INTRODUCTION
- Big data management is a challenging issue that is becoming more prominent by the year in the software industry.
- As cloud storage solutions grow in popularity and availability, companies are increasingly encountering issues with software that cannot scale to handle bloated databases or huge archives.
- Network Innovations works with customer data from a large array of devices, providing web-based access to this data.
- After three months customer data is sent to AWS S3, and can no longer be accessed.
- Our team is tasked with building a web-based solution that provides easy and fast access to this huge data archive.

OBJECTIVE AND SCOPE
The project objective was to create an easy-to-use custom web application to retrieve, filter, and display a user specified subset of data from an AWS S3 bucket containing raw data in parquet files. One new file is stored each day, spanning years of historical data, with a count of nearly 2 billion recorded entries (hundreds of gigabytes!).

Most importantly, the server-side queries are required to run very fast relative to this large amount of data. The primary goal is for results to return in 30 seconds or less for a 1-3 month time period (which represents the average load on the system for a normal use case). Each fetch query is required to evaluate data for a single device over a user-specified date range. From here, users must be able to interact with data in a tabular format, with the ability to apply custom filters to any column. Finally, data results must be easily exportable to a .csv format when requested.

METHODS AND TECHNOLOGIES
- Development started by building an MVP product using a simple React front-end and Flask back-end using Pandas and a small set of sample data. With the MVP created, development followed a cycle of: Test -> Explore changes & alternatives -> Improve solution -> Repeat
- A full set of mock data was created using the provided sample and a custom Python script.
- Various back-end solutions were explored to improve query times and memory optimization:
  - Pandas - Easy to use library, but has limitations with speed and scalability
  - Dask - Very good at handling large datasets but poor performance for frontend integration
  - Polars - Optimal balance between speed and scalability when implemented correctly
- The technologies included in our final solution include Python, Pandas, and Polars in a Flask back-end, with a React.js front-end. Code is hosted on an AWS EC2 cloud computing instance, deployed using Docker, and is built to access parquet data in an AWS S3 storage bucket.

CONCLUSION
Our team has successfully created a custom web-based solution that meets all of the needs of Network Innovations while overcoming the numerous challenges of big data management.

An iterative development cycle was executed within our team, resulting in large performance improvements over time.

The primary goal of returning 1-3 months worth of data in under 30 seconds was surpassed, with the application being able to fetch 3 months of requested data in an average time of ~20 seconds.