Dataset Information

This is a transactional data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

Data Available on the website : archive.ics.uci.edu

Data :

Variables Table OnlineRetail

Variable Name	Role	Type	Description
InvoiceNo	ID	Categorical	a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation
StockCode	ID	Categorical	a 5-digit integral number uniquely assigned to each distinct product
Description	Feature	Categorical	product name
Quantity	Feature	Integer	the quantities of each product (item) per transaction
InvoiceDate	Feature	Date	the day and time when each transaction was generated
UnitPrice	Feature	Continuous	product price per unit
CustomerID	Feature	Categorical	a 5-digit integral number uniquely assigned to each customer
Country	Feature	Categorical	the name of the country where each customer resides

Variables Table Supplier

Variable Name	Role	Type	Description
InvoiceNo	ID	Categorical	a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation
Supplier	ID	Categorical	A 4-character alphanumeric code that uniquely identifies a supplier. This code always starts with the letter 'F'.

Travail demandé :

Pour ce projet ETL, il est demandé d'utiliser la programmation orientée objet (POO) pour créer un projet Python. On va se concentrer sur le nettoyage, la gestion des transactions, des fournisseurs, et la transformation des données pour obtenir un

DataFrame enrichi et prêt pour l'analyse. Ce projet inclura des classes et méthodes permettant de structurer le code de manière maintenable et modulaire.

Classes et Méthodes à Créer

1. Classe DataCleaner

Cette classe sera responsable du nettoyage des données.

- **Attributs** :
    - df: Le DataFrame initial.
- **Méthodes** :
    - remove_duplicates(): Supprimer les lignes en double du DataFrame.
    - handle_missing_values(): Traiter les valeurs manquantes dans les colonnes critiques (CustomerID, Description, Quantity, etc.).
    - filter_valid_transactions(): Garder uniquement les transactions non annulées.

2. Classe TransactionProcessor

Cette classe gère la logique métier pour les transactions.

- **Attributs** :
    - df: Le DataFrame après nettoyage.
- **Méthodes** :
    - calculate_total_amount(): Calculer le montant total de chaque transaction (Quantity * UnitPrice) et ajouter une colonne TotalAmount.
    - group_by_country(): Regrouper les données par pays et calculer la somme totale des montants des transactions pour chaque pays.
    - aggregate_monthly_data(): Calculer des statistiques mensuelles, comme le montant total des ventes et le nombre de transactions.
    - calcul_stat_data():
      - Afficher le produit « **Description** » qui a rapporté le plus de gains en France,
      -Nous devons analyser l'heure à laquelle le nombre de transactions est le plus élevé. L'analyse se fera par tranches d'une heure, en regroupant les transactions pour identifier l'intervalle avec le plus grand volume.
    - aggregate_supplier_data(): besoin d'avoir les informations sur les fournisseurs (**fichier Supplier**) :
      - Agréger les résultats pour avoir le classement des fournisseurs selon le total du vente de leurs produits (**dans tous les pays**) => ne pas oublier d'éliminer les opérations annulées
      - Faire le même calcul pour l'année **2011** a « **United Kingdom** »

- aggregate_world_data(): Ajouter un fichier de mapping au projet permettant d'associer les pays présents dans le fichier à leurs continents :
  - Classer les continents selon les dépenses
  - Quel est le continent où il y a le plus d'opérations annulées ?

3. Classe ETLPipeline

Cette classe principale orchestre le processus ETL en appelant les méthodes des autres classes. Elle inclura également l'enregistrement final en parquet.

- **Attributs** :
  - df: Le DataFrame de départ.
- **Méthodes** :
  - run_pipeline(): Exécute le pipeline ETL complet, nettoie les données, applique les transformations, et execute les traitements par ordre.
  - save_as_parquet(path: String): Enregistre le DataFrame final sous forme de fichier parquet.

Critères d'acceptation :

1- Vous devez commenter le code avant chaque traitement ou un calcul a l'intérieur et en dehors de la fonction

2- Il faut afficher les logs pour faciliter la l'interprétation des résultats,

3 – Développer les Tests Unitaires pour toutes les fonctions développées (un script test pour chaque classe développée (sauf ETLPipeline) portant le nom : nomdelaclasse**Test)**

4 – Bien structurer le projet