**Query Builder Commands**

**1. Sélectionner des résultats**

**get() method :**

>>      $articles = DB::table('articles')->get();

**where() method :**

>>        $articles = DB::table('articles')->where('active','=','1')->get();

**first() method :**

>>       $articles = DB::table('articles')->first();

**find() method :** (it takes a number as a parameter, and it searches by ID)

>>         $articles = DB::table('articles')->find(1);

**2. Rangement et groupement**

**orderBy() method :** (it takes 2 parameters, the 1st parameter is the field you want to sort your items with and the 2nd parameter is either **ASC** or **DESC**)

>>       $articles = DB::table('articles')->orderBy('prix','ASC')->get();

OR :

        $articles = DB::table('articles')->orderBy('prix','DESC')->get();

**groupBy() method :** (it takes a single parameter, BUT it must be used after an operation method like **SUM(),MAX()..**)

>> $articles = DB::table('articles')->sum('prix')->groupBy('id')->get();

**chunk() method :** (we use it to split our data to chunks following the number we specified and while using a query builder, we must add the orderBy() methode, otherways it’s will throw an error, However, its better to use the Eloquent Model directly like which will not enforce you to add an order by clause manually :)

USING Query Builder :

>> DB::table('articles')->orderBy('id',"DESC")->chunk(1, function ($articles) {

            foreach ($articles as $article) {

                return $article;

            }

        });

USING Elequont Model :

>>        Article::chunk(10, function($product){

            ...});

**latest() and oldest() methods :**

>>        $articles = DB::table('articles')->latest('created\_at')->first();

>>       $articles = DB::table('articles')->oldest('created\_at')first();

**inRandomOrder() methods :**

        $articles = DB::table('articles')->inRandomOrder('id')->get();

**3. Les jointures**

**join() method :**

DB::table('articles')>join('sous\_familles','articles.sous\_famille\_id','=','sous\_familles.id')->get();

***Left join & right join methods :***

DB::table('articles')->leftJoin('sous\_familles','articles.sous\_famille\_id','=','sous\_familles.id')->get();

&&

DB::table('articles')->rightJoin('sous\_familles','articles.sous\_famille\_id','=','sous\_familles.id')->get();

**4. Insert, Update and Delete :**

**insertGetId() method : (it takes an array as a parameter, which contains our table’s fields and the value u want to insert in key => value pairs, and it’s returns the modifications/actions number.**

        $articles = DB::table('familles')->insert([

            "id" => 10,

            "famille"=> "testing query builder insert method",

            "active"=> 1,

            "photo\_famille"=> "test.png",

        ]);

        return $articles;

**update () method : u must use the where condition to specify the field u want to update other ways it’s will update every field and just like insert it takes an array of the table’s fields as a parameter on key=>value pairs**

        $articles = DB::table('familles')->whereId(10)->update([

            "id" => 5,

            "famille"=> "testing query builder update method",

            "active"=> 1,

            "photo\_famille"=> "test.png",

        ]);

        return $articles;

**updateOrInsert() method :(it does exactly the same as update, the only different is, in case the field u want to update didn’t exist, it’s will insert it as a new one)**

>>        $articles = DB::table('familles')->whereId(10)->**updateOrInsert**([

            "id" => 5,

            "famille"=> "testing query builder update method",

            "active"=> 1,

            "photo\_famille"=> "test.png",

        ]);

        return $articles;

incrementing() et decrementing() ; (increments the value of a field u specify with the pace you want, since the incsrement and descrement methods takes 1 main parameter and an optional 2nd parameter and it’s the pace)

>> $articles = DB::table('articles')->whereId(9)->increment('prix',2);

>> $articles = DB::table('articles')->whereId(9)->decrement('prix',2);

**5. Opérations numérique**

        $max\_price = DB::table('articles')->max('prix');

        $sum\_price = DB::table('articles')->sum('prix');

        $min\_price = DB::table('articles')->min('prix');

        $average\_price = DB::table('articles')->avg('prix');