**4: Zillow**

* **Tanvi Deosarkar**
* **Aakash Jadhav**
* **Faizalabbas Saiyed**

The Challenge: Design a Database and Implement for Your E-Business Using A Life Cycle Approach to Database Design

**1.** **Describe your understanding of the problem. How will you address the business requirements? What questions do you have and what assumption have you made to address these ambiguities?**

**Zillow Business requirements?**

While researching and analyzing e-commerce business Zillow. We found out the initial requirement is to build an end to end database management system.

**Problem statement** for building the database system is Zillow is getting huge amounts of data from different users and their aim was to store the data and connect the database tables and avoid the partial dependency or transitive dependency from the the data and all the chunk of data should be in 3rd normal form so that these data sets will be used to show front end information.

Users of the systems are Customer, Agent, and zillow website employees. This database will have data about the properties, location, mortgage, amenities, transactions etc

* Scalability: Designing the database system to handle a large volume of data and accommodate future growth as the number of properties and users increases.
* Performance: Optimizing the database system to provide fast and efficient access to property information, enabling users to search, browse, and retrieve data quickly.
* Security: Implementing robust security measures to protect sensitive user information and prevent unauthorized access to the database.

**Hypothetical questions and answers for the requirements:**

1. Which particular data points should be incorporated into the Zillow database design system?

* The main data points we can include are Property ID, Address, Listing Price, Number of Bedrooms, Number of Bathrooms, Square Footage, Property Type, Listing Status, and other basic property details. Additional data points such as property features, images, and historical data can also be incorporated.

2. What kind of structure and organization do you want the data to have inside the database?

* A Relational DataBase model with appropriate tables for entities like Property, Sell Property, Rent Property, Agents, Listings and many more. Then we need to define the relationship between these tables by using PK and FK. After that we need Normalization to avoid data redundancy.

3. Does the database design need to take any special conditions or limitations into account?

* The database should be designed in accordance with local real estate and privacy laws. For example we must manage user data and payment information securely while maintaining compliance to data protection requirements.

4. How often will fresh data need to be added to the database?

* Data will be continuously updated as new properties are listed or sold and user interactions take place. The structure of the database should allow for frequent data additions, updates and data retrieval while keeping the other data as it is.

5. Do you have a preference for the database management system or technological stack that will be employed?

* Yes, we would like to implement a database management system of MS access as the data in our database design will be from small to medium sized.

6. Will any particular search or filtering features need to be supported by the database?

* Websites like zillow require robust search to provide user friendly and efficient experience for property seekers. We are going to include filtering features such as Location Based search, Property Type, Price Range, Square footage, Property Status.

**Assumption:**

While creating relationships between the table we made few assumptions like

* Property to Lease Agreement: Here we assume that one property can have only one lease agreement if it is rented and zero lease agreement if it is active in listing and not yet rented.
* Property to location: Property search is mainly based on the location that’s why we created subtype location from the supertype property. It will have detailed information about the property location and searching location would be easy.
* Admin would be the person who will be managing the users, agents and properties that’s why admin and other entities are connected with one to many relationships
* Multiple User Roles: This statement makes the assumption that the database system would accommodate many user roles, each with a unique set of rights and access levels, such as administrators, agents, customer sellers, and buyers.
* Data Validation: Presuming that the database system would be equipped with features to verify and guarantee the accuracy of the data being input, like identifying legitimate addresses and avoiding duplicate entries.
* Scalability and High Availability: This refers to the expectation that the database system would be built to support numerous users at once and be expandable to meet future expansion, with safeguards in place to guarantee high availability and low downtime.

**2. Identify (list and describe) the Entities and Relationships you derived from your analysis**

1. **User:**

* Attributes: User\_ID(PK), Name, Email, Phone\_Number, Address.

1. **Property:**

* Attributes: Property\_ID(PK), Listing\_Price, Bedrooms, Bathrooms, Square\_Footage, Home\_Type, Property\_Type, Listing\_Status, User\_ID(FK).

1. **Agent:**

* Attributes: Agent\_ID(PK), User\_ID(FK), Property\_ID(FK) Name, Email, Phone\_Number, Agency\_Name.

1. **Review:**

* Attributes: Review\_ID(PK), User\_ID(FK), Property\_ID(FK), Rating, Comment, Review\_Date.

1. **Transaction:**

* Attributes: Transaction\_ID(PK), User\_ID(FK), Property\_ID(FK), Transaction\_Date, Sell\_Price.

1. **Location:**

* Attributes: Location\_ID(PK), Apt\_Number, Street, City, State, Country, Property\_ID(FK).

1. **Mortgage:**

Attributes: Mortgage\_ID(PK), Mortgage\_amount, Mortgage\_provider, Date, Amount, Tenure, User\_ID(FK), Property\_ID(FK).

1. **Amenities:**

* Attributes: Amenities\_ID(PK), Property\_ID(FK), Amenities\_Name, Description.

1. **Lease Agreement:**

* Attributes: Lease\_Agreement\_ID(PK), User\_ID(FK), Property\_ID(FK), Lease\_Start\_Date, Lease\_End\_Date, Terms\_and\_Conditions.

1. **Admin:**

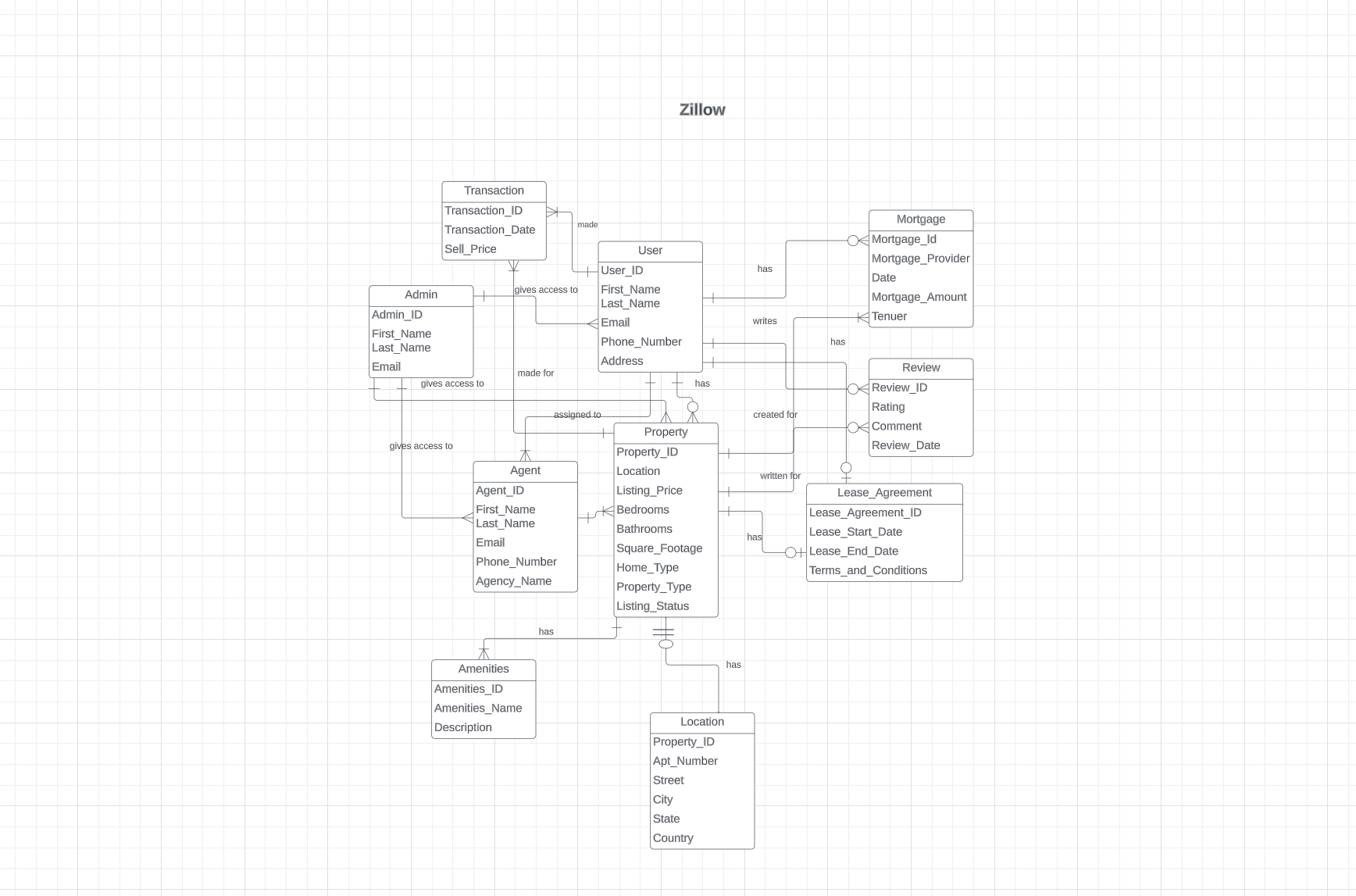
* Attributes: Admin\_ID(PK), Name, Email, Agent\_ID(FK), User\_ID(FK), Property\_ID(FK),

**Relationships of entities:**

1. Admin to Agent one to many
2. Admin to Property one to many
3. Admin to User one to many
4. User to Agent one to many
5. User to Property one to many
6. User to Review one to many
7. User to Lease Agreement one to one
8. User to Transaction one to many
9. User to Mortgage one to many
10. Property to Transaction one to many
11. Property to Mortgage one to many
12. Property to Amenities one to many
13. Property to Review one to many
14. Property to Lease agreement one to one

**3. Design a Fist Cut “Top Down” ERD (do not normalize) based on the initial set of information that has been provided. Show the basic entity relationships without concern for M:N relationship. Use Crowsfoot Notation. Remember the ERD is the initial output of your design process to show to the business and validate your understanding of its needs.**

**ERD diagram**

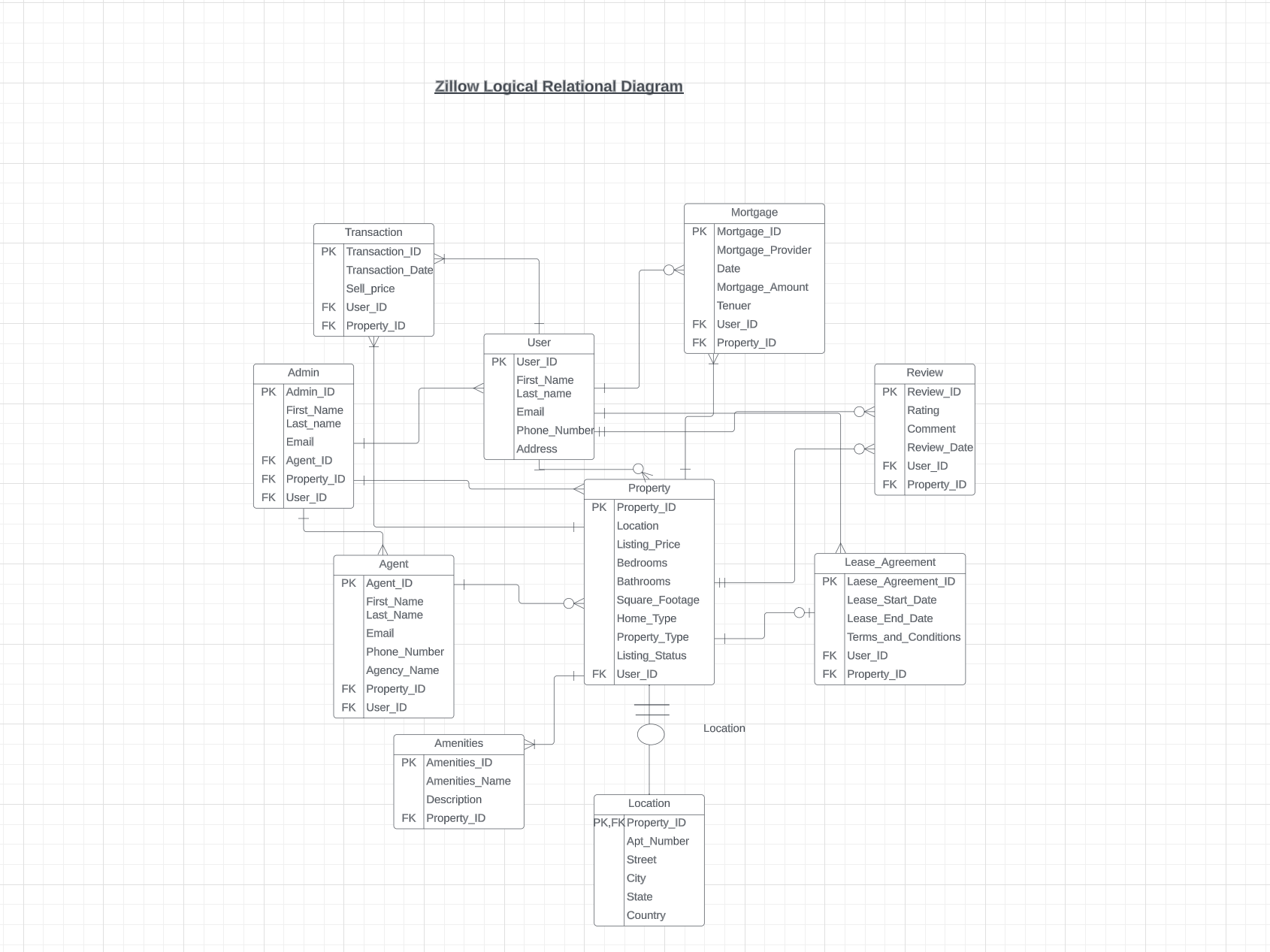


**4. Design a Logical Relational Model and Assign Primary and Foreign Keys Using Crowsfoot Notation. You must Resolve M:N Relationships and Note Cardinality in your model. Also:**

**a) Identify the Types of Entities in Your Model. For example: Composite, Super-type and Sub-type. Strong vs. Weak. Recursive.**

**b) Determine Entity Integrity and Referential Integrity (see Entity tables below).**

**Logical Relational Diagram**



**5. Describe Your Approach to Achieving a Fully Normalized Logical Model (with assumptions and tradeoffs that may exist). Are there redundant tables and 1:1 relationships that can be eliminated? Describe partial and transitive dependencies you encountered as you took your model from 1NF to 2NF to 3NF.**

**Answer:**

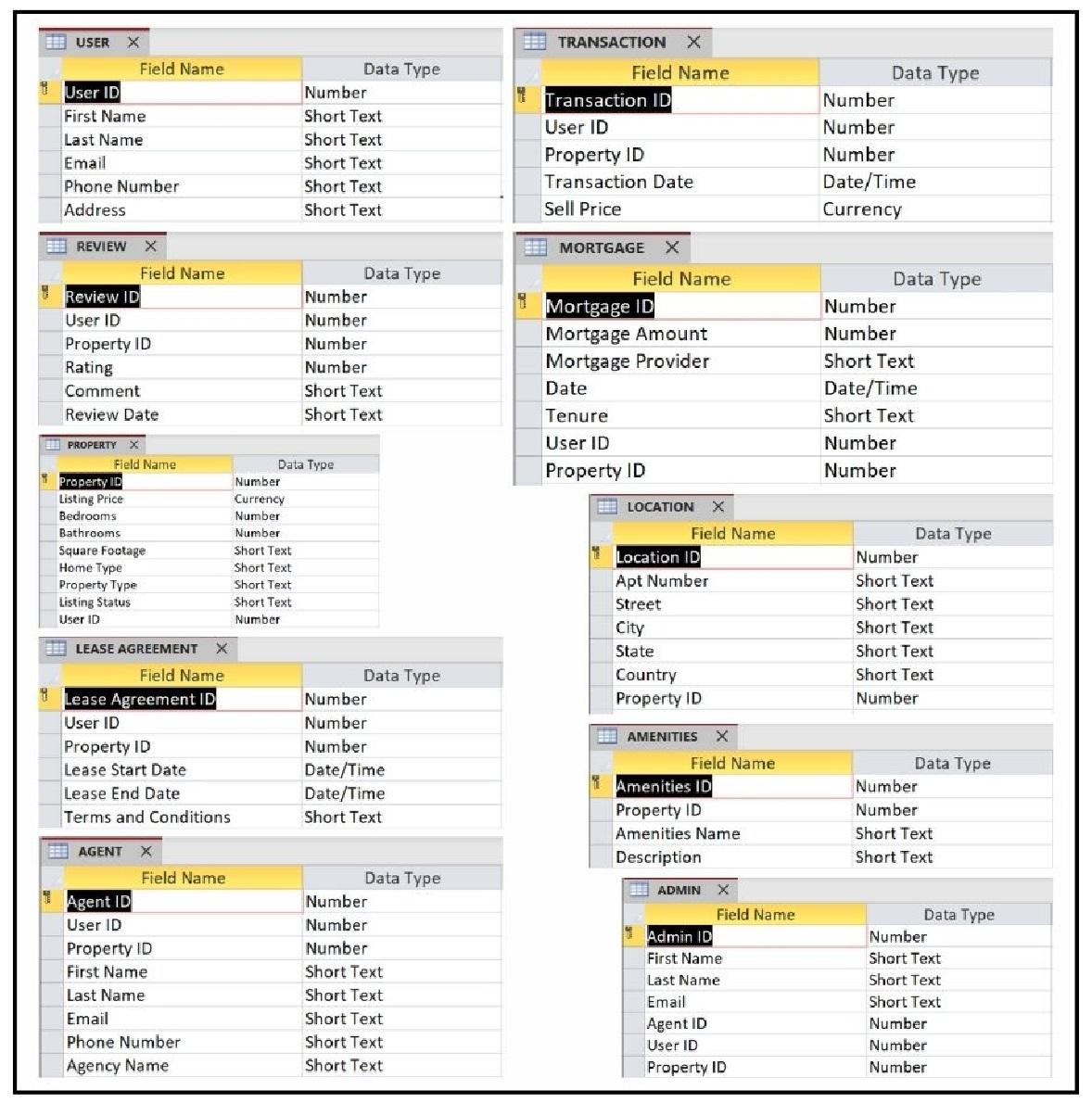
For the Normalization we got mainly the property data and we found that agent and user data is also present property data in that case we performed following operations

1NF: To achieve the first normal form we made sure there is no duplicate data and data is atomic. To clean the atomic data we separated the attributes like name to first\_name & last\_name. Address to apt\_no, street, city, state, country.

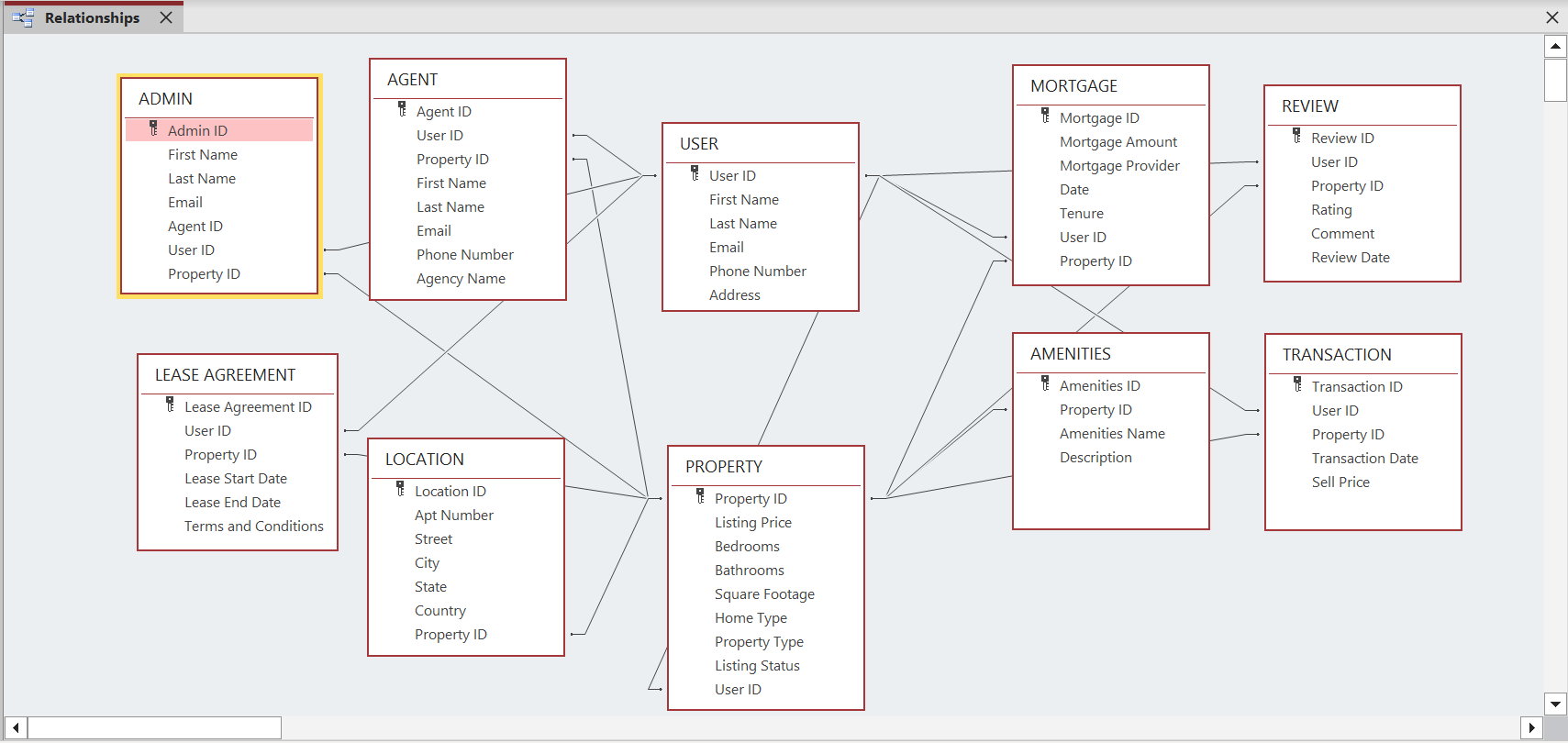
2NF: To achieve the second normal form we made sure it is in first normal form and separated the table to remove the partial dependency like separating User table and Agent table from Property table

3NF: To achieve the third normal form we made sure that data is in 1nf and 2nf and separated the transitive dependencies from each table. Transitive dependencies wear Amenities, Lease\_Agreement, Review, Mortgage.

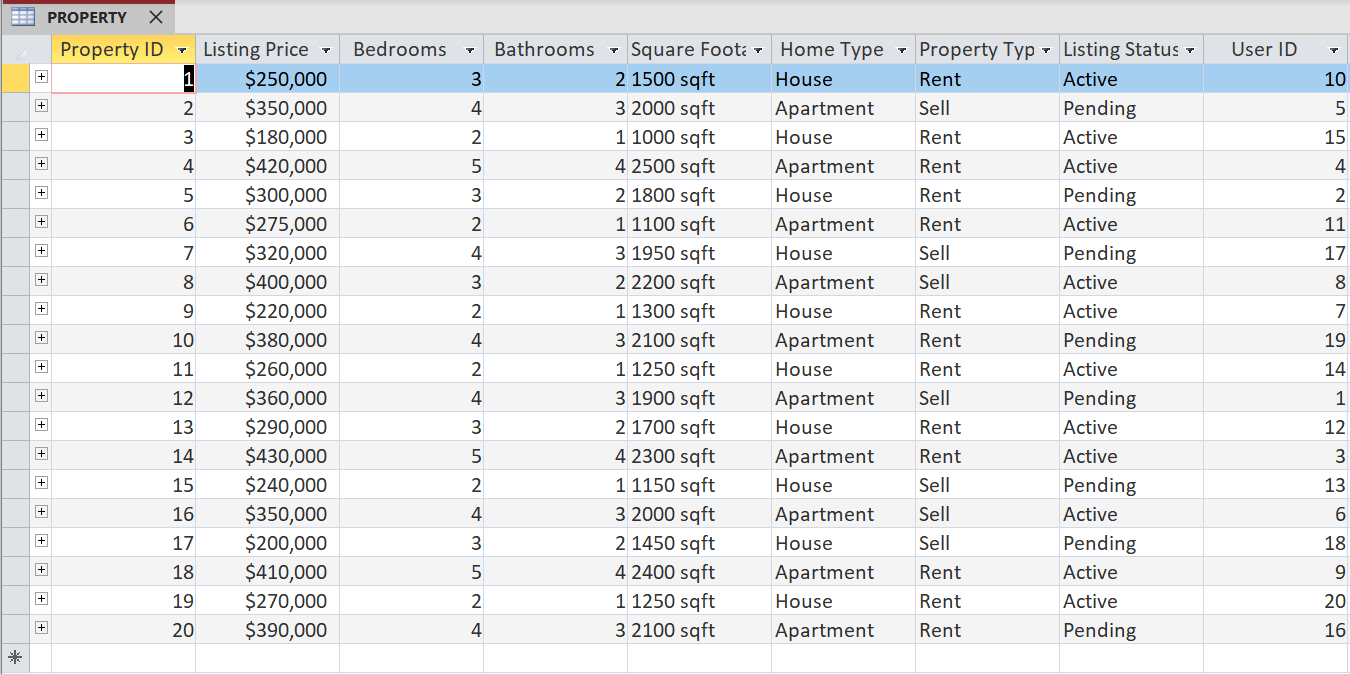
**6. Create a Data Dictionary. Create data dictionary for all the tables**

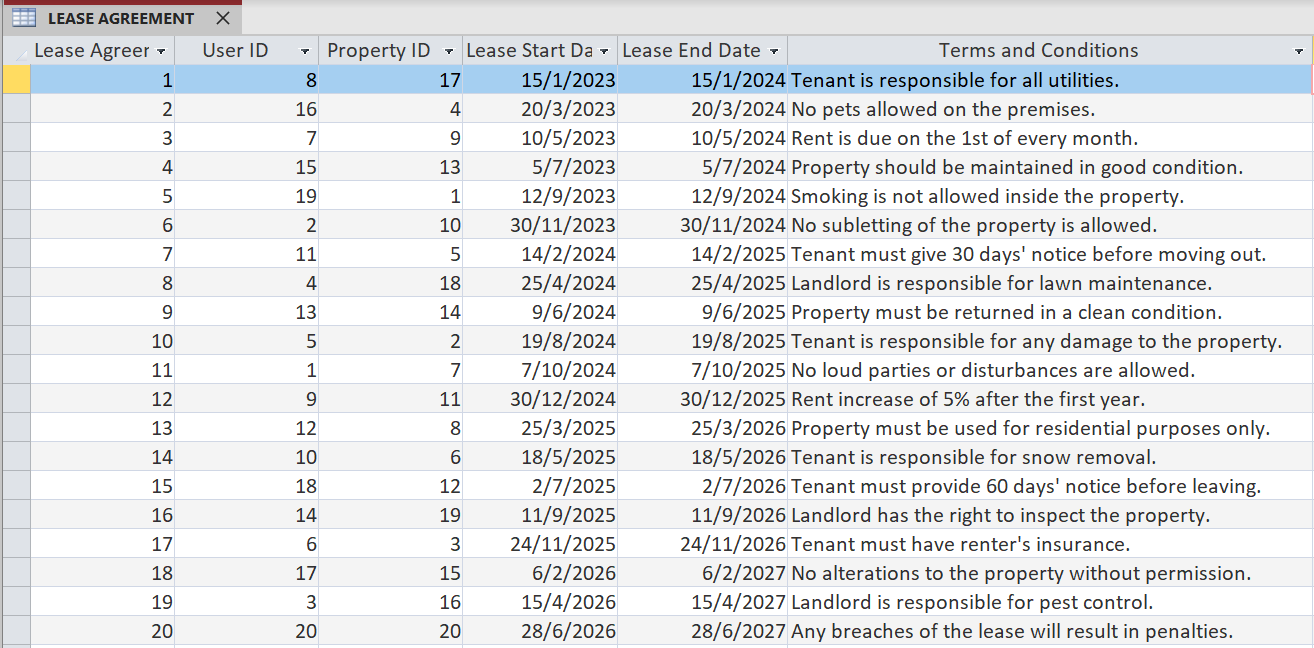


**7. Implement your Relational Logical Model in MS Access.**

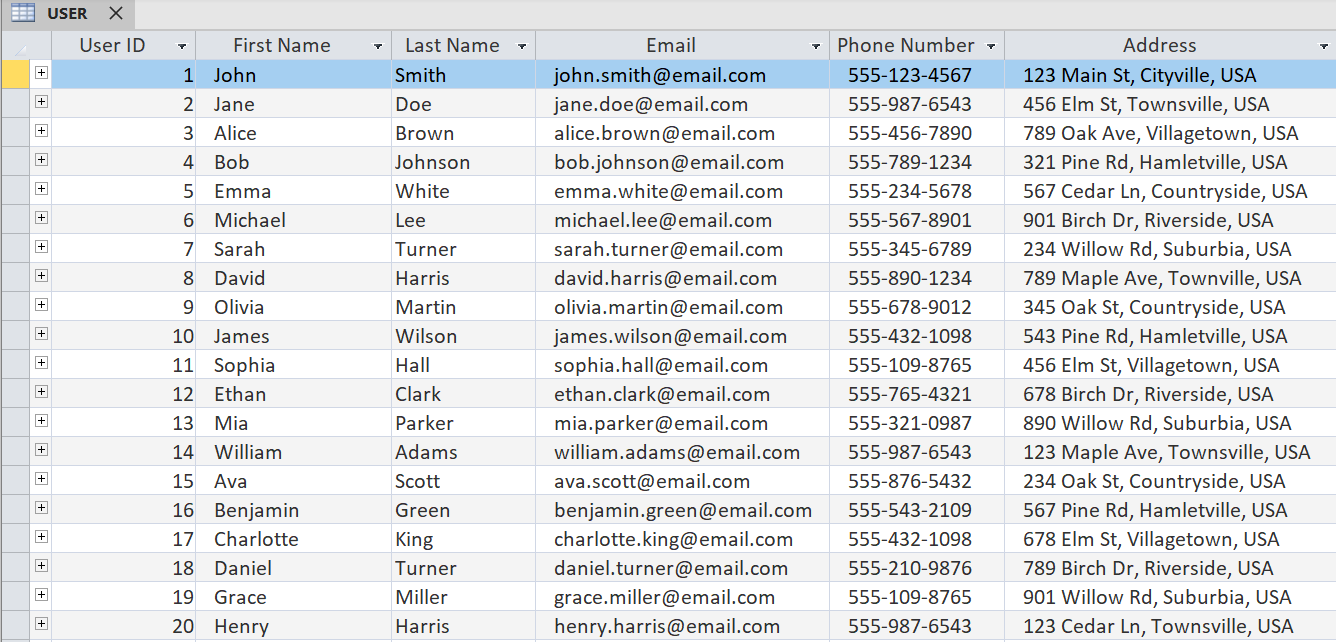
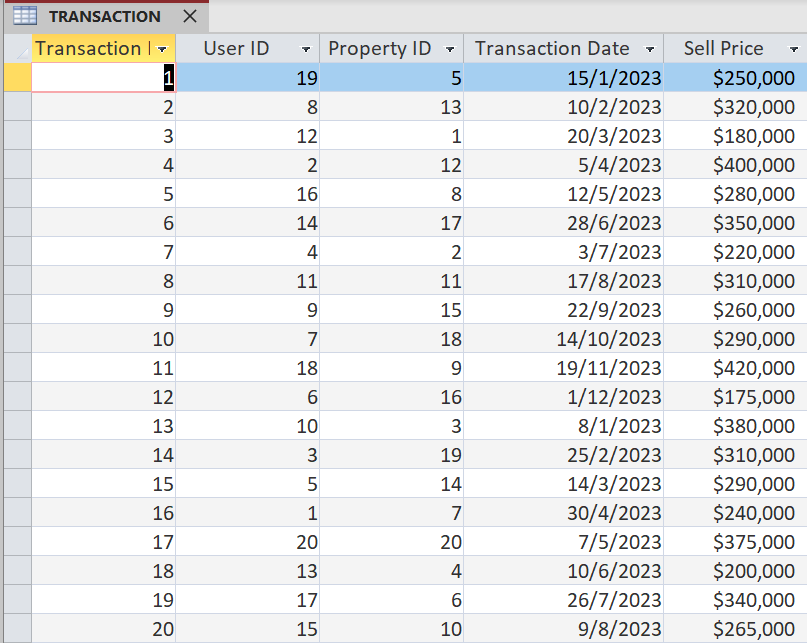
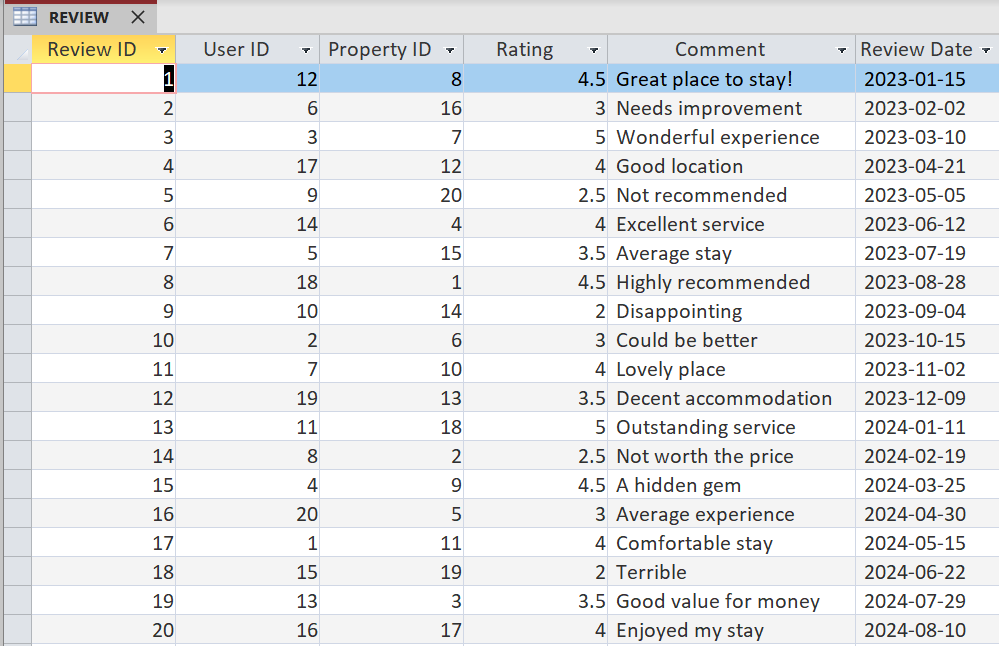
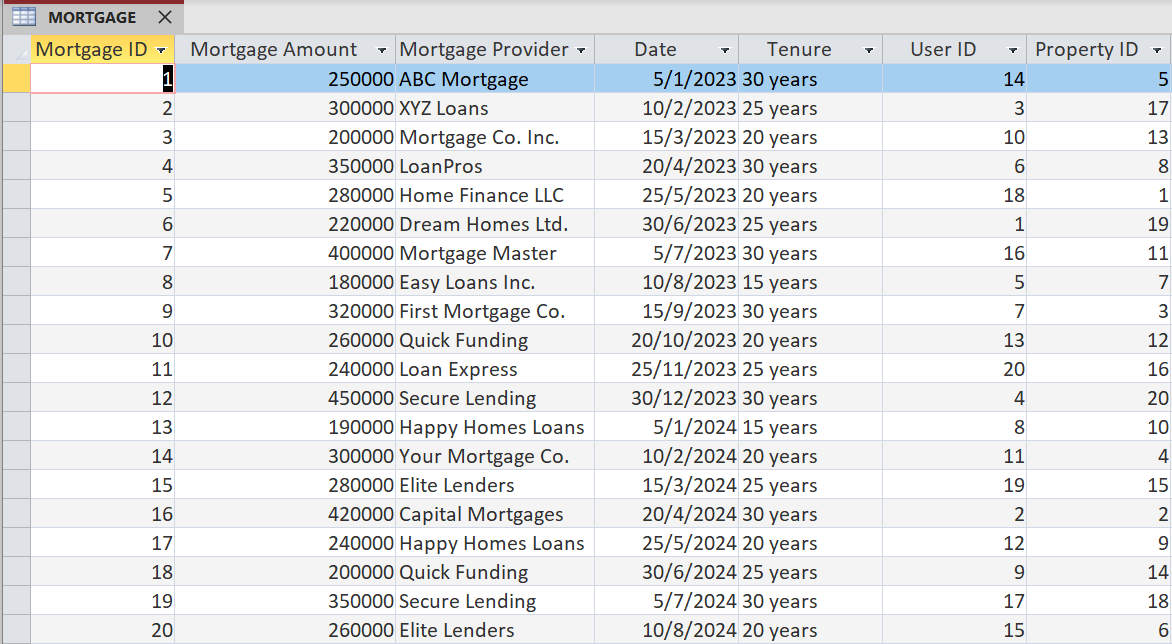


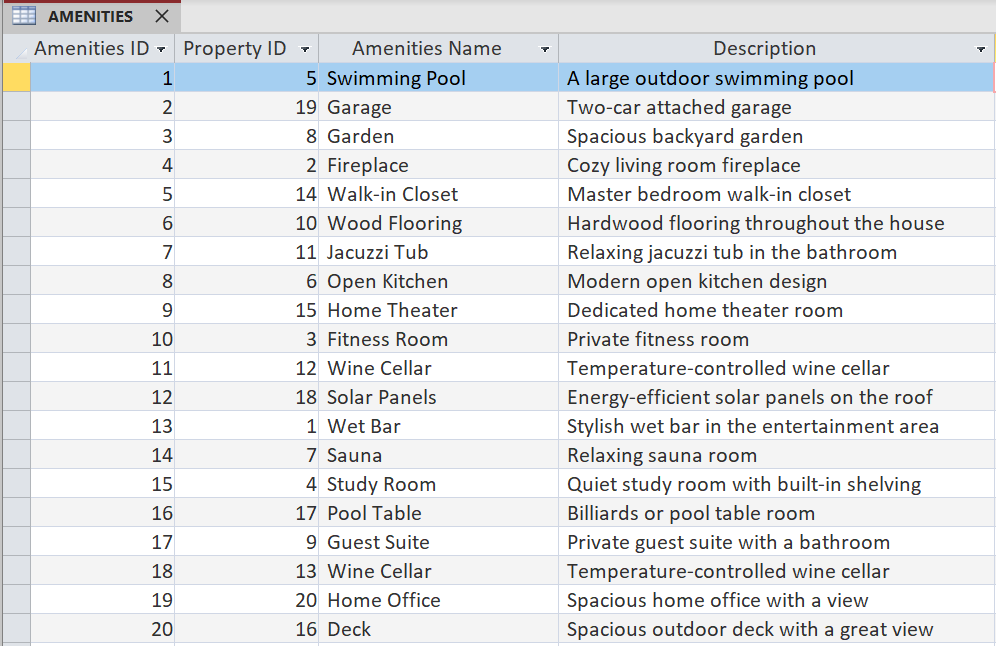
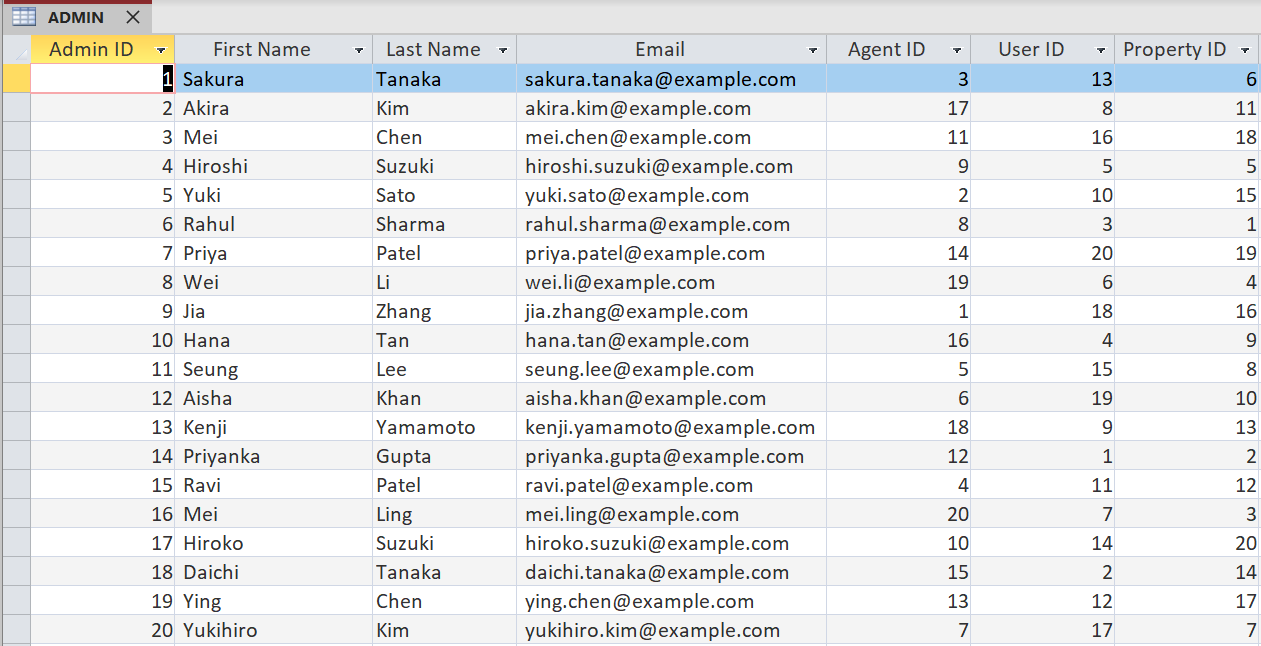
**8. Enter Sample Data in Each Table with at least 20 rows of data per table.**







A screenshot of a computer

Description automatically generated

**9. Select 10 Business/User Views and Implement Using Query and Reports (Produce Samples). Select 2-3 transaction inputs to populate your logical model tables and test against your physical model.**

1. Show User with specific property and mortgage details where tenure is more than 15 years

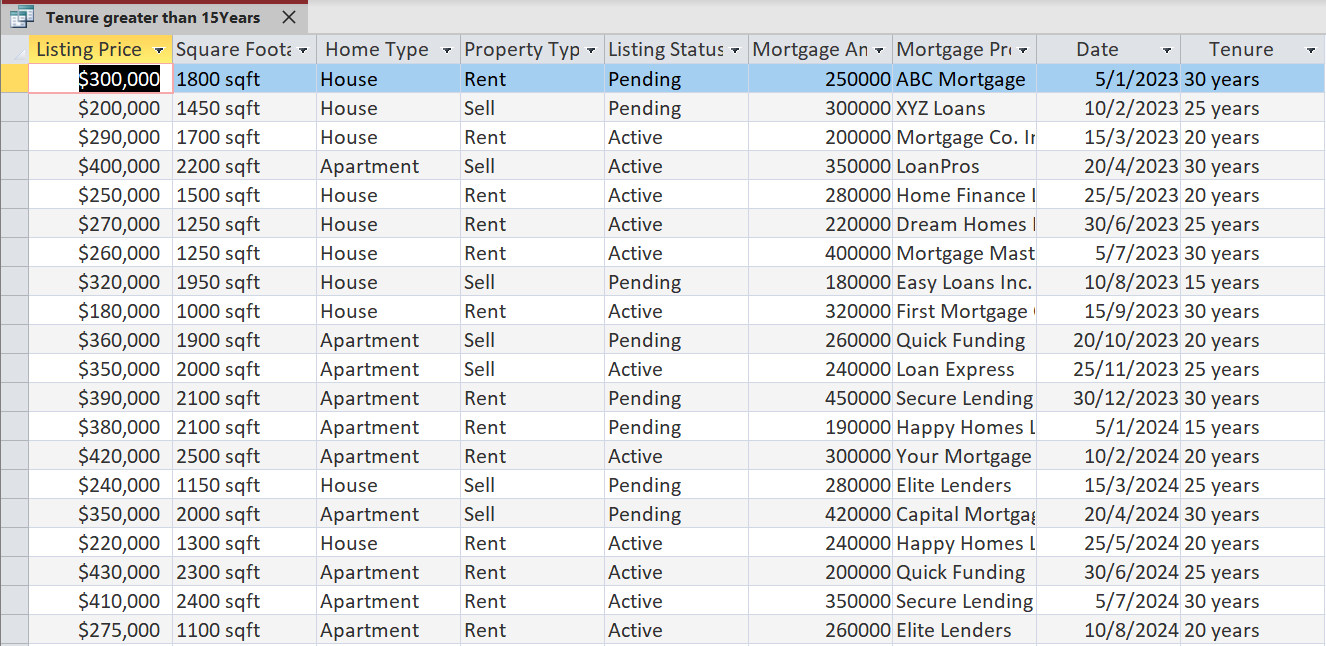
**Query:**

SELECT PROPERTY.[Listing Price], PROPERTY.[Square Footage], PROPERTY.[Home Type], PROPERTY.[Property Type], PROPERTY.[Listing Status], MORTGAGE.[Mortgage Amount], MORTGAGE.[Mortgage Provider], MORTGAGE.Date, MORTGAGE.Tenure

FROM PROPERTY INNER JOIN MORTGAGE ON PROPERTY.[Property ID] = MORTGAGE.[Property ID]

WHERE MORTGAGE.[Tenure] >= "15 years";

**Output:**



1. Show lease agreement details for user name “Emma White”

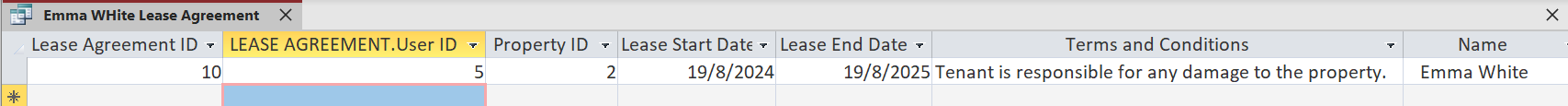
**Query:**

SELECT [LEASE AGREEMENT].\*, USER.Name

FROM [USER] INNER JOIN [LEASE AGREEMENT] ON USER.[User ID] = [LEASE AGREEMENT].[User ID]

WHERE (((USER.Name)="Emma White"));

**Output:**



1. Show Home Type, Square Footage, Amenities Name, Description, CIty and State where Listing Price is greater than $300000.

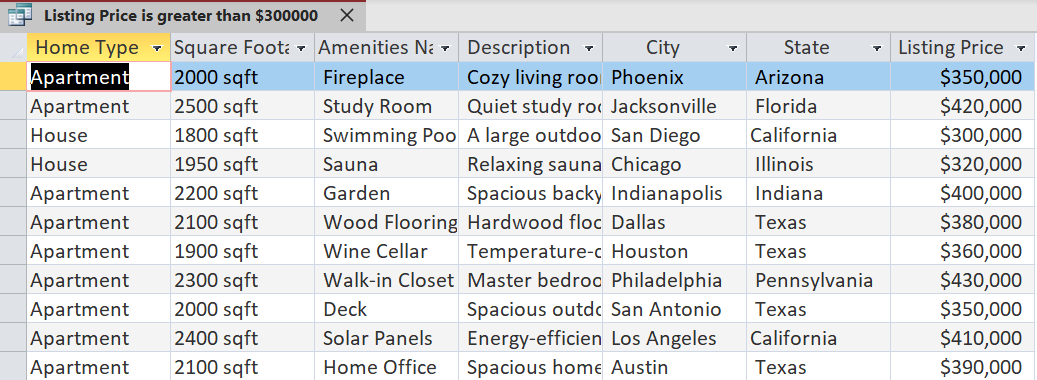
**Query:**

SELECT PROPERTY.[Home Type], PROPERTY.[Square Footage], AMENITIES.[Amenities Name], AMENITIES.Description, LOCATION.City, LOCATION.State, PROPERTY.[Listing Price]

FROM (PROPERTY INNER JOIN LOCATION ON PROPERTY.[Property ID] = LOCATION.[Property ID]) INNER JOIN AMENITIES ON PROPERTY.[Property ID] = AMENITIES.[Property ID]

WHERE (((PROPERTY.[Listing Price]) >=300000));

**Output:**



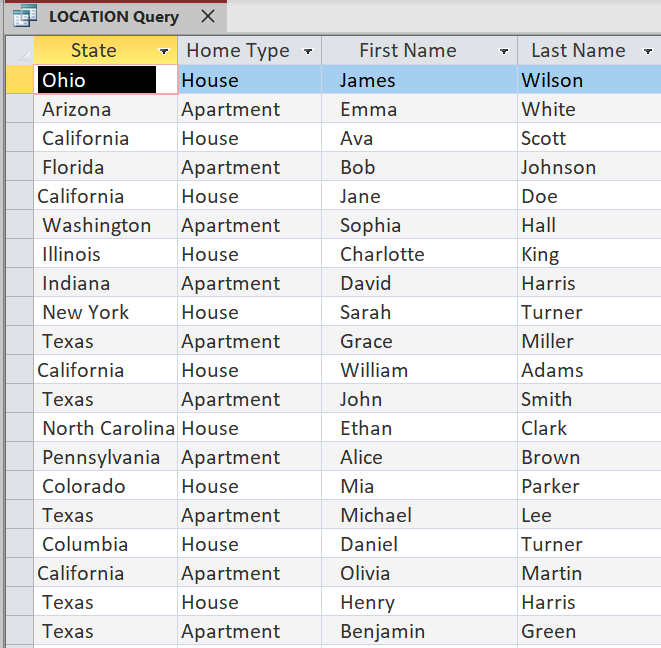
1. Show State, Home Type and Name of the person who owns that Property.

**Query:**

SELECT LOCATION.State, PROPERTY.[Home Type], USER.[First Name], USER.[Last Name]

FROM ([USER] INNER JOIN PROPERTY ON USER.[User ID] = PROPERTY.[User ID]) INNER JOIN LOCATION ON PROPERTY.[Property ID] = LOCATION.[Property ID];

**Output:**



1. Show Home Type, State and Count of Home Type.

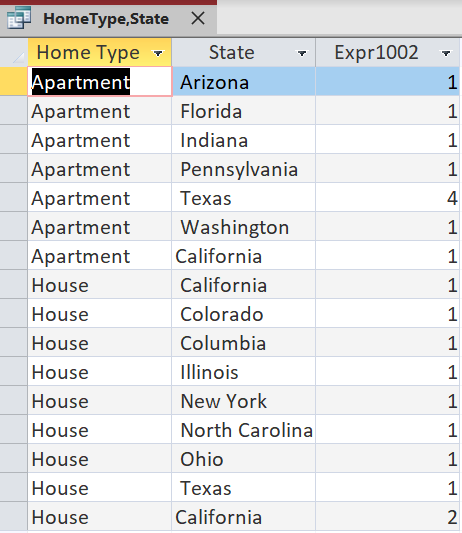
**Query:**

SELECT PROPERTY.[Home Type], LOCATION.[State], count(\*)

FROM PROPERTY INNER JOIN LOCATION ON PROPERTY.[Property ID] = LOCATION.[Property ID]

GROUP BY PROPERTY.[Home Type], LOCATION.[State];

**Output:**



1. Show the count of different types of Homes.

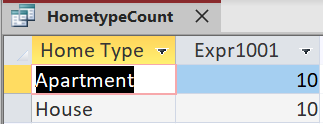
**Query:**

SELECT PROPERTY.[Home Type], COUNT (\*)

FROM PROPERTY

GROUP BY PROPERTY.[Home Type];

**Output:**



1. Show all details of Property where Property ID is less than 8

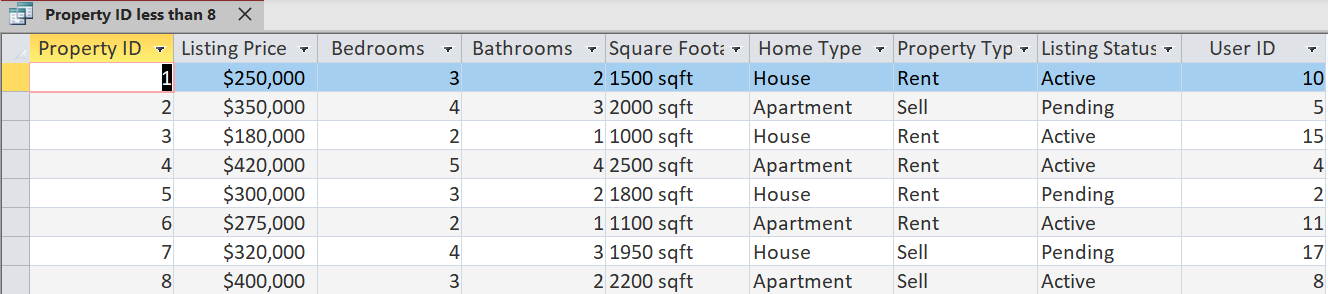
**Query:**

SELECT \*

FROM PROPERTY

WHERE PROPERTY.[Property ID] <= 8;

**Output:**



1. Show average sell price of Property Type.

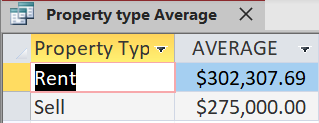
**Query:**

SELECT PROPERTY.[Property Type], AVG(TRANSACTION.[SELL PRICE]) AS AVERAGE

FROM PROPERTY INNER JOIN [TRANSACTION] ON PROPERTY.[Property ID] = TRANSACTION.[Property ID]

GROUP BY PROPERTY.[Property Type];

**Output:**



1. Show Property Type, Home Type, Rating, Amenities Name and state where rating is greater than 4.5 and state is california.

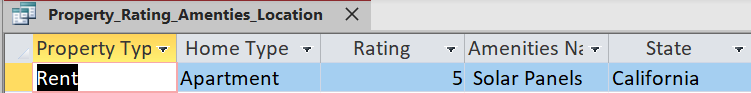
**Query:**

SELECT PROPERTY.[Property Type], PROPERTY.[Home Type], REVIEW.Rating, AMENITIES.[Amenities Name], LOCATION.State

FROM ((PROPERTY INNER JOIN AMENITIES ON PROPERTY.[Property ID] = AMENITIES.[Property ID]) INNER JOIN LOCATION ON PROPERTY.[Property ID] = LOCATION.[Property ID]) INNER JOIN REVIEW ON PROPERTY.[Property ID] = REVIEW.[Property ID]

WHERE (((REVIEW.Rating)>4.5) AND ((LOCATION.State)="California"));

**Output:**



1. Show average rating of a property based on its reviews

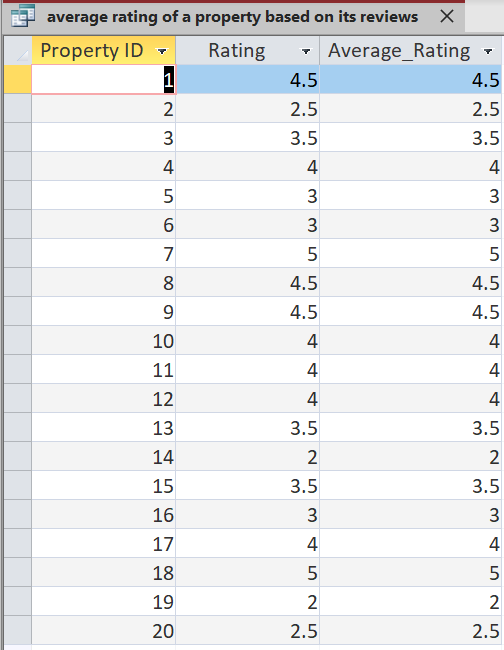
**Query:**

SELECT PROPERTY.[Property ID], REVIEW.Rating, Avg(REVIEW.Rating) AS Average\_Rating

FROM PROPERTY LEFT JOIN REVIEW ON PROPERTY.[Property ID] = REVIEW.[Property ID]

GROUP BY PROPERTY.[Property ID], REVIEW.Rating;

**Output:**



**10. Summary Conclusions Based on Your Design: How Confident are you that your design will meet the originally stated requirements. Describe how you feel about the life cycle process you have undertaken and whether you are comfortable using this approach in designing other database projects.**

**Conclusion:**

The business problem we came across has been solved using the database system we have designed. We are confident that the business requirements we took into consideration we achieved those. We followed every step of database system design starting from understanding the requirement, identifying the entities creating entity relationship diagrams by taking business rules under consideration. We created a logical relational diagram and assigned primary key and foreign key wherever required. We also normalized the overall data and remove duplicates, partial dependencies and transitive dependencies. We moved forward with creating a physical database using the Access tool. We performed multiple operations under access. We created the data dictionary. We tested our result from business and user point of view and we ran a few queries on the database.

All the above steps helped us learn more about how the database system looks and how every step is important to achieve the desired result. We are very much comfortable with solving other problems and designing multiple database systems. It is definitely a value adding project we worked upon. We were excited to work on this project and this project helped us learn collaboration and appreciate team work. We welcomed suggestions from each team member. We brainstormed more about it and came to the conclusion.