# Housing Rental Management System Database Design


## Team 8

**Qian Chen, Rongjing Huang, Xiyue Suo, Yuanyi Xie, Zihan Wan**


# Introduction & Objectives



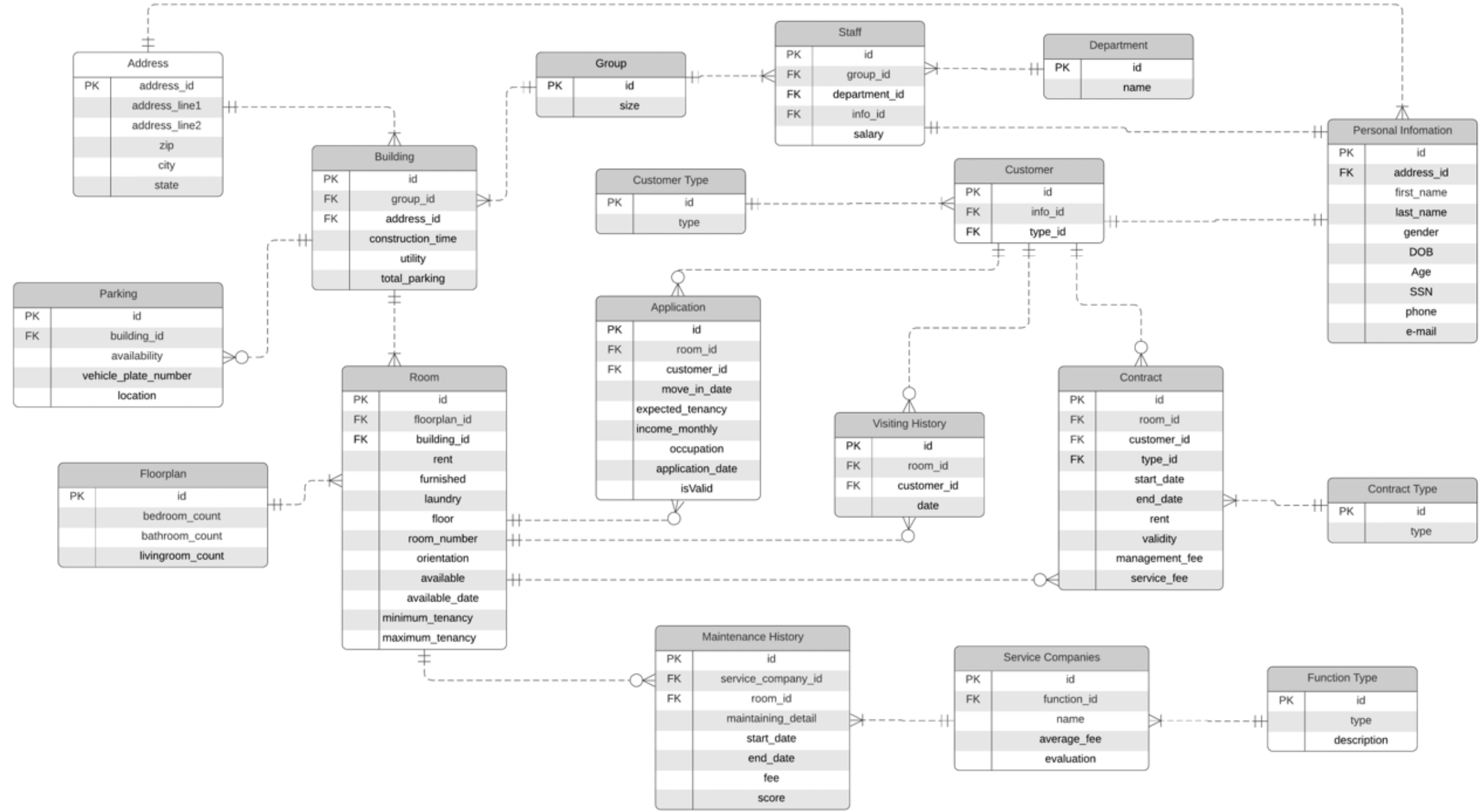
## Purposes

1. Design and implement a relational database
2. Maintain data for the housing rental management system

## Objectives

1. Design a database for staff and clients
2. Create an ERD
3. Implement simulated data
4. Analyze data

**Team 8 Project ERD**
**Online Housing Rental Management System**
**(Qian Chen, Rongjing Huang, Xiyue Suo, Yuanyi Xie, Zihan Wan)**



## Computed Columns

Personal Information – Age

```
CREATE TABLE dbo.PersonalInformation (
    id int IDENTITY(1,1) NOT NULL,
    first_name varchar(40) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    Last_name varchar(40) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    gender varchar(40) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    DOB date NOT NULL,
    Age AS (datediff(hour,[DOB],getdate())/(8766)),
    phone varchar(40) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    email varchar(40) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    address_id int NULL,
    SSN varbinary(250) NULL,
    CONSTRAINT PK__Personal__3213E83FB03E7FD7 PRIMARY KEY (id),
    CONSTRAINT FK__PersonalI__addre__3C69FB99 FOREIGN KEY (address_id) REFERENCES dbo.Address(address_id)
);
```

## Column Data Encryption

Personal Information -- SSN

```
--encryption key
CREATE MASTER KEY ENCRYPTION
BY PASSWORD = 'Team8Authority'

CREATE CERTIFICATE EncryptPersonalInfo
WITH SUBJECT = 'Team8Authority'

CREATE SYMMETRIC KEY SSNEncryptionKey
WITH ALGORITHM = AES_256 ENCRYPTION
BY CERTIFICATE EncryptPersonalInfo
```

## Computed Columns

Contract – management_fee

```
CREATE TABLE dbo.Contract (
    id int IDENTITY(1,1) NOT NULL,
    room_id int NOT NULL,
    customer_id int NULL,
    start_date date NULL,
    end_date date NULL,
    rent money NULL,
    validity varchar(100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    service_fee money NULL,
    type_id int NULL,
    management_fee AS ([service_fee]*(0.1)),
    CONSTRAINT Contract_PK PRIMARY KEY (id),
    CONSTRAINT Contract_FK FOREIGN KEY (room_id) REFERENCES dbo.Room(id),
    CONSTRAINT Contract_FK_1 FOREIGN KEY (customer_id) REFERENCES dbo.Customer(id),
    CONSTRAINT Contract_FK_2 FOREIGN KEY (type_id) REFERENCES dbo.ContractType(id)
);
```

## Table-Level Check Constraints

Check Availability

```
CREATE FUNCTION CheckAvailability (@Sdate date)
RETURNS int
AS
BEGIN
    DECLARE @BlankDays INT = 0;
    SELECT @BlankDays = DATEDIFF(DAY, available_date, end_date)
    FROM Contract c
    JOIN Room r
    ON r.id = c.room_id
    RETURN @BlankDays
END

ALTER TABLE Contract
ADD CONSTRAINT AvailabilityEvaluation CHECK (dbo.CheckAvailability(end_date)>=0);
```

# Trigger

on [application]

```
CREATE TRIGGER [dbo].[updateAppkicationStatus]
ON [dbo].[Application]
AFTER INSERT,UPDATE
AS BEGIN

    DECLARE @id int;
    DECLARE @move_in_date date;
    DECLARE @expected_tenancy int;
    DECLARE @income_monthly money;
    DECLARE @occupation varchar(100);
    DECLARE @minimum_tenancy int;
    DECLARE @maximum_tenancy int;
    DECLARE @avaliable_date date;
    DECLARE @age int;
    DECLARE @rent money;

    SELECT @minimum_tenancy=r.minimum_tenancy,
           @maximum_tenancy=r.maximum_tenancy,
           @avaliable_date=r.available_date,
           @occupation=i.occupation,
           @income_monthly=i.income_monthly,
           @id=i.id,@age=p.Age,
           @expected_tenancy=i.expected_tenancy,
           @move_in_date=i.move_in_date,
           @rent=r.rent

    FROM dbo.Room r
    FULL JOIN inserted i
    ON i.room_id=r.id
    FULL JOIN dbo.Customer c
    ON i.customer_id=c.id
    FULL JOIN dbo.PersonalInformation p
    ON c.info_id=p.id

    IF @expected_tenancy<=@maximum_tenancy AND @expected_tenancy>=@minimum_tenancy AND @move_in_date>=@avaliable_date
    AND (@occupation='student' OR @income_monthly>=1.5*@rent) AND @age>=18
        UPDATE dbo.Application SET isValid=1 WHERE id=@id

END ;
```

# Trigger

on [Contract]

```
CREATE TRIGGER [dbo].[changeAvailableDate]
ON [dbo].[Contract]
AFTER INSERT,UPDATE
AS BEGIN
    DECLARE @room_id int;
    DECLARE @end_date date;
    DECLARE @start_date date;
    DECLARE @contract_type int;
    DECLARE @customer_id int;
    DECLARE @available_date date;

    SELECT @room_id=ISNULL(i.room_id,d.room_id),
           @end_date=ISNULL(i.end_date,d.end_date),
           @start_date=ISNULL(i.start_date,d.start_date),
           @contract_type=ISNULL(i.type_id,d.type_id),
           @customer_id=ISNULL(i.customer_id,d.customer_id),
           @available_date=r.available_date

    FROM inserted i
    FULL JOIN deleted d
    ON i.room_id=d.room_id
    FULL JOIN dbo.Room r
    ON i.room_id=r.id

    IF(@contract_type=1)
    BEGIN
        IF EXISTS(SELECT room_id FROM INSERTED)
        BEGIN
            UPDATE dbo.Customer SET type=1 WHERE id=@customer_id
            IF(@available_date<DATEADD(day,1,@end_date))
                BEGIN
                UPDATE dbo.Room SET available_date=DATEADD(day,1,@end_date) WHERE id=@room_id
                END
        END
        ELSE
            UPDATE dbo.Room SET available_date=DATEADD(day,-1,@start_date) WHERE id=@room_id
    END
END;
```

# Trigger

on [Maintenance History] and [ Personal Information]

```
--add a trigger on MaintenanceHistory
CREATE TRIGGER update_average_fee
ON MaintenanceHistory
AFTER INSERT, UPDATE, DELETE
AS BEGIN
    UPDATE ServiceCompanies
    SET average_fee = (
        SELECT AVG(mh.fee)
        FROM MaintenanceHistory mh
        WHERE ServiceCompanies.id = mh.service_company_id )
END;

--add a trigger on MaintenanceHistory
CREATE TRIGGER update_evaluation
ON MaintenanceHistory
AFTER INSERT, UPDATE, DELETE
AS BEGIN
    UPDATE ServiceCompanies
    SET evaluation = (
        SELECT CAST(AVG(CONVERT(FLOAT, score)) AS DECIMAL(18,2))
        FROM MaintenanceHistory mh
        WHERE ServiceCompanies.id = mh.service_company_id )
END;

--add a trigger on PersonalInformation
CREATE TRIGGER dbo.addPerson
ON dbo.PersonalInformation
INSTEAD OF INSERT
AS BEGIN
    DECLARE @first_name varchar(40),
    @last_name varchar(40),
    @gender varchar(40),
    @DOB DATE,
    @SSN varbinary,
    @phone varchar(40),
    @email varchar(40),
    @address_id int

    SELECT @first_name=i.first_name,@last_name=i.Last_name,@gender=i.gender,
           @DOB=i.DOB,@SSN=i.SSN,@phone=i.phone,@email=i.email,@address_id=i.address_id
    FROM inserted i;

    OPEN SYMMETRIC KEY SSNEncryptionKey
    DECRYPTION BY CERTIFICATE EncryptPersonalInfo;

    INSERT INTO dbo.PersonalInformation(first_name,last_name,gender,DOB,SSN,phone,email,address_id)
    VALUES (@first_name,@last_name,@gender,@DOB,EncryptByKey(Key_GUID('SSNEncryptionKey')
    , @ssn,1, HashBytes('SHA1', CONVERT(varbinary(250),@phone))),@phone,@email,@address_id);
END;
```

# View 1 - Rooms Distribution in MA

```
CREATE VIEW roomDistribution
AS
SELECT r.id as [room_id],
       r.building_id,
       a.address_id,
       a.city,a.state
from Room r
left join Building b on b.id = r.building_id
left join Address a on a.address_id =b.address_id
where r.avaliable = 1;
```

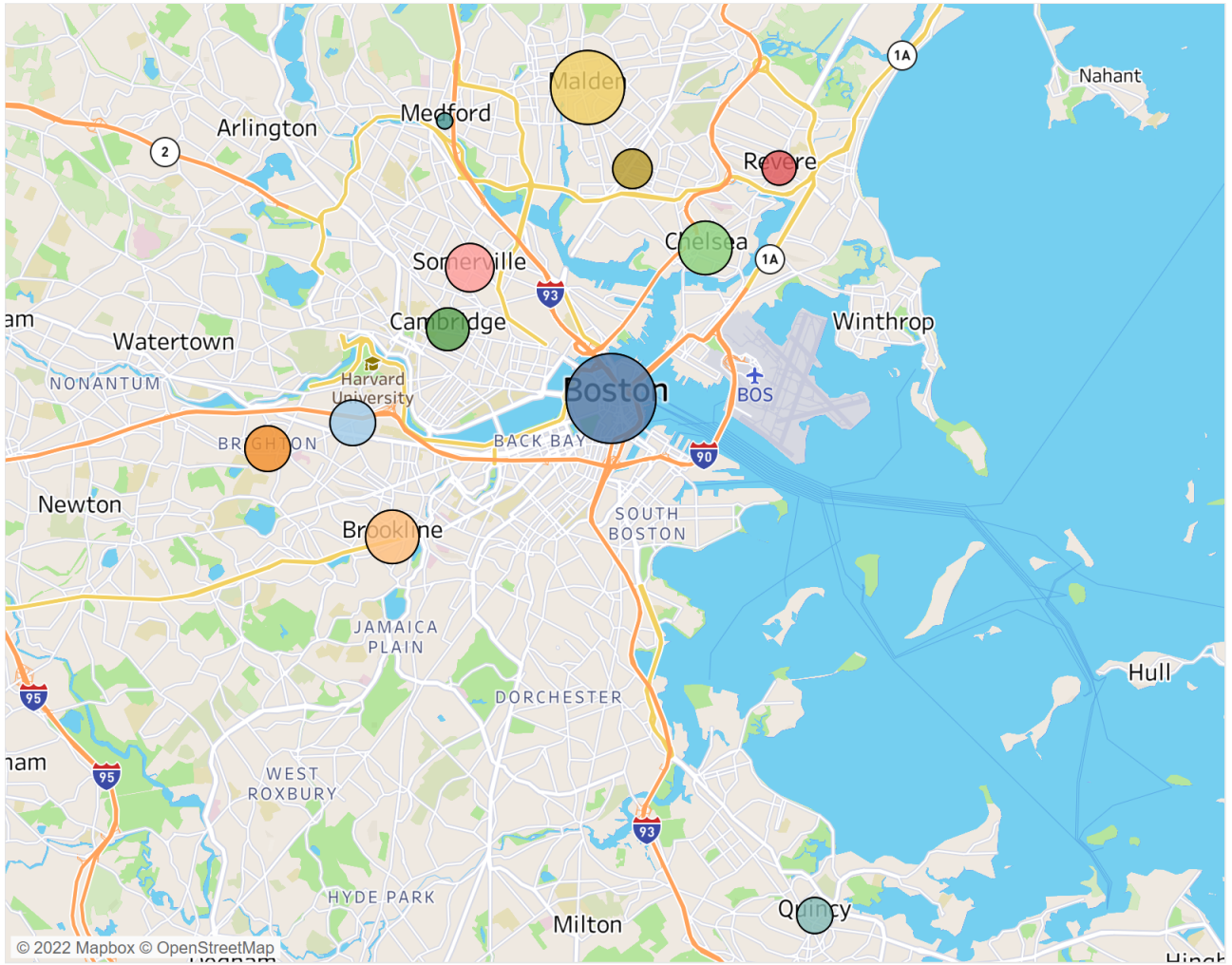
# Visualization Analysis Report 1

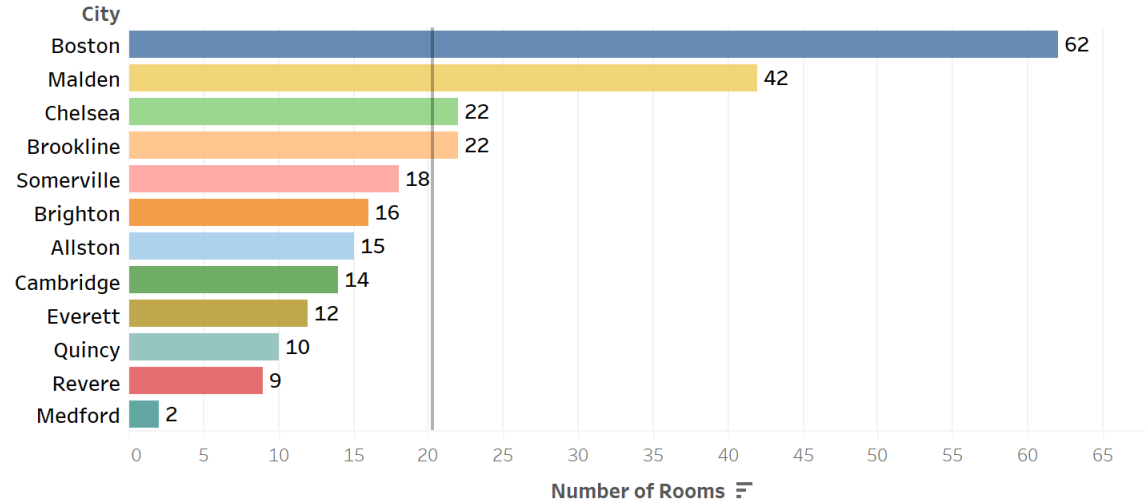
This is the distribution map of MA's rooms by city dimension.

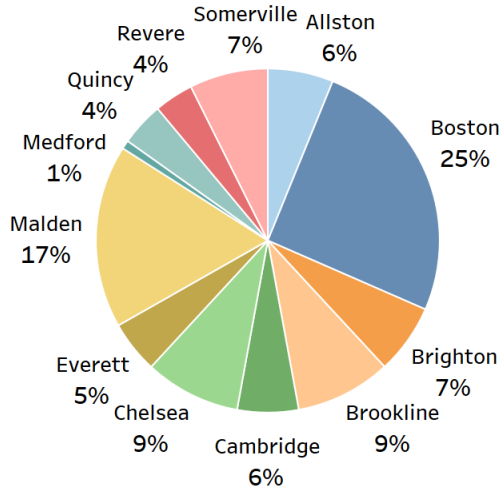
Boston has the most rooms, accounting for 25% of the total number of rooms.

Medford has the least rooms, only accounting for 1% of the total number of rooms.

## Room Distribution - By city dimension



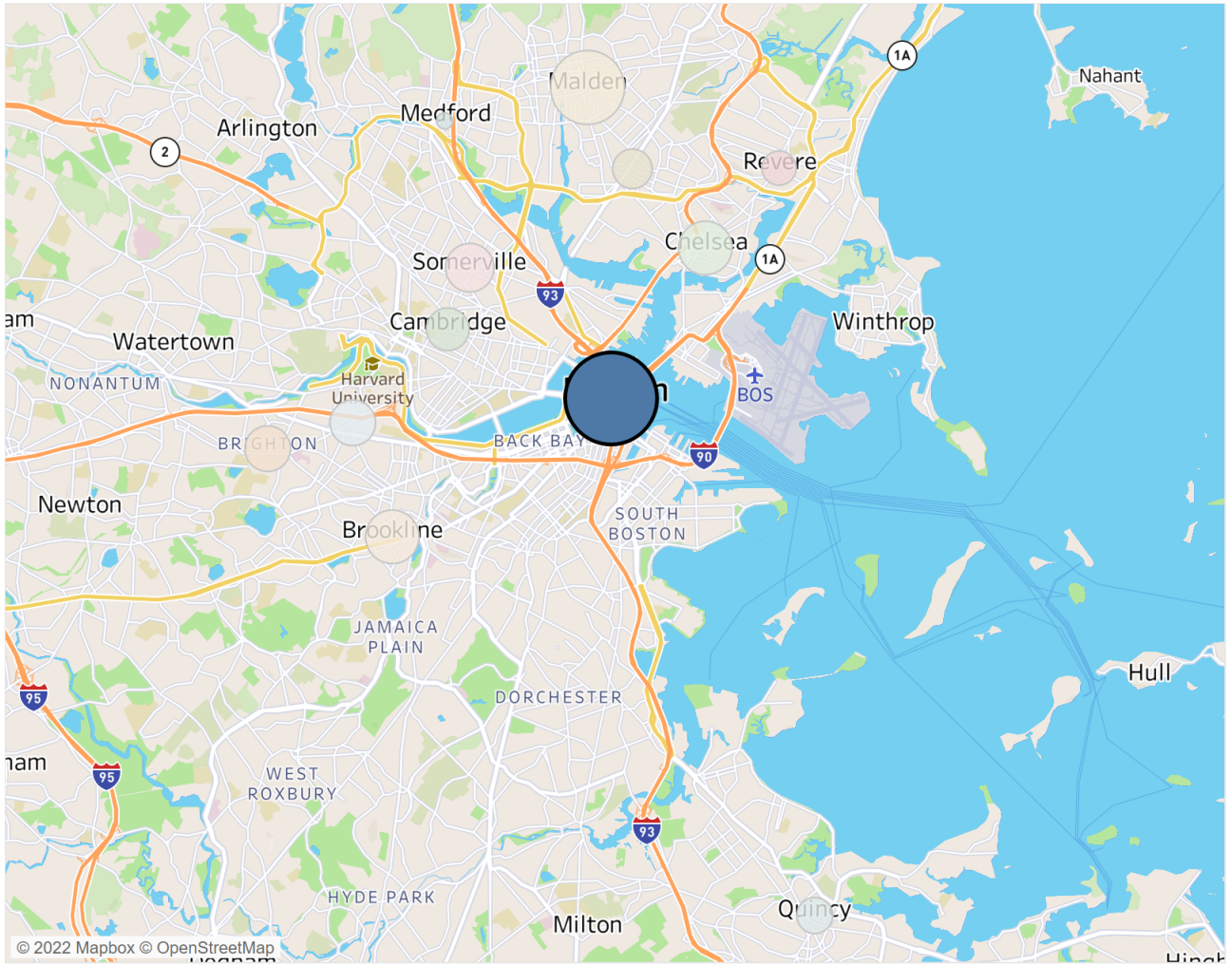
# Visualization Analysis Report 1

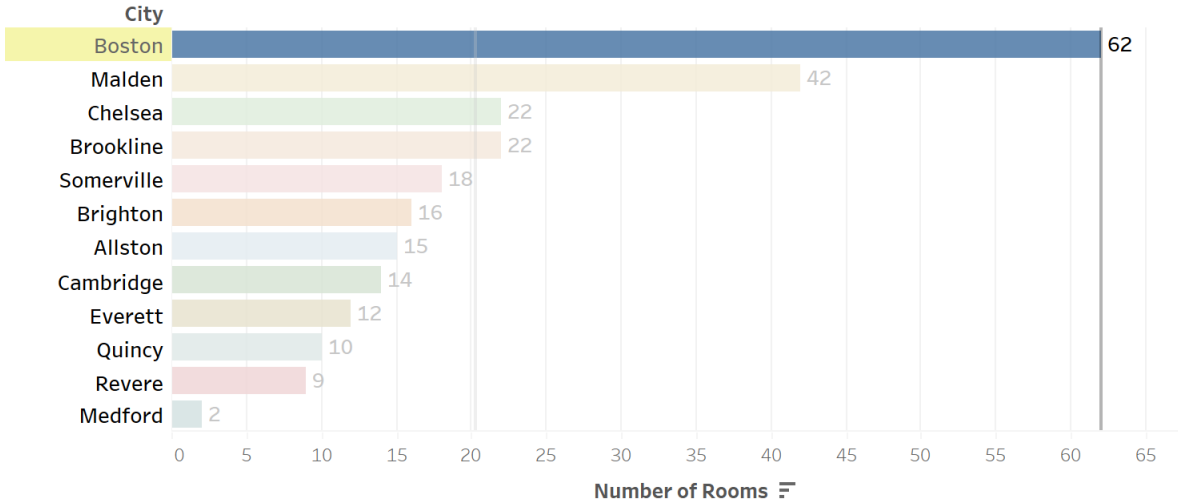
This is the distribution map of MA's rooms by city dimension.

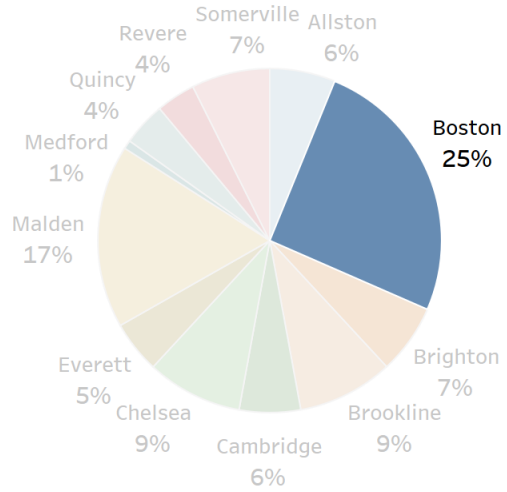
**Boston has the most rooms, accounting for 25% of the total number of rooms.**

Medford has the least rooms, only accounting for 1% of the total number of rooms.

## Room Distribution - By city dimension



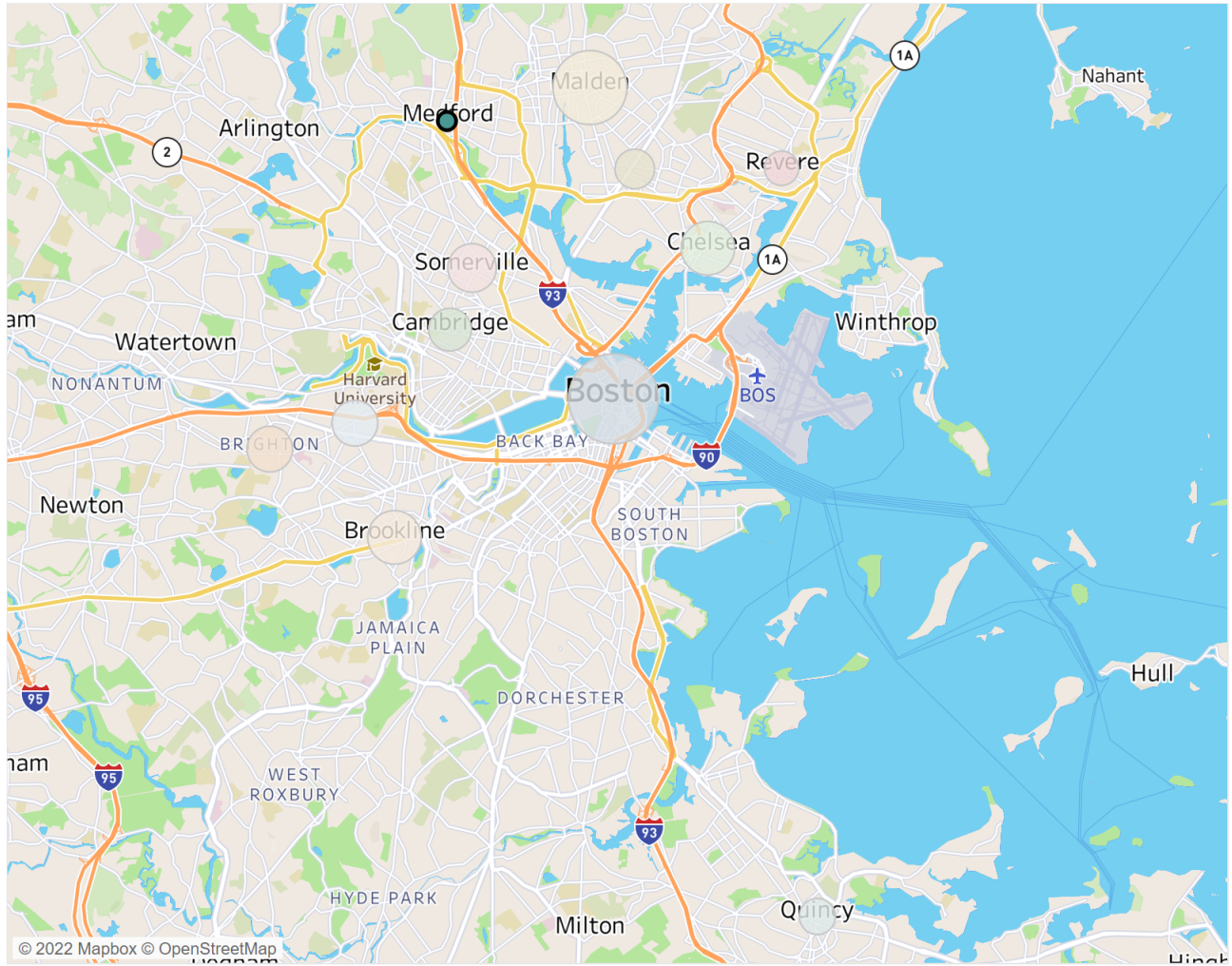
# Visualization Analysis Report 1

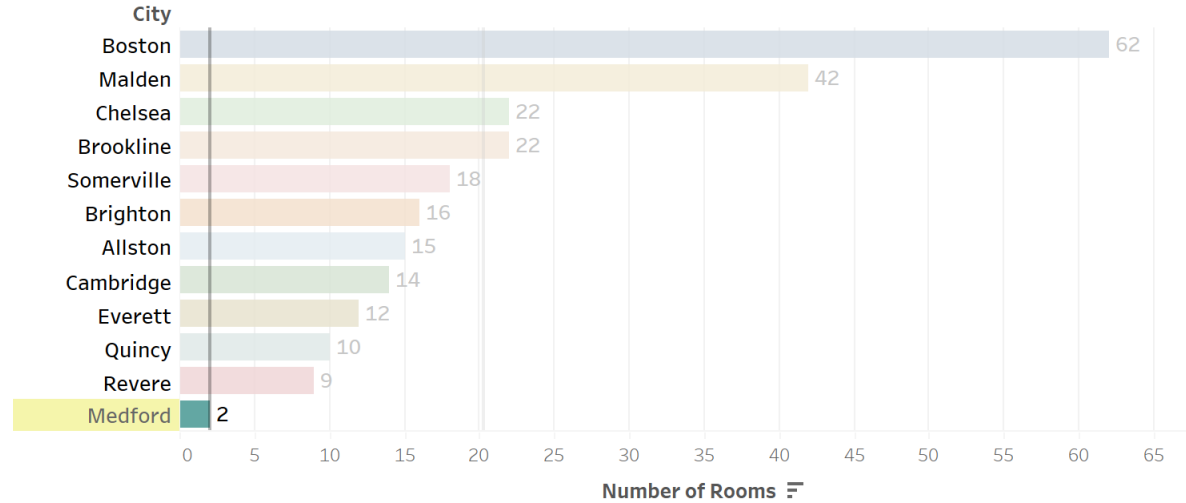
This is the distribution map of MA's rooms by city dimension.

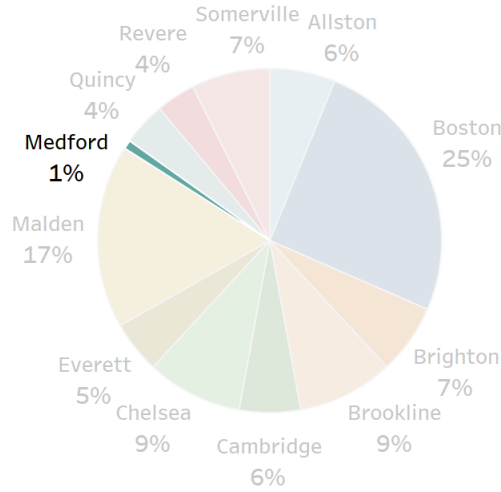
Boston has the most rooms, accounting for 25% of the total number of rooms.

Medford has the least rooms, only accounting for 1% of the total number of rooms.

## Room Distribution - By city dimension



# View 2 - Customer Portrait

```
CREATE VIEW applicationDistribution
AS
SELECT a.room_id,
       a2.city,
       f.id as [Room_Type],
       f.bedroom_count,
       f.bathroom_count,
       a.customer_id,
       p.gender,
       p.Age,
       a.occupation,
       a.income_monthly
from Application a
join Room r on r.id = a.room_id
join Floorplan f on f.id = r.floorplan_id
join Building b on b.id = r.building_id
join Address a2 on a2.address_id = b.address_id
join Customer c on c.id = a.customer_id
join PersonalInformation p on p.id = c.info_id;
```

# Visualization Analysis Report 2

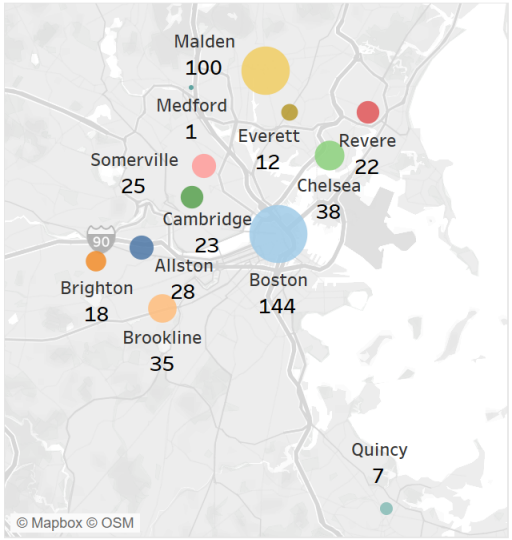
This is the distribution map of applications.

Students prefer to live in Boston and Malden.

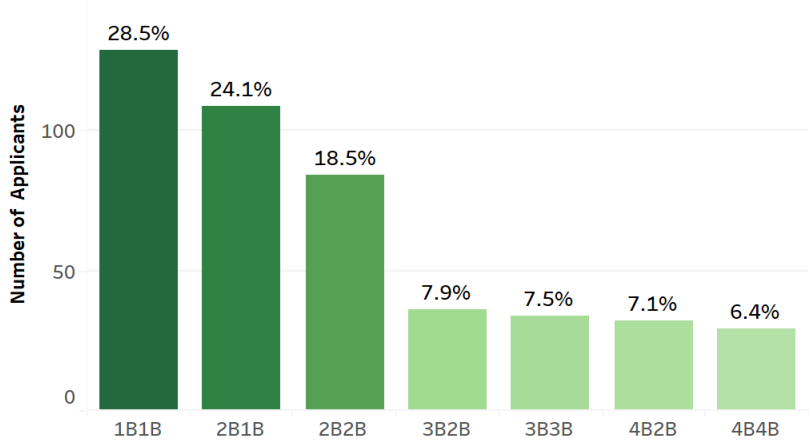
Females prefer room types with 1B or 2B.

## Application Distribution

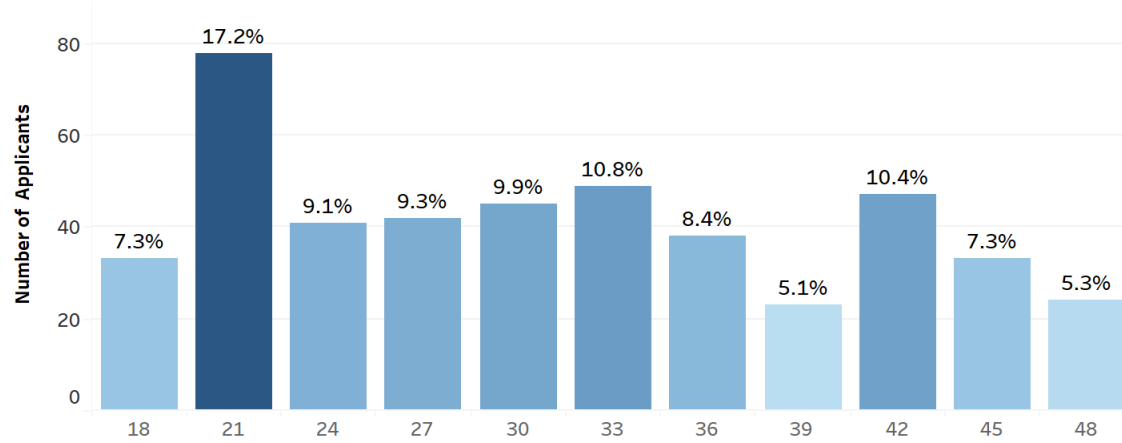
### - City



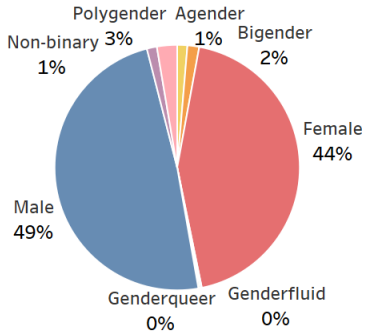

### - Room Type



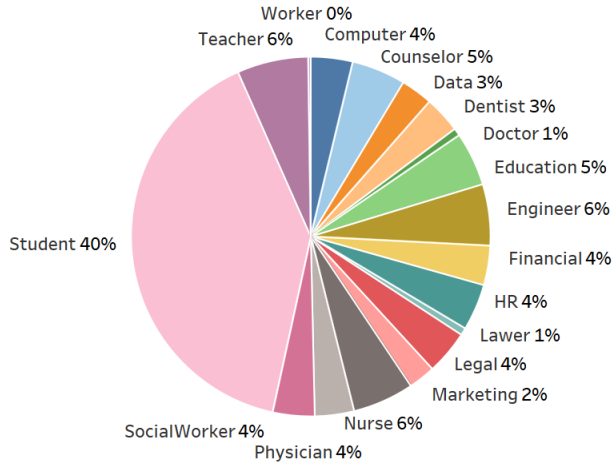

### - Age



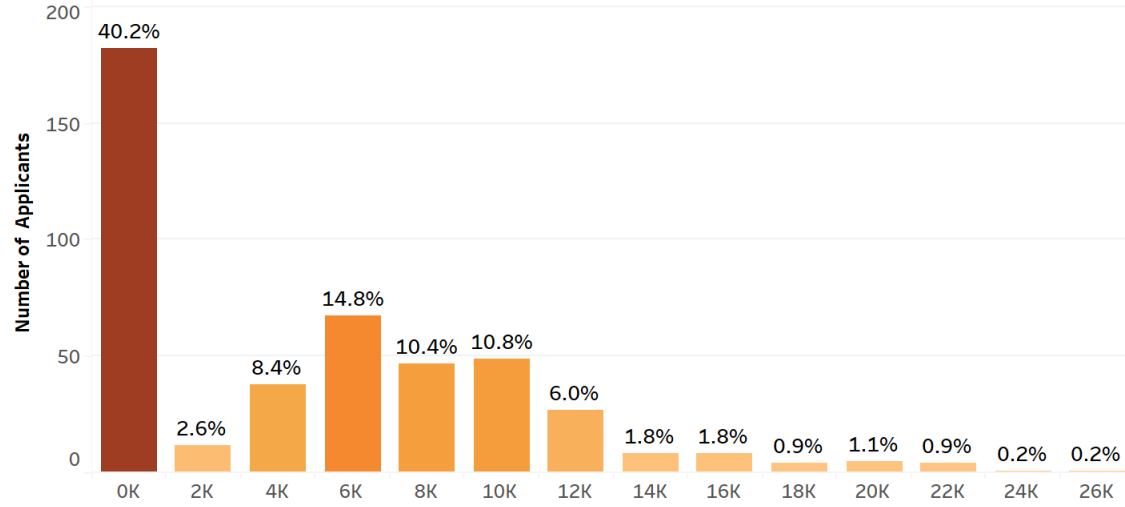

### - Gender




### - Job




### - Monthly Income



# Visualization Analysis Report 2

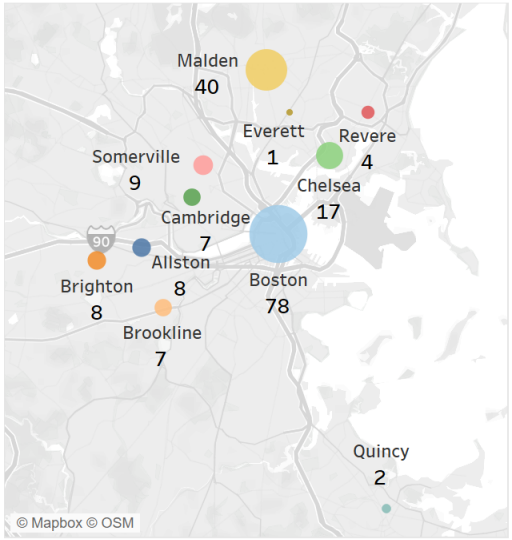
This is the distribution map of applications.

Students prefer to live in Boston and Malden.

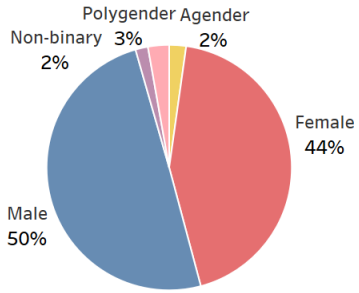
Females prefer room types with 1B or 2B.

## Application Distribution

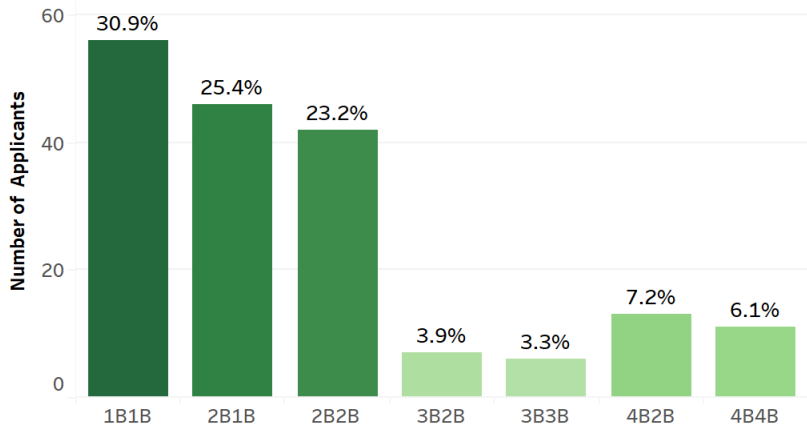
### - City



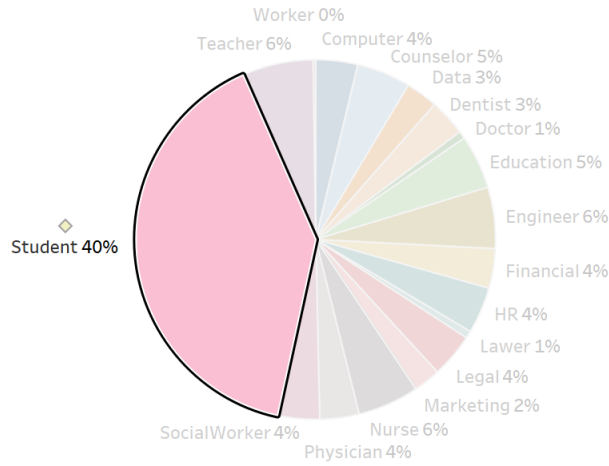

### - Gender



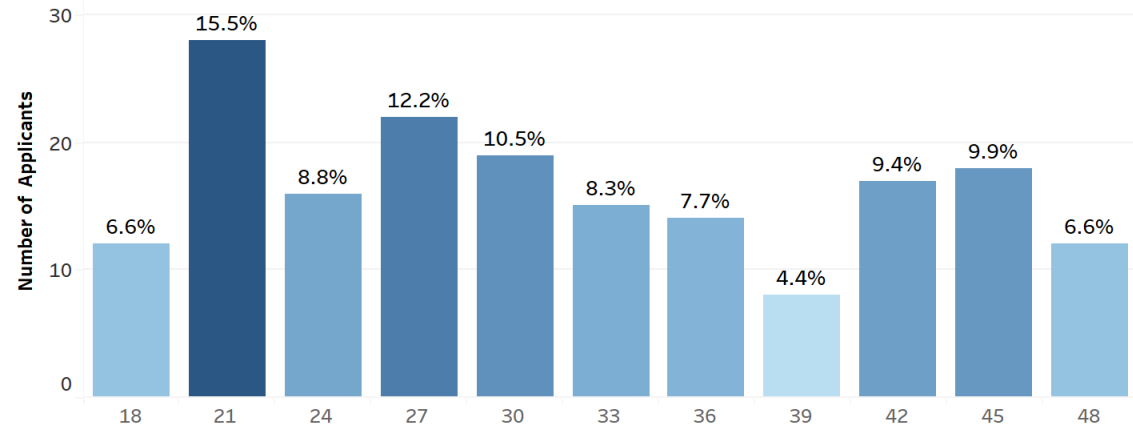

### - Room Type



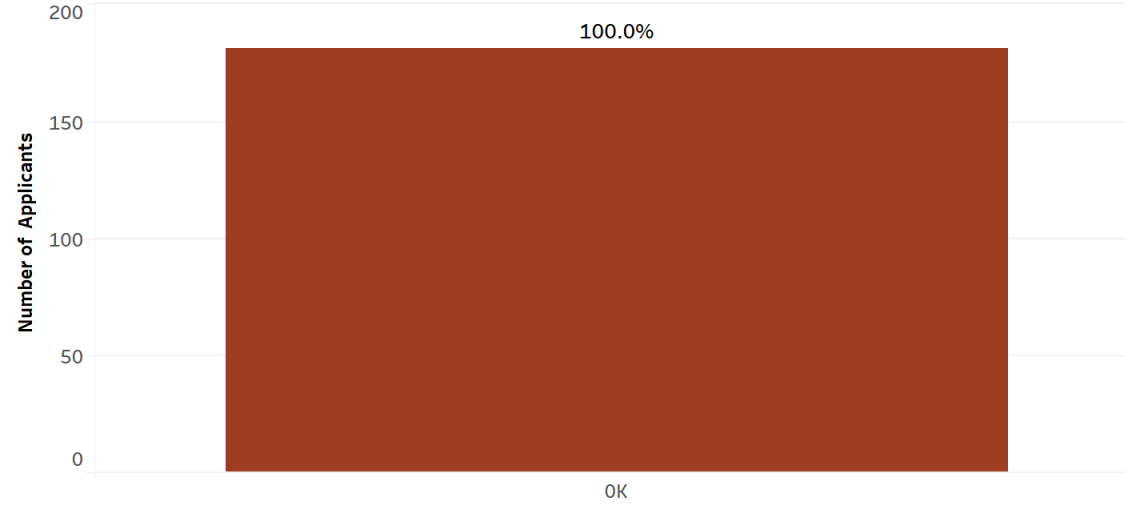

### - Job




### - Age




### - Monthly Income



# Visualization Analysis Report 2

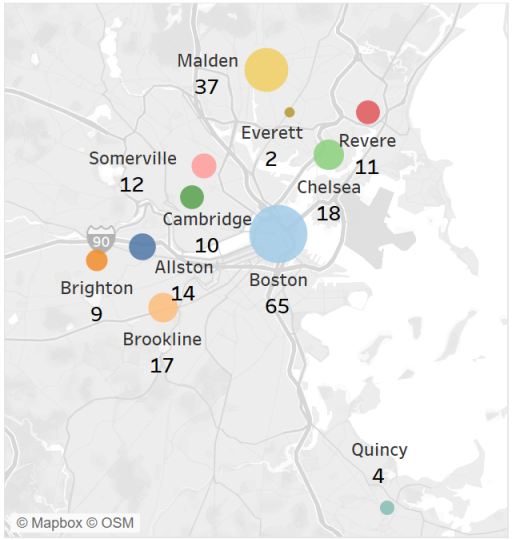
This is the distribution map of applications.

Students prefer to live in Boston and Malden.

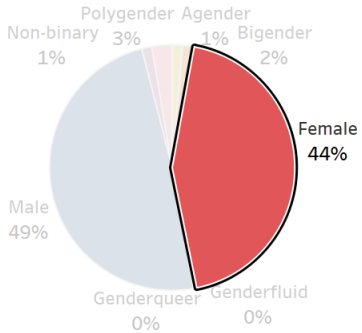
Females prefer room types with 1B or 2B.

## Application Distribution

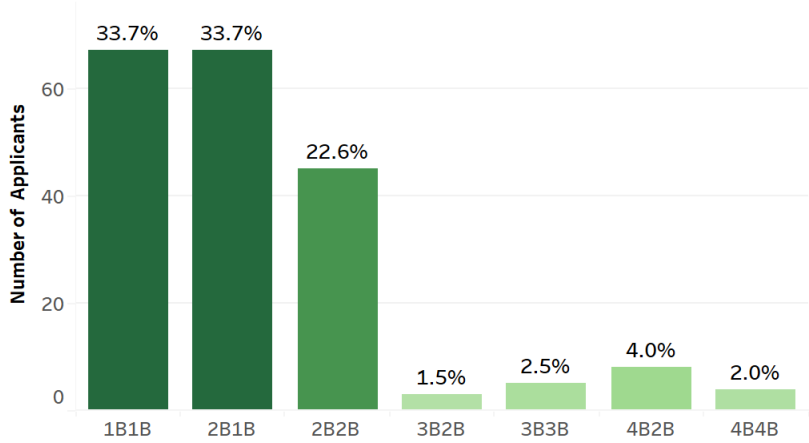
### - City



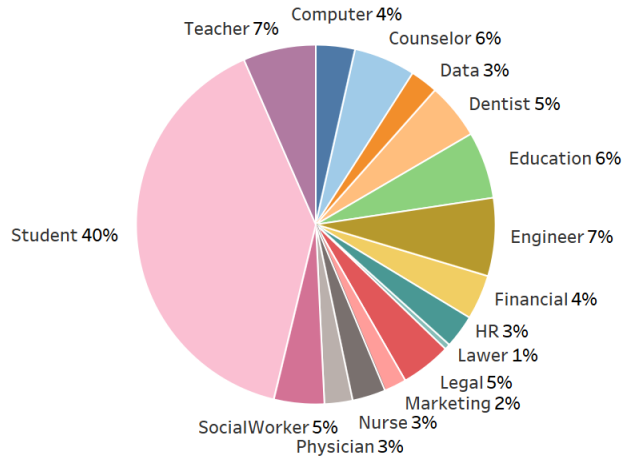

### - Gender



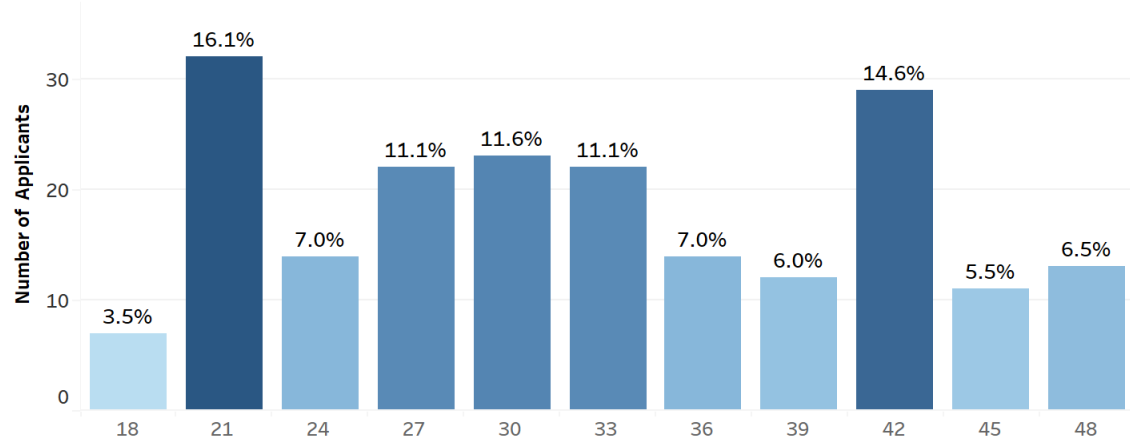

### - Room Type



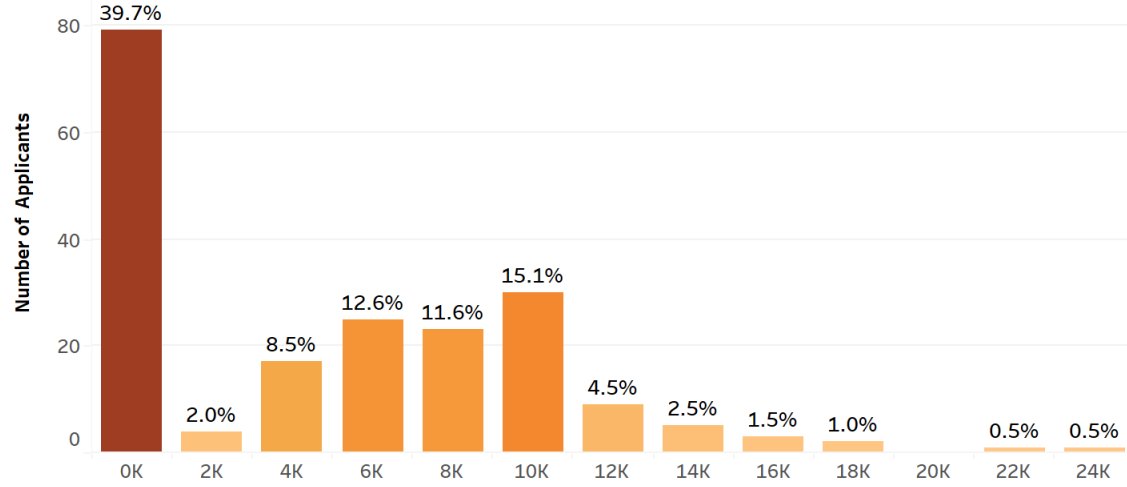

### - Job




### - Age




### - Monthly Income



# View 3 - Housing Services Company Evaluation

```
CREATE VIEW serviceCompanyCompare
AS
SELECT sc.name as [service_company_name],
       ft.[type] as [service_type],
       count(sc.id) as [times_of_service],
       average_fee,evaluation
FROM ServiceCompanies sc
join FunctionType ft ON ft.id = sc.function_id
join MaintenanceHistory mh ON mh.service_company_id = sc.id
group by sc.name,ft.[type],average_fee,evaluation;
```

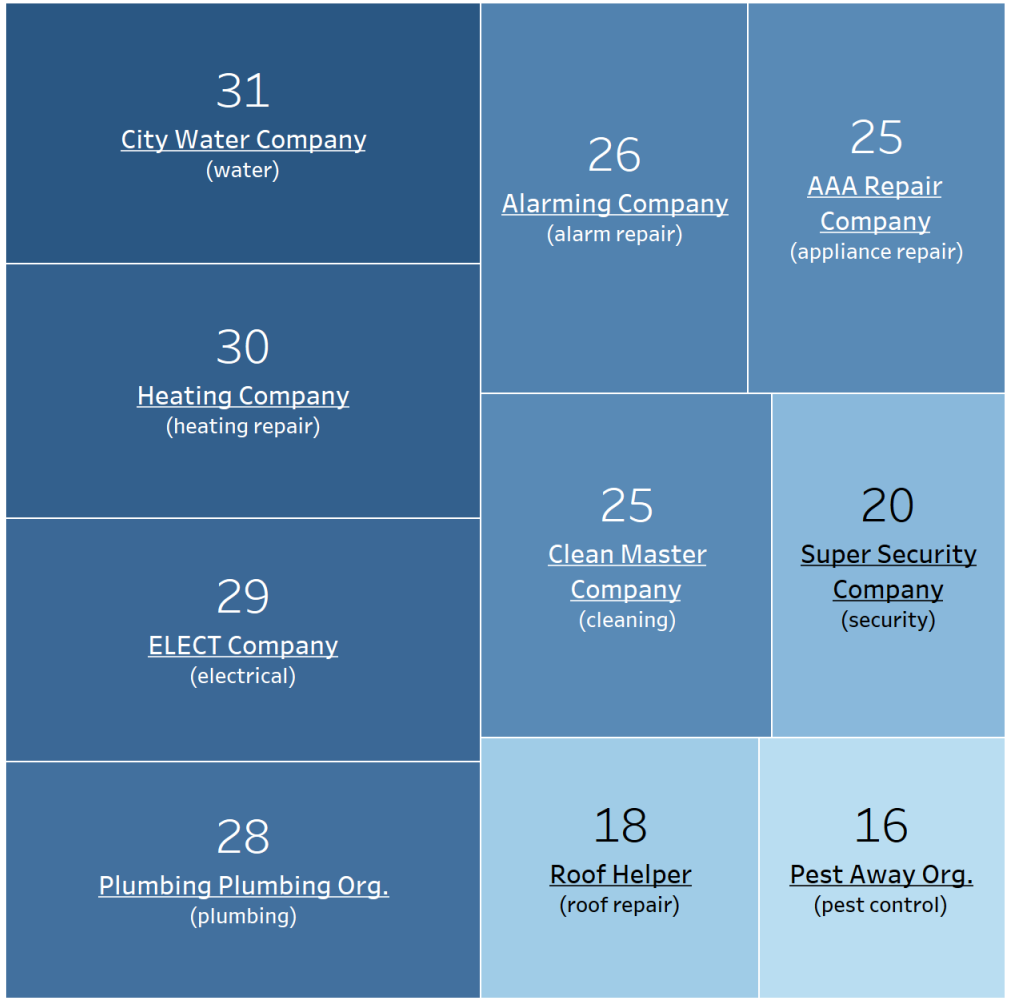
# Visualization Analysis Report 3

This is the times of service, average fees and ratings of each housing service company.

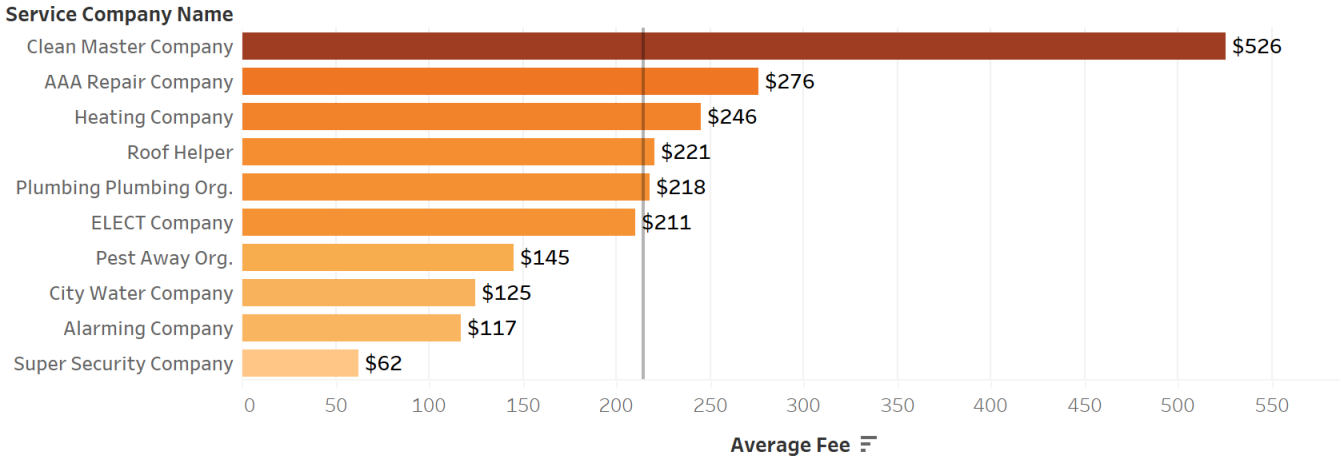
City Water Company has the highest number of requests from tenants and the highest rating with lower fees.

Clean Master Company has the highest fees for their services but the lowest ratings.

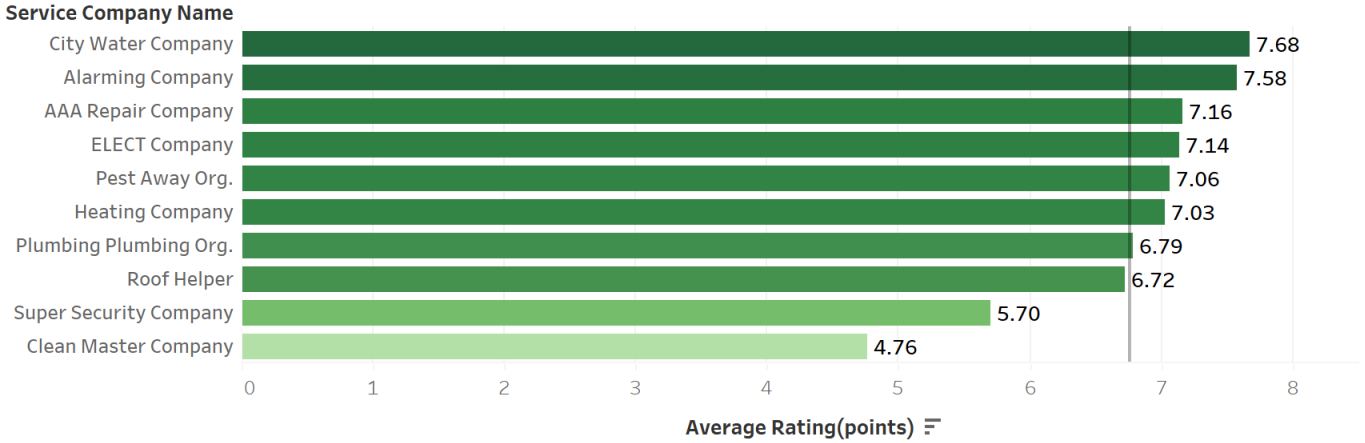
## Service Company Evaluation - Times of Service




## - Average Fee




## - Average Rating



# Visualization Analysis Report 3

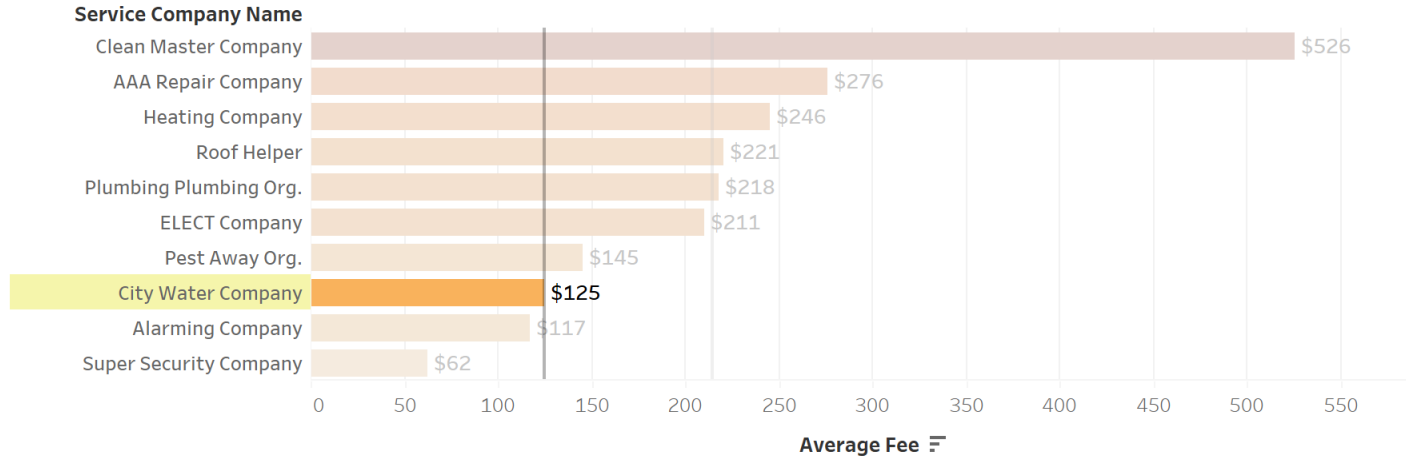
This is the times of service, average fees and ratings of each housing service company.

City Water Company has the highest number of requests from tenants and the highest rating with lower fees.

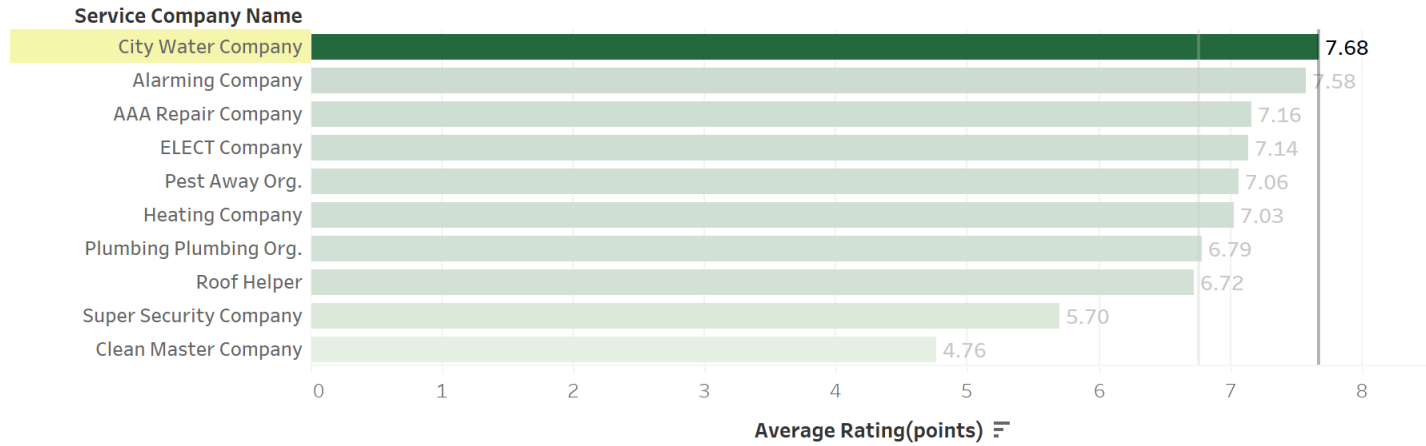
Clean Master Company has the highest fees for their services but the lowest ratings.

## Service Company Evaluation - Times of Service


## - Average Fee




## - Average Rating



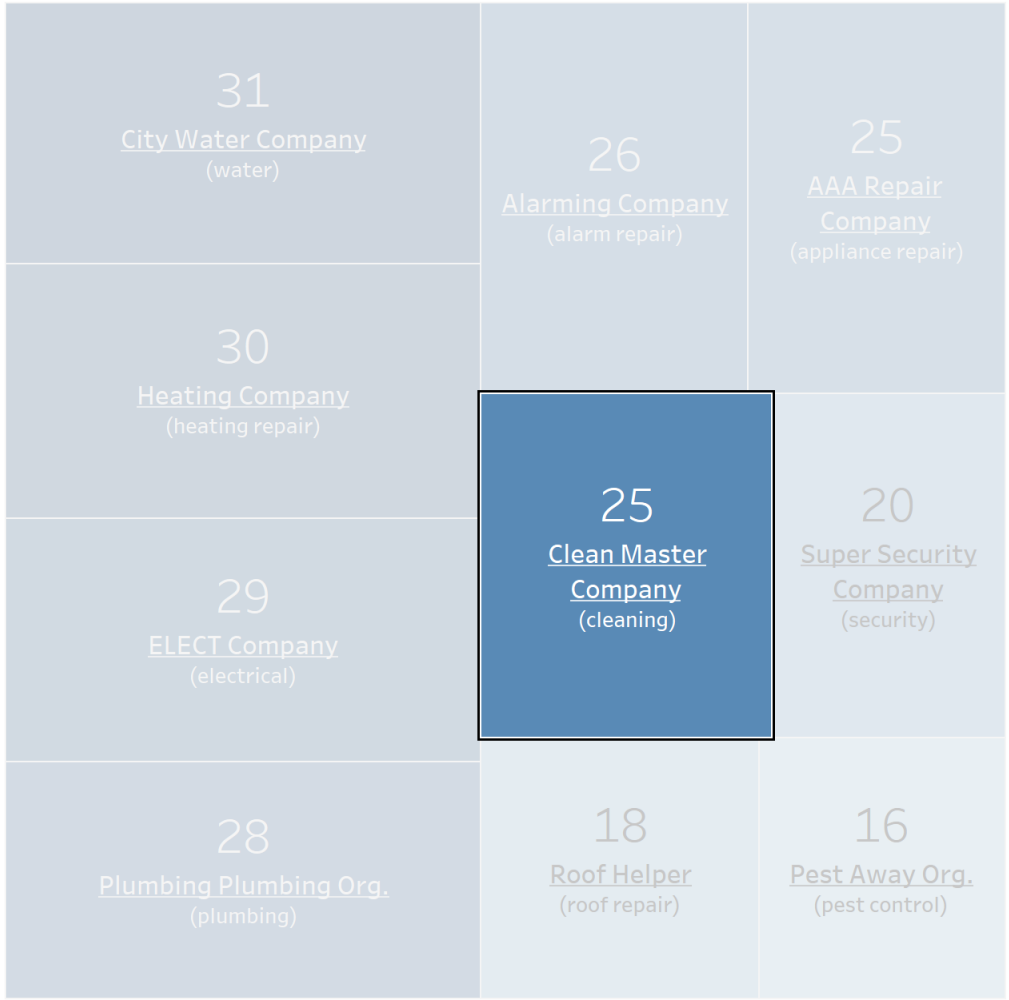
# Visualization Analysis Report 3

This is the times of service, average fees and ratings of each housing service company.

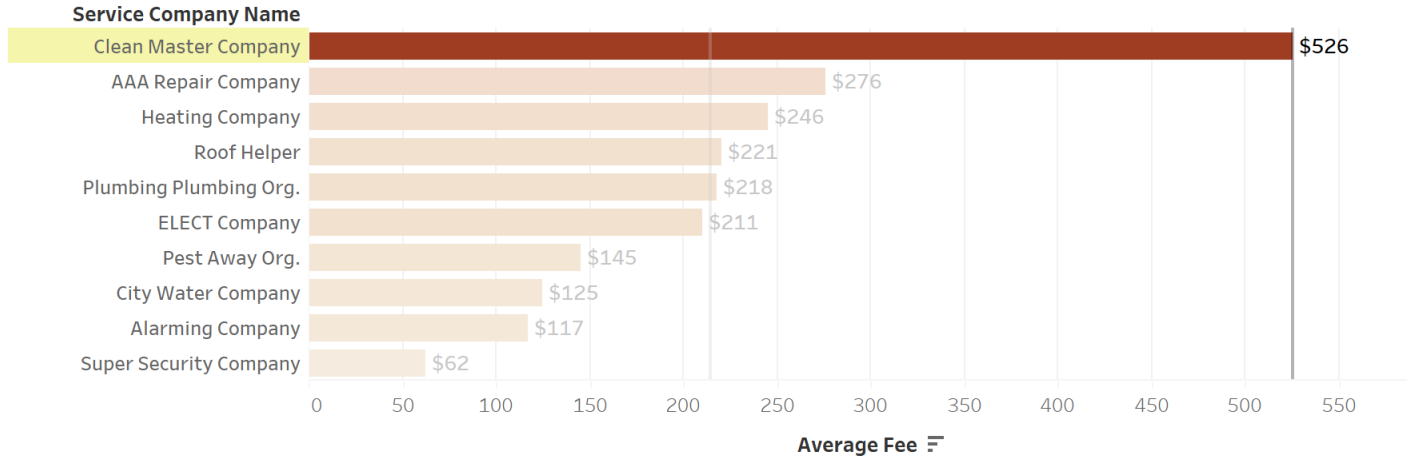
City Water Company has the highest number of requests from tenants and the highest rating with lower fees.

Clean Master Company has the highest fees for their services but the lowest ratings.

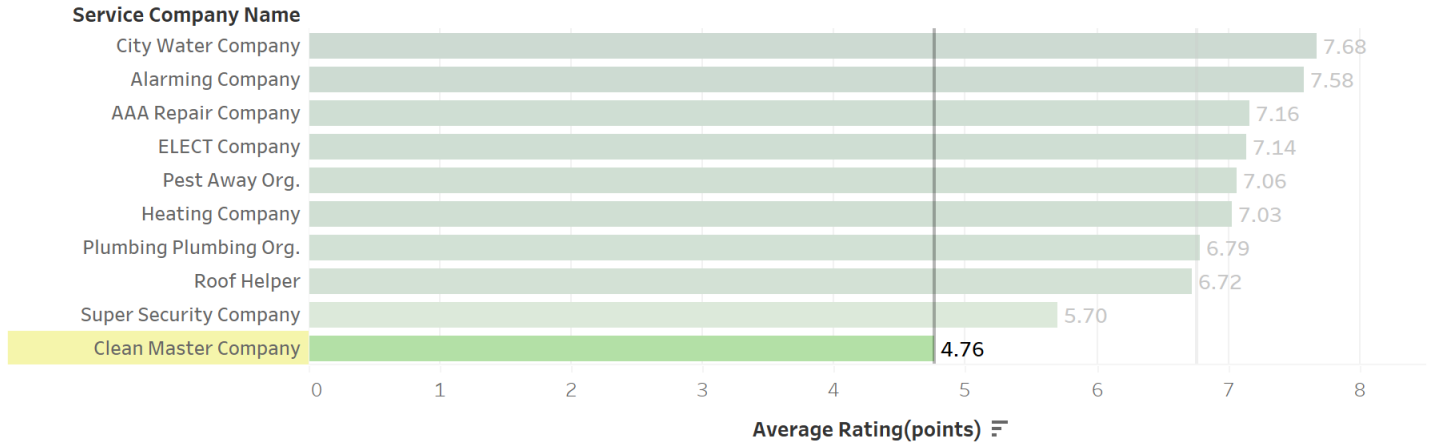
## Service Company Evaluation - Times of Service




## - Average Fee




## - Average Rating



# View 4 - Convention Rate between Application and Visiting History

```
CREATE VIEW application_vs_visit
AS
select t2.*,t3.num_of_visits
from
(select room_id,Room_Type,sum(app_id_not_null) as [num_of_applications]
from
(select r.id as [room_id],
       CONCAT(f.bedroom_count,'B',bathroom_count,'B') as [Room_Type],
       a.id as [application_id],
       case
           when a.id is null then 0
           else 1 end as [app_id_not_null]
from Room r
left join Application a on a.room_id = r.id
left join Floorplan f on f.id = r.floorplan_id) as temp
group by temp.room_id,temp.Room_Type) as t2

join

(select room_id,sum(visit_id_not_null) as [num_of_visits]
from
(select r.id as [room_id],
       v.id as [visit_id],
       case
           when v.id is null then 0
           else 1 end as [visit_id_not_null]
from Room r
left join VisitingHistory v on v.room_id = r.id ) as temp
group by temp.room_id) as t3 on t3.room_id = t2.room_id;
```

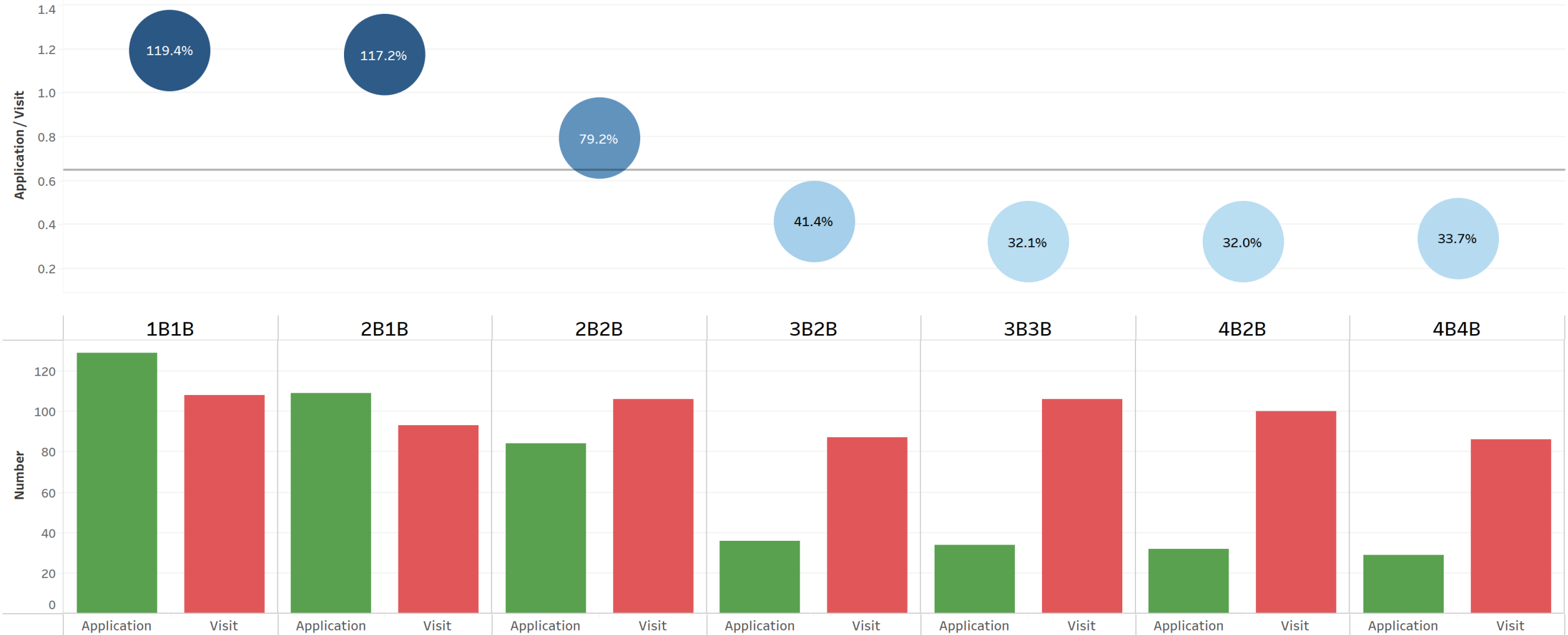
# Visualization Analysis Report 4

The number of applications and visits for each room type are counted.

The conversion rates of 1B1B and 2B1B room type are over 1, indicating the fact that there are many applications without visits.

However, the conversion rates of 3B2B, 3B3B,4B2B and 4B4B room types are below the average.

## Application vs. Visit - By room type dimension



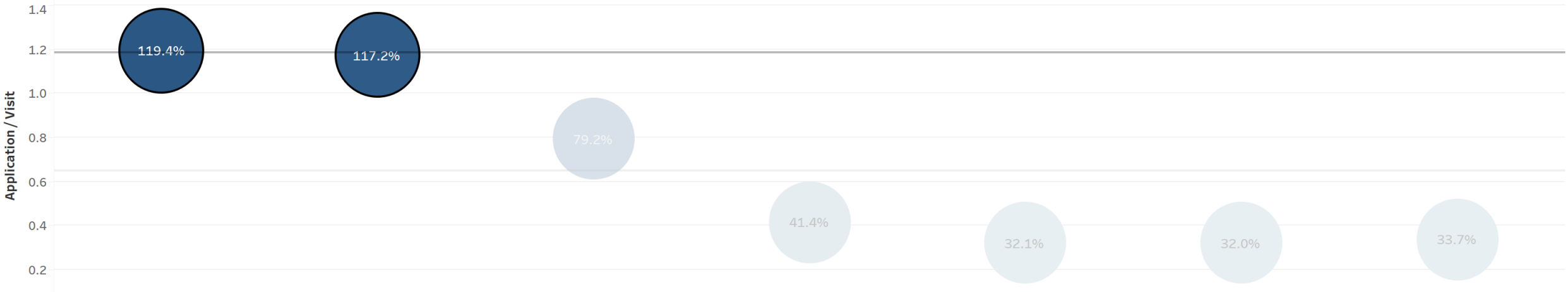
# Visualization Analysis Report 4

The number of applications and visits for each room type are counted.

The conversion rates of 1B1B and 2B1B room type are over 1, indicating the fact that there are many applications without visits.

However, the conversion rates of 3B2B, 3B3B,4B2B and 4B4B room types are below the average.

## Application vs. Visit - By room type dimension



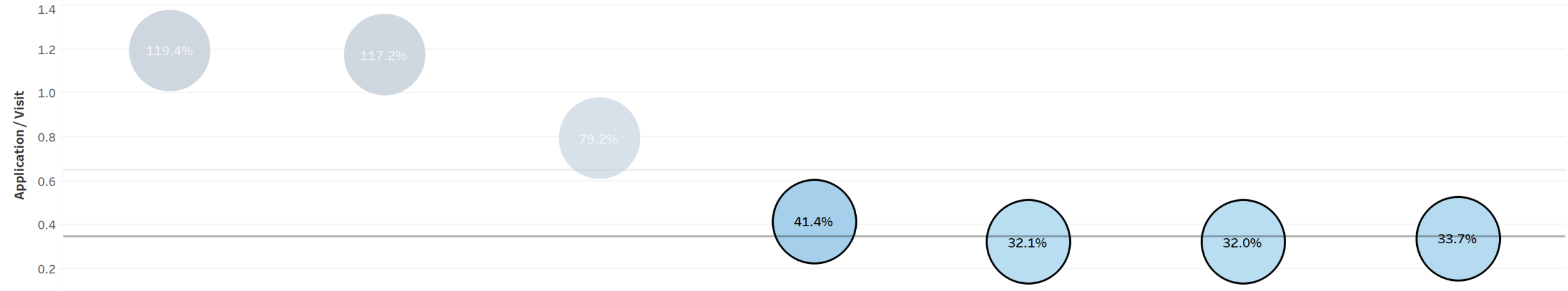

# Visualization Analysis Report 4

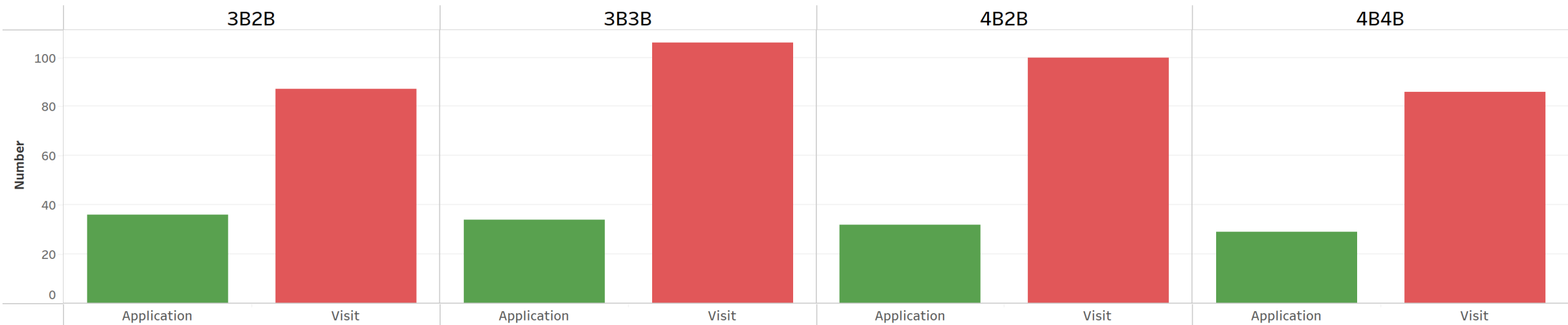
The number of applications and visits for each room type are counted.

The conversion rates of 1B1B and 2B1B room type are over 1, indicating the fact that there are many applications without visits.

However, the conversion rates of 3B2B, 3B3B,4B2B and 4B4B room types are below the average.

## Application vs. Visit - By room type dimension



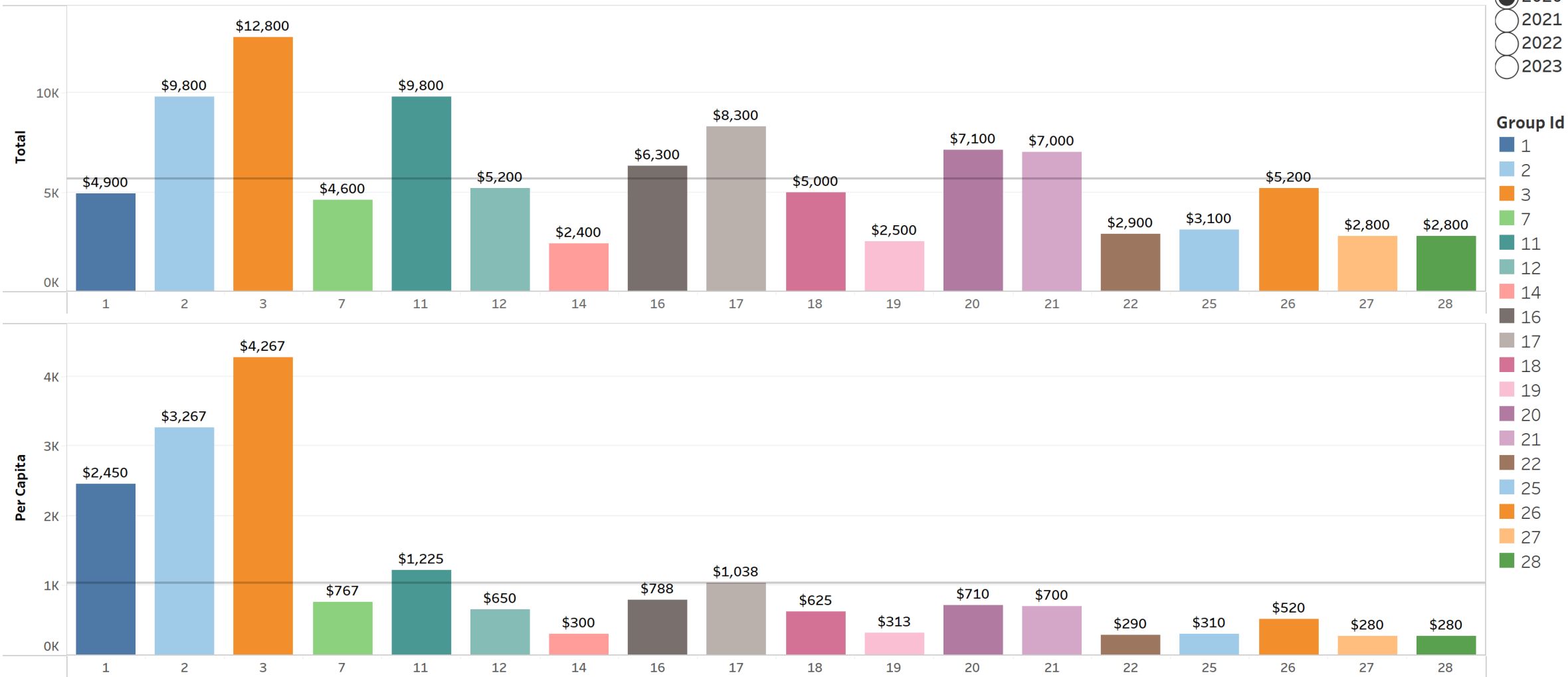
# View 5 - Performance Evaluation

```
create view performanceEvaluation
as
SELECT year(c.start_date) as [year],
       c.id as [contract_id],
       r.id as [room_id],
       c.rent,
       b.group_id as [group_id],
       g.[size] as [group_size]
FROM Contract c
join Room r on r.id = c.room_id
join Building b on b.id = r.building_id
join [Group] g on g.id = b.group_id;
```

# Visualization Analysis Report 5

| This is the total and per capita amount of rental contracts for each group for each year. | For 2021 | For 2022 |
|---|---|---|

## Performance Evaluation - By group dimension(based on contract rent)



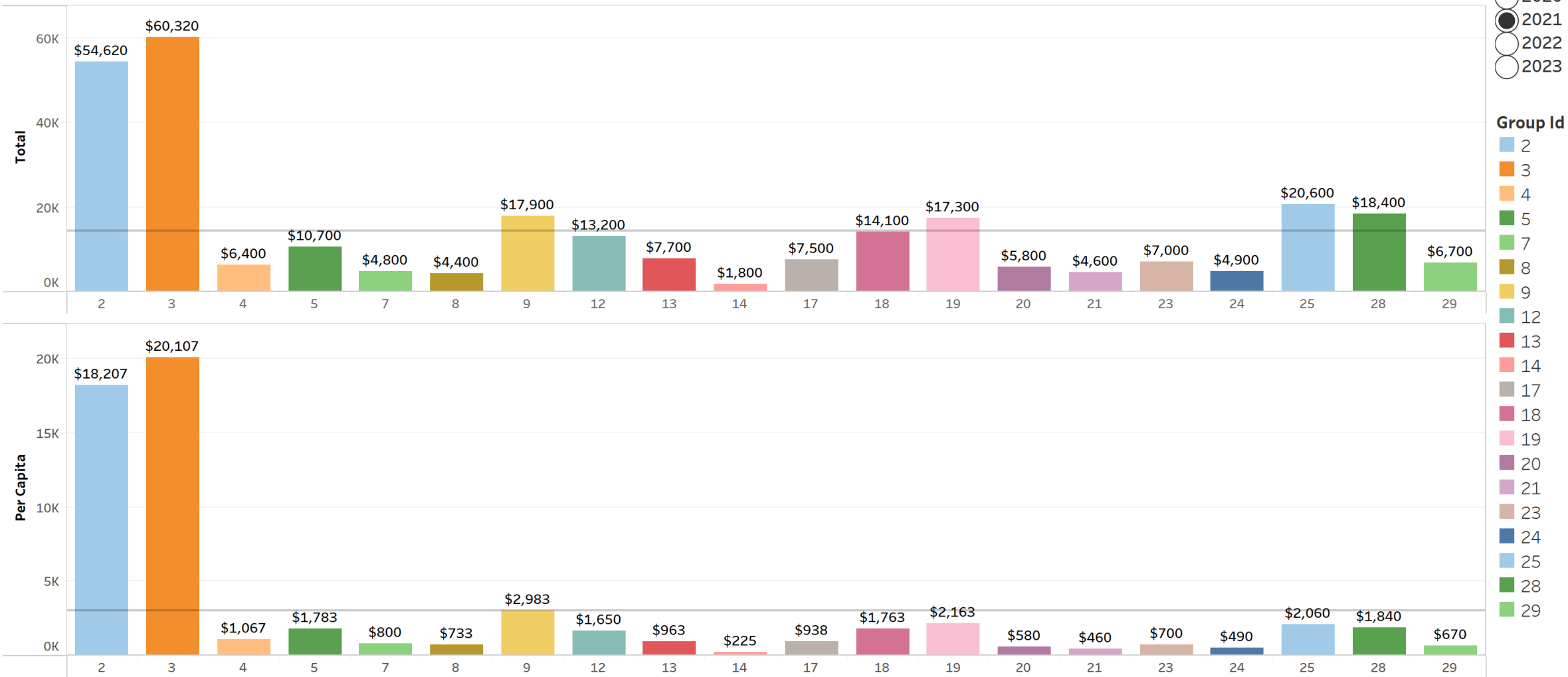
# Visualization Analysis Report 5

This is the total and per capita amount of rental contracts for each group for each year.

For 2021

For 2022

## Performance Evaluation - By group dimension(based on contract rent)



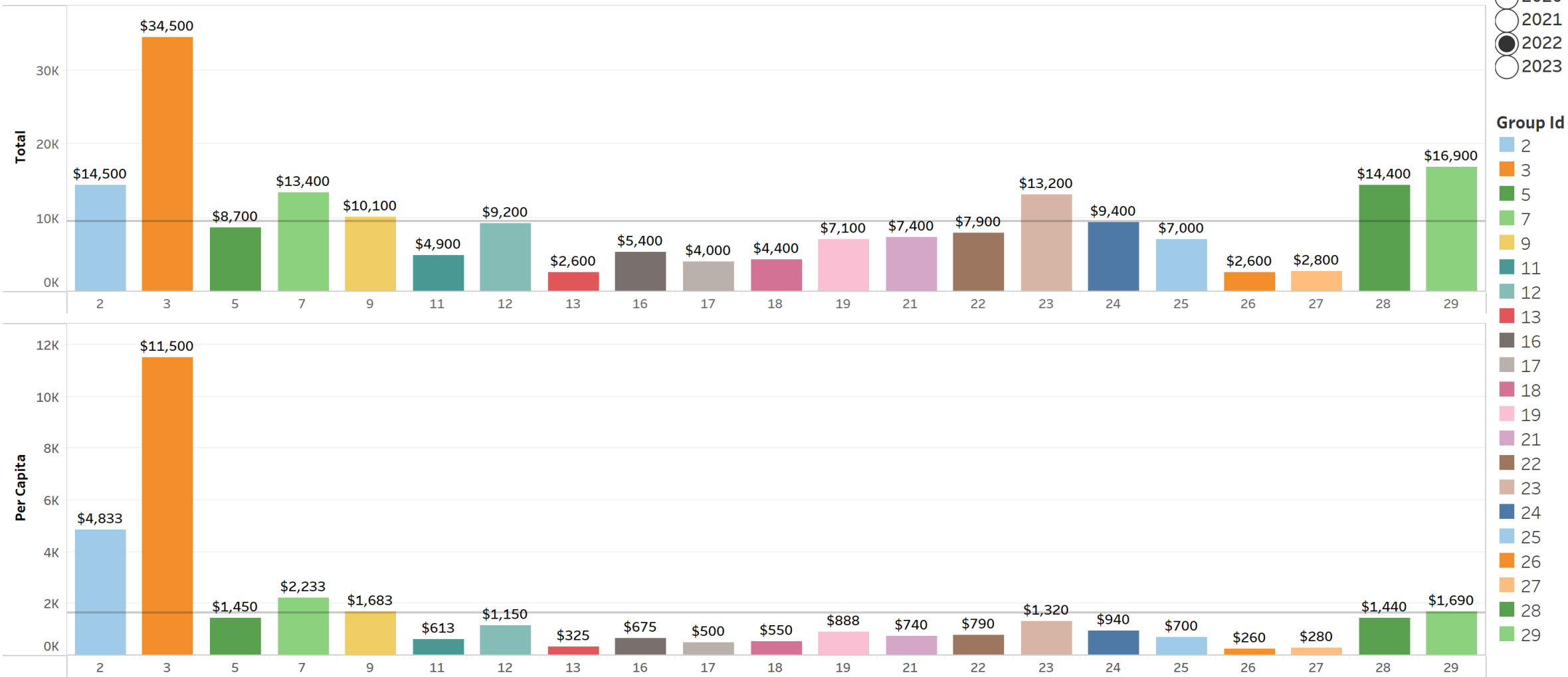
# Visualization Analysis Report 5

This is the total and per capita amount of rental contracts for each group for each year.

For 2021

For 2022

## Performance Evaluation - By group dimension(based on contract rent)



# Thank you!

**Team 8**

**Qian Chen, Rongjing Huang, Xiyue Suo, Yuanyi Xie, Zihan Wan**