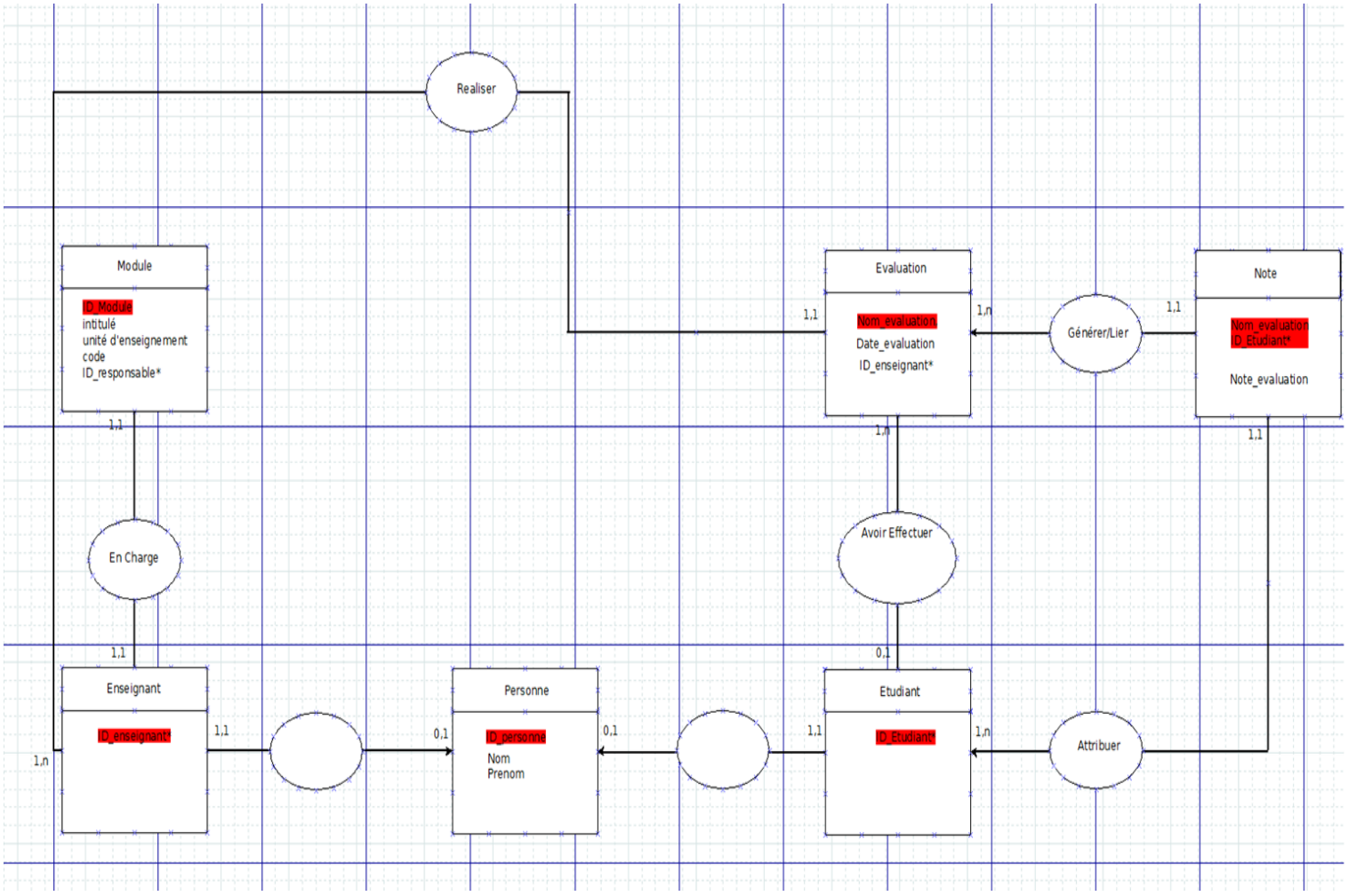
Abelard Rodlens INDRA BUT INFO 1

SAE104-BDD

2.1 Modélisation et script de création «sans AGL»

1. Modèle entités-association



2. Schéma relationnel

Schéma relationnel :

-personne (id_personne, nom, prenom)

-enseignant(id_enseignant) où id_enseignant est à la fois clé primaire du schéma de relation enseignant, et une clé étrangère qui fait référence au schéma de relation de personne.

-etudiant(id_etudiant) où id_etudiant est à la fois clé primaire du schéma de relation étudiant ,et une clé étrangère qui fait référence au schéma de relation de personne.

-module(id_module, intitulé, unité d'enseignement, code, id_responsable) où id_responsable est une clé étrangère qui fait référence au schéma de relation de enseignant.

-evaluation(nom_evaluation, date_evaluation, id_enseignant) où id_enseignant est une clé étrangère qui fait référence au schéma de relation de enseignant.

-note (nom_evaluation, id_etudiant, note_evaluation) où nom_evaluation et id_etudiant sont à la fois clé primaire de la relation note, et clé étrangère qui font respectivement référence aux schéma de relation etudiant et evaluation.

3. Script SQL de création des tables

Table personne:

```
CREATE TABLE personne
(
	id_personne INTEGER PRIMARY KEY,
	nom VARCHAR,
	prenom VARCHAR
) ;
```

Table enseignant :

```
CREATE TABLE enseignant
(
		id_enseignant INTEGER REFERENCES
personne(id_personne),
		PRIMARY KEY(id_enseignant)

);
```

Table etudiant :

```
CREATE TABLE etudiant
(
        id_etudiant INTEGER REFERENCES personne(id_personne),
        PRIMARY KEY(id_etudiant)

);

CREATE TABLE module
(
        id_responsable INTEGER REFERENCES
enseignant(id_enseignant),
        id_module INTEGER PRIMARY KEY,
        ue VARCHAR,
        code VARCHAR,
        intitule VARCHAR


);
```

Table evaluation :

```
CREATE TABLE evaluation
(
        id_enseignant INTEGER REFERENCES
enseignant(id_enseignant),
        nom_evaluation VARCHAR PRIMARY KEY,
        date_evaluation DATE


);
```

Table note :

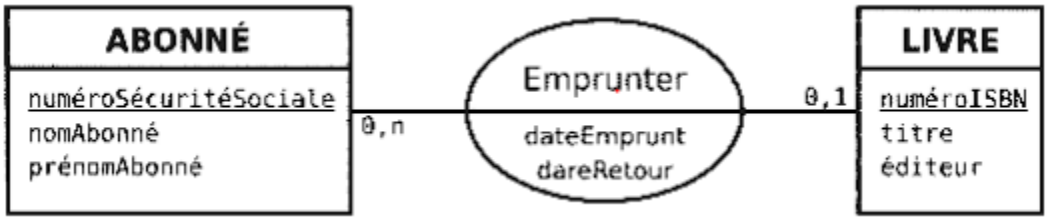
```
CREATE TABLE note
(
        id_etudiant INTEGER REFERENCES etudiant(id_etudiant),
        nom_evaluation VARCHAR REFERENCES
evaluation(nom_evaluation),
        note_evaluation FLOAT,
        PRIMARY KEY(nom_evaluation, id_etudiant)

);
```

2.2 Modélisation et script de création « Avec AGL »

1. type-association fonctionnel AGL

type-association fonctionnel Cours

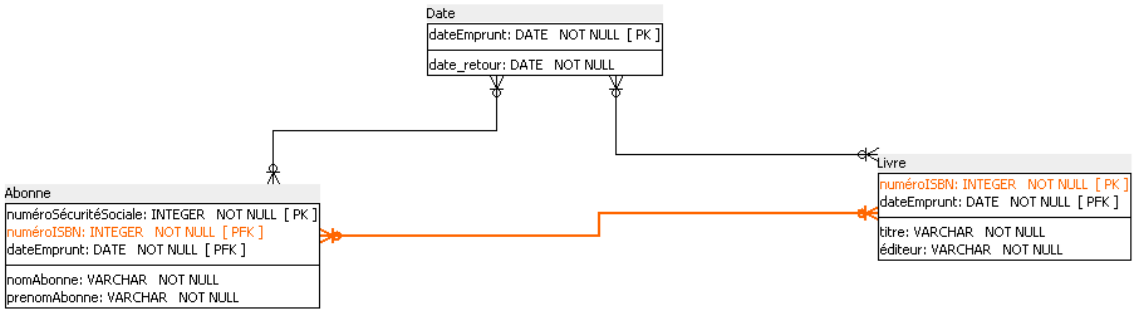


Dans le type-association AGL les type-association ne sont pas visible contrairement à la version du cours, les cardinalités ne sont visibles que par des schémas de liens (flèches qui lient les types entités) différents, alors que dans le cours les cardinalités sont représentées par 0,1 et n, de plus les clé primaire et étrangère sont spécifier et voyante dans les autres tables associé.

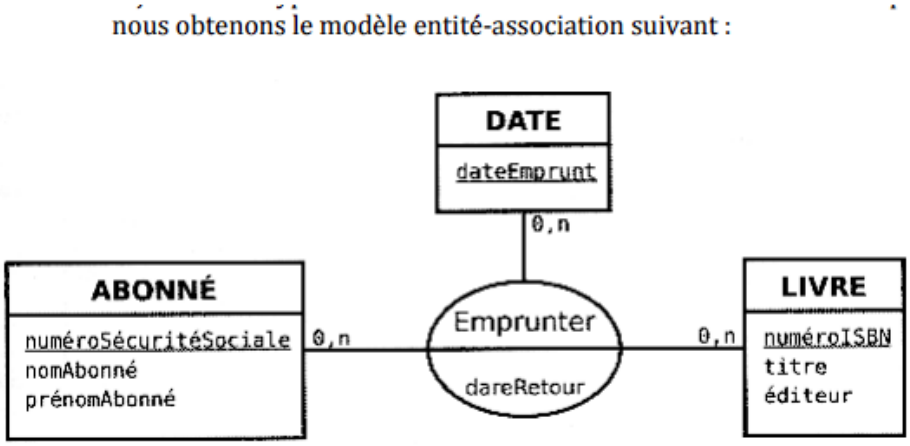
Dans la version AGL le type des attributs est spécifié (Varchar, Integer),la syntaxe est différente de celle vue en cours, de plus vue qu'il n'y a pas de type-association dans la version AGL, les 2 types-entité ne sont pas mis en relation car ils n'ont pas de clé étrangère.

2.2

Types-association maillé AGL

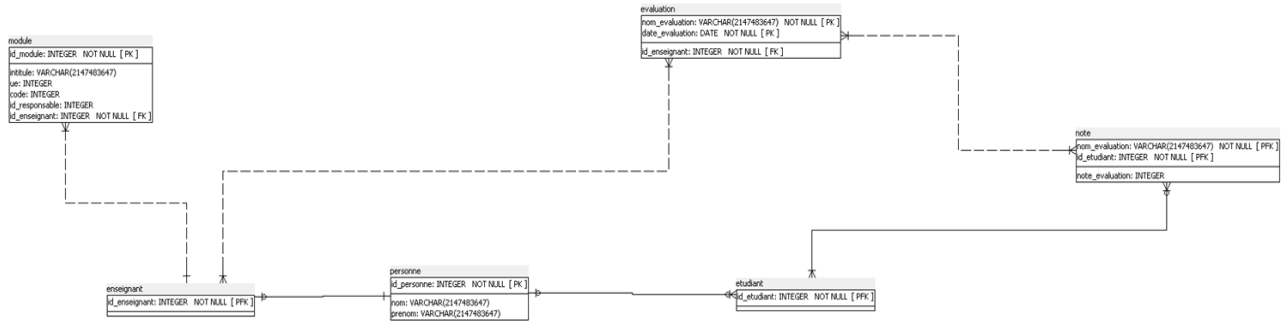


Types-association maillé Cours



C’est globalement comme la 2.1 les différences ne se trouvent que dans la syntaxe, types-association pas représenté …

4. Modèle entités-associations avec AGL



Je trouve le choix de cardinalité un peu limite, je ne peut pas mettre les cardinalité de personne, enseignant, et étudiant en adéquation avec mon schéma de base.

4 . Script SQL généré automatiquement par AGL

```
CREATE TABLE public.personne (
	id_personne INTEGER NOT NULL,
	nom VARCHAR,
	prenom VARCHAR,
	CONSTRAINT personne_pkey PRIMARY KEY (id_personne)
);


CREATE TABLE public.enseignant (
	id_enseignant INTEGER NOT NULL,
	CONSTRAINT enseignant_pkey PRIMARY KEY (id_enseignant)
);


CREATE TABLE public.evaluation (
	nom_evaluation VARCHAR NOT NULL,
	date_evaluation DATE NOT NULL,
	id_enseignant INTEGER NOT NULL,
	CONSTRAINT evaluation_pkey PRIMARY KEY (nom_evaluation,
date_evaluation)
);


CREATE TABLE public.module (
	id_module INTEGER NOT NULL,
	intitule VARCHAR,
	ue VARCHAR NOT NULL,
```

```
	code VARCHAR NOT NULL,
	id_responsable INTEGER,
	id_enseignant INTEGER NOT NULL,
	CONSTRAINT module_pkey PRIMARY KEY (id_module)
);


CREATE TABLE public.etudiant (
	id_etudiant INTEGER NOT NULL,
	CONSTRAINT etudiant_pkey PRIMARY KEY (id_etudiant)
);


CREATE TABLE public.note (
	nom_evaluation VARCHAR NOT NULL,
	id_etudiant INTEGER NOT NULL,
	note_evaluation REAL NOT NULL,
	CONSTRAINT note_pkey PRIMARY KEY (nom_evaluation, id_etudiant)
);


ALTER TABLE public.etudiant ADD CONSTRAINT etudiant_id_etudiant_fkey
FOREIGN KEY (id_etudiant)
REFERENCES public.personne (id_personne)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.enseignant ADD CONSTRAINT
enseignant_id_enseignant_fkey
FOREIGN KEY (id_enseignant)
REFERENCES public.personne (id_personne)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.module ADD CONSTRAINT module_id_responsable_fkey
FOREIGN KEY (id_enseignant)
REFERENCES public.enseignant (id_enseignant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.evaluation ADD CONSTRAINT
evaluation_id_enseignant_fkey
```

```
FOREIGN KEY (id_enseignant)
REFERENCES public.enseignant (id_enseignant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.note ADD CONSTRAINT note_nom_evaluation_fkey
FOREIGN KEY (nom_evaluation)
REFERENCES public.evaluation (nom_evaluation)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.note ADD CONSTRAINT note_id_etudiant_fkey
FOREIGN KEY (id_etudiant)
REFERENCES public.etudiant (id_etudiant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

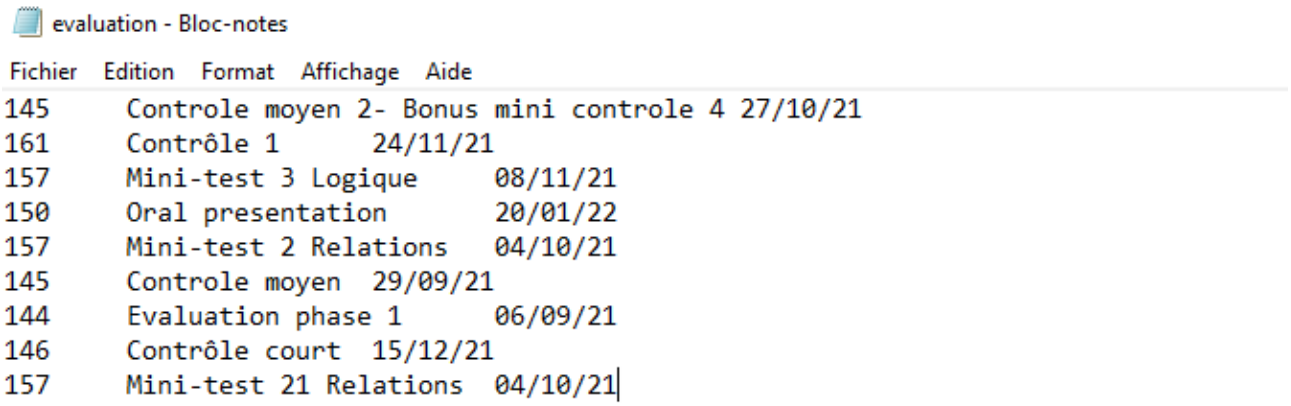
5.

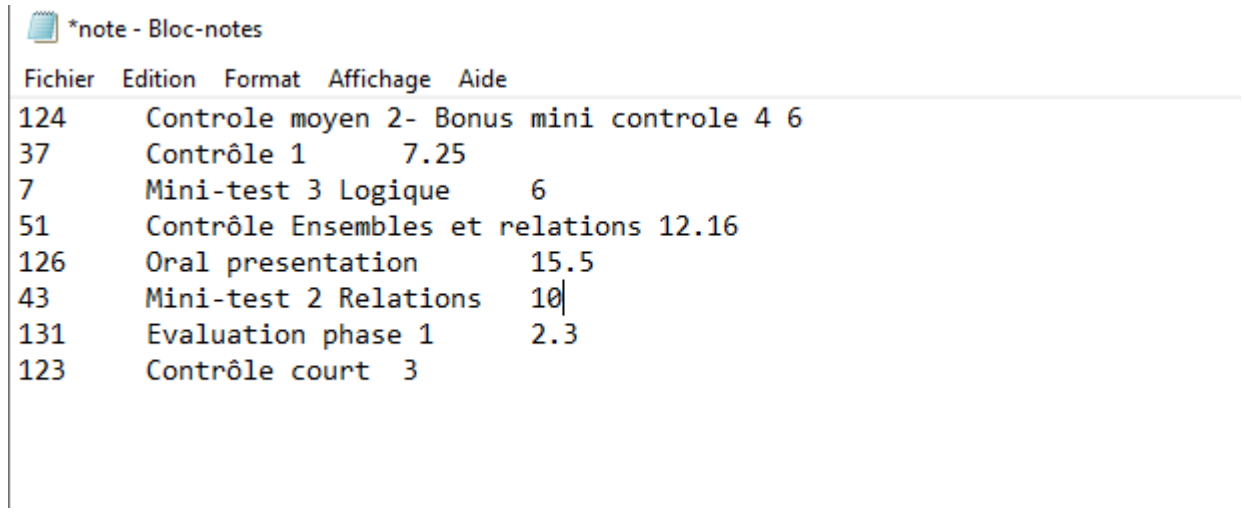
Le script généré par l'AGL est beaucoup plus complet que le mien, que ce soit pour les clé primaires/secondaires au niveau des contraintes , au niveau du type des clé, la suppression des lignes nuls , etc.

2.3 Peuplement des tables et requêtes

1. J'ai fais par morceau le script de peuplement en m'inspirant du fichier csv et je l'ai adapter a ma bdd pour vérifier son bon fonctionnement, pour cela j'ai réalisé des fichier .txt pour chacune des tables, que j'ai projeter à l'aide de la commande \copy <table> FROM <fichier>.

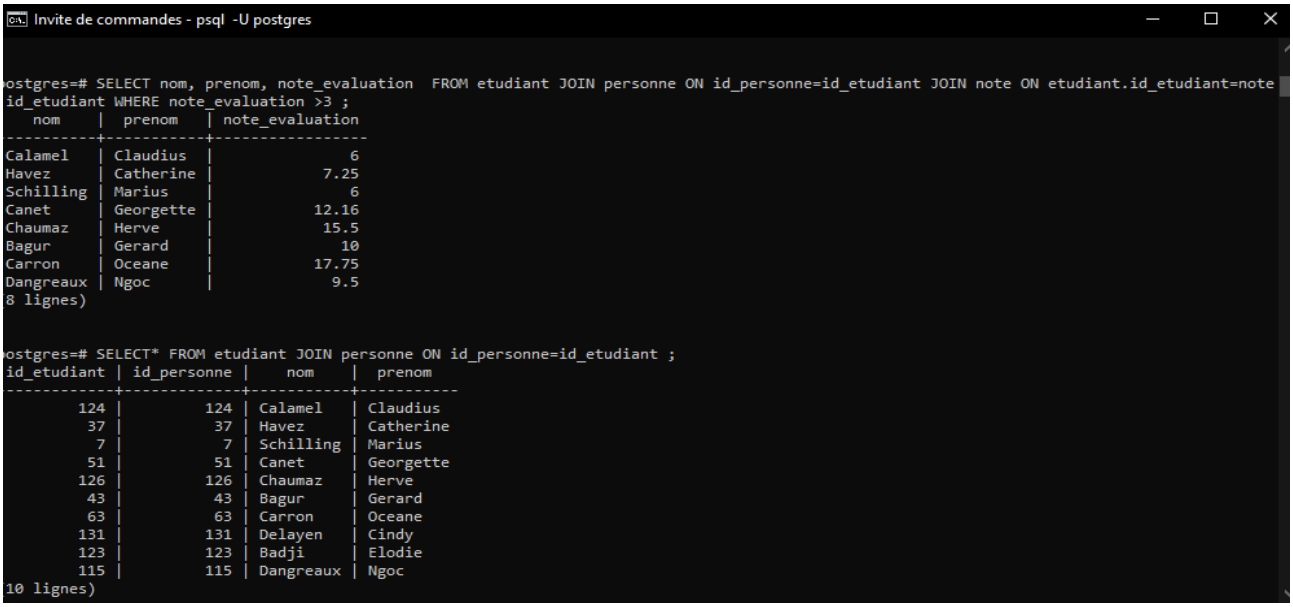
Exemple de mes fichier.txt :



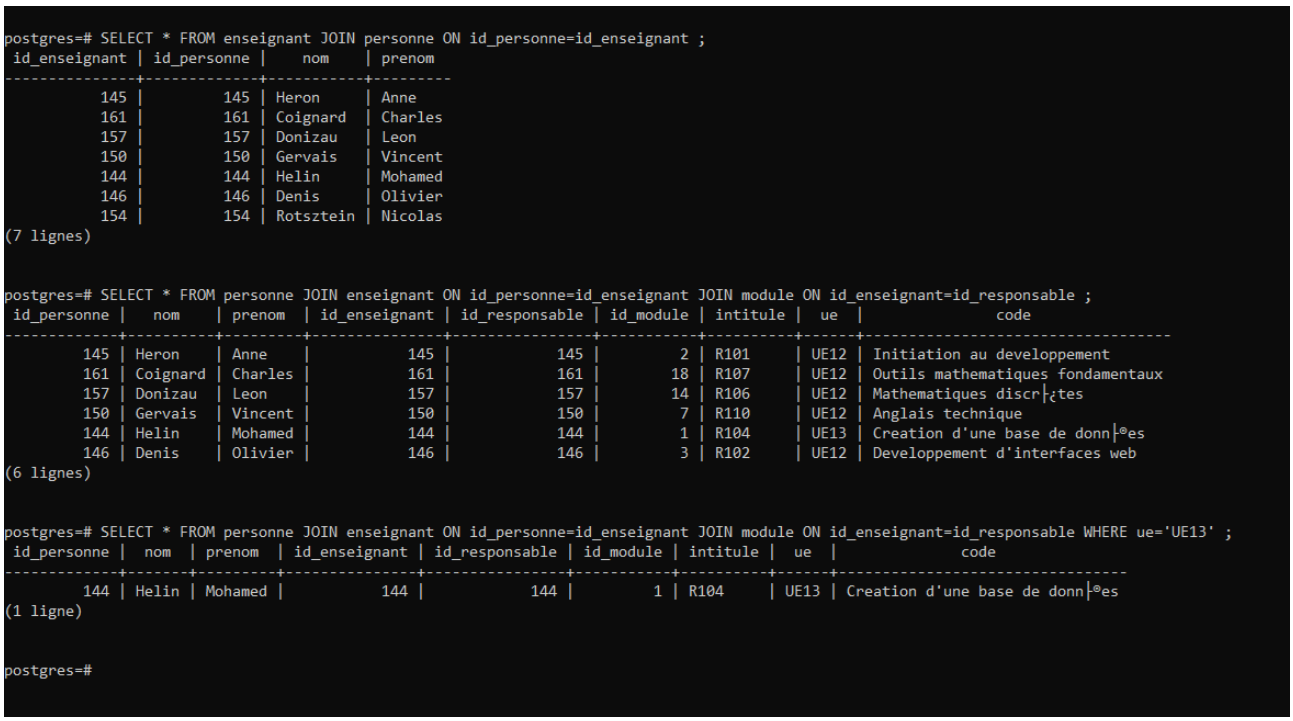


(il est possible que certaines infos ne soit pas visible dans mes fichiers de remplissage, car j'en ai ajouter certaines au fur et à mesure)

2.



J’ai effectuer une requête dans laquelle je demande à ma bdd les étudiant qui ont eu + de 3 à une évaluation, cela me permet de voir que ma bdd fais bien la différence entre les étudiants et les enseignants, et qu’elle sait aussi faire la différence entre les différente notes/évalutions.



Cette seconde requête me permet de voir que les enseignant son bien assigné à leur module, et que ma bdd sais faire la différence entre les différents responsable de module.

(ps, j’ai interverti certaine données lors du remplissage)