Trip Analytics Stack

What This Stack Does

This is a containerized data analytics stack that processes trip data from a PostgreSQL database. The stack consists of a Python application that connects to a PostgreSQL database, executes analytical queries to compute trip statistics (total trips, average fare by city, and top N longest trips), and outputs the results both to the console and as a JSON file. The entire stack is orchestrated using Docker Compose for easy deployment and management.

Prerequisites

- Docker and Docker Compose installed
- Make (optional, for using Makefile commands)

Commands to Run/Stop

Using Make (Recommended)

```
# Build and run the entire stack
make up

# Stop and clean up containers with volumes
make down

# Clean output directory and run fresh
make clean up

# Or simply run all (clean + up)
make all

# Build only (without running)
make build
```

Using Docker Compose Directly

```
# Build and run the stack
docker compose up --build

# Stop and remove containers with volumes
docker compose down -v

# Build only
docker compose build
```

Example Output

When the stack runs successfully, you'll see output like this in the console:

```
=== Summary ===
{
  "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
    },
    {
      "city": "Charlotte",
      "minutes": 12,
      "fare": 12.5
    },
    {
      "city": "San Francisco",
      "minutes": 11,
      "fare": 11.2
    },
    {
      "city": "New York",
      "minutes": 9,
      "fare": 10.9
    }
  ]
}
```

Where Outputs Are Written

- **Console**: Results are printed to standard output in JSON format
- **File**: Results are saved to `./out/summary.json` (mapped from `/out/summary.json` inside the container)
- **Log Files**: Container logs can be viewed with `docker compose logs`

Troubleshooting

Database Not Ready

Symptom: App shows "Waiting for database..." messages

```
app-1  | Waiting for database...
app-1  | Failed to connect to Postgres: connection failed...
```

Solution: Wait for the database to fully initialize. The app has built-in retry logic and will automatically connect once the database is ready. This is normal during first startup.

Permission Issues with `out/` Directory

Symptom: Permission denied errors when writing to output directory

```
app-1  | PermissionError: [Errno 13] Permission denied: '/out/summary.json'
```

Solutions:

- Ensure the `out/` directory exists and is writable: `mkdir -p out && chmod 755 out`
- Run `make clean` to reset the output directory
- On Linux/macOS, check that your user has write permissions to the project directory

Container Build Issues

Symptom: Docker build failures or missing dependencies **Solutions**:

- Ensure Docker is running: `docker --version`
- Clean up Docker cache: `docker system prune`
- Rebuild without cache: `docker compose build --no-cache`

Port Conflicts

Symptom: Port already in use errors **Solution**: Stop other PostgreSQL instances running on port 5432 or modify the port mapping in `compose.yml`

Environment Variable Issues

Symptom: Connection refused or authentication errors **Solution**: Verify that the environment variables in `compose.yml` match between the `db` and `app` services

Architecture

- **Database**: PostgreSQL 16 with sample trip data
- **Application**: Python 3.11 with psycopg library for database connectivity
- **Orchestration**: Docker Compose for multi-container management
- **Data**: Sample dataset with 6 trips across 3 cities (Charlotte, New York, San Francisco)

Configuration

The number of top trips returned can be configured by modifying the `APP_TOP_N` environment variable in `compose.yml` (default: 10).

Security Note

The credentials in this repository (`secretpw`, `appuser`, `appdb`) are safe to commit as they are demo credentials for a local development environment that only runs in isolated Docker containers. These are not production secrets and pose no security risk since they cannot be used to access any external systems.

Reflection

This project gave me hands-on lessons in containerized app development and data workflows. Using Docker Compose taught me the importance of orchestrating services with health checks, handling dependencies, and managing environment variables across containers. I also saw how crucial error handling and retry logic are in distributed systems.

If I improved the stack, I'd add structured logging, stronger data validation, better error handling for edge cases, and flexible configuration management. I'd also include monitoring and metrics to track performance and database queries.