UNIVERSITÉ ABDELMALEK ESSAÂDI
FACULTÉ DES SCIENCES ET TECHNIQUES
TANGER

Système ETL Automatisé avec Apache Airflow

Encadré par :
Pr. Yasyn EL YUSUFI

Réalisé par:
Said FRIKH
Anas ZENAGUI

Table de matière

1 .Introduction

Les entreprises modernes reposent sur l'analyse de données pour prendre des décisions stratégiques éclairées. Ce projet se concentre sur la création d'un pipeline ETL (Extract, Transform, Load) automatisé avec Apache Airflow. L'objectif est de collecter des données provenant de multiples sources, de les transformer pour les rendre exploitables et de les centraliser dans un entrepôt de données (data warehouse). Ces données pourront ensuite être utilisées pour générer des tableaux de bord et des rapports, facilitant ainsi la prise de décision.

Apache Airflow, un outil d'orchestration de workflows, est au cœur de ce projet. Il permet de planifier et d'automatiser chaque étape du processus ETL, garantissant que les données sont collectées, transformées et chargées efficacement dans le data warehouse.

2. Objectif du Projet

L'objectif principal du projet est de développer un pipeline de données qui :

1. **Ingestion des données** : Collecte des données provenant de sources variées telles que des APIs, des bases de données, et des fichiers locaux ou cloud.

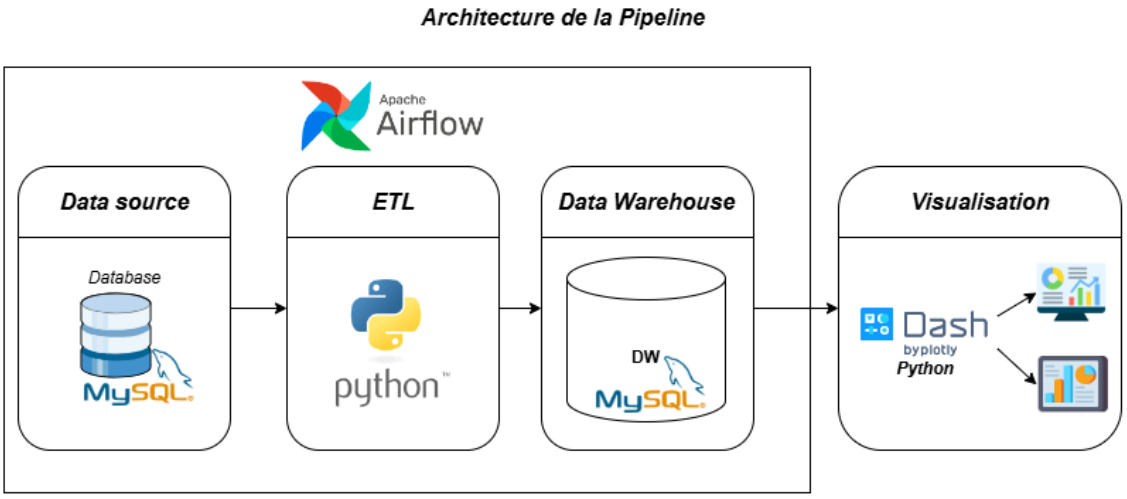
2. **Transformation des données** : Effectue des opérations de nettoyage, d'agrégation et de transformation des données pour les préparer aux analyses.

3. **Chargement des données** : Transfère les données traitées dans un entrepôt de données centralisé pour des analyses ultérieures.

4. **Visualisation des données** : Création d'un tableau de bord interactif pour représenter les indicateurs clés et générer des rapports basés sur les analyses de données.

En automatisant ces étapes, le projet vise à simplifier le processus d'analyse de données et à fournir une base fiable pour la prise de décision.

3. Architecture de la Pipeline



L'architecture de cette pipeline de données se compose de quatre étapes principales, illustrées dans le schéma :

1. **Source de Données**

Cette étape représente la base de données source où les données brutes sont collectées. Dans ce cas, la source est une base de données MySQL contenant plusieurs tables pour un site e-commerce :

- Table **customer** : Stocke les informations des clients, comme customer_id, first_name, last_name, email, etc.
- Table **product** : Contient des informations sur les produits, incluant product_id, product_name, product_description, product_price, et product_quantity.
- Table **order** : Représente les commandes, avec des champs tels que order_id, customer_id, order_date, et order_status. Cette table a une clé étrangère vers customer pour associer chaque commande à un client.
- Table **order-item** : Détaille les articles dans chaque commande, incluant order_item_id, order_id, product_id, product_price, et quantity. Elle possède des clés étrangères vers les tables order et product pour relier les articles aux commandes et aux produits.

Cette base de données source ecommercedb constitue la première étape où les données sont extraites pour être traitées.

2. ETL (Extraction, Transformation, Chargement)

Le processus ETL est responsable de l'extraction des données depuis la source, de leur transformation pour les structurer selon les besoins analytiques, et de leur chargement dans l'entrepôt de données.

Apache Airflow orchestre ce flux de travail ETL, en automatisant chaque étape. Des scripts Python sont utilisés pour appliquer les transformations nécessaires, par exemple, l'agrégation, le nettoyage ou l'enrichissement des données.

Airflow permet de planifier des tâches régulières et de surveiller l'ensemble du processus ETL, garantissant que les données dans l'entrepôt sont à jour.

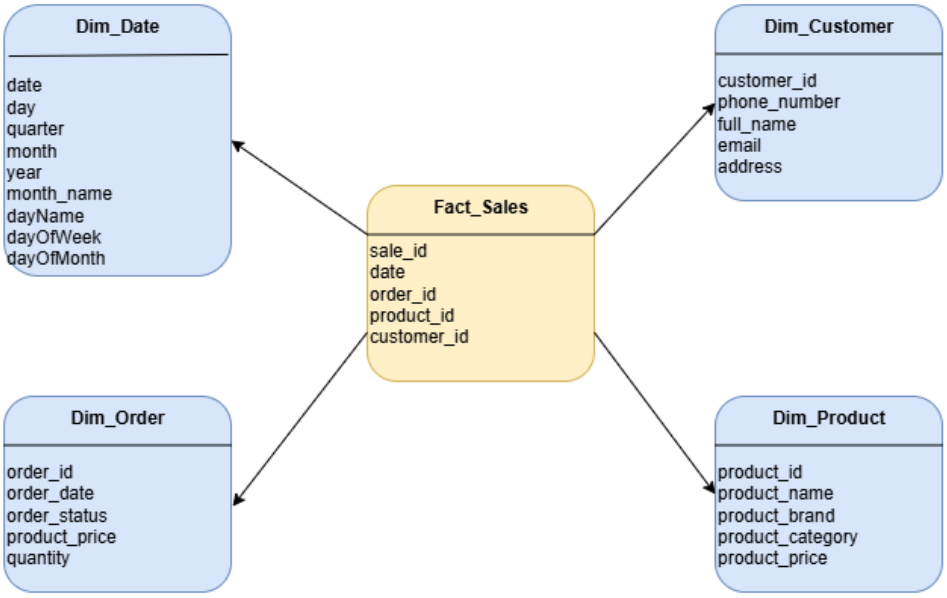
3. Entrepôt de Données (Data Warehouse)

Dans cette architecture, l'entrepôt de données est une base de données MySQL dédiée, qui stocke les données transformées prêtes pour l'analyse.

Le schéma de l'entrepôt de données suit une modélisation en étoile. La table des faits, appelée **sales**, contient les mesures quantitatives, telles que les montants des ventes et les quantités vendues.

Les tables de dimensions associées incluent :

- Dimension **date** : Stocke les informations temporelles.
- Dimension **product** : Contient les détails des produits.
- Dimension **customer** : Conserve les informations des clients.
- Dimension **order** : Enregistre les détails de chaque commande.



Ce schéma en étoile permet des requêtes rapides et facilite l'analyse des données pour obtenir des insights sur les ventes.

4. **Visualisation**

Cette dernière étape concerne la création de tableaux de bord et de visualisations pour présenter les données de manière compréhensible aux utilisateurs.

Dash by Plotly, un framework Python, est utilisé pour développer des tableaux de bord interactifs. Les données extraites de l'entrepôt sont présentées sous forme de graphiques, de diagrammes et de visualisations.

Ce tableau de bord permet aux utilisateurs d'explorer les tendances de vente, d'analyser le comportement des clients, et de prendre des décisions basées sur les données.

4. Installation et Configuration d'Apache Airflow

1. Préparation de l'environnement avec Docker Compose

Pour simplifier l'installation et la gestion des services d'Apache Airflow, nous allons utiliser **Docker Compose** avec un fichier **docker-compose.yaml** qui configure les différents services nécessaires : Airflow Webserver, Scheduler, Triggerer, ainsi qu'une base de données PostgreSQL pour stocker les informations d'Airflow.

Étapes :

1. **Télécharger** le fichier docker-compose.yaml
   Assurez-vous que le fichier docker-compose.yaml contient les services configurés pour Airflow, PostgreSQL, et Redis, comme dans le code fourni.

2. **Lancer les services Airflow**Exécutez la commande suivante pour démarrer tous les services en arrière-plan :

Cela lancera tous les conteneurs configurés pour Airflow, y compris le serveur web, le planificateur (scheduler), et le service de base de données PostgreSQL.

3. **Accéder à l'interface utilisateur d'Airflow**

Après avoir démarré les services, vous pouvez accéder à l'UI d'Airflow en ouvrant votre navigateur et en visitant l'URL suivante :

```
http://localhost:8085/dags
```

2. Création et Configuration d'un DAG

1. **Créer le fichier DAG**
   Créez un fichier appelé `data_transform_pipeline.py` dans le dossier `dags`, qui contient le code de votre flux de données Airflow.
   Exemple de code de base pour le fichier DAG :

```python
from airflow import DAG
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email_on_failure': False,
    'email_on_retry': False,
}

with DAG(
    'data_transform_pipeline',
    default_args=default_args,
    description='Un DAG pour transformer les données',
    schedule_interval='@daily',
    start_date=days_ago(1),
    catchup=False,
) as dag:

    start = DummyOperator(task_id='start')
    end = DummyOperator(task_id='end')

    start >> end
```

1. **Synchroniser le DAG dans l'interface d'Airflow**
   Le DAG sera automatiquement chargé et synchronisé dans l'UI d'Airflow. Vous pouvez visualiser et gérer ce DAG en accédant à la page `DAGs` à l'adresse `http://localhost:8085/dags`.

3. Vérification de la Configuration

Pour s'assurer que tout est configuré correctement :

- Accédez à la page des DAGs (`http://localhost:8085/dags`) pour vérifier que le DAG `data_transform_pipeline` est bien listé.
- Assurez-vous que le statut des services est actif et vérifiez les logs en cas de problèmes.

4. Exécution et Suivi du Pipeline ETL

1. Définition des tâches du pipeline

Le script `data_transform_pipeline.py` contient la configuration des tâches dans le pipeline ETL, qui inclut les étapes suivantes :

- **Extraction** des données depuis plusieurs sources : produits, clients, commandes, dates.
- **Transformation** des données extraites pour chaque entité.
- **Chargement** des données dans des tables dimensionnelles (dim).
- **Création de la table de faits** pour centraliser les informations de vente (`fact_sales`).

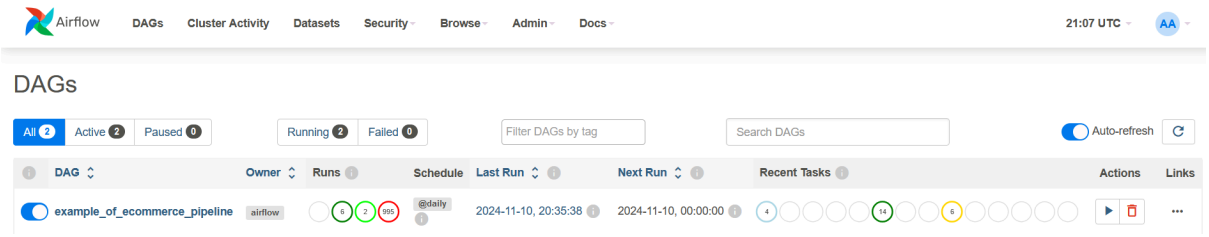
2. Voici l'ordre des tâches défini dans le pipeline

```
extract_customer_task >> transform_customer_task >>
load_customer_dim_task,
extract_product_task >> transform_product_task >> load_product_dim_task,
extract_order_task >> transform_order_task >> load_order_dim_task,
extract_date_task >> load_data_task
```

Toutes ces chaînes de tâches se rejoignent pour créer la table de faits avec la tâche `create_fact_sales`.

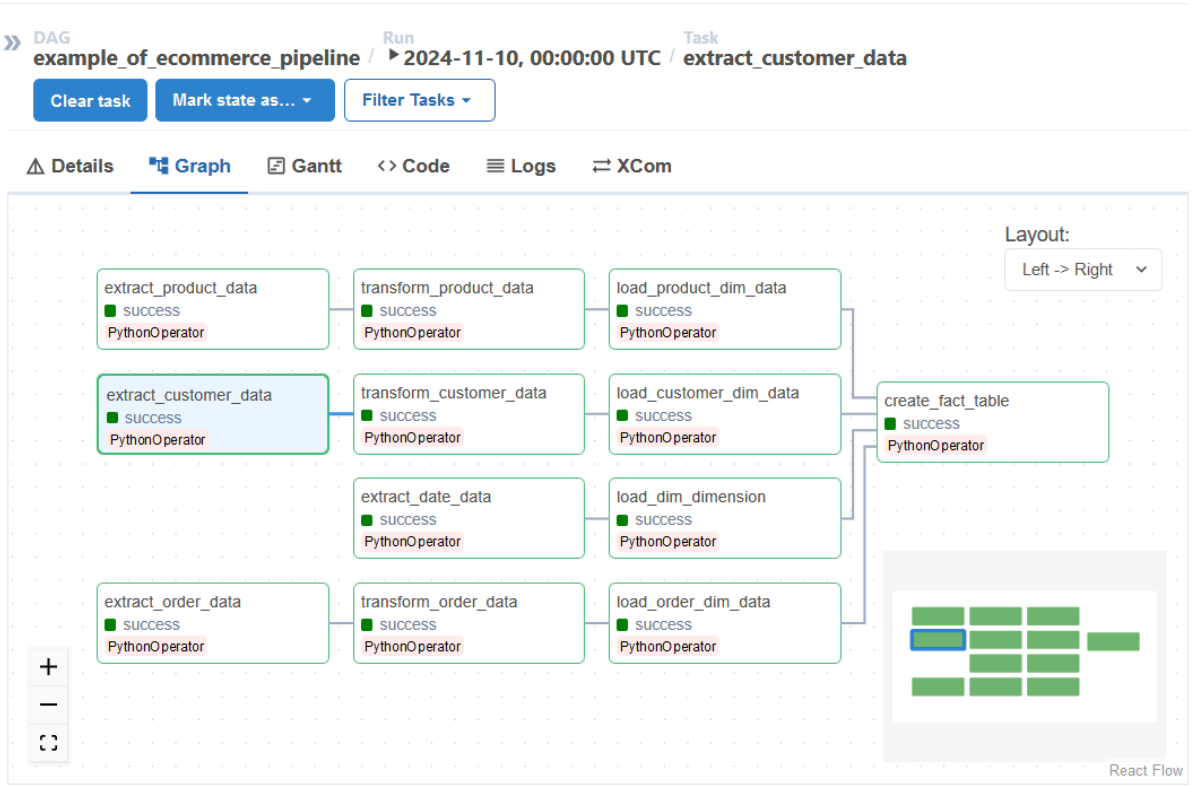
5. Vérification et Exécution dans l'interface utilisateur d'Airflow

Après avoir défini et ajouté le DAG dans le dossier `dags`, le DAG est apparu dans le tableau de bord d'Airflow (`http://localhost:8085/dags`). Vous pouvez visualiser la structure des tâches et suivre leur état.



6. Exécution du pipeline

En lançant le DAG depuis l'interface utilisateur, chaque tâche a été exécutée avec succès, comme le montre la capture d'écran ci-dessus. Les tâches ont toutes passé leur état à "success" (indiqué par les cases vertes), démontrant que le pipeline a fonctionné sans erreurs.



5. Tableau de Bord (Dashboard)

Objectif

Ce dashboard, créé avec la librairie Dash, permet d'analyser les tendances de ventes et d'interactions clients en utilisant des filtres interactifs. Il est connecté à une base de données MySQL pour extraire les données nécessaires, effectuer des transformations, et afficher les informations de manière visuelle et intuitive.

Connexion à la Base de Données

Le script `app.py` se connecte à une base de données MySQL locale en utilisant SQLAlchemy. La connexion est configurée pour extraire les données de plusieurs tables (produits, clients, commandes, dates), et les données sont traitées dans un dataframe Pandas pour simplifier les opérations de transformation et de filtrage.

Filtres Dynamiques

Le dashboard propose plusieurs filtres pour permettre aux utilisateurs de personnaliser l'analyse :

- **Produit**
- **Client**
- **Catégorie**
- **Marque**
- **Intervalle de Dates**

Ces filtres sont créés à partir des valeurs uniques des colonnes pertinentes, ce qui offre une expérience utilisateur fluide.

Interface Utilisateur

L'interface est divisée en deux sections principales :

1. **Section des Filtres** : Permet de sélectionner des options dans des menus déroulants et un sélecteur de plage de dates pour filtrer les données.
2. **Section des Graphiques** : Affiche plusieurs graphiques interactifs qui se mettent à jour dynamiquement en fonction des filtres.

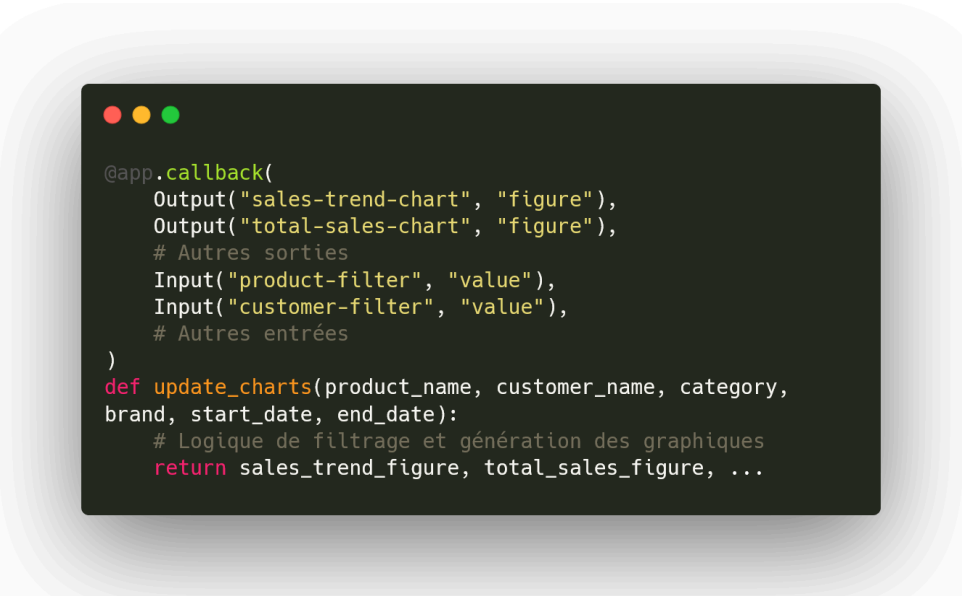
```
app.layout = html.Div(
    children=[
        # Filtres
        html.Div(...),
        # Graphiques
        html.Div(...),
    ]
)
```

Graphiques et Visualisations

Le dashboard comprend les visualisations suivantes :

- **Tendance Mensuelle des Ventes** : Un graphique en courbe montrant l'évolution des ventes mensuelles.
- **Total des Ventes** : Un graphique à barres empilées représentant le montant total des ventes par date.
- **Ventes par Catégorie et Trimestre** : Un graphique à barres empilées montrant les ventes totales par catégorie de produit et par trimestre.
- **Répartition des Produits par Marque** : Un graphique circulaire indiquant la part des ventes de chaque marque.
- **Répartition des Produits par Catégorie** : Un autre graphique circulaire pour la répartition des produits par catégorie.

Chaque graphique est mis à jour automatiquement lorsqu'un filtre est appliqué, grâce aux **callbacks** de Dash.



```
@app.callback(
    Output("sales-trend-chart", "figure"),
    Output("total-sales-chart", "figure"),
    # Autres sorties
    Input("product-filter", "value"),
    Input("customer-filter", "value"),
    # Autres entrées
)
def update_charts(product_name, customer_name, category,
brand, start_date, end_date):
    # Logique de filtrage et génération des graphiques
    return sales_trend_figure, total_sales_figure, ...
```

Exécution du Dashboard

En exécutant le script `app.py`, le dashboard est lancé et accessible via l'URL `http://localhost:8050/`. Cette interface permet une exploration visuelle et interactive des données de vente, en offrant une vue détaillée sur les tendances et les préférences des clients.

Image 1 : Vue d'ensemble et Filtres

- **Titre et Description** : Le tableau de bord est intitulé "Analytique des Ventes" avec un sous-titre indiquant "Analysez les tendances de vente et les interactions avec les clients".
- **Section de Filtres** : Juste en dessous du titre, une section de filtres permet aux utilisateurs de sélectionner divers critères pour affiner les données affichées dans les graphiques. Les filtres incluent :
    - Des menus déroulants pour différents attributs (ex : catégorie de produit, marque, ou période).
    - Un sélecteur de plage de dates permettant aux utilisateurs de définir une date de début et de fin spécifique pour les données qu'ils souhaitent visualiser.
- **Graphique en Barres : Ventes Totales par Catégorie de Produit et par Trimestre** :
    - Le graphique en dessous des filtres montre les ventes totales par catégorie de produit pour chaque trimestre.
    - Des catégories telles que **Téléphones**, **PC**, **Montres**, et **Casques** sont représentées par différentes couleurs.
    - L'axe Y représente le montant total des ventes, tandis que l'axe X représente les trimestres.
    - Ce graphique offre une comparaison visuelle rapide des performances de chaque catégorie de produit au fil des trimestres.

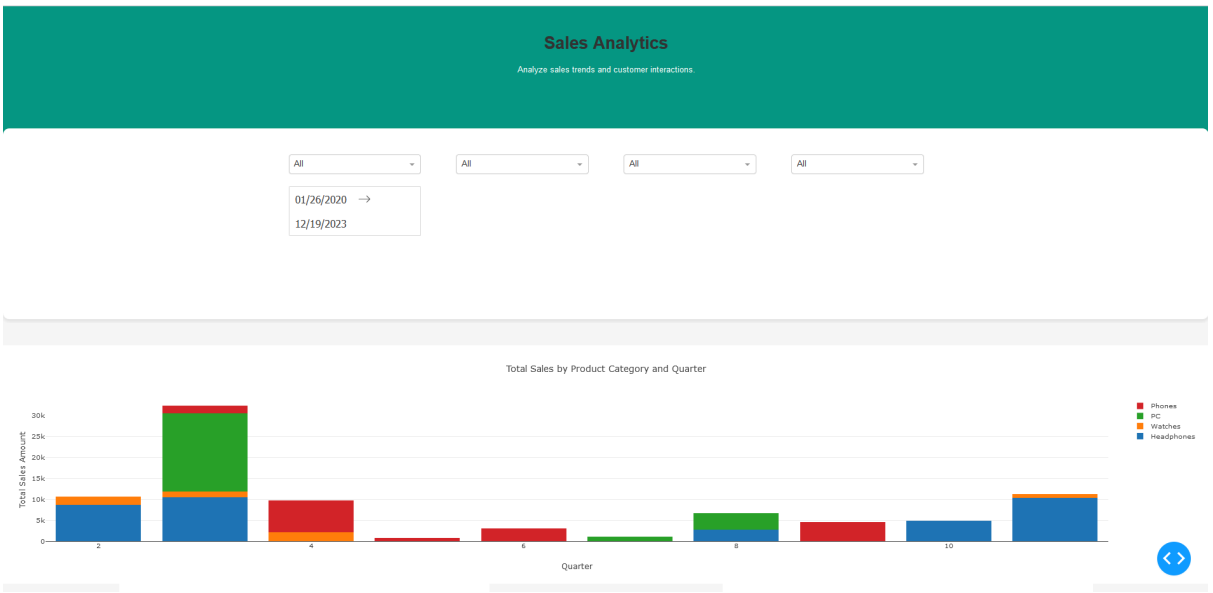
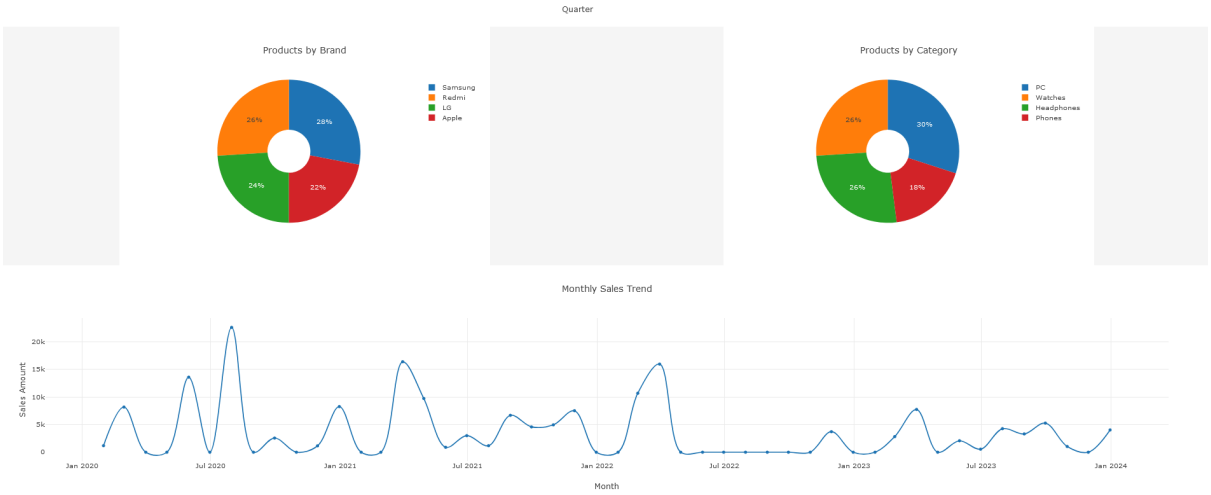
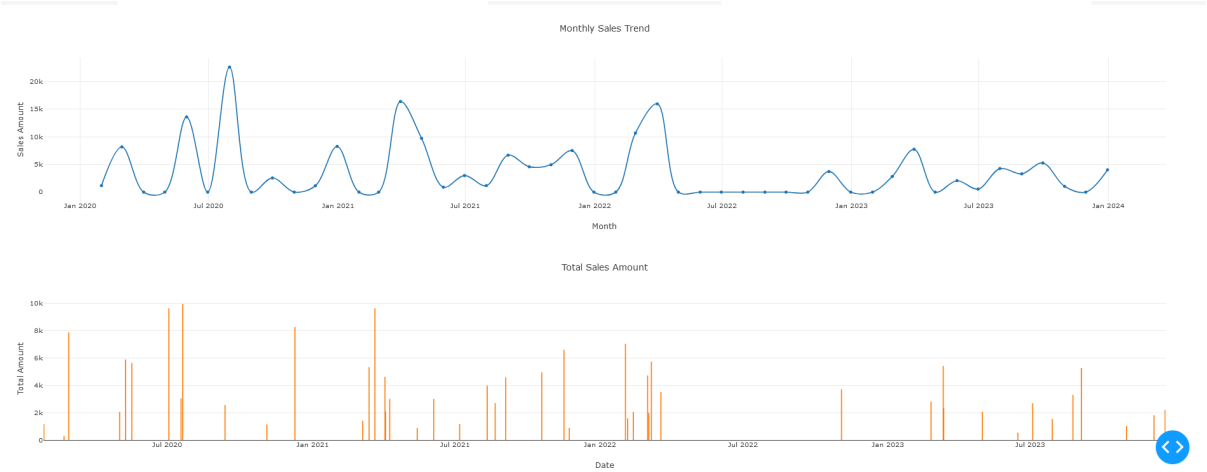


Image 2 : Analytique Détaillée avec Plusieurs Graphiques

Cette image contient trois principaux graphiques, disposés pour offrir une analyse plus approfondie des ventes selon différents aspects.

1. **Graphique en Donut : Produits par Marque** :
   - Ce graphique montre la répartition des ventes par marque.
   - Des marques comme **Samsung**, **Redmi**, **LG**, et **Apple** sont représentées par différentes couleurs, avec des pourcentages indiquant la contribution de chaque marque aux ventes totales.
   - Ce graphique permet aux utilisateurs de comprendre la part de marché de chaque marque sur la période ou selon les critères sélectionnés.
2. **Graphique en Donut : Produits par Catégorie** :
   - Similaire au graphique des marques, celui-ci montre la répartition des ventes par catégorie de produit.
   - Des catégories comme **PC**, **Montres**, **Casques**, et **Téléphones** sont mises en avant, chacune avec une couleur distincte et un pourcentage.
   - Ce graphique aide les utilisateurs à saisir rapidement quelles catégories de produits sont les plus populaires ou génèrent le plus de revenus.
3. **Graphique en Ligne : Tendance des Ventes Mensuelles** :
   - Ce graphique en ligne affiche les tendances des ventes au fil du temps, avec les **Mois** sur l'axe X et le **Montant des Ventes** sur l'axe Y.
   - Il représente les fluctuations mensuelles des ventes, avec des pics et des creux correspondant aux périodes de ventes élevées ou faibles.
   - Ce graphique fournit des informations sur les tendances saisonnières, aidant les utilisateurs à comprendre comment les ventes évoluent au fil du temps et à identifier les périodes de hausse ou de baisse des ventes.





6. Conclusion

En conclusion, ce projet a permis de mettre en place une architecture de pipeline de données complète, de l'extraction des données brutes à leur visualisation. L'utilisation d'Apache Airflow a automatisé le processus ETL, assurant des données à jour dans l'entrepôt, structuré en schéma étoile pour des requêtes rapides et efficaces. Avec Dash, nous avons créé un tableau de bord interactif, permettant aux utilisateurs d'explorer les données et de prendre des décisions éclairées. Ce pipeline démontre l'efficacité d'une architecture bien pensée et l'utilité des outils modernes pour transformer les données en informations exploitables.