P-OO\_SpaceInvaders



Auteur : Thomas Nardou

Date : 28.08.2023 – 03.11.2023

Périodes : 60

Chef de projet : Mathieu Meylan, Aurélie Curchod, Xavier Carrel

Table des matières

[Chapitre 1 : Partie programmation 1](#_Toc147752783)

[Chapitre 2 : Partie UX/UI 1](#_Toc147752784)

[3.1 : Introduction 1](#_Toc147752785)

[3.2 : Analyse 1](#_Toc147752786)

[3.2.1 : Conception centré utilisateur 1](#_Toc147752787)

[3.2.2 : Palette Graphique 2](#_Toc147752788)

[3.2.3 : Éco-Conception : 3](#_Toc147752789)

[3.2.4 : Accessibilité : 3](#_Toc147752790)

[3.3 : Conception : 3](#_Toc147752791)

[3.3.1 : définition des écrans 3](#_Toc147752792)

[3.3.2 : Choix effectués : 4](#_Toc147752793)

[3.4 : Tests 5](#_Toc147752794)

[Chapitre 3 : Partie Base de données 5](#_Toc147752795)

[4.1 : Importer la base de données 5](#_Toc147752796)

[4.2 : Requêtes 6](#_Toc147752797)

[4.2.1 : Requête N°1 6](#_Toc147752798)

[4.2.2 : Requête N°2 6](#_Toc147752799)

[4.2.3 : Requête N°3 7](#_Toc147752800)

[4.2.4 : Requête N°4 7](#_Toc147752801)

[4.2.5 : Requête N°5 8](#_Toc147752802)

[4.2.6 : Requête N°6 8](#_Toc147752803)

[4.2.7 : Requête N°7 9](#_Toc147752804)

[4.2.8 : Requête N°8 9](#_Toc147752805)

[4.2.9 : Requête N°9 10](#_Toc147752806)

[4.2.10 : Requête N°10 10](#_Toc147752807)

[4.12 : Création des types d’utilisateur et utilisateur 11](#_Toc147752808)

[4.13 : Assignation des permissions 12](#_Toc147752809)

[4.13.1 : Rôle "r\_admin" 12](#_Toc147752810)

[4.13.2 : Rôle "r\_player" 12](#_Toc147752811)

[4.13.3 : Rôle "r\_shopKeeper" 13](#_Toc147752812)

[4.14 : exporter / importer la base de données 13](#_Toc147752813)

[4.15 : Création des index 14](#_Toc147752814)

# Chapitre 1 : Partie programmation

## : Introduction

Dans le cadre du Projet P-OO il a été demandé de réaliser une réplique du célèbre jeu "Space Invaders". Le jeu doit contenir au minimum :

* Un vaisseau avec possibilité de tir et déplacement
* 10 ennemis qui descendent sur l’axe vertical

# Chapitre 2 : Partie UX/UI

## 3.1 : Introduction

Dans la cadre du projet UX il nous a été demandé de réaliser une maquette pour le menu du célèbre jeu d’arcade « Space Invaders ». On nous a chargés de concevoir des maquettes pour chaque page du menu, tant pour le mode console que pour le mode graphique.

## 3.2 : Analyse

### 3.2.1 : Conception centré utilisateur

Pendant ce projet, j'ai choisi de créer deux persona, un homme et une femme, afin de mieux comprendre les types d'utilisateurs potentiels de mon application, en prenant en compte les besoins et les préférences de chacun des sexes. Afin de réaliser cette partie j’ai dû utiliser l’IA chatGPT afin de me donner :

* Les informations personnelles de mes persona
* Leurs biographies
* Leurs envies
* Leurs frustrations

Lors de la réalisation de mes maquettes j’ai essayé de respecter les envies de mes persona le plus possible afin d’attirer un publique plus large. Voici mes persona :

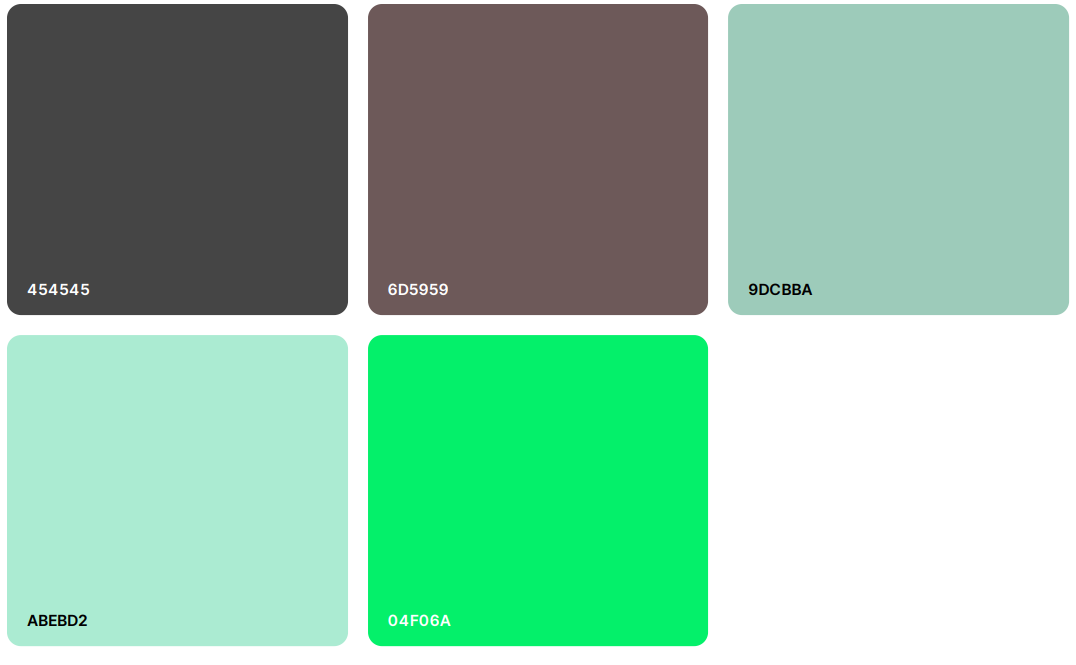




Vous trouverez aussi les persona en pièce jointe ou en appuyant sur [ce lien](Personna.pdf)

### 3.2.2 : Palette Graphique

Lors de la réalisation de mes maquettes j’ai dû choisir une palette de couleur afin d’avoir des pages plus jolies afin de réaliser cette partie je suis allé sur le site [coolors.co](https://coolors.co/) qui est un site qui génère des palettes de couleur et après avoir générer un certain nombre de palette je suis arrivé à ce résultat :



**678479**

J’ai décidé aussi de rajouter encore une nuance de vert (#678479) car je trouvais qu’il manquait des couleurs sombres donc j’ai décidé de la rajouter pour ajouter plus de contraste dans maquette tout en gardant une cohérence entre les couleurs. Si je me suis arrêté sur ces couleur c’est pour ces deux raisons :

1. Le vert et ses nuances peuvent faire rappeler les aliens et aussi le space Invaders original où le joueur est vert
2. Le noir et sa nuance peuvent faire rappeler l’espace c’est donc pour cela que j’ai choisi

### 3.2.3 : Éco-Conception :

Lors de conception des maquettes j’ai porté attention sur l’éco conception de mes maquettes pour pouvoir consommer le moins d’énergie possible notamment en choisissant un thème principal plutôt sombre qui sont les couleurs les moins énergivores de plus, j'ai décidé d’opter pour seulement deux polices d'écriture afin de limiter l'utilisation de ressources. Lors de la réalisation de l’application j’ai essayé de limiter le plus possible les transaction faite à la base de donnée hébergé sur un serveur

### 3.2.4 : Accessibilité :

Pour cette partie je porté attention aux personnes atteint de daltonisme lors du choix de la palette de couleur. Afin de garantir que ma palette de couleurs est accessible, j'ai consulté des personnes atteintes de daltonisme pour recueillir leurs impressions sur les couleurs qu'elles perçoivent. De plus, j'ai également utilisé un simulateur de daltonisme comme [color-blindness.com](https://www.color-blindness.com/coblis-color-blindness-simulator/) pour visualiser comment les personnes atteintes de différents types de daltonisme percevraient les couleurs de ma palette.

J’ai aussi porté attention aux personnes ayant des troubles de vue pour ce faire j’ai consulté des personnes atteint de problèmes de vue en leur demandant s'ils pouvaient clairement lire ce qui était écrit.

## 3.3 : Conception :

### 3.3.1 : définition des écrans

Lors de la réalisation de mes maquettes et de mon application j’ai décidé de prendre ces dimensions :

* Console : 759 x 638 si J'ai opté pour ces dimensions parce que les options en mode console sont assez restreintes. Par conséquent, j'ai choisi de réduire la taille de la page afin de réduire la sensation de vide qui peut survenir lorsque la page est trop grande par rapport à son contenu.
* Graphique : pleine écran si j’ai opté pour le plein écran c’est parce nous avons plus de choix pour le design du menu et par conséquent nous pouvons agrandir la résolution la deuxième raison est que généralement les jeux se joue en pleine écran pour avoir une meilleure immersion

### 3.3.2 : Choix effectués :

Voici l’une de mes maquettes que je vais utiliser pour illustrer mes choix :



Sur ce menu j’ai décidé de représenter l’espace avec différentes planètes dont deux qui ont déjà été envahi par des « invaders ». J’ai choisi d’utilisé cette typographie car c’est celle qui m’a le plus pensé au futur/espace. Pour les boutons j’ai décidé de mettre le texte en vert sur un fond brun pour bien pouvoir faire ressortir le vert. Pour tout ce qui est pour les boutons « Quitter » / « Retour » j’ai décidé de mettre le texte toujours en vert mais sur fond noir car cela permet de dire implicitement que ces boutons n’ont pas la même fonction que les autres.

## 3.4 : Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom : | Description : | Étape de test : | Données de test : | Sortie : | Condition : |
| Mode de jeu | Savoir si l’utilisateur peu choisir son mode de jeu | 1. Appuyer sur « jouer » dans la page d’accueil |  | La page de sélection de mode de jeu apparait |  |
| Commencer une partie en Local | Savoir si l’utilisateur va pour commencer une partie en Local | 1. Appuyer sur « Local » dans la page du mode de jeu |  | La partie se lance | Avoir déjà choisi l’option « jouer » dans la page d’accueil |
| Commencer une partie en multijoueur | Savoir si l’utilisateur va pour commencer une partie en ligne | 1. Appuyer sur « en ligne » dans la page du mode de jeu | - | La partie se lance | Avoir déjà choisi l’option « jouer » dans la page d’accueil |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Chapitre 3 : Partie Base de données

Pour cette partie on nous a demandé d’exécuter des requêtes SQL dont une qui sert un importer un dump d’une base de données existante depuis un invite de commande et les autres qui servent à aller chercher des informations qui se trouvent dans la base de données.

## 4.1 : Importer la base de données

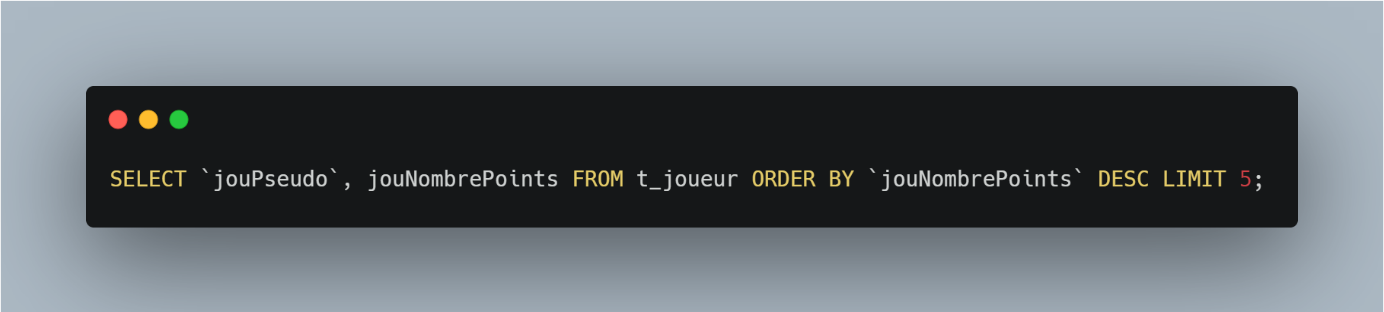
Pour cette partie voici la commande à utiliser pour pouvoir importer la base de données



Sur cette commande on peut voir un "**docker exec -i cc94b…**" cette partie signifie que l’on va exécuter la commande dans un conteneur docker qui possède l’id "**cc94b…** " la seconde partie va être la commande que l’on va exécuter dans le conteneur dans notre cas la commande va lancer MYSQL avec l’username (-u) "**root** " et le mot de passe (-p) "**root** " et la dernière partie signifie que le conteneur va exécuter le script SQL qui se trouve à l’emplacement indiqué.

## 4.2 : Requêtes

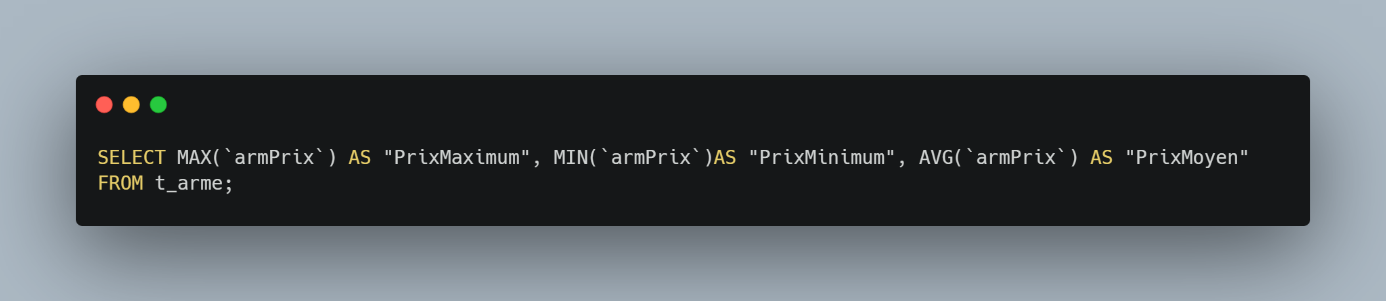
### 4.2.1 : Requête N°1

Voici la requête que qu’il faut utiliser pour répondre à la demande du cahier des charges : 

Pour cette requête il est nécessaire de sélectionner le pseudo des jours et les leurs nombres de points comme demander dans la requête dans la table qui leur correspond grâce au "**FROM**". Le "**ORDER BY**" va permettre d’ordonner les résultats (par défaut : par ordre croissant) le "**DESC**" quant à lui va permettre d’ordonner par ordre décroissant et pour finir le "**LIMIT 5**" va permettre de sélection seulement les 5 premier résultats.

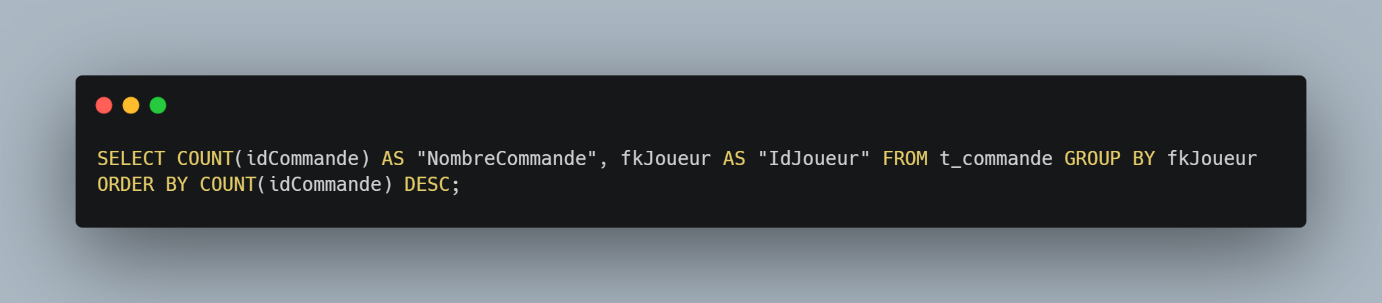
### 4.2.2 : Requête N°2

Voici la requête :



Sur cette requête il est indispensable de sélectionner le prix le plus haut, le plus bas et Moyen des armes grâce au fonctions **MAX(), MIN(), AVG()** il est essentiel renommer les colonnes par les noms imposés grâce au alias (**AS**) depuis la table **t\_arme**.

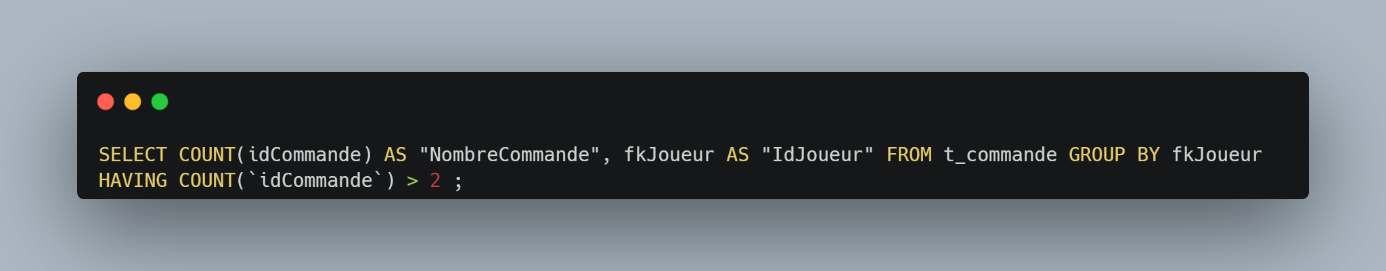
### 4.2.3 : Requête N°3

Voici la requête utilisée : 

Sur cette commande il est nécessaire de sélectionner le nombre de commande que chaque joueur à fait grâce à la fonction **COUNT()** et de sélectionner l’id des joueurs les colonnes. L’étape d’après consiste à renommer les colonnes avec des alias (**AS**) ensuite il est indispensable de regrouper le tout par joueur (**GROUP BY**). L’étape suivante consiste à ordonner le tout par le nombre de commande du plus grand au plus petit grâce à la clause "**DESC** ".

### 4.2.4 : Requête N°4

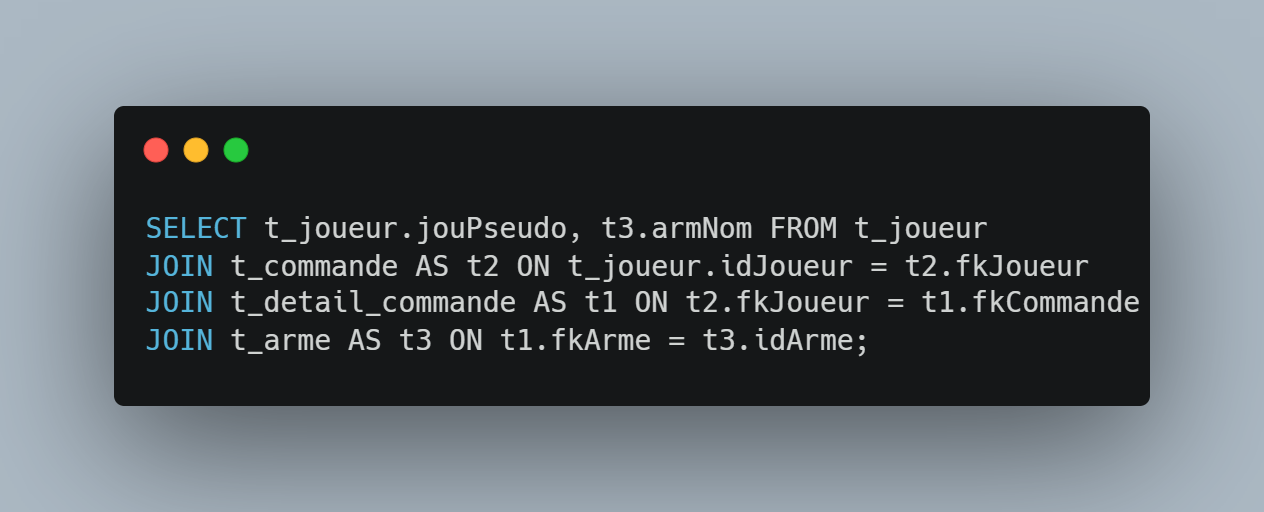
Voici la requête :



Sur cette requête les début et pareil que pour la requête N°3 mais en plus de regrouper le tout par joueur il faut regrouper par les joueurs dont le nombre de commande et plus grand que deux s’il est possible de réaliser cette partie c’est grâce à la clause "**HAVING**" qui va mettre une condition sur chaque groupement

### 4.2.5 : Requête N°5

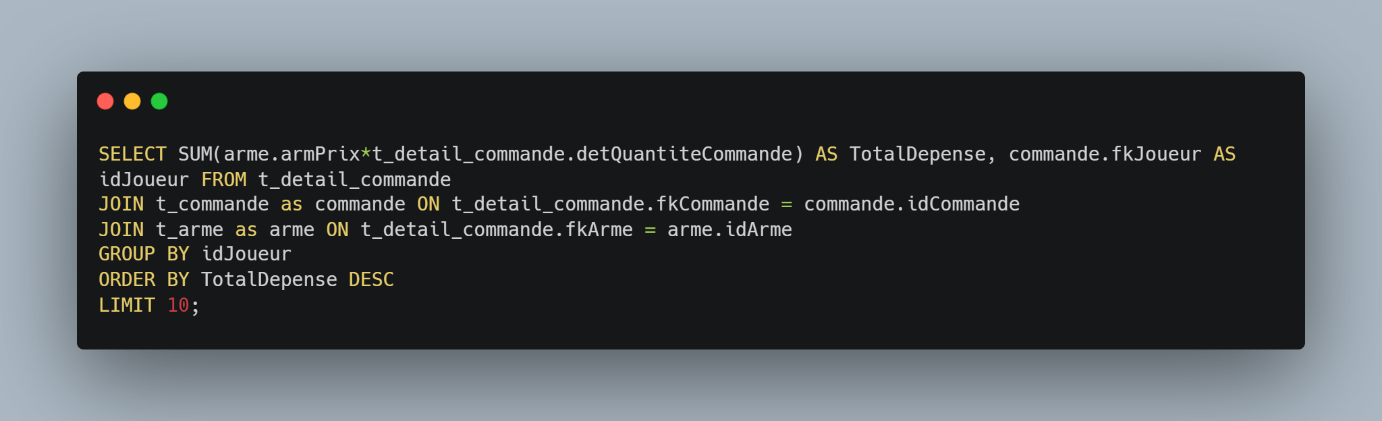
Pour cette question voici la requête à utiliser :



Pour cette requête il est demandé de sélectionner le pseudo des joueurs et le nom des armes de leurs commande ensuite il est nécessaire de faire des jointures entre les différentes tables pour pouvoir utiliser leurs données, il a été décidé de faire un inner join (**JOIN**) car il est demandé seulement les joueurs qui ont passé une commande.

### 4.2.6 : Requête N°6

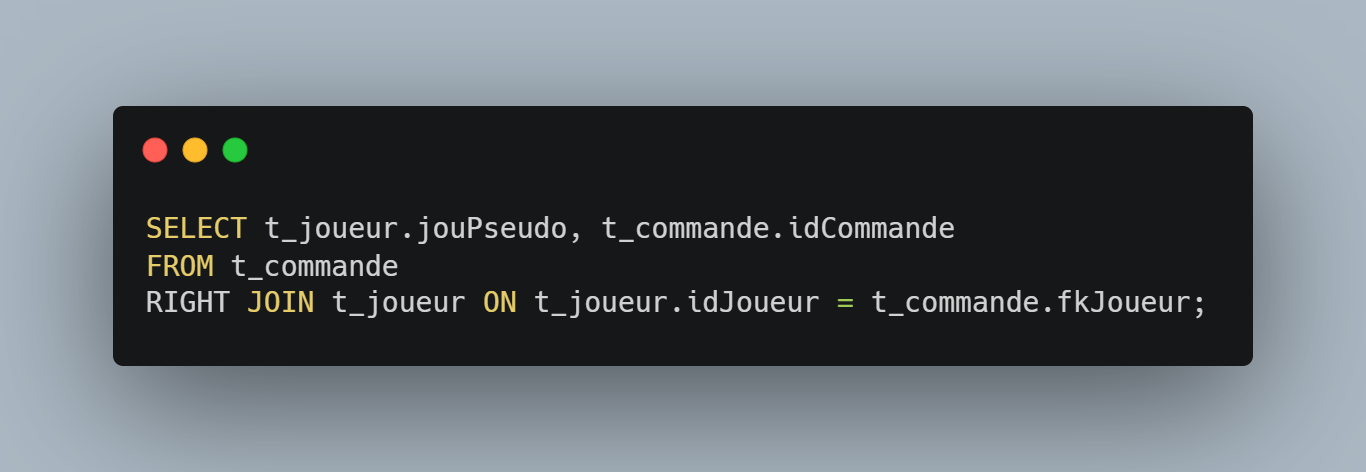
Voici ma requête :



Cette requête sélectionne le total dépensé un calculant la somme du prix de l’arme \* quantité d’arme dans la commande ensuite elle nomme la colonne "**TotalDepense** ". L’étape d’après la requête sélectionne les joueurs de la table commande et renomme la colonne en "**idJoueur** ". La suite consiste à joindre les tables pour pouvoir utiliser leurs données ensuite il est nécessaire de faire un **GROUP BY** pour connaitre le total par joueur ensuite il faut ordonner les résultats par ordre décroissant en fonction du total dépensé et pour finir la requête prend que les 10 premiers résultats

### 4.2.7 : Requête N°7

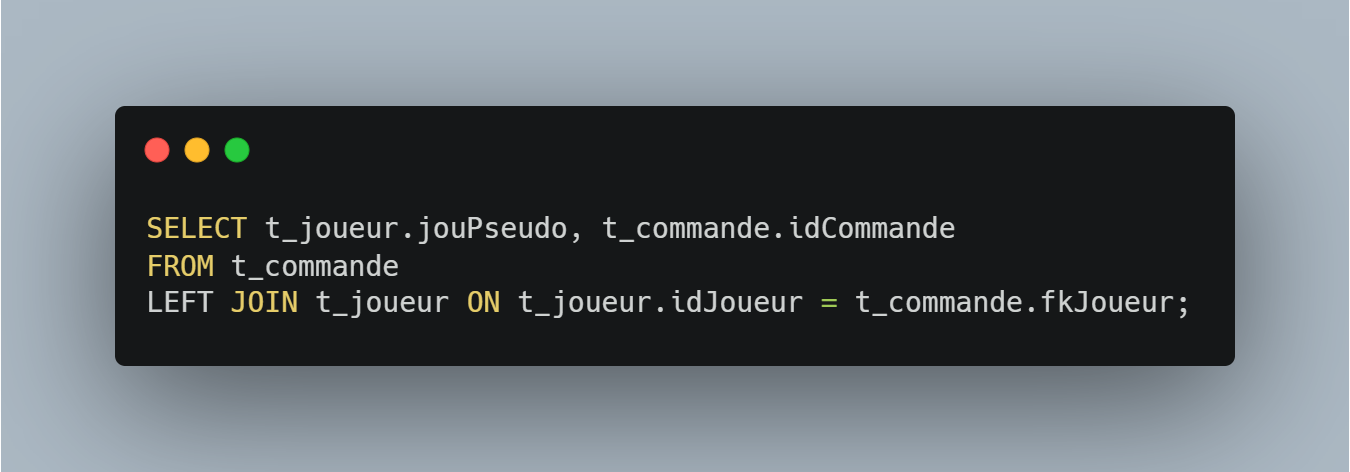
Voici la requête trouvée :



Cette requête sélectionne le pseudo des joueurs ainsi que l’id de leurs commandes ensuite elle joint avec un "**RIGHT JOIN**" car il est aussi demandé de sélectionner les joueurs qui n’ont pas passé de commande

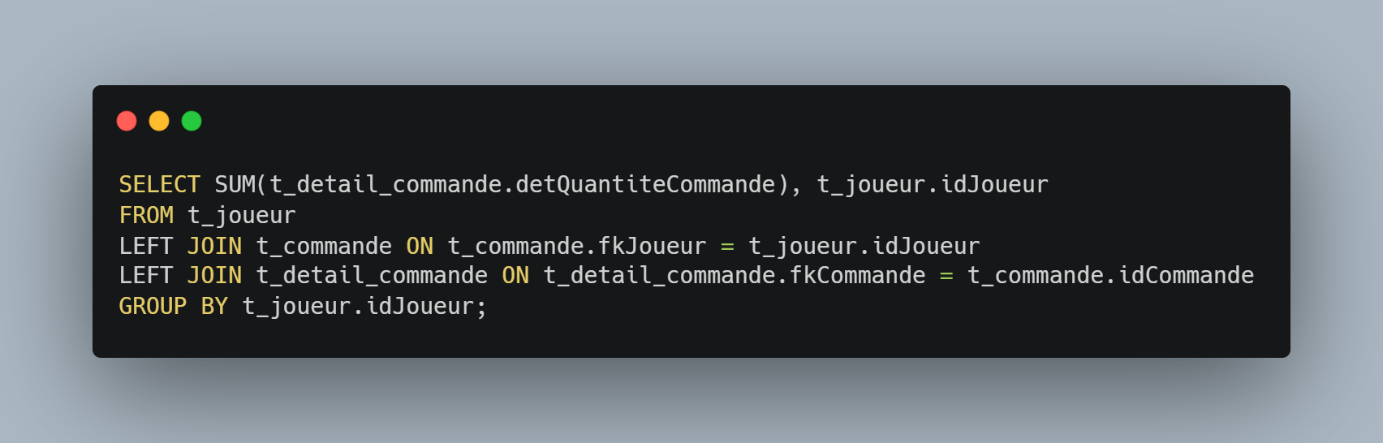
### 4.2.8 : Requête N°8

Voici la requête :



La requête est la même que dans la requête N°7 mais au lieu de faire un "**RIGHT JOIN**" elle fait un "**LEFT JOIN**" car il est demandé d’afficher les commandes même si elles existent ou pas

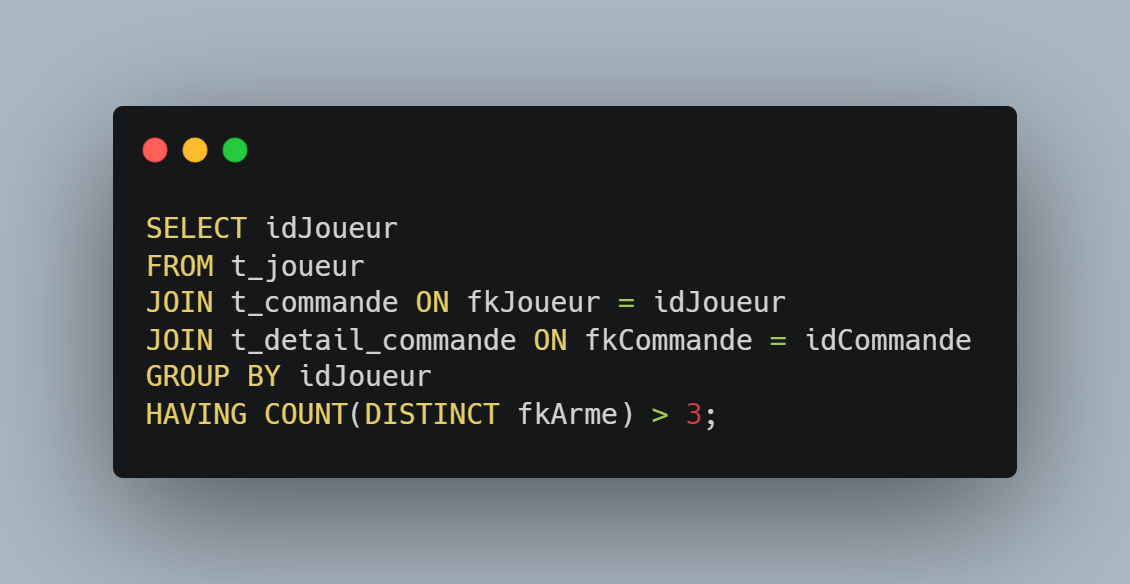
### 4.2.9 : Requête N°9

Voici ma requête : 

Cette requête sélectionne le nombre total de commande qu’un joueur a passé ensuite elle joint les tables entre elles avec un "**LEFT JOIN**" car il est aussi demandé de sélectionner les joueurs qui n’ont pas passé de commande ensuite la requête groupe le tout par joueur.

### 4.2.10 : Requête N°10

Voici la requête :



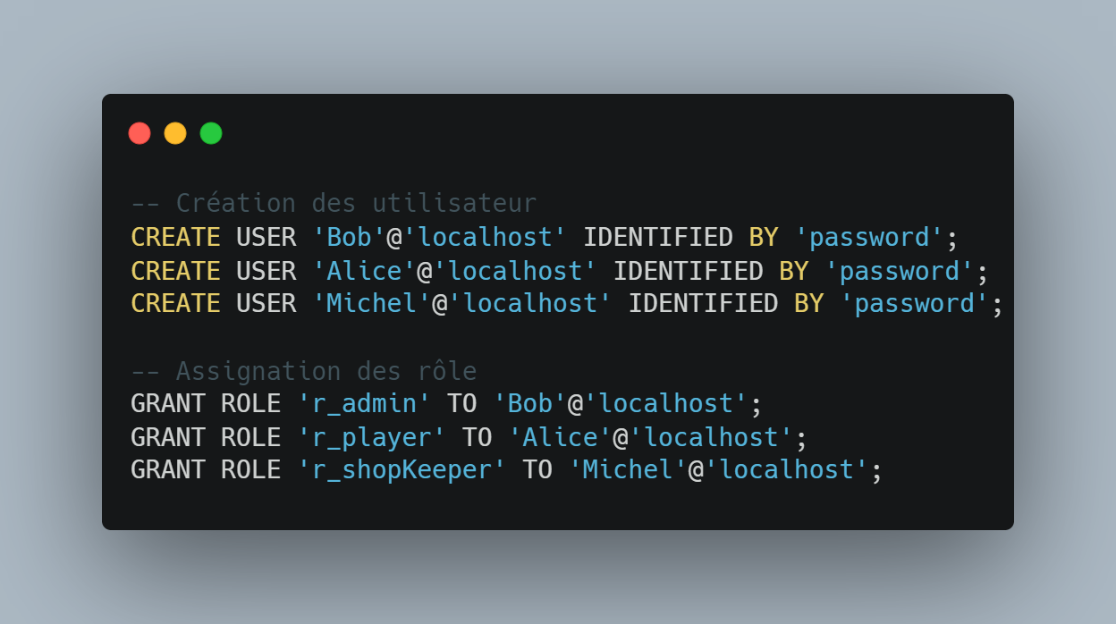
Cette requête sélectionne l’id des joueurs ensuite elle joint les tables entres elles avec un "**JOIN**" ("**INNER JOIN**") car nous voulons seulement les joueur qui ont passé commande ensuite elle trie le tout par joueur mais seulement si le nombre d’arme différentes (**COUNT(DISTINCT fkArme)**) et plus grand que trois

## 4.12 : Création des types d’utilisateur et utilisateur

Pour cette partie on nous a demandé de donner des permissions a des types d’utilisateurs mais pour ce faire nous devons d’abord créer ces types d’utilisateur et dans ce but il est nécessaire d’utilisé ces trois requêtes :



Il a été nécessaire de créer un utilisateur pas type voici les requêtes que qu’il faut utiliser pour les créer et assigné les utilisateurs à leurs rôles :

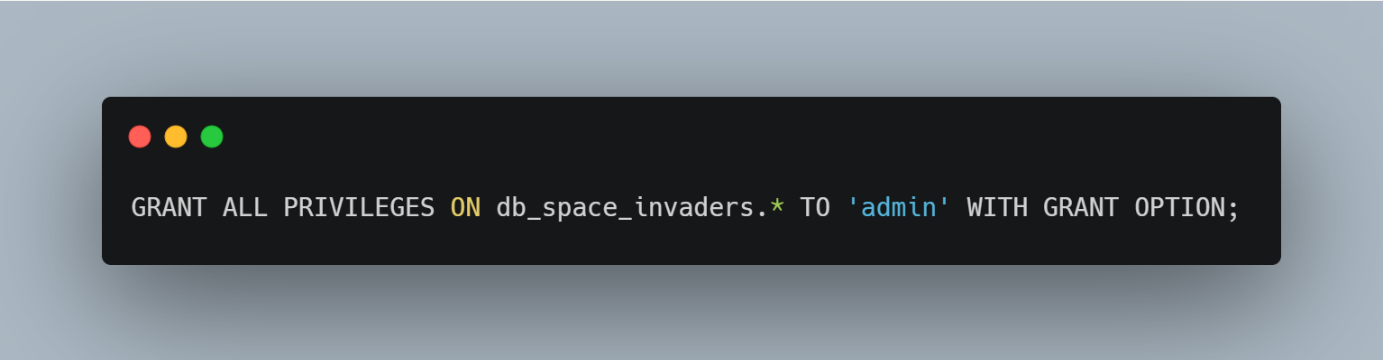


Dans la première partie la requête crée les utilisateurs dans l’host "*localhost*" et leurs donnent le mot de passe "*password*" grâce au "**IDENTIFIED BY**". Ensuite elle assigne les utilisateurs au rôle.

## 4.13 : Assignation des permissions

### 4.13.1 : Rôle "r\_admin"

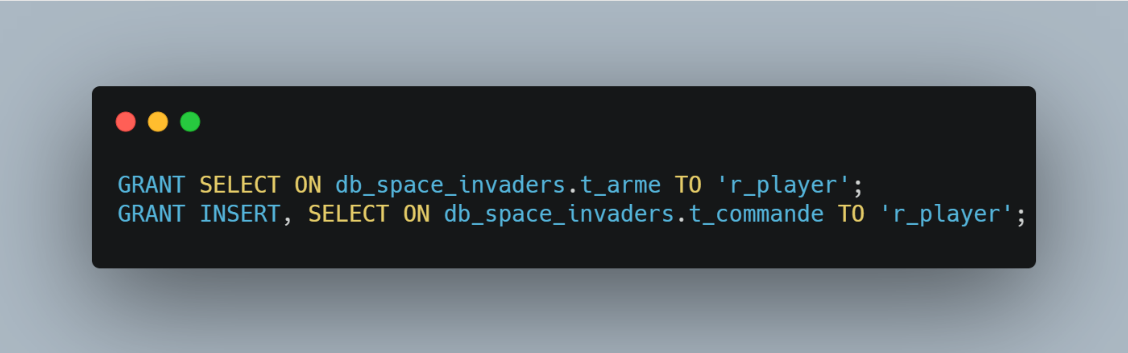
Voici la requête qu’il faut utiliser pour donner les permissions d’écriture, lecture, suppression, mis à jour, gestion des permissions pour ce rôle :



Sur cette requête donne toutes les permissions au rôle (select, insert, …) grâce au "**ALL PRIVILEGES**" sur la base de données **db\_space\_invaders** avec la permission de pouvoir gérer les utilisateurs

### 4.13.2 : Rôle "r\_player"

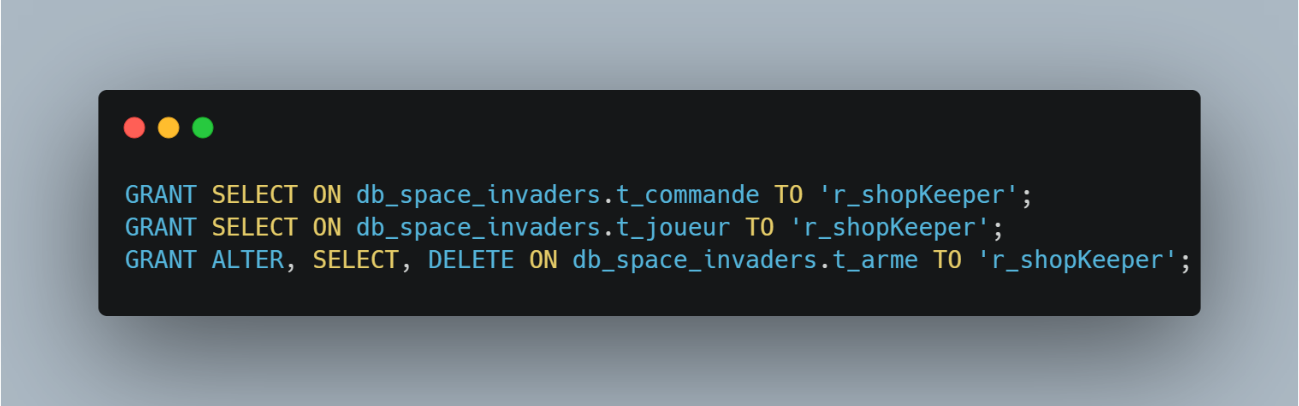
Voici la requête qui donner les permissions de Lire les informations des armes, créer une commande et lire toutes les commandes :



La première requête donne uniquement la permission de sélectionner les données de la table **t\_arme** pour pouvoir sélectionner les informations d’une arme. Ensuite la requête donne les permissions d’insérer et de sélectionner uniquement sur la table **t\_commande** comme demandé afin de créer des command et de voir les commandes en cours.

### 4.13.3 : Rôle "r\_shopKeeper"

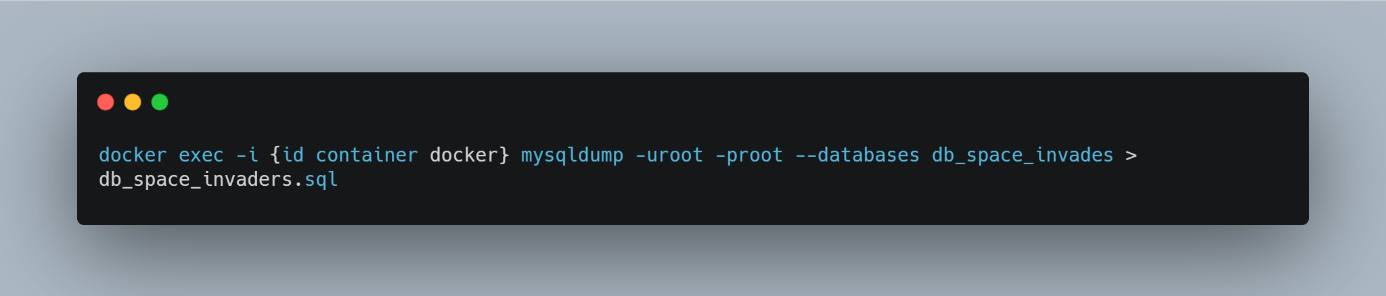
Voici la requête utilisée pour donner les permissions de Lire les informations sur tous les joueurs, mettre à jour, lire et supprimer des armes et lire toutes les commandes :



En premier lieu la requête donne les permissions de lire les informations sur les tables **t\_commande** (pour pouvoir lire toutes les commandes) et sur la table **t\_joueur** (pour pouvoir lire toutes les informations d’un joueur), ensuite je lui donne les permissions de modifier, lire et supprimer sur la table **t\_armes** pour pouvoir gérer la vente des armes.

## 4.14 : exporter / importer la base de données

Pour exporter un dump de la base de données voici la commande nécessaire afin d’exporter une base de données dans l’invite de commande windows :



Dans cette commande ouvre un terminale docker grâce au "**docker exec -i {id container docker}**" ensuite elle exécute la commande "**mysqldump"** qui va exporter ma base de données avec l’utilisateur root. Le "**--databases"** va permettre d’insérer la création et l’utilisation de la base de données dans le script car par défaut ils n’y sont pas ensuite la requête indique la base de données à exporter. L’étape d’après consiste à donner le chemin d’accès où le fichier va se créer le signe "**>**" va permettre d’importer VERS l’endroit indiqué.

Pour restaurer le dump de la base de données il est nécessaire d’exécuter cette commande toujours dans le terminal Windows :



Cette commande ouvre un terminale docker grâce au "**docker exec -i cc94…**" ensuite elle exécute la commande "**mysql**" qui va restaurer la base de données avec l’utilisateur root ensuite il est primordial d’indiquer le chemin d’accès du fichier .sql le signe "**<**" est important car il veut dire que cela va importer le fichier.

## 4.15 : Création des index

1. Car MySql génère automatiquement des index pour les clés primaires et étrangères afin de réaliser des jointures. Il génère aussi des index pour les champs uniques afin d’augmenter la rapidité des requêtes de type "**SELECT"**.
2. Un index va permettre d’optimiser les requêtes de type "**SELECT**" pour cela il va créer ce que l’on appelle un "*B-Tree*" qui un arbre auto-équilibré dans lequel chaque nœud contient les données sous une forme triée. L’inconvénient c’est que ça prend beaucoup de mémoire vive de plus les requête du type "**INSERT", "ALTER"** et **"DELETE"** sont ralenties car en plus de mettre à jour la table elles doivent mettre à jour l’index.
3. Il serait judicieux de créer un index composite pour les champs "**armNom**", "**armDescription**" et "**armPrix**" dans la table **t\_arme**, étant donné que ces valeurs ne sont pas susceptibles de changer fréquemment, et il est peu probable qu'il y ait fréquemment de nouvelles armes ajoutées.