Uber ETL Pipeline & Power BI Dashboard - Technical Report

# 1. Executive Summary

This technical report details the design, implementation, and deployment of the Uber ETL pipeline and Power BI dashboard. It covers system architecture, data ingestion, transformation logic, data warehouse schema, quality validation, deployment orchestration, and performance considerations.

# 2. System Architecture

• \*\*Data Ingestion Layer\*\*: Raw CSV files (`Uber\_Data.csv`) are ingested into a staging directory.  
• \*\*Processing Layer\*\*: Python scripts and Jupyter Notebook perform data cleaning, transformation, and enrichment.  
• \*\*Storage Layer\*\*: SQL Server hosts the data warehouse with fact and dimension tables.  
• \*\*Visualization Layer\*\*: Power BI dashboard connects to SQL Server to render interactive reports.

# 3. Source Data Description

The source dataset consists of timestamped trip records, including pickup/dropoff times, coordinates, trip distance, fare amount, tip, payment type, and vendor identifier. Quality considerations include missing values and inconsistent timestamps.

# 4. ETL Pipeline Implementation

\*\*Extract\*\*: Load `Uber\_Data.csv` into Pandas DataFrame.  
\*\*Transform\*\*:  
- Handle nulls in numeric fields by dropping or filling with median values.  
- Cast date strings to `datetime` objects.  
- Compute derived fields: trip duration, revenue per mile, distance bands.  
- Validate ranges and flag anomalies (e.g., zero-distance trips).  
\*\*Load\*\*: Write staging table via `BULK INSERT`, then merge into fact and dimensions using T‑SQL scripts.

# 5. Data Warehouse Schema

Defined in `Uber\_Data\_Warehouse\_Schema.SQL`:  
- \*\*Dimension Tables\*\*: `dim\_date`, `dim\_vendor`, `dim\_payment\_type`.  
- \*\*Fact Table\*\*: `fact\_trips` storing foreign keys to dimensions and measures (fare, tip, distance, duration).

# 6. Data Quality & Validation

- \*\*Schema Validation\*\*: Ensure CSV columns match expected schema.  
- \*\*Null Checks\*\*: Automated scripts to report missing or out‑of‑range values.  
- \*\*Record Counts\*\*: Pre‑ and post‑load counts to verify completeness.

# 7. Deployment & Orchestration

• \*\*Scheduling\*\*: Python `.py` script (`ETL\_and\_Advanced\_EDA.python`) is scheduled via cron or Windows Task Scheduler.  
• \*\*CI/CD\*\*: Integrate with GitHub Actions for automated testing of transformation logic and deployment to SQL Server.  
• \*\*Error Handling\*\*: Try/except blocks in Python log failures to a central monitoring table in SQL Server.

# 8. Performance & Scalability

- \*\*Batch Size Tuning\*\*: Adjust `BULK INSERT` batch sizes for optimal throughput.  
- \*\*Indexing\*\*: Apply clustered index on staging and fact tables for faster insert and query.  
- \*\*Parallel Processing\*\*: Use Python multiprocessing to transform partitions of data concurrently.

# 9. Security & Compliance

- \*\*Credentials\*\*: SQL connection strings stored securely via environment variables.  
- \*\*Data Privacy\*\*: No PII present in the dataset; any location data is aggregated.  
- \*\*Access Control\*\*: Power BI reports restricted to authorized users via workspace permissions.

# 10. Dependencies & Environment Setup

- \*\*Python\*\*: 3.8+ with Pandas, NumPy, pyodbc, SQLAlchemy.  
- \*\*SQL Server\*\*: 2019+ instance with necessary permissions.  
- \*\*Power BI Desktop\*\*: Latest version for `.pbix` file consumption.  
- \*\*Configuration\*\*: `config.yaml` for DB credentials, file paths, and parameters.

# 11. Usage & Contribution Guidelines

- \*\*Running ETL\*\*: `python ETL\_and\_Advanced\_EDA.python --config config.yaml`.  
- \*\*Notebook\*\*: Use `Uber\_Data\_Pipeline.ipynb` for ad-hoc analysis and prototyping.  
- \*\*Pull Requests\*\*: Fork, create feature branch, add tests for new transformations, submit PR.

# 12. Future Enhancements

- Real-time streaming ingestion (Kafka/Event Hubs).  
- Machine learning model integration for demand forecasting.  
- Geospatial clustering to identify hotspots.  
- Automated anomaly detection and alerting.