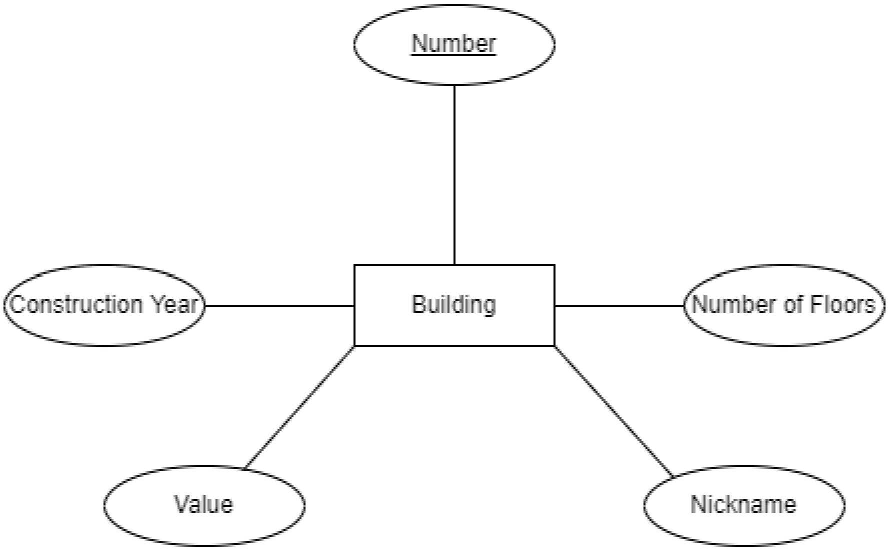
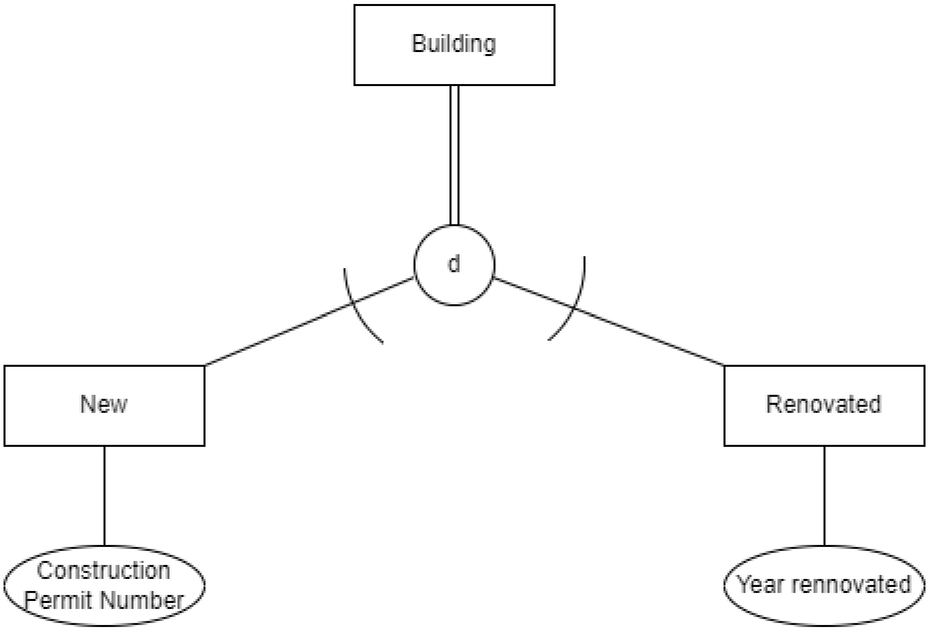
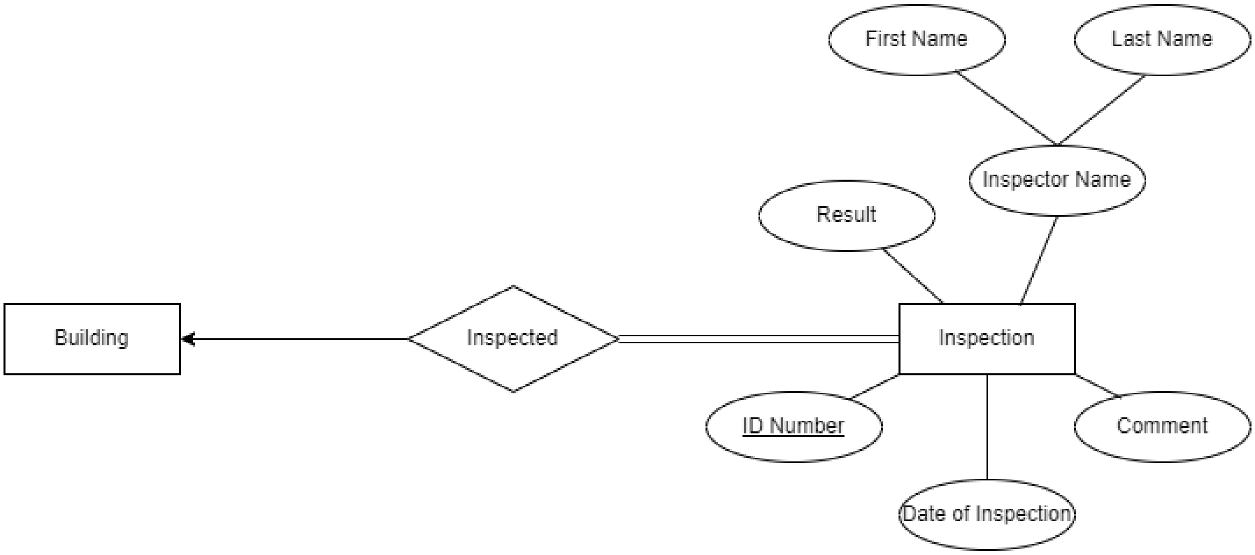
Apartments Database Project

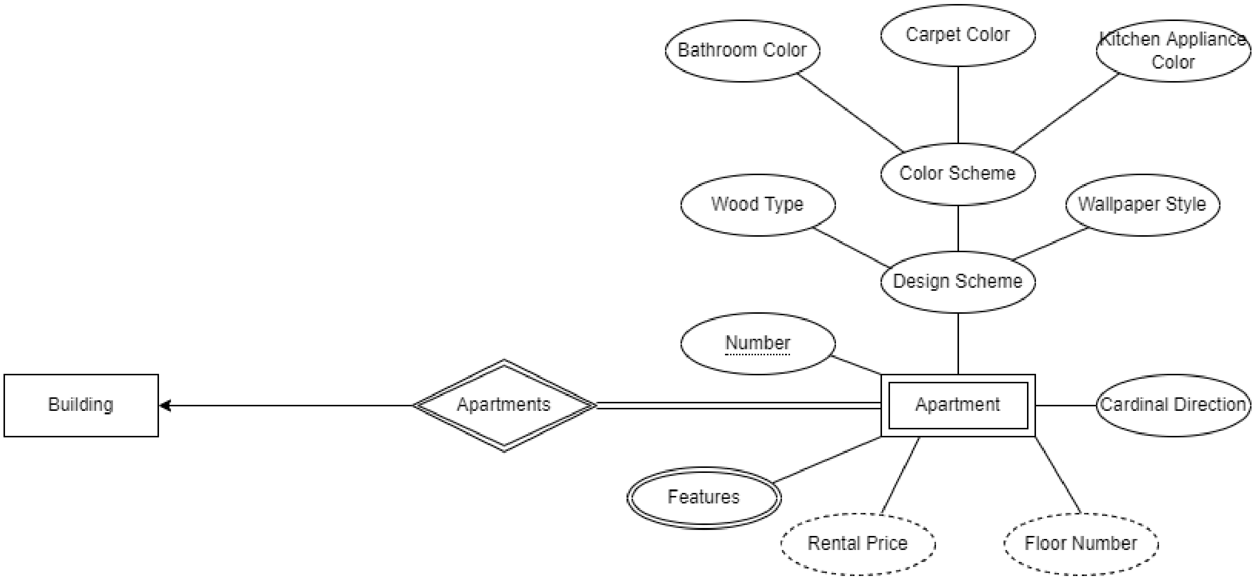
Josue Jovel

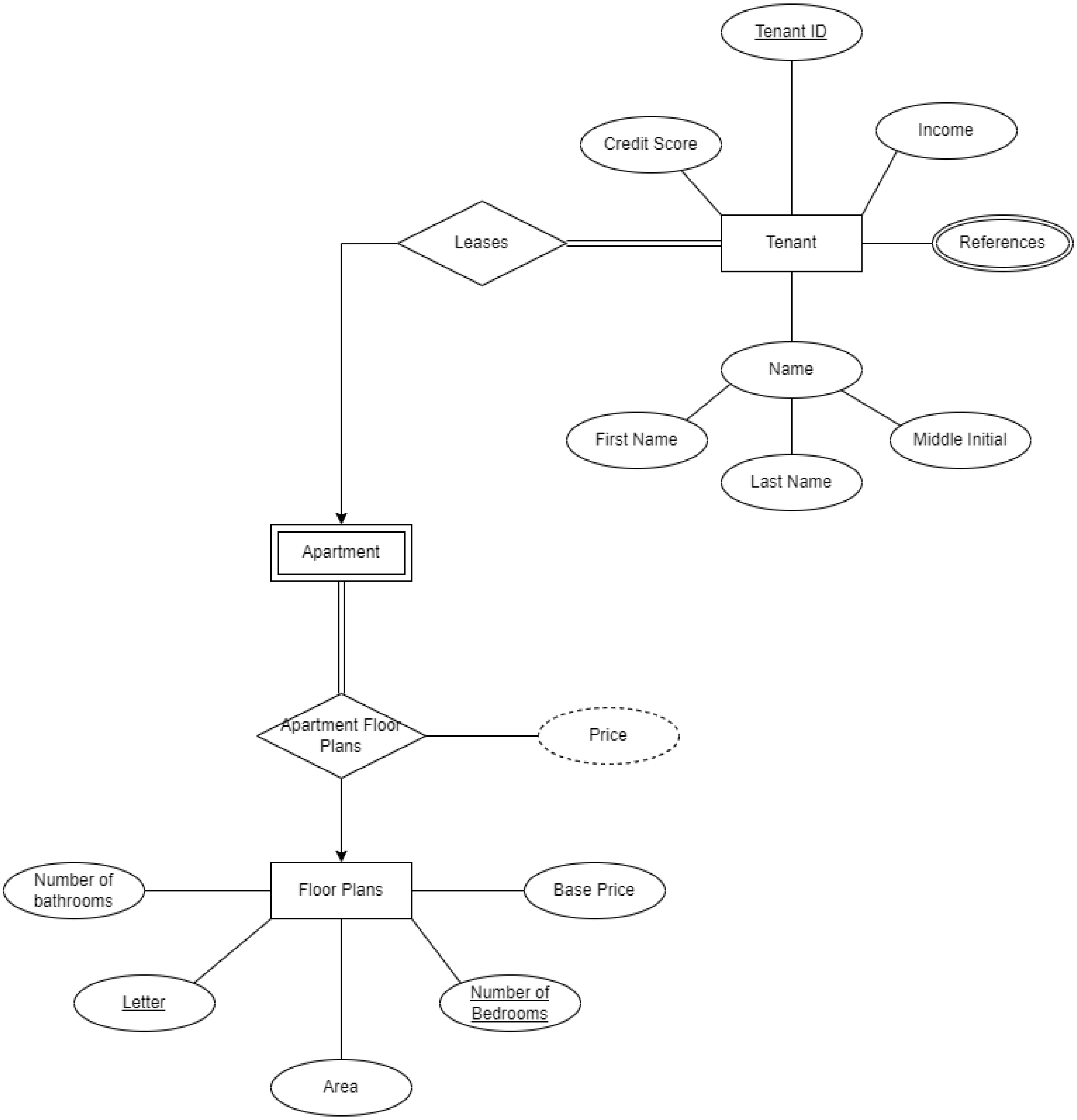
ER Diagrams


Number
Construction Year
Building
Number of Floors
Value
Nickname


Building
d
New
Renovated
Construction Permit Number
Year rennovated


First Name
Last Name
Inspector Name
Result
Building
Inspected
Inspection
ID Number
Comment
Date of Inspection


Bathroom Color
Carpet Color
Kitchen Appliance Color
Color Scheme
Wood Type
Wallpaper Style
Design Scheme
Number
Building
Apartments
Apartment
Cardinal Direction
Features
Rental Price
Floor Number


Tenant ID
Credit Score
Income
Leases
Tenant
References
Name
First Name
Middle Initial
Last Name
Apartment
Apartment Floor Plans
Price
Number of bathrooms
Floor Plans
Base Price
Letter
Number of Bedrooms
Area

Data Dictionary

ENTITIES:

Table 1: Building

Building (Building ID, Floors, Nickname, Value, Construction Year)

Name	Data Type	Description	Constraints
Building ID	Int	Identifier for building	Primary Key
Floors	Int	Number of floors	>0
Nickname	VarChar2(20)	Informal name of building	Not Null
Value	Double	Monetary value of building	>=0
Construction Year	Int	Year of building's construction	>0

Table 2: New Buildings

Building (Building ID, Construction Permit)

Name	Data Type	Description	Constraints
Building ID	Int	Identifier for building	Foreign Key, Primary Key
Construction Permit	Int	ID of Construction Permit	Not Null

Table 3: Renovated Buildings

Building (Building ID, Year Renovated)

Name	Data Type	Description	Constraints
Building ID	Int	Identifier for building	Foreign Key, Primary Key
Year Renovated	Int	Year that building was renovated	>0

Table 4: Inspections

Inspections (Inspection ID, Building ID, Comment, Result, First name, Last name, Date Inspected)

Name	Data Type	Description	Constraints
Inspection ID	Int	Identifier for inspection	Primary Key
Comment	VarChar2(99999)	Additional comments left by inspector	Default: 'None'
Result	Char(1)	Result of inspection	Must be 'P' OR 'F'
Inspector First name	VarChar2(50)	Inspector's first name	Not Null
Inspector Last name	VarChar2(50)	Inspector's last name	Not Null
Date Inspected	Date	Date of inspection	Not Null
Building ID	Int	Identifier for inspected building	Foreign Key

Table 5: Apartments

Apartments (Apartment Number, Building ID, Plan_Letter, Bedrooms, Direction, Wood Type, Wallpaper Style, Bathroom Color, Carpet Color, Kitchen Color)

Name	Data Type	Description	Constraints
Apartment Number	Number(3, 0)	3 digit Apartment Number	Composite Primary Key
Building ID	Int	Identifier for building	Foreign key, Composite Primary Key
Direction	Char(1)	Cardinal Direction of building	Must be IN ('N', 'S', 'E', 'W')
Wood Type	VarChar2(20)	Apartment's wood type	Not Null
Wallpaper style	VarChar2(20)	Apartment's wallpaper style	Not Null
Bathroom Color	VarChar2(20)	Apartment's bathroom color	Not Null
Carpet Color	VarChar2(20)	Apartment's carpet color	Not Null
Kitchen Color	VarChar2(20)	Apartment's kitchen appliance color	Not Null
Plan_Letter	Char(1)	Used in part to identify floor plan	Foreign Key
Bedrooms	Int	Number of bedrooms(used for referencing floor plan)	Foreign Key

Table 6: Features

Features (Apartment Number, Building ID, Feature)

Name	Data Type	Description	Constraints
Apartment Number	Number(3, 0)	3 digit Apartment Number	Foreign Key, Composite Primary Key
Building ID	Int	Identifier for building	Foreign key, Composite Primary Key
Feature	VarChar2(50)	Unique features of an apartment	NOT NULL

Table 7: Tenants

Tenants (Tenant ID, Apartment Number, Building ID, Credit Score, Income, First Name, Last Name, Middle Initial)

Name	Data Type	Description	Constraints
Tenant ID	Int	Identifier for tenant	Primary Key
Income	Double	Income of Tenant	>0
Credit Score	Int	Tentant's credit score	>0
Tenant First name	VarChar2(50)	Tenant's first name	Not Null
Tenant Last name	VarChar2(50)	Tenant's last name	Not Null
Tenant Middle Initial	Char(1)	Tenant's middle initial	Default: None
Apartment Number	Int	Number of tenant's apartment	Foreign Key
Building ID	Int	ID of tenant's building	Foreign key

Table 8: Tenant_References

Tenant_References (Tenant ID, Reference Name, Phone Number, Email)

Name	Data Type	Description	Constraints
Tenant ID	Int	Identifier for tenant	Composite Primary Key, Foreign Key
Reference Name	VarChar2(50)	Name of reference	NOT NULL
Phone Number	Int	Phone Number of reference, also used as a unique identifier	Composite Primary Key
Email	VarChar2(50)	Email of reference	NOT NULL

Table 9: Floor Plan

Floor Plan (Plan_Letter, Bedrooms, Bathrooms, Base Price, Area)

Name	Data Type	Description	Constraints
Plan_Letter	Char(1)	Used in part to identify floor plan	Composite Primary Key
Bedrooms	Int	Number of bedrooms	Composite Primary Key, >0
Bathrooms	Int	Number of bathrooms	>0
Base Price	Double	Unmodified price of floor plan	>0
Area	Int	Amount of square feet of the floor plan	>0

SQL Queries

1. Create all tables

```
CREATE TABLE Buildings (
        Building_ID INTEGER NOT NULL,
        Floors INTEGER CHECK (Floors > 0),
        Nickname VARCHAR2(20) NOT NULL,
        Value NUMBER (15, 2) CHECK (Value >= 0),
        Construction_Year INTEGER CHECK (Construction_Year > 0),
        PRIMARY KEY (Building_ID)
);
CREATE TABLE New_Buildings (
        Building_ID INTEGER NOT NULL,
        Construction_Permit INTEGER NOT NULL,
        FOREIGN KEY (Building_ID) REFERENCES Buildings(Building_ID),
        PRIMARY KEY (Building_ID, Construction_Permit)
);
CREATE TABLE Renovated_Buildings (
        Building_ID INTEGER NOT NULL,
        Renovation_Year INTEGER CHECK (Renovation_Year > 0),
        FOREIGN KEY (Building_ID) REFERENCES Buildings(Building_ID),
        PRIMARY KEY (Building_ID, Renovation_Year)
);
CREATE TABLE Inspections (
        Inspection_ID INTEGER NOT NULL,
        Result CHAR(1) CHECK(Result = 'P' OR Result = 'F'),
        Inspector_First_Name VARCHAR2(50) NOT NULL,
        Inspector_Last_Name VARCHAR2(50) NOT NULL,
        Date_Inspected DATE NOT NULL,
        Building_ID INTEGER NOT NULL,
        Comments LONG DEFAULT 'None',
        FOREIGN KEY (Building_ID) REFERENCES Buildings(Building_ID),
        PRIMARY KEY (Inspection_ID)
);

CREATE TABLE Floor_Plan (
        Plan_Letter CHAR(1) NOT NULL,
        Bedrooms INTEGER CHECK(Bedrooms > 0),
        Bathrooms INTEGER CHECK(Bathrooms > 0),
        Base_Price NUMBER (7, 2) CHECK(Base_Price > 0),
        Area INTEGER CHECK(Area > 0),
        PRIMARY KEY (Plan_Letter, Bedrooms)
);


CREATE TABLE Apartments (
        Apartment_Number NUMBER(3, 0) NOT NULL,
        Building_ID INTEGER NOT NULL,
        Direction CHAR(1) CHECK(Direction IN ('N', 'S', 'E', 'W')),
        Wood_Type VARCHAR2(20) NOT NULL,
        Wallpaper_Style VARCHAR2(20) NOT NULL,
        Bathroom_Color VARCHAR2(20) NOT NULL,
        Carpet_Color VARCHAR2(20) NOT NULL,
        Kitchen_Color VARCHAR2(20) NOT NULL,
        Plan_Letter CHAR(1) NOT NULL,
        Bedrooms INTEGER NOT NULL,
        FOREIGN KEY (Building_ID) REFERENCES Buildings(Building_ID),
        FOREIGN KEY (Plan_Letter, Bedrooms) REFERENCES Floor_Plan(Plan_Letter, Bedrooms),
	PRIMARY KEY (Apartment_Number, Building_ID)
);



CREATE TABLE Features (
	Apartment_Number NUMBER(3, 0) NOT NULL,
	Building_ID INTEGER NOT NULL,
	Feature VARCHAR2(50) NOT NULL,
	FOREIGN KEY (Apartment_Number, Building_ID) REFERENCES Apartments,
	PRIMARY KEY (Apartment_Number, Building_ID, Feature)
);
CREATE TABLE Tenants (
	Tenant_ID INTEGER NOT NULL,
	Income NUMBER(11, 2) CHECK(Income > 0),
	Credit_Score INTEGER CHECK(Credit_Score > 0),
	Tenant_First_Name VARCHAR2(50) NOT NULL,
	Tenant_Last_Name VARCHAR2(50) NOT NULL,
	Tenant_Middle_Initial VARCHAR(1) DEFAULT '',
	Apartment_Number INTEGER NOT NULL,
	Building_ID INTEGER NOT NULL,
	FOREIGN KEY (Apartment_Number, Building_ID) REFERENCES Apartments,
	PRIMARY KEY (Tenant_ID)
);
CREATE TABLE Tenant_References (
	Tenant_ID INTEGER NOT NULL,
	Reference_Name VARCHAR2(50) NOT NULL,
	Reference_Phone INTEGER NOT NULL,
        Reference_Email VARCHAR2(50) NOT NULL,
        FOREIGN KEY (Tenant_ID) REFERENCES Tenants(Tenant_ID),
        PRIMARY KEY (Tenant_ID, Reference_Phone)
);
```

2. Insert Values

```
INSERT INTO Buildings VALUES ('164', '3', 'Verdun', '6500211.34', '1984');
INSERT INTO Buildings VALUES ('592', '2', 'Passchendaele', '4678845.74', '1988');
INSERT INTO Buildings VALUES ('922', '5', 'Somme', '11679451.93', '1990');
INSERT INTO Buildings VALUES ('230', '4', 'Gallipoli', '9784389.44', '1979');
INSERT INTO Buildings VALUES ('628', '2', 'Argonne', '4734900.29', '1997');
INSERT INTO Buildings VALUES ('465', '4', 'Stalingrad', '9492593.50', '2008');
INSERT INTO Buildings VALUES ('160', '3', 'Monte Cassino', '6943030.44', '2010');
INSERT INTO Buildings VALUES ('371', '2', 'El Alamein', '4794993.70', '2009');
INSERT INTO Buildings VALUES ('581', '2', 'Iwo Jima', '3953434.32', '2012');
INSERT INTO Buildings VALUES ('985', '3', 'Bastogne', '6343022.00', '2011');
INSERT INTO Buildings VALUES ('784', '3', 'Saigon', '6529495.00', '2021');



INSERT INTO New_Buildings VALUES ('985', '493602');
INSERT INTO New_Buildings VALUES ('581', '949294');
INSERT INTO New_Buildings VALUES ('371', '492592');
INSERT INTO New_Buildings VALUES ('160', '572863');
INSERT INTO New_Buildings VALUES ('465', '394295');
INSERT INTO New_Buildings VALUES ('784', '454264');

INSERT INTO Renovated_Buildings VALUES ('164', '2007');
INSERT INTO Renovated_Buildings VALUES ('592', '2011');
INSERT INTO Renovated_Buildings VALUES ('922', '2014');
INSERT INTO Renovated_Buildings VALUES ('230', '2009');
INSERT INTO Renovated_Buildings VALUES ('628', '2010');

INSERT INTO Inspections VALUES ('40024', 'P', 'Wesley', 'Jacobs', '29-NOV-12', '985', 'Thermostat needs tuning, much too cold.');
INSERT INTO Inspections VALUES ('38183', 'P', 'Adam', 'Myers', '15-JAN-17', '581', 'Might want some air purifiers, humid as a jungle in there.');
INSERT INTO Inspections VALUES ('93924', 'P', 'John', 'Brown', '18-MAR-13', '371', 'AC needs to be reinstalled, every room was very hot, and it is not even summer yet.');
INSERT INTO Inspections VALUES ('29943', 'F', 'Guy', 'Manson', '06-APR-16', '922', 'Bad plumbing, dirty water, infestion problem. Smells like death in a lot of rooms.');
INSERT INTO Inspections VALUES ('51195', 'P', 'Bryan', 'Fisher', '05-SEP-15', '592', 'Might want to lay some cement on the ground in front. Gets really muddy out front.');
INSERT INTO Inspections VALUES ('58243', 'F', 'Slim', 'White', '16-MAR-12', '371', 'No Comment');
INSERT INTO Inspections VALUES ('58288', 'F', 'Slim', 'White', '16-JUN-12', '371', 'No Comment');

INSERT INTO Floor_Plan VALUES ('A', '2', '1', '850.00', '800');
INSERT INTO Floor_Plan VALUES ('A', '3', '2', '1020.00', '900');
INSERT INTO Floor_Plan VALUES ('B', '2', '1', '860.00', '820');
INSERT INTO Floor_Plan VALUES ('B', '3', '2', '1120.00', '940');
INSERT INTO Floor_Plan VALUES ('C', '1', '1', '680.00', '650');

INSERT INTO Apartments VALUES ('101', '164', 'N', 'Mahogany', 'Industrial', 'Blue', 'Wine', 'White', 'A', '2');
INSERT INTO Apartments VALUES ('204', '628', 'S', 'Walnut', 'Country', 'Teal', 'Black', 'Fuschia', 'A', '3');
INSERT INTO Apartments VALUES ('313', '160', 'E', 'Cherry', 'Bohemian', 'Navy', 'Honeydew', 'Brown', 'B', '2');
INSERT INTO Apartments VALUES ('421', '465', 'E', 'Oak', 'Oriental', 'Beige', 'Salmon', 'Aquamarine', 'B', '3');
INSERT INTO Apartments VALUES ('512', '922', 'W', 'Maple', 'Floral', 'Purple', 'Plum', 'Lavender', 'C', '1');

INSERT INTO Apartments VALUES ('166', '922', 'S', 'Birch', 'Contemporary', 'Blue', 'Navy', 'Black', 'C', '1');

INSERT INTO Apartments VALUES ('314', '160', 'E', 'Cherry', 'Bohemian', 'Navy', 'Honeydew', 'Brown', 'B', '2');

INSERT INTO Apartments VALUES ('422', '465', 'E', 'Oak', 'Oriental', 'Beige', 'Salmon', 'Aquamarine', 'B', '3');

INSERT INTO Apartments VALUES ('423', '465', 'E', 'Oak', 'Oriental', 'Beige', 'Salmon', 'Aquamarine', 'B', '3');

INSERT INTO Apartments VALUES ('423', '985', 'E', 'Oak', 'Oriental', 'Beige', 'Salmon', 'Aquamarine', 'B', '3');

INSERT INTO Apartments VALUES ('166', '784', 'W', 'Fir', 'Fortnite', 'Brown', 'Green', 'Yellow', 'C', '1');

INSERT INTO Apartments VALUES ('167', '784', 'W', 'Fir', 'Fortnite', 'Brown', 'Green', 'Yellow', 'C', '1');
```

```
INSERT INTO Features VALUES ('423', '465', 'Fireplace');

INSERT INTO Features VALUES ('422', '465', 'Sauna');

INSERT INTO Features VALUES ('314', '160', 'Patio');

INSERT INTO Features VALUES ('166', '922', 'Poolside');

INSERT INTO Features VALUES ('166', '922', 'Covered Parking Spot');

INSERT INTO Features VALUES ('314', '160', 'Sauna');

INSERT INTO Features VALUES ('204', '628', 'Balcony');

INSERT INTO Features VALUES ('101', '164', 'Balcony');

INSERT INTO Features VALUES ('314', '160', 'Balcony');

INSERT INTO Features VALUES ('422', '465', 'Balcony');

INSERT INTO Features VALUES ('423', '465', 'Balcony');

INSERT INTO Features VALUES ('423', '985', 'Balcony');

INSERT INTO Features VALUES ('101', '164', 'Sauna');
INSERT INTO Tenants VALUES ('568399', '35000', '350', 'Daisy', 'Baker', '', '101', '164');

INSERT INTO Tenants VALUES ('692853', '69000', '600', 'Scott', 'Foster', 'D', '204', '628');

INSERT INTO Tenants VALUES ('683953', '100000', '750', 'Michael', 'Henderson', 'L', '313', '160');

INSERT INTO Tenants VALUES ('582122', '20000', '320', 'Leo', 'Wood', 'W', '421', '465');

INSERT INTO Tenants VALUES ('429502', '45000', '400', 'Jude', 'Robinson', 'F', '512', '922');

INSERT INTO Tenants VALUES ('548683', '65000', '600', 'Johnson', 'Williams', 'G', '423', '985');
```

```
INSERT INTO Tenant_References VALUES ('692853', 'Max Richards', '6675821883',
'MRich123@gmail.com');

INSERT INTO Tenant_References VALUES ('683953', 'Curtis Conway', '6676883286',
'TheConway323@gmail.com');

INSERT INTO Tenant_References VALUES ('582122', 'Jeffery May', '6675843992',
'JeffM399@gmail.com');

INSERT INTO Tenant_References VALUES ('429502', 'Tyrell Pruitt', '8683961139',
'pizzaisawesome1337@gmail.com');

INSERT INTO Tenant_References VALUES ('429502', 'Josh Manson', '6673249600',
'iheartsql@gmail.com');
```

3.

```
CREATE VIEW Empty_Apartments AS

SELECT Apartments.Apartment_Number, Apartments.Building_ID FROM Apartments

LEFT JOIN Tenants ON Apartments.Apartment_Number = Tenants.Apartment_Number

AND Apartments.Building_ID = Tenants.Building_ID

WHERE Tenant_ID IS NULL;


SELECT Value, Nickname, COUNT(Apartment_Number) AS Apartment_Count FROM Buildings NATURAL
JOIN New_Buildings NATURAL JOIN Empty_Apartments

WHERE Construction_Year > (EXTRACT(year FROM CURRENT_DATE) – 3)

GROUP BY Value, Nickname

UNION

SELECT Value, Nickname, COUNT(Apartment_Number) AS Apartment_Count FROM Buildings NATURAL
JOIN Renovated_Buildings NATURAL JOIN Empty_Apartments

WHERE Renovation_Year > (EXTRACT(year FROM CURRENT_DATE) – 3)

GROUP BY Value, Nickname

ORDER BY Value DESC, Apartment_Count DESC;
```

4.

```
SELECT Construction_Permit, Floors, Construction_Year, Inspector_First_Name, Inspector_Last_Name,
Date_Inspected FROM Buildings NATURAL JOIN New_Buildings NATURAL JOIN Inspections

WHERE Result LIKE 'F'

ORDER BY Date_Inspected ASC;
```

5.

```
SELECT Nickname, Value, SUM(Area) FROM Buildings NATURAL JOIN Renovated_Buildings NATURAL
JOIN Apartments NATURAL JOIN Floor_Plan

GROUP BY Nickname, Value

ORDER BY SUM(Area) ASC;
```

6.

```
SELECT Apartment_Number, Direction, Nickname, SUBSTR(TO_CHAR(Apartment_Number), 1, 1) AS Floor, Base_Price + (100 * COUNT(FEATURE)) AS Rental_Price, Bathroom_Color, Carpet_Color, Kitchen_Color

FROM Empty_Apartments NATURAL JOIN Apartments NATURAL JOIN Features NATURAL JOIN Buildings NATURAL JOIN Floor_Plan

GROUP BY Apartment_Number, Direction, Nickname, Base_Price, Bathroom_Color, Carpet_Color, Kitchen_Color;
```

7.

```
CREATE VIEW Apartment_Stats AS

SELECT DISTINCT Apartments.Apartment_Number,  Apartments.Building_ID, COUNT(Features.Feature) AS Feature_Count, COUNT(Tenants.Tenant_ID) AS Tenant_Count FROM Features

INNER JOIN Apartments ON Features.Building_ID = Apartments.Building_ID

AND Features.Apartment_Number = Apartments.Apartment_Number

LEFT JOIN Tenants ON Apartments.Apartment_Number = Tenants.Apartment_Number

AND Apartments.Building_ID = Tenants.Building_ID

GROUP BY Apartments.Apartment_Number,  Apartments.Building_ID;
```

```
SELECT Apartments.Apartment_Number, Apartments.Wood_Type, CONCAT(Apartments.Bedrooms, Apartments.Plan_Letter) AS Floorplan, Floor_Plan.Base_Price + (100 * Apartment_Stats.Feature_Count) AS Rental_Price, Apartment_Stats.Tenant_Count FROM Apartment_Stats

INNER JOIN Apartments ON Apartment_Stats.Building_ID = Apartments.Building_ID

AND Apartment_Stats.Apartment_Number = Apartments.Apartment_Number

INNER JOIN Floor_Plan ON Apartments.Plan_Letter = Floor_Plan.Plan_Letter

AND Apartments.Bedrooms = Floor_Plan.Bedrooms

WHERE CONCAT(Apartments.Apartment_Number, Apartments.Building_ID) IN (Select CONCAT(Apartment_Number, Building_ID) FROM Features

WHERE Feature = 'Balcony')

ORDER BY Floorplan DESC, Apartment_Stats.Tenant_Count DESC;
```

8.

```
SELECT Tenant_First_Name, Tenant_Middle_Initial, Tenant_Last_Name, Credit_Score,
Apartment_Number, COUNT(Reference_Phone) AS Reference_Amount FROM Tenants NATURAL JOIN
Apartments NATURAL JOIN Tenant_References

HAVING COUNT(Reference_Phone) IN (SELECT MAX(Phone_Count)

FROM (SELECT COUNT(Reference_Phone) AS Phone_Count FROM Tenant_References NATURAL JOIN
Tenants

GROUP BY Tenant_ID))

GROUP BY Tenant_First_Name, Tenant_Middle_Initial, Tenant_Last_Name, Credit_Score,
Apartment_Number;
```

9.

```
SELECT CONCAT(Bedrooms, Plan_Letter) AS Floorplan, Bedrooms, Bathrooms, Base_Price + (100 *
COUNT(Feature)) AS Rental_Price, Area

FROM Apartments NATURAL JOIN Features NATURAL JOIN Floor_Plan NATURAL JOIN Tenants LEFT JOIN
Tenant_References ON Tenants.Tenant_ID = Tenant_References.Tenant_ID

WHERE Reference_Phone IS NULL

GROUP BY Plan_Letter, Bedrooms, Bathrooms, Base_Price, Area

ORDER BY Area DESC;
```

10.

```
CREATE VIEW Empty_Apartments AS

SELECT Apartments.Apartment_Number, Apartments.Building_ID FROM Apartments

LEFT JOIN Tenants ON Apartments.Apartment_Number = Tenants.Apartment_Number

AND Apartments.Building_ID = Tenants.Building_ID

WHERE Tenant_ID IS NULL;


UPDATE Apartments

SET Kitchen_Color = 'Fuschia', Carpet_Color = 'Lime', Bathroom_Color = 'Teal'
```

```
WHERE CONCAT(Apartment_Number, Building_ID) IN (SELECT CONCAT(Apartment_Number,
Building_ID) FROM Empty_Apartments NATURAL JOIN Renovated_Buildings NATURAL JOIN Apartments
WHERE Renovation_Year < (EXTRACT(year FROM CURRENT_DATE) – 2));
```

11.

```
DELETE FROM Features
WHERE Feature IN (
        SELECT Feature FROM Features NATURAL JOIN New_Buildings
        HAVING COUNT(Building_ID) = (
                SELECT MAX(Feature_Count)
                FROM (
                        SELECT Feature, COUNT(Building_ID) AS Feature_Count FROM Features
                        NATURAL JOIN New_Buildings
                        GROUP BY Feature
                )
        )
        GROUP BY Feature
);
```

12.

```
DROP VIEW Apartment_Stats;
DROP VIEW Empty_Apartments;
DROP TABLE Tenant_References;
DROP TABLE Tenants;
DROP TABLE Features;
DROP TABLE Apartments;
DROP TABLE Floor_Plan;
DROP TABLE Inspections;
DROP TABLE Renovated_Buildings;
DROP TABLE New_Buildings;

DROP TABLE Buildings;
```