A Case Study On Short Term Rental Investment Opportunities In New York

Prachi Tiwari

Objective

- Analysis of Airbnb and property sales and weather data of New York.
- Finding investment opportunities for short term rental market.
- Process, analyze and provide actionable data points through visualization and statistics

Agenda

- **Analytics Insights:** Present data analysis findings for Airbnb and Property Sales data.
- **Engineering Perspectives:** Discuss the technical aspects and methodologies used in the analysis, providing a deeper understanding of the process

Analyzing Airbnb

Insights

Assumptions

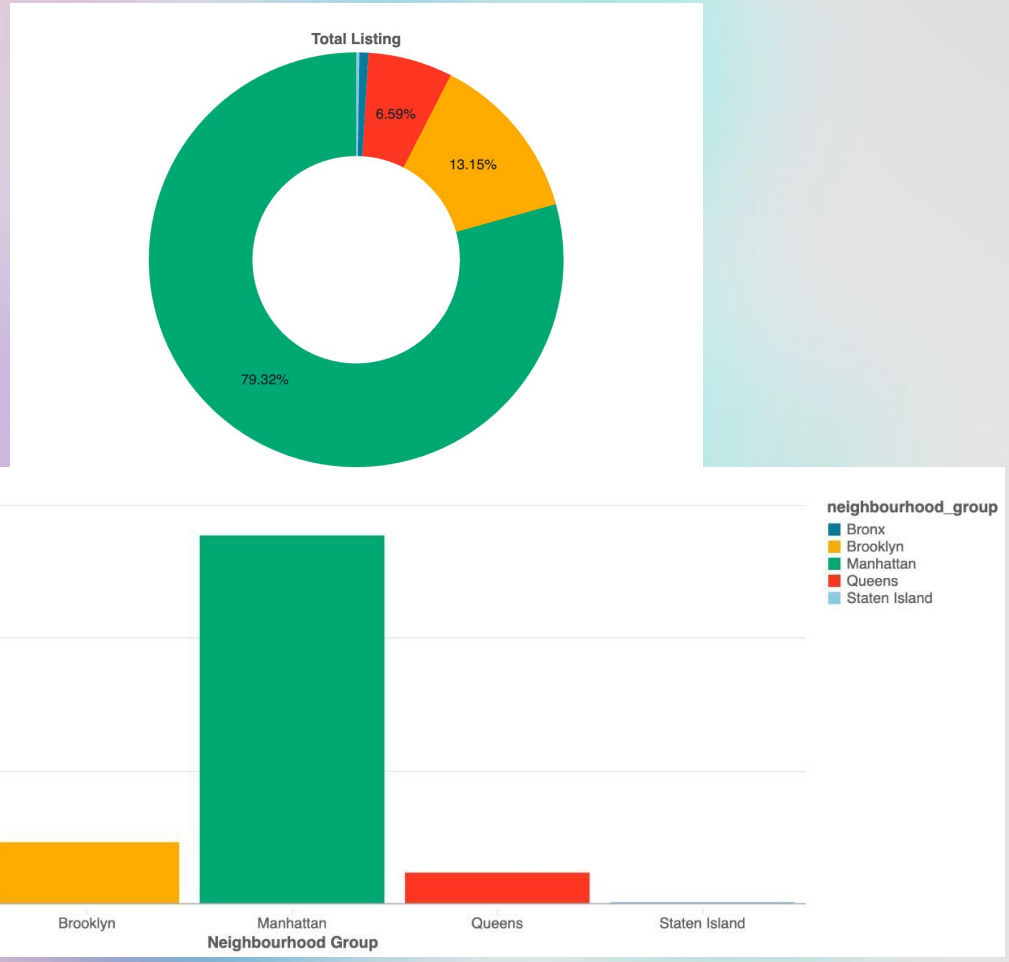
- Reviews are considered as positive user intent.

Listings

- Manhattan tops with ~277K listing and boost share of 79.32%
- Brooklyn(~46K) is in second place with 13.15% of share.
- Queens(~23K) contributes to 6.59%.
- Bronx(~2.5K) and Staten Island (~<1k) Listings are at the last place.

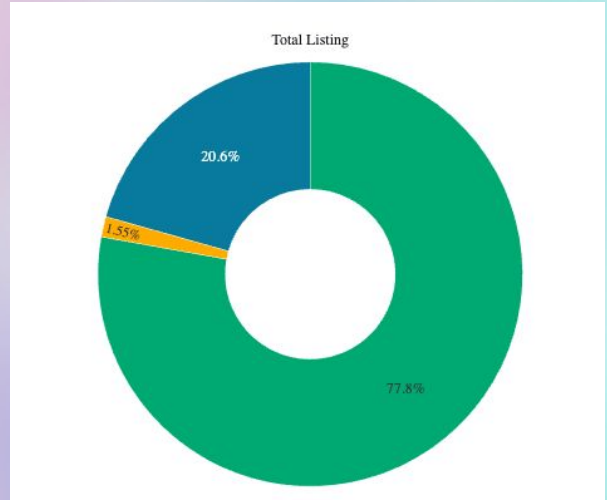
Key Data Point

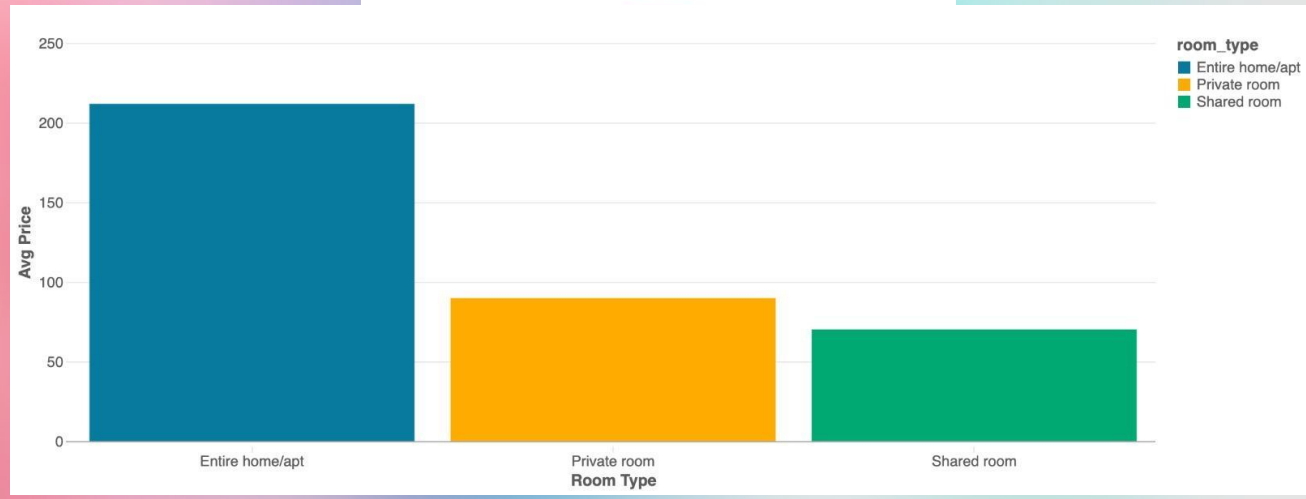
- Manhattan with 277K listing has significant higher number, resulting in the crowded rental market



Room Type

- Entire Home has the highest Listing and Revenue contribution.
- Average Price of Entire Home due to larger area is higher than Private Room or Shared Room.



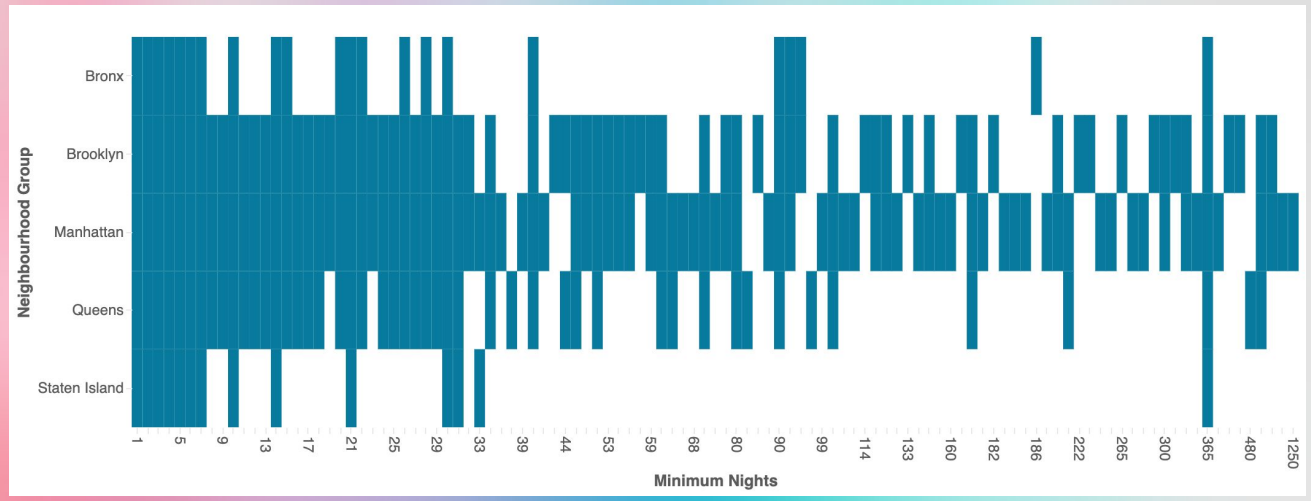


Key Data Point

- While average price of Shared Room is close to Private Room, low revenue generation clearly points to consumer preference towards Private Room and Entire Home.

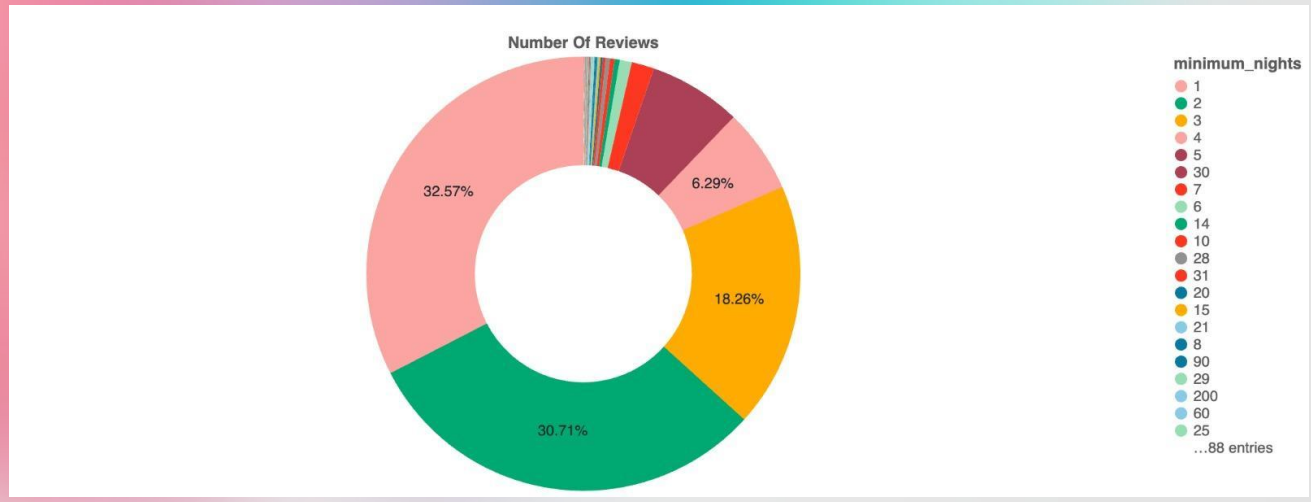
Minimum Nights

- Every neighborhood has listing for <8 minimum nights.
- Neighborhoods with higher number of listing and reviews has more tiers of minimum nights.



Key Data Point

- The popularity is inversely proportional to minimum nights.

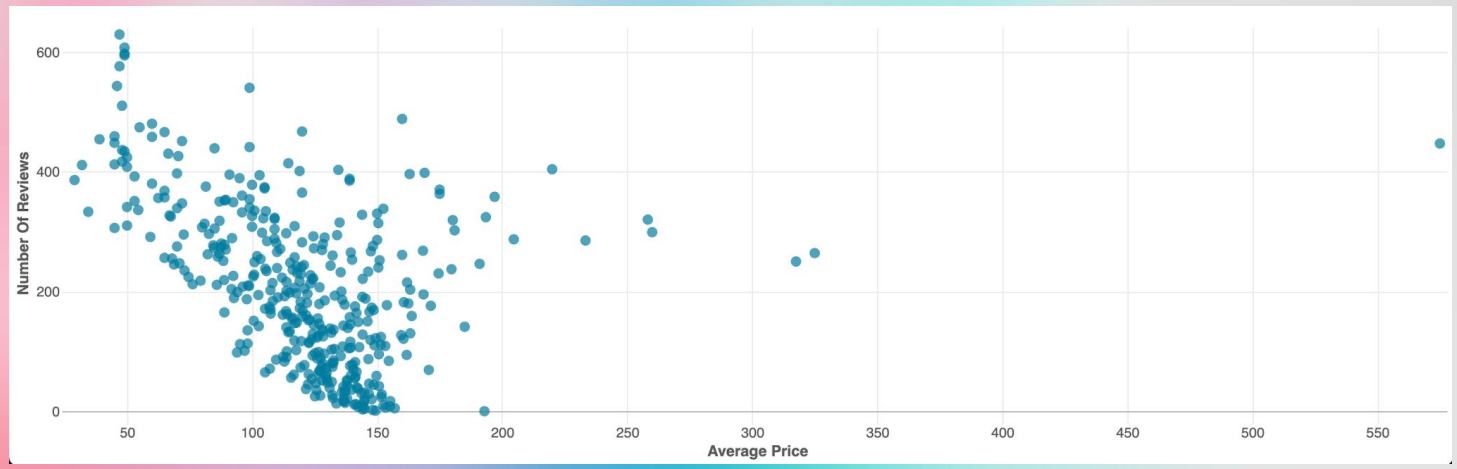


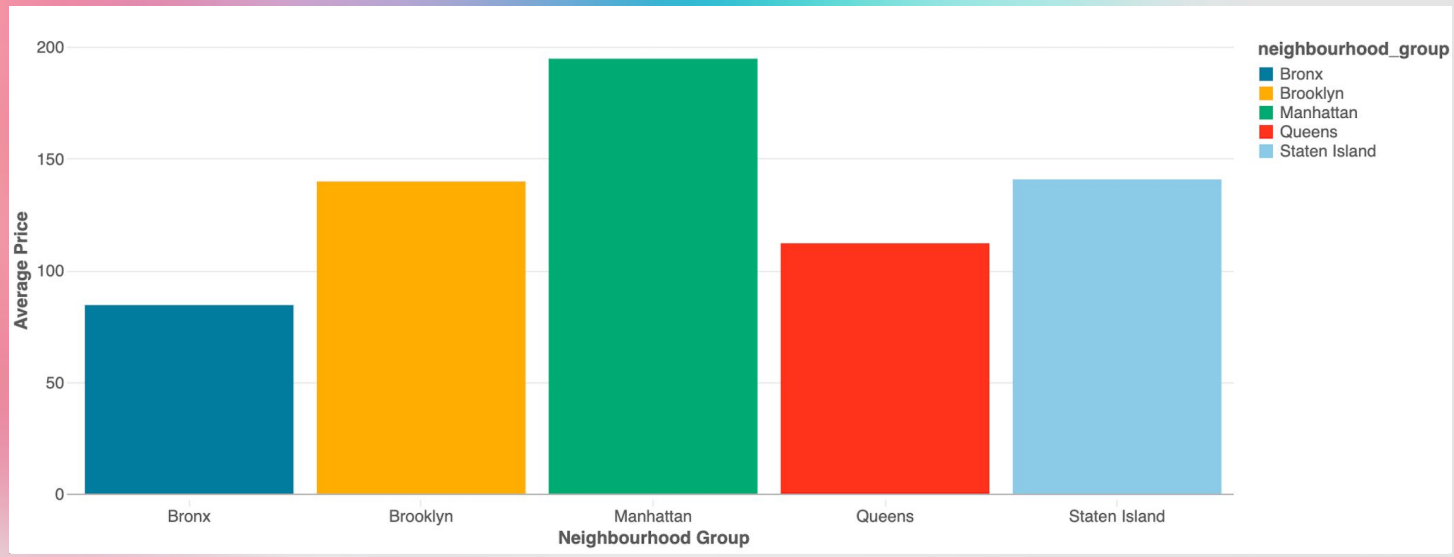
Average Rental Price

- Rental Price between 100 to 150 is has the more number of reviews.
- Manhattan has the highest average rental price ~196 dollars.
- Brooklyn and Staten Island has the average rental price ~126 and ~114 dollars.

Key Data Point

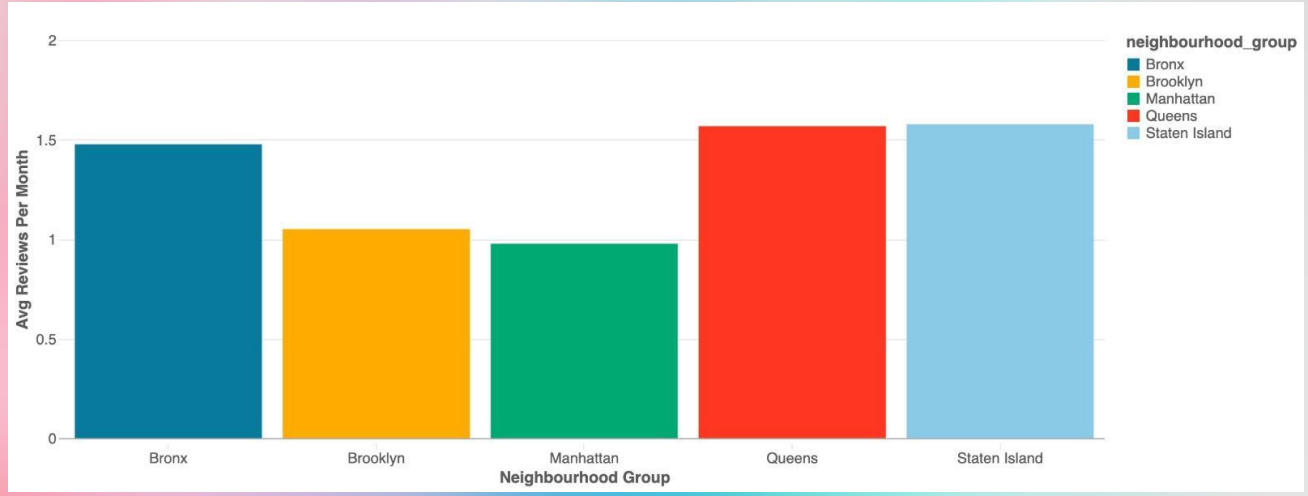
- Average rent price between 100-150 dollars are most popular.
- Rental in Manhattan is expensive.
- Brooklyn Rental price is sweet spot for the users.

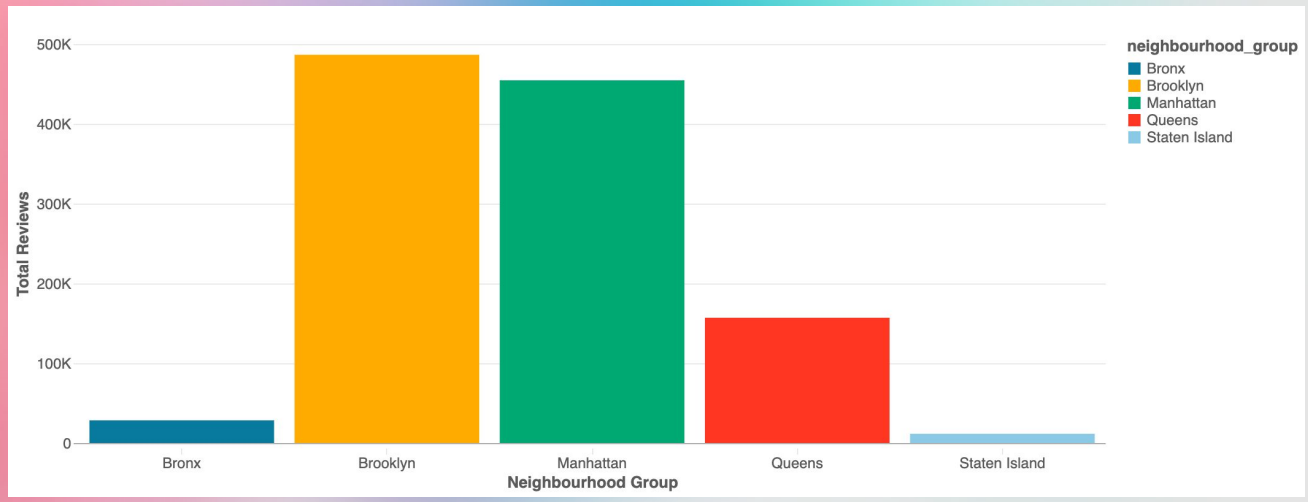




Reviews

- Manhattan though had highest number of listings, but Brooklyn leads the review count.
- Followed by Queens, Bronx and Staten Island



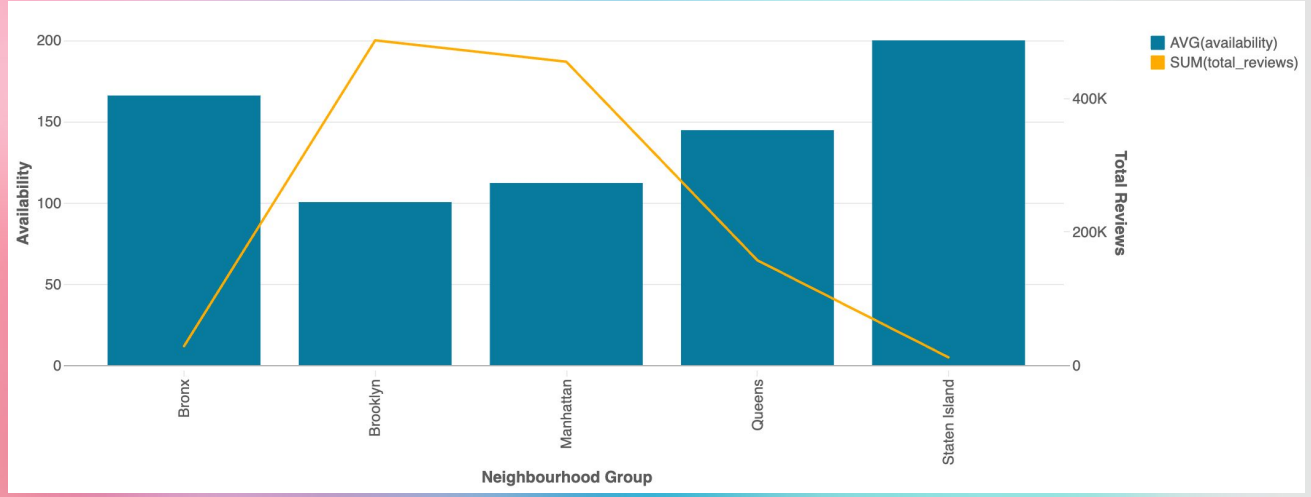


Key Data Point

- With lower listings Brooklyn is more popular.
- Neighborhoods with lower number of listing such as Bronx, Queens and Staten Island has higher average monthly reviews, which signifies high utilization rate.

Availability

- Brooklyn and Manhattan tends to have lower availability.
- Bronx, Queens and Staten Island have higher availability but are less popular.



Key Data Point

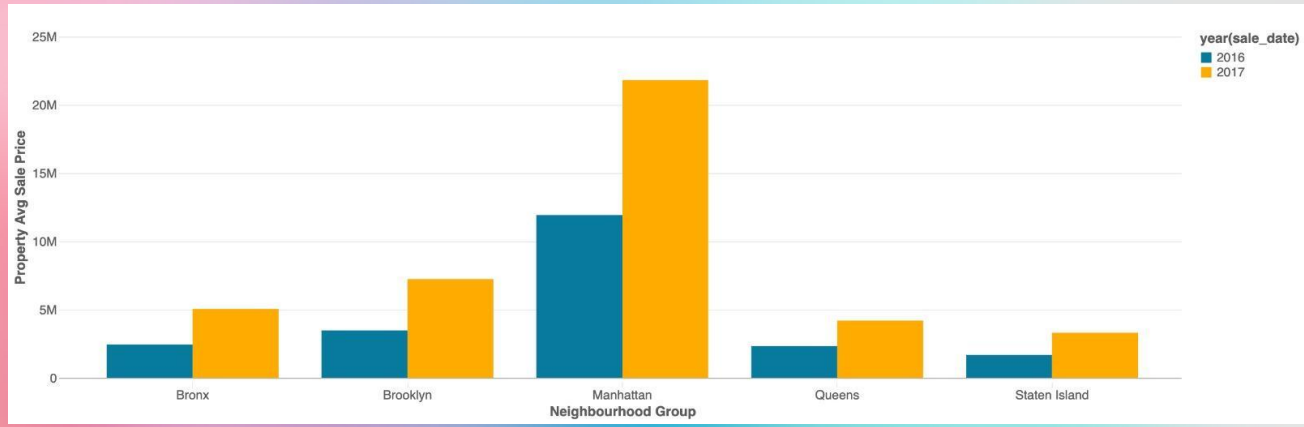
- Brooklyn listings with less availability are more popular than Manhattan

Analyzing Property Sales

Insights

Sale Price

- Comparative average sale price between 2016 and 2017

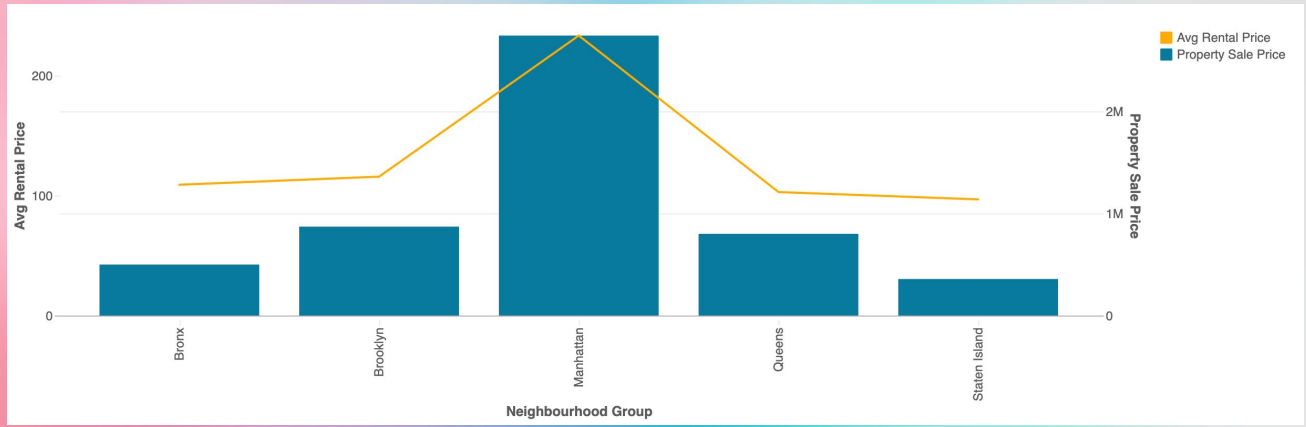


Key Data Point

- Manhattan real estate prices are significantly higher than any other neighborhood
- All neighborhood trends towards 2x appreciation

Sale Price and Rental Price

- Correlation between average property sale price and rental price



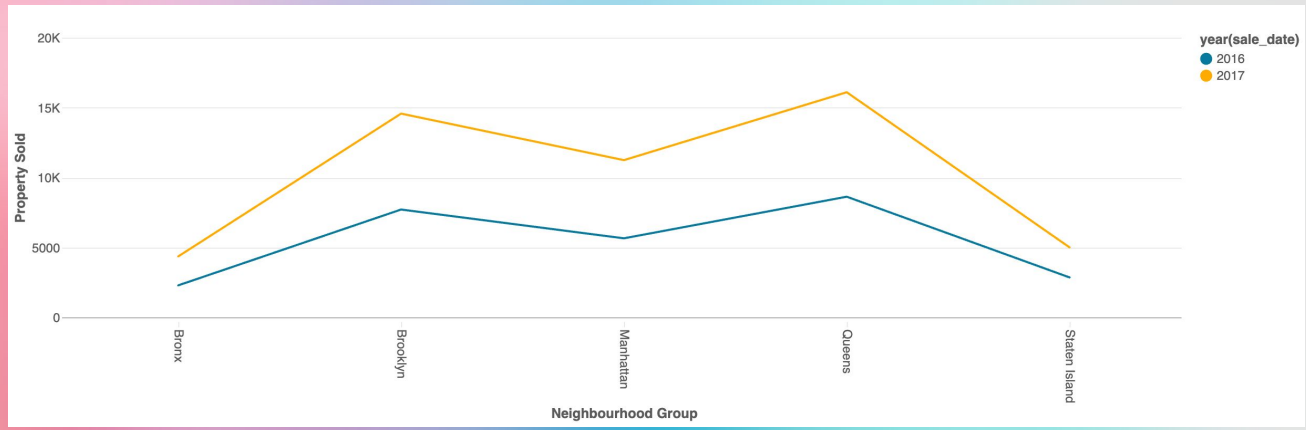
Neighbourhood Group	Avg Sale Price	Avg Rental Price
Manhattan	2744247.04	233.31
Brooklyn	871415.99	115.73
Queens	800951.02	102.91
Bronx	499714.87	109.01
Staten Island	358040.15	96.81

Key Data Point

- While Manhattan has average sale price 3x to Brooklyn, the average rental price is 2x.

Property Sold

- Sold properties between 2016 and 2017



Key Data Point

- Brooklyn and Queens which are cheaper options than Manhattan has higher number of property sold.
- This signifies Manhattan real estate market is expensive and saturated while Brooklyn and Queens is upcoming market.

Conclusion

Investment advice for short term rental market between Brooklyn and Manhattan

Manhattan

79%
Rental listings

High
Rental Availability

3x
Real Estate Price

2x
Average Rental Price

Brooklyn

13%
Rental listings

High
Rental Availability

1/3
Real Estate Price

1/2
Average Rental Price

Conclusion

Manhattan

- Capital intensive investment in a crowded rental market.
- Low monthly reviews implying low utilization of property.
- High value real estate market.
- Fits for various tiers of minimum days booking.
- Indication of market saturation based on the lower number of property sold in comparison with Brooklyn

Brooklyn

- Low capital investment in upcoming neighborhood.
- High review volume indicates higher popularity among users.
- Developing real estate market in terms of property appreciation and number of property sold (YoY)
- Fits for various tiers of minimum days booking.
- Lower average rental price is compensated by lower capital investment with high popularity indicated fast return on investment.

Engineering the Data

Filtering, Cleanup and Pipeline

Phases

1. ETL
    1. Loading data
    2. Data profiling
    3. Clean and Transform
    4. Data load
2. Scheduling
3. Testing
4. Deployment
5. Analytics

ETL

- Loading data
  - Import Airbnb, Property sales, weather data to appropriate environment.
- Data profiling
  - Understand the data by its schema and values to perform ETL.
  - Understand the numeric and nonnumeric data.
  - Verify data for null values, duplication and appropriate data types.
- Clean
  - Defining schema
  - Handling null values
  - Cast data types
  - Renaming columns
  - Removing duplicates
  - Adding derived columns
- Data load
  - Load the data to delta table using PySpark at clean stage
  - Perform dimension modeling
  - Load data to warehouse

ETL - AirBnB

Imports

```python
from pyspark.sql import functions as F
from pyspark.sql.functions import isnan, when, count, col
from pyspark.sql.types import StructType, StructField, DoubleType, StringType, TimestampType
```

Python

```python
PATH = 'sample-s3-location'
DB = 'sample_db'
```

Python

```python
%sql
CREATE WIDGET TEXT database DEFAULT "hive_metastore.sample_db"
```

Python

neighbou...	room_type	price	minim...	numbe...	last_re...	review...	calculated_h...
Brooklyn	Private room	149	1	9	2018-10-19	0.21	6
Manhattan	Entire home/apt	225	1	45	2019-05-21	0.38	2
Manhattan	Private room	150	3	0	1900-01-01	0	1
Brooklyn	Entire home/apt	89	1	270	2019-07-05	4.64	1
Manhattan	Entire home/apt	80	10	9	2018-11-19	0.1	1
Manhattan	Entire home/apt	200	3	74	2019-06-22	0.59	1
Brooklyn	Private room	60	45	49	2017-10-05	0.4	1

#Airbnb Data

+ Code + Markdown

```python
airbnb_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .option("multiLine", "true") \
    .option("escape", "\"") \
    .load("s3://{PATH}/tmp/Airbnb_data_New_York.csv")

#check schema
airbnb_df.printSchema()
#view sample data
airbnb_df.limit(5),display()
#chcek null values
print("columns with null values")
airbnb_df.select([count(when(col(c).isNull(), c)).alias(c) for c in airbnb_df.columns]).display()
#check duplicates
print("duplicate records in columns" )
duplicates_all_columns = airbnb_df.groupBy(airbnb_df.columns).count()

# Filter out rows that have a count greater than 1 (duplicates)
duplicates_all_columns = duplicates_all_columns.filter("count > 1")
duplicates_all_columns.display()

#fill null columns with appropriate values
airbnb_df = airbnb_df.fillna({'reviews_per_month': 0})
airbnb_df = airbnb_df.fillna({'last_review': '1900-01-01'})
airbnb_df= airbnb_df.drop('id','name','host_id','host_name')

#write cleaned data to deltatable
airbnb_df.write.format("delta").mode("overwrite").saveAsTable(f"{DB}.ny_airbnb")
```

Python

ETL – Property Sales

```
#Property Sales Data

df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .option("multiLine", "true") \
    .option("escape", "\"") \
    .load("s3://{PATH}/tmp/Property_sales_data_New_York.csv")

#Mapping of BOROUGH to BOROUGH_NAME
df = df.withColumn(
    "BOROUGH_NAME",
    F.when(F.col("BOROUGH") == 1, "Manhattan")
    .when(F.col("BOROUGH") == 2, "Bronx")
    .when(F.col("BOROUGH") == 3, "Brooklyn")
    .when(F.col("BOROUGH") == 4, "Queens")
    .when(F.col("BOROUGH") == 5, "Staten Island")
    .otherwise("Unknown")
)
```

	BOROUGH_NAME	NEIGHBORHOOD	RESIDENTIAL_UNITS	SALE_PRICE	SALE_DATE
1	Manhattan	ALPHABET CITY	0	866000	2017-01-09
2	Manhattan	CHELSEA	0	103640	2017-02-15
3	Manhattan	CHELSEA	0	849000	2016-12-15
4	Manhattan	CHELSEA	0	2495000	2016-12-01
5	Manhattan	CHELSEA	0	6033271	2017-06-27

```
#check schema
df.printSchema()
#view sample data
df.limit(5).display()
#column required and renaming column
df = df.select(F.col('BOROUGH_NAME'),
               F.col('NEIGHBORHOOD'),
               F.col('RESIDENTIAL UNITS').alias('RESIDENTIAL_UNITS'),
               F.col('SALE PRICE').alias('SALE_PRICE'),
               F.col('SALE DATE').alias('SALE_DATE'))

#chcek null values
print("columns with null values")
df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).display()

#check duplicates
print("duplicate records" )
duplicates_all_columns = df.groupBy(df.columns).count()
#duplicate rows
duplicates_all_columns = duplicates_all_columns.filter("count > 1")
duplicates_all_columns.limit(5).display()

before_drop = df.count()
print(f"Number of rows before dropping duplicates: {before_drop}")

# Perform drop duplicates
df = df.dropDuplicates()

after_drop = df.count()
print(f"Number of rows after dropping duplicates: {after_drop}")

#convert sale_price to double
df = df.withColumn(
    "SALE_PRICE",
    F.regexp_replace(F.col("SALE_PRICE"), "[^0-9.]", "").cast(DoubleType())
).withColumn("SALE_DATE", F.col("SALE_DATE").cast('date'))

cleaned_df =df.fillna({'SALE_PRICE': 0})

cleaned_df.write.format("delta").mode("overwrite").saveAsTable(f"{DB}.ny_property_sales")
```

ETL – Weather Data

#Weather Data

```
#Weather data first 9 rows contains the static information of location,latitude,temp_unit
# Actual data start from 10th row
# Define schema of the column present in 10th row
# Add static columns as additional columns

schema = StructType([
    StructField("date", TimestampType(), True),
    StructField("temp_max", DoubleType(), True),
    StructField("temp_min", DoubleType(), True),
    StructField("temp_avg", DoubleType(), True)
])

df = (spark.read.format("com.crealytics.spark.excel").option("header", "true")
    .option("inferSchema", "true")
    .option("dataAddress", "'0 daily'!A10")
    .schema(schema)
    .load("s3://{PATH}/tmp/Weather Data.xlsx"))

full_df = (df.withColumn("location", F.lit('New York'))
            .withColumn("latitude", F.lit(40.666534))
            .withColumn("longitude", F.lit(-74.0625))
            .withColumn("altitude", F.lit(44.787575))
            .withColumn("temp_unit", F.lit('°C'))
)

full_df.write.format("delta").mode("overwrite").saveAsTable(f"{DB}.daily_temperature")
```

	date	1.2 temp_max	1.2 temp_min	1.2 temp_avg	location	1.2 latitude	1.2 longitude
1	2016-09-01...	19.21612	9.236119	14.4386215	New York	40.666534	-74.0625
2	2016-09-02...	21.10612	11.196119	15.695287	New York	40.666534	-74.0625
3	2016-09-03...	16.05612	4.816119	7.6294513	New York	40.666534	-74.0625
4	2016-09-04...	10.626119	2.4861193	6.8319516	New York	40.666534	-74.0625

Scheduling

- Schedule the ETL pipeline based on the frequency using airflow.
- Define the dependency between the task
- Implement retry mechanisms for failed ETL jobs.
- Use logging and monitoring to track job statuses and errors.
- Set up alerts and notifications for failures.

Testing

- Unit testing for individual ETL components.
- Integration testing for end-to-end data flow.
- Performance testing to ensure ETL processes can handle large volumes of data efficiently.

Deployment

- Use separate environments for development, testing, and production.
- Use a version control tool such as git to maintain the version of codes.
- Automate the build, testing, and deployment process using CI/CD tools like Travis CI.

Analytics

- There are various tools which we can use to perform analytics.
- For this case study, the analytics is done on delta table using PySpark.
- Analytics using Snowflake, Tableau or PowerBI, which will have following steps:
  - Load data from Delta Table to Snowflake.
  - Create views to have access controls.
  - Dashboard will be created on views using reporting tools.

References

- Data Preparation using pyspark notebook
  - Notebook
- Dashboard creation using pyspark notebook
  - Dashboard