Real-Time Taxi Trip Data Pipeline for Data Engineering Portfolio

# PROJECT BACKGROUND

As a newly assigned data engineer, your first challenge is to build a real-time data pipeline from scratch that simulates how companies process, transform, and visualize streaming data. This hands-on project will showcase your ability to work with tools like Apache Kafka, Python, Azure, and GitHub—all on a Linux-based system.

# OBJECTIVE

Design, implement, and deploy a real-time data processing system that ingests NYC taxi trip data, processes it, stores it, and visualizes key insights.

# TOOLS & TECHNOLOGIES

- Linux (Ubuntu)  
- Python  
- Apache Kafka (local or Confluent Cloud)  
- Azure SQL Database or Azure Blob Storage  
- Power BI (Service) or Streamlit for visualization  
- Git & GitHub

# PROJECT TASKS

Phase 1: Ingestion (Kafka Producer)  
- Obtain a sample NYC taxi trip CSV dataset  
- Install Kafka locally or run it via Docker  
- Write a Python script that reads rows from the CSV and sends them one at a time to a Kafka topic  
  
Phase 2: Processing (Kafka Consumer)  
- Create a Python Kafka consumer that listens to the Kafka topic  
- Process each row (e.g., compute trip duration, fare/km, validate fields)  
- Output to console or a local file for testing  
  
Phase 3: Storage  
- Store processed data in:  
 - Azure SQL Database (via pyodbc or SQLAlchemy), OR  
 - Azure Blob Storage as Parquet or CSV (via azure-storage-blob SDK)  
  
Phase 4: Visualization  
- Option A: Build a Power BI dashboard using Azure data source (if on cloud)  
- Option B: Create a Streamlit dashboard showing:  
 - Trip counts by hour  
 - Average fare/km  
 - Longest and shortest trips  
  
Phase 5: Version Control  
- Organize your code into a GitHub repo:  
 - /ingestion — Kafka producer code  
 - /processing — Kafka consumer and transformation code  
 - /storage — Azure interaction scripts  
 - /visualization — Streamlit or Power BI files  
 - README.md — Project overview, setup steps, and screenshots

# MILESTONES

1. Kafka setup and producer script tested  
2. Consumer script working and saving to local file  
3. Azure SQL or Blob storage integrated  
4. Streamlit or Power BI dashboard created  
5. GitHub repository organized and published

# NOTES FOR LINUX SETUP

- Use systemd or cron jobs for scheduling  
- Use virtual environments for Python (venv)  
- Test scripts from terminal and document errors/fixes  
- Optionally use Docker Compose to manage Kafka stack

# DELIVERABLES

- GitHub link with working code and README  
- Screenshots of dashboard  
- Optional: Blog-style README write-up  
- This document (as downloadable project brief)

# SUPPORT

You have a senior data engineer (ChatGPT) assigned to guide you. Ask for help anytime on code, error logs, or deployment strategies.