PROJECT DESCRIPTION:

Every year with an influx of about 3000 students and professionals in Buffalo, problems arise in searching for affordable and verified housing options. One must investigate various unverified groups, connect with alums and realtors, and buy expensive subscriptions of realtor websites. With limited options and less scope of verifying the listings people fall prone to frauds or end up paying more.

With this database, we aim to solve the problem and make the rental and leasing process easy for tenants and property owners alike.

OBJECTIVE:

Build a platform where stakeholders and customers can interact with transparency and make the house hunting hassle free.

KEY STAKEHOLDERS AND OTHER PLAYERS:

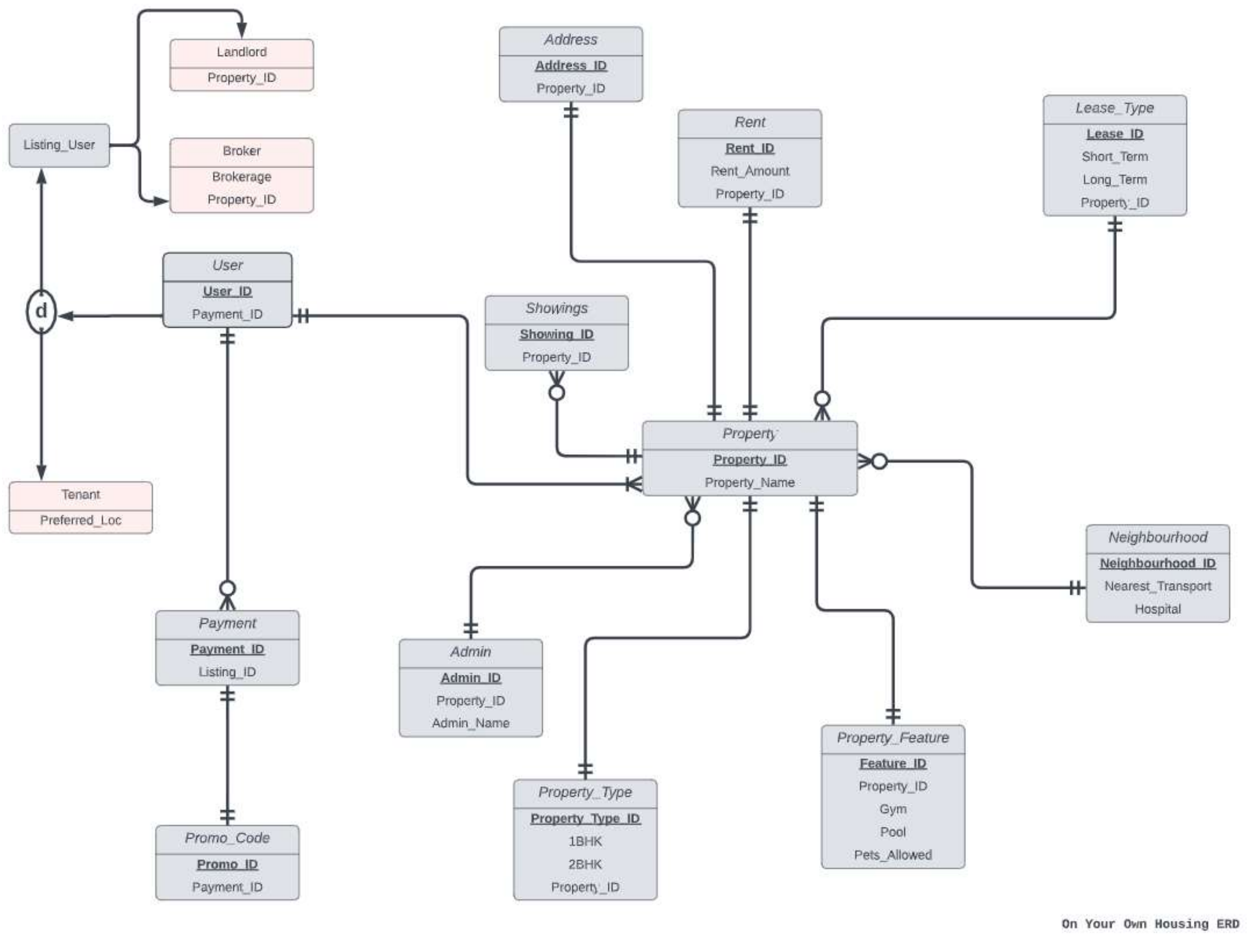
Students, professionals, and property owners

PROJECT SCOPE:

Deliverables- We are projecting a one stop database solution for our key stakeholders with all the possible housing options in Buffalo. The verified data reduces the possibilities of frauds with multiple screening by adding only the registered companies, with valid ID proofs and multiple fraud screening. The database covers the listing of long-term rentals and leases.

Exclusions- Our database is limited to the Buffalo area with a scope of expanding it in future if necessary. The database does not encompass holiday homes and short-term rentals. The database is limited to rental options only and does not cover the sale of properties.

ER DIAGRAM



BUSINESS RULES:

1. A user must be either a listing user or a tenant. A Listing user can be further divided into a broker or a landlord. We can distinctly differentiate between listing user and a tenant using their respective attribute. Landlord and broker will be having a common attribute as property id and are separated entity on brokerage collected by a broker. Tenant can be uniquely identified based on preferred location.

2. Each property must have a listing user; One listing user must have at least one property.

3. Each property must have a property type; Property type must have a property.

4. Each property must have at least one property feature; A property feature must have at least one property.

5. Each Property must have one property feature detail during property listing. For example: A property must have details of heating, plumbing, fire safety, gym, pool etc. Each entry in property feature must have a property associated with it.

6. Each property can have at least one lease type. For example: A property can be leased out for either 3 months, or 6 months, or 12 months or more. Each lease type can be associated with any number of properties.

7. A property must have a neighborhood. A neighborhood may have none or many properties. Neighborhood includes nearest hospitals, schools, bus stops and metro stations.

8. A property must have an address. An address must have a property.

9. A property must have rent listed. Rent must be associated with property. Example: Rent for a property can be paid in one go or in installments.

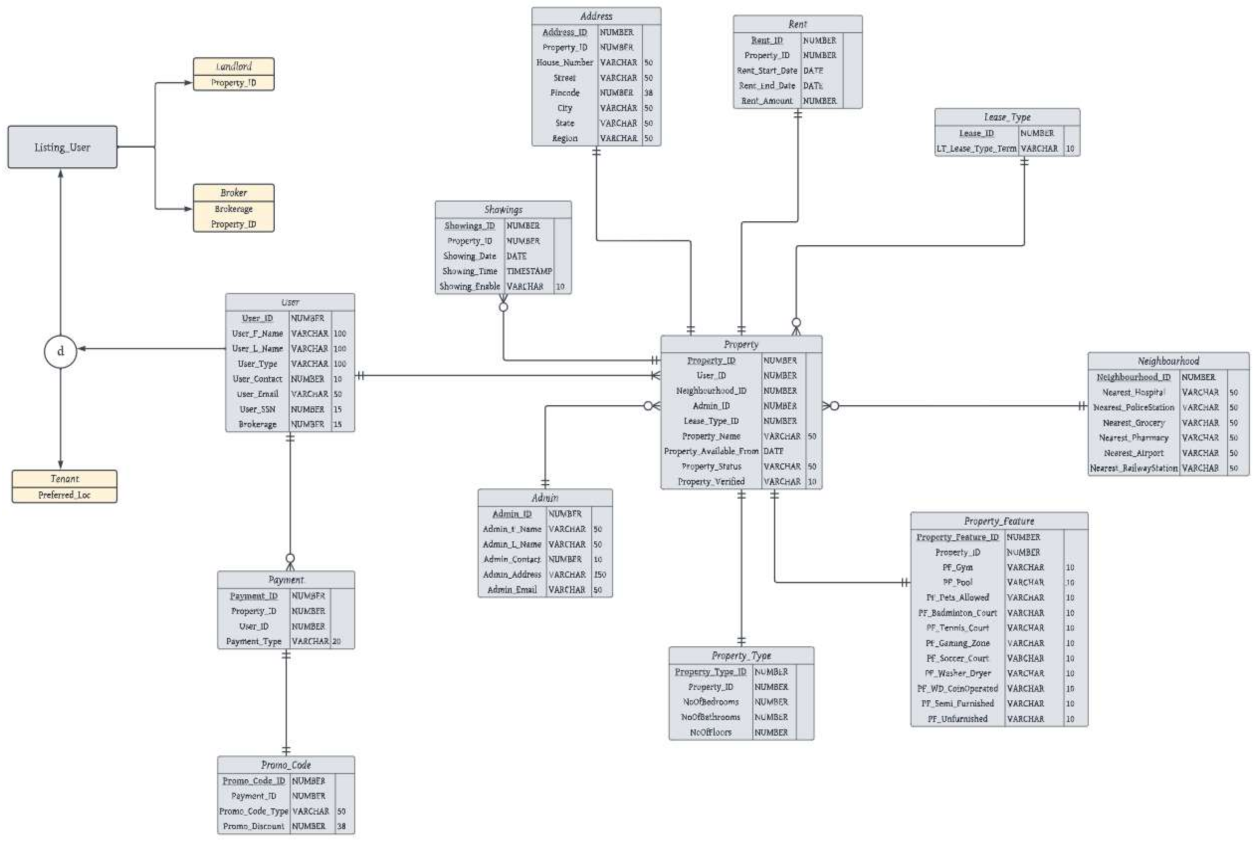
10. Employees at On Your Own housing are admins who can verify the property before they are listed. Each property must have an admin for verification. An admin can have a property to verify or can be a standalone employee.

11. Each listing user/broker must have a payment method for listing their property. A payment method may have none or many users.

12. Payment may change based on the promotional code offered for the user. Each payment must have only one promo code.

13. A property may have none or many showings. A showing must have a property. For ex: A property can be seen multiple times by a user. Once a tenant requests for a showing, a relationship is created between listing user and tenant.

HIGH LEVEL ERD/SCHEMA DESIGN



DDL COMMANDS:

```
CREATE TABLE Users
(
User_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 5000),
User_F_Name VARCHAR(100) NOT NULL,
User_L_Name VARCHAR(100) NOT NULL,
User_Type VARCHAR(100) NOT NULL CHECK(User_Type IN ('Tenant','Landlord','Broker')),
User_Contact Number(10) NOT NULL,
User_Email VARCHAR(50),
User_SSN Number(15) NOT NULL,
Brokerage Number(15),
CONSTRAINT User_PK PRIMARY KEY(User_ID),
CONSTRAINT User_UK UNIQUE (User_Contact,User_Email,User_SSN)
);
```

```
 =================================================================================
CREATE TABLE Admin
(
Admin_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 10),
Admin_F_Name VARCHAR(50) NOT NULL,
Admin_L_Name VARCHAR(50) NOT NULL,
Admin_Contact NUMBER(10) NOT NULL,
Admin_Address VARCHAR(150) NOT NULL,
Admin_Email VARCHAR(50),
CONSTRAINT Admin_PK PRIMARY KEY(Admin_ID),
CONSTRAINT Admin_CONTACT UNIQUE (Admin_Contact, Admin_Email)
);
 =================================================================================

CREATE TABLE Neighbourhood
(
Neighbourhood_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 2000),
Nearest_Hospital Varchar(50),
Nearest_PoliceStation Varchar(50),
Nearest_Grocery Varchar(50),
Nearest_Pharmacy Varchar(50),
Nearest_Airport Varchar(50),
Nearest_RailwayStation Varchar(50),
CONSTRAINT Neighbourhood_PK PRIMARY KEY(Neighbourhood_ID)
);

==================================================================================
CREATE TABLE Lease_Type
(
Lease_Type_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 2000),
LT_Lease_Type_Term VARCHAR(10) CHECK(LT_Lease_Type_Term IN ('Short','Long')) NOT
NULL,
CONSTRAINT Lease_Type_PK PRIMARY KEY(Lease_Type_ID)
);

==================================================================================
CREATE TABLE Property
(
Property_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
```

```
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 2000),
User_ID NUMBER NOT NULL,
Neighbourhood_ID NUMBER NOT NULL,
Admin_ID NUMBER NOT NULL,
Lease_Type_ID  NUMBER NOT NULL,
Property_Name VARCHAR(50),
Property_Available_From DATE NOT NULL,
Property_Status VARCHAR(50) NOT NULL,
Property_Verified VARCHAR(10) NOT NULL CHECK(Property_Verified IN('Y','N')),
CONSTRAINT Property_PK PRIMARY KEY(Property_ID),
CONSTRAINT Property_FK FOREIGN KEY(User_ID) REFERENCES Users(User_ID),
CONSTRAINT Property_FK1 FOREIGN KEY(Neighbourhood_ID) REFERENCES
Neighbourhood(Neighbourhood_ID),
CONSTRAINT Property_FK2 FOREIGN KEY(Admin_ID) REFERENCES Admin(Admin_ID),
CONSTRAINT Property_FK3 FOREIGN KEY(Lease_Type_ID) REFERENCES
Lease_Type(Lease_Type_ID)
);
 ===================================================================================
 CREATE TABLE Property_Feature
(
Property_Feature_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 3000),
Property_ID NUMBER NOT NULL,
PF_Gym VARCHAR(10) CHECK(PF_GYM IN ('Y','N')),
PF_Pool VARCHAR(10) CHECK(PF_Pool IN ('Y','N')),
PF_Pets_Allowed VARCHAR(10) CHECK(PF_Pets_Allowed IN ('Y','N')),
PF_Badminton_Court VARCHAR(10) CHECK(PF_Badminton_Court IN ('Y','N')),
PF_Tennis_Court VARCHAR(10) CHECK(PF_Tennis_Court IN ('Y','N')),
PF_Gaming_Zone VARCHAR(10) CHECK(PF_Gaming_Zone IN ('Y','N')),
PF_Soccer_Court VARCHAR(10) CHECK(PF_Soccer_Court IN ('Y','N')),
PF_Washer_Dryer VARCHAR(10) CHECK(PF_Washer_Dryer IN ('Y','N')),
PF_WD_CoinOperated VARCHAR(10) CHECK(PF_WD_CoinOperated IN ('Y','N')),
PF_SemiFurnished VARCHAR(10) CHECK(PF_SemiFurnished IN ('Y','N')),
PF_Unfurnished VARCHAR(10) CHECK(PF_Unfurnished IN ('Y','N')),
CONSTRAINT Property_Feature_PK PRIMARY KEY(Property_Feature_ID),
CONSTRAINT Property_Feature_FK FOREIGN KEY(Property_ID) REFERENCES
Property(Property_ID)
);
 ===================================================================================
 CREATE TABLE Showings
(
Showing_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 3000
),
Property_ID NUMBER NOT NULL,
Showing_Date Date NOT NULL,
Showing_Time Timestamp NOT NULL,
Showing_Enable VARCHAR(10) CHECK(Showing_Enable IN ('Y','N')),
CONSTRAINT Showings_PK PRIMARY KEY(Showing_ID),
CONSTRAINT Showings_FK FOREIGN KEY(Property_ID) REFERENCES Property(Property_ID)
);
 ==================================================================================
CREATE TABLE Payment
(
Payment_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 5000),
Property_ID NUMBER NOT NULL,
User_ID NUMBER NOT NULL,
Payment_Type VARCHAR(20) NOT NULL CHECK(Payment_Type IN ('ACH','Credit Card')),
CONSTRAINT Payment_PK PRIMARY KEY(Payment_ID),
CONSTRAINT Payment_Property_FK FOREIGN KEY(Property_ID) REFERENCES
Property(Property_ID),
CONSTRAINT Payment_User_FK FOREIGN KEY(User_ID) REFERENCES Users(User_ID)
);
 ==================================================================================
CREATE TABLE Promo_Code
(
Promo_Code_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000),
Payment_ID NUMBER NOT NULL,
Promo_Code_Type VARCHAR(50) NOT NULL,
Promo_Discount NUMBER(38) NOT NULL,
CONSTRAINT Promo_Code_PK PRIMARY KEY(Promo_Code_ID),
CONSTRAINT Promo_Code_FK FOREIGN KEY(Payment_ID) REFERENCES Payment(Payment_ID)
);
 ==================================================================================
CREATE TABLE Address
(
Address_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 3000),
Property_ID NUMBER NOT NULL,
House_Number VARCHAR(50) NOT NULL,
Street VARCHAR(50) NOT NULL,
PinCode NUMBER(38) NOT NULL,
City VARCHAR(50) NOT NULL,
State VARCHAR(50) NOT NULL,
Region VARCHAR(50),
CONSTRAINT Address_PK PRIMARY KEY(Address_ID),
CONSTRAINT Address_FK FOREIGN KEY(Property_ID) REFERENCES Property(Property_ID)
);
==================================================================================
CREATE TABLE Property_Type

(

Property_Type_ID NUMBER GENERATED BY DEFAULT AS IDENTITY

(

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 5000),

Property_ID NUMBER NOT NULL,

NoofBedrooms NUMBER NOT NULL,
NoOfBathrooms NUMBER NOT NULL,
NoOfFloors NUMBER NOT NULL,

CONSTRAINT Property_Type_PK PRIMARY KEY(Property_Type_ID),

CONSTRAINT Property_Type_FK FOREIGN KEY(Property_ID) REFERENCES
Property(Property_ID));
===================================================================================
CREATE TABLE Rent
(
Rent_ID NUMBER GENERATED BY DEFAULT AS IDENTITY
(
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 5000),
Property_ID NUMBER NOT NULL,
Rent_Start_Date DATE NOT NULL,
Rent_End_Date DATE NOT NULL,
Rent_Amount NUMBER NOT NULL,
CONSTRAINT Rent_PK PRIMARY KEY(Rent_ID),
CONSTRAINT Rent_FK FOREIGN KEY(Property_ID) REFERENCES Property(Property_ID));
```

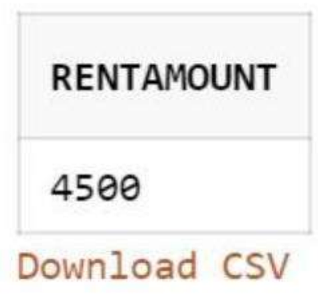
GP4: POPULATED DATA & QUERY OUTPUT

QUERY 1: Extracts the second lowest rent for the properties available

```
select
   min(Rent_Amount)
      from
        (select distinct Rent_Amount from Rent order by Rent_Amount desc)where
rownum<=2;
```

Concepts Used: MIN, DISTINCT, ORDERBY, OPERATORS, SUBQUERY

Query Output:



RENTAMOUNT
4500

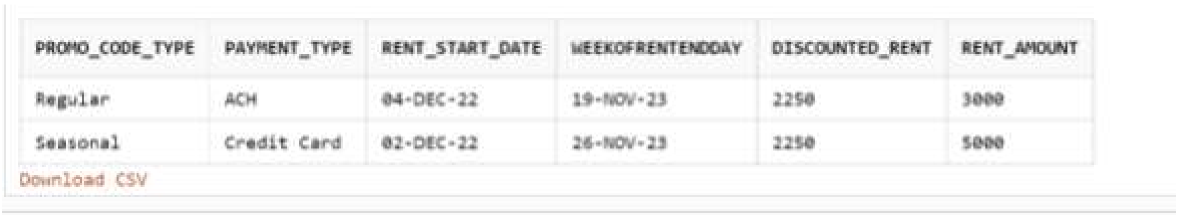
Download CSV

QUERY 2: Extracts promo code types, rent start and end dates, rent amounts for all the payments that have been made under discount of greater than 20 %

```
SELECT
   PROMO_CODE.PROMO_CODE_TYPE,
   payment.payment_type,
   rent.rent_start_date,
   trunc(rent.RENT_END_DATE,'DAY') as WeekOfRentEndDay,
   -- calculate rent after applying promo discount
   rent.rent_amount*promo_discount/100 as discounted_rent,
   rent.rent_amount
from promo_code
   left join payment on
      promo_code.payment_id=payment.payment_id
   left join rent on
      payment.property_id=rent.property_id
         where promo_code.promo_discount>20
            and rent.rent_start_date >SYSDATE+15
```

Concept Used: LEFT JOIN, OPERATORS, ARITHMETIC OPERATIONS ON DATE, TRUNC FUNCTION ON DATE

Query Output:



PROMO_CODE_TYPE	PAYMENT_TYPE	RENT_START_DATE	WEEKOFRENTENDDAY	DISCOUNTED_RENT	RENT_AMOUNT
Regular	ACH	04-DEC-22	19-NOV-23	2250	3000
Seasonal	Credit Card	02-DEC-22	26-NOV-23	2250	5000

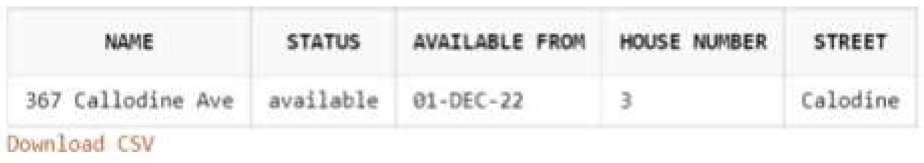
Download CSV

QUERY 3: Address details of the property with 'tops' as nearest grocery in the neighborhood and its availability

```
SELECT
   INITCAP(Property.Property_Name) AS PropertyName,
   NVL(Property.Property_Status,'Rented') AS Status,
   Property.Property_Available_From AS "AVAILABLE FROM",
   Address.House_Number AS "HOUSE NUMBER",
   Address.Street AS Street
FROM Property
   JOIN Address ON
      Property.Property_ID=Address.Property_ID
WHERE Property.Property_Name=
   (SELECT Property.Property_Name
FROM Neighbourhood
   JOIN Property ON
      Neighbourhood.Neighbourhood_ID=Property.Neighbourhood_ID
         WHERE Neighbourhood.Nearest_Grocery='Tops')
```

Concept Used: JOIN, SUB-QUERY, ALIASING, INITCAP, NVL

Query Output:



NAME	STATUS	AVAILABLE FROM	HOUSE NUMBER	STREET
367 Callodine Ave	available	01-DEC-22	3	Calodine

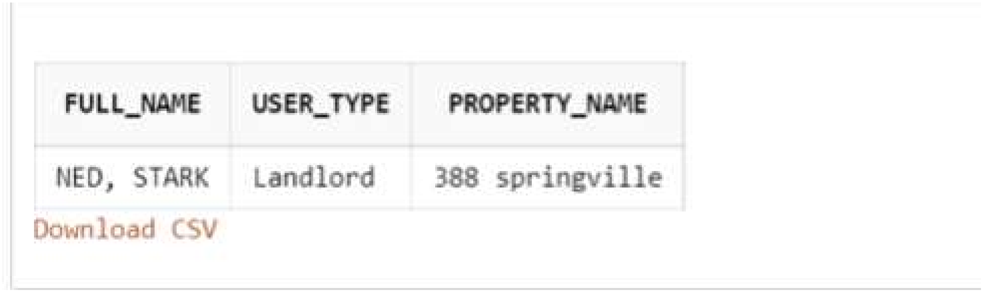
Download CSV

QUERY 4: Find listed user, user type and listed property with highest showings

```
WITH
   highestShowingProperty AS
      (select property.property_name, count(*), property.property_id from property
         join showings on property.property_id = showings.property_id
            group by property.property_name, property.property_id  order by 2 desc
fetch first 1 row only)
select UPPER(CONCAT(CONCAT(users.user_f_name, ', '), users.user_l_name)) as
Full_Name,
   users.user_type, property.property_name
      from users join property on
             users.user_id = property.user_id
            where property.property_id in (select property_id from
highestShowingProperty);
```

Concept Used: JOIN, UPPER, CONCAT, SUBQUERY IN FROM, GROUP BY, ORDER BY, FETCH, ALIASING

Query Output:



FULL_NAME	USER_TYPE	PROPERTY_NAME
NED, STARK	Landlord	388 springville

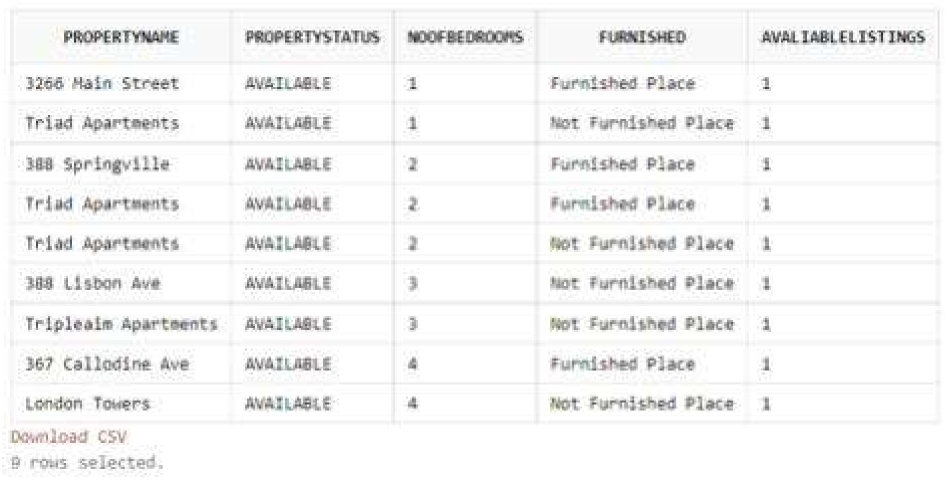
Download CSV

QUERY 5: Selected details of property where rent is between 1000 and 4500

```
select
   INITCAP(pt.property_name) as PROPERTYNAME,
   UPPER(pt.property_status)as PROPERTYSTATUS,
   ptype.noofbedrooms,
   --casing because data is for unfurnished
   CASE pf.pf_unfurnished
      when 'N' then 'Furnished Place'
         else 'Not Furnished Place'
      end as FURNISHED,
   count(*) as AvaliableListings
from property pt left outer join property_type ptype on
   pt.property_id=ptype.property_id left outer join property_feature pf on
   ptype.property_id=pf.property_id left outer join rent r on
pf.property_id=r.property_id
   --condition for rent range
   where r.rent_amount between 1000 and 4500
      group by pt.property_name, pt.property_status,
ptype.noofbedrooms,pf.pf_unfurnished having count(*) >=1
   --order by number of bedrooms
       order by ptype.noofbedrooms;
```

Concept Used: INITCAP, UPPER, CASE, COUNT, ALIAS, JOIN, GROUPBY, ORDERBY, ALIASING

Query Output:



PROPERTYNAME	PROPERTYSTATUS	NOOFBEDROOMS	FURNISHED	AVALIABLELISTINGS
3266 Main Street	AVAILABLE	1	Furnished Place	1
Triad Apartments	AVAILABLE	1	Not Furnished Place	1
388 Springville	AVAILABLE	2	Furnished Place	1
Triad Apartments	AVAILABLE	2	Furnished Place	1
Triad Apartments	AVAILABLE	2	Not Furnished Place	1
388 Lisbon Ave	AVAILABLE	3	Not Furnished Place	1
Tripleaim Apartments	AVAILABLE	3	Not Furnished Place	1
367 Callodine Ave	AVAILABLE	4	Furnished Place	1
London Towers	AVAILABLE	4	Not Furnished Place	1

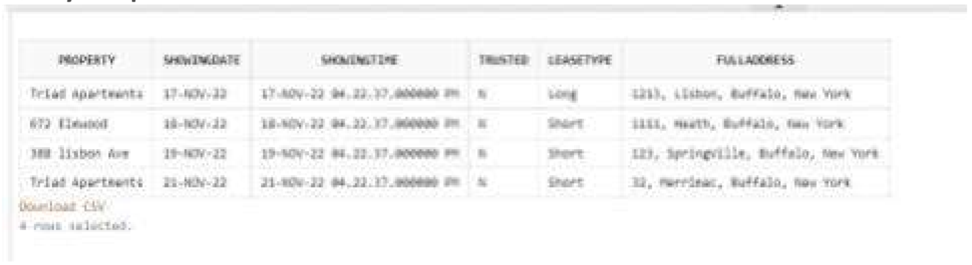
Download CSV
9 rows selected.

QUERY 6: Checking the property details for the next 4 showings for broker

```
select
   property.property_name as PROPERTY,
   showings.showing_date as SHOWINGDATE,
   showings.showing_time as SHOWINGTIME,
   property.property_verified as TRUSTED,
   lease_type.lt_lease_type_term as LeaseType,
   CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(addr.house_number,', '),addr.street),', 
'),addr.city),', '),addr.state) as FullAddress
from property join showings on
   property.property_id=showings.property_id join users on
      property.user_id=users.user_id join lease_type on
             property.lease_type_id= lease_type.lease_type_id join
            address addr on property.property_id=addr.property_id
where showings.showing_enable='Y'
   and showings.showing_date>sysdate and users.user_type like 'Bro%'
      order by showings.showing_date asc fetch first 4 rows only;
```

Concept Used: ALIAS, NESTED CONCAT, JOIN, CONDITIONAL OPERATOR, LIKE, ORDER BY, LIMITING

Query Output:



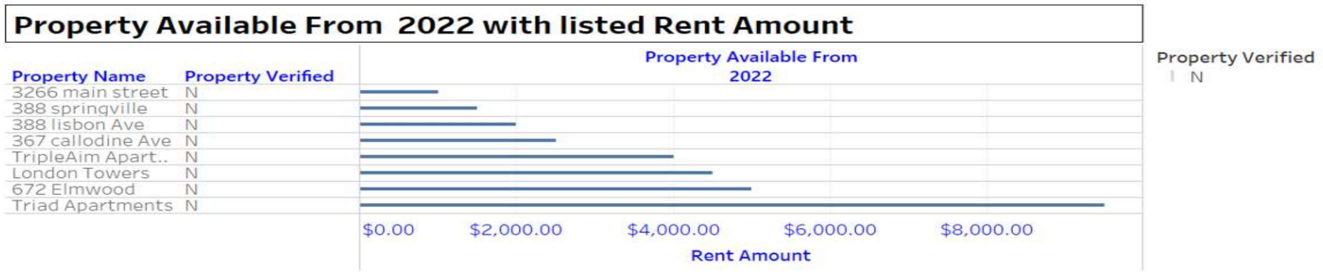
PROPERTY	SHOWINGDATE	SHOWINGTIME	TRUSTED	LEASETYPE	FULLADDRESS
Triad Apartments	17-NOV-22	17-NOV-22 04.22.37.000000 PM	N	Long	1213, Lisbon, Buffalo, New York
672 Elmwood	18-NOV-22	18-NOV-22 04.22.37.000000 PM	N	Short	1111, Heath, Buffalo, New York
388 lisbon Ave	19-NOV-22	19-NOV-22 04.22.37.000000 PM	N	Short	123, Springville, Buffalo, New York
Triad Apartments	21-NOV-22	21-NOV-22 04.22.37.000000 PM	N	Short	32, Merrimac, Buffalo, New York

Download CSV
4 rows selected.

GP5: QUERY VISUALIZATION

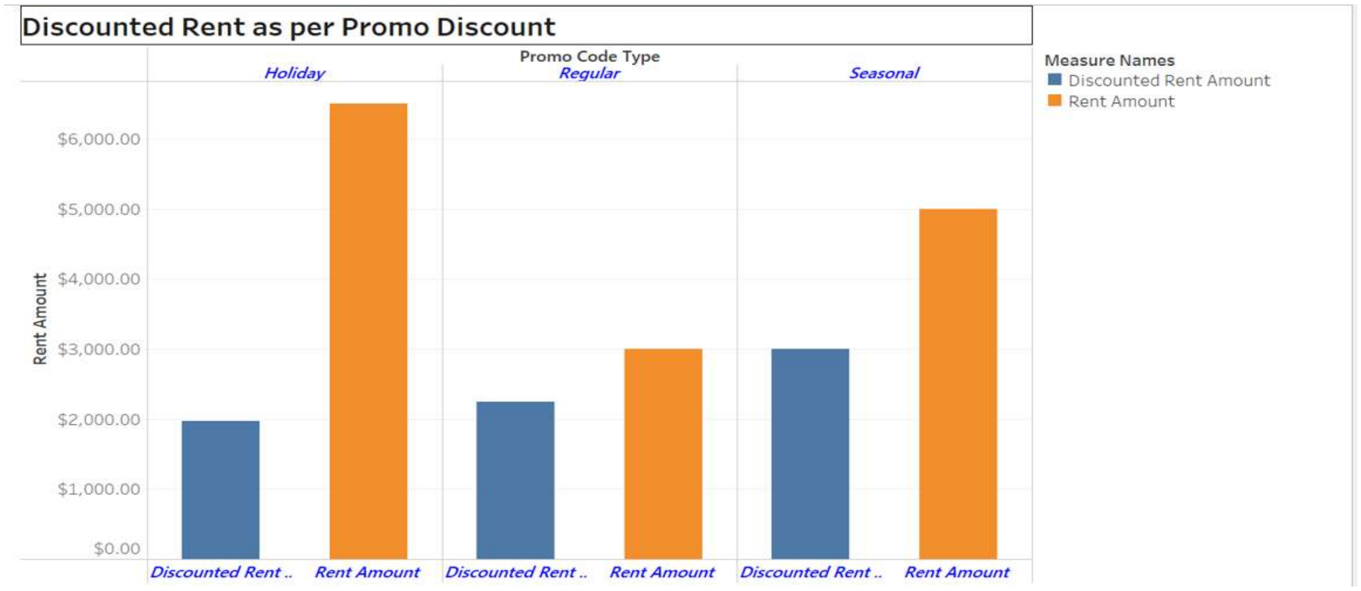
VISUALIZATION 1:

Display the available properties based on rent amount filter set from lowest to highest.



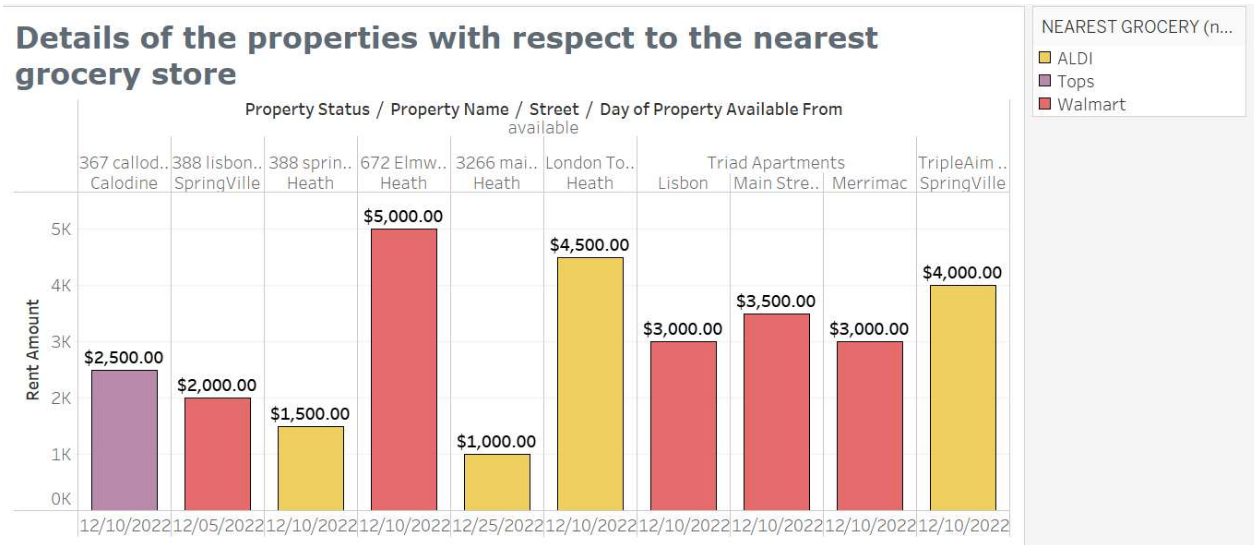
VISUALIZATION 2:

Display the discounted rent amount after applying promo code.



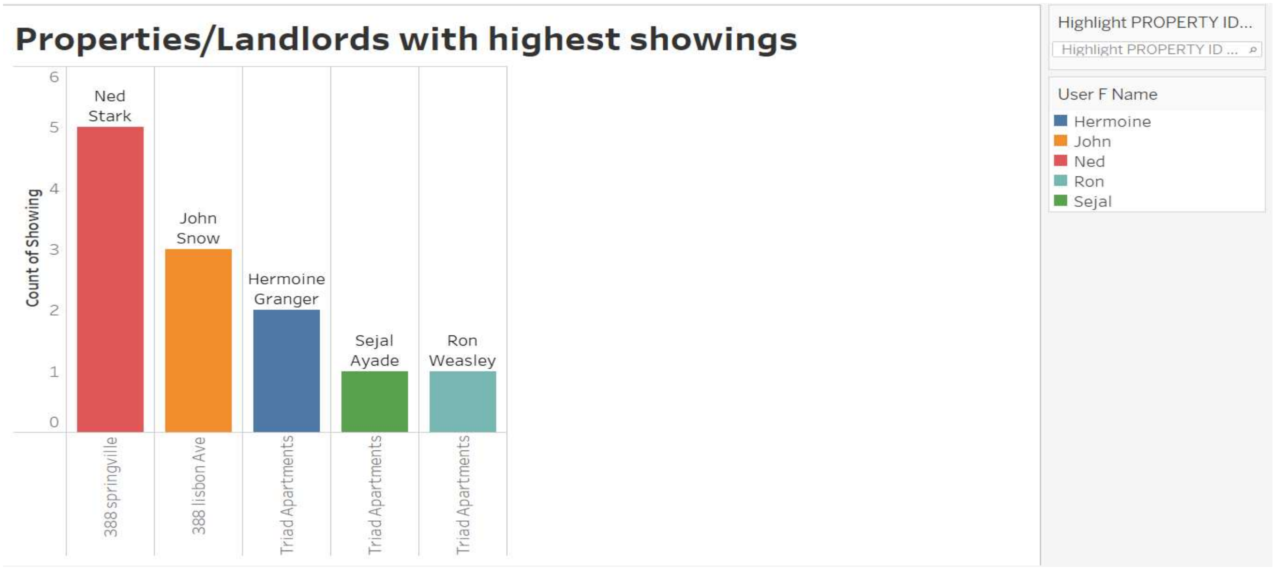
VISUALIZATION 3:

Display the property details such as rent amount, property status, date of availability filtered on nearest grocery stores.



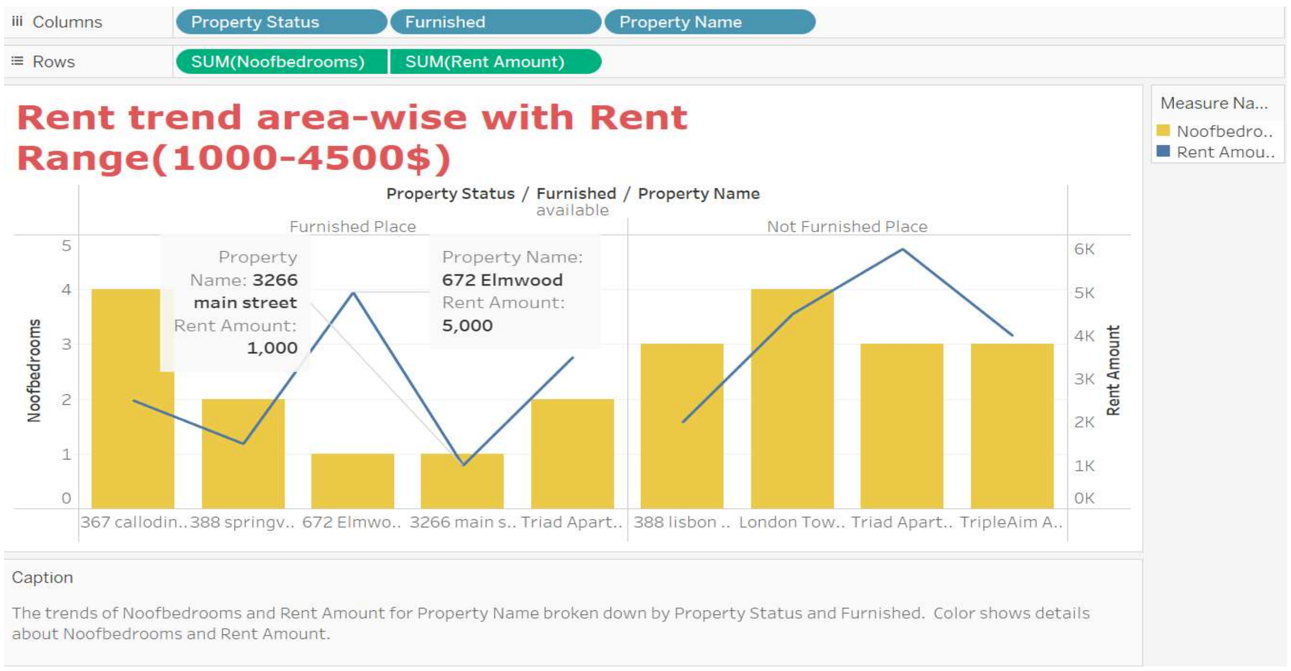
VISUALIZATION 4:

Display the top 3 properties with highest showings. Also display the property name and landlord details.



VISUALIZATION 5:

Display the rent amount trend of property filtered by furnished and unfurnished property type.

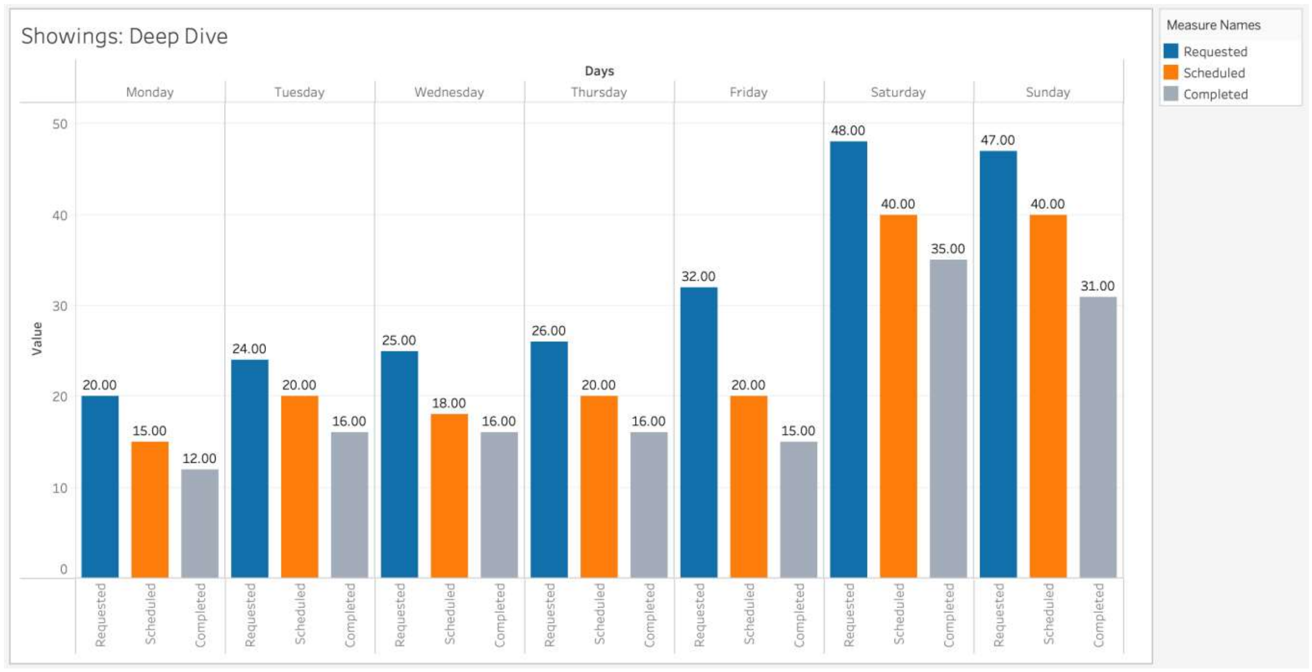


Caption

The trends of Noofbedrooms and Rent Amount for Property Name broken down by Property Status and Furnished. Color shows details about Noofbedrooms and Rent Amount.

VISUALIZATION 6:

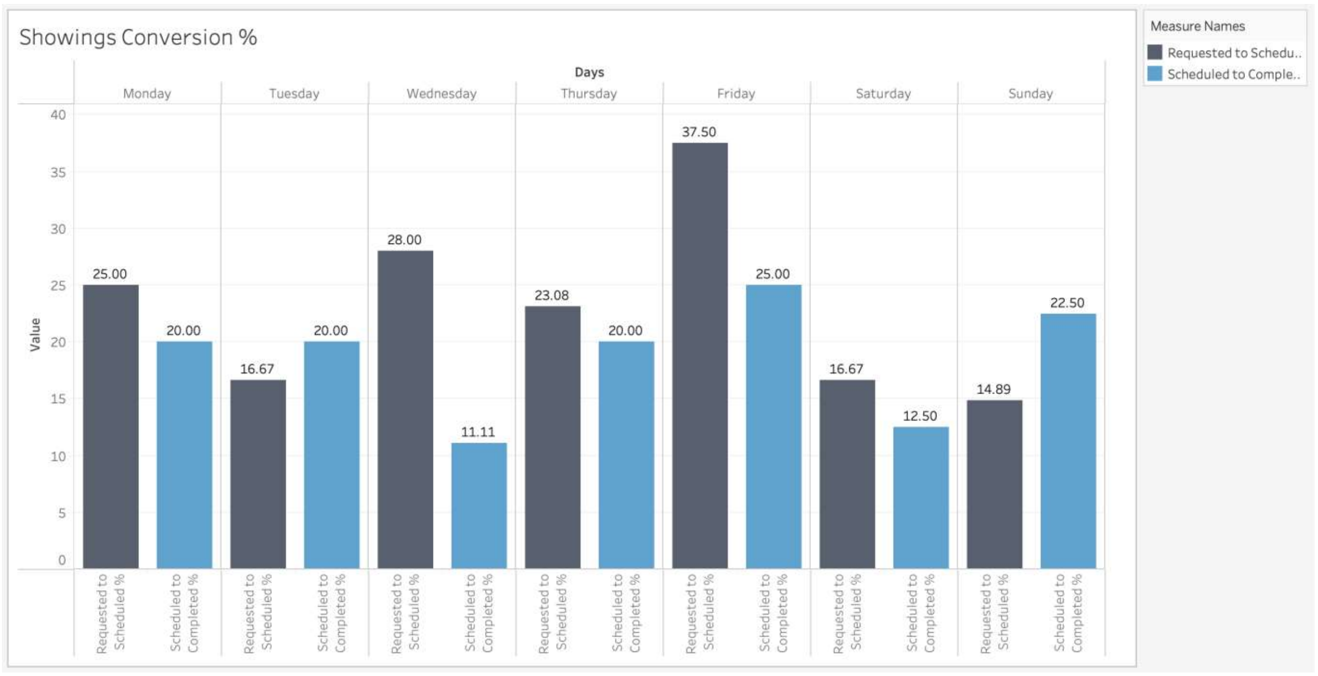
Displays the trend of showing requests spread across the week.



VISUALIZATION 7:

Displays the trend of two buckets:

- Rescheduled to Scheduled%
- Scheduled to Completed%



Days	Requested to Scheduled %	Scheduled to Completed %
Monday	25.00	20.00
Tuesday	16.67	20.00
Wednesday	28.00	11.11
Thursday	23.08	20.00
Friday	37.50	25.00
Saturday	16.67	12.50
Sunday	14.89	22.50