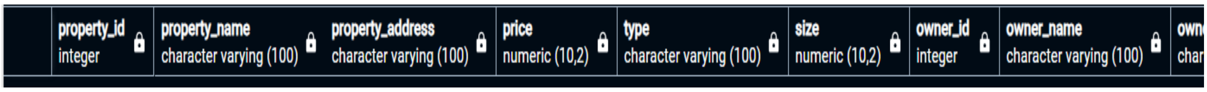
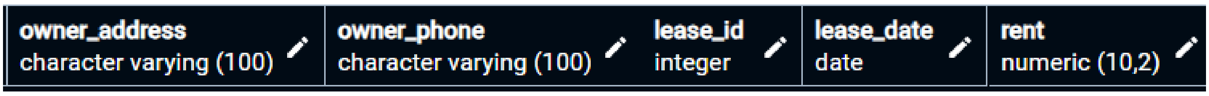
Guidelines: DBMS Project Review II

1. Construct a universal table related to your assigned project. The table should consist of at least 12 attributes.



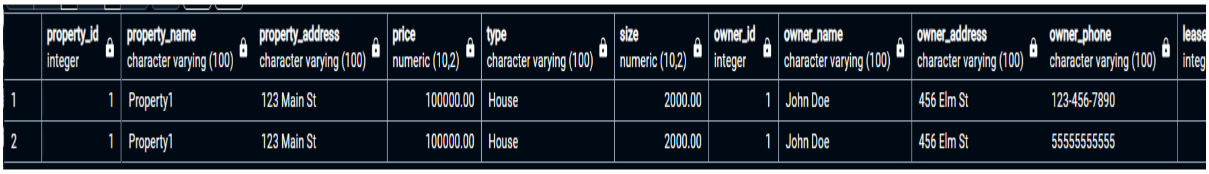
	property_id integer	property_name character varying (100)	property_address character varying (100)	price numeric (10,2)	type character varying (100)	size numeric (10,2)	owner_id integer	owner_name character varying (100)	own char



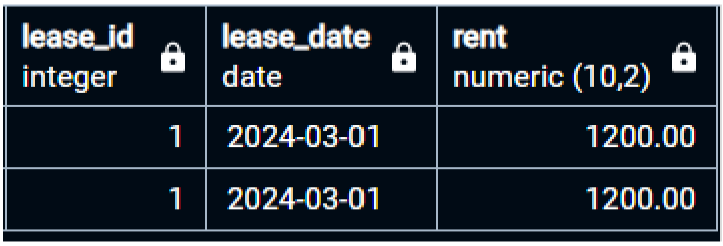
owner_address character varying (100)	owner_phone character varying (100)	lease_id integer	lease_date date	rent numeric (10,2)

2. Perform normalization process to ensure that these relations satisfy 1 NF, 2NF and 3 NF.

1NF: achieving the 1NF by having all the atomic attributes (Phone number is divided into two rows containing two phone numbers of the owner)



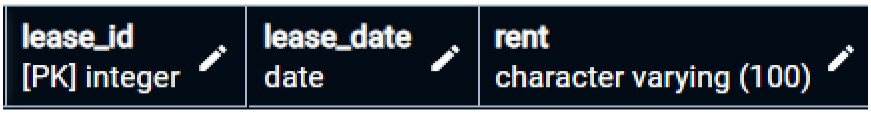
	property_id integer	property_name character varying (100)	property_address character varying (100)	price numeric (10,2)	type character varying (100)	size numeric (10,2)	owner_id integer	owner_name character varying (100)	owner_address character varying (100)	owner_phone character varying (100)	lease integ
1	1	Property1	123 Main St	100000.00	House	2000.00	1	John Doe	456 Elm St	123-456-7890	
2	1	Property1	123 Main St	100000.00	House	2000.00	1	John Doe	456 Elm St	55555555555	



lease_id integer	lease_date date	rent numeric (10,2)
1	2024-03-01	1200.00
1	2024-03-01	1200.00

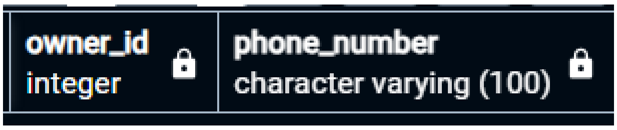
2NF: Property,Lease

Lease Table: (Identified as partial dependency)



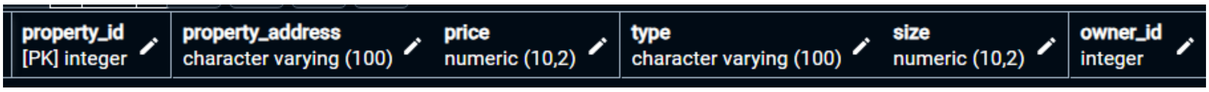
lease_id [PK] integer	lease_date date	rent character varying (100)

owner_phone(owner_id as a foreign key) from 1NF



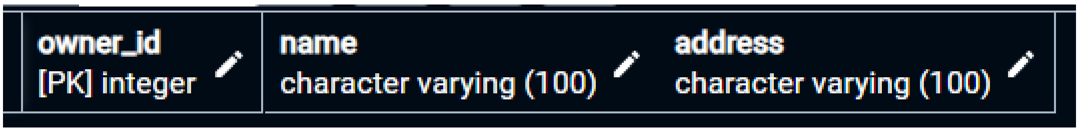
owner_id integer	phone_number character varying (100)

3NF: Identified Transitive Dependency



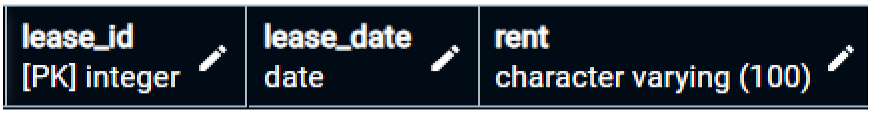
property_id	property_address	price	type	size	owner_id
[PK] integer	character varying (100)	numeric (10,2)	character varying (100)	numeric (10,2)	integer

Property Table



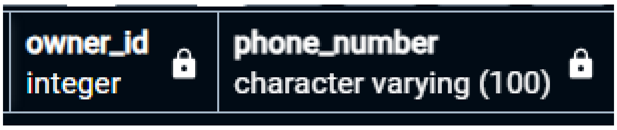
owner_id	name	address
[PK] integer	character varying (100)	character varying (100)

Owner Table



lease_id	lease_date	rent
[PK] integer	date	character varying (100)

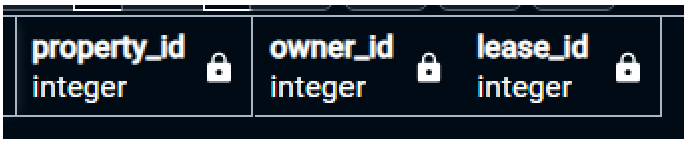
Lease Table



owner_id	phone_number
integer	character varying (100)

Owner_phone Table

A Table which helps to get data associated with property,owner and lease by using foreign key which references to property,owner,lease tables



property_id	owner_id	lease_id
integer	integer	integer

3. Based on this set of normalized relations obtained, create at least 4 tables by writing proper DDL statements (WITH CONSTRAINTS SET WHEREVER NECESSARY).

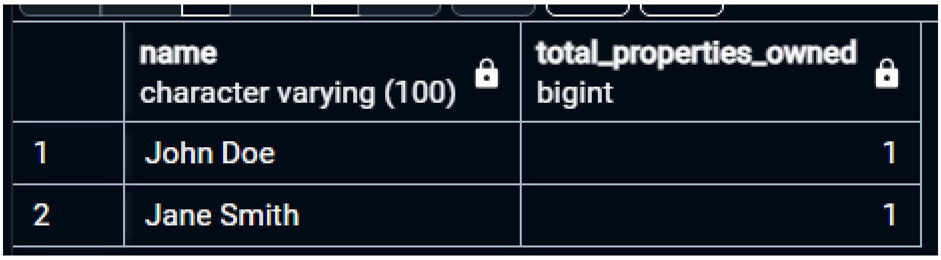
```
CREATE TABLE owner_phone (
    Owner_id INT,
    Phone_number VARCHAR(100),
Foreign key (Owner_id) REFERENCES Owner(Owner_id)
);
create table Lease (lease_id int primary key,lease_date date,Rent varchar(100));

CREATE TABLE Property (
    Property_id INT PRIMARY KEY,
    Property_address VARCHAR(100),
    Price NUMERIC(10,2),
    Type VARCHAR(100),
    Size NUMERIC(10,2),
    Owner_id INT,
    FOREIGN KEY (Owner_id) REFERENCES Owner(Owner_id)
);

CREATE TABLE Owner (
    Owner_id INT PRIMARY KEY,
    Name VARCHAR(100),
    Address VARCHAR(100)
);

CREATE TABLE sample (
    Property_id INT,
    Owner_id INT,
    Lease_id INT,
    FOREIGN KEY (Property_id) REFERENCES Property(Property_id)
);
```

4. Insert appropriate data into the tables and generate 10 queries.

The queries must be based on the following:

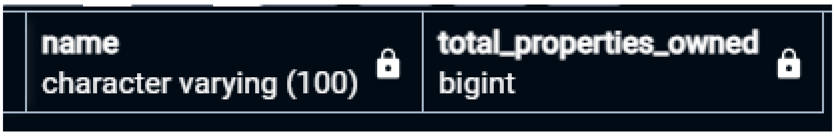
i. Aggregate functions, Group by…having :

```
SELECT o.Name, COUNT(p.Property_id) AS
Total_properties_owned
FROM Owner o
LEFT JOIN Property p ON o.Owner_id =
p.Owner_id
GROUP BY o.Name;
```



	name character varying (100)	total_properties_owned bigint
1	John Doe	1
2	Jane Smith	1

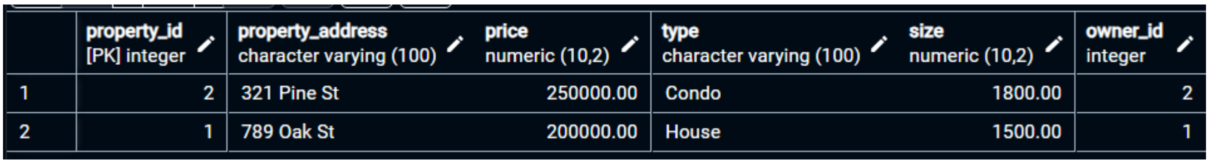
```
SELECT o.Name, COUNT(p.Property_id) AS
Total_properties_owned
FROM Owner o
LEFT JOIN Property p ON o.Owner_id =
p.Owner_id
GROUP BY o.Name
HAVING COUNT(p.Property_id) > 1;
```



name character varying (100)	total_properties_owned bigint

ii. Order by

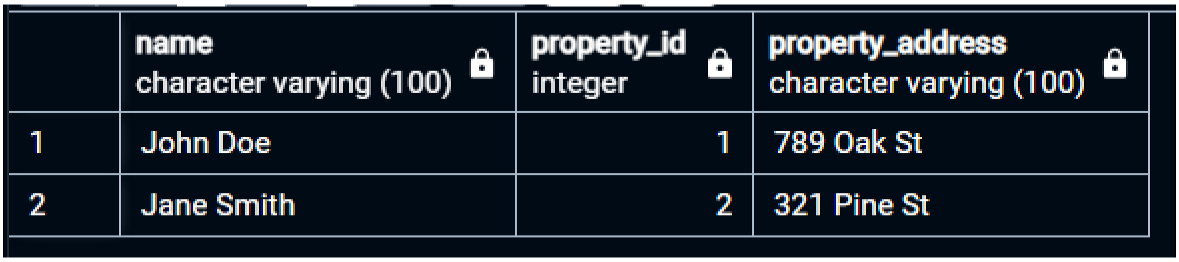
```
SELECT * FROM Property
ORDER BY Price DESC;
```



	property_id [PK] integer	property_address character varying (100)	price numeric (10,2)	type character varying (100)	size numeric (10,2)	owner_id integer
1	2	321 Pine St	250000.00	Condo	1800.00	2
2	1	789 Oak St	200000.00	House	1500.00	1

iii. Join, Outer Join

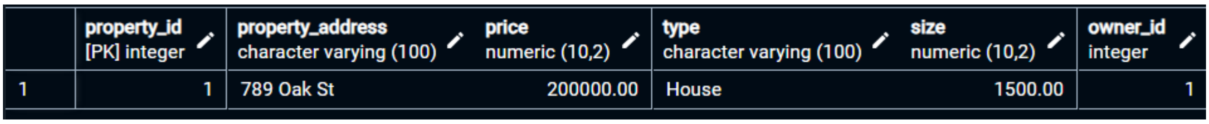
SELECT o.Name, p.Property_id,
p.Property_address
FROM Owner o
LEFT JOIN Property p ON o.Owner_id =
p.Owner_id;



	name character varying (100)	property_id integer	property_address character varying (100)
1	John Doe	1	789 Oak St
2	Jane Smith	2	321 Pine St

iv. Query having Boolean operators

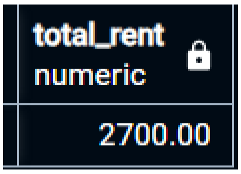
SELECT * FROM Property
WHERE Owner_id = (SELECT Owner_id FROM Owner
WHERE Name = 'John Doe')
AND Price > 20000.00;



	property_id [PK] integer	property_address character varying (100)	price numeric (10,2)	type character varying (100)	size numeric (10,2)	owner_id integer
1	1	789 Oak St	200000.00	House	1500.00	1

v. Query having arithmetic operators

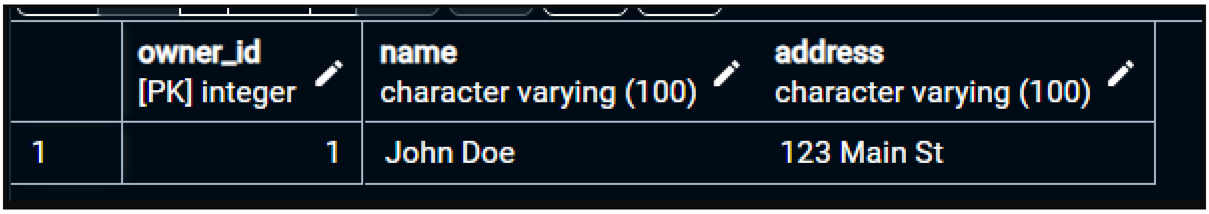
SELECT SUM(Rent) AS Total_rent FROM Lease;



total_rent numeric
2700.00

vi. A search query using string operators

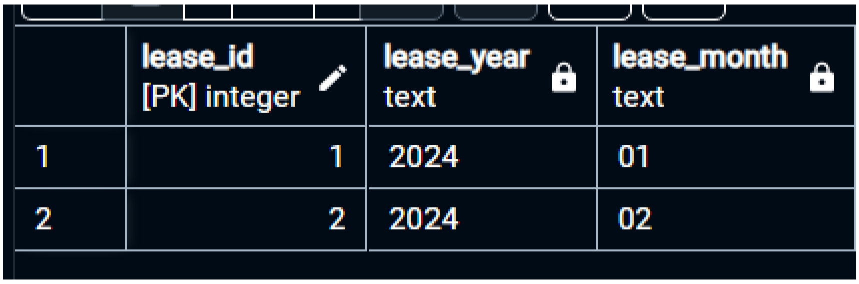
SELECT * FROM Owner
WHERE Address LIKE '%Main%';



	owner_id [PK] integer	name character varying (100)	address character varying (100)
1	1	John Doe	123 Main St

vii. Usage of to_char, extract

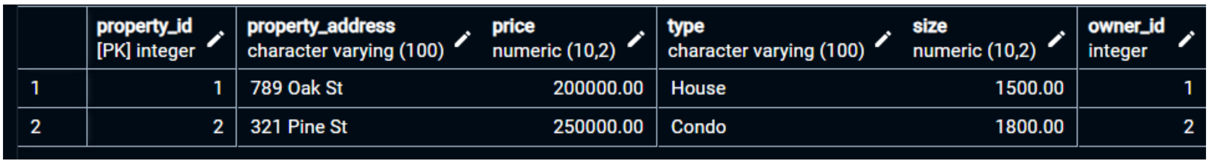
SELECT lease_id, TO_CHAR(lease_date, 'YYYY') AS lease_year, TO_CHAR(lease_date, 'MM') AS lease_month
FROM Lease;



	lease_id [PK] integer	lease_year text	lease_month text
1	1	2024	01
2	2	2024	02

viii. Between, IN, Not between, Not IN

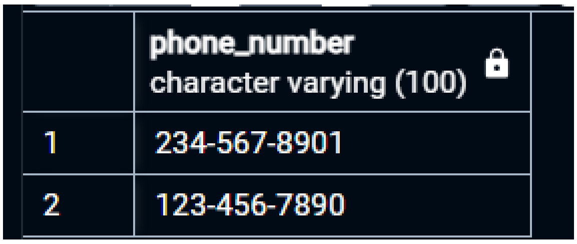
SELECT * FROM Property
WHERE Owner_id IN (1, 2);



	property_id [PK] integer	property_address character varying (100)	price numeric (10,2)	type character varying (100)	size numeric (10,2)	owner_id integer
1	1	789 Oak St	200000.00	House	1500.00	1
2	2	321 Pine St	250000.00	Condo	1800.00	2

ix. Set operations

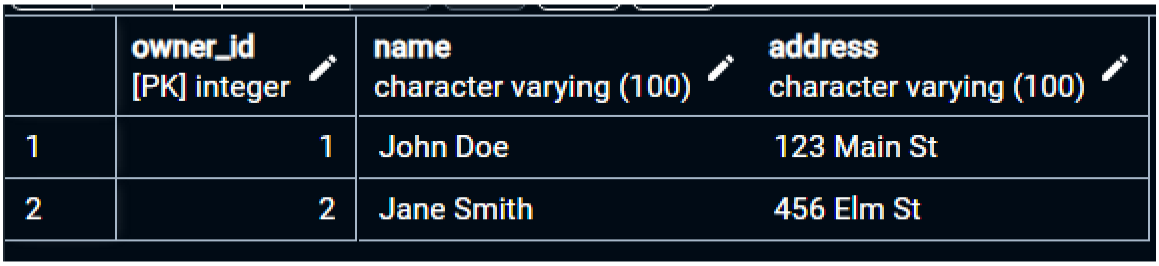
SELECT DISTINCT Phone_number FROM owner_phone;



	phone_number character varying (100)
1	234-567-8901
2	123-456-7890

x. Subquery using EXISTS / NOT EXISTS, ANY, ALL

```
SELECT * FROM Owner o
WHERE EXISTS (
  SELECT 1 FROM Property p
  WHERE p.Owner_id = o.Owner_id
);
```

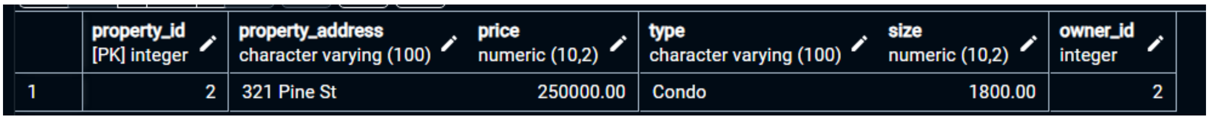


	owner_id [PK] integer	name character varying (100)	address character varying (100)
1	1	John Doe	123 Main St
2	2	Jane Smith	456 Elm St

```
SELECT * FROM Property
WHERE Price > ALL (
  SELECT Price FROM Property
  WHERE Owner_id = (SELECT Owner_id FROM Owner WHERE Name = 'Jane
Smith')
);
```

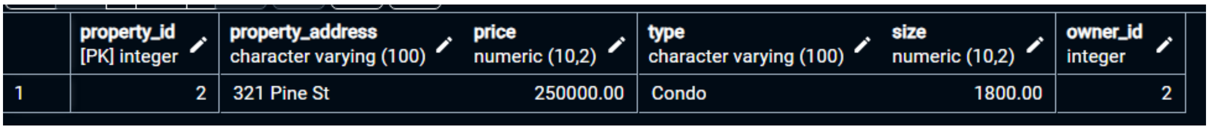
SELECT * FROM Property

```
WHERE Price > ALL (
   SELECT Price FROM Property
   WHERE Owner_id = (SELECT Owner_id FROM Owner WHERE Name = 'Jane
Smith')
);
```



	property_id [PK] integer	property_address character varying (100)	price numeric (10,2)	type character varying (100)	size numeric (10,2)	owner_id integer
1	2	321 Pine St	250000.00	Condo	1800.00	2

```
SELECT * FROM Owner o
WHERE NOT EXISTS (
   SELECT 1 FROM Property p
   WHERE p.Owner_id = o.Owner_id
);
```



	property_id [PK] integer	property_address character varying (100)	price numeric (10,2)	type character varying (100)	size numeric (10,2)	owner_id integer
1	2	321 Pine St	250000.00	Condo	1800.00	2

Execute the queries and paste the screenshots with results.

Additional Information

a. Assumptions/Constraints

b. Tables in 1NF: Form tables that are in 2NF and give proper justification for the same.

This table has thirteen attributes: Property_id, Property_name, Property_Address, Price, Type, Size, Owner_id, owner_name, Owner_Address, owner_phone, Lease_id, Lease_Date, and rent.

All attributes seem to be atomic, meaning they contain single values.

There are no repeating groups or multiple values within a single cell.

Therefore, the master table appears to satisfy the requirements for the first normal form (1NF).

c. Tables in 2NF: Form tables that are in 2NF and give proper justification for the same.

master2

This table has nine attributes: Property_id, Property_name, Property_Address, Price, Type, Size, Owner_id, owner_name, and Owner_Address.

Property_id is the primary key.

All attributes are atomic, satisfying the 1NF requirement.

Non-prime attributes (Property_name, Property_Address, Price, Type, Size, Owner_id, owner_name, and Owner_Address) are all functionally dependent on the entire primary key (Property_id).

Therefore, the master2 table is in 2NF.

owner_phone

This table has two attributes: Owner_id and Phone_number.

Owner_id is not a composite key.

All attributes are atomic, satisfying the 1NF requirement.

There are no partial dependencies as both attributes are directly related to the primary key.

Therefore, the owner_phone table is in 2NF.

Lease

This table has three attributes: lease_id, lease_date, and Rent.

lease_id is the primary key.

All attributes are atomic, satisfying the 1NF requirement.

Non-prime attributes (lease_date and Rent) are fully functionally dependent on the entire primary key (lease_id).

Therefore, the Lease table is in 2NF.

d. Tables in 3NF: Form tables that are in 3NF and give proper justification for the same.

owner_phone

This table has two attributes: Owner_id and Phone_number.

Both attributes are atomic.

There are no transitive dependencies as both attributes are directly related to the Owner_id, which is the primary key.

Therefore, the owner_phone table is in 3NF.

Lease

This table has three attributes: lease_id, lease_date, and Rent.

All attributes are atomic.

There are no transitive dependencies as each attribute is directly related to the lease_id, which is the primary key.

Therefore, the Lease table is in 3NF.

Property

This table has six attributes: Property_id, Property_address, Price, Type, Size, and Owner_id.

All attributes are atomic.

There is a transitive dependency between Owner_id and Name and Address. However, since Name and Address are dependent on Owner_id, which is the primary key, this table satisfies 3NF.

Therefore, the Property table is in 3NF.

Owner

This table has three attributes: Owner_id, Name, and Address.

All attributes are atomic.

There are no transitive dependencies as each attribute is directly related to the Owner_id, which is the primary key.

Therefore, the Owner table is in 3NF.

PROPERTY_OWNER_TENANT:

This table has three attributes: Property_id, Owner_id, and Lease_id.

All attributes are atomic.

There are no transitive dependencies as each attribute is directly related to the Property_id, Owner_id, and Lease_id, which are the primary keys.

Therefore, the sample table is in 3NF.