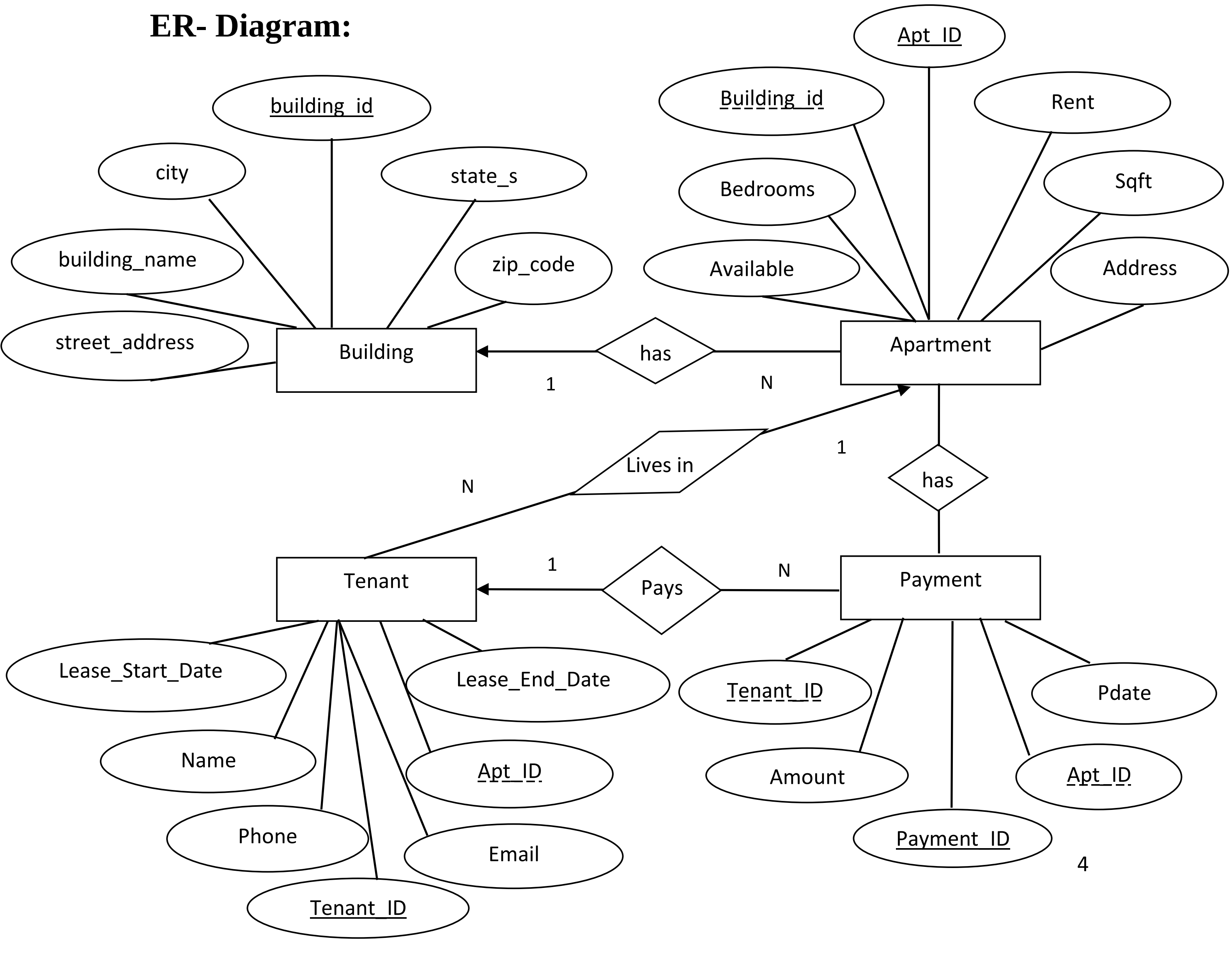
The system should provide functionality to generate reports on various aspects of apartment management, such as rent payments, maintenance requests, and occupancy rates.

6. Security:

The system should ensure the security of resident and apartment unit data. The system should provide user authentication and authorization to ensure that only authorized personnel can access sensitive data.

With these requirements in mind, we can start designing and implementing the database system using SQL.

ER- Diagram:



ER to Table: By following the ER model we have the following four tables-

Table 1. Building (building_id primary key, building_name, street_address, city, state_s, zip_code).

Table 2. Apartment(with relation has) (Apt_ID primary key, Address, Bedrooms, Bathrooms, Sqft, Rent, Available, building_id foreign key).

Table 3. Tenant(with relation lives in) (Tenant_ID primary key, Name, Phone, Email, Apt_ID foreign key, Lease_Start_Date, Lease_End_Date).

Table 4. Payment(with relation pays,has) (Payment_ID primary key, Tenant_ID and Apt_ID foreign key, Amount, Pdate).

Normalization:

To apply Third Normal Form (3NF) on the given tables, we need to eliminate any transitive dependencies, partial dependencies that exist.

1. Building table: The Building table is already in 3NF, so no changes are necessary.

2. Apartment table: The Apartment table is currently in 2NF. We need to split the table to create a new table for Building information and another table for Availability.

```
CREATE TABLE Building_Info (

   building_id INT PRIMARY KEY,

   building_name VARCHAR(30),

   street_address VARCHAR(30),

   city VARCHAR(20),

   state_s VARCHAR(20),

   zip_code INT );
```

```
CREATE TABLE Availability (

    apt_id INT PRIMARY KEY,

    available VARCHAR(20),

    FOREIGN KEY (apt_id) REFERENCES Apartment(Apt_ID)

);

ALTER TABLE Apartment

    ADD building_id INT,

    ADD CONSTRAINT fk_building_id

    FOREIGN KEY (building_id) REFERENCES Building_Info(building_id);
```

Now the Apartment table only contains information about the apartment itself and a reference to the Building_Info table that contains information about the building. The Availability table stores the availability of each apartment.

3. Tenant table: The Tenant table is currently in 2NF. We need to create a new table for Tenant information, and another table for Lease information to remove the transitive dependency between Tenant and Lease.

```
CREATE TABLE Tenant_Info (

    tenant_id INT PRIMARY KEY,

    name VARCHAR(50),

    phone VARCHAR(15),

    email VARCHAR(50)

);
```

```
CREATE TABLE Lease (
    lease_id INT PRIMARY KEY,
    apt_id INT,
    tenant_id INT,
    lease_start_date DATE,
    lease_end_date DATE,
    FOREIGN KEY (apt_id) REFERENCES Apartment(apt_id),
    FOREIGN KEY (tenant_id) REFERENCES Tenant_Info(tenant_id)
);
ALTER TABLE Tenant
    ADD lease_id INT,
    ADD CONSTRAINT fk_lease_id
    FOREIGN KEY (lease_id) REFERENCES Lease(lease_id);
```

Now the Tenant table only contains information about the tenant, and a reference to the Lease table that contains information about the lease.

4. Payment table: The Payment table is currently in 2NF. We need to remove the transitive dependency by creating a new table for Payment information and referencing Tenant and Apartment tables.

```
CREATE TABLE Payment_Info (
    payment_id INT PRIMARY KEY,
    amount DECIMAL(10, 2),
    pdate DATE
);
```

```
ALTER TABLE Payment
    ADD payment_id INT,
    ADD CONSTRAINT fk_payment_id
    FOREIGN KEY (payment_id) REFERENCES Payment_Info(payment_id);
```

Now the Payment table only contains information about the payment, and a reference to the Payment_Info table that contains information about the payment. After the above changes, the tables are in 3NF, which means that there are no transitive dependencies in the tables.

SQL & PL/SQL with Output:

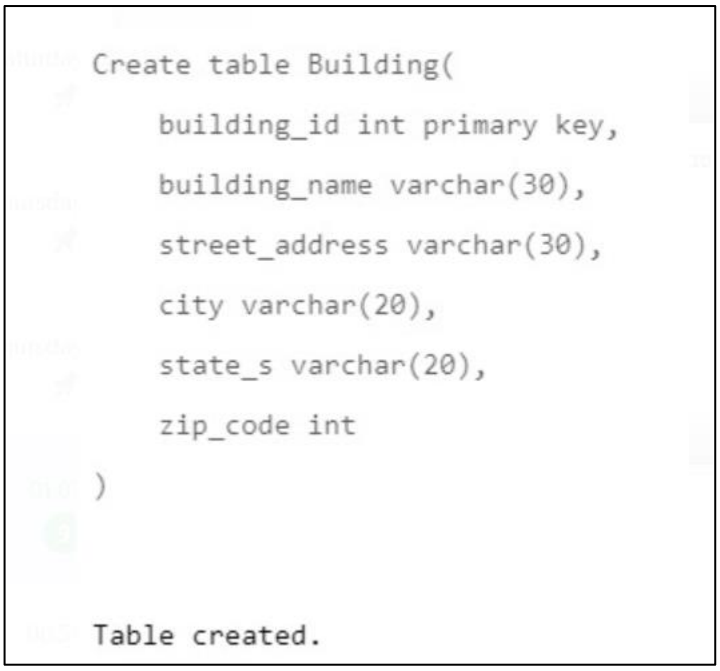
```
Create table Building(
    building_id int primary key,
    building_name varchar(30),
    street_address varchar(30),
    city varchar(20),
    state_s varchar(20),
    zip_code int
);
CREATE TABLE Apartment (
    Apt_ID INT PRIMARY KEY,
    Address VARCHAR(100),
    Bedrooms INT,
    Bathrooms INT,
    Sqft INT,
    Rent DECIMAL(10, 2),
    Available varchar(20),
    building_id int, foreign key (building_id) references Building(building_id)
);
CREATE TABLE Tenant (
    Tenant_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Phone VARCHAR(15),
    Email VARCHAR(50),
    Apt_ID INT,
    Lease_Start_Date DATE,
    Lease_End_Date DATE,
    FOREIGN KEY (Apt_ID) REFERENCES Apartment(Apt_ID)
);
CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY,
    Tenant_ID INT,
    Apt_ID INT,
    Amount DECIMAL(10, 2),
    Pdate DATE,
    FOREIGN KEY (Tenant_ID) REFERENCES Tenant(Tenant_ID),
    FOREIGN KEY (Apt_ID) REFERENCES Apartment(Apt_ID)
);
```

```
insert into Building values (1, 'Central Towers', '123 Main St', 'New York', 'NY',
10001);

insert into Building values (2, 'The Palms', '456 Elm St', 'Los Angeles', 'CA',
90001);

insert into Building values (3, 'City View Apartments', '789 Oak St', 'Chicago',
'IL', 60601);

insert into Building values (4, 'Park Place', '234 Maple Ave', 'San Francisco',
'CA', 94101);

insert into Building values (5, 'Riverfront Residences', '567 Pine St', 'Miami',
'FL', 33101);


insert into Apartment  values (1,'A101',2,2,4.1,1500,'yes',1);

insert into Apartment  values (2,'A102',3,2,4.5,1500,'no',2);

insert into Apartment  values (3,'A103',4,3,5.0,1500,'yes',3);

insert into Apartment  values (4,'A103',4,2,4.0,1500,'no',4);

insert into Apartment  values (5,'A105',3,2,5.0,1500,'yes',5);


insert into Tenant  values (4,'Manav','9874562565','manav5@gmail.com',3,'24-
may-2000','24-may-2010');

insert into Tenant  values (5,'Sam','8874562565','Sam5@gmail.com',1,'4-may-
2000','24-may-2011');

insert into Tenant  values (3,'anav','7874562565','anav5@gmail.com',2,'2-may-
2000','24-may-2009');

insert into Tenant  values (1,'arnav','7864562565','arnav5@gmail.com',4,'2-may-
2001','2-may-2011');

insert into Tenant  values (2,'manav','7984562565','manav5@gmail.com',5,'3-
may-2001','4-may-2009');
```

```
insert into Payment values (7,4,3,500,'30-may-2000');

insert into Payment values (8,5,1,800,'24-may-2000');

insert into Payment values (9,3,2,700,'20-may-2000');

insert into Payment values (16,1,4,800,'21-may-2000');

insert into Payment values (17,2,5,900,'22-may-2000');


--QUERIES-

select * from Building;

select * from Apartment;

select * from Tenant;

select * from Payment;

select * from  Apartment where Available = 'yes';


SELECT * FROM Tenant WHERE Tenant_ID = 5;

SELECT * FROM Payment WHERE Tenant_ID = 2;

SELECT * FROM building WHERE building_name= 'Riverfront Residences';

UPDATE Apartment SET Rent = 2000 WHERE Apt_ID = 3;


SELECT Tenant.Tenant_ID, Tenant.Name, Tenant.Phone, Tenant.Email,
Apartment.Address, Apartment.Rent FROM Tenant JOIN Apartment ON
Tenant.Apt_ID = Apartment.Apt_ID;


SELECT Apartment.Address, Tenant.Name, Tenant.Phone, Tenant.Email

FROM Apartment LEFT JOIN Tenant ON Apartment.Apt_ID = Tenant.Apt_ID

WHERE Apartment.Available = 'yes';
```

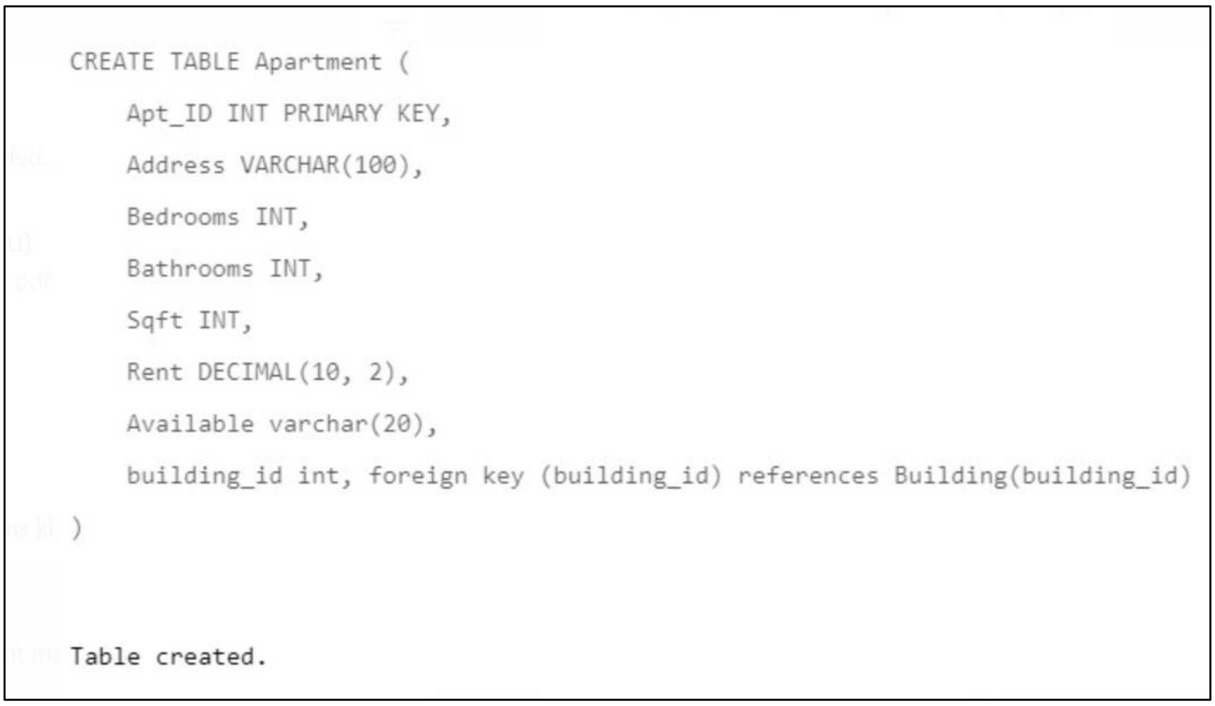
```
SELECT Tenant.Name, Apartment.Address, Payment.Amount, Payment.Pdate

FROM Tenant JOIN Apartment ON Tenant.Apt_ID = Apartment.Apt_ID

JOIN Payment ON Tenant.Tenant_ID = Payment.Tenant_ID;


SELECT Tenant.Name, Apartment.Address, Payment.Amount, Payment.Pdate

FROM Tenant Right JOIN Apartment ON Tenant.Apt_ID = Apartment.Apt_ID

JOIN Payment ON Tenant.Tenant_ID = Payment.Tenant_ID;


SELECT Apartment.Address, Building.building_Name

FROM Apartment JOIN Building ON Apartment.building_ID =
Building.building_ID;


-- SQL function to get the total rent paid by a tenant for all apartments they have
rented:

SELECT Tenant.Name, SUM(Payment.Amount) AS TotalRentPaid

FROM Tenant JOIN Payment ON Tenant.Tenant_ID = Payment.Tenant_ID

GROUP BY Tenant.Name;


-- SQL function to get the average rent for apartments in a building:

SELECT Building.building_name, AVG(Apartment.Rent) AS AvgRent

FROM Apartment JOIN Building ON Apartment.Address LIKE '%' ||
Building.street_address || '%'  GROUP BY Building.building_name;
```

Table Creation- Table 1



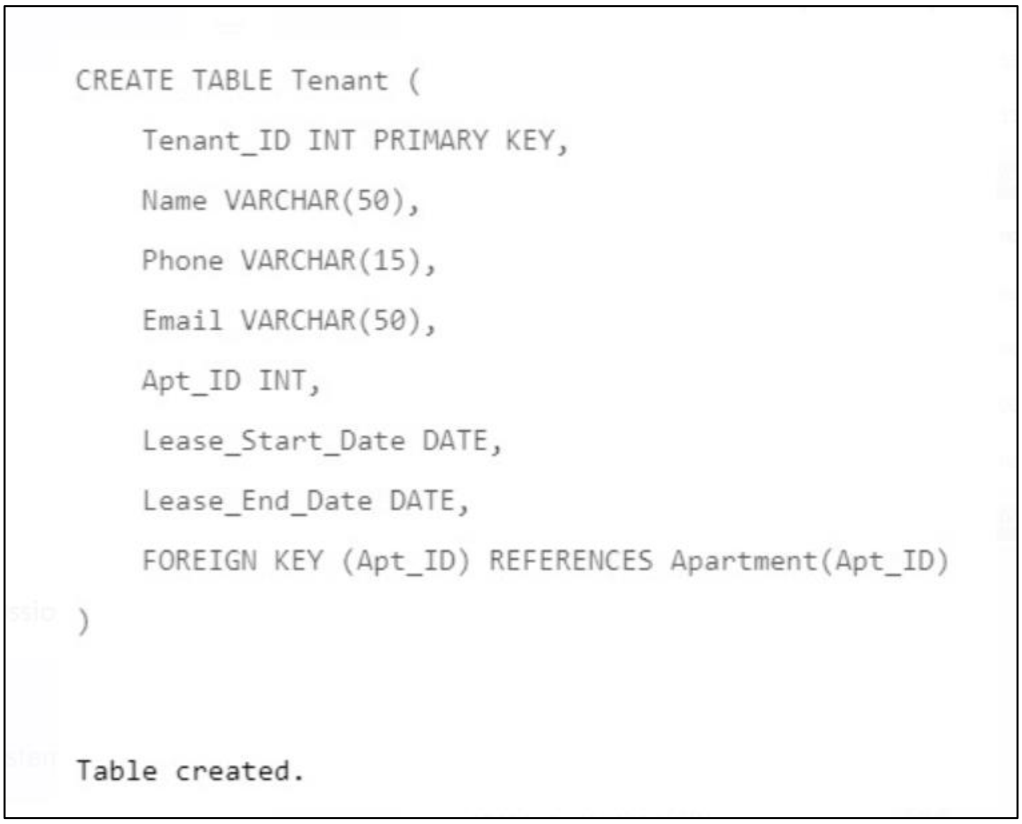
```
Create table Building(
    building_id int primary key,
    building_name varchar(30),
    street_address varchar(30),
    city varchar(20),
    state_s varchar(20),
    zip_code int
)


Table created.
```

Table 2



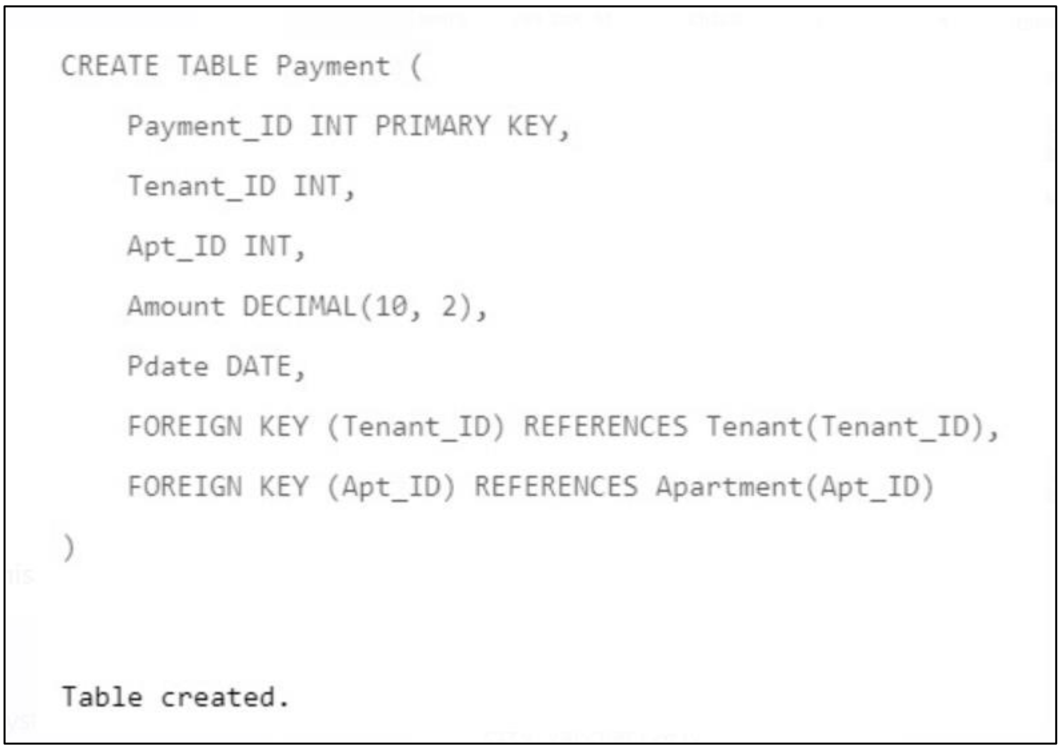
```
CREATE TABLE Apartment (
    Apt_ID INT PRIMARY KEY,
    Address VARCHAR(100),
    Bedrooms INT,
    Bathrooms INT,
    Sqft INT,
    Rent DECIMAL(10, 2),
    Available varchar(20),
    building_id int, foreign key (building_id) references Building(building_id)
)


Table created.
```

Table 3



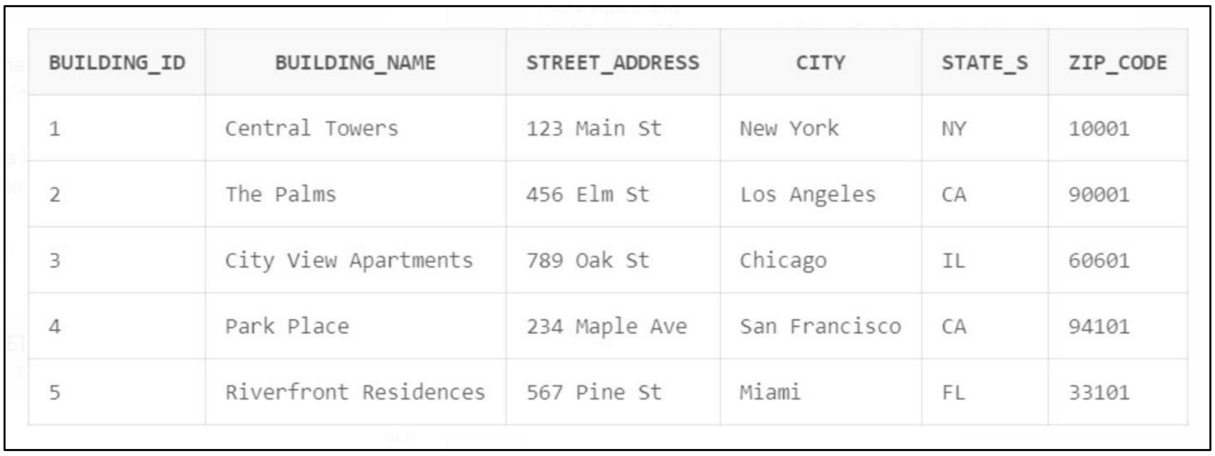
```
CREATE TABLE Tenant (
    Tenant_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Phone VARCHAR(15),
    Email VARCHAR(50),
    Apt_ID INT,
    Lease_Start_Date DATE,
    Lease_End_Date DATE,
    FOREIGN KEY (Apt_ID) REFERENCES Apartment(Apt_ID)
)


Table created.
```

Table 4

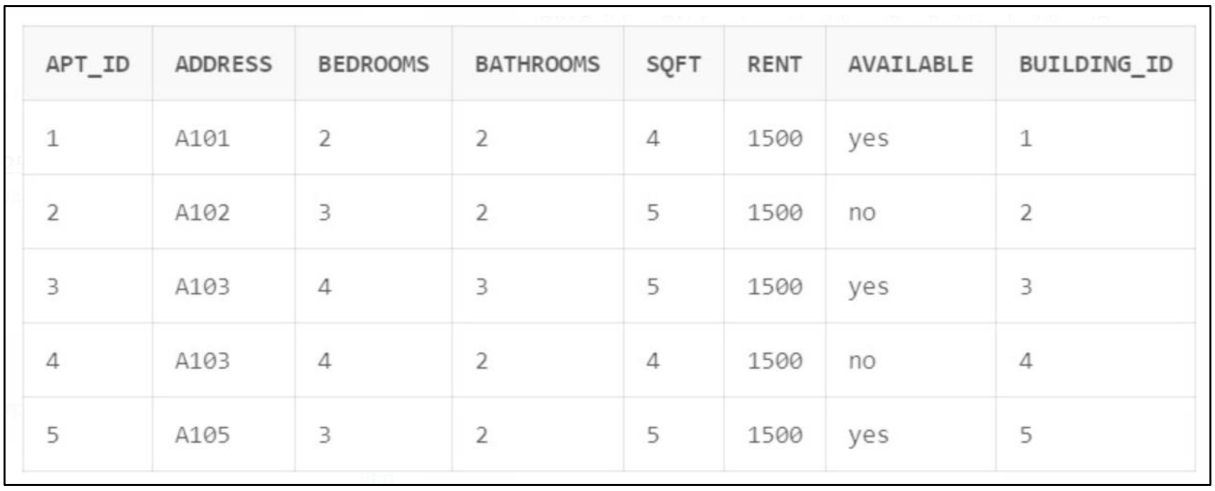


```
CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY,
    Tenant_ID INT,
    Apt_ID INT,
    Amount DECIMAL(10, 2),
    Pdate DATE,
    FOREIGN KEY (Tenant_ID) REFERENCES Tenant(Tenant_ID),
    FOREIGN KEY (Apt_ID) REFERENCES Apartment(Apt_ID)
)


Table created.
```

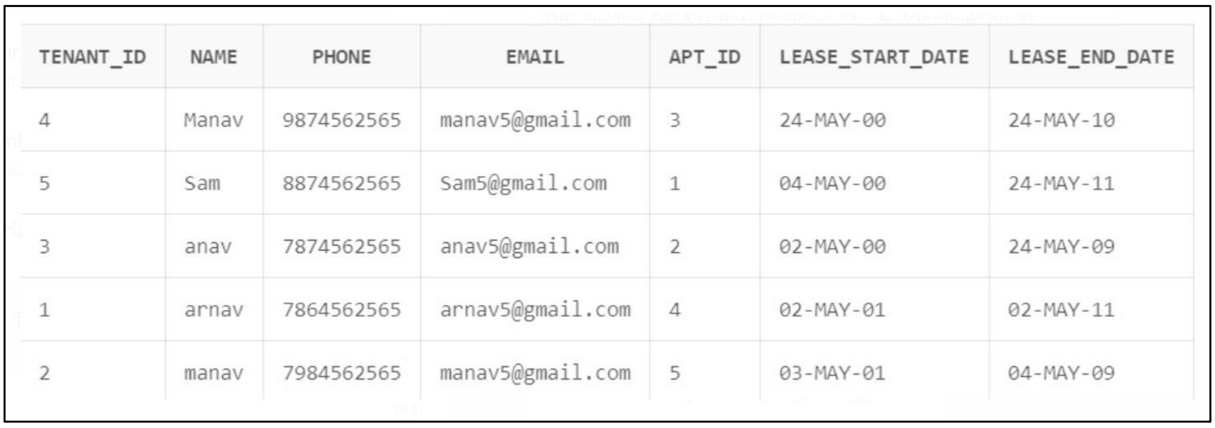
Outputs-



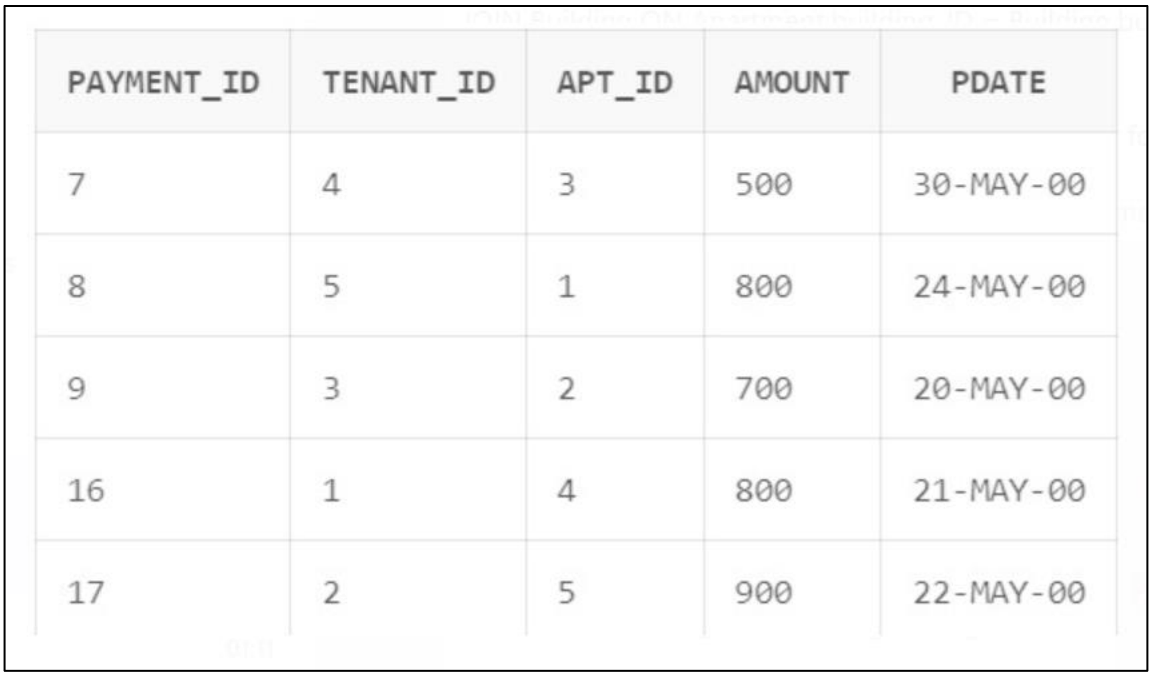
BUILDING_ID	BUILDING_NAME	STREET_ADDRESS	CITY	STATE_S	ZIP_CODE
1	Central Towers	123 Main St	New York	NY	10001
2	The Palms	456 Elm St	Los Angeles	CA	90001
3	City View Apartments	789 Oak St	Chicago	IL	60601
4	Park Place	234 Maple Ave	San Francisco	CA	94101
5	Riverfront Residences	567 Pine St	Miami	FL	33101



APT_ID	ADDRESS	BEDROOMS	BATHROOMS	SQFT	RENT	AVAILABLE	BUILDING_ID
1	A101	2	2	4	1500	yes	1
2	A102	3	2	5	1500	no	2
3	A103	4	3	5	1500	yes	3
4	A103	4	2	4	1500	no	4
5	A105	3	2	5	1500	yes	5

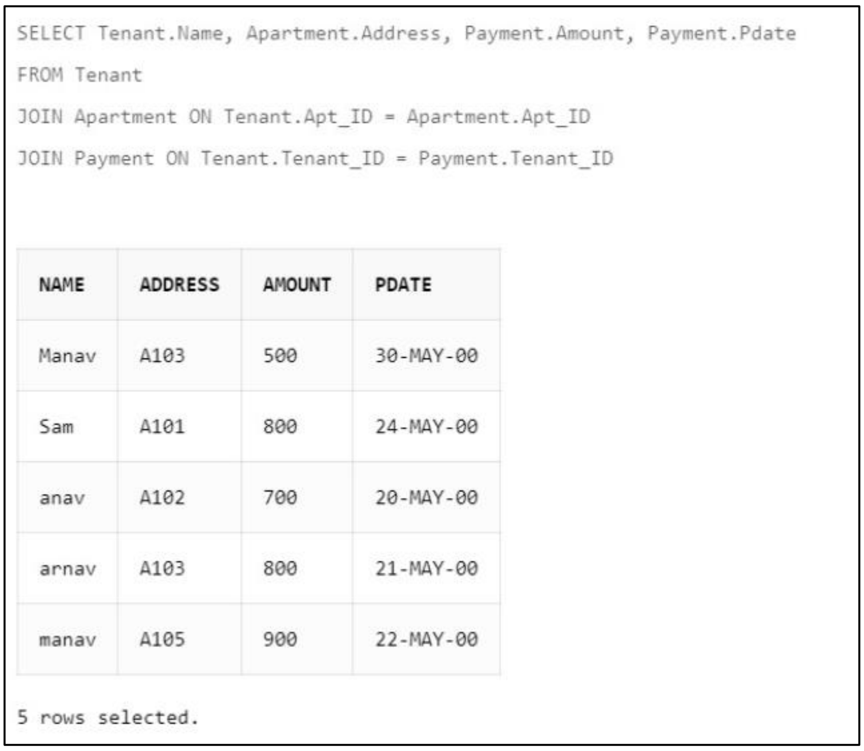


TENANT_ID	NAME	PHONE	EMAIL	APT_ID	LEASE_START_DATE	LEASE_END_DATE
4	Manav	9874562565	manav5@gmail.com	3	24-MAY-00	24-MAY-10
5	Sam	8874562565	Sam5@gmail.com	1	04-MAY-00	24-MAY-11
3	anav	7874562565	anav5@gmail.com	2	02-MAY-00	24-MAY-09
1	arnav	7864562565	arnav5@gmail.com	4	02-MAY-01	02-MAY-11
2	manav	7984562565	manav5@gmail.com	5	03-MAY-01	04-MAY-09



PAYMENT_ID	TENANT_ID	APT_ID	AMOUNT	PDATE
7	4	3	500	30-MAY-00
8	5	1	800	24-MAY-00
9	3	2	700	20-MAY-00
16	1	4	800	21-MAY-00
17	2	5	900	22-MAY-00

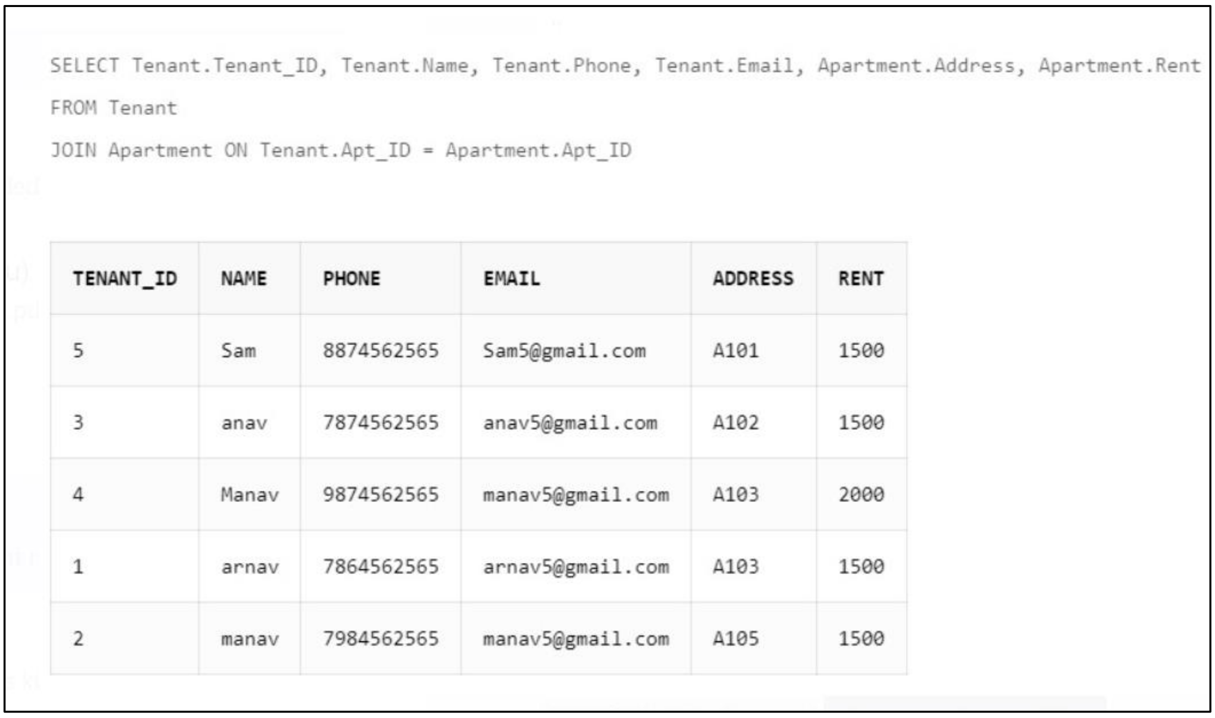
Some Queries-



```
SELECT Tenant.Name, Apartment.Address, Payment.Amount, Payment.Pdate
FROM Tenant
JOIN Apartment ON Tenant.Apt_ID = Apartment.Apt_ID
JOIN Payment ON Tenant.Tenant_ID = Payment.Tenant_ID
```

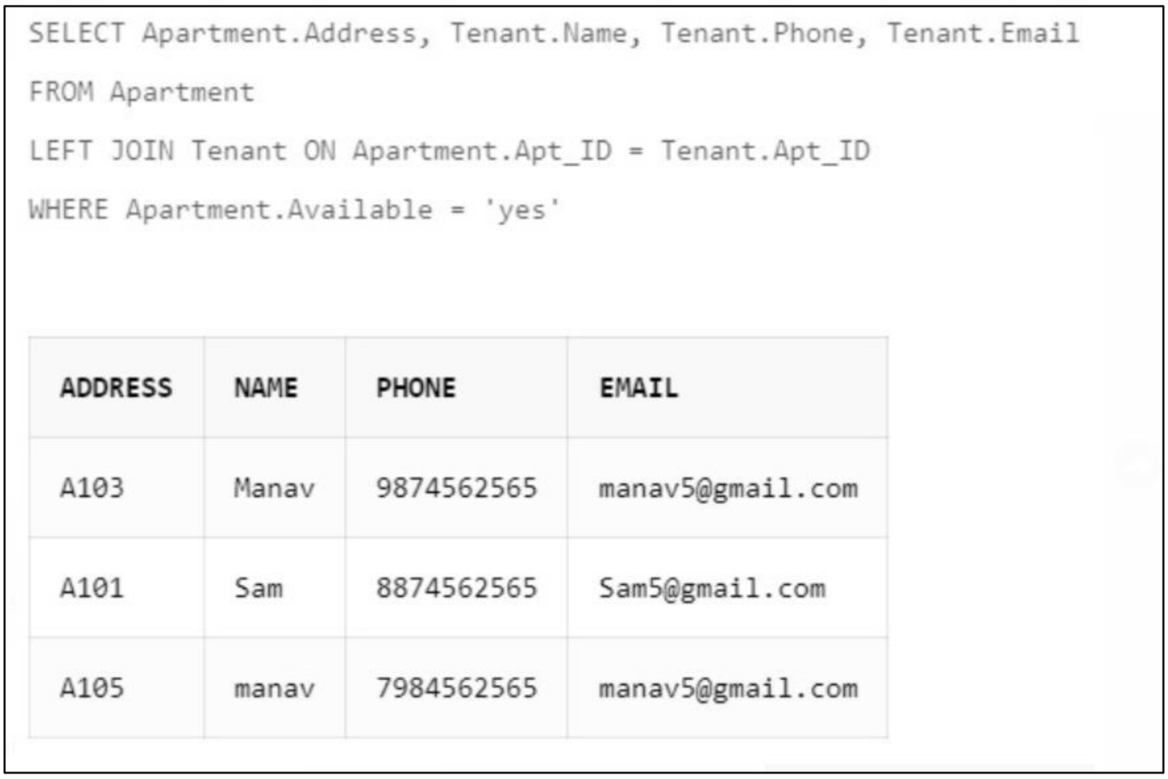
NAME	ADDRESS	AMOUNT	PDATE
Manav	A103	500	30-MAY-00
Sam	A101	800	24-MAY-00
anav	A102	700	20-MAY-00
arnav	A103	800	21-MAY-00
manav	A105	900	22-MAY-00

```
5 rows selected.
```



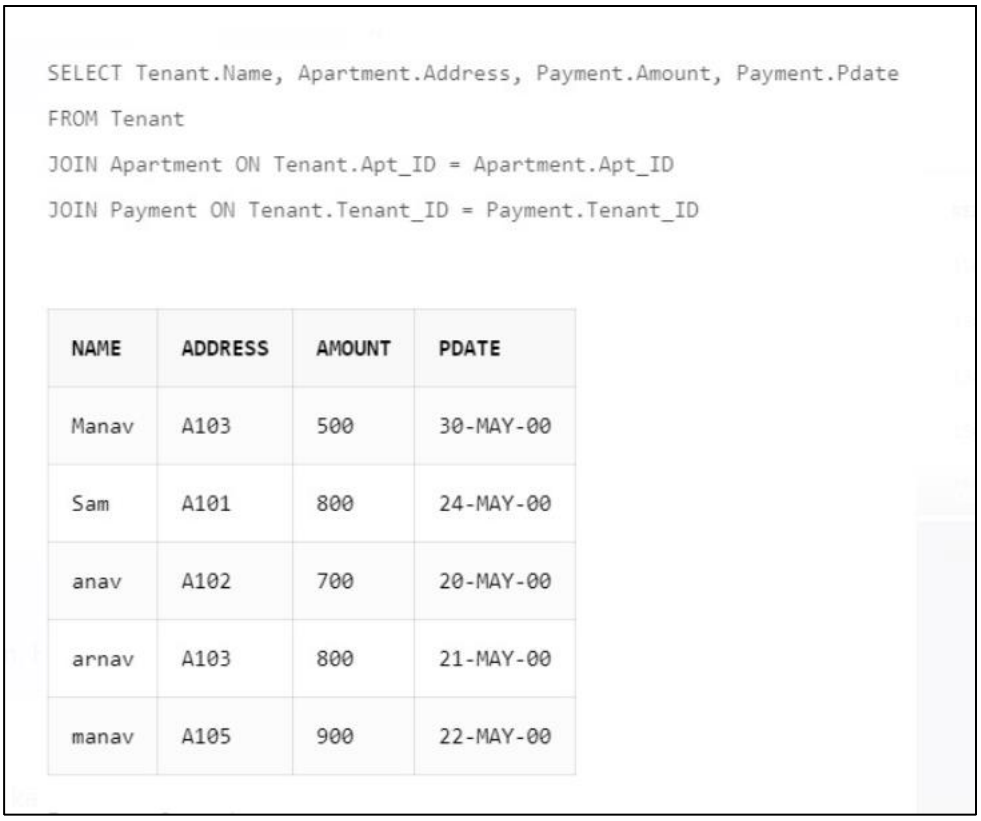
```
SELECT Tenant.Tenant_ID, Tenant.Name, Tenant.Phone, Tenant.Email, Apartment.Address, Apartment.Rent
FROM Tenant
JOIN Apartment ON Tenant.Apt_ID = Apartment.Apt_ID
```

TENANT_ID	NAME	PHONE	EMAIL	ADDRESS	RENT
5	Sam	8874562565	Sam5@gmail.com	A101	1500
3	anav	7874562565	anav5@gmail.com	A102	1500
4	Manav	9874562565	manav5@gmail.com	A103	2000
1	arnav	7864562565	arnav5@gmail.com	A103	1500
2	manav	7984562565	manav5@gmail.com	A105	1500



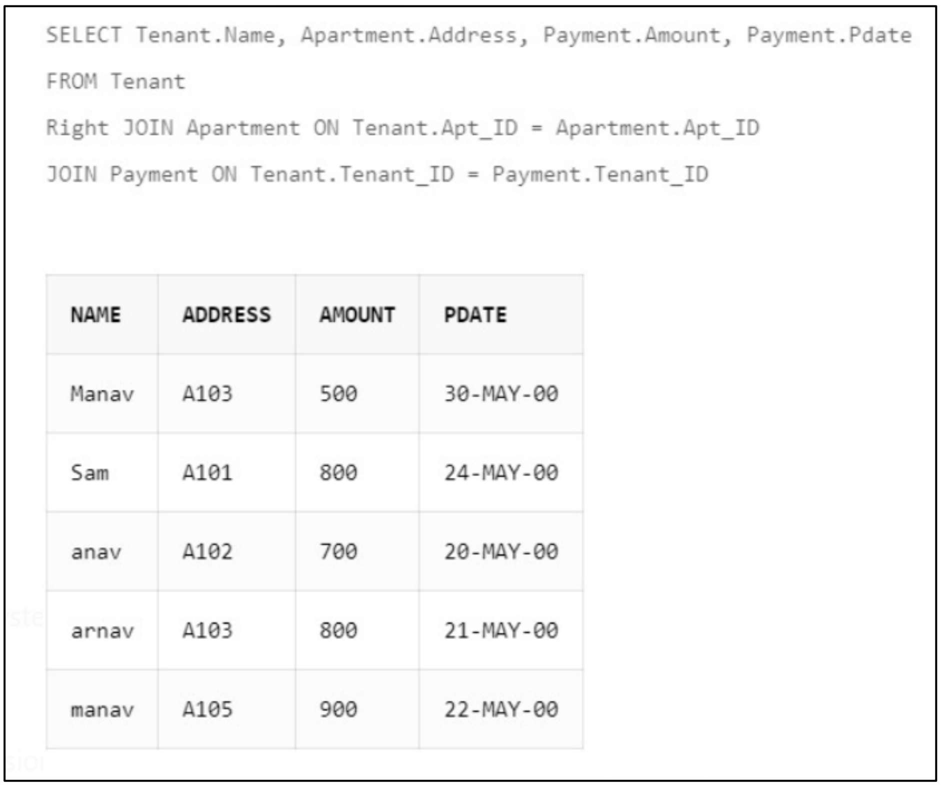
```
SELECT Apartment.Address, Tenant.Name, Tenant.Phone, Tenant.Email
FROM Apartment
LEFT JOIN Tenant ON Apartment.Apt_ID = Tenant.Apt_ID
WHERE Apartment.Available = 'yes'
```

ADDRESS	NAME	PHONE	EMAIL
A103	Manav	9874562565	manav5@gmail.com
A101	Sam	8874562565	Sam5@gmail.com
A105	manav	7984562565	manav5@gmail.com



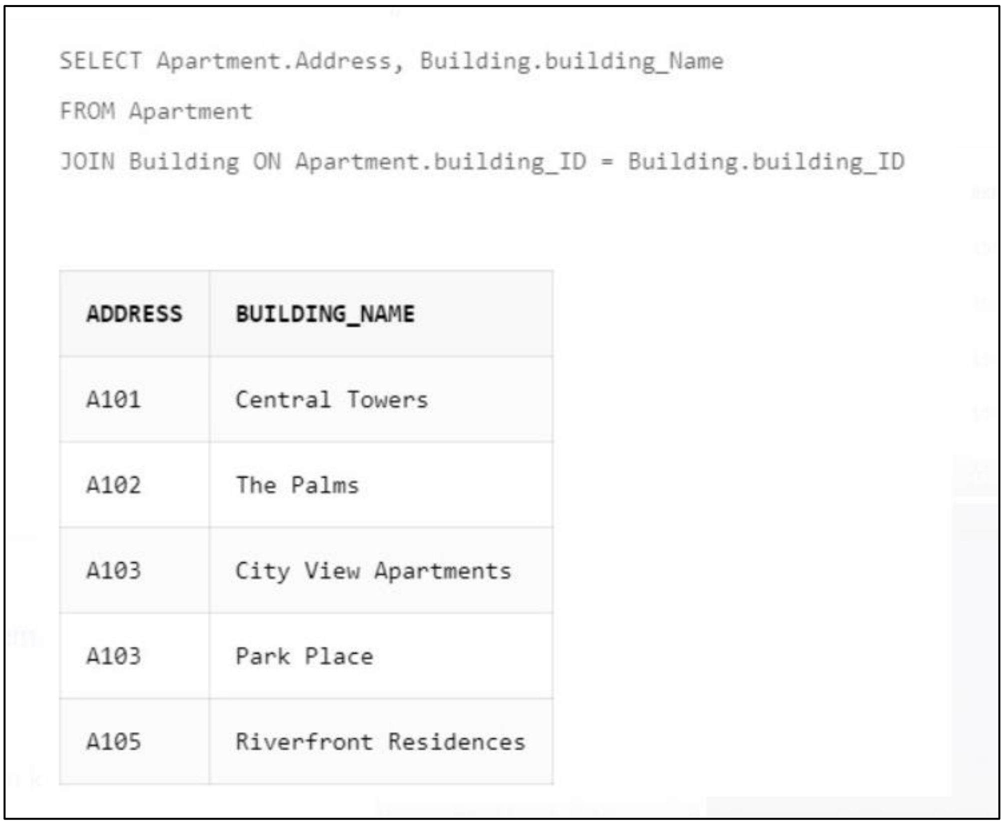
```
SELECT Tenant.Name, Apartment.Address, Payment.Amount, Payment.Pdate
FROM Tenant
JOIN Apartment ON Tenant.Apt_ID = Apartment.Apt_ID
JOIN Payment ON Tenant.Tenant_ID = Payment.Tenant_ID
```

NAME	ADDRESS	AMOUNT	PDATE
Manav	A103	500	30-MAY-00
Sam	A101	800	24-MAY-00
anav	A102	700	20-MAY-00
arnav	A103	800	21-MAY-00
manav	A105	900	22-MAY-00



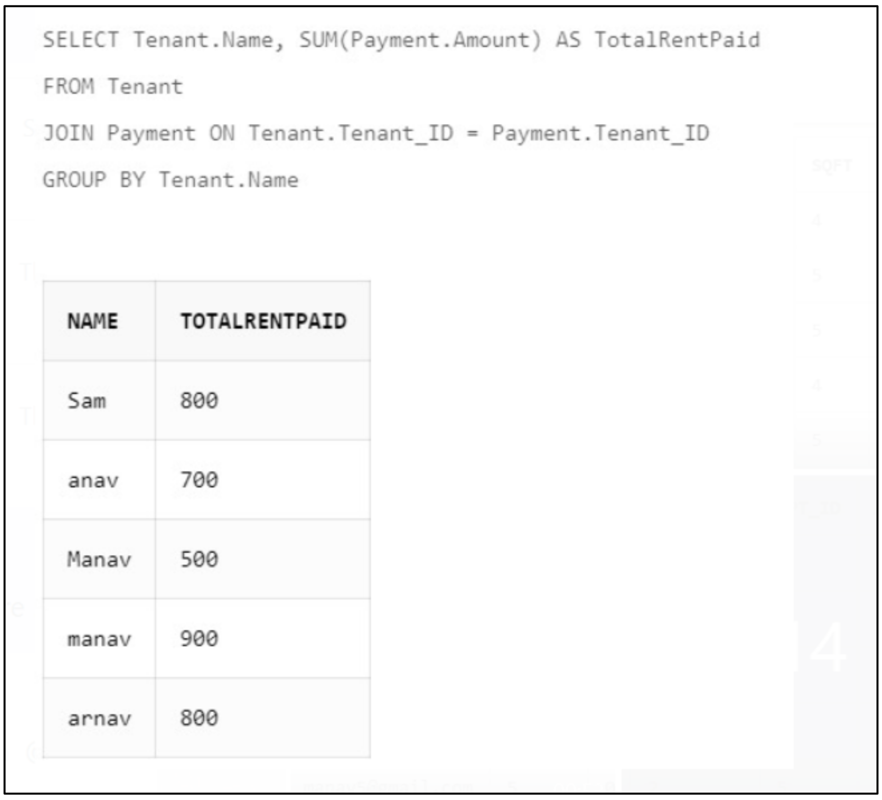
```
SELECT Tenant.Name, Apartment.Address, Payment.Amount, Payment.Pdate
FROM Tenant
Right JOIN Apartment ON Tenant.Apt_ID = Apartment.Apt_ID
JOIN Payment ON Tenant.Tenant_ID = Payment.Tenant_ID
```

NAME	ADDRESS	AMOUNT	PDATE
Manav	A103	500	30-MAY-00
Sam	A101	800	24-MAY-00
anav	A102	700	20-MAY-00
arnav	A103	800	21-MAY-00
manav	A105	900	22-MAY-00



```
SELECT Apartment.Address, Building.building_Name
FROM Apartment
JOIN Building ON Apartment.building_ID = Building.building_ID
```

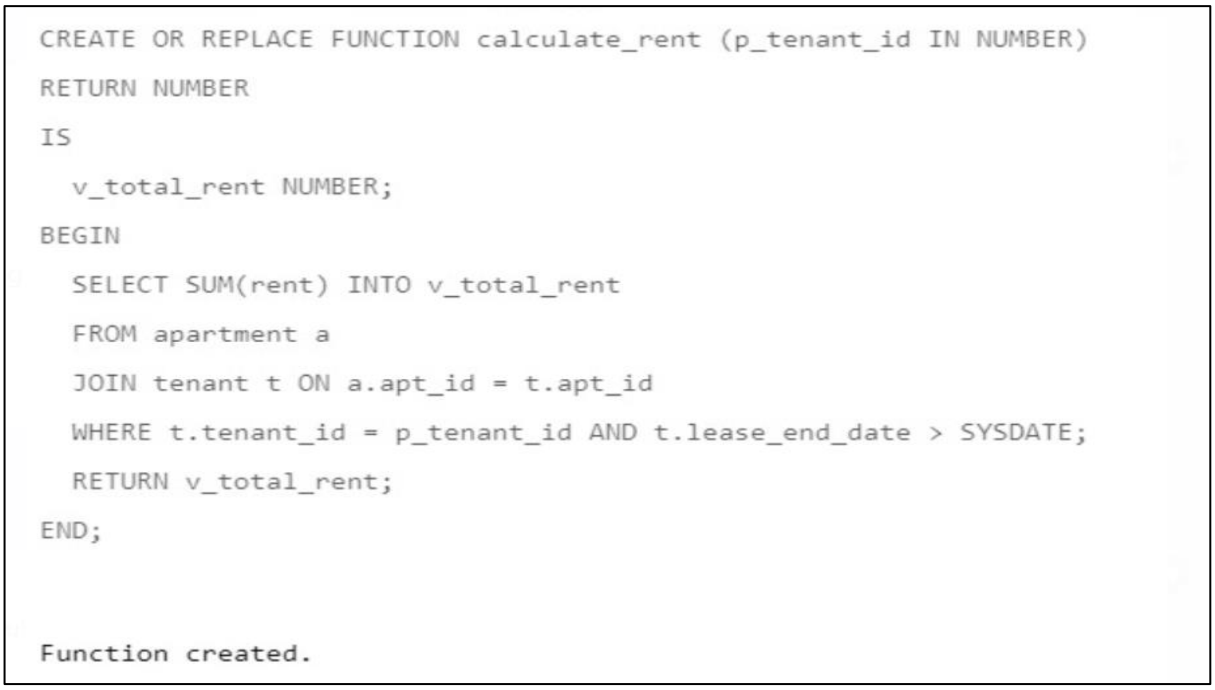
ADDRESS	BUILDING_NAME
A101	Central Towers
A102	The Palms
A103	City View Apartments
A103	Park Place
A105	Riverfront Residences



```
SELECT Tenant.Name, SUM(Payment.Amount) AS TotalRentPaid
FROM Tenant
JOIN Payment ON Tenant.Tenant_ID = Payment.Tenant_ID
GROUP BY Tenant.Name
```

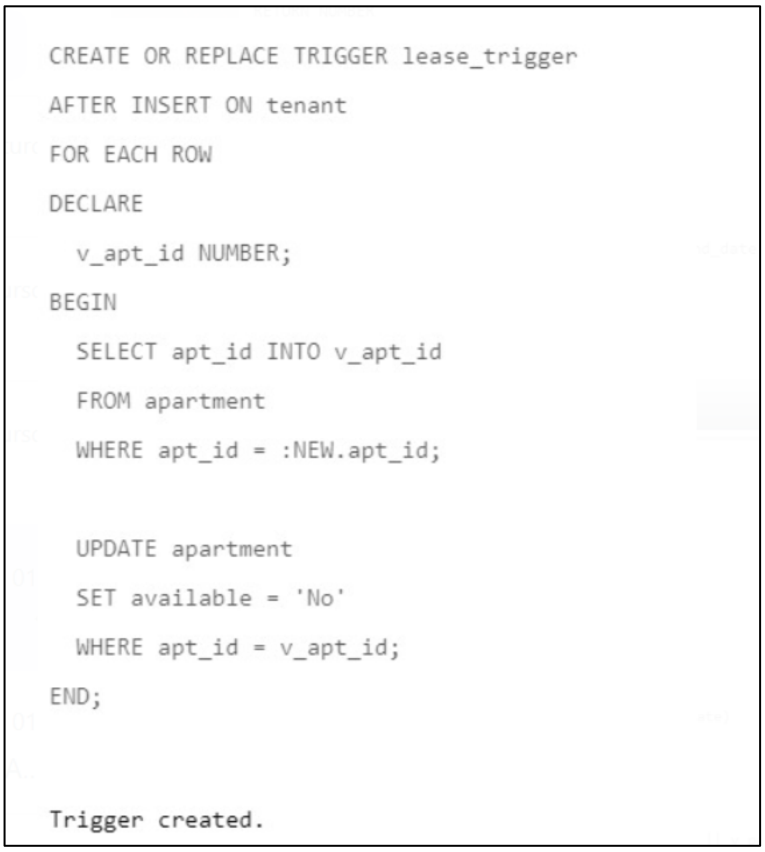
NAME	TOTALRENTPAID
Sam	800
anav	700
Manav	500
manav	900
arnav	800

Function to calculate the total rent owed by a tenant:



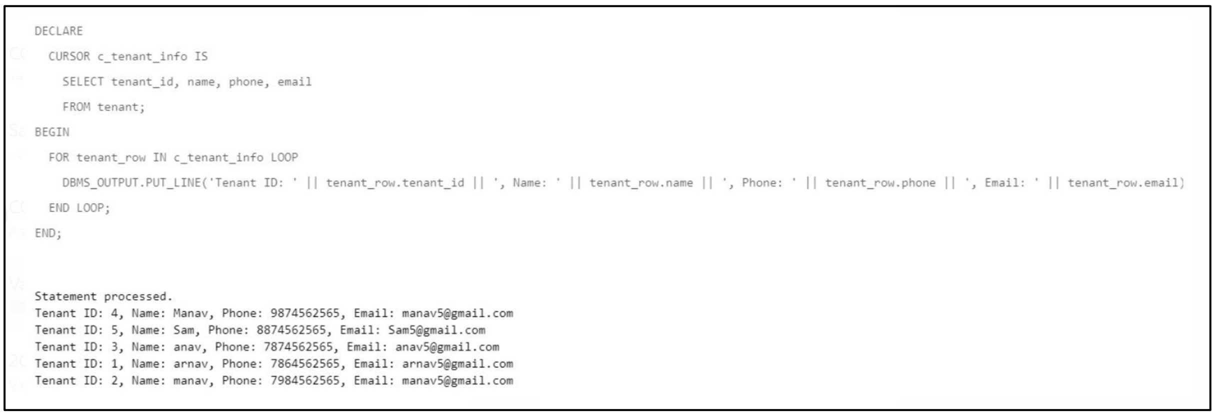
```
CREATE OR REPLACE FUNCTION calculate_rent (p_tenant_id IN NUMBER)
RETURN NUMBER
IS
  v_total_rent NUMBER;
BEGIN
  SELECT SUM(rent) INTO v_total_rent
  FROM apartment a
  JOIN tenant t ON a.apt_id = t.apt_id
  WHERE t.tenant_id = p_tenant_id AND t.lease_end_date > SYSDATE;
  RETURN v_total_rent;
END;


Function created.
```

Trigger to update the availability of an apartment when a new lease is created:



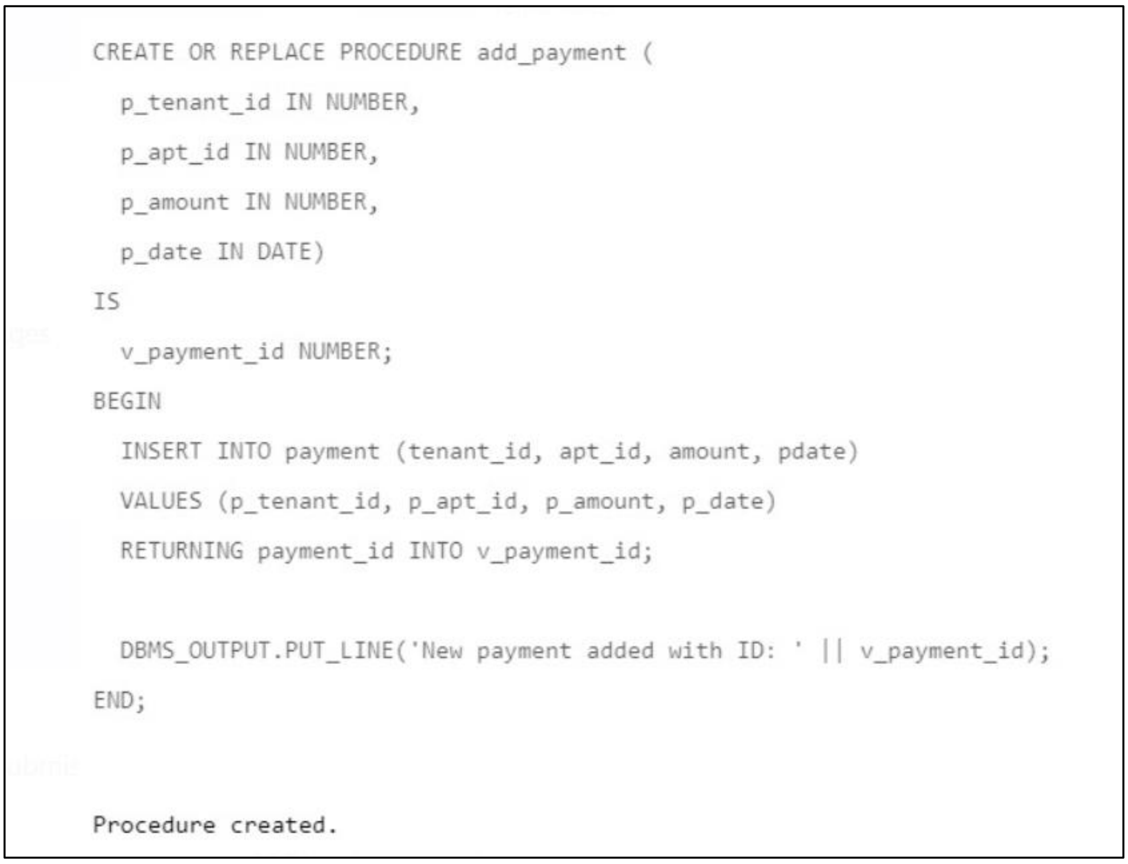
```
CREATE OR REPLACE TRIGGER lease_trigger
AFTER INSERT ON tenant
FOR EACH ROW
DECLARE
  v_apt_id NUMBER;
BEGIN
  SELECT apt_id INTO v_apt_id
  FROM apartment
  WHERE apt_id = :NEW.apt_id;

  UPDATE apartment
  SET available = 'No'
  WHERE apt_id = v_apt_id;
END;


Trigger created.
```

Cursor to retrieve a list of tenants and their contact information:



```
DECLARE
  CURSOR c_tenant_info IS
    SELECT tenant_id, name, phone, email
    FROM tenant;
BEGIN
  FOR tenant_row IN c_tenant_info LOOP
    DBMS_OUTPUT.PUT_LINE('Tenant ID: ' || tenant_row.tenant_id || ', Name: ' || tenant_row.name || ', Phone: ' || tenant_row.phone || ', Email: ' || tenant_row.email)
  END LOOP;
END;


Statement processed.
Tenant ID: 4, Name: Manav, Phone: 9874562565, Email: manav5@gmail.com
Tenant ID: 5, Name: Sam, Phone: 8874562565, Email: Sam5@gmail.com
Tenant ID: 3, Name: anav, Phone: 7874562565, Email: anav5@gmail.com
Tenant ID: 1, Name: arnav, Phone: 7864562565, Email: arnav5@gmail.com
Tenant ID: 2, Name: manav, Phone: 7984562565, Email: manav5@gmail.com
```

Procedure to add a new payment to the system:



```
CREATE OR REPLACE PROCEDURE add_payment (
  p_tenant_id IN NUMBER,
  p_apt_id IN NUMBER,
  p_amount IN NUMBER,
  p_date IN DATE)
IS
  v_payment_id NUMBER;
BEGIN
  INSERT INTO payment (tenant_id, apt_id, amount, pdate)
  VALUES (p_tenant_id, p_apt_id, p_amount, p_date)
  RETURNING payment_id INTO v_payment_id;

  DBMS_OUTPUT.PUT_LINE('New payment added with ID: ' || v_payment_id);
END;


Procedure created.
```

Conclusion: We have designed an apartment management system. Here we have created four tables based on ER diagrams : Buildings, Apartment, Tenants and Payment. We have inserted some sample values in all the tables. Further, we have used functions like joins, alter , update , where etc and also implemented pl/sql commands such as trigger, cursor and procedure. We have run some queries to use database management system. We have also normalised the tables to 3rd normal form.