

A Natural Language Interface for Exploring NYC Open Data: Enhancing Public Access to Data

Tak Watanabe

Thesis
Major Studio II
Data Visualization

“Submitted in partial fulfillment of the requirements for the degree of Master Science in Data Visualization at Parsons School of Design.”

Thesis Faculty: Daniel Sauter

May, 2025

Abstract

Open-data portals hold vast civic insights, yet many residents cannot exploit them because exploration still demands specialist tools.

This thesis presents a natural-language interface (NLI) that lets users query New York City's 311 Service Requests in plain English. A curated semantic model grounds field names, types, and synonyms, while an LLM-based pipeline—comprising a Query Translator, SQL Generator, and Insight Generator—turns free-text questions into executable SQL and concise narrative summaries. To ensure reliable visuals, a rule-based Chart Selector chooses among twelve chart families (bars, lines, treemaps, choropleths, etc.) based on dimension types, additivity, and cardinality.

Powered by a modern large-language model, the prototype answers queries such as “Which borough saw the sharpest rise in rat complaints last year?” in real time, supplying an appropriate chart, title, and concise insight. In an author-run evaluation spanning ten tasks—overview, trend, geographic drill-down, follow-up, and “why” questions—the system consistently produced context-appropriate visualisations and substantively correct answers.

Although currently limited to a single dataset, the modular architecture demonstrates how recent advances in language models can translate the promise of open data into practical, inclusive access for journalists, policymakers, and the public.

Table of Contents

Abstract	2
Table of Contents	3
1. Introduction	4
1.1 Current Open Data Challenges	4
1.1.1 Data Aggregation Challenges	4
1.1.2 Data Visualization Challenges	6
1.2 The Proposed Natural Language Interface	8
1.2.1 Why Natural Language Query Is a Viable Solution	8
1.2.2 Why This Approach Is Feasible Today	9
1.2.3 Relevant Initiatives in the Business Intelligence Field	10
2. Treatment	10
2.1 Dataset	10
2.2 System Design	10
2-2-1. Semantic Model	11
2-2-2. Query Translator	13
2-2-3. SQL Generator	14
2-2-4. Database	15
2-2-5. Chart Selector	16
2-2-6. Insight Generator	23
2-2-7. How LLM is used in this system	25
2-2-8. System Design Summary	26
2.3 User Interface Design	26
2-3-1. Top Page	26
2-3-2. Application Page	28
3. Evaluation	29
3-1. Question List	29
3-2. Evaluation Summary	39
4. Room for Improvement	39
5. Conclusion	40

1. Introduction

1.1 Current Open Data Challenges

Open data portals have boosted transparency, sparked innovation, and engaged communities. Still, non-technical users often struggle to make full use of these platforms.

NYC Open Data, which today hosts nearly 4,000 datasets¹, faces this issue as well. A 2017 study found that many users saw it as a site for "a niche community of civic hackers and tech journalists"² only. This indicates that the range of audience is limited to those with specialized skills.

This research focuses on overcoming technical barriers in the processes of aggregating and visualizing data. Although these steps are necessary for gaining insights from open data, they implicitly require specialized knowledge.

The next section explains how data aggregation and visualization can be obstacles for users trying to convert raw data into useful information.

1.1.1 Data Aggregation Challenges

Data aggregation is the task of transforming raw data into the level of detail required for analysis. This is typically achieved through row filtering and value grouping. Since the raw granularity of a dataset rarely matches the desired output exactly, aggregation is essential for most analyses.

To perform data aggregation with open data, there are two main ways:

Approach 1: Download and Process Locally

Portals often allow you to download files in CSV, Excel, or JSON format. With these files, users can reshape them on their own computers. However, large files with over a million rows can crash an ordinary laptop without powerful hardware or a cloud environment. This restricts large-scale work to those with specialized hardware, which undermines the goal of open access.

Approach 2: Use Portal's Built-In Tools

¹ SCOUT Project, NYC Open Data, accessed May 14, 2025.

<https://opendata.cityofnewyork.us/projects/scout/>

² Understanding the Users of Open Data: Research Findings. Prepared by Reboot for NYC Open Data. May 2017.

https://opendata.cityofnewyork.us/wp-content/uploads/2017/07/Understanding-the-Users-of-Open-Data_Reboot.pdf

Some portals offer aggregation features that allow users to filter and group data before downloading it. This shifts the computational burden to the portal's servers, eliminating the need for users to rely on powerful local machines.

Figures 1 and 2 illustrate the 'Query Data' screens on NYC Open Data, where users can set filters and configure group-and-aggregate options.

Figure 1. NYC Open Data “Query Data” Interface – Setting Filters ³

The screenshot shows the NYC Open Data "Query Data" interface. At the top, there is a navigation bar with links for Home, Data, About, Learn, Contact Us, and Sign In. Below the navigation is a search bar and an export button. The main area displays a table of data with columns: Unique Key, Created Date, Closed Date, Agency, Agency Name, Complaint Type, Descriptor, and Location. The table lists numerous complaints from the NYPD, categorized by type like Drinking, Noise, and Illegal Parking, and further detailed by descriptor and location. A footer note indicates 1-100 of 3988309 rows. Below the table is a sidebar titled "Filters" with a "Clear all" link. It contains a dropdown for "Created Date" set to "is between" May 12, 03:42:55 and May 12, 03:42:55, connected by an "AND" operator. There is also a "+ AND" button and an "Apply" button at the bottom of the sidebar.

³ Screenshot of the Query Data filter interface from the NYC Open Data portal. Accessed May 12, 2025. <https://opendata.cityofnewyork.us/>.

Figure 2. NYC Open Data “Query Data” Interface – Specifying Group & Aggregate⁴

The screenshot shows the NYC Open Data "Query Data" interface. At the top, there is a navigation bar with links for Home, Data, About, Learn, Contact Us, a search bar, and a Sign In button. Below the navigation bar is a table displaying data from the "complaints" dataset. The table has columns for Unique Key, Created Date, Closed Date, Agency, Agency Name, Complaint Type, Descriptor, and Location Type. The first three rows of the table are highlighted.

Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type
64905737	05/11/2025 01:50:46 AM		NYPD	New York City Police Department	Drinking	In Public	Residential Buildi
64905268	05/11/2025 01:50:42 AM		NYPD	New York City Police Department	Noise - Residential	Loud Talking	Residential Buildi
64905963	05/11/2025 01:50:40 AM		NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk

Below the table, there is a "Group & aggregate" section. It includes a "Group by" section where "Based on" is set to "Created Date" and an "Aggregate by" section where "Unique Key" is aggregated with "count". An "Apply" button is located at the bottom of this section.

These tools provide flexible data manipulation capabilities but implicitly require users to be familiar with basic data transformation knowledge. This includes understanding data types, the difference between "Count" and "Count Distinct," and how "Group By" works. Without this knowledge, many users may find the interface confusing, limiting its reach to people with data skills.

1.1.2 Data Visualization Challenges

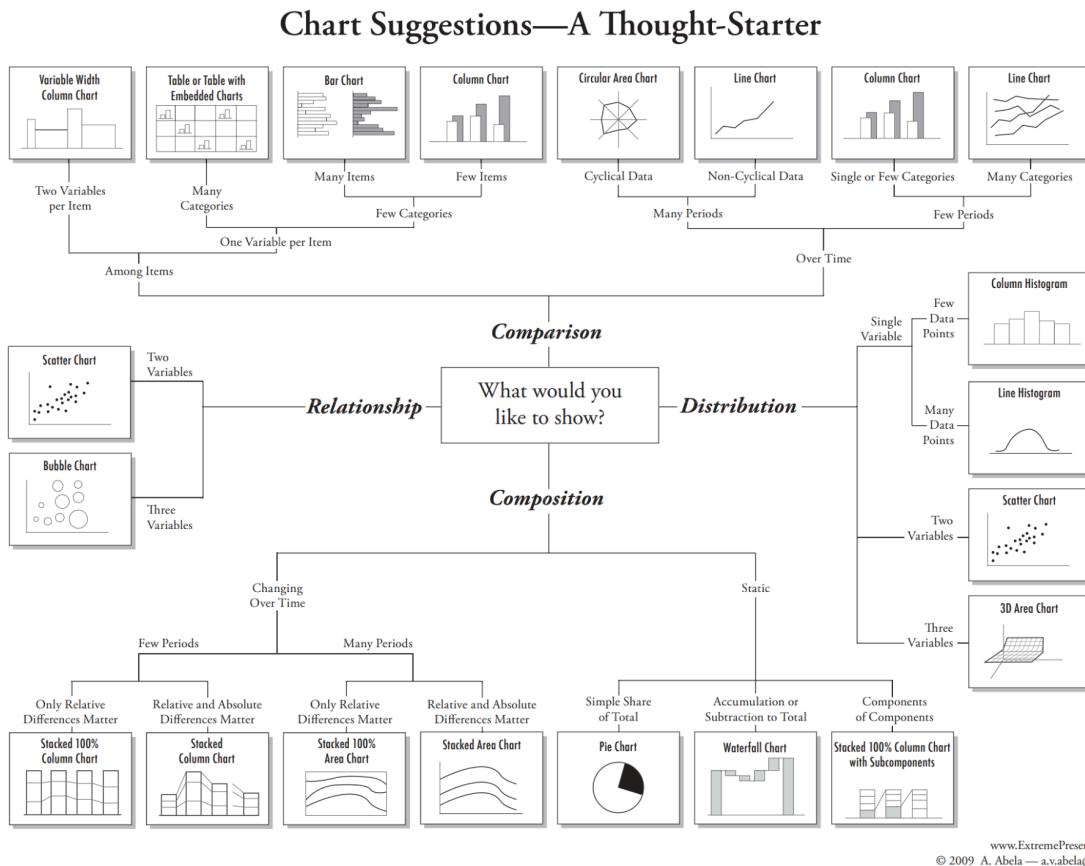
Even after aggregating data to an appropriate level of detail, users are still faced with the challenge of data visualization.

1. Choosing the Right Chart

To efficiently identify patterns or communicate insights from data, it is important to choose an appropriate chart type (e.g., bar, line, or area) based on the intended message. Figure 3 presents Dr. Andrew Abela's 2009 chart selection guide, which outlines how to select the most suitable chart based on factors such as the type of analysis, number of variables, and volume of data points. This highlights the complexity involved in designing an effective chart selection algorithm.

⁴ Screenshot of the Query Data Group & Aggregate interface from the NYC Open Data portal. Accessed May 12, 2025. <https://opendata.cityofnewyork.us/>.

Figure 3. Chart selection diagram (“Chart Chooser”) created in 2009 by Dr. Andrew Abela⁵



2. Software or Coding Proficiency

Even when users have a specific chart in mind, creating it typically requires software such as Excel or Tableau, or programming languages like Python or R. While some knowledge workers are familiar with spreadsheet tools like Microsoft Excel, a 2022 survey of 1,001 office workers found that only 44% considered chart creation easy⁶. This implies that more than half lack confidence in basic charting tasks, highlighting that even widely used tools can present challenges for data visualization.

In short, despite the abundance of open data, the processes of aggregating and visualizing it often act as barriers, potentially preventing interested users from effectively engaging with the

⁵ Andrew Abela, *Chart Chooser*, Extreme Presentation, accessed May 12, 2025, https://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html

⁶ New Research Shows Almost Half of Users Struggle to Make Sense of Data in Spreadsheets. ThoughtSpot, October 27, 2022. Accessed May 14, 2025. <https://www.thoughtspot.com/press-releases/new-research-shows-almost-half-of-users-struggle-to-make-sense-of-data-in-spreadsheets>

data. The following chapters will demonstrate how a natural-language interface can help reduce these obstacles and enable a broader audience to realize the full potential of open data.

1.2 The Proposed Natural Language Interface

This thesis proposes a natural language interface (NLI) as a solution to the technical challenges outlined above. Through this interface, users can pose questions directly to the data, with the system automatically handling data aggregation using the underlying database and generating visualizations on the front end. The following section explains why natural language is well-suited to these tasks and how recent advances in large language models (LLMs) have made such capabilities increasingly practical.

1.2.1 Why Natural Language Query Is a Viable Solution

When seeking insights from data, people often begin with a question such as, 'When are certain types of complaints most prevalent?' or 'Where are the hotspots for noise complaints?' This paper refers to such questions as natural-language queries (NLQs).

Although NLQs are expressed informally, an NLQ usually carries the main ingredients required for a formal analytical query. In business-intelligence and OLAP practice, those ingredients fall into three object types⁷:

- **Dimension:** A descriptive attribute that organises or groups data (e.g. Complaint Type, Borough). In SQL a dimension usually appears in the GROUP BY clause.
- **Measure:** A value to be summarised, usually numeric (e.g. Number of Complaints, Average response time). Measures are calculated via aggregation functions such as SUM, AVG, MIN, MAX, or COUNT.
- **Filter:** A predicate that restricts the dataset, applied either before aggregation (WHERE) or after aggregation (HAVING / front-end slicers) in SQL.

The specific combination of dimensions, measures, and filters constitutes an Aggregation Definition. Once this triplet is identified, the engine can translate the natural-language query (NLQ) into SQL and execute it against the database.

Consider the NLQ: "*How have 311 noise complaints changed over the past five years in Manhattan?*" From this sentence we can extract the aggregation definition:

- **Dimensions:** Time (weekly or daily)
- **Measure:** Count of noise complaints
- **Filters:**
 - **Area** is Manhattan
 - **Complaint Type** is Noise

⁷ Ahmed Yasir Khan, *What is OLAP (Online Analytical Processing): Cube, Operations & Types*, accessed May 14, 2025,

<https://storage.googleapis.com/kaggle-forum-message-attachments/884701/15858/What%20is%20OLAP.pdf>

- **Time Span** is Five years

With these elements the query generator emits SQL that (i) filters on borough, complaint type, and date, (ii) groups by the chosen time grain, and (iii) counts complaints—returning a time-series result that can be plotted.

At the same time, NLQs often imply not only what to aggregate but how to visualise it. Following Dr Andrew Abela's Chart Chooser⁸, common linguistic cues map to chart families:

- **Time-series language (“changed over time”)** → **Line chart**
- **Relationship language (“more frequent in denser areas”)** → **Scatter plot**
- **Comparison language (“which borough had the biggest rise?”)** → **Bar chart**

Additional chart-selection criteria can also be automatically derived from the aggregation definition or its resulting dataset:

- **Number of variables** → **count of Measures in the Aggregation Definition**
- **Data points** → **number of rows returned by the query**

In summary, by (1) extracting an Aggregation Definition from the user's question and (2) applying chart-selection heuristics, the NLI can move from an NLQ to an actionable visual insight without human intervention.

1.2.2 Why This Approach Is Feasible Today

Recent advances in large-language-model research have dramatically improved computers' understanding of everyday English. Gemini 1.5, for example, can resolve pronouns like “the protagonist” across an entire novel-length context⁹. In production, Gemini in BigQuery already turns free-form questions into fully-formed SQL, handling synonyms and comparison language without rigid keyword lists¹⁰, underscoring how far general LLM comprehension has advanced.

Another helpful advancement is the Long Context Window. For instance, Google's Gemini 2.0 Flash model provides users with a one-million-token context window for each prompt, equivalent to eight average-length English novels¹¹. With this feature, AI can attempt to understand user queries by consulting the data dictionary, which stores metadata in the database.

Since the model handles the complexity, the rest of the system remains simple. A short prompt converts natural language into the query format of NYC Open Data, and a small code snippet sends it to a charting tool that handles visualization. This leap in language understanding not

⁸ Abela, Chart Chooser

⁹ *Gemini 1.5: Unlocking Multimodal Understanding Across Millions of Tokens of Context*. Google DeepMind Technical Report, 2024. Accessed May 14, 2025.

<https://storage.googleapis.com/cloud-samples-data/generative-ai/pdf/2403.05530.pdf>

¹⁰ “Write Queries with Gemini Assistance.” Google Cloud BigQuery Documentation. Accessed May 14, 2025. <https://cloud.google.com/bigquery/docs/write-sql-gemini>

¹¹ “Long Context.” Gemini API Documentation. Google AI for Developers. Last updated April 24, 2025. Accessed May 14, 2025. <https://ai.google.dev/gemini-api/docs/long-context>

only makes interfaces smarter, but also makes the entire system leaner. This enables powerful features to run in lightweight projects.

1.2.3 Relevant Initiatives in the Business Intelligence Field

The business intelligence (BI) field has recently seen significant growth in tools that incorporate natural language and conversational analytics. These tools aim to make data querying more accessible, especially for non-technical users. Two notable examples include:

ThoughtSpot: Spotter¹²

ThoughtSpot, a leader in search-driven analytics, introduced Spotter, an AI-powered assistant that not only responds to user queries but also proactively suggests insights and visualizations. This represents a shift toward AI-driven, self-service analytics in the BI industry.

Tableau: Tableau Next¹³

At the 2025 Tableau Conference, Tableau introduced Tableau Next—a fully redesigned platform that integrates autonomous analytics powered by AI. This new platform allows users to generate visualizations simply by asking questions in natural language, moving beyond the traditional drag-and-drop approach.

These innovations serve as inspiration for improving open data portals. Incorporating conversational interfaces and proactive assistance can dramatically reduce barriers to data use, particularly for users without technical backgrounds.

2. Treatment

This section outlines how the proposed natural language interface is constructed and how its core components interact within the system.

2.1 Dataset

As a proof of concept for natural language interfaces (NLIs) applied to open data, this thesis focuses on the 311 Service Requests from 2010 to Present dataset¹⁴, available through NYC Open Data. This dataset was selected for its richness and granularity, as it captures a broad spectrum of public service complaints—ranging from noise and heating issues to rodent sightings and sanitation concerns—submitted by residents across New York City.

¹² “ThoughtSpot Spotter” ThoughtSpot. Accessed May 14, 2025.

<https://www.thoughtspot.com/product/ai-analyst>

¹³ “Tableau Next.” Tableau. Accessed May 14, 2025. <https://www.tableau.com/products/tableau-next>

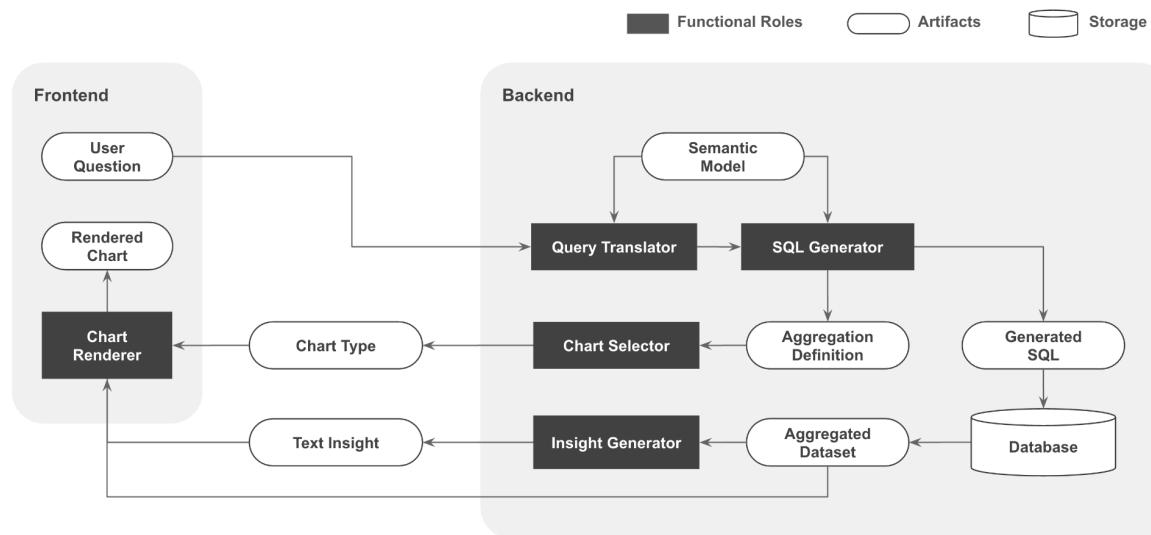
¹⁴ “311 Service Requests from 2010 to Present.” NYC Open Data. Accessed May 14, 2025.

https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9/about_data

2.2 System Design

Figure 4 (below) illustrates the overall workflow, outlining the process from a user's plain-language query to the generation of a final chart. The diagram includes several key components. The following subsection describes the role of key components and explains its importance within the natural language interface, following the order of data flow.

Figure 4. Overall Workflow of a natural language interface



2-2-1. Semantic Model

The Semantic Model serves as a dictionary for the data stored in the NYC 311 dataset. It plays a critical role in enabling the LLM to infer relevant field names from user-submitted natural language queries.

The semantic model stores key metadata, including:

- the physical name of each field in the database,
- the display name shown to users,
- the data type,
- a detailed description intended for the AI,
- a concise description intended for users, and
- relevant synonyms.

For dimensions, the model explicitly distinguishes between categories such as *geographic dimensions* (e.g., borough or neighborhood) and *temporal dimensions* (e.g., report created date and closed date).

For measures, the semantic model includes the corresponding SQL expressions used by the system for aggregation. This allows the LLM to apply the formulas directly, without needing to resolve dialect-specific variations in SQL syntax.

The example of Semantic Model (Dimension)

```
{  
    "physical_name": "created_date",  
    "display_name": "Request Date",  
    "data_type": "date",  
    "description": "DATE (YYYY-MM-DD) of service request creation.",  
    "description_to_user": "The exact date when people submitted their service  
requests to the city.",  
    "synonym": [  
        "date",  
        "day",  
        "submission date",  
        "date submitted",  
        "reported date",  
        "date reported"  
    ],  
    "data_source_id": 1  
},
```

The example of Semantic Model (Measure)

```
{  
    "physical_name": "requests_per_day",  
    "display_name": "# of Requests per Day",  
    "expression": "round(count(1) / count(distinct created_date), 2)",  
    "data_type": "float",  
    "description": "Average number of requests per day.",  
    "description_to_user": "The average number of service requests submitted each  
day.",  
    "synonym": [  
        "daily requests",  
        "requests per day",  
        "daily average",  
        "average daily requests",  
        "requests per day average"  
    ],  
    "data_source_id": 1  
},
```

,

2-2-2. Query Translator

The Query Translator plays a central role in the natural language interface by converting free-form user queries into machine-readable, structured formats. This leverages the LLM begins by identifying the type of query submitted and then translates it into structured data aggregation logic. This process involves extracting key components such as dimensions, measures, and filters.

In the prototype system, queries are classified into nine categories, each associated with a specific reasoning strategy that guides the generation of an optimized aggregation definition.

Query Type	Example Questions	Reasoning Strategy
Simple Data Aggregation	"Show me all noise complaints in Brooklyn" "What are the most common 311 complaints?"	Identify the primary entity and measure of interest. Determine if filtering is needed. Focus on delivering a straightforward snapshot of the data without time progression.
Composition/Percentage	"What percentage of complaints come from Manhattan?" "How is the distribution of complaint types across boroughs?"	Understand how parts relate to the whole. Avoid filtering out the comparison basis so that relative proportions can be calculated accurately.
Trend Analysis	"How have rat complaints changed over the past year?" "What's the trend of heating complaints during winter months?"	Look for time-related terms. Focus on showing value changes over time with appropriate granularity based on the analyzed time span.
Follow-Up Query	"What about in Queens?" (after seeing Brooklyn data) "Can you show this by month instead?"	Maintain continuity by preserving existing dimensions and measures while modifying only the explicitly changed components in the follow-up.
Dimension Drill-Down	"What specific types of noise complaints occur in Williamsburg?" "Break down these heating issues by specific complaint type"	Identify signals for deeper exploration. Move from broader to more specific categories while preserving existing filters and adding new ones as needed.
Top-N Query	"Show me the top 5 neighborhoods with the most rat sightings" "What are the 3 agencies with the slowest response times?"	Recognize explicit ranking requests. Sort by the measure of interest and limit results accordingly. Distinguish from general superlatives without specified counts.

Why-Type Question	"Why are there more noise complaints in summer?" "What explains the high number of complaints in this neighborhood?"	Identify if the question seeks explanation beyond the data. Determine whether it can be addressed with a breakdown or requires external contextual knowledge.
Locations Query	"Where are most heating complaints reported?" "Which streets have the most parking violations?"	Detect geographic intent. Start with broad geographic units (e.g., neighborhoods), then narrow down if further filtering is present.
Vague Queries	"Show me that" "Yes, please" "Let's do it"	Reference prior conversation history to infer the user's intent. Identify how the current input builds on previous context.

These classification rules are conveyed to LLM through system instructions, guiding the AI to interpret and handle user queries in a way that aligns with common expectations.

As a result, Query translator turns the user query into the structured aggregated definition as below:

Input :

What were the common complaint types in Manhattan last year?"

Output :

- Dimensions: ['complaint_type_large']
- Measures: ['num_of_requests']
- PreAggregationFilters: "borough must be exactly 'MANHATTAN' AND created_date must be within last year"

Input :

Which neighborhoods report the most heating issues in winter?

Output :

```
Dimensions: ['neighborhood_name']
Measures: ['num_of_requests']
PreAggregationFilters: "complaint_type_middle must be exactly 'Heat/Hot Water' AND created_month_datepart must be one of 1, 2, or 12"
```

2-2-3. SQL Generator

The SQL Generator is responsible for converting the structured output from the Query Translator into a database-specific query format using the LLM.

As a result, the SQL Generator produces queries using predefined DuckDB-specific syntax. Below is a subset of system instructions that is passed to the LLM be followed when generating SQL statements, particularly for date and location-based filtering:

- **Current Date:**
 - CURRENT_DATE
- **Date Truncation:**
 - date_trunc('YEAR', created_date)
 - date_trunc('MONTH', created_date)
- **Intervals:**
 - Three months ago: CURRENT_DATE - INTERVAL 3 MONTH
 - One year ago: CURRENT_DATE - INTERVAL 1 YEAR

As a result, Query translator turns the aggregated definition into the SQL as below:

Input :

```
Dimensions: ['complaint_type_large']
Measures: ['num_of_requests']
PreAggregationFilters: "borough must be exactly 'MANHATTAN' AND
created_date must be within last year"
```

Output:

```
SELECT
    complaint_type_large
    , count(1) AS num_of_requests
FROM requests_311
WHERE borough = 'MANHATTAN'
    AND created_date >= CURRENT_DATE - INTERVAL 1 YEAR
GROUP BY 1
ORDER BY num_of_requests DESC
LIMIT 5000;
```

2-2-4. Database

For this project, we use DuckDB¹⁵, an open-source, in-memory database capable of handling millions of records without requiring a dedicated database server. Using DuckDB allows this system to efficiently aggregate and transform data, including handling JSON and geospatial queries.

In addition to running the SQL query passed by the SQL Generator, DuckDB retrieves metadata about the dataset, such as

- Number of data points after data aggregation
- Basic statistics (average, median, and standard deviation) of the measure.
- The top and bottom three values of the dataset and their dimensions.

¹⁵ “DuckDB”. Accessed May 14, 2025. <https://duckdb.org/>

These values are passed to the chart selector and insights generator. This enables the subsequent system to provide more appropriate chart selection and insight generation based on the dataset's characteristics.

2-2-5. Chart Selector

The Chart Selector determines the most appropriate chart type for each natural language query (NLQ). This system supports the following twelve chart types, selected to cover a wide range of analytical needs:

1. Table

Displays data in a tabular format. While clean and familiar, it is less effective for large datasets, as users may need to scroll or navigate multiple pages.

Figure 5. Table

Complaint Type	Borough	# of Requests
Noise Issues	BRONX	1,154,096
Housing & Building	BRONX	1,089,185
Parking & Street Access	BROOKLYN	1,071,786
Noise Issues	MANHATTAN	1,046,033
Housing & Building	BROOKLYN	993,729
Parking & Street Access	QUEENS	965,270
Noise Issues	BROOKLYN	960,739
Sanitation Issues	BROOKLYN	774,991
Noise Issues	QUEENS	741,452
Housing & Building	MANHATTAN	707,502
Sanitation Issues	QUEENS	545,935
Housing & Building	QUEENS	491,116
Parking & Street Access	BRONX	465,582
Sanitation Issues	BRONX	355,698
Sanitation Issues	MANHATTAN	353,833

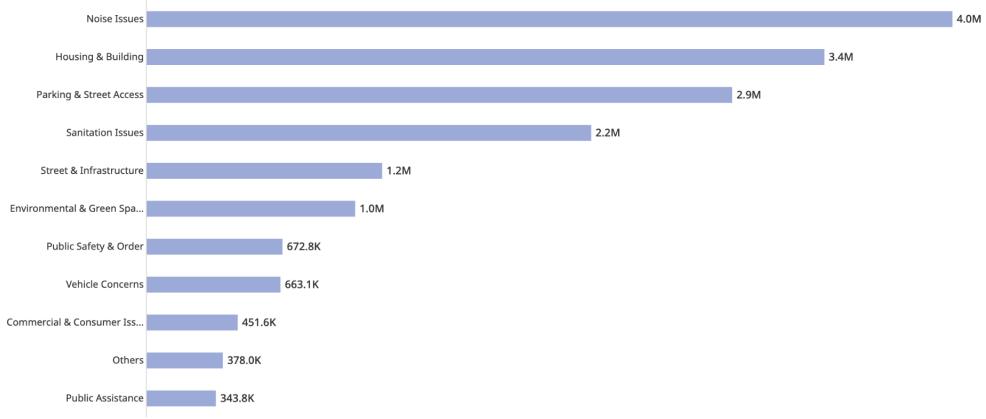
Showing 1 to 15 of 55 results

Previous | **1** | 2 | 3 | ... | 4 | Next

2. Single Bar Chart

A traditional horizontal bar chart used for comparing values across categories. Horizontal orientation improves readability of category labels.

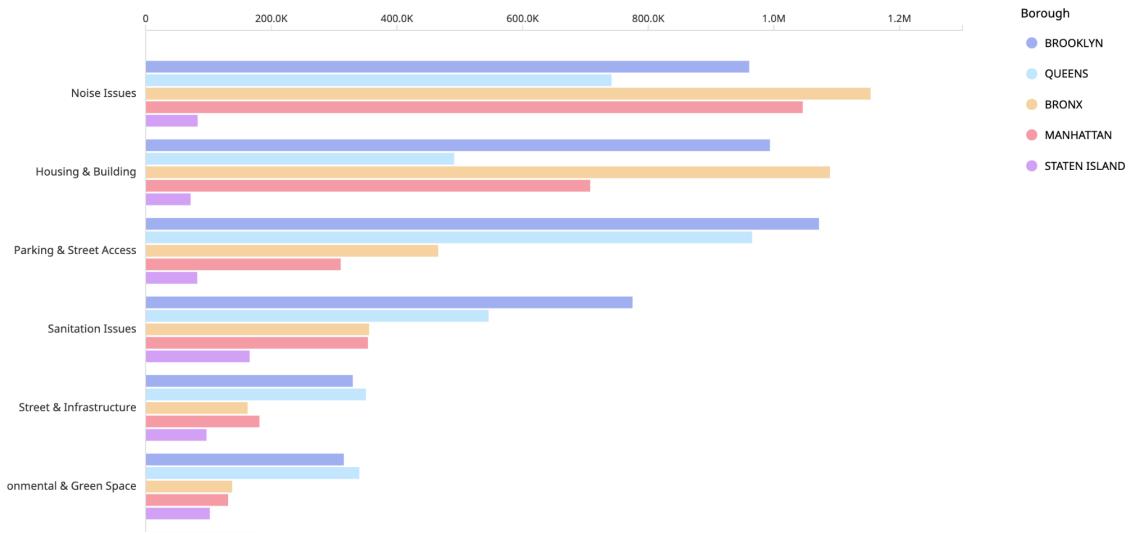
Figure 6. Single Bar Chart



3. Grouped Bar Chart

Displays multiple bars within each category, grouped by an additional dimension (e.g., time or sub-category). Useful for comparing values across subgroups.

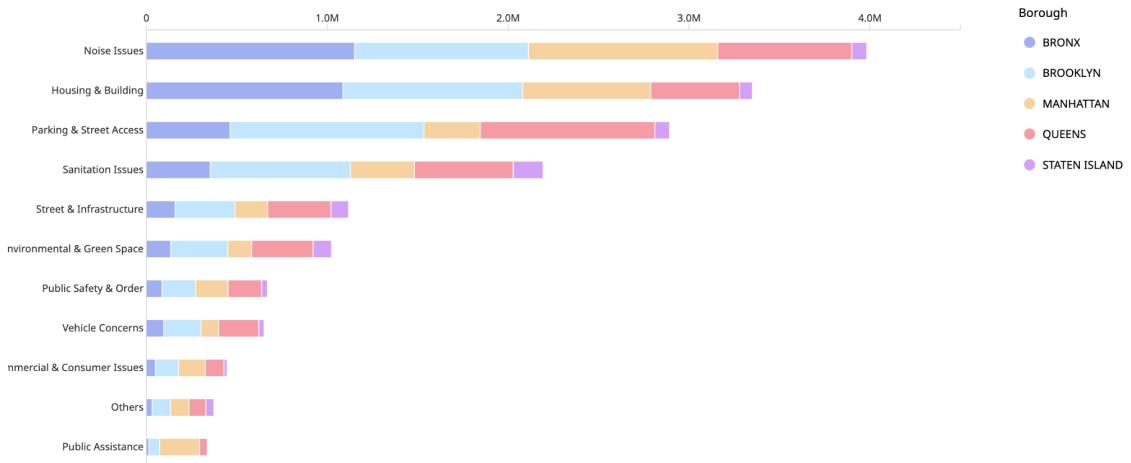
Figure 7. Grouped Bar Chart



4. Stacked Bar Chart

Stacks values within a single bar to show the total alongside a breakdown of components. Ideal for understanding composition within categories.

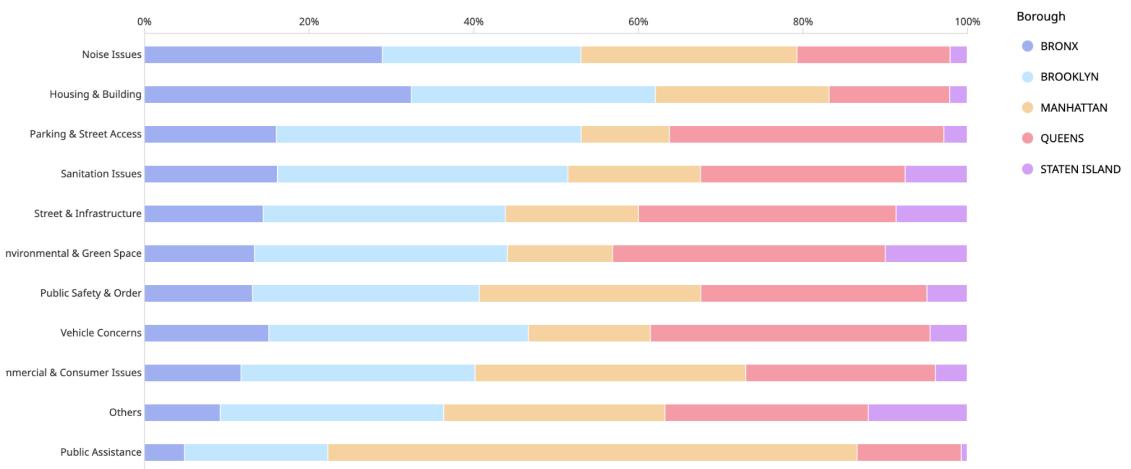
Figure 8. Stacked Bar Chart



5. Stacked Bar Chart 100%

A variation of the stacked bar chart where each bar represents 100%, allowing comparison of relative proportions rather than absolute values.

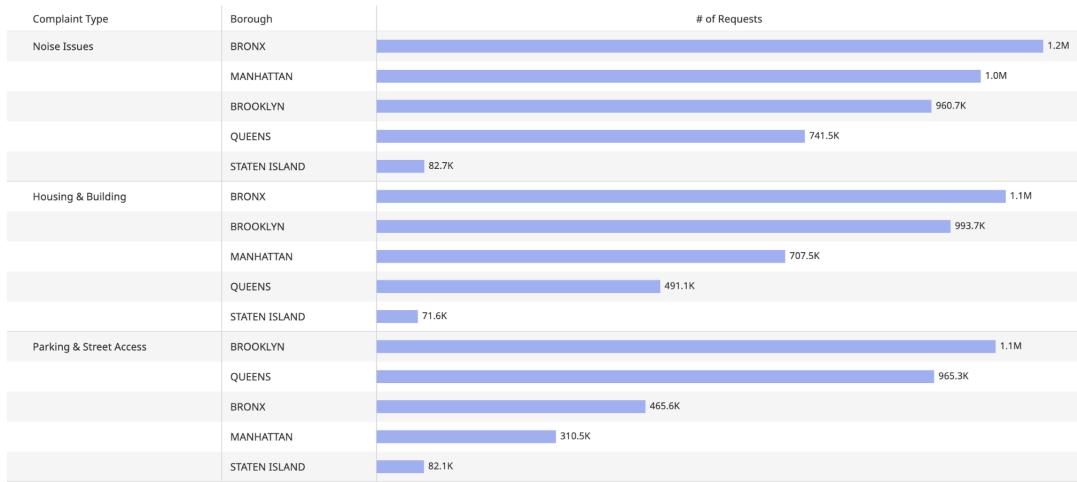
Figure 9. Stacked Bar Chart 100%



6. Nested Bar Chart

Displays hierarchical groupings across multiple categorical levels. Useful for comparing values within subgroups while maintaining a clear structural hierarchy.

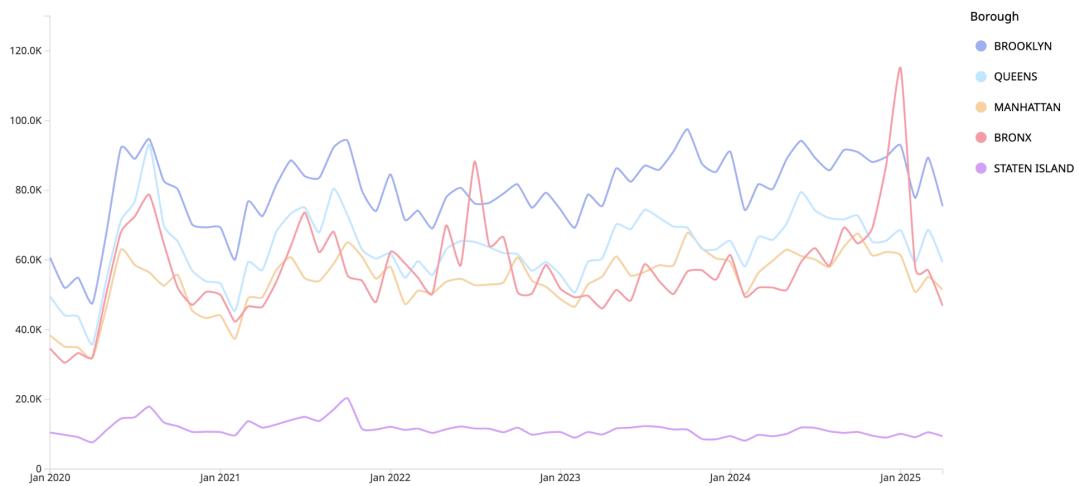
Figure 10. Nested Bar Chart



7. Line Chart

Ideal for trend analysis over time. Requires a time-related dimension and at least one measure. Can also include a categorical dimension for comparing multiple series.

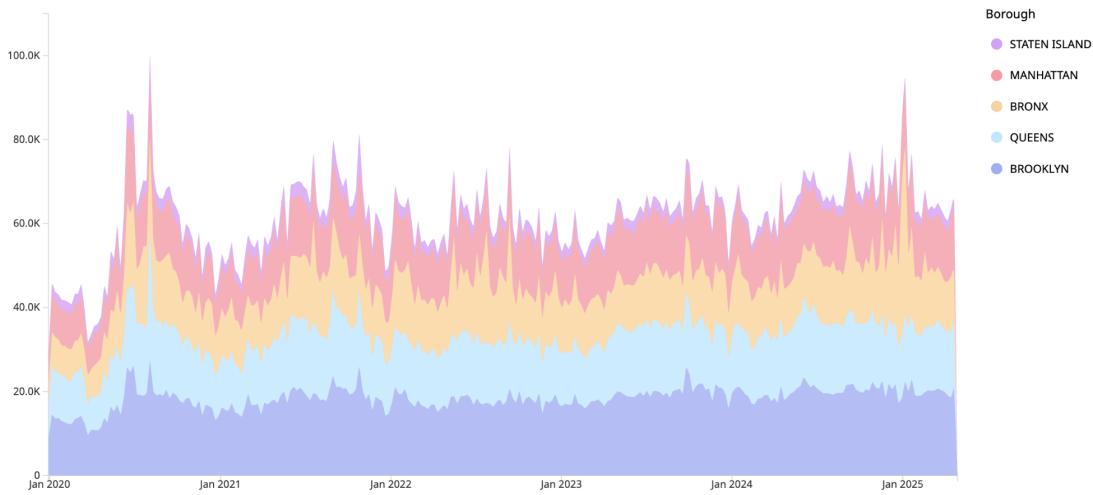
Figure 11. Line Chart



8. Stacked Area Chart

Visualizes how quantitative values change over time, with stacked areas illustrating the cumulative total and contributions of each category.

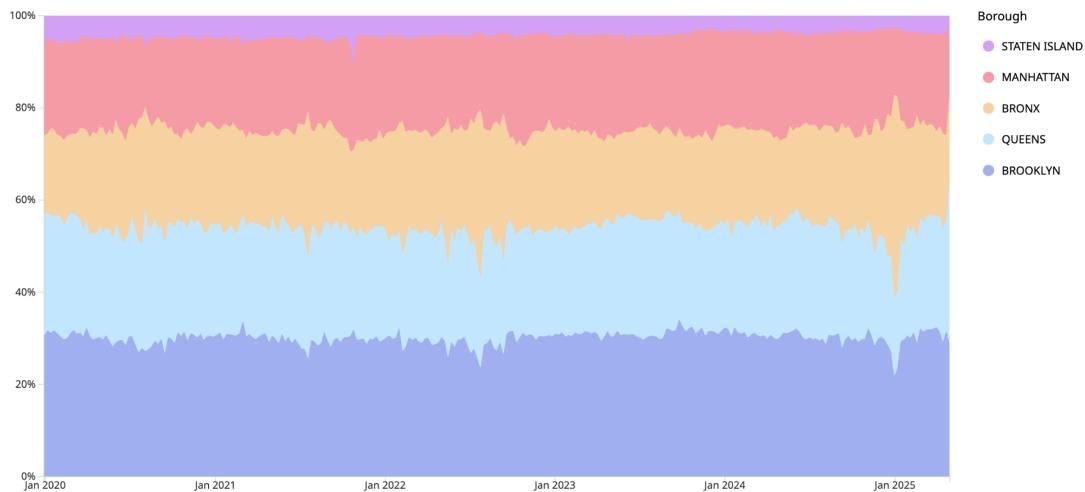
Figure 12. Stacked Area Chart



9. Stacked Area Chart 100%

Normalizes stacked area values to 100% at each time point, highlighting relative contributions rather than absolute totals over time.

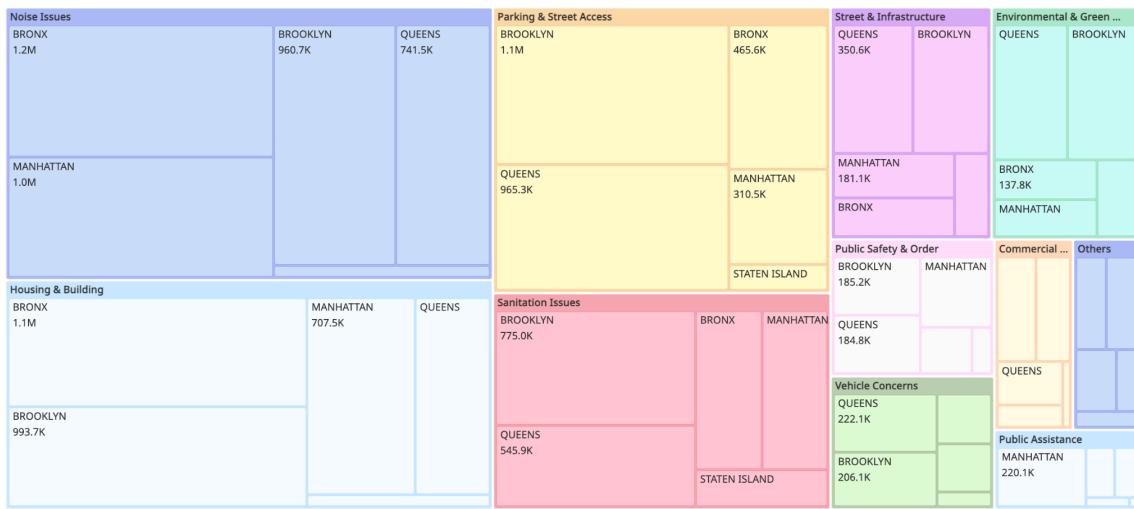
Figure 13. Stacked Area Chart 100%



10. Treemap

Represents hierarchical data using nested rectangles sized according to a quantitative measure. Useful for visualizing proportions within categories in a compact format.

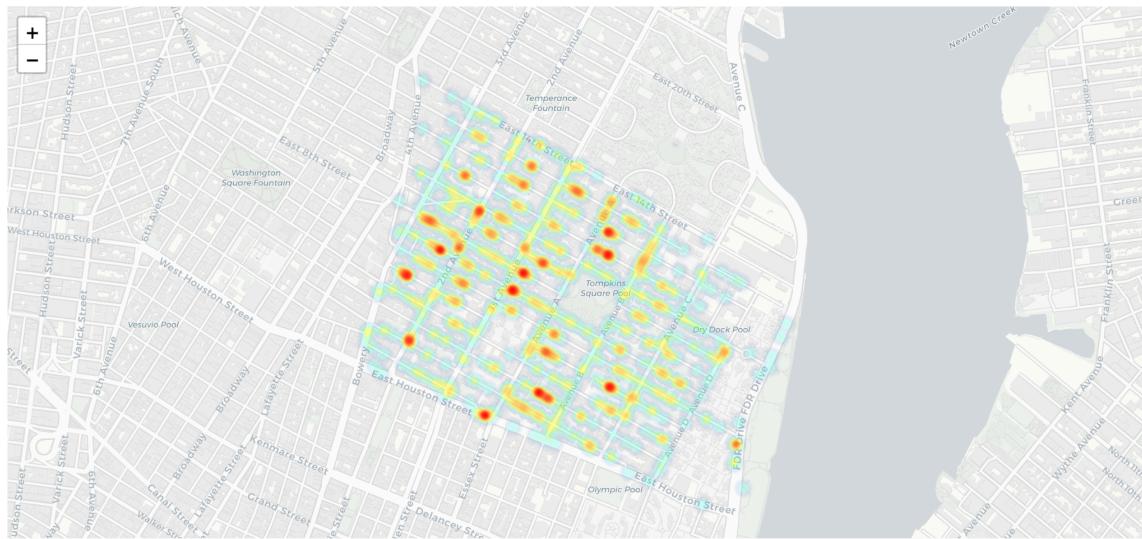
Figure 14. Treemap



11. Heatmap

Applies intensity-based color gradients to a geographic map to represent data concentration. Effective for identifying geographic hotspots or density distributions.

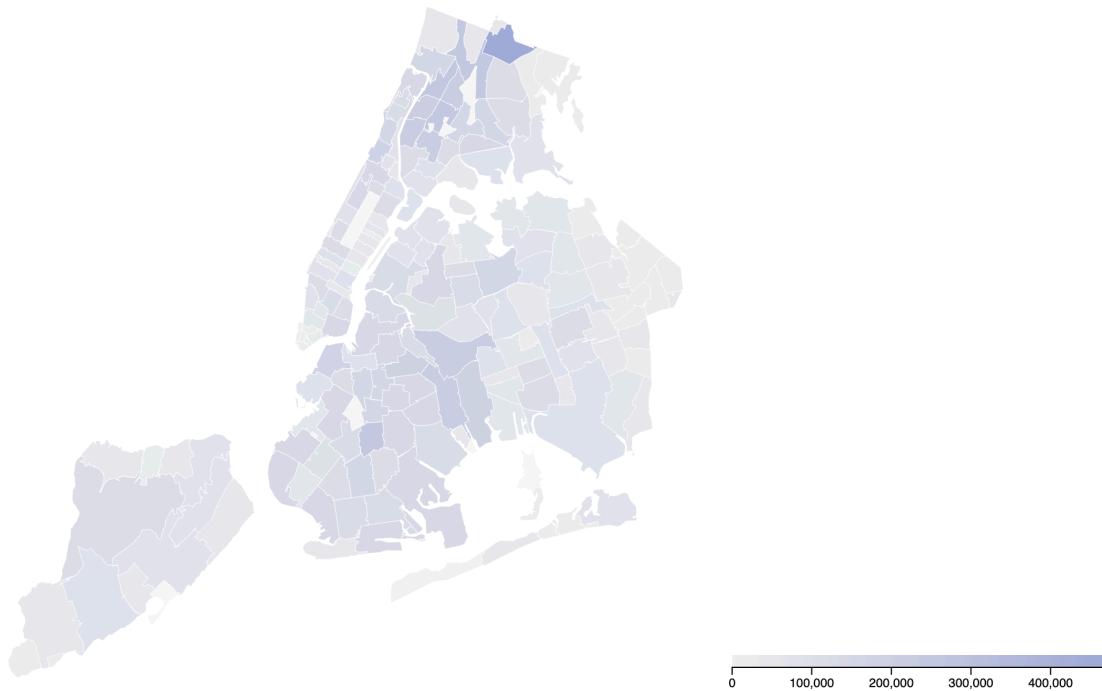
Figure 15. Heatmap



12. Choropleth Map

Uses color shading to represent data values across predefined geographic regions. Useful for analyzing spatial patterns and regional differences.

Figure 16. Choropleth Map



To ensure consistency and stability in chart selection, this system uses a rule-based approach rather than relying on one-shot AI predictions. The process begins by classifying dimensions and measures into more granular categories based on their data types and characteristics.

Dimension Classification

Dimensions are categorized as follows:

- **Time Dimension**
 - A date-type dimension, typically used as the x-axis in time-series charts such as line charts or area charts.
 - Examples: 311 request date
- **Categorical Dimension:**
 - A non-date, discrete value, including geographic dimension, used for grouping or breakdowns in various charts.
 - Examples: Complaint Category, Borough
- **Geographic Dimension:**
 - A spatial attribute used in map-based visualizations, such as choropleth maps or heatmaps.
 - Examples: Borough, Zip Code, Location Coordinates

Measure Classification

Measures are divided based on whether they support additive or compositional analysis:

- **Additive Measures:**

- These can be used as regular measures in various charts and are also meaningfully additive, making them suitable for stacked or compositional visualizations such as stacked bar charts, stacked area charts, or treemaps.
- Examples: Number of Complaints, Population
- **Non-Additive measure**
 - These do not retain their meaning when summed, as they often represent ratios or averages. As such, they should only be used with chart types that do not rely on stacked or compositional representations.
 - Examples: Number of Complaints per Day, Average Response Time

Additional Selection Criteria

To avoid generating cluttered or unreadable visualizations, the system also considers cardinality:

- **Cardinality of Dimensions**
 - Refers to the number of unique values within a dimension. High-cardinality dimensions may produce overly complex or messy charts.
 - Example: If a dimension used to break down each bar in a stacked bar chart has high cardinality, the chart is excluded from selection.

Based on the criteria outlined above, the system assigns chart types that are compatible with the NLQ. The table below demonstrates how these assignment rules are applied in practice.

Chart Type	Dimension	Measure	Other Requirements
Table	Any	Any	-
Single Bar Chart	1 categorical dimension	1 additive/non-additive measure	- No time dimensions - Dimension is not "location" (coordinate)
Grouped Bar Chart	Exactly 2 categorical dimensions	1 additive/non-additive measure	Not high cardinality
Stacked Bar Chart	Exactly 2 categorical dimensions	1 additive measure	Not high cardinality
Stacked Bar Chart 100%	Exactly 2 categorical dimensions	1 additive measure	Not high cardinality
Nested Bar Chart	1-2 dimensions	1-2 additive/non-additive measure	Either multiple categorical dimensions or multiple measures
Line Chart	- 1 time dimension - Up to 1 categorical dimension	1 additive/non-additive measure	Not high cardinality
Stacked Area Chart	- 1 time dimension - 1 categorical dimension	1 additive measure	Not high cardinality
Stacked Area Chart	- 1 time dimension	1 additive measure	Not high cardinality

100%	- 1 categorical dimension		
Treemap	1-2 categorical dimensions	1 additive measure	<ul style="list-style-type: none"> • No time dimensions
 • Not "location" dimension
 • Not a TOP-N query
Heatmap	1 "location" dimension (coordinate)	1 additive/non-additive measure	<ul style="list-style-type: none"> • None
Choropleth Map	1 geographic dimension (borough, county, neighborhood, zip)	1 additive/non-additive measure	<ul style="list-style-type: none"> • None

This method enables users to explore data using various and suitable visualizations.

2-2-6. Insight Generator

Insight generator provides a chart title and insight summary to provide efficient interpretation to users at a glance in the user interface. Below is one example.

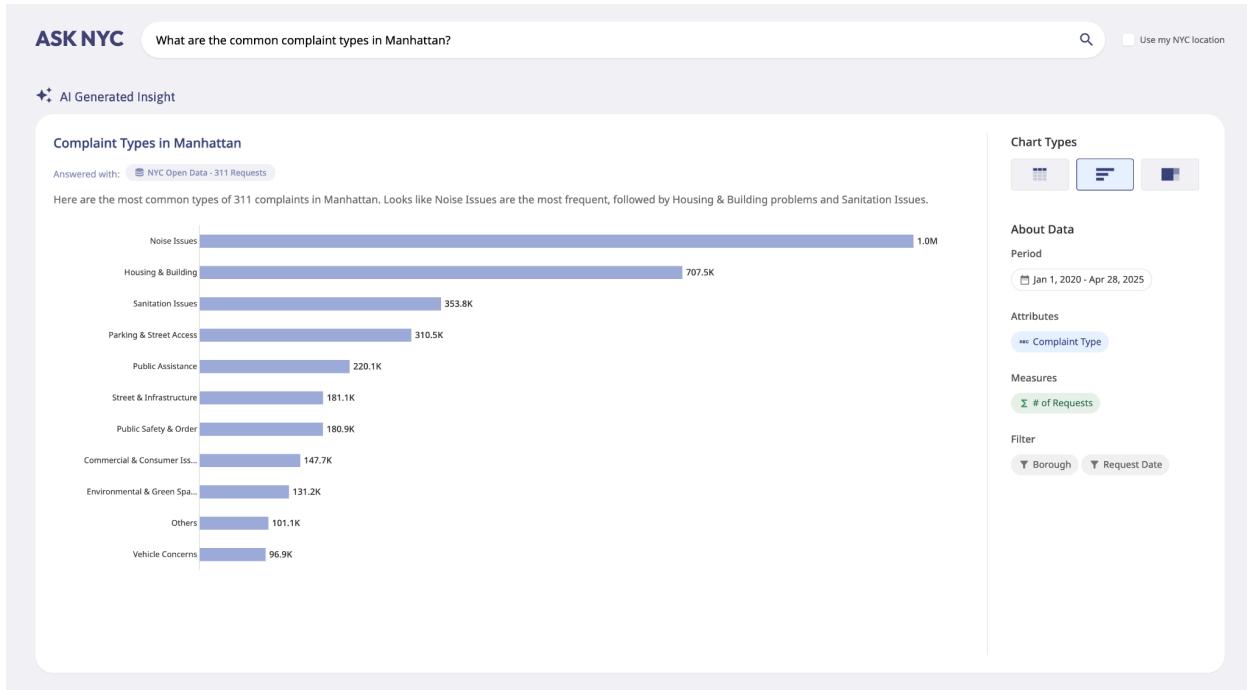
Input:

What are the common complaint types in Manhattan?

Insight Summary:

- Chart Title: Complaint Types in Manhattan
- Description: Here are the most common types of 311 complaints in Manhattan. Looks like Noise Issues are the most frequent, followed by Housing & Building problems and Sanitation Issues.

User Interface:



This description is generated by LLM by passing the following information.

- Subset of the result
- Metadata about the result set
 - Information about the field included in the result set
 - The time span used for the data
 - Number of data points in the dataset
 - Basic statistics (average, median, and standard deviation) of the measure.
 - The top and bottom three values of the dataset and their dimension values.

This also generates the explanation when the result set is slightly different from the user's expectations. Below is the example of when a user asks about "potholes" complaints but actually 311 dataset shows the data based on "street condition" complaints. This capability helps avoid users from misunderstanding the data while providing relevant information to users.

Input:

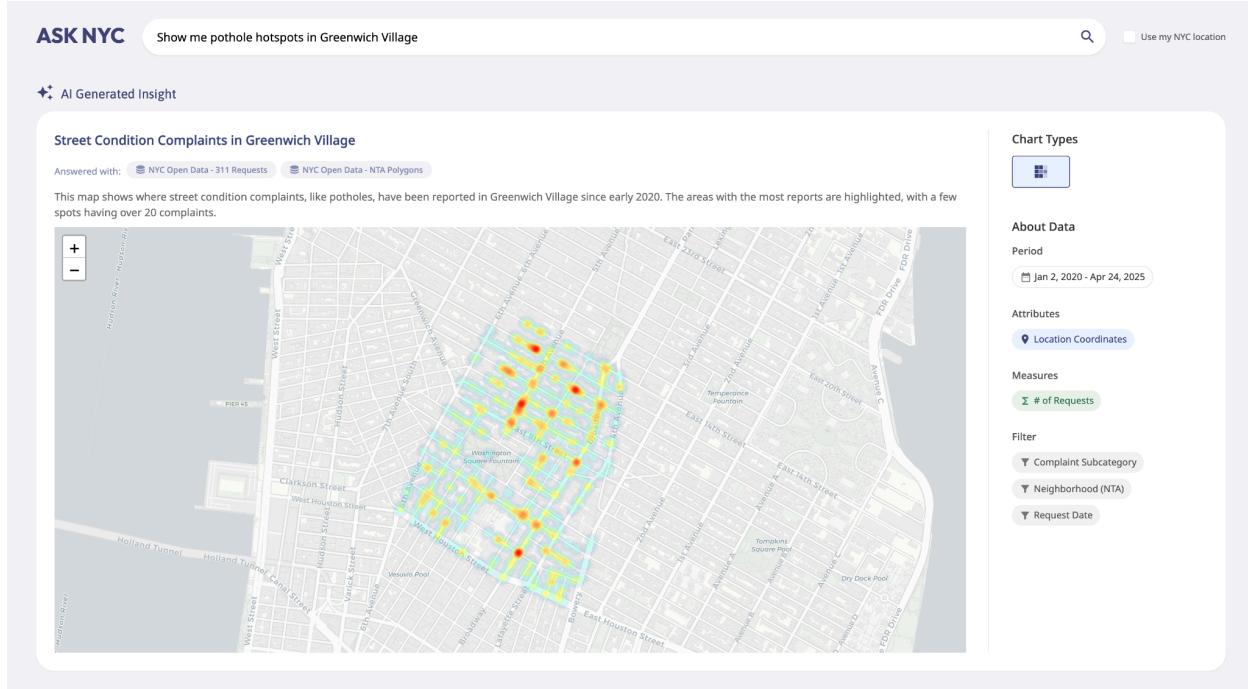
Show me pothole hotspots in Greenwich Village

Insight Summary:

- Chart Title: Street Condition Complaints in Greenwich Village
- Description: This map shows where street condition complaints, like potholes, have been reported in Greenwich Village since early 2020.

The areas with the most reports are highlighted, with a few spots having over 20 complaints.

User Interface:



2-2-7. How LLM is used in this system

Large language models (LLMs) play a critical role in this system, particularly within the Query Translator, SQL Generator, and Insight Generator components. Rather than adopting a single-shot approach—where the AI produces all necessary outputs (from query interpretation to insight generation) in one prompt—this system implements an agentic architecture. In this architecture, three AI-powered components each focus on a specific task: interpreting user queries (Query Translator), generating SQL syntax (SQL Generator), and producing insights (Insight Generator). This modular design enhances the overall quality and reliability of the output.

For the underlying AI model, various factors were considered, including accuracy, long-context support, cost, and performance. At the time of writing this thesis, the system utilizes Gemini 2.5 Flash¹⁶, provided by Google AI Studio¹⁷. This model was selected for its strong balance of reasoning capabilities, long context window capabilities, responsiveness, and cost-efficiency. It is well-suited for delivering real-time responses while supporting the complex reasoning needed for data aggregation and insight generation.

¹⁶ “Gemini 2.5 Flash – Generative AI on Vertex AI.” Google Cloud. Last updated May 14, 2025. Accessed May 14, 2025. <https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash>

¹⁷ “Google AI Studio.” Google. Accessed May 14, 2025. <https://aistudio.google.com>

Given the rapid advancement of LLMs, regularly evaluating and adopting the latest models may further enhance system performance and user experience.

2-2-8. System Design Summary

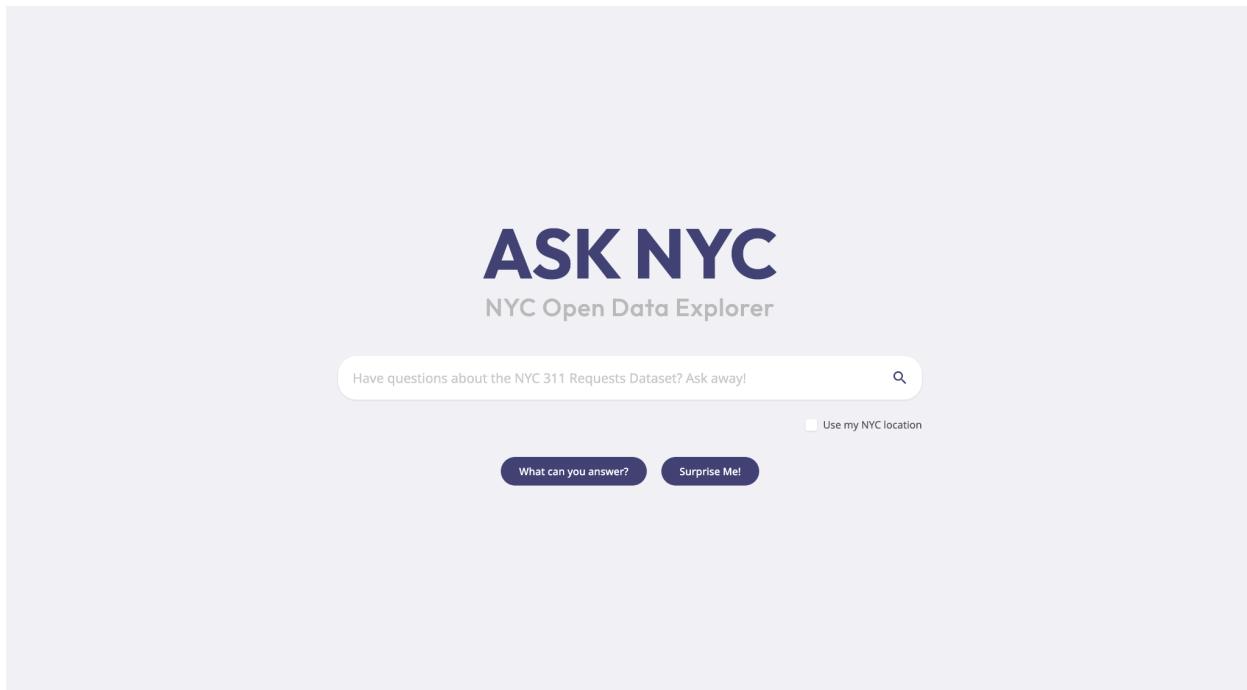
As demonstrated above, the system delivers comprehensive information in response to a single query by integrating aggregated data, recommended chart types, and concise insights—leveraging modern AI technologies. The following section outlines how this information is presented within the user interface.

2.3 User Interface Design

2-3-1. Top Page

The top page features a clean, search engine–like design, consisting of a logo, a search box, and a few functional buttons.

Figure 17. Top Page



Beneath the search box are two buttons designed to assist users who may be unfamiliar with the interface:

“What Can You Answer?” Button

This button submits a predefined question about the system to the AI and directs users to the application page, where the response outlines the types of information the platform can provide. This helps users better understand its scope and capabilities.

Figure 18. “What can you answer?” response

The screenshot shows the ASK NYC interface. At the top, there's a header with "ASK NYC" and a search bar containing "What can you answer?". A "Use my NYC location" checkbox is also present. Below the header, a section titled "AI Generated Insight" displays the following text:

I can help you explore data about NYC 311 service requests! I have information on things like:

- * Different types of complaints (like noise, heating, or parking issues)
- * Where complaints are reported (by borough, neighborhood, or even specific locations)
- * When complaints were made (over time, by month, or day of the week)
- * How long it takes to resolve complaints
- * Which city agencies handle different issues

You can ask me questions like "Show me the most common complaint types" or "What are the noise complaints in Brooklyn?"

What would you like to know about NYC 311 requests?

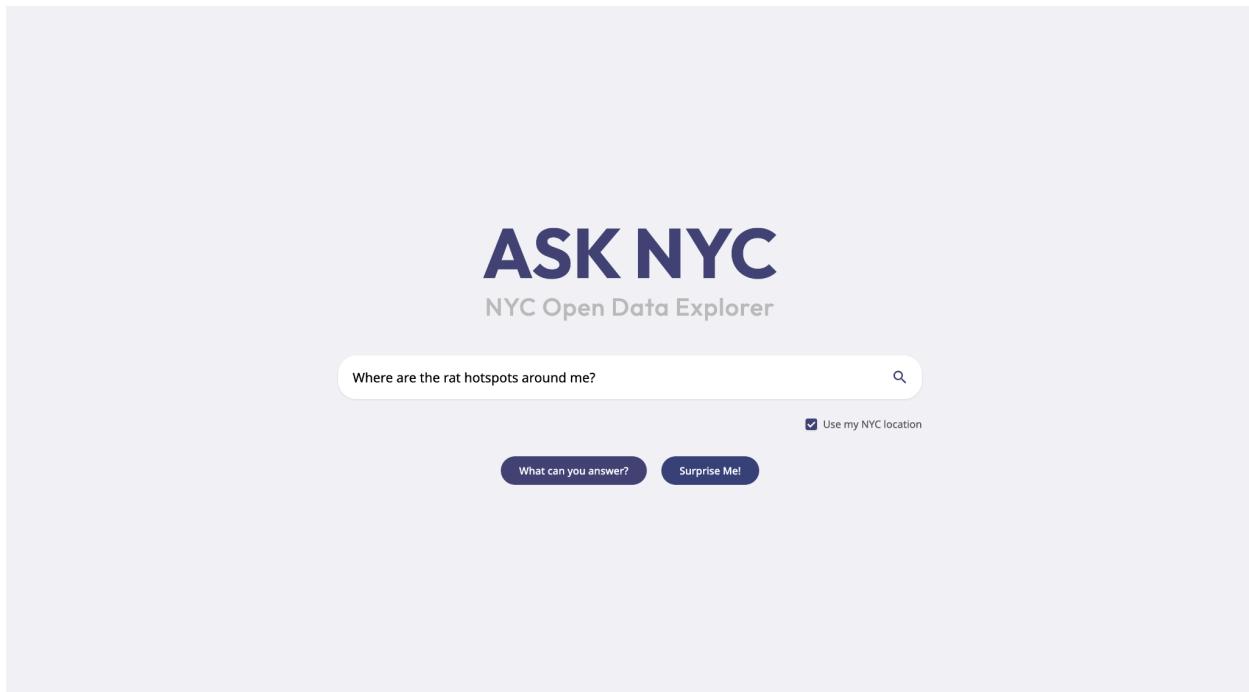
On the right side of the interface, there's a sidebar titled "Chart Types" which has a button labeled "Tr". Below this are several sections with status messages:

- About Data: No time period specified
- Attributes: No attributes in this visualization
- Measures: No measures in this visualization
- Filter: No filters applied

“Surprise Me!” Button

This button randomly inserts sample queries into the search box, showcasing the types of questions the system can answer. It highlights queries that are likely to be engaging to a broad range of users—such as “Where are the rat hotspots around me?” or “Which neighborhoods had the most noise complaints?”—encouraging exploration and discovery, even without prior knowledge of the underlying data structure.

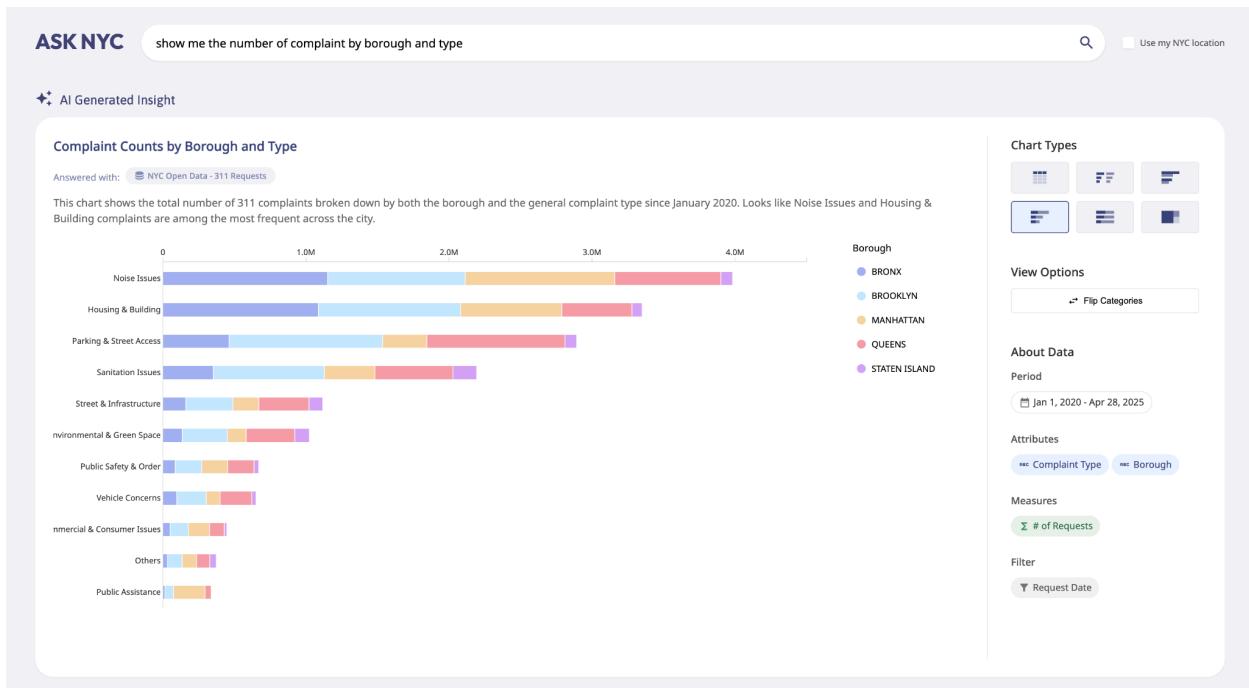
Figure 19. “Surprise Me!” button response



2-3-2. Application Page

The Application Page provides the five main components.

Figure 20. Application Page Design



Search Section (Top)

Positioned at the very top of the page, the Search Section allows users to enter natural language queries. It functions as the main input point for interacting with the system, optionally supporting location-based queries to tailor results geographically.

Title & Insight Section (Below Search Bar)

Directly below the search bar, this section displays a dynamic chart title and a concise AI-generated insight generated by insight generator (3-2-6). It provides an immediate summary of key findings based on the query, helping users understand the context and significance of the results at a glance.

Visualization Section (Center)

Located at the center of the page, the Visualization Section renders the data as a visual chart used with the chart type and aggregated dataset. It allows users to explore trends and comparisons visually and interactively.

Chart Control Section (Right Sidebar, Upper)

Found on the upper part of the right sidebar, the Chart Control Section offers options to switch chart types and toggle category display settings using the available chart types derived from the Chart Selector(3-2-5).

About Data Section (Right Sidebar, Lower)

Located just below the chart controls in the right sidebar, the About Data Section presents metadata including the time range, dimensions, measures, and filters used. This contextual information supports transparency and helps users interpret the visualization accurately.

3. Evaluation

In this section, author-run tests are conducted to assess whether the system can effectively respond to natural language queries.

3-1. Question List

The following questions are submitted to the system. These questions are designed to sound conversational rather than technical.

#	Type	Question
1	Simple overview	Top 10 complaint types citywide this year?
2	Trend analysis	Daily rat trend in Manhattan, 5 yrs?
3	Composition / percentage	Borough % split for 311 last year?
4	Top-N ranking	Top 5 spots for heat complaints this

		winter?
5	Location hotspot	Pothole hot-spots in Greenwich Village?
6	Drill-down (needs the result of Q1)	Break those noise complaints down.
7	Follow-up change of filter (needs the result of Q6)	Cool—now do Queens.
8	Why-type explanatory query	(After seeing the monthly chart of heat complaints): Why's January way worse than July?
9	Vague follow-up asking for a different grain (builds on Q2)	Show that by month?
10	Secondary drill-down (builds on Q4)	Which ZIP leads in BK?

Q1. Simple overview:

"Top 10 complaint types citywide this year?"

Output:

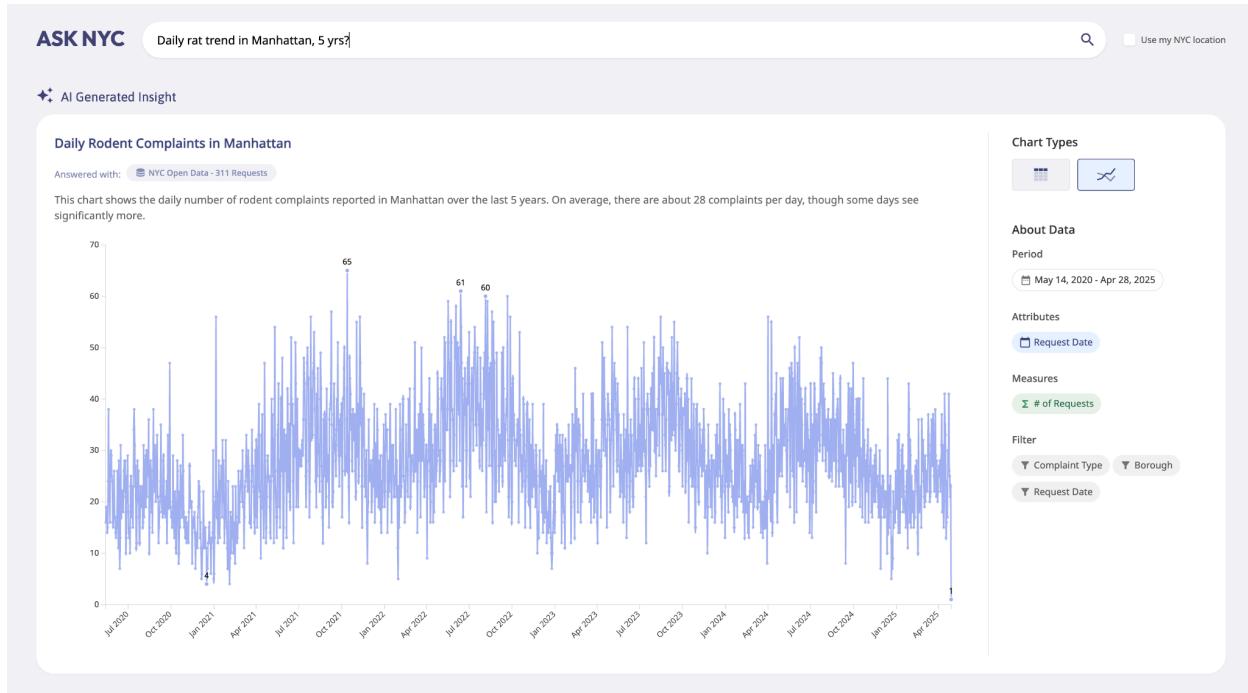


Evaluation: It successfully addressed the most common complaint types within the specified date range (this year). A bar chart appears to be an appropriate visualization for these results.

Q2. Trend analysis

“Daily rat trend in Manhattan, 5 yrs?”

Output:

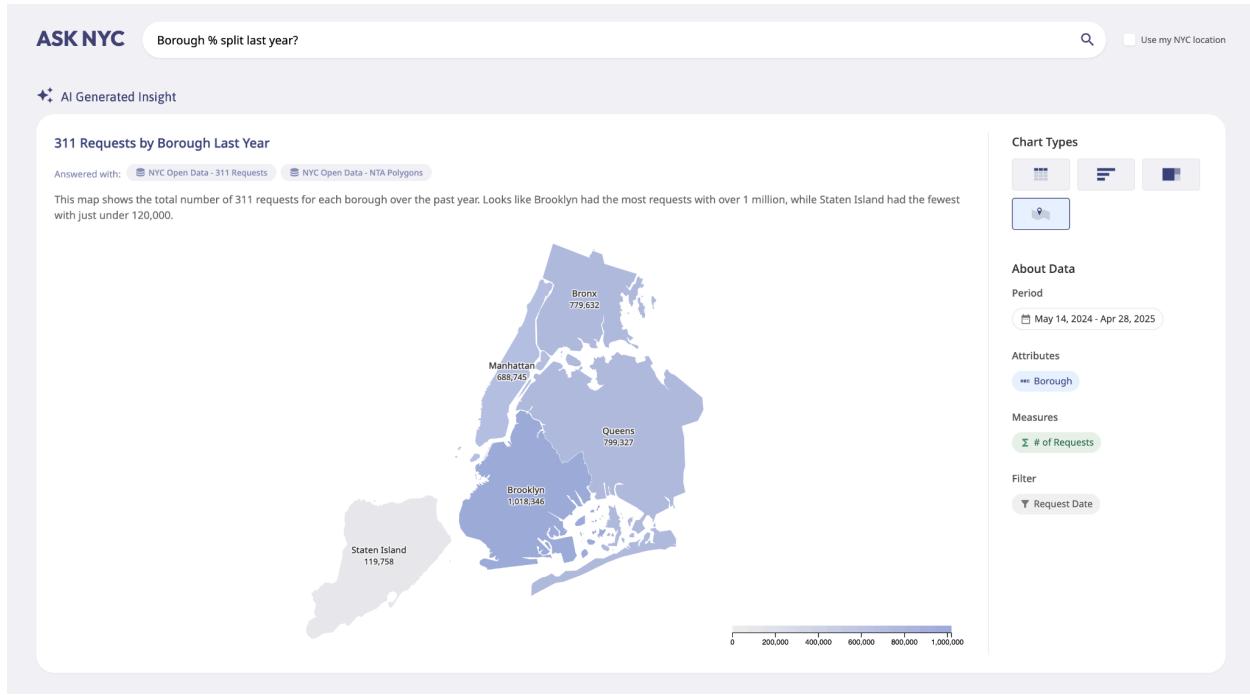


Evaluation: Although the chart itself is difficult to interpret, the system successfully generates the requested line chart. The accompanying text insight also mentions fluctuations, possibly based on standard deviation data.

Q3. Composition / percentage

“Borough % split last year?”

Output:

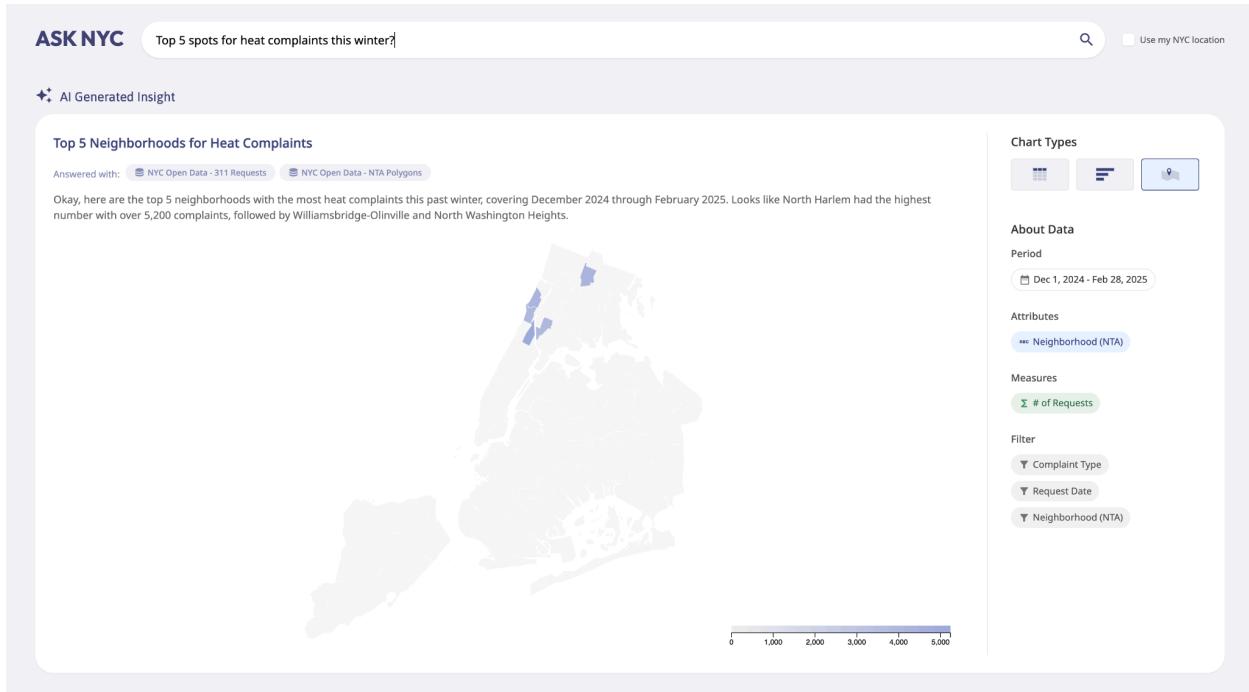


Evaluation: The borough-level choropleth map displaying last year's number of requests was successfully retrieved. However, the map does not directly present percentage data, requiring users to calculate it manually. This limitation suggests that the system is not optimized for this task, indicating potential for improvement.

Q4. Top-N ranking

"Top 5 spots for heat complaints this winter?"

Output:

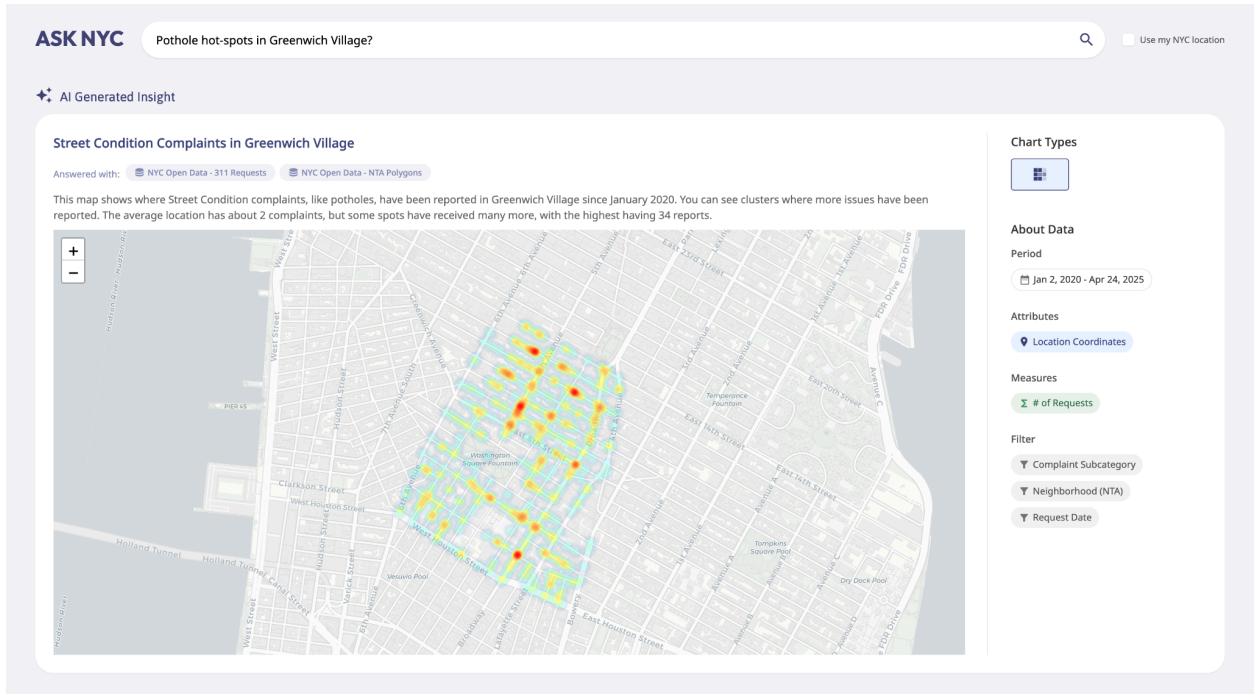


Evaluation: The system successfully identified the top five neighborhoods with the most heat and hot water complaints. Although a choropleth map may not be the most suitable visualization for this query—a bar chart or table might be more appropriate given the limited number of data points—it still provides an acceptable result.

Q5. Location hotspot (map-style)

"Pothole hot-spots in Greenwich Village?"

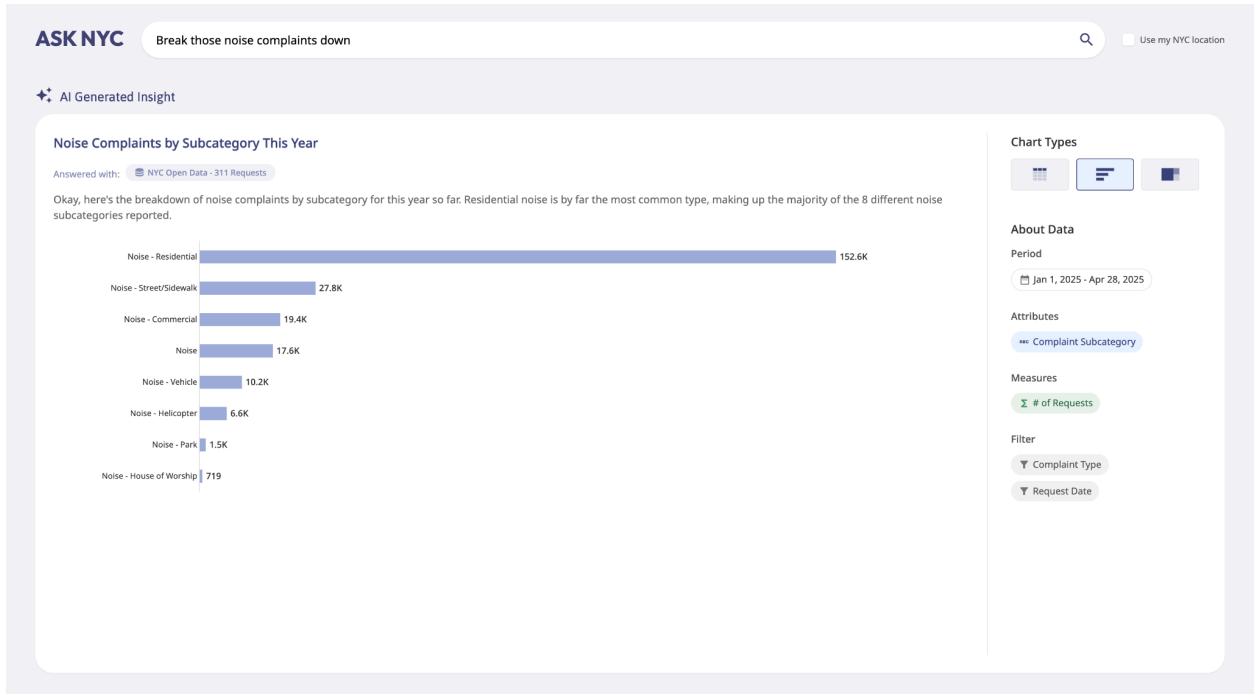
Output:



Evaluation: The system successfully generated a heat map of street condition complaints in Greenwich Village, enabling users to identify complaint hotspots.

Q6. Drill-down (needs the result of Q1) “Break those noise complaints down”

Output:



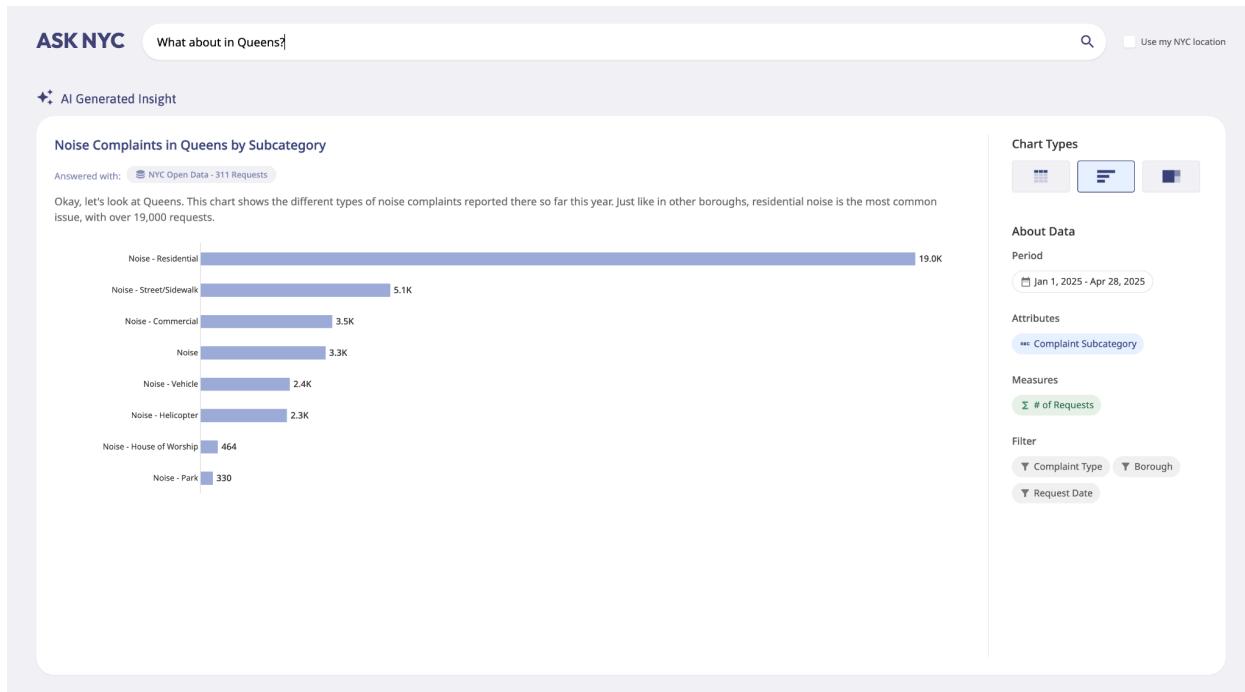
Evaluation:

The system successfully generated a breakdown of noise complaints by granular dimension while preserving the filter conditions for this year and Q1.

Q7. Follow-up change of filter (needs the result of Q6)

"What about in Queens?"

Output:



Evaluation:

The system successfully applied a Queens filter to the previous result.

Q8. Why-type explanatory query

"Why's January way worse for heat complaints than July?"

Output:

This system returned the following answer.

That's a great observation! As the chart shows, there's a really significant difference in heat complaints between January and July.

The data itself shows ***when*** the complaints happen, but it doesn't tell us ***why***. However, based on general knowledge, January is typically the coldest month in NYC, requiring heating systems to work constantly, while July is a warm summer month when heating isn't needed. This difference in temperature is the most likely reason for the pattern you see.

Would you like to explore heat complaints in other ways, perhaps by borough or agency, to see if there are differences there?

The screenshot shows the ASK NYC interface. At the top, there's a search bar with the query "Why's January way worse than July?". To the right of the search bar are a magnifying glass icon and a checkbox labeled "Use my NYC location". Below the search bar, the text "AI Generated Insight" is displayed with a small AI icon.

The main content area contains the following text:

That's a great observation! As the chart shows, there's a really significant difference in heat complaints between January and July. The data itself shows *when* the complaints happen, but it doesn't tell us *why*. However, based on general knowledge, January is typically the coldest month in NYC, requiring heating systems to work constantly, while July is a warm summer month when heating isn't needed. This difference in temperature is the most likely reason for the pattern you see.

Would you like to explore heat complaints in other ways, perhaps by borough or agency, to see if there are differences there?

On the right side of the interface, there's a sidebar titled "Chart Types" which has a "Tr" button highlighted in blue. Below this are sections for "About Data", "Attributes", "Measures", and "Filter", each with a corresponding message indicating no specific information was provided.

Evaluation:

The system addressed two points. First, it acknowledged its inability to determine the exact reasons why January is worse than July. Second, it provided a plausible explanation based on general knowledge from the language model. Overall, the response appears reasonable.

Q9. Vague follow-up asking for a different grain (builds on Q2)

"Show that by month?"

Output:



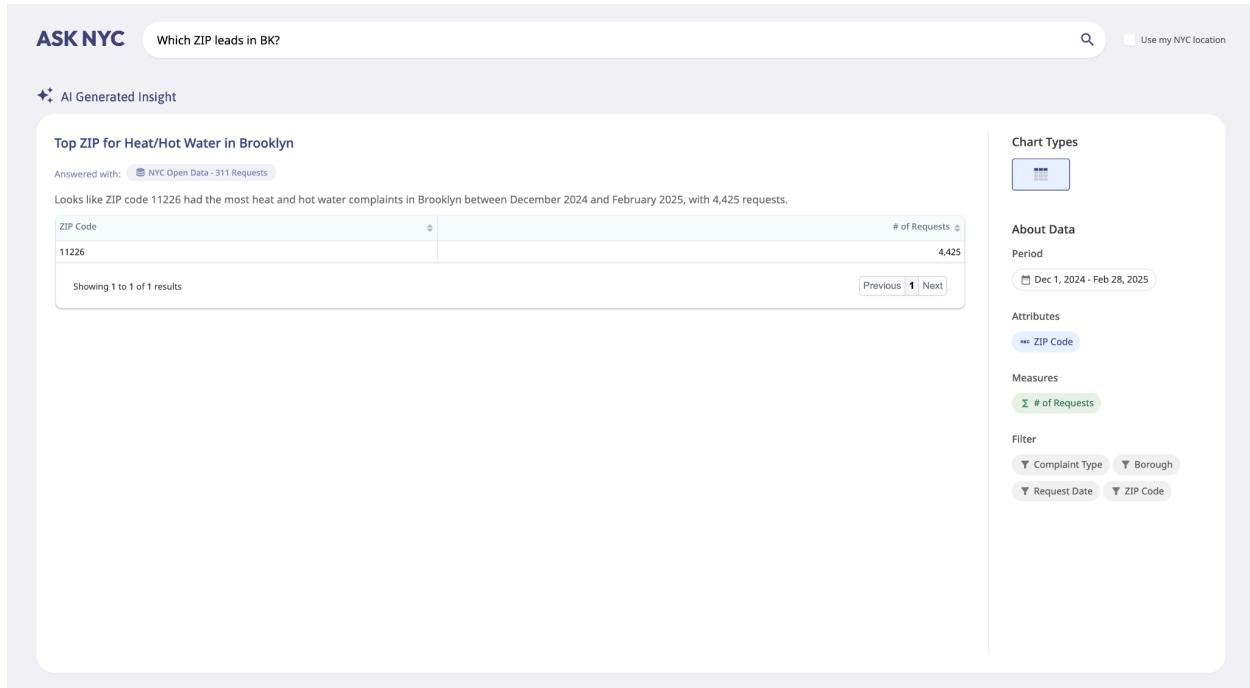
Evaluation:

The system successfully converted the time grain from daily to monthly while preserving the required filters—Manhattan only and rodent complaints.

Q10. Secondary drill-down (builds on Q4)

“Which ZIP leads in BK?”

Output:



Evaluation:

The system successfully identified the top ZIP code in Brooklyn while maintaining the required filter—"Complaint Type"—which was applied for this winter.

3-2. Evaluation Summary

Overall, the evaluation demonstrates that the NLI reliably translates a wide range of natural-language queries into appropriate data aggregations and visualizations, even when contextual understanding is required.

Minor limitations were noted in visualization selection and output formatting—for example, the borough percentage split required manual calculation due to the absence of a built-in percentage chart, and the top-N heat complaints map might have been more effective as a bar chart.

Despite these limitations, the system's performance shows strong potential, highlighting its capability to broaden access to open data for non-technical users, while leaving room for further improvement.

4. Room for Improvement

Although the assessment results appear acceptable for most queries, the broader initiative of democratizing open data for all users still has significant room for improvement.

1. Replace self-evaluation with user testing

The current assessment is author-run and focuses on whether the system returns a result, not whether the result is useful to novices. Recruiting a small panel of city employees, journalists and community advocates, running think-aloud sessions, and coding their frustrations would reveal pain points invisible to the builder and provide quantitative baselines for future iterations.

2. Broaden dataset coverage

Because the prototype is hard-wired to the 311 Service Requests feed, the semantic model, query patterns, and visualization rules are based on the structure of that dataset. However, since open data encompasses a broader range of datasets, supporting a wider variety of data sources is essential to uncover the deeper challenges inherent in democratizing open data for non-technical users.

3. Expand the visualization palette

The rule-based Chart Selector supports 12 chart families, yet it excludes staples such as scatterplots, box-and-whisker or small-multiples grids that are essential for correlation, distribution and relational questions mentioned earlier in the paper. Adding these types (and enabling users to override the automatic pick) would surface subtler insights and avoid the “choropleth-for-everything” fallback observed in the evaluation.

4. Advanced Analytics Capability

Some advanced users may seek more sophisticated analytical capabilities, such as building regression models or running k-means clustering to gain deeper insights. Expanding the system in this direction could encourage power users to more fully engage with the open data portal.

5. Conclusion

This thesis presents a natural-language interface that combines a semantic data model, AI-powered query translation, and rule-based chart selection to enable non-technical users to query New York City’s 311 Open Data feed in plain English.

In a ten-question evaluation covering overview, trend, follow-up, and explanatory queries, the prototype produced correct aggregations and context-appropriate visualizations—demonstrating the significant potential of large language model components to overcome common barriers in data aggregation and visualization.

While some limitations remain—such as the need for a broader visualization library, integration of additional datasets, and user-centered evaluations—these results suggest that natural-language querying can substantially improve the practical accessibility of civic data portals.