**Chicago Business Intelligence Reports**

**For Strategic Planning**

MSDS 432 DL SEC 56

Fall 2021

Siyuan Liu

Nov 21, 2021

**Project Summary**

COVID-19 pandemic outbreak has brought great challenge to both social economy and public health management of U.S. in past two years. In the process of monitoring and assessing the pandemic hit to local communities, big data has played an important role by enabling timely tracking of both economic and public health metrics such as transportation, unemployment, daily cases/ deaths and so on. The city of Chicago has laid out data infrastructure with 16 categories of datasets in place, this project aims to build data lake based on 3 main categories including Transportation, Buildings, and Health & Human Services, to enable capability of generating different intelligence reports for various purposes such as strategic planning, disease control, and infrastructure investment.

**Project Process**

1. **Data Sources and Collection**

The datasets used for the data lake construction of this project are collected from <https://data.cityofchicago.org/>.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset Name | Data Owner | Category | Frequency |
| COVID-19 Daily Cases, Deaths, and Hospitalizations | Department of Public Health | Health & Human Services | Daily |
| COVID-19 Cases, Tests, and Deaths by ZIP Code | Department of Public Health | Health & Human Services | Weekly |
| Chicago COVID-19 Community Vulnerability Index (CCVI) | Department of Public Health | Health & Human Services | / |
| Taxi Trips | Department of Business Affairs & Consumer Protection | Transportation | Monthly |
| Transportation Network Providers - Trips | Chicago Department of Business Affairs & Consumer Protection | Transportation | Monthly |
| Building Permits | Department of Buildings | Buildings | Daily |
| Public Health Statistics – Selected public health indicators by Chicago community area | Epidemiology and Public Heath Informatics, Chicago Department of Public Health (CDPH) | Economy (Unemployment) | / |

**Table 1: Datasets in Scope**

There are multiple requirements with specific output that have been planned in the scope of this project, which involves different attributes from datasets aforementioned. To fulfill each specific requirement, datasets and associated attributes are discussed as below.

**1.1 Requirement 1**

|  |  |
| --- | --- |
| Dataset | Relevant Attributes |
| Taxi Trips | * Trip ID / Taxi ID (used to identify and group taxi drivers who have been to zip codes with risk of being super spreader * Pickup\_centroid\_latitude, Pickup\_centroild\_longtitude, Dropoff\_centroid\_latitude, Dropoff\_centroild\_longtitude (use to map taxi trips to different zip code |
| Transportation Network Providers - Trips | * Trip ID * Pickup\_centroid\_latitude, Pickup\_centroild\_longitude, Dropoff\_centroid\_latitude, Dropoff\_centroild\_longitude (use to map trips to different zip code |
| COVID-19 Daily Cases, Deaths, and Hospitalizations (Daily) | * Case-total * Deaths – Total * Hospitalization - Total |

*Data processing:*

The trip data from Taxi Trips and Transportation Network Providers – Trip are aggregated to form the completed list of trips, each of which is mapped to zip code based on their pickup or drop-off latitude and longitude. At the same time, daily COVID-19 data in terms of cases / deaths / hospitalization is mapped to zip code, enabling distribution location-based COVID alert to the drivers who have been to the relevant local community.

*Issues with datasets:*

* The zip code column within COVID-19 dataset is shown as unknown for location outside of Chicago. We may have to drop off these rows
* The daily COVID-19 dataset is not available by zip code, instead the COVID-19 data with zip code breakdown is updated weekly only. We will either need to figure out a way to map daily COVID-19 data by zip code or have to compromise to distribute weekly data to drivers.
* To map trips to zip code, we will need to bring in the coordinate range of zip code boundaries, potentially using Nominatim geocoder within geopy library in Python

**1.2 Requirement 2**

|  |  |
| --- | --- |
| Data Source | Relevant Attributes |
| Taxi Trips | * Trip ID / Taxi ID (used to identify and group taxi drivers who have been to zip codes with risk of being super spreader * Pickup\_centroid\_latitude, Pickup\_centroild\_longitude * Dropoff\_centroid\_latitude, Dropoff\_centroild\_longitude |
| Transportation Network Providers - Trips | * Trip ID * Pickup\_centroid\_latitude, Pickup\_centroild\_longitude, Dropoff\_centroid\_latitude, Dropoff\_centroild\_longitude (use to map trips to different zip code |
| COVID-19 Cases, Tests, and Deaths by ZIP Code  (Weekly) | * ZIP Code Location (used to map to drop-off locations of taxi trips) * Case – Weekly (metrics for COVID-19 monitoring) |

*Data processing:*

Pickup\_centroid\_latitude and Pickup\_centroild\_longitude are used to narrow down to the trips that started from Ohare (41.9803° N, 87.9090° W) or Midway airport (41.7868° N, 87.7522° W). Then trips can be grouped by zip code based on column Dropoff\_centroid\_latitude and Dropoff\_centroild\_longitude and joint with positive case number and test rate.

Issues with datasets:

* The zip code column within COVID-19 dataset is shown as unknown for location outside of Chicago. We may have to drop off these rows
* The daily COVID-19 dataset is not available by zip code, instead the COVID-19 data with zip code breakdown is updated weekly only. We will either need to figure out a way to map daily COVID-19 data by zip code or have to compromise to distribute weekly data to drivers.
* To map trips to zip code, we will need to bring in the coordinate range of zip code boundaries, potentially using Nominatim geocoder within geopy library in Python

**1.3 Requirement 3**

|  |  |
| --- | --- |
| Data Source | Relevant Attributes |
| Taxi Trips | * Trip ID / Taxi ID * Pickup\_centroid\_latitude, Pickup\_centroild\_longtitude (use to filter down and look up the trips starting from coordinates of two airports (Ohare, Midway) * Dropoff\_centroid\_latitude, Dropoff\_centroild\_longtitude (use to map different destinations of taxt trips starting from the two airports) |
| Chicago COVID-19 Community Vulnerability Index (CCVI) | * Community Area or ZIP Code * CCVI Category (used as a filter to boil down to neighborhood with vulnerability category as “HIGH”) |

*Data processing:*

Zip codes or community areas with “HIGH” as value of CCVI Category are selected. Both location coordinates of pickup trips and drop-off trips are used to map to zip codes that are filtered based on CCVI category.

*Issues with datasets:*

* Community area numbers and zip codes are mixed in one column. We may need to differentiate to avoid duplication
* To map trips to zip code, we will need to bring in the coordinate range of zip code boundaries, potentially using Nominatim geocoder within geopy library in Python

**1.4 Requirement 3**

|  |  |
| --- | --- |
| Data Source | Relevant Attributes |
| Taxi Trips | * Trip ID / Taxi ID (used to aggregate to measure traffic by trips per neighborhood/zip code * Pickup\_centroid\_latitude, Pickup\_centroild\_longtitude, Dropoff\_centroid\_latitude, Dropoff\_centroild\_longtitude (use to map taxi trips to different zip code * Trip Start Timestamp, Trip End Timestamp (used to aggregate the data period by daily, weekly and monthly |

*Data processing:*

Trips are aggregated by time period (daily, weekly and monthly) and zip code (based on location coordinate data). Then time series algorithm such as LSTM could be applied to forecast the traffic pattern by zip code.

**1.5 Requirement 5**

|  |  |
| --- | --- |
| Data Source | Relevant Attributes |
| Building Permits | * ID / PERMIT # * LATITUDE, LONGITUDE (used to map associated buildings that locate in the top 5 neighborhood with highest unemployment rate and poverty rate) |
| Public Health Statistics- Selected public health indicators by Chicago community area | * Unemployment (used to identified top 5 neighborhoods with highest unemployment rate and poverty rate) * Below Poverty Level * community\_area\_name |

*Data processing:*

Top five neighborhoods with highest unemployment rate and poverty rate are selected based on column “Unemployment” and “Below Poverty Level”. Building permits data are mapped and grouped to zip code or community areas based on latitude and longitude. Permits ID that are within those five neighborhoods are screened and included in the output for fee waive consideration.

**1.6 Requirement 6**

|  |  |
| --- | --- |
| Data Source | Relevant Attributes |
| Building Permits | * PERMIT\_TYPE (used as filter to boil down to PERMIT – NEW CONSTRUTION permit applications) * LATITUDE, LONGITUDE |
| Public Health Statistics- Selected public health indicators by Chicago community area | * Unemployment * Per Capita Income (used as filter to locate neighborhood with PER CAPITA INCOME less than $30,000) * community\_area\_name (used to map to neighborhood / zip code) |

*Data processing:*

Only permits with type of new construction are selected and aggregated at zip code level. Zip codes are narrowed down to ones with PER CAPITA INCOME less than $30,000, and then sorted by counted number of permits. The zip code with lowest new construction applications is selected for loan offering.

**2. Infrastructure and Technology Stack**

**2.1 Database Engine**

**PostgreSQL** is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL 14 brings a variety of features that help developers and administrators deploy their data-backed applications. PostgreSQL continues to add innovations on complex data types, including more convenient access for JSON and support for noncontiguous ranges of data.

**2.2 Micro Service Development**

**Python**is used as programming language in this project to construct the microservices which will be tasked to extract data from Chicago Data Portal, interact with Postgres Engine and insert data into database. A few python packages are utilized as key components to execute functionalities including data extraction, ingestion, and transformation. **Golang** has also been reviewed as alternative language which provides multiple advantages over Python.

**2.3 Micro Service Deployment**

**Docker** is an excellent tool for managing and deploying microservices. Each microservice can be further broken down into processes running in separate Docker containers, which can be specified with Dockerfiles and Docker Compose configuration files.

[**Kubernetes**](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/)**,** also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications

**2.4 Time Series Forecast**

To develop time-series forecast capability of projecting taxi traffic patterns, TensorFlow LSTM algorithm from Google and Prophet from Facebook are under consideration.

**TensorFlow/Keras LSTM,** Long Short-Term Memory layer, is one of Recurrent neural networks (RNN) that is powerful for modelling sequence data such as time series. Schematically, a RNN layer uses a for loop to iterate over the timesteps of a sequence, while maintaining an internal state that encodes information about the timesteps it has seen so far.

**Prophet** is an open source library released by Facebook. It delivers good performance, especially when working with time series that have strong seasonal effects fitting with non-linear trends.

**3. Architecture and Design**

******

**Figure 1** Overview of Architecture & Design of CBI Project

**3.1 Data Collection and Collection**

Data Ingestion and collection is designed as and fulfilled by the microservice built with Python API, containerized by Docker, and deployed via Kubernetes resources. The python app is built with data extraction and insertion functionalities that interact with data lake on Postgres Engine hosted on local machine. To implement the functionalities, several Python libraries are utilized and built into the app:

* Pandas: a widely used for data processing and manipulation.
* Sodapy: python client for the Socrata Open Data API to communicate with Chicago Data Portal (<https://github.com/xmunoz/sodapy>)
* Psycopy2: one of popular PostgreSQL database adapter for python that is used to interact with data lake constructed on Postgres engine
* Schedule: package for job scheduling (<https://schedule.readthedocs.io/en/stable/>)

Then, the app is put into a container via Docker and deployment via Kubernetes. To avoid technical issues with connection, a virtual environment has been initiated via minikube. The image built into container will be pushed to Docker Hub and deployed by Kubernetes. The microservice is scheduled on daily basis with automatic execution

**3.2 Data Storage**

The data storage layer is built upon a data lake on PostgreSQL engine. The fragmented data collected from Chicago Data Portal will be stored into “chiproject\_postgres” database hosted on local machine (Port:5432). Within the data base, six tables are created as show below to store data that is extracted with selected fields from respective API regarding each data categories described in phase 1.

**3.3 Data Processing**

In this layer, the fragmented data that is stored in PostgreSQL database is queried with SQL statements and loaded into Jupyter notebook using Psycopy2 depending on the specific requirements. The data is then manipulated and processed through several foundational steps such as removing null value, mapping out zip code with community area number, mapping location coordinates (latitude, longitude) to zip code, etc. Cleaned data with lean relevant fields are expected to be generated ready for visualization and forecast in next Insight layer.

**3.4 Insights and visualization**

In this layer, analytical algorithms such as TensorFlow LSTM algorithm from Google and Prophet from Facebook are brought in and used for time series forecast on traffic pattern in daily, weekly and monthly window. Also, some tabular-format output is generated to meet the requirements by City of Chicago, for example, the top 5 neighborhoods with highest permits but highest unemployment rate and poverty rate are identified and delivered as one of projects outputs.

**4. Development**

**4.1 Data lake construction**

The data lake that hosts the raw data extracted from Chicago Data Portal was constructed with Postgres Database engine. The database and tables were defined through pgAdmin4 and hosted on local machine (port 5432) as shown in Figure 2.

Graphical user interface, application, table, Excel

Description automatically generated

**Figure 2** Snapshot of Data Lake - Postgres

**4.2 Data fetching API**

The API that is tasked to fetch data from Chicago Data Portal in a batch manner was developed with Python environment and containerized and deployed using Docker/Kubernetes. All the coding was done in Visual Studio Code, which integrates well with different programming language. As shown in Figure.3, the data fetching functionalities was programed in main.py, then it was containerized into Docker image (Figure.4) and further defined and run via yaml file (Figure.5).

Text

Description automatically generated

**Figure 3** Screenshot of Python app

Text

Description automatically generated

**Figure 4** Screenshot of Docker file

Text

Description automatically generated

**Figure 5** Screenshot of Yaml file

***Text

Description automatically generated***

**Figure 6** Build Docker Container Image

***Text

Description automatically generated***

Text

Description automatically generated

**Figure 7** Kubernetes Deployment

**4.3 Data processing and analysis**

Data processing and analysis were conducted in Python environment. The functionalities that were developed and implemented in this section are purposed to meet the described requirements and enable City of Chicago to make strategic planning with analytical and forecasting capabilities. The implementation process of each requirement is documented as below:

*Requirement 1*

* Taxi trip and Covid-19 data was fetched through PostgreSQL queries and read into DataFrame format
* Data was then cleaned up by removing null values and transformed to the right data type, for example, latitude and longitude were retyped to numeric values
* Uszipcode library was imported to map taxi trips to zip codes by latitude and longitude to generate both pick-up and drop-off zip code for each trip as shown in Figure. 6
* For Covid-19 data with cases and deaths, zip code information was extracted from column “zip\_code\_location” (str class) using Uszipcode package as well
* Covid-19 data were grouped by zip code with aggregated cases and deaths numbers, the information could be then embedded into messages and sent to taxi drivers who drop off or pick up passengers to/from specific zip code as shown in Figure. 7

Table

Description automatically generated with medium confidence

**Figure 6** Zip Code Mapping – Requirement 1

Text

Description automatically generated with medium confidence

**Figure 7** Output Test – Requirement 1

*Requirement 2*

* The coordinates (latitude, longitude) of Chicago Ohare airport and Midway airport were looked up and set as global variables
* Taxi trips that have approximate coordinates with that of Ohare and Midway airport were filtered (rel\_tol = 0.01) and loaded into separate DataFrames
* For each DataFrame, the taxi trips were grouped by zip code and date and aggregated into number of trips. The output shows the traffic pattern – number of trip per day, from one of two airports to each zip code as shown Figure. 8

Graphical user interface, text, application

Description automatically generatedTable

Description automatically generated

**Figure 8** Trips From Ohare/Midway Airport – Requirement 2

*Requirement 3*

* CCVI data records with only column ‘ccvi\_category” as ‘HIGH” were fetched from data lake through PostgreSQL query
* The taxi trip data generated from previous step and CCVI data were both mapped into community areas based on community area boundaries defined by latitude and longitude Reference community geocoding data was imported from file “Neighborhoods\_2012\_polygons.json”
* The taxi trip data was then grouped by community area with aggregated traffic (number of trips) and mapped into each neighborhood with ccvi\_category as ‘HIGH’ as shown in Figure. 9

Table

Description automatically generated

**Figure 9** Output of Requirement 3

*Requirement 4*

* Three additional features were firstly generated based on taxi trip data. For each trip, latitude and longitude of each trip was mapped to specific zip code. The date was extracted out of “start time stamp” attribute and convert to week number and month number.
* The transformed taxi trip data was then grouped by date and zip code with aggregated number of trips. Due to computing intensity of mapping trip to zip code on more than 3 million rows, I ended up looking at only pick-ups
* The daily traffic data was then fed into pre-defined Prophet model to generate time series forecast on daily, weekly and monthly traffic

*Requirement 5*

* Top 5 community with highest unemployment rate were fetched from “economy” table, the 5 values of community area were convert into a list named “inv\_list”
* The building permit data were fetched from “bldgpmt” table from the data lake
* The permit records with community area number that are within the “inv\_list” were filtered and selected as output with permit id, so city of Chicago can waive the fees for these applications to support the economy in these neighborhoods as shown Figure. 10

Graphical user interface, text, application

Description automatically generatedTable

Description automatically generated

**Figure 10** Output of Requirement 5

*Requirement 6*

* The building permit instances with “PERMIT\_TYPE” as “PERMIT – NEW CONSTRUCTION” were fetched from the data lake using SQL query and loaded into a DataFrame
* The neighborhoods with “PER\_CAPITA\_INCOME” less than $30,000 were selected from “economy” table
* The building permit data was filtered based on neighborhood which is on the list generated from previous step, grouped by neighborhood with aggregated number of building permit application, and then sorted in ascending manner
* Top 5 neighborhoods with lowest number of building permit applications were filtered and selected as output

*Table

Description automatically generated*

**Figure 11** Output of Requirement 6

**5. Forecasting and Strategic Planning**

As described in section 4 – requirement 4, the forecast of traffic pattern has been generated using Prophet model at daily, weekly, and monthly basis. Due to computing intensity driven by large data size, 2021 taxi trip date was extracted from date lake and used for demo. Also, stratified sampling method has been used to proportionally sample the taxi trip data by date.

The figures below demonstrate the output of times series forecast by Prophet model.

Chart, scatter chart

Description automatically generated

**Figure 12** Daily Traffic Forecast – Zip Code 60657

Chart, scatter chart

Description automatically generated

**Figure 13** Weekly Traffic Forecast – Zip Code 60657

Chart, line chart, scatter chart

Description automatically generated

**Figure 14** Monthly Traffic Forecast – Zip Code 60657

Chart, line chart

Description automatically generated

**Figure 15** Daily Traffic Forecast – All Zip Codes

Chart, line chart

Description automatically generated

**Figure 16** Weekly Traffic Forecast – All Zip Codes

Chart, line chart

Description automatically generated

**Figure 17** Monthly Traffic Forecast – All Zip Codes