APARTMENT LISTING SYSTEM

PRESENTED BY :

GROUP 3

TEAM MEMBERS:

Aditi Ashutosh Deodhar . Drashti Bhingradiya

Kavya Rachana Malluvalasa . Vineeth Reddy Singireddy

PROFESSOR:

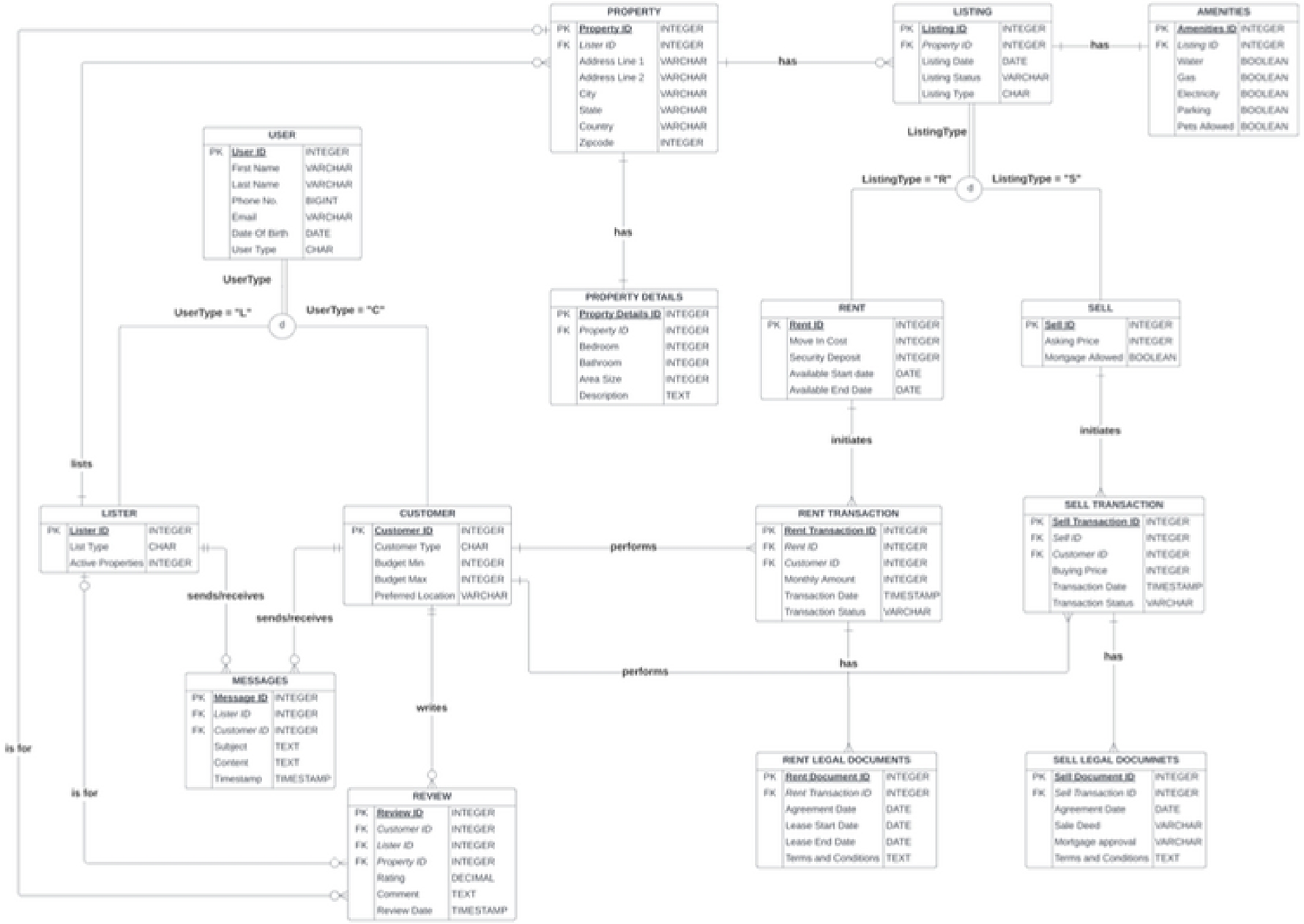
Manuel Montrond

OVERVIEW

- The Apartment Listings System serves as a multifaceted tool designed to simplify property-related tasks and enhance transparency and efficiency within the real estate market.

- Main Features:
  - User Profiles and Reviews
  - Listing Management
  - Search facility

ER DIAGRAM


PROPERTY
PK Property ID INTEGER
FK Lister ID INTEGER
Address Line 1 VARCHAR
Address Line 2 VARCHAR
City VARCHAR
State VARCHAR
Country VARCHAR
Zipcode INTEGER
has
LISTING
PK Listing ID INTEGER
FK Property ID INTEGER
Listing Date DATE
Listing Status VARCHAR
Listing Type CHAR
has
AMENITIES
PK Amenities ID INTEGER
FK Listing ID INTEGER
Water BOOLEAN
Gas BOOLEAN
Electricity BOOLEAN
Parking BOOLEAN
Pets Allowed BOOLEAN
ListingType
ListingType = "R"
d
ListingType = "S"
USER
PK User ID INTEGER
First Name VARCHAR
Last Name VARCHAR
Phone No. BIGINT
Email VARCHAR
Date Of Birth DATE
User Type CHAR
UserType
UserType = "L"
d
UserType = "C"
has
PROPERTY DETAILS
PK Property Details ID INTEGER
FK Property ID INTEGER
Bedroom INTEGER
Bathroom INTEGER
Area Size INTEGER
Description TEXT
RENT
PK Rent ID INTEGER
Move In Cost INTEGER
Security Deposit INTEGER
Available Start date DATE
Available End Date DATE
initiates
SELL
PK Sell ID INTEGER
Asking Price INTEGER
Mortgage Allowed BOOLEAN
initiates
lists
LISTER
PK Lister ID INTEGER
List Type CHAR
Active Properties INTEGER
CUSTOMER
PK Customer ID INTEGER
Customer Type CHAR
Budget Min INTEGER
Budget Max INTEGER
Preferred Location VARCHAR
performs
RENT TRANSACTION
PK Rent Transaction ID INTEGER
FK Rent ID INTEGER
FK Customer ID INTEGER
Monthly Amount INTEGER
Transaction Date TIMESTAMP
Transaction Status VARCHAR
SELL TRANSACTION
PK Sell Transaction ID INTEGER
FK Sell ID INTEGER
FK Customer ID INTEGER
Buying Price INTEGER
Transaction Date TIMESTAMP
Transaction Status VARCHAR
sends/receives
sends/receives
MESSAGES
PK Message ID INTEGER
FK Lister ID INTEGER
FK Customer ID INTEGER
Subject TEXT
Content TEXT
Timestamp TIMESTAMP
writes
performs
has
has
is for
is for
REVIEW
PK Review ID INTEGER
FK Customer ID INTEGER
FK Lister ID INTEGER
FK Property ID INTEGER
Rating DECIMAL
Comment TEXT
Review Date TIMESTAMP
RENT LEGAL DOCUMENTS
PK Rent Document ID INTEGER
FK Rent Transaction ID INTEGER
Agreement Date DATE
Lease Start Date DATE
Lease End Date DATE
Terms and Conditions TEXT
SELL LEGAL DOCUMNETS
PK Sell Document ID INTEGER
FK Sell Transaction ID INTEGER
Agreement Date DATE
Sale Deed VARCHAR
Mortgage approval VARCHAR
Terms and Conditions TEXT

DDL OBJECTS

STORED PROCEDURE

- **GetListingsByDaysAvailable**
- Retrieves available listings for the number of days provided.
- Parameters : **@maxDays**
- Output : Filter the results to include only listings with a number of days available less than or equal to the value specified by the **@maxDays** parameter.

```sql
CREATE OR ALTER PROCEDURE GetListingsByDaysAvailable
    @maxDays INT
AS
BEGIN
    DECLARE @currentDate DATE = GETDATE();

    SELECT *,
           DATEDIFF(DAY, Listing_Date, @currentDate) AS Days_Available
    FROM Listing
    WHERE DATEDIFF(DAY, Listing_Date, @currentDate) <= @maxDays
        AND DATEDIFF(DAY, Listing_Date, @currentDate) > 0;
END;

-- Demonstartion of stored procedure GetListingsByDaysAvailable
EXEC GetListingsByDaysAvailable 100;
```

Results Messages

	Listing_ID	Property_ID	Listing_Date	Listing_Status	Listing_Type	Days_Available
1	1	1	2024-03-17	Active	R	35
2	2	2	2024-03-17	Active	S	35
3	3	3	2024-02-17	Active	R	64
4	4	4	2024-04-19	Active	S	2
5	6	6	2024-01-17	Active	S	95
6	7	7	2024-03-27	Active	R	25
7	9	9	2024-02-17	Active	R	64
8	12	12	2024-03-17	Active	S	35
9	40	41	2024-03-01	Active	R	51
10	41	42	2024-02-15	Active	R	66

USER DEFINED FUNCTION

- **CalculateNeighborhoodScore**
- computes the average rating and total no. of reviews for properties within a given zip code.
- Parameters : **@Zipcode**
- Output : Filter the results to include only listings within the zip code with ratings and reviews.

```sql
CREATE OR ALTER FUNCTION CalculateNeighborhoodScore
(
    @Zipcode INTEGER
)
RETURNS TABLE
AS RETURN
(
    SELECT
        P.Zipcode, ROUND(AVG(R.Rating),2) AS AvgRating, COUNT(R.Review_ID) AS NumReviews
    FROM Property P
    INNER JOIN Review R ON P.Property_ID = R.Property_ID
    WHERE P.Zipcode = @Zipcode
    GROUP BY P.Zipcode
);

--- to test the function
SELECT
    DISTINCT p.Zipcode,
    p.[City],
    p.[State],
    d.AvgRating,
    d.NumReviews,
    COUNT(*) AS NoOfProperties
FROM Property p
CROSS APPLY dbo.CalculateNeighborhoodScore(p.Zipcode) d
GROUP BY p.Zipcode, p.[City], p.[State], d.AvgRating, d.NumReviews;
```

Results Messages

	Zipcode	City	State	AvgRating	NumReviews	NoOfProperties
1	2108	Boston	MA	2.000000	3	2
2	10001	New York	NY	3.200000	5	3
3	19103	Philadelphia	PA	2.500000	2	2
4	30303	Atlanta	GA	2.000000	1	1
5	33101	Miami	FL	2.500000	2	2
6	60611	Chicago	IL	4.000000	2	2
7	77002	Houston	TX	3.500000	2	2
8	90001	Los Angeles	CA	4.000000	2	2
9	90002	Los Angeles	CA	3.000000	2	1

ENCRYPTION

- **Encryption setup for User's Phone no.**
- A security framework within SQL Server involving a master key, a certificate, and a symmetric key to encrypt sensitive user phone no., using AES algorithm.

```sql
-- Creating a master key for column encryption
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'ApartmentFinder@123';

-- Creating a certificate for column encryption
CREATE CERTIFICATE MyCertificate
  WITH SUBJECT = 'My Column Encryption Certificate';

-- Creating a symmetric key for column encryption
CREATE SYMMETRIC KEY UserDataSymmetricKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE MyCertificate;

-- Opening the symmetric key for decryption
OPEN SYMMETRIC KEY UserDataSymmetricKey
DECRYPTION BY CERTIFICATE MyCertificate

-- Creating a trigger to encrypt the Phone_No column when an insert or update is being performed
CREATE OR ALTER TRIGGER EncryptPhoneNo
ON [User]
AFTER INSERT, UPDATE
AS
BEGIN
    -- Updating the Encrypted_Phone_No only when the Phone_No is given
    UPDATE u
    SET u.Encrypted_Phone_No =
        CASE WHEN i.Phone_No IS NOT NULL
        THEN ENCRYPTBYKEY(KEY_GUID('UserDataSymmetricKey'), CONVERT(VARBINARY(MAX), i.Phone_No))
        ELSE NULL
            END
    FROM [User] u
    INNER JOIN inserted i ON u.User_ID = i.User_ID;
END;
```

Results Messages

	User_ID	First_Name	Last_Name	Email	Date_Of_Birth	Phone_no	User_Type
1	1	Gabriel	Parker	gabriel.parker@example.com	1984-09-30	0x006A9EAB0AE3BC4A8253F91DC6F948370200000073E0875F5C…	C
2	2	Natalie	Gonzalez	natalie.gonzalez@example.com	1997-12-12	0x006A9EAB0AE3BC4A8253F91DC6F948370200000014C5ABC17F…	L
3	3	Henry	Martinez	henry.martinez@example.com	1986-03-25	0x006A9EAB0AE3BC4A8253F91DC6F948370200000008B4BF3FB67…	C
4	4	Ella	Cooper	ella.cooper@example.com	1990-07-08	0x006A9EAB0AE3BC4A8253F91DC6F948370200000028A68265A7…	L

TRIGGER

- **UpdateActiveProperties**
- An After Insert, Delete on the Property table.
- automatically updates the Active_Properties column in the Lister table whenever a new property is inserted or deleted in the Property table by that Lister_ID.

```sql
CREATE OR ALTER TRIGGER UpdateActiveProperties
ON Property
AFTER INSERT, DELETE
AS
BEGIN
    -- Update Active_Properties for each affected Lister due to INSERT operation
    UPDATE L
    SET Active_Properties = (
        SELECT COUNT(*)
        FROM Property P
        WHERE P.Lister_ID = L.Lister_ID
    )
    FROM Lister L
    INNER JOIN inserted I ON L.Lister_ID = I.Lister_ID;

    -- Update Active_Properties for each affected Lister due to DELETE operation
    UPDATE L
    SET Active_Properties = (
        SELECT COUNT(*)
        FROM Property P
        WHERE P.Lister_ID = L.Lister_ID
    )
    FROM Lister L
    INNER JOIN deleted D ON L.Lister_ID = D.Lister_ID;

END;
```

ges

6:20 PM Started executing query at Line 1

Commands completed successfully.

Total execution time: 00:00:00.030

DASHBOARD

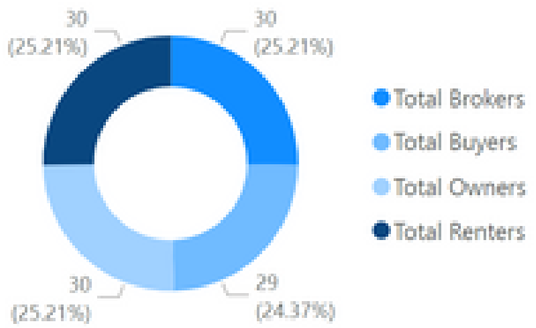
Apartment Listing System

3.00
Avg. Customer Satisfaction Score

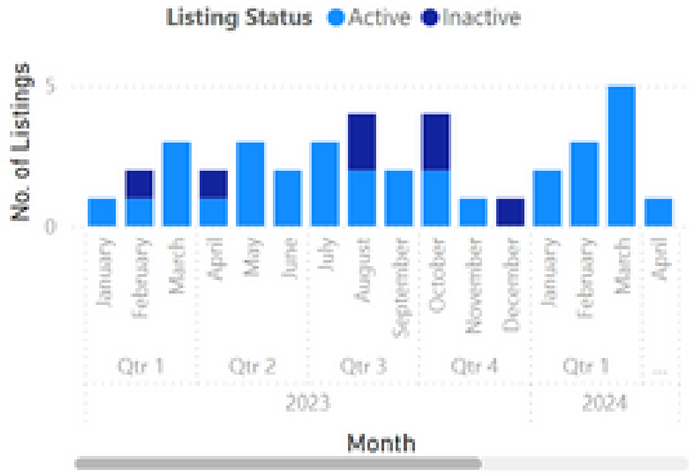
326.50K
Avg. Buying Price

Customer Popular Location Choices

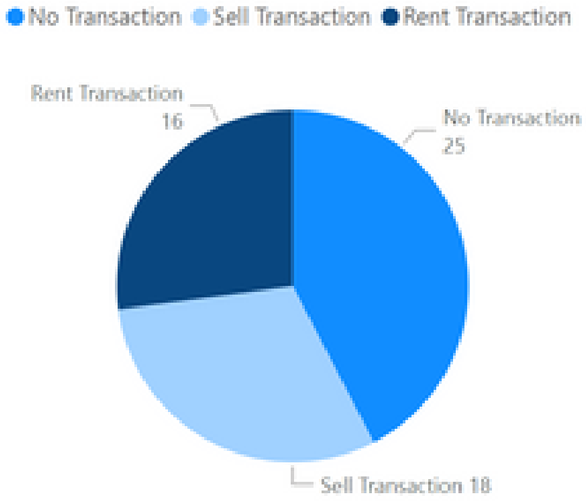
User Segmentation



No. of Listings posted per Quarter



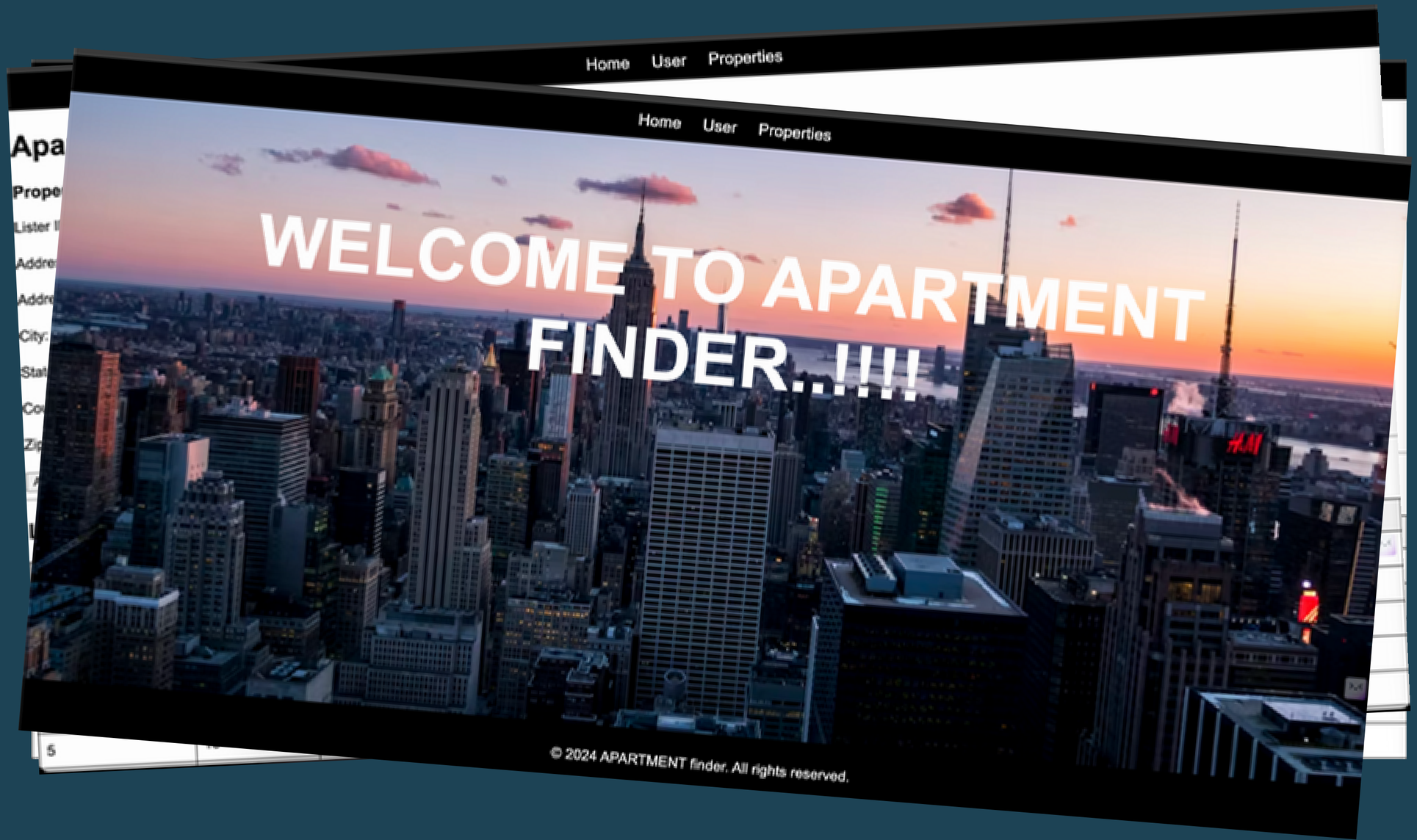
Customer Transaction Summary



Top Rated Listers

Lister	Avg Rating
Alexander Green	5.00
James Jackson	4.00
Michael Brown	4.00
Michael Gonzalez	4.00
William Anderson	4.00
Daniel Taylor	3.50
Matthew Wright	3.50

GUI


Home User Properties
Home User Properties
WELCOME TO APARTMENT FINDER..!!!!
© 2024 APARTMENT finder. All rights reserved.

DEMO

- SQL Stored Procedure
- SQL User Defined Function
- GUI

THANK YOU!