NYC Taxi Trip Explorer - Technical Report

1. Problem Framing and Dataset Analysis

Dataset Context

The NYC Taxi Trip Dataset contains 1.4 million trip records from 2016, providing a comprehensive view of urban mobility patterns in New York City. This dataset presents unique challenges and opportunities for understanding transportation dynamics.

Data Challenges Identified

- **Missing Values**: 17,065 records had invalid coordinates or missing critical data
- **Outliers**: Extreme trip durations (0-86,400 seconds) and distances (0-100+ km)
- **Data Quality**: Inconsistent borough classifications and coordinate precision
- **Scale**: Large dataset requiring memory-efficient processing strategies

Unexpected Observation

A significant discovery was the presence of "zero-distance" trips (same pickup/dropoff coordinates) representing 2.3% of all trips. These trips, averaging 8.5 minutes duration, likely represent passenger cancellations or meter errors, providing insights into operational inefficiencies.

2. System Architecture and Design Decisions

Architecture Overview

```
   Frontend              Backend              Database
   (HTML/CSS/JS)        (Express.js)        (PostgreSQL)

 • Interactive         • REST API            • Normalized
   Dashboard           • Data Cleaning         Schema
 • Visualizations      • Custom              • Indexing
 • Filtering             Algorithms          • Performance
```

Technology Stack Justification

- **Node.js/Express.js**: Chosen for rapid development and JavaScript ecosystem consistency
- **PostgreSQL**: Selected for ACID compliance and advanced indexing capabilities

- **HTML/CSS/JavaScript**: Ensures broad compatibility and no additional dependencies
- **Plotly.js**: Provides professional-grade visualizations with minimal code

Design Trade-offs

- **Memory vs. Performance**: Implemented streaming data processing to handle 1.4M records
- **Simplicity vs. Features**: Focused on core functionality with clean, maintainable code
- **Real-time vs. Batch**: Chose batch processing for data consistency and performance

3. Algorithmic Logic and Data Structures

Custom K-means Clustering Implementation

Problem: Group similar taxi trips based on geographic location and temporal patterns to identify mobility hotspots.

Approach: Manual implementation of K-means algorithm without external libraries.

Pseudo-code:

```
FUNCTION kMeans(data, k, maxIterations):
    IF data.length == 0 OR k <= 0:
        RETURN empty array

    centroids = initializeCentroids(data, k)

    FOR iteration = 1 TO maxIterations:
        clusters = createEmptyClusters(k)

        FOR each point IN data:
            distances = calculateDistances(point, centroids)
            nearestCluster = findMinimumDistance(distances)
            clusters[nearestCluster].add(point)

        newCentroids = calculateNewCentroids(clusters)

        IF converged(centroids, newCentroids):
            BREAK

        centroids = newCentroids

    RETURN filterNonEmptyClusters(clusters)
```

Time Complexity: O(n × k × i × d) where n=points, k=clusters, i=iterations, d=dimensions **Space Complexity**: O(n + k) for storing points and centroids

Key Features: - Handles empty clusters by reinitializing with random points - Uses Haversine distance for geographic accuracy - Normalizes duration dimension for balanced clustering - Implements convergence detection for efficiency

4. Insights and Interpretation

Insight 1: Manhattan Dominance

Derivation: Borough analysis query showing 78% of trips originate in Manhattan **Visualization**: Pie chart showing trip distribution by borough **Interpretation**: Manhattan's central business district generates the highest taxi demand, indicating economic activity concentration and transportation needs.

Insight 2: Rush Hour Patterns

Derivation: Hourly trip analysis revealing peak times at 8 AM and 6 PM **Visualization**: Bar chart showing trips per hour **Interpretation**: Clear commuter patterns with morning and evening peaks, suggesting work-related travel dominates taxi usage.

Insight 3: Speed-Distance Relationship

Derivation: Scatter plot analysis of trip speed vs. distance **Visualization**: Interactive scatter plot with color-coded duration **Interpretation**: Shorter trips show higher speed variability, while longer trips maintain more consistent speeds, indicating different traffic conditions and trip purposes.

5. Reflection and Future Work

Technical Challenges

- **Memory Management**: Large dataset required streaming processing and batch operations
- **Algorithm Optimization**: K-means convergence needed careful parameter tuning
- **Data Type Handling**: PostgreSQL string returns required frontend type conversion

Lessons Learned

- Streaming data processing is essential for large datasets
- Custom algorithms provide better control and understanding
- Data validation at multiple layers prevents runtime errors

Future Enhancements

- **Real-time Processing**: Implement WebSocket connections for live data updates
- **Machine Learning**: Add predictive models for trip duration and demand
- **Mobile Optimization**: Responsive design improvements for mobile devices
- **Advanced Analytics**: Time series analysis and seasonal pattern detection

Production Considerations

- Database connection pooling for concurrent users
- Caching layer for frequently accessed data
- API rate limiting and authentication
- Horizontal scaling with load balancers

Technical Specifications: - Database: 1,441,579 valid trip records - Processing Time: <200ms average query response - Memory Usage: <500MB peak during data import - Browser Compatibility: Modern browsers with ES6 support