

RAPPORT DE STAGE (SI)



OUAZZANI GHALI

2^{ème} année GI

Organisme d'accueil :



Campus BMCE Bank, Bouskoura Green City, Casablanca, Maroc

Responsable du stage :

Mme. El Boukhari Ibtissam

Présidente du directoire

Avant-propos

Ce stage d'ingénieur s'inscrit dans le cadre de la formation d'ingénieur proposée par l'Ecole Hassania des Travaux Publics. C'est un stage permettant l'apprentissage en situation réelle du métier de l'ingénieur que ce soit sur le plan technique ou encore sur celui du management des hommes et des organisations.

C'est donc en ce sens que j'ai, en tant que stagiaire, intégré la société EURAFRIC INFORMATION et précisément son département informatique, afin de décrire son organisation et de travailler sur une étude de conception et d'expérimentation aboutissant à une production réelle.

Remerciements

Je remercie d'abord Madame El Boukhari Ibtissam, présidente du directoire, pour l'opportunité d'effectuer le stage chez Eurafric Informations.

Je remercie ensuite Madame Chadqui Loubna et Monsieur Kilani Reda pour l'organisation des réunions de mise au point et pour leur encadrement tout au long du stage.

Je remercie également Madame Bouzidi Oumaima pour l'encadrement technique avec les différents outils informatiques utilisés ainsi que la résolution des blocages.

Enfin, je remercie Madame Dernour Rajaâ, Madame Bouchrit Khadija et Madame Bousagi Aicha pour l'installation chez Eurafric Informations et ma prise en charge en tant que stagiaire.

Sommaire

- I. Présentation de l'organisme d'accueil
 - a. Bank of Africa, BMCE Group
 - b. Eurafric Information
- II. Problématique du stage
 - a. Big Data & Data Engineering
 - b. EDD & Data Quality
- III. Contexte du stage
 - a. Le projet INTELAKA
 - b. Diagramme BPMN de demande d'un crédit
- IV. Présentation du modèle de Data Quality réalisé
 - a. Outils utilisés
 - b. Contrôles de qualité à effectuer
 - c. Quelques captures d'écran avec commentaires
- V. Ouverture

I. Présentation de l'organisme d'accueil

a. Bank of Africa, BMCE Group



Bank of Africa (anciennement BMCE Bank of Africa et Banque marocaine du Commerce extérieur) est une banque commerciale marocaine.

Fondée en 1959 en tant que banque publique, qui contribue en plus de son activité bancaire classique, au développement du commerce extérieur du Maroc, elle fut la première banque devenant filiale du groupe Finance Com.

Avec la multitude et diversification de projets informatiques qui nécessite la performance et la maturité pour un tel organisme, d'où l'implication d'une filiale IT : Eurafric Informations, pour répondre à une clientèle qui ne cesse de croître.

I. Présentation de l'organisme d'accueil

b. EURAFRIC INFORMATION



Eurafric Information gère le Système d'Information et des services clés en main de + 15 Grands Partenaires (dont Bank of Africa BMCE Group) exerçant des métiers divers : Banque, Assurance, Finance, Technologies, Crédits à la consommation et bien d'autres.

EURAFRIC INFORMATION (EAI) a été créée en octobre 2008, elle regroupe actuellement plus de 350 ingénieurs de haut niveau lauréats de grandes écoles marocaines et internationales, dotés d'expertises pointues dans plusieurs domaines : Développement, Réseaux & Télécoms, Système et Bases de Données, Sécurité Informatique, Business Intelligence, Digital...

II. Problématique du stage

a. Big Data & Data Engineering

Le Big Data fait référence à l'explosion du volume des données dans l'entreprise et des nouveaux moyens technologiques proposés par les éditeurs, en particulier de la Business Intelligence, pour y répondre.

Les objectifs de ces solutions sont de traiter un volume très important de données aussi bien structurées que non structurées, se trouvant sur des terminaux variés (PC, smartphones, tablettes, objets communicants...), produites ou non en temps réel.



Quant au data engineering, il crée des solutions pour le traitement de données volumineuses.

Le Data Engineer ou Ingénieur Data en français est l'expert qui représente le premier maillon de la chaîne du traitement des données. Grâce à son expertise, cet ingénieur veille au bon fonctionnement de l'aspect pratique de la compréhension des datas. Il est l'auteur des configurations et outils nécessaires à la collecte et à l'analyse de données.

Le data engineering est donc une discipline qui requiert des compétences spécifiques dans les infrastructures IT et l'analytique du Big Data.

II. Problématique du stage

b. Data Quality & Entrepôt de données

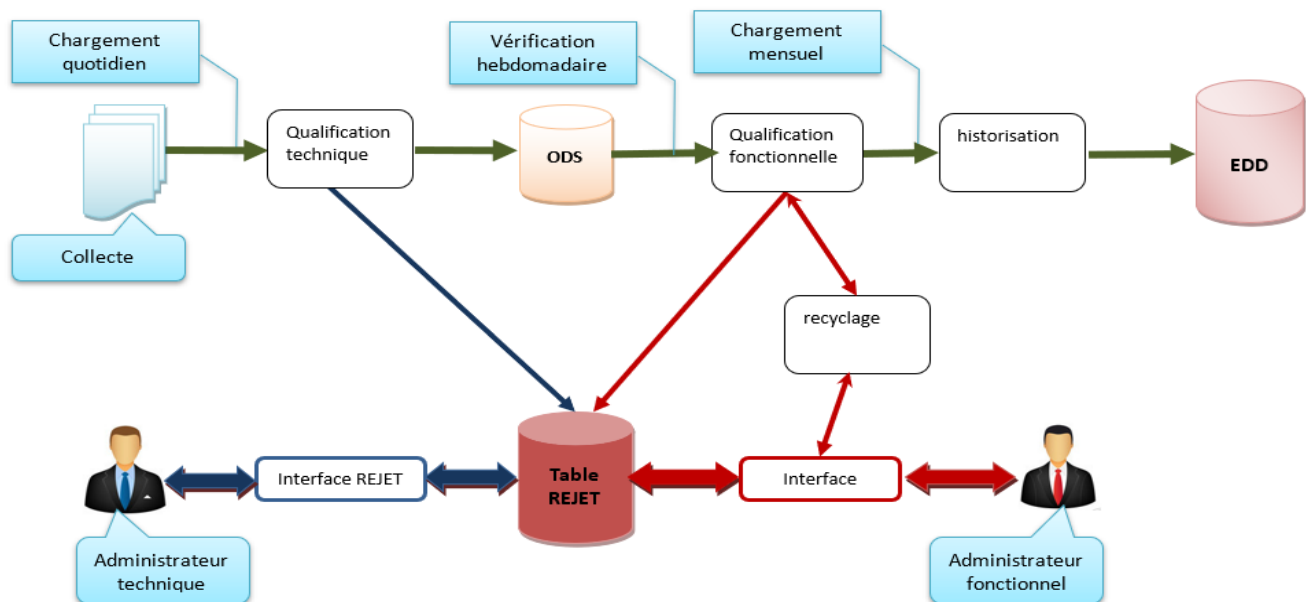
La qualité des données est très importante pour réaliser l'interopérabilité de systèmes complexes. En particulier, elle intervient dans les exigences de traçabilité, qui se manifestent dans plusieurs secteurs économiques :

- Santé et pharmacie,
- Agroalimentaire et grande distribution,
- Chimie,
- Automobile...

En informatique , la qualité de données se réfère à la conformité des données aux usages prévus, dans les modes opératoires, les processus, les prises de décision, ainsi que la planification.

De même, les données sont jugées de grande qualité si elles représentent correctement la réalité à laquelle elles se réfèrent.





❑ L'architecture de l'EDD prévoit deux administrateurs chargés de traiter les enregistrements rejetés :

- Un administrateur technique qui doit :
 - ✓ Suivre les chargements quotidiens,
 - ✓ Prévenir l'entité responsable en cas de rejet du fichier source ;
- Un administrateur fonctionnel qui doit :
 - ✓ Corriger ou forcer une donnée rejetée (non renseignée ou incorrecte),
 - ✓ Prévenir l'entité responsable pour correction à la source.

Data Quality

III. Contexte du stage

a. Le projet INTELAKA

Le projet INTELAKA est un projet lancé en février 2020 par le gouvernement marocain sous hautes instructions royales. Son but est d'encourager les entrepreneurs, les porteurs de projets ainsi que les entreprises de moins de 24 mois d'existence à financer des projets.

Ces financements bénéficient notamment d'une garantie publique à travers la caisse centrale de garantie (CCG). La CCG est un organisme assimilé à un organisme de crédit public spécialisé dans l'octroi de garantie.

Le programme de financement inclut les dépenses d'investissement ainsi que les frais d'exploitation initiaux.

Chaque mois, un reporting est réalisé et envoyé à Bank Al Maghrib.

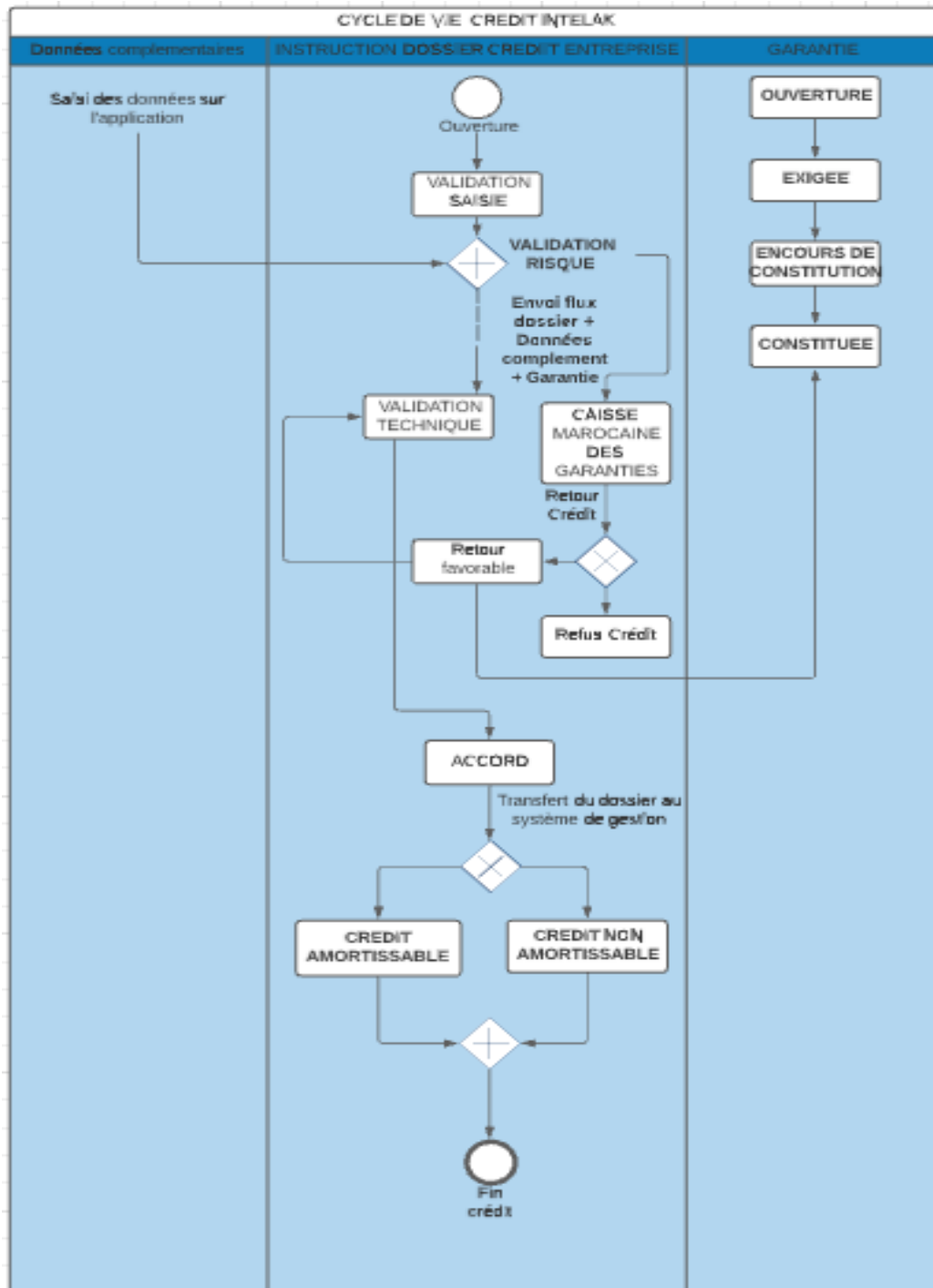
Ce reporting regroupe mensuellement les différents adhérents au programme INTILAKA qui sont répartis selon différents critères : par exemple par genre, par chiffre d'affaires de l'entreprises, par revenus, etc...

Il en résulte des tables contenant un nombre important de lignes et de colonnes et qui donc devra être sujet à de minutieux examen, car tout oubli ou toute erreur résulte en un avertissement voire en une amende de Bank al Maghrib.

D'où l'objectif de ce stage, qui consiste à réaliser un modèle de Data Quality sur le périmètre des crédits et des garanties, afin de garantir la cohérence et la fiabilité des données.

III. Contexte du stage

b. Diagramme BPMN de demande de crédit



IV. Présentation du modèle réalisé

a. Outils utilisés



Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique.



Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet.



Jupyter est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont Python, Julia, Ruby, R, ou encore Scala. C'est un projet communautaire dont l'objectif est de développer des logiciels libres, des formats ouverts et des services pour l'informatique interactive.



Spark (ou Apache Spark) est un framework open source de calcul distribué.

C'est un framework de calcul Big Data dont le rôle est d'exécuter des analyses sophistiquées. Rapide et facile d'utilisation, il peut être développé de plusieurs façons comme dans le streaming de données, le Machine Learning ou encore le traitement de graphiques

Remarques :

- Dans ce rapport, nous omettrons de montrer les étapes d'installation de ces différents outils. Dans la section qui suit, on partira du principe que tous les outils sont installés pour se focaliser uniquement sur la présentation du modèle.
- Pour des raisons de confidentialité, certains champs seront floutés.

IV. Présentation du modèle réalisé

b. Contrôles de qualité à effectuer

Les contrôles à effectuer sur les tables en entrées sont les suivants :

- ✓ Vérification des doublons fonctionnels ;
- ✓ Vérification des champs obligatoires (valeurs « null ») ;
- ✓ Contrôle des valeurs permises ;
- ✓ Traitements de cohérences entre les différentes tables.

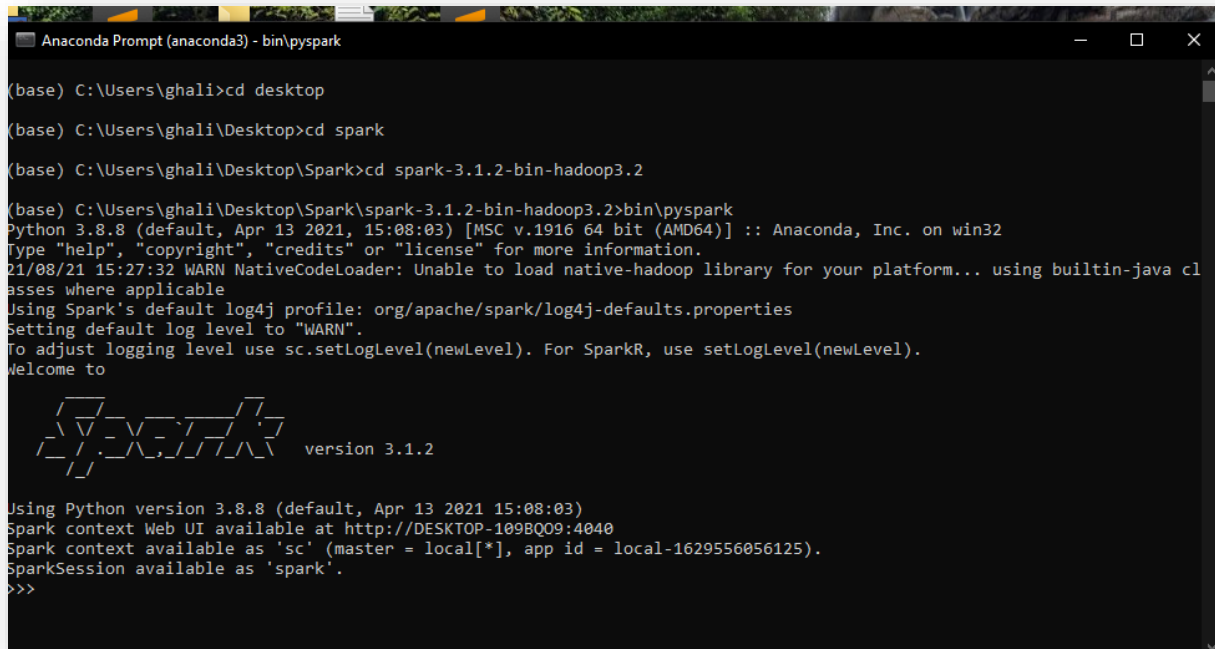
En sortie, on devra obtenir des table regroupant les différentes anomalies détectées sur les tables en entrées.

Par la suite, ces table des anomalies (ou encore tables des « rejets ») serviront à corriger les tables en entrée pour ainsi garantir la fiabilité et la cohérence des données.

IV. Présentation du modèle réalisé

c. Quelques captures d'écran avec commentaires

- Démarrage de Spark :



```
Anaconda Prompt (anaconda3) - bin\pyspark
(base) C:\Users\ghali>cd desktop
(base) C:\Users\ghali\Desktop>cd spark
(base) C:\Users\ghali\Desktop\Spark>cd spark-3.1.2-bin-hadoop3.2
(base) C:\Users\ghali\Desktop\Spark\spark-3.1.2-bin-hadoop3.2>bin\pyspark
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
21/08/21 15:27:32 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

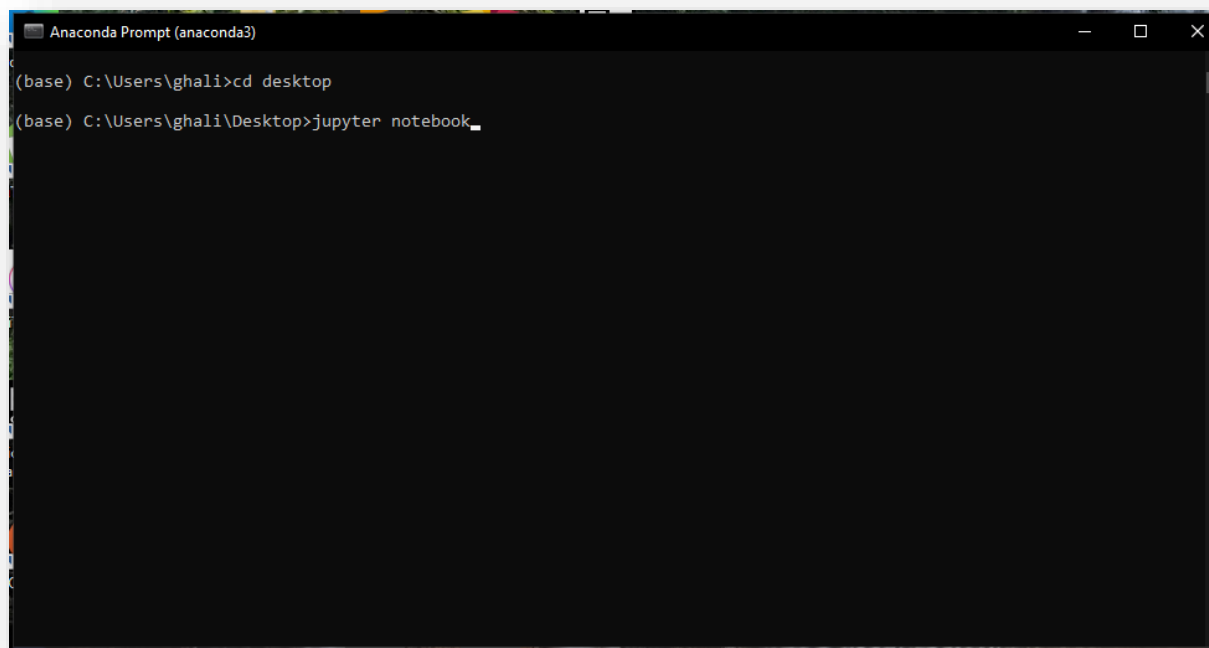
  ____      _
 / ___|    / \
 \___ \  / _ \
  ___) / / ___\
 /____/_/_\___\

version 3.1.2

Using Python version 3.8.8 (default, Apr 13 2021 15:08:03)
Spark context Web UI available at http://DESKTOP-109BQ09:4040
Spark context available as 'sc' (master = local[*], app id = local-1629556056125).
SparkSession available as 'spark'.
>>>
```

Cette succession de commandes permet de démarrer et d'initialiser le framework Spark et de charger ses différents composants et ses librairies natives.

- **Démarrage de Jupyter :**

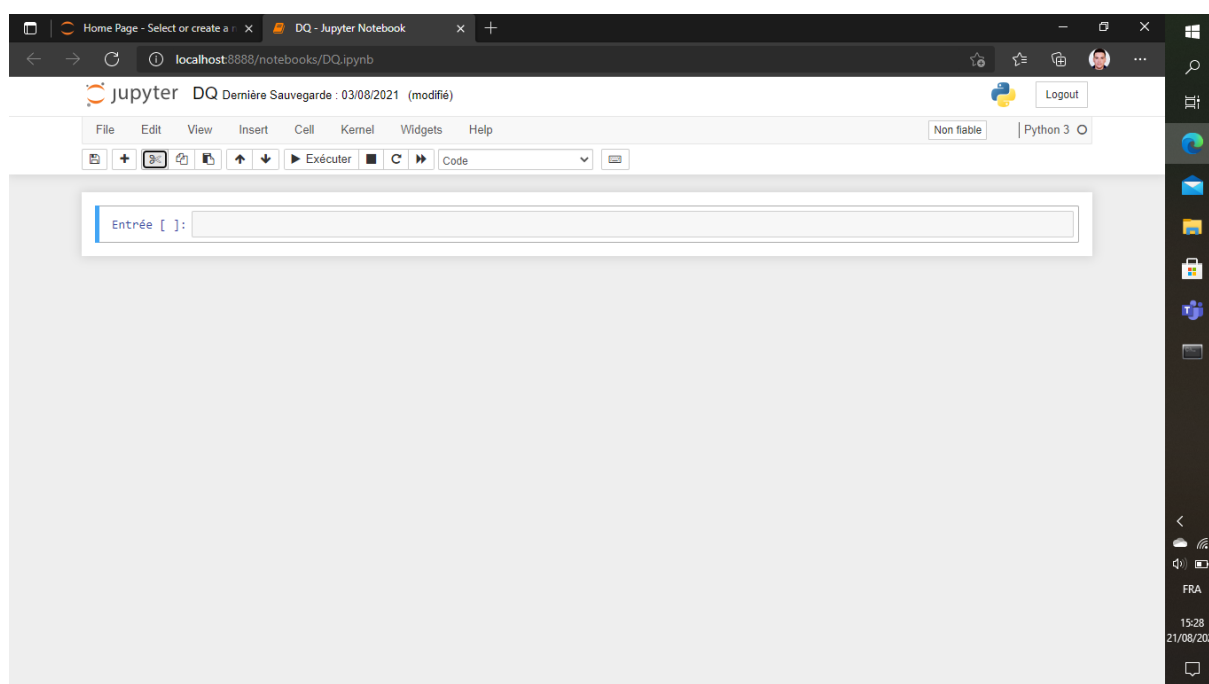


```

Anaconda Prompt (anaconda3)

(base) C:\Users\ghali>cd desktop
(base) C:\Users\ghali\Desktop>jupyter notebook_
  
```

Ces deux commandes permettent de lancer le notebook de Jupyter sur lequel sera d'abord créé un fichier Python puis où sera écrit et exécuté l'ensemble du code. (Voir capture d'écran ci-dessous)



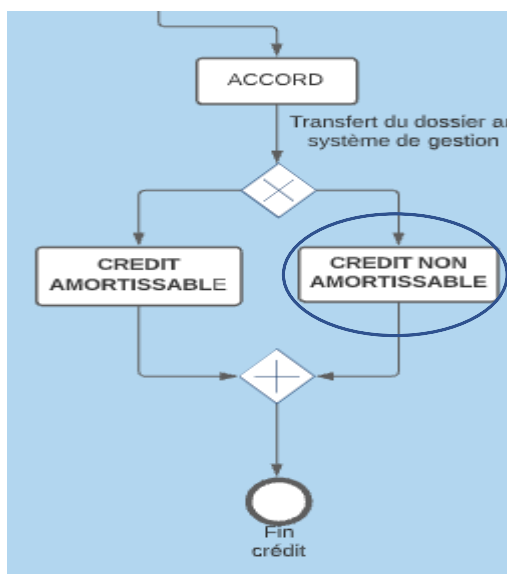
- Importation des composants et des fonctions nécessaires aux traitements de contrôles :

Dans la capture ci-dessus, il est à noter que nous avons également créé la table qui, par la suite, regroupera les différentes anomalies détectées (table des « rejets »).

```
Entrée [1]: import findspark
findspark.init()
import pyspark
from pyspark import SparkContext
sc = SparkContext("local", "Hello World App")
from pyspark.sql import SparkSession
spark = SparkSession(sc)
from pyspark.sql.functions import col
import pyspark.sql.functions as F
col_rejetee = ""
mot_rejet = ""
nom_table = ""
from datetime import date
today = date.today()
columns = ['NOM_TABLE', 'COL_REJETEE', 'ANC_VAL', 'MOT_REJET', 'DATE_REJET', 'VAL_CLE_FONC', 'NUM_LIGNE']
vals = [(0, 0, 0, 0, 0, 0, 0)]
dfr = spark.createDataFrame(vals, columns)
dfr.show()
```

NOM_TABLE	COL_REJETEE	ANC_VAL	MOT_REJET	DATE_REJET	VAL_CLE_FONC	NUM_LIGNE
0	0	0	0	0	0	0

Dans les captures d'écran qui vont suivre, nous prendrons comme exemple un échantillon de la table concernant les crédits non amortissables :



- Chargement de la table des crédits non amortissables :

```
Entrée [3]: from_table = CREDIT_NA
df6 = spark.read.format('csv').option('delimiter', ';').option('header', True).option('encoding', 'UTF-8').load('CREDIT_NA.csv')

Entrée [4]: df6.show()
```

IDENTIFIANT	MONTANT	REFERENCE	DATE_C	DATE_E	CODE	NUM_LIGNE	
00000000010000,000	B	214CIM02	M	2009-04-01 00:00:...	2013-03-31 00:00:...	TER	83
000000000170000,000	B	200CF350	M	2008-12-04 00:00:...	2029-08-31 00:00:...	GES	84
000000000043000,000	B	214CIM02	M	2009-04-01 00:00:...	2014-03-31 00:00:...	TER	85
000000000036000,000	B	214CIM02	M	2009-03-31 00:00:...	2014-03-31 00:00:...	TER	86
000000000025000,000	B	214CIM02	M	2009-04-01 00:00:...	2014-03-31 00:00:...	TER	87
000000000000000,000	L	234	M	null	null	OUV	88
000000000020000,000	B	214CIM02	M	2013-04-03 00:00:...	2016-01-19 00:00:...	TER	89
000000000050000,000	B	214CIM02	M	2009-05-04 00:00:...	2015-04-25 00:00:...	TER	90
000000000048000,000	B	214CIM02	M	2009-04-16 00:00:...	2014-03-24 00:00:...	TER	91
000000000100000,000	B	214CIM02	M	2009-04-16 00:00:...	2013-01-04 00:00:...	TER	92
000000000030000,000	B	214CIM02	M	2009-04-21 00:00:...	2014-03-24 00:00:...	TER	93
000000000041000,000	B	214CIM02	M	2009-04-20 00:00:...	2016-02-29 00:00:...	TER	94
000000000220000,000	B	200IHV08	M	2009-04-17 00:00:...	2034-08-31 00:00:...	GES	95
000000000150000,000	B	214CIM02	M	2009-04-28 00:00:...	2015-03-29 00:00:...	TER	96
000000000060000,000	B	214CIM02	M	2009-04-28 00:00:...	2014-03-31 00:00:...	TER	97
00000000001000,000	L	120	M	2012-03-12 00:00:...	9999-12-31 00:00:...	TER	98
000000000185000,000	B	200IHV08	M	2009-04-06 00:00:...	2034-04-30 00:00:...	GES	99
000000000110000,000	B	214CIM02	M	2013-09-04 00:00:...	2015-09-03 00:00:...	TER	100
000000000197000,000	B	200CF350	M	2012-11-26 00:00:...	2035-12-31 00:00:...	GES	101
000000000270000,000	B	200CF350	M	2005-06-09 00:00:...	2025-06-30 00:00:...	GES	102

only showing top 20 rows

Pour cette table, nous effectuerons dans un premier temps les contrôles suivants :

- Doublons fonctionnels sur le champ « IDENTIFIANT » ;
- Valeurs obligatoires sur tous les champs sauf « NUM_LIGNE » ;
- Valeurs permises sur le champ « CODE » ;

Puis nous effectuerons un traitement de cohérence avec une autre table qui sera présentée par la suite.

- **Contrôle des doublons fonctionnels** :

```
Entrée [16]: col_rejetee = "IDENTIFIANT"
mot_rejet = "DOUBLON"
dfd6i=df6.groupBy("IDENTIFIANT").count().filter("count > 1")
dfd6 = dfd6i.join(df6,dfd6i.IDENTIFIANT == df6.IDENTIFIANT)
dfd6.count()

Out[16]: 4

Entrée [17]: for i in range(dfd6.count()):
    newRow = spark.createDataFrame([(nom_table, col_rejetee, dfd6.collect()[i][0], mot_rejet, today.strftime("%d/%m/%Y"), dfd6.col
    dfr = dfr.union(newRow)
dfr.toPandas().to_csv('t_rejets_p.csv', sep=';', index=False)
```

La dernière ligne du premier bloc de code contient la fonction count() avec un output de 4 : cela signifie que le contrôle des doublons a détecté 4 lignes, chacune d'elles n'étant pas unique. Dans ce cas précis, et comme il sera présenté plus tard, ces 4 doublons correspondent à deux dossiers en double (2*2=4).

Par la suite, le deuxième bloc de code est constitué d'une boucle for qui alimentera la table des rejets avec les 4 lignes détectées.

- **Contrôle des champs obligatoires :**

```
Entrée [19]: df6 = spark.read.format('csv').option('delimiter',';').option('header',True).option('encoding','UTF-8').load('CREDIT_NA.csv')
mot_rejet = "VALEURS OBLIGATOIRES"
L=df6.columns
x=""
for i in range(len(L)-1):
    x=L[i]
    dfo6=df6.filter(F.col(x).isNull())
    print(dfo6.count())
    for j in range(dfo6.count()):
        if dfo6.collect()[j][0] is None :
            newRow = spark.createDataFrame([(nom_table, x, 'null', mot_rejet,today.strftime("%d/%m/%Y"),'null', dfo6.collect()[j][1])])
            dfr = dfr.union(newRow)
        elif dfo6.collect()[j][0] is not None :
            newRow = spark.createDataFrame([(nom_table, x, 'null', mot_rejet,today.strftime("%d/%m/%Y"),dfo6.collect()[j][1],dfo6.collect()[j][2])])
            dfr = dfr.union(newRow)
dfr.toPandas().to_csv('t_rejets_p.csv', sep=';', index=False)
```

0
0
0
10
10
0

Ici, l'output obtenu signifie :

- Qu'au niveau du 4^{ème} champ, 10 lignes ont une valeur null (champ vide) ;
- Qu'au niveau du 5^{ème} champ, 10 lignes ont une valeur null (champ vide).

La suite du code permet d'alimenter la table des rejets.

- **Contrôle des valeurs permises** :

Ici, le contrôle des valeurs permises porte sur le champ « CODE » : c'est-à-dire que toutes les valeurs contenues dans ce champ doivent nécessairement être également contenues dans une liste prédéfinie.

```
Entrée [21]: df6 = spark.read.format('csv').option('delimiter',';').option('header',True).option('encoding','UTF-8').load('CREDIT_NA.csv')
col_rejetee = "CODE"
mot_rejet = "VALEURS PERMISES"
df_p = spark.read.format('csv').option('delimiter',';').option('header',True).option('encoding','UTF-8').load('E_P.csv')
df_p = df_p.filter('NOM_TABLE == "CREDIT_NA"')
dfp6=df_p.join(df6, col('CODE_S') == col('CODE'), "right").filter('CODE_S IS NULL')
dfp6.count()
```

Out[21]: 2

Dans ce cas-ci, nous obtenons un output de 2 : cela signifie qu'il y a 2 lignes où le code n'appartient pas à la liste des codes prédéfinie.

S'en suit alors l'alimentation de la table de rejets :

```
Entrée [23]: for i in range(dfp6.count()):
    newRow = spark.createDataFrame([(nom_table, col_rejetee, dfp6.collect()[i][9], mot_rejet, today.strftime("%d/%m/%Y"), dfp6.col.
    dfr = dfr.union(newRow)
dfr.toPandas().to_csv('t_rejets_p.csv', sep=';', index=False)
```


- Table des rejets :

Pour récapituler, nous avons obtenu :

- 4 lignes pour les doublons ;
- 10 lignes pour les champs obligatoires ;
- 2 lignes pour les valeurs permises.

```

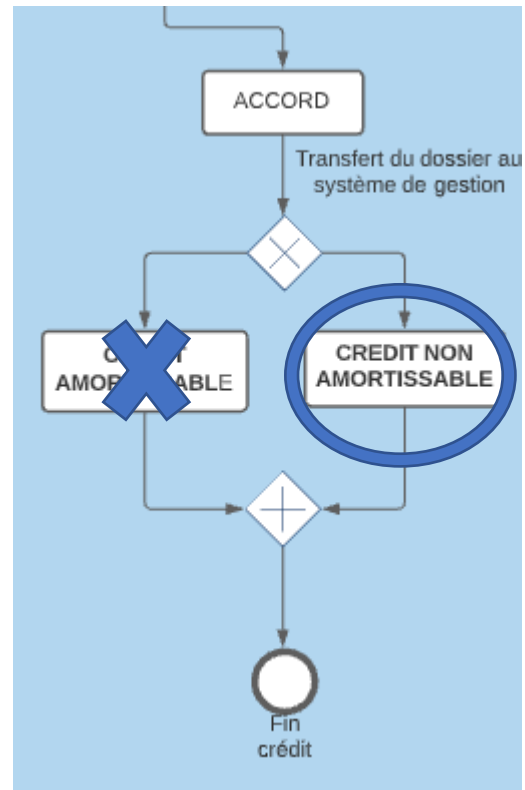
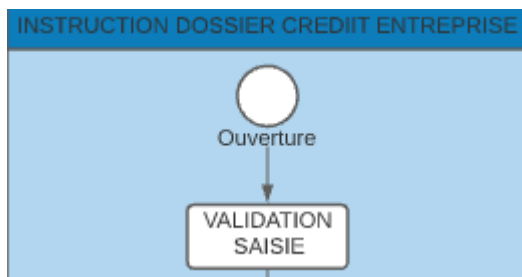
C:\Users\ghali\Desktop\t_rejets_p.csv - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

t_rejets_p.csv
1 NOM_TABLE;COL_REJETEE;ANC_VAL;MOT_REJET;DATE_REJET;VAL_CLE_FONC;NUM_LIGNE
2 0;0;0;0;0;0;0
3 CREDIT_NA;IDENTIFIANT;00000000000000000000;DOUBLON;27/08/2021;00000000000000000000;83
4 CREDIT_NA;IDENTIFIANT;00000000000000000000;DOUBLON;27/08/2021;00000000000000000000;833
5 CREDIT_NA;IDENTIFIANT;00000000000000000000;DOUBLON;27/08/2021;00000000000000000000;1079
6 CREDIT_NA;IDENTIFIANT;00000000000000000000;DOUBLON;27/08/2021;00000000000000000000;10799
7 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;88
8 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;60
9 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;109
10 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;115
11 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;121
12 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;300
13 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;494
14 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;496
15 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;698
16 CREDIT_NA;DATE_C;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;775
17 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;88
18 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;60
19 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;109
20 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;115
21 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;121
22 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;300
23 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;494
24 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;496
25 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;698
26 CREDIT_NA;DATE_E;null;VALEURS OBLIGATOIRES;27/08/2021;00000000000000000000;775
27 CREDIT_NA;CODE;XXX;VALEURS PERMISES;27/08/2021;00000000000000000000;833
28 CREDIT_NA;CODE;YYY;VALEURS PERMISES;27/08/2021;00000000000000000000;10799
29

```

- **Traitement de cohérence :**

Pour le traitement de cohérence, nous allons dans un premier temps revenir au modèle BPMN présenté précédemment :



Le but de ce traitement de cohérence est de détecter les dossiers qui ont été transmis au système de gestion des crédits non amortissables mais qui ne sont pas présents dans l'instruction dossier crédit entreprises : logiquement, chaque dossier saisi dans la table instruction dossier doit nécessairement être également contenu dans la table des crédits non amortissables.

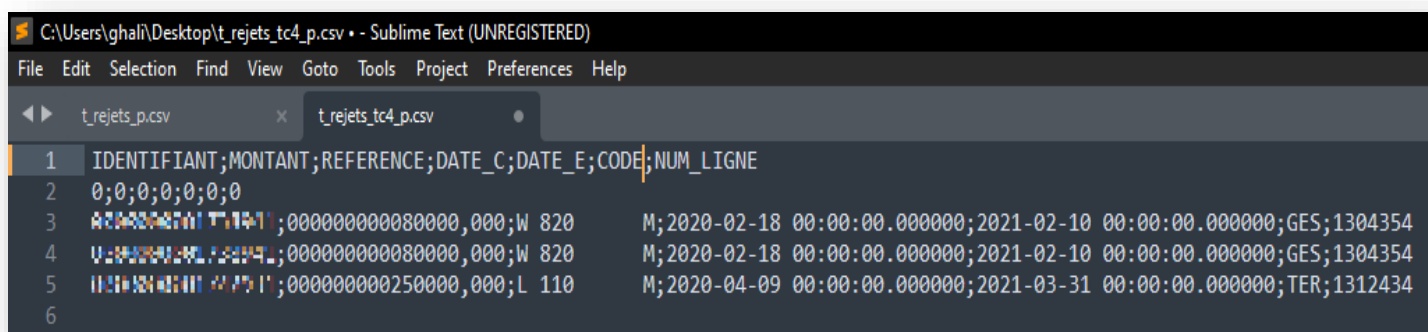
```
Entrée [25]: columns = ['IDENTIFIANT','MONTANT','REFERENCE','DATE_C','DATE_E','CODE','NUM_LIGNE']
vals = [[0, 0, 0, 0, 0, 0, 0]]
df_tc4 = spark.createDataFrame(vals, columns)
df4 = spark.read.format('csv').option('delimiter',';').option('header',True).option('encoding','UTF-8').load('G_D.csv')
df4i = df4.filter('CODE_G_D IN ("01", "02")')
df5 = spark.read.format('csv').option('delimiter',';').option('header',True).option('encoding','UTF-8').load('G_C.csv')
df4i = df4i.join(df5, df4i.IDENTIFIANT == df5.IDENTIFIANT).select('IDENTIFIANT_REFERENCE')
df6 = spark.read.format('csv').option('delimiter',';').option('header',True).option('encoding','UTF-8').load('CREDIT_NA.csv')
df4i = df6.join(df4i, col('IDENTIFIANT') == col('IDENTIFIANT_REFERENCE'))
df2 = spark.read.format('csv').option('delimiter',';').option('header',True).option('encoding','UTF-8').load('IDCE_A.csv')
df_tc4i = df4i.join(df2, col('IDENTIFIANT') == col('REFERENCE_SECONDAIRE'), "left").filter('REFERENCE_SECONDAIRE IS NULL')
df_tc4i.count()

Out[25]: 682

Entrée [26]: df_tc4i = df_tc4i.limit(3)
for i in range(df_tc4i.count()):
    newRow = spark.createDataFrame([(df_tc4i.collect()[i][0], df_tc4i.collect()[i][1], df_tc4i.collect()[i][2], df_tc4i.collect()[i][3], df_tc4i.collect()[i][4], df_tc4i.collect()[i][5], df_tc4i.collect()[i][6])])
    df_tc4 = df_tc4.union(newRow)
df_tc4.toPandas().to_csv('t_rejets_tc4_p.csv', sep=';', index=False)
```

Dans ce cas-ci, nous obtenons un output de 682 : c'est-à-dire qu'il y a 682 dossiers présents dans la table des crédits non amortissables mais absents de la table des instructions dossiers.

Comme le nombre de dossiers ici est important, nous avons limité le résultat des contrôles à seulement 3 lignes pour accélérer le processus d'alimentation de la table des anomalies.



```
C:\Users\ghali\Desktop\t_rejets_tc4_p.csv - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
t_rejets_p.csv x t_rejets_tc4_p.csv
1 IDENTIFIANT;MONTANT;REFERENCE;DATE_C;DATE_E;CODE;NUM_LIGNE
2 0;0;0;0;0;0;0
3 00000000000000000000;000;W 820 M;2020-02-18 00:00:00.000000;2021-02-10 00:00:00.000000;GES;1304354
4 00000000000000000000;000;W 820 M;2020-02-18 00:00:00.000000;2021-02-10 00:00:00.000000;GES;1304354
5 00000000000000000000;000;L 110 M;2020-04-09 00:00:00.000000;2021-03-31 00:00:00.000000;TER;1312434
6
```

V. Ouverture

Finalement, l'intérêt majeur de ces contrôles de Data Qualité est de pouvoir retracer les anomalies détectées dans les tables de rejets à la source, pour pouvoir idéalement les corriger ou au pire des cas les forcer.

En ce sens, ces rejets seront remontés dans l'application de gestion de l'EDD afin d'être traités par l'administrateur fonctionnel.

Ce stage a été évidemment enrichissant sur plusieurs points, notamment les deux suivants :

- D'abord car il m'a permis de découvrir et de me familiariser avec de nouveaux outils informatiques dans le monde du Big Data ;
- Mais aussi car il m'a permis de découvrir en profondeur le programme INTELAKA avec ses différents paramètres de crédits et garanties.