

Certification Développeur Data

Rapport Projet Final :

Au-delà de notre système Solaire



Table des matières :

I - Introduction

II - Analyse de la demande

- 1 - Objectif et enjeux
- 2 - Périmètre du projet
- 3 - Etat de l'art
- 4 - Schémas

III – Gestion du projet

- 1 - Planification
- 2- Outils technologiques

IV - Mise en œuvre du projet

- 1 - Les données
 - A - Source des données
 - B - Règlementation sur l'utilisation des données
 - C - Chartes des colonnes
- 2 - Analyse / Nettoyage des données
 - A - Analyse
 - B - Nettoyage
- 3 – Architecture
- 5 - Création et insertion dans la base de données
 - A – Création
 - B – Insertion
- 6 – Sauvegarde
 - 1 – Base de données
 - 2 - Métadonnées
- 7 – Visualisation

V - Amélioration

VI – Conclusion du projet

VII – Remerciements

VIII – Ressources

I - Introduction

En cette période 2021 l'activité spatiale est plus qu'active et apporte plein de découvertes et d'annonces excitantes.

Les scientifiques venant de tous les horizons sont des précurseurs essentiels pour analyser et apporter des mots, des rapports sur les nombreuses données spatiales qui sont distribuées de jours en jours.

Mais, ne vivant pas dans un monde utopique ces données n'arrivent pas par magie, des instruments scientifiques en sont les créateurs en scrutant notre Voie Lactée et même plus loin l'Univers.

National Aeronautics and Space Administration (NASA) est l'une des agences les plus connues dans le monde et met à disposition leurs données en Open-Sources afin d'aider les scientifiques ou même les curieux et amateurs de l'espace qui souhaiteraient s'informer et de créer des projets sur le sujet.

La Professeure de Sciences-Physiques Léa Martinez qui exerce au sein d'un collège et moi-même faisons partie de ces personnes qui suivent les découvertes spatiales avec intérêt et ce projet est né lors d'une conversation qui tournait autour de l'espace.

C'est finalement au terme de cette discussion que c'est dégagé le projet qui a été nommé "Au delà de notre système Solaire", le but est de préparer un support interactif au Professeure Léa Martinez pour aborder ludiquement la partie de son chapitre " Les Exoplanètes" afin d'amener avec légèreté des notions scientifiques complexes, l'analyse des données et ce que nous pouvons en tirer pour permettre une sensibilisation divertissante auprès de ses élèves.

II – Analyse de la demande

1 – Objectif et enjeux

L'objectif de ce projet est de répondre aux attentes du client qui sera réalisé grâce à une analyse des données au préalable, la création d'une base de données, l'insertion des données dans celle-ci qui viendra à être par la suite questionnée afin d'en sortir des visualisations.

Le second objectif, est que le projet devra être réalisé le plus proprement possible et plus clair afin de permettre au Professeur Léa Martinez et de ses élèves son utilisation avec aisance.

L'enjeu au cours du projet, pour moi, est donc d'appréhender la problématique et les attentes du client, afin d'en sortir l'aspect technique de la demande.

La démarche sera expliquée tout du long de ce rapport.

2 - Périmètre du projet

Interroger une base de données relationnelles afin d'en tirer des informations pour élaborer un dashboard interactif, tout en respectant le cahier des charges.

3 – Etat de l’art

Au cours de mes recherches sur le thème du projet et l’étude de celui-ci, je suis tombé sur une application en phase bêta de la NASA “Eyes on Exoplanets” qui représente l’Univers scientifiquement précis en 3D et nous permet d'explorer des milliers de systèmes planétaires connus pour orbiter autour d'étoiles lointaines. Cette application permet donc, de parcourir les planètes et leurs étoiles et d’obtenir des renseignements sur celles-ci en cliquant dessus :

<https://eyes.nasa.gov/>

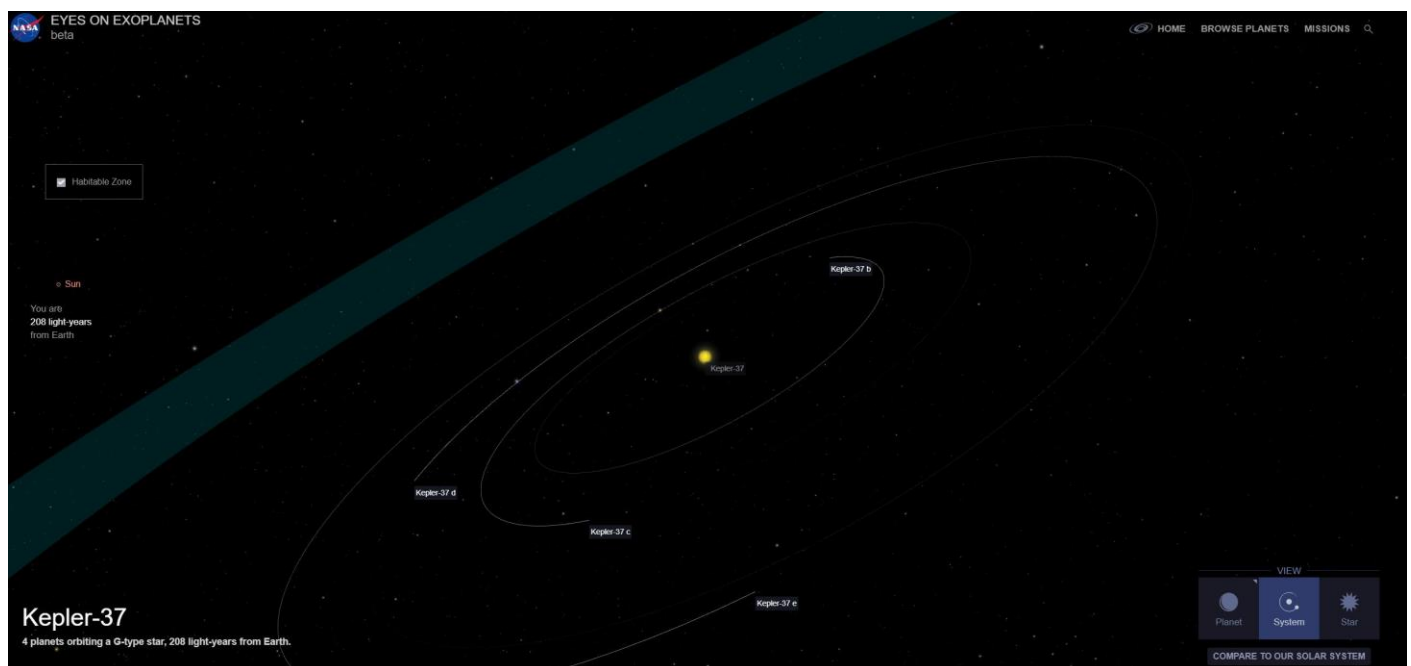
Puis un autre site de la NASA basé au California Institute of Technology (Caltech) qui permet l’accès à un catalogue astronomique et des bases de données en ligne et open sources dédiées aux exoplanètes et leurs étoiles hôtes :

Page officiel :

<https://exoplanetarchive.ipac.caltech.edu/>

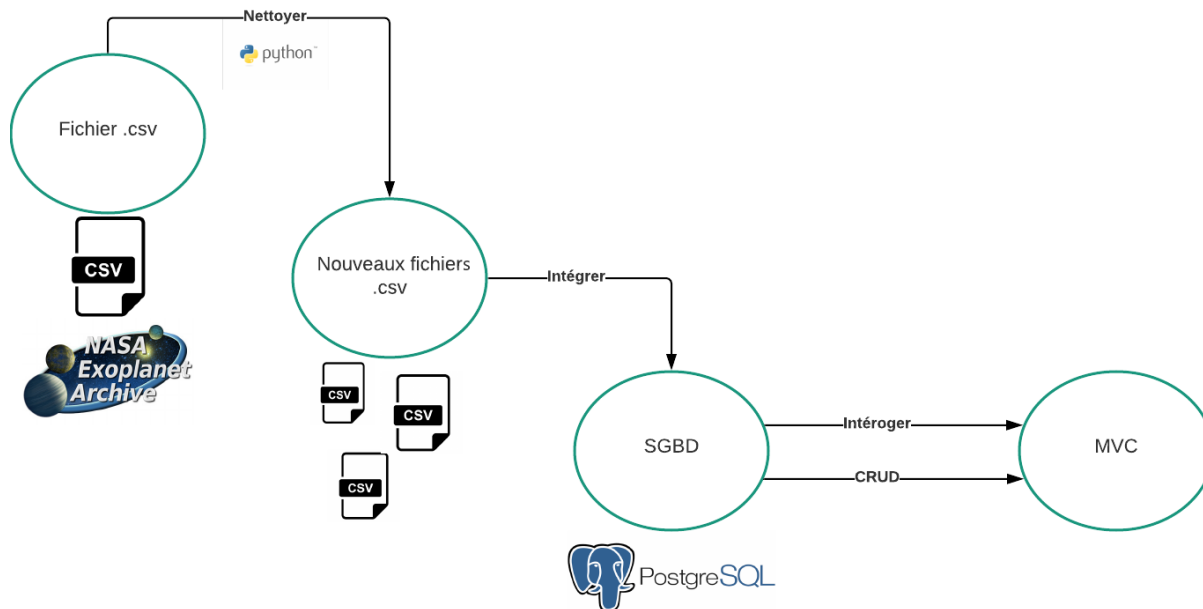
Datasets :

<https://exoplanetarchive.ipac.caltech.edu/docs/data.html>



4- Schémas

Schéma fonctionnel :

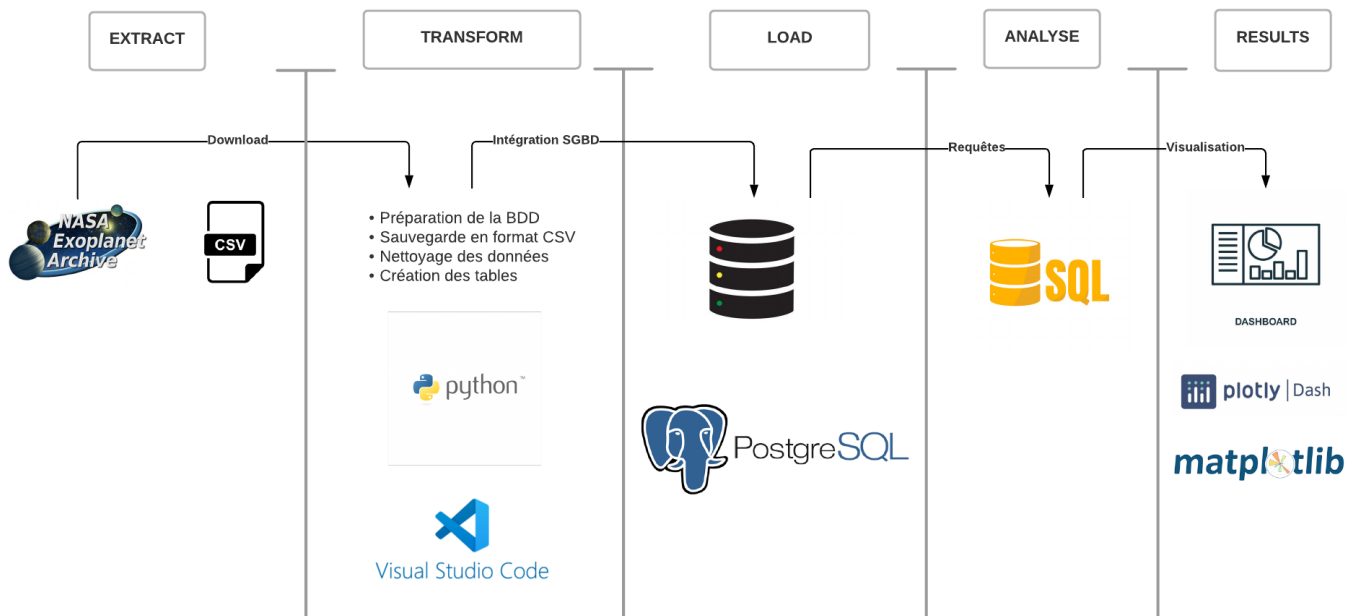


Les données d'entrée, provenant d'Exoplanet Archive ont pour format .csv, et ce fichier est situé dans le dossier **Data>csv_brut**.

Après un traitement, nettoyage et une mise en forme des données du fichier csv, celles-ci vont être sauvegarder automatiquement dans le dossier **Data>csv_cleaned**.

Par la suite, ces nouveaux fichiers csv, vont être utilisés lors de l'insertion dans la base de données.

Schéma ETL :



Dans un premier temps nous extrayons les données de **Nasa Exoplanet Archive** en les téléchargeant, ensuite nous les transformons grâce à python, puis créons en même temps la base de données.

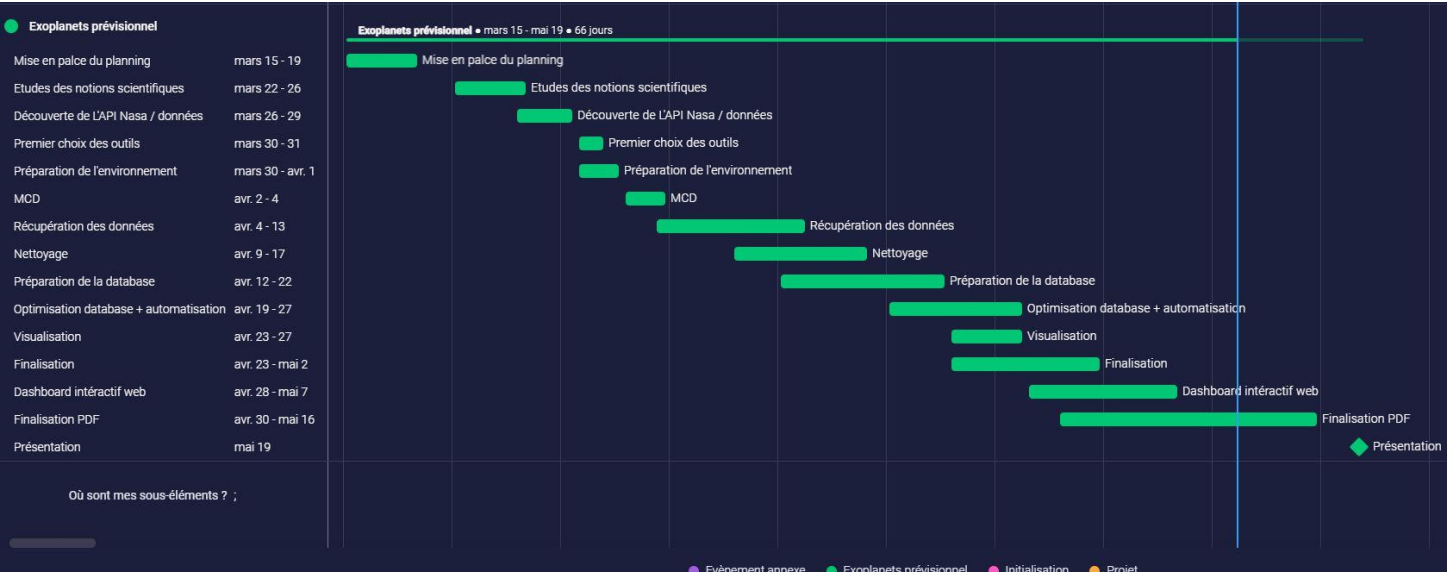
Dans un troisième temps nous faisons l'insertion dans la base de données de PostgreSQL.

Suite à l'insertion nous avons une base de données opérationnelle donc nous pouvons l'interroger grâce à des requêtes SQL enfin d'obtenir un Dashboard interactif.

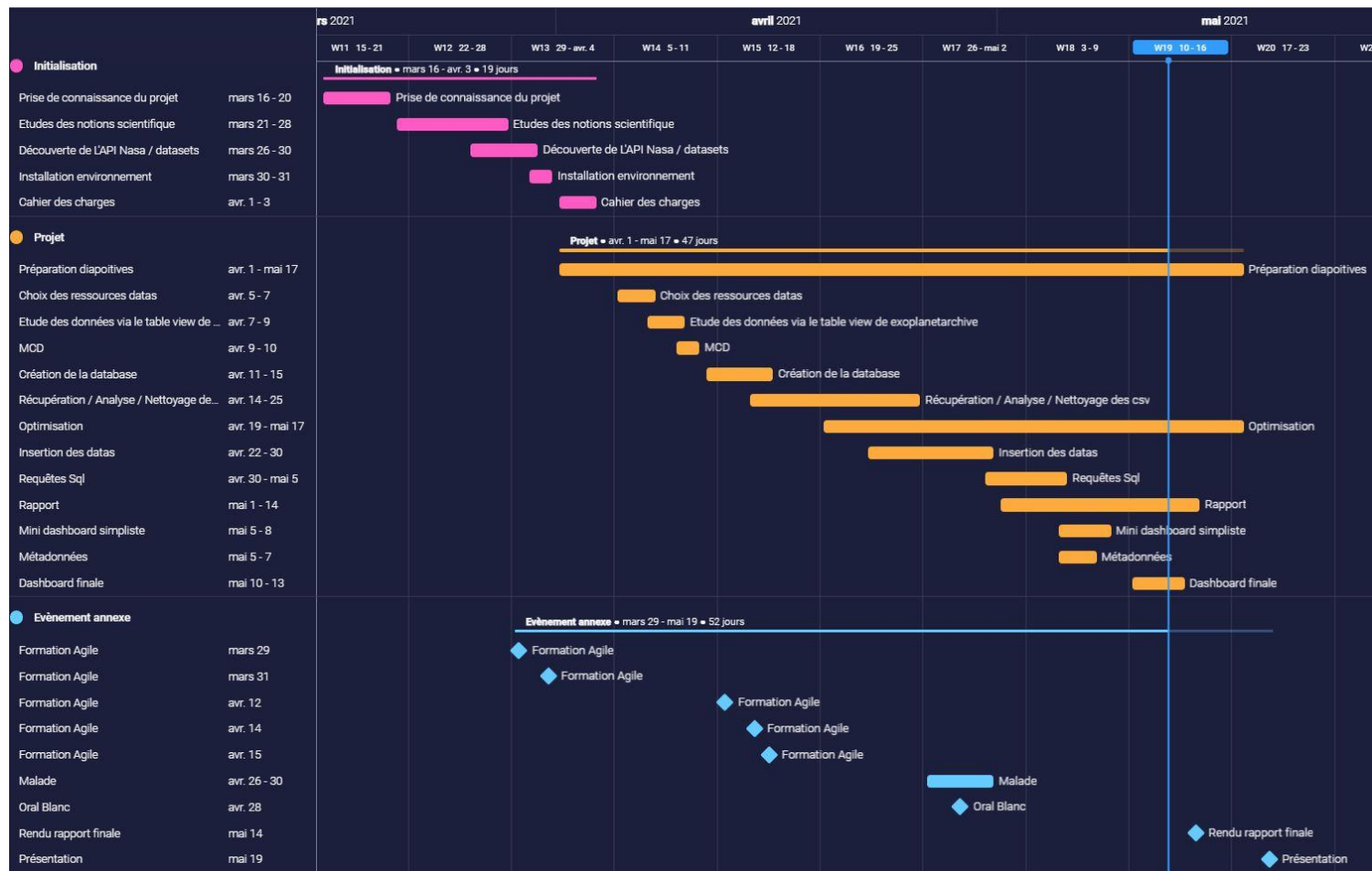
III – Gestion du projet

1 – Planification

Dans un premier j’ai établi pour le client un planning prévisionnel très simple qui décrit les grandes lignes du projet :



Puis au cours du projet un planning a été établi tout du long de celui-ci et est par conséquent plus évolué et devient plus complet :



En **rose**, nous avons l'initialisation :

Dans cette première phase nous avons la prise de connaissance des consignes et fait une petite approche de potentielles ressources.

En **orange**, nous avons le projet :

Dans cette deuxième phase nous avons la mise en œuvre du projet comprenant toutes les étapes clefs jusqu'à son rendu et sa présentation.

Puis en **bleu**, nous avons les évènements annexes :

Dans cette dernière phase nous avons tous les évènements annexes qui sont survenus au cours du projet.

2 – Outils technologiques

Ci-dessous l'ensemble des outils technologiques Utilisés :

Gestion de projet
<ul style="list-style-type: none">• Gantt de Monday.com : Logiciel libre de gestion de projet, utilisé actuellement pour la planification.• Github/Gitlab : Outils de versionning, permettant aux développeurs de stocker et partager, dans une sphère publique ou privée, son projet ou code qu'ils créent.
Logiciels
<ul style="list-style-type: none">• Jupyter : Jupyter est une application open source qui vous permet de créer des notebooks, dans ce cas-ci, le logiciel est utilisé dans une première approche de test des scripts, et utilisé pour de l'analyses.• Visual Studio Code 1.56 : éditeur de code développé par Microsoft, il sera utilisé pour l'exécution des scripts python.• DBeaver 7.3 : un logiciel permettant l'administration et le requêtage de base de données.• Exiftool 12.25 : ExifTool est un logiciel gratuit et open source pour lire, écrire et manipuler des métadonnées.
Base de données
<ul style="list-style-type: none">• PostgreSQL 4.29: système de gestion de base de données relationnelles et objets avec PgAdmin4• MongoDB 1.26 : système de gestion de base de données orienté documents, ce système sera utilisé pour les métadonnées.
Langage
<ul style="list-style-type: none">• Python 3.8 : langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Un fichier requirements.txt sera fourni pour les versions des librairies utilisées.• SQL : langage informatique normalisé servant à exploiter des bases de données relationnelles (Structured Query Language).• Dash : petit framework open-source pour python basé sur Flask. Il permet la création d'applications web analytiques, destinée à l'analyse, l'exploration, la visualisation, ou la modélisation de données.

IV – Mise en œuvre du projet







1 – Les données

A – Source des données

Les données qui vont être utilisées durant le projet proviennent du site :

<https://exoplanetarchive.ipac.caltech.edu/>

Dans la section Data le jeu de données choisi sera “Confirmed Planets - Planetary Systems Composite Data”

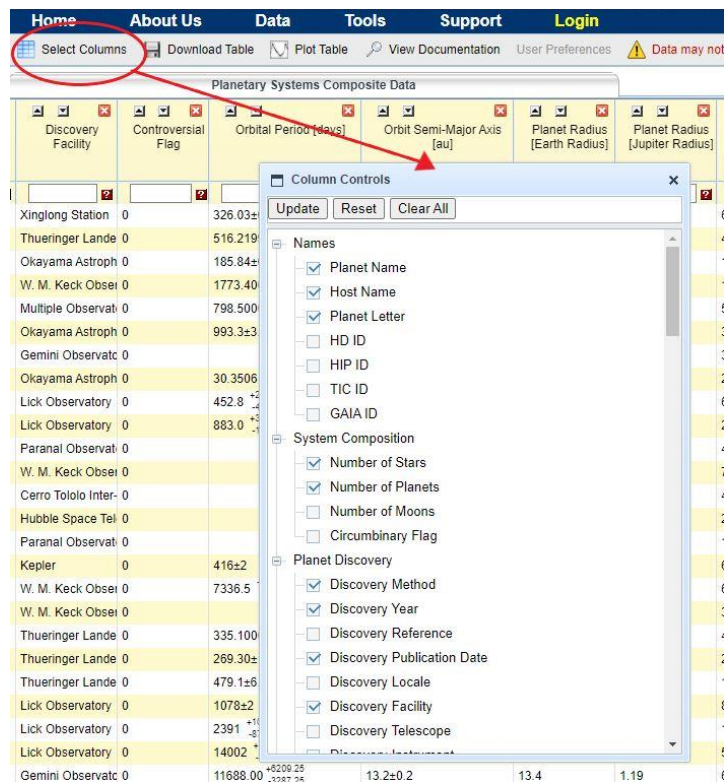
Home		About Us		Data		Tools		Support		Login	
	Select Columns		Download Table		Plot Table		View Documentation		User Preferences	 Data may not be self-consistent if drawn from multiple sources. (Learn more)	
Planetary Systems Composite Data											
Planet Name		Host Name	Number of Stars	Number of Planets	Discovery Method	Discovery Year	Discovery Facility	Controversial Flag	Orbital Period [days]	Orbit	
<input checked="" type="checkbox"/>	11 Com b	11 Com	2	1	Radial Velocity	2007	Xinglong Station	0	326.03±0.32	1.29±0.0	
<input checked="" type="checkbox"/>	11 UMi b	11 UMi	1	1	Radial Velocity	2009	Thuringer Lande	0	516.21997±3.20000	1.53±0.0	
<input checked="" type="checkbox"/>	14 And b	14 And	1	1	Radial Velocity	2008	Okayama Astroph	0	185.84±0.23	0.83	
<input checked="" type="checkbox"/>	14 Her b	14 Her	1	1	Radial Velocity	2002	W. M. Keck Obser	0	1773.40002±2.50000	2.93±0.0	
<input checked="" type="checkbox"/>	16 Cyg B b	16 Cyg B	3	1	Radial Velocity	1996	Multiple Observat	0	798.50000±1.00000	1.66±0.0	
<input checked="" type="checkbox"/>	18 Del b	18 Del	2	1	Radial Velocity	2008	Okayama Astroph	0	993.3±3.2	2.6	
<input checked="" type="checkbox"/>	1RXS J160929.1-210524 b	1RXS J160929.1-	1	1	Imaging	2008	Gemini Observat	0		330	
<input checked="" type="checkbox"/>	24 Boo b	24 Boo	1	1	Radial Velocity	2018	Okayama Astroph	0	30.3506 ^{+0.0078} _{-0.0077}	0.190 ⁺⁰ ₋₀	
<input checked="" type="checkbox"/>	24 Sex b	24 Sex	1	2	Radial Velocity	2010	Lick Observatory	0	452.8 ^{+2.1} _{-4.5}	1.333 ⁺⁰ ₋₀	
<input checked="" type="checkbox"/>	24 Sex c	24 Sex	1	2	Radial Velocity	2010	Lick Observatory	0	883.0 ^{+32.4} _{-13.8}	2.08 ⁺⁰ ₋₀	
<input checked="" type="checkbox"/>	2MASS J01033563-5515561 AB b	2MASS J0103356	2	1	Imaging	2013	Paranal Observat	0		84	
<input checked="" type="checkbox"/>	2MASS J01225093-2439505 b	2MASS J0122509	1	1	Imaging	2013	W. M. Keck Obser	0		52±6	
<input checked="" type="checkbox"/>	2MASS J02192210-3925225 b	2MASS J0219221	1	1	Imaging	2015	Cerro Tololo Inter-	0		156±10	
<input checked="" type="checkbox"/>	2MASS J04414489+2301513 b	2MASS J0441448	1	1	Imaging	2010	Hubble Space Tel	0		15.0	
<input checked="" type="checkbox"/>	2MASS J12073346-3932539 b	2MASS J1207334	1	1	Imaging	2004	Paranal Observat	0		55	
<input checked="" type="checkbox"/>	2MASS J19383260+4603591 b	2MASS J1938326	2	1	Eclipse Timing Va	2015	Kepler	0	416±2	0.92±0.0	

Il y aura cependant 2 étapes à suivre avant la récupération des données sous un format .csv :

1 – Le choix des colonnes (1 rouge) :

Pour le choix des colonnes que l'on va récupérer, veuillez vous référer à la charte des données du rapport.

Cliquez simplement sur “Select Columns”, puis cochez les champs demandés, puis update :



2 – Le choix du format de l’enregistrement (2 rouge) :

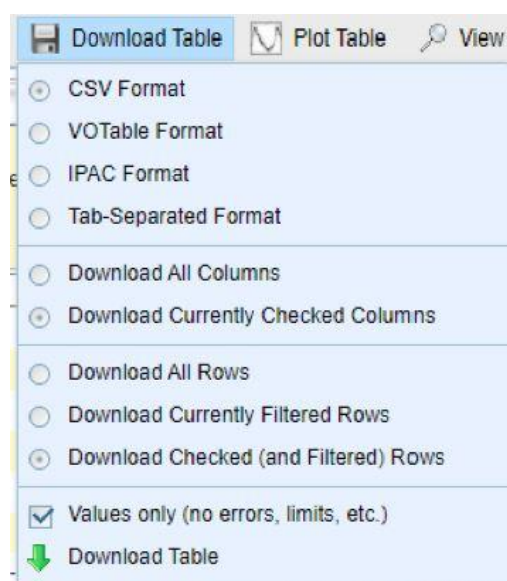
Lorsque les colonnes ont été bien choisies nous pouvons dès à présent enregistrer la table personnalisée, pour cela nous devons simplement cocher :

CSV Format : correspond au format des données.

Download currently checked Columns : télécharge les colonnes actuellement vérifiées

Download Checked (and filtered) Rows : télécharge les lignes filtrées et vérifiées.

Values only (no errors, limits, etc.) : les valeurs uniquement sans erreurs, limites, etc.



Une demande de téléchargement va alors apparaître, veuillez à l’enregistrer sous le nom exoplanets-stars.csv et le placer dans **Data>csv_brut**.

B - Règlements sur l'utilisation des données

En se référant aux deux liens suivants :

<https://exoplanetarchive.ipac.caltech.edu/docs/privacy.html>

<https://exoplanetarchive.ipac.caltech.edu/docs/acknowledge.html>

Nous sommes libres d'utiliser les données, cependant s'il vient à y avoir une publication du projet, il faut spécifier ce paragraphe :

"Cette recherche a utilisé les archives des exoplanètes de la NASA, qui sont gérées par le California Institute of Technology, sous contrat avec la National Aeronautics and Space Administration dans le cadre du programme d'exploration des exoplanètes."

C – Chartes des données

Planetary_system :

Nom colonnes	Nom complet	Description
ps_id	Planetary_system_id	L'id qui permet de définir la table
system_name	System_name	Le nom du system, autre appellation hostname
sy_snum	Nombre d'étoile	Nombre d'étoile dans le système planétaire
sy_pnum	Nombre de planètes	Nombre de planètes dans le système planétaire
sy_mnum	Nombre de satellites naturel	Nombre de satellites naturel dans le système planétaire
stars_id	Stars_id	Le champ qui sert de relation foreign key avec la table stars.

pl_id	Planets_id	Le champ qui sert de relation foreign key avec la table planets.
-------	------------	--

Stars :

Nom colonnes	Nom complet	Description
stars_id	Stars_id	L'id qui permet de définir la table
hostname	Nom de l'étoile	Le nom de l'étoile le plus couramment utilisé dans la littérature
ssp_id	Stars_specification_id	Le champ qui sert de relation foreign key avec la table stars_specification

Stars specification :

Nom colonnes	Nom complet	Description
ssp_id	Stars_specification_id	L'id qui permet de définir la table
st_spectype	Type spectrale	Le nom de l'étoile le plus couramment utilisé
st_teff	Température effective de l'étoile[K]	Température de l'étoile modélisée par un corps noir émettant la même quantité totale de rayonnement électromagnétique
st_rad	Rayon de l'étoile [Solar Radius]	Longueur d'un segment de ligne du centre de l'étoile à sa surface, mesurée en unités de rayon du Soleil
st_mass	Masse de l'étoile [Solar mass]	Quantité de matière contenue dans l'étoile, mesurée en unités de masse du Soleil
st_lum	Luminosité de l'étoile [log(Solar)]	Quantité d'énergie émise par une étoile par unité de temps, mesurée en unités de luminosités solaires
st_logg	Gravité de la surface stellaire [log10 (cm / s ** 2)]	Accélération gravitationnelle ressentie à la surface stellaire
st_age	Age de l'étoile [Gyr]	Age de l'étoile hôte du système planétaire
st_dens	Densité de l'étoile [g/cm**3]	Quantité de masse par unité de volume de l'étoile

st_vsin	Vitesse de rotation stellaire [km / s]	Vitesse de rotation à l'équateur de l'étoile multipliée par le sinus de l'inclinaison
sy_gaiamag	Magnitude de Gaia	Luminosité de l'étoile hôte mesurée en utilisant la bande Gaia en unités de magnitudes

Planets :

Nom colonnes	Nom complet	Description
pl_id	planets_id	L'id qui permet de définir la table
pl_name	Nom de la planète	Nom de la planète le plus couramment utilisé dans la littérature
pl_letter	Lettre de la planète	Lettre attribuée à la composante planétaire d'un système planétaire
date_id	Date_id	Le champ qui sert de relation foreign key avec la table date
disc_id	Disc_id	Le champ qui sert de relation foreign key avec la table discovery
psp_id	Planetary_specification_id	Le champ qui sert de relation foreign key avec la table Planetary_specification

Planets specification :

Nom colonnes	Nom complet	Description
psp_id	Planets_specification_id	L'id qui permet de définir la table
pl_orbper	Période orbitale [jours]	Temps nécessaire à la planète pour faire une orbite complète autour de l'étoile ou du système hôte
pl_rade	Rayon de la planète [Rayon de la Terre]	Longueur d'un segment de ligne du centre de la planète à sa surface, mesurée en unités de rayon de la Terre
pl_bmasse	Masse ou masse planétaire * sin (i) [Masse terrestre]	Meilleure estimation de la masse planétaire disponible
pl_dens	Densité de la planète [g / cm ** 3]	Quantité de masse par unité de volume de la planète
pl_orbeccen	Excentricité	Quantité par laquelle l'orbite de la planète s'écarte d'un cercle parfait

pl_insol	Insolation Flux [Earth Flux]	Le flux d'insolation est une autre façon de donner la température d'équilibre
pl_eqt	Température d'équilibre [K]	La température d'équilibre de la planète
pl_orbincl	Inclinaison [deg]	Angle du plan de l'orbite par rapport au plan perpendiculaire à la ligne de visée de la Terre à l'objet
pl_ratdor	Rapport de l'axe semi-majeur au rayon stellaire	La distance entre la planète et l'étoile à mi-parcours divisée par le rayon stellaire.

Date :

Nom colonnes	Nom complet	Description
date_id	date_id	L'id qui permet de définir la table
disc_year	Année de la découverte	Année de la découverte de la planète
disc_pubyear	Date de publication de la découverte	Date de publication de la publication du référent Planet Discovery

Discovery :

Nom colonnes	Nom complet	Description
disc_id	discovery_id	L'id qui permet de définir la table
discoverymethod	Méthode de découverte	Méthode par laquelle la planète a été identifiée pour la première fois
disc_locale	Localité de découverte	Lieu d'observation de la découverte de la planète (sol ou espace)
disc_facility	Centre de découverte	Nom de l'installation d'observation de la découverte de la planète
disc_telescope	Télescope Discovery	Nom du télescope des observations de découverte de la planète

Source : https://exoplanetarchive.ipac.caltech.edu/docs/API_PS_columns.html

2 - Analyse / nettoyage des données

A - Analyse

Dans cette première partie nous allons analyser le csv principal qui se situe dans **Data>csv_brut**, via un notebook sous jupyter-lab et les bibliothèques pandas et seaborn.

Cette analyse se fait en 3 étapes :

1 - Préparation du csv :

	A	B
1	# This file was produced by the NASA Exoplanet Archive http://exoplanetarchive.ipac.caltech.edu	
2	# Thu May 13 05:10:36 2021	
3	#	
4	# User preference: *	
5	#	
6	# COLUMN pl_name: Planet Name	
7	# COLUMN hostname: Host Name	
8	# COLUMN pl_letter: Planet Letter	
9	# COLUMN sy_snum: Number of Stars	
10	# COLUMN sy_pnum: Number of Planets	
11	# COLUMN sy_mnum: Number of Moons	
12	# COLUMN discoverymethod: Discovery Method	
13	# COLUMN disc_year: Discovery Year	
14	# COLUMN disc_pubdate: Discovery Publication Date	
15	# COLUMN disc_locale: Discovery Locale	
16	# COLUMN disc_facility: Discovery Facility	
17	# COLUMN disc_telescope: Discovery Telescope	
18	# COLUMN pl_orbper: Orbital Period [days]	
19	# COLUMN pl_rade: Planet Radius [Earth Radius]	
20	# COLUMN pl_bmasse: Planet Mass or Mass*sin(i) [Earth Mass]	
21	# COLUMN pl_dens: Planet Density [g/cm**3]	
22	# COLUMN pl_orbeccen: Eccentricity	
23	# COLUMN pl_insol: Insolation Flux [Earth Flux]	
24	# COLUMN pl_eqt: Equilibrium Temperature [K]	
25	# COLUMN pl_orbinc: Inclination [deg]	
26	# COLUMN pl_ratdor: Ratio of Semi-Major Axis to Stellar Radius	
27	# COLUMN st_spectype: Spectral Type	
28	# COLUMN st_teff: Stellar Effective Temperature [K]	
29	# COLUMN st_rad: Stellar Radius [Solar Radius]	
30	# COLUMN st_mass: Stellar Mass [Solar mass]	
31	# COLUMN st_lum: Stellar Luminosity [log(Solar)]	
32	# COLUMN st_logg: Stellar Surface Gravity [log10(cm/s**2)]	
33	# COLUMN st_age: Stellar Age [Gyr]	
34	# COLUMN st_dens: Stellar Density [g/cm**3]	
35	# COLUMN st_vrot: Stellar Rotational Velocity [km/s]	

Comme nous remarquons sur l'image ci-dessus, nous avons, dans notre csv des commentaires (désignés par #).

Nous allons réaliser une petite manipulation manuelle pour supprimer ces commentaires pour éviter une erreur de lecture lorsque l'on voudra le lire avec python.

Manipulation :

- Sélectionner les ligne de 1 à x
- Clic droit
- Supprimer
- Supprimer des lignes entières
- Enregistrer sous format CSV

	A	B	C	D	E	F	G	H	
1	pl name	hostname	pl letter	sv_snum	sv_pnum	sv_mnum	discoverymethod	disc_year	dis
2	11 Com b	11 Com	b	2	1	0	Radial Velocity	2007	20
3	11 UMi b	11 UMi	b	1	1	0	Radial Velocity	2009	20
4	14 And b	14 And	b	1	1	0	Radial Velocity	2008	20
5	14 Her b	14 Her	b	1	1	0	Radial Velocity	2002	20
6	16 Cyg B b	16 Cyg B	b	3	1	0	Radial Velocity	1996	19
7	18 Del b	18 Del	b	2	1	0	Radial Velocity	2008	20
8	1RXS J160929.1-210524 b	1RXS J160929.1-210524	b	1	1	0	Imaging	2008	20
9	24 Boo b	24 Boo	b	1	1	0	Radial Velocity	2018	20
10	24 Sex b	24 Sex	b	1	2	0	Radial Velocity	2010	20
11	24 Sex c	24 Sex	c	1	2	0	Radial Velocity	2010	20
12	2MASS J01033563-5515561 AB b	2MASS J01033563-5515561 A	b	2	1	0	Imaging	2013	20
13	2MASS J01225093-2439505 b	2MASS J01225093-2439505	b	1	1	0	Imaging	2013	20
14	2MASS J02192210-3925225 b	2MASS J02192210-3925225	b	1	1	0	Imaging	2015	20
15	2MASS J04414489+2301513 b	2MASS J04414489+2301513	b	1	1	0	Imaging	2010	20
16	2MASS J12073346-3932539 b	2MASS J12073346-3932539	b	1	1	0	Imaging	2004	20
17	2MASS J19383260+4603591 b	2MASS J19383260+4603591	b	2	1	0	Eclipse Timing Variations	2015	20
18	2MASS J21402931+1625183 A b	2MASS J21402931+1625183 A	b	1	1	0	Imaging	2009	20
19	2MASS J22362452+4751425 b	2MASS J22362452+4751425	b	1	1	0	Imaging	2016	20
20	30 Ari B b	30 Ari B	b	4	1	0	Radial Velocity	2009	20

2– Visualisation d'un échantillon

Dans un second temps, nous allons afficher un échantillon du csv avec la fonction `head(*nombre_de_ligne*)`

```
# On importe la bibliothèque pandas
import pandas as pd

# On définit notre dataframe (df)
df1 = pd.read_csv("C:\projet\projet-chef-d-oeuvre\Data\Csv_brut/exoplanets&stars.csv")

# Puis on en sort un échantillon
df1.head(5)
```

	pl_name	hostname	pl_letter	sy_snum	sy_pnum	sy_mnum	discoverymethod	disc_year	disc_pubdate	disc_locale	...
0	11 Com b	11 Com	b	2	1	0	Radial Velocity	2007	2008-01	Ground	..
1	11 UMi b	11 UMi	b	1	1	0	Radial Velocity	2009	2009-10	Ground	..
2	14 And b	14 And	b	1	1	0	Radial Velocity	2008	2008-12	Ground	..
3	14 Her b	14 Her	b	1	1	0	Radial Velocity	2002	2003-01	Ground	..
4	16 Cyg B b	16 Cyg B	b	3	1	0	Radial Velocity	1996	1997-07	Ground	..

5 rows × 28 columns

Nous pouvons voir que nous avons 28 colonnes, nous allons donc voir un peu plus en détail celles-ci et les données qu'elles comportent :

Il y a donc 4383 entrées de 0 à 4382 et nous avons 28 colonnes avec au total 3 formats différents :

object -> text

int64 -> integer

Float64 -> float/real

On peut voir, qu'il y a des données manquantes dans certaines colonnes, nous passons donc à l'étape 2.

```
df0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4383 entries, 0 to 4382
Data columns (total 31 columns):
#   Column                Non-Null Count  Dtype
---  -
0   pl_name                4383 non-null   object
1   hostname               4383 non-null   object
2   pl_letter              4383 non-null   object
3   sy_snum                4383 non-null   int64
4   sy_pnum                4383 non-null   int64
5   sy_mnum                4383 non-null   int64
6   discoverymethod        4383 non-null   object
7   disc_year              4383 non-null   int64
8   disc_pubdate           4383 non-null   object
9   disc_locale            4383 non-null   object
10  disc_facility           4383 non-null   object
11  disc_telescope          4383 non-null   object
12  pl_orbper               4241 non-null   float64
13  pl_rade                 4369 non-null   float64
14  pl_bmasse               4361 non-null   float64
15  pl_dens                 4279 non-null   float64
16  pl_orbeccen             3901 non-null   float64
17  pl_insol                2796 non-null   float64
18  pl_eqt                  3271 non-null   float64
19  pl_orbincl              3340 non-null   float64
20  pl_ratdor               3261 non-null   float64
21  st_spectype             1570 non-null   object
22  st_teff                 4267 non-null   float64
23  st_rad                  4249 non-null   float64
24  st_mass                 4379 non-null   float64
25  st_lum                  4253 non-null   float64
26  st_logg                 4242 non-null   float64
27  st_age                  3650 non-null   float64
28  st_dens                 4033 non-null   float64
29  st_vsin                 1589 non-null   float64
30  sy_gaiamag              4228 non-null   float64
dtypes: float64(18), int64(4), object(9)
```

3 – Les données manquantes

Puis dans un troisième temps, nous allons visualiser les données manquantes grâce à la fonction `heatmap()` de `seaborn` et nous ferons un comptage de ces données avec `isnull()` et `sum()` de `pandas`



Conclusion :

Nous possédons un csv de très bonne qualité au vu de la complexité des données proposées, mais un entretien avec le client a eu lieu pour savoir, si oui ou non, les données manquantes pouvaient l'intéresser. Et nous avons convenu que les données manquantes avaient leurs importances pour montrer aux élèves, qu'il y a un long chemin à faire pour les scientifiques chevronnés pour récupérer efficacement toutes les mesures qu'ils souhaitent.

Nous pouvons donc passer au nettoyage du csv et de ses données, enfin de créer les csv/table adéquats pour la base de données.

B – Nettoyage

Avec les fonctions `read_csv()`, `rename()`, `drop_duplicates()`, `to_csv()` de pandas, on définit la dataframe 0 comme mère, elle vient lire le fichier csv brut.

Par la suite, nous créons des variables qui correspondront aux nouveaux csv.

Exemple pour la csv/table planetary_system :

```
df0 = pd.read_csv("../Data/csv_brut/exoplanets-stars.csv")

# Création du csv planetary_system

try:
    planetary_system = df0[["hostname", "sy_snum", "sy_pnum", "sy_mnum"]]

    planetary_system.rename(columns={"hostname": "system_name"}, inplace=True)

    planetary_system = planetary_system.drop_duplicates("system_name")

    planetary_system.to_csv("../Data/csv_cleaned/planetary_system.csv", sep=";", encoding="utf-8", index=False)
```

Dans un premier temps on définit la variable planetary_system avec **df0** puis les colonnes que l'on souhaite

On renomme la colonnes "hostname" en "system_name"

On réalise un `drop_duplicates()`, ce qui permet d'enlever les doublons et éviter une redondance de la donnée.

Avant :

26	51 Peg	1	1	0
27	55 Cnc	2	5	0
28	55 Cnc	2	5	0
29	55 Cnc	2	5	0
30	55 Cnc	2	5	0
31	55 Cnc	2	5	0
32	6 Lyn	1	1	0
33	61 Vir	1	3	0
34	61 Vir	1	3	0
35	61 Vir	1	3	0

Après :

23	51 Peg	1	1	0
24	55 Cnc	2	5	0
25	6 Lyn	1	1	0
26	61 Vir	1	3	0
27	7 CMa	1	2	0
28	70 Vir	1	1	0
29	75 Cet	2	1	0

Puis nous terminons par la fonction `to_csv()`, en précisant le chemin de sauvegarde, le séparateur “,”, l’encoding, et l’index.

Nous avons donc un csv `planetary_system` propre !

Voir le scripts ***cleaning_data.py*** pour tous les csv.

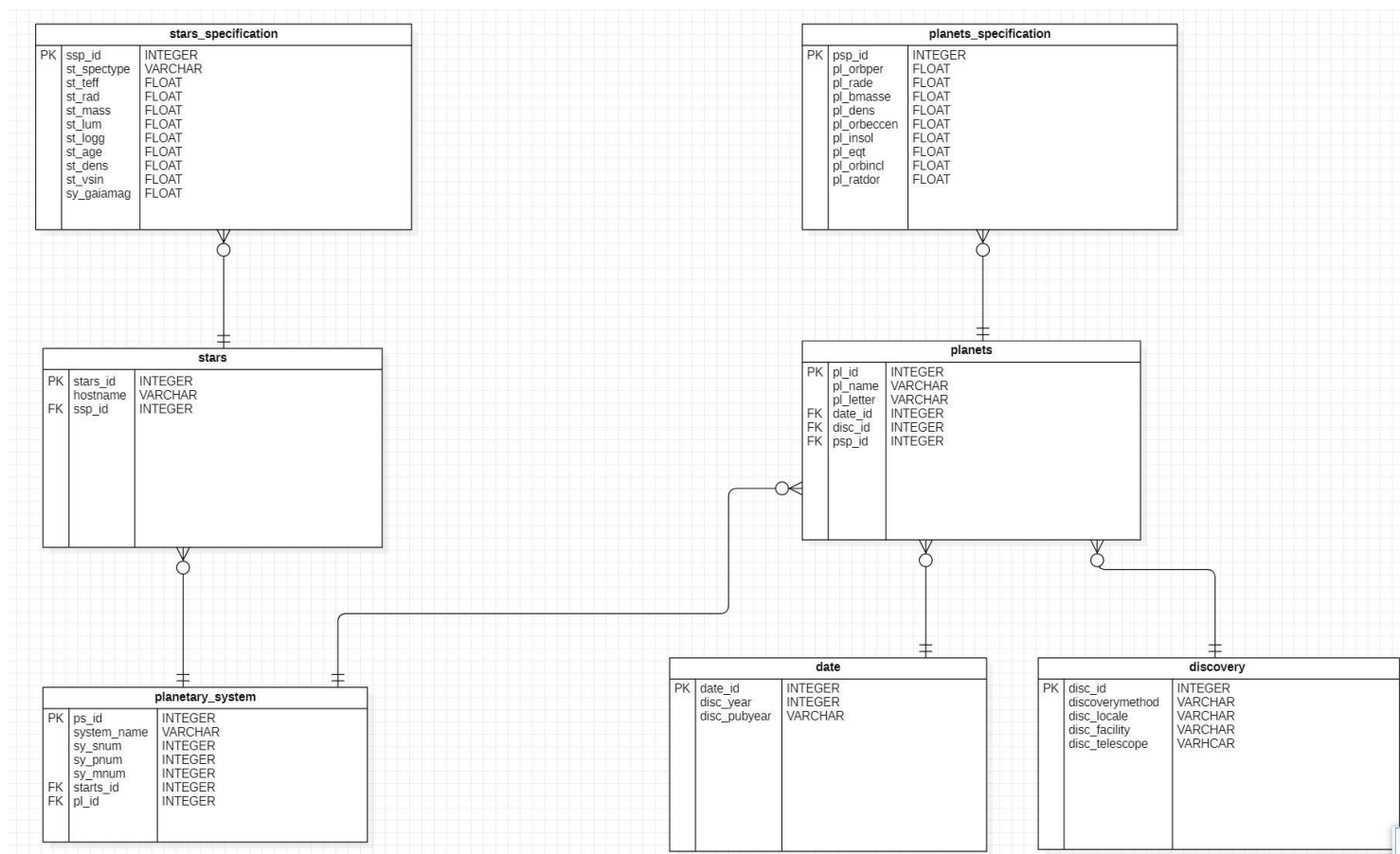
3 – Architecture

Suite à l’analyse des données téléchargées (`exoplanets-stars.csv`), l’architecture retenue sera une base de données relationnelles SQL.

Pourquoi une base de données relationnelles ?

En effet, le choix se porte sur cette forme car en utilisant des données scientifiques il est important de créer des relations fortes entre les données et leur tables respectives, tout en ayant une gestion de la base de données propre et claire en gardant au mieux les propriétés A.C.I.D.

Voici donc l'architecture retenue :



Nous avons donc 7 tables avec des relations entre elles :

La table mère, **planetary_system** qui possédera 7 colonnes, dont une clef primaire et deux clefs étrangères

- *Ps_id* : serial clef primaire.
- *Stars_id* : clef étrangère pour créer la relation avec la table de la table **stars**, relation one-to-many.
- *Pl_id* : clef étrangère pour créer la relation avec la table de la table **planets**, relation one-to-many.
-

La table **stars**, qui possède 6 colonnes, dont une clef primaire et une clef étrangère :

- *Stars_id* : serial clef primaire.
- *Ssp_id* : clef étrangère pour créer la relation avec la table **stars_specification**, relation one-to-many.

La table **stars_specification**, qui possède 8 colonnes, dont une clef primaire :

- *Ssp_id* : serial clef primaire.

La table **planets**, qui possède 3 colonnes, dont une clef primaire et trois clef étrangères :

- *pl_id* : serial clef primaire
- *date_id* : clef étrangère pour créer la relation avec la table **date**, relation one-to-many.
- *Psp_id* : clef étrangère pour créer la relation avec la table **planets_specification**, relation one-to-many.
- *Disc_id* : clef étrangère pour créer la relation avec la table **discovery**, relation one-to-many.

La table **date**, qui possède 3 colonnes, dont une clef primaire :

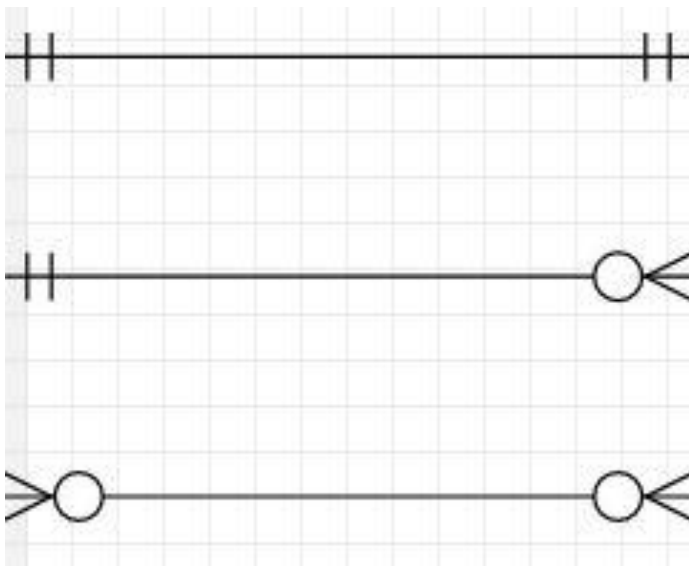
- *date_id* : serial clef primaire.

La table **planets_specification**, qui possède 9 colonnes, dont une clef primaire :

- *psp_id* : serial clef primaire.

La table **discovery**, qui possède 5 colonnes, dont une clef primaire :

- *disc_id* : serial clef primaire.



Comme vous pouvez le remarquer, les relations entre les tables sont seulement one-to-many, cependant il existe d'autres relations dans le monde de la base de données :

One to one

One to many

Many to many

5 - Création et insertion dans la base de données

A – Création

Dans un premier temps nous nous connectons à la base de données postgres et sa base de données par défaut “postgres”.

La requête alors utilisée pour la création de notre database “Exoplanets” :

```
create = '''CREATE DATABASE exoplanets;'''
```

Ensuite nous fermons la connexion à la base de données par défaut puis ouvrons de nouveau une connexion sur celle présentement créée.

Lorsque nous sommes connectées à celle-ci, nous pouvons désormais créer nos tables dans un ordre précis pour permettre la création des clefs étrangères sans défaut :

Scripts create_database.py (requête de création) :

Exemple pour la table planetary_system :

```
# Création de la table planetary_system

cursor.execute('''CREATE TABLE IF NOT EXISTS planetary_system(
    ps_id SERIAL PRIMARY KEY NOT NULL,
    system_name TEXT,
    sy_snum INTEGER,
    sy_pnum INTEGER,
    sy_mnum INTEGER,
    stars_id SERIAL,
    FOREIGN KEY(stars_id)
        REFERENCES stars(stars_id),
    pl_id SERIAL,
    FOREIGN KEY(pl_id)
        REFERENCES planets(pl_id));
''')
```

La requête SQL “create table if not exists *nom_table*” permet de créer une table si elle n’existe pas.

Ensuite nous spécifions les colonnes avec leurs formats :

Spécification
<ul style="list-style-type: none">• Serial : permet de créer un index automatiquement qui correspondra à l'identifiant de l'enregistrement, utilisé ici en clef primaire• Integer : la donnée a un format de nombre entier.• Real : la donnée a un format de nombre à virgules.• Text : la donnée a un format textuel.• Foreign key : permet de créer une relation, il faut spécifier proprement la référence de celle-ci.

B – Insertion

Pour l'insertion des données dans la base de données **exoplanets**, nous allons utiliser la fonction `to_sql()` de la bibliothèque **pandas**.

Exemple vierge :

```
dataframe.to_sql('nom_table', engine, if_exists='append', index=False)
```

Pour un exemple concret nous avons donc besoin de définir dans un premier temps notre dataframe et ensuite à spécifier les arguments de la fonction `to_sql()` qui sont :

- Le nom de la table visée (les colonnes dataframe / table doivent correspondre)
- Définir engine, qui est simplement la connection à la base de données grâce à la bibliothèque **SQLAlchemy**.
- Si la table existe (if_exists), insère les données dans la table existante ('append')
- Puis nous ne souhaitons pas avoir d'index (False)

Exemple concret :

```
planets = pd.read_csv("../Data/csv_cleaned/planets.csv")
```

Cette fonction nous permet donc d'insérer les enregistrements stockés dans **planets.csv** (voir 4 – Nettoyage des données), dans la table **planets** de la base de données **exoplanets**.

6 – Sauvegarde

A– Base de données

Pour assurer une protection sur la durabilité des données donc de la base de données, un script a été créé afin de permettre un backup de celle-ci (backup_database.py)

Pour le bon fonctionnement du script, il nécessaire de spécifier **3 variables primordiales** :

La base de données à sauvegarder :

```
OBJECT_TO_BACKUP = 'C:\\Program Files\\PostgreSQL\\13\\data\\base\\*base de données*'
```

La localisation du dossier qui va accueillir la sauvegarde :

```
BACKUP_DIRECTORY = 'C:\\Exoplanets\\projet-chef-d-oeuvre\\Backup'
```

Le nombre de sauvegarde autorisé dans le dossier d'accueil :

```
MAX_BACKUP_AMOUNT = 1
```

Ensuite, le script va réaliser une suite de code/fonction qui va permettre de créer un fichier .zip de la base données souhaitée.

B - Métadonnées

Les métadonnées sont vraiment partout, toutes nos activités sur Internet ou même sur notre bureau en local n'y échappent pas bien au contraire. Nous laissons toujours une multitude de traces qu'il est possible de voir et qui donne sur nos fichiers l'information sur sa source, son auteur et plein d'autres données importantes.

Nous avons donc utilisé un script pour récupérer les métadonnées du projet Exoplanets et de les enregistrer en JSON dans une base de données NoSQL MongoDB.

Script *metadata.py* :

```
path = "C:\\Exoplanets\\projet-chef-d-oeuvre"
exclude = set([".ipynb_checkpoints", ".git", ".vscode"])

shpfiles = []

✓ for dirpath, subdirs, files in os.walk(path):
    subdirs[:] = [d for d in subdirs if d not in exclude]
    ✓ for x in files:
        shpfiles.append(os.path.join(dirpath, x))

    #print(shpfiles)

✓ for l in shpfiles:
    input_file = l.replace("\\", "/")
    exe = "C:/Exiftool/exiftool.exe"
    process = subprocess.Popen([exe, input_file], stdout=subprocess.PIPE, stderr=subprocess.STDOUT, universal_newlines=True)
    print(input_file) #si ça bloque tu sais quel fichier n'est pas lu correctement
    metadata = {}
    ✓ for output in process.stdout:
        line = (output.strip().split(":",1))
        metadata[line[0].strip()] = line[1].strip()
```

Nous spécifions dans la variable **path** le chemin de notre dossier, puis dans **exclude**, les fichiers que nous souhaitons exclure.

Ensuite, on définit une liste vide, **shpfiles** puis nous créons une boucle qui va venir analyser tous les documents du dossier puis une autre boucle qui, avec **Exiftool** récupère les métadonnées et les ajoute dans le dictionnaire **metadata**.

Puis nous les ajoutons dans la base de données **metadonnee** :

```
#permet la création du client qui va se connecter à MongoDB
client = MongoClient()

#Préparation à la création de la base de données mabdd.
db = client.metadonnee

collection = db.metadata_collection

result = collection.insert_one(metadata)

#permet de vérifier la liste des collections créées :
db.list_collection_names()
```

7 – Visualisation

Deux exemples de questions du Professeure Léa Martinez :

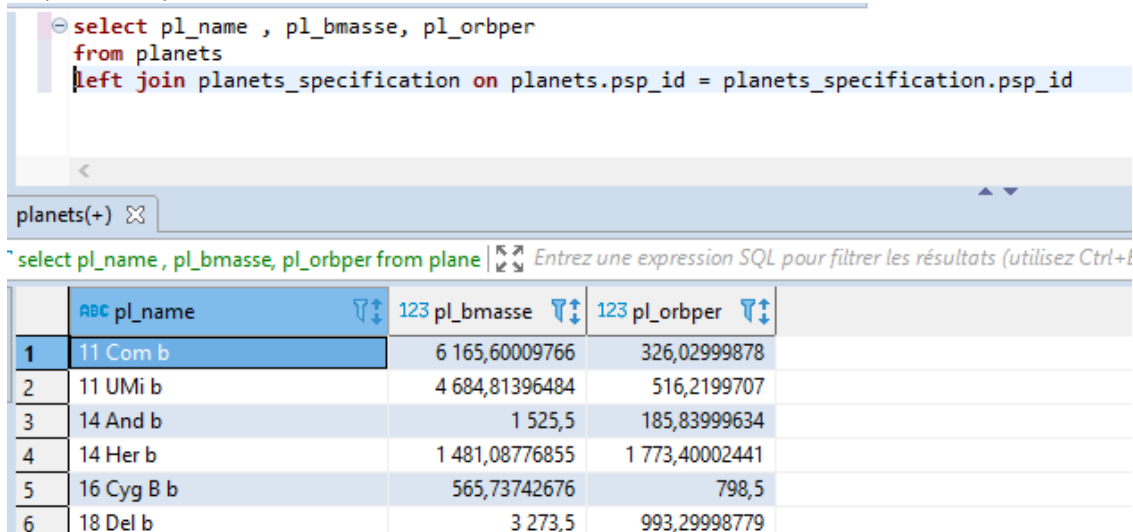
1 - Est-ce que la masse de l'exoplanètes influe telle sa période orbitale ?

2 – Diagramme de Hertzsprung Russell

Exemple de démarche :

1 - Est-ce que la masse de l'exoplanètes influe telle sa période orbitale ?

Pour répondre à cette question nous avons dans un premier temps utilisé DBeaver pour faire une première requête simple.



The screenshot shows the DBeaver SQL editor with a query window titled 'planets(+)' containing the following SQL query:

```
select pl_name , pl_bmasse, pl_orbper
from planets
left join planets_specification on planets.psp_id = planets_specification.psp_id
```

Below the query, the results are displayed in a table with the following columns: **pl_name**, **pl_bmasse**, and **pl_orbper**. The table contains 6 rows of data.

	ABC pl_name	123 pl_bmasse	123 pl_orbper
1	11 Com b	6 165,60009766	326,02999878
2	11 UMi b	4 684,81396484	516,2199707
3	14 And b	1 525,5	185,83999634
4	14 Her b	1 481,08776855	1 773,40002441
5	16 Cyg B b	565,73742676	798,5
6	18 Del b	3 273,5	993,29998779

Détails de la requête SQL :

Nous faisons un **SELECT** sur les colonnes **pl_name** de la table **planets**. Ensuite **pl_bmasse** , **pl_orbper** de la table **planets_specification** puis nous joignons celle-ci grâce au **LEFT JOIN** en spécifiant les foreigns keys.

Le résultat nous convient, donc nous transposons le code dans le script queries.py :

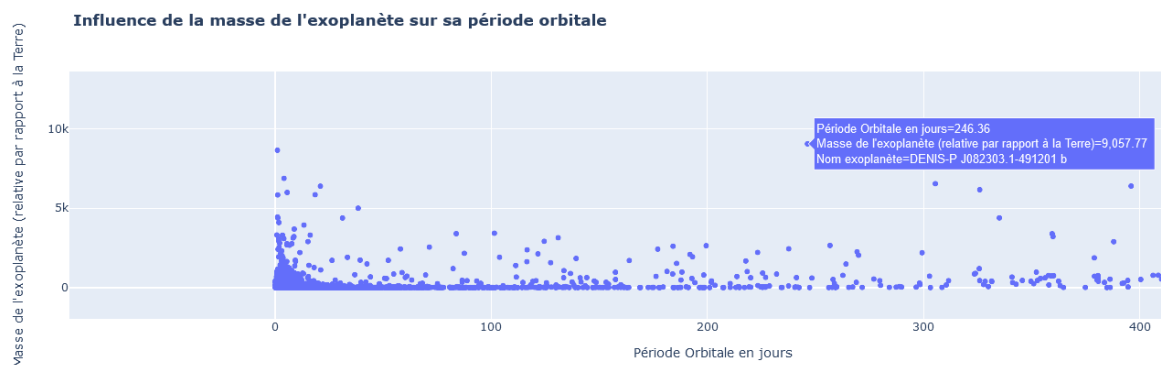
```
# Masse de l'exoplanète / période orbitale

df2 = pd.read_sql("""
SELECT pl_name , pl_bmasse, pl_orbper
FROM planets
LEFT JOIN planets_specification ON planets.psp_id = planets_specification.psp_id
""", engine)

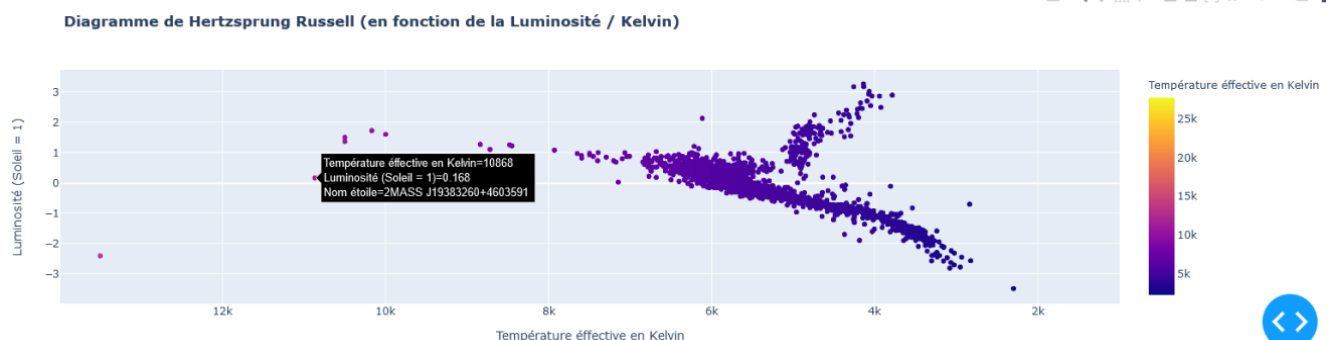
fig2 = px.scatter(df2,
x="pl_orbper",
y="pl_bmasse",
hover_data=['pl_name'],
labels=dict(pl_orbper="Période Orbitale en jours",
pl_bmasse="Masse de l'exoplanète (relative par rapport à la Terre)",
pl_name="Nom exoplanète"),
title="Influence de la masse de l'exoplanète sur sa période orbitale <b>"
)

fig2.update_xaxes(range=[-3000, 40000])
fig2.update_yaxes(range=[-3000, 40000])
```

La requête SQL est définie dans la variable df2 avec à la fin la fonction engine qui est définie en amont et permet de se connecter à la database et de la questionner directement
 Ensuite nous créons la figure pour la visualisation (fig2) en utilisant la bibliothèque de potly et la fonction px.scatter en définissant ces arguments.



2 – Diagramme de Hertzsprung Russell :



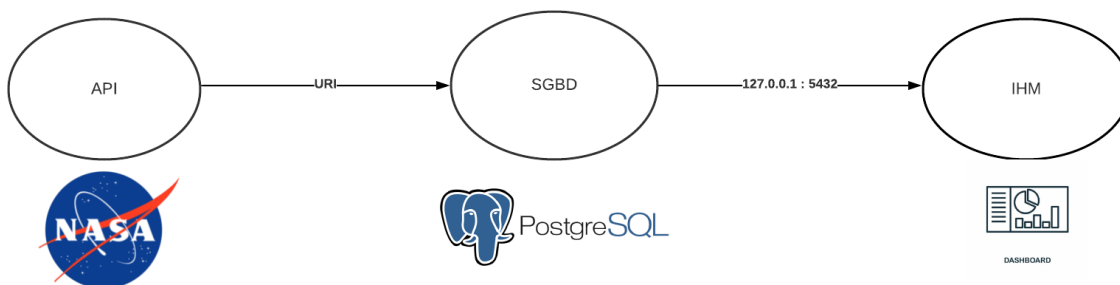
En astronomie, le diagramme de Hertzsprung-Russell, est un graphique dans lequel est indiqué la luminosité d'un ensemble d'étoiles en fonction de leur température effective permettant de les classer, exemples : naines blanche, géante.

V - Amélioration

Pour la suite du projet, j'envisage dans un premier temps de poursuivre sur les données actuelles pour pousser au maximum leurs utilisations.

Ensuite l'utilisation d'une API fournit par la NASA pourrait être un choix intelligent pour permettre de fournir une application interactive et ludique aux élèves ou encore aux amateurs.

Schéma fonctionnel d'une API :



Et au summum du projet, pourquoi ne pas développer un petit jeu interactif où les adolescents seront eux même les scientifiques à explorer les databases ou datasets et chercheront par eux même les informations adéquates pour établir des visualisations simples mais intéressante.

VI– Conclusion du projet

Après ces semaines studieuses dans la compréhension des enjeux et de la demande client, ce fut très formateur d'appliquer de A à Z toutes les notions et compétences acquises durant ma formation de développeur data, que ce soit de l'analyse, la gestion de base de données à la mise en œuvre des visualisations, sans pour autant mettre toutes les petites étapes cachées de l'iceberg qu'est le développement.

Cependant je ne compte pas laisser ce projet de côté, puisque la partie sur laquelle je souhaite me consacrer est d'améliorer le projet en général, en particulier la base de données et ce qui la compose, puis de pousser l'utilisation de celles-ci afin d'en fournir des visualisations plus approfondies sur ces exoplanètes éloignées, trop éloignées.

“To confine our attention to terrestrial matters would be to limit the human spirit.”

~Stephen Hawking~

VII– Remerciement

Tout d'abord, je souhaite remercier SIMPLON qui m'a donné la chance de poursuivre la formation. C'était primordial pour moi d'intégrer le milieu informatique afin de développer des compétences pour mon futur professionnel.

Ensuite je remercie mes formateurs Mr. Pierre Jaumier, Mr. Vincent Lhoste et tout particulièrement Mr. Christophe Chevalier, qui pendant ces mois de confinement ont assuré le suivi.

Florence Sahal pour ces contacts réguliers et bienveillants.

Les apprenant.e.s, Joris Teyssier, Corentin Pingeon, Maëlle Coriou, Jonathan Goergen, Alicia Schouman, Mayel Pèllé, Lamia Adjir et Jordan Troadec qui m'ont été d'une grande aide et d'un soutien sans égal.

Merci à vous lecteurs !

VIII – Ressources

<https://www.nasa.gov/>

<https://stackoverflow.com/>

<https://www.wikipedia.org/>

<https://plotly.com/>

<https://www.eso.org/public/norway/images/potw1611a/?lang>