Vendor Performance Data Analytics Project README

Contents

1 Introduction

This document provides a comprehensive overview of the Vendor Performance Data Analytics Project, detailing the functionality, requirements, and usage instructions for each component file. The project processes and analyzes vendor performance data for inventory management, including data ingestion, transformation, and visualization.

2 File: `get_vendor_summary.py`

2.1 Features

- **Data Aggregation**: Combines data from multiple tables (`vendor_invoice`, `purchases`, `purchase_prices`, and `sales`) to create a comprehensive vendor summary using SQL queries.
- **Data Cleaning**: Cleans the aggregated data by handling missing values, ensuring correct data types, and removing whitespace from categorical columns.
- **Derived Metrics**: Calculates key performance indicators such as `GrossProfit`, `ProfitMargin`, `StockTurnover`, and `SalesPurchaseRatio` for analysis.
- **Database Integration**: Ingests the cleaned summary data into a SQLite database table named `vendor_sales_summary`.
- **Logging**: Logs key operations (e.g., table creation, data cleaning, and ingestion) to a file (`logs/ingestion_db.log`) for debugging and monitoring.

2.2 Requirements

- **Python**: Version 3.6 or higher
- **Libraries**:
  - `pandas` (for data manipulation)
  - `sqlite3` (for database connectivity)
  - `logging` (for logging operations)
- **Dependencies**:
  - The `ingestion_db.py` module must be available in the same directory for the `ingest_db` function.
- **Database**:
  - A SQLite database named `inventory.db` with the following tables:
    - `vendor_invoice` (with `VendorNumber` and `freight` columns)
    - `purchases` (with `VendorNumber`, `VendorName`, `Brand`, `Description`, `Quantity`, and `Dollars` columns)
    - `purchase_prices` (with `Brand`, `PurchasePrice`, `volume`, and `price` columns)
    - `sales` (with `VendorNo`, `Brand`, `salesDollars`, `SalesPrice`, `SalesQuantity`, and `ExciseTax` columns)
- **File System**:
  - A `logs` directory to store the `ingestion_db.log` file.

2.3 Usage

1. **Ensure Dependencies**: Verify that `inventory.db` exists and contains the required tables. Ensure the `logs` directory exists.
2. **Run the Script**:

```
python get_vendor_summary.py
```

3. **Output**:
   - A table named `vendor_sales_summary` is created in `inventory.db` with the aggregated and cleaned data.
   - Logs are written to `logs/ingestion_db.log`, including the top rows of the summary and cleaned data.
4. **Notes**:
   - The script assumes the database connection is valid and the required tables are populated.
   - Errors (e.g., missing columns) are logged and raised as exceptions.

3 File: `ingestion_db.py`

3.1 Features

- **CSV Data Ingestion**: Loads CSV files from a specified directory and ingests them into a SQLite database as tables.
- **Database Connectivity**: Uses SQLAlchemy to connect to a SQLite database (`inventory.db`).
- **Logging**: Records ingestion progress and timing details in `logs/ingestion_db.log`.
- **Performance Tracking**: Measures and logs the total time taken for ingestion.

3.2 Requirements

- **Python**: Version 3.6 or higher
- **Libraries**:
  - `pandas` (for data manipulation)
  - `sqlalchemy` (for database connectivity)
  - `os` (for file system operations)
  - `logging` (for logging operations)
  - `time` (for performance tracking)
- **File System**:
  - A directory containing CSV files (e.g., `C:\PROJECT_2025\powerbi2025\Vendor Performance Data Analytics\dataset\data\data`).
  - A `logs` directory for the `ingestion_db.log` file.
- **Database**:
  - A SQLite database named `inventory.db`.

3.3 Usage

1. **Prepare Data**:
   - Place CSV files (e.g., `begin_inventory.csv`, `purchases.csv`, `purchase_prices.csv`, `sales.csv`, `vendor_invoice.csv`) in the specified directory.
2. **Run the Script**:

```
python ingestion_db.py
```

3. **Output**:
   - Each CSV file is ingested as a table in `inventory.db`, with the table name derived from the file name (without the `.csv` extension).
   - Logs are written to `logs/ingestion_db.log`, including file ingestion details and total time taken.
4. **Notes**:
   - The script overwrites existing tables in the database (`if_exists='replace'`).
   - Ensure the directory path in the script matches the location of your CSV files.

4 File: `Exploratory data analysis.ipynb`

4.1 Features

- **Data Ingestion**: Replicates the functionality of `ingestion_db.py` to load CSV files into `inventory.db`.
- **Vendor Summary Creation**: Replicates the functionality of `get_vendor_summary.py` to create and clean the `vendor_sales_summary` table.
- **Data Exploration**: Queries the `vendor_sales_summary` table to display its contents and saves it back to the database.
- **Data Shape Reporting**: Prints the shape of each ingested CSV file for verification.

4.2 Requirements

- **Python**: Version 3.6 or higher
- **Libraries**:
  - `pandas` (for data manipulation)
  - `sqlalchemy` (for database connectivity)
  - `sqlite3` (for database operations)
  - `logging` (for logging operations)
  - `time` (for performance tracking)
- **Dependencies**:
  - The `ingestion_db.py` module must be available for the `ingest_db` function.
- **File System**:
  - A directory containing CSV files (same as `ingestion_db.py`).

  - A `logs` directory for the `ingestion_db.log` file.
- **Database**:
  - A SQLite database named `inventory.db`.
- **Environment**:
  - Jupyter Notebook environment (e.g., JupyterLab or Jupyter Notebook).
  - Python kernel version 3.13.5 (as specified in the notebook metadata).

4.3 Usage

1. **Set Up Environment**:
   - Launch Jupyter Notebook:

     ```
     jupyter notebook
     ```

   - Open `Exploratory data analysis.ipynb`.
2. **Run Cells**:
   - Execute the cells sequentially to:
     (a) Ingest CSV files into `inventory.db`.
     (b) Create and clean the `vendor_sales_summary` table.
     (c) Query and display the `vendor_sales_summary` table.
3. **Output**:
   - CSV files are ingested into `inventory.db`.
   - The `vendor_sales_summary` table is created and saved.
   - Logs are written to `logs/ingestion_db.log`.
   - The notebook displays the shape of ingested files and the contents of `vendor_sales_summar`
4. **Notes**:
   - The notebook assumes the same directory structure as `ingestion_db.py`.
   - Errors (e.g., missing columns) are logged and raised.

5 File: `vendor_performance_analysis.ipynb`

5.1 Features

- **Data Visualization**: Creates a histogram with kernel density estimation (KDE) to compare the profit margins of top and low vendors.
- **Statistical Analysis**: Calculates 95% confidence intervals for profit margins of top and low vendors using `scipy.stats`.
- **Data Loading**: Loads the `vendor_sales_summary` table from `inventory.db` for analysis.
- **Plot Customization**: Uses `seaborn` and `matplotlib` to create a detailed plot with confidence intervals and mean lines.

5.2 Requirements

- **Python**: Version 3.6 or higher
- **Libraries**:
  - `pandas` (for data manipulation)
  - `matplotlib.pyplot` (for plotting)
  - `numpy` (for numerical operations)
  - `seaborn` (for enhanced visualizations)
  - `scipy.stats` (for statistical calculations)
  - `sqlite3` (for database connectivity)
  - `warnings` (to suppress warnings)
- **Database**:
  - A SQLite database named `inventory.db` with the `vendor_sales_summary` table (populated by `get_vendor_summary.py` or `Exploratory data analysis.ipynb`).
- **Environment**:
  - Jupyter Notebook environment with Python kernel version 3.13.5.

5.3 Usage

1. **Set Up Environment**:
   - Launch Jupyter Notebook:

     ```
     jupyter notebook
     ```

   - Open `vendor_performance_analysis.ipynb`.
2. **Prepare Data**:
   - Ensure the `vendor_sales_summary` table exists in `inventory.db`.
3. **Run Cells**:
   - Execute the cells to:
     (a) Load the `vendor_sales_summary` table.
     (b) Calculate confidence intervals for top and low vendors.
     (c) Generate a comparative histogram plot.
4. **Output**:
   - A plot titled "Confidence Interval Comparison: Top vs. Low Vendors (Profit Margin)" is displayed.
   - Confidence interval values and means are printed for top and low vendors.
5. **Notes**:
   - The notebook assumes `top_vendors` and `low_vendors` are defined (likely as subsets of `vendor_sales_summary` based on `ProfitMargin`).

- The plot is customized for clarity with legends, gridlines, and labeled confidence intervals.

6 General Notes

- **Database**: All files interact with a SQLite database named `inventory.db`. Ensure it is accessible and properly structured.
- **Logging**: Logs are stored in `logs/ingestion_db.log`. Check this file for debugging or monitoring ingestion and processing activities.
- **Directory Structure**: Update file paths in `ingestion_db.py` and `Exploratory data analysis.ipynb` to match your local environment.
- **Error Handling**: Scripts and notebooks include error handling for missing columns and log errors for traceability.