● *Problem Statement Formation:*

The City of Seattle has created an on-street paid parking occupancy dataset and is providing access to this dataset for public use for research and entrepreneurial purposes under the City's Open Data Program. This data set contains records year to date in 2020 (with the seven data delay).

● *Context*

On-Street Paid Parking Occupancy Data

The data can be used to answer some of the following questions

- Where can I find a parking spot near me?
- Where is a street/block to find a parking spot right now? or in 1 hour?
- How much will be the parking cost?

Paid Parking Information Data

The City of Seattle Department of Transportation (SDOT) is providing this data set of transaction records at parking pay stations for paid street parking within city limits and transactions from the City's mobile payment vendor, called **PayByPhone**. The dataset is downloaded nightly with the prior days' paid parking transaction data. SDOT has transaction records dating back to January 2012. This data set is provided in conjunction with the Paid **Blockface** data. Purpose SDOT publishes these data on the City's data.seattle.gov. The mission of data.seattle.gov is to provide timely and accurate City information to increase government transparency and access to useful and well-organized data by the general public, non-governmental organizations, and City of Seattle employees. Specifically, with respect to parking, SDOT is providing these datasets to encourage programmers to develop mobile or other applications that can help people make smarter decisions to find parking faster and spending less time circling, stuck in traffic.

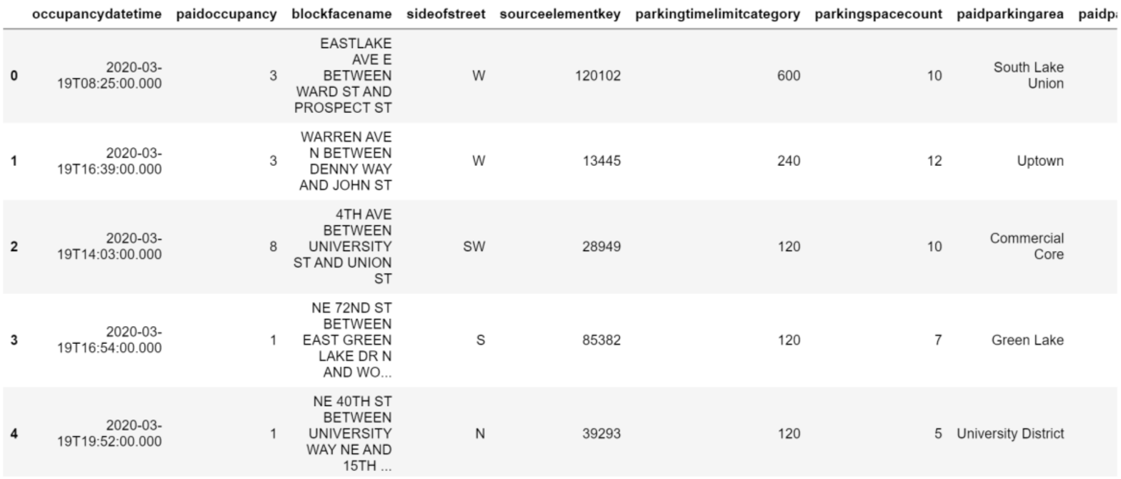
● *Data source(s)*

Year: 2020

Street Parking Occupancy data (Processed data)

2020 Paid Parking Occupancy (Year-to-date) | City of Seattle Open Data portal

- Seattle.gov update the data with a week delay.
- Granularity of the data is by the minute.
- About 131 million records in a year. (~45GB)
- Approx. total of 1.4 billion records since 2012. (~320GB)



	occupancydatetime	paidoccupancy	blockfacename	sideofstreet	sourceelementkey	parkingtimelimitcategory	parkingspacecount	paidparkingarea	paidp
0	2020-03-19T08:25:00.000	3	EASTLAKE AVE E BETWEEN WARD ST AND PROSPECT ST	W	120102	600	10	South Lake Union	
1	2020-03-19T16:39:00.000	3	WARREN AVE N BETWEEN DENNY WAY AND JOHN ST	W	13445	240	12	Uptown	
2	2020-03-19T14:03:00.000	8	4TH AVE BETWEEN UNIVERSITY ST AND UNION ST	SW	28949	120	10	Commercial Core	
3	2020-03-19T16:54:00.000	1	NE 72ND ST BETWEEN EAST GREEN LAKE DR N AND WO...	S	85382	120	7	Green Lake	
4	2020-03-19T19:52:00.000	1	NE 40TH ST BETWEEN UNIVERSITY WAY NE AND 15TH ...	N	39293	120	5	University District	

Street Parking Transaction data (Raw data)

Data Source Link

- Updated daily in the morning.

Instructions for Data Access

Please go to this URL source below.

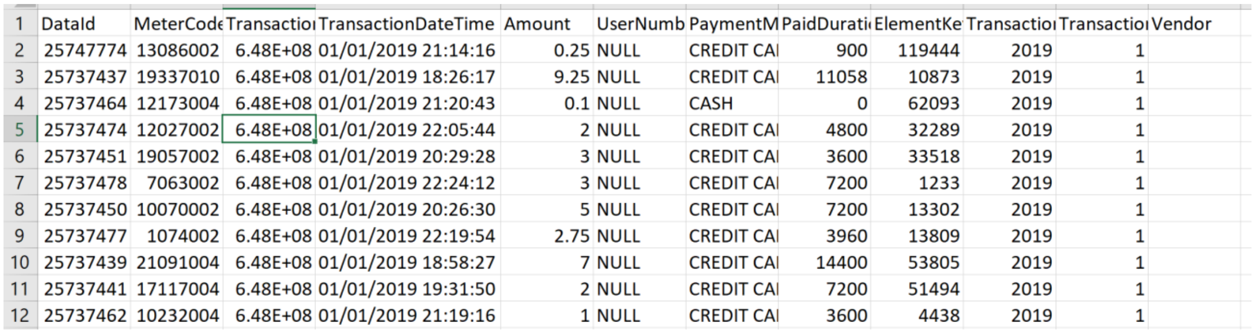
http://web6.seattle.gov/SDOT/wapiParkingStudy/api/ParkingTransaction?from=XXXXXXX&to=XXXXXXX

Be sure to enter the "from" and "to" date ranges requested. The date range of your request cannot exceed more than 7 days.

Request Parameter

Parameter	Details
From	From date range, format mmddyyyy.
To	To date range, format mmddyyyy

web6.seattle.gov/SDOT/wapiParkingStudy/api/ParkingTransaction?from=01012019&to=01072019

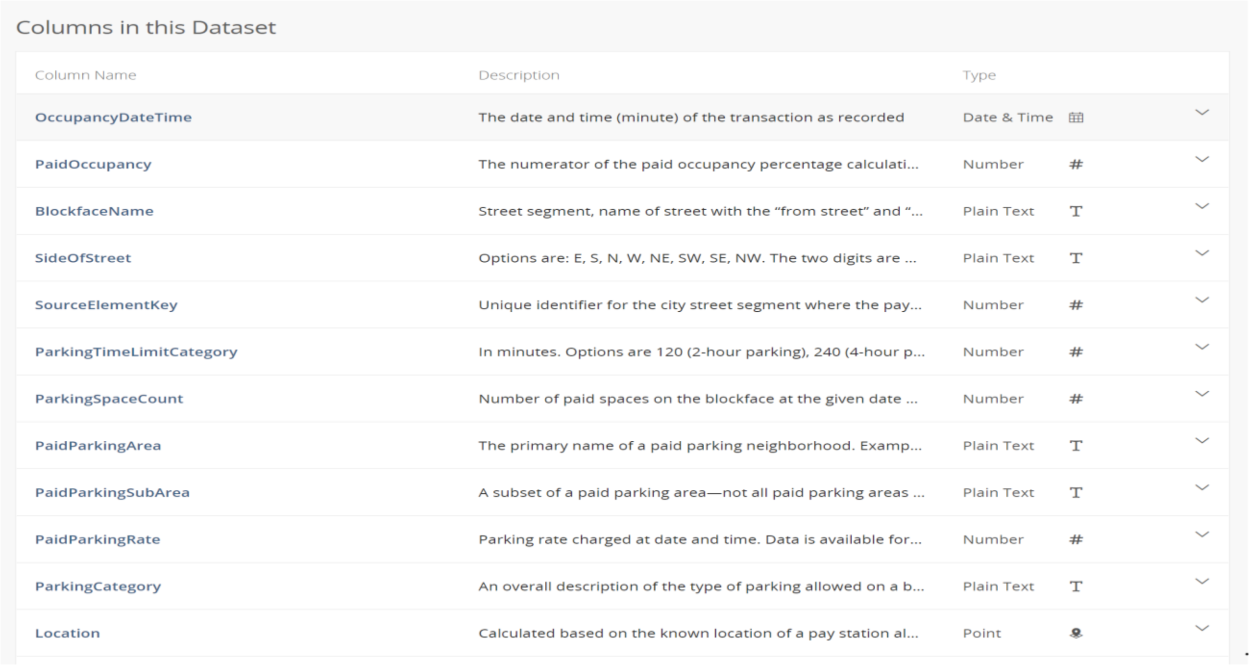


1	DataId	MeterCode	Transactioı	TransactionDateTime	Amount	UserNumb	PaymentM	PaidDurati	ElementKe	Transactioı	Transactioı	Vendor
2	25747774	13086002	6.48E+08	01/01/2019 21:14:16	0.25	NULL	CREDIT CAI	900	119444	2019	1	
3	25737437	19337010	6.48E+08	01/01/2019 18:26:17	9.25	NULL	CREDIT CAI	11058	10873	2019	1	
4	25737464	12173004	6.48E+08	01/01/2019 21:20:43	0.1	NULL	CASH	0	62093	2019	1	
5	25737474	12027002	6.48E+08	01/01/2019 22:05:44	2	NULL	CREDIT CAI	4800	32289	2019	1	
6	25737451	19057002	6.48E+08	01/01/2019 20:29:28	3	NULL	CREDIT CAI	3600	33518	2019	1	
7	25737478	7063002	6.48E+08	01/01/2019 22:24:12	3	NULL	CREDIT CAI	7200	1233	2019	1	
8	25737450	10070002	6.48E+08	01/01/2019 20:26:30	5	NULL	CREDIT CAI	7200	13302	2019	1	
9	25737477	1074002	6.48E+08	01/01/2019 22:19:54	2.75	NULL	CREDIT CAI	3960	13809	2019	1	
10	25737439	21091004	6.48E+08	01/01/2019 18:58:27	7	NULL	CREDIT CAI	14400	53805	2019	1	
11	25737441	17117004	6.48E+08	01/01/2019 19:31:50	2	NULL	CREDIT CAI	7200	51494	2019	1	
12	25737462	10232004	6.48E+08	01/01/2019 21:19:16	1	NULL	CREDIT CAI	3600	4438	2019	1	

- *Sneak Peak into the Data*

Seattle Parking Lot Occupancy Data

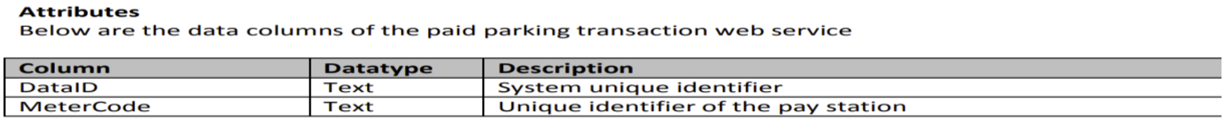
Each record in the dataset has following 12 columns:



Columns in this Dataset

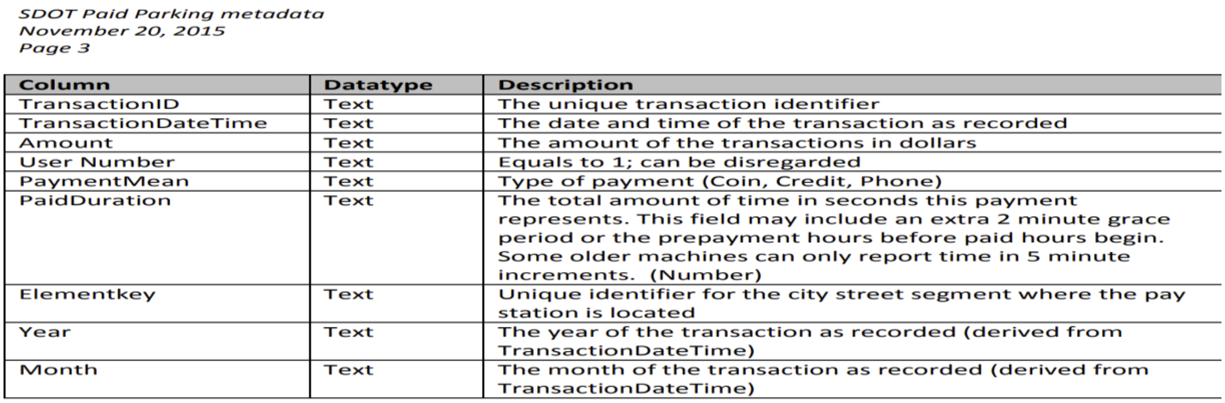
Column Name	Description	Type
OccupancyDateTime	The date and time (minute) of the transaction as recorded	Date & Time 📅
PaidOccupancy	The numerator of the paid occupancy percentage calculati...	Number #
BlockfaceName	Street segment, name of street with the "from street" and "...	Plain Text T
SideOfStreet	Options are: E, S, N, W, NE, SW, SE, NW. The two digits are ...	Plain Text T
SourceElementKey	Unique identifier for the city street segment where the pay...	Number #
ParkingTimeLimitCategory	In minutes. Options are 120 (2-hour parking), 240 (4-hour p...	Number #
ParkingSpaceCount	Number of paid spaces on the blockface at the given date ...	Number #
PaidParkingArea	The primary name of a paid parking neighborhood. Examp...	Plain Text T
PaidParkingSubArea	A subset of a paid parking area—not all paid parking areas ...	Plain Text T
PaidParkingRate	Parking rate charged at date and time. Data is available for...	Number #
ParkingCategory	An overall description of the type of parking allowed on a b...	Plain Text T
Location	Calculated based on the known location of a pay station al...	Point ●

Paid Parking Transaction Data



Attributes
Below are the data columns of the paid parking transaction web service

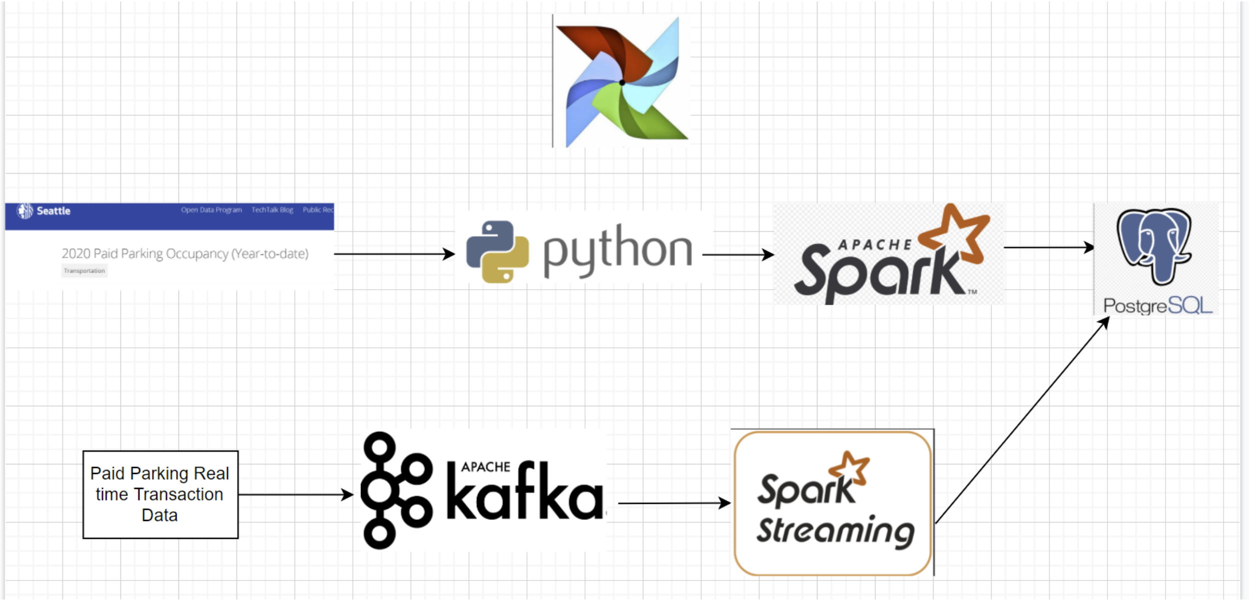
Column	Datatype	Description
DataID	Text	System unique identifier
MeterCode	Text	Unique identifier of the pay station



SDOT Paid Parking metadata
November 20, 2015
Page 3

Column	Datatype	Description
TransactionID	Text	The unique transaction identifier
TransactionDateTime	Text	The date and time of the transaction as recorded
Amount	Text	The amount of the transactions in dollars
User Number	Text	Equals to 1; can be disregarded
PaymentMean	Text	Type of payment (Coin, Credit, Phone)
PaidDuration	Text	The total amount of time in seconds this payment represents. This field may include an extra 2 minute grace period or the prepayment hours before paid hours begin. Some older machines can only report time in 5 minute increments. (Number)
Elementkey	Text	Unique identifier for the city street segment where the pay station is located
Year	Text	The year of the transaction as recorded (derived from TransactionDateTime)
Month	Text	The month of the transaction as recorded (derived from TransactionDateTime)

- *Proposed architecture for the solution and rationale behind it*



Data will be extracted from 2020 Seattle Parking Lot Dataset using Python libraries, transformed into usable format using Python library of Apache Spark i.e. Pyspark and loaded into Postgres Database. Airflow will be used to automate this ETL data pipeline for historical and future data

TBD: Real time processing

Apache Kafka and Spark Streaming will work together to extract the real time Paid Parking Transaction Data and push to Postgres Database.

The ETL solution will be scaled to Azure Cloud

- *Choice of technology for the solution and rationale*

Python:

Python is known for being the swiss army knife of programming languages. It’s especially useful in data science, backend systems, and server-side scripting. That’s because Python has strong typing, simple syntax, and an abundance of third-party libraries to use. Pandas, SciPy, Tensorflow, SQLAlchemy, and NumPy are some of the most widely used libraries in production across different industries.

Most importantly, Python decreases development time, which means fewer expenses for companies. For a data engineer, most code execution is database-bound, not CPU-bound. Because of this, it makes sense to capitalize on Python’s simplicity, even at the cost of slower performance when compared to compiled languages such as C# and Java.

Spark/Pyspark:

Apache Spark is a unified analytics engine for large-scale data processing. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for incremental computation and stream processing.

Postgres:

A multifunctional DBMS capable of processing complex queries and supporting massive databases

Airflow:

Airflow (https://airflow.apache.org/) is a configuration-as-code OSS solution for workflow automation. It is purely Python-based and there is no XML, YAML, etc. An Airflow workflow is defined as a DAG (Directed Acyclic Graph)coded in Python as a sequence of Tasks. It was originally developed at Airbnb in 2014; a top-level Apache Software Foundation project as of January 2019. It offers developers a way to programmatically author, schedule for execution, and monitor highly configurable complex workflows.

Apache Kafka:

Apache Kafka is an open-source stream-processing software platform developed by the Apache Software Foundation, written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Kafka can connect to external systems via Kafka Connect and provides Kafka Streams, a Java stream processing library. Kafka uses a binary TCP-based protocol that is optimized for efficiency and relies on a "message set" abstraction that naturally groups messages together to reduce the overhead of the network roundtrip. This "leads to larger network packets, larger sequential disk operations, contiguous memory blocks which allows Kafka to turn a stream of random message writes into linear writes."

Spark Streaming:

Apache Spark Streaming is a scalable fault-tolerant streaming processing system that natively supports both batch and streaming workloads. Spark Streaming is an extension of the core Spark API that allows data engineers and data scientists to process real-time data from various sources including (but not limited to) Kafka, Flume, and Amazon Kinesis. This processed data can be pushed out to file systems, databases, and live dashboards.

Azure Cloud:

TO DO