12/6/2018

Housing Management System

Contents

Current System:

The current system at most of the places around the country are either completely manual or semi-automated. The manual system starts all the way by calling the leasing office for availability and enquiring about the price and signing the lease, making payment and other manual process. This could be good for small housing company but for a company which has properties based in multiple locations this becomes tough, Also the manual process introduces lot of labor cost and window for potential errors. The semi-automated procedure also involves manual procedures but some things like payments are automated.

Proposed Solution:

We are proposing a solution that maintains centralized database to store data about the company's properties located in different places and total number of apartments under each house and details about properties that are already sold or available for sale, leased or vacant apartments. The system would store all the details about each and every property and minimize the manual efforts reducing the overcall costs. The system also contains the capability to receive and respond to complaints and intimating proper concerned department for appropriate solution. The system will also have the capability for the storing and retrieving historical data on demand.

Requirement:

Housing companies wants to maintain a database that contains details of each of its properties at different locations and all their details. The property must contain property Id, name, location, apartment types associated with each property (2BHK,3BHK,1BHK etc.) apartment count which is a composite attribute of leased, vacant and owned apartment along with details on pet's restriction (pets allowed or not).

1. Each property contains many apartments built under it and each apartment contains Apartment ID which is a combination of Apartment number (apartment number) and identifier (building ID), details about rent called payment which is composite attribute of payment date, paid amount and actual annual lease or purchased amount where all attributes can be understood based of apartment current status.

2. Each property is administered by at most one Leasing Office. Each leasing office has Office ID, Manager_SSN, name who manages the office and office location.

3. Leasing office leases or sells apartment to individuals and stores details about Apartment current status like (leased, owned or vacant), lease start date or purchased date wr.t to each

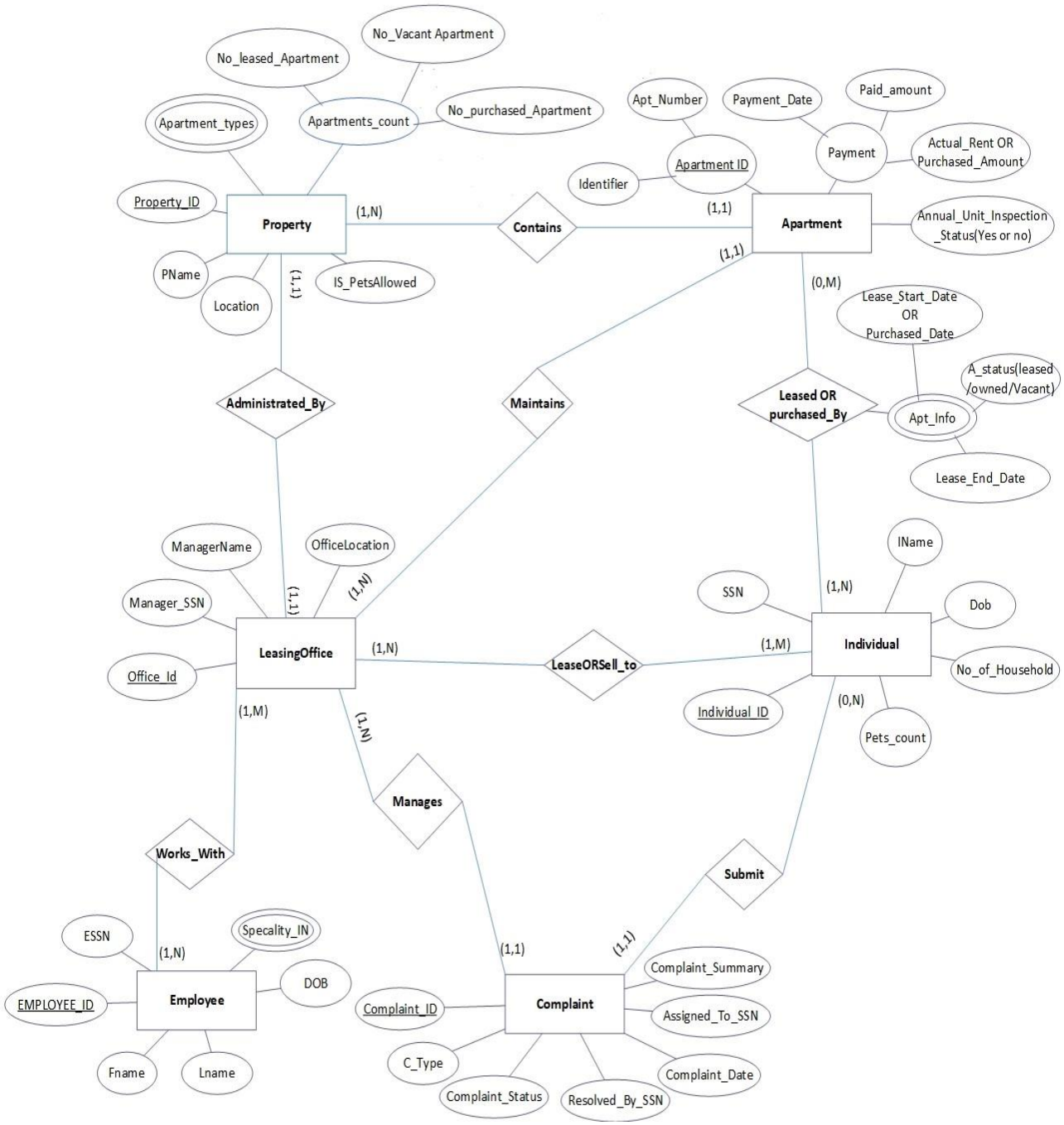
apartment and individual. Leasing office can lease apartments to any number of individuals.

4. Apartments are leased or owned by the individuals whose details contains data about individual ID(SSN), Individual name, DOB, number of household and pets stay in leased or owned apartment. One individual can lease one or more apartments and one apartment can be leased or purchased by one or more individuals who live in that apartment. Individual who leases the apartment can purchase same or different apartment. In addition to this each apartment has the annual inspection status populated.

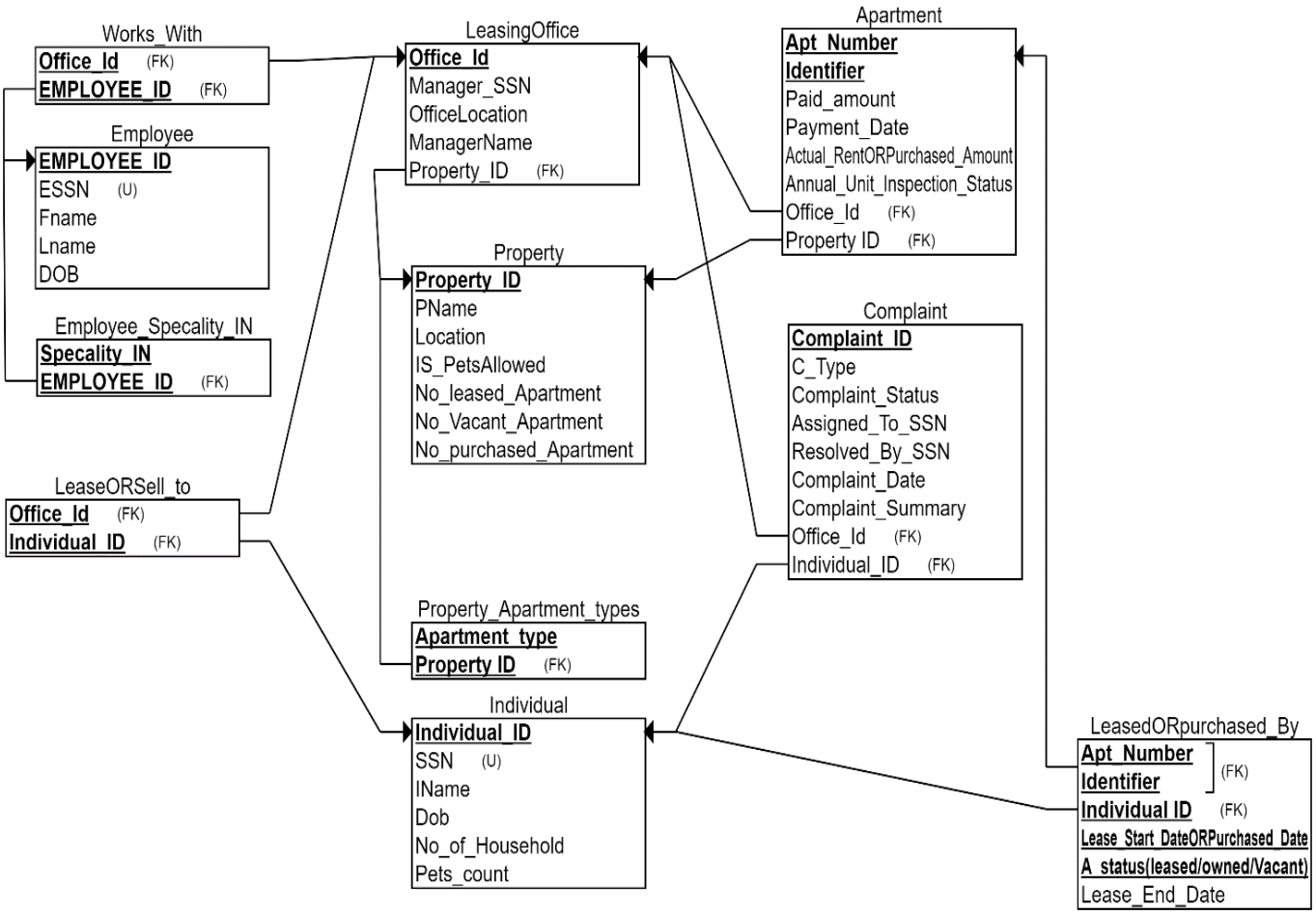
5. An individual can submit number of complaints which will be managed by the leasing office of his resident apartment. Every complaint has complaint id, complaint type, status, assigned person SSN and resolved person SSN who actually resolved the complaint along with date of complaint registered. One complaint can be logged by one individual and managed or resolved by any one the leasing office, but an individual can log any number of complaints and leasing office can manage any number of complaints.

6. Every leasing office has employees who work with them. An employee has employee id, SSN, fname, lanme, DOB and specialization. An employee can work for one or more leasing office but should work for at least at one leasing office. A leasing office can contain more than one employee.

ER Diagram


No_Vacant Apartment
No_leased_Apartment
Apartment_types
Apartments_count
No_purchased_Apartment
Apt_Number
Payment_Date
Paid_amount
Payment
Actual_Rent OR Purchased_Amount
Apartment ID
Identifier
Property_ID
Property
(1,N)
Contains
(1,1)
Apartment
Annual_Unit_Inspection _Status(Yes or no)
PName
Location
(1,1)
IS_PetsAllowed
(1,1)
(0,M)
Lease_Start_Date OR Purchased_Date
A_status(leased /owned/Vacant)
Administrated_By
Maintains
Leased OR purchased_By
Apt_Info
Lease_End_Date
ManagerName
OfficeLocation
(1,1)
(1,N)
IName
SSN
(1,N)
Dob
Manager_SSN
LeasingOffice
(1,N)
LeaseORSell_to
(1,M)
Individual
Office_Id
No_of_Household
(1,M)
(1,N)
Individual_ID
(0,N)
Pets_count
Manages
Works_With
Submit
ESSN
Specality_IN
(1,N)
(1,1)
(1,1)
Complaint_Summary
DOB
Employee
Complaint
EMPLOYEE_ID
Complaint_ID
Assigned_To_SSN
C_Type
Complaint_Date
Fname
Lname
Complaint_Status
Resolved_By_SSN

Relational Database Schema



SQL Create Schema

I have created a schema in MySQL named "HMS" technically the name of the database in which I will be creating the tables using the below queries.

Property Table:

```
use HMS;
CREATE TABLE Property
(
 Property_ID INT AUTO_INCREMENT PRIMARY KEY,
 PName VARCHAR(15) NOT NULL UNIQUE ,
 Location VARCHAR(30)NOT NULL,
 IS_PetsAllowed CHAR(3) NOT NULL,
 No_leased_Apartment INT NOT NULL DEFAULT '0',
 No_Vacant_Apartment INT NOT NULL DEFAULT '0',
 No_purchased_Apartment INT NOT NULL DEFAULT '0'
);
```

Leasing Office Table:

```
CREATE TABLE LeasingOffice
(
 Office_Id INT AUTO_INCREMENT PRIMARY KEY,
 Manager_SSN VARCHAR(9) UNIQUE,
 OfficeLocation VARCHAR(30)NOT NULL,
 ManagerName  VARCHAR(15)NOT NULL,
 Property_ID INT NOT NULL ,
 FOREIGN KEY (Property_ID) REFERENCES Property(Property_ID)ON DELETE CASCADE ON UPDATE CASCADE
);
```

Apartment Table:

```
CREATE TABLE Apartment
(
  Apt_Number INT AUTO_INCREMENT,
  Identifier varchar(5),
  Payment_Date DATE,
  Paid_amount FLOAT ,
  Actual_Annual_RentORPurchased_Amount FLOAT NOT NULL ,
  Annual_Unit_Inspection_Status CHAR(3) NOT NULL DEFAULT 'NO', # yes or No
  Office_Id  INT NOT NULL,
  Property_ID INT NOT NULL,
  PRIMARY KEY(Apt_Number,Identifier),
  FOREIGN KEY (Office_Id) REFERENCES LeasingOffice(Office_Id) ON DELETE CASCADE ON UPDATE
CASCADE,
  FOREIGN KEY (Property_ID) REFERENCES Property(Property_ID)ON DELETE CASCADE ON UPDATE
CASCADE
);
```

Individual Table:

```
CREATE TABLE Individual
(
  Individual_ID INT AUTO_INCREMENT PRIMARY KEY,
  SSN VARCHAR(9) UNIQUE,
  IName VARCHAR(15) NOT NULL,
  Dob DATE NOT NULL,
  No_of_Household INT NOT NULL DEFAULT '0',
  Pets_count INT NOT NULL DEFAULT '0'
);
```

Employee Table:

```
CREATE TABLE Employee
(
 EMPLOYEE_ID INT AUTO_INCREMENT PRIMARY KEY,
 ESSN VARCHAR(9) UNIQUE,
 FName VARCHAR(15) NOT NULL,
 LName VARCHAR(15) NOT NULL,
 DOB DATE NOT NULL
);
```

Employee_Specality_IN Table:

```
CREATE TABLE Employee_Specality_IN
(
Specality_IN VARCHAR(15),
EMPLOYEE_ID INT,
PRIMARY KEY(Specality_IN,EMPLOYEE_ID),
FOREIGN KEY (EMPLOYEE_ID) REFERENCES Employee(EMPLOYEE_ID)ON DELETE CASCADE ON UPDATE CASCADE
);
```

Complaint Table:

```
CREATE TABLE Complaint
(
 Complaint_ID INT PRIMARY KEY AUTO_INCREMENT,
 C_Type VARCHAR(10),
 Complaint_Status VARCHAR(10) NOT NULL DEFAULT 'OPEN' CHECK (Complaint_Status='OPEN' OR Complaint_Status='ASSIGNED' OR Complaint_Status='INPROGRESS' OR Complaint_Status='RESOLVED'),
 Assigned_To_SSN VARCHAR(9),
 Resolved_By_SSN VARCHAR(9),
```

```
 Complaint_Date DATE,
 Complaint_Summary TEXT NOT NULL, #TEXT has no limit of characters MySQL dont support CLOB BLOB.
 Office_Id INT NOT NULL,
 FOREIGN KEY (Office_Id) REFERENCES LeasingOffice(Office_Id) ON DELETE CASCADE ON UPDATE CASCADE,
 Individual_ID INT NOT NULL,
 FOREIGN KEY(Individual_ID) REFERENCES Individual(Individual_ID) On DELETE CASCADE ON UPDATE CASCADE
);
```

Works_With Table:

```
CREATE TABLE Works_With
(
 Office_Id INT,
 EMPLOYEE_ID INT,
 PRIMARY KEY (Office_Id, EMPLOYEE_ID),
 FOREIGN KEY (Office_Id) REFERENCES LeasingOffice(Office_Id) ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (EMPLOYEE_ID) REFERENCES Employee(EMPLOYEE_ID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

LeaseORSell_to Table:

```
CREATE TABLE LeaseORSell_to
(
 Individual_ID INT ,
 Office_Id INT,
 PRIMARY KEY (Office_Id, Individual_ID),
 FOREIGN KEY (Office_Id) REFERENCES LeasingOffice(Office_Id) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
 FOREIGN KEY (Individual_ID) REFERENCES Individual(Individual_ID) ON DELETE CASCADE ON UPDATE
CASCADE

);
```

LeasedORpurchased_By Table

```
CREATE TABLE LeasedORpurchased_By

(

 Apt_Number INT,

 Identifier varchar(5),

 Individual_ID INT,

 A_status   VARCHAR(8) NOT NULL, #leased or owned or vacant

 Lease_Start_DateORPurchased_Date DATE NOT NULL,

 Lease_End_Date DATE,

 PRIMARY KEY (Apt_Number,Identifier, Individual_ID,A_status,Lease_Start_DateORPurchased_Date),

 FOREIGN KEY (Apt_Number,Identifier) REFERENCES Apartment(Apt_Number,Identifier) ON DELETE
CASCADE ON UPDATE CASCADE,

 FOREIGN KEY (Individual_ID) REFERENCES Individual(Individual_ID) ON DELETE CASCADE ON UPDATE
CASCADE

);
```

Property_Apartment_type Table:

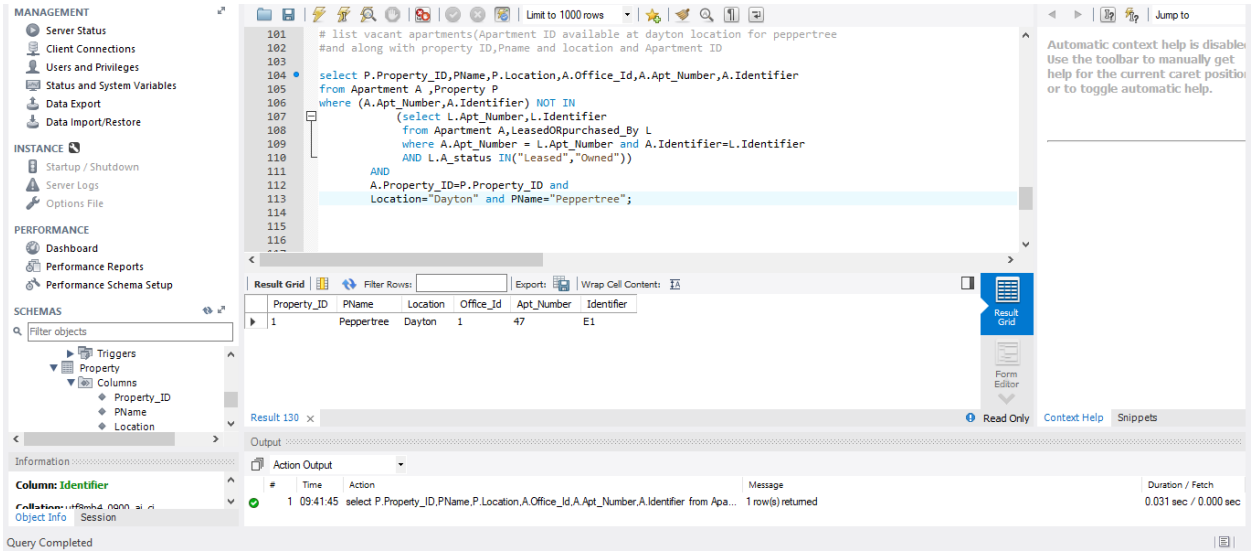
```
CREATE TABLE Property_Apartment_type

(

 Property_ID INT,

 Apartment_type VARCHAR(5) CHECK(Apartment_type='2BHK' OR Apartment_type='3BHK' OR
Apartment_type='1BHK'),

 PRIMARY KEY(Property_ID,Apartment_type),

 FOREIGN KEY (Property_ID) REFERENCES Property(Property_ID) ON DELETE CASCADE ON UPDATE
CASCADE

 );
```

SQL Queries

##list vacant apartments(Apartment ID available at Dayton location for peppertree and along with property ID, Pname and location and Apartment ID.

```
select P.Property_ID,PName,P. Location,A.Office_Id,A.Apt_Number,A.Identifier

from Apartment A , Property P

where (A.Apt_Number,A.Identifier) NOT IN (select L.Apt_Number,L.Identifier

                                          from Apartment A,LeasedORpurchased_By L

                                          where A.Apt_Number = L.Apt_Number

                                          AND  A.Identifier=L.Identifier

                                          AND L.A_status IN("Leased","Owned"))

                                          AND  A.Property_ID=P.Property_ID

                                          AND Location="Dayton" and PName="Peppertree";
```

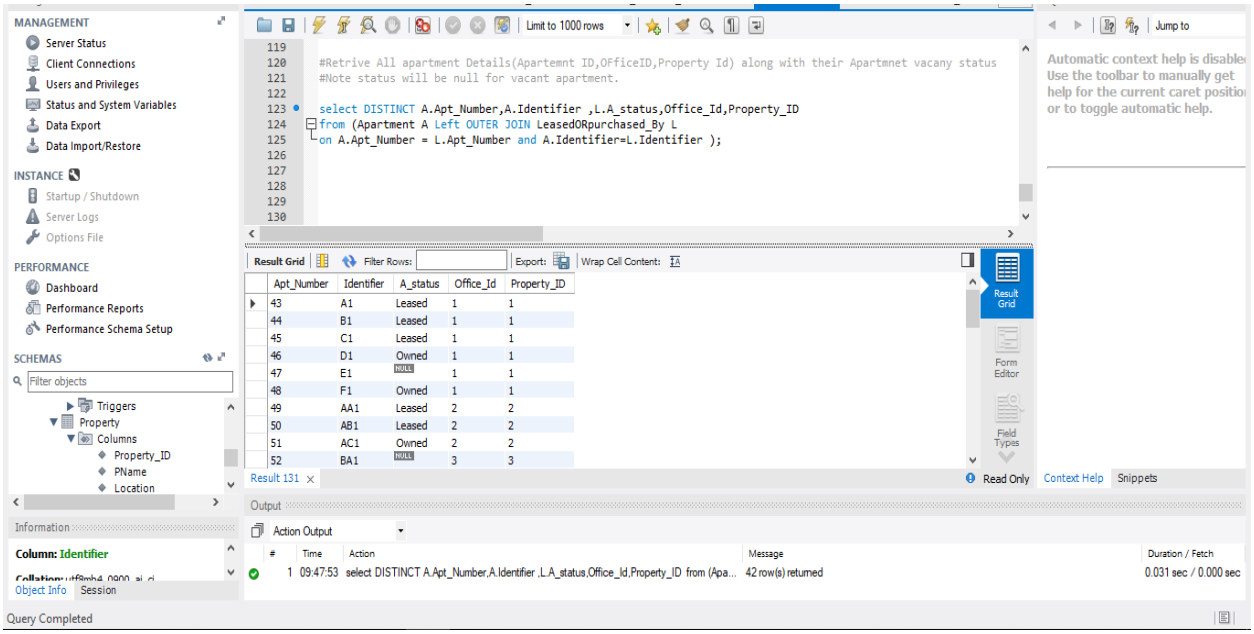
Result:



##Retrive All apartment Details(Apartemnt ID,OFficeID,Property Id) along with their Apartmnet vacany status (Note status will be null for vacant apartment.)

```
select DISTINCT A.Apt_Number,A.Identifier ,L.A_status,Office_Id,Property_ID

from (Apartment A Left OUTER JOIN LeasedORpurchased_By L

on A.Apt_Number = L.Apt_Number and A.Identifier=L.Identifier );
```

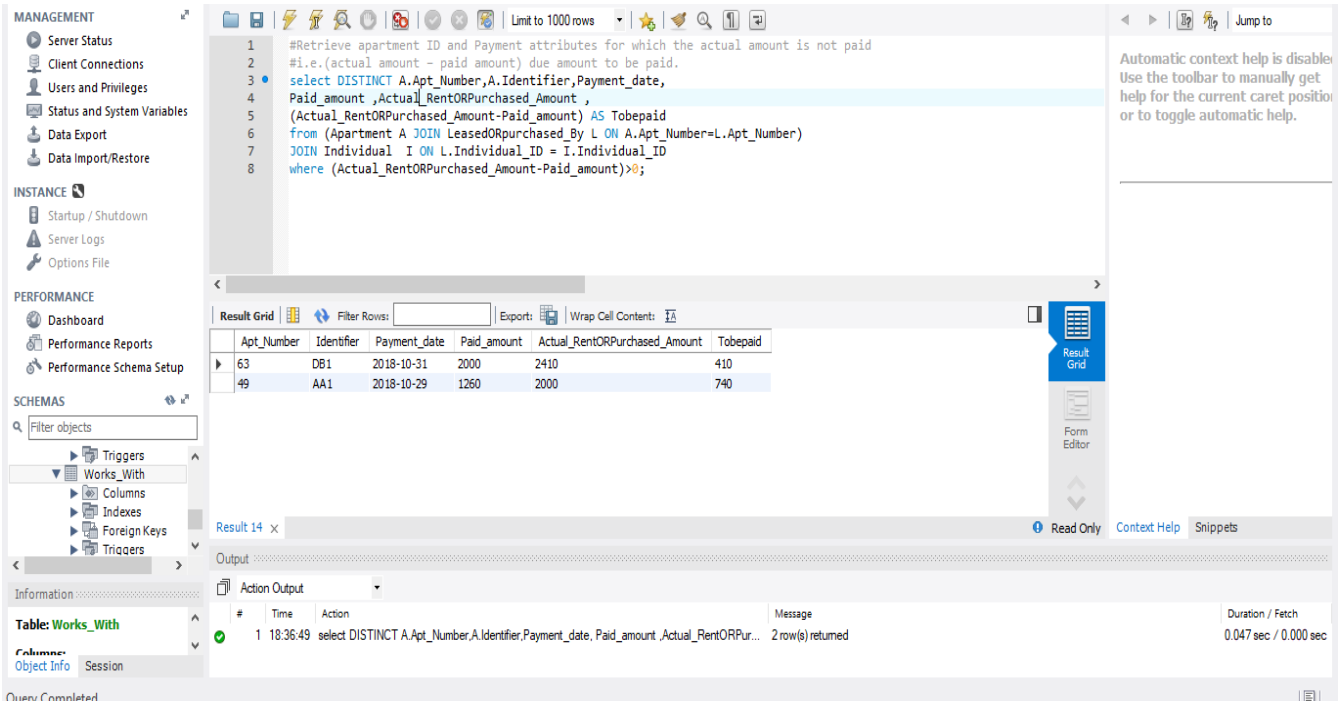
Result:



Retrieve apartment ID and Payment attributes for which the actual amount is not paid (actual amount – paid amount) due amount to be paid.

```
select DISTINCT A.Apt_Number,A.Identifier,Payment_date,
Paid_amount, Actual_RentORPurchased_Amount ,
(Actual_RentORPurchased_Amount-Paid_amount) AS Tobepaid
from (Apartment A JOIN LeasedORpurchased_By L ON A.Apt_Number=L.Apt_Number)
JOIN Individual  I ON L.Individual_ID = I.Individual_ID  where (Actual_RentORPurchased_Amount-Paid_amount)>0;
```

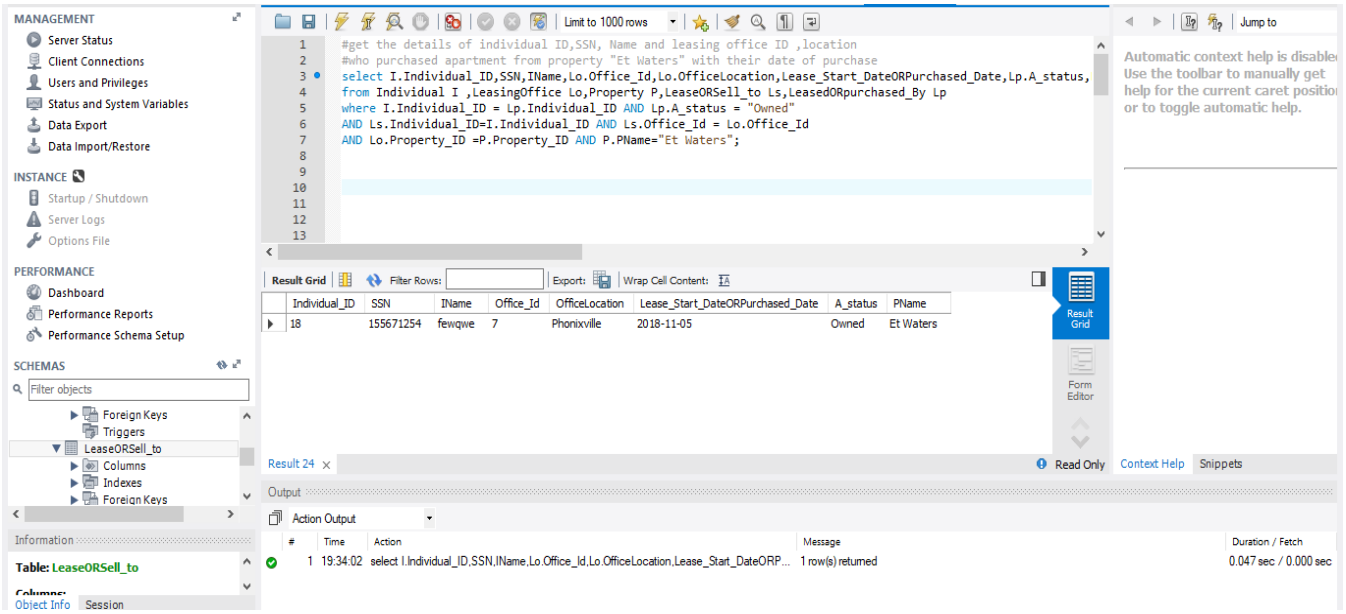
Result:



##Get the details of individual ID,SSN, Name and leasing office ID ,location who purchased apartment from property "Et Waters" with their date of purchase

```
Select I.Individual_ID,SSN,IName,Lo.Office_Id,
Lo.OfficeLocation,Lease_Start_DateORPurchased_Date,Lp.A_status,P.PName
from Individual I ,LeasingOffice Lo,Property P,LeaseORSell_to Ls,LeasedORpurchased_By Lp
where I.Individual_ID = Lp.Individual_ID AND Lp.A_status = "Owned"
AND Ls.Individual_ID=I.Individual_ID AND Ls.Office_Id = Lo.Office_Id
AND Lo.Property_ID =P.Property_ID AND P.PName="Et Waters";
```

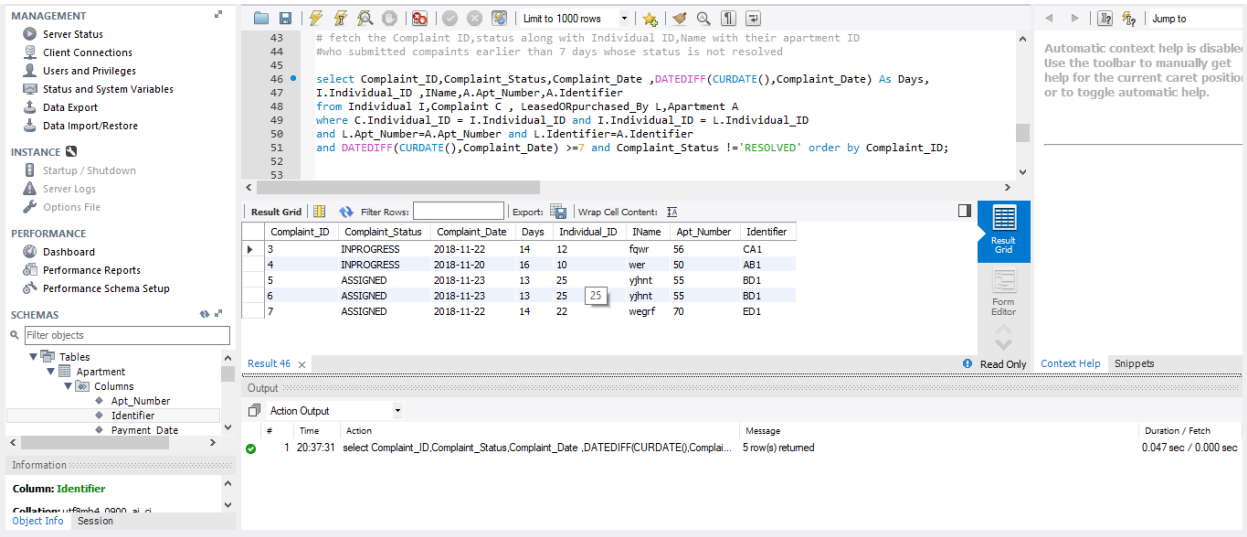
Result:



 # #fetch the Complaint ID,status along with Individual ID,Name with their apartment ID who submitted compaints earlier than 7 days whose status is not resolved.

```
select Complaint_ID,Complaint_Status,Complaint_Date ,DATEDIFF(CURDATE(),Complaint_Date) As Days,
I.Individual_ID ,IName,A.Apt_Number,A.Identifier
from Individual I,Complaint C , LeasedORpurchased_By L,Apartment A
where C.Individual_ID = I.Individual_ID and I.Individual_ID = L.Individual_ID
and L.Apt_Number=A.Apt_Number and L.Identifier=A.Identifier
and DATEDIFF(CURDATE(),Complaint_Date) >=7 and Complaint_Status !='RESOLVED' order by Complaint_ID;
```

Result:

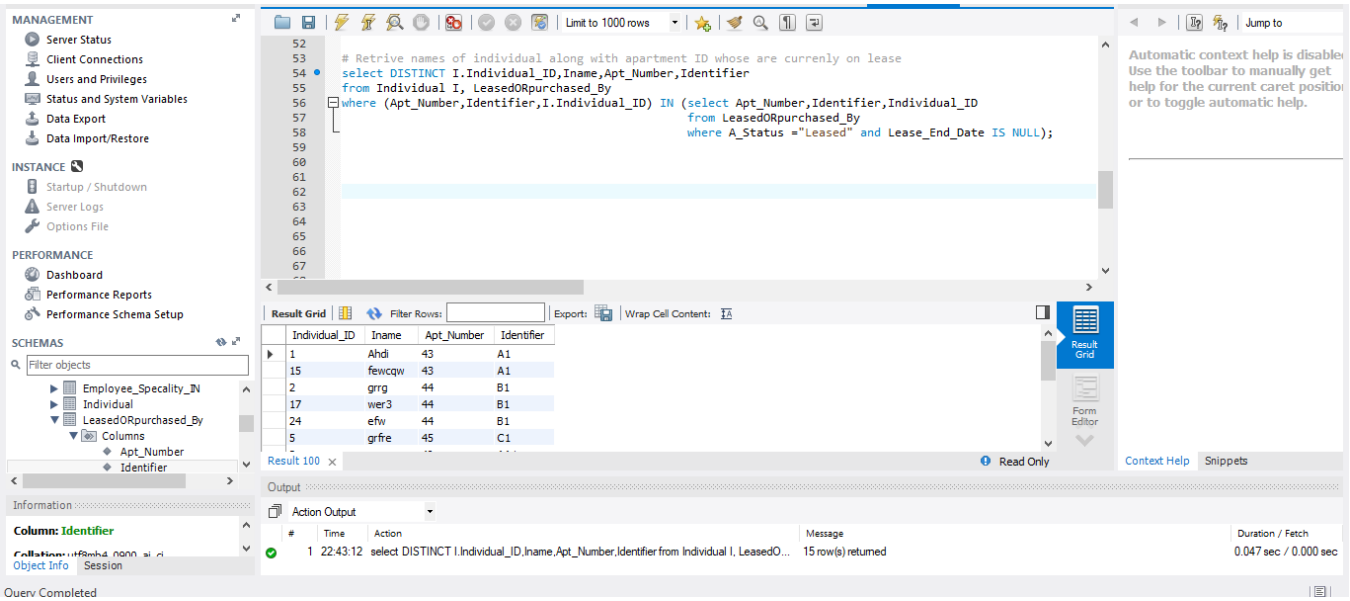


Retrive names of individual whose are currenly on lease.

select DISTINCT I.Individual_ID,Iname,Apt_Number,Identifier from Individual I, LeasedORpurchased_By

where (Apt_Number,Identifier,I.Individual_ID) IN (select Apt_Number,Identifier,Individual_ID from LeasedORpurchased_By where A_Status ="Leased" and Lease_End_Date IS NULL);

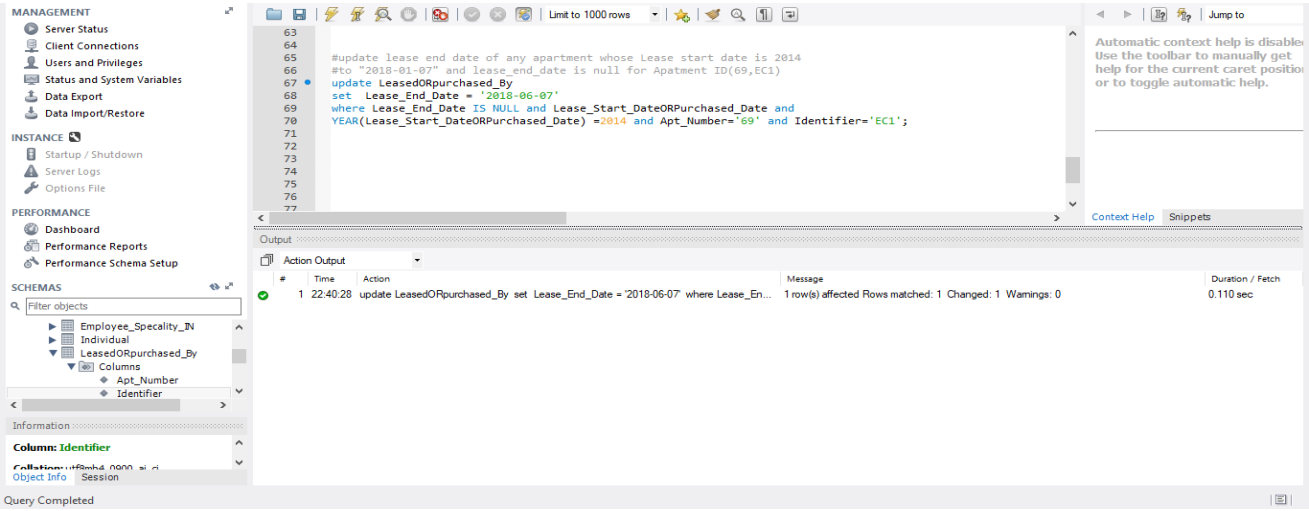
Result:



update lease end date of any apartment whose Lease start date is 2014 to "2018-01-07" and lease_end_date is null for Apatment ID(69,EC1)

update LeasedORpurchased_By set Lease_End_Date = '2018-06-07' where Lease_End_Date IS NULL and Lease_Start_DateORPurchased_Date and YEAR(Lease_Start_DateORPurchased_Date) =2014 and Apt_Number='69' and Identifier='EC1';

Result:



Insert values intlo table Property

INSERT INTO Property(

PName ,Location,IS_PetsAllowed ,No_leased_Apartment ,No_Vacant_Apartment ,

No_purchased_Apartment)

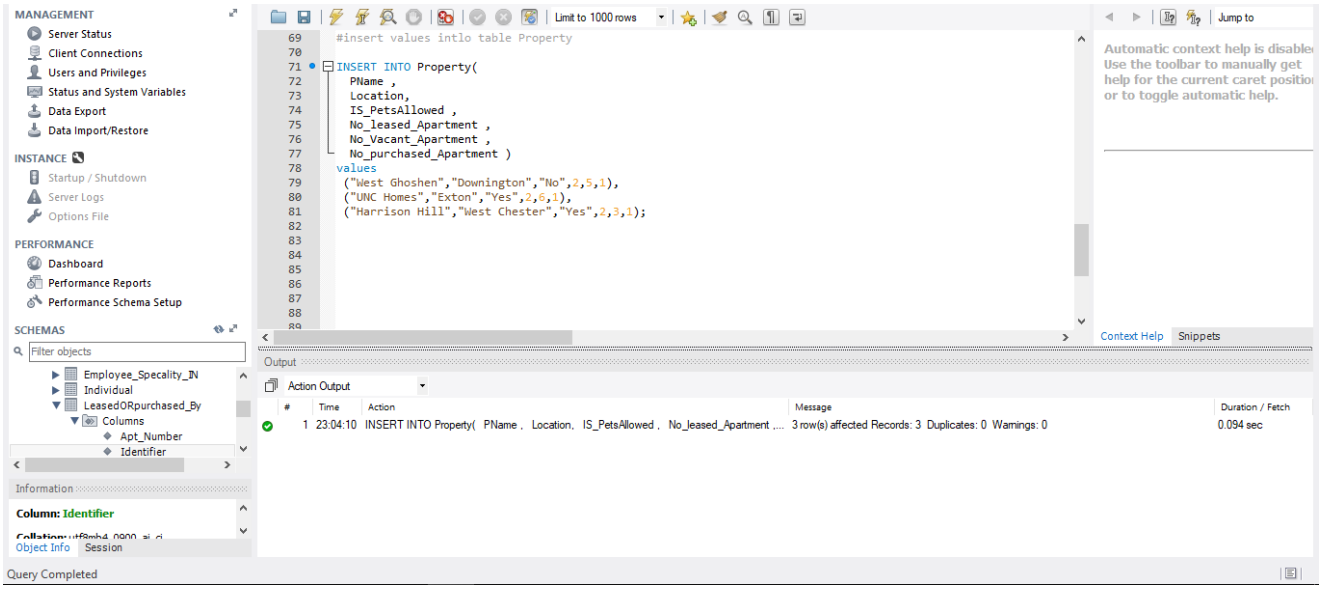
values

("West Ghoshen","Downington","No",2,5,1),

("UNC Homes","Exton","Yes",2,6,1),

("Harrison Hill","West Chester","Yes",2,3,1);

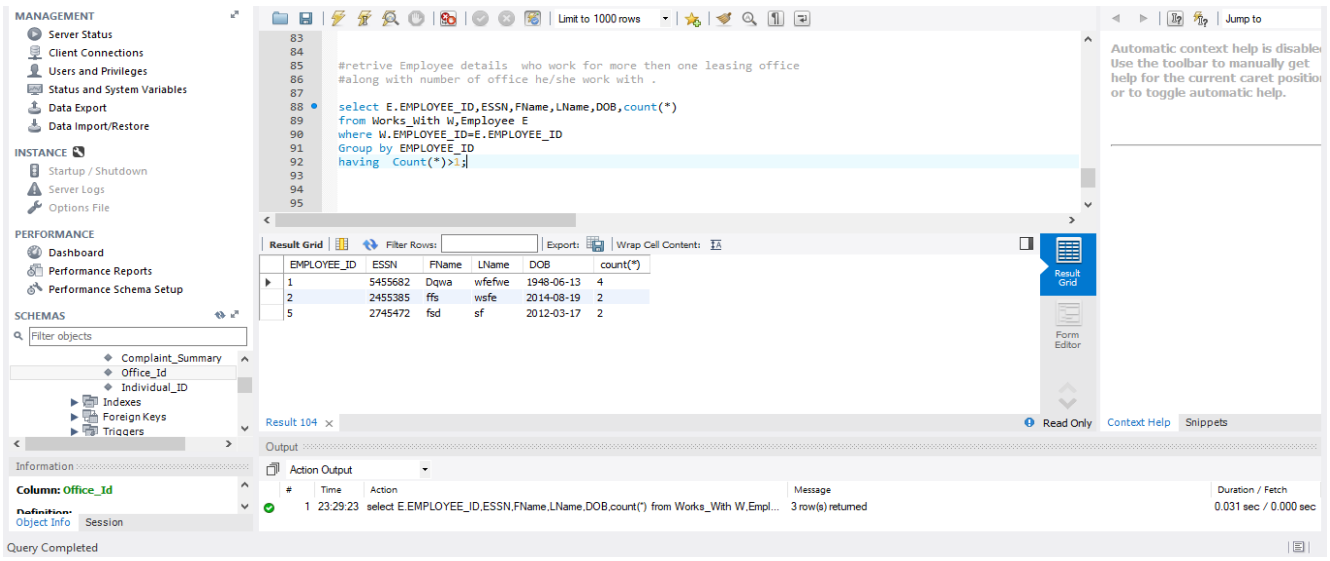
Result:



Retrive Employee details who work for more then one leasing office along with number of office he/she work with.

```
select E.EMPLOYEE_ID,ESSN,FName,LName,DOB,count(*)

from Works_With W,Employee E

where W.EMPLOYEE_ID=E.EMPLOYEE_ID

Group by EMPLOYEE_ID

having  Count(*)>1;
```

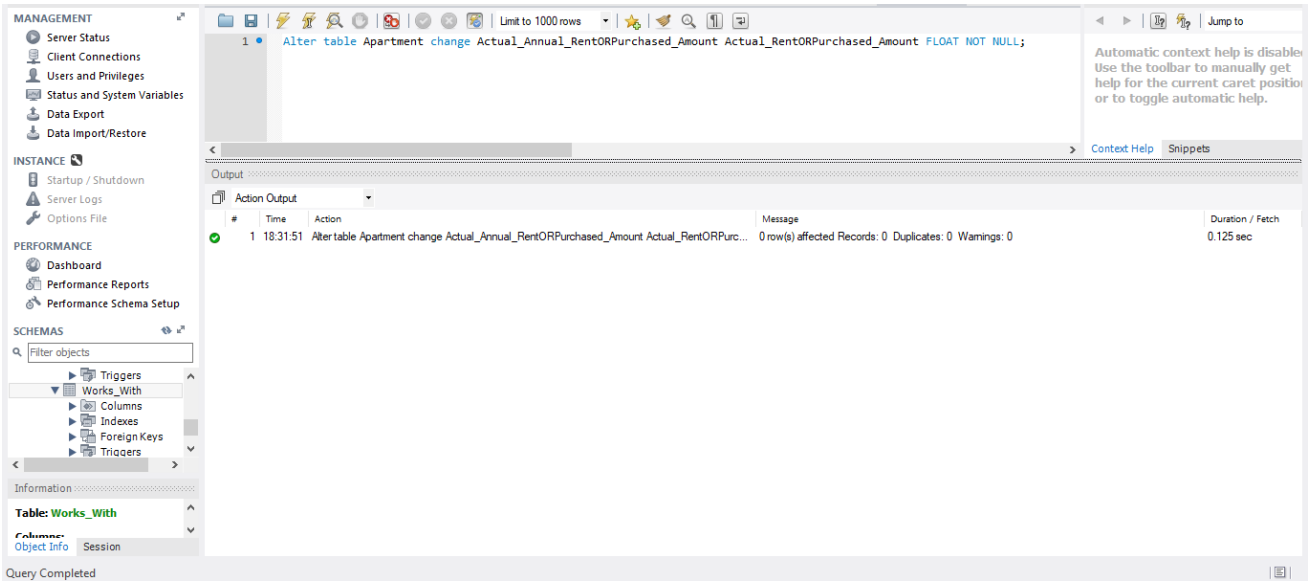
Result:

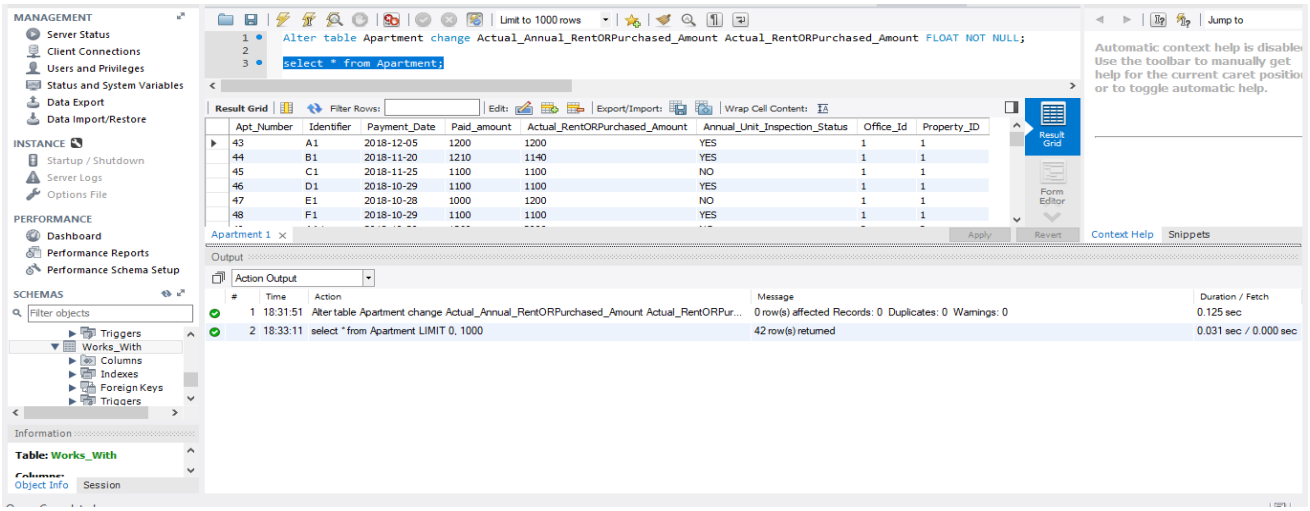


Alter Table:

Alter table Apartment change Actual_Annual_RentORPurchased_Amount Actual_RentORPurchased_Amount FLOAT NOT NULL;

Result:





Discussion points

I initially started this project with the question what database I can design, eventually I came across designing a database for Housing Management System (HMS). While doing this project starting from requirement gathering it has given me an exposure to the real-world experience of entity, attribute and relation along with structural constraints. While converting requirements to ER diagram and to relational schema theoretically everything looks fine until we actually implement it. While implementing the relational database schema I came across some situations when I needed to revisit the ER diagram and realized the mistake of my existing problem about the relation between two entities. My existing knowledge about relation was while defining structural constraint I have defined them in ER diagram with respect to relation itself instead of considering constraint w.r.t to two entity with action item w.r.t to relation.

For example: In the initial ER diagram I initially defined cardinality between individual and Apartment w.r.t leasing information or purchase information as many to many relation as individual can lease or own any number of apartments. But the question comes when we think about the status of an apartment along with purchaser leasing date. Which can me multivalued and composite data as we need to store historical data about on which date who purchased or leased the apartment along with lease end date

With this mistake I had to update my ER diagram and relational schema which resulted in rework which made me clear that clear requirements and correct understanding of the requirement is essential to avoid rework that may occur huge costs in real time projects.

Apart from this I also learned about some datatypes in MySQL. MySQL uses TEXT datatype rather than BLOB/CLOB and we can't use TEXT as primary key so instead we need VARCHAR.Initially When I designed ER diagram with Complaint ID and Complaint summary as relation attributes I came across situation of added TEXT data type as primary key but encountered SQL error ,that is when I realized TEXT can't be used as primary key since the size is not fixed.Later after through analysis of the requirement the attribute "Compaint_ID" was being used with Entity rather than an attribute on Relation which uniquely identify the tuple in Database instance. I also came to know about the normalization and de-normalization process and their importance. In my project most of the tables are normalized and that has given the advantages but also it made the realize about the big queries that needs to be written to get the data. Also, when the table size grows, and joins increase execution time becomes the issue and that is when de normalization is required. I came to know that not always normalization is helpful.

I have used Amazon AWS EC2 instance to run the MySQL database and used the MySQL workbench to connect to the database. This was a learning curve which gave me exposure to AWS and hosting and running DB instances on AWS.

I also learned that in real world scenario there is no physical delete operations that are performed on the data. Instead the delete operation are actually update on a column (like Active_Indicator which is either 0 or 1) and which identifies a row as active or inactive. This helps to maintain the history of the data. However I have not implemented Active indicators in the current Database because it was late by the time I realized it and did not want to make changes at last moment due to time constraints.

Finally by concluding the discussion I would like to add future scope that this system can be further enhanced to support the features like scheduling visits, notifying the future residents about schedules, reminding tenants for their rent payments and a system for notifying users who subscribe about the availability and other details.