Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Семинарска работа по предметот:
Рударење на масивни податоци

Визуелизација на настани на мапа во реално време со користење на Apache Flink, Apache Kafka и Web Sockets

Скопје, Септември 2022

Ментор:
Проф. Д-р Ѓорѓи Мацаров

Изработена од:
Милена Трајаноска 181004

Содржина

Вовед

Целта на оваа семинарска работа е да се изработи едноставна Веб апликација за прикажување на настани на мапа во реално време. За таа цел се користат технологии за процесирање на потоци од податоци (stream processing). Конкретно, за целите на оваа семинарска работа се користи Apache Flink за процесирање на податоците со помош на лизгачки прозорци.

Резултатите од процесирањето се запишуваат на посебен топик на Apache Kafka редици на чекање. Овие редици се конфигурирани да ги чуваат податоците се додека не бидат прочитани (конзумирани) од некоја апликација.

Резултатите се читаат од Веб апликација и со помош на Веб сокети се испраќаат пораките до интерфејсот каде се прилажуваат како настани на мапа. Податоците се освежуваат во интервали од 1 минута, а најчесто претставуваат агрегации на настани од минатите 10 минути.

Податочно множество

Архитектурата на апликацијата е изработена со цел прикажување на настани на мапа во реално време со помош на стриминг податоци. За целите на изработка на овој прокет се користат податоци од NYC Trip Record Data (“TLC Trip Record Data - TLC”). Од оваа база, беа земени податоци од множеството 2016 NYC Yellow Cab trip record data.

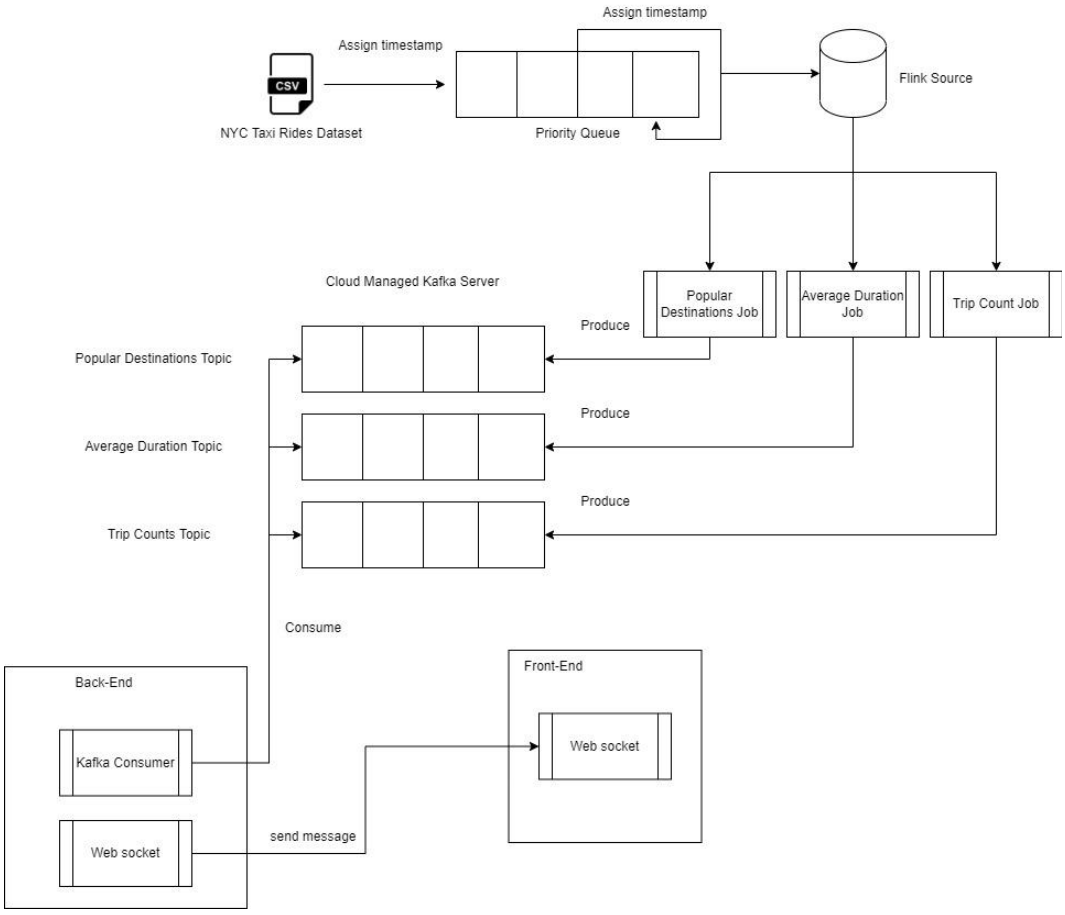
Податоците се достапни во parquet и csv формат и ги содржат следните колони:

- **id** - уникатен идентификатор на секое патување
- **vendor_id** - код кој што ја идентификува такси компанијата одговорна за патувањето
- **pickup_datetime** - датум и време на почеток на патувањето
- **dropoff_datetime** - датум и време на завршување на патувањето
- **passenger_count** - број на патници во возилото за време на патувањето
- **pickup_longitude** - латитуда од каде бил преземен патникот
- **pickup_latitude** - лонгитуда од каде бил преземен патиникот
- **dropoff_longitude** - латитуда на целната дестинација
- **dropoff_latitude** - лонгитуда на целната дестинација
- **store_and_fwd_flag** - индикатор кој покажува дали податоците биле чувани во меморија на возилото пред да се испратат до такси компанијата поради пад на конекција
- **trip_duration** - времетраење на патувањето во секунди

Податочното множество се состои од вкупно 2,083,778 патувања, во рамките на Њујорк и неговата непосредна близина. Истите овие податоци се процесираат на соодветен начин, кој е објаснет подоцна, за да се симулира добивање на податоци во реално време.

Архитектура на апликацијата

Архитектурата на апликацијата се состои од повеќе компоненти. Овие компоненти се детално објаснети во следните подглави. Целосната архитектура е прикажана на Слика 1.



Слика 1. Архитектура за процесирање на настани во реално време

Податочен извор

Податочниот извор претставува csv датотека со податоци за патувања. Се работи за податочното множество опишано во претходната глава. Целта на апликацијата е да се прикажат податоци за патувањата во реално време. Од оваа причина, поради недостаток на вистински извор кој ќе генерира податоци од такси возилата,

и кои можат да се вчитаат во апликацијата, направена е симулација на таков процес. Истите 2,083,778 патувања постојано се процесираат во непрекинат циклус, со додавање на случајно доцнење на времето за процесирање.

Стратегија за доделување на временски печат

Секоја редица, еквивалентна на едно патување, се сместува во приоритетна редица врз база на доделениот временски печат.

Начинот на доделување на временскиот печат е следниот, се зема моменталното системско време кога е прочитана редицата од датотеката, и на истото се додава случајно доцнење во интервал од 0 до 50 секунди. Вака креираните настани се додаваат во приоритетната редица, кадешто настани со пониски вредности на временскиот печат имаат поголем приоритет да бидат, и ќе бидат први прочитани од редицата.

Откако ќе се прочита одреден настан од редицата, истиот се процесира, потоа му се доделува нов временски печат кој се изведува од моменталното процесорско време и додаденото доцнење во рамки од 0 до 50 секунди. Овој настан повторно се додава во приоритетната редица, со новопресметаниот приоритет. Процесот се повторува за секој настан во непрекинат циклус.

Flink процесори на податочни потоци

Следната компонента во архитектурата на апликацијата се процесорите на податоци во реално време. За нивната имплементација се користи Apache Flink рамката (“Apache Flink”). Постојат три типа на настани кои се следат во рамките на апликацијата а тоа се: агрегирање на бројот на патници кои пристигнуваат на одредена локација во временска рамка од 10 минути, вториот процес ги агрегира просечните времетрења на патувањата во рамките на минатите 10 минути, и последниот процесор ги пресметува вкупниот број на патувања направени во изминатите 10 минути.

Процесирање на популарни дестинации

Првиот процесор го пресметува вкупниот број на патници кои пристигнуваат на одредена локација во временски период од 10 минути. Овие податоци се ажурираат за секоја следна минута со користење на лизгачки прозорец.

За пресметување на дестинацијата, се користата податоците за латитуда и лонгитуда на дестинацијата на патувањето. Меѓутоа овие податоци не се користат во формтот во кој се добиваат, бидејќи речиси никогаш нема да има повеќе од 1 патување до истата локација по латитуда и лонгитуда во толку краток временски период. Наместо оваа стратегија, за агрегирање на податоците дестинациите се групираат по блокови. Овие блокови имаат висина од 0.0014 степени и ширина од 0.00125 степени. Со тоа се постигнува целата површина на Њујорк да се подели на 100 блокови со речиси еднаква плоштина.

За секоја дестинација се пресметува во кој блок припаѓа и се агрегира според латитудата и лонгитудата на центарот на блокот. Ова претставува поток на податоци агрегиран по клуч, каде што клучот е централната точка на блокот на кој припаѓа дестинацијата. Во агрегацијата се врши пресметка на вкупниот број на патници пристигнати на таа локација во временскиот интервал од 10 минути.

Резултатите од процесирањето се запишуваат на соодветниот sink кој ќе биде подетално објаснет во наредните глави. Како брокер во архитектурата, се користи Apache Kafka (“Apache Kafka”). Овие sink-ови се всушност Kafka producer-и кои запишиваат податоци во форма на настани на Kafka топици, кои се дефинирани за секој процесор посебно.

Процесирање на просечно времетраење на патувања

Вториот процесор пресметува просечно времетраење на сите патувања во временски интервал од 10 минути. Овие пресметки се ажурираат на секоја минута. За разлика од претходниот процесор, во овој случај не се агрегираат податоците по клуч, бидејќи секоја редица од податочното множество претставува едно единствено патување, различно од сите останати.

Од податоците се користи колоната за времетраење на патувањето во секунди, и истото се претвора во минути и се пресметува просек од сите патувања во минатите 10 минути.

Процесираните податоци се запишуваат на Kafka топикот направен за запишување на настаните од пресметка на времетраењето на патувањата.

Процесирање на вкупен број на патувања

Последниот процесор пресметува вкупен број на паувања направени во последните 10 минтуи. Како и претходниот процесор, така и овој процесор не прави агрегација по клуч, туку ги агрегира податоците во рамките на целиот поток.

Пресметаните податоци за бројот на патувања се запишуваат на соодветниот Kafka топик.

Kafka producer / sink

Следната компонента во архитектурата претставува Kafka producer. Оваа компонента е одговорна за запишување на резултатите од процесирањето на настаните во соодветниот Kafka топик.

Изработени се 3 sink-ови кои одговараат на настаните кои се процесираат, имено: настан за популарна дестинација, настан за просечно времетраење на патување и настан за вкупен број на патувања.

За подобра организација на кодот се користи дизајн шаблонот “Factory Method“ (“Factory Method Design Pattern”) за различните типови на Kafka sink-ови.

Структурата на податоците кои се запишуваат на топикот за популарни дестинации е следна:

```
PopularDestination {
      locationCenter: Point{
            latitude: double,
            longitude: double
      },
      passengers: long
}
```

Структурата на податоците кои се запишуваат на топикот за просечно времетраење на патување е следна:

```
TripDuration {
      sumDurations: long,
      numberOfTrips: long,
      kmPassed: double,
      date: LocalDateTime
}
```

Структурата на податоците кои се запишуваат на топикот за вкупен број на патувања е следна:

```
TripHourMintueCount {
      hour: int,
      minute: int,
      count: long,
}
```

Сите настани се серијализираат така што најпрво се трансформираат во JSON формат, а потоа се претвораат во бајти и се запишуваат на соодветниот топик.

Kafka сервер во облак

За управување со различните Kafka редици на чекање, беше искористена бесплатната верзија на Kafka сервер во облак CloudKafka од планот DeveloperDuck (“Documentation”).

За потребите на апликацијата беа креирани 3 топика, имено: popular-destinations за запишување на настаните кои ги бројат патниците кои пристигнуваат на одредена дестинација, потоа trip-duration каде што се запишуваат настаните за пресметување на просечното времетраење на патување, и trip-count каде што се запишуваат настаните кои го означуваат кумулативниот број на патувања во интервали од по 10 минути.

Конфигурацијата за сите топици е еднаква. Секој топик има по 5 партиции, репликацискиот фактор по редица на чекање е 1 (односно не се реплицираат настаните за да се овозможи редундантност и толеранција на грешки), вкупниот број на бајти кои ги собира во една редица е 1048576, и времето на задржување на одреден настан пред да биде избришан е 86400000 ms.

На различните топици запишуваат Kafka sink-овите кои беа објаснети во претходната глава. Од топиците чита Kafka consumer кој е дел од Веб апликацијата за прикажување на настаните на мапа. Најпрво, consumer-от треба да се претплати на секој од топиците за да ги добива настаните како што ќе бидат запишувани на топикот. Откако ќе бидат прочитани, пораките се испраќаат на Веб интерфејсот на апликацијата за да бидат визуелно прикажани.

Kafka Consumer

Kafka consumer-от е посебна нишка која е дел од Back-End-от на Веб апликацијата. Consumer-от се претплатува на секој од претходно објаснетите Kafka топици, и постојано слуша дали ќе се запишат нови настани.

На consumer-от е прикачена callback функција, која за секој нов настан кој ќе биде прочитан од топикот, го препраќа со помош на Веб сокет кон Front-End-от на апликацијата. Освен самиот настан, се праќа и топикот кој служи за да се идентификува која функција да се повика за приказ на настанот, на пример за popular-destinations топикот се повикува функцијата која го исцртува настанот на мапа.

Consumer-от секој пат започнува да чита од почетокот на топикот, односно ги чита пораките кои биле запишани а не биле прочитани ниту еднаш до конкретниот момент. Сите останати пораки кои биле прочитани барем еднаш се бришат од топикот.

Веб апликација

Веб апликацијата претставува следната компонента во архитектурата. Изработена е со користење на рамката Express (“Express Node.js”) во Node.js (“Node.js”) за Back-End и JavaScript (“JavaScript”) со HTML (“HTML”) и CSS (“CSS”) на Front-End. Се работи за апликација која се состои од само една страница, на која се прикажуваат сите настани.

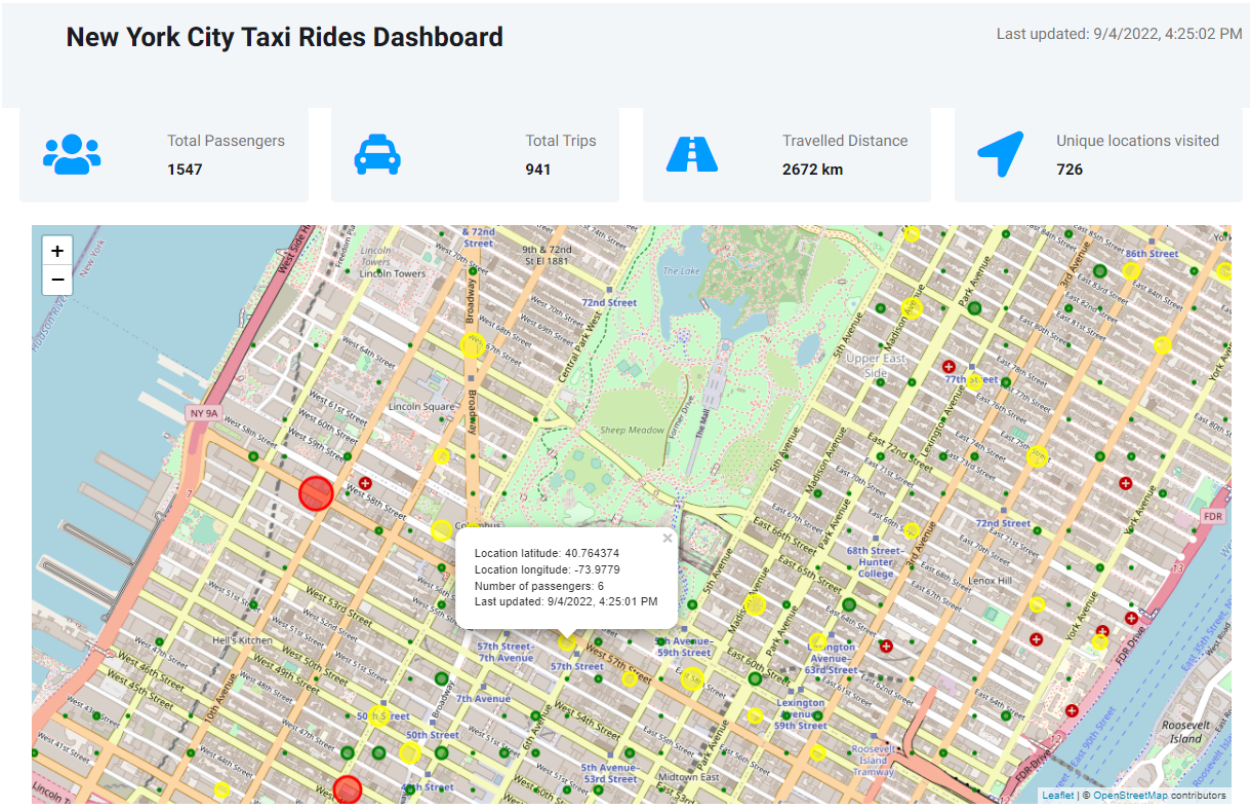
Front-End-от и Back-End-от на Веб апликацијата комуницираат преку Веб сокет (Kitamura). Во конкретната апликација, се отвора Веб сокет на страна на Веб серверот, на порта 7071. При отворање на нова конекција со сокетот, се стартува Kafka consumer-от доколку веќе не е стартуван, и до Front-End-от на апликацијата се испраќаат сите пораки кои ги прима consumer-от преку оваа конекција.

Се чува листа од клиентски сокети кои оствари конекција со серверскиот сокет, и на секој од клиентите му се препраќаат пораките од consumer-от. Кога некој од клиентските сокети ќе ја затвори конекцијата, истата се затвора и на серверска страна и клиентскиот сокет се отстранува од листата со конекции.

Визуелизација на настаните

Front-End-от на Веб апликацијата е напишан во JavaScript и ги визуелизира процесираните настани на мапа со користење на библиотеката leaflet.js (“Leaflet.js”). Дополнително, времетраењето на патувањата и вкупниот број на патувања се прикажуваат на график со помош на библиотеката Chart.js (“Chart.js”). Веб интерфејсот се состои од една страница на кој се прикажани сите настани.

На Слика 2. прикажан е дел од Веб интерфејсот кој е одговорен за исцртување на настаните за популарни дестинации на мапа. Круговите ги претставуваат дестинациите на кои во конкретниот временски интервал од 10 минути пристигнал барем еден патник. Големината и бојата на круговите ја претставува визуелно густината на сообраќајот во овие области. Поголеми кругови значат повеќе патници пристигнале на конкретната дестинација, дополнително, доколку кругот е црвен значи дека на таа дестинација пристигнуваат многу патници, доколку е жолт значи дека има релативно голем број на патници кои пристигнуваат таму, а кај зелените кругови пристигнуваат помалку патници.

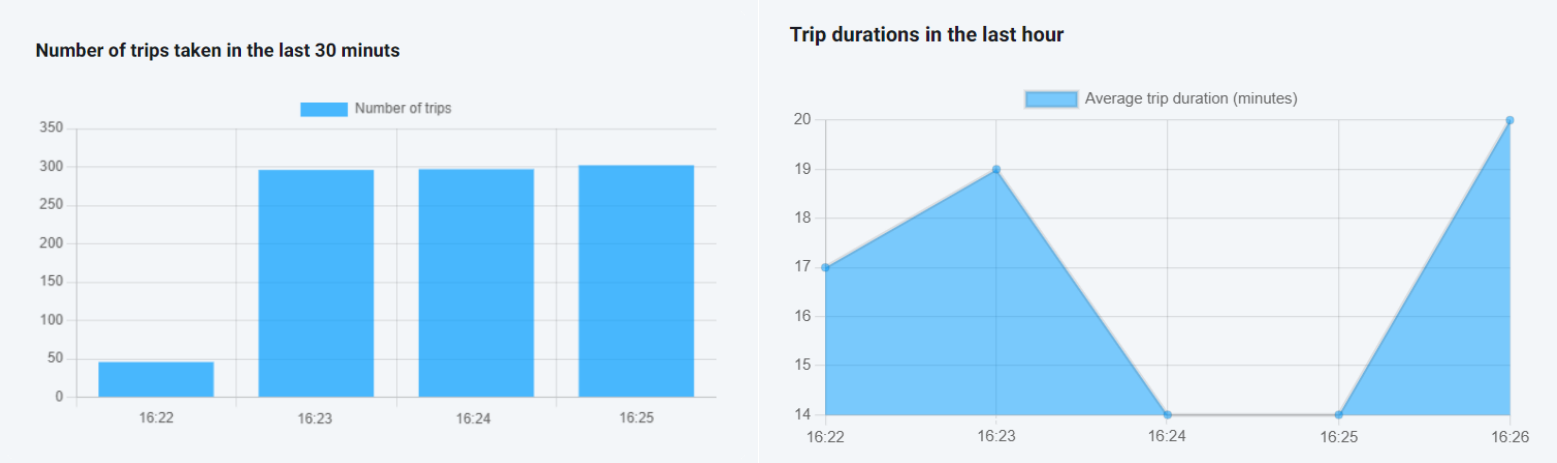


Слика 2. Исцртување на популарни дестинации на мапа

За секоја дестинација може да се видат информации за латитудата и лонгитудата, бројот на патници во минатите 10 минути кои пристигнале таму, како и времето на последно ажурирање на податоците за таа дестинација.

Освен исцртаните настани на мапа, дополнително се прикажани и одредени статистики кои го вклучуваат вкупниот број на патници, вкупниот број на патувања, вкупно изминатата километража на сите патувања и бројот на уникатни посетени дестинации.

На Слика 3. претставени се графици на кои се прикажани вкупниот број на патувања кои завршиле во дадената минута, и просечното времетраење на истите патувања, по минута.



Слика 3. Исцртување на графици за вкупниот број на патувања и просечното времетраење на патувањата.

Заклучок

Во рамките на оваа семинарска работа беше изработена едноставна Веб апликација за прикажување на настани од потоци од податоци во реално време на мапа.

За таа цел беа користени повеќе технологии меѓу кои Apache Flink, Apache Kafka, Node.js express, Web sockets, JavaScript, HTML, CSS.

Беа направени 3 Kafka producer-и кои запишуваат процесирани настани на посебни Kafka топици. Во рамките на Веб апликацијата беше креиран Kafka consumer кој ги чита запишаните податоци за процесираните настани, и соодветно

преку Веб сокет овие податоци беа препраќани на Front-End-от на Веб апликацијата каде се прикажуваат визуелно во реално време.

Беше имплементирана Proof of Concept апликација за прикажување на настани кои произелгуваат од такси возења во рамките на Њујорк. Преку оваа апликација се покажува како секоја од посебните компоненти на оваа stream-processing архитектура функционираат засебно и како е воспоставена комуникацијата меѓу секоја од компонентите за да се постигне процесирањето во реално време.

Референци

“Apache Flink.” *Apache Flink: Stateful Computations over Data Streams*, 2014, https://flink.apache.org/. Accessed 4 September 2022.

“Apache Kafka.” *Apache Kafka*, 2011, https://kafka.apache.org/. Accessed 4 September 2022.

“Chart.js.” *Chart.js | Open source HTML5 Charts for your website*, 2014, https://www.chartjs.org/. Accessed 4 September 2022.

“CSS.” *Cascading Style Sheets home page*, 1996, https://www.w3.org/Style/CSS/Overview.en.html. Accessed 4 September 2022.

“Documentation.” *CloudKarafka*, 2015-2022, https://www.cloudkarafka.com/docs/index.html. Accessed 4 September 2022.

“Express Node.js.” *Express - Node.js web application framework*, 2017, https://expressjs.com/. Accessed 4 September 2022.

“Factory Method Design Pattern.” *Javatpoint*, 2011-2022, https://www.javatpoint.com/factory-method-design-pattern. Accessed 3 September 2022.

“HTML.” *HTML Standard*, 1993, https://html.spec.whatwg.org/. Accessed 4 September 2022.

"JavaScript." *JavaScript.com*, 1995, https://www.javascript.com/. Accessed 4 September 2022.

Kitamura, Eiji. "Introducing WebSockets - Bringing Sockets to the Web." *web.dev*, 20 October 2012, https://web.dev/websockets-basics/. Accessed 4 September 2022.

"Leaflet.js." *Leaflet - a JavaScript library for interactive maps*, 2010, https://leafletjs.com/. Accessed 4 September 2022.

"Node.js." *Node.js*, 2009, https://nodejs.org/en/. Accessed 4 September 2022.

"TLC Trip Record Data - TLC." *NYC.gov*, 2009, https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page. Accessed 3 September 2022.