# Livrable du Projet : Système ETL pour  l’Analyse des Données NBA

## 1. Introduction

Ce document présente le livrable du  projet de système ETL (Extract,  Transform, Load) conçu pour collecter,  traiter et préparer des données relatives  à la NBA (National Basketball  Association). L’objectif principal de ce  projet est de créer une infrastructure  robuste et automatisée capable d’ingérer  des données provenant de diverses  sources, de les nettoyer, de les  transformer et de les rendre disponibles  pour des analyses approfondies. Ce  système permettra aux utilisateurs  d’obtenir des informations précieuses sur  les performances des équipes, les  statistiques des joueurs, les résultats  des matchs et d’autres aspects pertinents  du basketball professionnel. L’importance  d’un tel système réside dans sa capacité  à fournir des données fiables et  structurées, essentielles pour la prise  de décision basée sur les données, que ce  soit pour des analyses sportives, des  stratégies de paris ou des recherches  académiques.

## 2. Présentation du Projet

Le projet vise à construire un pipeline  ETL complet pour les données NBA. Ce  pipeline est conçu pour être modulaire et  extensible, permettant l’intégration de  nouvelles sources de données et  l’adaptation à des exigences d’analyse  évolutives. Le processus ETL se décompose  en trois phases principales :

***Extraction (Extract)****:  Récupération des données brutes à partir  de sources hétérogènes telles que des  APIs, des pages web (web scraping), des  fichiers plats (CSV, JSON), des bases de  données relationnelles (SQL) et des  systèmes de Big Data.*    **Transformation (Transform)** :  Nettoyage, validation, enrichissement et  restructuration des données extraites  pour les rendre cohérentes et prêtes à  l’analyse. Cette phase inclut la gestion  des valeurs manquantes, la  standardisation des formats et  l’agrégation des informations. \*   **Chargement (Load)** : Stockage des  données transformées dans une base de  données cible optimisée pour l’analyse,  telle que PostgreSQL pour les données  structurées et MongoDB pour les données  semi-structurées ou non structurées.

Le système est développé en Python, un  langage largement reconnu pour sa  flexibilité, sa richesse en bibliothèques  dédiées à la manipulation de données et  sa facilité d’intégration avec diverses  technologies de bases de données et de  Big Data. L’automatisation du pipeline  garantit que les données sont toujours à  jour et prêtes à être consommées par des  outils de visualisation ou des modèles  d’apprentissage automatique.

## 3. Spécifications Techniques

Le projet s’appuie sur un ensemble de  technologies et de bibliothèques Python  pour assurer l’efficacité et la  robustesse du pipeline ETL :

***Langage de Programmation****: Python  3.x est le langage principal utilisé pour  le développement de toutes les  composantes du système ETL. Sa vaste  écosystème de bibliothèques est un atout  majeur.*    **Gestion des Requêtes HTTP** : La  bibliothèque requests est utilisée pour  effectuer des requêtes HTTP vers les APIs  et les serveurs web, permettant  l’extraction de données structurées  (JSON, XML) et de pages HTML. ***Web Scraping****: BeautifulSoup  est employée pour le parsing (analyse  syntaxique) des documents HTML et XML,  facilitant l’extraction de données  spécifiques à partir de pages web non  structurées. Elle est souvent utilisée en  conjonction avec requests.*    **Manipulation et Analyse de  Données** : pandas est la bibliothèque  fondamentale pour la manipulation de  données. Elle offre des structures de  données performantes comme les  DataFrames, idéales pour le nettoyage, la  transformation et l’agrégation de grands  ensembles de données tabulaires. ***Bases de Données Relationnelles****:  sqlite3 est utilisée pour interagir  avec des bases de données SQLite locales,  servant de base de données temporaire ou  de source de données relationnelle pour  l’extraction. Pour des bases de données  plus robustes comme PostgreSQL, des  bibliothèques comme psycopg2 ou  SQLAlchemy seraient utilisées.*    **Bases de Données NoSQL** :  pymongo est la bibliothèque Python  officielle pour interagir avec MongoDB,  une base de données NoSQL orientée  document, utilisée pour stocker des  données semi-structurées ou des logs. ***Orchestration****: Bien que non  explicitement mentionné, un système  d’orchestration comme Apache Airflow ou  un simple script main.py avec des  appels séquentiels peut être utilisé pour  gérer l’exécution des différentes étapes  du pipeline ETL.*    **Gestion des Dépendances** :  requirements.txt liste toutes les  bibliothèques Python nécessaires au  projet, assurant un environnement de  développement et de déploiement cohérent.

Ces technologies, combinées, forment une  architecture flexible et puissante,  capable de gérer des volumes de données  variés et de s’adapter aux besoins  spécifiques du projet.

## 4. Structure du Projet

Cette organisation permet une séparation claire des responsabilités : les scripts extract\_\*.py gèrent l’extraction, les scripts load\_\*.py gèrent le chargement, main.py orchestre le tout, et models.py centralise la structure des données. Le dossier data/raw est crucial pour stocker les données brutes avant toute transformation, ce qui est essentiel pour la traçabilité et le débogage.

## 5. Extraction des Données

La phase d’extraction est la première étape du pipeline ETL, où les données brutes sont collectées à partir de diverses sources. Chaque source nécessite une approche spécifique :

### 5.1. Extraction des Données via API (extract\_api.py)

Ce module est responsable de la récupération des données à partir d’APIs externes. Il utilise la bibliothèque requests pour envoyer des requêtes HTTP (GET, POST) aux endpoints des APIs. Les données sont généralement reçues au format JSON ou XML, puis parsées et converties en DataFrames pandas pour faciliter les étapes suivantes. Des exemples d’APIs pertinentes pour les données NBA pourraient inclure des APIs de statistiques sportives ou des APIs de données de matchs. Le script gère l’authentification (clés API), la pagination et la gestion des erreurs de requête.

### 5.2. Extraction des Données des Pages Web (Web Scraping) (extract\_web.py)

Ce module se concentre sur l’extraction de données à partir de pages web HTML. Il utilise requests pour télécharger le contenu des pages et BeautifulSoup pour analyser la structure HTML. Des sélecteurs CSS ou des expressions XPath sont employés pour localiser et extraire les éléments de données spécifiques (tableaux, listes, texte). Cette méthode est particulièrement utile lorsque les données ne sont pas disponibles via une API structurée. Une attention particulière est portée à la robustesse du scraping pour gérer les changements de structure des pages web et éviter le blocage par les sites.

### 5.3. Extraction des Données des Fichiers (extract\_csv.py)

Ce module gère l’ingestion de données à partir de fichiers locaux ou distants, tels que des fichiers CSV, JSON, Excel ou Parquet. La bibliothèque pandas est largement utilisée ici pour lire ces formats de fichiers directement dans des DataFrames. Le script doit gérer les différents délimiteurs, les encodages de caractères, les en-têtes et les types de données pour assurer une lecture correcte des fichiers. Il peut également inclure des logiques pour l’extraction de fichiers compressés ou l’itération sur plusieurs fichiers dans un répertoire.

### 5.4. Extraction des Données des Bases de Données Relationnelles (extract\_sql.py)

Ce module est conçu pour se connecter à des bases de données relationnelles (ex: SQLite, MySQL, PostgreSQL) et extraire des données via des requêtes SQL. Il utilise des connecteurs de bases de données Python (comme sqlite3 ou psycopg2) et pandas.read\_sql\_query pour exécuter des requêtes et charger les résultats directement dans des DataFrames. Le script peut contenir des requêtes complexes avec des jointures, des filtres et des agrégations pour extraire des sous-ensembles de données spécifiques ou des vues matérialisées.

### 5.5. Extraction des Données des Systèmes Big Data (extract\_big.py)

Ce module est dédié à l’extraction de données à partir de systèmes de Big Data, tels que Apache Hadoop (HDFS), Apache Spark, ou des entrepôts de données cloud (ex: Google BigQuery, Amazon Redshift). L’implémentation dépendra du système spécifique, mais elle pourrait impliquer l’utilisation de bibliothèques comme pyarrow pour Parquet, pyspark pour Spark, ou des SDKs cloud pour interagir avec les services de Big Data. L’objectif est d’extraire efficacement de grands volumes de données distribuées.

## 6. Interrogation des Données

Une fois les données extraites et potentiellement transformées, la capacité à les interroger est cruciale pour la validation et l’analyse exploratoire. Deux types d’interrogation sont envisagés :

### 6.1. Interrogation de Type SQL (query\_sql.py)

Ce module, s’il est implémenté, permettrait d’exécuter des requêtes de type SQL sur les données stockées localement (par exemple, dans une base SQLite ou des DataFrames pandas en mémoire). Il est essentiel pour vérifier l’intégrité des données après l’extraction et la transformation, et pour effectuer des analyses ad-hoc. L’utilisation de pandasql ou de requêtes directes sur une base SQLite permettrait de simuler un environnement de base de données relationnelle pour l’interrogation.

### 6.2. Interrogation des Systèmes Big Data

Pour les données stockées dans des systèmes Big Data, l’interrogation se ferait via les interfaces spécifiques de ces systèmes (ex: HiveQL pour Hive, Spark SQL pour Spark, ou les APIs de requêtes des services cloud). Ce type d’interrogation est optimisé pour les grands volumes de données et les requêtes complexes, permettant des analyses à l’échelle du cluster. Le module extract\_big.py pourrait être étendu pour inclure des fonctionnalités d’interrogation ou un nouveau module query\_big.py pourrait être créé.

## 7. Agrégation des Données

La phase d’agrégation est une étape clé de la transformation, où les données brutes sont consolidées et structurées pour l’analyse. Cette étape est actuellement en cours de développement et comprend :

* **Nettoyage des Données** : Identification et correction des erreurs, suppression des doublons, gestion des valeurs aberrantes. Cela inclut la standardisation des chaînes de caractères (ex: noms d’équipes, noms de joueurs).
* **Suppression des Entrées Corrompues** : Élimination des enregistrements incomplets ou invalides qui pourraient fausser les analyses.
* **Homogénéisation des Formats** : Conversion des types de données (ex: dates, nombres) pour assurer la cohérence entre les différentes sources. Par exemple, s’assurer que toutes les dates sont au même format AAAA-MM-JJ.
* **Agrégation Logique** : Regroupement des données par des clés communes (ex: ID de match, ID de joueur, date) pour créer des résumés ou des vues agrégées. Par exemple, calculer la moyenne de points par match pour chaque joueur.

Cette étape est cruciale pour garantir la qualité et l’exploitabilité des données pour les analyses ultérieures. Elle implique souvent l’utilisation intensive des fonctionnalités de groupby et agg de pandas.

## 8. Préparation des Données

La préparation des données est une extension de la transformation, se concentrant sur l’optimisation des données pour des cas d’utilisation spécifiques, notamment l’analyse statistique et l’apprentissage automatique. Cette étape est en attente de développement et inclura :

* **Gestion des Données Partielles/Manquantes** : Imputation des valeurs manquantes (moyenne, médiane, mode, régression) ou suppression des lignes/colonnes si l’imputation n’est pas appropriée. Des stratégies spécifiques seront définies en fonction de la nature des données.
* **Normalisation des Données** : Mise à l’échelle des valeurs numériques pour qu’elles se situent dans une plage spécifique (ex: 0-1) ou aient une distribution standard (moyenne 0, écart-type 1). Ceci est essentiel pour de nombreux algorithmes d’apprentissage automatique.
* **Création de Caractéristiques (Feature Engineering)** : Dérivation de nouvelles variables à partir des données existantes pour améliorer la puissance prédictive des modèles. Par exemple, calculer le ratio de victoires/défaites d’une équipe ou le pourcentage de tirs réussis d’un joueur.
* **Encodage des Variables Catégorielles** : Conversion des variables textuelles (ex: noms d’équipes, positions de joueurs) en formats numériques utilisables par les algorithmes (ex: One-Hot Encoding, Label Encoding).

## 9. Création du Jeu de Données Final

Le jeu de données final est le résultat des étapes d’extraction, de transformation, d’agrégation et de préparation. Il s’agit d’un ensemble de données propre, cohérent et structuré, prêt à être chargé dans la base de données cible et utilisé pour l’analyse. Ce jeu de données sera optimisé pour les performances de requête et la facilité d’utilisation. Il pourrait s’agir d’un ou plusieurs DataFrames pandas qui seront ensuite persistés.

## 10. Création de la Base de Données

Cette section décrira la conception et la mise en œuvre de la base de données cible. Pour les données structurées, une base de données relationnelle comme PostgreSQL est privilégiée, avec un schéma bien défini (tables, colonnes, clés primaires/étrangères, index). Pour les données semi-structurées ou les logs, MongoDB serait utilisée. Le script de création de la base de données (load\_pg.py, load\_mongo.py) inclura les définitions de schéma (DDL) et les logiques d’insertion (DML).

## 11. Modélisation des Données

La modélisation des données implique la conception du schéma de la base de données pour représenter au mieux les relations entre les entités (joueurs, équipes, matchs, statistiques). Cela inclut la création de diagrammes entité-relation (ERD) et la normalisation des tables pour éviter la redondance et assurer l’intégrité des données. Le fichier models.py contiendrait les définitions des classes ou des schémas qui mappent les données aux structures de la base de données.

## 12. Développement d’API

Pour rendre les données accessibles à d’autres applications ou services, une API RESTful pourrait être développée. Cette API permettrait aux utilisateurs d’interroger les données transformées et agrégées via des requêtes HTTP standardisées. Des frameworks comme Flask ou FastAPI en Python seraient utilisés pour construire cette API, offrant des endpoints pour récupérer des statistiques de joueurs, des résultats de matchs, etc. Cette API serait une couche d’abstraction au-dessus de la base de données finale.

## 13. Améliorations Futures

Le projet est conçu pour être évolutif, et plusieurs améliorations futures sont envisagées pour enrichir ses fonctionnalités et ses performances :

* **Optimisation des Performances ETL** : Amélioration de la vitesse d’exécution du pipeline ETL, notamment pour les grands volumes de données, par l’optimisation des requêtes, l’utilisation de techniques de parallélisation ou de traitement distribué.
* **Intégration de Sources de Données Supplémentaires** : Ajout de nouvelles APIs (ex: données de paris sportifs, réseaux sociaux), de bases de données tierces ou de flux de données en temps réel pour enrichir le jeu de données.
* **Mise en Œuvre de Tableaux de Bord Visuels** : Intégration avec des outils de Business Intelligence (BI) comme Tableau, Power BI, ou des bibliothèques Python de visualisation (Matplotlib, Seaborn, Plotly, Dash) pour créer des tableaux de bord interactifs et des rapports visuels.
* **Application de Modèles d’Apprentissage Automatique pour les Prévisions** : Utilisation des données préparées pour développer et déployer des modèles de Machine Learning (ex: prédiction des résultats de matchs, performances des joueurs, détection de tendances). Cela pourrait inclure des modèles de régression, de classification ou de séries temporelles.
* **Surveillance et Alertes** : Mise en place de systèmes de surveillance pour le pipeline ETL afin de détecter les erreurs, les retards ou les anomalies dans les données, avec des alertes automatiques.
* **Conteneurisation et Déploiement** : Utilisation de Docker pour conteneuriser l’application et de Kubernetes pour l’orchestration du déploiement, assurant une portabilité et une scalabilité accrues.

Ce livrable détaillé fournit une vue d’ensemble complète du projet ETL, de ses spécifications techniques à ses perspectives d’évolution. Il sert de document de référence pour toutes les parties prenantes.