ETL Report

Anthony Rondos, Nathan Williamson, Erick Sanmartin, Peter Morris

Introduction

To uncover real estate sales trends in New York City and predict the cost of residential property within the city, our team investigated datasets from real estate sales website Streeteasy, rolling sales information provided by the New York City Department of Finance, and the Census Economic Survey. Exploration of these datasets was crucial for understanding the characteristics of the NYC market: we chose datasets that provided price breakdown by zip code, property features, date of sale, and economic indicators.

Data Sources

New York City Department of Finance. (2022). *Rolling Sales Data*. https://www1.nyc.gov/site/finance/taxes/property-rolling-sales-data.page

Street Easy. (2010-2022). *Sales in NYC, All Home Type, Median Sale Price*. https://streeteasy.com/blog/data-dashboard/?agg=Median&metric=Recorded%20Price&type=Sales

Street Easy. (2010-2022). *Rentals in NYC, All Bedroom Count, Median Asking Price*. https://streeteasy.com/blog/data-dashboard/?agg=Median&metric=Asking%20Rent&type=Rentals

United States Census Bureau. (2017). *Economy-Wide Key Statistics 2017*. https://www.census.gov/data/developers/data-sets/economic-census/2017.html

United States Census Bureau. (2020). *County Business Patterns: 2020*. https://www.census.gov/data/datasets/2020/econ/cbp/2020-cbp.html

United States Census Bureau. (2020). *ZIP Code Tabulation Areas*. https://www.census.gov/cgi-bin/geo/shapefiles/index.php?year=2020&layergroup=ZIP+Code+Tabulation+Areas

---

Extraction

Rolling Sales

1. Go to the website https://www1.nyc.gov/site/finance/taxes/property-rolling-sales-data.page.

2. Download each borough file as an MS Excel

3. As an alternative to #2, copy the MS Excel link on the webpage.

4. Paste into a Jupyter notebook and set equal to ‘link’,

5. Make an imports cell to include ‘pandas’.

6. Read in the Excel file via the link in #4, and skip the first 3 rows as a parameter.

Street Easy

1. Go to the StreetEasy website https://streeteasy.com/blog/data-dashboard.

2. In a Jupyter notebook, make an imports cell for the packages ‘pandas’, ‘requests’, ‘zipfile’, ‘io’, and ‘urlopen’, as per the guidelines of towardsdev.com/nyc-streeteasy-get-real-estate-housing-data-using-python-dc9632d5a609.

3. In a Jupyter notebook, define a read_zip function to read contents from urls.

4. Define a melt_dataframe function to transform the month columns into rows inside the dataframe, leaving the other columns intact.

5. Obtain the StreetEasy .zip files from the website divided by property type for sales (3 files) and by number of bedrooms for rentals (4 files). This is done in the dropdown of the Download Data section.

    a. Select the links from the ‘Median Sale Price’ among the property types and ‘Median Asking Rent’ among the bedroom counts.

6. Declare an empty dictionary in a new Jupyter cell.

7. Create new columns according to the property types or bedroom counts, setting each equal to its respective link (7 in total).

8. Declare empty lists, one for ownership and the other for rentals.

9. Iterate through the urls using the .items() method for ownership (3 urls) and again for rentals (4 urls). Inside the loop, do the following:

    a. Call the functions from steps 3 and 4.

    b. Add new columns to the lists by melting in the name of item + median price from each group mentioned.

    c. Append to the appropriate master list

10. Construct new dataframes out of the zeroth index of each list in step 9.

11. Iterate through each of the remaining dataframes for the own and rent groups and append new columns in order to combine them into their own tables.

Census

1. Open the economy-wide key statistics 2017 on the website https://www.census.gov/data/developers/data-sets/economic-census/2017.html.

2. Copy and paste the provided url https://api.census.gov/data/2017/ecnbasic into a Jupyter notebook cell, setting it to a variable.

3. Create several parameters in a dictionary: “get:” the columns "NAICS2017_LABEL, NAICS2017,GEO_ID,EMP,NAME,FIRM,RCPTOT,RCPTOT_F,PAYANN, SECTOR", “for:” all counties selected (*), “in:” the state number for New York (36), and “key:” the api key associated with your US Census account.

4. Read in the url and params as a json through the get_requests method, stored as a variable.

5. Build a Pandas dataframe from the json, specifying row index 1 onward and the zeroth index column, stored as a new variable “econNYC”.

6. Repeat steps 2 to 5 for County Business Patterns file https://api.census.gov/data/2020/cbp, which can be found on the Census website here: https://www.census.gov/data/datasets/2020/econ/cbp/2020-cbp.html.

7. Make the following edits: “get:” NAME,NAICS2017_LABEL,ESTAB,PAYANN,PAYQTR1,EMP; “for:” all zip codes (*). Same api key. Comment out the “in:” statement of params. Assign it to object “cbp”.

8. Read in the zip code tabulation area file ‘zcta/tl_2020_us_zcta520.shp’ via geopandas gpd, assigning it to “zcta”. The file is downloadable here: https://www.census.gov/cgi-bin/geo/shapefiles/index.php?year=2020&layergroup=ZIP+Code+Tabulation+Areas.

9. Read in the .txt file 'https://www2.census.gov/geo/docs/maps-data/data/rel2020/zcta520/tab20_zcta520_place20_natl.txt' downloaded on your local machine or via the url, assigning it to “places”. Set the delimiter equal to “ | ”.

Transform

<u>Rolling Sales</u>

1. Load the Brooklyn, Queens, Manhattan, Bronx, and Staten Island rolling sales csv's into a Pandas dataframe via url.

2. Drop the following columns in all five datasets: 'EASEMENT', 'APARTMENT NUMBER', 'TAX CLASS AT PRESENT', 'BUILDING CLASS AT PRESENT'

3. Replace nulls in the following columns with 0.0 in all five datasets: 'GROSS SQUARE FEET', 'LAND SQUARE FEET', 'TOTAL UNITS', 'SALE PRICE', 'YEAR BUILT'

4. Drop all na's in all five columns

5. Concatenate all five dataframes into a new dataframe

6. Export as csv called nyrollingproperty

<u>Street Easy</u>

1. With the 'month' column inside the dataframe from the extraction phase, add two new columns titled 'Year' and 'Month' by splitting the strings on the dash '-'.

2. Drop the 'month' column.

3. Create a copy of this dataframe.

4. Sort the df values on 'areaName' and 'Year'.

5. Filter out the 'areaName' strings that are neither neighborhoods nor submarkets except for "Staten Island," i.e., remove "Manhattan", "Bronx", "Brooklyn", "Queens," and "NYC".

6. Filter out all rows containing 'submarket' in the 'areaType' column.

7. Drop all null values from the 'areaName' column.

8. Drop the 'areaType' column completely.

9. Drop rows in the entire dataframe that are missing more than 2 prices in the price columns of ownership, or missing more than 3 prices in the price columns of rentals.

10. Reset the index of the dataframe with "drop=True" to prevent new column formation.

11. Order the dataframe columns by 'areaName', 'Borough', 'Year', 'Month', and the various price columns (leave the price order as it is in the Street Easy file) for own/rent.

12. Cast each of the price columns and the ‘Year’ and ‘Month’ columns as type ‘Int64’, which allows for the possibility of null values.

13. Map the names of the NYC boroughs from strings to their corresponding integer labels.

14. Rename the ‘areaName’ header to ‘Neighborhood’.

Census

1. With the dataframe “econNYC”, API call (step 6 of Extraction), slice on “NAME” equalling each of the five NYC borough names, joined by “or” operators (|). Store this object as “NYC”.

2. Drop all duplicates from the columns NAICS2017_LABEL and NAME of the NYC dataframe.

3. Cast the following columns as “Int64” type: “EMP”, “FIRM”, “RCPTOT”, “PAYANN”.

4. Rename (with their logical equivalent) and reorder the columns to fit the DDL schema as such: 'boroughID','industryLabel', 'industryLabelID', 'geoID', 'numEmployees', 'numFirms', 'RCPtot', 'RCPtotFlag', 'annualPay', 'sectorNum'.

5. Define a function to map the string borough names to their numerical designation.

6. Modify the column “boroughID” by applying the function in the previous step in a lambda expression.

7. Export the dataframe as a CSV named “EconomicSurvey2017.csv”.

8. With the dataframe “econNYC” (step 5 of Extraction), reduce the dataframe by filtering the column "NAMELSAD_PLACE_20" by “New York City”.

9. Drop the null values in the column “GEOID_ZCTA5_20”.

10. Insert the clean dataframe into a lambda expression in which you type-cast unique inputs as a composition of integer (inner) and string (outer). Assign the result to “nyc_zcta_list”.

11. Filter zcta by nyc_zcta_list through a .isin method. Assign the result to “zcta_nyc”.

12. Merge zcta_nyc with cbp through a left join on “ZCTA5CE20” and a right join on “zip code”.

13. Cast the columns “EMP”, “PAYANN”, “PAYQTR1”, and “ESTAB” as integer type.

14. Cast the columns “INTPTLAT20” and “INTPTLON20” as float type.

15. Cast the column “ZIPCODE” as string type.

16. Drop the columns "GEOID20", "CLASSFP20", "MTFCC20", "FUNCSTAT20", "ZCTA5CE20", "NAME", and "geometry".

17. Rename (the logical equivalent) and reorder the columns like so: 'zipcode','landArea', 'waterArea', 'latitude', 'longitude', 'industryLabel','numEstablishments', 'annPayroll', 'q1Payroll', 'numEmployees'.

18. Export this dataframe as a CSV “CountyBusinessPatterns.csv”.

Load

<u>Rolling Sales</u>

1. Load nyrollingproperty csv into spark dataframe

2. Drop column ‘_c0’

3. Change column ‘SALE DATE’ data type to string

4. For Borough SQL table:

   a. Extract distinct values of ‘BOROUGH’ column to a dataframe
   b. Sort dataframe in ascending order
   c. Append dataframe to Borough SQL table

5. For Neighborhood SQL table:

   a. Extract distinct rows of ‘BOROUGH’ and ‘NEIGHBORHOOD’ into a separate dataframe. Both columns are used to account for neighborhoods with the same name in different boroughs
   b. Import *pyspark.sql.functions*
   c. Use the monotonically_increasing_id function to generate a column of Id values in this dataframe
   d. Name new column ‘Neighborhood ID’
   e. Rename ‘BOROUGH’ column to ‘BoroughID’
   f. Append dataframe to Neighborhood SQL table
   g. Inner join this dataframe to original nyrollingproperty dataframe on unique combinations of ‘NEIGHBORHOOD’ and ‘BOROUGH’ for use in following step

6. For RollingSales SQL table:

   a. Create a dataframe from nyrollingproperty dataframe, dropping ‘BOROUGH’ and ‘NEIGHBORHOOD’ columns
   b. Rename the following columns to fit SQL database schema:
      i. ‘BUILDING CLASS CATEGORY’ to ‘buildingCat’
      ii. ‘BLOCK’ to ‘block’
      iii. ‘LOT’ to ‘lot’
      iv. ‘ADDRESS’ to ‘address’
      v. ‘RESIDENTIAL UNITS’ to ‘resUnits’
      vi. ‘COMMERCIAL UNITS’ to ‘comUnits’

   vii. ‘TOTAL UNITS’ to ‘totUnits’
   viii. ‘LAND SQUARE FEET’ to ‘landSqrFt’
   ix. ‘GROSS SQUARE FEET’ to ‘grossSqrFt’
   x. ‘YEAR BUILT’ to ‘yearBuilt’
   xi. ‘TAX CLASS AT TIME OF SALE’ to ‘taxClass’
   xii. ‘BUILDING CLASS AT TIME OF SALE to ‘buildClass’
   xiii. ‘SALE PRICE’ to ‘salePrice’
   xiv. ‘SALE DATE’ to ‘saleDate’
   xv. ‘ZIP CODE’ to ‘zipcode’

c. Append dataframe to RollingSales SQL table

<u>Street Easy</u>

1. Export the Pandas dataframes as CSV’s, index=False, saved to your local drive.

2. In a Databricks Jupyter cell, specify the credentials for the SQL database.

3. Read in the formatted CSV’s as PySpark dataframes, header=True, via the mount point referencing the Azure container file.

4. Rename all the columns to match the SQL DDL schema, i.e., lowercase, underscore-spaced strings converted to camelCase, and ‘borough’ changed to ‘boroughID’.

5. Revert the price columns to type Int64, as they may have been automatically uploaded as Float type in DataBricks.

6. Having the SQL server connection established, write in the own/rent dataframes with the “append” mode enabled.

<u>Census</u>

1. In Databricks, set the mount point to be “/mnt/gp6”.

2. Import the EconomicSurvey2017.csv to Databricks as a PySpark dataframe called “econNYC”.

3. Write the file into the SQL database with “append” mode.

4. Import the CountyBusinessPatterns.csv to Databricks as a Pandas dataframe called “cbp”.

5. Drop column “_c0”.

6. Convert the Pandas dataframe to a PySpark dataframe.

7. Write the file into the SQL database with “append” mode.

Conclusion

All of the clean, processed datasets are now in the SQL database titled “group-6-finance”. All of the steps are necessary in order to create visualizations, get the machine-learning model running, and start the Kafka producer-consumer factory.