**API routes for inferencing**

To solve this problem, we need to optimize the delivery of orders from the warehouse in Cincinnati, OH, using the available trucks and drivers. The goal is to minimize the overall cost, which includes factors like driver salaries, fuel consumption, and truck capacity utilization.

**Understanding the Data**

1. **item\_info.csv**: Contains details about items, including:
   * ItemId: Unique identifier for each item.
   * weight (pounds): Weight of each item.
   * warehouse origin: All items originate from *Cincinnati, OH*.
2. **orders.csv**: Contains details about orders, including:
   * Company: The customer.
   * Item: The item ID.
   * Number of Units: The quantity of the item ordered.
   * Destination: The city and state for delivery.
   * There are two unnamed columns that appear to be empty.
3. **trucks.xlsx (Sheet1)**: Contains details about available trucks and drivers:
   * Truck ID: Unique truck identifier.
   * Driver Name: Assigned driver.
   * Years Experience: Experience level of the driver.
   * Truck Type (length in m): The truck size.
   * Weight Capacity (kg): The maximum weight capacity.
   * Top Speed (km/h): The truck’s speed.
   * Driver Hourly Rate: The cost per hour for the driver.
   * Fuel Tank Size (gallon): The fuel tank capacity.
   * Spare Tires: Number of spare tires available.
   * Truck Age (years): The age of the truck.

**Next Steps**

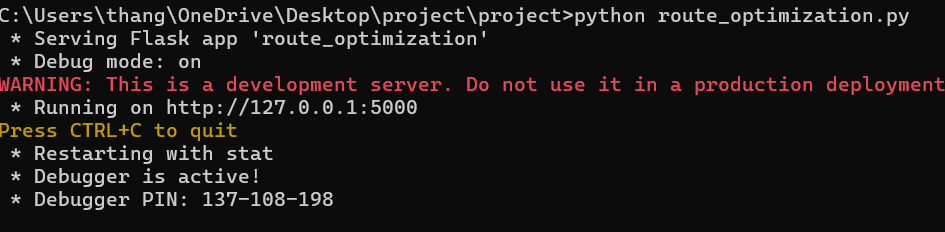
1. **Optimization Approach**:
   * Assign orders to trucks based on weight capacity.
   * Group orders by location to minimize route costs.
   * Plan truck routes to visit multiple locations efficiently.
   * Consider driver working limits (14-hour shifts, 10-hour breaks).
   * Minimize fuel and labour costs.
2. **Python Implementation**:
   * API to accept orders and return optimized delivery plans.
   * Route optimization using *Google OR-Tools* or *Graph Algorithms*.
   * Cost estimation based on fuel and driver wages.
3. **Deployment on Azure**:
   * API deployed using **Azure App Service**.
   * **Azure Functions** for automated scheduling.
   * **Azure Cosmos DB** for storing order and route data.
   * **Azure Machine Learning** for cost predictions (optional).
   * **Docker** for containerized deployment.
4. **Infrastructure Configuration**:
   * **YAML file** for Azure services.
   * **Docker file** for containerized API deployment.

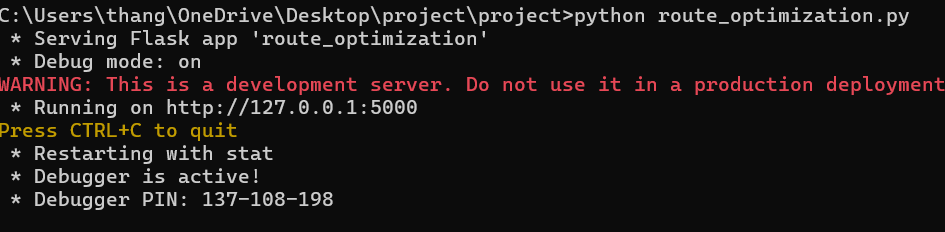
**Assumptions**

* All trucks start and end their journey at the Cincinnati warehouse.
* Fuel consumption is directly proportional to distance travelled.
* Driver costs are based on the time taken to complete the route.
* The road network and distances between cities are known and can be used for route planning.

Now, I'll implement the **Python script for route optimization and API deployment.**

|  |
| --- |
| **from** flask **import** Flask**,** request**,** jsonify  **import** pandas **as** pd  **import** networkx **as** nx  # Load Data  item\_info **=** pd**.**read\_csv**(**r"data\item\_info.csv"**)**  orders **=** pd**.**read\_csv**(**r"data\orders.csv"**)**  trucks **=** pd**.**read\_excel**(**r"data\trucks.xlsx"**,** sheet\_name**=**"Sheet1"**)**  # Initialize Flask App  app **=** Flask**(**\_\_name\_\_**)**  # Build Graph for Route Optimization  G **=** nx**.**Graph**()**  # Placeholder: Add nodes (cities) and edges (distances between them)  # In a real-world case, we'd use an API like Google Maps to get distances  # API Endpoint for Delivery Optimization  *@app***.**route**(**"/optimize\_routes"**,** methods**=[**"POST"**])**  **def** optimize\_routes**():**  data **=** request**.**get\_json**()**  orders **=** data**.**get**(**"orders"**,** **[])**  optimized\_routes **=** **[]**  **for** order **in** orders**:**  destination **=** order**[**"Destination"**]**  weight **=** **int(**order**[**"Number of Units"**])** **\*** **int(**item\_info**[**item\_info**[**"ItemId"**]** **==** order**[**"Item"**]][**"weight (pounds)"**].**values**[**0**])** # Convert to int  truck **=** trucks**.**loc**[**trucks**[**"Weight Capacity (kg)"**]** **>=** weight**].**iloc**[**0**]**    optimized\_routes**.**append**({**  "Truck ID"**:** **int(**truck**[**"Truck ID"**]),** # Convert int64 to int  "Destination"**:** destination**,**  "Weight"**:** **int(**weight**)** # Convert int64 to int  **})**    **return** jsonify**({**"optimized\_routes"**:** optimized\_routes**})**  # Run API  **if** \_\_name\_\_ **==** "\_\_main\_\_"**:**  app**.**run**(**debug**=True)** |





**API outcomes:**

