Information Management Project
By Tanmay Kaushik
Student No. - 18308341

Estate Agency Database

I have designed and implemented a database for an Estate Agency Company called Casa Hunt. This company has branches spread over the country which helps people buy and sell their properties. The database that I have implemented keeps track of the agency's branches, employees, buyers, sellers, properties and some additional things like features, commissions, property viewings and sale records. This is done by using tables and implementing constraints and triggers on which ensures the database's integrity and smooth functioning.

The Branch table has 6 attributes - Branch_ID which is the primary key, Name, Phone_No, Street, Town and County. A constraint have been added on Phone_No. is a 10 digit number.

The Employee table has 8 attributes - Employee_ID which is the primary key, FName, SName, Emp_Position which tells if the employee is a Manager or Executive, Emp_Status which indicates if the employee is an active employee or has left the company, Salary, PPS and Branch_ID. Branch_ID is the foreign key which is used to know the branch where the employee works at. To ensure that the PPS number is right, I have added a constraint that the PPS number must be of length 9 and also must start with be a letter. Constraints on Emp_Position checks that the input is either Manager or Executive and on Emp_Status checks it is either Active or Inactive.

The Owner table has 5 attributes - Owner ID which is the primary key, FName, SName, Phone_No. and Email. Similar to Branch table a constraint has been added on

Phone_No. which ensures that the length is equal to 10 numbers. A constraint on Email also verifies that there is an @ and . symbol with text before and after @. The Contact Number has been selected as NOT NULL because the Owner of the house has be contacted.

The Buyer Table stores the information of customers who are looking to buy a property from the agency. This table has 7 attributes - Buyer_ID which is the primary key, FName, SName, Budget, Phone_No., Bed_Requirement and Bath_Requirement. Budget, Bed and Bath requirement columns are optional in case the Buyer is looking for different options.

The Features table is a table which has a list of features that a property can have. It has 3 attributes - Feature_ID (Primary Key), Feat_Name and Feat_Description. The description field is optional as sometimes feature name is enough to convey thr information. Due to a many to many relationship with property, the table is normalised using primary keys of both tables and make a separate table which I call Property_Features.

Property table has 13 attributes - a primary key Property_ID, Property_Status which indicates if the property is still on the market, Date_Updated which indicates the date when the details of the property were updated, Price, Street, Town, County, EIRCODE, Bedroom showing the number of bedrooms, Bathroom, Area and Type which tells if the property is a House, Apartment, Duplex, Bungalow, Cottage or Other and Owner_ID which is a foreign key showing the owner of the property. I have added multiple constraints on various columns of the table like - a constraint on EIRCODE should always start with a letter and it must be of length 7. I have made sure that the Property status must be either For Sale, Sale Agreed or Sold, else it must be deleted from the database.

I have made sure using triggers that when the property is sold, the buyer becomes the new owner and its Owner_ID is tagged with the house.

The Viewing table keeps a record of all the viewings that the employee has organised for prospective buyers for a particular property. The table has 6 attributes - Viewing_ID, Viewing_Time, Viewing_Date, Employee_ID, Property_ID and Buyer_ID. The last 3 attributes are foreign keys which tracks the employee who held the viewing for a particular property and for which buyer.

I have created a View which can be seen by all called Viewing_List which shows Viewing ID, Property Name, Visitor's Name, Date, Time and Employee's details who showed the property.

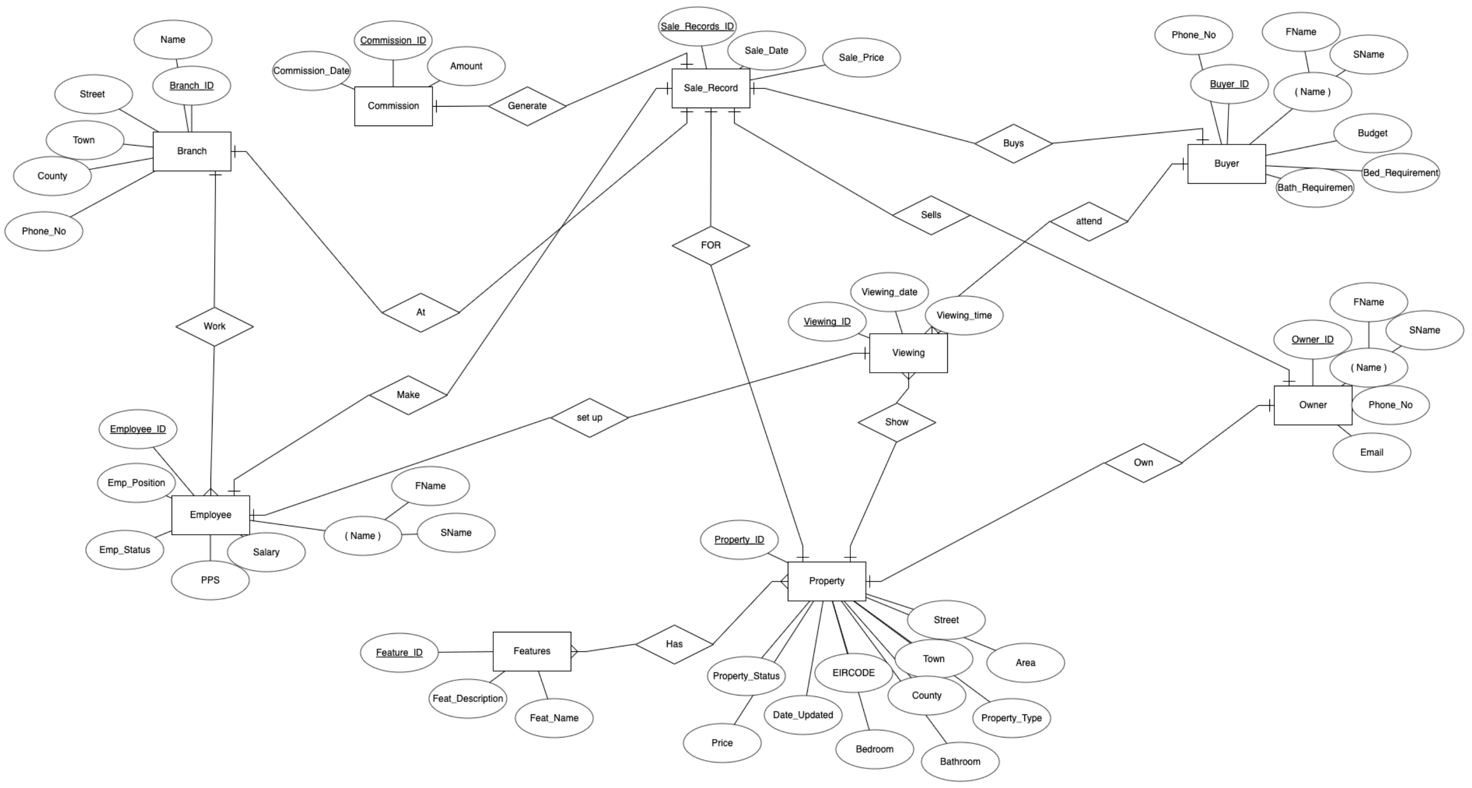
Sale_Record keeps track of all the property sales done in the agency as a central system. When the property is sold (not sale agreed), the manager enters the details of all the related parties and employee. It has 8 attributes - Sale_Record_ID which is the primary key, Sale_Date, Sale_Price, Branch_ID, Property_ID, Buyer_ID, Employee_ID and Owner_ID. The last 5 attributes are foreign keys used to store the details of Branch where the sale was made, Property information and also the information of Buyer, Seller and Employee.

I have made 2 triggers based on this table which is responsible for changing the ownership of the property automatically and also generates commission in the commission table (that too automatically).

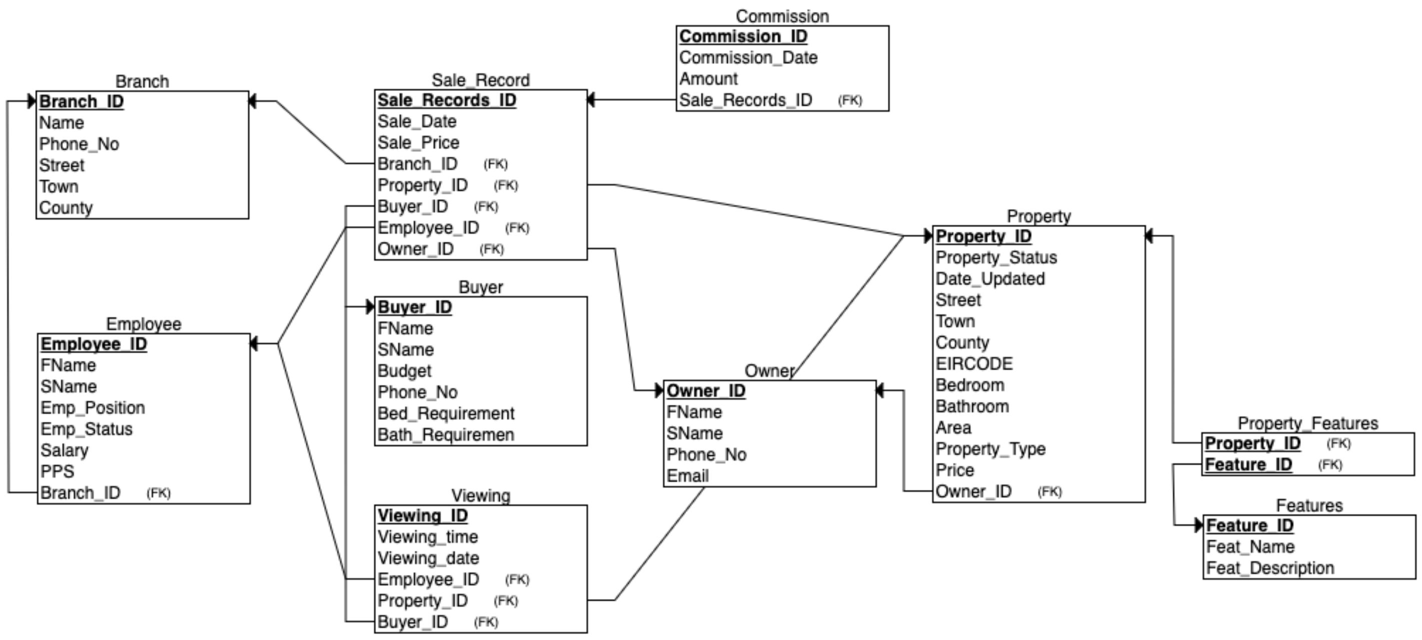
The Commission table has 4 attributes. This table is used to calculate the commission of employee when a property is sold. The attributes are - Commission_ID (Primary Kay), Date, Amount, Sales_Records_ID.

The entries in these tables are generated when the Manager adds the sale information of a property in the Sale Record.

ENTITY RELATIONSHIP DIAGRAM

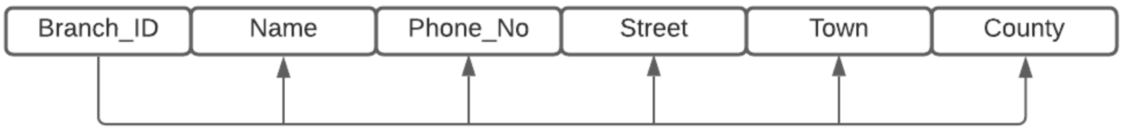


RELATIONAL SCHEMA DIAGRAM

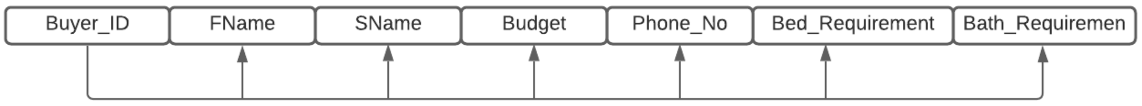


FUNCTIONAL DEPENDENCY DIAGRAM

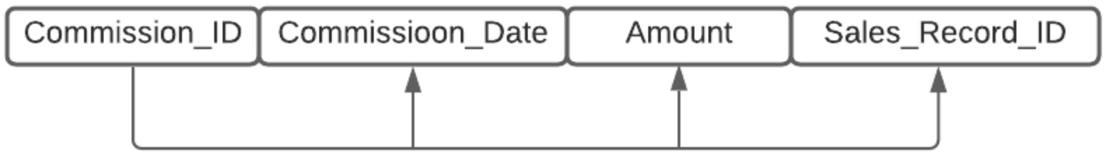
BRANCH



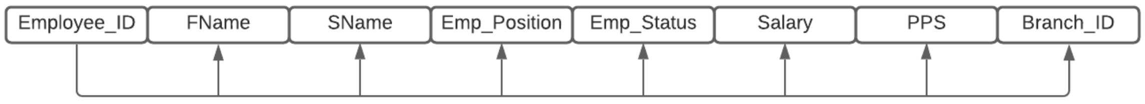
Buyer



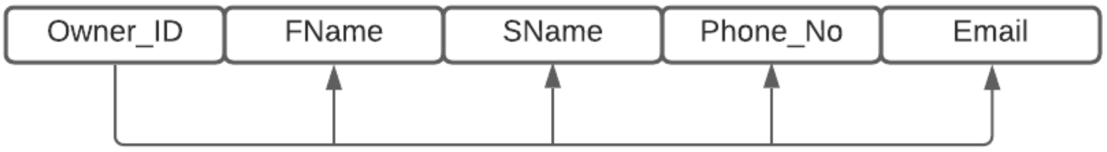
COMMISSION



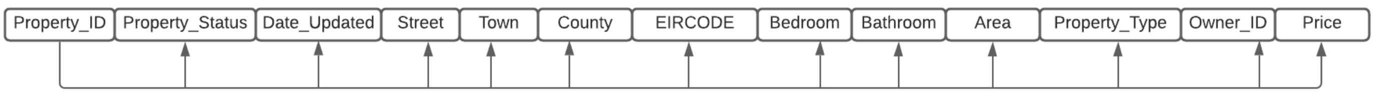
EMPLOYEE



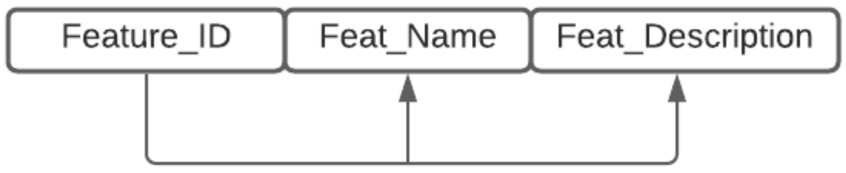
OWNER



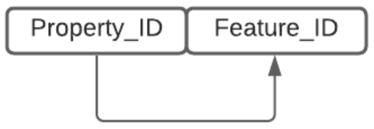
PROPERTY



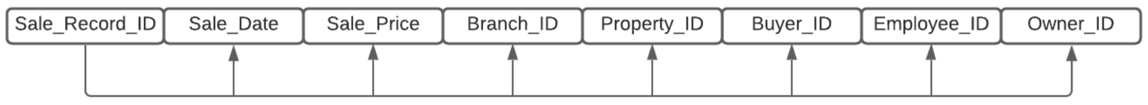
FEATURES



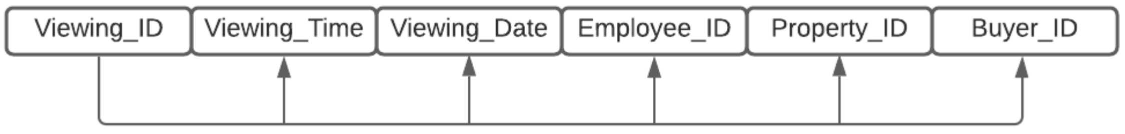
Property_Features



Sale_Record



VIEWING



SEMANTIC CONSTRAINTS

Key Constraints :

All the tables in the database have a unique Primary Key. I have not used PPS number or any sensitive information as a Primary Key in any of the table to ensure security of people's personal data.

The database uses a Composite Primary Key in only one table which is Property_Features. It uses the primary keys of Property (i.e. Property_ID) and Features (i.e. Feature_ID).

Entity Integrity Constraints :

Constraints have been added on all Primary Keys to ensure that it cannot be NULL. Along with that, Auto_Increment is used in case the count of entries is not available.

Referential Integrity Constraints :

Referential Integrity is maintained in the database with constraints to ensure that the foreign keys must not be deleted when used by another table. Most of the foreign keys in my database is the data that might never be deleted to maintain a proper record of all the owners and properties.

Additional Constraints

The database uses various constraints to ensure the length, format of some entries and sometimes the entry has to be a particular value from the options.

Some of them are -

```
CONSTRAINT check_PPS CHECK(CHARACTER_LENGTH(PPS) = 9
AND SUBSTRING(PPS FROM 1 FOR 1) BETWEEN 'A' AND 'Z')
```

This constraint ensures that the PPS number entered is of length 9 and starts with a letter. This is the correct format mentioned by the Government of Ireland website.

```
CONSTRAINT check_Email CHECK ( Email  LIKE '%_@__%.__%' )
```

This constraint ensures the following things :

1. A @ symbol is used as well as '.' is used.

2. There is at least one character before the @ symbol and in between @ and '.'
3. There is at least one character after the '.' symbol.

```
CONSTRAINT check_Property_Type CHECK(Property_Type IN
( 'House','Apartment', 'Bungalow', 'Duplex','Cottage', 'Other'))
```

This constraint uses CHECK keyword to ensure that the Property type must be either of these types.

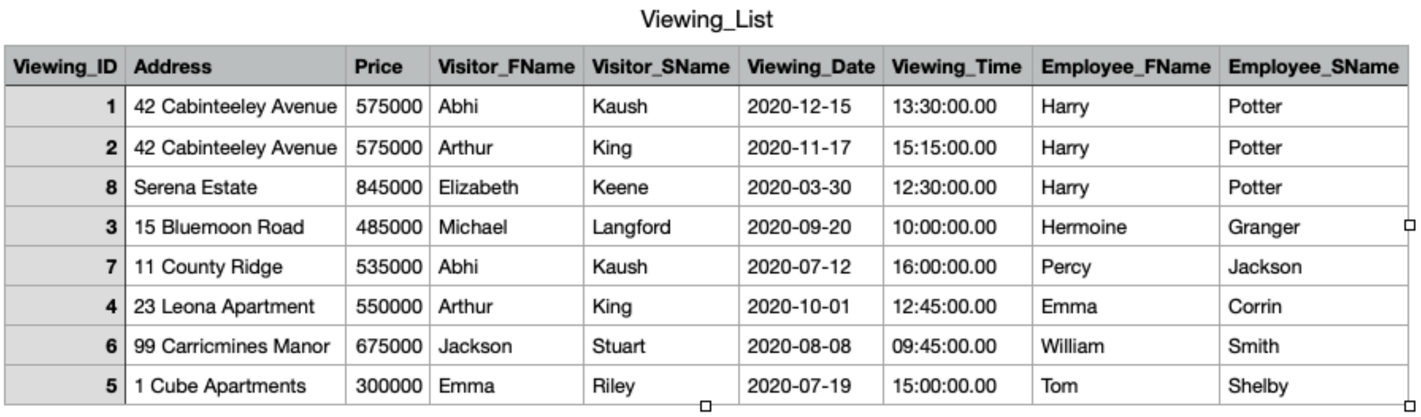
VIEWS and TRIGGERS

VIEWS:

The Estate Agency database uses 3 Views which are mentioned below. 1 of them is only visible to the managers of the agencies and the rest are visible to both Managers and Executives. This is done by adding security measures which are explained later.

```
CREATE VIEW Viewing_List AS
  (SELECT
     Viewing.Viewing_ID,
     Property.Street AS Address,
     Property.Price,
     Buyer.FName AS Visitor_FName,
     Buyer.SName AS Visitor_SName,
     Viewing.Viewing_Date AS Viewing_Date,
     Viewing.Viewing_Time,
     Employee.FName AS Employee_FName,
     Employee.SName AS Employee_SName
  FROM
     Viewing,
     Property,
     Employee,
     Buyer
  WHERE
     Viewing.Buyer_ID = Buyer.Buyer_ID
         AND Viewing.Employee_ID = Employee.Employee_ID
         AND Viewing.Property_ID = Property.Property_ID);
```

This is the first view created which can be viewed by both Manager and Executives. This view gives a detailed report of all the property viewings that have been set up by employees for visitors. The table shows the date, time and location of the viewings as well.



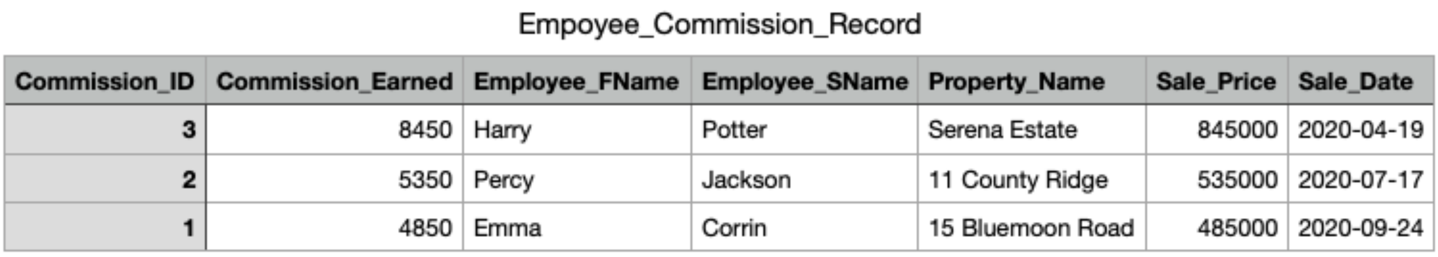
Viewing_List

Viewing_ID	Address	Price	Visitor_FName	Visitor_SName	Viewing_Date	Viewing_Time	Employee_FName	Employee_SName
1	42 Cabinteeley Avenue	575000	Abhi	Kaush	2020-12-15	13:30:00.00	Harry	Potter
2	42 Cabinteeley Avenue	575000	Arthur	King	2020-11-17	15:15:00.00	Harry	Potter
8	Serena Estate	845000	Elizabeth	Keene	2020-03-30	12:30:00.00	Harry	Potter
3	15 Bluemoon Road	485000	Michael	Langford	2020-09-20	10:00:00.00	Hermoine	Granger
7	11 County Ridge	535000	Abhi	Kaush	2020-07-12	16:00:00.00	Percy	Jackson
4	23 Leona Apartment	550000	Arthur	King	2020-10-01	12:45:00.00	Emma	Corrin
6	99 Carricmines Manor	675000	Jackson	Stuart	2020-08-08	09:45:00.00	William	Smith
5	1 Cube Apartments	300000	Emma	Riley	2020-07-19	15:00:00.00	Tom	Shelby

```
CREATE VIEW Empoyee_Commission_Record AS
    (SELECT
        Commission.Commission_ID,
        Commission.Amount AS Commission_Earned,
        Employee.FName AS Employee_FName,
        Employee.SName AS Employee_SName,
        Property.Street AS Property_Name,
        Property.Price AS Sale_Price,
        Sale_Record.Sale_Date
    FROM
        Commission,
        Employee,
        Property,
        Sale_Record
    WHERE
        Commission.Sale_Records_ID = Sale_Record.Sale_Records_ID
            AND Sale_Record.Employee_ID = Employee.Employee_ID
            AND Sale_Record.Property_ID = Property.Property_ID
            );
```

This view is only visible to the employees of the agency at a Manager level. This table shows the details of all the Commissions that is generated from property sales. It displays the amount the employee earned on a sale, employee's details and property details.

The result looks like this:

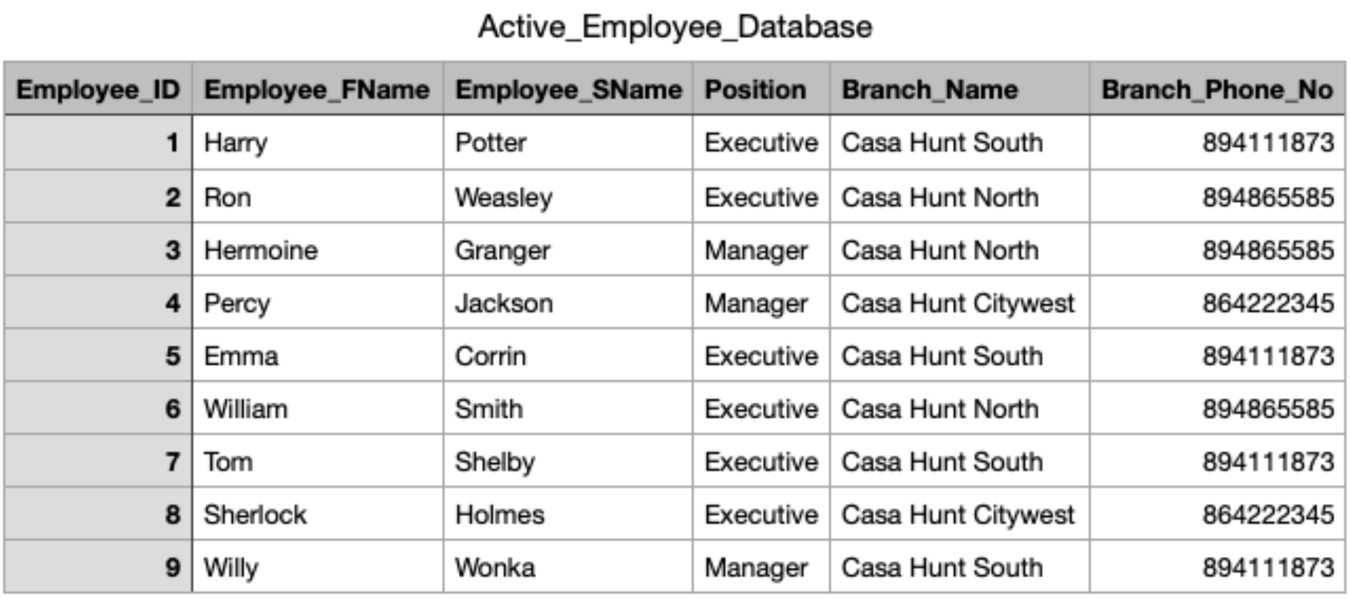

Empoyee_Commission_Record

Commission_ID	Commission_Earned	Employee_FName	Employee_SName	Property_Name	Sale_Price	Sale_Date
3	8450	Harry	Potter	Serena Estate	845000	2020-04-19
2	5350	Percy	Jackson	11 County Ridge	535000	2020-07-17
1	4850	Emma	Corrin	15 Bluemoon Road	485000	2020-09-24

```
CREATE VIEW Active_Employee_Database AS
    SELECT
        Employee.Employee_ID,
        Employee.FName AS Employee_FName,
        Employee.SName AS Employee_SName,
        Emp_Position AS Position,
        Branch.Name AS Branch_Name,
        Branch.Phone_No AS Branch_Phone_No
    FROM
        Employee, Branch
        WHERE
            Employee.Emp_Status = 'Active' AND Branch.Branch_ID =
Employee.Branch_ID;
```

This View is created with the purpose of showing details of Employees and where they work at in the Agency which other employees can use to find details about their colleagues. This can be viewed by all the employees regardless of their roles.

The View looks like this:



Active_Employee_Database

Employee_ID	Employee_FName	Employee_SName	Position	Branch_Name	Branch_Phone_No
1	Harry	Potter	Executive	Casa Hunt South	894111873
2	Ron	Weasley	Executive	Casa Hunt North	894865585
3	Hermoine	Granger	Manager	Casa Hunt North	894865585
4	Percy	Jackson	Manager	Casa Hunt Citywest	864222345
5	Emma	Corrin	Executive	Casa Hunt South	894111873
6	William	Smith	Executive	Casa Hunt North	894865585
7	Tom	Shelby	Executive	Casa Hunt South	894111873
8	Sherlock	Holmes	Executive	Casa Hunt Citywest	864222345
9	Willy	Wonka	Manager	Casa Hunt South	894111873

TRIGGERS:

In the database, there are 2 triggers which are initiated when a new record is stored in the Sale_Record table. Their code and details are mentioned below:

```
DELIMITER $$
CREATE TRIGGER Update_Property_Details
AFTER INSERT ON Sale_Record FOR EACH ROW
begin
  DECLARE New_Owner varchar(50);
```

```
        -- Creating Buyer into a New Owner
        INSERT INTO Owner(Owner_ID, FName, SName, Phone_No) select NULL,
Buyer.FName, Buyer.SName, Buyer.Phone_No from Buyer where Buyer.Buyer_ID =
NEW.Buyer_ID;

        -- Tagging the new Owner as the Owner of the property
         Select Max(Owner_ID) from Owner into New_Owner;
         -- Status to Sold and Price to Sale Price
           UPDATE Property
         SET Property.Property_Status = 'Sold', Property.Price = New.Sale_Price,
Property.Owner_ID = New_Owner, Property.Date_Updated = New.Sale_Date
        WHERE New.Property_ID = Property.Property_ID;
    END;
    $$
    DELIMITER ;
```

The first trigger is called Update_Property_Details. This is initiated after a new record is inserted in the Sales_Record table. It does a number of things -

1. As the property is sold, the property has a new Owner now. Thus the property's Owner_ID is outdated. To update the Owner_ID, firstly it creates a new Owner_ID using the details of the buyer with the help of Buyer_ID. After the new Owner_ID is created, the outdated Owner_ID is updated to the new one.
2. It updates the Property Status to Sold and the price of the property to the price at which the property was purchased at.
3. Lastly, it updates the Date_Refreshed column which tells us when these changes on the property made.

```
DELIMITER $$
CREATE TRIGGER Allocate_Commmission
AFTER INSERT ON Sale_Record FOR EACH ROW
begin
```

```
    -- Giving Commission to the Employee in the sale
     INSERT INTO Commission(Commission_ID, Commission_Date, Amount,
Sale_Records_ID) SELECT null, New.Sale_Date, (0.01*New.Sale_Price),
New.Sale_Records_ID;
  END;
  $$
  DELIMITER ;
```

The second trigger is called Allocate_Commmission. It is used to allocate commission to the employees when the property has been sold and sale has been recorded. After the entry in Sale_Record is finished, a new entry is inserted in the Commission table. It generates a new commission_id, takes in the date from the Sale_Record, calculates the commission amount (which is 1%) and tags the Sale_Record_ID. In case the manager wants to change the commission amount or any other field, manager can do that using the UPDATE command.

SECURITY of the Database

Estate Agency has a lot of sensitive data of employees, buyers and sellers. Protecting information like PPS number, salary, commissions and many other things is hugely important in a database.

To ensure this, I have created two roles in my database - Manager and Executive.

```
CREATE ROLE IF NOT EXISTS MANAGER;
CREATE ROLE IF NOT EXISTS EXECUTIVE;
```

The Manager has access to all the tables and views whereas the Executives cannot see some of the sensitive information. If we talk about the views, I have used the following commands to hide the Commission View from the Executives but available to Managers.

```
  GRANT ALL ON Empoyee_Commission_Record TO MANAGER WITH GRANT
OPTION;
  GRANT ALL ON Viewing_List TO MANAGER WITH GRANT OPTION;
```

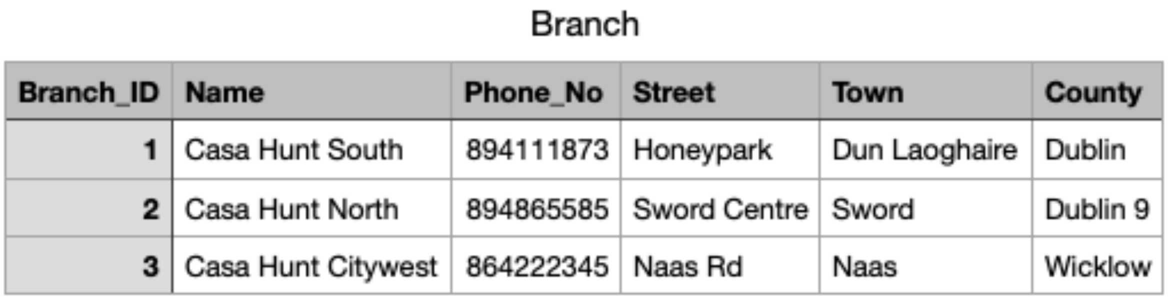
GRANT ALL ON Active_Employee_Database TO MANAGER WITH GRANT OPTION;

GRANT INSERT, SELECT, UPDATE ON Viewing_List TO EXECUTIVE;
GRANT SELECT, UPDATE ON Active_Employee_Database TO EXECUTIVE;

However, if the Manager wants to share the access or visibility of any particular view with any executive, he/she can do so as they have a Grant Option which is used to grant access to anyone in the database.

Outputs of Database -

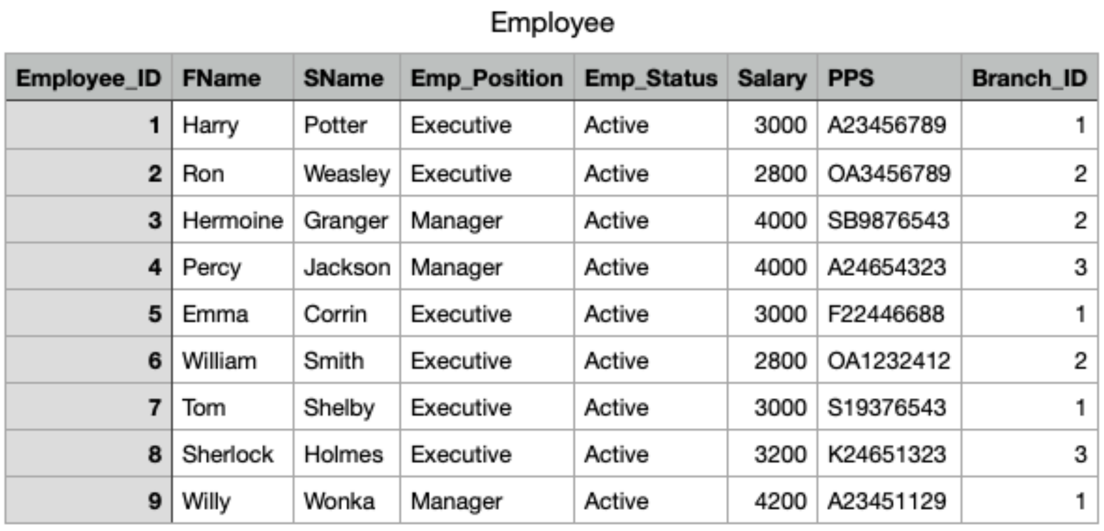
SELECT * FROM BRANCH;



Branch

Branch_ID	Name	Phone_No	Street	Town	County
1	Casa Hunt South	894111873	Honeypark	Dun Laoghaire	Dublin
2	Casa Hunt North	894865585	Sword Centre	Sword	Dublin 9
3	Casa Hunt Citywest	864222345	Naas Rd	Naas	Wicklow

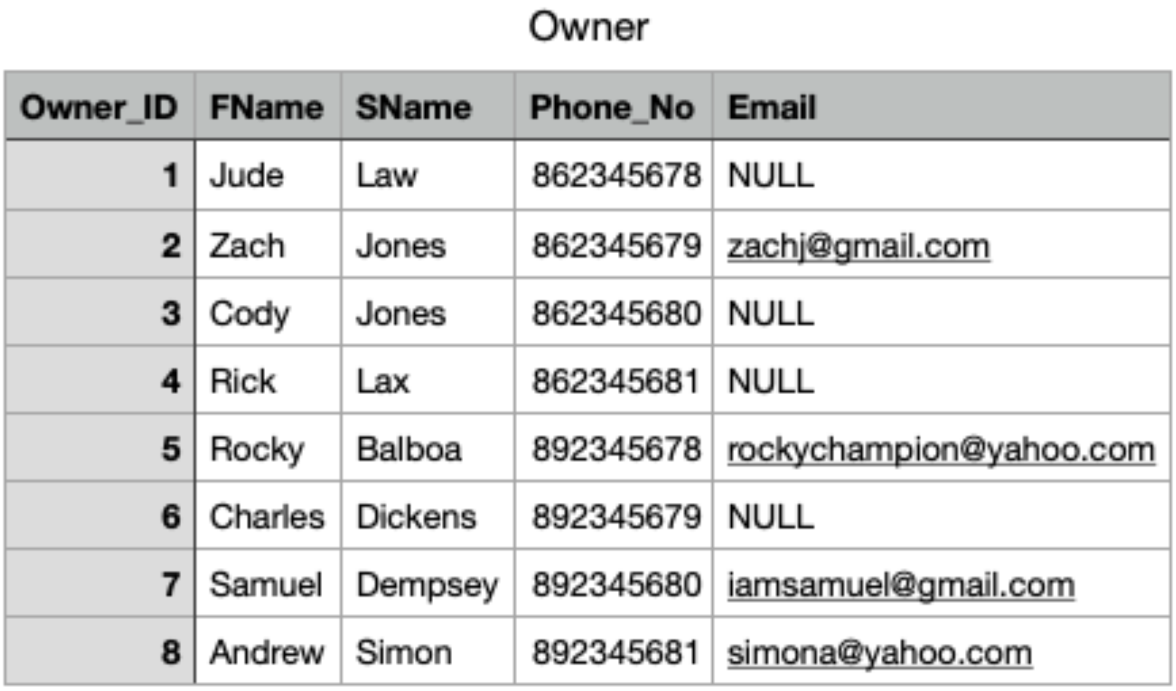
SELECT * FROM Employee;



Employee

Employee_ID	FName	SName	Emp_Position	Emp_Status	Salary	PPS	Branch_ID
1	Harry	Potter	Executive	Active	3000	A23456789	1
2	Ron	Weasley	Executive	Active	2800	OA3456789	2
3	Hermoine	Granger	Manager	Active	4000	SB9876543	2
4	Percy	Jackson	Manager	Active	4000	A24654323	3
5	Emma	Corrin	Executive	Active	3000	F22446688	1
6	William	Smith	Executive	Active	2800	OA1232412	2
7	Tom	Shelby	Executive	Active	3000	S19376543	1
8	Sherlock	Holmes	Executive	Active	3200	K24651323	3
9	Willy	Wonka	Manager	Active	4200	A23451129	1

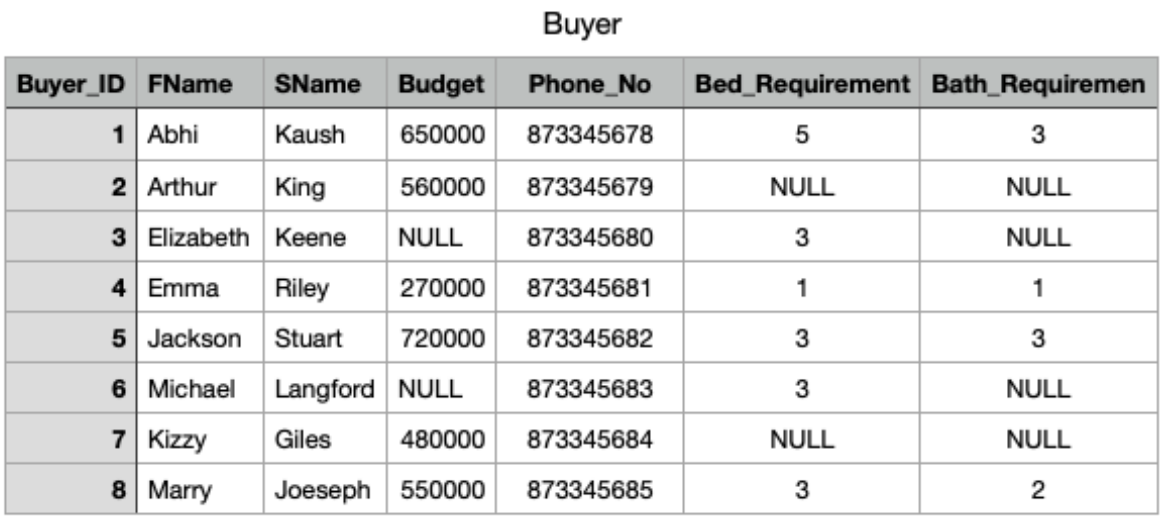
SELECT * FROM OWNER;



Owner

Owner_ID	FName	SName	Phone_No	Email
1	Jude	Law	862345678	NULL
2	Zach	Jones	862345679	zachj@gmail.com
3	Cody	Jones	862345680	NULL
4	Rick	Lax	862345681	NULL
5	Rocky	Balboa	892345678	rockychampion@yahoo.com
6	Charles	Dickens	892345679	NULL
7	Samuel	Dempsey	892345680	iamsamuel@gmail.com
8	Andrew	Simon	892345681	simona@yahoo.com

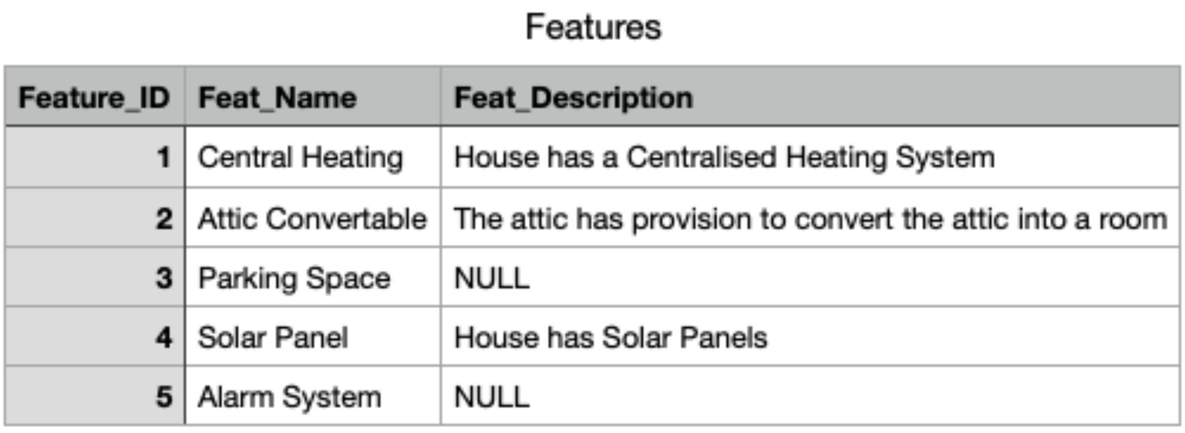
SELECT * FROM Buyer;



Buyer

Buyer_ID	FName	SName	Budget	Phone_No	Bed_Requirement	Bath_Requiremen
1	Abhi	Kaush	650000	873345678	5	3
2	Arthur	King	560000	873345679	NULL	NULL
3	Elizabeth	Keene	NULL	873345680	3	NULL
4	Emma	Riley	270000	873345681	1	1
5	Jackson	Stuart	720000	873345682	3	3
6	Michael	Langford	NULL	873345683	3	NULL
7	Kizzy	Giles	480000	873345684	NULL	NULL
8	Marry	Joeseph	550000	873345685	3	2

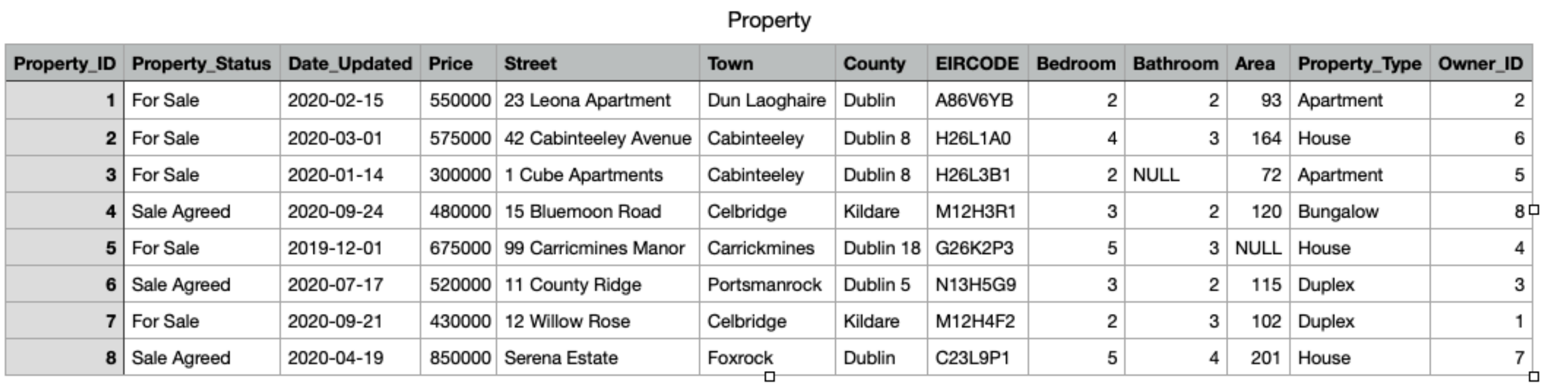
SELECT * FROM FEATURES;



Features

Feature_ID	Feat_Name	Feat_Description
1	Central Heating	House has a Centralised Heating System
2	Attic Convertable	The attic has provision to convert the attic into a room
3	Parking Space	NULL
4	Solar Panel	House has Solar Panels
5	Alarm System	NULL

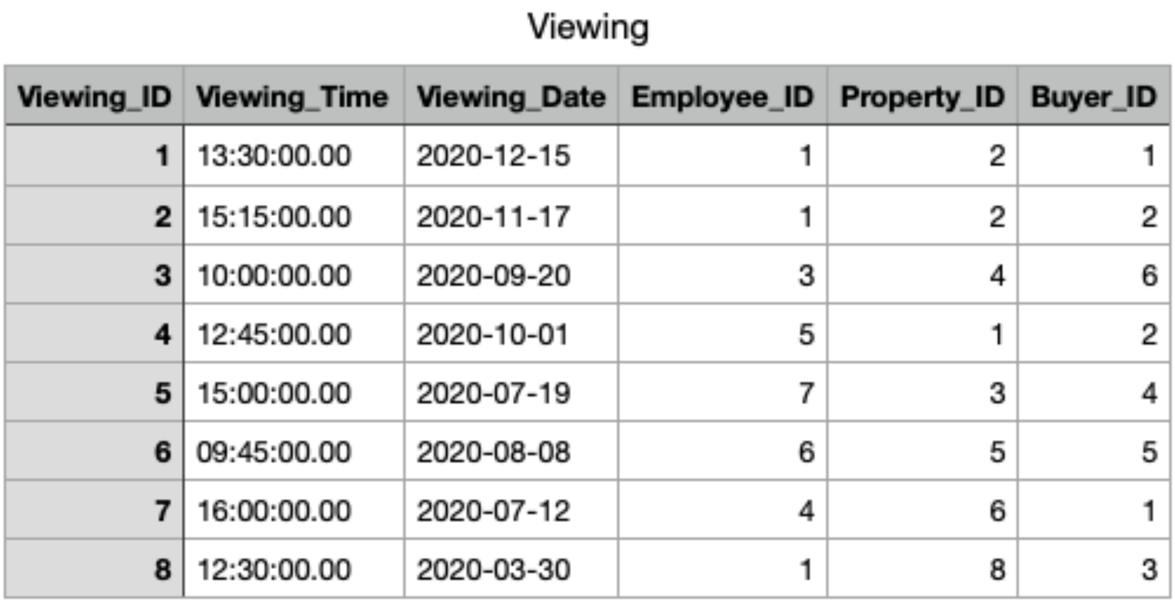
SELECT * FROM Property;



Property

Property_ID	Property_Status	Date_Updated	Price	Street	Town	County	EIRCODE	Bedroom	Bathroom	Area	Property_Type	Owner_ID
1	For Sale	2020-02-15	550000	23 Leona Apartment	Dun Laoghaire	Dublin	A86V6YB	2	2	93	Apartment	2
2	For Sale	2020-03-01	575000	42 Cabinteeley Avenue	Cabinteeley	Dublin 8	H26L1A0	4	3	164	House	6
3	For Sale	2020-01-14	300000	1 Cube Apartments	Cabinteeley	Dublin 8	H26L3B1	2	NULL	72	Apartment	5
4	Sale Agreed	2020-09-24	480000	15 Bluemoon Road	Celbridge	Kildare	M12H3R1	3	2	120	Bungalow	8
5	For Sale	2019-12-01	675000	99 Carricmines Manor	Carrickmines	Dublin 18	G26K2P3	5	3	NULL	House	4
6	Sale Agreed	2020-07-17	520000	11 County Ridge	Portsmanrock	Dublin 5	N13H5G9	3	2	115	Duplex	3
7	For Sale	2020-09-21	430000	12 Willow Rose	Celbridge	Kildare	M12H4F2	2	3	102	Duplex	1
8	Sale Agreed	2020-04-19	850000	Serena Estate	Foxrock	Dublin	C23L9P1	5	4	201	House	7

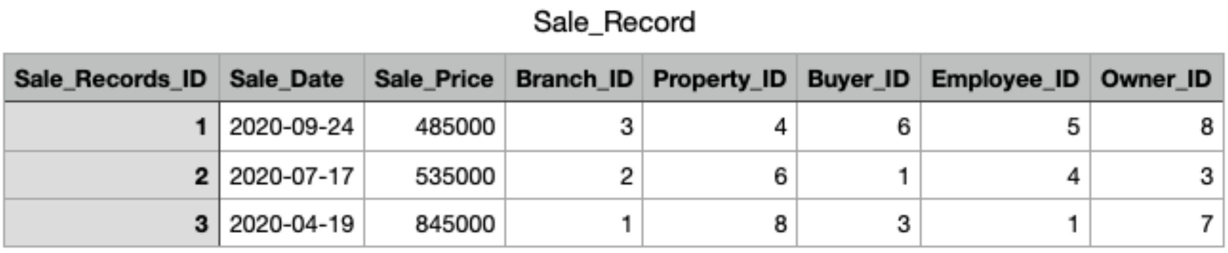
SELECT * FROM Viewing;



Viewing

Viewing_ID	Viewing_Time	Viewing_Date	Employee_ID	Property_ID	Buyer_ID
1	13:30:00.00	2020-12-15	1	2	1
2	15:15:00.00	2020-11-17	1	2	2
3	10:00:00.00	2020-09-20	3	4	6
4	12:45:00.00	2020-10-01	5	1	2
5	15:00:00.00	2020-07-19	7	3	4
6	09:45:00.00	2020-08-08	6	5	5
7	16:00:00.00	2020-07-12	4	6	1
8	12:30:00.00	2020-03-30	1	8	3

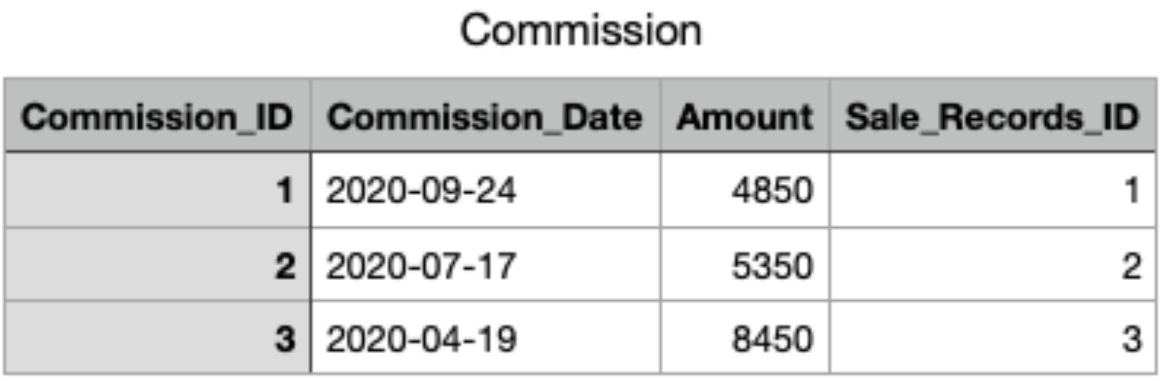
SELECT * FROM Sale_Record;



Sale_Record

Sale_Records_ID	Sale_Date	Sale_Price	Branch_ID	Property_ID	Buyer_ID	Employee_ID	Owner_ID
1	2020-09-24	485000	3	4	6	5	8
2	2020-07-17	535000	2	6	1	4	3
3	2020-04-19	845000	1	8	3	1	7

SELECT * FROM Commission;

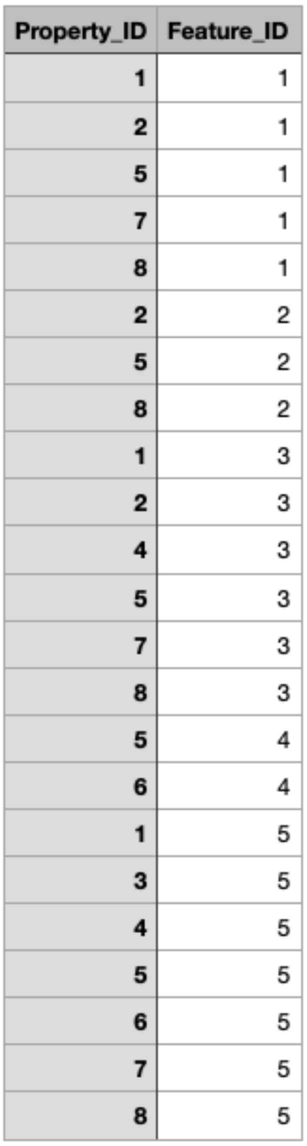


Commission

Commission_ID	Commission_Date	Amount	Sale_Records_ID
1	2020-09-24	4850	1
2	2020-07-17	5350	2
3	2020-04-19	8450	3

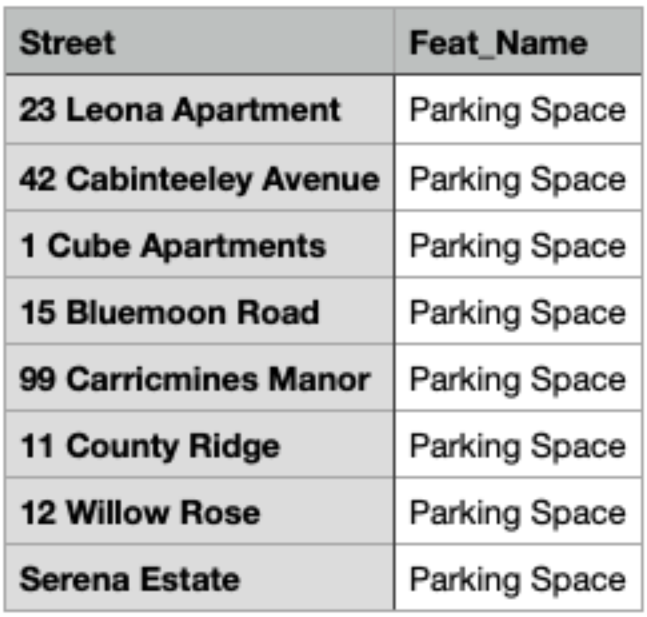
```
Select * From Property_Features;
```

Property_Features



Property_ID	Feature_ID
1	1
2	1
5	1
7	1
8	1
2	2
5	2
8	2
1	3
2	3
4	3
5	3
7	3
8	3
5	4
6	4
1	5
3	5
4	5
5	5
6	5
7	5
8	5

Property_With_Parking_Space



Street	Feat_Name
23 Leona Apartment	Parking Space
42 Cabinteeley Avenue	Parking Space
1 Cube Apartments	Parking Space
15 Bluemoon Road	Parking Space
99 Carricmines Manor	Parking Space
11 County Ridge	Parking Space
12 Willow Rose	Parking Space
Serena Estate	Parking Space

```
SELECT distinct Property.Street,
Features.Feat_Name
FROM Property, Features, Property_Features
WHERE Features.Feat_Name = 'Parking
Space' ;
```

APPENDIX

```
Drop Database IF EXISTS Estate_Agency;

CREATE DATABASE Estate_Agency;
USE Estate_Agency;

        CREATE TABLE IF NOT EXISTS Branch
        (
          Branch_ID INT NOT NULL AUTO_INCREMENT,
          Name CHAR(255) NOT NULL,
          Phone_No INT  NOT NULL,
          Street CHAR(255),
          Town CHAR(255),
          County CHAR(255),
          PRIMARY KEY (Branch_ID),
          CONSTRAINT check_Phone_No CHECK ( Phone_No < 10000000000 )
        );


        Insert Into Branch Values(001, 'Casa Hunt South', 894111873, 'Honeypark', 'Dun Laoghaire', 'Dublin');
        Insert Into Branch Values(002, 'Casa Hunt North', 0894865585, 'Sword Centre', 'Sword', 'Dublin 9');
        Insert Into Branch Values(003, 'Casa Hunt Citywest', 864222345, 'Naas Rd', 'Naas', 'Wicklow');


        CREATE TABLE IF NOT EXISTS Employee
        (
          Employee_ID INT NOT NULL AUTO_INCREMENT,
          FName VARCHAR(20) NOT NULL,
          SName VARCHAR(20) NOT NULL,
          Emp_Position VARCHAR(20) NOT NULL,
          Emp_Status VARCHAR(20) NOT NULL,
          Salary INT NOT NULL,
          PPS VARCHAR(9) UNIQUE NOT NULL,
          Branch_ID INT NOT NULL,
          PRIMARY KEY (Employee_ID),
          FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID),
          CONSTRAINT check_PPS CHECK(CHARACTER_LENGTH(PPS) = 9 AND SUBSTRING(PPS FROM 1
FOR 1) BETWEEN 'A' AND 'Z'),
          CONSTRAINT check_Position CHECK(Emp_Position = 'Manager' OR Emp_Position = 'Executive'),
          CONSTRAINT check_Status CHECK(Emp_Status IN ('Active','Inactive'))
        );
```

```
INSERT INTO Employee VALUES( 0001, 'Harry', 'Potter', 'Executive', 'Active', 3000, 'A23456789', 001);
INSERT INTO Employee VALUES( 0002, 'Ron', 'Weasley', 'Executive', 'Active', 2800, 'OA3456789', 002);
INSERT INTO Employee VALUES( 0003, 'Hermoine', 'Granger', 'Manager', 'Active', 4000, 'SB9876543', 002);
INSERT INTO Employee VALUES( 0004, 'Percy', 'Jackson', 'Manager', 'Active', 4000, 'A24654323', 003);
INSERT INTO Employee VALUES( 0005, 'Emma', 'Corrin', 'Executive', 'Active', 3000, 'F22446688', 001);
INSERT INTO Employee VALUES( 0006, 'William', 'Smith', 'Executive', 'Active', 2800, 'OA1232412', 002);
INSERT INTO Employee VALUES( 0007, 'Tom', 'Shelby', 'Executive', 'Active', 3000, 'S19376543', 001);
INSERT INTO Employee VALUES( 0008, 'Sherlock', 'Holmes', 'Executive', 'Active', 3200, 'K24651323', 003);
INSERT INTO Employee VALUES( 0009, 'Willy', 'Wonka', 'Manager', 'Active', 4200, 'A23451129', 001);



CREATE TABLE IF NOT EXISTS Owner
(
  Owner_ID INT NOT NULL auto_increment,
  FName VARCHAR(20) NOT NULL,
  SName VARCHAR(20) NOT NULL,
  Phone_No INT  NOT NULL,
Email varchar(50),
  PRIMARY KEY (Owner_ID),
  CONSTRAINT check_Phone CHECK ( Phone_No < 10000000000 ),
CONSTRAINT check_Email CHECK ( Email  LIKE '%_@__%.__%' )
);


INSERT INTO Owner VALUES( 001, 'Jude', 'Law', '862345678',NULL);
INSERT INTO Owner VALUES( 002, 'Zach', 'Jones', '862345679','zachj@gmail.com');
INSERT INTO Owner VALUES( 003, 'Cody', 'Jones', '862345680',NULL);
INSERT INTO Owner VALUES( 004, 'Rick', 'Lax', '862345681',NULL);
INSERT INTO Owner VALUES( 005, 'Rocky', 'Balboa', '892345678', 'rockychampion@yahoo.com');
INSERT INTO Owner VALUES( 006, 'Charles', 'Dickens', '892345679',NULL);
INSERT INTO Owner VALUES( 007, 'Samuel', 'Dempsey', '892345680', 'iamsamuel@gmail.com');
INSERT INTO Owner VALUES( 008, 'Andrew', 'Simon', '892345681','simona@yahoo.com');


CREATE TABLE IF NOT EXISTS Buyer
(
  Buyer_ID INT NOT NULL AUTO_INCREMENT,
  FName VARCHAR(20) NOT NULL,
  SName VARCHAR(20) NOT NULL,
  Budget INT,
  Phone_No INT  NOT NULL,
  Bed_Requirement INT,
  Bath_Requiremen INT,
  PRIMARY KEY (Buyer_ID),
  CONSTRAINT check_Phone_No3 CHECK ( Phone_No < 10000000000 )
);

INSERT INTO Buyer VALUES( 001, 'Abhi', 'Kaush', 650000, '873345678', 5, 3);
INSERT INTO Buyer VALUES( 002, 'Arthur', 'King', 560000, '873345679', NULL, NULL);
INSERT INTO Buyer VALUES( 003, 'Elizabeth', 'Keene', NULL, '873345680',3, NULL);
INSERT INTO Buyer VALUES( 004, 'Emma', 'Riley', 270000, '873345681', 1, 1);
INSERT INTO Buyer VALUES( 005, 'Jackson', 'Stuart', 720000, '873345682', 3, 3);
INSERT INTO Buyer VALUES( 006, 'Michael', 'Langford', NULL, '873345683', 3, NULL);
INSERT INTO Buyer VALUES( 007, 'Kizzy', 'Giles', 480000, '873345684', NULL, NULL);
INSERT INTO Buyer VALUES( 008, 'Marry', 'Joeseph', 550000, '873345685', 3, 2);



CREATE TABLE IF NOT EXISTS Features
(
  Feature_ID INT NOT NULL AUTO_INCREMENT,
  Feat_Name  VARCHAR(60) NOT NULL,
  Feat_Description VARCHAR(255),
  PRIMARY KEY (Feature_ID)
);

INSERT INTO Features VALUES(001, 'Central Heating', 'House has a Centralised Heating System');
INSERT INTO Features VALUES(002, 'Attic Convertable', 'The attic has provision to convert the attic into a
room');
INSERT INTO Features VALUES(003, 'Parking Space', NULL);
INSERT INTO Features VALUES(004, 'Solar Panel', 'House has Solar Panels');
INSERT INTO Features VALUES(005, 'Alarm System', NULL);


CREATE TABLE IF NOT EXISTS Property
(
  Property_ID INT NOT NULL AUTO_INCREMENT,
  Property_Status VARCHAR(20) NOT NULL,
  Date_Updated DATE NOT NULL,
  Price INT NOT NULL,
  Street CHAR(255) NOT NULL,
  Town CHAR(255) NOT NULL,
  County CHAR(255) NOT NULL,
  EIRCODE VARCHAR(7) UNIQUE,
  Bedroom INT,
  Bathroom INT,
            Area INT,
            Property_Type VARCHAR(20) NOT NULL,
            Owner_ID INT NOT NULL,
            PRIMARY KEY (Property_ID),
            FOREIGN KEY (Owner_ID) REFERENCES Owner(Owner_ID),
            CONSTRAINT check_Property_Status CHECK(Property_Status = 'For Sale' OR Property_Status = 'Sale
Agreed' OR Property_Status = 'Sold'),
            CONSTRAINT check_EIRCODE CHECK(CHARACTER_LENGTH(EIRCODE) = 7 AND
SUBSTRING(EIRCODE FROM 1 FOR 1) BETWEEN 'A' AND 'Z'),
            CONSTRAINT check_Property_Type CHECK(Property_Type IN ( 'House','Apartment', 'Bungalow',
'Duplex','Cottage', 'Other'))
        );



        INSERT INTO Property VALUES(001, 'For Sale', '2020-02-15', 550000, '23 Leona Apartment', 'Dun Laoghaire',
'Dublin', 'A86V6YB', 2, 2, 93, 'Apartment', 002);
        INSERT INTO Property VALUES(002, 'For Sale', '2020-03-1', 575000, '42 Cabinteeley Avenue', 'Cabinteeley',
'Dublin 8', 'H26L1A0', 4, 3, 164, 'House', 006);
        INSERT INTO Property VALUES(003, 'For Sale', '2020-01-14', 300000, '1 Cube Apartments', 'Cabinteeley',
'Dublin 8', 'H26L3B1', 2, NULL, 72, 'Apartment', 005);
        INSERT INTO Property VALUES(004, 'Sale Agreed', '2020-09-24', 480000, '15 Bluemoon Road', 'Celbridge',
'Kildare', 'M12H3R1', 3, 2, 120, 'Bungalow', 008);
        INSERT INTO Property VALUES(005, 'For Sale', '2019-12-01', 675000, '99 Carricmines Manor', 'Carrickmines',
'Dublin 18', 'G26K2P3', 5, 3, NULL, 'House', 004);
        INSERT INTO Property VALUES(006, 'Sale Agreed', '2020-7-17', 520000, '11 County Ridge', 'Portsmanrock',
'Dublin 5', 'N13H5G9', 3, 2, 115, 'Duplex', 003);
        INSERT INTO Property VALUES(007, 'For Sale', '2020-09-21', 430000, '12 Willow Rose', 'Celbridge', 'Kildare',
'M12H4F2', 2, 3, 102, 'Duplex', 001);
        INSERT INTO Property VALUES(008, 'Sale Agreed', '2020-04-19', 850000, 'Serena Estate', 'Foxrock', 'Dublin',
'C23L9P1', 5, 4, 201, 'House', 007);



/*
(Prints Property ID, Street, Price, Owner Name)
SELECT Property.Property_ID, Property.Street, Property.Price, Owner.FName, Owner.SName FROM Property, Owner
where Property.Owner_ID = Owner.Owner_ID;
*/

  CREATE TABLE IF NOT EXISTS Viewing
        (
            Viewing_ID INT NOT NULL AUTO_INCREMENT,
            Viewing_Time TIME(2) NOT NULL,
            Viewing_Date DATE NOT NULL,
            Employee_ID INT NOT NULL,
            Property_ID INT NOT NULL,
            Buyer_ID INT NOT NULL,
                PRIMARY KEY (Viewing_ID),
                FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID),
                FOREIGN KEY (Property_ID) REFERENCES Property(Property_ID),
                FOREIGN KEY (Buyer_ID) REFERENCES Buyer(Buyer_ID)
              );

              INSERT INTO Viewing VALUES (001,'13:30', '2020-12-15', 001, 002, 001);
    INSERT INTO Viewing VALUES (002,'15:15', '2020-11-17', 001, 002, 002);
    INSERT INTO Viewing VALUES (003,'10:00', '2020-09-20', 003, 004, 006);
              INSERT INTO Viewing VALUES (004,'12:45', '2020-10-1', 005, 001, 002);
    INSERT INTO Viewing VALUES (005,'15:00', '2020-07-19', 007, 003, 004);
    INSERT INTO Viewing VALUES (006,'09:45', '2020-08-08', 006, 005, 005);
    INSERT INTO Viewing VALUES (007,'16:00', '2020-07-12', 004, 006, 001);
    INSERT INTO Viewing VALUES (008,'12:30', '2020-03-30', 001, 008, 003);
```

```
 /*
 (Prints Property ID Owner ID Owner Name and Date Updated for the House whose sale is agreed. Can be used by manager)
    select property_id, Property.owner_id, Owner.FName, date_updated, Property_Status from Property, Owner where Property_Status = 'Sale Agreed' AND Owner.Owner_ID = Property.Owner_ID;
 */

CREATE TABLE Sale_Record
(
  Sale_Records_ID INT NOT NULL AUTO_INCREMENT,
  Sale_Date DATE NOT NULL,
  Sale_Price INT NOT NULL,
  Branch_ID INT NOT NULL,
  Property_ID INT NOT NULL,
  Buyer_ID INT NOT NULL,
  Employee_ID INT NOT NULL,
  Owner_ID INT NOT NULL,
  PRIMARY KEY (Sale_Records_ID),
  FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID),
  FOREIGN KEY (Property_ID) REFERENCES Property(Property_ID),
  FOREIGN KEY (Buyer_ID) REFERENCES Buyer(Buyer_ID),
  FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID),
  FOREIGN KEY (Owner_ID) REFERENCES Owner(Owner_ID)
);
```

```
CREATE TABLE Commission
(
  Commission_ID INT NOT NULL AUTO_INCREMENT,
  Commission_Date DATE NOT NULL,
  Amount INT NOT NULL,
  Sale_Records_ID INT NOT NULL,
  PRIMARY KEY (Commission_ID),
  FOREIGN KEY (Sale_Records_ID) REFERENCES Sale_Record(Sale_Records_ID)
);




CREATE TABLE Property_Features
(
  Property_ID INT NOT NULL,
  Feature_ID INT NOT NULL,
  PRIMARY KEY (Property_ID, Feature_ID),
  FOREIGN KEY (Property_ID) REFERENCES Property(Property_ID),
  FOREIGN KEY (Feature_ID) REFERENCES Features(Feature_ID)
);

INSERT INTO Property_Features VALUES(001, 001);
INSERT INTO Property_Features VALUES(001, 003);
INSERT INTO Property_Features VALUES(001, 005);
INSERT INTO Property_Features VALUES(002, 001);
INSERT INTO Property_Features VALUES(002, 002);
INSERT INTO Property_Features VALUES(002, 003);
INSERT INTO Property_Features VALUES(003, 005);
INSERT INTO Property_Features VALUES(004, 003);
INSERT INTO Property_Features VALUES(004, 005);
INSERT INTO Property_Features VALUES(005, 001);
INSERT INTO Property_Features VALUES(005, 002);
INSERT INTO Property_Features VALUES(005, 003);
INSERT INTO Property_Features VALUES(005, 004);
INSERT INTO Property_Features VALUES(005, 005);
INSERT INTO Property_Features VALUES(006, 004);
INSERT INTO Property_Features VALUES(006, 005);
INSERT INTO Property_Features VALUES(007, 001);
INSERT INTO Property_Features VALUES(007, 003);
INSERT INTO Property_Features VALUES(007, 005);
INSERT INTO Property_Features VALUES(008, 001);
INSERT INTO Property_Features VALUES(008, 002);
INSERT INTO Property_Features VALUES(008, 003);
INSERT INTO Property_Features VALUES(008, 005);


-- select distinct Property.Street, Features.Feat_Name from Property, Features, Property_Features where Features.Feat_Name
= 'Parking Space' ;


-- 2 TRIGGERS
DELIMITER $$
CREATE TRIGGER Update_Property_Details
AFTER INSERT ON Sale_Record FOR EACH ROW
begin
        DECLARE New_Owner varchar(50);
  -- Creating Buyer into a New Owner
  INSERT INTO Owner(Owner_ID, FName, SName, Phone_No) select NULL, Buyer.FName, Buyer.SName,
Buyer.Phone_No from Buyer where Buyer.Buyer_ID = NEW.Buyer_ID;

  -- Tagging the new Owner as the Owner of the property
   Select Max(Owner_ID) from Owner into New_Owner;
   -- Status to Sold and Price to Sale Price
    UPDATE Property
   SET Property.Property_Status = 'Sold', Property.Price = New.Sale_Price, Property.Owner_ID = New_Owner,
Property.Date_Updated = New.Sale_Date
  WHERE New.Property_ID = Property.Property_ID;
END;
$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER Allocate_Commmission
AFTER INSERT ON Sale_Record FOR EACH ROW
begin
        -- Giving Commission to the Employee in the sale
  INSERT INTO Commission(Commission_ID, Commission_Date, Amount, Sale_Records_ID) SELECT null,
New.Sale_Date, (0.01*New.Sale_Price), New.Sale_Records_ID;
END;
$$
DELIMITER ;


-- These Values result into Sale of 3 Houses
```

```
INSERT INTO Sale_Record VALUES(001, '2020-09-24', 485000, 003, 004,006,005, 008);
INSERT INTO Sale_Record VALUES(002, '2020-07-17', 535000, 002, 006, 001, 004, 003);
INSERT INTO Sale_Record VALUES(003, '2020-04-19', 845000, 001, 008, 003, 001, 007);


-- SECURITY and VIEWS
/* 3 VIEWS CREATED */
CREATE VIEW Viewing_List AS
  (SELECT
    Viewing.Viewing_ID,
    Property.Street AS Address,
    Property.Price,
    Buyer.FName AS Visitor_FName,
    Buyer.SName AS Visitor_SName,
    Viewing.Viewing_Date AS Viewing_Date,
    Viewing.Viewing_Time,
    Employee.FName AS Employee_FName,
    Employee.SName AS Employee_SName
  FROM
    Viewing,
    Property,
    Employee,
    Buyer
  WHERE
    Viewing.Buyer_ID = Buyer.Buyer_ID
      AND Viewing.Employee_ID = Employee.Employee_ID
      AND Viewing.Property_ID = Property.Property_ID);


CREATE VIEW Empoyee_Commission_Record AS
  (SELECT
    Commission.Commission_ID,
    Commission.Amount AS Commission_Earned,
    Employee.FName AS Employee_FName,
    Employee.SName AS Employee_SName,
    Property.Street AS Property_Name,
    Property.Price AS Sale_Price,
    Sale_Record.Sale_Date
  FROM
    Commission,
    Employee,
    Property,
    Sale_Record
    WHERE
        Commission.Sale_Records_ID = Sale_Record.Sale_Records_ID
            AND Sale_Record.Employee_ID = Employee.Employee_ID
            AND Sale_Record.Property_ID = Property.Property_ID
            );



CREATE VIEW Active_Employee_Database AS
    SELECT
        Employee.Employee_ID,
        Employee.FName AS Employee_FName,
        Employee.SName AS Employee_SName,
        Emp_Position AS Position,
        Branch.Name AS Branch_Name,
        Branch.Phone_No AS Branch_Phone_No
    FROM
        Employee, Branch
    WHERE
        Employee.Emp_Status = 'Active' AND Branch.Branch_ID = Employee.Branch_ID;




CREATE ROLE IF NOT EXISTS MANAGER;
CREATE ROLE IF NOT EXISTS EXECUTIVE;

GRANT ALL ON Empoyee_Commission_Record TO MANAGER WITH GRANT OPTION;
GRANT ALL ON Viewing_List TO MANAGER WITH GRANT OPTION;
GRANT ALL ON Active_Employee_Database TO MANAGER WITH GRANT OPTION;

GRANT INSERT, SELECT, UPDATE ON Viewing_List TO EXECUTIVE;
GRANT SELECT, UPDATE ON Active_Employee_Database TO EXECUTIVE;

-- Test Tables :

-- SELECT * FROM Branch;
-- SELECT * FROM Employee;
-- SELECT * FROM OWNER;
-- SELECT * FROM Buyer;
-- SELECT * FROM FEATURES;
-- SELECT * FROM Property;
-- SELECT * FROM Viewing;
-- Select * From Property_Features;
-- SELECT * FROM Sale_Record;
-- SELECT * FROM Commission;

-- Test Views :

-- SELECT * From Viewing_List;
-- SELECT * From Empoyee_Commission_Record;
-- SELECT * FROM Active_Employee_Database;
```