BourseTrack : Suivi et Analyse des Données Financières et Actualités du Marché

I - Description du sujet

Dans le domaine des marchés financiers, la collecte et le traitement des données boursières, ainsi que la compréhension des actualités qui influencent les entreprises, sont essentiels pour toute analyse des tendances économiques. « **BourseTrack** » est une plateforme conçue pour centraliser, structurer et afficher des informations financières pertinentes, en s'appuyant notamment sur des APIs .

L'objectif principal du projet est de développer une **page web simple et intuitive** permettant de suivre les performances financières des grandes entreprises sur uniquement **les 100 derniers jours ouvrés**. Les données collectées incluent les prix moyens d'ouverture et de fermeture, les volumes d'échange, ainsi que d'autres indicateurs tels que la volatilité moyenne, les tendances globales et les rendements moyens journaliers. Ces informations sont enrichies par des actualités pertinentes concernant les entreprises suivies, offrant ainsi un contexte complémentaire pour l'analyse.

Le projet se concentre sur le suivi de grandes entreprises comme Apple, Amazon, Tesla et Microsoft, ... avec la possibilité pour les utilisateurs d'examiner les fluctuations de leurs données financières et les actualités associées. Par exemple, les tendances boursières positives ou négatives sont mises en avant pour chaque entreprise, permettant une compréhension rapide de leur performance.

Techniquement, le projet repose sur l'utilisation de Python et l'approche orientée objet pour l'extraction et la transformation des données via des API, ainsi que sur des outils de visualisation pour présenter les informations sous forme de graphiques et de tableaux clairs. Les utilisateurs peuvent ainsi accéder facilement à des données synthétiques, prêtes à être analysées.

II - APIs à utiliser

Dans le cadre du projet « **BourseTrack** », des APIs essentielles seront utilisées pour collecter les données financières et les actualités nécessaires. Tout d'abord, l'API **Stock Price de Alpha Vantage** sera utilisée pour obtenir des informations détaillées sur les prix des actions. Cette API est accessible via l'URL suivante : https://www.alphavantage.co/. Ensuite, pour enrichir l'analyse avec les actualités pertinentes ayant un impact sur le marché, l'API News API sera exploitée, disponible à l'adresse : https://newsapi.org.

Remarque importante : Attention, Ces APIs sont soumises à des limitations concernant le nombre de requêtes autorisées dans un certain laps de temps. Chaque API nécessite une clé d'accès individuelle, qui devra être obtenue et configurée avant de pouvoir être utilisée.

III - Étapes du projet

→ Étape N°1 : se familiariser avec l'utilisation des APIs ci dessus.

Avant de débuter la collecte des données via les deux APIs mentionnées, il est essentiel d'identifier le symbole boursier de l'entreprise concernée. Un **symbole boursier** est un code unique attribué à chaque entreprise cotée en bourse, qui sert à l'identifier sur les marchés financiers. Par exemple, le symbole boursier de **Tesla, Inc**. est **TSLA**, celui de **Apple Inc**. est **AAPL**, et celui de **Microsoft Corporation** est **MSFT**. Ces symboles servent à interroger les APIs pour récupérer les données financières et les actualités propres à chaque entreprise.

Pour explorer les tendances des actions des entreprises et leurs actualités, le site https://www.tradingview.com peut être consulté. Ce site fournit une vue d'ensemble des performances des actions et des nouvelles associées sur le marché financier.

Dans ce projet, nous nous intéressons aux entreprises dont les informations sont organisées sous forme de dictionnaire Python. Les clés de ce dictionnaire correspondent aux symboles boursiers des entreprises, et chaque symbole est associé à des informations telles que le nom de l'entreprise et son secteur d'activité :

```
NOMS_ENTREPRISES = {
    "TSLA": {"nom": "Tesla, Inc.", "secteur": "Automobile"},
    "F": {"nom": "Ford Motor Company", "secteur": "Automobile"},
    "TM": {"nom": "Toyota Motor Corp.", "secteur": "Automobile"},
    "AMZN": {"nom": "Amazon.com, Inc.", "secteur": "Commerce électronique"},
    "BABA": {"nom": "Alibaba Group Holding", "secteur": "Commerce
    électronique"},
    "AAPL": {"nom": "Apple Inc.", "secteur": "Technologie"},
    "MSFT": {"nom": "Microsoft Corporation", "secteur": "Technologie"}
}
```

Ensuite, afin de découvrir et de vous familiariser avec les deux APIs :

→ **Consulter la documentation des APIs** qui fournit des informations sur les points de terminaison (endpoints) disponibles, la configuration des requêtes, les paramètres nécessaires, ainsi que des exemples d'utilisation.

→ **Obtenir la clé API et configurer les paramètres** : Chaque API requiert une clé d'accès unique. Une fois cette clé obtenue, vous devez préparer un dictionnaire de paramètres pour chaque requête.

→ Pour débuter, vous pouvez utiliser le symbole TSLA (Tesla) comme exemple afin de tester les appels aux APIs. Voici les informations à collecter pour cette entreprise :

- **Données financières des 100 derniers jours ouvrés** : Récupérer des informations telles que le prix d'ouverture, le prix de clôture, le prix le plus haut, le prix le plus bas, le volume des échanges (nombre d'actions échangées sur une journée), et la date.
- **Actualités** : Collecter les dix dernières actualités concernant Tesla, incluant les champs suivants : le titre, la date de publication, l'URL, l'auteur, et une description.

Une fois les informations pour Tesla récupérées avec succès, vous pouvez tester le processus avec une autre entreprise, comme Apple (AAPL) ou Microsoft (MSFT). L'objectif est de répéter ces étapes pour chaque entreprise figurant dans le dictionnaire **`NOMS_ENTREPRISES`**.

À ce stade, vous pouvez réaliser une classe nommée **`GestionnaireAPI`** qui permet de configurer une API et de récupérer les données sous format JSON.

→ Étape N°2 : Sauvegarde et Lecture dans un fichier CSV

Étant donné que le nombre de requêtes aux API est limité, il est essentiel de sauvegarder localement les données récupérées (qu'il s'agisse de données financières ou d'actualités). Cela permet de réduire le nombre d'appels aux API en réutilisant les données déjà collectées. Pour chaque entreprise, les informations seront stockées dans des fichiers CSV distincts. Si les données sont déjà enregistrées dans un fichier CSV, elles peuvent être consultées directement, évitant ainsi un nouvel appel à l'API et respectant les limites de requêtes.

Pour chaque entreprise, deux fichiers CSV distincts seront générés : un fichier pour les données financières (**`symbole_financial.csv`**, par exemple **`TSLA_financial.csv`**) et un autre pour les actualités (**`symbole_news.csv`**, par exemple **`TSLA_news.csv`**).

À ce stade, vous pouvez réaliser une classe nommée **`GestionnaireFichier`** qui permet de sauvegarder et de lire depuis des fichiers CSV.

→ Étape N°3 : Réalisation du modèle relationnel de données et des classes métiers

Remarque : La base de données utilisée est SQLite.

Dans ce projet, il est important d'organiser vos données et votre code de manière logique pour simplifier le développement et la maintenance. Deux approches peuvent être envisagées : l'approche simplifiée basée sur un gestionnaire unique pour la base de données, ou une approche avancée utilisant des classes métiers.

→ **Réalisation d'un MLD : Avant de choisir une approche, la première étape incontournable est la réalisation d'un modèle relationnel de données (MLD).** Pour ce faire, il faut analyser les données collectées et à identifier les entités principales, leurs attributs, et les relations entre elles. Voici un exemple de MLD contenant deux tables **`FILMS`** et **`GENRES`** reliés par une relation **`un-plusieurs`** :

```
FILMS(num_film: INTEGER,titre: VARCHAR(80), #num_genre: INTEGER, annee_sortie: INTEGER)

GENRES(num_genre: INTEGER, nom_genre: VARCHAR(50))
```

Les clés primaires sont soulignés et les clés étrangères sont préfixées par **`#`**.

→ **Utilisation d'un gestionnaire unique pour la base de données (`GestionnaireBDD`) :** Pour tous des groupes, nous recommandons une approche simplifiée basée sur une classe unique dédiée à la gestion de la base de données, appelée **`GestionnaireBDD`**. Cette classe gère toutes les interactions avec la base de données, telles que l'établissement et la fermeture de la connexion, la création des tables et les opérations d'insertion ou de récupération des données. **Attention, il faut créer une seule instance de connexion (singleton)**.

Cette approche a l'avantage d'être simple, rapide à mettre en œuvre et flexible, car elle peut évoluer si vous décidez d'ajouter d'autres fonctionnalités. Les données collectées peuvent être manipulées à l'aide de structures simples comme des dictionnaires ou des listes, sans recourir à des classes métiers spécifiques pour représenter les entités (par exemple, **`Entreprise`**, **`Action`**, etc.).

Pour les plus avancés : Intégration de classes métiers

Pour les groupes les plus avancés ou ceux qui souhaitent approfondir leurs compétences, une approche orientée objet basée sur des classes métiers peut être envisagée. Dans cette approche, chaque table du MLD est représentée par une classe dédiée, avec ses attributs et méthodes correspondants. Par exemple, une

classe **`Entreprise`** avec des attributs comme **`nom`**, **`secteur`**, **`symbole`** et une méthode **`lister_entreprises()`** pour récupérer la liste des entreprises, etc.

Ces classes seraient ensuite connectées à la base de données via une classe comme **`GestionnaireBDD`**. Les avantages de cette approche sont : une meilleure organisation où les données et les fonctionnalités associées à chaque entité sont regroupées, ce qui améliore la lisibilité et la maintenabilité du code, l'encapsulation car les classes métiers permettent de mieux protéger et structurer les données, et la modularité permettant d'ajouter de nouvelles fonctionnalités ou modifier des entités sans affecter tout le projet.

Conseil : pour résoudre un problème complexe, procéder toujours par « diviser pour mieux régner », c'est à dire, décomposer le problème complexe en des sous-problèmes simples à résoudre. Proposer pour chaque sous-problème, une sous-solution. Finalement, regrouper les sous-solutions pour trouver la solution finale au problème.

→ Étape N°4 : Génération d'une page HTML dynamique pour afficher les résultats

Au lieu d'utiliser une interface console classique avec des menus, vous allez concevoir une solution où les données financières et les actualités sont collectées, traitées et affichées directement dans une page HTML générée dynamiquement. À ce stade, il est recommandé de créer un fichier nommé « **`gestionnaire_app.py`** » qui servira à orchestrer et gérer les différents traitements de votre application.

Voici comment cette étape est structurée:

1- Tracer et enregistrer le volume total des échanges par entreprise sous forme de graphique circulaire (pie chart) :

→ À partir des données stockées dans la base de données SQLite, vous calculez l**e volume total des échanges pour chaque entreprise.**
→Le graphique circulaire est généré avec la fonction **`afficher_volume_total()`** et enregistré sous forme d'image (fichier JPG) pour être intégré dans la page HTML. Utiliser les modules **`matplotlib`** et **`seaborn`** pour générer le graphique.

2- Calculer les indicateurs financiers pour chaque entreprise :

→Les indicateurs sont générés à partir des données stockées dans la base de données SQLite. Ils offrent une vue synthétique des performances financières de

chaque entreprise et sont présentés sous forme de tableau dans la page HTML. Les indicateurs calculés sont les suivants :

-**Prix moyen d'ouverture et prix moyen de clôture** : moyenne des prix d'ouverture /de fermeture des actions sur les 100 derniers jours.

Un prix d'ouverture élevé peut indiquer que l'entreprise suscite beaucoup d'intérêt dès le début de la journée de trading, tandis qu'un prix de fermeture reflète la valeur perçue de l'entreprise à la fin des sessions de trading. La comparaison des deux permet d'identifier une tendance générale (hausse ou baisse).

- **Volatilité moyenne** : moyenne des écarts entre les prix haut et bas des actions sur les 100 derniers jours.

Une volatilité élevée indique un risque ou un potentiel de profit important pour les traders, tandis qu'une volatilité faible signifie une action plus stable.

-**Volume moyen** : Moyenne des volumes d'échange des actions.

Il reflète l'intérêt des investisseurs pour l'action. Un volume élevé peut indiquer une liquidité importante.

-**Tendance globale**: différence entre le prix moyen de clôture et le prix moyen d'ouverture.

Une tendance positive (clôture > ouverture) indique que l'action a tendance à augmenter dans la journée, tandis qu'une tendance négative (clôture < ouverture) peut indiquer une pression à la vente.

- **Rendement moyen journalier** : moyenne des variations journalières en pourcentage. Sa formule est : `100 * (Close – Open)/Open`

Il permet de comparer la performance relative entre différentes entreprises.

Voici un aperçu des résultats à obtenir :

Performances Boursières des Grandes Entreprises

Entreprise	PrixMoyenOuverture	PrixMoyenFermeture	VolatiliteMoyenne	VolumeMoyen	TendanceGlobale	RendementMoyenJournalier
Alibaba Group Holding	88.644	88.714	1.99	17531052.61	0.07	0.099
Amazon.com, Inc.	186.592	186.433	4.171	39852324.58	-0.159	-0.074
Apple Inc.	224.813	225.123	4.227	50165864.86	0.31	0.154
Ford Motor Company	11.223	11.225	0.276	57214978.49	0.001	0.015
Microsoft Corporation	425.062	424.523	7.413	20016322.29	-0.539	-0.122
Tesla, Inc.	240.306	240.624	11.061	97121563.77	0.317	0.174
Toyota Motor Corp.	182.036	181.909	2.309	354503.89	-0.127	-0.059

Intégrer un style CSS pour différencier visuellement les entreprises selon leur tendance : par exemple, une couleur pour une tendance positive (hausse) et une autre pour une tendance négative (baisse).

3- Récupérer les actualités pertinentes pour chaque entreprise :

→ Les actualités collectées via l'API (et stockées dans la base de données) sont récupérées pour être affichées dans la page HTML. **On affiche les actualités des entreprises ayant une tendance ≥ 0.15 ou ≤ -0.15 uniquement**.

Vous pouvez créer une fonction **`generer_html()`**, qui inclut: une image du graphique circulaire représentant les volumes d'échange, un tableau contenant les indicateurs financiers, et les actualités pertinentes.

IV - Modalités du rendu de projet et de Soutenance orale

Voici les principales modalités à respecter :

1- Fichiers Python contenant les **Docstrings et commentaires pertinents** :
Toutes les fonctions, classes, et méthodes doivent inclure des ***docstrings*** détaillées.. Une chaîne documentaire doit expliquer ce que fait la fonction/classe/méthode, quels sont ses arguments, et ce qu'elle renvoie. Cela aide le correcteur et les autres membres de votre équipe à comprendre et à utiliser vos fonctions. En plus des *docstrings*, vous devriez ajouter des commentaires pertinents dans le code pour expliquer les parties complexes ou importantes de votre implémentation.

2- Qualité du code :
Vous devez éviter la duplication de code autant que possible. Si plusieurs fonctions ont besoin de faire la même chose, envisager de créer une fonction utilitaire pour éviter de répéter le code. Utiliser des noms de variables/classes/méthodes clairs et explicites pour que le code soit facile à comprendre. Éviter les noms de variables obscurs ou trop courts.

3- Dépôt sur Ecampus (jusqu'au 19/12 à 18h)
Une archive compressée : **`NomW_NomX_NomY.zip`** (**`W`**, **`X`** et **`Y`** sont les noms des étudiants) contenant les :
→ **fichiers Python y compris docstrings et commentaires.**
→ **fichier de présentation (PDF)**
→ **Poster (PDF)**
Le groupe dont le travail est non soumis pour évaluation aura 0 comme note finale de projet. Tout retard ou plagiat (codes identiques entre groupes) entraînera des pénalités sur la note finale.

4- Soutenance orale (09/12) :

Chaque groupe disposera d'environ 30-40 minutes pour effectuer une présentation orale de son projet. Chacun des étudiants présentera son travail pendant une durée de 5 minutes, en mettant en évidence les parties du projet qu'il a développées, les défis/obstacles qu'il a rencontrés, les solutions adoptées, les compétences qu'il a acquises et les possibilités d'amélioration. Ensuite, chaque étudiant sera interrogé individuellement, en mettant l'accent particulièrement sur la participation individuelle de chaque membre au projet. Il est important de noter que pendant la présentation, l'examinateur n'interviendra pas. Les 5 minutes allouées à chaque présentateur étant limitées, il est essentiel de bien structurer son discours pour mettre en valeur son implication personnelle dans le projet. Après les présentations individuelles, l'examinateur disposera de 5 minutes pour poser des questions et évaluer le travail présenté par le groupe.