NYC Housing Market Analysis

Problem Statement

While the COVID-19 pandemic has affected nearly every aspect of daily life, perhaps the housing market has been affected in the most unexpected way. Early news reports suggested that shutting down COVID-19 would mean the destruction of the U.S. housing market - a market that, for better or worse, is the largest driver of the economy. Instead, government intervention in the form of lower interest rates, wage protection, and other measures contributed to an unprecedented boom in the sector and housing prices, encouraging many renters to become homeowners.

At the same time, many companies that had a work-from-home policy early in the pandemic announced that they would make work-from-home permanent or adopt a hybrid model, with one or two days in the office and the rest at home. This opened up a new world of housing options for families who moved to the cities or near major towns. It has also led to increased demand in suburban and rural areas, which are generally unaffected by rising home prices.

During the Covid-19 pandemic, rents and sales in the United States have fallen sharply, especially in pricey markets such as New York and Boston. Last year, New York's real estate industry was on the ropes since many New Yorks fled to the suburbs. To reduce losses, real estate companies are cutting rents and giving away free months to attract tenants, which has led to a downturn in the rental market. Also, house prices in various boroughs of New York have also fallen to varying degrees. However, as the outbreak improves and more people flock to big cities like New York, the house prices and rents are even higher than before the Covid outbreak, especially in Manhattan. Rental prices have increased by nearly 20% which is very expensive for people to afford (Zumper). New York house prices also increased by 1% to 10% in varying degrees (Santarelli, 2022). Our group is a technical team working for a real estate company in New York, and we want to use the analytical knowledge and techniques to achieve the goals of our company. The ultimate goal of our company is to increase our profits in the following years.

Proposal

To achieve this goal, our group first searches for the housing data in New York City and collects articles that are related to the housing market in New York. After comparing the data from various sources, our group decided to use the data source from StreetEasy, which is a widely used website for customers who are looking for houses. We plan to establish our database system based on the data we found by merging datasets, normalizing the data tables from 1NF to 3NF, and using query tools to analyze. Our dataset contains areas, boroughs, median asking price, inventory, bedroom type, property type, rent cuts, and so on from both rental and sales.

Our project aims to give the company a general idea of what the housing market in New York City looks like under the pandemic, such as the rental and sales situation in each area, and the rental and sales inventory of various regions of New York City. Furthermore, based on our analysis of sales and rental inventory in NYC, our company can gain an insight into which areas or types are worth investing in to help our company get more profits. To be illustrated, the sales

median asking price can somehow reveal the value of this house in people's minds, and the sales inventory can represent whether a certain type of room is popular in a certain area. Real estate companies can use our reports to introduce customers to properties in areas that fit their budgets, and also come up with a better property investment plan for the next fiscal year. Furthermore, our project can also help local workers and students who want to rent in New York City to choose the properties they are interested in based on the rental prices in different areas.

To implement this process, we first design a relational database schema for our project. Besides the technical model, we also come up with a conceptual model that contains not only the rental and sales data we have but also the information of renters, sellers, agents, and the insurance companies they corporated with. In this way, our project can do a better job of linking the data resources we have with the real-world housing situation. Second, we plan to use Python to do the normalization and ETL process. Then, we will design the queries to resolve questions in the following aspects.

Area Related

The location of an investment property is one of the most essential factors in its profitability and potential returns. When we talk about location, we are not referring to a city, but to the specific area in which the investment is made. Although it is beneficial to know how a city's real estate market is performing, it doesn't provide the most accurate information. Each city consists of several local real estate markets, some of which are suitable for real estate investment, but some of them are not. Therefore, the first step in housing market analysis is an assessment and evaluation of the neighborhood area. Real estate investors should always make sure that the neighborhoods they want to buy in are not only good but also desirable. There are some key points that can help attract good tenants and increase the value of your investment property.

As a result, we have formed a table called ‘area’, in which the column ‘area_name’ stands exactly for the location of different neighborhoods in NYC. By analyzing how the factor of area affects the inventory and price of both rentals and sales, we can help our company to get a full picture of whether buying a property in this area will provide a good return.

Type Related

Once we have known the area and decided that it is a profitable place to start investing in housing properties, the next step in analyzing the housing market is to look for comparable

properties. Comparable properties help real estate investors determine the general housing situation in an area. Although no property can be considered completely comparable, we try to develop a general comparison among different bedroom types for the rental properties, and also among various property types for the properties for sale.

Consequently, we have developed two tables called ‘room’ and ‘house’, in which the columns ‘bedroom_type’ and ‘property_type’ refer to the features of the property across different neighborhoods in NYC. By assessing how those features influence the inventory and price of both rentals and sales, we can help our company to get more insights into which type of

bedroom/property attracts the greatest number of customers, and whether buying a property of this type will lead to a big profit in the long run.

Finally, we will draw the conclusions from the analysis we have made, and visualize the results into Metabase Dashboards to help the audience understand our project more intuitively.

1NF of Our Datasets

Normalization

As we said before, the datasets used for this study are all from StreetEasy.com. From StreetEasy, we downloaded the three rent datasets in NYC with all bedroom counts and five sales datasets in NYC with all home types, including inventory, median asking price, median sales price, the share of price cuts, etc.

After we downloaded the datasets, we found that the dates of all values are stored in columns instead of rows. Therefore, the first step to merging all separate datasets together is to convert date columns into rows. Through the melt function, we can easily convert the date value into a single column with the specific month in each row. Then, we merge the separate datasets into two main datasets - rent and sales. The following code is an example of how we use melt function to merge datasets together:

```
sales_inventory = sales_inventory.melt(id_vars =['areaName','Borough','areaType','PropertyType'])
sales_inventory = sales_inventory.rename(columns = {"variable" : "date", "value" : "sales_inventory"})
```

After merging the separate datasets together,
The rent dataset, based on Figure 1, contains areaname, borough, areatype, bedroomtype, date, rent_inventory, rent_asking_price, and rent_price_cut. The sales dataset, based on Figure 2, contains areaname, borough, areatype, propertytype, date, sales_inventory, sales_recorded, sales_price, sales_asking_price, and sales_price_cut.

Figure 1: 1NF of Our Datasets Rent Dataset and its Attributes

Sales Dataset and its Attributes

According to StreetEasy, the column rent_inventory contains the number of rental listings available on StreeEasy. The rent_asking_price is the exact middle asking rent among all asking rents of units listed during the given period. The rent_price_cut is the percent of total rental

	areaName	Borough	areaType	BedroomType	date	rent_inventory	rent_asking_price	rent_price_cut
0	All Downtown	Manhattan	submarket	Studio	2020-01	1062.0	2995.0	0.169
1	All Midtown	Manhattan	submarket	Studio	2020-01	817.0	2695.0	0.162
2	All Upper East Side	Manhattan	submarket	Studio	2020-01	435.0	2250.0	0.110
3	All Upper Manhattan	Manhattan	submarket	Studio	2020-01	229.0	1795.0	0.109
4	All Upper West Side	Manhattan	submarket	Studio	2020-01	352.0	2475.0	0.250

	areaName	Borough	areaType	PropertyType	date	sales_inventory	sales_recorded	sales_price	sales_asking_price	sales_price_cut
0	All Downtown	Manhattan	submarket	Condo	2020-01	1644.0	156.0	1900000.0	2650000.0	0.099
1	All Midtown	Manhattan	submarket	Condo	2020-01	1055.0	89.0	1357500.0	1995000.0	0.102
2	All Upper East Side	Manhattan	submarket	Condo	2020-01	579.0	66.0	1261404.0	2500000.0	0.138
3	All Upper Manhattan	Manhattan	submarket	Condo	2020-01	304.0	30.0	843111.0	997000.0	0.095
4	All Upper West Side	Manhattan	submarket	Condo	2020-01	486.0	93.0	1660000.0	2492500.0	0.121

listings receiving a cut in the asking rent .posted on StreetEasy. This metric indicates how common discounts are in a given area and time period (StreetEasy, n.d.).

Due to the difference in market patterns, the sales dataset contains more columns, namely sales_recorded and sales_price, compared to the rent dataset. In the rental market, customers can only rent rooms, so it only has inventory records to show. The sales_recorded displays the number of homes sold during the given period based on records from the New York City Department of Finance (StreetEasy, n.d.). There has the median asking price and the median sales price in the sales market. The asking price, also known as the listing price, means how much a seller has listed a property for, while the sales price refers to the amount it actually sells for (*List Price vs Sale Price: Know the Difference*, 2021). The concept of median sales price, which is the sales_price column in the sales dataset, is defined by StreetEasy as the exact middle price among all final, recorded sales prices for homes that closed during the given period (StreetEasy, n.d.).

3NF of Our Datasets

After we merge our separate CSV files into 2 main datasets - rent and sales, we start to do the normalization and ETL based on our ERD graph. Because our dataset does not include information on sellers, renters, sales agents, rental agents, and insurance, we do not include these tables during the normalization process. We did, however, create these tables in our SQL database for future research purposes. If the company's research team can provide us with the relevant information, we can load this data into the tables we created.

According to our ERD, Our main tables will be 'area', 'room', 'house', 'rent_inventory_info', 'rent_price_info', 'sales_inventory_info', 'sales_price_info', and two relational tables 'area_house' and 'area_room' table. First, consider to ETL process, we convert all column names to lowercase.

For the 'area', 'room', 'house' table, the process of normalization is similar. First, we extract the needed column from the main data frame, drop the duplicate value of each column, and save it in a new data frame. Second, we generate the unique identifier for the table. For 'room' and 'house', since there are just a few rows, we generate incrementing integers as the unique identifier. Since the process of creating the room and house tables is the same, only the code for how we create the area and room tables is shown here:

‘area table’:

```
# extract the area table

area = rent[["areaname","borough","areatype"]]
area.rename(columns = {'areaname':'area_name', 'areatype':'area_type'}, inplace = True) # drop duplicate value in area df
area = area.drop_duplicates(keep='first')

# generate unique identifier for area

area["area_id"] = random.sample(range(100,999),area.shape[0]) # rearrange the order
cols = area.columns.tolist()
cols = cols[-1:] + cols[:-1]

area = area[cols]
```

‘room table’:

```
# create room df with unique bedroom type
```

Then, we insert the unique identifier of each table - 'area_id', 'bedroom_type_id', 'property_type_id' back to the main data frame 'rent' and 'sales' for next steps.

```
# Map area_id
```

After that, we generate 'rent_id' and 'sales_id' as the unique identifier through random sample function for 'rent_inventory_info', 'rent_price_info', 'sales_inventory_info', 'sales_price_info' tables. Next, we extract the needed columns for each table with the foreign key information (e.g. 'area_id') included in the table. Take the code of rent_inventory_info as an example to show how we can accomplish this step:

```
# generate rent_id for rent inventory, median asking price, price cut

rent["rent_id"] = random.sample(range(10000,99999),rent.shape[0])
```

To build the relation table between two tables('area_room' and 'area_house'), we extract the unique identifier of the related data frame and save it together, the results are shown on Table 8 and 9. After finishing the normalization, we load our data to PgAdmin through ETL.

Figure 2: 3NF of Our Dataset

```
room = pd.DataFrame(rent.bedroomtype.unique(), columns=['bedroomtype']) room.rename(columns = {'bedroomtype':'bedroom_type'}, inplace = True)
# add incrementing integers
room.insert(0, 'bedroom_type_id', range(1, 1 + len(room)))
```

```
area_id_list = [area.area_id[area.area_name == i].values[0] for i in rent.areaname] # add area_id to rent data frame
rent.insert(0, 'area_id', area_id_list)

rent.rename(columns = {'rent_asking_price':'rent_median_asking_price'}, inplace = True)
# extract rent_inventory to rent_inventory_info df
rent_inventory_info = rent[["rent_id","date","rent_inventory","area_id","bedroom_type_id"]]
```

'area table' and its attributes

'house' table and its attributes

'rent_price_info' table and its attributes

'sales_inventory_info' table and its attributes

'sales_price_info' table and its attributes

'room' table and its attributes

'rent_inventory_info' table and its attributes

	area_id	area_name	borough	area_type
0	974	All Downtown	Manhattan	submarket
1	263	All Midtown	Manhattan	submarket
2	966	All Upper East Side	Manhattan	submarket
3	701	All Upper Manhattan	Manhattan	submarket
4	638	All Upper West Side	Manhattan	submarket

	bedroom_type_id	bedroom_type
0	1	Studio
1	2	One bed
2	3	Two beds
3	4	Three or more beds

	rent_id	date	rent_inventory	area_id	bedroom_type_id
0	12855	2020-01	1062.0	974	1
1	14064	2020-01	817.0	263	1
2	82948	2020-01	435.0	966	1
3	63265	2020-01	229.0	701	1
4	19356	2020-01	352.0	638	1

	property_type_id	property_type
0	11	Condo
1	12	Co-Op
2	13	Single Family/Townhouse

	rent_id	date	rent_median_asking_price	rent_price_cut	area_id	bedroom_type_id
0	12855	2020-01	2995.0	0.169	974	1
1	14064	2020-01	2695.0	0.162	263	1
2	82948	2020-01	2250.0	0.110	966	1
3	63265	2020-01	1795.0	0.109	701	1
4	19356	2020-01	2475.0	0.250	638	1

	sales_id	date	sales_inventory	sales_recorded	area_id	property_type_id
0	61232	2020-01	1644.0	156.0	974	11
1	39199	2020-01	1055.0	89.0	263	11
2	23892	2020-01	579.0	66.0	966	11
3	77462	2020-01	304.0	30.0	701	11
4	85025	2020-01	486.0	93.0	638	11

	sales_id	date	median_sales_price	sales_median_asking_price	sales_price_cut	area_id	property_type_id
0	61232	2020-01	1900000.0	2650000.0	0.099	974	11
1	39199	2020-01	1357500.0	1995000.0	0.102	263	11
2	23892	2020-01	1261404.0	2500000.0	0.138	966	11
3	77462	2020-01	843111.0	997000.0	0.095	701	11
4	85025	2020-01	1660000.0	2492500.0	0.121	638	11

‘area_room’ table and its attributes ‘area_house’ table and its attributes

ETL Process

ETL stands for extracting data from the original source system into the staging area, transforming data, and loading newly transformed data into the database. Before we can do the ETL, we need to connect Python to PostgreSQL. In this process, we create an engine used to connect with PostgreSQL and establish a connection. The following code shows how we build the connection between Python and PostgreSQL:

```
conn_url='postgresql://pgsql4:columbia4@pgsql4.c0ttp4tkwo66.us-east-1.rds.amazonaws.com:54
32/pgsql4'
engine = create_engine(conn_url)
connection = engine.connect()
```

Furthermore, the second step is to use SQL statements to create tables. The first part creates the tables in which we have the actual data, and the second query creates the tables based on the conceptual model. The following code is the query of creating tables with actual data:

```
stmt = """ CREATE TABLE area( area_id integer,

area_name varchar(100) NOT NULL, borough varchar(100),
area_type varchar(100),
primary key (area_id)

);
CREATE TABLE room(

bedroom_type_id integer,
bedroom_type varchar(100) NOT NULL, primary key (bedroom_type_id)
);
```

CREATE TABLE house(
property_type_id integer,
property_type varchar(100) NOT NULL,

	area_id	property_type_id
0	944	11
1	970	11
2	656	11
3	424	11
4	223	11

	area_id	bedroom_type_id
0	944	1
1	970	1
2	656	1
3	424	1
4	223	1

primary key (property_type_id)

);
CREATE TABLE rent_inventory_info(

rent_id integer,
date varchar(7) NOT NULL,
rent_inventory integer,
area_id integer,
bedroom_type_id integer,
primary key (rent_id),
foreign key (area_id) references area(area_id),
foreign key (bedroom_type_id) references room(bedroom_type_id));

CREATE TABLE rent_price_info(rent_id integer,

date varchar(7) NOT NULL, rent_median_asking_price integer, rent_price_cut numeric(4,3), area_id integer,

bedroom_type_id integer,
primary key (rent_id),
foreign key (area_id) references area(area_id),
foreign key (bedroom_type_id) references room(bedroom_type_id));

CREATE TABLE sales_inventory_info(sales_id integer,

date varchar(7) NOT NULL,
sales_inventory integer,
sales_recorded integer,
area_id integer,
property_type_id integer,
primary key (sales_id),
foreign key (area_id) references area(area_id),
foreign key (property_type_id) references house(property_type_id));

CREATE TABLE sales_price_info(sales_id integer,

```
date varchar(7) NOT NULL, median  sales  price integer, sales  median  asking  price integer,
sales_price_cut numeric(4,3), area_id integer,

property_type_id integer,
primary key (sales_id),
foreign key (area_id) references area(area_id),
foreign key (property_type_id) references house(property_type_id));

CREATE TABLE area_room( area_id integer,

bedroom_type_id integer,
foreign key (bedroom_type_id) references room(bedroom_type_id), foreign key (area_id) references
area(area_id));

CREATE TABLE area_house( area_id integer,

property_type_id integer,
foreign key (property_type_id) references house(property_type_id), foreign key (area_id) references
area(area_id))

""" connection.execute(stmt)
```

The first table is called ‘area’, ‘area_id’ is the primary key of this table and its class is an integer that is generated in the normalization part. The rest of the attributes are in varchar datatype since they are varying characters and ‘area_name’ should not contain null values. The primary key of table ‘room’ is ‘bedroom_type_id’. The class of ‘bedroom_type_id’ is an integer that is generated randomly and ‘bedroom_type’ is varying characters that should not contain null values. The third table is ‘house’ and ‘property_type_id’ is the primary key for this table and the data type is an integer. Another attribute in this table is ‘property_type’ which is a varying character and should not contain null values. The primary key of ‘rent_inventory_info’ is ‘rent_id’, and the foreign keys are ‘area_id’ and ‘bedroom_type_id’. ‘area_id’ references table ‘area’ and ‘bedroom_type_id’ references table ‘room’. The fifth table is ‘rent_price_info’ and its primary key is ‘rent_id’. The foreign key ‘area_id’ references table ‘area’ and ‘bedroom_type_id’ references table ‘room’. Table ‘sales_inventory_info’ has a primary key ‘sales_id’ and two foreign keys. The ‘area_id’ references table area and ‘property_type_id’ references table house. The following table is ‘sales_price_info’ and the primary key is ‘sales_id’. The foreign keys are ‘area_id’ references table ‘area’, ‘property_type_id’ references table ‘house’. The ‘area_room’ is a junction table that connects tables ‘room’ and ‘area’ using ‘bedroom_type_id’ and ‘area_id’. Another junction table is ‘area_house’ which connects tables ‘house’ and ‘area’. The foreign keys are ‘property_type_id’ and ‘area_id’. And the following code shows tables we built in a conceptual model.

```
stmt= """ CREATE TABLE insurance( insurance_company_id integer,

insurance_company_name varchar(100) NOT NULL,

primary key (insurance_company_id)); CREATE TABLE renter_building(
```

```
renter  id integer,
renter_name varchar(100) NOT NULL,

renter_location text NOT NULL, elevator boolean,
gym boolean,
doorman boolean, distance_to_MTA numeric(6,2), insurance_company_id integer, primary key
(renter_id),

foreign key (insurance_company_id) references

insurance(insurance_company_id)); CREATE TABLE seller_building(

seller_id integer,
seller_name varchar(100) NOT NULL, seller_location text NOT NULL, elevator boolean,
distance_to_MTA numeric(6,2), amenity_fee numeric(12,2), building_age integer,
down_payment numeric(12,2), primary key (seller_id));

CREATE TABLE building_agent_rent( agent_id integer,

renter_id integer NOT NULL,
agent_name varchar(100) NOT NULL,
phone_number varchar(10) NOT NULL,
fax_number varchar(10) NOT NULL,
primary key (agent_id),
foreign key (renter_id) references renter_building(renter_id));

CREATE TABLE building_agent_sales( agent_id integer,

seller_id integer NOT NULL,
agent_name varchar(100) NOT NULL,
phone_number varchar(10) NOT NULL,
fax_number varchar(10) NOT NULL,
primary key (agent_id),
foreign key (seller_id) references seller_building(seller_id));

CREATE TABLE room_renter(
bedroom_type_id integer,
renter_id integer,
foreign key (bedroom_type_id) references room(bedroom_type_id), foreign key (renter_id) references
renter_building(renter_id));

CREATE TABLE house_seller( seller_id integer,

property_type_id integer,
foreign key (property_type_id) references house(property_type_id), foreign key (seller_id) references
seller_building(seller_id))

""" connection.execute(stmt)
```

Our conceptual model contains seven tables including: ‘insurance’, ‘renter_building’, seller_building’, ‘building_agent_rent’, ‘building_agent_sales’, ‘bedroom_renter’ and ‘seller_property’. Table ‘insurance’ contains the primary key ‘insurance_company_id’ and ‘insurance_company_name’. Table ‘renter_building’ has a primary key ‘renter_id’ and a foreign key ‘insurance_company_id’ references table insurance. The primary key of table ‘seller_building’ is ‘seller_id’. Table ‘building_agent_rent’ has a primary key ‘agent_id’ and a foreign key ‘renter_id’ references table ‘renter_building’. The primary key of table ‘building_agent_sales’ has a primary key ‘agent_id’ and a foreign key ‘seller_id’ references table ‘seller_building’. Furthermore, the following two tables are junction tables ‘bedroom_renter’ and ‘seller_property’. In ‘room_renter’, ‘bedroom_type_id’ is used to connect with table ‘room’ and ‘renter_id’ is used to refer to the table ‘renter_building’. In the ‘house_seller’ table, foreign key ‘property_type_id’ references table ‘house’ and ‘seller_id’ references table ‘seller_building’.

The final process of ETL is to load data into PostgreSQL through the to_sql function. The following code shows the inserting process:

area.to_sql(name='area', con=engine, if_exists='append', index=**False**) room.to_sql(name='room', con=engine, if_exists='append', index=**False**)

house.to_sql(name='house', con=engine, if_exists='append', index=**False**)
rent_inventory_info.to_sql(name='rent_inventory_info', con=engine, if_exists='append', index=**False**)
rent_price_info.to_sql(name='rent_price_info', con=engine, if_exists='append', index=**False**)
sales_inventory_info.to_sql(name='sales_inventory_info', con=engine, if_exists='append', index=**False**)
sales_price_info.to_sql(name='sales_price_info', con=engine, if_exists='append', index=**False**)
area_room.to_sql(name='area_room', con=engine, if_exists='append', index=**False**)
area_house.to_sql(name='area_house', con=engine, if_exists='append', index=**False**)

Analytical Insights

Procedures Analytical Procedures

Our team aims to help companies gain insight into which areas or types of rooms or homes are worth investing in the rental market and the sales market. We conduct the analysis of each market separately. We first investigate the overall market performance of the rental and sales

market by comparing price changes between January 2020 and January 2021 and between January 2021 and February 2022. We then drill down to discover which typical areas, bedroom types and property types are the most valuable to invest in by looking at the median rent, rental inventory, median sales, median sales price and price cut columns. Here we only show 4 queries and results, since it would be too long to show them all.

The complete query and related outputs can be accessed through this link

https://docs.google.com/document/d/12plUf8FqTA_Plesfu3BtLk_mIRmkzBBZMM5IpDh5cRg/edit?usp=sharing

The SQL file with all queries can be accessed and downloaded through this link:

https://drive.google.com/file/d/1cO_MUoMl5t2N3QTlMWTmAkbcM_dK5aov/view?usp=sharing

What is the median price of rent per bedroom type in 2020-01 (highest point before Covid) compared to 2021-01 (lowest point)?

```
WITH before_covid AS
(SELECT bedroom_type_id, round(AVG(rent_median_asking_price),2) before_covid FROM
rent_price_info rp
WHERE rp.date = '2020-01'
GROUP BY bedroom_type_id),
lowest AS
(SELECT bedroom_type_id, round(AVG(rent_median_asking_price),2) lowest FROM rent_price_info rp
WHERE rp.date = '2021-01'
GROUP BY bedroom_type_id)

SELECT bedroom_type, before_covid, lowest, before_covid - lowest AS diff, round((before_covid -
lowest)/before_covid*100,2) drop_perc
FROM before_covid bc
LEFT JOIN lowest l

USING (bedroom_type_id) LEFT JOIN room r USING (bedroom_type_id)
```

	bedroom_type character varying (100)	before_covid numeric	lowest numeric	diff numeric	drop_perc numeric
1	Studio	2189.55	1892.40	297.15	13.57
2	One bed	2519.57	2198.67	320.90	12.74
3	Two beds	3372.43	2900.43	472.00	14.00
4	Three or more beds	5156.50	4162.79	993.71	19.27

Is there any overlap between the area with the highest rent median asking price and the area with highest inventory?

```
SELECT i.area_id, i.area_name, i.borough, i.total_inventory, p.avg_rent FROM

(SELECT r.area_id, r.area_name, rt.total_inventory, r.borough FROM(
SELECT ri.area_id, sum(ri.rent_inventory) AS total_inventory FROM rent_inventory_info AS ri

WHERE ri.rent_inventory IS NOT null GROUP BY ri.area_id) rt
LEFT JOIN AREA AS r
ON rt.area_id = r.area_id

WHERE area_type = 'neighborhood' ORDER BY rt.total_inventory DESC LIMIT 20) AS i
```

```
INNER JOIN (
SELECT r.area_id, r.area_name, rt.avg_rent
FROM(
SELECT rp.area_id, round(AVG(rp.rent_median_asking_price),2) AS avg_rent FROM rent_price_info
AS rp
WHERE rp.rent_median_asking_price IS NOT null
GROUP BY rp.area_id) rt
LEFT JOIN AREA AS r
ON rt.area_id = r.area_id
WHERE area_type = 'neighborhood'
ORDER BY rt.avg_rent DESC
LIMIT 20) AS p

ON i.area_id = p.area_id
```

area_id [PK] integer	area_name character varying (100)	borough character varying (100)	total_inventory bigint	avg_rent numeric
218	Upper East Side	Manhattan	97383	4112.75
487	Upper West Side	Manhattan	83533	4204.95
203	Midtown East	Manhattan	68529	4017.41
315	Midtown West	Manhattan	47039	3878.90
822	Chelsea	Manhattan	29381	5236.95
215	Long Island City	Queens	24005	3867.06
386	Financial District	Manhattan	23218	4682.44
952	West Village	Manhattan	19005	4575.46

What is the most popular property type by recorded sales?

```
SELECT h.property_type_id, h.property_type, i.total_records FROM(
SELECT property_type_id, sum(sales_recorded) AS total_records FROM sales_inventory_info AS ri

WHERE ri.sales_recorded IS NOT null GROUP BY property_type_id) i
LEFT JOIN house AS h
ON h.property_type_id = i. property_type_id
```

Is there any overlap between the area with the highest median sales price and the area with highest recorded sales?

```
SELECT i.area_id, i.area_name, i.borough, p.total_recorded, i.avg_sales FROM (SELECT r.area_id,
r.area_name, r. borough, rt.avg_sales

FROM(
SELECT sp.area_id, round(AVG(sp.median_sales_price),2) AS avg_sales FROM sales_price_info AS sp
WHERE sp.median_sales_price IS NOT null
GROUP BY sp.area_id) rt
LEFT JOIN AREA AS r
ON rt.area_id = r.area_id
WHERE area_type = 'neighborhood'
```

```
ORDER BY rt.avg  sales DESC
LIMIT 20) AS i

INNER JOIN (
SELECT r.area_id, r.area_name, rt.total_recorded
FROM(
SELECT ri.area_id, sum(ri.sales_recorded) AS total_recorded FROM sales_inventory_info AS ri
WHERE ri.sales_recorded IS NOT null
GROUP BY ri.area_id) rt
LEFT JOIN AREA AS r
ON rt.area_id = r.area_id
WHERE area_type = 'neighborhood'
ORDER BY rt.total_recorded DESC
LIMIT 20) AS p
```

	property_type_id [PK] integer	property_type character varying (100)	total_records bigint
1	11	Condo	92324
2	12	Co-Op	114452
3	13	Single Family/Townhouse	146661

```
ON i.area_id = p.area_id
```

Insights

From the brief overview of the rental market, we noticed the median asking price had dropped drastically since the beginning of 2020. The downward trend continued until the end of 2020. At the beginning of 2021, the median asking price started to rise until today. The change in prices is directly linked to the crisis of COVID-19. The drastic change in price provides us a good opportunity to identify which kind of houses are more likely to preserve their value during a crisis.

Room type is an important factor we need to consider for the investment, especially when each room behaves differently during the pandemic. After we compared all the room types, we found that rooms with three or four bedrooms tend to lose value quicker than others and the rate of increase was also slower, which should be excluded from investment. 1 bedroom room price was the most stable during the crisis, it has higher risk resilience. The studio is more “high-risk high return”: it had the most rapid growth after the crisis.

It is also important that we picked a suitable area to gain the max profit. We examined all the median asking prices and inventory in each area then we chose the area with the highest median asking price and the highest inventory. The area with high inventory is often associated with areas with high population density, it also provides us opportunities to scale the market quicker; areas with higher median asking price have higher potential to gain profit. By analyzing the overlapping region, we selected the 9 best areas: Upper East Side, Upper West Side, Midtown East, Midtown West, Chelsea, Long Island City, Financial District, and West Village. All these

areas are in Manhattan except Long Island City. We also found the area with the most rapid growth after the crisis: Whitestone, Midtown, Soho, East Tremont, Downtown Brooklyn, and Tribeca.

The analysis of the sales market used a similar approach. The price change in the sales market is not as big as in the rental market but is also distinct. Overall, Condo has the least recorded sales

	area_id [PK] integer	area_name character varying (100)	borough character varying (100)	total_recorded bigint	avg_sales numeric
1	986	Greenwich Village	Manhattan	1124	1809940.24
2	211	Park Slope	Brooklyn	1323	1628990.62
3	218	Upper East Side	Manhattan	5765	1440243.04
4	822	Chelsea	Manhattan	1480	1428514.46
5	487	Upper West Side	Manhattan	5255	1328070.85
6	102	Williamsburg	Brooklyn	1968	1275907.20

in the past two years, and it did not perform well during the crisis: it has the biggest drop in price and the lowest increase rate. On the contrary, Single Family/Townhouse had the highest recorded sales and it showed high-risk resilience with the highest profitability after the crisis. It should be our first choice in investment. Upper East Side, Upper West Side, Midtown East, Flushing, and Williamsburg have the highest record sales by region; Fort Greene, Tribeca, Midtown, Soho, and Central Park South have the highest average sales price.

Dashboard

Implement for Analysts and for "C" Level Officers

Our company is a New York-based real estate company. Through cloud sharing, the company's personnel have access to the database system and our analysis results from which they understand the current New York real estate market trends and possible areas and types of properties to put emphasis on.

We will show our results using Tableau for C-level officers. CEOs have to deal with a lot of work every day, so presenting the results of our analysis in an intuitive and vivid way in the form of graphs would greatly enhance the efficiency. The executives can acquire their targeted information such as rent price or inventory amount for specific dates via interactive visual charts created through tableau.

Here is the dashboard for the C-level officers to have an overview of the housing market in NYC:

https://public.tableau.com/views/5310_Group4_Dashboard/HouseMarketAnalysisinNYC?:language=en-US&:display_count=n&:origin=viz_share_link

We will present our results by Metabase for analysts. Through reading the query and code in detail, they can gain a more accurate understanding of our system construction. In addition, through Metabase, a perfect place for analysts to dive into the database and analytical procedures, analysts can run the code for the analytical procedures we designed and even write their own code to do further analysis.

Here is the dashboard for analysts, where they read the query and access the database:

http://localhost:3000/public/dashboard/1bfbd20d-e3f3-448c-90aa-ca65ab684981

Plan for Redundancy and Performance

We processed the initial data by integrating multiple data tables as master data, for example, conforming different room type tables of the same area together. At the same time, we removed categories that were not relevant to the purpose of the research, such as rental and sales

information from several years ago. This operation ensures better data consistency and accuracy, and when the fragment information in it changes, we only need to process and update this part of the data. Secondly, we normalized the data, using the same appearance for the fields to ensure that the data can be read in a specific way. Sometimes data redundancy may also result from inefficient coding, overly complex data storage processes, or unexpected occurrences on the storage server. Therefore, we gradually refined the encoding of queries and used cloud storage to reduce duplicate sharing or the presence of unnecessary fields.

In the future, we will further improve the dashboard and query. For example, the dashboard now uses a line graph format to give customers visual feedback on average rental inventory, prices, etc. The horizontal coordinates repeat the same dates, with a break in the line between various regions due to differences in information. We might be able to optimize the chart by picking a specific date range and displaying the rental inventory and price information vertically for different regions. This approach provides a comparison and reduces the repetition and redundancy of queries.

Apply Cloud Hosting Services

Based on our customers' needs and case scenarios, our team decided to adopt cloud hosting which is managed by a third-party provider and available for many objectives. Cloud infrastructure provides elastic computing resources, including servers, storage, databases, and applications on a pay-as-you-go basis. This approach offers significant cost savings while enabling features such as regular data backups, modifications, and easy scaling to meet the demands of high workloads. The IT team does not need to invest effort in installing, managing, and upgrading technology. A large amount of redundancy in the cloud provides high reliability

and allows organizations to quickly configure resources and change the scale quickly based on business needs.

Limitation

For the analysis of the New York real estate market after the recovery from the epidemic, we created a database system based on the data package available on StreetEasy and analyzed it with a query tool. We focused on categorizing New York by region and housing type to determine the most suitable segments for real estate companies to invest in and the potential development advantages. Throughout the process, we were most constrained by the completeness and diversity of the data set.

Firstly, we had some missing values in the existing dataset, such as areas, boroughs, median asking price, inventory, bedroom type, property type, and rent cuts for both rent and sale. This information affects the overall analysis and the result of the feedback to the client to some extent.

The second point is that we focus on the comprehensiveness of the analysis and therefore try to collect more data sets such as insurance, building, building facilities, agent, etc. However, as we are in the initial steps of building the model, we do not have access to these specific datasets, and possibly some of them require official consent for information protection. Based on this, we have constructed the conceptual model and built the associated tables, but there is no data insertion. In the future, after the database system is more mature and put into commercial use, the data set will be further enriched and populated as well.

The third point, regarding data storage, is that we prefer to store the data on an external server, which offers cost-saving benefits as well as features such as regular data backup and easy scalability. We can link and interact with our audience and database, and facilitate timely updates of property information, ensuring more efficient real-time monitoring of the entire market. However, there are pros and cons to everything, and one of the disadvantages of relying on the Internet to store files is that Internet outages can completely disrupt user access to important files. Also delegating the management of data to another party can create management and certain information security issues. What’s more, on a deeper level, we hope that the database system will not only stop at analyzing New York properties but that in the future this proven framework might be applied to other states' markets, contributing to the development and strategic blueprint of real estate companies.

Conclusion

According to the analysis of our relational database, we have come up with the following findings. First, for the part of the rental, we have found the top 5 neighborhoods with higher demand and more competitive rent prices. They are the **Upper East Side, Upper West Side, Midtown East, Midtown West, and Chelsea.** Not surprisingly, all of those popular areas are in the borough of Manhattan.

Second, since the price of rentals with different types of bedrooms highly depends on the size of the area, which is not available in our database, we can not simply compare the inventory and

rent price to draw the conclusion of which type of bedroom is more popular. Consequently, we have analyzed their risk resilience, seeing if the rent prices for different bedroom types had dropped significantly under the effects of COVID 19, and come up with the finding that the **risk resilience** of **One bed > Studio > Two beds > Three or more beds**. Moreover, we also calculate the growth rate to see if the rent prices for different bedroom types have increased dramatically

after the pandemic. We concluded that the growth rate of **Studio > Two beds > One bed >Three or more beds.**

Furthermore, we have made a similar analysis on the sales part and found that the **Upper East Side, Upper West Side, Midtown East, Flushing, and Williamsburg** are the top 5 popular neighborhoods where the numbers of housing transactions are higher than in any other neighborhoods.

Finally, we also rank the property type by its risk resilience, growth rate, and the number of transitions that have been made to detect which property types are more popular. The rank of risk resilience is **Co-Op > Single Family/ Townhouse > Condo**, the rank of growth rate is **Single Family/ Townhouse > Co-Op > Condo**, and the rank of transaction number is **Single Family/ Townhouse > Co-Op > Condo**.

Based on our findings through the analysis, we have summarized several recommendations for the company's next round of property investment plan. Among all areas, we suggest investing in the **Upper East Side, Upper West Side, Midtown East, Midtown West, and Chelsea** areas for rental properties, and in the **Upper East Side, Upper West Side, Midtown East, Flushing, and Williamsburg areas** for sales properties, which are potentially more profitable than other areas. Furthermore, for the choice of type, we highly recommend further investing in the **Studio** for the rental properties and **Single Family/ Townhouse** for the sales properties.

Reference

Hamed (2019, January 5). REAL ESTATE ANALYSIS: 5 STEPS TO CONDUCTING AN ACCURATE RENTAL MARKET ANALYSIS. MASHVISOR. https://www.mashvisor.com/blog/5-steps-rental-market-analysis/

List price vs sale price: Know the difference. (2021, November 26). RECO Website. https://www.reco.on.ca/buyer-seller-news/list-price-vs-sale-price-know-the-difference/#:%7E:text=Listing%20price%2C%20also%20known%20as,their%20listing%20or%20asking %20price.

Santarelli, M. (2022, April 15). *NYC housing market: Prices: Trends: Forecasts 2022*. Norada Real Estate Investments. Retrieved April 20, 2022, from https://www.noradarealestate.com/blog/new-york-real-estate-market/

StreetEasy. (n.d.). StreetEasy Blog – NYC Real Estate Trends and Data, Tips and Advice. StreetEasy Blog. Retrieved April 20, 2022, from https://streeteasy.com/blog/

Zumper. (n.d.). *Average rent in New York, NY and cost information*. Zumper. Retrieved April 20, 2022, from https://www.zumper.com/rent-research/new-york-ny