CPSC 304: Milestone 4

Hunt: Showdown
Project Implementation
Github Repo

Victor Shen 73048563 e1w5d Ranyi (Jake) Tao 27268507 s1w7b Yun (Brina) Li 16840829 t9k4s

#### **Project Description:**

Our application represents a simplified model of the *Hunt: Showdown* equipment system, and also models locations and monster entities within the game. The database includes information about the different Hunters featured in the game, including the various traits that they can acquire. Each Hunter is expected to carry a firearm, and they may also choose to wield a tool and bring various consumables. Each firearm, tool and consumable all have different specifications and descriptions; firearms also require ammunition which may be of different types. Hunters may be located at different locations on the map, and at each location, there may be monsters that spawn. Names of the various locations, along with coordinates, are included in the databases. Information about the different monsters that spawn at each location are also modeled accordingly.

Our application allows users to view all data tables within the database including Hunters, Consumables, Firearms, etc. Additionally, users may insert tuples into the *Trait* table, delete tuples from the *Consumable* table, and update tuples of the *Tool* table based on user input. Additionally, we provided functionality to drop tables and reset the database utilizing our GUI to aid ease of use for presentation purposes. We also successfully implemented queries which allow for user input to demonstrate selection, projection, and joins. More complex queries involving aggregation with group by, aggregation with having, nested aggregation with group by, and division are hardcoded, but query based on meaningful inquiries a potential user might make. All backend implementations have been successfully linked with our frontend and as such, all functionality may be demoed via our GUI. We also took the extra step and made an effort to polish up the appearance of our application utilizing CSS.

Our final schema did not differ from the schema we previously submitted. However, minor changes were made to the SQL initialization statements we previously submitted in order to ensure compatibility with Oracle, and to provide more tuples which may be meaningfully queried. Additionally, since the previous milestone, we made changes to some of our queries (aggregation with having and nested aggregation with group by) in order to better reflect the requirements of the rubric, and a few very minor changes were made to our intended GUI elements as well as to be more user-friendly. We have attached a copy of our schema below, as well as screenshots to show all data present post-initialization. All operations and queries are also clearly documented below with references to where in the code they are implemented.

#### **Schema:**

Firearm(<u>firearmName: char(100)</u>, firearmDescription: text(8000), firearmCapacity: int, firearmRateOfFire: int, firearmHandling: int, firearmCost: int)

- firearmDescription must be NOT NULL
- firearmCapacity must be NOT NULL
- firearmRateOfFire must be NOT NULL
- firearmHandling must be NOT NULL
- firearmCost must be NOT NULL

Ammo(<u>firearmName: char(100)</u>, <u>ammoType: char(100)</u>, ammoDescription: text(8000), ammoDamage: int, ammoEffectiveRange: int, ammoVelocity: int, ammoCost: int)

- ammoDescription must be NOT NULL
- ammoDamage must be NOT NULL
- ammoEffectiveRange must be NOT NULL
- ammoVelocity must be NOT NULL
- ammoCost must be NOT NULL and DEFAULT 0
- ammoType is a PARTIAL KEY

Consumable(<u>consumableName: char(100)</u>, consumableDescription: text(8000), consumableCost: int)

- consumableDescription must be NOT NULL
- consumableCost must be NOT NULL

Syringe(consumableName: char(100), syringeHealing: int, syringeEffectDuration: int)

Explosive(consumableName: char(100), explosiveEffectiveRange: int, explosiveEffectiveRadius: int, explosiveDamage: int)

- explosiveEffectiveRange must be NOT NULL
- explosiveEffectiveRadius must be NOT NULL
- explosiveDamage must be NOT NULL

Tool(<u>toolName</u>: char(<u>100</u>), toolDescription: text(8000), toolMeleeDamage: int, toolHeavyMeleeDamage: int, toolCost: int)

- toolDescription must be NOT NULL
- toolCost must be NOT NULL

Trait(traitName: char(100), traitDescription: text(800), traitCost: int)

- traitDescription must be NOT NULL
- traitCost must be NOT NULL

Location(locationName: char(100), locationCoordinates: char(100))

- locationCoordinates must be NOT NULL UNIQUE and is a CANDIDATE KEY

Monster(<u>monsterName</u>: char(100), monsterDescription: text(8000), monsterType: char(100), monsterSize: char(100), monsterHealth: int)

- monsterDescription must be NOT NULL
- monsterType must be NOT NULL
- monsterSize must be NOT NULL
- monsterHealth must be NOT NULL

Hunter(<u>hunterName: char(100)</u>, hunterDescription: text(8000), hunterFunds: int, hunterLevel: int, hunterHealth: int, **locationName: char(100)**, **firearmName: char(100)**)

- hunterDescription must be NOT NULL
- hunterFunds must be NOT NULL and DEFAULT 0
- hunterLevel must be NOT NULL and DEFAULT 1
- hunterHealth must be NOT NULL
- locationName must be NOT NULL
- firearmName must be NOT NULL

Consumes(<u>hunterName: char(100)</u>, <u>consumableName: char(100)</u>)

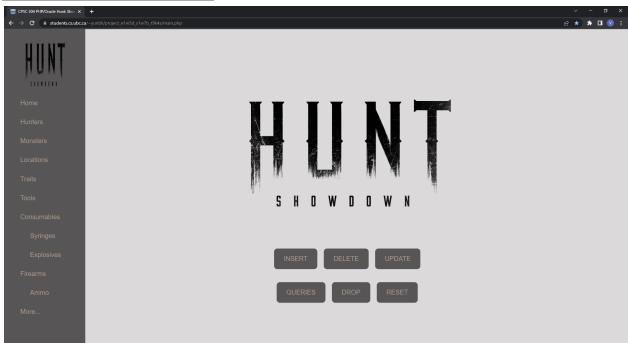
Wields(<u>hunterName: char(100</u>), <u>toolName: char(100</u>))

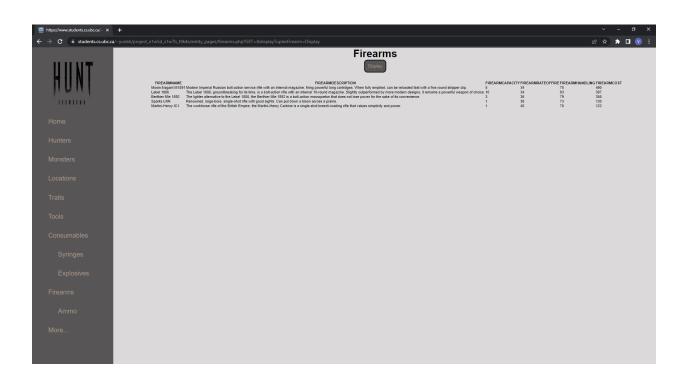
Acquires(<u>hunterName: char(100)</u>, <u>traitName: char(100)</u>)

Hunts(hunterName: char(100), monsterName: char(100))

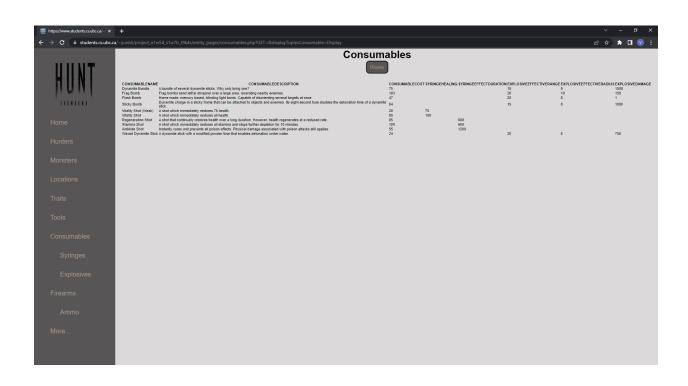
Spawns(<u>locationName: char(100)</u>, <u>monsterName: char(100)</u>)

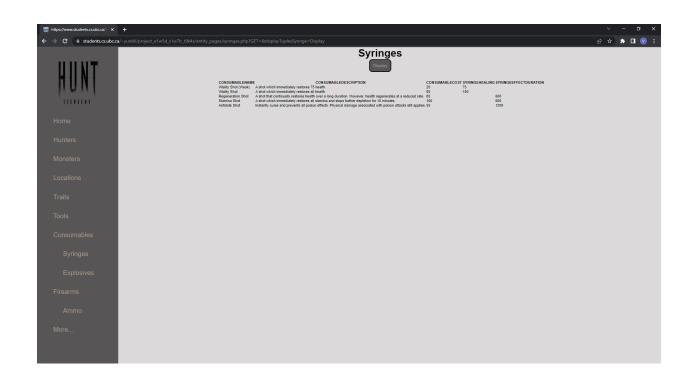
### **Post-Initialization Screenshots:**

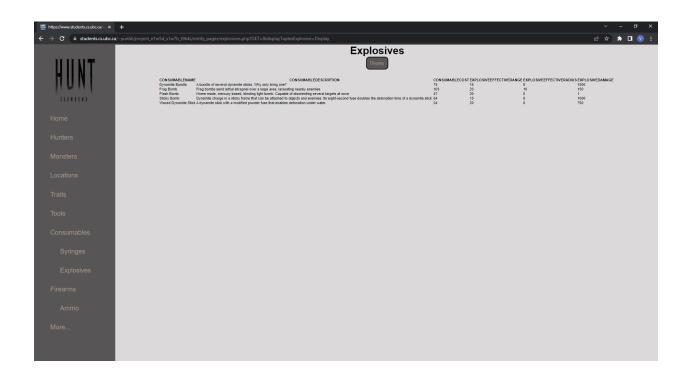


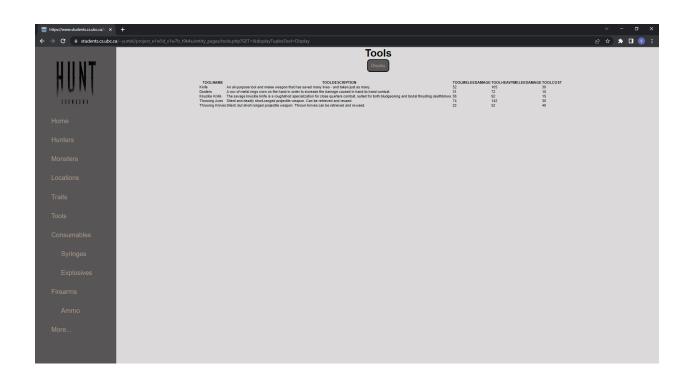






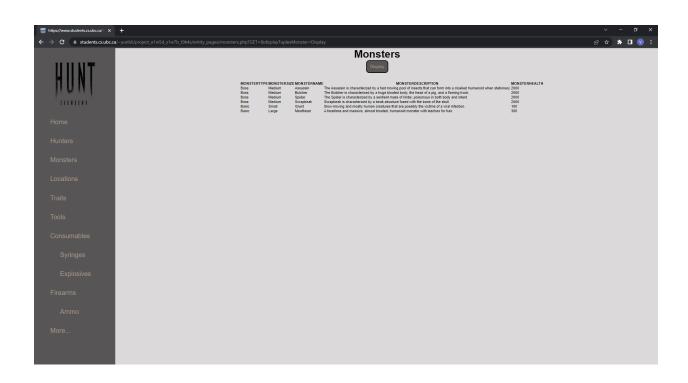


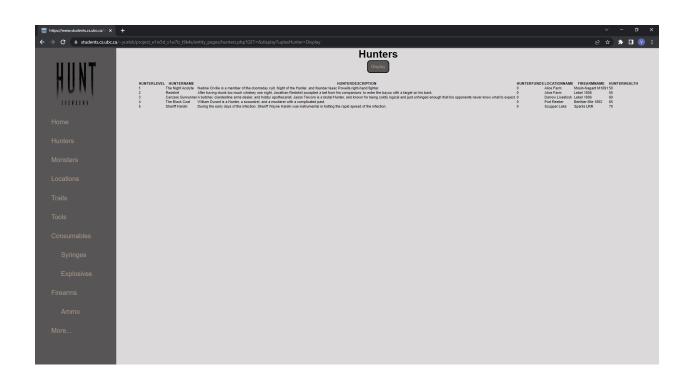


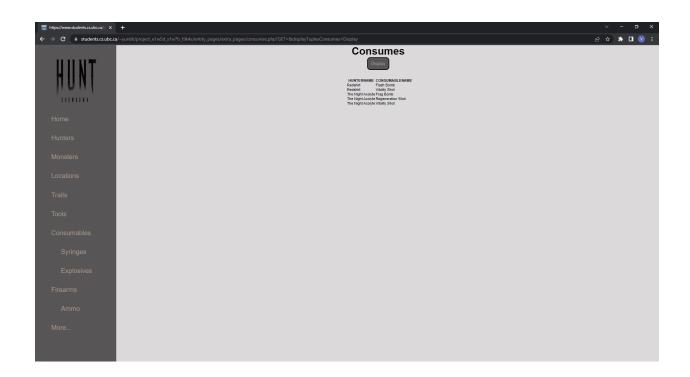


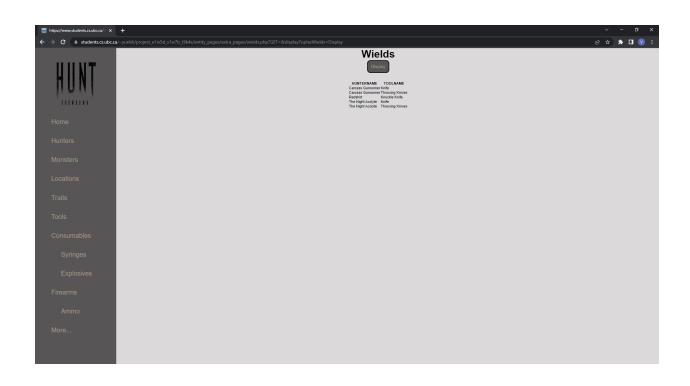


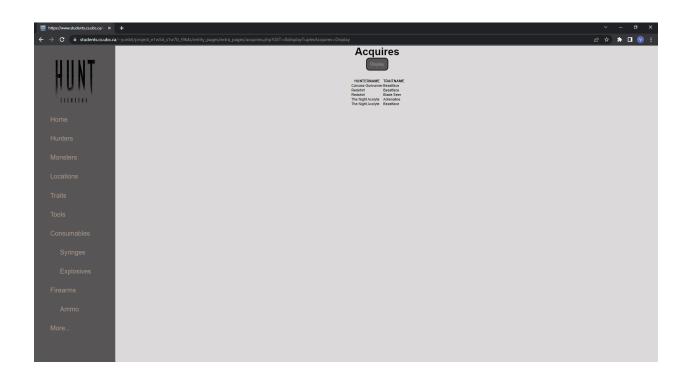


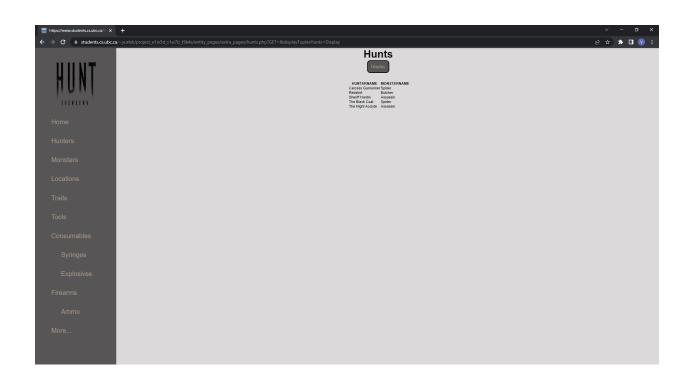






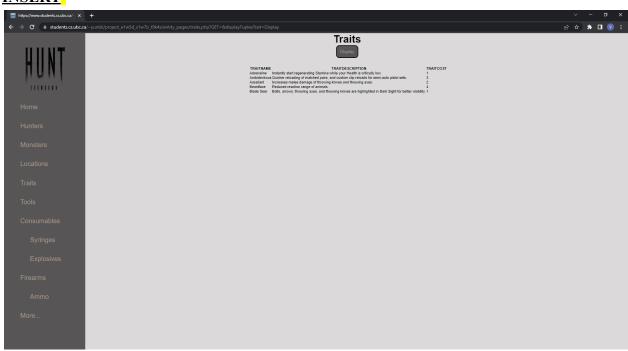


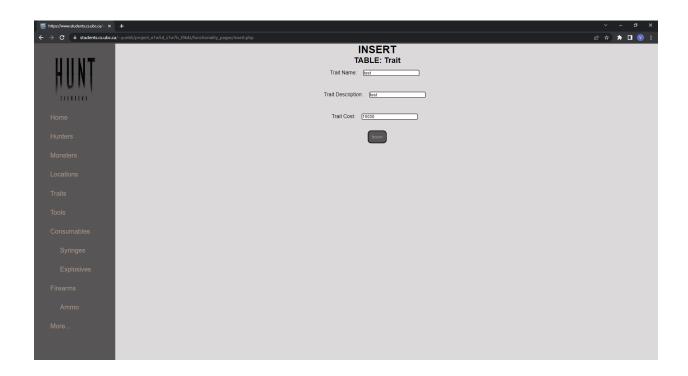






### INSERT<sup>1</sup>





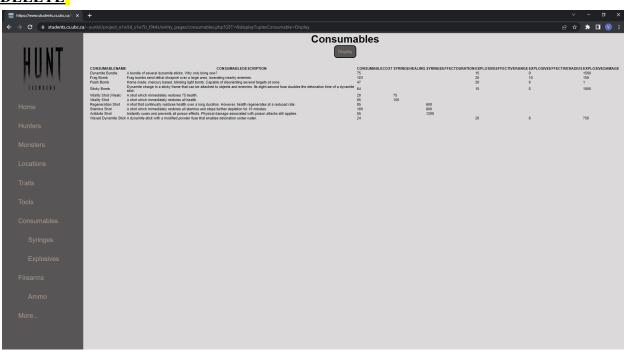
<sup>&</sup>lt;sup>1</sup> This operation satisfies the insert rubric item for our project

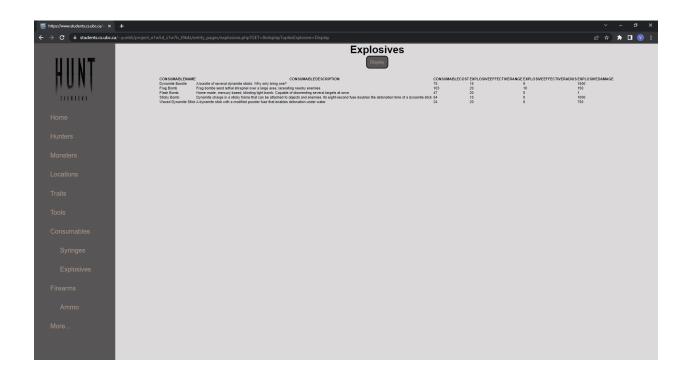


In this insert operation, the user is able to input values for all trait attributes for an insertion into the *Trait* table.

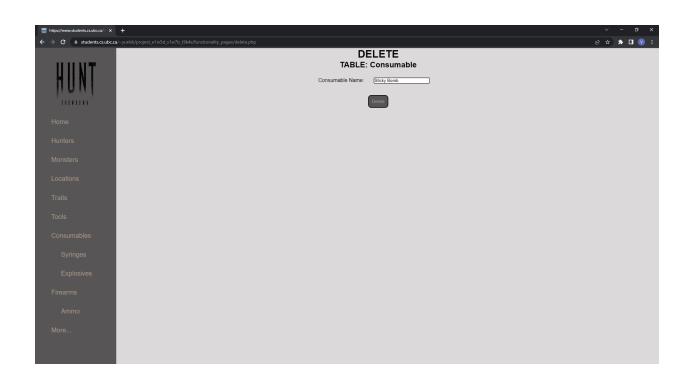
The backend code for this operation functionality may be found within the *handleInsertRequest()* function in ./backend\_functionality\handlers.php

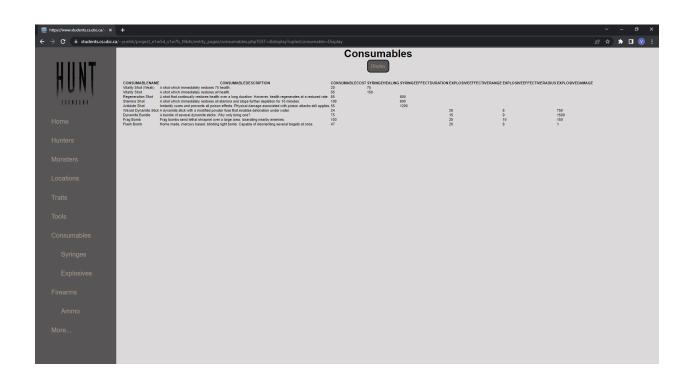
### **DELETE**<sup>2</sup>

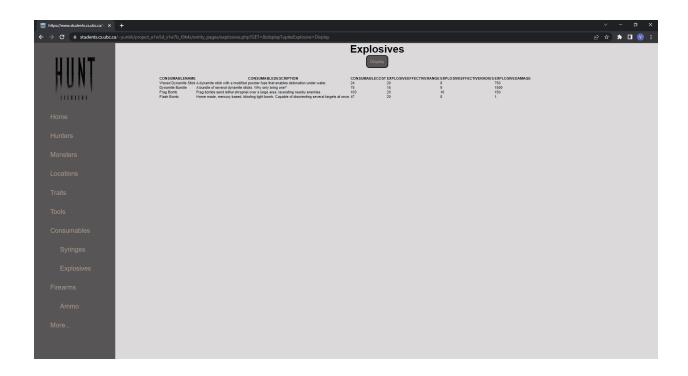




<sup>&</sup>lt;sup>2</sup> This operation satisfies the delete rubric item for our project



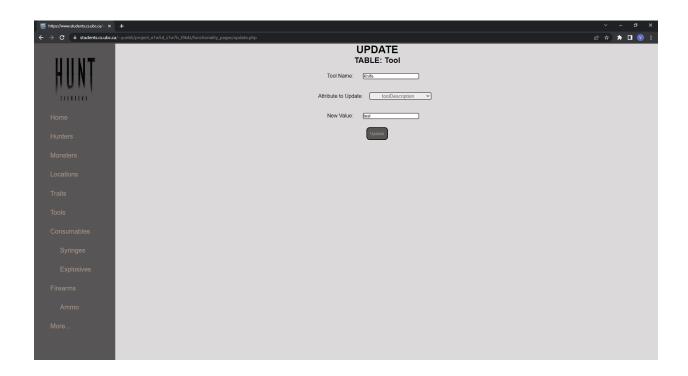




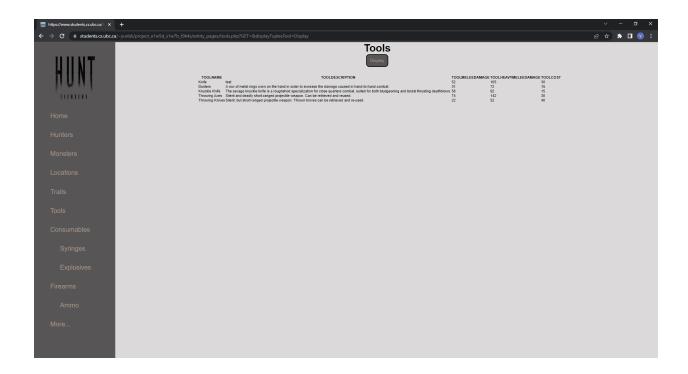
In this delete operation, the user is able to input a consumable name to identify which consumable they would like to delete from the *Consumable* table. Note the cascade-on-delete scenario between the *Consumable* table and the *Explosive* table in the provided example. The backend code for this operation functionality may be found within the <code>handleDeleteRequest()</code> function in <code>./backend\_functionality\handlers.php</code>

## **UPDATE**<sup>3</sup>





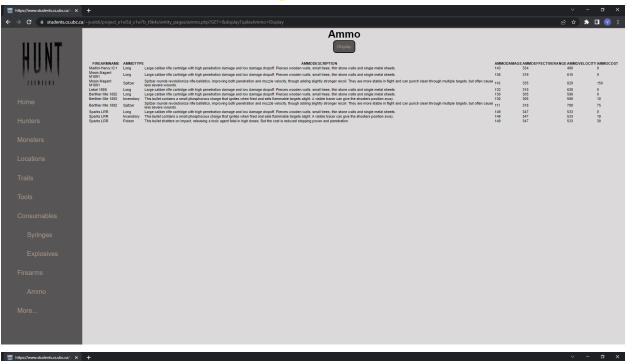
<sup>&</sup>lt;sup>3</sup> This operation satisfies the update rubric item for our project

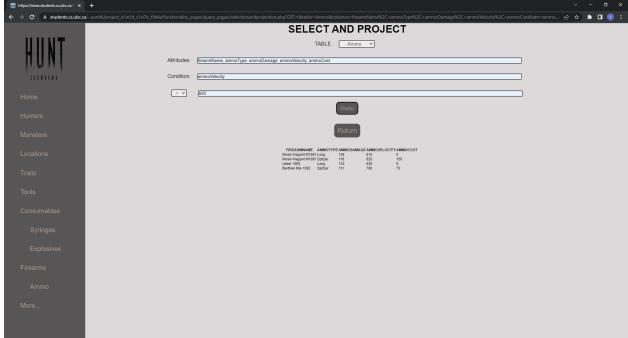


In this update operation, the user is able to input a tool name to identify which tool they would like to update from the *Tool* table. Additionally, they may specify which tool attribute they would like to update as well as a new value for said tool attribute.

The backend code for this operation functionality may be found within the <code>handleUpdateRequest()</code> function in <code>./backend functionality\handlers.php</code>

### **SELECTION & PROJECTION QUERY**

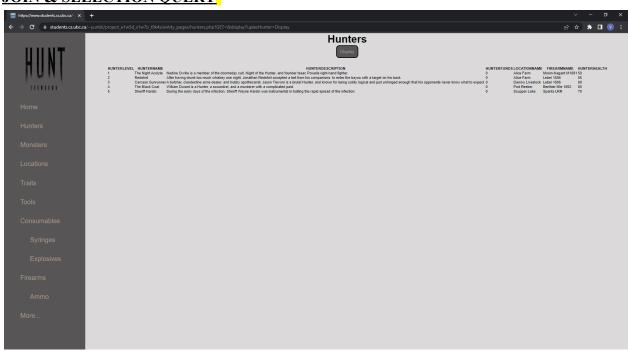


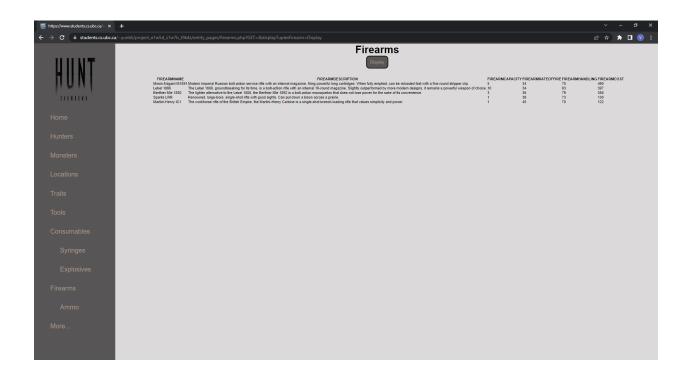


This query projects *firearmName*, *ammoType*, *ammoDamage*, *ammoVelocity*, and *ammoCost* from the *Ammo* table, with the selection condition that the *ammoVelocity* > 600. The backend code for this query functionality may be found within the *handleSelectionAndProjectionRequest()* function in ./backend functionality\handlers.php

<sup>&</sup>lt;sup>4</sup> This query satisfies both selection and projection rubric items for our project

### JOIN & SELECTION QUERY<sup>5</sup>



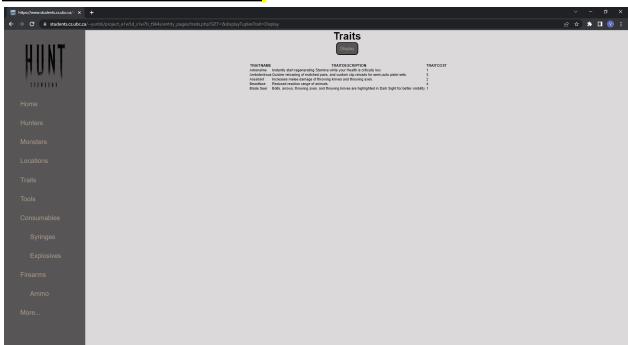


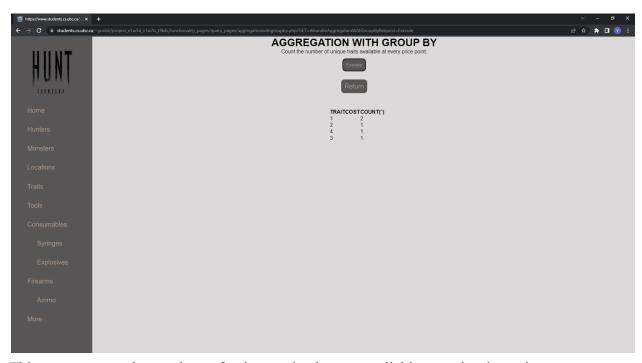
<sup>&</sup>lt;sup>5</sup> This query only satisfies the join rubric item for our project



This query joins the *Hunter* and *Firearm* tables and subsequently selects tuples given the conditions that hunterLevel >= 3 and said hunter is carrying a firearm of firearmCost > 350. The backend code for this query functionality may be found within the handleJoinAndSelectionRequest() function in  $./backend\_functionality handlers.php$ 

### **AGGREGATION WITH GROUP BY**

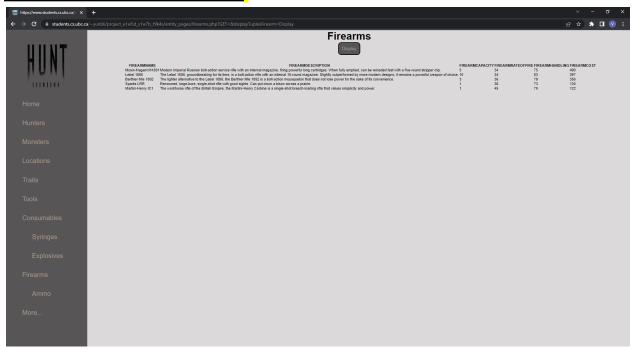




This query counts the numbers of unique traits that are available at each price point. The backend code for this query functionality may be found within the <code>handleAggregationWithGroupByRequest()</code> function in <code>./backend\_functionality\handlers.php</code>

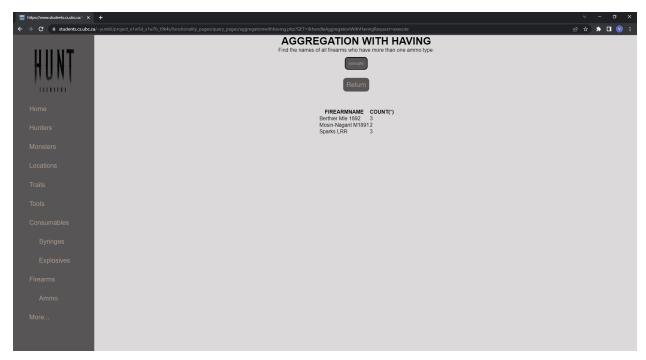
<sup>&</sup>lt;sup>6</sup> This query satisfies the aggregation with group by rubric item for our project

# AGGREGATION WITH HAVING<sup>1</sup>



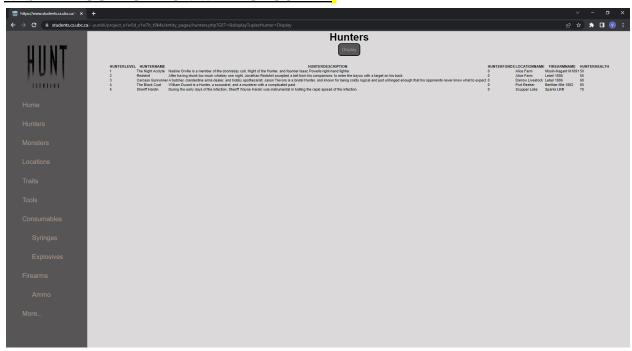


<sup>&</sup>lt;sup>7</sup> This query satisfies the aggregation with having rubric item for our project



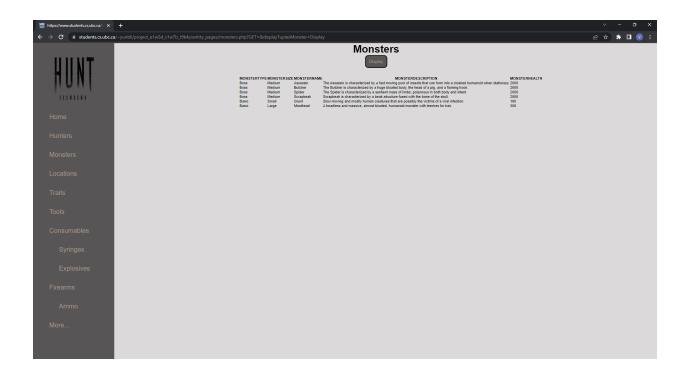
This query finds the names of all firearms who have more than one ammo type. The backend code for this query functionality may be found within the <code>handleAggregationWithHaving()</code> function in <code>./backend\_functionality\handlers.php</code>

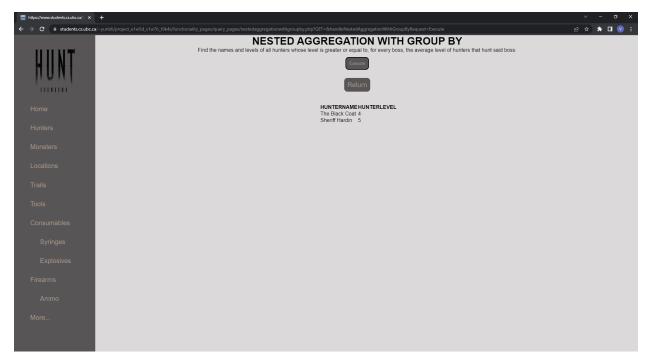
### **NESTED AGGREGATION WITH GROUP BY**<sup>8</sup>





<sup>&</sup>lt;sup>8</sup> This query satisfies the nested aggregation with group by rubric item for our project

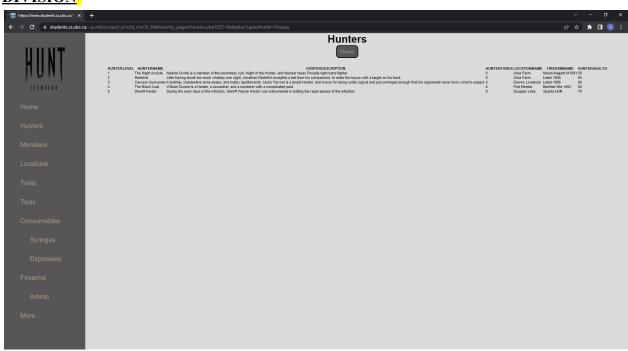




This query finds the names and levels of all hunters whose level is greater or equal to, for every boss, the average level of hunters that hunt said boss.

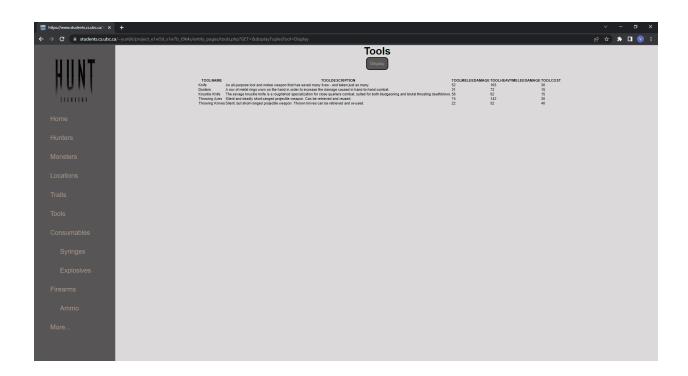
The backend code for this query functionality may be found within the handleNestedAggregationWithGroupByRequest() function in ./backend\_functionality\handlers.php

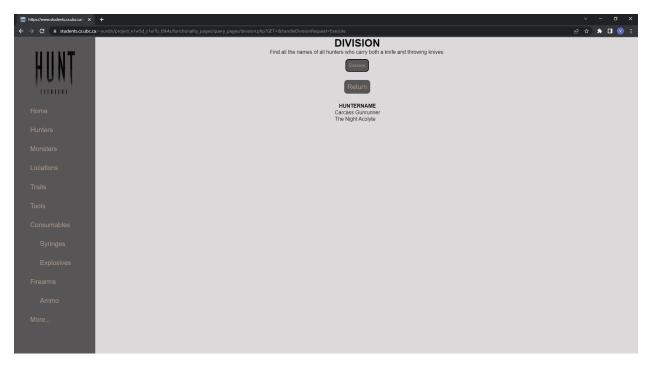
# **DIVISION**<sup>9</sup>





<sup>&</sup>lt;sup>9</sup> This query satisfies the division rubric item for our project





This query finds the names of all hunters who carry both a knife and throwing knives. The backend code for this query functionality may be found within the <code>handleDivisionRequest()</code> function in <code>./backend\_functionality\handlers.php</code>