Docker Trip Data Analysis Stack

Introduction

This project demonstrates a multi-container Docker application for analyzing transportation trip data using PostgreSQL and Python. Built for the Cloud Computing for Data Analysis course (ITCS 6190/8190), this assignment introduces fundamental containerization concepts including service orchestration, database integration, and automated deployment workflows.

The application consists of two containerized services: a PostgreSQL database that stores trip records (city, duration, fare) and a Python analytics service that connects to the database, executes statistical queries, and generates comprehensive reports in both console and JSON formats.

Key Learning Objectives: - Understand multi-container Docker applications - Learn service networking and DNS resolution - Practice database initialization and seeding - Implement container health checks and dependencies - Master Docker Compose orchestration - Build reproducible development workflows

Prerequisites

Before starting this project, ensure you have the following tools installed:

Required Software

- **Docker Desktop** (v20.10+) - Container runtime and management
- **Docker Compose** (v2.0+) - Multi-container orchestration
- **Make** - Build automation (installation guide below)
- **Git** - Version control
- **IDE** - VS Code, PyCharm, or similar

Installation Verification

Test your Docker installation:

```
docker --version
docker compose version
docker run hello-world
```

How to Install Make Using Homebrew

For macOS Users

Step 1: Install Homebrew (if not already installed)

```
# Install Homebrew package manager
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.s
```

```
# Verify Homebrew installation
brew --version
```

Step 2: Install Make

```
# Install make using Homebrew
brew install make

# Verify installation
make --version
```

Step 3: Update PATH (if necessary) Some systems may need to update the PATH to use the Homebrew version of make:

```
# Add to your shell profile (~/.zshrc or ~/.bash_profile)
echo 'export PATH="/opt/homebrew/bin:$PATH"' >> ~/.zshrc

# Reload your shell configuration
source ~/.zshrc

# Verify make is working
which make
make --version
```

Project Structure

```
docker-trip-analysis/
  app/                          # Python Analytics Service
     main.py                   # Analytics application with database queries

  db/                           # PostgreSQL Database Service
     Dockerfile                # Database container configuration
     init.sql                  # Schema creation and seed data

  out/                          # Generated Output Directory
     summary.json              # Analytics results (created at runtime)

  compose.yml                   # Multi-container orchestration configuration
  Dockerfile                    # Python container configuration
  Makefile                     # Build automation and common commands
  README.md                    # Project documentation (this file)
```

File Descriptions

app/Dockerfile - Defines the Python container with psycopg PostgreSQL driver
app/main.py - Analytics engine that connects to database and computes

statistics
db/Dockerfile - PostgreSQL container with automatic initialization
db/init.sql - Database schema and sample trip data
compose.yml - Service definitions, networking, and dependencies
Makefile - Automation scripts for building, running, and cleaning

How to Run

Method 1: Using Make (Recommended)

The Makefile provides convenient commands for common operations:

```
# Build and run the complete stack (recommended for first run)
make all
```

Method 2: Using Docker Compose Directly

If you prefer direct Docker Compose commands:

```
# Build and start all services
docker compose up --build
```

Sample output:

When successful, you'll see analytics output like:

```
{
  "total_trips": 6,
  "avg_fare_by_city": [
    {"city": "Charlotte", "avg_fare": 16.25},
    {"city": "New York", "avg_fare": 19.0},
    {"city": "San Francisco", "avg_fare": 20.25}
  ],
  "top_by_minutes": [
    {"city": "San Francisco", "minutes": 28, "fare": 29.3},
    {"city": "New York", "minutes": 26, "fare": 27.1},
    {"city": "Charlotte", "minutes": 21, "fare": 20.0}
  ]
}
```

The JSON output is also saved in the out/summary.json file.

Troubleshooting

Common Issues and Solutions

1. Make Command Not Found Problem: `make: command not found` **Solution:**

```
# macOS - Install using Homebrew
brew install make

# Windows - Install using Chocolatey
choco install make

# Linux - Install build tools
sudo apt-get install build-essential
```

2. Docker Desktop Not Running **Problem:** `Cannot connect to the Docker daemon` **Solution:** - Start Docker Desktop application - Wait for Docker to fully initialize - Verify with: `docker ps`

3. Port 5432 Already in Use **Problem:** `Port 5432 is already allocated` **Solution:**

```
# Find what's using the port
lsof -i :5432  # macOS/Linux
netstat -ano | findstr :5432  # Windows

# Stop conflicting PostgreSQL service
brew services stop postgresql  # macOS
sudo service postgresql stop  # Linux

# Or modify compose.yml to use different port
```

4. Database Connection Failures **Problem:** App cannot connect to database **Solution:**

```
# Check database health
docker compose exec db pg_isready -U appuser -d appdb

# View database logs
docker compose logs db

# Restart with clean slate
make clean && make all
```

Suggested Improvements

1. Security Enhancements Replace hardcoded passwords with Docker secrets to improve security in production environments. This prevents sensitive credentials from being exposed in configuration files or environment variables.

2. Advanced Analytics Implement more sophisticated statistical analysis including revenue trends, trip duration distributions, and outlier detection. This

would provide deeper insights into transportation patterns and help identify anomalies in the data.

3. Data Visualization Add chart generation capabilities using matplotlib or plotly to create visual reports alongside JSON output. This would make the analytics more accessible and easier to interpret for stakeholders.