NYC Neighborhood Finder: Final Report

December 14, 2021

CIS 550: Database and Information Systems

Professor Val Tannen

University of Pennsylvania

Group 27

Parsa Fallahi: parsaf@seas.upenn.edu, parsa-fallahi

Yizhe Hang: yizheh@seas.upenn.edu, hangyhan

Thomas Donnelly: tomvdon@seas.upenn.edu, tomvdon

Noah Eagle: nkeagle@seas.upenn.edu, Noah-Eagle

Deployment Link: http://nyc-client3.us-east-2.elasticbeanstalk.com

Section 1: Introduction

Project Goals/Target Problem

Finding an apartment in New York City (NYC) is far from an easy task. With so many neighborhoods for prospective renters to choose from, it's nigh impossible for most renters to fully investigate and consider each neighborhood that best fits their wants and needs, especially with the time constraints associated with today's fast-paced, busy lifestyles. We sought to alleviate this problem by compiling neighborhood rental and crime statistics and allowing users to filter neighborhoods based on average rent prices and various crime statistics. By selecting neighborhoods they are interested in or that match their criteria, users can quickly view specialized rent and crime reports. Additionally, in case users want more general information to determine a neighborhood's relative standing, they also have the ability to view borough-wide and city-wide rental and crime reports.

Description of Application Functionality

The Dashboard page aims to give an overview of historical NYC rental and crime statistics. For the rental market, we plotted the monthly average rental price from 2020-01 to 2021-08, which will help our users better understand rent fluctuations and make wiser financial decisions. For crime statistics, we displayed the percentages that each crime level and age group takes up in all crime events from 2010-01 to 2021-08.

The Borough Information page allows users to see average rent and crime count information for each borough for a selected year. Year options for the user to choose from range from 2010 to 2020. Data is presented in both tabular and graphical formats. There is also a button that will take users to a separate Borough Trends page where they can select a borough and see rent and crime count trends over time, side-by-side.

The Filter page allows users to filter neighborhoods that meet their financial and safety criteria. With regard to financial filtering, users can provide the minimum and maximum they are willing to pay for an apartment and we will only return those neighborhoods whose average rent falls within that range, along with their average rent, minimum rent, maximum rent, and rent range. Users can then click on a neighborhood's name to be redirected to the search page where they can see its individualized neighborhood report. With regard to crime filtering, users can choose a level of offense (felony, misdemeanor, or violation) and whether they want the top 1, 5, 10, or all neighborhood(s) for this level of offense. Similarly, they can filter for the top 1, 5, 10, or all for a selected gender (M or F) and a selected age group (<18, 18-24, 25-44, 45-64, 65+). This way, the neighborhoods returned would have to be in the top X for the chosen level of offense (top being "safest"), top Y for the chosen gender, and top Z for the chosen age group. Users can also toggle

between “highest” and “lowest” (meaning “most dangerous” and “safest,” respectively), and this applies to all filters. For each returned neighborhood, we include its name, and the number of crimes at the chosen level of offense, the number of crimes with the specified gender as the victim, and the number of crimes with the specified age group as the victim.

The Search page allows users to search for neighborhoods based on the neighborhood's name, ZIP codes, or borough. The page returns all neighborhoods that match a user's search criteria as well as the ZIP codes inside those neighborhoods. When a user clicks on a neighborhood, details such as the neighborhood’s crime and rental rankings compared to other neighborhoods in the borough and a line graph of crime and rental prices over time are displayed.

Section 2: Architecture

Architecture and Technologies

The application uses a MySQL database hosted on Amazon RDS for its backend that is accessed using a Node.js express server. The application client is written in Node.js React. The stack is deployed using AWS Elastic Beanstalk. The client sends HTTP requests to the backend server which then queries the database for a result. The resulting JSON is sent back to the client as a HTTP response and corresponding information displayed. Additionally, Pandas was used for data cleaning.

Section 3: Data

Datasets Used

NYPD Complaint Data Historic (“Crime Dataset”)

This dataset includes all valid felony, misdemeanor, and violation crimes that were reported to the New York Police Department (NYPD) from 2006 through the end of 2020. We use this data to allow users to query for neighborhoods that meet certain safety criteria with regards to the crime counts in terms of offense severity levels, gender victimizations, and age group victimizations. We also use this crime data to provide neighborhood-wide, borough-wide, and city-wide reports on crime trends and fluctuations over the past decade.

Rows: 7.38 million, Columns: 35
Level of Offense Distribution: 56% Misdemeanor, 30% Felony, 14% Violation
Most Common Victim Age Group: 25-44

StreetEasy Neighborhood Median Rents (“Rent Dataset”)

Please Note: Scroll down to “Download Data”, set “Property Type” to “Rentals,” then click on the “Median Asking Rent” report to download it.

This dataset contains the median rent for each month for each neighborhood, submarket, and borough from the start of 2010 through the Fall of 2021. Similar to the crime dataset, we use this rental data to allow users to query for neighborhoods within a specified budget. We also use this rental data to showcase neighborhood-wide, borough-wide, and city-wide reports on the changes in rental prices over the past decade. Additionally, in combination with the crime dataset, we are able to provide comparisons between rental prices and crime prevalence at the neighborhood, borough, and city levels.

Rows: 200, Columns: 143, Max Median Rent: $9,000, Minimum Median Rent: $763
Average Median Rent: $2,491, Median Rent Standard Deviation: $957

NYC Neighborhoods by ZIP Codes (“Neighborhood/ZIP Dataset”)

Please Note: If the link is not working, see Table 1 in the appendix for a screenshot of the information.

This dataset maps NYC ZIP codes to their respective neighborhoods. We needed this dataset so that we could connect the crime and rental datasets. This is discussed further in the next section.

Neighborhoods: 42, ZIP Codes: 178

Section 4: Database

Data Wrangling/Cleaning and Entity Resolution

We first cleaned the Neighborhood/ZIP dataset. The PDF was converted to a CSV file using an online tool so it could be loaded into Pandas. After some initial whitespace cleaning, the ZIP codes were split into lists, since they were initially in string format. Then, the lists were exploded so that each row had one ZIP code and one neighborhood.

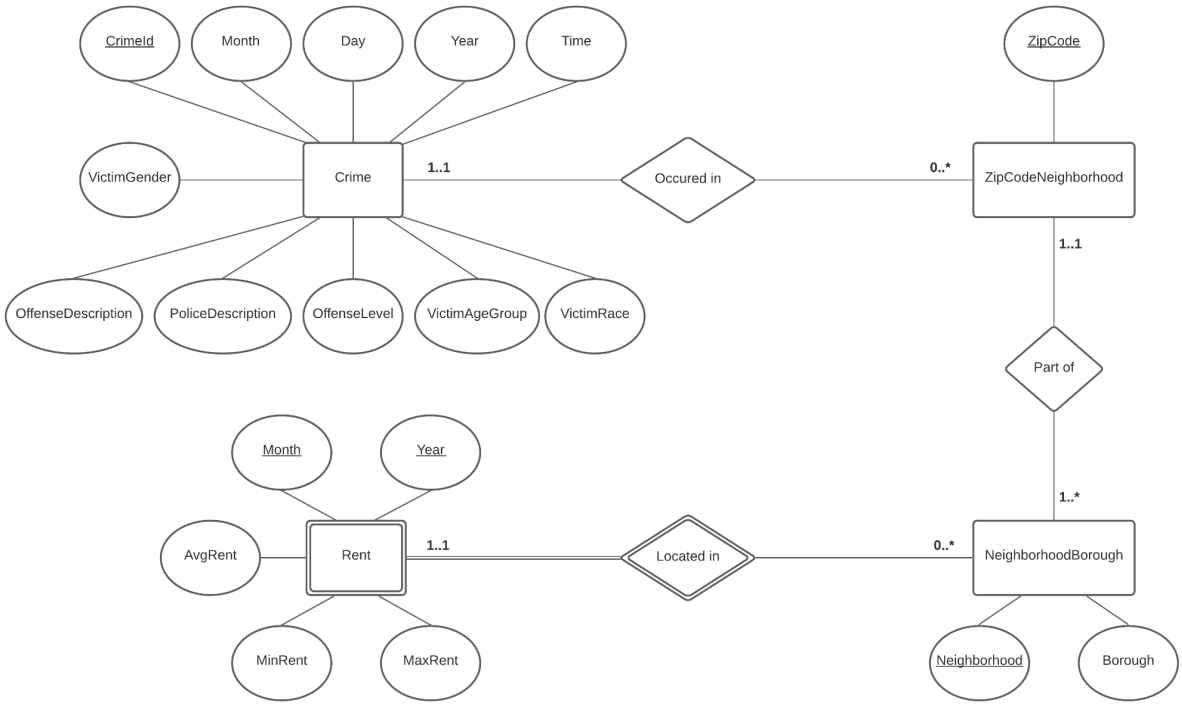
For the Rent Dataset, records were first filtered to only include sub-neighborhoods. The dataframe was “melted” to convert it from a “wide” format to a “long” format, which was far easier to query and analyze. Since the dataset had rent records by sub-neighborhood, we had to determine to which neighborhood (from list of neighborhoods in Table 1) the sub-neighborhood belonged. For example, the sub-neighborhood “Astoria” belongs to “Northwest Queens.” Due to conflicting/missing information, web scraping was not an option, so this step was done manually to ensure the neighborhood mappings were accurate. After obtaining the neighborhoods, we

dropped the sub-neighborhoods and calculated average, minimum, and maximum rents for each month of each year for every neighborhood, and split the date into month and year.

For the Crime Dataset, we first dropped any unnecessary columns and any rows that had null values for date or latitude/longitude, since these attributes were critical to our analysis, and split date into three columns: month, day, and year. Crimes were filtered to remove those from before 2010, and only "completed" crimes were kept. Attributes were systematically examined to find nonsensical values (e.g. negative numbers for the victim age group attribute), and such values were set to null. To obtain the ZIP codes from latitude/longitude, we used the uszipcode library. However, as discussed later, we split the dataframe by year; after getting the ZIP codes, we concatenated all 11 dataframes back together and merged the result with the dataframe mapping ZIP codes to a neighborhood to ensure that we removed extraneous ZIP codes. Finally, we cleaned up the text columns and dropped the date.

The main entity resolution goal was to ensure that everything was in terms of neighborhoods. For the Crime Dataset, this meant getting ZIP codes from coordinates for all crimes, which we could then map to neighborhoods. For the Rent Dataset, due to the domain knowledge required, neighborhoods were determined manually. Ultimately, having neighborhoods for both the crime and rent datasets allowed us to compare rental and crime statistics at the neighborhood level.

Entity Relationship Diagram



Number of Instances in Each Table

The Crime table has 4,374,938 instances. The ZipCodeNeighborhood table has 178 instances. The NeighborhoodBorough table has 42 instances. The Rent table has 4,785 instances.

Schemas, Normal Forms, and Justification

NeighborhoodBorough(Neighborhood, Borough)

The NeighborhoodBorough relation is in BCNF. Since Neighborhood is the key and Borough is functionally dependent on Neighborhood, for every X → A over R, X is a superkey.

Rent(Neighborhood, Month, Year, AvgRent, MinRent, MaxRent)
    FOREIGN KEY Neighborhood REFERENCES NeighborhoodBorough(Neighborhood)

The Rent relation is in BCNF. Since {Neighborhood, Month, Year} is the key and AvgRent, MinRent, and MaxRent are functionally dependent on all three of Neighborhood, Month, and Year, for every X → A over R, X is a superkey.

ZipCodeNeighborhood(ZipCode, Neighborhood)
    FOREIGN KEY Neighborhood REFERENCES NeighborhoodBorough(Neighborhood)
    INDEX ON Neighborhood

The ZipCodeNeighborhood relation is in BCNF. Since ZipCode is the key and Neighborhood is functionally dependent on ZipCode, for every X → A over R, X is a superkey.

Crime(CrimeId, Month, Day, Year, Time, OffenseDescription, PoliceDescription, OffenseLevel, VictimAgeGroup, VictimRace, VictimGender, ZipCode)
    FOREIGN KEY ZipCode REFERENCES ZipCodeNeighborhood(ZipCode)
    INDEX ON ZipCode, INDEX ON (Month, Year), INDEX ON (ZipCode, Month, Year)

The Crime relation is in BCNF. Since CrimeId is the key and every attribute is functionally dependent on CrimeId, for every X → A over R, X is a superkey.

Section 5: Queries

Query 1 with Explanation

```
    WITH IDC AS (
    SELECT ZCN.Neighborhood, TC.OffenseDescription, COUNT(IF (TC.OffenseLevel = 'Felony', 1,
NULL)) AS FCOUNT,  COUNT(IF (TC.OffenseLevel = 'Misdemeanor', 1, NULL)) AS MCOUNT, COUNT(*) AS
TCOUNT
    FROM 2020Crimes TC RIGHT JOIN ZipCodeNeighborhood ZCN ON TC.ZipCode = ZCN.ZipCode
```

```
    WHERE ZCN.Neighborhood = '${id}'
    GROUP BY (TC.OffenseDescription)),
    FMOST AS (
    SELECT Neighborhood, OffenseDescription AS FMOST
    FROM IDC
    WHERE IDC.FCOUNT >= ALL(SELECT FCOUNT FROM IDC)),
    MMOST AS (
    SELECT Neighborhood, OffenseDescription AS MMOST
    FROM IDC
    WHERE IDC.MCOUNT >= ALL(SELECT MCOUNT FROM IDC)),
    NB AS (
    SELECT *
    FROM NeighborhoodBorough
    WHERE Neighborhood = '${id}'),
    NBS AS (
    SELECT NB2.Neighborhood, NB2.Borough
    FROM NeighborhoodBorough NB2, NB
    WHERE NB.Borough = NB2.Borough),
    NBR AS (
    SELECT R.Neighborhood, AVG(R.AvgRent) AS AvgRent
    FROM Rent R
    WHERE Year = 2020
    GROUP BY R.Neighborhood),
    NBSZ AS (
    SELECT ZCN.ZipCode, NBS.Neighborhood, NBS.Borough
    FROM ZipCodeNeighborhood ZCN JOIN NBS ON ZCN.Neighborhood=NBS.Neighborhood),
    Counts AS (
    SELECT NBSZ.Neighborhood, COUNT(TC.CrimeId) AS Total, COUNT(IF (TC.OffenseLevel = 'Felony',
1, NULL)) AS Felonies,
    COUNT(IF (TC.OffenseLevel = 'Misdemeanor', 1, NULL)) AS Misdemeanors, NBR.AvgRent,
NBSZ.Borough
    FROM 2020Crimes TC JOIN NBSZ ON NBSZ.ZipCode = TC.ZipCode LEFT JOIN NBR ON NBSZ.Neighborhood
= NBR.Neighborhood
    GROUP BY NBSZ.Neighborhood
    UNION
    SELECT NBSZ.Neighborhood, COUNT(TC.CrimeId) AS Total, COUNT(IF (TC.OffenseLevel = 'Felony',
1, NULL)) AS Felonies,
    COUNT(IF (TC.OffenseLevel = 'Misdemeanor', 1, NULL)) AS Misdemeanors, NBR.AvgRent,
NBSZ.Borough
    FROM NBR RIGHT JOIN NBSZ ON NBR.Neighborhood = NBSZ.Neighborhood LEFT JOIN 2020Crimes TC ON
NBSZ.ZipCode = TC.ZipCode
    GROUP BY NBSZ.Neighborhood),
    NBRank AS (
    SELECT Counts.Borough, Counts.Neighborhood, RANK() OVER (ORDER BY Counts.Total DESC) AS
TRank, RANK() OVER (ORDER BY Counts.Felonies DESC) AS FRank, RANK() OVER (ORDER BY
Counts.Misdemeanors DESC) AS MRank,
    CASE WHEN Counts.AvgRent IS NULL THEN NULL ELSE RANK() OVER (ORDER BY Counts.AvgRent DESC)
END AS RentRank
    FROM Counts)
    SELECT *
    FROM NBRank NATURAL JOIN FMOST NATURAL JOIN MMOST
```

This query calculates the ranking of a neighborhood with respect to its borough for total crime, felony crime, misdemeanor crime, and average rent for the year 2020. This is used as a table on the search page for when a user wants more detailed information on a neighborhood.

Query 2 with Explanation

```
WITH Least_Offenses AS (
    SELECT ZipCodeNeighborhood.Neighborhood, COUNT(*) AS Offense_Count
    FROM ZipCodeNeighborhood JOIN 2020Crimes ON ZipCodeNeighborhood.ZipCode = 2020Crimes.ZipCode
    WHERE 2020Crimes.OffenseLevel = '${level}'
    GROUP BY ZipCodeNeighborhood.Neighborhood
    ORDER BY Offense_Count ${ordering}
    LIMIT ${numLevelResults}),
    Least_Gender_Victimizations AS (
    SELECT ZipCodeNeighborhood.Neighborhood, COUNT(*) AS Gender_Victimizations
    FROM ZipCodeNeighborhood JOIN 2020Crimes ON ZipCodeNeighborhood.ZipCode = 2020Crimes.ZipCode
    WHERE 2020Crimes.VictimGender = '${gender}'
    GROUP BY ZipCodeNeighborhood.Neighborhood
    ORDER BY Gender_Victimizations ${ordering}
    LIMIT ${numGenderResults}),
    Least_Age_Victimizations AS (
    SELECT ZipCodeNeighborhood.Neighborhood, COUNT(*) AS Age_Group_Victimizations
    FROM ZipCodeNeighborhood JOIN 2020Crimes ON ZipCodeNeighborhood.ZipCode = 2020Crimes.ZipCode
    WHERE 2020Crimes.VictimAgeGroup = '${agerange}'
    GROUP BY ZipCodeNeighborhood.Neighborhood
    ORDER BY Age_Group_Victimizations ${ordering}
    LIMIT ${numAgeResults})
    SELECT Least_Offenses.Neighborhood, FORMAT(Least_Offenses.Offense_Count, 0) AS Offense_Count,
FORMAT(Least_Gender_Victimizations.Gender_Victimizations, 0) AS Gender_Victimizations,
    FORMAT(Least_Age_Victimizations.Age_Group_Victimizations, 0) AS Age_Group_Victimizations
    FROM Least_Offenses JOIN Least_Gender_Victimizations ON Least_Offenses.Neighborhood =
Least_Gender_Victimizations.Neighborhood
    JOIN Least_Age_Victimizations ON Least_Offenses.Neighborhood =
Least_Age_Victimizations.Neighborhood
```

This query is used to allow users to filter neighborhoods to find those that meet their safety criteria. More specifically, it returns the neighborhoods that are in the top X safest in terms of the counts of a certain offense level AND in the top Y safest in terms of the counts of victimizations of a certain gender AND in the top Z safest in terms of the counts of victimizations of a certain age group. The offense level (felony, misdemeanor, or violation), gender (male or female), and age group (<18, 18-24, 25-44, 45-64, 65+) are user-determined, and so are the top X, Y, Z values for each of the three criteria. Users can also choose to switch from a “safest” neighborhoods search to a “most dangerous” neighborhoods search by toggling a highest/lowest variable. That is, selecting lowest would lead the search to find the neighborhoods in the bottom X, bottom Y, and bottom Z for the three criteria respectively.

Query 3 with Explanation

```
    WITH CRIME_STAT (Month, Year, Crime_Count,Neighborhood) AS (
```

```
    SELECT C.Month, C.Year, COUNT(C.ZipCode) as Crime_Count, zip.Neighborhood
    FROM Crime as C JOIN ( SELECT ZipCode, Neighborhood
    FROM ZipCodeNeighborhood ZCN
    WHERE ZCN.Neighborhood= '${id}') zip on zip.ZipCode = C.ZipCode
    GROUP BY C.Month,C.Year),
    DS AS (
    SELECT Rent.Neighborhood, Rent.Month, Rent.Year, Rent.AvgRent, Rent.MinRent, Rent.MaxRent,
    CRIME_STAT.Crime_Count, Concat(CAST(Rent.Year AS CHAR(4)), '-', CAST(Rent.Month AS
CHAR(2)),'-', '01') as datestring
    FROM CRIME_STAT RIGHT JOIN Rent ON Rent.Month = CRIME_STAT.Month AND Rent.Year =
CRIME_STAT.Year
    WHERE Rent.Neighborhood ='${id}'
    UNION
    SELECT Rent.Neighborhood, Rent.Month, Rent.Year,  Rent.AvgRent, Rent.MinRent, Rent.MaxRent,
CRIME_STAT.Crime_Count, Concat(CAST(CRIME_STAT.Year AS CHAR(4)), '-', CAST(CRIME_STAT.Month AS
CHAR(2)), '-', '01') as datestring
    FROM CRIME_STAT LEFT JOIN Rent ON Rent.Month = CRIME_STAT.Month AND Rent.Year =
CRIME_STAT.Year AND Rent.Neighborhood = CRIME_STAT.Neighborhood)
    SELECT *,  str_to_date(datestring, '%Y-%m-%d') as date
    FROM DS
    ORDER BY date;
```

This query returns the crime totals and rent average for a given neighborhood for every month/day combination in the database. This is used to populate a line graph on the Search page over time of rent and crime for a given neighborhood as details.

Query 4 with Explanation

```
    WITH NB AS (
    SELECT DISTINCT ZCN.Neighborhood, COUNT(ZCN.ZipCode) AS NumZipCodes
    FROM ZipCodeNeighborhood AS ZCN
    WHERE ZCN.Neighborhood LIKE '${name}'
    GROUP BY ZCN.Neighborhood
    UNION
    SELECT DISTINCT ZCN.Neighborhood, COUNT(ZCN.ZipCode) AS NumZipCodes
    FROM ZipCodeNeighborhood AS ZCN
    WHERE ZCN.ZipCode LIKE '${name}'
    GROUP BY ZCN.Neighborhood
    UNION
    SELECT DISTINCT ZCN.Neighborhood, COUNT(ZCN.ZipCode) AS NumZipCodes
    FROM ZipCodeNeighborhood AS ZCN JOIN NeighborhoodBorough NB ON ZCN.Neighborhood =
NB.Neighborhood
    WHERE NB.Borough LIKE '${name}'
    GROUP BY ZCN.Neighborhood)
    SELECT NB.Neighborhood, NB.NumZipCodes, GROUP_CONCAT(ZCN.ZipCode ORDER BY ZCN.ZipCode
SEPARATOR ', ') AS ZipCodes
    FROM NB JOIN ZipCodeNeighborhood ZCN ON NB.Neighborhood = ZCN.Neighborhood
    GROUP BY ZCN.Neighborhood
```

This query finds the neighborhoods that match either the corresponding zip code, borough, or neighborhood using unions and places them into a CTE. This CTE is then used to return the

neighborhoods, number of zip codes, and concatenation of zip codes into a single element. This is used on a table in the Search page for users to search for neighborhoods.

Query 5 with Explanation

```
WITH Borough_Rents AS (
SELECT NeighborhoodBorough.Borough, AVG(Rent.AvgRent) AS Average_Rent
FROM NeighborhoodBorough JOIN Rent ON NeighborhoodBorough.Neighborhood = Rent.Neighborhood
WHERE Rent.Year = ${year}
GROUP BY NeighborhoodBorough.Borough),
Borough_Crimes AS (
SELECT BoroughCrimesAllYears.Borough, BoroughCrimesAllYears.Crime_Count
FROM BoroughCrimesAllYears
WHERE BoroughCrimesAllYears.Year = ${year})
SELECT Borough_Crimes.Borough, CONCAT('$', FORMAT(IFNULL(Borough_Rents.Average_Rent, 0), 2)) AS
Average_Rent, FORMAT(Borough_Crimes.Crime_Count, 0) AS Crime_Count
FROM Borough_Crimes LEFT OUTER JOIN Borough_Rents ON Borough_Crimes.Borough =
Borough_Rents.Borough
ORDER BY Borough_Crimes.Borough;
```

This query calculates the average rent and total crime count for every borough for a user-specified year. In the Borough_Rents CTE, NeighborhoodBorough is joined with Rent in order to calculate average rent by borough for the selected year. In the Borough_Crimes CTE, the number of crimes for the selected year is retrieved for every borough. Then, Borough_Crimes is left joined with Borough_Rents; left join was used since there is no rent data for Staten Island. Rents and crime counts were formatted appropriately and the null value for Staten Island average rent was replaced with 0 to improve readability in the table.

Section 6: Performance Evaluation

Optimization	**Query 1 Time (seconds)**
None	48
Index on Crime(year)	37
Intermediate table 2020Crimes	2.3

Optimization	**Query 2 Time (seconds)**
None	40
Intermediate table 2020Crimes	< 1

Optimization	**Query 3 Time (seconds)**
None	42
Pushing down selection	37
Index on Crime(zipcode)	22
Index on Crime(zipcode, month, year)	1.4

Optimization	Query 5 Time (seconds)
None	40
Intermediate table BoroughCrimesAllYears	< 1

Each query was tested using DataGrip with id = ‘Borough Park’ when applicable. For Query 3, ZipCodeNeighborhood was originally joined to Crime with a where clause on the neighborhood. This was pushed to be joined inside a selection on ZipCodeNeighborhood which reduced the size of the join and sped up computation. Next, indexes were used to try and speed up the query by allowing for Index Nested Block Joins on ZIP code. Finally an index on (zipcode, month, year) was created for Crime which allowed for index-based aggregation instead of accessing the data record and significantly reduced computation time. Intermediate tables also improved performance because they either eliminated costly joins or reduced the size of tables participating in joins.

Section 7: Technical Challenges

Challenges Encountered and Solutions

To obtain ZIP codes from coordinates for the Crime table, we initially wanted to use the Google Maps API. However, we quickly realized that the API had daily call limits and throttled the number of requests made per second. If we were to use the Google Maps API, it would have taken much too long to process all of our crime records. To resolve this, we turned to the uszipcode library, which has no such limitations.

However, with millions of records, even uszipcode would take quite a long time to process all the records. Considering that Google Colab has a maximum runtime of 12 hours, running uszipcode on the entire dataframe at once was not an option. We decided to split the dataframe by year, creating a separate dataframe for each year from 2010 to 2020 (11 dataframes total). This would allow us to distribute processing among our group members by running different sections concurrently, increasing efficiency, while also keeping us well below the 12-hour runtime limit.

Another challenge we encountered was that some of our queries involving the Crime table would run for a very long time due to the large size of the table. To resolve this, we created various indexes and intermediate tables (further details can be found in section 6), which significantly improved query speed.

Appendix

Table 1: ZIP codes and borough associated with each New York City Neighborhood

List of Zip Codes by Neighborhood, New York City Five Boroughs (Bronx, Brooklyn, Manhattan, Queens, and Staten Island

Bronx	
Central Bronx	10453, 10457, 10460
Bronx Park and Fordham	10458, 10467, 10468
High Bridge and Morrisania	10451, 10452, 10456
Hunts Point and Mott Haven	10454, 10455, 10459, 10474
Kingsbridge and Riverdale	10463, 10471
Northeast Bronx	10466, 10469, 10470, 10475
Southeast Bronx	10461, 10462,10464, 10465, 10472, 10473
Brooklyn	
Central Brooklyn	11212, 11213, 11216, 11233, 11238
Southwest Brooklyn	11209, 11214, 11228
Borough Park	11204, 11218, 11219, 11230
Canarsie and Flatlands	11234, 11236, 11239
Southern Brooklyn	11223, 11224, 11229, 11235
Northwest Brooklyn	11201, 11205, 11215, 11217, 11231
Flatbush	11203, 11210, 11225, 11226
East New York and New Lots	11207, 11208
Greenpoint	11211, 11222
Sunset Park	11220, 11232
Bushwick and Williamsburg	11206, 11221, 11237
Manhattan	
Central Harlem	10026, 10027, 10030, 10037, 10039
Chelsea and Clinton	10001, 10011, 10018, 10019, 10020, 10036
East Harlem	10029, 10035
Gramercy Park and Murray Hill	10010, 10016, 10017, 10022
Greenwich Village and Soho	10012, 10013, 10014
Lower Manhattan	10004, 10005, 10006, 10007, 10038, 10280
Lower East Side	10002, 10003, 10009
Upper East Side	10021, 10028, 10044, 10065, 10075, 10128
Upper West Side	10023, 10024, 10025
Inwood and Washington Heights	10031, 10032, 10033, 10034, 10040
Queens	
Northeast Queens	11361, 11362, 11363, 11364
North Queens	11354, 11355, 11356, 11357, 11358, 11359, 11360
Central Queens	11365, 11366, 11367
Jamaica	11412, 11423, 11432, 11433, 11434, 11435, 11436
Northwest Queens	11101, 11102, 11103, 11104, 11105, 11106
West Central Queens	11374, 11375, 11379, 11385
Rockaways	11691, 11692, 11693, 11694, 11695, 11697
Southeast Queens	11004, 11005, 11411, 11413, 11422, 11426, 11427, 11428, 11429
Southwest Queens	11414, 11415, 11416, 11417, 11418, 11419, 11420, 11421
West Queens	11368, 11369, 11370, 11372, 11373, 11377, 11378
Staten Island	
Port Richmond	10302, 10303, 10310
South Shore	10306, 10307, 10308, 10309, 10312
Stapleton and St. George	10301, 10304, 10305
Mid-Island	10314