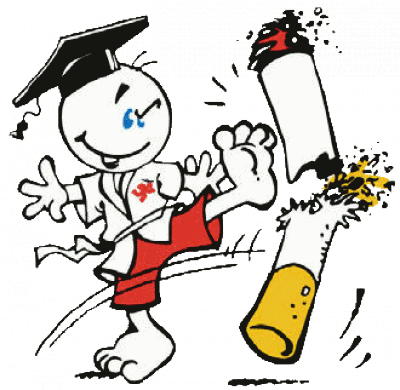
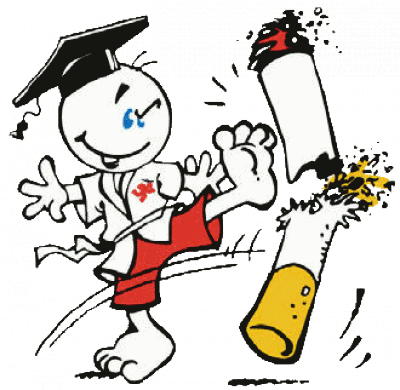
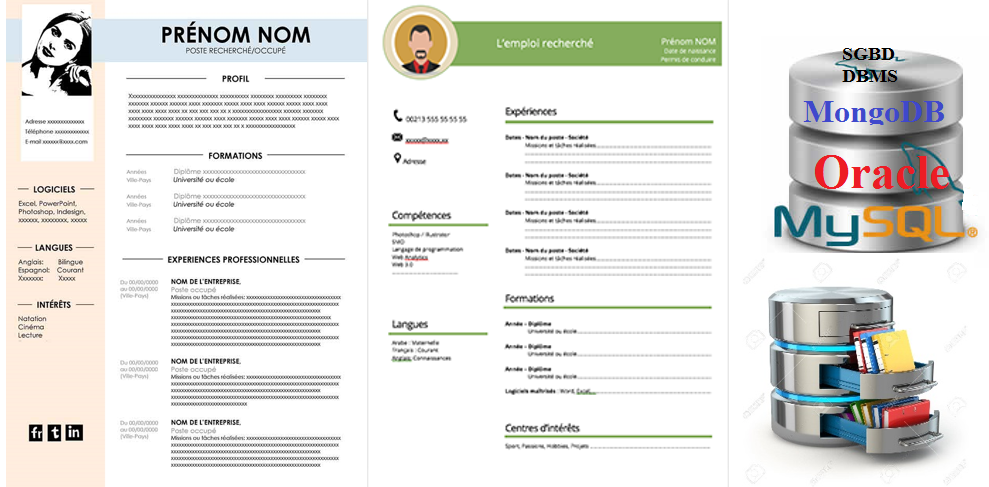
|  |  |
| --- | --- |
| **Université Sorbonne Paris Nord (USPN)** | Année Universitaire |
| Institut Galilée (**IG**) | 2020 / 2021 |
| Laboratoire d’Informatique de Paris Nord (**LIPN**) | ***La Maison des Sciences Numérique (La MSN)*** |
| ***La Data e(s)t le monde de demain*** | **M. F. Boufarès** |
| faouzi.boufares@sorbonne-paris-nord.fr | [boufares@lipn.univ-paris13.fr](mailto:boufares@lipn.univ-paris13.fr) |
| **La Data/La Donnée** | ***Bases de Données Avancées - Entrepôts de Données*** |





**Projet Annuel : DES BASES AUX ENTREPÔTS DE DONNÉES,**

**Données structurées ou NON structurées (Oracle, MySQL ou MongoDB ?)**

***From PDF Files to SQL or NoSQL Data Bases***

Exploration des Curriculum-Vitae (CV) des étudiant.e.s

qui candidatent pour le Maser 2 Informatique Exploration Informatique des Données et Décisionnel.

Réalisé le 15 Mars 2021 (Partie 2) par l’équipe :

|  |  |  |  |
| --- | --- | --- | --- |
| ***Groupe*** | ***Binôme*** | ***NOM*** | ***Prénom*** |
| G1 | B18 | BERHILI | Faïza |
| BOUSSAC | Emiliano |
| B19 | ALSADIK | Ammar |
| CHAMBRIER | Julian |
| ***Encadrant (Enseignant chercheur) Monsieur F.B BOUFARES*** | | | |



Table des matières

[Avant propos](#_heading=h.30j0zll) **3**

[Introduction](#_heading=h.1fob9te) **3**

[Schémas de Base de Données : Structures de données & Métadonnées](#_heading=h.3znysh7) **4**

[***2.1. Le schéma conceptuel de données (EER)***](#_heading=h.rxr5gd2xj4k0) ***4***

[***2.2. Le schéma conceptuel de données (UML)***](#_heading=h.q9eydaek71rf) ***7***

[***2.3. Le schéma logique de données (Schéma relationnel)***](#_heading=h.xf8qn0nk2kdo) ***8***

[***2.4. Le dictionnaire de données & les métadonnées***](#_heading=h.3gselp5taam7) ***9***

[***2.5. Expression des contraintes sur les données***](#_heading=h.3mupjjabnz3)11

[3. Exemple de contenu de la base de données (instance de BD)](#_heading=h.2s8eyo1) **14**

[***3.1. Exemple d’instances de la BD des CV***](#_heading=h.17dp8vu) ***15***

[***3.2. Création des structures de données (SQL-ORACLE)***](#_heading=h.3rdcrjn) ***20***

[***3.2.1. Extraction automatique des données.***](#_heading=h.26in1rg) ***21***

[***3.2.3. Insertion des données***](#_heading=h.ho1610olko8n) ***22***

[***3.2.4. Qualité des données.***](#_heading=h.lnxbz9) ***24***

[***3.2.5. Machine Learning.***](#_heading=h.35nkun2) ***25***

[***3.2.6. Comparaison des SGBD.***](#_heading=h.1ksv4uv) ***27***

[***3.2.7 Data Integration - Intégration des données de bases de données hétérogènes - Talend***](#_heading=h.44sinio) ***28***

[***3.2.8 Business Intelligence - Tableau***](#_heading=h.2jxsxqh) ***32***

[4. Sujet du Projet Annuel](#_heading=h.1y810tw) **35**

[***4.1. : Conception & Implantation de Bases de Données***](#_heading=h.4i7ojhp) ***35***

[***4.2. : Conception & Implantation d’Entrepôts de Données***](#_heading=h.2xcytpi) ***35***

[Conclusion](#_heading=h.1ci93xb) **36**

# Avant propos

Afin d'exécuter le projet et voir toutes les fonctionnalités possibles, il est nécessaire de se référencer au fichier nommé README.txt. Ce document reprend pas à pas les instructions à exécuter afin de tester le projet.

Le document G1\_2021-MFB-ProjetM2EID2-SQLnoSQL.docx (ce document) permet d'expliquer les choix d'implémentation et d'avoir plus de détails sur le projet. En effet, il permet d'expliquer le choix de structure, de Data Cleaning, de Machine Learning, de Data Integration …

# Introduction

Le système d’information de l’Université Sorbonne Paris Nord (USPN) est articulé autour de la base de données de nom BD-USPN. Elle contient les données relatives aux : (i) Ressources Humaines (Tels que les Enseignant.e.s, les Chercheur.se.s , les Ingénieur.e.s, les Administratifs) (ii) aux Établissements (Tels que les Universités, les Écoles, les Instituts, Hôpitaux, Entreprises, Organismes, Éditeurs) partenaires, (iii) Étudiant.e.s (Tels que les anciens, les actuels, les candidats nationaux et internationaux retenus et non retenus). Une partie de cette base de données fera l’objet de notre étude.

En effet, dans le cadre de notre projet annuel intitulé «  **DES BASES AUX ENTREPÔTS DE DONNÉES, données structurées ou NON structurées (Oracle, MySQL ou MongoDB ?) ; *From PDF Files to SQL or NoSQL Data Bases*** ».

Il s’agit de la construction de données structurées à partir de données NON structurées (à partir de documents papiers ou numériques qui correspondent aux Curriculum-Vitae (CV) des étudiant.e.s qui candidatent pour le Master 2 Informatique Exploration Informatique des Données et Décisionnel. Cette BD devra contenir les candidatures sur plusieurs années afin d’étudier le profil des candidat.e.s en entrée et évidemment ceux en sortie c’est-à-dire les diplômé.e.s.

Elle est décrite par le schéma conceptuel ainsi que le schéma relationnel (les tables) ci-après.

Ce document se présente comme suit. La première partie traite de la construction des entrepôts de données, avec la conception et la création des tables nécessaires au bon fonctionnement de notre entrepôt. La seconde partie traite des insertions dans ces tables. Nous verrons que ces insertions se font après la vérification préalable de contraintes stockées dans des méta-tables. La dernière partie correspond à la validation de notre entrepôt de données grâce à des requêtes. Ces requêtes permettent aussi d'observer les tendances, les habitudes, les profils qui se dégagent pour l'acceptation ou le refus des candidats dans le Master 2 - Exploration Informatique des Données et Décisionnel.

La chose primordiale pour notre équipe dans ce projet, était de construire un algorithme automatique avec de bon résultats, sans réelle interaction humaine afin d’extraire les informations des CV et les insérer dans des tables.

On ne voulait pas simplement faire des Insert Into et Create table. On voulait vraiment quelque chose digne d’un projet et tout ça de manière entièrement automatisé. On gère donc plusieurs gestionnaires de bases de données et des outils d’intégration et de visualisation avec toutes les spécificités nécessaires. On va donc vous présenter nos concepts.

# Schémas de Base de Données : Structures de données & Métadonnées

Le schéma conceptuel de données (SCD) ou le modèle conceptuel de données (MCD) peut être décrit selon plusieurs formalismes.

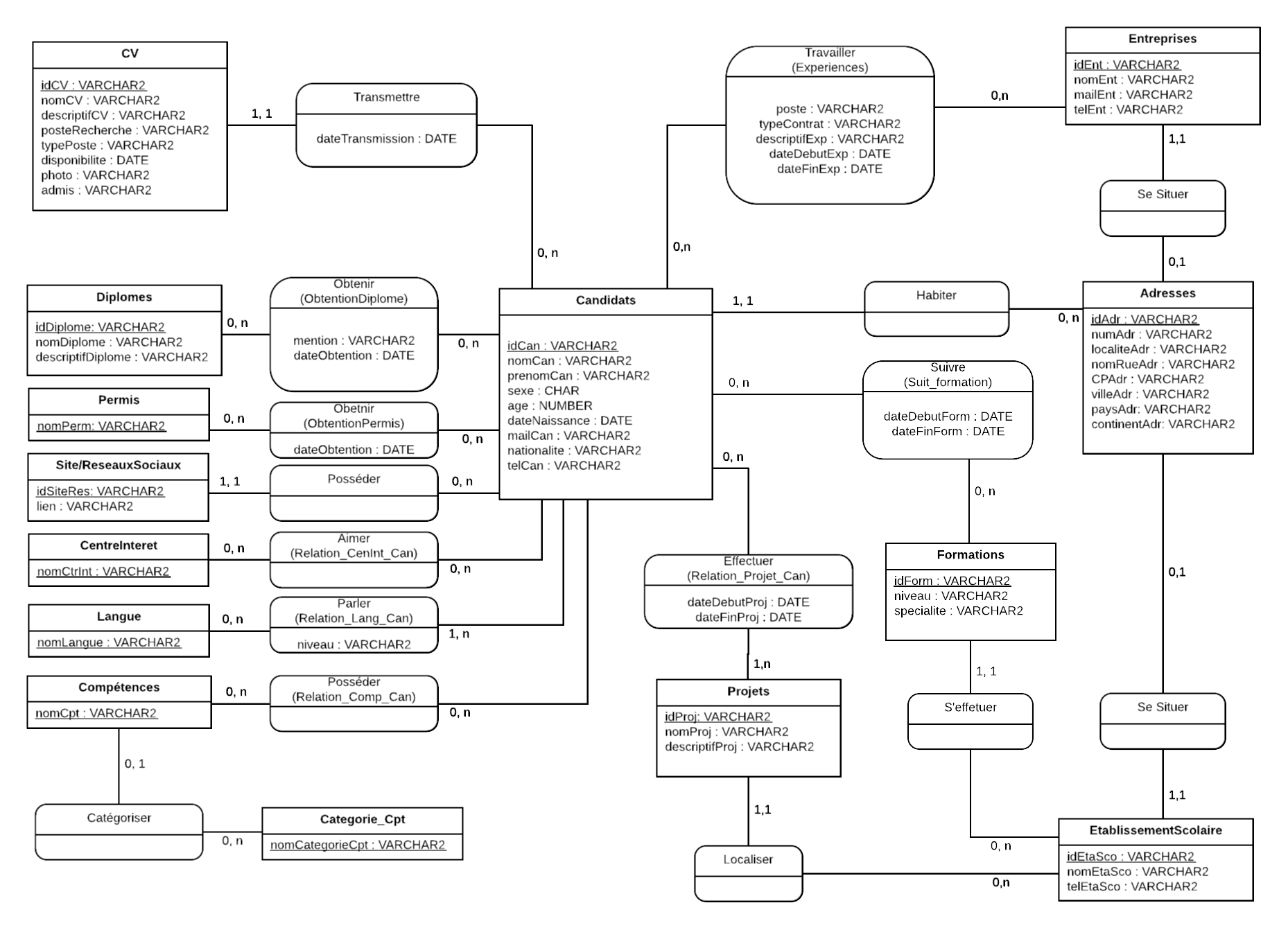
Dans le formalisme « Entity-Relationship (ER) ou Extended Entity-Relationship (EER), appelé aussi EA », la description des données comporte plusieurs entités (ou type d’entité = les rectangles) reliées entre elle par des associations (ou type d’association = les ovales), les arcs sont étiquetés par des contraintes de cardinalité (Exemple 1,1 ; 1,n et 0,n). La figure 1 représente le schéma conceptuel de la Base de Données, selon le formalisme Entité/Association (EA).

Dans le formalisme « Unified Modeling Language : UML », La description des données comporte plusieurs classes reliées entre elles par des arcs étiquetés par des contraintes de multiplicité (Exemple 1 ; 1,\* et \*) :

La structure (le schéma) de la BD est donnée ci-dessous.

## ***2.1. Le schéma conceptuel de données (EER)***

Le **schéma conceptuel de données** (SCD) ou **le modèle conceptuel de données** (MCD) est décrit ci-dessous selon formalisme « Entity-Relationship(ER, EA) ou Extended Entity-Relationship(EER) ».





**Les entités :**

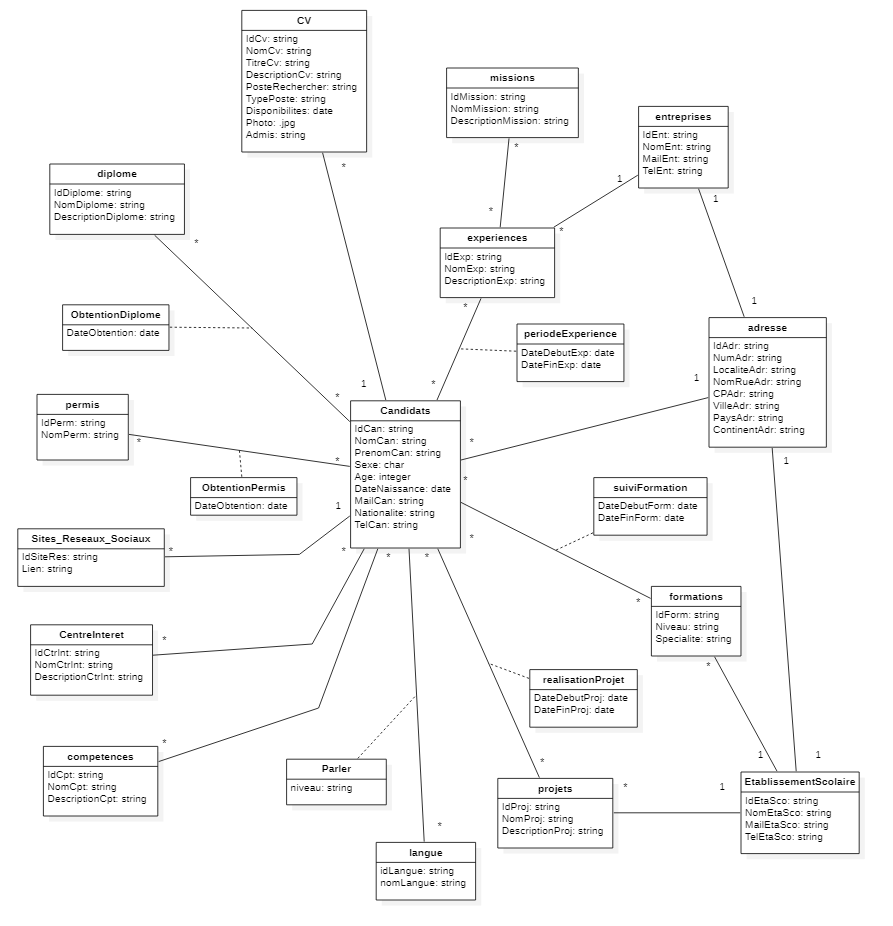
|  |  |
| --- | --- |
| **Détails des Entités** | |
| **Entité** | **Attributs** |
| *ADRESSES* | **IdAdr**, NumAdr, LocaliteAdr, NomRueAdr, CPAdr, VilleAdr, PaysAdr, ContinentAdr |
| *CANDIDATS* | **IdCan**, NomCan, PrenomCan, Sexe, Age, DateNaissance, MailCan, Nationalite, TelCan |
| *CATEGORIE\_CPT* | **NomCategorie** |
| *CENTREINTERETS* | **NomCtrInt** |
| *COMPETENCES* | **NomCpt** |
| *CV* | **IdCv**, NomCv, TitreCv, DescriptionCv, PosteRechercher, TypePoste, Disponibilites, Photo, Admis |
| *DIPLOMES* | **IdDiplome**, NomDiplome, DescriptifDiplome |
| *ENTREPRISES* | **IdEnt**, NomEnt, MailEnt,TelEnt |
| *ETABLISSEMENTSCOLAIRE* | **IdEtaSco**, NomEtaSco, MailEtaSco,TelEtaSco |
| *FORMATIONS* | **IdForm**, Niveau, Specialite |
| *LANGUES* | **NomLangue** |
| *PROJETS* | **IdProj**, NomProj, DescriptionProj |
| *PERMIS* | **NomPerm** |
| *RESEAUX* | **IdSiteRes**, Lien |

**Les associations :**

|  |  |  |
| --- | --- | --- |
| **Détails des Associations** | | |
| Association | Nom définitif de l'association | Attributs supplémentaires |
| *AIMER* | *RELATION\_CENTINT\_CAN* | Aucun attribut supplémentaire |
| *CATEGORISER* | *CATEGORISER* | Aucun attribut supplémentaire |
| *EFFECTUER* | *RELATION\_PROJET\_CAN* | DateDebutProj, DateFinProj |
| *HABITER* | *HABITER* | Aucun attribut supplémentaire |
| *LOCALISER* | *LOCALISER* | Aucun attribut supplémentaire |
| *OBTENIR* | *OBTENTIONDIPLOME* | DateObtention, Mention |
| *OBTENIR* | *OBTENTIONPERMIS* | DateObtention |
| *PARLER* | *RELATION\_LANG\_CAN* | Niveau |
| *POSSEDER* | *POSSEDER* | Aucun attribut supplémentaire |
| *POSSEDER* | *RELATION\_COMP\_CAN* | Aucun attribut supplémentaire |
| *S'EFFECTUER* | *S'EFFECTUER* | Aucun attribut supplémentaire |
| *SE SITUER(1)* | *SE SITUER(1)* | Aucun attribut supplémentaire |
| *SE SITUER(2)* | *SE SITUER(2)* | Aucun attribut supplémentaire |
| *SUIVRE* | *SUIT\_FORMATION* | DateDebutForm, DateFinForm |
| *TRANSMETTRE* | *TRANSMETTRE* | DateTransmission |
| *TRAVAILLER* | *EXPERIENCES* | Poste, TypeContrat, DescriptifExp, DateDebutProj, DateFinProj |

## ***2.2. Le schéma conceptuel de données (UML)***

Le **schéma conceptuel de données** (SCD) ou **le modèle conceptuel de données** (MCD) est décrit ci-dessous selon formalisme « Unified Modeling Language : UML ».



## ***2.3. Le schéma logique de données (Schéma relationnel)***

**Le Schéma relationnel de la Base de Données (Modèle Logique de Données MLD) est composé de plusieurs tables.** Elles sont présentées ci-dessous :

|  |  |  |  |
| --- | --- | --- | --- |
| **Schéma relationnel** | | | |
| Nom de la table | Clé primaire  | Clé(s) étrangère(s) 🡩 | Autre(s) attribut(s) |
| **ADRESSES** | IdAdr |  | NumAdr, LocaliteAdr, NomRueAdr, CPAdr, VilleAdr, PaysAdr, ContinentAdr NumAdr, LocaliteAdr, NomRueAdr, CPAdr, VilleAdr, PaysAdr, ContinentAdr |
| **CANDIDATS** | IdCan | IdAdr | NomCan, PrenomCan, Sexe, Age, DateNaissance, MailCan, Nationalite, TelCan |
| **CATEGORIE\_CPT** | NomCategorie |  |  |
| **CENTREINTERET** | NomCtrInt |  |  |
| **COMPETENCES** | NomCpt | NomCategorie |  |
| **CV** | IdCv | IdCan | NomCv, DescriptionCv, PosteRechercher, TypePoste, Disponibilites, Photo, Admis, DateTransmission |
| **DIPLOMES** | IdDiplome |  | NomDiplome, Mention, DescriptifDiplome |
| **ENTREPRISES** | IdEnt | IdAdr | NomEnt, MailEnt, TelEnt |
| **ETABLISSEMENTSCOLAIRE** | IdEtaSco | IdAdr | NomEtaSco, TelEtaSco |
| **EXPERIENCES** | IdCan, IdEnt | IdCan, IdEnt | Poste, TypeContrat, DescriptifExp, DateDebutExp, DateFinExp |
| **FORMATIONS** | IdForm | IdEtaSco | Niveau, Specialite |
| **LANGUES** | NomLangue |  |  |
| **OBTENTIONDIPLOME** | IdCan, IdDiplome | IdCan, IdDiplome | DateObtention |
| **OBTENTIONPERMIS** | IdCan, IdPermis | IdCan , IdPermis | DateObtention |
| **PERMIS** | NomPerm |  |  |
| **PROJETS** | IdProj | IdEtaSco | NomProj, DescriptionProj |
| **RELATION\_COMP\_CAN** | IdCan, NomCpt | IdCan, NomCpt |  |
| **RELATION\_CENTINT\_CAN** | IdCan, IdCtrInt | IdCan, IdCtrInt |  |
| **RELATION\_LANG\_CAN** | IdCan, NomLangue | IdCan, NomLangue | Niveau |
| **RELATION\_PROJET\_CAN** | IdCan, IdProj | IdCan, IdProj | DateDebutProj, DateFinProj |
| **SITES\_RESEAUX** | IdSiteRes | IdCan | Lien |
| **SUIT\_FORMATIONS** | IdCan, IdForm | IdCan, IdForm | DateDebutForm, DateFinForm |

Un **dictionnaire de données** contient les détails de toutes les descriptions des différentes colonnes de toutes les tables de la BD.

Ces descriptifs constituent une partie des **métadonnées** de la BD.

Le dictionnaire de données de la BD est détaillé ci-dessous :

## ***2.4. Le dictionnaire de données & les métadonnées***

## 

Le **dictionnaire de données** de la BD est le suivant (ces descriptifs constituent un sous ensemble des **métadonnées** de la BD) :

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribut**  **Colonne** | **Signification - Description** | **Domaine -Type de données**  **Contrainte Naturelles** | **Contrainte (Meta Table)**  **Expression Régulière** |
| Admis | Réponse d'acceptation ou de refus du candidat | Caractères/Texte, 150, ACCEPTE, REFUSE ou INCONNU | '^(ACCEPTE|REFUSE|INCONNU)$' |
| Age | Age du candidat | Entier, Maximum 3, >= 0 | '^([0-9]{1,3})|NULL$' |
| ContinentAdr | Adresse (Nom du continent) | Caractères/Texte, 150, en MAJUSCULE | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| CPAdr | Adresse (Code Postal) | Caractères/Texte, 150 | '^([0-9]{5})|NULL$' |
| DateDebutExp | Date de début d'expérience | Date |  |
| DateDebutForm | Date de début de formation | Date |  |
| DateDebutProj | Date de début de projet | Date |  |
| DateFinExp | Date de fin d'expérience | Date |  |
| DateFinForm | Date de fin de formation | Date |  |
| DateFinProj | Date de fin de projet | Date |  |
| DateNaissance | Date de naissance du candidat | Date |  |
| DateTransmission | Date de transmission du Cv | Date |  |
| DescriptifCpt | Descriptif de la compétence | Caractères/Texte, 2000 |  |
| DescriptifCtrInt | Descriptif du centre d'intérêt | Caractères/Texte, 2000 |  |
| DescriptifCv | Descriptif du cv | Caractères/Texte, 2000 |  |
| DescriptifDiplome | Description du diplôme | Caractères/Texte, 2000 |  |
| DescriptifExp | Descriptif de l'expérience | Caractères/Texte, 2000 |  |
| DescriptifProj | Description du projet | Caractères/Texte, 2000 |  |
| Disponibilite | Disponibilité du candidat | Date |  |
| **IdAdr** | Numéro de l'adresse (Unique par adresse) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdCan** | Numéro du candidat (Unique par candidat) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdCv** | Numéro de Cv (Unique par Cv) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdDiplome** | Numéro de diplôme (Unique par diplôme) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdEtaSco** | Numéro de l'établissement scolaire (Unique par établissement) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdEnt** | Numéro de l'entreprise (Unique par entreprise) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdForm** | Numéro de la formation (Unique par formation) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdProj** | Numéro du projet (Unique par projet) | Caractères/Texte, 150 | '^[0-9]+$' |
| **IdSiteRes** | Numéro du site/réseau social (Unique par site/réseau social) | Caractères/Texte, 150 | '^[0-9]+$' |
| Lien | Lien vers le site / Réseau social | Caractères/Texte, 150 |  |
| LocaliteAdr | Adresse (Type de la voie) | Caractères/Texte, 150, en MAJUSCULE | '^(RUE|BOULEVARD|AVENUE|QUAI|IMPASSE|PONT|PLACE|SQUARE|ALLEE|ALLEES|VOIE|MONTEE|ESPLANADE|ROUTE|VOIRIE|CITE|CHEMIN|PARVIS)|NULL$' |
| MailCan | Mail du candidat | Caractères/Texte, 150 | '^([A-Za-z]+[A-Za-z0-9--\.\_]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4})|NULL$' |
| MailEnt | Mail de l'entreprise | Caractères/Texte, 150 | '^([A-Za-z]+[A-Za-z0-9--\.\_]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4})|NULL$' |
| Mention | Mention obtenue | Caractères/Texte, 150 | '^[A-Za-z]+[A-Za-z0-9.]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$' |
| Nationalite | Nationalité du candidat | Caractères/Texte, 150 | '^([A-Z]+)|NULL$' |
| Niveau | Le niveau de langue du candidat | Caractères/Texte, 150 | '^[A-Z][a-zç]$' |
| NomCan | Nom du candidat | Caractères/Texte, 150en MAJUSCULE | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| **NomCategorie** | Nom de la catégorie des compétences (Unique pour chaque catégorie) | Caractères/Texte, 150 | ''^([A-Z]+[(-| )?[A-Z]+]\*)|NULL$' |
| **NomCpt** | Nom de la compétence (Unique par compétence) | Caractères/Texte, 150 | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| **NomCtrInt** | Nomdu centre d'intérêt (Unique par centre d'intérêt) | Caractères/Texte, 150 | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| NomCv | Nom du Cv | Caractères/Texte, 150 | '^[A-Za-z0-9\_\-/]+\.pdf$' |
| NomDiplome | Nom du diplôme | Caractères/Texte, 150 |  |
| NomEnt | Nom de l'entreprise | Caractères/Texte, 150; en MAJUSCULE | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| NomEtaSco | Nom de l'établissement Scolaire | Caractères/Texte, 150en MAJUSCULE | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| NomExp | Nom de l'expérience | Caractères/Texte, 150 |  |
| **NomLangue** | Nom de la langue (Unique par langue) | Caractères/Texte, 150en MAJUCULE | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| **NomPerm** | Nom de permis (Unique par permis) | Caractères/Texte, 150dans la liste des permis existants | '^(AM|BSR|A|A1|A2|B|B1|B2|BE|BVA|C|C1|CE|C1E|D|D1|D2|DE|DE1)$' |
| NomProj | Nom du projet | Caractères/Texte, 150 |  |
| NomRueAdr | Adresse (Nom de la voie) | Caractères/Texte, 150 | '^[A-Za-z0-9]+((\-| )[A-Za-z0-9]+)?$' |
| NumAdr | Adresse (Numéro dans la voie) | Caractères/Texte, 150 | '^([0-9]{1,5}( BIS| TER)?)|NULL$' |
| PaysAdr | Adresse (Nom du Pays) | Caractères/Texte, 150 en MAJUSCULE | '^[A-Z]+[(-| )?[A-Z]+]\*$' |
| Poste | Poste occupé lors d'une expérience | Caractères/Texte, 150 en MAJUSCULE |  |
| PosteRechercher | Nom du poste recherché | Caractères/Texte, 150 |  |
| PrenomCan | Prénom du candidat | Caractères/Texte, 150, 1 ère lettre de chaque prénom en MAJUSCULE, le reste en minuscule | '^[A-Z][a-zéïàèîôû]+((\-| )[A-Z][a-zéïàèîôû]+)?$' |
| Sexe | Sexe du candidat | Caractères/Texte, 1, M (Masculin) ou F (Féminin) | '^(M|F)|NULL$' |
| TelCan | Téléphone du candidat | Caractères/Texte, 150 | '^((\+)?[0-9]+)|NULL$' |
| TelEnt | Téléphone de l'entreprise | Caractères/Texte, 150 | '^((\+)?[0-9]+)|NULL$' |
| TelEtaSco | Téléphone de l'établissement Scolaire | Caractères/Texte, 150 | '^((\+)?[0-9]+)|NULL$' |
| TypeContrat | Type de contrat lors d'une expérience | Caractères/Texte, 150 en MAJUSCULE |  |
| TypePoste | Type de poste recherché | Caractères/Texte, 150 |  |
| VilleAdr | Adresse (Nom de la ville) | Caractères/Texte, 150en MAJUSCULE | '^[A-Z]+[(-| )?[A-Z]+]\*$' |

## ***2.5. Expression des contraintes sur les données***

L’expression des contraintes sur les données est une étape primordiale pour valider les données. La liste des contraintes à valider est donnée ci-dessous.

La majorité des contraintes sont gérées avec des expressions régulières présentes dans des méta-tables sous les noms *META\_NomTable*.

- Adresses :

Le **numéro d'identification** doit être unique pour chaque adresse. Le **numéro de la rue** doit être un entier éventuellement suivi de BIS ou TER. Le **type de la voie dans une adresse** prend une valeur dans la liste suivante : {RUE, BOULEVARD, AVENUE, QUAI, IMPASSE, PONT, PLACE, SQUARE, ALLÉE, ALLÉES, VOIE, MONTÉE, ESPLANADE, ROUTE, VOIRIE, CITÉ, CHEMIN, PARVIS…}, les valeurs sont du type alphabétique. Le **code postal** doit être un être composé de 5 chiffres. Les **villes, pays et continents** sont constitués essentiellement de majuscules avec éventuellement des espaces ou des tirets pour ceux avec des noms composés.

- Candidats :

Le **numéro d'identification** doit être unique pour chaque candidat. Le **numéro d'identification** de l'adresse doit être présent dans la table adresse. Le **nom** d'un candidat est une chaîne alphabétique entièrement en majuscule avec éventuellement des espaces ou tirets pour les noms composés. Le **prénom** d'un candidat est une chaîne alphabétique dont la première lettre de chaque prénom (pour les prénoms composés) doit être en majuscule et tout le reste des prénoms doit être en minuscule. L'**âge** d'un candidat est un entier positif avec maximum 3 chiffres. Le **sexe** d'un candidat est représenté par un caractère présent dans la liste {M, F} pour masculin et féminin respectivement. Le **numéro de téléphone** d'un candidat doit commencer par +33 ou 0 suivi de 9 chiffres. Le **mail** d'un candidat doit correspondre à l'expression régulière des mails.

- Établissements :

Le **numéro d'identification** doit être unique pour chaque établissement scolaire. Le **numéro d'identification** de l'adresse doit être présent dans la table adresse. Le **nom de l'établissement** doit être en majuscule. Le **mail** de l'établissement doit correspondre à l'expression régulière des mails. Le **numéro de téléphone** d'un établissement doit commencer par +33 ou 0 suivi de 9 chiffres.

- Entreprises :

Le **numéro d'identification** doit être unique pour chaque entreprise. Le **numéro d'identification** de l'adresse doit être présent dans la table adresse. Le **nom de l'entreprise** doit être en majuscule. Le **mail** de l'entreprise doit correspondre à l'expression régulière des mails. Le **numéro de téléphone** d'un établissement doit commencer par +33 ou 0 suivi de 9 chiffres.

- Projets :

Le **numéro d'identification** doit être unique pour chaque projet. Le **numéro d'identification** de l'établissement scolaire doit être présent dans la table etablissement scolaire. Le **nom du projet** est composé d’une suite de lettres, de tirets ou d’espaces ou de chiffres. La **description du projet** peut utiliser n’importe quel caractère ou aucun.

- Langues :

Le **nom** de la langue doit être unique pour chaque langue. Il doit être essentiellement constitué de majuscules.

- Catégorie\_Cpt :

Le **nom** de la catégorie doit être unique pour chaque catégorie.

- Compétences :

Le **nom de la compétence** doit être unique pour chaque compétence et en lettres majuscules, avec acceptation de certains caractères spéciaux. La catégorie d'une compétence doit exister dans la table Categorie\_Cpt.

- Centre Intérêt:

Le **nom du centre d’intérêt** doit être unique pour chaque centre d'intérêt et peut être constitué de n’importe quel caractère.

- Sites/Réseaux sociaux :

Le **numéro d'identification** doit être unique pour chaque site/réseau social. Le **numéro d’identification** du candidat doit être présent dans la table des candidats. Le **lien** du site ou du réseau social correspond à l’expression régulière d’une URL.

- Permis :

Le **nom du permis** doit être unique pour chaque permis et doit être présent dans la liste { AM,BSR,A,A1,A2,B,B1,B2,BE,BVA,C,C1,CE,C1E,D,D1,D2,DE,DE1}.

- Diplômes :

Le **numéro d'identification** doit être unique pour chaque diplôme. Le **nom du diplôme** est une chaîne de caractères alphanumériques et sa **description** peut être faite par l’utilisation de différents caractères.

- CV :

Le **numéro d'identification** doit être unique pour chaque Cv. Le **numéro d’identification** du candidat doit être présent dans la table des candidats. Le **nom** du CV et la **photo** (correspond au chemin)ne contiennent pas d’espace, la **description** du CV et le **poste recherché** sont constitués d’une chaîne de caractères alphanumérique. **Description**, **type de poste** et **disponibilité** peuvent être constitués de n’importe quel caractère. Le champ **admis** peut avoir soit la valeur ‘ACCEPTE’ ou ‘REFUSE’.

- Expériences :

Les numéros d'identification du candidat et de l'entreprise doivent exister dans les tables Candidats et Entreprises respectivement. Le type de contrat et le poste occupé doivent être en majuscule.

- Formations :

Le **numéro d'identification** doit être unique pour chaque formation. Le **numéro d’identification** de l’établissement scolaire doit être présent dans la table des établissements scolaires. Le **niveau** et la **spécialité** sont constitués de caractères alphanumériques.

Les clés étrangères correspondantes à l'association des numéros d'identification doivent être présentes dans les tables associées.

# 3. Exemple de contenu de la base de données (instance de BD)

Une instance de la base de données des CV est donnée ci-dessous. Les scripts SQL correspondants sont données dans les fichiers :

- de création des structures des tables de nom : Gx\_CreatDon\_CV.sql

- de création des données de nom : Gx\_InsertDon\_CV.sql

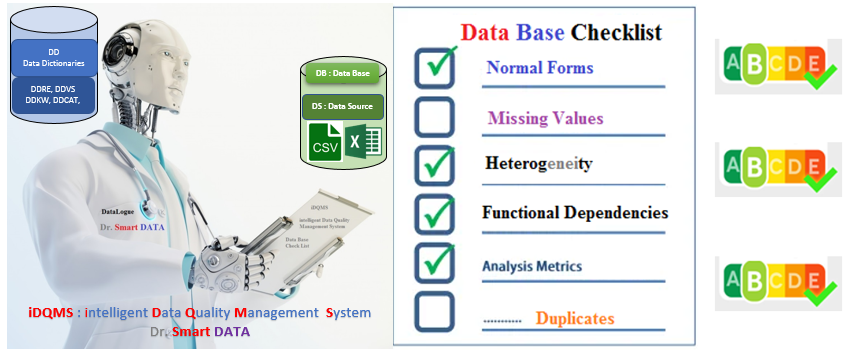
- de manipulation des données de nom : Gx\_ManipDon\_CV.sql

Un jeu de test est primordial pour valider notre travail. Nous donnons, dans ce qui suit, les détails de ce jeu de données.

Il va falloir définir très précisément des « règles » (des contraintes très précises) sur les données (Construire/Définir les métadonnées) afin d’assurer/améliorer la qualité des données

Exemple : Le nom d’un(e) client(e) doit être :

1. NON vide, alphabétique,
2. en MAJUSCULE,
3. sans espace superflu (devant et entre les mots),
4. les seuls caractères spéciaux autorisés sont – et les lettres françaises avec accent…



## ***3.1. Exemple d’instances de la BD des CV***

Une instance inspirée des CV de la promotion M2EID2 de l’année 2020-2021 est donnée ci-dessous.

Afin de ne pas trop surcharger l'affichage, nous avons décidé de vous montrer une instance avec les CV des quatre membres de ce groupe. Il est évident que nos systèmes gèrent tous les CV (voire bien plus) fournis.

**Instance en SQL :**

Table CV :

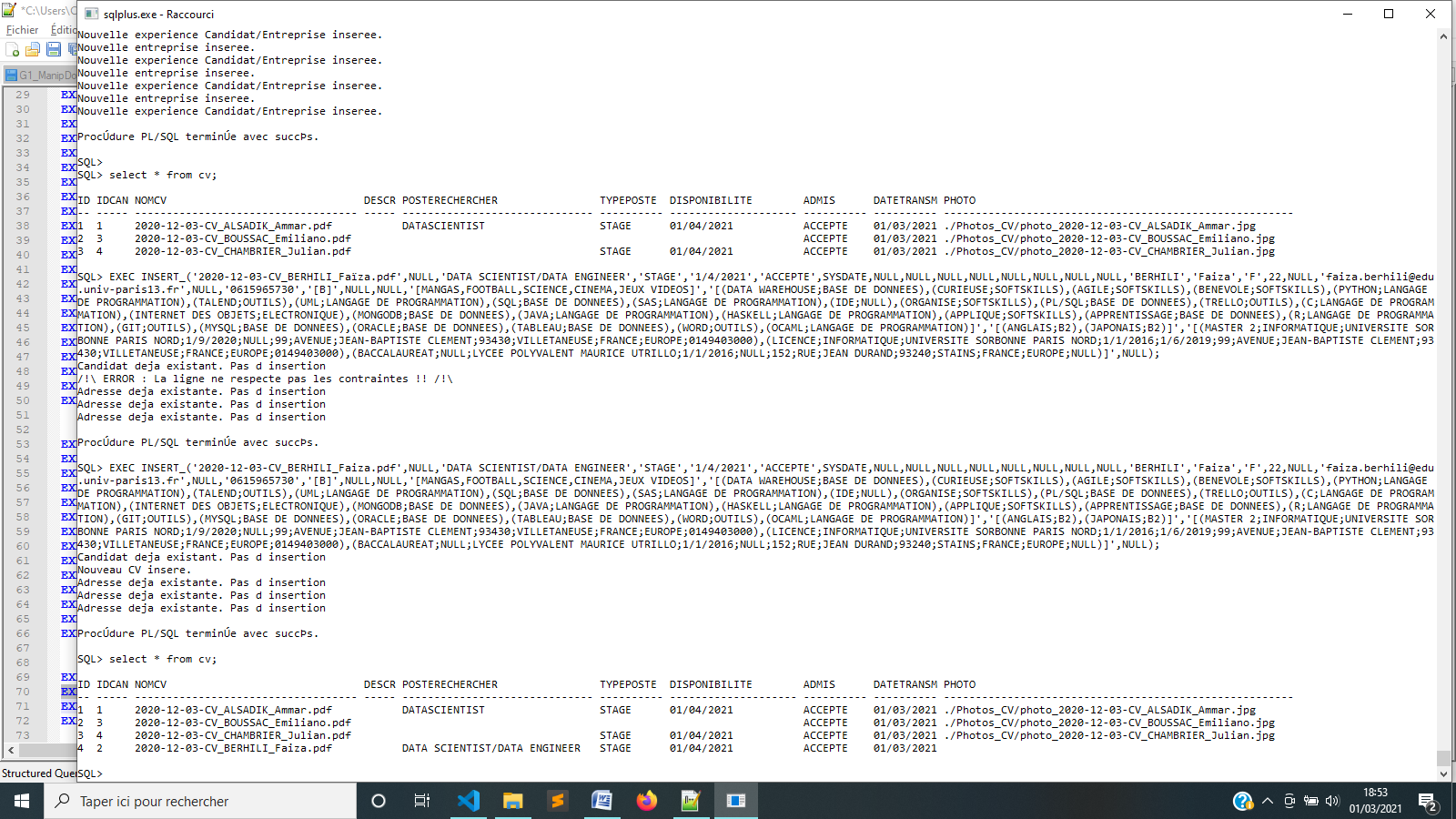


Table Adresses :

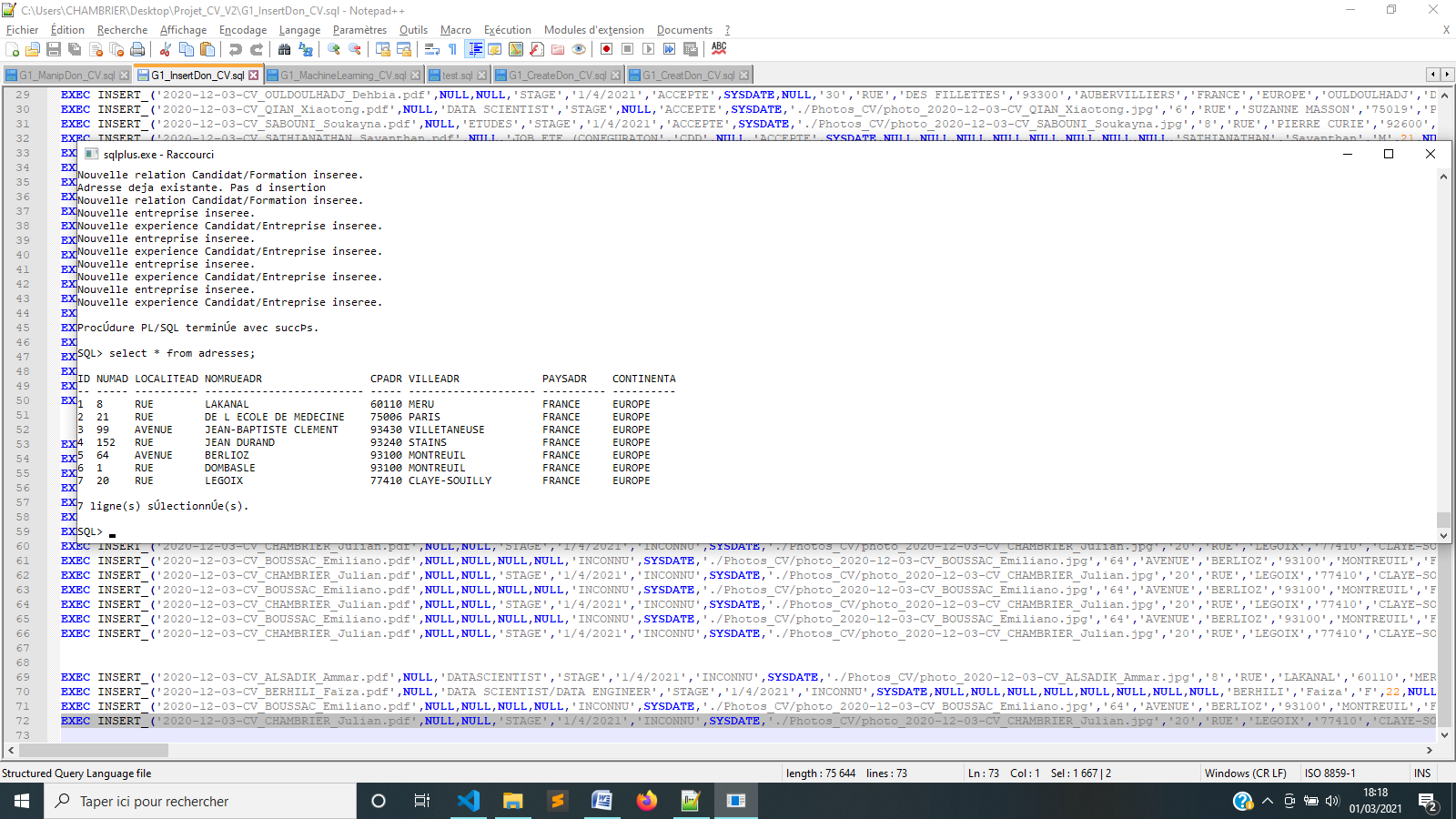
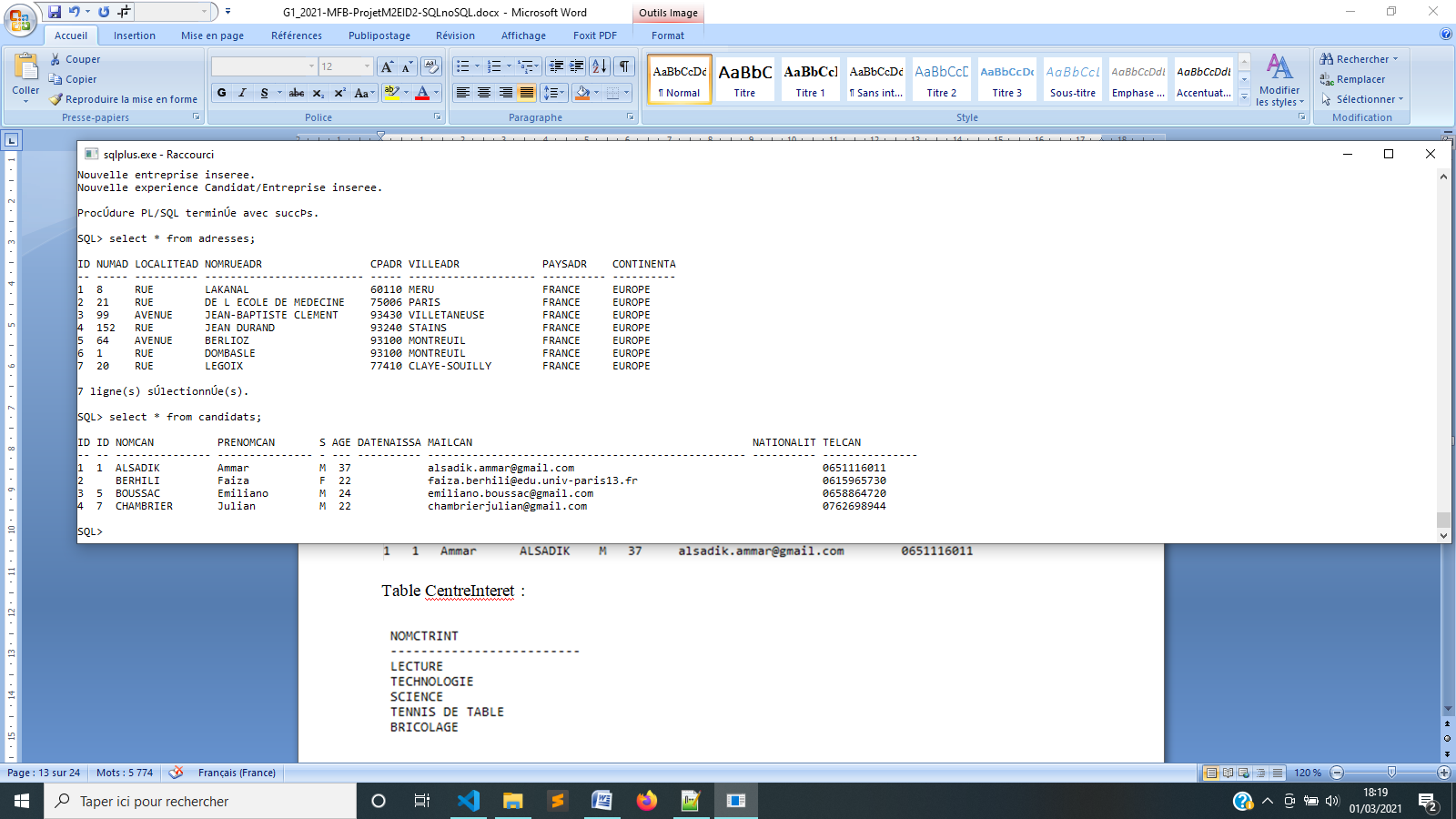
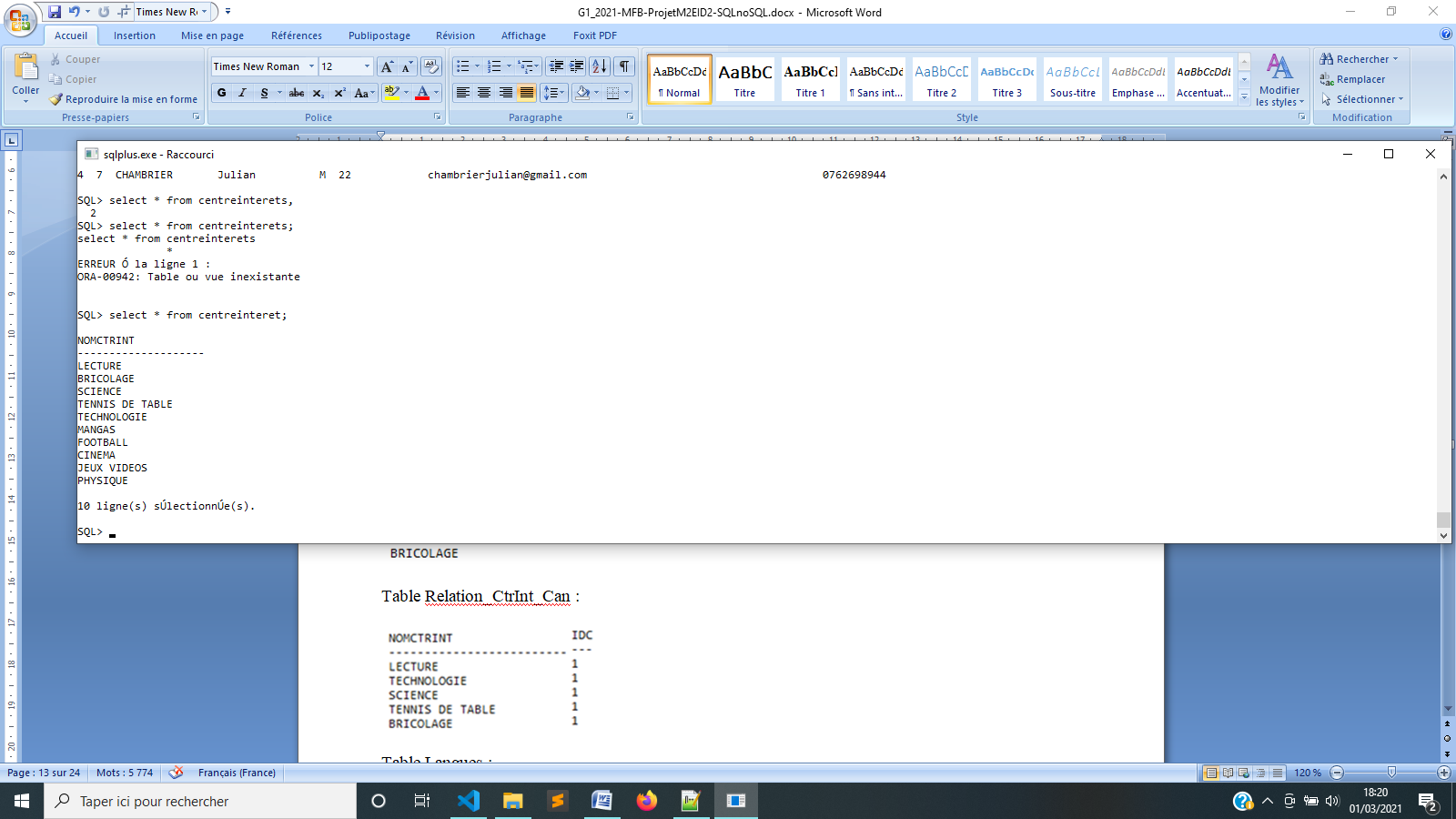
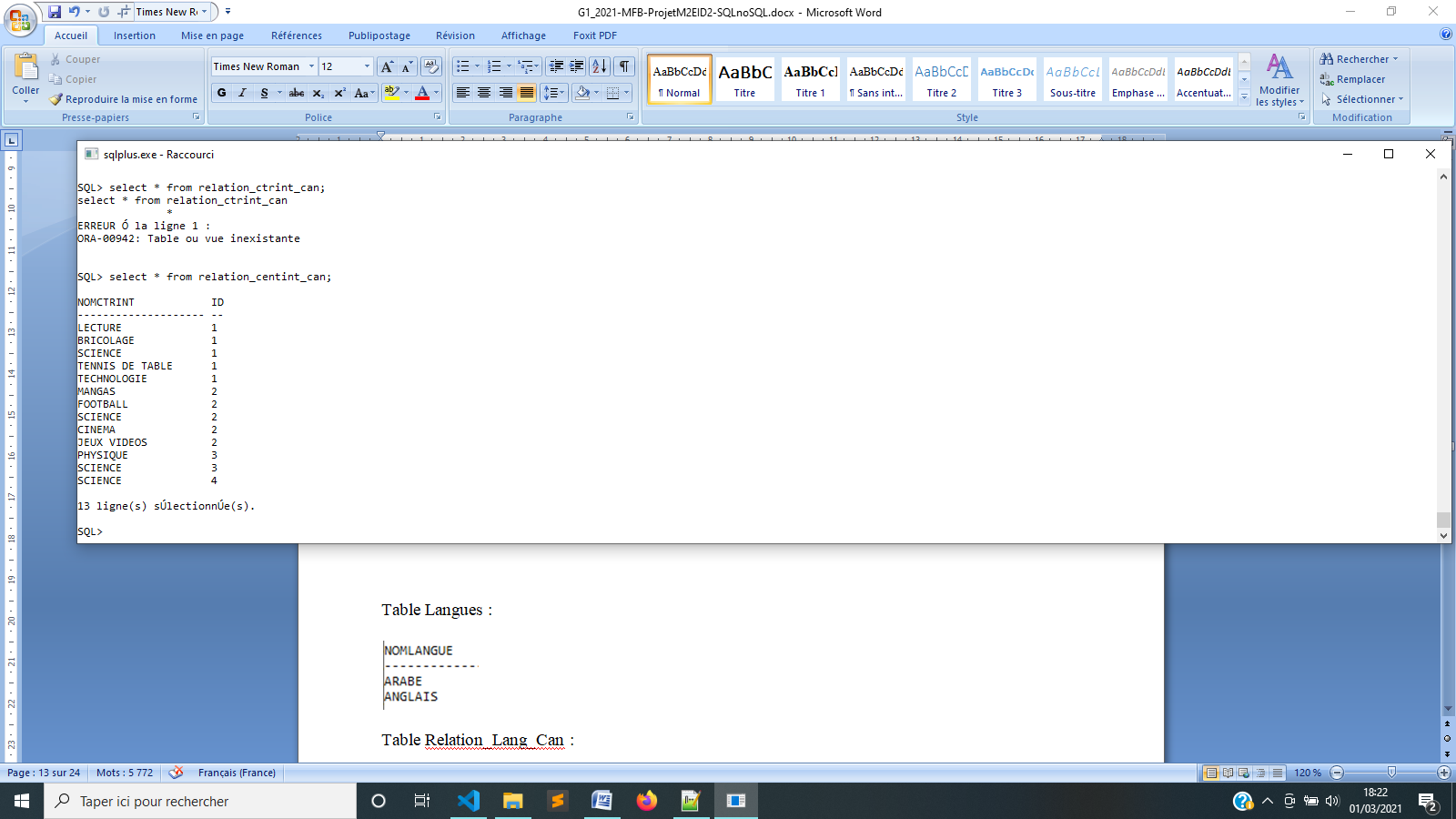


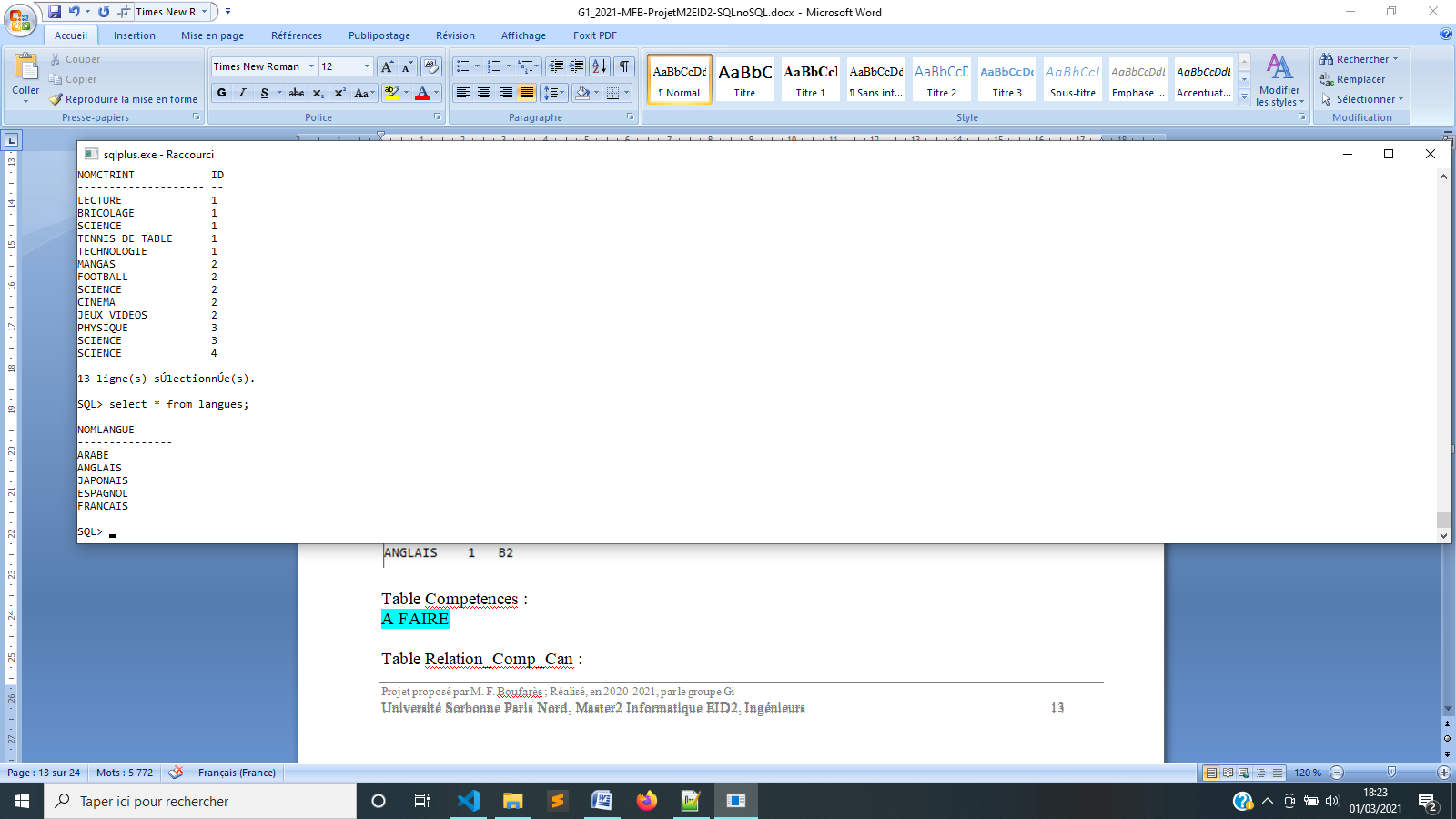
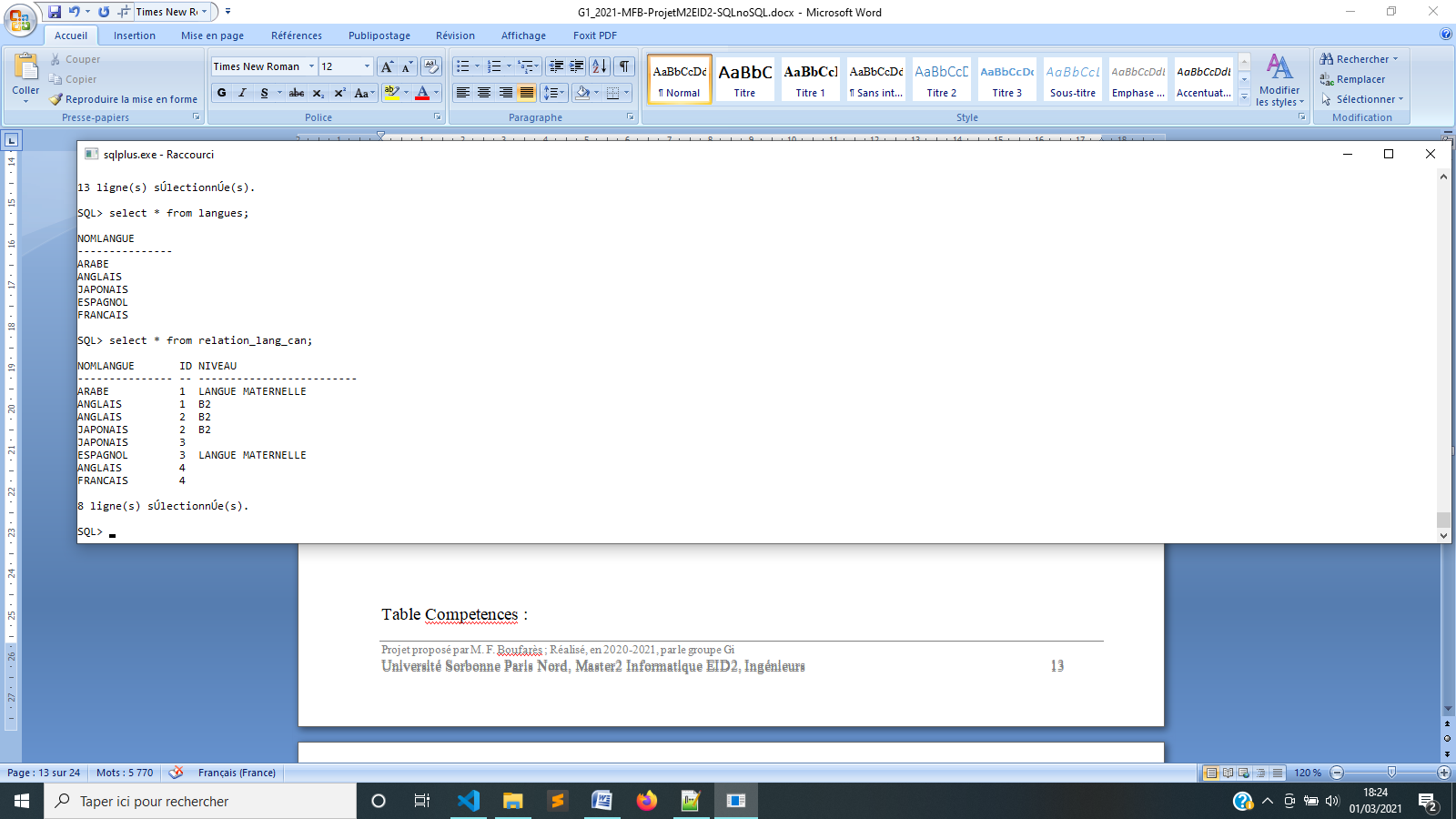
Table Candidats :



Tables CentreInteret et Relation\_CentInt\_Can:

Tables Langues et Relation\_Lang\_Can:

Tables Competences, Relation\_Comp\_Can et Categorie\_Cpt:

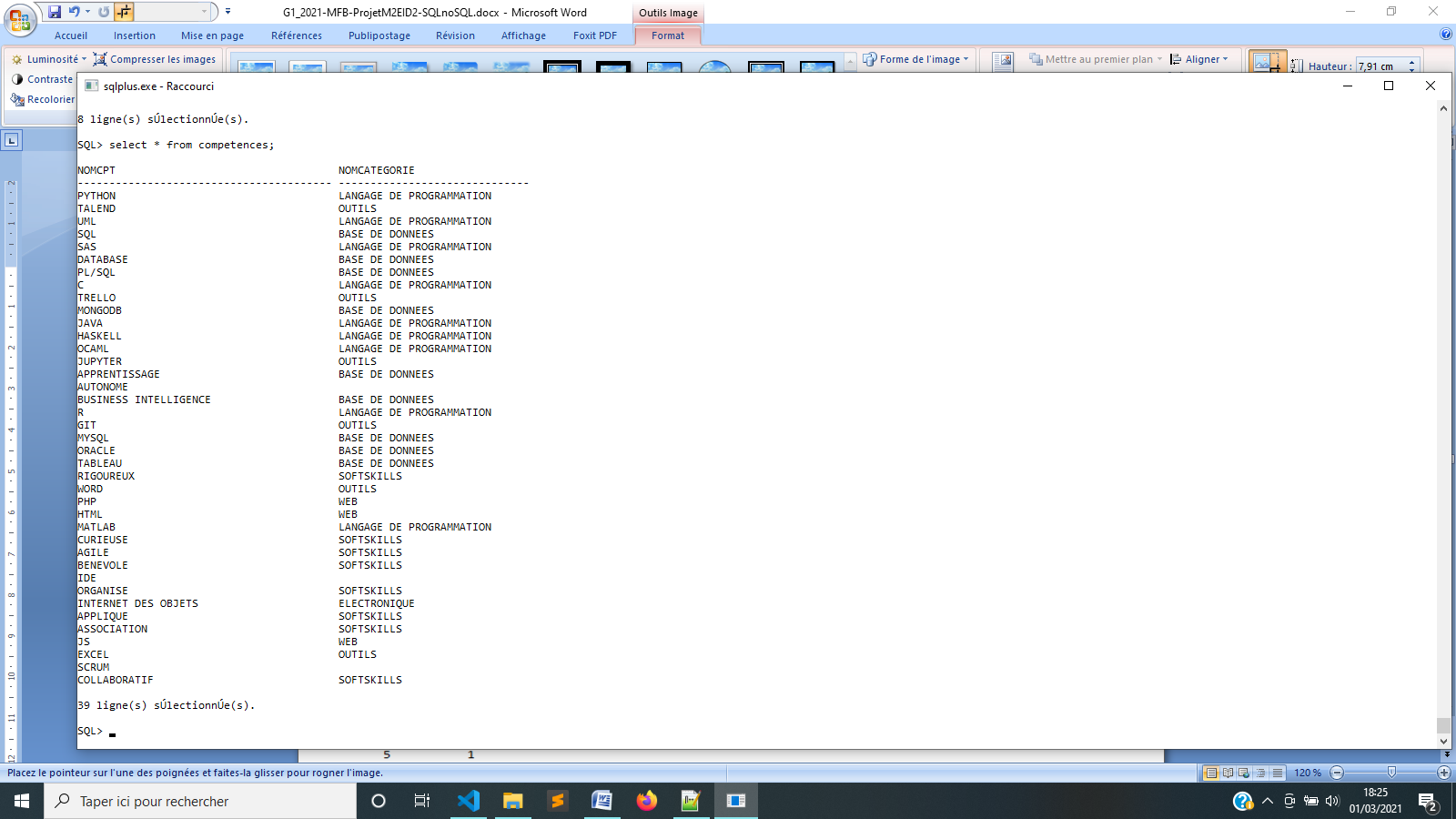
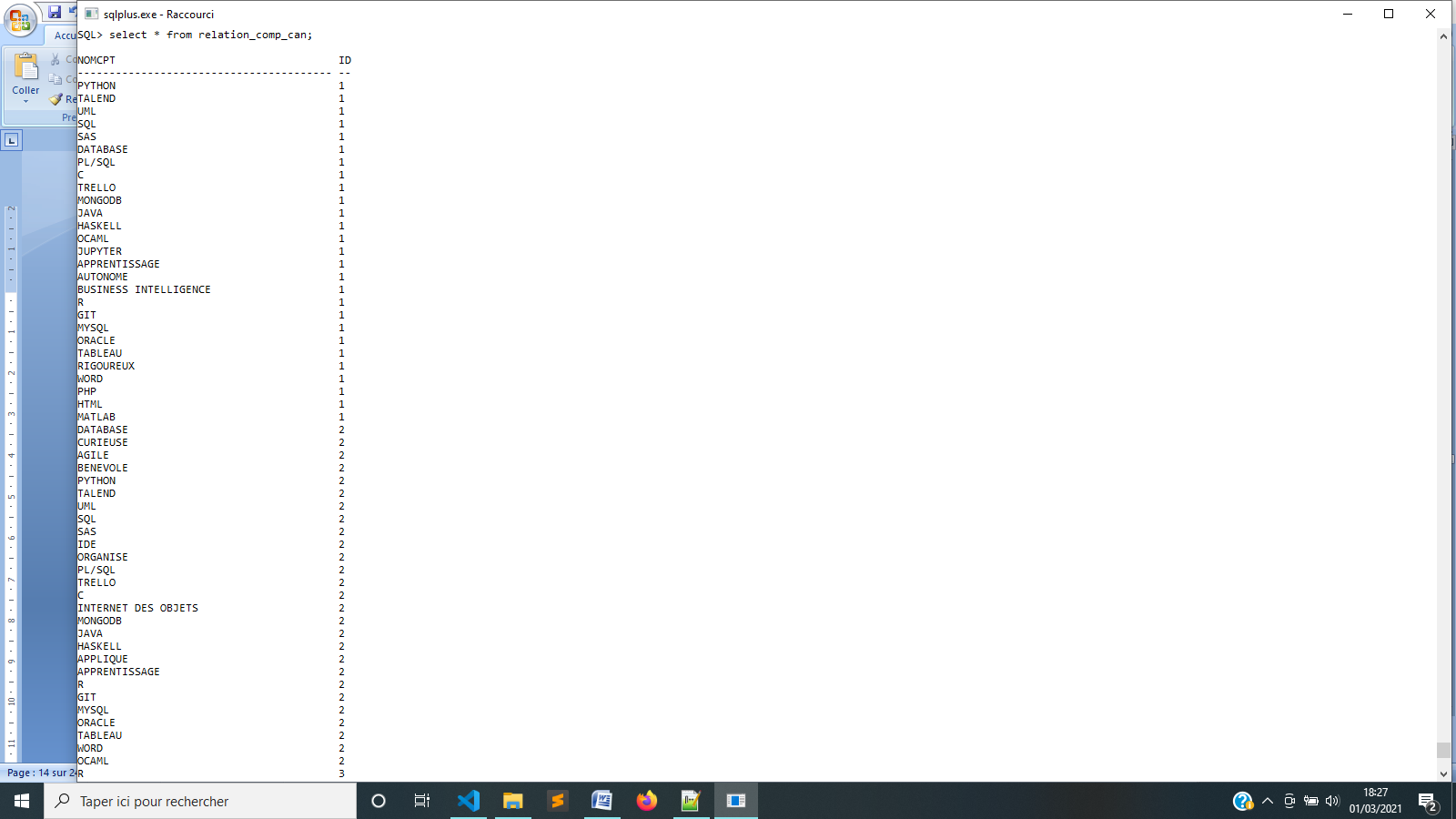
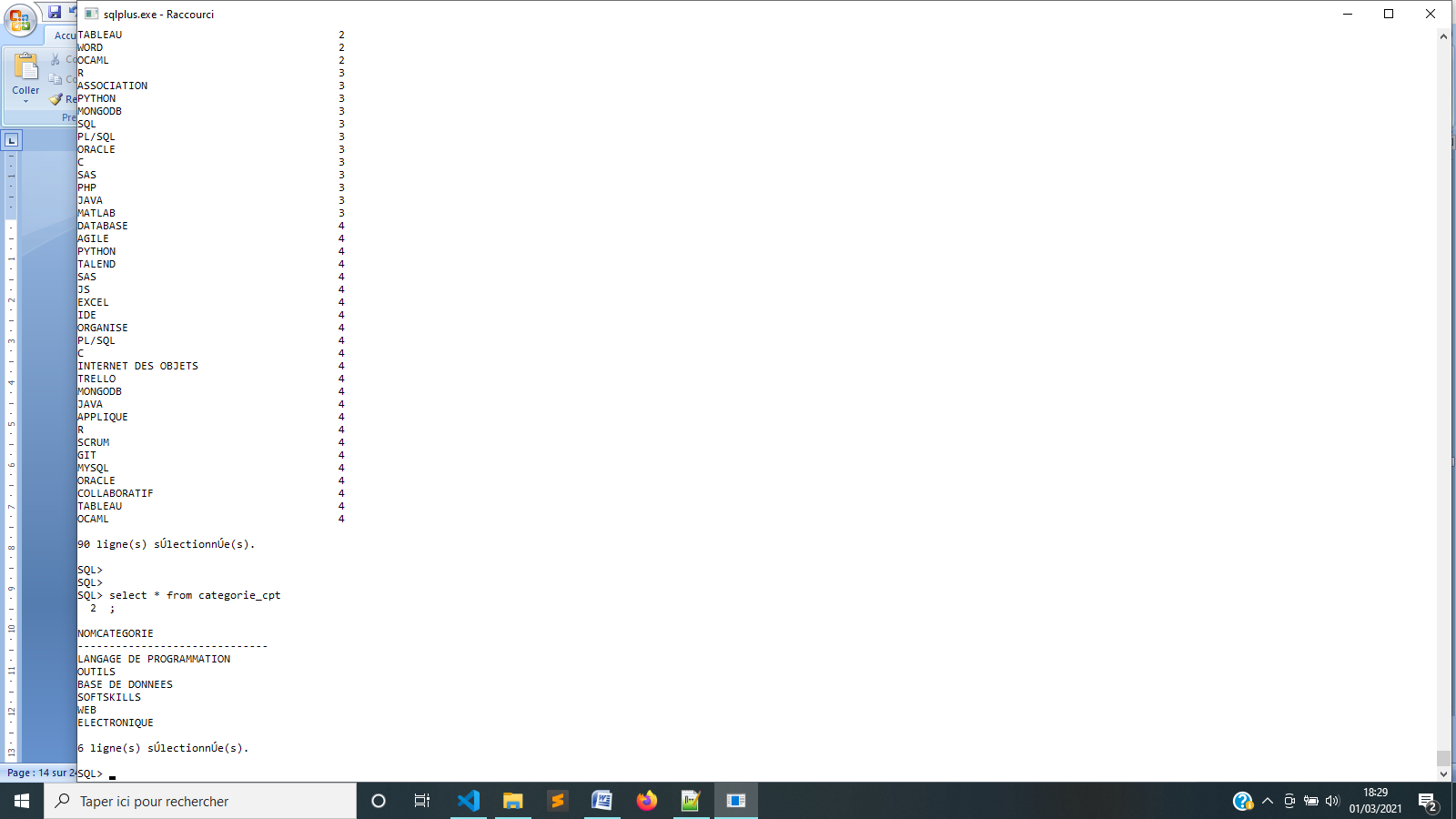
  

Table EtablissementScolaires :

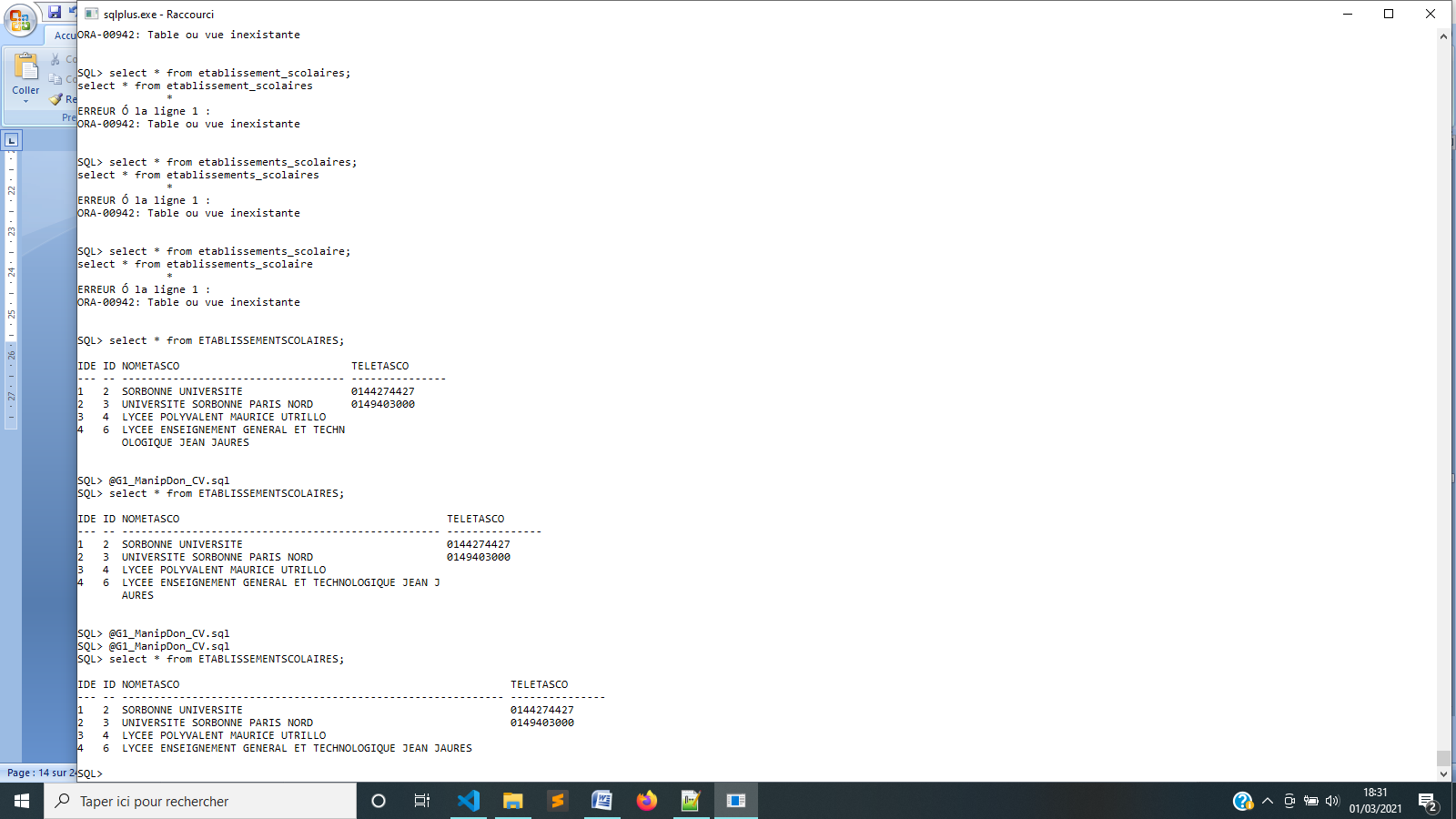


Table Entreprises :

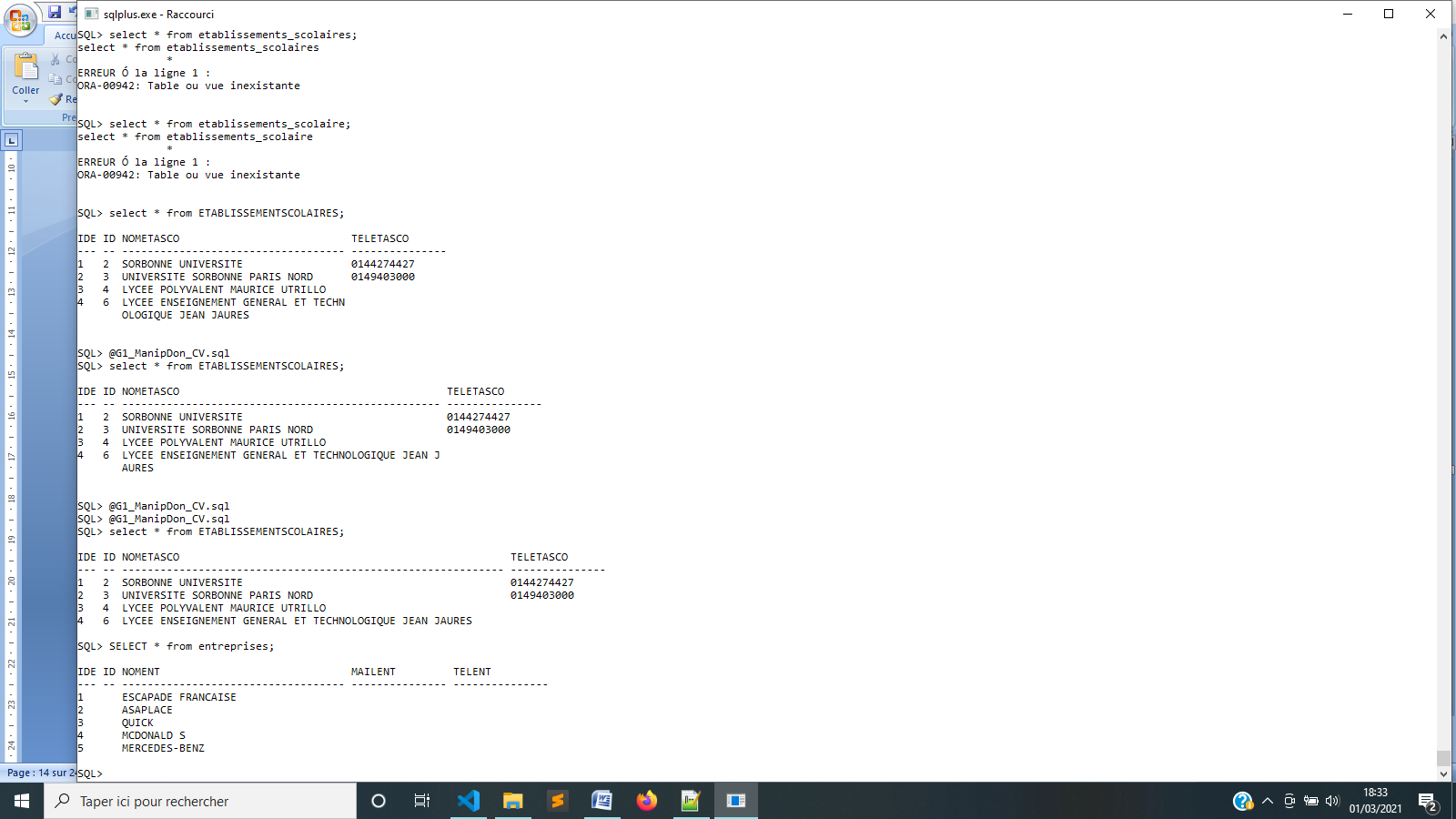
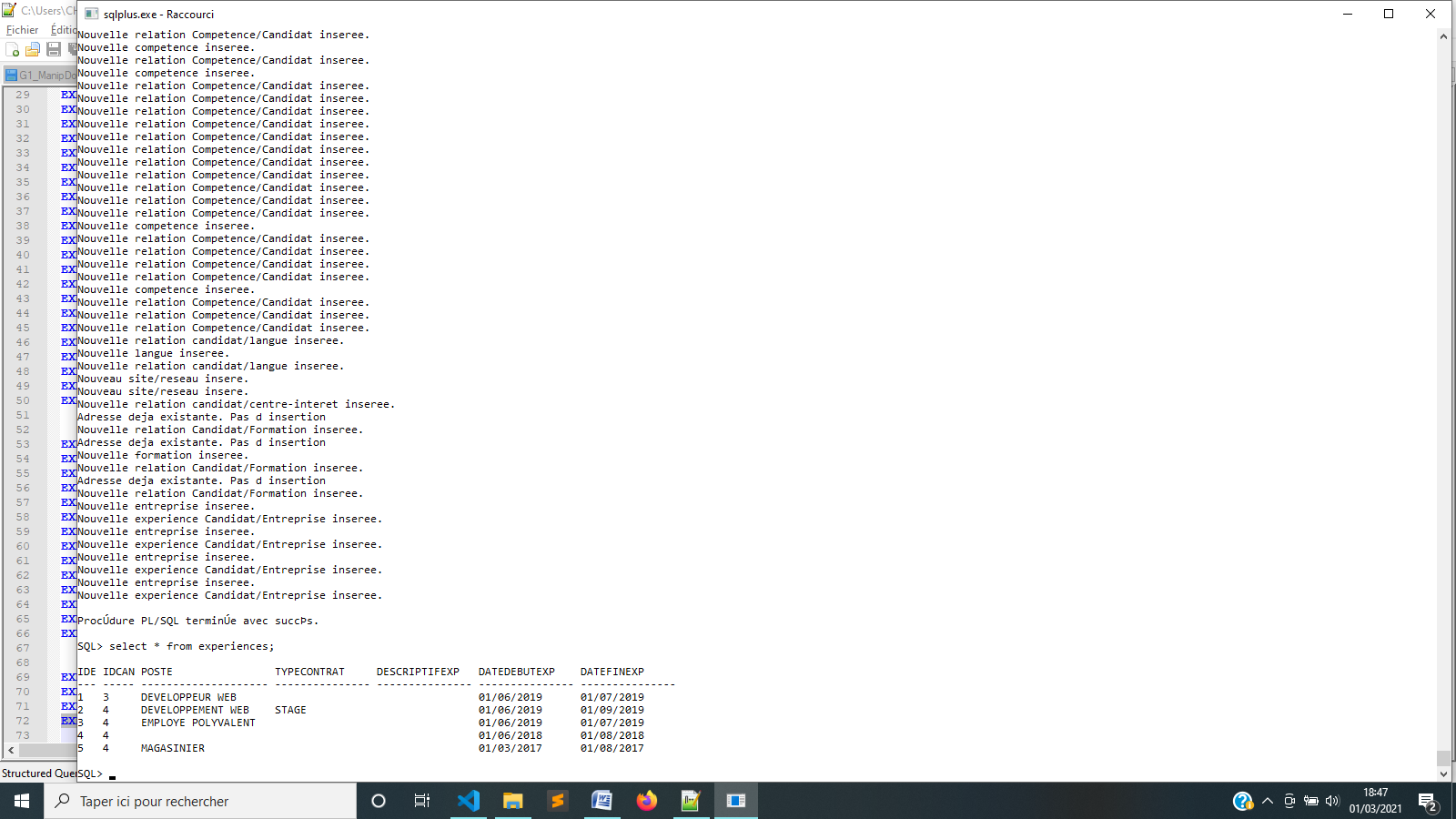
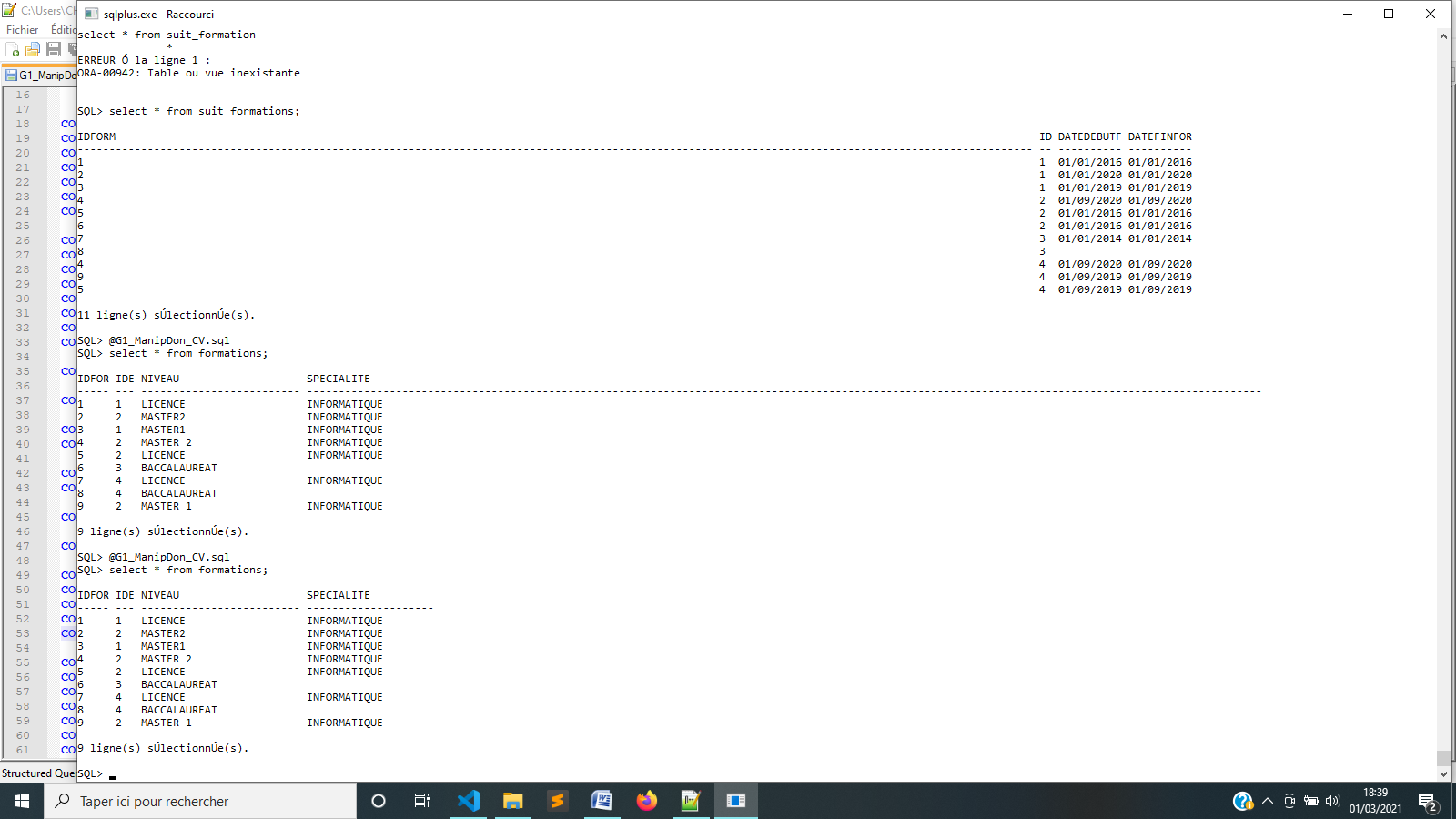
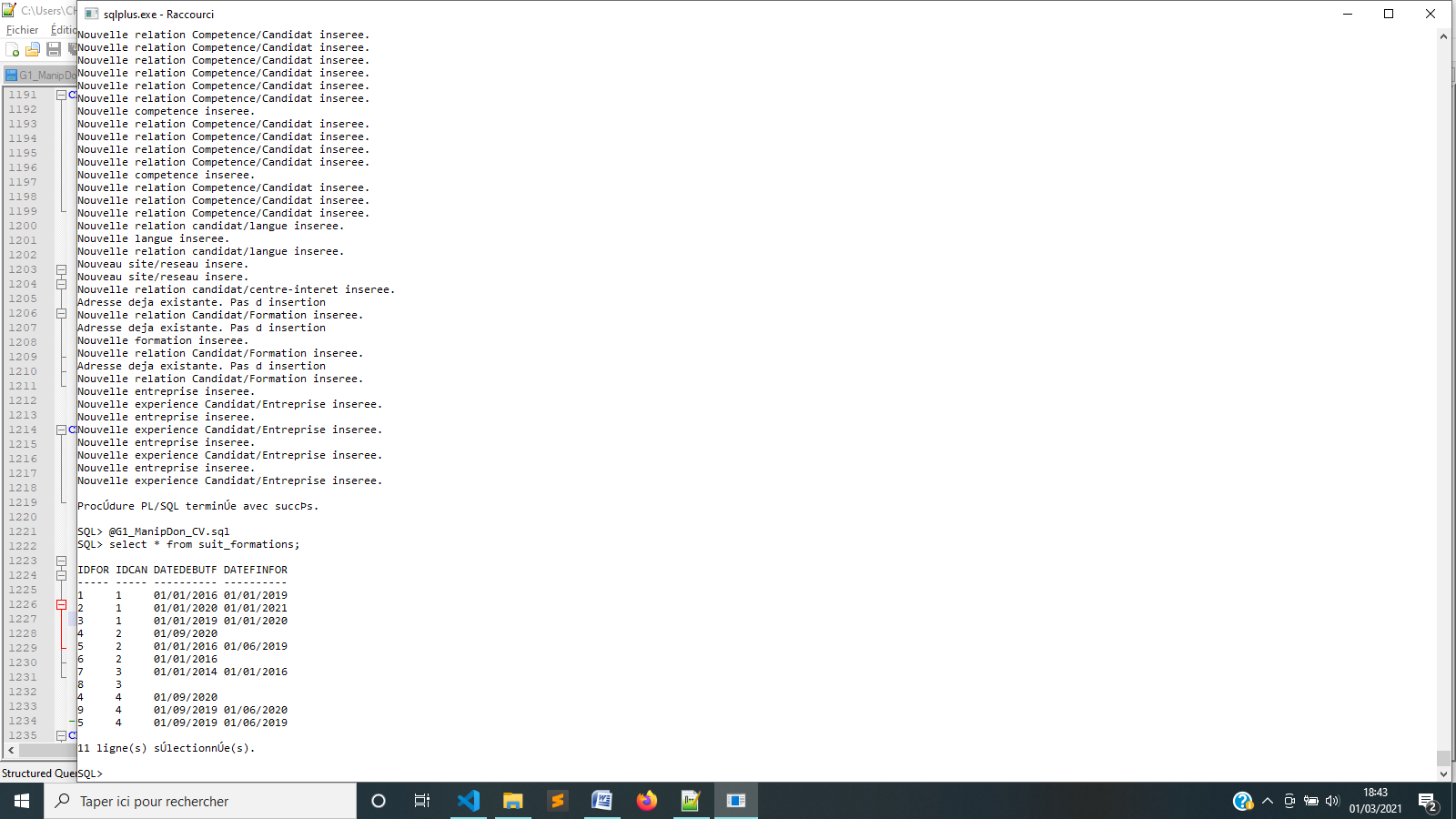


Table Expériences :



Tables Formations et Suit\_Formations :

Tables Permis et ObtentionPermis :

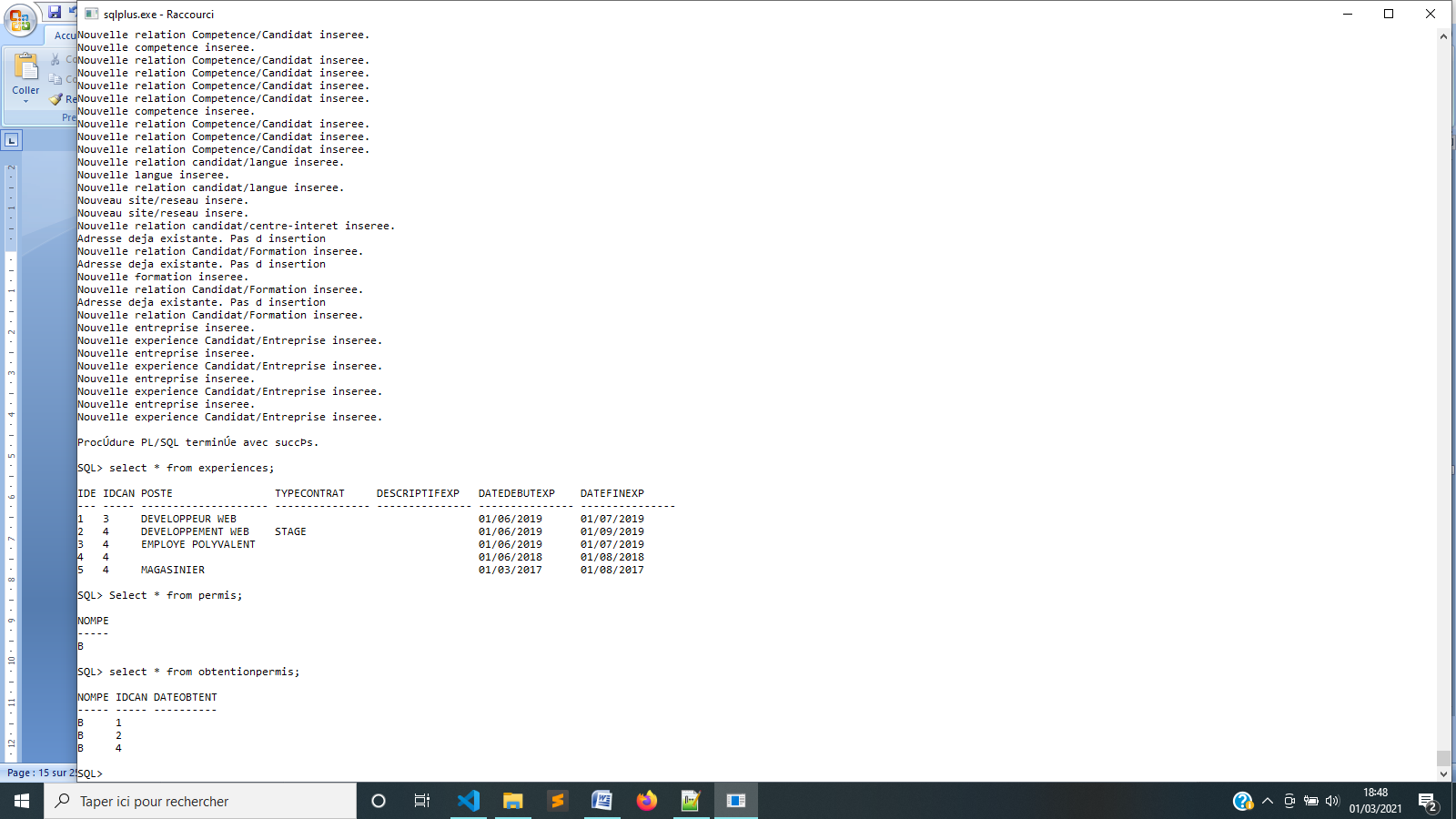
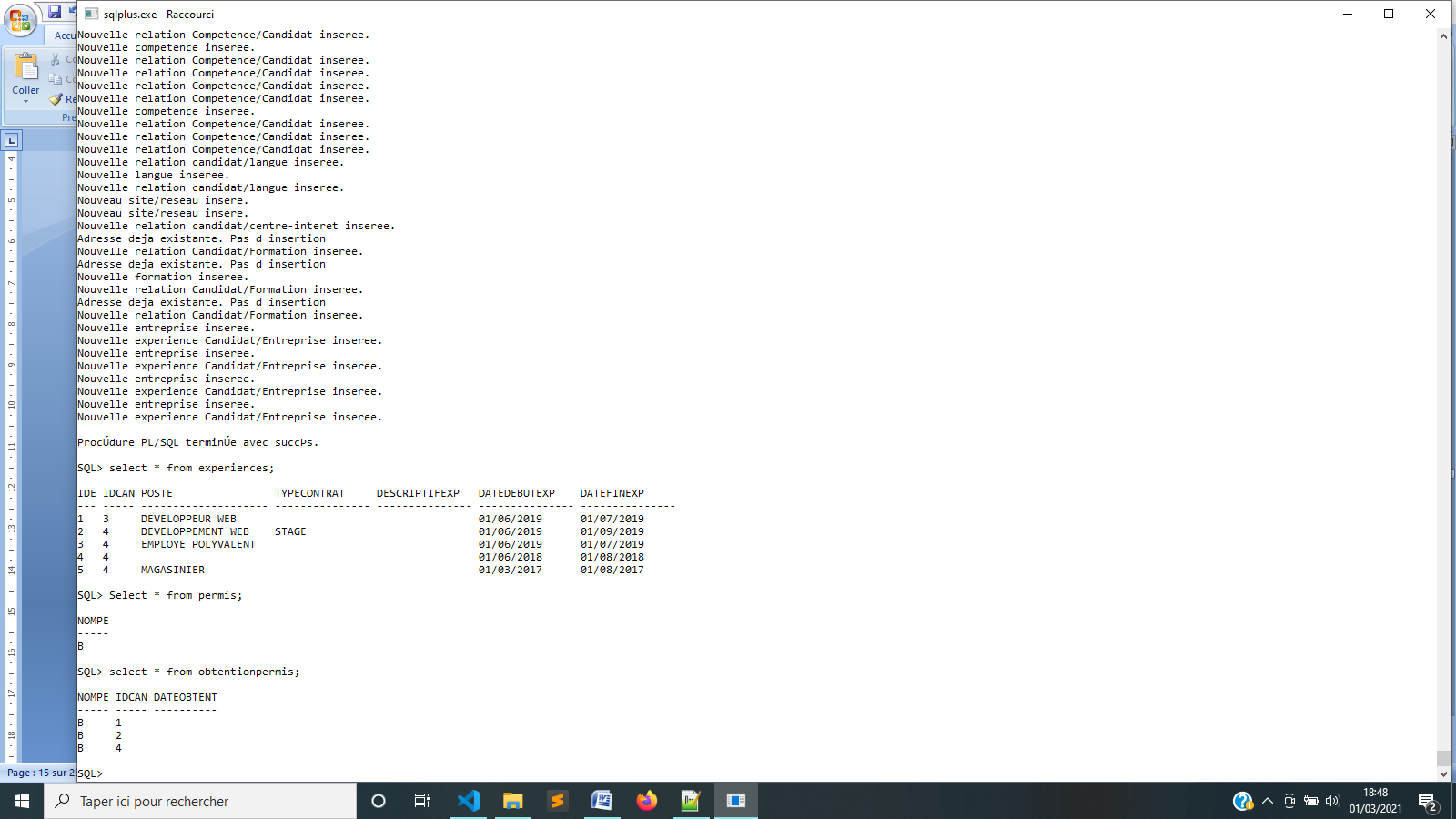
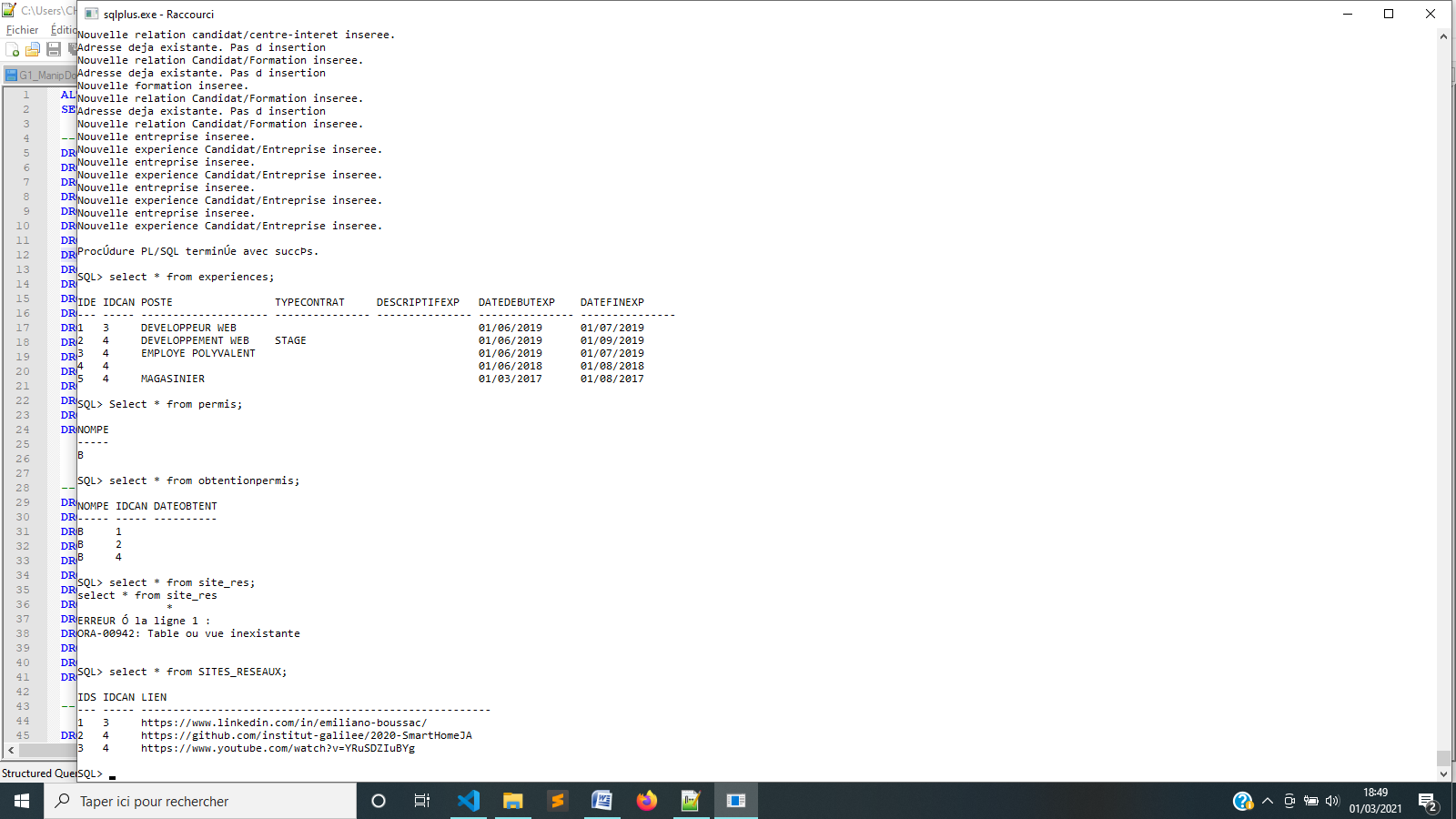
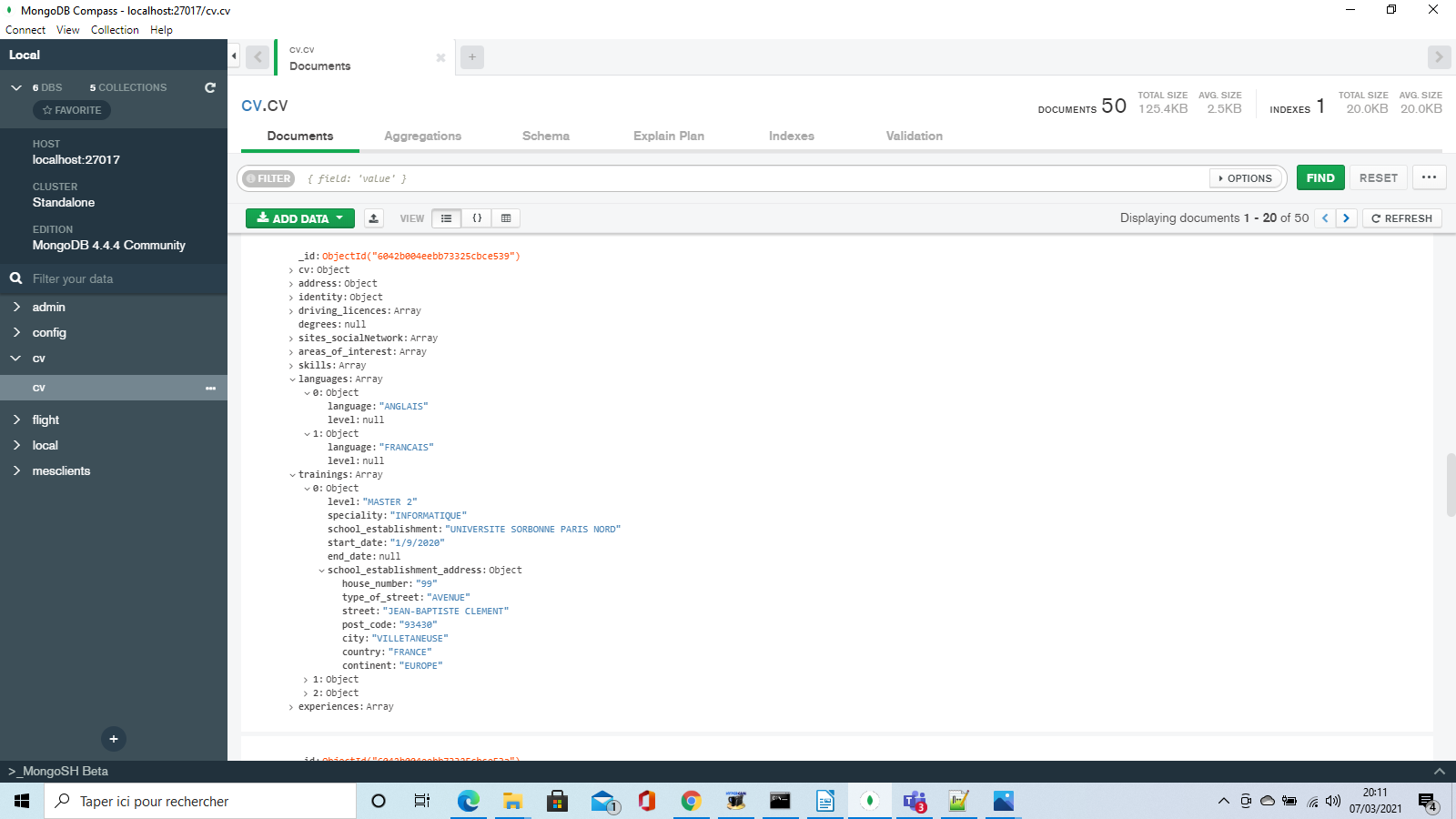
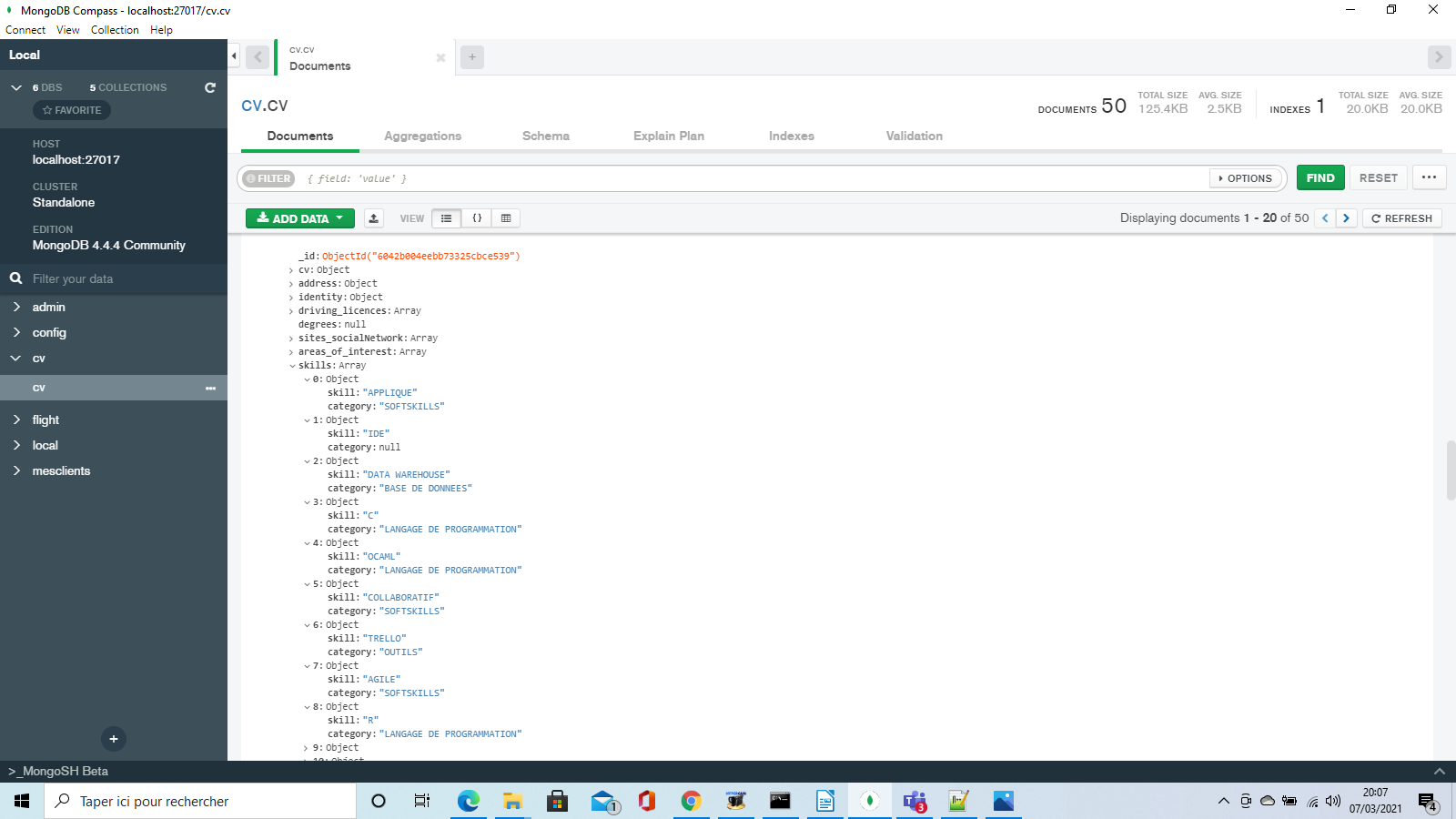
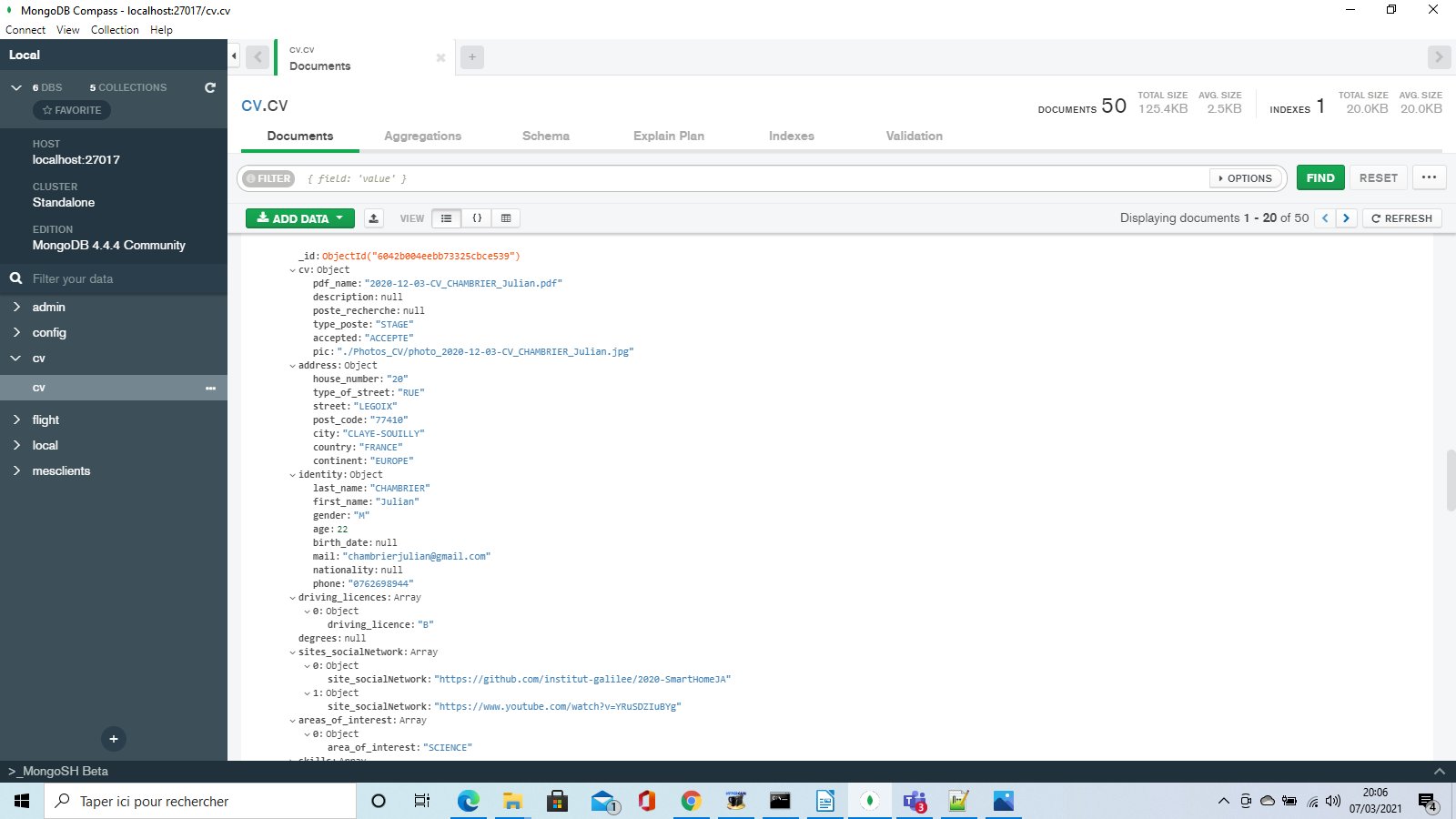
 

Table Sites\_Reseaux :



**Instance en MongoDB :**

Voici une instance des CV en MongoDB.



On observe que les deux instances ne sont pas représentées de la même manière. D'un côté (en SQL), nous avons des tables, les données sont structurées et de l'autre (en MongoDB), les données se trouvent dans une collection et on a directement accès à l'information.

Cela nous permet de travailler avec des données dans des outils hétérogènes. Ainsi, outre l'hétérogénéité des données, nous voyons aussi l’aspect de la diversification des outils de gestion de bases de données.

## ***3.2. Création des structures de données (SQL-ORACLE)***

**Pour SQL :**

La création des tables, en SQL2, est détaillée ci-dessous. Les tables sont créées dans le fichier G1\_CreatDon\_CV.sql.

Notre entrepôt de données est constitué de 22 tables et 14 méta-tables.

La création des tables se fait avec les commandes

CREATE TABLE NomTable(

Attr1 Type1,

Attr2 Type2,

…

CONSTRAINT PKNomTable PRIMARY KEY(Attr),

[CONSTRAINT FK\_NomTab1\_NomTab2\_Attr FOREIGN KEY(Attr) REFERENCES NomTable2(Attr),...]

);

De plus, la création des méta-tables qui permettent la vérification des contraintes sur les colonnes sont créées avec les commandes :

CREATE TABLE META\_NomTable

(

CONST\_Attr1 VARCHAR2(2000) DEFAULT NULL

[CONST\_Attr2 VARCHAR2(2000) DEFAULT NULL,...]

);

Les CONST\_Attr permettent de stocker les expressions régulières que doivent respecter les attributs.

Elles vont donc permettre :

- 1) On a facilement accès à la visualisation des contraintes sur les colonnes. En faisant simplement un 'SELECT \*' sur les méta-tables.

- 2) On peut facilement ajouter des contraintes sur un attribut quand bon nous semble, avec un simple INSERT INTO dans la méta-table avec l'expression régulière que l'on souhaite ajouter. Ainsi, si on souhaite faire évoluer notre table avec d'autres données, il est assez facile de faire ce procédé.

- 3) Enfin les méta-tables vont permettre des insertions plus sûres. En effet, avec une procédure on va pouvoir vérifier les contraintes lorsque l'on souhaite insérer une donnée. Toutes nos procédures d'insertions de nouvelles données vont vérifier que les contraintes des méta-tables sont vérifiées.

**- Pour MongoDB :**

MongoDB ne demande pas de création particulière de structure.

## ***3.2.1. Extraction automatique des données.***

L'extraction de données des CV au format PDF se fait avec des scripts Python. Cette extraction se fera avec plusieurs étapes.

**Pré-Requis** : Trois dossiers comportant les CV acceptés (CV\_ACCEPTE), les CV non acceptés (CV\_REFUSE) et les CV dont on ne connaît pas la classe (CV\_INCONNU) sous format PDF que l'on souhaite traiter. Idéalement, il est préférable que les fichiers PDF aient le nom : *2020-12-03-CV\_NOM\_Prenom.*pdf afin d’accélérer le traitement des informations. Vérifiez bien que les CV se trouvant dans CV\_INCONNU ne se trouvent pas dans CV\_ACCEPTE, ni dans CV\_REFUSE. Sinon ça serait faire du sur-apprentissage (pour la partie Machine Learning).

La seconde étape est de lancer le Script Python main.py. Pendant l'exécution de ce script on peut voir l'avancement de l'extraction des données grâce à l'affichage d'une barre de progression que l'on à implémenter sur le Terminal. En fonction de la puissance de calcul de votre machine, le traitement des CV peut prendre quelques minutes.

1) Le script commence par ouvrir le fichier PDF avec des fonctions que l’on a implémentées

2) Puis le script va convertir le fichier PDF en texte.

3) Il va **convertir le texte en FRANÇAIS** (ex : United Kingdom -> Royaume Unis, Francia -> France). Ainsi, nous n'avons plus de problème liés aux langues. On passe de **N'IMPORTE QUELLES LANGUES** au français.

4) Puis il réalise la détections et l'extraction des informations contenues dans le texte

La plupart des extractions se fait à l'aide d'expressions régulières sauf pour le nom et le prénom qui sont simplement des chaînes de caractères pouvant facilement se confondre avec n'importe quel mot.

Pour pallier le problème, nous avons implémentés une solution :

Elle consiste à trouver les mots qui se trouvent dans l'adresse mail et/ou dans le nom du fichier. En effet,  la plupart des gens ont leur prénom dans leur adresse mail ou dans le nom du fichier qu'il transmette.

De ce fait, l'extraction des données textuelles permet déjà de faire un premier tri, dans le respect des contraintes, avant même le passage par les procédures de validation.

Maintenant que nous avons extrait toutes les informations textuelles via des expressions ou des algorithmes de similarité, il reste une donnée à extraire : la photo.

Nous avons fait le choix de ne pas utiliser le dossier des photos, pour une raison simple : quand on pense plus grand dans l'extension de notre entrepôt de données. Il est très peu fréquent qu'une personne envoie sa photo en plus de son CV.

On a donc voulu extraire la photo directement à partir du CV, si elle existe.

Pour ce faire, on va convertir le fichier PDF en Image.

A partir de cette image, nous allons appliquer un algorithme de détection de visage, puis on va sauvegarder, dans un dossier Photo\_CV, la détection de cet algorithme. A la fin de l’exécution du script, on aura donc un dossier avec toutes les photos des CV comportant une photo.

Une fois toutes les informations du CV extraites, le script Python va générer automatiquement des requêtes SQL (pour ORACLE) et un fichier JSON (pour MongoDB) correspondantes aux différentes insertions dans les tables SQL et MongoDB et va les ajouter à la suite de nos fichiers SQL et MongoDB d'insertion G1\_InsertDon\_CV.sql et G1\_InsertDonMongoDB\_CV.json.

Là encore, sans aucune intervention de notre part, tout est automatique.

Ainsi, après l'exécution de ce script, on peut exécuter les fichiers SQL et MongoDB, soit en copiant/collant des commandes SQL ou MongoDB soit en exécutant directement le script. Après l’exécution de ces requêtes, on aura les tables remplies. On aura construit notre entrepôt de données.

## ***3.2.3. Insertion des données***

Les données précédemment extraites sont insérées en SQL et en MongoDB. Comme dit précédemment, elles se situent dans le fichier G1\_InsertDon\_CV.sql, pour SQL, et G1\_InsertDonMongoDB\_CV.json, pour MongoDB.

**En SQL :**

Pour chaque CV, une seule requête d’insertion est créée. En comparaison avec la première phase du projet, nous avions une trentaine de requêtes (candidat, compétences, centres d'intérêt, adresse, …) par CV. Cela a pour avantage de rendre plus lisible le fichier d’insertions. En effet, dans la première phase, nous avions près de 1500 lignes pour 50 CV, il était donc facile de s’y perdre. Maintenant, pour 50 CV nous avons 50 requêtes. Ces lignes sont certes plus longues mais restent faciles à lire et à comprendre.

La requête d’insertion pour un CV est constituée de toutes les informations extraites avec le script Python et cette requête se fonde sur une procédure appelée INSERT\_. Cette procédure prend beaucoup de paramètres représentant les informations sur le candidat, ses compétences, ses centres d’intérêts, ses formations et expériences, … Afin de facilement la comprendre, cette grande procédure fait appel à plusieurs autres procédures, chacune ayant un rôle défini comme l’insertion du candidat.

Notons bien que les identifiants uniques de tous les éléments sont gérés par SQL, on a dû mettre en place des séquences et des renvois d’identifiant par les procédures afin qu’il puisse être réutilisé par d’autres procédures qui en ont besoin. Par exemple, quand on insère un candidat, il est important de stocker son identifiant pour insérer toutes les relations (compétences, centres d’intérêt, langues, formations,...) qu’il existe avec ce candidat. Il en va de même pour les adresses, par exemple.

Dans la première partie de notre projet, Python se chargeait de générer des identifiants uniques. Mais cela avait plusieurs inconvénients. Par exemple, cela imposait de stocker un fichier d'identifiants quelque part auquel personne ne devait toucher, sous peine de dérégler les identifiants. Maintenant, nous n'avons plus aucun problème, tout est géré avec SQL.

Afin de mieux comprendre une insertion, nous allons faire un exemple. Bien entendu, il est très facile de comprendre les procédures et leur fonctionnement en se référant au fichier G1\_CreateDon\_CV.sql où elles sont implémentées.

Imaginons qu’on veuille insérer ces informations via la procédure INSERT\_:

Jean DUPOND, 20, RUE, NULPART, 77000, NOWHERE, FRANCE, EUROPE, [(SQL;BASE DE DONNEES)].

La première étape est l’insertion de l’adresse du candidat via la procédure INSERT\_ADRESSES. Cette procédure va vérifier si les données à insérer respectent les contraintes des méta-tables et si tout s’est bien passé elle va insérer l’adresse et renvoyer l'identifiant unique généré au cours de cette insertion.

Ensuite, c’est au tour du candidat via la procédure INSERT\_CANDIDATS. Mais cette procédure à besoin de l'identifiant de l’adresse. Heureusement, il a été renvoyé par l’appel à la précédente procédure (la procédure INSERT\_ADRESSE renvoie NULL s’il n’y a pas d’adresse ou si l’adresse s’est mal insérée). Ensuite, comme précédemment, la procédure vérifie les contraintes liées aux données avant d’insérer. Puis, elle renvoie l'identifiant unique du candidat généré au cours de cette insertion.

Enfin, dans notre exemple, on insère en dernier la compétence via la procédure INSERT\_COMPETENCES. Ici, on a besoin de l’identifiant du candidat pour insérer la relation entre le candidat et la compétence. Comme précédemment, il est renvoyé par la précédente procédure et l’insertion peut se faire facilement après vérification des contraintes.

Ici, l’exemple est assez simple et réduit, mais nous faisons ce principe pour toutes les données à insérer. On voit bien qu’il y a un jeu de transfert d'identification afin d’assurer la bonne relation entre les données.

Notons que les contraintes vérifiées sont stockées dans les méta-tables et que les données similaires sont aussi vérifiées. Si une donnée similaire a été détectée, la procédure n’insère pas la nouvelle donnée, complète éventuellement celle existante (explication dans la section suivante) et renvoie l’identifiant unique de la donnée existante. Ainsi, par exemple, si un candidat existe déjà, la procédure renverra son identifiant et toutes les données à insérer ensuite comme les compétences seront associées à cet identifiant.

**En MongoDB :**

MongoDB n’impose la mise en place de procédés pour la vérification des insertions. Il est non structuré.

Les données seront insérées telles qu’elles sont extraites et formatées par Python (déjà de très bonne qualité) et les éventuelles erreurs (similaires, ….) seront traités lors de l’intégration des données en Talend.

## ***3.2.4. Qualité des données.***

Dans un objectif de qualité des données, nous avons mis en place plusieurs procédés afin d'assurer un bon Data Cleaning.

La qualité des données est assurée de l'extraction de la donnée jusqu'à son insertion dans les tables.

Comme dit précédemment, l'extraction faite en Python est la première étape de vérification de la donnée. En effet, la plupart des extractions des données se font via des expressions régulières. Ainsi, on extrait la donnée seulement si elle vérifie une certaine forme et donc certaines contraintes.

La seconde étape, toujours en Python, est la standardisation et l'homogénéisation. Ces étapes sont réalisées dans les différentes fonctions d'extractions et de création des requêtes après l'extraction de la donnée. On va ainsi pouvoir mettre en majuscule, minuscule, corriger les caractères de ponctuations, les espaces, homogénéiser les numéros de téléphone… avant de créer les requêtes d'insertions SQL et MongoDB.

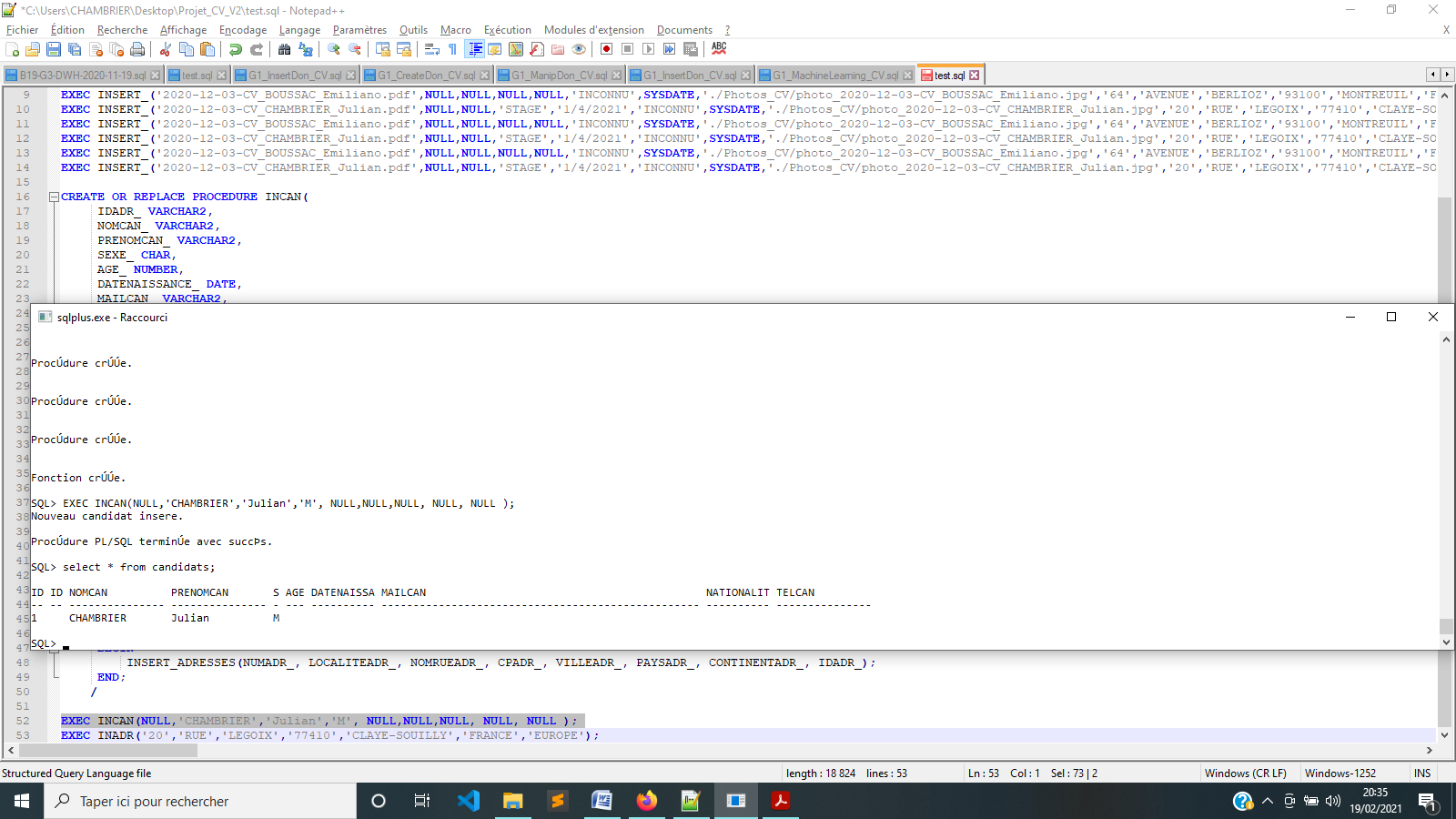
A ce stade, la donnée a déjà le format attendu.

La troisième étape est faite en SQL. En effet, nous avons développé des procédures d'insertions des données qui fonctionnent avec des appels à des tables où se trouvent des métadonnées. Ainsi, à chaque insertion d'une donnée, les procédures vérifieront que la donnée vérifie bien la métadonnée (une expression régulière).

Enfin, la dernière étape, au niveau SQL, est d'éviter d'insérer des données similaires (ou doublons). On ne veut pas qu'un candidat, une compétence ou encore un centre d'intérêt ne soit inséré plusieurs fois dans nos tables simplement car quelques caractères fluctuent.

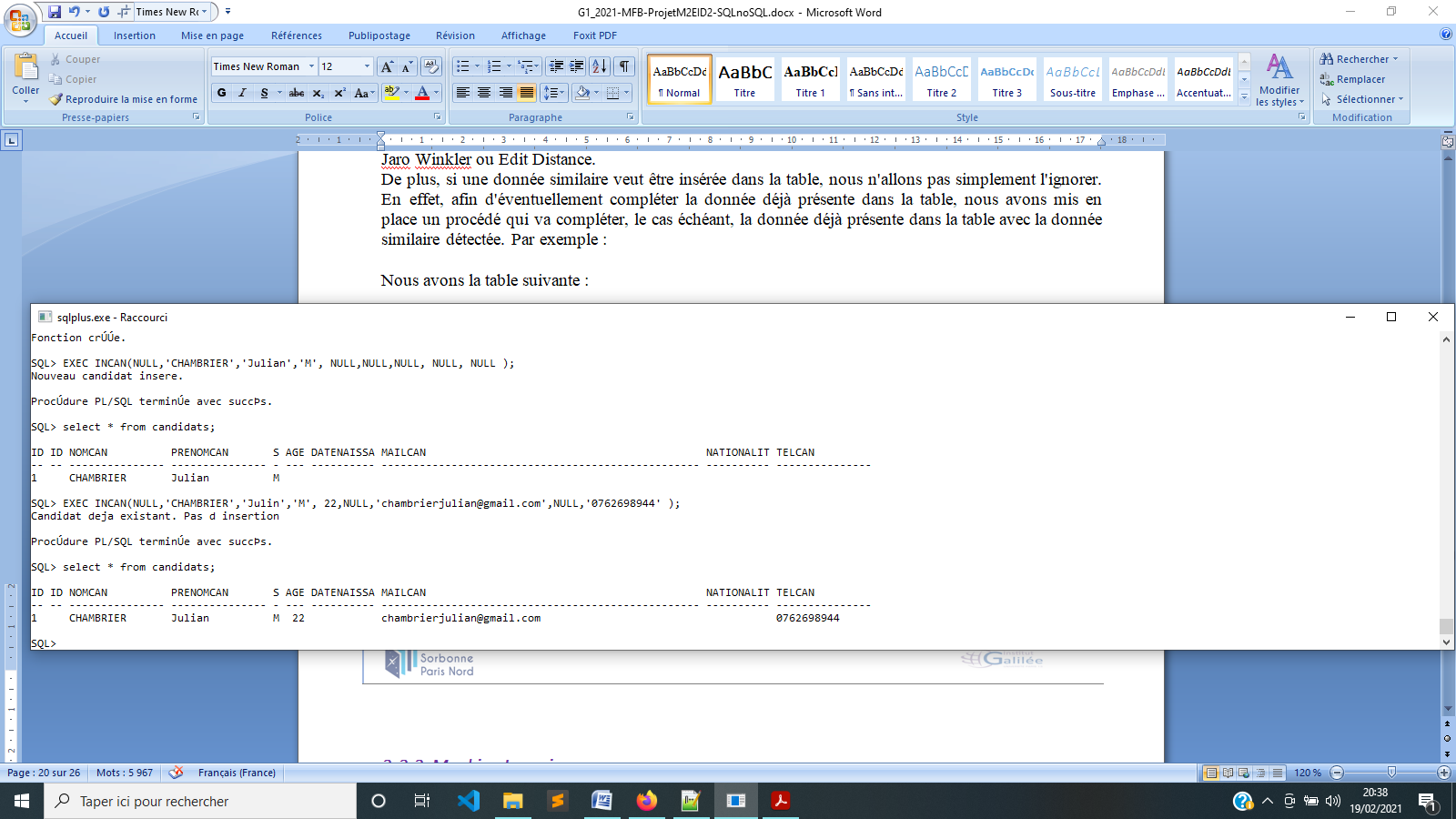
Pour pallier le problème, nous avons développé, dans nos procédures d'insertion, des procédés qui vérifient qu'une donnée n'a pas de similaire dans les tables avant de l'insérer. On se base sur des algorithmes vus durant le premier semestre notamment Jaro Winkler ou Edit Distance.

De plus, si une donnée similaire veut être insérée dans la table, nous n'allons pas simplement l'ignorer. En effet, afin d'éventuellement compléter la donnée déjà présente dans la table, nous avons mis en place un procédé qui va compléter, le cas échéant, la donnée déjà présente dans la table avec la donnée similaire détectée. Par exemple, nous avons la table suivante :



Et nous souhaitons insérer la donnée Julian, CHAMBRIE, M, 22 ans, chambrierjulian@gmail.com, 0762698944.

Elle sera considérée comme similaire malgré la faute dans le nom de famille et son insertion donnera le résultat suivant :



On observe que la donnée similaire n'a pas été insérée mais elle a permis de compléter la donnée déjà présente dans la table. Cela permet de nous inscrire dans une politique d'avoir des données le plus complète possible.

Enfin, la dernière étape de correction des données se situe lors de l’intégration des données de gestionnaire de bases de données effectué avec Talend. En effet, cette section sera détaillée plus bas, mais Talend corrige les données afin que la fusion entre Oracle et MongoDB n’apporte pas d’erreur, ni de redondance.

Pour résumer, nous avons mis en place différents procédés permettant d'avoir des données de bonnes qualités. Cela nous permet d'avoir des données facilement exploitables dans la réalisation de nos différentes applications.

## ***3.2.5. Machine Learning.***

Nous avons ajouté, à ce projet, un apprentissage automatique sur les données extraites. Cette partie va permettre de prédire et de conseiller l'utilisateur pour la classe (ACCEPTE ou REFUSE) des CV se trouvant dans le dossier CV\_INCONNU.

Pour ce faire :

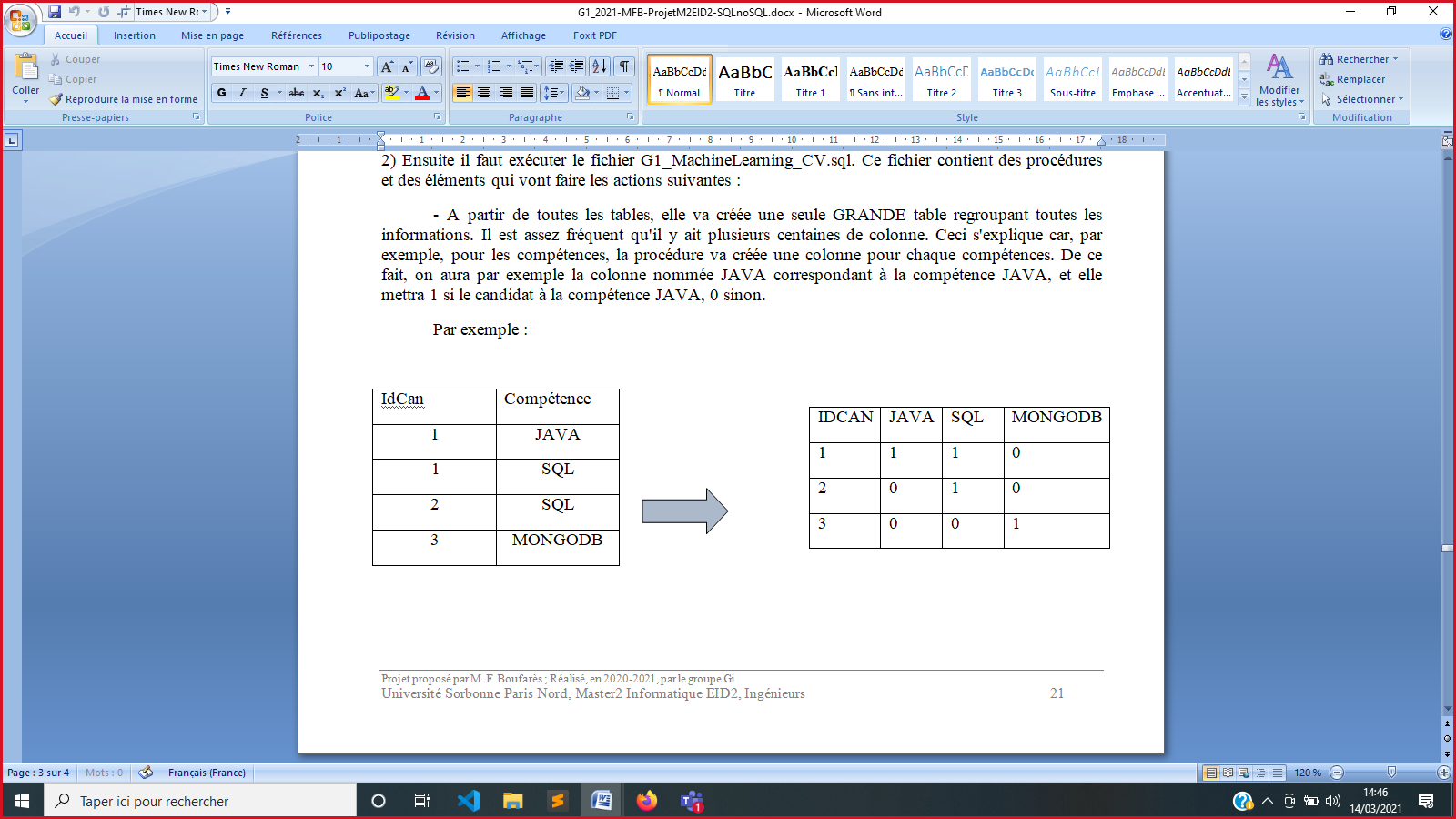
1) Lorsque le fichier d'insertions SQL (G1\_InsertDon\_CV.sql) a bien été exécuté, nous avons un entrepôt de données avec une multitude de données dans les tables.

Remarque : Les CV situés dans le fichier CV\_INCONNU sont aussi traités afin d'extraire leurs informations.

2) Ensuite il faut exécuter le fichier G1\_MachineLearning\_CV.sql. Ce fichier contient des procédures et des éléments qui vont faire les actions suivantes :

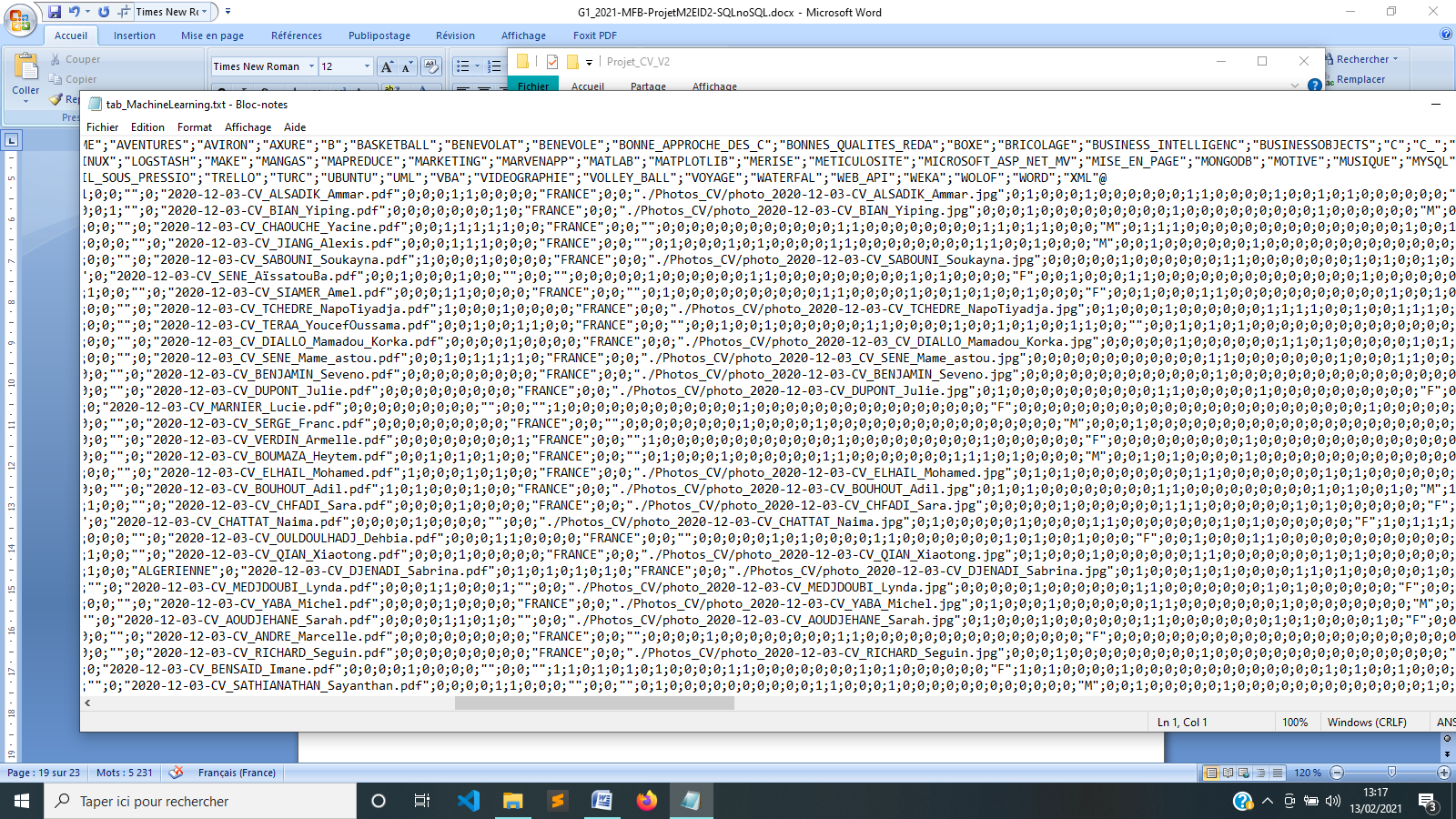
- A partir de toutes les tables, elle va créer une seule GRANDE table regroupant toutes les informations. Il est assez fréquent qu'il y ait plusieurs centaines de colonnes. Ceci s'explique car, par exemple, pour les compétences, la procédure va créer une colonne pour chaque compétences. De ce fait, on aura par exemple la colonne nommée JAVA correspondant à la compétence JAVA, et elle mettra 1 si le candidat possède la compétence JAVA, 0 sinon.

Par exemple :



Cela permettra d'avoir une grande table avec, en général, beaucoup de 0 ou 1 qui signifie que le candidat a ou n'a pas tel ou tel attribut. On fait la même chose pour les centres d'intérêts, les permis, les langues, … et on va simplement ajouter les colonnes à la grande table.

- Ensuite, l'exécution du fichier permettra d'écrire la GRANDE table dans le fichier tab\_MachineLearning.txt. La table sera ainsi disponible pour le script Python.



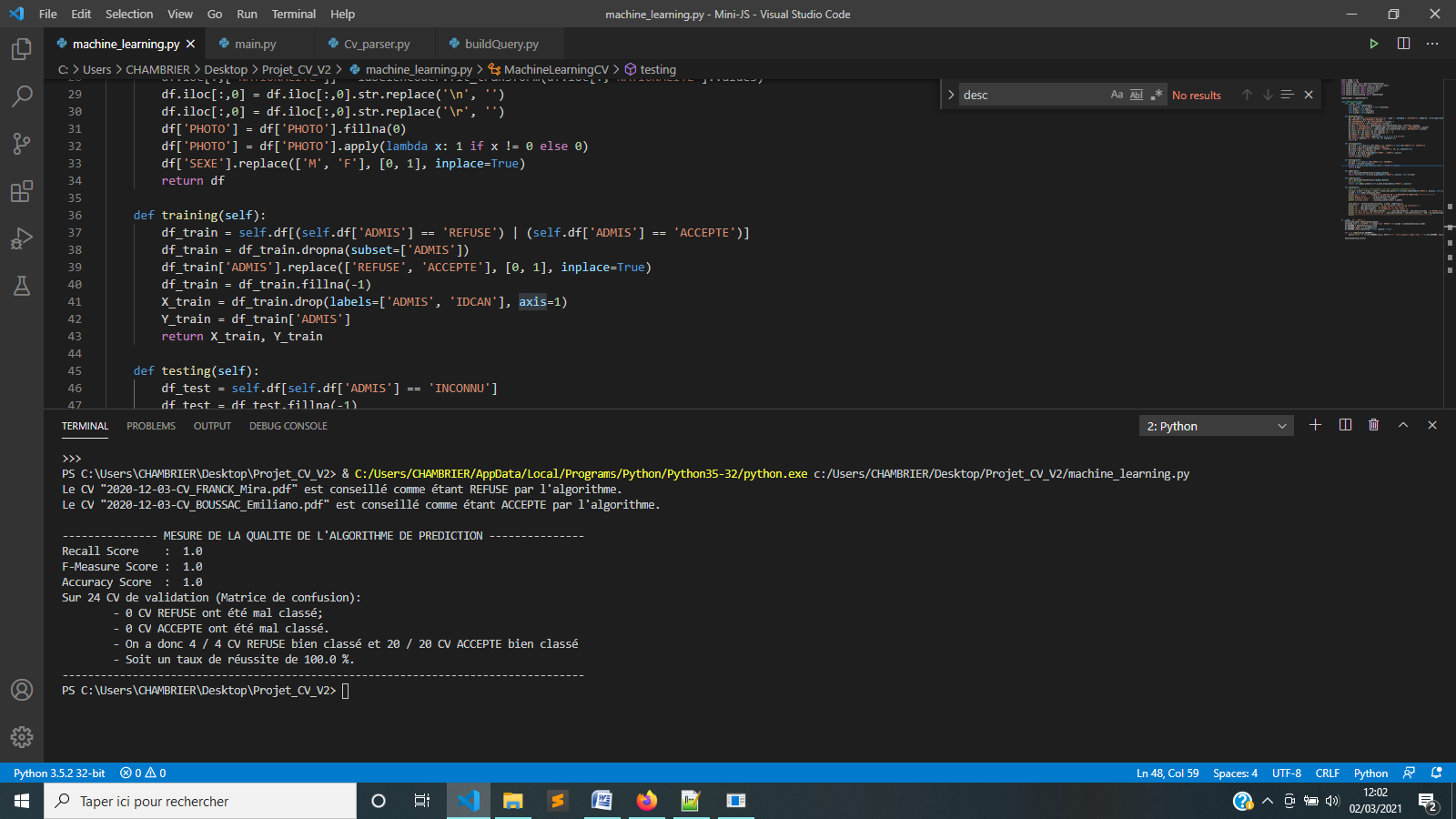
Ci-dessus, une partie de la table créée par la procédure du fichier G1\_MachineLearning\_CV.sql. On remarque qu'il y a toutes les informations qu'on a besoin pour faire du Machine Learning.

3) Une fois qu'on a obtenu une seule et unique table, on peut exécuter le script python machine\_learning.py.

Dans un premier temps, le script va permettre de formater les données. En effet, nous utilisons un arbre de décision afin d'apprendre sur les données, mais il a besoin de données numériques. Ainsi, par exemple, pour le pays d'habitation, il mettra par exemple 0 pour France, 1 pour Espagne, 2 pour Maroc... Ou encore il mettra 1 si le candidat à une photo, 0 sinon.

Ce script va permettre d'apprendre sur la table précédemment obtenue. Elle va apprendre seulement sur les données dont l'admission est connue (ADMIS = ACCEPTE ou ADMIS = REFUSE). De ce fait, elle va apprendre les caractéristiques des CV acceptés et ceux refusés afin de pouvoir prédire un nouveau CV qui entre dans la base.

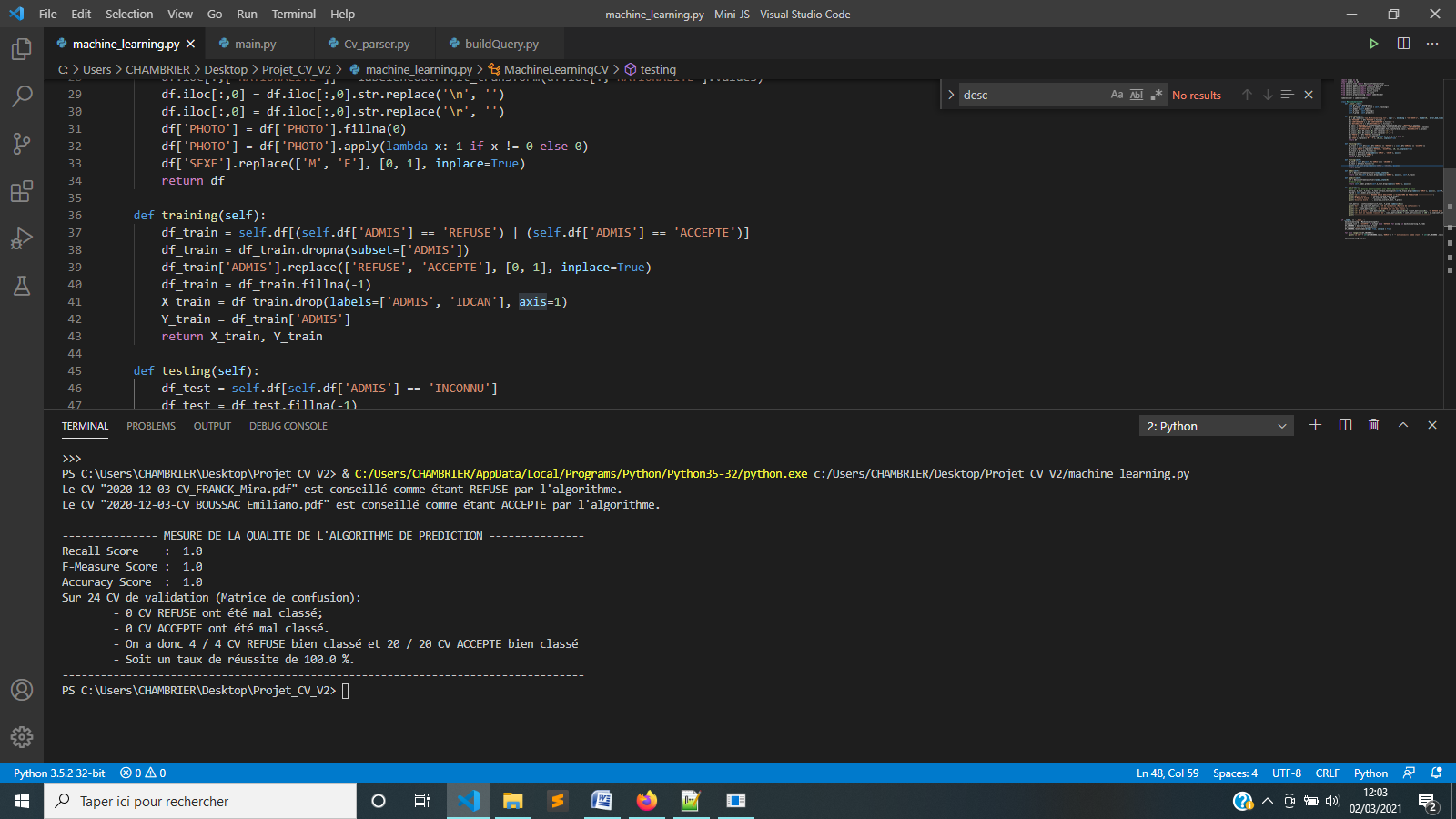
Ainsi, elle va pouvoir prédire sur les données dont la classe d'admission est inconnue (ADMIS = INCONNU). Par exemple avec les CV suivants :



Exemple sur un des CV qui a été mis dans CV\_INCONNU et retiré de CV\_ACCEPTE/CV\_REFUSE

Dans ce cas, le programme a bien prédit les classes des deux CV qu'on lui a demandés. En effet, le CV "*2020-12-03-CV\_BOUSSAC\_Emiliano.pdf"* est bien un CV d'un étudiant accepté dans la formation et le CV *"2020-12-03-CV\_FRANCK\_Mira.pdf"* est un CV qu'on a créé de toutes pièces et qui n'est pas dans la formation.

De plus, nous avons ajouté une fonction qui va permettre de calculer les scores (Recall, F-Measure, Accuracy et matrice de confusion) de notre modèle afin de d'assurer de sa fiabilité. Bien entendu cette fonction se base sur les données dont la classe est connue pour calculer son score. Elle va simplement scinder en deux les données d'apprentissage, une partie pour faire l'apprentissage (50% de données de chaque classe) et l'autre partie pour tester (50% de données de chaque classe). Ainsi, elle pourra vérifier si la classe prédite est la bonne.



Voici un exemple d'exécution.

Notons bien que l'algorithme de Machine Learning conseille simplement l'utilisateur. En pratique, l'utilisateur peut éventuellement aussi regarder le CV. L'algorithme est simplement une aide à la décision.

## ***3.2.6. Comparaison des SGBD.***

Au cours de ce projet, nous avons utilisé deux SGBD différents : Oracle et MongoDB. Les deux apportent des caractéristiques différentes.

Dans Oracle, nous avons des données structurées, les données sont rangées dans des tables sous certaines conditions. Cela à l'avantage de pouvoir assurer une bonne qualité des données (Data Cleaning). En effet, leur insertion est soumise à plusieurs vérifications :

- Contraintes de tables;

- Contraintes des méta-tables;

- Contraintes des procédures.

On ne peut donc pas insérer n'importe quelle donnée n'importe où. On va donc perdre en flexibilité mais gagner en qualité.

Ce genre de système oblige à construire une architecture de base de données robuste avec la mise en place de schéma, de tables, de contraintes, … qui peut être lourde et énergivore en temps. De plus, elle demande souvent l'intervention d'expert en gestion de bases de données afin d'assurer sa pérennité et sa sécurité. Il sera ainsi plus cher à mettre en place.

De plus, les données évoluant toujours, on risque, au fil du temps, de bloquer certaines données (qui ont évoluées), si on oublie de revoir les contraintes. Par exemple, aujourd'hui l'identifiant international des numéros des téléphones français est le +33, mais demain qui nous dit qu'il ne changera pas ? Avant la mise à jour des contraintes, on risque, durant un lapse de temps, de bloquer l'insertion de données et donc d'en perdre. Il faut donc être constamment vigilant quant-à l'évolution des choses.

Mais son gros avantage est que les données insérées sont souvent de bonne qualité (si on à mis les procédés adéquates en place). Ainsi, la manipulation des données et l'extraction d'information supplémentaire sera beaucoup plus simple.

Le second SGBD est MongoDB. Il est non structuré et donc beaucoup plus permissif. Cela à pour avantage de pouvoir insérer toutes les données possibles et inimaginables. On n'a donc pas à se soucier directement des contraintes lors d'insertions et demande peu de réflexion qu'en à la mise en place d'une architecture de base de données complexe.

Néanmoins son gros désavantage est lié au précédent paragraphe. En effet, vu qu'il n'y a presque pas de contrôle sur l'insertion des données, il peut être facile d'insérer des données qui n'ont aucun rapport les unes des autres. On risque de créer des collections très hétérogènes et leur manipulation par la suite sera d'autant plus difficile. Il faudra donc donner beaucoup plus de temps et d'argent à la correction des données en vue de leur exploitation.

Pour résumer, Oracle va demander beaucoup d'attention au début du processus afin de s'assurer que les données insérées respectent les conditions et que les tables soient bien construites pour que le système soit robuste et qu'il dure dans le temps. Grâce à cela, on pourra facilement extraire les données afin de les manipuler et de les intégrer pour les différentes applications qu'on souhaite faire. Il n'en reste pas moins nécessaire d'appliquer des procédés de Data Cleaning afin de supprimer les éventuelles erreurs (doublon, similaires,…) qui peuvent être présentes.

Du côté de MongoDB, on aura moins à faire au début du processus, les données pourront être insérées sans un réel contrôle. Néanmoins, si on veut manipuler ces données, il va falloir investir plus de temps et d'argent pour procéder à leur nettoyage.

On a donc deux SGBD qui demandent de l'investissement mais pas aux mêmes étapes.

Enfin dans notre projet, nous avons quand même décidé de procéder à un contrôle des insertions dans MongoDB afin d'assurer un bon Data Cleaning et une bonne qualité des données. Ce procédé se fait en Python lors de la création du fichier JSON. Les données sont formatées afin de coller aux contraintes qu'on a fixées plus haut dans ce rapport. De plus, les données des deux SGBD sont corrigées et formatées dans Talend afin d'assurer leur bonne intégration.

Cela nous permet de faciliter les tâches lors de l'intégration des données en Talend et d'avoir un meilleur rendu.

## ***3.2.7 Data Integration - Intégration des données de bases de données hétérogènes - Talend***

L'intégration des données est importante dans notre projet. Elle consiste à prendre des bases de données dans différents gestionnaires de bases de données, en l'occurrence Oracle/MySQL et MongoDB. Pour faire cette phase, nous avons utilisé l'outil Talend.

Cette intégration des données se fait en plusieurs étapes.

1) Récupération des données :

La première est de récupérer les données des différents gestionnaires de bases de données.

- Pour Oracle:

Nous avons créé un script SQL (G1\_SQLtoCSV\_CV.sql) permettant de convertir nos tables SQL en plusieurs fichiers .txt. Ces fichiers correspondent simplement aux données des tables mais au format texte séparé par des ';'.

- Pour MongoDB :

Nous n'avons pas fait de processus supplémentaire. En effet, le fichier JSON généré lors de l'extraction des données suffit pour faire de l'intégration.

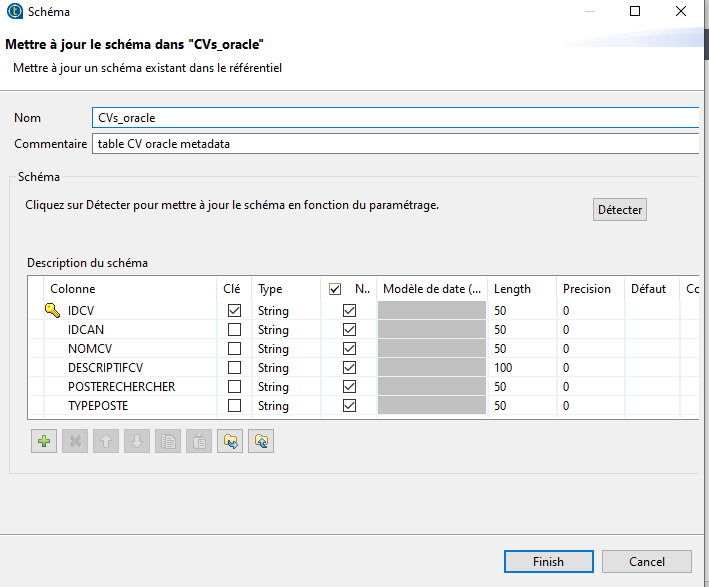
De ce fait, dans Talend, nous pouvons importer les données de ces différents fichiers.

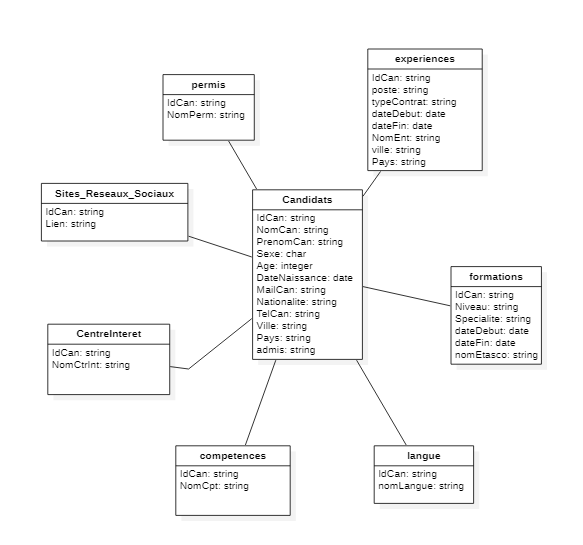
Pour construire le début de notre workflow pour chaque job on doit ouvrir ses différents fichiers. On utilise l’outils métadonnées qui permet de donner le schéma de chaque table ainsi que le type et la longueur de chaque champ.

Pour les données qui proviennent de la base Oracle, on utilise la métadonnée

Pour les données qui proviennent de la base MongoDB, on utilise la métadonnée

Par exemple pour la table CV d' Oracle, le schéma de la table se présente ainsi

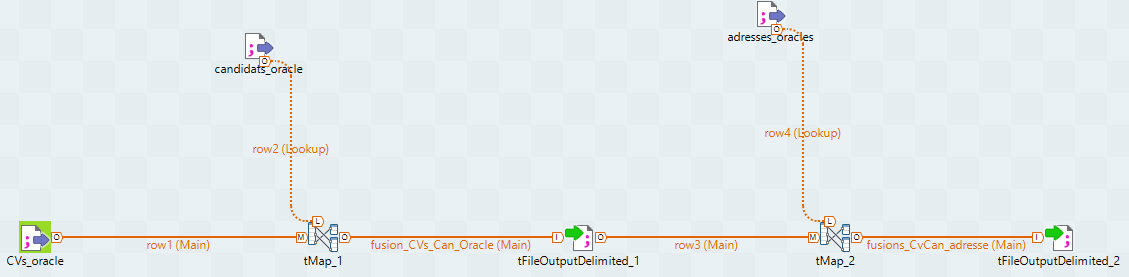




Pour la création des table du data warehouse on suit le schéma en étoile suivant :

Pour créer les différentes tables qui vont être dans le data warehouse, on va en premier créer des jobs pour fusionner les table Oracle en gardant seulement les attributs qui nous intéresse pour obtenir les différentes tables qui composeront l'entrepôt de données.

Les étapes des workflows sont l’ouverture des fichiers texte pour lire les données puis l’utilisation de l’outil Tmap pour faire des jointures avec d’autres tables et pour sélectionner seulement les colonnes qui nous intéresse

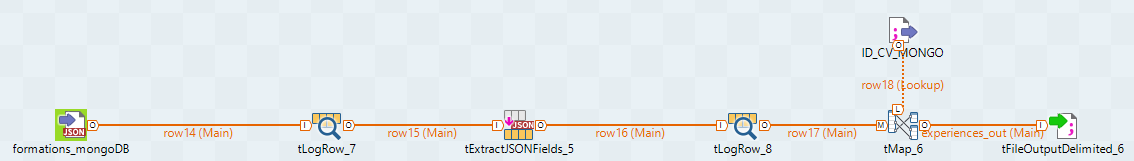


Dans cet exemple, on ouvre les table CV, candidats et adresses d’Oracle. On fait d’abord une jointure entre la table CV et Candidat grâce à la clé étrangère puis on refait encore une jointure avec la table Adresse et ainsi on crée la table principale du data warehouse qui est celle du candidat

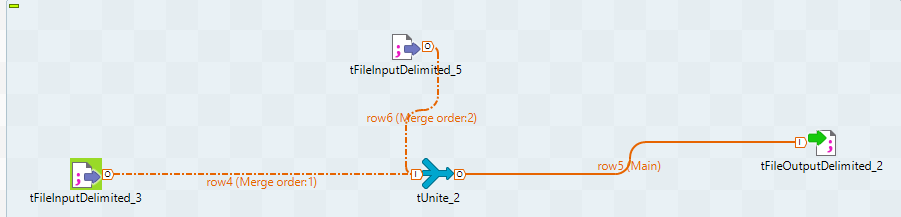
Pour effectuer les jobs Json et pour pouvoir créer des tables avec des ID comme pour les tables Oracle, on doit préalablement créer un job qui donne un ID pour chaque candidat, on utilisera ces ID à chaque fois en faisant des jointures sur le nom et prénom.

Puis on va créer des jobs pour extraire et diviser les données de la base de mongoDB qui sont dans les fichiers json pour aussi créer les mêmes tables.

Ici, pour chaque job, les étapes principales sont d’ouvrir le fichier json qui contient toutes les données puis d’extraire les champs qui nous intéresse et de créer la table du data warehouse



Et enfin on va fusionner les tables identiques pour avoir les donnée d’oracle et de mongoDB dans une même table:

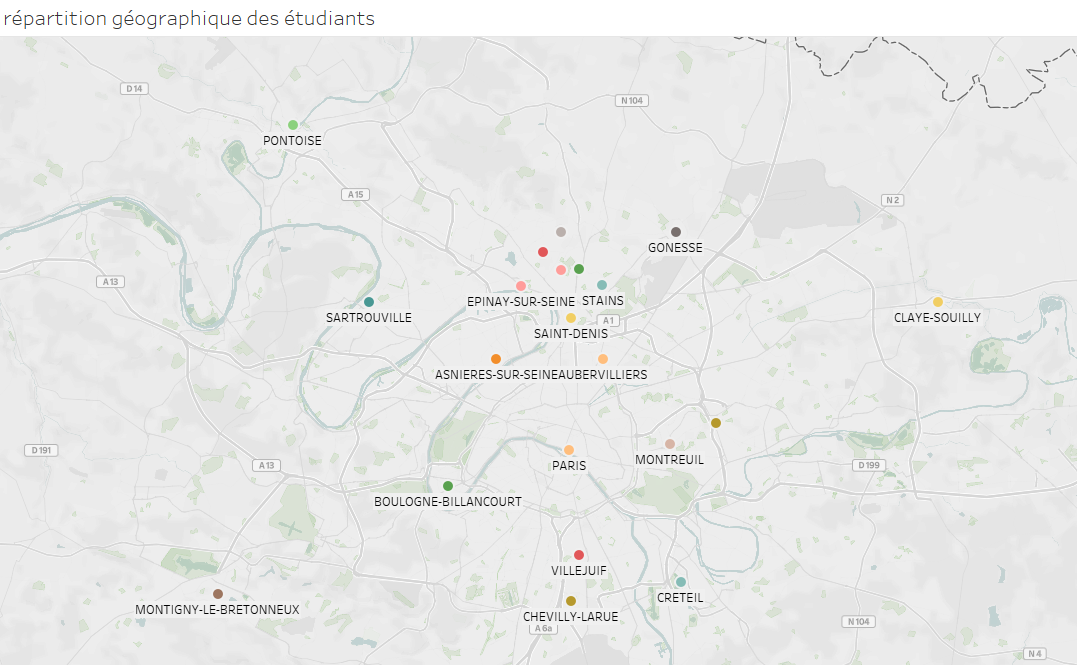


La fusion passe par une étape de traitement des éventuels doublons ou de données similaires.

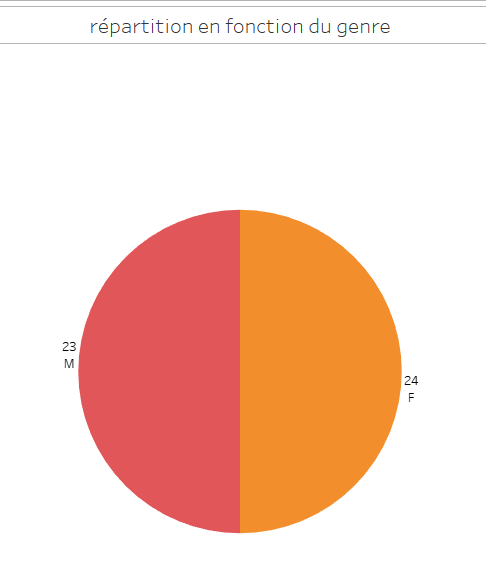
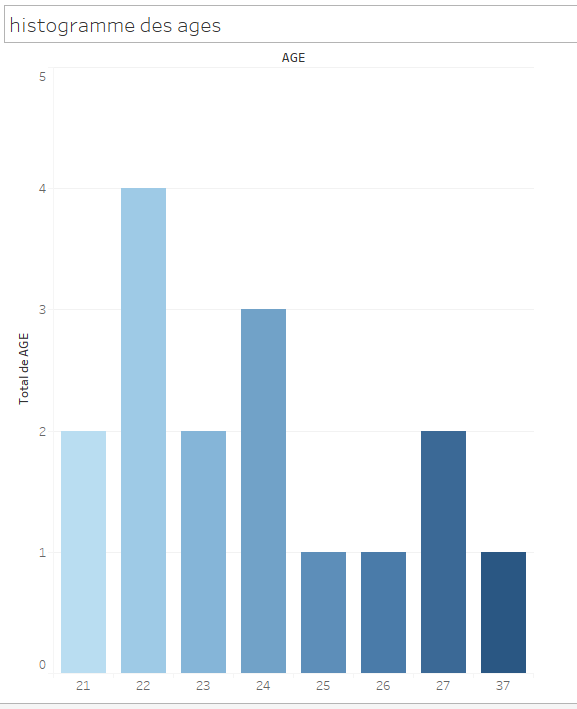
## ***3.2.8 Business Intelligence - Tableau***

Grâce à toutes les étapes précédentes, il est possible de faire de la visualisation des données de manière professionnelle et claire. Pour ce faire, on utilise l’outil Tableau.

On peut ainsi avoir, grâce aux données extraites des CV des informations sur les compétences requises, les localisations des personnes acceptées ou encore les tranches d’âges.



Grâce à cette carte, on peut voir par exemple que les étudiants se situent majoritairement dans le sud de l’Île-de-France, proche de l’Université Sorbonne Paris Nord. De plus, on peut aussi voir que certains étudiants sont assez loin de leur lieu d’étude. Par exemple, un étudiant vient chaque matin de Montigny-Le-Bretonneux pour étudier.



Grâce aux deux diagrammes ci-dessus, on observe que les âges des candidats se situent entre 21 et 37 ans. Néanmoins, les âges sont majoritairement situés entre 21 et 24 ans. De plus, on peut voir que la répartition des hommes et des femmes est presque égalitaire. En effet, nous avons 24 femmes, pour 23 hommes dans cette formation. Ainsi, on peut conclure que la formation respecte la parité homme/femme et ne fait pas preuve de sexisme.

## 

## 

Avec le diagramme en bâton juste au dessus, on voit que les compétences les plus représentées sont Python, Java, SQL, SAS, Oracle et MySQL. On voit donc que les compétences travaillées durant ce projet sont majoritairement misent sur les CV et sont pour les candidats de réels atouts vis-à-vis des entreprises.

Les différentes langues présentes dans les CVs :

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

Avec le tableau suivant, on voit l’hétérogénéité des langues parlées par les candidats de cette formation. Les candidats de cette formation représentent donc une grande diversité de cultures.

## 

## 

# 4. Sujet du Projet Annuel

Le sujet de notre projet annuel est « DES BASES AUX ENTREPÔTS DE DONNÉES, données structurées ou NON structurées (Oracle, MySQL ou MongoDB) ». Il s’agit d’un travail à réaliser par groupe d’étudiant.e.s.

Le travail à faire consiste à :

- Concevoir & développer une interface…

- Présentation orale à faire la semaine qui précède l’examen du mois de mars (fin des cours).

- Rapport (Word/PDF) + Diapositives (PPT/PDF)

- Codes (Oracle, MySQL et MongoDB)

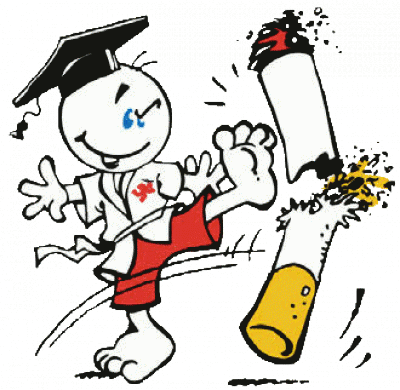
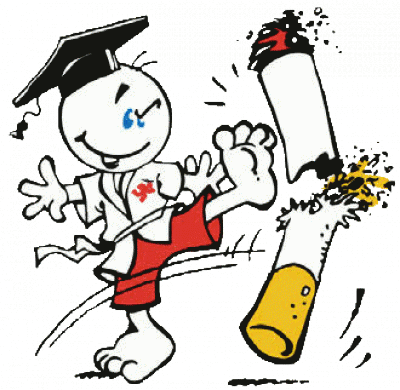
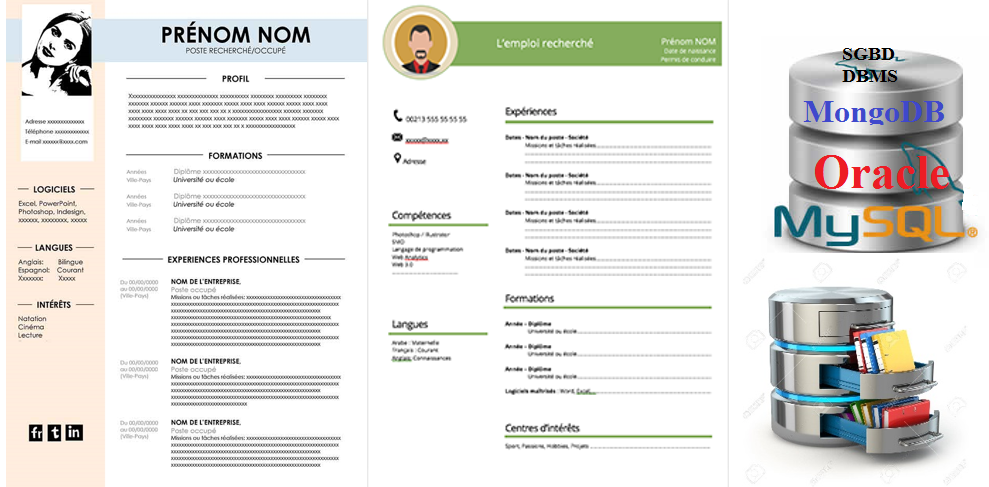
## ***4.1. : Conception & Implantation de Bases de Données***

Construction de **données structurées** à partir de **données NON structurées** (à partir de documents papiers ou numériques qui correspondent à des CVs ; ***From PDF Files to SQL or NoSQL Data Bases****).*

**8.1.1. 🡪 Donnez le schéma conceptuel** (en utilisant l’un des deux formalismes **UML** ou **E/A)**, le **schéma logique relationnel structuré SQL**et le **schéma logique Non-relationnel Non-structuré NoSQL** de la BD qui permet de stocker les données pertinentes que l’on peut trouver dans les CVs des personnes qui candidatent pour une inscription en Master 2 Informatique « Exploration Informatique et Décisionnel » (M2EID2).

**8.1.2. 🡪 Donnez une instance** (quelques lignes, au moins 50, ou objets ou enregistrements réalistes) de votre BD !

**8.1.3. 🡪 Donnez TROIS implémentations sur Oracle, MySQL et MongoDB**



## ***4.2. : Conception & Implantation d’Entrepôts de Données***

En disposant de données (stockées dans la BD des CV) sur une période de 10 années,

**4.2.1. 🡪 Concevez un entrepôt de données** qui permet d’analyser profondément le profil des candidats en M2EID2 ! (Vous êtes libres de prendre les hypothèses que vous voulez !).

**4.2.2. 🡪Donnez une instance** (quelques lignes, au moins 50, ou objets ou enregistrements réalistes) de votre ED !

**4.2.3. 🡪 Donnez TROIS implémentations sur Oracle, MySQL et MongoDB**

# Conclusion

Dans cette seconde partie du projet, nous avons étoffé notre projet. Dans un premier temps, nous avons développé toutes les fonctions d'extractions qui nous manquaient lors de la première partie. Ensuite, nous avons optimisé le code et lui faire gagner en robustesse grâce à une optimisation poussée du code afin de gagner en temps d'exécution mais aussi en qualité des données. Par exemple, chaque adresse extraite est vérifiée et complétée via une API. Cela permet de nous ancrer dans une politique de qualité de données.

Comme pour la première partie, l'utilisateur a simplement à lancer des scripts et tout se fait automatiquement, de l’extraction des données jusqu'à la création de rapport BI en passant par à l’insertion dans les tables.

De plus, lors de la première partie nous voulions mettre en place une partie Machine Learning dans notre projet afin de rapprocher encore plus le projet de notre formation. Nous l'avons fait et une des fonctionnalités de ce projet est de conseiller l'utilisateur pour un nouveau CV arrivant dans la base. En effet, nous avons entraîné un modèle afin de prédire si un CV peut être accepté ou non dans la formation en se basant sur les anciens CV (acceptés ou non).

On a donc allié plusieurs matières au sein de ce projet. C'est souvent ce qu'on retrouve dans le monde professionnel. Il nous permet donc de nous immerger au plus près des attentes d'une entreprise.

De plus, nous avons intégré plusieurs gestionnaires de bases de données. En effet, notre projet permet de générer des bases de données Oracle, MySQL et MongoDB rapidement.

Enfin afin d'apporter de la Data Integration, nous utilisons Talend qui permet de regrouper les bases de données hétérogènes en une seule et procéder à du Data Cleaning afin de réaliser des rapports BI par la suite grâce à l'outil Tableau.

En effet, comme nous regroupons plusieurs systèmes de bases de données, il est possible qu'une même donnée se retrouve répétée lors de la fusion. Talend nous permet de nettoyer cela et de formater les données afin de tirer des visualisations. Là encore, on s'inscrit dans les attentes d'une entreprise en proposant des rapports détaillés comportant des études sur les données afin d'aider à la décision.

En ce qui concerne la visualisation avec Tableau, elle permet de présenter les données afin d’aider à la prise de décisions lors de réunions par exemple. De plus, elle permet d’avoir une approche plus ludique des données avec des graphiques, schémas ou encore des cartes. Avec une bonne légende, cela permet à toutes personnes d’avoir une bonne intuition sur les données.