

REPUBLIQUE DU SENEGAL



Un peuple –un but –une foi

**MINISTERE DE L'ENSEIGNEMENT SUPERIEURE DE LA RECHERCHE ET
DE L'INNOVATION**

DIRECTION GENERALE DE L'ENSEIGNEMENT SUPERIEUR



**UNIVERSITE
GASTON BERGER**

L'excellence au service du développement

Université Gaston Berger



UFR Institut Polytechnique de Saint-Louis

Projet Final: Data Engineering avec Apache Spark

Présenté par :

Bassirou SAGNA

ING 3 Info-Telecom

Sous la direction de :

Dr. Djibril MBOUP

Plan :

Introduction

Schéma des Données

Processus de Traitement des Données

Sauvegarde des Données

Codes et résultats

Conclusion

Introduction :

Dans le monde de la gestion des données massives, il est essentiel d'adopter des mesures efficaces pour traiter et stocker les données afin de garantir un niveau élevé de qualité et de performance des systèmes d'information. Ce rapport met l'accent sur la manière dont les données sont traitées et sauvegardées en utilisant **Apache Spark**, qui est un moteur de traitement de données distribué connu pour sa rapidité et son adaptabilité. Le but principal est d'exposer le processus entier, de la lecture des données non transformées jusqu'à leur stockage définitif en utilisant le format Parquet, tout en assurant l'exactitude et l'intégrité des données.

L'objectif du projet est de manipuler un groupe de fichiers JSON qui contiennent des données sur les sources de trafic et les revenus qui y sont rattachés. Analyser les performances des campagnes publicitaires et optimiser les stratégies marketing repose sur l'importance de ces données. Afin de satisfaire aux besoins en termes de performance et d'évolutivité, nous avons opté pour Apache Spark qui assure la réalisation d'opérations sophistiquées de transformation et d'agrégation sur des volumes considérables de données à travers un système distribué.

L'objectif de ce rapport est de présenter une vue exhaustive des différentes étapes nécessaires pour traiter et sauvegarder les données, en fournissant des détails précis à chaque étape. Il aborde spécifiquement :

- La lecture des fichiers JSON, incluant le schéma de données et la méthode d'importation associée.
- Expliquer les opérations appliquées pour préparer les données en vue de l'analyse, notamment le nettoyage et la transformation des données.
- Les données transformées sont sauvegardées sous forme de fichiers Parquet, qui est un format de stockage colonne conçu pour maximiser les performances des requêtes et la compression.
- En plus de cela, le rapport soulignera les difficultés qui ont été rencontrées pendant le processus, comme les erreurs de compilation et les incompatibilités des types de données, ainsi que les mesures prises pour y remédier. Enfin, nous procéderons à l'évaluation des résultats obtenus, en prenant en compte la validation des données ainsi que les performances du traitement.

Pour mener à bien cette étude nous allons utiliser Apache Spark qui est un framework de traitement distribué pour gérer efficacement des données massives. Il propose des options efficaces pour la manipulation en mémoire, la conversion de données et l'analyse à grande échelle. Mais aussi le Format Parquet qui est un format de fichier en colonnes conçu pour offrir des performances optimales lors de la lecture et de l'écriture dans les systèmes distribués. Il assure une compression efficace et un accès rapide aux données.

Schéma des Données :

Toute opération de traitement et d'analyse efficace repose sur le schéma des données. Il établit la configuration des données, en précisant les catégories de données et leur interrelation. Il est essentiel d'avoir une compréhension claire du schéma afin de s'assurer que les données sont traitées correctement, interprétées et stockées tout au long du processus.

Notre projet repose sur des fichiers JSON contenant des données essentielles concernant les sources de trafic et les revenus générés. Voici une vue d'ensemble du schéma des données, tel qu'il apparaît dans les fichiers JSON :

- **traffic_source** : Donne une représentation de la source de trafic sous forme d'une chaîne de caractères. On se sert de ce champ pour repérer l'origine des visiteurs, tels que les campagnes publicitaires ou les sources de référencement.
- **total_rev** : Veuillez fournir le montant total du revenu généré, exprimé en chiffres. Ce champ enregistre la somme totale des revenus liés à une source de trafic particulière.
- **avg_rev** : Donne une représentation sous forme de chaîne de caractères du revenu moyen généré. Ce champ permet d'étudier la moyenne de revenus par type de trafic.

Le schéma en JSON est structuré comme suit :

```
{
  "type" : "struct",
  "fields" : [ {
    "name" : "traffic_source",
    "type" : "string",
    "nullable" : true,
    "metadata" : { }
  }, {
    "name" : "total_rev",
    "type" : "string",
    "nullable" : true,
    "metadata" : { }
  }, {
    "name" : "avg_rev",
    "type" : "string",
    "nullable" : true,
    "metadata" : { }
  } ]
}
```

Il est essentiel de vérifier que le schéma JSON est correctement adapté lors de la conversion des données au format Parquet, car ce dernier est un format de fichier colonne utilisé pour

sauvegarder les données. Le schéma Parquet correspondant est défini comme suit :

```
and corresponding Parquet message type:
message spark_schema {
  optional binary traffic_source (STRING);
  optional binary total_rev (STRING);
  optional binary avg_rev (STRING);
}
```

Il est spécifié que les champs **traffic_source**, **total_rev** et **avg_rev** sont des chaînes de caractères binaires (binary) et leur l'étiquette est indiquée comme étant facultative. En optant pour ce type, la compatibilité avec les types de données d'origine est assurée, tout en offrant une compression et un accès plus efficaces. Une compréhension précise du schéma est cruciale pour plusieurs raisons :

- **Validation des Données :** Veillez à ce que les données lues et écrites soient conformes à la structure attendue afin de réduire au minimum les risques d'erreurs et d'interprétation erronée des données.
- **Transformation Efficace :** Il est possible d'effectuer les transformations appropriées lors du traitement des données afin de garantir une manipulation correcte des types de données.
- **Performance de Sauvegarde :** Améliorer les performances globales des opérations de lecture et d'écriture en optimisant le stockage et la récupération des données sous format Parquet.

Pour résumer, le schéma de données joue un rôle crucial en orientant toutes les étapes du traitement des données, depuis la lecture des informations brutes jusqu'à leur stockage définitif. Pour assurer l'intégrité des données et optimiser les opérations, il est essentiel de prêter une attention particulière à la définition précise et à la concordance des schémas.

Processus de Traitement des Données :

Pour pouvoir utiliser et compter sur les données, il est essentiel de passer par une étape cruciale : le traitement des données. Dans cette partie, les étapes du flux de traitement sont décrites en détail depuis la lecture des données initiales jusqu'à leur sauvegarde finale au format Parquet. En utilisant Apache Spark comme moteur principal, chaque étape est soigneusement conçue afin d'assurer la qualité des données et d'optimiser les performances du traitement.

- **Lecture des Données :** Pour commencer le processus, il faut d'abord lire les données brutes provenant des fichiers JSON. Les fichiers JSON sont fréquemment utilisés en raison de leur polyvalence et de leur compatibilité avec différents systèmes de manipulation des données. À l'aide d'Apache Spark, on importe les données dans un DataFrame qui est une structure de données spécialement conçue pour le traitement distribué. Pour faire le **Chargement des Données** : On utilise la méthode spark pour charger les fichiers JSON. **read.** Utiliser la fonction **jsonToDataFrame()** pour créer un DataFrame contenant les données brutes. Ensuite on a **Validation du Schéma** : On vérifie le schéma des données afin de s'assurer qu'il correspond aux attentes, et on signale toute incohérence détectée pour correction.
- **Nettoyage des Données :** Pour garantir l'intégrité des données, il est primordial de procéder à une étape cruciale qui consiste à nettoyer les données afin d'éliminer toutes erreurs, valeurs manquantes et incohérences présentes dans les données brutes. Ce processus garantit que les données sont en bon état pour être utilisées dans des transformations et analyses futures. Pour ce faire nous procédons en une **Identification des Valeurs Manquantes** en détectant et traitent les valeurs manquantes en les imputant avec des valeurs par défaut ou bien en supprimant les lignes concernées. Puis on passe à la **Correction des Incohérences** c'est dire on corrige les incohérences dans les données, telles que la normalisation des formats de date et la conversion des types de données. Et enfin à la **Validation des Données Nettoyées** ce qui veut dire que l'on vérifie le DataFrame nettoyé afin de s'assurer que toutes les anomalies ont été rectifiées et que les données sont prêtes pour la transformation.
- **Transformation des Données :** Lors de cette étape de transformation des données, différentes opérations de modification et regroupement sont appliquées aux données nettoyées afin qu'elles soient prêtes pour l'analyse. Il est possible d'effectuer diverses opérations telles que des calculs, des regroupements et des filtrages afin de récupérer les données pertinentes. Nous pouvons dans un cas bien précis faire le **Calcul des**

Revenus Totaux et Moyens pour se faire nous pouvons calculer les revenus totaux et moyens par source de trafic, des transformations particulières sont mises en œuvre. Comme le **Regroupement des Données** pour pouvoir obtenir des résumés agrégés, les données sont classées par catégories pertinentes comme la source de trafic. Ensuite nous avons la **Validation des Données Transformées** en d'autre terme on vérifie les résultats des transformations pour s'assurer de leur précision et de leur conformité aux attentes.

- **Sauvegarde des Données :** En transformant les données en format Parquet, il est possible d'optimiser le stockage et les performances des requêtes futures lors de leur sauvegarde. Le format Parquet est un type de fichier qui organise les données en colonnes, offrant une compression optimale et une récupération rapide des données. Au moment de l'**Écriture des Données en Format Parquet** on a le DataFrame qui est enregistré après avoir été transformé à l'aide de la méthode **df. write. mode("overwrite"). spécifiez(path)**, où path est le chemin de destination pour le fichier Parquet. Puis la **Validation de la Sauvegarde** c'est quand on effectue une vérification de la sauvegarde afin de garantir que les données ont été écrites correctement et que le fichier Parquet pourra être consulté ultérieurement.

Le processus de traitement des données, depuis la lecture initiale jusqu'à la sauvegarde finale, est élaboré afin d'assurer une transformation efficace et fiable des données. On garantit la qualité des données et on maximise les performances du traitement en exécutant soigneusement chaque étape du processus. Ce processus de conversion permet de transformer les données brutes en informations exploitables et prêtes à être analysées, facilitant ainsi la prise de décision.

Sauvegarde des Données :

La sauvegarde correcte des résultats transformés est une étape essentielle dans l'optimisation des opérations de traitement de données massives. Dans cette partie, nous plongerons plus en profondeur dans le processus de sauvegarde des données une fois qu'elles auront été traitées avec Apache Spark. L'utilisation du format Parquet pour la sauvegarde des données est primordiale afin de conserver la qualité des informations et optimiser les performances lorsqu'il s'agit d'exécuter des requêtes par la suite.

Une fois que vous avez terminé de nettoyer et transformer les données brutes provenant des fichiers JSON, il est essentiel de les enregistrer dans un format qui garantit à la fois une compression efficace et un accès rapide. Nous avons opté pour le format Parquet, qui est un format de stockage en colonnes. Ce choix répond aux besoins du projet car il permet d'avoir une capacité de stockage compacte tout en offrant des opérations de lecture et d'écriture optimisées.

Le processus de sauvegarde des données vise à atteindre les objectifs suivants :

- **Conservation de la Structure et de la Qualité des Données :** Veiller à ce que les données transformées soient enregistrées sans perte d'intégrité ou de modification.
- **Optimisation des Performances de Lecture :** En optant pour le format Parquet, vous pourrez bénéficier de la compression et du stockage en colonnes, ce qui facilitera les opérations de lecture et réduira le temps d'accès aux données.
- **Flexibilité et Scalabilité :** Il est essentiel que le processus de sauvegarde puisse s'ajuster aux quantités croissantes de données et aux éventuels changements dans les besoins en matière de stockage.

Le processus de sauvegarde des données se compose des étapes suivantes :

- **Préparation des Données :** Les données sont nettoyées et transformées avant la sauvegarde afin de garantir leur qualité et leur pertinence. Dans ce processus, on supprime les anomalies, on normalise les formats et on applique les calculs nécessaires.
- **Configuration du Format de Sauvegarde :** La mise en place de la configuration pour sauvegarder au format Parquet est terminée. Cela inclut la description des choix de compression, le mode de conservation sélectionné et l'indication du chemin où les fichiers seront sauvegardés.

- **Exécution de la Sauvegarde :** Ensuite, les données transformées sont écrites dans le format Parquet et sauvegardées au chemin spécifié. Spark est utilisé dans cette étape pour traiter les données de manière efficace grâce à ses capacités distribuées.
- **Validation et Vérification :** Une fois la sauvegarde effectuée, il est procédé à des vérifications afin de garantir que les données sont bien enregistrées et disponibles. Cela implique de vérifier l'intégrité et les performances de la sauvegarde afin d'en valider la qualité.

Ainsi pour mener à bien cette partie du travail nous avons eu à utiliser :

- **Apache Spark :** Grâce à ses capacités de traitement et de gestion des données distribuées, il est utilisé pour transformer les données et les sauvegarder au format Parquet.
- **Format Parquet :** Sélectionné pour sa capacité de compression hautement efficace et ses performances optimales lors de la lecture et de l'écriture.

Dans le cycle de gestion des données massives, la sauvegarde des données est une étape essentielle. Ce processus garantit une performance élevée et l'intégrité des données pour les futures analyses grâce à l'utilisation d'Apache Spark et du format Parquet, assurant ainsi une sauvegarde efficace et optimisée des données transformées.

Codes et résultats :

Le processus de traitement des données à l'aide d'Apache Spark implique plusieurs étapes clés, chacune illustrée par des extraits de code Scala. Ces extraits montrent comment les données sont lues, transformées, et sauvegardées. Voici un aperçu des principaux morceaux de code utilisés dans ce projet :

- **Load.scala**

```
import org.apache.spark.sql.{DataFrame, SaveMode}

object Load {

  def saveData(df: DataFrame, saveMode: String, format: String, path: String): Unit = {
    try {
      df.write
        .format(format)           // Spécifie le format de sauvegarde (e.g., parquet, csv)
        .mode(saveMode)          // Spécifie le mode de sauvegarde (e.g., overwrite, append)
        .save(path)              // Spécifie le chemin de sauvegarde
    } catch {
      case e: Exception => {
        Utils._log.error(s"Erreur lors de la sauvegarde des données: ${e.getMessage}")
        throw e
      }
    }
  }
}
```

- **Extract.scala**

```
import org.apache.spark.sql.{DataFrame, SparkSession}

object Extract {

  def read_source_file(path: String, format : String): DataFrame = {
    try {
      val df = Utils._spark.read.format(format)
        .option("header", "true")
        .option("inferSchema", "true")
        .load(path)
      df
    } catch {
      case e: Exception => {
        Utils._log.error(s"The input path defined is incorrect or " +
          s"the file format does not match data." +
          s"or file format is incorrect. " + e.getMessage)
        throw e
      }
    }
  }
}
```

- Utils.scala

```
You, il y a 1 heure | 3 authors (sopeKhadim and others)
import org.apache.spark.sql.Session
import org.slf4j.{Logger, LoggerFactory}

object Utils {

  val _spark: Session = Session.builder.
    appName("Project Spark").
    config("spark.ui.port", "0").
    master("local[*]").
    getOrCreate()

  val _log: Logger = LoggerFactory.getLogger(Main.getClass)
}
```

- Transform.scala

```
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.functions._

object Transform {

  /**
   * Cette fonction permet de nettoyer les données brutes.
   * @param df : DataFrame
   * @return DataFrame
   */
  def cleanData(df: DataFrame): DataFrame = {
    /**
     * 0. Traiter toutes les colonnes en date timestamp vers YYYY/MM/DD HH:MM SSS.
     * On suppose que les colonnes de type timestamp sont `event_previous_timestamp`, `event_t
     */
    // Convertir les colonnes timestamp
    val firstDF = df.withColumn("event_previous_timestamp", to_timestamp(col("event_previous
      .withColumn("event_timestamp", to_timestamp(col("event_timestamp")))
      .withColumn("user_first_touch_timestamp", to_timestamp(col("user_first_touch_timestamp

    /**
     * 1. Extraire les revenus d'achat pour chaque événement
     * - Ajouter une nouvelle colonne nommée revenue en faisant l'extraction de ecommerce.pur
     */
    val revenueDF = firstDF.withColumn("revenue", col("ecommerce.purchase_revenue_in_usd"))
  }
```

- Main.scala

```
object Main {
  def main(args: Array[String]): Unit = {
  def main(args: Array[String]): Unit = {
    if (args.length <= 1) {
      println("Missing two arguments.")
      println("usage : You have to give the source file path ou the output path")
      System.exit(1)
    }
    //get the path to the JSON file
    val jsonFilePath = args(0)
    val outputPath = args(1)

    val df = Extract.read_source_file(jsonFilePath, "json")
    df.show()

    val cleanDF = Transform.cleanData(df)
    cleanDF.show()

    val transformDF = Transform.computeTrafficRevenue(cleanDF)
    transformDF.show()

    Load.saveData(transformDF, "overwrite", "parquet", outputPath)
  }
}
```

Une fois la configuration du code terminer nous pouvons aller executer le code pour ce faire nous avons choisi d'executer le code en ligne de commande avec SBT (Simple Build Tool) est un outil de construction utilisé principalement pour les projets Scala, bien qu'il puisse également être utilisé avec des projets Java. Il facilite la gestion de la construction, des dépendances, et des tâches de développement pour les projets basés sur JVM (Java Virtual Machine).

Pour compiler le projet nous allons en premier lieu lancer la commande La commande **sbt clean** qui est une des commandes courantes dans SBT, et elle permet le **Nettoyage du Projet** en supprimant les fichiers de compilation précédents, les fichiers temporaires, et les artefacts générés lors des compilations précédentes. Cela inclut les fichiers dans les répertoires target (où les compilations sont stockées) et project/target (où les fichiers de construction SBT sont stockés).







```
C:\Users\lenovo\Desktop\Documents\project_scala_spark>sbt clean
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.jline.terminal.impl.exec.ExecTerminalProvider$ReflectionRedirectPipeCreator (file:/C:/Users/lenovo/.sbt/boot/scala-2.12.19/org
.scala-sbt/sbt/1.10.1/jline-terminal-3.24.1.jar) to constructor java.lang.ProcessBuilder$RedirectPipeImpl()
WARNING: Please consider reporting this to the maintainers of org.jline.terminal.impl.exec.ExecTerminalProvider$ReflectionRedirectPipeCreator
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[info] welcome to sbt 1.10.1 (OpenLogic Java 11.0.23)
[info] loading project definition from C:\Users\lenovo\Desktop\Documents\project_scala_spark\project
[info] loading settings for project root from build.sbt ...
[info] set current project to project_scala_spark (in build file:/C:/Users/lenovo/Desktop/Documents/project_scala_spark/)
[success] Total time: 0 s, completed 8 août 2024 à 20:42:14
←[0]
```

La commande **sbt compile** est utilisée pour compiler le code source de notre projet en utilisant SBT (Simple Build Tool). Elle fait **Compilation du Code, Gestion des Dépendances, Détection des Changements, Gestion des Erreurs et la Génération des Fichiers de Sortie** :

```
C:\Users\lenovo\Desktop\Documents\project_scala_spark>sbt compile
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.jline.terminal.impl.exec.ExecTerminalProvider$ReflectionRedirectPipeCreator (f
.scala-sbt/sbt/1.10.1/jline-terminal-3.24.1.jar) to constructor java.lang.ProcessBuilder$RedirectPipeImpl()
WARNING: Please consider reporting this to the maintainers of org.jline.terminal.impl.exec.ExecTerminalProvider$Reflecti
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[info] welcome to sbt 1.10.1 (OpenLogic Java 11.0.23)
[info] loading project definition from C:\Users\lenovo\Desktop\Documents\project_scala_spark\project
[info] loading settings for project root from build.sbt ...
[info] set current project to project_scala_spark (in build file:/C:/Users/lenovo/Desktop/Documents/project_scala_spark/)
[info] Executing in batch mode. For better performance use sbt's shell
[info] compiling 5 Scala sources to C:\Users\lenovo\Desktop\Documents\project_scala_spark\target\scala-2.12\classes ...
[success] Total time: 17 s, completed 8 août 2024 à 20:42:44
←[0]
```

La commande **sbt package** est utilisée pour créer un fichier JAR (Java ARchive) à partir du code source de votre projet. Il permet entre autre la **Création du JAR**, de faire l'**Inclusion des Dépendances**.

```
C:\Users\lenovo\Desktop\Documents\project_scala_spark>sbt package
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.jline.terminal.impl.exec.ExecTe
.scala-sbt/sbt/1.10.1/jline-terminal-3.24.1.jar) to constructor java.lang
WARNING: Please consider reporting this to the maintainers of org.jline.t
WARNING: Use --illegal-access=warn to enable warnings of further illegal
WARNING: All illegal access operations will be denied in a future release
[info] welcome to sbt 1.10.1 (OpenLogic Java 11.0.23)
[info] loading project definition from C:\Users\lenovo\Desktop\Documents\
[info] loading settings for project root from build.sbt ...
[info] set current project to project_scala_spark (in build file:/C:/User
[success] Total time: 3 s, completed 8 août 2024 à 20:43:01
←[0]
```

Nom	Modifié le	Type	Taille
 classes	08/08/2024 20:42	Dossier de fichiers	
 sync	08/08/2024 20:42	Dossier de fichiers	
 update	08/08/2024 18:18	Dossier de fichiers	
 zinc	08/08/2024 20:42	Dossier de fichiers	
 project_scala_spark_2.12-0.1.0.jar	08/08/2024 20:43	Executable Jar File	10 Ko

La commande **sbt run** est utilisée pour exécuter le programme principal dans un projet Scala.

Pour ce projet comme nous n'avons pas défini le chemin où se trouve le fichier input voilà ce nous obtenons :

```
C:\Users\lenovo\Desktop\Documents\project_scala_spark>sbt run
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.jline.terminal.impl.exec.ExecTerminalProvider$ReflectionRedirectPipeCreator (file:/C:/Users/lenovo/.sbt/boot/scala-2.12.19/org.jline-sbt/sbt/1.10.1/jline-terminal-3.24.1.jar) to constructor java.lang.ProcessBuilder$RedirectPipeImpl()
WARNING: Please consider reporting this to the maintainers of org.jline.terminal.impl.exec.ExecTerminalProvider$ReflectionRedirectPipeCreator
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[info] welcome to sbt 1.10.1 (OpenLogic Java 11.0.23)
[info] loading project definition from C:\Users\lenovo\Desktop\Documents\project_scala_spark\project
[info] loading settings for project root from build.sbt ...
[info] set current project to project_scala_spark (in build file:/C:/Users/lenovo/Desktop/Document/project_scala_spark/)
[info] running Main
Missing two arguments.
usage: You have to give the source file path ou the output path
```

Une fois que nous donnons le bon chemin du fichier JSON que nous étudions comme ceci :

sbt "run

C:/Users/lenovo/Desktop/Documents/project_scala_spark/datasets/events/events.json

C:/Users/lenovo/Desktop/Documents/project_scala_spark/output"

On a :

```
C:\Users\lenovo\Desktop\Documents\project_scala_spark>sbt "run C:/Users/lenovo/Desktop/Documents/project_scala_spark/datasets/events/events.json C:/Users/lenovo/Desktop/Documents/project_scala_spark/output"
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.jline.terminal.impl.exec.ExecTerminalProvider$ReflectionRedirectPipeCreator (file:/C:/Users/lenovo/.sbt/boot/scala-2.12.19/org.jline-sbt/sbt/1.10.1/jline-terminal-3.24.1.jar) to constructor java.lang.ProcessBuilder$RedirectPipeImpl()
WARNING: Please consider reporting this to the maintainers of org.jline.terminal.impl.exec.ExecTerminalProvider$ReflectionRedirectPipeCreator
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[info] welcome to sbt 1.10.1 (OpenLogic Java 11.0.23)
[info] loading project definition from C:\Users\lenovo\Desktop\Documents\project_scala_spark\project
[info] loading settings for project root from build.sbt ...
[info] set current project to project_scala_spark (in build file:/C:/Users/lenovo/Desktop/Document/project_scala_spark/)
[info] running Main C:/Users/lenovo/Desktop/Documents/project_scala_spark/datasets/events/events.json C:/Users/lenovo/Desktop/Documents/project_scala_spark/output
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/08/08 20:43:41 INFO SparkContext: Running Spark version 3.5.1
24/08/08 20:43:41 INFO SparkContext: OS info Windows 10, 10.0, amd64
24/08/08 20:43:41 INFO SparkContext: Java version 11.0.23
24/08/08 20:43:42 INFO ResourceUtils: =====
24/08/08 20:43:42 INFO ResourceUtils: No custom resources configured for spark.driver.
24/08/08 20:43:42 INFO ResourceUtils: =====
24/08/08 20:43:42 INFO SparkContext: Submitted application: Project Spark
```


device	ecommerce	event_name	event_previous_timestamp	event_timestamp	geo	items	traffic_source	user_first_touch_timestamp	
macOS	NULL, NULL, NULL	warranty	1593878899217692	1593878946592107	{Montrose, MI}	[]	google	1593878899217692	UA00000107379500
Windows	NULL, NULL, NULL	press	1593876662175340	1593877011756535	{Northampton, MA}	[]	google	1593876662175340	UA00000107359357
macOS	NULL, NULL, NULL	add_item	1593878792892652	1593878815459100	{Salinas, CA}	{NULL, M_STAN_T,...}	youtube	1593878455472030	UA00000107375547
iOS	NULL, NULL, NULL	mattresses	1593878178791663	1593878809276923	{Everett, MA}	[]	facebook	1593877903116176	UA00000107370581
Windows	NULL, NULL, NULL	mattresses	NULL	1593878628143633	{Cottage Grove, MN}	[]	google	1593878628143633	UA00000107377108
Windows	NULL, NULL, NULL	main	NULL	1593878634344194	{Medina, MN}	[]	youtube	1593878634344194	UA00000107377161
iOS	NULL, NULL, NULL	main	NULL	1593877936171803	{Mount Pleasant, UT}	[]	direct	1593877936171803	UA00000107370851
macOS	NULL, NULL, NULL	main	NULL	1593876843215329	{Piedmont, AL}	[]	instagram	1593876843215329	UA00000107360961
Android	NULL, NULL, NULL	warranty	1593878529774474	1593879213196400	{Rancho Santa Mar...}	[]	instagram	1593878529774474	UA00000107376205
Windows	NULL, NULL, NULL	main	NULL	1593876713246514	{Elyria, OH}	[]	facebook	1593876713246514	UA00000107359805
iOS	NULL, NULL, NULL	original	1593878068949001	1593878170903989	{Longview, WA}	[]	google	1593877826716812	UA00000107369909
Linux	NULL, NULL, NULL	main	NULL	1593878036347579	{Lyndhurst, OH}	[]	direct	1593878036347579	UA00000107371743
Android	NULL, NULL, NULL	down	1593879057792999	1593879125815755	{Jackson, MO}	[]	facebook	1593879057792999	UA00000107380961
Linux	NULL, NULL, NULL	main	NULL	1593878672173087	{Cedar Rapids, IA}	[]	google	1593878672173087	UA00000107377487
macOS	NULL, NULL, NULL	main	NULL	1593876429415452	{Phoenix, AZ}	[]	google	1593876429415452	UA00000107357350
iOS	NULL, NULL, NULL	mattresses	NULL	1593876687337581	{Warwick, RI}	[]	google	1593876687337581	UA00000107359573
macOS	NULL, NULL, NULL	premium	1593877223736871	1593877973962436	{Everett, WA}	[]	instagram	1593877223736871	UA00000107364369
Windows	NULL, NULL, NULL	reviews	1593876442432487	1593876944661570	{Concord, CA}	[]	direct	1593876442432487	UA00000107357467
iOS	NULL, NULL, NULL	original	1593877781854634	1593877788141768	{Dunwoody, GA}	[]	google	1593877781854634	UA00000107369512
iOS	NULL, NULL, NULL	main	1593877445670953	1593877497207417	{Rochester, MN}	[]	facebook	1593877300577217	UA00000107355056

device	ecommerce	event_previous_timestamp	event_timestamp	geo	items	traffic_source	user_first_touch_timestamp	us
Linux	{1195.0, 1, 1}	+238333-03-29 18:...	+238333-05-13 17:...	{Shawnee, KS}	{NULL, M_STAN_K,...}	google	+238273-02-10 13:...	UA000000107362263
iOS	{1045.0, 1, 1}	+238320-04-19 16:...	+238320-05-14 03:...	{Detroit, MI}	{NULL, M_STAN_Q,...}	facebook	+238280-07-11 11:...	UA000000107364432
Android	{595.0, 1, 1}	+238302-09-14 23:...	+238335-07-18 08:...	{East Chicago, IN}	{NULL, M_STAN_T,...}	google	+238269-09-24 03:...	UA000000107361347
iOS	{2290.0, 2, 2}	+238293-10-30 10:...	+238293-11-22 17:...	{Warwick, RI}	{NULL, M_PREM_F,...}	google	+238263-04-28 09:...	UA000000107359573
macOS	{945.0, 1, 1}	+238341-05-30 03:...	+238342-11-17 02:...	{Boonville, MO}	{NULL, M_STAN_F,...}	facebook	+238324-01-15 01:...	UA000000107376872
Windows	{595.0, 1, 1}	+238302-01-12 14:...	+238305-07-23 02:...	{Hampton, VA}	{NULL, M_STAN_T,...}	google	+238274-04-21 11:...	UA000000107362622
Android	{945.0, 1, 1}	+238316-03-11 21:...	+238325-04-01 00:...	{White Bear Lake,...}	{NULL, M_STAN_F,...}	direct	+238275-10-15 23:...	UA000000107363039
Chrome OS	{1095.0, 1, 1}	+238338-12-12 15:...	+238342-09-07 10:...	{San Antonio, TX}	{NULL, M_PREM_T,...}	instagram	+238278-02-05 08:...	UA000000107363715
macOS	{1045.0, 1, 1}	+238286-09-18 22:...	+238286-11-04 15:...	{Searcy, AR}	{NULL, M_STAN_Q,...}	direct	+238268-07-08 13:...	UA000000107361027
iOS	{1045.0, 1, 1}	+238336-02-14 08:...	+238338-01-27 04:...	{Southport, IN}	{NULL, M_STAN_Q,...}	instagram	+238259-10-02 13:...	UA000000107358614
Windows	{1045.0, 1, 1}	+224257-02-11 11:...	+224257-03-08 14:...	{Columbus, OH}	{NULL, M_STAN_Q,...}	instagram	+224179-12-16 02:...	UA000000106096620
macOS	{1640.0, 2, 2}	+224317-03-04 02:...	+224318-05-29 16:...	{St. Petersburg, FL}	{NULL, M_STAN_T,...}	facebook	+224227-11-13 22:...	UA000000106011218
Chrome OS	{654.0, 2, 2}	+224031-02-05 10:...	+224052-11-10 01:...	{Farley, IA}	{NULL, M_STAN_T,...}	google	+223945-09-16 05:...	UA000000105991167
Windows	{1045.0, 1, 1}	+222782-02-25 18:...	+222789-01-10 22:...	{Houston, TX}	{NULL, M_STAN_Q,...}	direct	+222646-08-31 10:...	UA000000105957034
Windows	{1795.0, 1, 1}	+223144-12-29 06:...	+223148-02-06 00:...	{Monroe, LA}	{NULL, M_PREM_Q,...}	facebook	+223108-11-04 09:...	UA000000105978828
Android	{1195.0, 1, 1}	+223671-02-01 23:...	+223673-05-21 02:...	{Donaldsonville, LA}	{NULL, M_STAN_K,...}	google	+223658-03-16 10:...	UA000000105983663
iOS	{1195.0, 1, 1}	+222982-09-23 02:...	+223017-08-31 17:...	{Shreveport, LA}	{NULL, M_STAN_K,...}	google	+222796-03-15 05:...	UA000000105968663
iOS	{1128.6, 2, 2}	+234676-10-11 12:...	+234677-06-03 15:...	{San Ramon, CA}	{NEWBED10, M_STA...}	email	+224411-02-05 14:...	UA000000106033948
Android	{1075.5, 1, 1}	+227246-07-03 12:...	+227264-09-16 19:...	{La Vernia, TX}	{NEWBED10, M_STA...}	email	+222693-07-03 06:...	UA000000105961848
Windows	{595.0, 1, 1}	+224396-10-17 16:...	+224420-12-28 08:...	{Boston, MA}	{NULL, M_STAN_T,...}	facebook	+224283-05-01 19:...	UA000000106017264

traffic_source	total_rev	avg_rev
email	36,935.40	998.25
google	28,936.00	964.53
facebook	12,952.00	996.31
direct	9,129.00	1,141.12
instagram	8,160.00	1,020.00


```
{
  "type" : "struct",
  "fields" : [ {
    "name" : "traffic_source",
    "type" : "string",
    "nullable" : true,
    "metadata" : { }
  }, {
    "name" : "total_rev",
    "type" : "string",
    "nullable" : true,
    "metadata" : { }
  }, {
    "name" : "avg_rev",
    "type" : "string",
    "nullable" : true,
    "metadata" : { }
  } ]
}
and corresponding Parquet message type:
message spark_schema {
  optional binary traffic_source (STRING);
  optional binary total_rev (STRING);
  optional binary avg_rev (STRING);
}
```

```
[success] Total time: 50 s, completed 8 août 2024 à 20:44:06
+-----+
24/08/08 20:44:07 INFO SparkContext: Invoking stop() from shutdown hook
24/08/08 20:44:07 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/08/08 20:44:07 INFO SparkUI: Stopped Spark web UI at http://B-SAGNA:55718
24/08/08 20:44:07 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
24/08/08 20:44:07 INFO MemoryStore: MemoryStore cleared
24/08/08 20:44:07 INFO BlockManager: BlockManager stopped
24/08/08 20:44:07 INFO BlockManagerMaster: BlockManagerMaster stopped
24/08/08 20:44:07 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
24/08/08 20:44:07 INFO SparkContext: Successfully stopped SparkContext
24/08/08 20:44:07 INFO ShutdownHookManager: Shutdown hook called
24/08/08 20:44:07 INFO ShutdownHookManager: Deleting directory C:\Users\lenovo\AppData\Local\Temp\spark-93912a69-55d0-494e-ad57-2dae32129871
```

Conclusion :

Ce rapport a présenté une analyse détaillée des processus de transformation et de sauvegarde des données à l'aide d'Apache Spark, avec une attention particulière portée à la gestion des formats de données et des modes de sauvegarde. Nous avons exploré les aspects techniques des opérations réalisées sur des datasets.

Nous avons d'abord détaillé les étapes de traitement des données, en expliquant les différentes transformations appliquées aux datasets, telles que le nettoyage et la transformation des données. L'exploration des erreurs rencontrées, telles que les colonnes non résolues et les modes de sauvegarde incorrects, a permis de comprendre les défis techniques spécifiques et de proposer des solutions appropriées pour résoudre ces problèmes.

La mise en œuvre des processus de sauvegarde des données a été examinée avec une attention particulière à la gestion des différents formats de fichiers, y compris Parquet et JSON. L'utilisation des modes de sauvegarde, comme `overwrite`, a été discutée pour assurer une gestion efficace des données lors des écritures dans les systèmes de fichiers.

Ce rapport souligne l'importance d'une gestion rigoureuse des formats de données et des modes de sauvegarde pour garantir la qualité et la fiabilité des processus de traitement des données. En conclusion, ce rapport fournit un aperçu complet des méthodes et des meilleures pratiques pour le traitement et la sauvegarde des données avec Apache Spark, tout en offrant des solutions pratiques aux défis rencontrés. Les leçons tirées de cette analyse serviront de base solide pour des projets futurs, garantissant une gestion des données efficace et fiable dans des environnements complexes.