📄 Python Script Documentation: Order Enrichment and Sales Report Generator

🧩 Overview

This script processes customer orders, enriches them with geolocation data based on IP addresses using the ipinfo.io API, and generates a quarterly sales report for a given state and year. It uses:

- **SQLite** for persistent local storage.
- **Pandas** for data manipulation.
- **Requests** for API interaction.
- **Multiprocessing** for IP data enrichment (planned).
- **OpenPyXL** for Excel file generation.

📁 Input Files

- orders_file.csv: Contains raw order data (including sales info).
- ip_addresses.csv: Contains order numbers with IP addresses to be matched.

🗃️ Database Structure

- **1. `orders` table:** Stores enriched order details.

order_number TEXT PRIMARY KEY

date TEXT

city TEXT

state TEXT

Zip TEXT

$ sale TEXT

ip_address TEXT

- **2. ip_data table:** Stores IP address and corresponding geolocation data.

ip_address TEXT PRIMARY KEY

city TEXT

state TEXT

zip_code TEXT

Function Breakdown

✅ **create_tables()**

- Creates the required SQLite tables (orders, ip_data) if not already present.

✅ **alter_orders_table()**

- Ensures ip_address column exists in the orders table.

✅ **validate_ip(ip)**

- Validates and normalizes IP address strings using Python’s ipaddress module.

✅ **load_orders_data(file_path)**

- Loads order data from a CSV file into the orders table.
- Skips duplicate order_number values using INSERT OR IGNORE.

✅ **load_ip_data(file_path)**

- Loads unique, valid IP addresses from a CSV file into the ip_data table.
- Avoids inserting duplicates.

✅ **merge_ips_into_orders()**

- Associates IP addresses from ip_addresses.csv to the corresponding orders by order_number.

✅ **update_ip_data()**

- Fetches geolocation data in bulk from ipinfo.io only for Ips not yet enriched (where city IS NULL or blank).
- Designed to avoid redundant processing of already-known Ips.
- Parallel processing logic to be added for efficiency.

✅ **update_orders_table()**

- Updates the orders table with city, state, and Zip values from ip_data.
- Only updates rows where city is NULL or empty (avoiding unnecessary rewrites).

✅ **generate_sales_report(state, year)**

- Filters orders for a given state and year.
- Aggregates $ sale values by city and quarter.
- Creates a pivot table showing quarterly sales per city.
- Exports the data to an Excel file named “{state}_state_sales_report_{year}_generated.xlsx”.

▶️ main() Function Flow

1. Creates database tables.
2. Adds missing columns if required.
3. Loads order data from CSV.
4. Merges IPs into orders.
5. Loads and stores new IPs.
6. Updates geolocation data for unknown IPs.
7. Enriches orders using geolocation info.
8. Generates sales report for Ontario in 2024.

🛡️ Safeguards

- **Duplicate protection** via INSERT OR IGNORE.
- **Null checks** on IP and geolocation fields.
- **Data integrity** with validation of IP addresses.
- **Selective updates** to prevent overwriting enriched data.

📈 Output

- An Excel file with a summarized sales report per city and quarter.
    - Sheet name: Ontario_state_sales_report_2024_generated
    - Format: Cities as rows, Quarters as columns (Q1 to Q4), and totals.