



# Rapport de Travaux Pratiques

## Projet ETL avec Pentaho Data Integration

*Étude de cas : Tour de France Cycliste*

**École :** ENSA Tanger

**Encadré par :** Pr. Hassan BDIR  
Doctorant et Expert en Sciences des Données  
Chef de Filière

**Réalisé par :** ALLAM ELARBI

**Année universitaire :** 2025-2026

20 octobre 2025

# Table des matières

<b>Introduction</b>	<b>7</b>
<b>1 Présentation générale du projet</b>	<b>9</b>
1.1 Contexte et données du projet . . . . .	9
1.1.1 Description du jeu de données . . . . .	9
1.1.2 Sources de données . . . . .	10
1.2 Outils utilisés . . . . .	10
1.2.1 Pentaho Data Integration (PDI/Spoon) . . . . .	10
1.2.2 MySQL et MySQL Workbench . . . . .	10
1.3 Organisation du projet . . . . .	11
1.3.1 Architecture des dossiers . . . . .	11
<b>2 Configuration de l'environnement</b>	<b>12</b>
2.1 Installation des outils . . . . .	12
2.1.1 Prérequis . . . . .	12
2.1.2 Vérification de l'installation PDI . . . . .	12
2.2 Configuration des bases de données . . . . .	12
2.2.1 Création du schéma BD_S (Base de données Source) . . . . .	12
2.2.2 Création du schéma BD_CIBLE (Base de données Cible) . . . . .	15
2.3 Configuration des connexions dans Pentaho PDI . . . . .	15
2.3.1 Lancement de l'interface Spoon . . . . .	15
2.3.2 Création de la connexion à BD_S . . . . .	15
2.3.3 Création de la connexion à BD_CIBLE . . . . .	16
<b>3 Réalisation des transformations ETL</b>	<b>18</b>
3.1 T00 : Chargement des données PAYS depuis Excel . . . . .	18
3.1.1 Objectif . . . . .	18
3.1.2 Création de la transformation . . . . .	18
3.1.3 Configuration des composants . . . . .	19
3.1.4 Flux complet de la transformation . . . . .	22
3.1.5 Exécution et résultats . . . . .	23

3.1.6	Amélioration pour la ré-exécutabilité . . . . .	24
3.2	T01 : Chargement des données EQUIPE depuis un fichier texte . . . . .	25
3.2.1	Objectif . . . . .	25
3.2.2	Création de la transformation . . . . .	25
3.2.3	Configuration des composants . . . . .	25
3.2.4	Exécution et validation . . . . .	26
3.3	T02 : Export des données PAYS vers Excel . . . . .	28
3.3.1	Objectif . . . . .	28
3.3.2	Création de la transformation . . . . .	28
3.3.3	Configuration des composants . . . . .	28
3.3.4	Exécution et validation . . . . .	30
3.4	T03 : Chargement des données ETAPE depuis XML . . . . .	32
3.4.1	Objectif . . . . .	32
3.4.2	Création de la transformation . . . . .	32
3.4.3	Configuration des composants . . . . .	32
3.4.4	Exécution et validation . . . . .	37
3.4.5	Population des autres tables . . . . .	38
3.5	T04 : Calcul du budget par coureur . . . . .	39
3.5.1	Objectif . . . . .	39
3.5.2	Préparation : Ajout de la colonne calculée . . . . .	39
3.5.3	Configuration des composants . . . . .	39
3.5.4	Exécution et validation . . . . .	42
3.6	T05 : Chargement trié des pays vers BD_CIBLE . . . . .	44
3.6.1	Objectif . . . . .	44
3.6.2	Configuration des composants . . . . .	44
3.6.3	Exécution et validation . . . . .	46
3.7	T06 : Jointure Coureur-Équipe . . . . .	48
3.7.1	Objectif . . . . .	48
3.7.2	Architecture de la transformation . . . . .	48
3.7.3	Configuration des composants . . . . .	48
3.7.4	Exécution et validation . . . . .	50
3.7.5	Analyse des résultats . . . . .	51
<b>4</b>	<b>Orchestration des transformations avec les Jobs</b>	<b>52</b>
4.1	Introduction aux Jobs . . . . .	52
4.2	Correction préalable : Ré-exécutabilité de T00 . . . . .	52
4.2.1	Problème identifié . . . . .	52
4.2.2	Solution appliquée . . . . .	52
4.3	J01_Sec_Main_Job : Job de chargement initial . . . . .	53

---

4.3.1	Objectif . . . . .	53
4.3.2	Structure du job . . . . .	53
4.3.3	Configuration des liens . . . . .	53
4.3.4	Exécution et résultats . . . . .	53
4.4	J01_Sec_Job : Job de traitement . . . . .	54
4.4.1	Objectif . . . . .	54
4.4.2	Structure du job . . . . .	54
4.4.3	Exécution et résultats . . . . .	54
4.5	Orchestration globale . . . . .	55
4.5.1	Ordre d'exécution recommandé . . . . .	55
4.5.2	Avantages de l'approche modulaire . . . . .	55
4.5.3	Démonstration vidéo . . . . .	55
<b>5</b>	<b>Analyse et bilan du projet</b>	<b>56</b>
5.1	Synthèse des transformations réalisées . . . . .	56
5.1.1	Récapitulatif des objectifs atteints . . . . .	56
5.1.2	Volumétrie des données traitées . . . . .	56
5.2	Compétences acquises . . . . .	57
5.2.1	Maîtrise des concepts ETL . . . . .	57
5.2.2	Utilisation de Pentaho Data Integration . . . . .	57
5.3	Difficultés rencontrées et solutions . . . . .	58
5.3.1	Problèmes techniques et résolutions . . . . .	58
5.3.2	Leçons apprises . . . . .	58
5.4	Avantages de l'approche ETL avec Pentaho . . . . .	59
5.4.1	Points forts identifiés . . . . .	59
5.4.2	Cas d'usage adaptés . . . . .	59
5.5	Perspectives d'amélioration . . . . .	60
5.5.1	Améliorations techniques possibles . . . . .	60
5.5.2	Extensions fonctionnelles . . . . .	60
5.6	Application dans un contexte professionnel . . . . .	61
5.6.1	Transférabilité des compétences . . . . .	61
5.6.2	Conformité et gouvernance des données . . . . .	61
<b>Conclusion</b>		<b>62</b>
<b>A Scripts SQL</b>		<b>65</b>
A.1	Script de création de la base de données source . . . . .	65
A.2	Script d'ajout de la colonne calculée . . . . .	66
A.3	Script de création de la base cible . . . . .	66

<b>B Configuration des transformations</b>	<b>67</b>
B.1 Paramètres de connexion aux bases de données . . . . .	67
B.1.1 Connexion Conn_BD_S . . . . .	67
B.1.2 Connexion Conn_BD_CIBLE . . . . .	67
B.2 Liste des composants PDI utilisés . . . . .	68
<b>C Guide de reproduction du projet</b>	<b>69</b>
C.1 Prérequis système . . . . .	69
C.2 Installation étape par étape . . . . .	69
C.2.1 Étape 1 : Installation de MySQL . . . . .	69
C.2.2 Étape 2 : Installation de MySQL Workbench . . . . .	69
C.2.3 Étape 3 : Installation de Pentaho Data Integration . . . . .	69
C.2.4 Étape 4 : Configuration des bases de données . . . . .	70
C.2.5 Étape 5 : Configuration dans PDI . . . . .	70
C.3 Ordre d'exécution recommandé . . . . .	70
C.4 Vérifications à effectuer . . . . .	70
<b>D Glossaire</b>	<b>72</b>
<b>E Bibliographie et ressources</b>	<b>73</b>
E.1 Documentation officielle . . . . .	73
E.2 Tutoriels et guides . . . . .	73
E.3 Ouvrages recommandés . . . . .	73

# Table des figures

2.1	Exécution du script de création des tables dans MySQL Workbench . . . . .	14
2.2	Vérification de la création des tables dans le schéma BD_S . . . . .	14
2.3	Configuration de la connexion Conn_BD_S . . . . .	16
2.4	Test réussi de la connexion à BD_S . . . . .	16
3.1	Création et enregistrement de la transformation T00 . . . . .	18
3.2	Ajout du composant Microsoft Excel Input . . . . .	19
3.3	Sélection du fichier Pays.xls . . . . .	19
3.4	Sélection de la feuille de données . . . . .	20
3.5	Aperçu des colonnes extraites du fichier Excel . . . . .	20
3.6	Configuration du filtre pour éliminer les lignes invalides . . . . .	21
3.7	Configuration du composant Table Output . . . . .	22
3.8	Flux complet : Excel Input → Filter → Table Output . . . . .	22
3.9	Exécution réussie de la transformation T00 . . . . .	23
3.10	Vérification des données dans la table pays de MySQL . . . . .	23
3.11	Création de la transformation T01_Load_EQUIPE . . . . .	25
3.12	Configuration du Text File Input avec aperçu des données . . . . .	26
3.13	Exécution réussie de T01 . . . . .	26
3.14	Validation : 22 équipes insérées dans la table equipe . . . . .	27
3.15	Création de la transformation T02_Export_PAYS_to_Excel . . . . .	28
3.16	Établissement de la connexion à BD_S . . . . .	29
3.17	Configuration de l'extraction avec test de la requête . . . . .	29
3.18	Configuration du Microsoft Excel Output . . . . .	30
3.19	Exécution réussie de la transformation T02 . . . . .	30
3.20	Fichier Excel généré avec les données des pays . . . . .	31
3.21	Création de la transformation T03_Load_ETAPE . . . . .	32
3.22	Configuration du XML Input avec test d'extraction . . . . .	33
3.23	Configuration du XPath pour parser le fichier XML . . . . .	34
3.24	Configuration du filtre de contrôle qualité . . . . .	35
3.25	Configuration de l'insertion dans la table etape . . . . .	36
3.26	Connexion à la base de données BD_S . . . . .	36

3.27	Exécution réussie de la transformation T03 . . . . .	37
3.28	Validation : 21 étapes insérées dans la table etape . . . . .	37
3.29	Ajout de la colonne BUDGET_PAR_COUREUR . . . . .	39
3.30	Configuration de l'extraction des données budgétaires . . . . .	40
3.31	Configuration du Calculator pour le calcul du budget par coureur . . . . .	40
3.32	Configuration du composant Insert / Update . . . . .	41
3.33	État de la table avant l'exécution (colonne vide) . . . . .	42
3.34	Exécution réussie de la transformation T04 . . . . .	42
3.35	Validation : colonne BUDGET_PAR_COUREUR calculée et remplie . . . . .	43
3.36	Configuration du tri sur le nom des pays . . . . .	44
3.37	Établissement de la connexion à BD_CIBLE . . . . .	45
3.38	Script SQL généré pour créer la table PAYS_TRIE . . . . .	46
3.39	Exécution réussie de la transformation T05 . . . . .	46
3.40	Table PAYS_TRIE créée avec données triées alphabétiquement . . . . .	47
3.41	Exécution réussie de la transformation T06 . . . . .	50
3.42	Table COUREUR_ENRICHED avec les données jointes . . . . .	50
4.1	Exécution réussie du job J01_Sec_Main_Job . . . . .	53
4.2	Exécution réussie du job J01_Sec_Job . . . . .	54

# Introduction

Ce rapport présente de manière détaillée la réalisation d'un projet ETL (Extract, Transform, Load) utilisant Pentaho Data Integration (PDI), également connu sous le nom de Spoon. Le projet porte sur l'exploitation de données liées au Tour de France Cycliste.

## Objectifs du projet

Les principaux objectifs de ce travail pratique sont :

- Maîtriser les concepts fondamentaux de l'ETL et leur mise en œuvre pratique
- Apprendre à utiliser Pentaho Data Integration pour créer des transformations de données
- Manipuler différents formats de données (Excel, XML, fichiers texte, bases de données MySQL)
- Concevoir et exécuter des flux de données complexes incluant des jointures, des calculs et des tris
- Orchestrer plusieurs transformations à l'aide de jobs

## Contexte

Le jeu de données utilisé concerne le Tour de France Cycliste et comprend des informations sur les coureurs, les équipes, les étapes et les pays participants. Ces données sont stockées dans différents formats et doivent être intégrées dans une base de données MySQL pour permettre des analyses et des traitements ultérieurs.

## Structure du rapport

Ce rapport est organisé en plusieurs chapitres correspondant aux différentes phases du projet :

- Configuration de l'environnement de travail
- Création et configuration des bases de données

- Réalisation des transformations ETL individuelles
- Orchestration globale avec les jobs
- Conclusion et perspectives

# Chapitre 1

## Présentation générale du projet

### 1.1 Contexte et données du projet

#### 1.1.1 Description du jeu de données

Le jeu de données utilisé dans le cadre de cet atelier est volontairement simple afin de faciliter l'apprentissage des concepts ETL. Il comprend des données liées au Tour de France Cycliste, organisées dans plusieurs tables.

#### Structure de la base de données source (BD\_S)

La base de données MySQL BD\_S contient 5 tables principales :

1. **COUREUR** : Informations sur les coureurs participants

- Dossard\_Coureur
- Identite\_Coureur
- Code\_Pays\_Coureur
- Code\_Equipe\_Coureur

2. **EQUIPE** : Informations sur les équipes

- Code\_Equipe
- Nom\_Equipe

3. **EQUIPE\_BUDGET** : Données budgétaires des équipes

- Code\_Equipe
- Nom\_Equipe
- Budget\_Equipe
- NB\_Coureur\_Equipe

4. **ETAPE** : Informations sur les étapes du Tour

- Num\_Etape

- Ville\_Depart\_Etape
  - Ville\_Arrivee\_Etape
5. **PAYS** : Liste des pays participants
- Code\_Pays
  - Nom\_Pays

### 1.1.2 Sources de données

Les données proviennent de différentes sources, illustrant la diversité des formats rencontrés dans un contexte ETL réel :

- **Fichiers Excel** : Pays.xls contenant la liste des pays
- **Fichiers texte délimité** : Equipe.txt avec les informations des équipes
- **Fichiers XML** : Etape.xml décrivant les étapes du Tour
- **Bases de données MySQL** : Tables pré-remplies pour certaines données

## 1.2 Outils utilisés

### 1.2.1 Pentaho Data Integration (PDI/Spoon)

Pentaho Data Integration, également appelé Spoon, est un outil open-source puissant pour la conception et l'exécution de processus ETL. Il offre :

- Une interface graphique intuitive de type glisser-déposer
- Une large gamme de composants (steps) pour l'extraction, la transformation et le chargement de données
- La possibilité d'orchestrer plusieurs transformations via des jobs
- Un support natif de nombreux formats de données et systèmes de bases de données

### 1.2.2 MySQL et MySQL Workbench

MySQL est utilisé comme système de gestion de bases de données relationnelles pour stocker les données sources et cibles. MySQL Workbench sert d'interface pour :

- Créer et gérer les schémas de bases de données
- Exécuter des scripts SQL
- Vérifier l'intégrité des données après les transformations
- Effectuer des requêtes de validation

## 1.3 Organisation du projet

### 1.3.1 Architecture des dossiers

Le projet est organisé selon l'arborescence suivante :

Projet\_ETL\_TourDeFrance/

  Data\_Sources/

    Equipe.txt

    Etape.xml

    Export\_PAYS.xls

    Pays.xls

  Jobs/

    J00\_Main\_Job.kjb

    J01\_Sec\_Job.kjb

  SQL/

    Create\_Schema\_BD\_S.sql

    Populate\_BD\_S.sql

  Transformations/

    T00\_Load\_PAYS.ktr

    T01\_Load\_EQUIPE.ktr

    T02\_Export\_PAYS\_to\_Excel.ktr

    T03\_Load\_ETAPE.ktr

    T04\_Calcul\_BUDGET.ktr

    T05\_Load\_PAYS\_CIBLE.ktr

    T06\_Jointure\_Coureur\_Equipe.ktr

  README.md

Cette organisation permet une gestion claire et modulaire du projet, facilitant la maintenance et l'évolution.

# Chapitre 2

## Configuration de l'environnement

### 2.1 Installation des outils

#### 2.1.1 Prérequis

Avant de commencer le projet, il est nécessaire de vérifier la présence des outils suivants :

- **Pentaho Data Integration** (version 8.x ou supérieure)
- **MySQL Server** (version 5.7 ou supérieure)
- **MySQL Workbench** pour l'administration graphique
- **Pilote JDBC MySQL Connector/J** (doit être placé dans le dossier `PDI_HOME/lib`)
- **Java JDK** (version 8 ou supérieure)

#### 2.1.2 Vérification de l'installation PDI

Après l'installation de Pentaho Data Integration, il est important de vérifier :

- Le bon lancement de l'interface Spoon
- La présence du driver JDBC MySQL dans le dossier `lib`
- Les droits d'écriture dans les répertoires du projet

### 2.2 Configuration des bases de données

#### 2.2.1 Crédation du schéma BD\_S (Base de données Source)

La première étape consiste à créer le schéma de la base de données source qui accueillera les données du Tour de France.

## Exécution du script de création

Le fichier `Create_Schema_BD_S.sql` contient les instructions pour créer toutes les tables nécessaires. Ce script définit la structure suivante :

Listing 2.1 – Script de création de la base de données source

```
1 USE BD_S;

2

3 -- Structure de la table 'coureur'
4 CREATE TABLE IF NOT EXISTS 'coureur' (
5     'DOSSARD_COUREUR' char(3) NOT NULL DEFAULT '',
6     'IDENTITE_COUREUR' char(40) NOT NULL DEFAULT '',
7     'CODE_PAYS_COUREUR' char(3) NOT NULL DEFAULT '',
8     'CODE_EQUIPE_COUREUR' char(3) NOT NULL DEFAULT ''
9 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

10

11 -- Structure de la table 'equipe'
12 CREATE TABLE IF NOT EXISTS 'equipe' (
13     'CODE_EQUIPE' char(3) NOT NULL DEFAULT '',
14     'NOM_EQUIPE' char(20) DEFAULT NULL,
15     PRIMARY KEY ('CODE_EQUIPE')
16 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

17

18 -- Structure de la table 'equipe_budget'
19 CREATE TABLE IF NOT EXISTS 'equipe_budget' (
20     'CODE_EQUIPE' char(3) NOT NULL DEFAULT '',
21     'NOM_EQUIPE' char(20) DEFAULT NULL,
22     'BUDGET_EQUIPE' double NOT NULL,
23     'NB_COUREURS_EQUIPE' int(11) NOT NULL,
24     PRIMARY KEY ('CODE_EQUIPE')
25 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

26

27 -- Structure de la table 'etape'
28 CREATE TABLE IF NOT EXISTS 'etape' (
29     'NUM_ETAPE' int(10) unsigned NOT NULL DEFAULT '0',
30     'VILLE_DEPART' char(30) DEFAULT NULL,
31     'VILLE_ARRIVEE' char(30) DEFAULT NULL,
32     PRIMARY KEY ('NUM_ETAPE')
33 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

34

35 -- Structure de la table 'pays'
36 CREATE TABLE IF NOT EXISTS 'pays' (
```

```

37   'CODE_PAYS' char(3) NOT NULL DEFAULT '' ,
38   'NOM_PAYS' char(20) DEFAULT NULL ,
39   PRIMARY KEY ('CODE_PAYS')
40 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

```

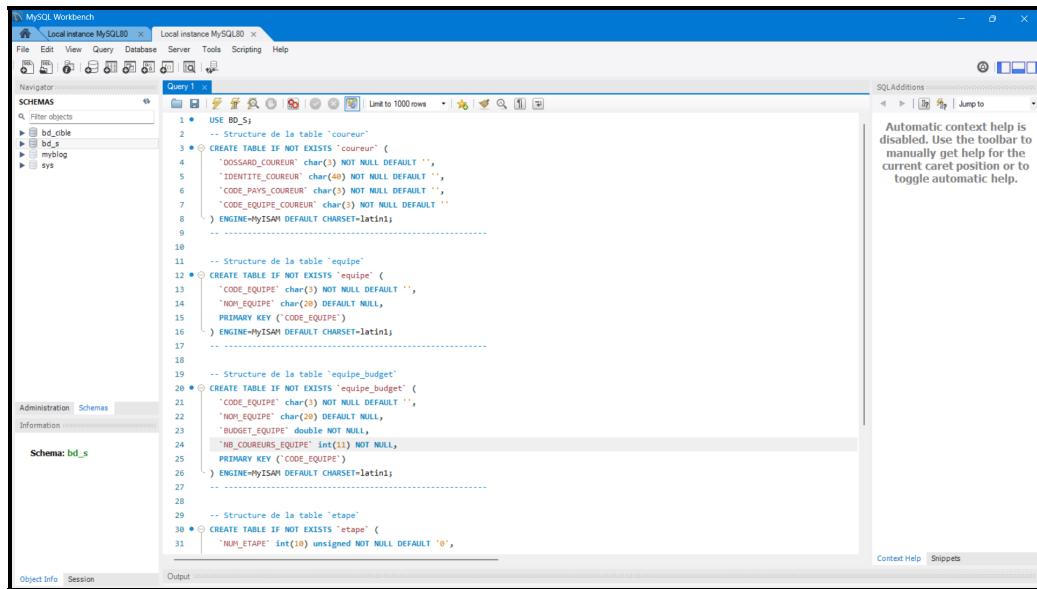


FIGURE 2.1 – Exécution du script de création des tables dans MySQL Workbench

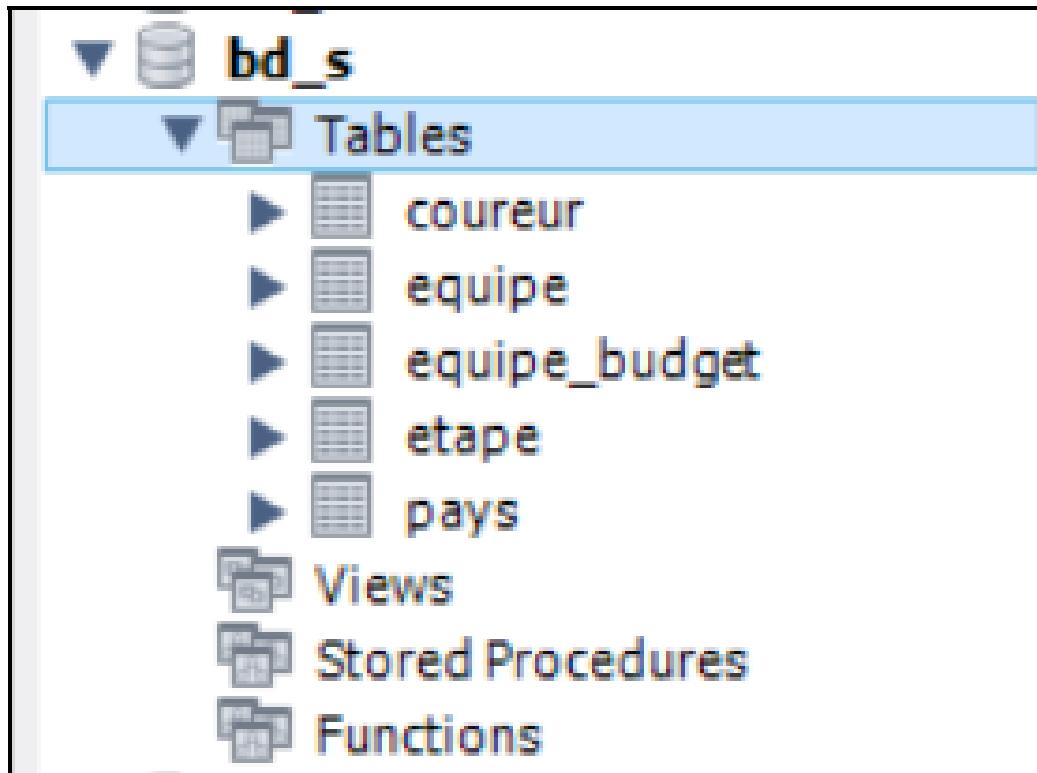


FIGURE 2.2 – Vérification de la création des tables dans le schéma BD\_S

## Résultat

Les 5 tables (`coureur`, `equipe`, `equipe_budget`, `etape`, `pays`) ont été créées avec succès dans le schéma BD\_S.

### 2.2.2 Création du schéma BD\_CIBLE (Base de données Cible)

De la même manière, un schéma BD\_CIBLE a été créé pour accueillir les données transformées. Ce schéma recevra les résultats des différentes transformations, notamment les données triées et les jointures.

## 2.3 Configuration des connexions dans Pentaho PDI

### 2.3.1 Lancement de l'interface Spoon

Après avoir lancé Pentaho Data Integration, l'interface graphique Spoon s'affiche. C'est depuis cette interface que nous allons créer nos transformations et configurer nos connexions aux bases de données.

### 2.3.2 Création de la connexion à BD\_S

Pour établir la connexion à la base de données source :

1. Dans le panneau de gauche, faire un clic droit sur **Database connections**
2. Sélectionner **New**
3. Configurer les paramètres de connexion :
  - Connection Name : Conn\_BD\_S
  - Connection Type : MySQL
  - Access : Native (JDBC)
  - Host Name : localhost
  - Database Name : BD\_S
  - Port Number : 3306
  - Username : root (ou votre utilisateur MySQL)
  - Password : [votre mot de passe]
4. Cliquer sur **Test** pour vérifier la connexion

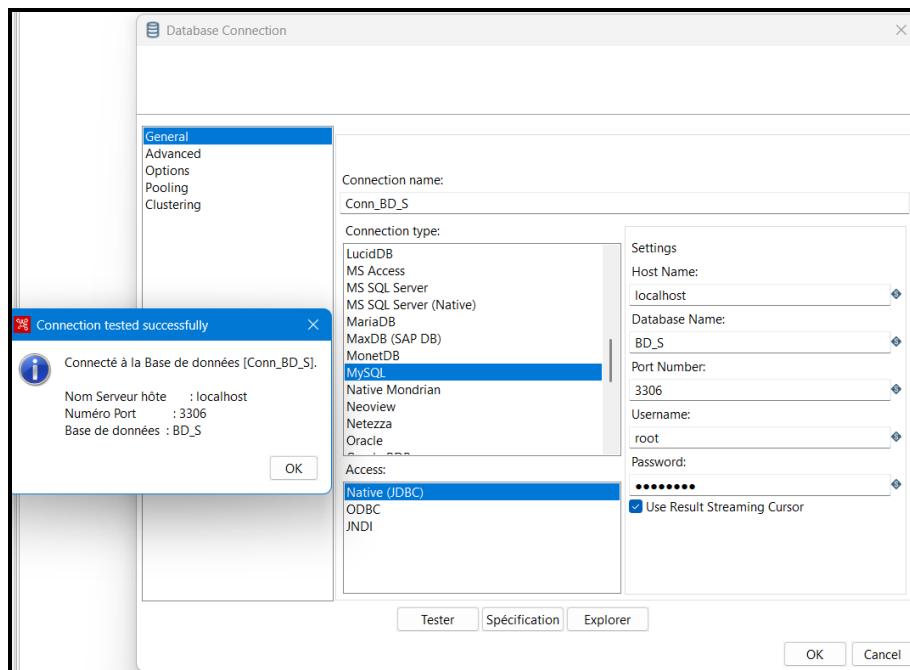


FIGURE 2.3 – Configuration de la connexion Conn\_BD\_S

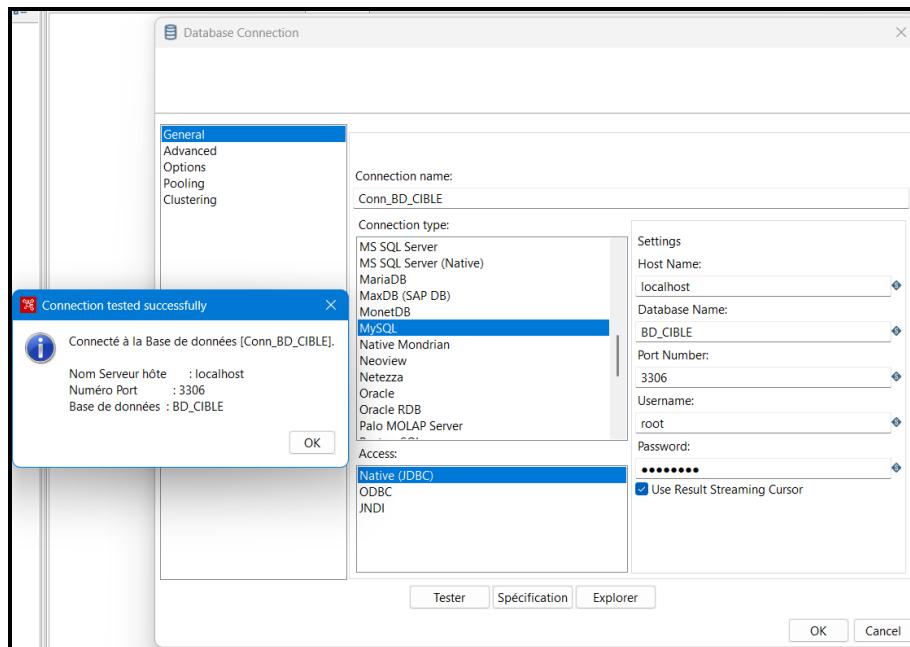


FIGURE 2.4 – Test réussi de la connexion à BD\_S

### 2.3.3 Crédation de la connexion à BD\_CIBLE

Le même processus est répété pour créer la connexion Conn\_BD\_CIBLE pointant vers la base de données cible.

### Configuration identique

Les paramètres de connexion sont identiques à ceux de Conn\_BD\_S, seul le nom de la base de données change pour BD\_CIBLE.

# Chapitre 3

## Réalisation des transformations ETL

Ce chapitre détaille la création et l'exécution de chaque transformation ETL. Chaque section présente les objectifs, la configuration et les résultats obtenus.

### 3.1 T00 : Chargement des données PAYS depuis Excel

#### 3.1.1 Objectif

Cette première transformation a pour but d'extraire les données des pays depuis un fichier Excel (`Pays.xls`) et de les charger dans la table `pays` de la base de données `BD_S`.

#### 3.1.2 Création de la transformation

1. Créer une nouvelle transformation : **File → New → Transformation**
2. Sauvegarder sous le nom : `T00_Load_PAYS.ktr` dans le dossier Transformations

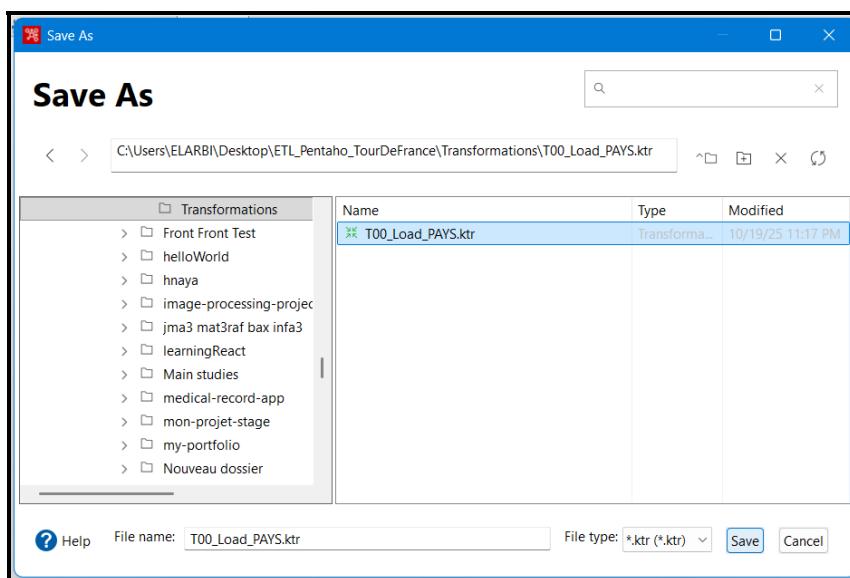


FIGURE 3.1 – Création et enregistrement de la transformation T00

### 3.1.3 Configuration des composants

#### Étape 1 : Microsoft Excel Input

Ce composant permet d'extraire les données du fichier Excel source.

##### Configuration :

- Emplacement du fichier : Data\_Sources/Pays.xls
- Sélection de la feuille contenant les données
- Récupération automatique des champs depuis l'en-tête

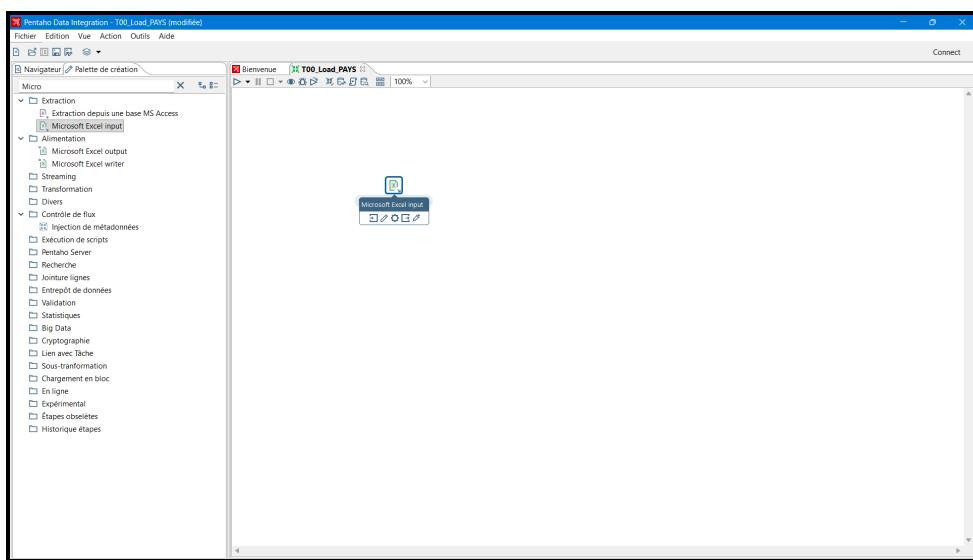


FIGURE 3.2 – Ajout du composant Microsoft Excel Input

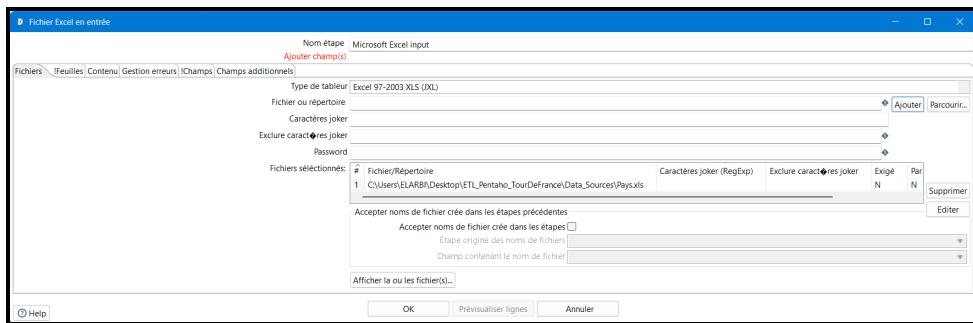


FIGURE 3.3 – Sélection du fichier Pays.xls

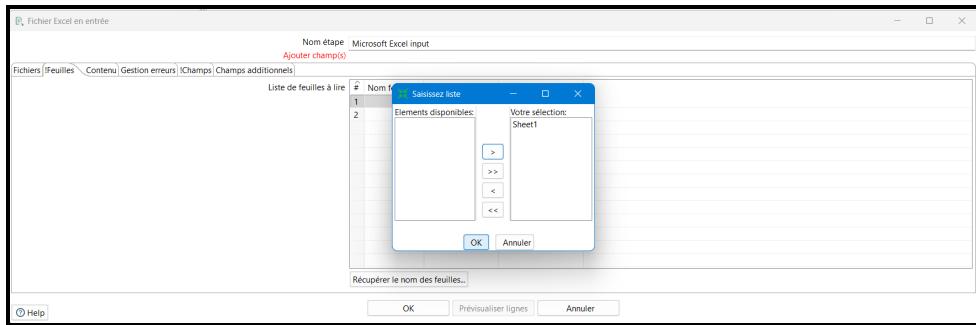


FIGURE 3.4 – Sélection de la feuille de données

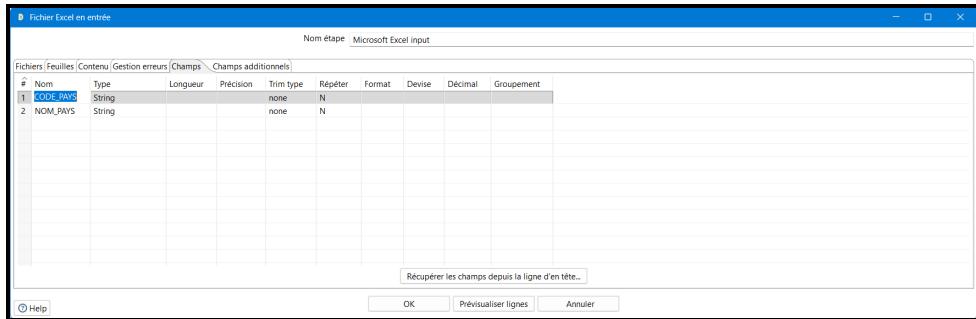


FIGURE 3.5 – Aperçu des colonnes extraites du fichier Excel

## Étape 2 : Filtre pour éliminer les lignes vides

Lors de la première exécution, une erreur s'est produite en raison de lignes avec des codes pays vides, ce qui violait la contrainte NOT NULL de la base de données.

### Erreur rencontrée

Des lignes avec des valeurs CODE\_PAYS vides ont causé l'échec de l'insertion dans la table pays.

**Solution :** Ajout d'un composant **Filter Rows** pour filtrer les lignes où CODE\_PAYS est vide ou NULL.

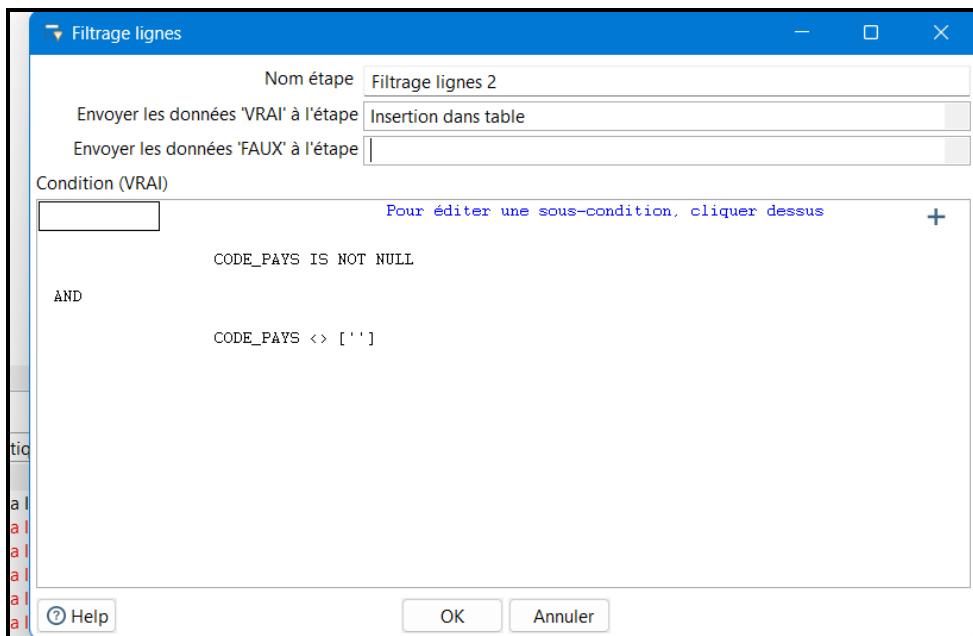


FIGURE 3.6 – Configuration du filtre pour éliminer les lignes invalides

#### Condition du filtre :

```
CODE_PAYS IS NOT NULL AND CODE_PAYS <> ''
```

#### Étape 3 : Insertion dans la table

Le composant **Table Output** permet d'insérer les données filtrées dans la table pays.

#### Configuration :

- Connexion : Conn\_BD\_S
- Table cible : pays
- Mapping des champs vérifié

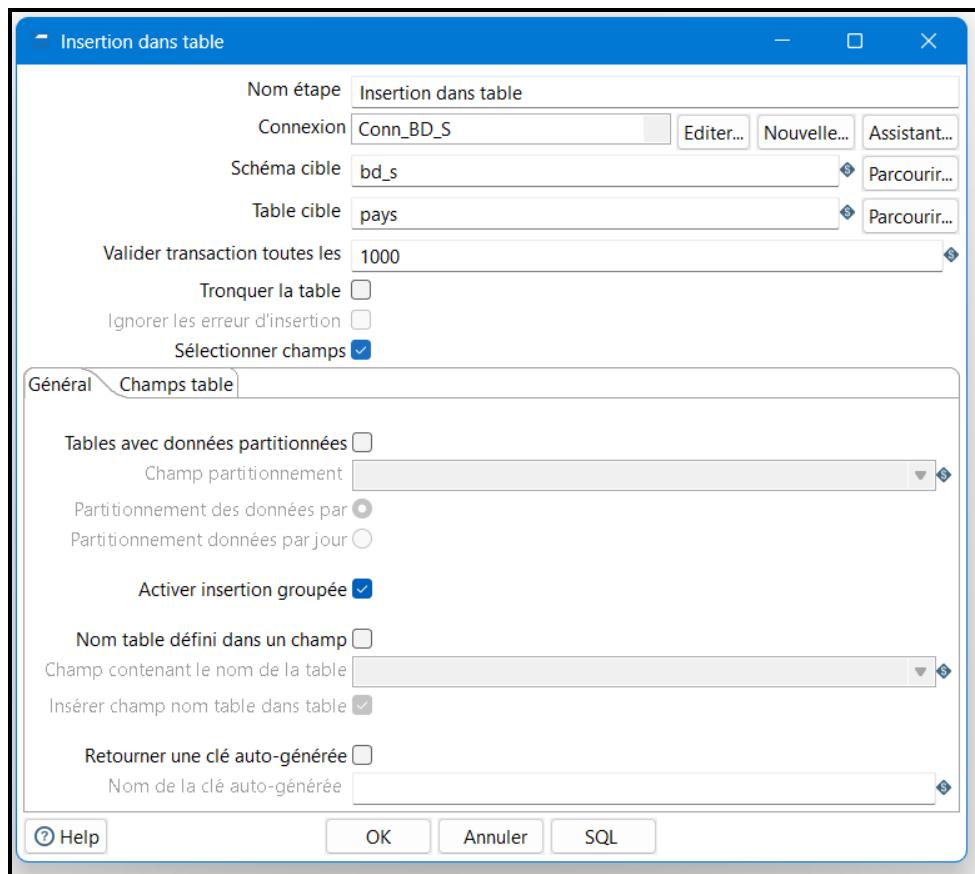


FIGURE 3.7 – Configuration du composant Table Output

### 3.1.4 Flux complet de la transformation

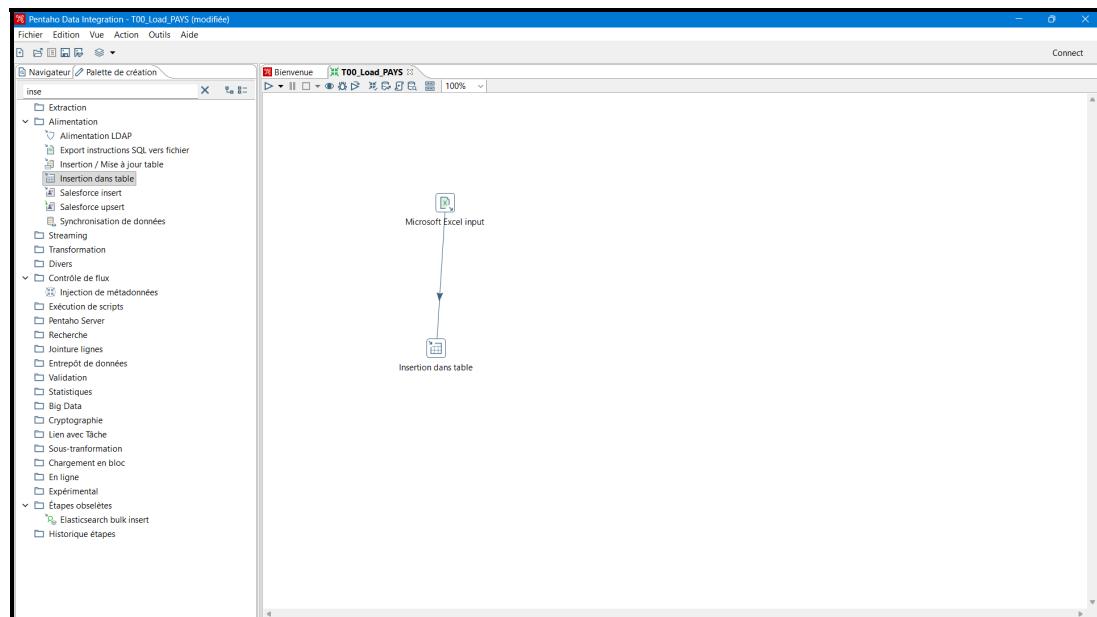


FIGURE 3.8 – Flux complet : Excel Input → Filter → Table Output

### 3.1.5 Exécution et résultats

Après l'ajout du filtre, la transformation s'exécute avec succès.

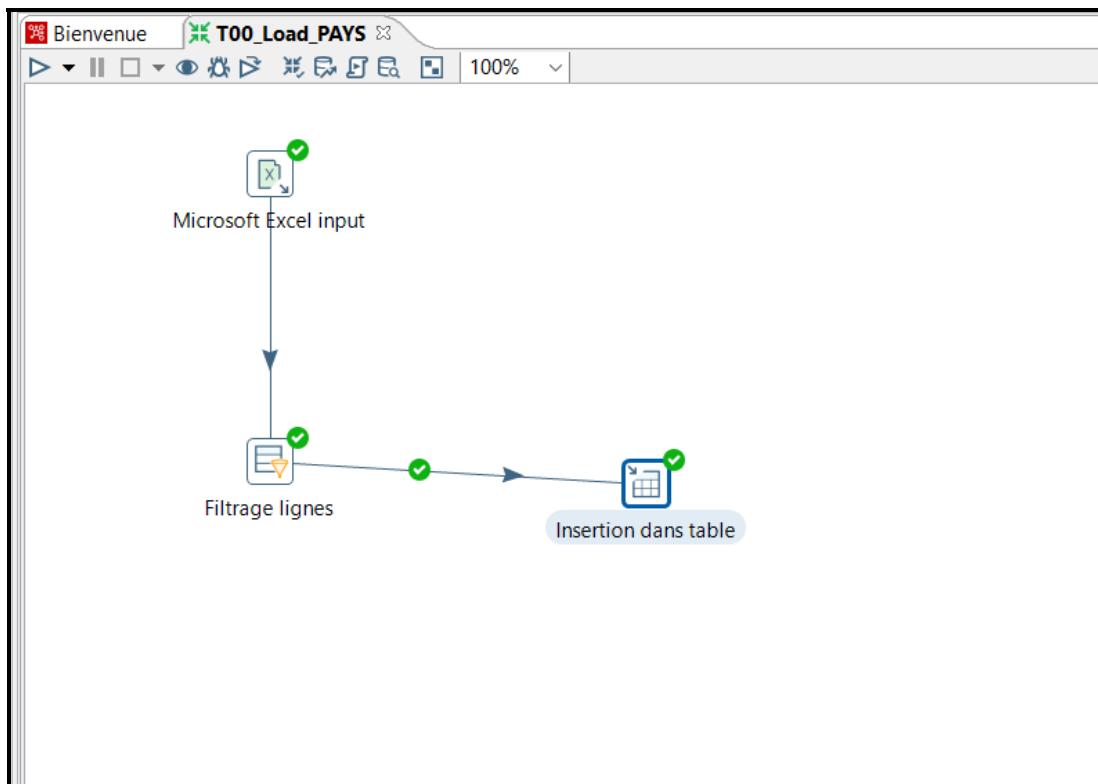


FIGURE 3.9 – Exécution réussie de la transformation T00

CODE_PAYs	NOM_PAYs
BEL	Belgique
COL	Colombie
CRO	Croatie
CZE	Macédoine/Tchéquie
DEN	Danemark
ESP	Espagne
EST	Estonie
FRA	France
GBR	Grande-Bretagne
GER	Allemagne
HUN	Hongrie
ITA	Italie
KAZ	Kazakhstan
LAT	Lettanie
NED	Pays-Bas
NOR	Norvège
POK	Porko-Pok
RSA	Afrique du Sud
RUS	Russie
SLO	Slovénie
SUI	Suisse
UKR	Ukraine
USA	USA
VEN	Vénézuela

FIGURE 3.10 – Vérification des données dans la table pays de MySQL

### Résultat

26 lignes de données pays ont été insérées avec succès dans la table **pays** de la base de données BD\_S.

### 3.1.6 Amélioration pour la ré-exécutabilité

Pour permettre la ré-exécution de la transformation sans erreur de clé primaire dupliquée, le composant **Table Output** a été remplacé par **Insert / Update**.

**Configuration de Insert / Update :**

- Champ clé : CODE\_PAYS
- Action : INSERT si la clé n'existe pas, UPDATE si elle existe

Cela permet d'exécuter la transformation plusieurs fois sans avoir à vider la table au préalable.

## 3.2 T01 : Chargement des données EQUIPE depuis un fichier texte

### 3.2.1 Objectif

Extraire les données des équipes depuis un fichier texte délimité (`Equipe.txt`) et les charger dans la table `equipe` de `BD_S`.

### 3.2.2 Création de la transformation

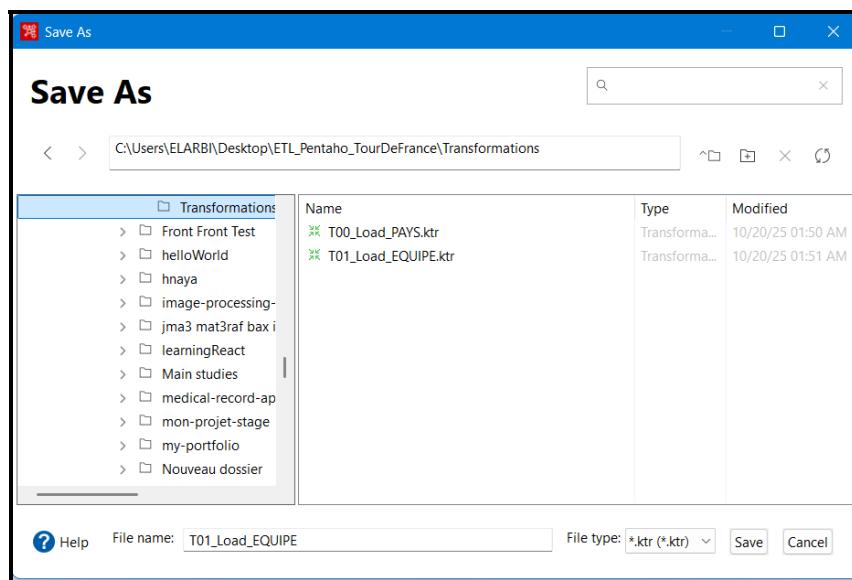


FIGURE 3.11 – Crédit de la transformation T01\_Load\_EQUIPE

### 3.2.3 Configuration des composants

#### Text File Input

Ce composant permet de lire des fichiers texte avec différents formats (délimité, largeur fixe, etc.).

##### Configuration :

- Fichier : `Data_Sources/Equipe.txt`
- Séparateur : Tabulation (ou point-virgule selon le format)
- Champs définis :
  - `CODE_EQUIPE` (String, 3 caractères)
  - `NOM_EQUIPE` (String, 40 caractères)

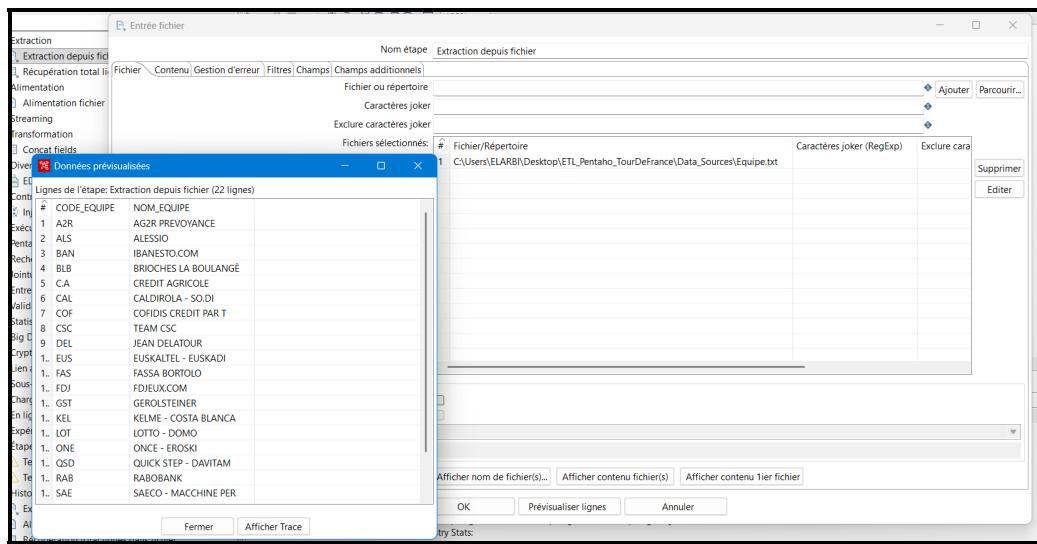


FIGURE 3.12 – Configuration du Text File Input avec aperçu des données

## Table Output

Insertion directe dans la table `équipe`.

### 3.2.4 Exécution et validation

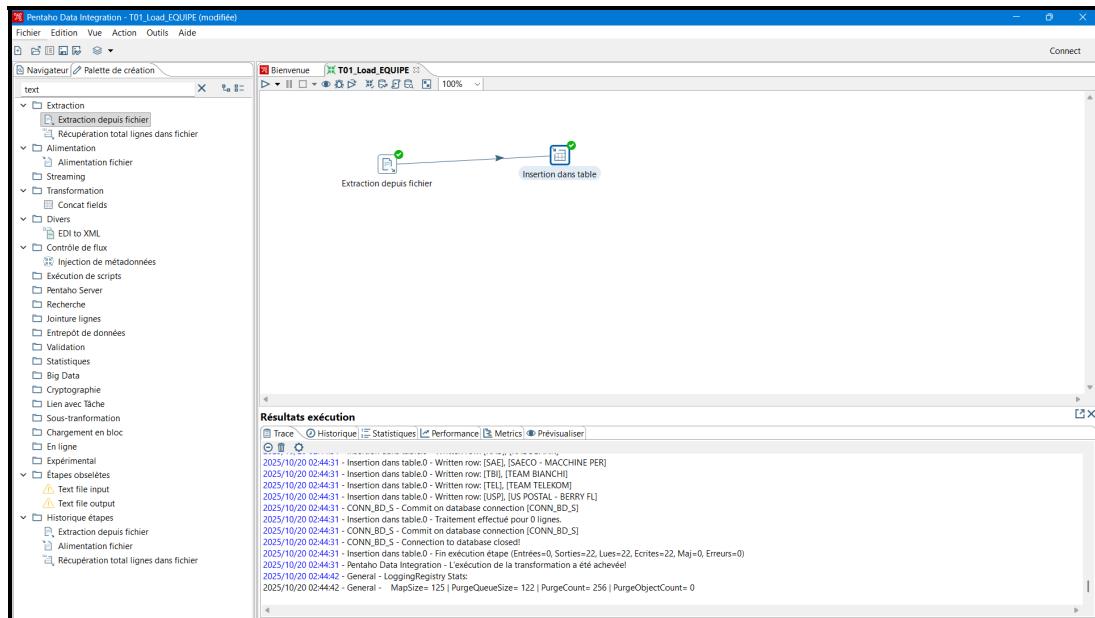


FIGURE 3.13 – Exécution réussie de T01

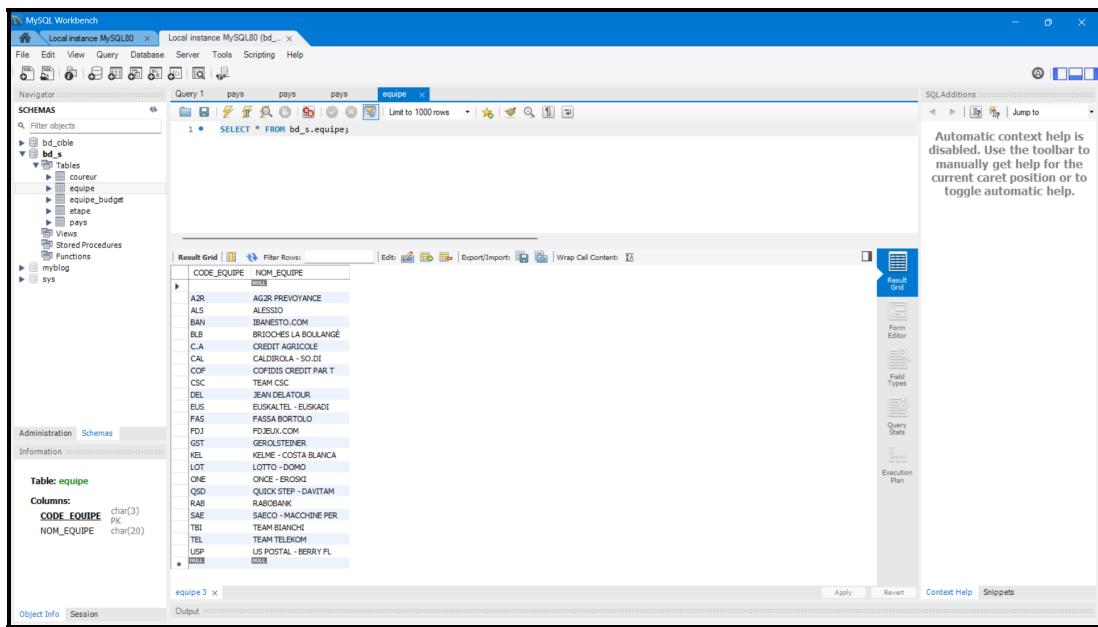


FIGURE 3.14 – Validation : 22 équipes insérées dans la table equipe

## Résultat

22 lignes ont été lues depuis le fichier Equipe.txt et insérées avec succès dans la table equipe.

### 3.3 T02 : Export des données PAYS vers Excel

#### 3.3.1 Objectif

Cette transformation effectue l'opération inverse : extraire les données de la table pays de la base de données et les exporter vers un fichier Excel.

#### 3.3.2 Création de la transformation

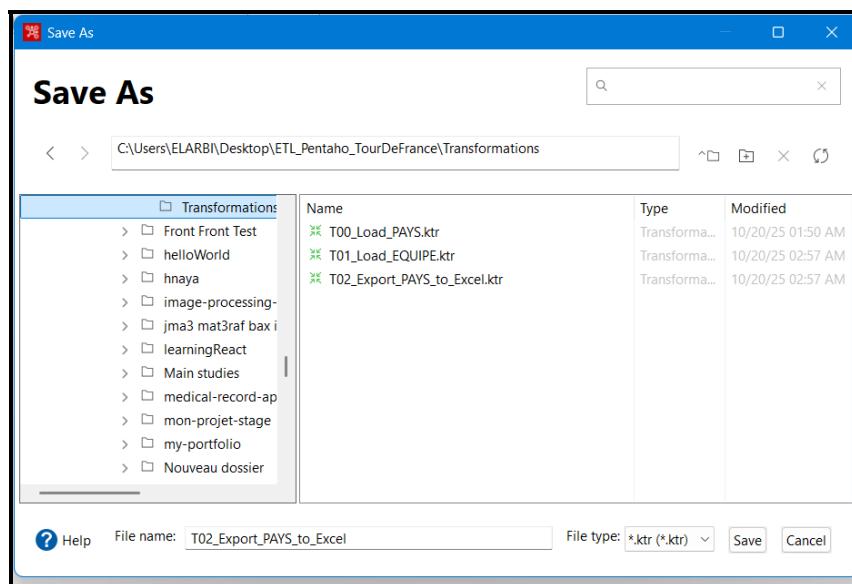


FIGURE 3.15 – Crédit de la transformation T02\_Export\_PAYS\_to\_Excel

#### 3.3.3 Configuration des composants

##### Table Input

Extraction des données depuis la base de données MySQL.

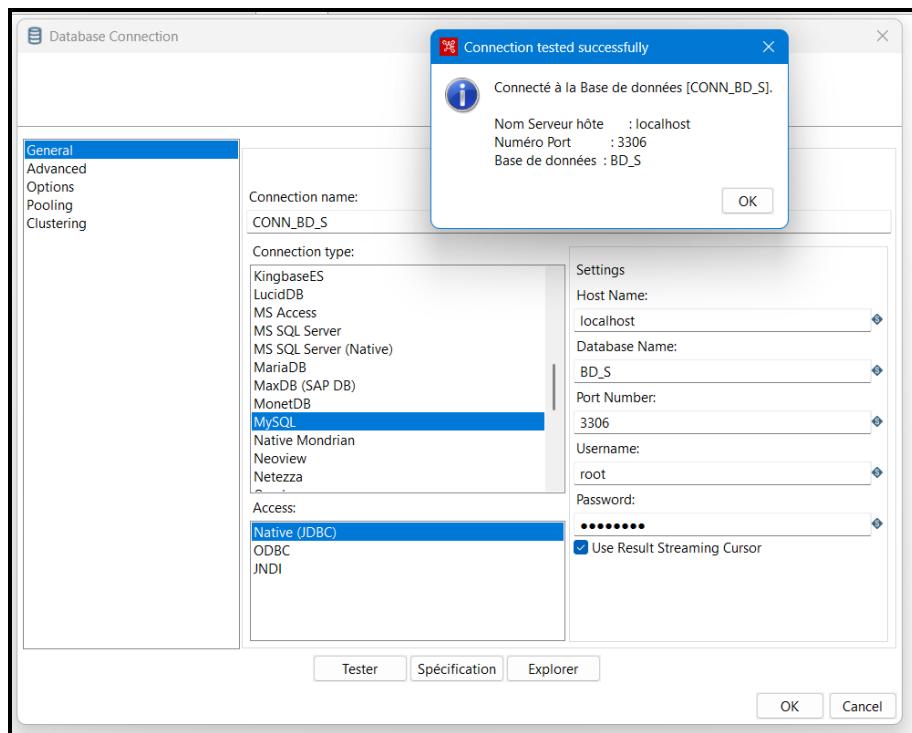


FIGURE 3.16 – Établissement de la connexion à BD\_S

**Requête SQL :**

```
1 SELECT CODE_PAYS , NOM_PAYS
2 FROM pays ;
```

The screenshot shows the 'Extraction Table' dialog box. On the left, a preview window titled 'Données prévisualisées' shows a table with columns '#', 'CODE\_PAYS', and 'NOM\_PAYS', containing 26 rows of country data. On the right, the 'Extraction Table' dialog has a 'Nom étape' field set to 'Extraction PAYS depuis BD\_S' and a 'Connexion' field set to 'CONN\_BD\_S'. The 'SQL' section contains the query: 'SELECT CODE\_PAYS , NOM\_PAYS FROM bd\_s.pays'. Below the SQL are several checkboxes: 'Ligne 1 Colonne 0', 'Store column info in step meta data', 'Repousser conversion de type', 'Remplacer les variables dans le script' (which is checked), 'Insérer données à partir de', 'Exécuter pour chaque ligne', and 'Limite: 0'. Buttons at the bottom include 'Help', 'Fermer', 'Afficher Trace', 'OK', 'Prévisualiser', and 'Annuler'.

FIGURE 3.17 – Configuration de l'extraction avec test de la requête

**Microsoft Excel Output**

Création d'un fichier Excel avec les données extraites.

**Configuration :**

- Fichier de sortie : Data\_Sources/Export\_PAYs.xlsx

- Nom de la feuille : Pays
- Inclure l'en-tête des colonnes : Oui

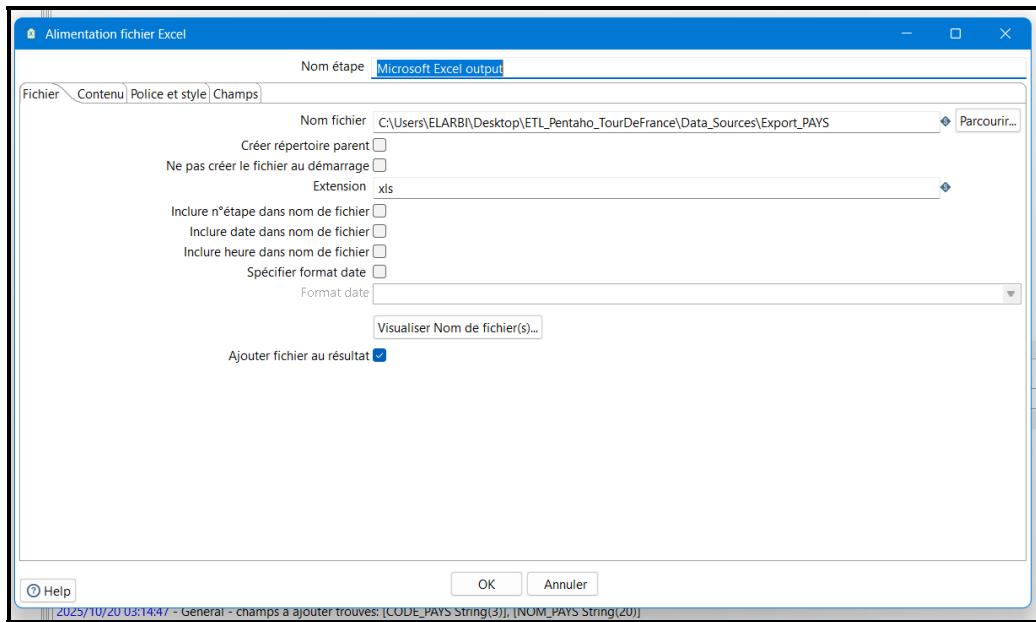


FIGURE 3.18 – Configuration du Microsoft Excel Output

### 3.3.4 Exécution et validation

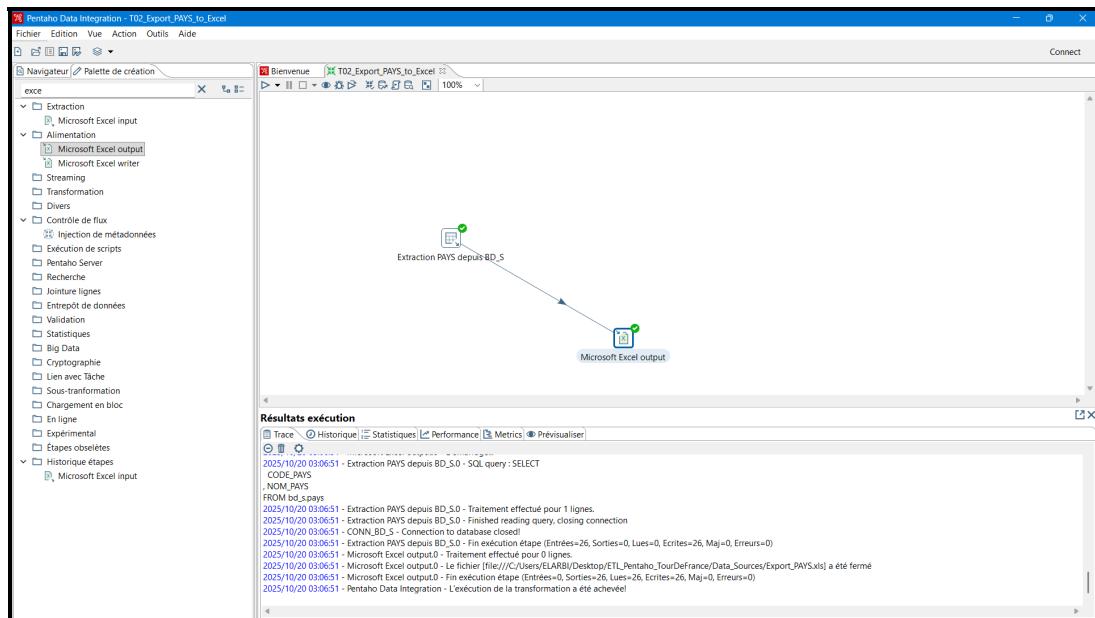


FIGURE 3.19 – Exécution réussie de la transformation T02

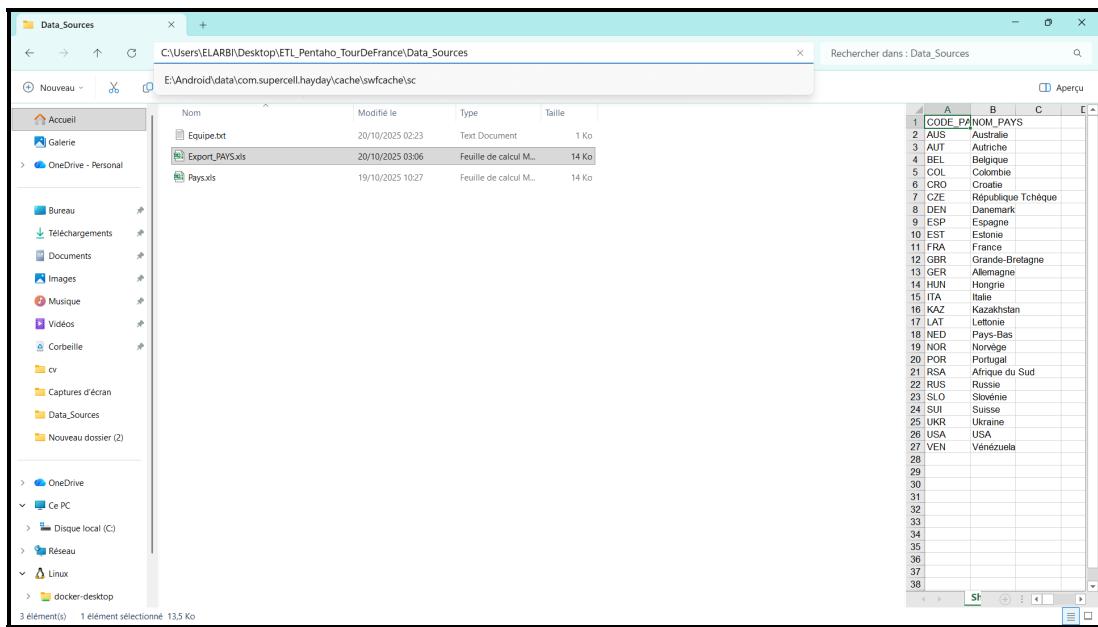


FIGURE 3.20 – Fichier Excel généré avec les données des pays

## Résultat

Le fichier Export\_PAYS.xlsx a été créé avec succès dans le dossier Data\_Sources, contenant les 26 pays extraits de la base de données.

## 3.4 T03 : Chargement des données ETAPE depuis XML

### 3.4.1 Objectif

Cette transformation illustre un cas d'usage important en ETL : la lecture et l'intégration de données structurées au format XML dans une base de données relationnelle.

### 3.4.2 Création de la transformation

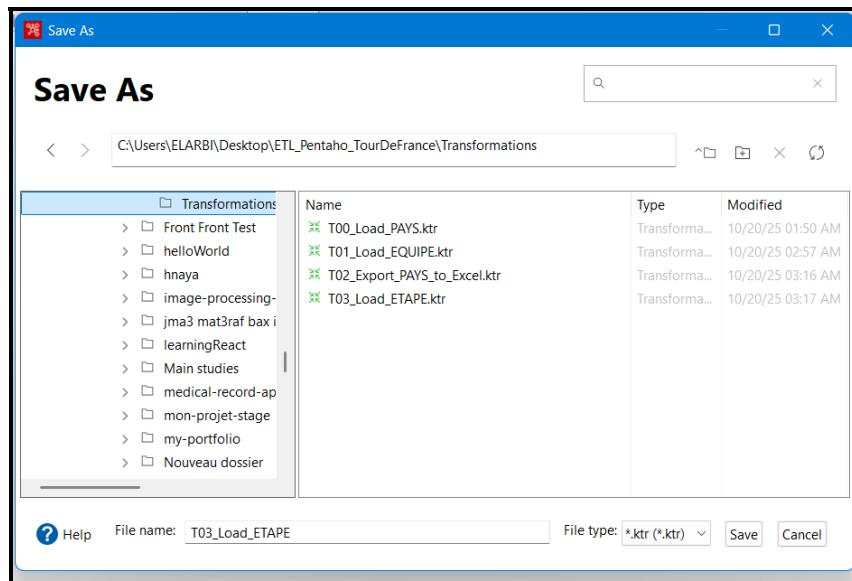


FIGURE 3.21 – Crédit de la transformation T03\_Load\_ETAPE

### 3.4.3 Configuration des composants

#### XML Input

Le composant **Get data from XML** permet de parser des fichiers XML et d'en extraire les données selon une structure définie par XPath.

##### Configuration :

- Fichier source : `Data_Sources/Etape.xml`
- Loop XPath : `/BDD/ETAPE` (définit le nœud de répétition)
- Champs extraits :
  - `NUM_ETAPE` (Integer) - XPath : `NUM_ETAPE`
  - `VILLE_DEPART` (String) - XPath : `VILLE_DEPART`
  - `VILLE_ARRIVEE` (String) - XPath : `VILLE_ARRIVEE`

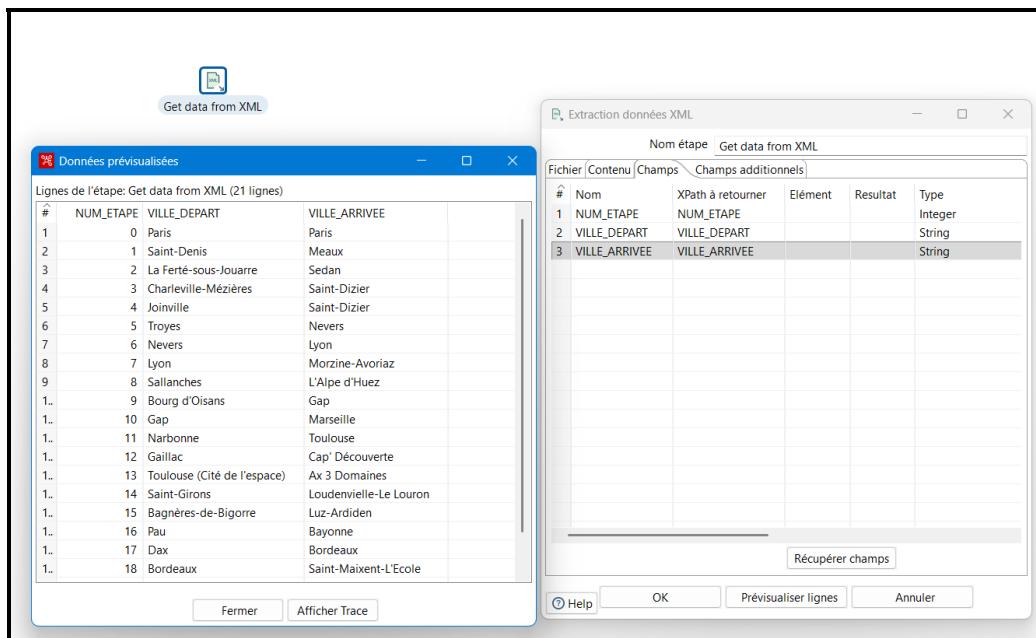


FIGURE 3.22 – Configuration du XML Input avec test d'extraction

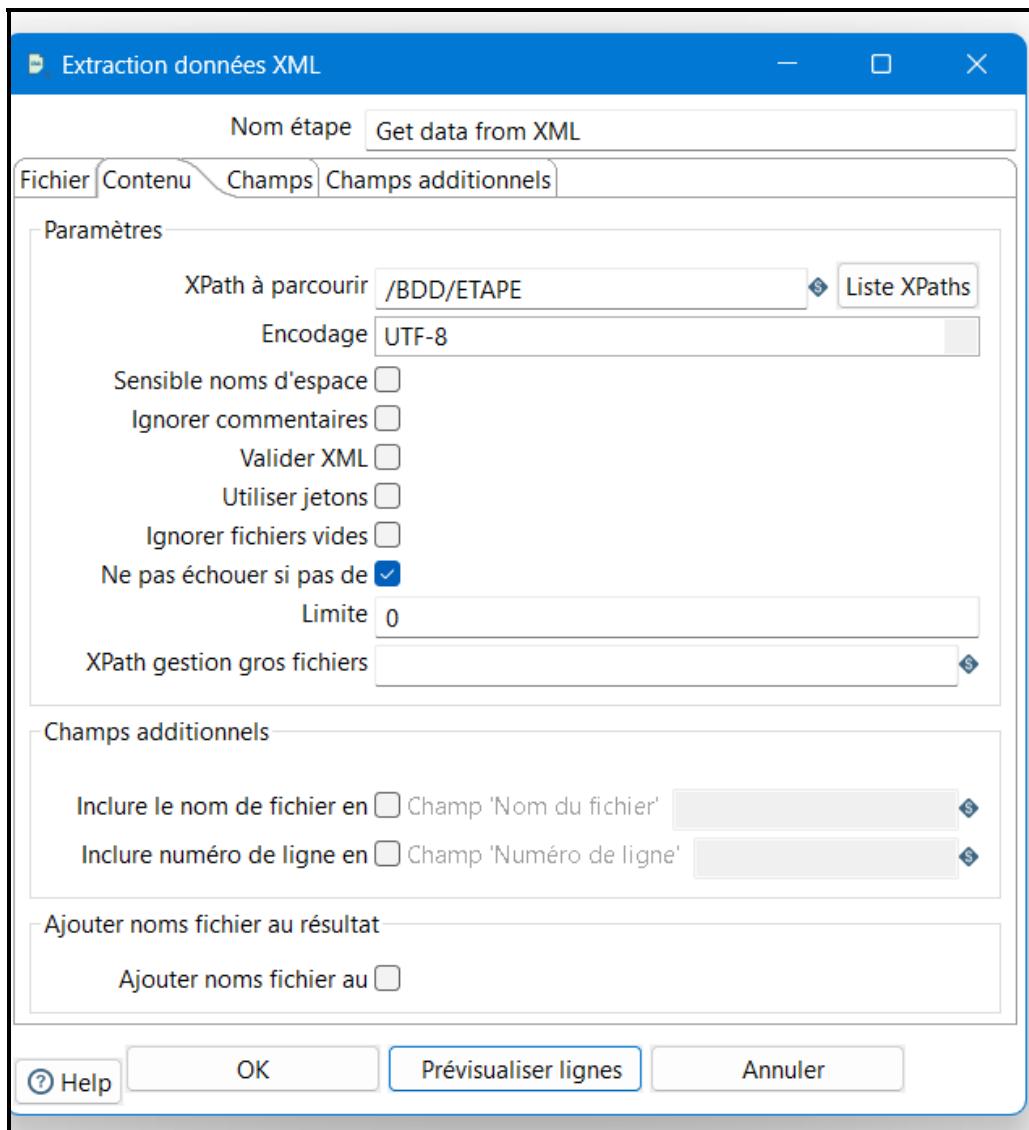


FIGURE 3.23 – Configuration du XPath pour parser le fichier XML



### Structure XML

Le fichier `Etape.xml` suit une structure hiérarchique où chaque étape est un élément `<ETAPE>` contenant les sous-éléments `<NUM_ETAPE>`, `<VILLE_DEPART>` et `<VILLE_ARRIVEE>`.

### Filter Rows

Pour garantir la qualité des données, un filtre est ajouté pour s'assurer qu'aucune ligne avec un numéro d'étape vide ou NULL n'est insérée.

#### Condition :

`NUM_ETAPE IS NOT NULL`

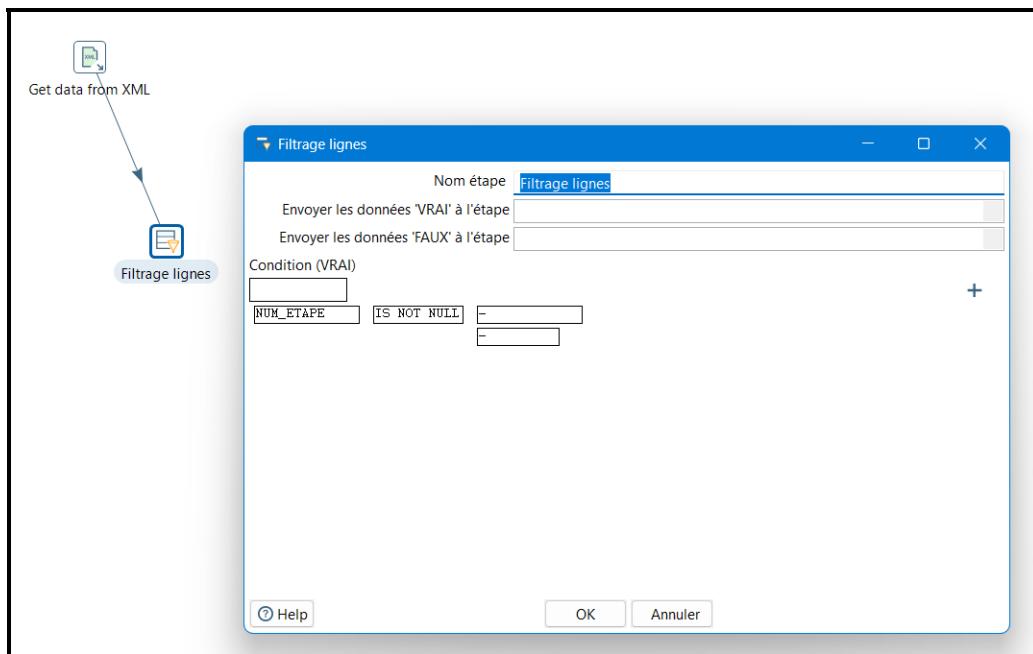


FIGURE 3.24 – Configuration du filtre de contrôle qualité

### Table Output

Insertion des données dans la table `etape`.

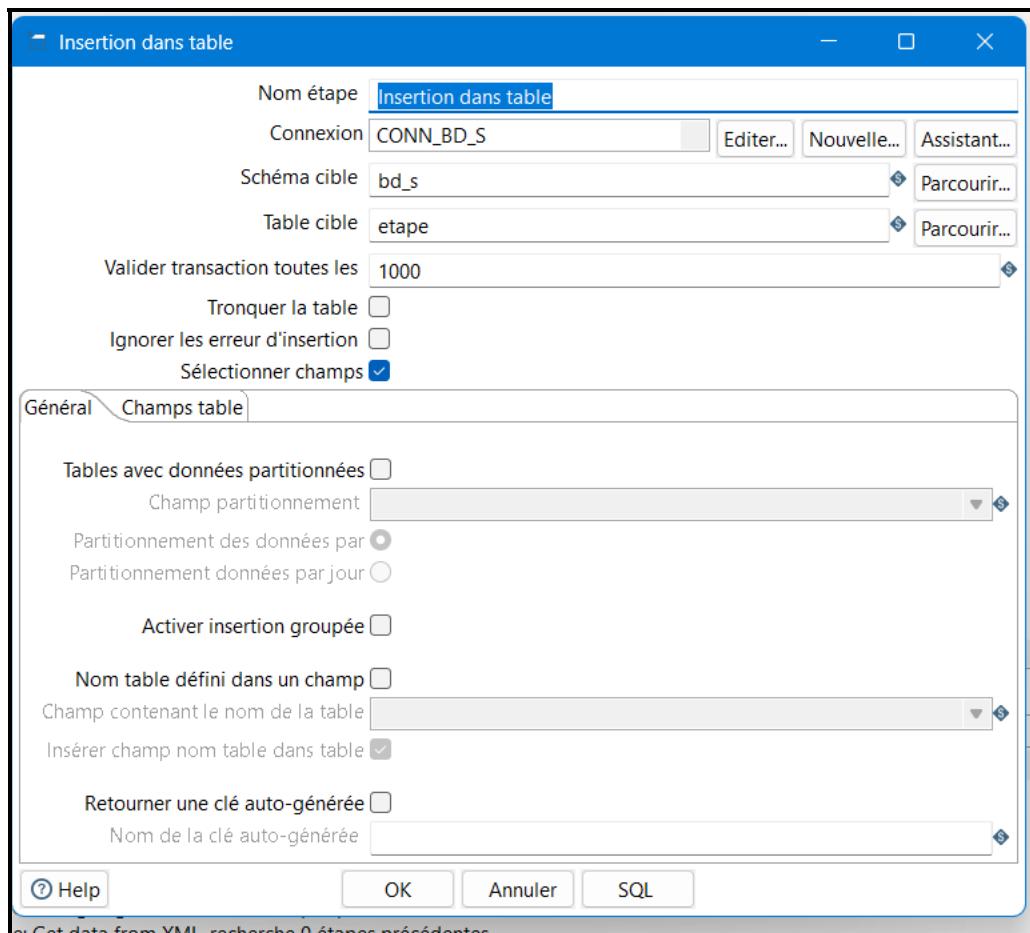


FIGURE 3.25 – Configuration de l’insertion dans la table etape

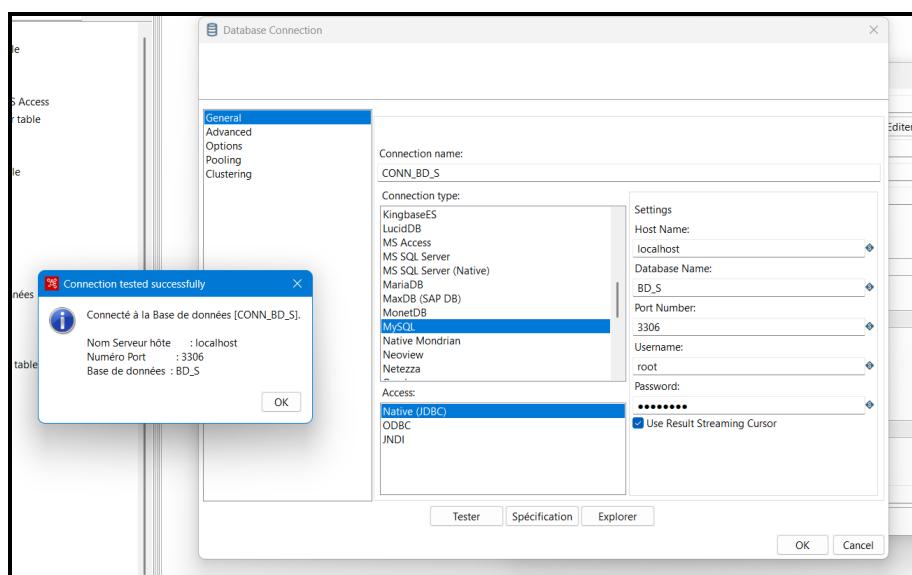


FIGURE 3.26 – Connexion à la base de données BD\_S

### 3.4.4 Exécution et validation

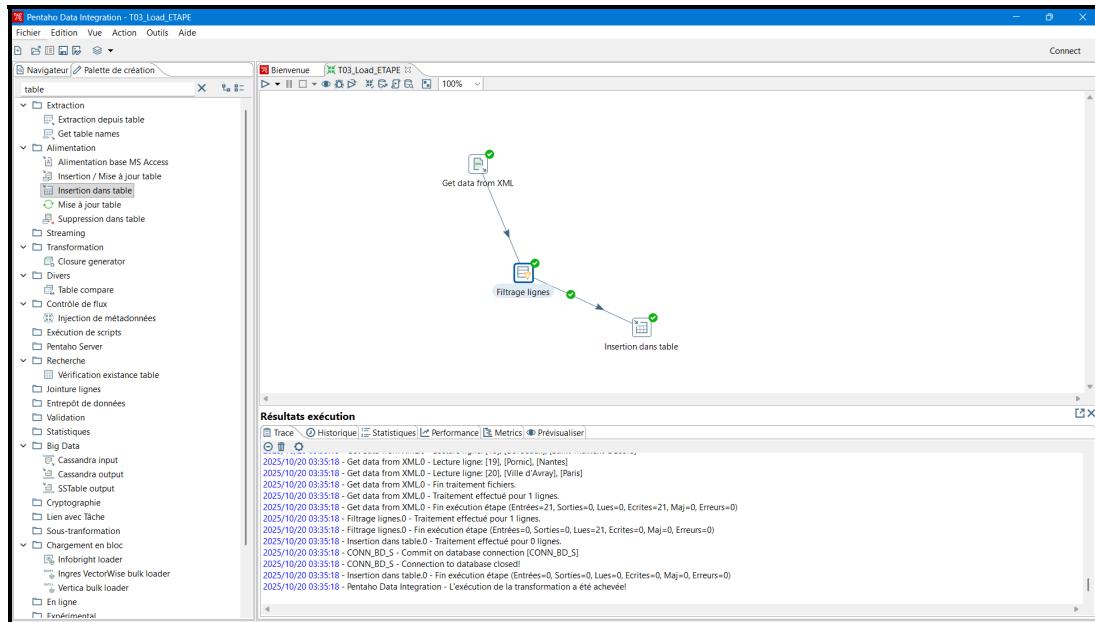


FIGURE 3.27 – Exécution réussie de la transformation T03

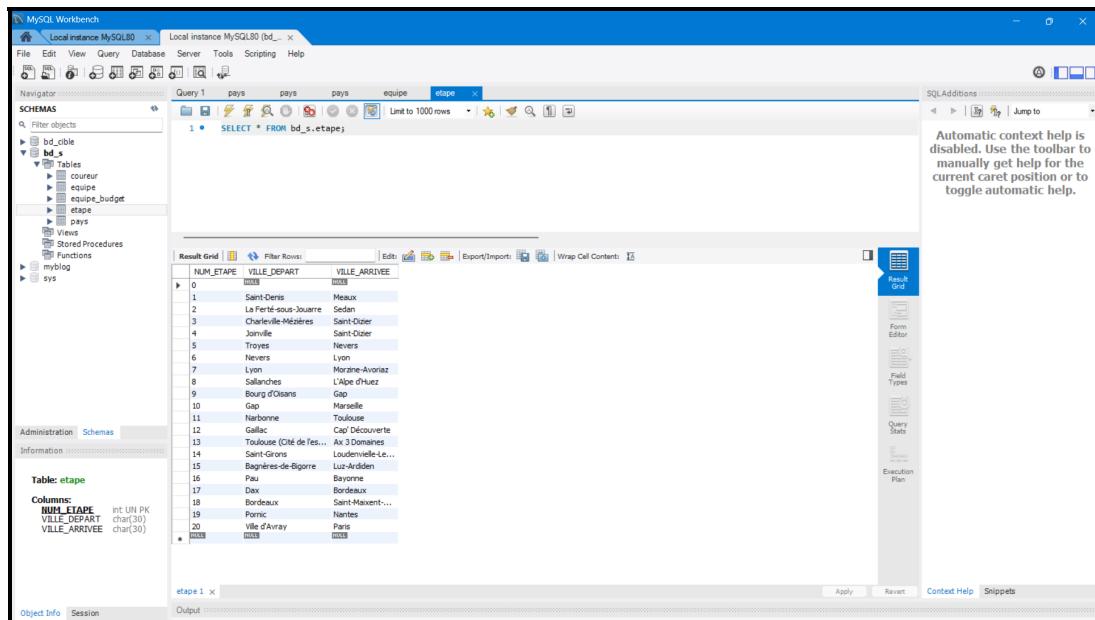


FIGURE 3.28 – Validation : 21 étapes insérées dans la table etape

#### Résultat

21 étapes ont été extraites du fichier XML et insérées avec succès dans la table **etape** de la base de données.

### 3.4.5 Population des autres tables

À ce stade, pour pouvoir continuer avec les transformations suivantes (notamment les calculs et jointures), il est nécessaire de peupler les tables `coureur` et `equipe_budget`. Cela est réalisé en exécutant le script SQL `Populate_BD_S.sql`.

## 3.5 T04 : Calcul du budget par coureur

### 3.5.1 Objectif

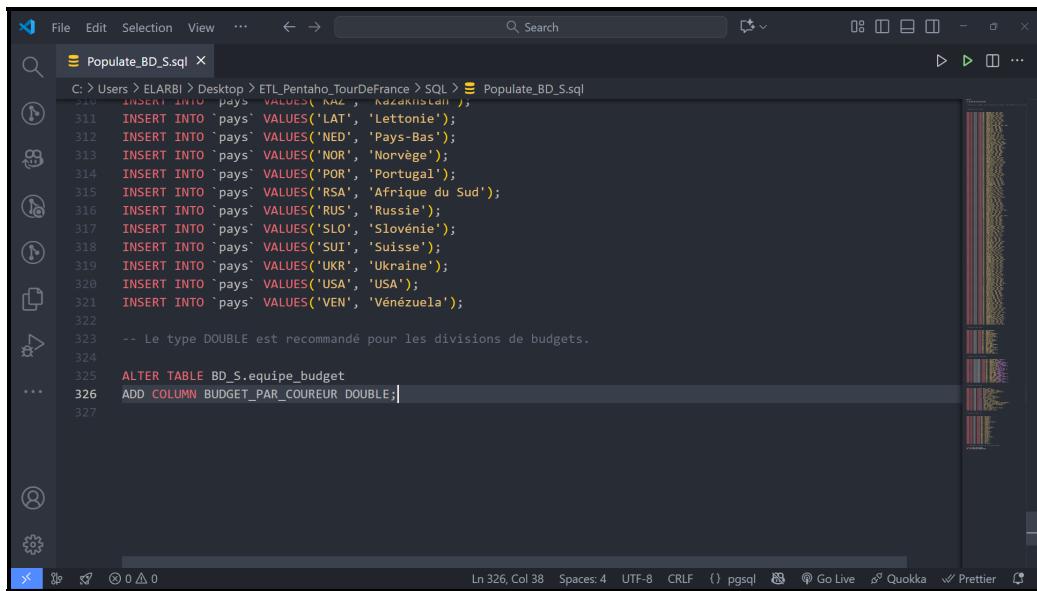
Cette transformation démontre la capacité de PDI à effectuer des calculs sur les données et à mettre à jour des tables existantes. L'objectif est de calculer le budget alloué par coureur pour chaque équipe.

### 3.5.2 Préparation : Ajout de la colonne calculée

Avant de lancer la transformation, il est nécessaire d'ajouter une colonne dans la table `equipe_budget` pour stocker le résultat du calcul.

Commande SQL exécutée :

```
1 ALTER TABLE BD_S.equipe_budget
2 ADD COLUMN BUDGET_PAR_COUREUR DOUBLE;
```



```
C:\> Users > ELARBI > Desktop > ETL_Pentaho_TourDeFrance > SQL > Populate_BD_S.sql
310 INSERT INTO `pays` VALUES('NED', 'Pays-Bas');
311 INSERT INTO `pays` VALUES('LAT', 'Lettanie');
312 INSERT INTO `pays` VALUES('NED', 'Pays-Bas');
313 INSERT INTO `pays` VALUES('NOR', 'Norvège');
314 INSERT INTO `pays` VALUES('POR', 'Portugal');
315 INSERT INTO `pays` VALUES('RSA', 'Afrique du Sud');
316 INSERT INTO `pays` VALUES('RUS', 'Russie');
317 INSERT INTO `pays` VALUES('SLO', 'Slovénie');
318 INSERT INTO `pays` VALUES('SUI', 'Suisse');
319 INSERT INTO `pays` VALUES('UKR', 'Ukraine');
320 INSERT INTO `pays` VALUES('USA', 'USA');
321 INSERT INTO `pays` VALUES('VEN', 'Vénézuela');
322
323 -- Le type DOUBLE est recommandé pour les divisions de budgets.
324
325 ALTER TABLE BD_S.equipe_budget
326 ADD COLUMN BUDGET_PAR_COUREUR DOUBLE;
327
```

FIGURE 3.29 – Ajout de la colonne BUDGET\_PAR\_COUREUR

### 3.5.3 Configuration des composants

#### Table Input

Extraction des données nécessaires au calcul.

Requête SQL :

```
1 SELECT CODE_EQUIPE ,
2     BUDGET_EQUIPE ,
3     NB_COUREURS_EQUIPE
4 FROM equipe_budget;
```

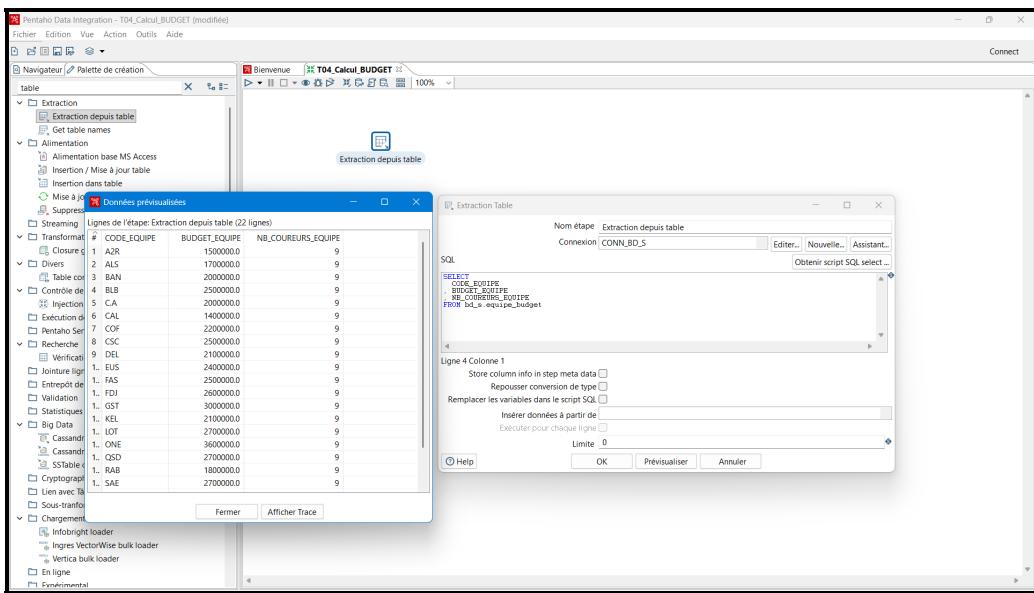


FIGURE 3.30 – Configuration de l'extraction des données budgétaires

## Calculator

Le composant **Calculator** permet d'effectuer des opérations mathématiques sur les champs.

### Configuration du calcul :

- Nouveau champ : BUDGET\_PAR\_COUREUR
- Calcul : A / B
- Champ A : BUDGET\_EQUIPE
- Champ B : NB\_COUREURS\_EQUIPE
- Type : Number

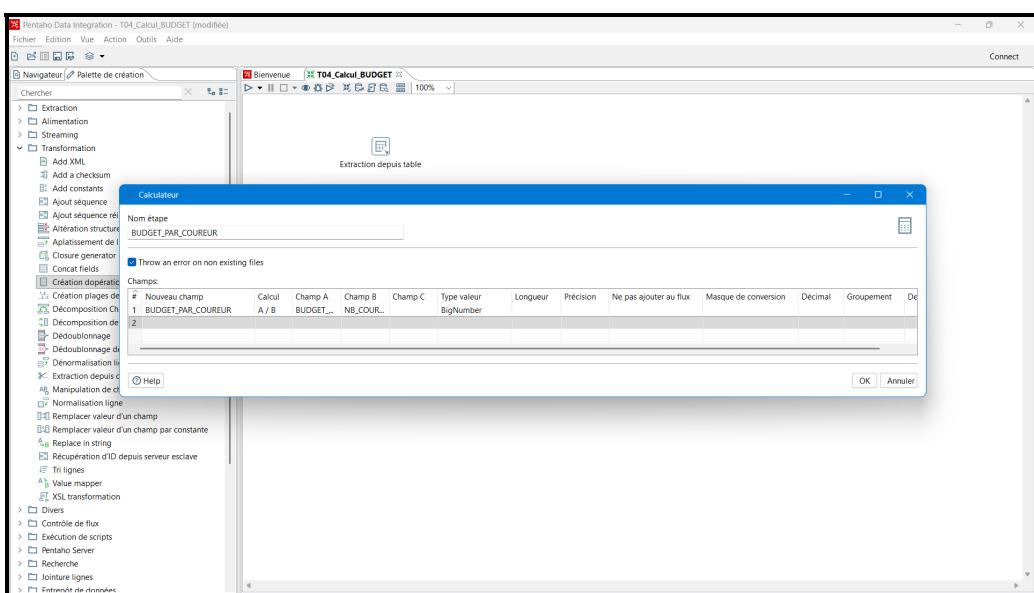


FIGURE 3.31 – Configuration du Calculator pour le calcul du budget par coureur

## Insert / Update

Ce composant permet de mettre à jour les lignes existantes plutôt que d'en insérer de nouvelles.

### Configuration :

- Connexion : Conn\_BD\_S
- Table : equipe\_budget
- Champ clé de recherche : CODE\_EQUIPE
- Champ à mettre à jour : BUDGET\_PAR\_COUREUR

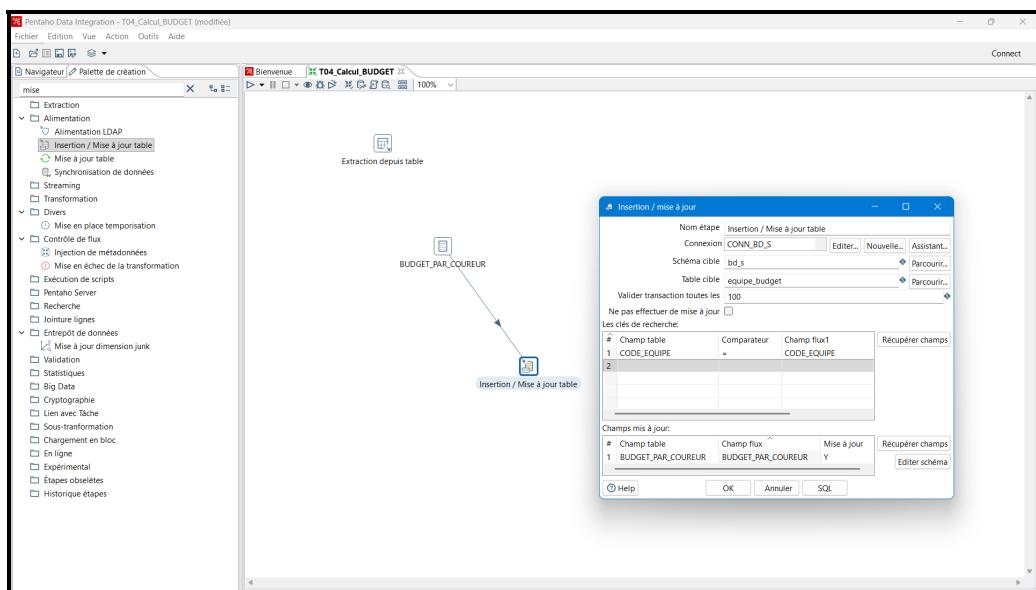


FIGURE 3.32 – Configuration du composant Insert / Update

### 3.5.4 Exécution et validation

The screenshot shows the MySQL Workbench interface with the 'equipe\_budget' table selected. The table structure is as follows:

	CODE_EQUIPE	NOM_EQUIPE	BUDGET_EQUIPE	NB_COURREURS_EQUIPE	BUDGET_PAR_COURSEUR
1	A2R	AG2R PREVoyance	1500000	9	
2	ALS	ALESSIO	1700000	9	
3	BAN	IBANESER.COM	2000000	9	
4	BEDOIS PERIER DOLLANGE	EDP	1000000	9	
5	C.A	CREDIT AGRICOLE	2000000	9	
6	CAL	CALDROLA - SO.DI	1400000	9	
7	COF	COFIDIS CREDIT PAR T	2000000	9	
8	CSC	TEAM CSC	2500000	9	
9	DEL	DELEUX TOURS	2100000	9	
10	FAS	EUSKALTEL - EUSKADI	2400000	9	
11	FAS	FASSA BORTOLO	2500000	9	
12	FDJ	FDJEUX.COM	2600000	9	
13	GST	GEROLSTEINER	3000000	9	
14	KEL	KELME - COSTA BLANCA	2100000	9	
15	LDF	LOTTO DÉPARTEMENT	2700000	9	
16	ONE	ONCE - EROSKI	3600000	9	
17	QSC	QUICK STEP - DAVIDAM	2700000	9	
18	RABOBANK	RABOBANK	1800000	9	
19	SAE	SACCO - MACHINERIE	2700000	9	
20	TEA	TEAM ENDURA	2100000	9	
21	TEL	TEAM TELKOM	4900000	9	
22	USP	US POSTAL - BERRY PL	5400000	9	

FIGURE 3.33 – État de la table avant l'exécution (colonne vide)

The screenshot shows the Pentaho Data Integration (PDI) interface with the transformation 'T04\_Calcul\_BUDGET'. The transformation diagram consists of three main components connected sequentially:

- An 'Extraction depuis table' node pointing to a 'BUDGET\_PAR\_COURSEUR' database table.
- The 'BUDGET\_PAR\_COURSEUR' table.
- An 'Insertion / Mise à jour table' node.

The execution results window at the bottom displays the log of operations performed:

```

2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Valeur pour la recherche: [TEL]
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Ligne pour mise à jour trouvée: [TEL], [4500000.0], [9], [500000.0]
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Mise à jour ligne avec: [TEL]
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Valeur pour la recherche: [USP]
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Ligne pour mise à jour trouvée: [USP], [5400000.0], [9], [600000.0]
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Mise à jour ligne avec: [USP]
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Traitement effectué pour 0 lignes.
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Commit en database connection [CONN_BO_S]
2025/10/20 04:48:25 - CONN_BO_S - Connection on database connection [CONN_BO_S]
2025/10/20 04:48:25 - CONN_BO_S - Connection to database closed!
2025/10/20 04:48:25 - Insertion / Mise à jour table:0 - Fin exécution étape [Entrées=22, Sorties=0, Lues=22, Ecrites=22, Maj=22, Erreurs=0]
2025/10/20 04:48:25 - Pentaho Data Integration - L'exécution de la transformation a été achevé!

```

FIGURE 3.34 – Exécution réussie de la transformation T04

The screenshot shows the MySQL Workbench interface with three tabs open: 'Local instance MySQL80' (selected), 'Local instance MySQL80 (bd\_cible)', and 'Local instance MySQL80 (bd\_s...)'. The left sidebar shows the 'bd\_cible' schema with tables like 'bd\_cible', 'bd\_s...', 'courreurs', 'equipe', and 'equipe\_budget'. The main area shows a query window with the following SQL code:

```
1 • SELECT * FROM bd_s.equipe_budget;
```

The results grid displays the following data:

CODE_EQUIPE	NOM_EQUIPE	BUDGET_EQUIPE	NB_COUREURS_EQUIPE	BUDGET_PAR_COUREUR
A26	ASSURANCE PREVOSTINCE	1500000	9	166666.6666666666
ALB	ALÉSIO	1700000	9	188888.8888888889
BAN	IBANEOSTO.COM	2000000	9	222222.2222222222
BIB	BIBOCHES LA BOULANGÈ	2500000	9	277777.7777777775
C.A	CREDIT AGRICOLE	2000000	9	222222.2222222222
CAU	CAUCUS C.A. - SO.DI	1400000	9	155555.5555555556
COF	COFIDES CREDIT PART	2200000	9	244444.4444444444
CSC	TEAM CSC	2500000	9	277777.7777777775
DEL	JEAN DELATOURE	2100000	9	233333.3333333334
EUS	EUSKALTEL - EUSKADI	2400000	9	266666.6666666667
FAS	FAIRFIELD BORTOLLO	2500000	9	277777.7777777775
FBI	FOURIER BIKE	2600000	9	288888.8888888889
GST	GEROLSTEINER	3000000	9	333333.3333333333
KEL	KELME - COSTA BLANCA	2100000	9	233333.3333333334
LOT	LOTTO - DOMO	2700000	9	300000
ONE	ONCE - ERGOSKI	3600000	9	400000
QAD	QUANTIPOP - DAVIDTAH	2300000	9	300000
RAB	RABOBANK	1800000	9	200000
SAE	SACCO - MACCHINE PIER	2700000	9	300000
TBL	TEAM BIANCHI	2700000	9	300000
TEL	TEAM TELEKOM	4500000	9	500000
USP	US POSTAL - BERRY FL	5400000	9	600000

FIGURE 3.35 – Validation : colonne BUDGET\_PAR\_COUREUR calculée et remplie

## Résultat

Le budget par coureur a été calculé avec succès pour les 22 équipes. La colonne BUDGET\_PAR\_COUREUR contient maintenant le résultat de la division BUDGET\_EQUIPE / NB\_COUREURS\_EQUIPE.

## 3.6 T05 : Chargement trié des pays vers BD\_CIBLE

### 3.6.1 Objectif

Cette transformation illustre le transfert de données d'une base vers une autre avec application d'un tri. Les données de la table **pays** de BD\_S sont extraites, triées par nom de pays, puis chargées dans une nouvelle table **PAYS\_TRIE** de BD\_CIBLE.

### 3.6.2 Configuration des composants

#### Table Input

Extraction des données depuis BD\_S.

**Requête SQL :**

```

1 SELECT CODE_PAYS , NOM_PAYS
2 FROM pays;

```

#### Sort Rows

Le composant **Sort rows** permet de trier les données selon un ou plusieurs champs.

**Configuration :**

- Champ de tri : **NOM\_PAYS**
- Ordre : Ascendant
- Sensible à la casse : Non

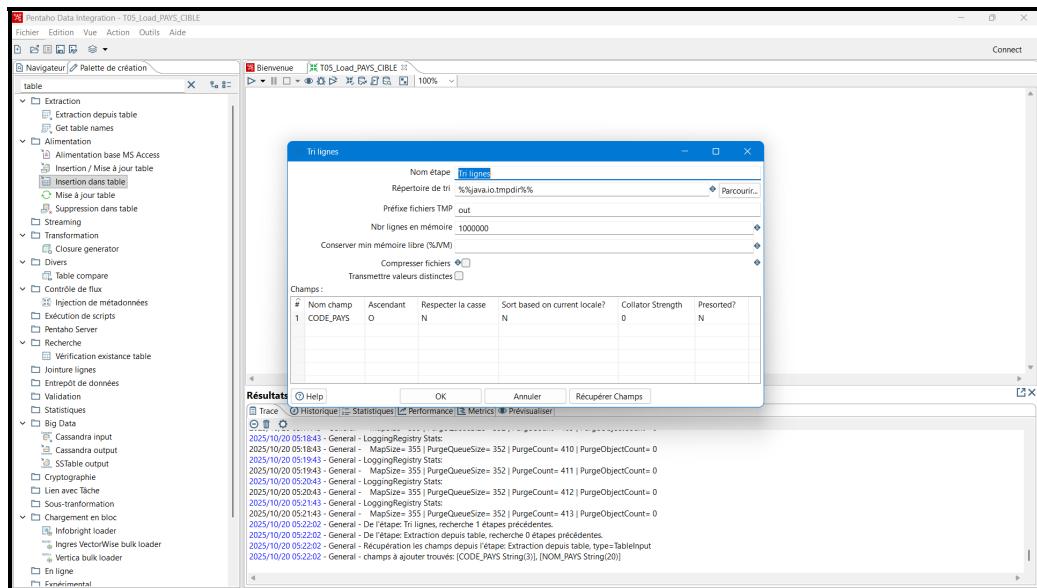


FIGURE 3.36 – Configuration du tri sur le nom des pays

## Table Output - Crédation de la table cible

Lors de la première exécution, une erreur s'est produite car la table PAYS\_TRIE n'existe pas dans BD\_CIBLE.

### ✗ Erreur rencontrée

Table 'bd\_cible.pays\_trie' doesn't exist

### Solution :

1. Dans la configuration du Table Output, cliquer sur "Récupérer les champs"
2. Cliquer sur le bouton "SQL" pour générer le script CREATE TABLE
3. Exécuter le script directement dans PDI pour créer la table

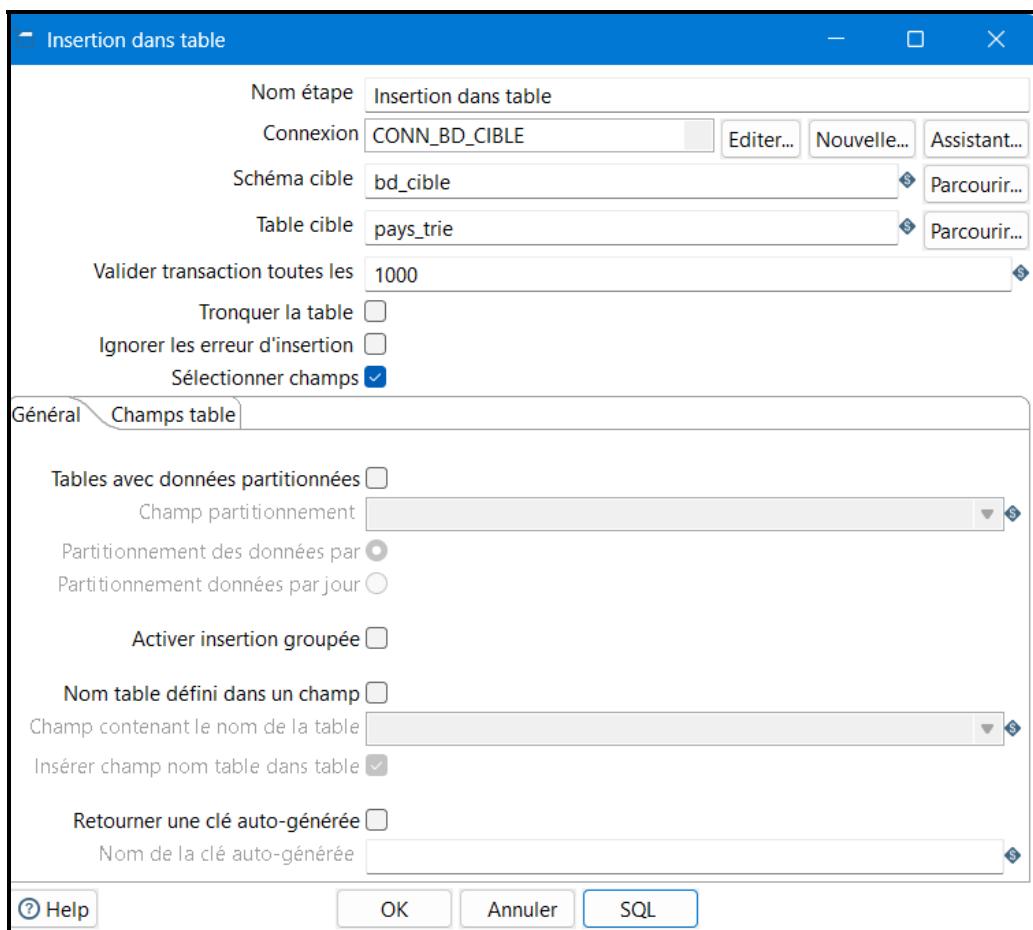


FIGURE 3.37 – Établissement de la connexion à BD\_CIBLE

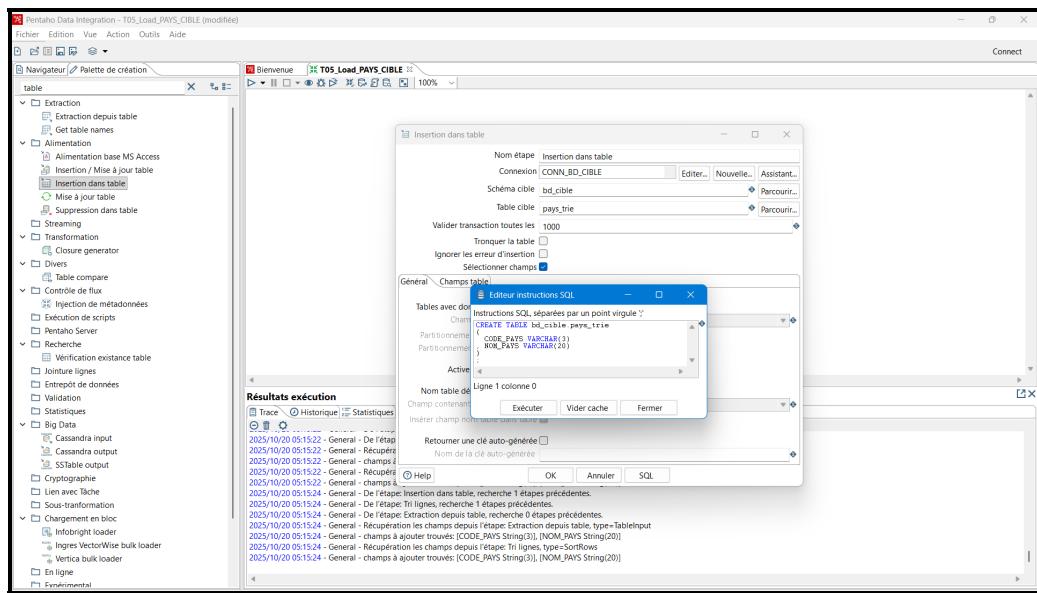


FIGURE 3.38 – Script SQL généré pour créer la table PAYS\_TRIE

### 3.6.3 Exécution et validation

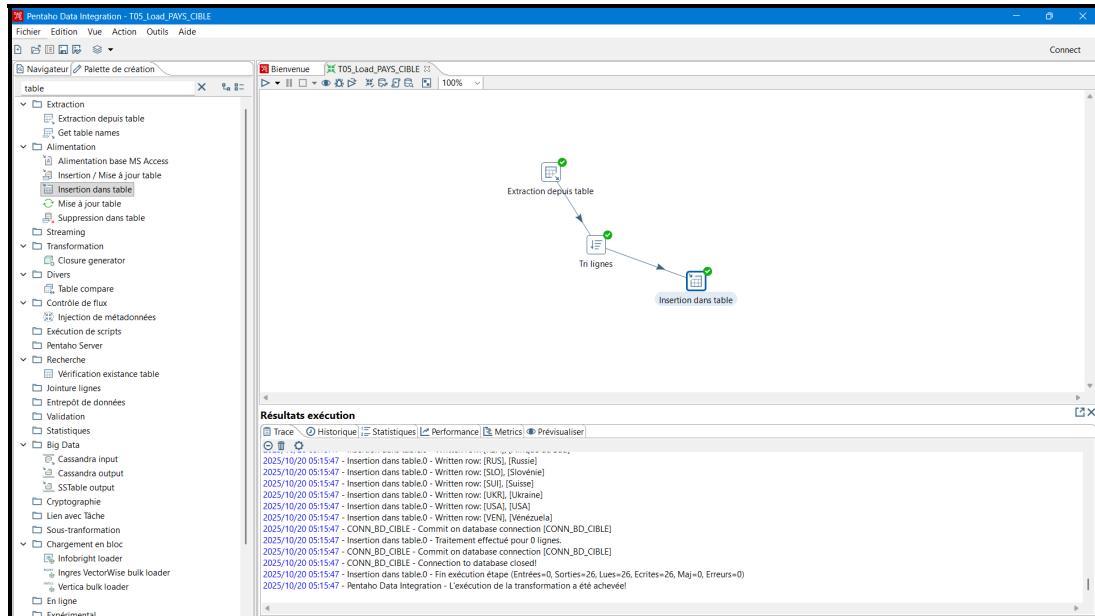


FIGURE 3.39 – Exécution réussie de la transformation T05

The screenshot shows the MySQL Workbench interface with three tabs open: 'Local instance MySQL80', 'Local instance MySQL80 (bd\_s)', and 'Local instance MySQL80 (bd\_cible)'. The 'bd\_cible' tab is active, displaying the 'pays\_trie' table. The table has two columns: 'CODE\_PAYS' and 'NOM\_PAYS'. The data is as follows:

CODE_PAYS	NOM_PAYS
AUS	Australie
AUT	Autriche
BEL	Belgique
COL	Colombie
CRO	Croatie
CZE	République Tchèque
DEN	Danemark
ESP	Espagne
EST	Estonie
FRA	France
GGR	Grande-Bretagne
GER	Allemagne
HUN	Hongrie
ITA	Italie
KAZ	Kazakhstan
LVA	Lettanie
NED	Pays-Bas
NOR	Norvège
POR	Portugal
RSA	Afrique du Sud
RUS	Russie
SLO	Slovénie
SUI	Suisse
UKR	Ukraine
USA	USA

FIGURE 3.40 – Table PAYS\_TRIE créée avec données triées alphabétiquement

### Résultat

26 pays ont été transférés de BD\_S vers BD\_CIBLE dans la table PAYS\_TRIE, triés par ordre alphabétique de nom.

## 3.7 T06 : Jointure Coureur-Équipe

### 3.7.1 Objectif

Cette transformation démontre la réalisation d'une jointure entre deux tables (opération fondamentale en ETL). L'objectif est d'enrichir les données des coureurs en y ajoutant le nom de leur équipe respective.

### 3.7.2 Architecture de la transformation

La jointure nécessite deux flux de données :

- **Flux principal (Main Stream)** : Les données des coureurs
- **Flux de recherche (Lookup Stream)** : Les données des équipes (triées sur la clé de jointure)

### 3.7.3 Configuration des composants

#### Flux de recherche : Équipes

##### Étape 1 - Table Input "Entrée Equipe" :

```
1 SELECT CODE_EQUIPE, NOM_EQUIPE
2 FROM bd_s.equipe;
```

##### Étape 2 - Sort Rows :

- Champ de tri : CODE\_EQUIPE
- Ordre : Ascendant

#### 💡 Importance du tri

Le tri du flux de recherche sur la clé de jointure est **obligatoire** pour utiliser le composant Stream Lookup. Cela permet d'optimiser les performances de la recherche.

#### Flux principal : Coureurs

##### Table Input "Entrée Coureur" :

```
1 SELECT DOSSARD_COUREUR,
2      IDENTITE_COUREUR,
3      CODE_EQUIPE_COUREUR
4 FROM bd_s.coureur;
```

## Stream Lookup - Configuration de la jointure

Le composant **Stream Lookup** effectue une recherche dans le flux trié pour enrichir le flux principal.

**Configuration initiale (avec erreurs) :**

### ✗ Erreur 1 - Tentative 1

Le flux de recherche était incorrectement configuré pour pointer vers "Entrée Coureur" au lieu de "Tri lignes".

**Correction 1 :** Configurer le flux de recherche pour utiliser "Tri lignes" (les équipes triées).

### ✗ Erreur 2 - Tentative 2

L'algorithme de cache était configuré sur "entier-pair" alors que les clés sont des chaînes de caractères.

**Correction 2 :** Utiliser l'algorithme par défaut (Hashmap) adapté aux clés de type String.

**Configuration finale correcte :**

- Flux de recherche : "Tri lignes" (équipes triées)
- Algorithme : Hashmap
- Clés de jointure :
  - Champ du flux (Coureur) : CODE\_EQUIPE\_COUREUR
  - Champ de recherche (Équipe) : CODE\_EQUIPE
- Champs ajoutés : NOM\_EQUIPE

## Table Output - Crédation de la table enrichie

Les données enrichies sont chargées dans une nouvelle table **COURSEUR\_ENRICHED** dans **BD\_CIBLE**.

**Configuration :**

- Connexion : Conn\_BD\_CIBLE
- Table : COURSEUR\_ENRICHED
- Crédation de la table via génération de script SQL

### 3.7.4 Exécution et validation

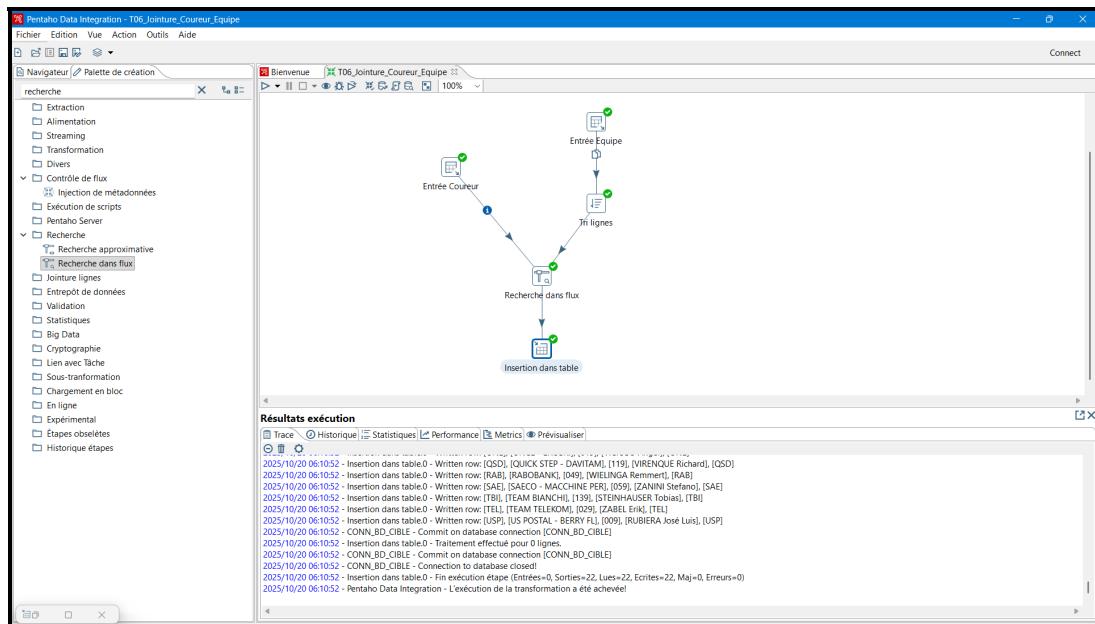


FIGURE 3.41 – Exécution réussie de la transformation T06

CODE_EQUIPE	NOM_EQUIPE	DOSSARD_COUREUR	IDENTITE_COUREUR	CODE_EQUIPE_COUREUR
ASR	AGOR PREVISION	189	TURPIN Ludovic	ASR
ALS	ALESSIO	209	PELLIZOTTI Fabrice	ALS
BAN	IBANEOSTE.COM	039	ZANDIO Xavier	BAN
BLB	BRIOCHES LA BOLANGÈ	189	VOECKLER Thomas	BLB
C.A	CREDIT AGRICOLE	129	VOIGT Jens	C.A
CAL	CALDIROLA - SO.DI	169	ZAMPERERI Steve	CAL
COF	COFIDEA CREDIT PAR T	069	VASSEUR Cédric	COF
CSC	TEAM CSC	079	SCHMIDT Sébastien	CSC
DEL	JEAN DELATOUR	219	NAZON Jean-Patrick	DEL
EUS	EUSKATEL - EUSKADI	179	ZUBELDIA Iñaki	EUS
FAS	FASSA BORTOLO	089	VELO Marco	FAS
FDJ	FDJEURO.COM	099	WILSON Matthew	FDJ
GST	GEROLSTEINER	199	ZBERG Marcus	GST
KEL	KELME - COSTA BLANCA	109	USANO Julian	KEL
LOT	LOTTO GIOMO	149	VALVERDE SAK	LOT
ONE	ONCE - ENAV	019	VECIOSO Angel	ONE
QSD	QUICK STEP - DAVIDAMI	119	VIRENIQUE Richard	QSD
RAB	RABOBANK	049	WIELINGA Remco	RAB
SAE	SACCO - MACCHINE PERI	059	ZANINI Stefano	SAE
TBL	TEAM BIANCHI	139	STEINHAUSER Tobias	TBL
TEL	TEAM TELEKOM	029	ZABEL Erik	TEL
USP	US POSTAL - BERRY FL	009	RUBIERA José Luis	USP

FIGURE 3.42 – Table COUREUR\_ENRICHED avec les données jointes

### Résultat

297 coureurs ont été lus, enrichis avec le nom de leur équipe respective (22 équipes), et les 297 lignes résultantes ont été insérées dans la table COUREUR\_ENRICHED de BD\_CIBLE.

### 3.7.5 Analyse des résultats

La jointure a permis de créer une table dénormalisée contenant :

- Le dossard du coureur
- L'identité du coureur
- Le code de l'équipe
- **Le nom de l'équipe (ajouté par la jointure)**

Cette table enrichie facilite les requêtes et analyses ultérieures en évitant de devoir effectuer des jointures répétées.

# Chapitre 4

## Orchestration des transformations avec les Jobs

### 4.1 Introduction aux Jobs

Les **Jobs** dans Pentaho Data Integration permettent d'orchestrer l'exécution de plusieurs transformations de manière séquentielle ou parallèle. Ils offrent :

- Le contrôle du flux d'exécution (séquences, conditions, boucles)
- La gestion des erreurs et des succès
- La planification et l'automatisation des processus ETL
- La modularité et la réutilisabilité du code

### 4.2 Correction préalable : Ré-exécutabilité de T00

#### 4.2.1 Problème identifié

Lors de l'orchestration, la transformation T00 échouait en cas de ré-exécution en raison d'erreurs de clé primaire dupliquée.

##### Erreur de clé dupliquée

Duplicate entry 'AUS' for key 'PRIMARY'

Cette erreur se produisait car le composant **Table Output** tentait d'insérer des lignes dont les clés primaires existaient déjà.

#### 4.2.2 Solution appliquée

Remplacement du composant **Table Output** par **Insert / Update** dans la transformation T00.

### Configuration :

- Champ clé de recherche : CODE\_PAYS
  - Action : INSERT si la clé n'existe pas, UPDATE si elle existe
- Cette modification permet d'exécuter le job plusieurs fois sans erreur.

## 4.3 J01\_Sec\_Main\_Job : Job de chargement initial

### 4.3.1 Objectif

Ce job orchestré l'extraction initiale de toutes les sources de données (Excel, TXT, XML) vers la base de données source BD\_S, ainsi que l'exportation vers Excel.

### 4.3.2 Structure du job

Le job enchaîne les transformations dans l'ordre suivant :

1. **START** (point de départ)
2. **T00\_Load\_PAYS** : Chargement des pays depuis Excel
3. **T01\_Load\_EQUIPE** : Chargement des équipes depuis fichier texte
4. **T03\_Load\_ETAPE** : Chargement des étapes depuis XML
5. **T02\_Export\_PAYS\_to\_Excel** : Export des pays vers Excel

### 4.3.3 Configuration des liens

Entre chaque transformation, des liens de type **Success** (ligne verte) sont configurés pour garantir l'exécution séquentielle. Une transformation ne démarre que si la précédente s'est terminée avec succès.

### 4.3.4 Exécution et résultats

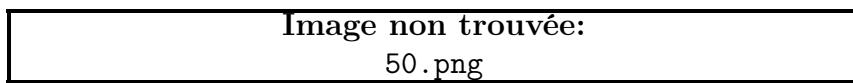


FIGURE 4.1 – Exécution réussie du job J01\_Sec\_Main\_Job

#### Résultat du job

Toutes les transformations se sont exécutées avec succès dans l'ordre défini. Les données sources ont été chargées dans BD\_S et exportées vers Excel.

## 4.4 J01\_Sec\_Job : Job de traitement

### 4.4.1 Objectif

Ce second job gère les opérations de traitement sur les données déjà chargées : calculs, tris, et jointures.

### 4.4.2 Structure du job

1. START
2. T04\_Calcul\_BUDGET : Calcul du budget par coureur
3. T05\_Load\_PAYS\_CIBLE : Transfert trié vers BD\_CIBLE
4. T06\_Jointure\_Coureur\_Equipe : Enrichissement par jointure

### 4.4.3 Exécution et résultats

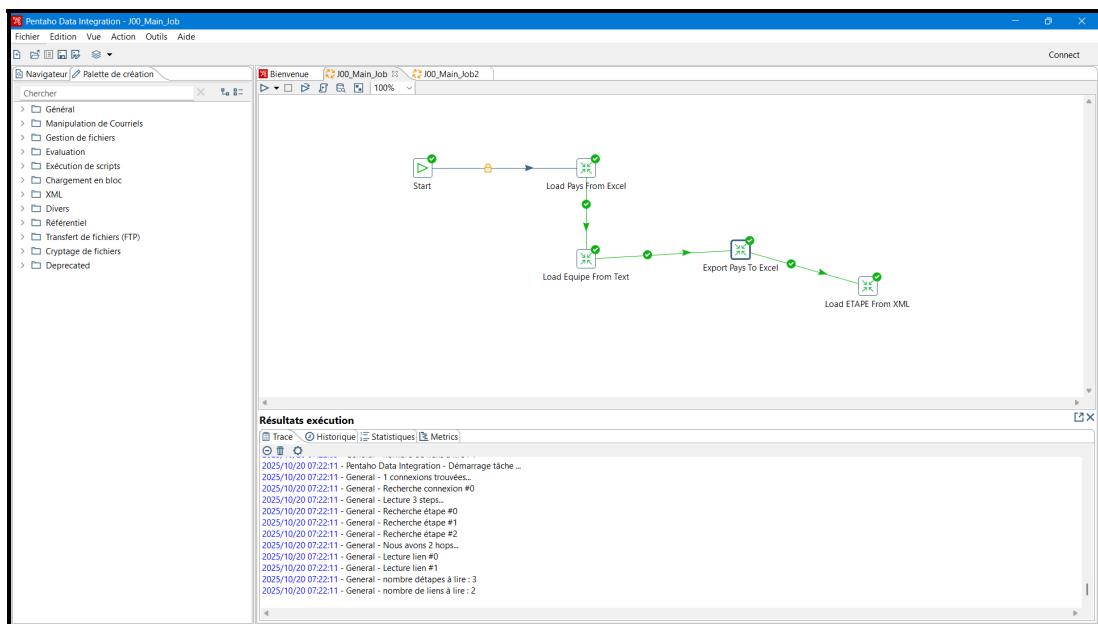


FIGURE 4.2 – Exécution réussie du job J01\_Sec\_Job

### Résultat du job

Toutes les transformations de traitement se sont exécutées correctement. Les calculs ont été effectués, les données triées et transférées, et les jointures réalisées avec succès.

## 4.5 Orchestration globale

### 4.5.1 Ordre d'exécution recommandé

Pour un processus ETL complet, l'ordre d'exécution est :

1. **J01\_Sec\_Main\_Job** : Chargement des données sources
2. Exécution du script `Populate_BD_S.sql` pour peupler les tables additionnelles
3. **J01\_Sec\_Job** : Traitement et enrichissement

### 4.5.2 Avantages de l'approche modulaire

La séparation en deux jobs distincts offre plusieurs avantages :

- **Clarté** : Séparation logique entre chargement et traitement
- **Maintenance** : Facilite les modifications et le débogage
- **Réutilisabilité** : Chaque job peut être exécuté indépendamment
- **Performance** : Possibilité d'optimiser chaque phase séparément
- **Gestion d'erreurs** : Localisation plus facile des problèmes

### 4.5.3 Démonstration vidéo

Une vidéo démonstrative a été réalisée pour illustrer l'exécution complète des deux jobs et présenter les détails de chaque étape. Cette vidéo permet de visualiser :

- Le démarrage et le déroulement de chaque job
- Les indicateurs de succès/échec à chaque étape
- Les logs d'exécution
- La validation des résultats dans MySQL Workbench

# Chapitre 5

## Analyse et bilan du projet

### 5.1 Synthèse des transformations réalisées

#### 5.1.1 Récapitulatif des objectifs atteints

Le tableau suivant résume les différentes transformations ETL réalisées et leurs objectifs :

Transformation	Objectif	Statut
T00	Chargement PAYS depuis Excel	✓ Réussi
T01	Chargement EQUIPE depuis TXT	✓ Réussi
T02	Export PAYS vers Excel	✓ Réussi
T03	Chargement ETAPE depuis XML	✓ Réussi
T04	Calcul budget par coureur	✓ Réussi
T05	Transfert trié vers BD_CIBLE	✓ Réussi
T06	Jointure Coureur-Équipe	✓ Réussi

TABLE 5.1 – Récapitulatif des transformations ETL réalisées

#### 5.1.2 Volumétrie des données traitées

- **Pays** : 26 enregistrements
- **Équipes** : 22 enregistrements
- **Étapes** : 21 enregistrements
- **Coureurs** : 297 enregistrements
- **Total des lignes traitées** : Plus de 600 lignes à travers toutes les transformations

## 5.2 Compétences acquises

### 5.2.1 Maîtrise des concepts ETL

Ce projet a permis de comprendre et d'appliquer les trois phases fondamentales de l'ETL :

#### 1. Extract (Extraction) :

- Lecture de fichiers Excel avec Microsoft Excel Input
- Parsing de fichiers XML avec Get data from XML
- Extraction depuis fichiers texte délimités
- Requêtes SQL sur bases de données relationnelles

#### 2. Transform (Transformation) :

- Filtrage de données avec Filter Rows
- Tri de données avec Sort Rows
- Calculs mathématiques avec Calculator
- Jointures avec Stream Lookup
- Contrôle qualité des données

#### 3. Load (Chargement) :

- Insertion dans tables MySQL avec Table Output
- Mise à jour conditionnelle avec Insert / Update
- Export vers Excel avec Microsoft Excel Output
- Création dynamique de tables

### 5.2.2 Utilisation de Pentaho Data Integration

#### Compétences techniques

- Configuration et test de connexions à des bases de données
- Utilisation de l'interface graphique Spoon
- Création et chaînage de composants (steps)
- Débogage et résolution d'erreurs
- Utilisation des logs pour le diagnostic
- Génération de scripts SQL depuis PDI

#### Bonnes pratiques apprises

- Nommage cohérent des transformations et jobs
- Organisation modulaire des fichiers

- Documentation du code et des flux
- Tests unitaires de chaque transformation avant orchestration
- Gestion de la ré-exécutabilité des processus
- Validation systématique des résultats dans MySQL Workbench

## 5.3 Difficultés rencontrées et solutions

### 5.3.1 Problèmes techniques et résolutions

Le tableau suivant détaille les principales difficultés rencontrées et les solutions apportées :

Transformation	Problème	Solution
T00	Lignes avec CODE_PAYS vide	Ajout d'un filtre pour éliminer les valeurs NULL
T00	Erreur clé dupliquée à la ré-exécution	Remplacement Table Output par Insert/Update
T03	Configuration XPath complexe	Étude de la structure XML et tests itératifs
T04	Colonne BUDGET_PAR_COUREUR inexistante	Ajout de la colonne via ALTER TABLE
T05	Table PAYS_TRIE inexistante	Génération et exécution du script CREATE TABLE
T06	Mauvaise configuration du Stream Lookup	Correction du flux de recherche et de l'algorithme

TABLE 5.2 – Problèmes rencontrés et solutions apportées

### 5.3.2 Leçons apprises

#### Importance de la validation des données

Les erreurs rencontrées avec les valeurs NULL dans T00 ont mis en évidence l'importance de :

- Toujours inspecter les données sources avant traitement
- Ajouter des contrôles qualité systématiques (filtres, validations)
- Anticiper les cas d'erreur possibles

#### Conception pour la ré-exécutabilité

L'utilisation d'Insert/Update au lieu de Table Output simple permet :

- D'exécuter les transformations en mode idempotent

- 
- De faciliter les tests et le débogage
  - D'automatiser les processus sans intervention manuelle

## Documentation et nommage

Une convention de nommage claire (T00, T01, etc.) et une organisation structurée des fichiers ont grandement facilité :

- La maintenance du projet
- La compréhension du flux global
- Le travail en équipe

## 5.4 Avantages de l'approche ETL avec Pentaho

### 5.4.1 Points forts identifiés

1. **Interface graphique intuitive** : Le développement par glisser-déposer accélère la création de flux complexes
2. **Versatilité des sources/cibles** : Support natif de nombreux formats (Excel, XML, TXT, bases de données)
3. **Composants riches** : Large bibliothèque de transformations pré-construites
4. **Visualisation du flux** : Compréhension immédiate du processus ETL
5. **Gestion d'erreurs** : Logs détaillés et indicateurs visuels de succès/échec
6. **Modularité** : Possibilité de réutiliser des transformations dans plusieurs jobs
7. **Performance** : Traitement en flux (streaming) pour des volumes importants

### 5.4.2 Cas d'usage adaptés

Pentaho Data Integration est particulièrement adapté pour :

- L'intégration de données hétérogènes
- La migration de bases de données
- L'alimentation de datawarehouses
- La consolidation de données multi-sources
- Les processus ETL complexes avec transformations multiples

## 5.5 Perspectives d'amélioration

### 5.5.1 Améliorations techniques possibles

#### Gestion des erreurs avancée

- Implémenter des stratégies de retry en cas d'échec temporaire
- Créer des tables d'erreurs pour capturer les lignes rejetées
- Ajouter des notifications par email en cas d'échec
- Mettre en place des logs structurés pour le monitoring

#### Optimisation des performances

- Utiliser le traitement parallèle pour les grandes volumétries
- Implémenter des stratégies de chargement incrémental
- Optimiser les requêtes SQL dans les Table Input
- Utiliser des index appropriés dans les bases de données

#### Automatisation complète

- Créer un job maître orchestrant J01\_Sec\_Main\_Job et J01\_Sec\_Job
- Planifier l'exécution automatique via cron ou Windows Task Scheduler
- Intégrer avec des systèmes de workflow d'entreprise
- Implémenter des mécanismes de checkpoint pour reprendre après erreur

### 5.5.2 Extensions fonctionnelles

#### Enrichissement des données

- Ajouter des dimensions temporelles (date de chargement, timestamp)
- Calculer des indicateurs statistiques sur les coureurs et équipes
- Créer des agrégations par pays, par équipe, par étape
- Implémenter une gestion d'historique (SCD - Slowly Changing Dimensions)

#### Qualité des données

- Ajouter des règles de validation métier
- Implémenter des contrôles de cohérence inter-tables
- Créer des rapports de qualité de données
- Mettre en place des processus de nettoyage automatique

## Analyse et reporting

- Créer des vues analytiques sur les données enrichies
- Générer des rapports automatiques en PDF ou Excel
- Intégrer avec des outils de Business Intelligence (Tableau, Power BI)
- Développer des dashboards de suivi du Tour de France

## 5.6 Application dans un contexte professionnel

### 5.6.1 Transférabilité des compétences

Les compétences acquises dans ce projet sont directement applicables dans des contextes professionnels variés :

- **Finance** : Consolidation de données de différentes filiales
- **Retail** : Intégration des ventes multi-canaux
- **Santé** : Agrégation de dossiers patients de sources diverses
- **Logistique** : Consolidation de données de suivi de livraisons
- **Marketing** : Unification de données clients multi-sources

### 5.6.2 Conformité et gouvernance des données

Dans un environnement professionnel, il serait important d'ajouter :

- Des mécanismes d'audit trail (traçabilité complète)
- La gestion des données sensibles (anonymisation, chiffrement)
- La conformité RGPD pour les données personnelles
- La documentation technique complète pour la certification
- Des tests automatisés de non-régression

# Conclusion

## Synthèse générale

Ce projet de travaux pratiques sur l'ETL avec Pentaho Data Integration a permis d'acquérir une expérience complète et concrète des processus d'extraction, de transformation et de chargement de données.

À travers la réalisation de sept transformations distinctes et de deux jobs d'orchestration, nous avons pu manipuler des données issues de sources hétérogènes (Excel, XML, fichiers texte, bases de données MySQL) et les intégrer dans un système cohérent et structuré.

## Objectifs atteints

L'ensemble des objectifs fixés au début du projet a été atteint avec succès :

- **Transfert de données** entre bases MySQL avec et sans transformation
- **Lecture et écriture** de fichiers Excel bidirectionnelle
- **Parsing de fichiers XML** et intégration dans une base relationnelle
- **Traitement de fichiers texte** délimités
- **Application de tris** sur les données transférées
- **Réalisation de jointures** entre tables
- **Calculs sur les données** (budget par coureur)
- **Orchestration** des transformations via des jobs

## Compétences développées

Ce projet nous a permis de développer des compétences techniques et méthodologiques essentielles :

### Compétences techniques :

- Maîtrise de Pentaho Data Integration (Spoon)
- Manipulation avancée de MySQL et MySQL Workbench
- Compréhension approfondie des processus ETL
- Gestion de différents formats de données
- Techniques d'optimisation et de débogage

### Compétences méthodologiques :

- Approche structurée de résolution de problèmes
- Gestion de projet modulaire
- Documentation technique
- Tests et validation systématiques
- Analyse et correction d'erreurs

## Valeur ajoutée du projet

Au-delà de l'apprentissage technique, ce projet a démontré l'importance de :

1. **La qualité des données** : Les contrôles de validation sont essentiels pour garantir l'intégrité du système d'information
2. **La ré-exécutabilité** : Concevoir des processus idempotents facilite l'automatisation et la maintenance
3. **La modularité** : Découper les processus en transformations atomiques améliore la maintenabilité
4. **La documentation** : Une documentation claire est indispensable pour le travail collaboratif

## Perspectives professionnelles

Les compétences acquises dans ce projet sont hautement valorisées sur le marché du travail, notamment dans les domaines de :

- L'ingénierie des données (Data Engineering)
- L'analyse de données (Data Analytics)
- La Business Intelligence
- L'architecture de systèmes d'information
- Le développement de datawarehouses

Les outils ETL comme Pentaho sont largement utilisés dans l'industrie pour gérer les flux de données complexes et multi-sources, faisant de cette compétence un atout majeur pour tout professionnel de la donnée.

## Remerciements

Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à :

- **Professeur Hassan BDIR**, Doctorant et Expert en Sciences des Données, Chef de Filière, pour l'honneur qu'il m'a fait en acceptant d'encadrer ce projet. Son expertise pointue dans le domaine de la Data, ses conseils éclairés et son accompagnement rigoureux m'ont permis de développer une compréhension approfondie des processus ETL et des enjeux de l'ingénierie des données. Sa vision stratégique et sa maîtrise technique ont été déterminantes dans la réussite de ce travail
- **L'École Nationale des Sciences Appliquées de Tanger (ENSA Tanger)** pour la qualité de la formation dispensée et pour la mise à disposition des ressources matérielles et logicielles nécessaires à la réalisation de ce travail pratique
- **La communauté open-source Pentaho** pour la qualité et la robustesse de l'outil Pentaho Data Integration, ainsi que pour la richesse de la documentation mise à disposition

Ce projet constitue une étape déterminante dans mon parcours de formation en ingénierie des données et m'ouvre des perspectives prometteuses vers des projets d'envergure dans le domaine de la gestion, de l'analyse et de la valorisation des données.

*ALLAM ELARBI*

*ENSA Tanger - Année universitaire 2025/2026*

# Annexe A

## Scripts SQL

### A.1 Script de création de la base de données source

Listing A.1 – Contenu complet de Create\_Schema\_BD\_S.sql

```
1  -- Creation du schema BD_S
2  CREATE SCHEMA IF NOT EXISTS BD_S;
3  USE BD_S;
4
5  -- Structure de la table 'coureur'
6  CREATE TABLE IF NOT EXISTS 'coureur' (
7      'DOSSARD_COUREUR' char(3) NOT NULL DEFAULT '',
8      'IDENTITE_COUREUR' char(40) NOT NULL DEFAULT '',
9      'CODE_PAYS_COUREUR' char(3) NOT NULL DEFAULT '',
10     'CODE_EQUIPE_COUREUR' char(3) NOT NULL DEFAULT ''
11 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
12
13 -- Structure de la table 'equipe'
14 CREATE TABLE IF NOT EXISTS 'equipe' (
15     'CODE_EQUIPE' char(3) NOT NULL DEFAULT '',
16     'NOM_EQUIPE' char(20) DEFAULT NULL,
17     PRIMARY KEY ('CODE_EQUIPE')
18 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
19
20 -- Structure de la table 'equipe_budget'
21 CREATE TABLE IF NOT EXISTS 'equipe_budget' (
22     'CODE_EQUIPE' char(3) NOT NULL DEFAULT '',
23     'NOM_EQUIPE' char(20) DEFAULT NULL,
24     'BUDGET_EQUIPE' double NOT NULL,
25     'NB_COUREURS_EQUIPE' int(11) NOT NULL,
26     PRIMARY KEY ('CODE_EQUIPE')
```

```
27 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;  
28  
29 -- Structure de la table 'etape'  
30 CREATE TABLE IF NOT EXISTS 'etape' (  
31   'NUM_ETAPE' int(10) unsigned NOT NULL DEFAULT '0',  
32   'VILLE_DEPART' char(30) DEFAULT NULL,  
33   'VILLE_ARRIVEE' char(30) DEFAULT NULL,  
34   PRIMARY KEY ('NUM_ETAPE')  
35 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;  
36  
37 -- Structure de la table 'pays'  
38 CREATE TABLE IF NOT EXISTS 'pays' (  
39   'CODE_PAYS' char(3) NOT NULL DEFAULT '',  
40   'NOM_PAYS' char(20) DEFAULT NULL,  
41   PRIMARY KEY ('CODE_PAYS')  
42 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

## A.2 Script d'ajout de la colonne calculée

Listing A.2 – Ajout de BUDGET\_PAR\_COUREUR

```
1 -- Ajout de la colonne pour le calcul du budget par coureur  
2 USE BD_S;  
3  
4 ALTER TABLE equipe_budget  
5 ADD COLUMN BUDGET_PAR_COUREUR DOUBLE;
```

## A.3 Script de création de la base cible

Listing A.3 – Création de BD\_CIBLE

```
1 -- Creation du schema BD_CIBLE  
2 CREATE SCHEMA IF NOT EXISTS BD_CIBLE;  
3 USE BD_CIBLE;  
4  
5 -- La structure des tables sera creee dynamiquement  
6 -- par les transformations PDI
```

## Annexe B

# Configuration des transformations

## B.1 Paramètres de connexion aux bases de données

### B.1.1 Connexion Conn\_BD\_S

Paramètre	Valeur
Connection Name	Conn_BD_S
Connection Type	MySQL
Access Method	Native (JDBC)
Host Name	localhost
Database Name	BD_S
Port Number	3306
Username	root

TABLE B.1 – Paramètres de connexion à BD\_S

### B.1.2 Connexion Conn\_BD\_CIBLE

Paramètre	Valeur
Connection Name	Conn_BD_CIBLE
Connection Type	MySQL
Access Method	Native (JDBC)
Host Name	localhost
Database Name	BD_CIBLE
Port Number	3306
Username	root

TABLE B.2 – Paramètres de connexion à BD\_CIBLE

## B.2 Liste des composants PDI utilisés

Composant	Usage dans le projet
Microsoft Excel Input	Lecture de fichiers Excel (Pays.xls)
Microsoft Excel Output	Export vers Excel (Export_PAYS.xlsx)
Text File Input	Lecture de fichiers texte (Equipe.txt)
Get data from XML	Parsing de fichiers XML (Etape.xml)
Table Input	Extraction depuis bases MySQL
Table Output	Insertion dans tables MySQL
Insert / Update	Insertion ou mise à jour conditionnelle
Filter Rows	Filtrage de lignes selon conditions
Sort Rows	Tri de données
Calculator	Calculs mathématiques
Stream Lookup	Jointure entre flux de données

TABLE B.3 – Composants PDI utilisés dans le projet

# Annexe C

## Guide de reproduction du projet

### C.1 Préquis système

- Système d’exploitation : Windows 10/11, Linux, ou macOS
- RAM : Minimum 4 GB (8 GB recommandé)
- Espace disque : Minimum 2 GB libres
- Java JDK 8 ou supérieur installé et configuré

### C.2 Installation étape par étape

#### C.2.1 Étape 1 : Installation de MySQL

1. Télécharger MySQL Server depuis <https://dev.mysql.com/downloads/>
2. Installer avec les paramètres par défaut
3. Définir un mot de passe root
4. Démarrer le service MySQL

#### C.2.2 Étape 2 : Installation de MySQL Workbench

1. Télécharger depuis <https://dev.mysql.com/downloads/workbench/>
2. Installer et configurer la connexion locale
3. Tester la connexion à localhost :3306

#### C.2.3 Étape 3 : Installation de Pentaho Data Integration

1. Télécharger PDI depuis <https://sourceforge.net/projects/pentaho/>
2. Extraire l’archive dans un dossier (ex : C :\Pentaho)
3. Télécharger MySQL Connector/J (driver JDBC)

- 
4. Copier le fichier .jar dans PDI\_HOME/lib
  5. Lancer Spoon.bat (Windows) ou spoon.sh (Linux/Mac)

#### C.2.4 Étape 4 : Configuration des bases de données

1. Ouvrir MySQL Workbench
2. Créer les schémas BD\_S et BD\_CIBLE
3. Exécuter le script Create\_Schema\_BD\_S.sql
4. Vérifier la création des tables

#### C.2.5 Étape 5 : Configuration dans PDI

1. Lancer Spoon
2. Créer les connexions Conn\_BD\_S et Conn\_BD\_CIBLE
3. Tester les connexions
4. Préparer l'arborescence des dossiers du projet

### C.3 Ordre d'exécution recommandé

Pour reproduire le projet dans l'ordre :

1. Créer et exécuter T00\_Load\_PAYS
2. Créer et exécuter T01\_Load\_EQUIPE
3. Créer et exécuter T02\_Export\_PAYS\_to\_Excel
4. Créer et exécuter T03\_Load\_ETAPE
5. Exécuter le script Populate\_BD\_S.sql
6. Créer et exécuter T04\_Calcul\_BUDGET
7. Créer et exécuter T05\_Load\_PAYS\_CIBLE
8. Créer et exécuter T06\_Jointure\_Coureur\_Equipe
9. Créer et exécuter J01\_Sec\_Main\_Job
10. Créer et exécuter J01\_Sec\_Job

### C.4 Vérifications à effectuer

Après chaque transformation, vérifier :

- Aucune erreur dans les logs PDI

- Le nombre de lignes traitées correspond aux attentes
- Les données sont présentes dans les tables cibles
- Les données sont correctes (pas de NULL inappropriés, formats corrects)

# Annexe D

## Glossaire

**ETL** Extract, Transform, Load - Processus d'extraction, transformation et chargement de données

**PDI** Pentaho Data Integration - Outil open-source d'ETL développé par Pentaho

**Spoon** Interface graphique de Pentaho Data Integration

**Transformation** Unité de traitement dans PDI contenant des steps connectés

**Step** Composant élémentaire effectuant une opération (extraction, transformation, chargement)

**Job** Orchestrateur permettant d'enchaîner plusieurs transformations

**Hop** Lien entre deux steps dans une transformation

**JDBC** Java Database Connectivity - API Java pour se connecter aux bases de données

**XPath** Langage de requête pour parcourir des documents XML

**Stream Lookup** Composant PDI pour effectuer des jointures en mémoire

**Insert/Update** Composant PDI permettant d'insérer ou mettre à jour selon une clé

**Idempotence** Propriété d'une opération donnant le même résultat qu'on l'exécute une ou plusieurs fois

**Datawarehouse** Entrepôt de données centralisé pour l'analyse

**SCD** Slowly Changing Dimension - Technique de gestion de l'historique dans les entrepôts de données

## Annexe E

# Bibliographie et ressources

### E.1 Documentation officielle

- Pentaho Data Integration Documentation :  
<https://help.pentaho.com/Documentation/>
- MySQL Reference Manual :  
<https://dev.mysql.com/doc/refman/8.0/en/>
- XPath Tutorial :  
[https://www.w3schools.com/xml/xpath\\_intro.asp](https://www.w3schools.com/xml/xpath_intro.asp)

### E.2 Tutoriels et guides

- Pentaho Community :  
<https://community.hitachivantara.com/>
- PDI Step Reference :  
<https://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps>
- ETL Best Practices :  
Articles et white papers sur les meilleures pratiques ETL

### E.3 Ouvrages recommandés

- *The Data Warehouse Toolkit* - Ralph Kimball
- *Building a Data Warehouse : With Examples in SQL Server* - Vincent Rainardi
- *Pentaho Data Integration Beginner's Guide* - María Carina Roldán