| **Type de données** | **Fournisseur direct** | **Source réelle** | **Ownership** | **Remarques** |
| --- | --- | --- | --- | --- |
| **Prépaiements** | DFIN | SGRF DAT / ALM | SGRF/ALM | Fournies pour RACER |
| **Origination volume** | DFIN | SGRF DAT / ALM | SGRF/ALM | Fournies pour RACER |
| **Pertes (PES, PED)** | RISQ CRE / ERM | RISQ CRE / ERM | RISQ | Données certifiées par RISQ |
| **Encours sains/défaut** | DFIN (extraction SAFIR) | SAFIR (RISQ) | RISQ | DFIN n’est **pas propriétaire** de ces données |
| **Provisions** | DFIN (extraction SAFIR) | SAFIR (RISQ) | RISQ | À valider avec RISQ CRE/ERM |

## Traitement des données

Le traitement des données collectées s’effectue dans l’environnement SAS puis dans Dataiku avec Python, après leur récupération depuis le S3 ou les dossiers SAS exportés.

Les principales étapes de traitement sont :

* **Filtrage du périmètre** : seules les lignes correspondant au périmètre titrisé sont conservées (filtrage par code portefeuille bâlois, CTR, rating system, etc.).
* **Normalisation et renommage des colonnes** : les noms des variables sont harmonisés conformément au fichier variable\_mapping, afin de garantir une cohérence des données en entrée des calculs.
* **Jointures entre les différentes bases** : les données issues des différentes sources (PD, LGD, etc.) sont jointes entre elles selon les clés définies dans le mapping.
* **Pré-traitement spécifique** : des règles métiers sont appliquées pour certaines variables (ex. : gestion des dates, ajustements spécifiques à la titrisation, etc.).
* **Préparation à l’analyse qualité et au calcul** : les données sont préparées dans un format structuré pour être utilisées dans les modules de contrôle qualité (DataQuality) puis dans le calcul des indicateurs (Indicator).

Ce traitement vise à obtenir des jeux de données propres, alignés avec les exigences du reporting titrisation, et compatibles avec les templates de production.

Les **données nécessaires aux calculs des indicateurs PD** sont en grande partie **préparées dans SAS**, puis consommées directement dans Python via l’outil d’automatisation.

Les **données LGD**, quant à elles, sont également prétraitées sous SAS, mais leur traitement se poursuit dans Python, en particulier à travers la classe RdsLgd.

Cette classe permet de construire la table de recouvrements à partir des données historiques LGD décrites dans la partie Inputs (voir section 4). Elle constitue une étape clé pour l’analyse de la qualité des données LGD et le calcul des indicateurs LGD nets et bruts.

Une image contenant texte, document, capture d’écran

Le contenu généré par l’IA peut être incorrect.

**Classe RdsLgd : Construction de la table de recouvrements LGD**

La classe RdsLgd permet de créer une table structurée des recouvrements (Recoveries RDS) à partir des flux de recouvrements, des dates de défaut, et des caractéristiques des contrats.

**Méthode principale : rds\_recovery(amount\_df, base\_lgd\_df)**

**Paramètres :**

* amount\_df : DataFrame contenant les flux financiers (OAD, recouvrements, etc.)
* base\_lgd\_df : DataFrame contenant les informations des contrats en défaut (date de défaut, EAD, portefeuille, etc.)

**Étapes principales :**

* Suppression des concours OAD sans montant :
* Identifie les concours dont les flux sont de type OAD (libelle\_sens\_flux == "1. OAD") et dont le montant est manquant (isnan(montant\_conv\_actu)).
* Ces concours sont exclus de la suite de la table de recouvrement.
* Récupération des recouvrements :
* Sélectionne les lignes de amount\_df où le flux est un recouvrement (libelle\_sens\_flux == "2. recouvrement").
* Calcule un horizon mensuel entre la date de situation et la date de défaut.
* Agrège les montants mensuels par contrat et horizon.
* Création d’une table pivotée :
* Construit une table avec une colonne par mois (M0, M1, M2, …) représentant les recouvrements mensuels.
* Utilise .pivot() pour reformater les données et .fillna(0) pour remplir les valeurs manquantes.
* Jointure avec les données LGD :
* Joint la table de recouvrement avec base\_lgd\_df selon l’ID du contrat et la date de défaut.
* Applique des filtres : défauts fermés (closing\_defaut\_indicator == 1), portefeuilles ciblés (ccr\_cdpotf in [13, 14]), entités RBDF ou CDN.
* Filtrage sur le chiffre d’affaires et segment :
* Garde uniquement les professionnels/entreprises pertinentes selon les seuils de chiffre d’affaires (bdd\_ca\_obligor1) et la classification BtoB/BtoC.
* Création finale du RDS Recovery :
* Conserve uniquement les colonnes pertinentes : identifiants, EAD, segments, mois de recouvrement (M0, M1, …), etc.

**Résultat :**

La méthode retourne un DataFrame Recoveries RDS :

* Une ligne par contrat en défaut.
* Des colonnes représentant les montants recouvrés chaque mois depuis le défaut.
* Les informations nécessaires au calcul des indicateurs LGD (recouvrement net, taux de récupération, aggravations, etc.).

## Analyse de la qualité des données

L’analyse de la qualité des données constitue une étape essentielle du processus de production des datapacks. Elle permet d’identifier les problèmes susceptibles d’impacter le calcul des indicateurs (PD, LGD, etc.) et d'assurer la robustesse des livrables.

L’objectifs de l'analyse qualité est :

* Détecter les valeurs manquantes dans les champs critiques (note interne, CA, EAD, dates défauts, etc.).
* Identifier les doublons dans les bases de contrats, de contreparties ou de flux.
* Repérer les outliers (valeurs extrêmes) sur des variables clés (CA, montant recouvré, taux de défaut).
* Vérifier la cohérence des données via des sanity checks :
  + Nombre d’obligors par cohorte, entité de booking et rating system (SME/LC et VLC).
  + Montant d’EAD (Exposition au défaut) agrégé par cohorte et rating system.
  + Statistiques descriptives des notations internes : Moyenne, médiane, écart-type, min, max, par cohorte et rating system.