*● 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.

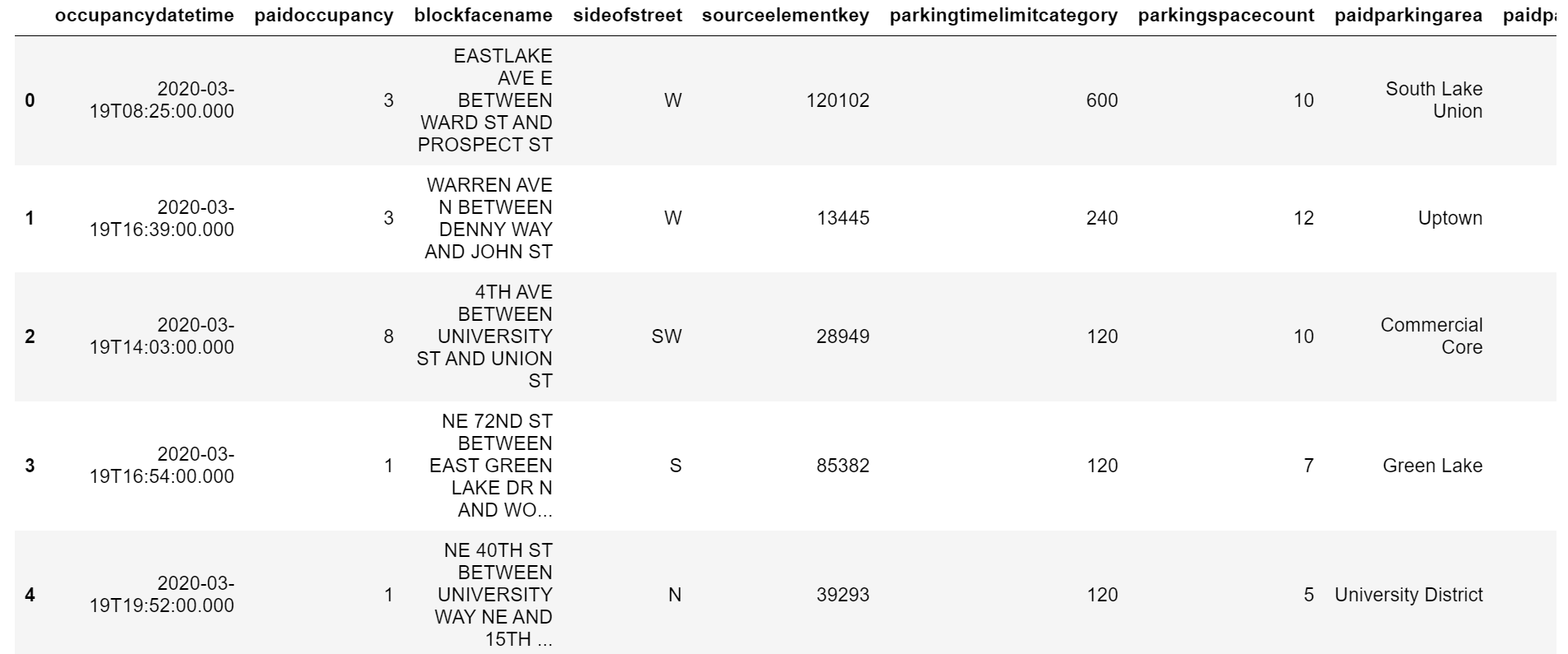
*● Data source(s)*

Year: 2020

#### Street Parking Occupancy data (Processed data)

[2020 Paid Parking Occupancy (Year-to-date) | City of Seattle Open Data portal](https://data.seattle.gov/Transportation/2020-Paid-Parking-Occupancy-Year-to-date-/wtpb-jp8d)

* 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)



#### Street Parking Transaction data (Raw data)

[Data Source Link](http://www.seattle.gov/Documents/Departments/SDOT/ParkingProgram/data/SeattlePaidTransactMetadata.pdf)

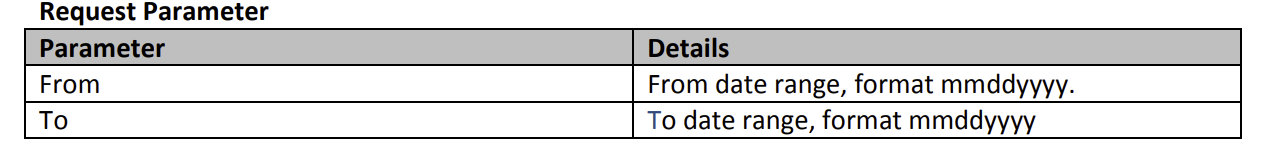
* Updated daily in the morning.

Instructions for Data Access

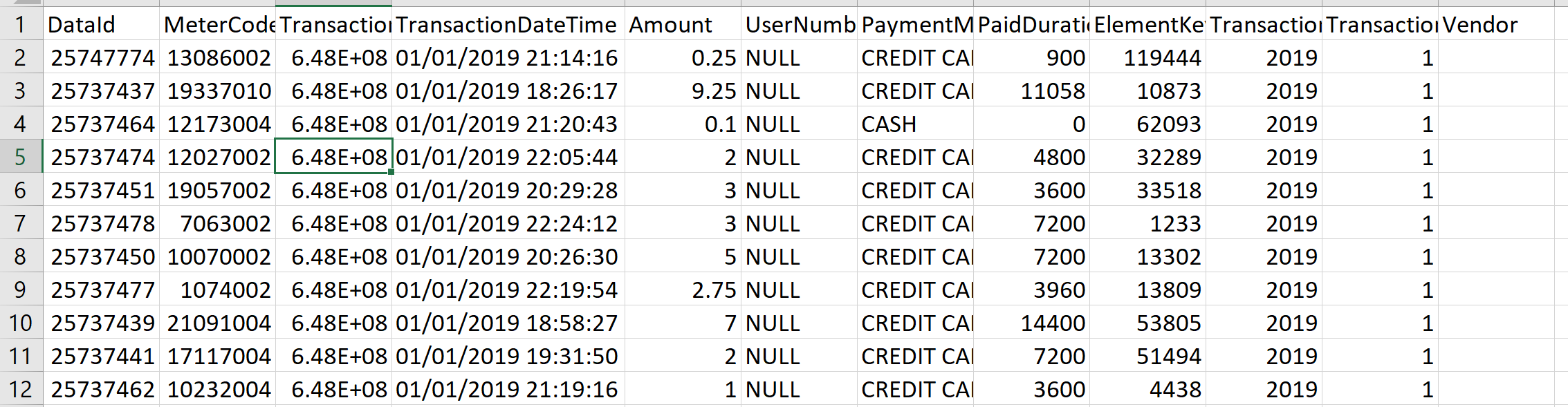
Please go to this URL source below.

<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.



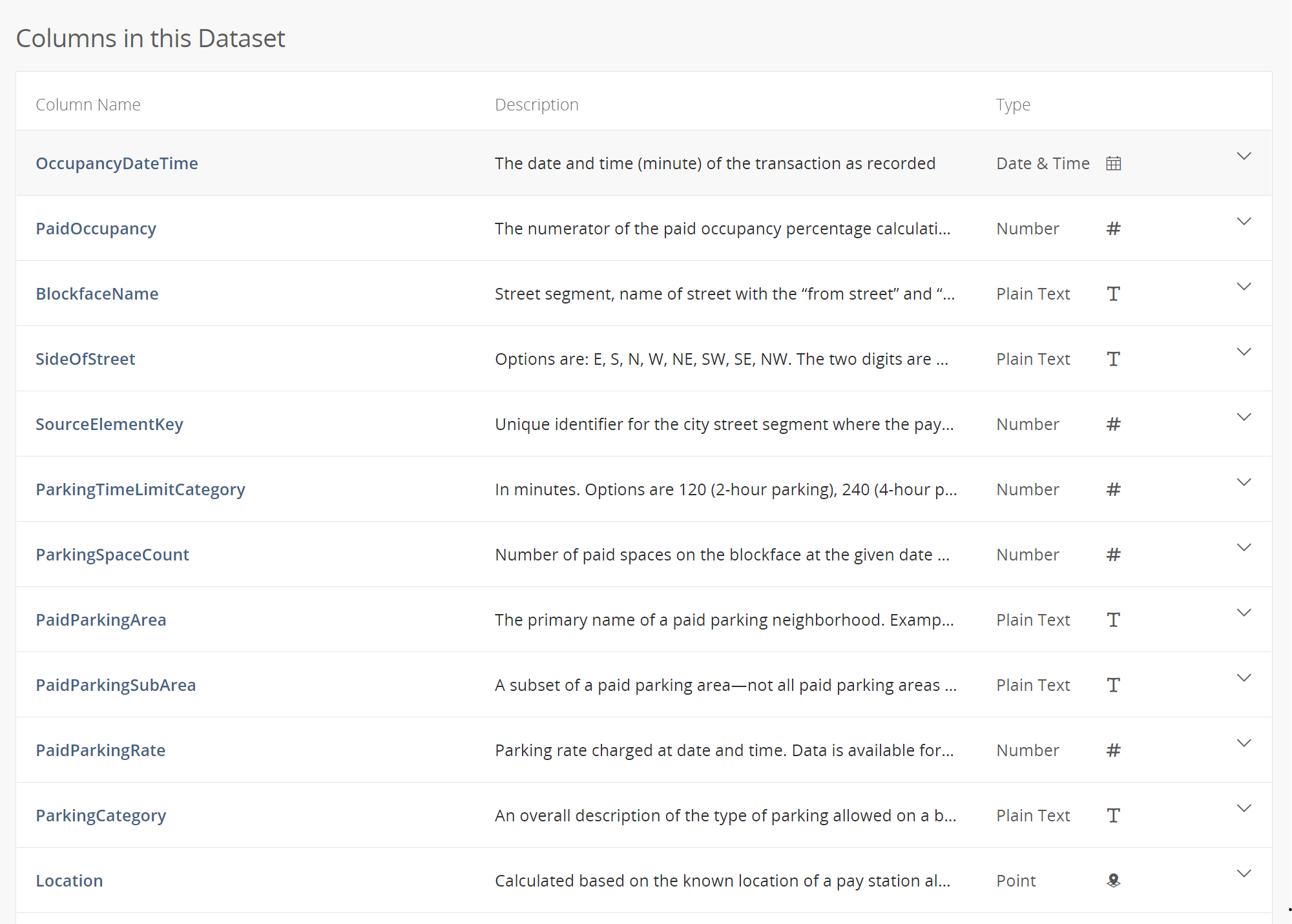
web6.seattle.gov/SDOT/wapiParkingStudy/api/ParkingTransaction?from=01012019&to=01072019



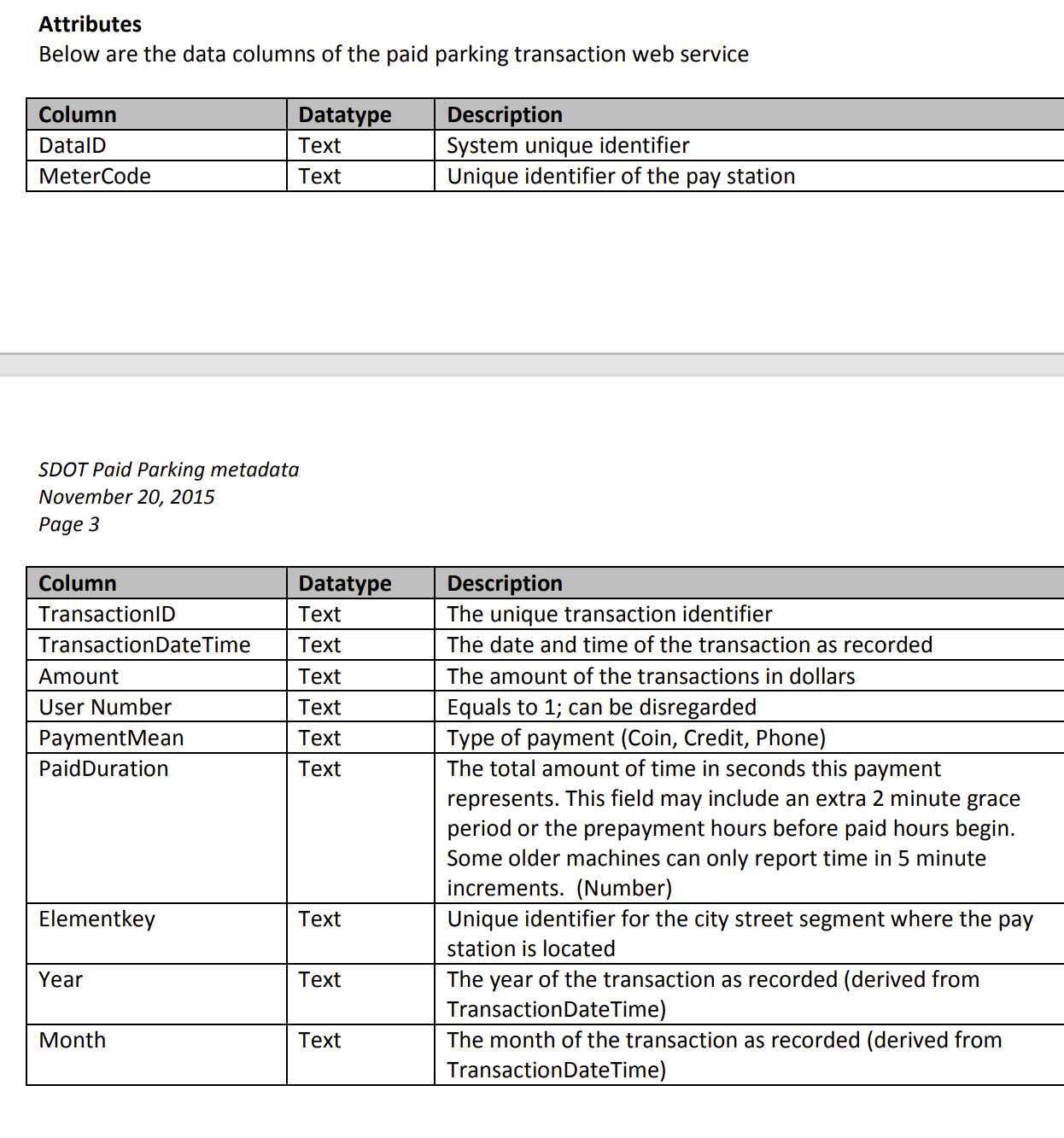
*● Sneak Peak into the Data*

Seattle Parking Lot Occupancy Data

Each record in the dataset has following 12 columns:



Paid Parking Transaction Data



● *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](https://realpython.com/courses/idiomatic-pandas-tricks-features-you-may-not-know/), SciPy, [Tensorflow](https://realpython.com/numpy-tensorflow-performance/), [SQLAlchemy](https://realpython.com/flask-connexion-rest-api/), and [NumPy](https://realpython.com/numpy-array-programming/) 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](https://realpython.com/oop-in-python-vs-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](https://spark.apache.org/docs/latest/sql-programming-guide.html) for SQL and structured data processing, [MLlib](https://spark.apache.org/docs/latest/ml-guide.html) for machine learning, [GraphX](https://spark.apache.org/docs/latest/graphx-programming-guide.html) for graph processing, and [Structured Streaming](https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html) 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