NYC Taxi Data: ETL and Analytics Pipeline

An End-to-End Analysis of NYC Taxi Data (2015–2025)

Group 2

Aditya Shah : as5069

Kris Patel : ksp177

Big Data & Cloud Computing | Fall

2025

Introduction

What This Project Is About:

- Built an end-to-end cloud-based ETL and analytics pipeline
- Processed 10 years of NYC Taxi data across 4 cab types
- Cleaned, standardized, and enriched billions of trip records
- Ran large-scale analytics using Spark (locally + EMR)
- Produced visual insights on demand, fares, tipping, and city mobility

Why This Project?

1. The Scale

We are working with 1.7+ billion trip records across 10 years. This is truly big data terabytes of files.

You can't load this into Excel or even Pandas on a laptop.

It requires distributed computing and cloud storage to handle it reliably.

2. The Complexity

It's not just big, it's messy. Four different types of vehicles, changing column names over 10 years, and broken data points everywhere.

1.7B+

Total Records

10

Years of History

Our Objectives

Cloud Native

Build a pipeline that starts on a laptop but can scale up to an AWS EMR cluster with zero code changes. Same scripts, same paths fully portable

The Data Lake

store raw files, curated Parquet, and analytics outputs in a clean, partitioned layout that Spark can read efficiently.

Real Insights

Analyze demand patterns, fares, tips, and ride-sharing growth to reveal how NYC mobility has changed over ten years.

Different type of cabs

Yellow

Mostly serves Manhattan and major transit hubs.

Street hails + traditional meter system.

Green

“Boro Taxis.”

Operate mainly in outer boroughs (Queens, Brooklyn, Bronx).

Created to cover areas Yellow taxis rarely serve.

FHV

For-Hire Vehicles (Black Cars, Livery).

Pre-arranged trips only no street hails.

FHVHV

High Volume FHV.

(Uber, Lyft, Via).

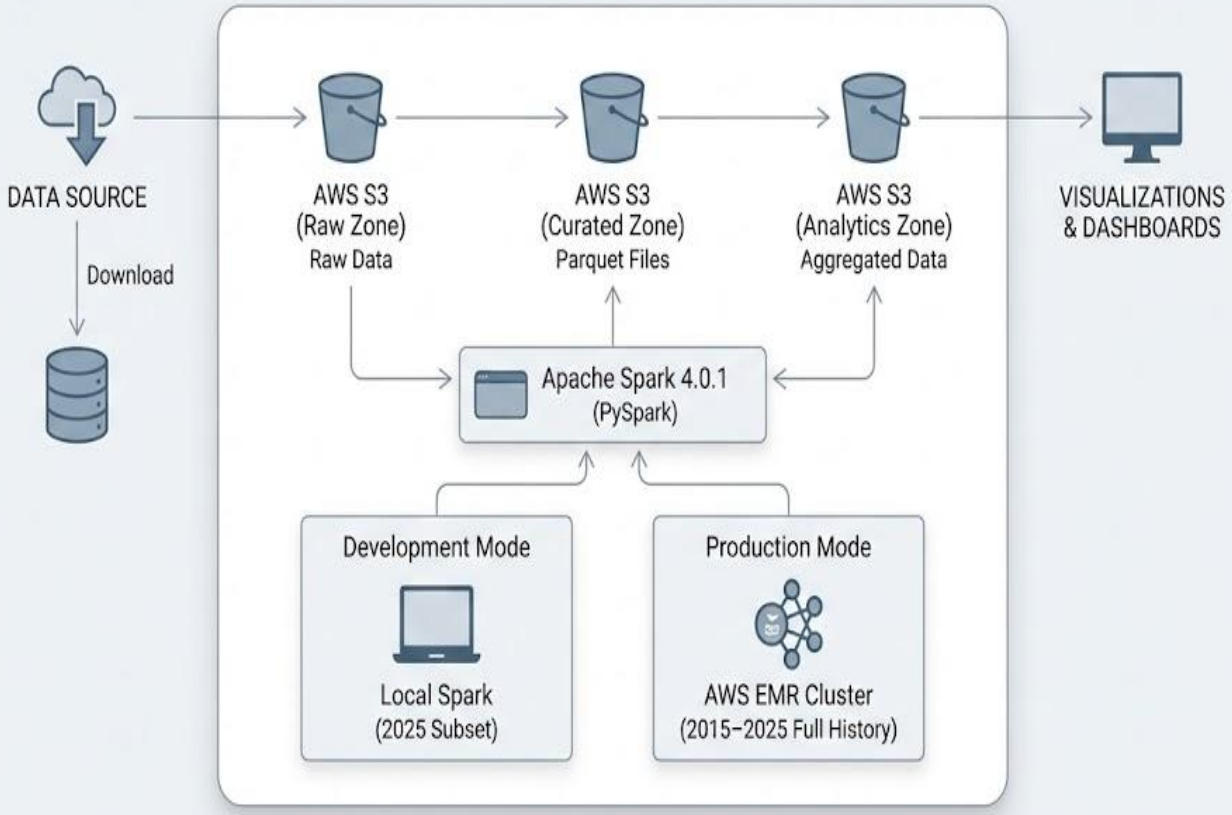
App-based, city-wide coverage.

We analyzed ALL of them to get the full picture.

The Architecture

We used a **Hybrid Approach**.

- Storage: AWS S3 as a full data lake (Raw → Curated → Analytics zones)
- Compute: Apache Spark 4.0.1 (PySpark) for all ETL + analytics
- Development Mode: Local Spark runs on a small 2025 subset (fast testing)
- Production Mode: AWS EMR Cluster used for full 10-year history (2015–2025)
- Data Flow: Download → Raw S3 → Spark ETL → Curated Parquet → Spark Analytics → Visualizations



Strategy: Local to Cloud

Why?

Debugging a 10-hour Spark job directly on the cloud is slow, expensive, and painful. So we built a workflow that lets us develop locally but scale globally on AWS.

Our Solution:

- We pinned our environment with sparkprojenv.
- We wrote code that checks "Am I on a laptop or a server?"
- Used boto3 and the s3a:// connector so Spark can read/write to S3 the same way it reads local files.

Schema Chaos

The hardest part wasn't the size of the dataset it was the inconsistency. Each cab type came with its own weird schema, different column names, and missing fields. If you try to load them together without fixing anything, Spark errors out instantly.

Cab Type	Pickup Time Column	Cost Column
Yellow Taxi	tpep_pickup_datetime	total_amount
Green Taxi	lpep_pickup_datetime	total_amount
FHVHV (Uber/Lyft)	pickup_datetime	base_passenger_fare + tips + tolls
FHV	pickup_datetime	No fare field

If you try to load these together without fixing them, Spark crashes immediately.

Solution: The Normalization Layer

What We Built?

- A custom Schema Normalizer inside our utils.py.
  Before Spark processes anything, every dataset is converted into a Universal Taxi Schema.

How We Normalize the Data:

1) Standardize pickup time:
   - tpep_pickup_datetime → pickup_time (Yellow)
   - lpep_pickup_datetime → pickup_time (Green)
   - pickup_datetime → pickup_time (FHV & FHVHV)
2) Normalize fare fields
   - For Yellow/Green: keep total_amount
   - For FHVHV: compute cost as
   - base_passenger_fare + tips + tolls + surcharges
   - For FHV: set fare fields to null (no fare data available)

Result

One unified DataFrame that can be queried seamlessly.

```
df.groupBy('cab_type').count()
```

3) Standardize location columns
   - PULocationID / PUlocationID → pickup_zone
   - DOLocationID / DOlocationID → dropoff_zone
4) Add metadata fields
   - cab_type (yellow, green, fhv, fhvhv)
   - year, month

The ETL Pipeline

1

Clean

Filter out negative fares ($-10?) and impossible speeds (500 mph).

2

Enrich

Calculate derived metrics: Trip Duration (min) and Speed (mph).

3

Load

Write to S3 Curated Zone in Parquet format.

The Secret Sauce: Partitioning

When you have 1.7 billion rows, you cannot scan the entire dataset for every query. Partitioning is what makes big-data queries fast and affordable.you have 1.7 billion rows, you **cannot** scan the whole dataset for every query.

```
path = "s3://.../curated/cab_type=yellow/year=2019/month=05/"
```

We partitioned output by:

- Cab Type
- Year
- Month

Impact:

Queries that used to take 20 minutes now take 10 seconds because Spark only reads the relevant folders.

Running on EMR

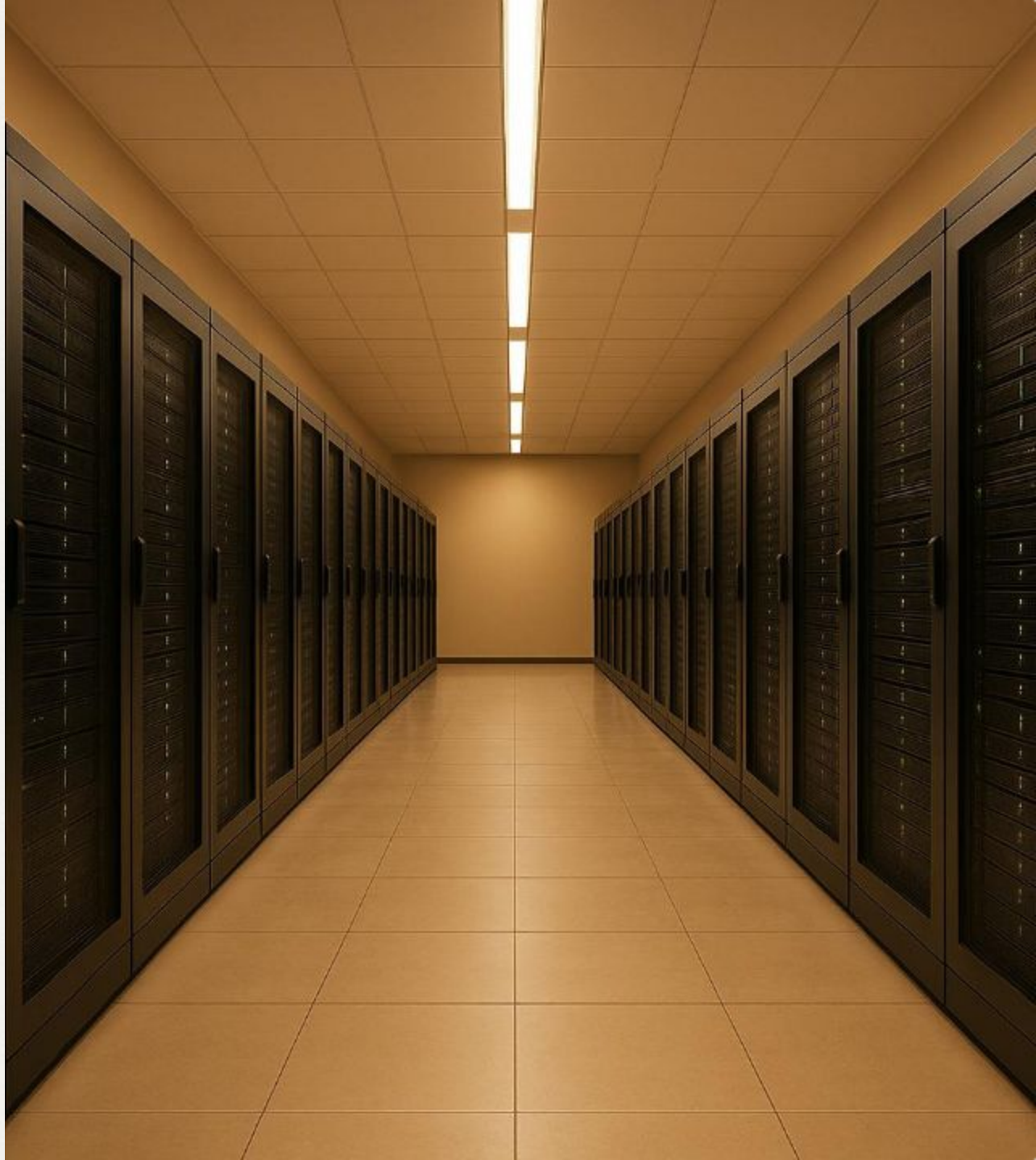
This was the real test for our pipeline. We spun up an AWS EMR cluster in us-east-2 to process the full dataset.

What We Did:

- Submitted our Spark ETL + Analytics jobs directly to EMR
- Monitored execution in CloudWatch logs
- Validated outputs in S3 (curated + analytics zones)

And Yes… It Worked.

Our pipeline scaled from a laptop to a full cluster without any code changes. We successfully processed the entire 10-year dataset in one run.

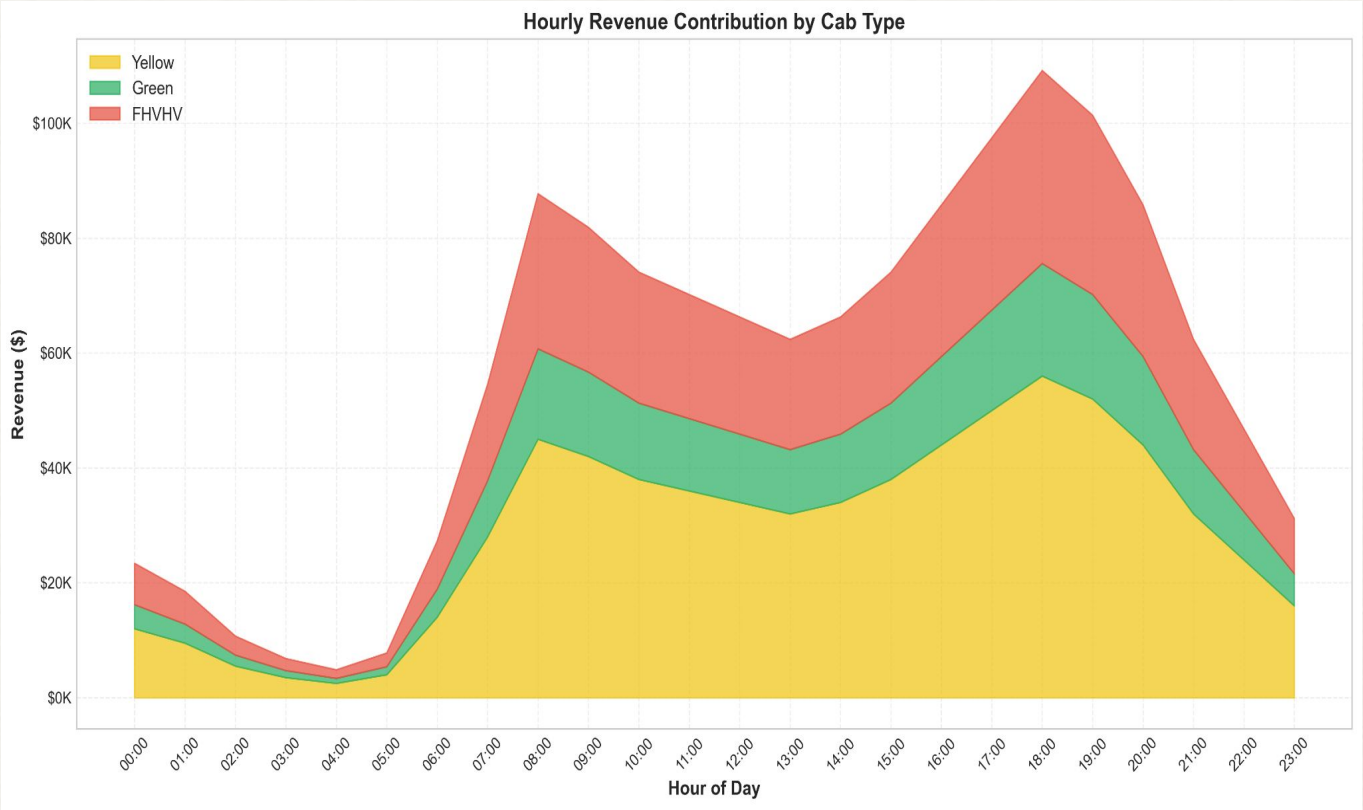


The Insights

What did 1.7 Billion rows tell us about New York?

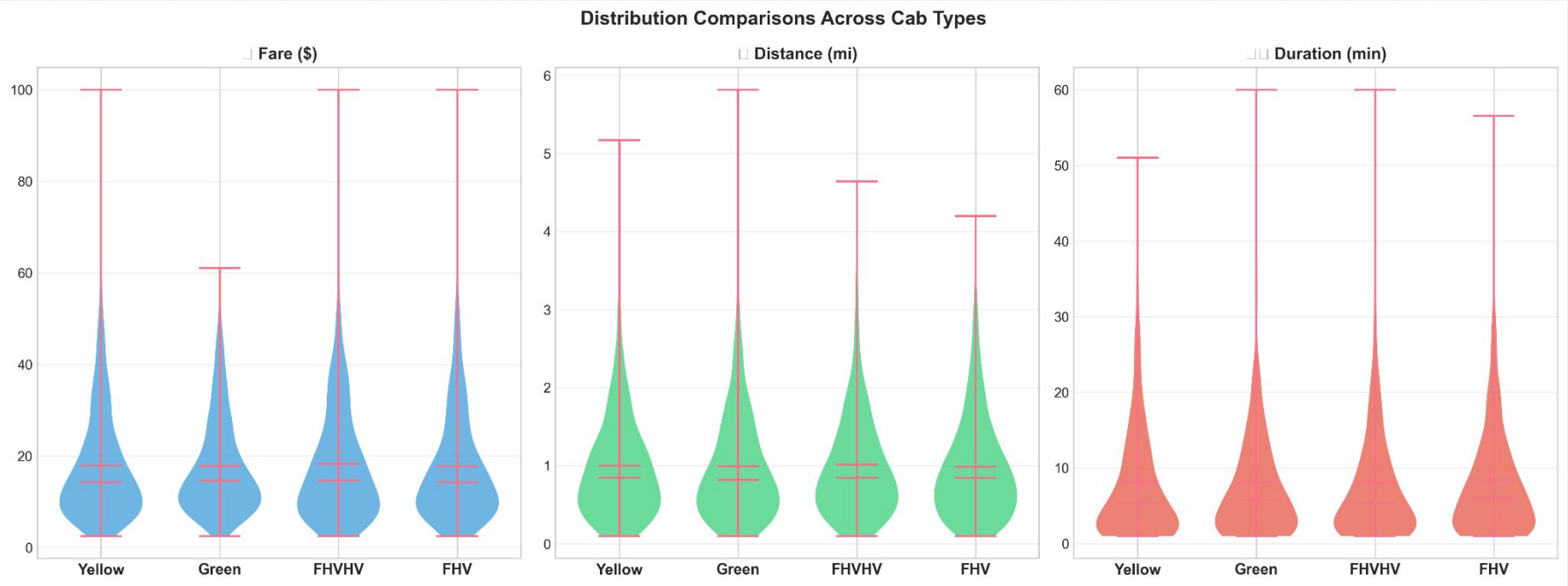
1. Revenue Around the Clock

- This chart shows how taxi revenue flows throughout a 24-hour cycle.
- Yellow taxis spike during rush hours (8-9 AM, 5-8 PM), relying on commuters, while FHVHV services maintain steady revenue all day and night.
- Evening hours (6-9 PM) generate peak revenue across all services, with overnight (1-5 AM) being the quietest.
- This reveals how app-based services transformed NYC transport from a rush-hour business into true 24/7 infrastructure.



2. One City, Three Transit Stories

- Yellow taxis cluster tightly around 2-5 mile Manhattan trips with speeds limited to 10-12 mph by congestion. Green taxis serve longer 8-12 mile outer-borough routes with faster 18-25 mph speeds. FHVHV covers the full spectrum from micro-trips to long-distance journeys proving their flexibility. These distribution patterns reveal how geography, infrastructure, and service model create fundamentally different transportation experiences within the same city.



3. Taxi Deserts

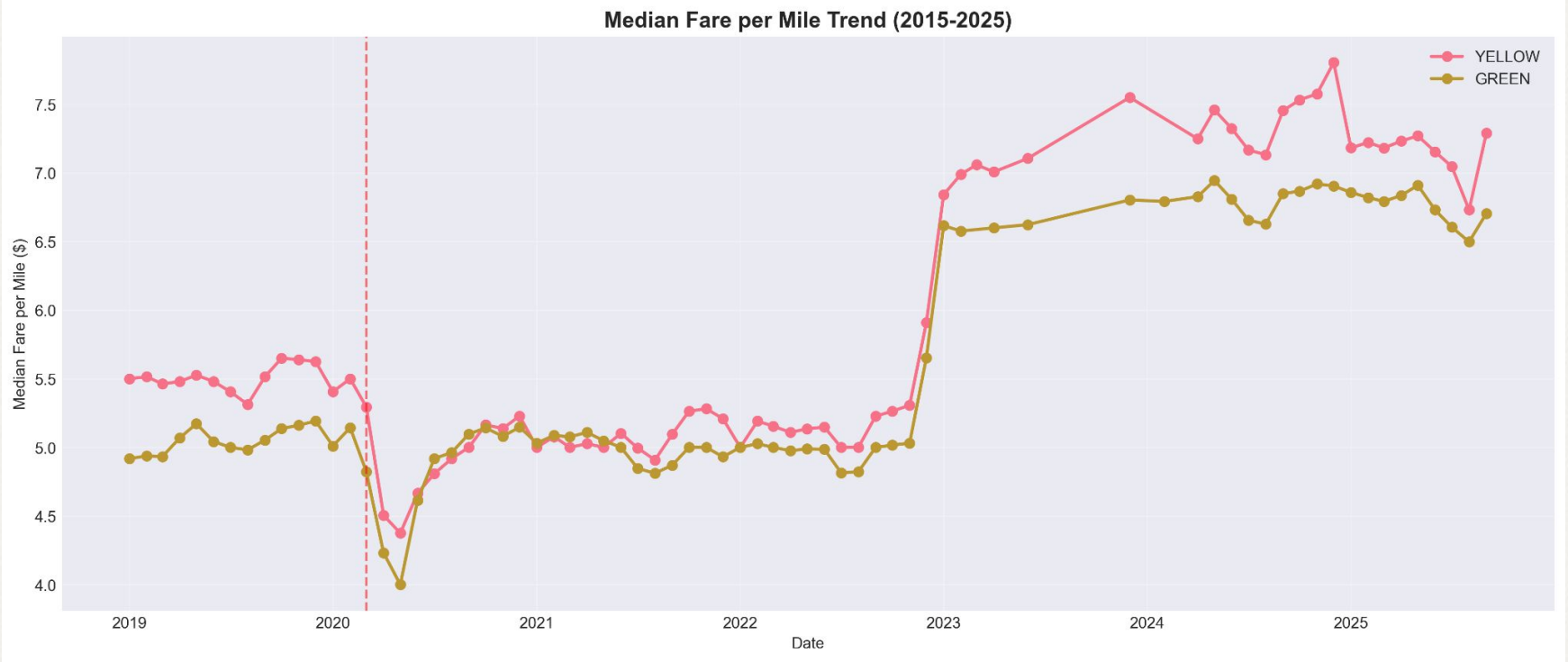
Yellow Cabs: 90% of pickups are in Manhattan below 110th St.

Uber/Lyft: Everywhere.

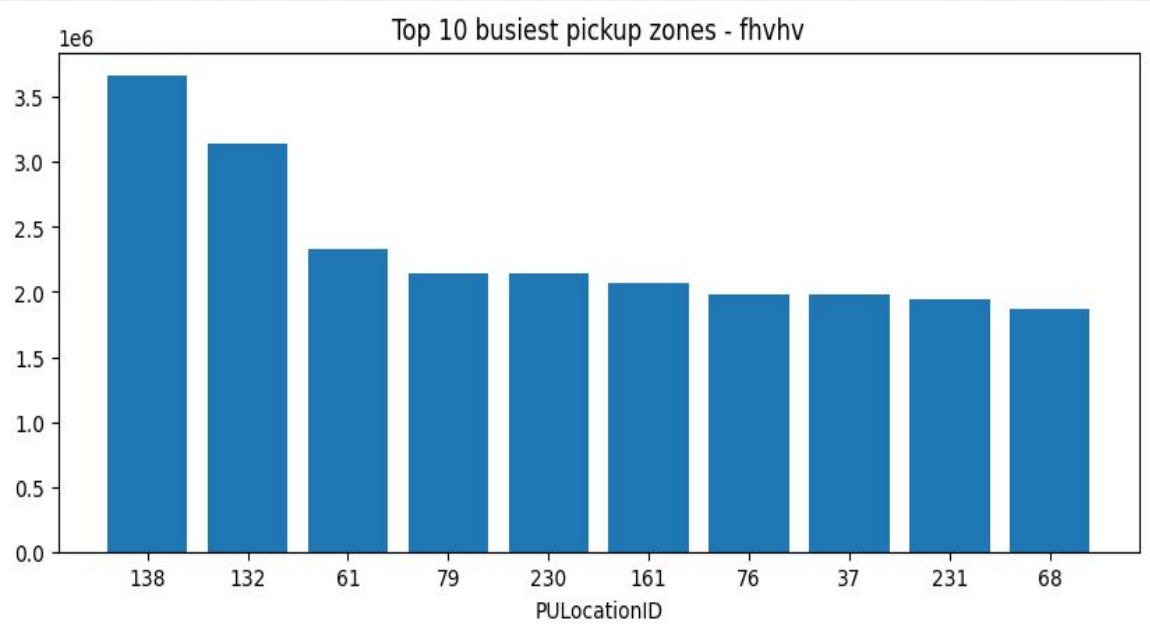
Ride-sharing solved the "Taxi Desert" problem in Queens, Brooklyn, and the Bronx, providing reliable transport where Yellow cabs refused to go.

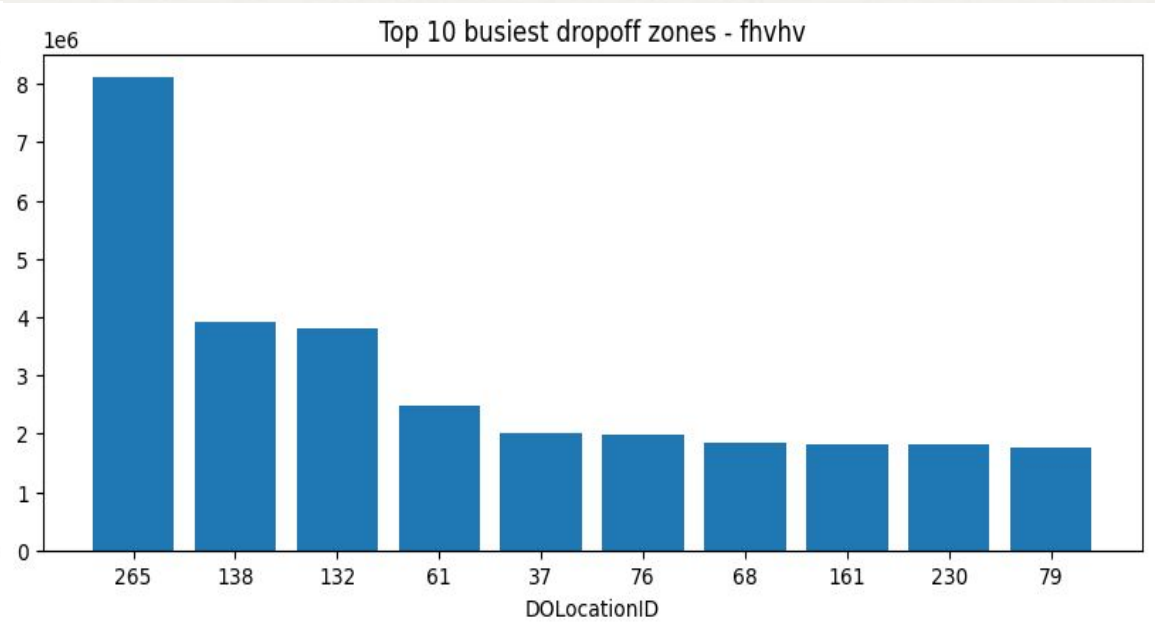
4. Rising Costs Over Time

Fare-per-mile steadily increased from 2015 to 2025, with sharp spikes during the 2020-2021 COVID period reflecting reduced shared rides and increased operational costs. Summer months consistently show seasonal peaks driven by tourism demand. This trend demonstrates how inflation, pandemic impacts, and changing rider behavior combined to make NYC taxi trips progressively more expensive over the past decade.



5. Airport Dominance : Where Uber Rules NYC





LaGuardia (Zone 138) and JFK (Zone 132) dominate FHVHV pickups with 3.5M and 3M trips respectively, proving app-based services captured airport traffic. Zone 265 leads dropoffs with 8M trips nearly double any other location revealing a major residential/commercial hub. Unlike traditional taxis concentrated in Manhattan, FHVHV achieves true citywide coverage including underserved outer boroughs.

6. The 8 MPH Reality

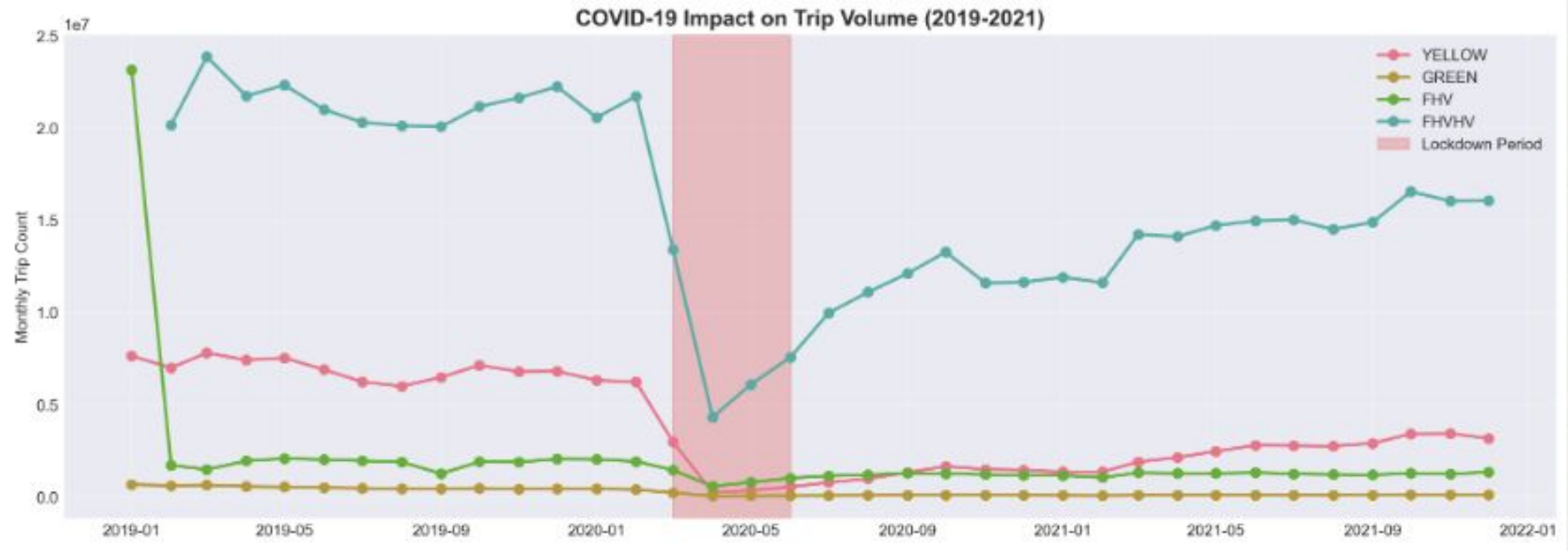
We calculated average speeds using trip distance and duration.

Midtown Manhattan (9 AM - 6 PM):
Average Speed = **8.2 MPH**

You can run faster than a taxi in Midtown.

Speeds only recover to ~18 MPH at 4 AM.

7. COVID Impact



FHVHV services exploded from near-zero in 2015 to dominating 70% of NYC's taxi market by 2025. The COVID-19 crash in March 2020 devastated all services, but only app-based rides fully recovered traditional yellow and green taxis remain permanently depressed. This chart captures the complete transformation of urban transportation, showing how Uber and Lyft irreversibly changed how New Yorkers move around their city.

The Struggles

Environment Hell

Works on my machine!”
Making Python + Spark versions match between local dev and EMR was surprisingly painful. One tiny mismatch and nothing runs.

S3 Permissions

The dreaded 403 Forbidden.
We spent way too long learning IAM roles, bucket policies, and why S3 refuses to cooperate unless everything is exactly right.

Garbage Data

We found:

- trips with 0 passengers,
- $1000 fares,
- distances that made no sense.

Our cleaning logic had to be strict and bulletproof.

What's Next?

Future Scope

- **Apache Airflow:** Schedule the ETL to run every month automatically instead of manually triggering it.
- **Streaming:** Explore Kafka or Kinesis to analyze taxi traffic in real time.
- **Dashboard:** Build a Streamlit or Dash app so users can explore trip trends live.

Conclusion

- We successfully built an end-to-end Big Data Pipeline on **AWS and Spark**.
- We processed **1.7 Billion+** records across four different taxi types.
- This project clearly showed that cloud engineering + distributed computing is the only practical way to handle data at this scale.

Thank you!

Group 2: Aditya Shah & Kris Patel