CS4347 Database Systems
Final Project Deliverable 2

MMO Virtual Item Marketplace

Group Members: Alexis Kaufman, Amila Haque, Brandon Maweu, Justin Jeirles, Kashvi Asokan, Brian Pham, Quintin Mitchell-Reyes, Ilan Ali, Shahreen Iqbal

Description

Title: MMO Virtual Item Marketplace

GitHub repository details: https://github.com/BNP200000/4347-005-MMO_Marketplace.git

Team Members:

- Alexis Kaufman: will be the frontend lead, whose duties will include setting up the development environment using React TSX, working on the frontend elements of the project, and onboarding other frontend team members through the project setup.
- Amila Haque: will act alongside Kashvi as a project manager, helping schedule meetings and delegating tasks throughout the project. Amila will also work in the database portion of the backend but has skills that they will apply to the front end, such as having familiarity with React and art.
- Brandon Maweu: will be assisting in the backend development for the project.
- Justin Jeirles will be working on the backend development and assisting with front-end development when needed.
- Kashvi Asokan: will be one of the project managers responsible for setting deadlines and assigning tasks, and will step in wherever else is needed.
- Brian Pham: will be working on the backend development for the project.
- Quintin Mitchell-Reyes: will assist with frontend development and implementing game universe-related ideas into the marketplace, and will create simple art to display to the user.
- Ilan Ali will work on backend development and assist in implementing ideas into the marketplace.
- Shahreen Iqbal: will work on the front end and assist with the back end along with any graphics required for the project.
- Everybody can assist with the database code itself when needed.

Our Motivation

Our team formed due to a mutual enjoyment of video games and game development. As video game players ourselves, we wanted to try a hand at implementing functions from one. Since we are interested in this topic, we expect this to show in our final product, as we believe it is a pleasure to work on something you are passionate about. We appreciate the flexibility of the professor in choosing our topic. The final product can be considered for use as a component of a new/existing game.

Project Timeline and Delegation of Tasks

Our team meets regularly to decide how to delegate tasks and discuss the implementation of our ideas. Our first meeting after Deliverable 1 was released was on September 29th, in which we determined which group members would work on specific tasks, and set a team deadline of October 6th to do research for the background section. All members of the team contributed to the research, while Amila and Ilan were assigned to work on the EER diagram. Quintin worked on the schema diagram, and Alexis, Brandon, and Brian were assigned to work on creating and populating the databases. Justin and Shahreen were assigned to work on the query execution, and Kashvi worked on the description, introduction, and references. We then met on October 8th to go over the specifics of the database so everyone could start working on their assigned parts.

Introduction

This project centers on building a virtual item marketplace for an MMO (massively multiplayer online) game. There are many games out there that have their own marketplaces, and we wanted to make one of our own to create a secure and user-friendly platform that would enhance the user's experience. While other marketplaces may face some challenges with protecting players' information because of how their marketplace was developed and stored within the game itself, our marketplace is completely separate from the game itself and thus adds an extra layer of security. This means that even if the main game is hacked, the information stored in the marketplace database is completely safe. We also aim to make implementation easier, which makes it faster for other games to implement this marketplace into their own game without much hassle. Although this means that our marketplace won't come with features that would be unique to a specific game, we are providing the base database and basic features that come with an MMO marketplace, making it easier for games to simply add or edit a few things to make it cater to their game specifically.

Background and Related Work

FIFA Marketplace (2008) [1]

- The FIFA Marketplace is a good example of an in-game marketplace. This is where players of the game FIFA can buy or sell items available in the game, such as player cards or consumables, using real money. There are a few different filters that players can use to sort through items, such as the Rarity filter or the Price Range. Players can also search through different categories to browse different items. In addition, FIFA Marketplace has a search filter that allows the player to find exactly what they're looking for. We will be implementing filters into our database as well, but they will differ from the FIFA Marketplace filters as we will tailor the filters to our specific content. While our database will also allow players to trade items for currency, we will not be using

real-world currency and instead will be using an in-game currency that players earn in some way.

Second Life Marketplace (2010) [2]

- Second Life Marketplace is a virtual marketplace where users can buy and sell virtual goods such as clothing, buildings, vehicles, and even services. This marketplace complements the game Second Life which is a 3D online space where players can explore, socialize, and create their own digital items and environments. The marketplace is extensive, and users can create their own items for sale. The Marketplace is a key part of the game's economy, allowing users to buy, sell, and trade virtual goods, ranging from clothing and avatars to buildings, vehicles, and even services like scripting or custom designs. The marketplace also has its own currency, which is reflected in the game, called the Linden Dollar (L).

Unity Asset Store (November 2010) [3]

- The Unity Asset Store is a good example of a digital marketplace that facilitates the selling and purchasing of assets for game developers or other creative professionals. It provides a platform where users can access a wide variety of assets from 3D models, VFX, and even entire game systems which can accelerate development workflows. The site lets users list and sell assets that other users can directly import into their Unity development environment. The site itself uses a structured database to maintain detailed information about each asset and the store contains important metadata regarding titles, descriptions, tags, and system requirements. In addition, it provides a useful search and filtering option where users can search for assets by using keywords, pricing, and price range. Lastly, it also manages transactions, licensing, and user access rights because when users purchase an asset it grants the user a license to download and use that asset for example. The Unity Asset store is very similar to what we are attempting to do with the MMO marketplace, like keeping track of transactions and allowing users to list their own items/assets. One of the biggest differences between our MMO marketplace and the Unity Asset store is that the Unity Asset store requires less concurrency control since users are buying digital rights to use an asset while in the MMO marketplace, you're buying digital items. It is more similar to a virtual economy since users would be buying a specific item hence the need for locking mechanisms to prevent two users from buying the same item at the same time.

Steam Community Market (December 12, 2012) [4]

- The Steam Community Market is a virtual marketplace within the Steam platform – a digital game distribution service and store developed by Valve – and is one of the largest digital distribution platforms for PC gaming, facilitating the trading of in-game items. It uses a digital currency, "Funds", stored in a user's "Steam Wallet" and purchased using real-life money. Within the Steam Community Market, users can purchase any item they wish for any game. One distinct similarity that our MMO Marketplace can draw from this example is that, like the Steam Community Market, our database also exists as an

external, virtual market for users to purchase MMO-related items – that can be filtered into categories – and translated back into the game. What makes the MMO Marketplace different from the Community Market is that the MMO Marketplace is essentially a virtual market for one game; The Steam Community Market is a multi-game virtual market, meaning that users can purchase any items that exist within that particular game.

Eorzea Database (FFXIV, The Lodestone) (August 2013) [5]

- The Eorzea Database is a user-friendly way of accessing most of FFXIV's data on items, raids, quests, etc. I'll focus on items here. Like in our database brainstorming, items are split into several categories, including arms/weapons, armor, accessories, materials, and more. Every type of item we listed has some sort of parallel in the Eorzea Database, which could be used to justify our item tuples. However, this public database only lists types of items, not specific instances or player inventories.
- Every item has information on it, much of which depends on what type of item it is. For example, the terrible magic staff pictured above lists its damage, cast time, stat modifiers, which classes can equip it, how much it can be upgraded, crafting and repair information, and sell prices for NPC vendors and players. Although it can be sold by players on the game's marketboard, only NPC vendors and quests are listed as sources of items on this website. This still may be comparable to any search functions or relationships between items and sellers in our database.
- In comparison to the staff, a fish item would have little of the same info. Instead, it has a description used for the fisher class, a note if it's able to be displayed in a player's home, and a list of crafting recipes it's used in.
- Overall, this source might be best compared to our item entity, its attributes, and how it's related to other entities such as characters, classes, and maybe transactions.

BitSkins (July 21st, 2015) [6]

- BitSkins is a marketplace that allows players from Counter-Strike: Global Offensive (CS) and Dota 2 to buy, sell, and trade in-game items for real-world currency. Because BitSkins involves items from MMOs, many of the attributes and filtered views of its database can apply to ours as well. BitSkins also showcases a transparent pricing system, enabling users to see the current market value of items, which aids in making informed buying and selling decisions. In virtual item trading, where prices can fluctuate based on demand and rarity, this transparent pricing can be a functionality that we implement in our marketplace. However, our virtual marketplace differs in that we would only be using our in-game monetary system, so the security measures that BitSkins needs to employ for real-world transactions aren't as necessary for our database. Furthermore, unlike BitSkins, which mainly focuses on a general trade of skins for all users, our marketplace will incorporate class-specific items, so our database will reflect how certain items are restricted to specific player classes.

CryptoPunks (2017) [7]

- CryptoPunks is an example of a system that has digital collectibles, specifically on the blockchain. While our project has a simpler scope and does not involve NFTs, this site lets users buy, sell, and trade digital assets in the form of 24x24 pixel art characters. Furthermore, each character has many attributes, such that characters can have special eyes, facial hair, mouth, ears, and more. Then, the rarity of any given character is measured by the combination and frequency of these attributes; in the image, one is much rarer than the other due to its many rare traits. The system thusly allows players to determine the uniqueness and value of each "Punk", so the database schema must not only accommodate ownership and trade history, but also it must calculate and compare "Total Rarity Scores". These characters all come together to form a digital marketplace similar to the one we are designing, such that different items (or in this case, characters), will have special values and not all be worth the same amount of currency.

Runescape Marketplace (June 7, 2021) [8]

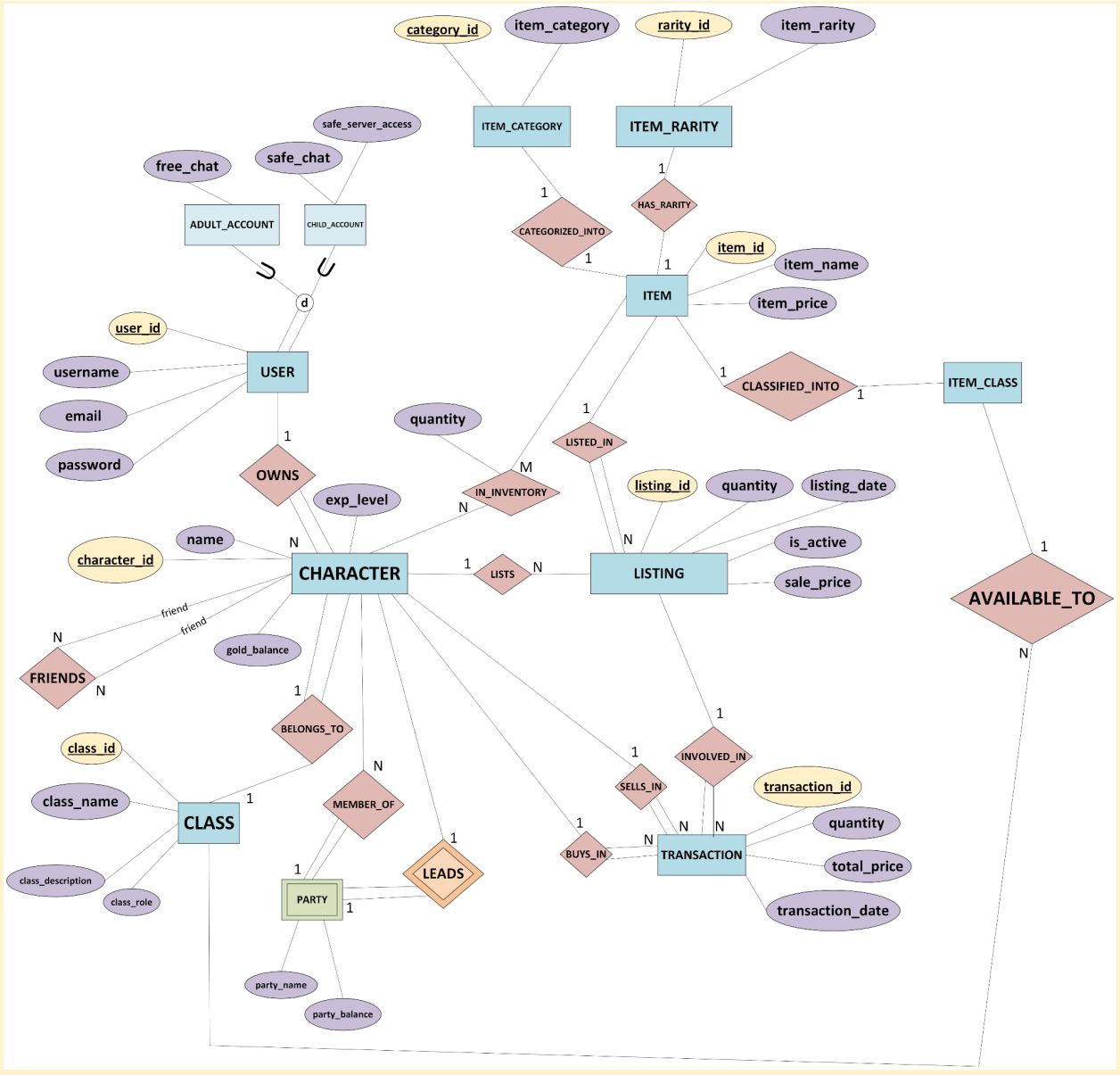
- This source is a good example of an MMO marketplace. Despite the game being released in 2001, the marketplace is a relatively new addition to the game and is a replacement for several older features in the game. Our database is similar to the marketplace because our goal is to create a shop where players can buy or sell items. The items will be categorized and have different attributes. The marketplace in the game is very similar to what we plan to create because it sells several things like consumables, armor/costumes, animations, pets, etc. The marketplace uses in-game called oddments. There are several seasonal currencies too, but that will not be something we add to our database shop.

Design & Implementation

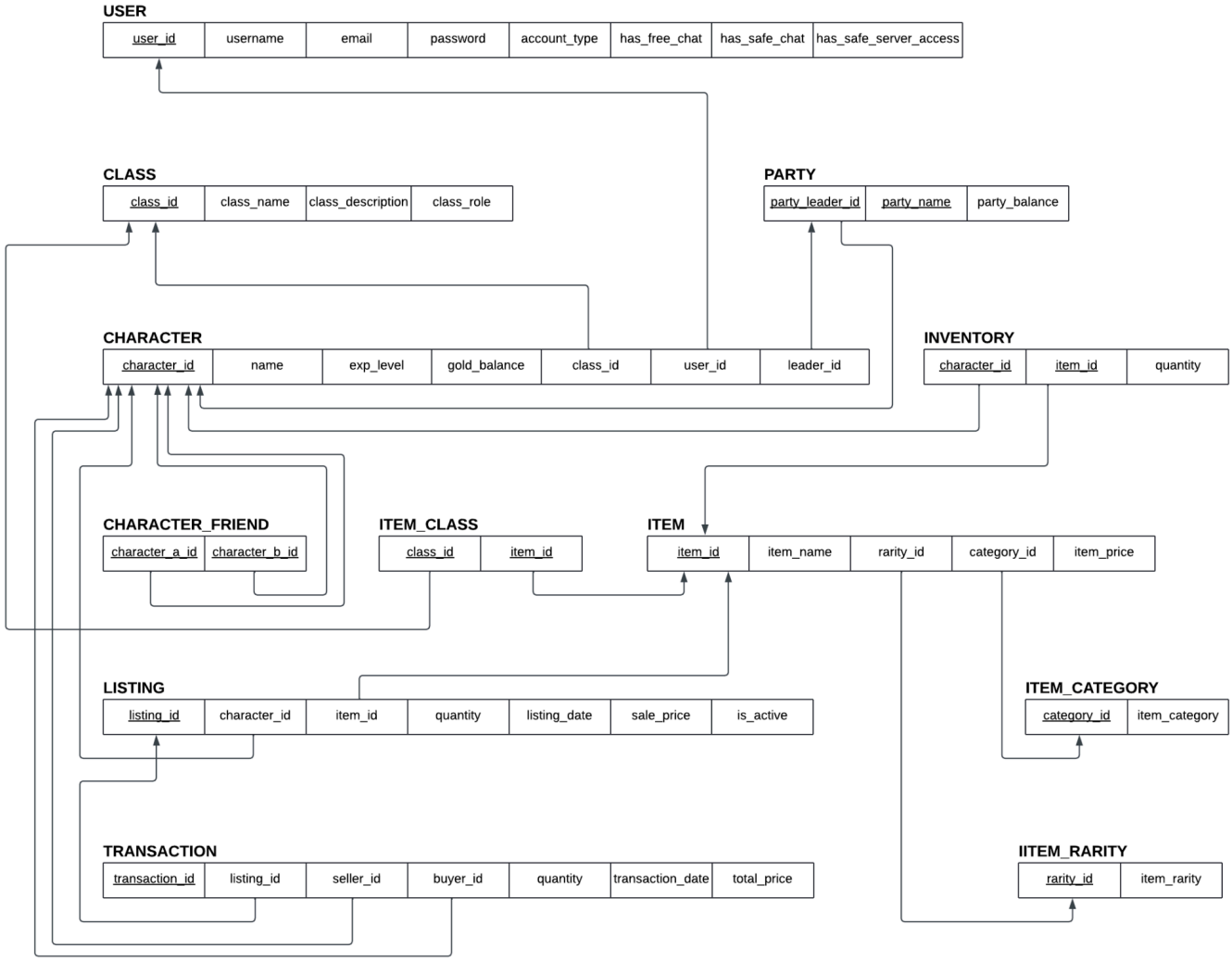
3.1. Normalization of EER Conceptual Data Model

The changes made to normalize our database revolved around the item table since it wasn't normalized. Since our allowed_classes column contained a set/array of data which breaks the 1NF. The first step we did was to break up the allowed_classes column and create another table called item_class, a table made up of a composite primary key which are item_id and class_id which refer to their respective tables item and class. The Item_Class table would hold information regarding the association between an item and class, so if a specific class could use an item it's item_id and the class_id would be an entry in the table. This would sufficiently normalize our previous allowed_classes column. We later realized the columns item_category and item_rarity were only in 2NF and not 3NF because there was a transitive dependency with item_id → item_name → item_category and item_id → item_name → item_rarity. The way we normalized this was by creating two new tables, item_category and item_rarity. Item_Category table had two columns, the primary key was the category_id and the second column was the category_name. Similar to the Item_Category table, the Item_Rarity table had two columns. The primary key was rarity_id and the second column was the rarity_name. We then changed the

columns for the items and replaced item_category with category_id which references the item_category(category_id) table and item_rarity with rarity_id which references the item_rarity(rarity_id) table. With all these changes we sufficiently put our database in 3NF form.



3.2. Relational Data Model Design Using Normalized EER Diagram:



3.3. Create your Normalized Database and Populate:

Database Creation

```
CREATE TABLE
    IF NOT EXISTS "USER" (
        user_id VARCHAR(36) PRIMARY KEY,
        username VARCHAR(25) UNIQUE NOT NULL,
        email VARCHAR(25) NOT NULL,
        password VARCHAR(25) NOT NULL,
        account_type VARCHAR(25),
        has_free_chat BOOLEAN,
        has_safe_chat BOOLEAN,
        has_safe_server_access BOOLEAN
    );

▷ Run | New Tab | Copy
CREATE TABLE
    IF NOT EXISTS "CLASS" (
        class_id VARCHAR(36) PRIMARY KEY,
        class_name VARCHAR(25) UNIQUE,
        class_description VARCHAR(25) NOT NULL,
        class_role VARCHAR(25) NOT NULL
    );

▷ Run | New Tab | Copy
CREATE TABLE
    IF NOT EXISTS "CHARACTER" (
        character_id VARCHAR(36) PRIMARY KEY,
        character_name VARCHAR(25) UNIQUE,
        exp_level INT CHECK(exp_level >= 0) NOT NULL,
        gold_balance INT CHECK(gold_balance >= 0) NOT NULL,
        owner_id VARCHAR(36) NOT NULL,
        character_class VARCHAR(36) NOT NULL,
        leader_id VARCHAR(36),
        FOREIGN KEY (owner_id) REFERENCES "USER" (user_id) ON DELETE CASCADE ON UPDATE CASCADE, -- "OWNS" relationship, 1 to N
        FOREIGN KEY (character_class) REFERENCES "CLASS" (class_id) ON DELETE CASCADE ON UPDATE CASCADE -- "BELONGS TO" relationship, 1 to 1
        -- NOTE; The foreign key constraint for "leader_id" is located in "insert_tables.sql" to avoid circular dependency. It relies on "PARTY"'s party_leader.
    );
```

```
CREATE TABLE
    IF NOT EXISTS "CHARACTER_FRIEND" (
        character_a_id VARCHAR(36),
        character_b_id VARCHAR(36),
        PRIMARY KEY (character_a_id, character_b_id),
        FOREIGN KEY (character_a_id) REFERENCES "CHARACTER" (character_id),
        FOREIGN KEY (character_b_id) REFERENCES "CHARACTER" (character_id),
        CHECK (character_a_id <> character_b_id) -- Reject adding self as friend
    );

▷ Run | New Tab | Copy
CREATE TABLE
    IF NOT EXISTS "PARTY" (
        party_name VARCHAR(50) UNIQUE,
        party_leader VARCHAR(36) UNIQUE,
        party_balance INT CHECK(party_balance >= 0) NOT NULL,
        PRIMARY KEY (party_name, party_leader),
        FOREIGN KEY (party_leader) REFERENCES "CHARACTER" (character_id) ON DELETE CASCADE ON UPDATE CASCADE
    );

▷ Run | New Tab | Copy
CREATE TABLE
    IF NOT EXISTS "ITEM_CATEGORY" (
    category_id SERIAL PRIMARY KEY,
    item_category VARCHAR(25) NOT NULL UNIQUE
);

▷ Run | New Tab | Copy
CREATE TABLE
    IF NOT EXISTS "ITEM_RARITY" (
    rarity_id SERIAL PRIMARY KEY,
    item_rarity VARCHAR(25) NOT NULL UNIQUE
);
```

```
CREATE TABLE
    IF NOT EXISTS "ITEM" (
        item_id SERIAL PRIMARY KEY,
        item_name VARCHAR(50) NOT NULL UNIQUE,
        category_id INT REFERENCES "ITEM_CATEGORY"(category_id),
        rarity_id INT REFERENCES "ITEM_RARITY"(rarity_id),
        item_price NUMERIC(10, 0) CHECK(item_price > 0) NOT NULL
    );
```

▷ Run | New Tab | Copy

```
CREATE TABLE
    IF NOT EXISTS "ITEM_CLASS" (
        item_id SERIAL NOT NULL REFERENCES "ITEM"(item_id) ON DELETE CASCADE,
        class_id VARCHAR(36) NOT NULL REFERENCES "CLASS"(class_id) ON DELETE CASCADE,
        PRIMARY KEY (item_id, class_id)
);
```

▷ Run | New Tab | Copy

```
CREATE TABLE
    IF NOT EXISTS "IN_INVENTORY" (
        character_id VARCHAR(36) NOT NULL,
        item_id SERIAL NOT NULL,
        quantity int NOT NULL CHECK(quantity >= 0),
        PRIMARY KEY (character_id, item_id),
        FOREIGN KEY (character_id) REFERENCES "CHARACTER" (character_id) ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (item_id) REFERENCES "ITEM" (item_id) ON DELETE CASCADE ON UPDATE CASCADE
    );
```

```
CREATE TABLE
    IF NOT EXISTS "LISTING" (
        listing_id SERIAL PRIMARY KEY,
        character_id VARCHAR(36) NOT NULL,
        item_id SERIAL NOT NULL,
        quantity int NOT NULL CHECK(quantity >= 0),
        listing_date DATE NOT NULL,
        is_active BOOLEAN NOT NULL,
        sale_price NUMERIC(10, 0) CHECK(sale_price >= 0),
        FOREIGN KEY (character_id) REFERENCES "CHARACTER" (character_id) ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (item_id) REFERENCES "ITEM" (item_id) ON DELETE CASCADE ON UPDATE CASCADE
    );
```

▷ Run | New Tab | Copy

```
CREATE TABLE
    IF NOT EXISTS "TRANSACTION" (
        transaction_id VARCHAR(36) PRIMARY KEY,
        listing_id SERIAL NOT NULL,
        seller_id VARCHAR(36) NOT NULL,
        buyer_id VARCHAR(36) NOT NULL,
        quantity INT NOT NULL CHECK(quantity > 0),
        total_price INT,
        transaction_date DATE NOT NULL,
        FOREIGN KEY (seller_id) REFERENCES "CHARACTER" (character_id) ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (buyer_id) REFERENCES "CHARACTER" (character_id) ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (listing_id) REFERENCES "LISTING" (listing_id) ON DELETE CASCADE ON UPDATE CASCADE,
        CHECK (seller_id <> buyer_id) -- Reject transaction of self
    );
```

```sql
-- Caluculate the total price value for the TRANSACTION table
▷ Run | Copy
CREATE OR REPLACE FUNCTION calculate_total_price()
RETURNS TRIGGER AS $$
BEGIN
    SELECT sale_price INTO NEW.total_price
    FROM "LISTING" as listing
    WHERE listing.listing_id = NEW.listing_id;
    NEW.total_price := NEW.total_price * NEW.quantity;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- CREATE trigger if it does not exist
▷ Run | Copy
DO $$
BEGIN
    IF NOT EXISTS(
        SELECT 1 FROM pg_trigger
        WHERE tgname = 'set_total_price'
    ) THEN
        CREATE TRIGGER set_total_price
        BEFORE INSERT OR UPDATE ON "TRANSACTION"
        FOR EACH ROW
        EXECUTE FUNCTION calculate_total_price();
    END IF;
END $$;
```

```sql
-- Update total_price when TRANSACTION is updated
▷ Run | Copy
CREATE OR REPLACE FUNCTION update_total_price()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE "TRANSACTION"
    SET total_price = NEW.quantity * NEW.sale_price
    WHERE listing_id = NEW.listing_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

▷ Run | Copy
DO $$
BEGIN
    IF NOT EXISTS(
        SELECT 1 FROM pg_trigger
        WHERE tgname = 'update_total_price'
    ) THEN
        CREATE TRIGGER update_total_price
        AFTER UPDATE OF quantity, sale_price ON "LISTING"
        FOR EACH ROW
        EXECUTE FUNCTION update_total_price();
    END IF;
END $$;
```

Database Insertion

```sql
INSERT INTO
    "CHARACTER" (character_id, exp_level, character_name, gold_balance, owner_id, character_class, leader_id)
VALUES
    ('1a2b3c456', 5, 'Alduin', 300, '1x2y3z456', '1m2n3o456', NULL),
    ('2b3c4d567', 3, 'Lyria', 150, '2y3z4a567', '2n3o4p567', '1a2b3c456'),
    ('3c4d5e678', 7, 'Gorath', 500, '3z4a5b678', '3o4p5q678', '1a2b3c456'),
    ('4d5e6f789', 6, 'Serana', 250, '4a5b6c789', '1m2n3o456', NULL),
    ('5e6f7g890', 10, 'Theron', 800, '5b6c7d890', '2n3o4p567', '1a2b3c456'),
    ('6f7g8h901', 4, 'Miraak', 200, '6c7d8e901', '3o4p5q678', '2b3c4d567'),
    ('7g8h9i012', 12, 'Valdrin', 1000, '7d8e9f012', '4p5q6r789', '3c4d5e678'),
    ('8h9i0j123', 2, 'Nalya', 100, '8e9f0g123', '1m2n3o456', '2b3c4d567'),
    ('9i0j1k234', 9, 'Druin', 750, '9f0g1h234', '4p5q6r789', '3c4d5e678'),
    ('0j1k2l345', 8, 'Kael', 600, '0g1h2i345', '5q6r7s890', '1a2b3c456');

▷ Run | New Tab | Copy
INSERT INTO
    "PARTY" (party_name, party_leader, party_balance)
VALUES
    ('Lyrical', '1a2b3c456', 53422),
    ('Arcane', '4d5e6f789', 3234223),
    ('Dragon Slayer', '2b3c4d567', 1000),
    ('Witch Hunter', '3c4d5e678', 54223),
    ('Tuba Gang', '7g8h9i012', 546),
    ('Stroopwafel', '0j1k2l345', 564),
    ('Birds of Prey', '9i0j1k234', 32),
    ('The Fallen', '6f7g8h901', 7869),
    ('Masked Fools', '5e6f7g890', 435345),
    ('Asgard', '8h9i0j123', 34523);

▷ Run | New Tab
ALTER TABLE "CHARACTER"
ADD CONSTRAINT FK_leader_id
FOREIGN KEY (leader_id) REFERENCES "PARTY" (party_leader);
```

```sql
INSERT INTO
    "CHARACTER_FRIEND" (character_a_id, character_b_id)
VALUES
    ('1a2b3c456', '2b3c4d567'),
    ('1a2b3c456', '3c4d5e678'),
    ('2b3c4d567', '6f7g8h901'),
    ('3c4d5e678', '9i0j1k234'),
    ('4d5e6f789', '1a2b3c456'),
    ('5e6f7g890', '2b3c4d567'),
    ('6f7g8h901', '7g8h9i012'),
    ('7g8h9i012', '3c4d5e678'),
    ('8h9i0j123', '9i0j1k234'),
    ('0j1k2l345', '5e6f7g890');
```

▷ Run | New Tab | Copy

```sql
INSERT INTO "ITEM_CATEGORY" (item_category) VALUES
    ('Consumable'),
    ('Weapon'),
    ('Armor'),
    ('Accessory'),
    ('Shield'),
    ('Headgear');
```

▷ Run | New Tab | Copy

```sql
INSERT INTO "ITEM_RARITY" (item_rarity) VALUES
    ('Common'),
    ('Uncommon'),
    ('Rare'),
    ('Epic'),
    ('Legendary');
```

```sql
INSERT INTO "ITEM" (item_name, category_id, rarity_id, item_price) VALUES
    ('Health Potion', 1, 1, 50),
    ('Greater Health Potion', 1, 2, 100),
    ('Mana Potion', 1, 1, 50),
    ('Elixir of Fortitude', 1, 3, 200),
    ('Steel Sword', 2, 3, 400),
    ('Iron Dagger', 2, 2, 250),
    ('Fire Staff', 2, 4, 700),
    ('Leather Armor', 3, 1, 150),
    ('Plate Armor', 3, 3, 600),
    ('Ring of Protection', 4, 3, 350),
    ('Necklace of Wisdom', 4, 4, 800),
    ('Wooden Shield', 5, 1, 100),
    ('Divine Shield', 5, 5, 1200),
    ('Wizard Hat', 6, 3, 500),
    ('Crown of the Bard King', 6, 5, 1000);


--Consumables
▷ Run | New Tab | Copy
INSERT INTO "ITEM_CLASS" (item_id, class_id) VALUES
    (1, '0v1w2x345'), (1, '1m2n3o456'), (1, '2n3o4p567'), (1, '3o4p5q678'), (1, '4p5q6r789'),
    (1, '5q6r7s890'), (1, '6r7s8t901'), (1, '7s8t9u012'), (1, '8t9u0v123'), (1, '9u0v1w234'),
    (2, '0v1w2x345'), (2, '1m2n3o456'), (2, '2n3o4p567'), (2, '3o4p5q678'), (2, '4p5q6r789'),
    (2, '5q6r7s890'), (2, '6r7s8t901'), (2, '7s8t9u012'), (2, '8t9u0v123'), (2, '9u0v1w234'),
    (3, '0v1w2x345'), (3, '1m2n3o456'), (3, '2n3o4p567'), (3, '3o4p5q678'), (3, '4p5q6r789'),
    (3, '5q6r7s890'), (3, '6r7s8t901'), (3, '7s8t9u012'), (3, '8t9u0v123'), (3, '9u0v1w234'),
    (4, '0v1w2x345'), (4, '1m2n3o456'), (4, '2n3o4p567'), (4, '3o4p5q678'), (4, '4p5q6r789'),
    (4, '5q6r7s890'), (4, '6r7s8t901'), (4, '7s8t9u012'), (4, '8t9u0v123'), (4, '9u0v1w234');
```

```
--Weapons, Armor, Headgear, Shield
▷ Run | New Tab | Copy
INSERT INTO "ITEM_CLASS" (item_id, class_id) VALUES
    (5, '1m2n3o456'), (5, '2n3o4p567'), (5, '5q6r7s890'),
    (6, '1m2n3o456'), (6, '2n3o4p567'),
    (7, '3o4p5q678'), (7, '7s8t9u012'),
    (8, '1m2n3o456'), (8, '5q6r7s890'),
    (9, '1m2n3o456'), (9, '5q6r7s890'),
    (10, '0v1w2x345'), (10, '4p5q6r789'), (10, '7s8t9u012'),
    (11, '3o4p5q678'), (11, '4p5q6r789'),
    (12, '1m2n3o456'), (12, '5q6r7s890'),
    (13, '4p5q6r789'), (13, '5q6r7s890'),
    (14, '3o4p5q678'), (14, '7s8t9u012'),
    (15, '9u0v1w234');


▷ Run | New Tab | Copy
INSERT INTO "LISTING" (character_id, quantity, listing_date, is_active, sale_price) VALUES
('1a2b3c456', 10, '2024-09-15', TRUE, 150),
('2b3c4d567', 5, '2024-08-23', TRUE, 350),
('3c4d5e678', 1, '2024-10-01', FALSE, 1000),
('4d5e6f789', 2, '2024-09-10', TRUE, 1200),
('5e6f7g890', 3, '2024-08-30', FALSE, 500),
('6f7g8h901', 7, '2024-09-25', TRUE, 250),
('7g8h9i012', 4, '2024-09-02', TRUE, 200),
('8h9i0j123', 12, '2024-07-20', FALSE, 300),
('9i0j1k234', 6, '2024-08-29', TRUE, 950),
('0j1k2l345', 1, '2024-09-18', FALSE, 50),
('1a2b3c456', 4, '2024-08-15', TRUE, 400),
('2b3c4d567', 6, '2024-07-28', FALSE, 180),
('3c4d5e678', 8, '2024-09-30', TRUE, 600),
('4d5e6f789', 9, '2024-08-25', TRUE, 1000),
('5e6f7g890', 10, '2024-09-01', FALSE, 800);
```

```sql
INSERT INTO "IN_INVENTORY" (character_id, item_id, quantity) VALUES
('0j1k2l345', 1, 5),
('0j1k2l345', 5, 1),
('1a2b3c456', 2, 3),
('1a2b3c456', 8, 1),
('2b3c4d567', 6, 2),
('2b3c4d567', 10, 1),
('3c4d5e678', 7, 1),
('3c4d5e678', 3, 4),
('4d5e6f789', 4, 2),
('4d5e6f789', 12, 1),
('5e6f7g890', 9, 1),
('5e6f7g890', 13, 1),
('6f7g8h901', 15, 1),
('7g8h9i012', 14, 1),
('8h9i0j123', 11, 1),
('9i0j1k234', 1, 6),
('9i0j1k234', 5, 1);
```

▷ Run | New Tab | Copy

```sql
INSERT INTO
    "TRANSACTION" (transaction_id, seller_id, buyer_id, quantity, transaction_date)
VALUES
    ('nfgu04bkz1', '1a2b3c456', '2b3c4d567', 9, '2024-10-22'),
    ('7lgij2an48', '2b3c4d567', '1a2b3c456', 5, '2024-10-28'),
    ('tiu0p1pmon', '3c4d5e678', '0j1k2l345', 4, '2024-11-04'),
    ('vt4ix91tle', '4d5e6f789', '5e6f7g890', 6, '2024-11-06'),
    ('urgd1d5h6i', '5e6f7g890', '1a2b3c456', 8, '2024-11-12'),
    ('o3jc0zpf5w', '6f7g8h901', '5e6f7g890', 10, '2024-11-23'),
    ('nTdbGTf8Mz', '7g8h9i012', '6f7g8h901', 2, '2024-11-30'),
    ('46onzox4ew', '8h9i0j123', '0j1k2l345', 1, '2024-12-14'),
    ('bf9ky0a20c', '9i0j1k234', '5e6f7g890', 1, '2024-12-21'),
    ('6UFAVp7Tok', '0j1k2l345', '7g8h9i012', 4, '2024-12-24');
```

Resulting Tables

1. USER

Query | Query History

```
1  select * from "USER";
```

Data Output | Messages | Notifications

	user_id [PK] character varying (36)	username character varying (25)	email character varying (25)	password character varying (25)	account_type character varying (25)	has_free_chat boolean	has_safe_chat boolean	has_safe_server_access boolean
1	1x2y3z456	dragonlord	dragonlord@example.com	pass123	adult	true	true	true
2	2y3z4a567	skywalker	skywalker@example.com	star123	child	false	true	false
3	3z4a5b678	ironfist	ironfist@example.com	punch789	adult	true	true	true
4	4a5b6c789	shadowblade	shadowblade@example.c...	shadow456	child	false	true	false
5	5b6c7d890	stormrider	stormrider@example.com	rider567	adult	true	true	true
6	6c7d8e901	firemage	firemage@example.com	fire321	child	false	true	false
7	7d8e9f012	windrunner	windrunner@example.com	wind111	adult	true	true	true
8	8e9f0g123	earthwarden	earthwarden@example.co...	earth222	child	false	true	false
9	9f0g1h234	waterseer	waterseer@example.com	water333	adult	true	true	true
10	0g1h2i345	lightbringer	lightbringer@example.com	light444	child	false	true	false

2. CLASS

Query | Query History

```
1  select * from "CLASS";
```

Data Output | Messages | Notifications

	class_id [PK] character varying (36)	class_name character varying (25)	class_description character varying (25)	class_role character varying (25)
1	1m2n3o456	Warrior	Frontline fighter	Tank
2	2n3o4p567	Rogue	Stealthy assassin	DPS
3	3o4p5q678	Mage	Master of magic	DPS
4	4p5q6r789	Cleric	Healer of the wounded	Healer
5	5q6r7s890	Paladin	Holy warrior	Tank
6	6r7s8t901	Ranger	Expert marksman	DPS
7	7s8t9u012	Druid	Nature spellcaster	Healer
8	8t9u0v123	Monk	Martial artist	DPS
9	9u0v1w234	Bard	Musical supporter	Support
10	0v1w2x345	Necromancer	Summoner of the dead	DPS

3. CHARACTER

Query | Query History

```
1  select * from "CHARACTER";
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	character_name character varying (25)	exp_level integer	gold_balance integer	owner_id character varying (36)	character_class character varying (36)	leader_id character varying (36)
1	1a2b3c456	Alduin	5	300	1x2y3z456	1m2n3o456	[null]
2	2b3c4d567	Lyria	3	150	2y3z4a567	2n3o4p567	1a2b3c456
3	3c4d5e678	Gorath	7	500	3z4a5b678	3o4p5q678	1a2b3c456
4	4d5e6f789	Serana	6	250	4a5b6c789	1m2n3o456	[null]
5	5e6f7g890	Theron	10	800	5b6c7d890	2n3o4p567	1a2b3c456
6	6f7g8h901	Miraak	4	200	6c7d8e901	3o4p5q678	2b3c4d567
7	7g8h9i012	Valdrin	12	1000	7d8e9f012	4p5q6r789	3c4d5e678
8	8h9i0j123	Nalya	2	100	8e9f0g123	1m2n3o456	2b3c4d567
9	9i0j1k234	Druin	9	750	9f0g1h234	4p5q6r789	3c4d5e678
10	0j1k2l345	Kael	8	600	0g1h2i345	5q6r7s890	1a2b3c456

4. CHARACTER_FRIEND

Query | Query History

```
1  select * from "CHARACTER_FRIEND";
```

Data Output | Messages | Notifications

	character_a_id [PK] character varying (36)	character_b_id [PK] character varying (36)
1	1a2b3c456	2b3c4d567
2	1a2b3c456	3c4d5e678
3	2b3c4d567	6f7g8h901
4	3c4d5e678	9i0j1k234
5	4d5e6f789	1a2b3c456
6	5e6f7g890	2b3c4d567
7	6f7g8h901	7g8h9i012
8	7g8h9i012	3c4d5e678
9	8h9i0j123	9i0j1k234
10	0j1k2l345	5e6f7g890

5. PARTY

Query Query History

```
1  select * from "PARTY";
```

Data Output Messages Notifications

	party_name [PK] character varying (50)	party_leader [PK] character varying (36)	party_balance integer
1	Lyrical	1a2b3c456	53422
2	Arcane	4d5e6f789	3234223
3	Dragon Slayer	2b3c4d567	1000
4	Witch Hunter	3c4d5e678	54223
5	Tuba Gang	7g8h9i012	546
6	Stroopwafel	0j1k2l345	564
7	Birds of Prey	9i0j1k234	32
8	The Fallen	6f7g8h901	7869
9	Masked Fools	5e6f7g890	435345
10	Asgard	8h9i0j123	34523

6. **ITEM_CATEGORY**

Query Query History

```
1  select * from "ITEM_CATEGORY"
```

Data Output Messages Notifications

	category_id [PK] integer	item_category character varying (25)
1	1	Consumable
2	2	Weapon
3	3	Armor
4	4	Accessory
5	5	Shield
6	6	Headgear

7. **ITEM_RARITY**

Query | Query History

```
1  select * from "ITEM_RARITY"
```

Data Output | Messages | Notifications

	rarity_id [PK] integer	item_rarity character varying (25)
1	1	Common
2	2	Uncommon
3	3	Rare
4	4	Epic
5	5	Legendary

8. **ITEM**

Query | Query History

```
1  select * from "ITEM"
```

Data Output | Messages | Notifications

	item_id [PK] integer	item_name character varying (50)	category_id integer	rarity_id integer	item_price numeric (10)
1	1	Health Potion	1	1	50
2	2	Greater Health Potion	1	2	100
3	3	Mana Potion	1	1	50
4	4	Elixir of Fortitude	1	3	200
5	5	Steel Sword	2	3	400
6	6	Iron Dagger	2	2	250
7	7	Fire Staff	2	4	700
8	8	Leather Armor	3	1	150
9	9	Plate Armor	3	3	600
10	10	Ring of Protection	4	3	350
11	11	Necklace of Wisdom	4	4	800
12	12	Wooden Shield	5	1	100
13	13	Divine Shield	5	5	1200
14	14	Wizard Hat	6	3	500
15	15	Crown of the Bard King	6	5	1000
16	16	Warhammer of the Iron Titan	2	4	10000
17	17	Sigil of the Saint	2	5	9000
18	18	Dragonbane Wand	2	3	1500

9. **ITEM_CLASS**

Query | Query History

```
1  select * from "ITEM_CLASS"
```

Data Output | Messages | Notifications

	item_id [PK] integer	class_id [PK] character varying (36)
1	1	0v1w2x345
2	1	1m2n3o456
3	1	2n3o4p567
4	1	3o4p5q678
5	1	4p5q6r789
6	1	5q6r7s890
7	1	6r7s8t901
8	1	7s8t9u012
9	1	8t9u0v123
10	1	9u0v1w234
11	2	0v1w2x345
12	2	1m2n3o456
13	2	2n3o4p567
14	2	3o4p5q678
15	2	4p5q6r789
16	2	5q6r7s890
17	2	6r7s8t901
18	2	7s8t9u012
19	2	8t9u0v123
20	2	9u0v1w234
21	3	0v1w2x345
22	3	1m2n3o456
23	3	2n3o4p567
24	3	3o4p5q678
25	3	4p5q6r789
26	3	5q6r7s890
27	3	6r7s8t901
28	3	7s8t9u012

10. IN_INVENTORY

Query | Query History

```
1  select * from "IN_INVENTORY";
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	item_id [PK] integer	quantity integer
1	0j1k2l345	1	5
2	0j1k2l345	5	1
3	1a2b3c456	2	3
4	1a2b3c456	8	1
5	2b3c4d567	6	2
6	2b3c4d567	10	1
7	3c4d5e678	7	1
8	3c4d5e678	3	4
9	4d5e6f789	4	2
10	4d5e6f789	12	1
11	5e6f7g890	9	1
12	5e6f7g890	13	1
13	6f7g8h901	15	1
14	7g8h9i012	14	1
15	8h9i0j123	11	1
16	9i0j1k234	1	6
17	9i0j1k234	5	1
18	8h9i0j123	16	11
19	0j1k2l345	17	10000
20	8h9i0j123	18	100

11. LISTING

Query | Query History

```
1  select * from "LISTING"
```

Data Output | Messages | Notifications

	listing_id [PK] integer	character_id character varying (36)	item_id integer	quantity integer	listing_date date	is_active boolean	sale_price numeric (10)
1	1	1a2b3c456	1	10	2024-09-15	true	150
2	2	2b3c4d567	2	5	2024-08-23	true	350
3	3	3c4d5e678	3	1	2024-10-01	false	1000
4	4	4d5e6f789	4	2	2024-09-10	true	1200
5	5	5e6f7g890	5	3	2024-08-30	false	500
6	6	6f7g8h901	6	7	2024-09-25	true	250
7	7	7g8h9i012	7	4	2024-09-02	true	200
8	8	8h9i0j123	8	12	2024-07-20	false	300
9	9	9i0j1k234	9	6	2024-08-29	true	950
10	10	0j1k2l345	10	1	2024-09-18	false	50
11	11	1a2b3c456	11	4	2024-08-15	true	400
12	12	2b3c4d567	12	6	2024-07-28	false	180
13	13	3c4d5e678	13	8	2024-09-30	true	600
14	14	4d5e6f789	14	9	2024-08-25	true	1000
15	15	5e6f7g890	15	10	2024-09-01	false	800

12. TRANSACTION

Query | Query History

```
1  select * from "TRANSACTION"
```

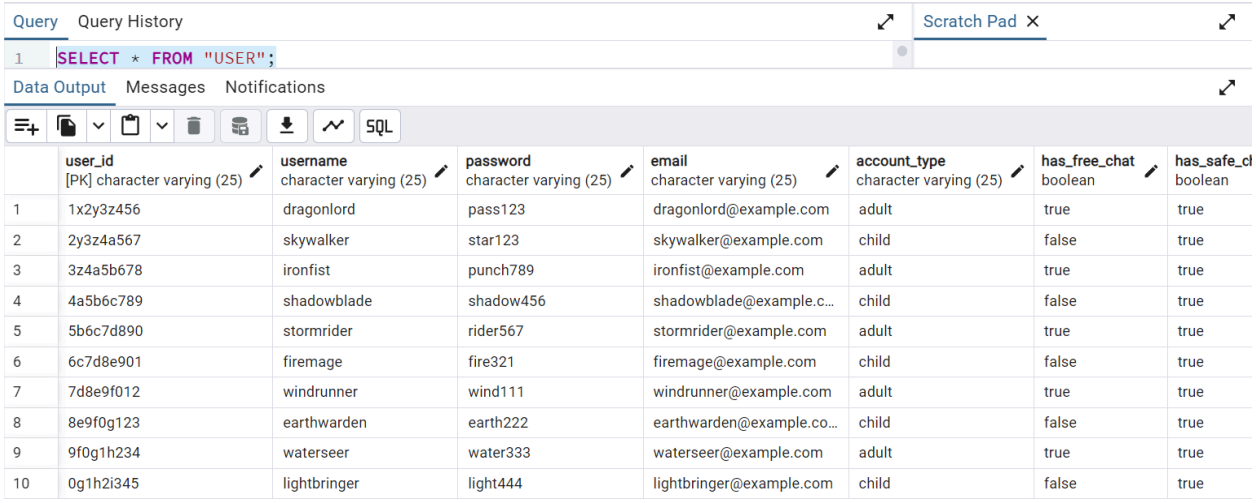
Data Output | Messages | Notifications

	transaction_id [PK] character varying (36)	listing_id integer	seller_id character varying (36)	buyer_id character varying (36)	quantity integer	total_price integer	transaction_date date
1	nfgu04bkz1	1	1a2b3c456	2b3c4d567	9	1350	2024-10-22
2	7lgij2an48	2	2b3c4d567	1a2b3c456	5	1750	2024-10-28
3	tiu0p1pmon	3	3c4d5e678	0j1k2l345	4	4000	2024-11-04
4	vt4ix91tle	4	4d5e6f789	5e6f7g890	6	7200	2024-11-06
5	urgd1d5h6i	5	5e6f7g890	1a2b3c456	8	4000	2024-11-12
6	o3jc0zpf5w	6	6f7g8h901	5e6f7g890	10	2500	2024-11-23
7	nTdbGTf8Mz	7	7g8h9i012	6f7g8h901	2	400	2024-11-30
8	46onzox4ew	8	8h9i0j123	0j1k2l345	1	300	2024-12-14
9	bf9ky0a20c	9	9i0j1k234	5e6f7g890	1	950	2024-12-21
10	6UFAVp7Tok	10	0j1k2l345	7g8h9i012	4	200	2024-12-24

- [15 POINTS] 3.4. Database Query Execution on your Normalized Database (from inside your SQL client): Use an SQL platform to provide sample executions for each of the following operations on each table of your normalized database: o Query – to perform operations such as list employees earning more than 150K per year o Insert – to add new tuple(s), and/or field(s) to your table(s) o Delete - to remove tuple(s), and/or field(s) from your table(s) o Update - to modify tuple(s), and/or field(s) from your table(s) **Provide a screenshot of each operation and each resulting table in your report exactly in this section.**

USER Table

1. **Query**



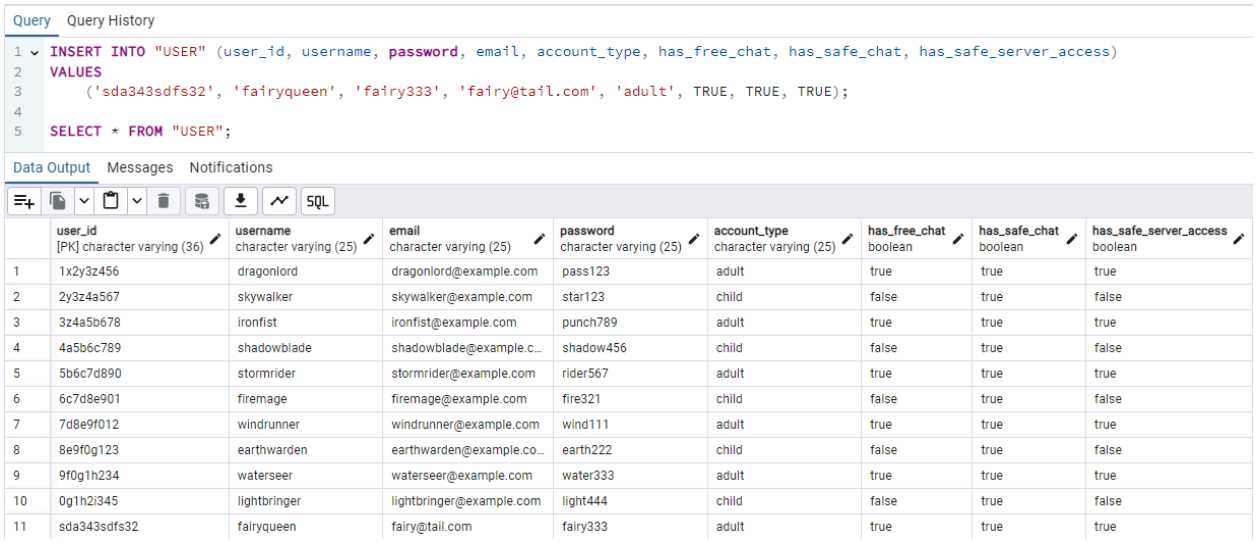
Query | Query History | Scratch Pad

```
SELECT * FROM "USER";
```

Data Output | Messages | Notifications

	user_id [PK] character varying (25)	username character varying (25)	password character varying (25)	email character varying (25)	account_type character varying (25)	has_free_chat boolean	has_safe_ch boolean
1	1x2y3z456	dragonlord	pass123	dragonlord@example.com	adult	true	true
2	2y3z4a567	skywalker	star123	skywalker@example.com	child	false	true
3	3z4a5b678	ironfist	punch789	ironfist@example.com	adult	true	true
4	4a5b6c789	shadowblade	shadow456	shadowblade@example.c...	child	false	true
5	5b6c7d890	stormrider	rider567	stormrider@example.com	adult	true	true
6	6c7d8e901	firemage	fire321	firemage@example.com	child	false	true
7	7d8e9f012	windrunner	wind111	windrunner@example.com	adult	true	true
8	8e9f0g123	earthwarden	earth222	earthwarden@example.co...	child	false	true
9	9f0g1h234	waterseer	water333	waterseer@example.com	adult	true	true
10	0g1h2i345	lightbringer	light444	lightbringer@example.com	child	false	true

2. **Insert**



Query | Query History

```
INSERT INTO "USER" (user_id, username, password, email, account_type, has_free_chat, has_safe_chat, has_safe_server_access)
VALUES
    ('sda343sdfs32', 'fairyqueen', 'fairy333', 'fairy@tail.com', 'adult', TRUE, TRUE, TRUE);

SELECT * FROM "USER";
```

Data Output | Messages | Notifications

	user_id [PK] character varying (36)	username character varying (25)	email character varying (25)	password character varying (25)	account_type character varying (25)	has_free_chat boolean	has_safe_chat boolean	has_safe_server_access boolean
1	1x2y3z456	dragonlord	dragonlord@example.com	pass123	adult	true	true	true
2	2y3z4a567	skywalker	skywalker@example.com	star123	child	false	true	false
3	3z4a5b678	ironfist	ironfist@example.com	punch789	adult	true	true	true
4	4a5b6c789	shadowblade	shadowblade@example.c...	shadow456	child	false	true	false
5	5b6c7d890	stormrider	stormrider@example.com	rider567	adult	true	true	true
6	6c7d8e901	firemage	firemage@example.com	fire321	child	false	true	false
7	7d8e9f012	windrunner	windrunner@example.com	wind111	adult	true	true	true
8	8e9f0g123	earthwarden	earthwarden@example.co...	earth222	child	false	true	false
9	9f0g1h234	waterseer	waterseer@example.com	water333	adult	true	true	true
10	0g1h2i345	lightbringer	lightbringer@example.com	light444	child	false	true	false
11	sda343sdfs32	fairyqueen	fairy@tail.com	fairy333	adult	true	true	true

3. Update

Query | Query History

```
1  UPDATE "USER"
2  SET password = 'sparkle791'
3  WHERE user_id = 'sda343sdfs32';
4
5  SELECT * FROM "USER";
```

Data Output | Messages | Notifications

	user_id [PK] character varying (36)	username character varying (25)	email character varying (25)	password character varying (25)	account_type character varying (25)	has_free_chat boolean	has_safe_chat boolean	has_safe_server_access boolean
1	1x2y3z456	dragonlord	dragonlord@example.com	pass123	adult	true	true	true
2	2y3z4a567	skywalker	skywalker@example.com	star123	child	false	true	false
3	3z4a5b678	ironfist	ironfist@example.com	punch789	adult	true	true	true
4	4a5b6c789	shadowblade	shadowblade@example.c...	shadow456	child	false	true	false
5	5b6c7d890	stormrider	stormrider@example.com	rider567	adult	true	true	true
6	6c7d8e901	firemage	firemage@example.com	fire321	child	false	true	false
7	7d8e9f012	windrunner	windrunner@example.com	wind111	adult	true	true	true
8	8e9f0g123	earthwarden	earthwarden@example.co...	earth222	child	false	true	false
9	9f0g1h234	waterseer	waterseer@example.com	water333	adult	true	true	true
10	0g1h2i345	lightbringer	lightbringer@example.com	light444	child	false	true	false
11	sda343sdfs32	fairyqueen	fairy@tail.com	sparkle791	adult	true	true	true

4. Delete

Query | Query History

```
1  DELETE FROM "USER"
2  WHERE user_id = 'sda343sdfs32';
3
4  SELECT * FROM "USER";
```

Data Output | Messages | Notifications

	user_id [PK] character varying (36)	username character varying (25)	email character varying (25)	password character varying (25)	account_type character varying (25)	has_free_chat boolean	has_safe_chat boolean	has_safe_server_access boolean
1	1x2y3z456	dragonlord	dragonlord@example.com	pass123	adult	true	true	true
2	2y3z4a567	skywalker	skywalker@example.com	star123	child	false	true	false
3	3z4a5b678	ironfist	ironfist@example.com	punch789	adult	true	true	true
4	4a5b6c789	shadowblade	shadowblade@example.c...	shadow456	child	false	true	false
5	5b6c7d890	stormrider	stormrider@example.com	rider567	adult	true	true	true
6	6c7d8e901	firemage	firemage@example.com	fire321	child	false	true	false
7	7d8e9f012	windrunner	windrunner@example.com	wind111	adult	true	true	true
8	8e9f0g123	earthwarden	earthwarden@example.co...	earth222	child	false	true	false
9	9f0g1h234	waterseer	waterseer@example.com	water333	adult	true	true	true
10	0g1h2i345	lightbringer	lightbringer@example.com	light444	child	false	true	false

CLASS Table

1. **Query**

Query | Query History

```
1  SELECT * FROM "CLASS";
```

Data Output | Messages | Notifications

	class_id [PK] character varying (36)	class_name character varying (25)	class_description character varying (25)	class_role character varying (25)
1	1m2n3o456	Warrior	Frontline fighter	Tank
2	2n3o4p567	Rogue	Stealthy assassin	DPS
3	3o4p5q678	Mage	Master of magic	DPS
4	4p5q6r789	Cleric	Healer of the wounded	Healer
5	5q6r7s890	Paladin	Holy warrior	Tank
6	6r7s8t901	Ranger	Expert marksman	DPS
7	7s8t9u012	Druid	Nature spellcaster	Healer
8	8t9u0v123	Monk	Martial artist	DPS
9	9u0v1w234	Bard	Musical supporter	Support
10	0v1w2x345	Necromancer	Summoner of the dead	DPS

2. Insert

Query | Query History

```
1  INSERT INTO
2      "CLASS" (class_id, class_name, class_description, class_role)
3  VALUES
4      ('12odk323eo', 'Hero', 'Savior of the world', 'DPS');
5
6  SELECT * FROM "CLASS";
```

Data Output | Messages | Notifications

	class_id [PK] character varying (36)	class_name character varying (25)	class_description character varying (25)	class_role character varying (25)
1	1m2n3o456	Warrior	Frontline fighter	Tank
2	2n3o4p567	Rogue	Stealthy assassin	DPS
3	3o4p5q678	Mage	Master of magic	DPS
4	4p5q6r789	Cleric	Healer of the wounded	Healer
5	5q6r7s890	Paladin	Holy warrior	Tank
6	6r7s8t901	Ranger	Expert marksman	DPS
7	7s8t9u012	Druid	Nature spellcaster	Healer
8	8t9u0v123	Monk	Martial artist	DPS
9	9u0v1w234	Bard	Musical supporter	Support
10	0v1w2x345	Necromancer	Summoner of the dead	DPS
11	12odk323eo	Hero	Savior of the world	DPS

3. **Update**

Query | Query History

```
1  UPDATE "CLASS"
2  SET class_description = 'Heroic savior'
3  WHERE class_name = 'Hero';
4
5  SELECT * FROM "CLASS";
```

Data Output | Messages | Notifications

	class_id [PK] character varying (36)	class_name character varying (25)	class_description character varying (25)	class_role character varying (25)
1	1m2n3o456	Warrior	Frontline fighter	Tank
2	2n3o4p567	Rogue	Stealthy assassin	DPS
3	3o4p5q678	Mage	Master of magic	DPS
4	4p5q6r789	Cleric	Healer of the wounded	Healer
5	5q6r7s890	Paladin	Holy warrior	Tank
6	6r7s8t901	Ranger	Expert marksman	DPS
7	7s8t9u012	Druid	Nature spellcaster	Healer
8	8t9u0v123	Monk	Martial artist	DPS
9	9u0v1w234	Bard	Musical supporter	Support
10	0v1w2x345	Necromancer	Summoner of the dead	DPS
11	12odk323eo	Hero	Heroic savior	DPS

4. **Delete**

Query | Query History

```
1  DELETE FROM "CLASS"
2  WHERE class_name = 'Hero';
3
4  SELECT * FROM "CLASS";
```

Data Output | Messages | Notifications

	class_id [PK] character varying (36)	class_name character varying (25)	class_description character varying (25)	class_role character varying (25)
1	1m2n3o456	Warrior	Frontline fighter	Tank
2	2n3o4p567	Rogue	Stealthy assassin	DPS
3	3o4p5q678	Mage	Master of magic	DPS
4	4p5q6r789	Cleric	Healer of the wounded	Healer
5	5q6r7s890	Paladin	Holy warrior	Tank
6	6r7s8t901	Ranger	Expert marksman	DPS
7	7s8t9u012	Druid	Nature spellcaster	Healer
8	8t9u0v123	Monk	Martial artist	DPS
9	9u0v1w234	Bard	Musical supporter	Support
10	0v1w2x345	Necromancer	Summoner of the dead	DPS

CHARACTER Table

1. **Query**

Query | Query History

```
1  SELECT * FROM "CHARACTER";
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	character_name character varying (25)	exp_level integer	gold_balance integer	owner_id character varying (36)	character_class character varying (36)	leader_id character varying (36)
1	1a2b3c456	Alduin	5	300	1x2y3z456	1m2n3o456	[null]
2	2b3c4d567	Lyria	3	150	2y3z4a567	2n3o4p567	1a2b3c456
3	3c4d5e678	Gorath	7	500	3z4a5b678	3o4p5q678	1a2b3c456
4	4d5e6f789	Serana	6	250	4a5b6c789	1m2n3o456	[null]
5	5e6f7g890	Theron	10	800	5b6c7d890	2n3o4p567	1a2b3c456
6	6f7g8h901	Miraak	4	200	6c7d8e901	3o4p5q678	2b3c4d567
7	7g8h9i012	Valdrin	12	1000	7d8e9f012	4p5q6r789	3c4d5e678
8	8h9i0j123	Nalya	2	100	8e9f0g123	1m2n3o456	2b3c4d567
9	9i0j1k234	Druin	9	750	9f0g1h234	4p5q6r789	3c4d5e678
10	0j1k2l345	Kael	8	600	0g1h2i345	5q6r7s890	1a2b3c456

2. Insert

Query | Query History

```
1  INSERT INTO
2      "CHARACTER" (character_id, exp_level, character_name, gold_balance, owner_id, character_class, leader_id)
3  VALUES
4      ('asj3i4ui32h', 5, 'Mongul', 300, '1x2y3z456', '1m2n3o456', NULL);
5
6  SELECT * FROM "CHARACTER";
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	character_name character varying (25)	exp_level integer	gold_balance integer	owner_id character varying (36)	character_class character varying (36)	leader_id character varying (36)
1	1a2b3c456	Alduin	5	300	1x2y3z456	1m2n3o456	[null]
2	2b3c4d567	Lyria	3	150	2y3z4a567	2n3o4p567	1a2b3c456
3	3c4d5e678	Gorath	7	500	3z4a5b678	3o4p5q678	1a2b3c456
4	4d5e6f789	Serana	6	250	4a5b6c789	1m2n3o456	[null]
5	5e6f7g890	Theron	10	800	5b6c7d890	2n3o4p567	1a2b3c456
6	6f7g8h901	Miraak	4	200	6c7d8e901	3o4p5q678	2b3c4d567
7	7g8h9i012	Valdrin	12	1000	7d8e9f012	4p5q6r789	3c4d5e678
8	8h9i0j123	Nalya	2	100	8e9f0g123	1m2n3o456	2b3c4d567
9	9i0j1k234	Druin	9	750	9f0g1h234	4p5q6r789	3c4d5e678
10	0j1k2l345	Kael	8	600	0g1h2i345	5q6r7s890	1a2b3c456
11	asj3i4ui32h	Mongul	5	300	1x2y3z456	1m2n3o456	[null]

3. Update

Query | Query History

```
1  UPDATE "CHARACTER"
2  SET character_class = '4p5q6r789'
3  WHERE character_name = 'Mongul';
4
5  SELECT * FROM "CHARACTER"
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	character_name character varying (25)	exp_level integer	gold_balance integer	owner_id character varying (36)	character_class character varying (36)	leader_id character varying (36)
1	1a2b3c456	Alduin	5	300	1x2y3z456	1m2n3o456	[null]
2	2b3c4d567	Lyria	3	150	2y3z4a567	2n3o4p567	1a2b3c456
3	3c4d5e678	Gorath	7	500	3z4a5b678	3o4p5q678	1a2b3c456
4	4d5e6f789	Serana	6	250	4a5b6c789	1m2n3o456	[null]
5	5e6f7g890	Theron	10	800	5b6c7d890	2n3o4p567	1a2b3c456
6	6f7g8h901	Miraak	4	200	6c7d8e901	3o4p5q678	2b3c4d567
7	7g8h9i012	Valdrin	12	1000	7d8e9f012	4p5q6r789	3c4d5e678
8	8h9i0j123	Nalya	2	100	8e9f0g123	1m2n3o456	2b3c4d567
9	9i0j1k234	Druin	9	750	9f0g1h234	4p5q6r789	3c4d5e678
10	0j1k2l345	Kael	8	600	0g1h2i345	5q6r7s890	1a2b3c456
11	asj3i4ui32h	Mongul	5	300	1x2y3z456	4p5q6r789	[null]

4. **Delete**

Query | Query History

```
1  DELETE FROM "CHARACTER"
2  WHERE character_name = 'Mongul';
3
4  SELECT * FROM "CHARACTER";
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	character_name character varying (25)	exp_level integer	gold_balance integer	owner_id character varying (36)	character_class character varying (36)	leader_id character varying (36)
1	1a2b3c456	Alduin	5	300	1x2y3z456	1m2n3o456	[null]
2	2b3c4d567	Lyria	3	150	2y3z4a567	2n3o4p567	1a2b3c456
3	3c4d5e678	Gorath	7	500	3z4a5b678	3o4p5q678	1a2b3c456
4	4d5e6f789	Serana	6	250	4a5b6c789	1m2n3o456	[null]
5	5e6f7g890	Theron	10	800	5b6c7d890	2n3o4p567	1a2b3c456
6	6f7g8h901	Miraak	4	200	6c7d8e901	3o4p5q678	2b3c4d567
7	7g8h9i012	Valdrin	12	1000	7d8e9f012	4p5q6r789	3c4d5e678
8	8h9i0j123	Nalya	2	100	8e9f0g123	1m2n3o456	2b3c4d567
9	9i0j1k234	Druin	9	750	9f0g1h234	4p5q6r789	3c4d5e678
10	0j1k2l345	Kael	8	600	0g1h2i345	5q6r7s890	1a2b3c456

PARTY Table

1. **Query**

Query | Query History

```
1  SELECT * FROM "PARTY";
```

Data Output | Messages | Notifications

	party_name [PK] character varying (50)	party_leader [PK] character varying (36)	party_balance integer
1	Lyrical	1a2b3c456	53422
2	Arcane	4d5e6f789	3234223
3	Dragon Slayer	2b3c4d567	1000
4	Witch Hunter	3c4d5e678	54223
5	Tuba Gang	7g8h9i012	546
6	Stroopwafel	0j1k2l345	564
7	Birds of Prey	9i0j1k234	32
8	The Fallen	6f7g8h901	7869
9	Masked Fools	5e6f7g890	435345
10	Asgard	8h9i0j123	34523

2. **Insert**

Query | Query History

```
INSERT INTO
    "PARTY" (party_name, party_leader, party_balance)
VALUES
    ('Chaotix', 'asj3i4ui32h', 50000);

SELECT * FROM "PARTY";
```

Data Output | Messages | Notifications

	party_name [PK] character varying (50)	party_leader [PK] character varying (36)	party_balance integer
1	Lyrical	1a2b3c456	53422
2	Arcane	4d5e6f789	3234223
3	Dragon Slayer	2b3c4d567	1000
4	Witch Hunter	3c4d5e678	54223
5	Tuba Gang	7g8h9i012	546
6	Stroopwafel	0j1k2l345	564
7	Birds of Prey	9i0j1k234	32
8	The Fallen	6f7g8h901	7869
9	Masked Fools	5e6f7g890	435345
10	Asgard	8h9i0j123	34523
11	Chaotix	asj3i4ui32h	50000

3. Update

Query | Query History

```
UPDATE "PARTY"
SET party_balance = party_balance * 1.1;

SELECT * FROM "PARTY";
```

Data Output | Messages | Notifications

	party_name [PK] character varying (50)	party_leader [PK] character varying (36)	party_balance integer
1	Lyrical	1a2b3c456	58764
2	Arcane	4d5e6f789	3557645
3	Dragon Slayer	2b3c4d567	1100
4	Witch Hunter	3c4d5e678	59645
5	Tuba Gang	7g8h9i012	601
6	Stroopwafel	0j1k2l345	620
7	Birds of Prey	9i0j1k234	35
8	The Fallen	6f7g8h901	8656
9	Masked Fools	5e6f7g890	478880
10	Asgard	8h9i0j123	37975
11	Chaotix	asj3i4ui32h	55000

4. **Delete**

```
Query  Query History
1 DELETE FROM "PARTY"
2 WHERE party_name = 'Chaotix';
3
4 SELECT * FROM "PARTY";
```

Data Output | Messages | Notifications

	party_name [PK] character varying (50)	party_leader [PK] character varying (36)	party_balance integer
1	Lyrical	1a2b3c456	58764
2	Arcane	4d5e6f789	3557645
3	Dragon Slayer	2b3c4d567	1100
4	Witch Hunter	3c4d5e678	59645
5	Tuba Gang	7g8h9i012	601
6	Stroopwafel	0j1k2l345	620
7	Birds of Prey	9i0j1k234	35
8	The Fallen	6f7g8h901	8656
9	Masked Fools	5e6f7g890	478880
10	Asgard	8h9i0j123	37975

CHARACTER_FRIEND Table

1. **Query**

```
Query  Query History
1 SELECT * FROM "CHARACTER_FRIEND";
```

Data Output | Messages | Notifications

	character_a_id [PK] character varying (36)	character_b_id [PK] character varying (36)
1	1a2b3c456	2b3c4d567
2	1a2b3c456	3c4d5e678
3	2b3c4d567	6f7g8h901
4	3c4d5e678	9i0j1k234
5	4d5e6f789	1a2b3c456
6	5e6f7g890	2b3c4d567
7	6f7g8h901	7g8h9i012
8	7g8h9i012	3c4d5e678
9	8h9i0j123	9i0j1k234
10	0j1k2l345	5e6f7g890

2. **Insert**

Query | Query History

```
1  INSERT INTO "CHARACTER_FRIEND" (character_a_id, character_b_id)
2  VALUES ('1a2b3c456', '4d5e6f789');
3
4  SELECT * FROM "CHARACTER_FRIEND";
```

Data Output | Messages | Notifications

	character_a_id [PK] character varying (36)	character_b_id [PK] character varying (36)
1	1a2b3c456	2b3c4d567
2	1a2b3c456	3c4d5e678
3	2b3c4d567	6f7g8h901
4	3c4d5e678	9i0j1k234
5	4d5e6f789	1a2b3c456
6	5e6f7g890	2b3c4d567
7	6f7g8h901	7g8h9i012
8	7g8h9i012	3c4d5e678
9	8h9i0j123	9i0j1k234
10	0j1k2l345	5e6f7g890
11	1a2b3c456	4d5e6f789

3. **Update**

Query | Query History

```
1  UPDATE "CHARACTER_FRIEND"
2  SET character_b_id = '9i0j1k234'
3  WHERE character_a_id = '1a2b3c456' and character_b_id = '4d5e6f789';
4
5  SELECT * FROM "CHARACTER_FRIEND";
```

Data Output | Messages | Notifications

	character_a_id [PK] character varying (36)	character_b_id [PK] character varying (36)
1	1a2b3c456	2b3c4d567
2	1a2b3c456	3c4d5e678
3	2b3c4d567	6f7g8h901
4	3c4d5e678	9i0j1k234
5	4d5e6f789	1a2b3c456
6	5e6f7g890	2b3c4d567
7	6f7g8h901	7g8h9i012
8	7g8h9i012	3c4d5e678
9	8h9i0j123	9i0j1k234
10	0j1k2l345	5e6f7g890
11	1a2b3c456	9i0j1k234

4. **Delete**

Query | Query History

```
1 DELETE FROM "CHARACTER_FRIEND"
2 WHERE character_a_id = '1a2b3c456' and character_b_id = '9i0j1k234';
3
4 SELECT * FROM "CHARACTER_FRIEND";
```

Data Output | Messages | Notifications

	character_a_id [PK] character varying (36)	character_b_id [PK] character varying (36)
1	1a2b3c456	2b3c4d567
2	1a2b3c456	3c4d5e678
3	2b3c4d567	6f7g8h901
4	3c4d5e678	9i0j1k234
5	4d5e6f789	1a2b3c456
6	5e6f7g890	2b3c4d567
7	6f7g8h901	7g8h9i012
8	7g8h9i012	3c4d5e678
9	8h9i0j123	9i0j1k234
10	0j1k2l345	5e6f7g890

ITEM_CATEGORY Table

1. **Query**

Query | Query History

```
1 SELECT * FROM "ITEM_CATEGORY";
```

Data Output | Messages | Notifications

	category_id [PK] integer	item_category character varying (25)
1	1	Consumable
2	2	Weapon
3	3	Armor
4	4	Accessory
5	5	Shield
6	6	Headgear

2. **Insert**

Query | Query History

```
1  INSERT INTO "ITEM_CATEGORY" (item_category) VALUES
2      ('Relic');
3
4  SELECT * FROM "ITEM_CATEGORY";
```

Data Output | Messages | Notifications

SQL

	category_id [PK] integer	item_category character varying (25)
1	1	Consumable
2	2	Weapon
3	3	Armor
4	4	Accessory
5	5	Shield
6	6	Headgear
7	7	Relic

3. **Update**

Query | Query History

```
1  UPDATE "ITEM_CATEGORY"
2  SET item_category = 'Miscellaneous'
3  WHERE item_category = 'Relic';
4
5  SELECT * FROM "ITEM_CATEGORY";
```

Data Output | Messages | Notifications

SQL

	category_id [PK] integer	item_category character varying (25)
1	1	Consumable
2	2	Weapon
3	3	Armor
4	4	Accessory
5	5	Shield
6	6	Headgear
7	7	Miscellaneous

4. **Delete**

Query | Query History

```
1  DELETE FROM "ITEM_CATEGORY"
2  WHERE item_category = 'Miscellaneous';
3
4  SELECT * FROM "ITEM_CATEGORY";
```

Data Output | Messages | Notifications

	category_id [PK] integer	item_category character varying (25)
1	1	Consumable
2	2	Weapon
3	3	Armor
4	4	Accessory
5	5	Shield
6	6	Headgear

ITEM_RARITY Table

1. **Query**

Query | Query History

```
1  SELECT * FROM "ITEM_RARITY";
```

Data Output | Messages | Notifications

	rarity_id [PK] integer	item_rarity character varying (25)
1	1	Common
2	2	Uncommon
3	3	Rare
4	4	Epic
5	5	Legendary

2. **Insert**

Query Query History

```
1 INSERT INTO "ITEM_RARITY" (item_rarity) VALUES
2     ('Mythical');
3
4 SELECT * FROM "ITEM_RARITY";
```

Data Output Messages Notifications

SQL

	rarity_id [PK] integer	item_rarity character varying (25)
1	1	Common
2	2	Uncommon
3	3	Rare
4	4	Epic
5	5	Legendary
6	6	Mythical

3. **Update**

Query Query History

```
1 UPDATE "ITEM_RARITY"
2 SET item_rarity = 'Mythic'
3 WHERE item_rarity = 'Mythical';
4
5 SELECT * FROM "ITEM_RARITY";
```

Data Output Messages Notifications

SQL

	rarity_id [PK] integer	item_rarity character varying (25)
1	1	Common
2	2	Uncommon
3	3	Rare
4	4	Epic
5	5	Legendary
6	6	Mythic

4. **Delete**

Query | Query History

```
1  DELETE FROM "ITEM_RARITY"
2  WHERE item_rarity = 'Mythic';
3
4  SELECT * FROM "ITEM_RARITY";
```

Data Output | Messages | Notifications

SQL

	rarity_id [PK] integer	item_rarity character varying (25)
1	1	Common
2	2	Uncommon
3	3	Rare
4	4	Epic
5	5	Legendary

ITEM Table

1. **Query**

Query | Query History

```
1  SELECT * FROM "ITEM";
```

Data Output | Messages | Notifications

	item_id [PK] integer	item_name character varying (50)	category_id integer	rarity_id integer	item_price numeric (10)
1	1	Health Potion	1	1	50
2	2	Greater Health Potion	1	2	100
3	3	Mana Potion	1	1	50
4	4	Elixir of Fortitude	1	3	200
5	5	Steel Sword	2	3	400
6	6	Iron Dagger	2	2	250
7	7	Fire Staff	2	4	700
8	8	Leather Armor	3	1	150
9	9	Plate Armor	3	3	600
10	10	Ring of Protection	4	3	350
11	11	Necklace of Wisdom	4	4	800
12	12	Wooden Shield	5	1	100
13	13	Divine Shield	5	5	1200
14	14	Wizard Hat	6	3	500
15	15	Crown of the Bard King	6	5	1000
16	16	Warhammer of the Iron Titan	2	4	10000
17	17	Sigil of the Saint	2	5	9000
18	18	Dragonbane Wand	2	3	1500

2. **Insert**

Query | Query History

```
1  INSERT INTO "ITEM" (item_name, category_id, rarity_id, item_price
2      ('Titanic Experience Bottle', 1, 1, 50000);
3
4  SELECT * FROM "ITEM";
```

Data Output | Messages | Notifications

	item_id [PK] integer	item_name character varying (50)	category_id integer	rarity_id integer	item_price numeric (10)
1	1	Health Potion	1	1	50
2	2	Greater Health Potion	1	2	100
3	3	Mana Potion	1	1	50
4	4	Elixir of Fortitude	1	3	200
5	5	Steel Sword	2	3	400
6	6	Iron Dagger	2	2	250
7	7	Fire Staff	2	4	700
8	8	Leather Armor	3	1	150
9	9	Plate Armor	3	3	600
10	10	Ring of Protection	4	3	350
11	11	Necklace of Wisdom	4	4	800
12	12	Wooden Shield	5	1	100
13	13	Divine Shield	5	5	1200
14	14	Wizard Hat	6	3	500
15	15	Crown of the Bard King	6	5	1000
16	16	Warhammer of the Iron Titan	2	4	10000
17	17	Sigil of the Saint	2	5	9000
18	18	Dragonbane Wand	2	3	1500
19	19	Titanic Experience Bottle	1	1	50000

3. **Update**

Query | Query History

```
1  UPDATE "ITEM"
2  SET rarity_id = 2
3  WHERE item_name = 'Titanic Experience Bottle';
4
5  SELECT * FROM "ITEM";
```

Data Output | Messages | Notifications

	item_id [PK] integer	item_name character varying (50)	category_id integer	rarity_id integer	item_price numeric (10)
1	1	Health Potion	1	1	50
2	2	Greater Health Potion	1	2	100
3	3	Mana Potion	1	1	50
4	4	Elixir of Fortitude	1	3	200
5	5	Steel Sword	2	3	400
6	6	Iron Dagger	2	2	250
7	7	Fire Staff	2	4	700
8	8	Leather Armor	3	1	150
9	9	Plate Armor	3	3	600
10	10	Ring of Protection	4	3	350
11	11	Necklace of Wisdom	4	4	800
12	12	Wooden Shield	5	1	100
13	13	Divine Shield	5	5	1200
14	14	Wizard Hat	6	3	500
15	15	Crown of the Bard King	6	5	1000
16	16	Warhammer of the Iron Titan	2	4	10000
17	17	Sigil of the Saint	2	5	9000
18	18	Dragonbane Wand	2	3	1500
19	19	Titanic Experience Bottle	1	2	50000

4. **Delete**

Query | Query History

```
1  DELETE FROM "ITEM"
2  WHERE item_name = 'Titanic Experience Bottle';
3
4  SELECT * FROM "ITEM";
```

Data Output | Messages | Notifications

	item_id [PK] integer	item_name character varying (50)	category_id integer	rarity_id integer	item_price numeric (10)
1	1	Health Potion	1	1	50
2	2	Greater Health Potion	1	2	100
3	3	Mana Potion	1	1	50
4	4	Elixir of Fortitude	1	3	200
5	5	Steel Sword	2	3	400
6	6	Iron Dagger	2	2	250
7	7	Fire Staff	2	4	700
8	8	Leather Armor	3	1	150
9	9	Plate Armor	3	3	600
10	10	Ring of Protection	4	3	350
11	11	Necklace of Wisdom	4	4	800
12	12	Wooden Shield	5	1	100
13	13	Divine Shield	5	5	1200
14	14	Wizard Hat	6	3	500
15	15	Crown of the Bard King	6	5	1000
16	16	Warhammer of the Iron Titan	2	4	10000
17	17	Sigil of the Saint	2	5	9000
18	18	Dragonbane Wand	2	3	1500

ITEM_CLASS Table

1. **Query**

Query Query History

```
1  SELECT * FROM "ITEM_CLASS";
```

Data Output Messages Notifications

	item_id [PK] integer	class_id [PK] character varying (36)
1	1	0v1w2x345
2	1	1m2n3o456
3	1	2n3o4p567
4	1	3o4p5q678
5	1	4p5q6r789
6	1	5q6r7s890
7	1	6r7s8t901
8	1	7s8t9u012
9	1	8t9u0v123
10	1	9u0v1w234

2. **Insert**

Query Query History

```
1  INSERT INTO "ITEM_CLASS" (item_id, class_id) VALUES
2      (21, '7s8t9u012'), (21, '8t9u0v123');
3
4  SELECT * FROM "ITEM_CLASS"
5  WHERE item_id = 21;
```

Data Output Messages Notifications

	item_id [PK] integer	class_id [PK] character varying (36)
1	21	7s8t9u012
2	21	8t9u0v123

3. **Update**

Query | Query History

```
1  UPDATE "ITEM_CLASS"
2  SET class_id = '3o4p5q678'
3  WHERE item_id = 21 and class_id = '7s8t9u012';
4
5  SELECT * FROM "ITEM_CLASS"
6  WHERE item_id = 21;
```

Data Output | Messages | Notifications

	item_id [PK] integer	class_id [PK] character varying (36)
1	21	8t9u0v123
2	21	3o4p5q678

4. **Delete**

Query | Query History

```
2  WHERE item_id = 21;
3
4  SELECT * FROM "ITEM_CLASS"
5  ORDER BY item_id DESC;
```

Data Output | Messages | Notifications

	item_id [PK] integer	class_id [PK] character varying (36)
1	18	0v1w2x345
2	18	3o4p5q678
3	17	5q6r7s890
4	17	3o4p5q678
5	16	5q6r7s890
6	15	9u0v1w234

LISTING Table

1. Query

Query | Query History

```
1  SELECT * FROM "LISTING";
```

Data Output | Messages | Notifications

	listing_id [PK] integer	character_id character varying (36)	item_id integer	quantity integer	listing_date date	is_active boolean	sale_price numeric (10)
1	2	2b3c4d567	2	5	2024-08-23	true	350
2	3	3c4d5e678	3	1	2024-10-01	false	1000
3	4	4d5e6f789	4	2	2024-09-10	true	1200
4	5	5e6f7g890	5	3	2024-08-30	false	500
5	6	6f7g8h901	6	7	2024-09-25	true	250
6	7	7g8h9i012	7	4	2024-09-02	true	200
7	8	8h9i0j123	8	12	2024-07-20	false	300
8	9	9i0j1k234	9	6	2024-08-29	true	950
9	10	0j1k2l345	10	1	2024-09-18	false	50
10	12	2b3c4d567	12	6	2024-07-28	false	180
11	13	3c4d5e678	13	8	2024-09-30	true	600
12	14	4d5e6f789	14	9	2024-08-25	true	1000
13	15	5e6f7g890	15	10	2024-09-01	false	800

2. Insert

Query | Query History

```
1  INSERT INTO "LISTING" (character_id, item_id, quantity, listing_date, is_active, sale_price) VALUES
2  ('1a2b3c456', 18, 10, '2024-09-15', TRUE, 150);
3
4  SELECT * FROM "LISTING";
```

Data Output | Messages | Notifications

	listing_id [PK] integer	character_id character varying (36)	item_id integer	quantity integer	listing_date date	is_active boolean	sale_price numeric (10)
1	2	2b3c4d567	2	5	2024-08-23	true	350
2	3	3c4d5e678	3	1	2024-10-01	false	1000
3	4	4d5e6f789	4	2	2024-09-10	true	1200
4	5	5e6f7g890	5	3	2024-08-30	false	500
5	6	6f7g8h901	6	7	2024-09-25	true	250
6	7	7g8h9i012	7	4	2024-09-02	true	200
7	8	8h9i0j123	8	12	2024-07-20	false	300
8	9	9i0j1k234	9	6	2024-08-29	true	950
9	10	0j1k2l345	10	1	2024-09-18	false	50
10	12	2b3c4d567	12	6	2024-07-28	false	180
11	13	3c4d5e678	13	8	2024-09-30	true	600
12	14	4d5e6f789	14	9	2024-08-25	true	1000
13	15	5e6f7g890	15	10	2024-09-01	false	800
14	22	1a2b3c456	18	10	2024-09-15	true	150

3. Update

Query | Query History

```
1  UPDATE "LISTING"
2  SET item_id = 15
3  WHERE listing_id = 22;
4
5  SELECT * FROM "LISTING"
```

Data Output | Messages | Notifications

SQL

	listing_id [PK] integer	character_id character varying (36)	item_id integer	quantity integer	listing_date date	is_active boolean	sale_price numeric (10)
1	2	2b3c4d567	2	5	2024-08-23	true	350
2	3	3c4d5e678	3	1	2024-10-01	false	1000
3	4	4d5e6f789	4	2	2024-09-10	true	1200
4	5	5e6f7g890	5	3	2024-08-30	false	500
5	6	6f7g8h901	6	7	2024-09-25	true	250
6	7	7g8h9i012	7	4	2024-09-02	true	200
7	8	8h9i0j123	8	12	2024-07-20	false	300
8	9	9i0j1k234	9	6	2024-08-29	true	950
9	10	0j1k2l345	10	1	2024-09-18	false	50
10	12	2b3c4d567	12	6	2024-07-28	false	180
11	13	3c4d5e678	13	8	2024-09-30	true	600
12	14	4d5e6f789	14	9	2024-08-25	true	1000
13	15	5e6f7g890	15	10	2024-09-01	false	800
14	22	1a2b3c456	15	10	2024-09-15	true	150

4. Delete

Query | Query History

```
1  DELETE FROM "LISTING"
2  WHERE listing_id = 22;
3
4  SELECT * FROM "LISTING";
```

Data Output | Messages | Notifications

	listing_id [PK] integer	character_id character varying (36)	item_id integer	quantity integer	listing_date date	is_active boolean	sale_price numeric (10)
1	2	2b3c4d567	2	5	2024-08-23	true	350
2	3	3c4d5e678	3	1	2024-10-01	false	1000
3	4	4d5e6f789	4	2	2024-09-10	true	1200
4	5	5e6f7g890	5	3	2024-08-30	false	500
5	6	6f7g8h901	6	7	2024-09-25	true	250
6	7	7g8h9i012	7	4	2024-09-02	true	200
7	8	8h9i0j123	8	12	2024-07-20	false	300
8	9	9i0j1k234	9	6	2024-08-29	true	950
9	10	0j1k2l345	10	1	2024-09-18	false	50
10	12	2b3c4d567	12	6	2024-07-28	false	180
11	13	3c4d5e678	13	8	2024-09-30	true	600
12	14	4d5e6f789	14	9	2024-08-25	true	1000
13	15	5e6f7g890	15	10	2024-09-01	false	800

IN_INVENTORY Table

1. Query

Query | Query History

```
1  SELECT * FROM "IN_INVENTORY";
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	item_id [PK] integer	quantity integer
1	0j1k2l345	1	5
2	0j1k2l345	5	1
3	1a2b3c456	2	3
4	1a2b3c456	8	1
5	2b3c4d567	6	2
6	2b3c4d567	10	1
7	3c4d5e678	7	1
8	3c4d5e678	3	4
9	4d5e6f789	4	2
10	4d5e6f789	12	1
11	5e6f7g890	9	1
12	5e6f7g890	13	1
13	6f7g8h901	15	1
14	7g8h9i012	14	1
15	8h9i0j123	11	1
16	9i0j1k234	1	6
17	9i0j1k234	5	1
18	8h9i0j123	16	11
19	0j1k2l345	17	10000
20	8h9i0j123	18	100

2. **Insert**

Query | Query History

```
1  INSERT INTO "IN_INVENTORY" (character_id, item_id, quantity) VALUES
2  ('0j1k2l345', 18, 500);
3
4  SELECT * FROM "IN_INVENTORY";
```

Data Output | Messages | Notifications

SQL

	character_id [PK] character varying (36)	item_id [PK] integer	quantity integer
1	0j1k2l345	1	5
2	0j1k2l345	5	1
3	1a2b3c456	2	3
4	1a2b3c456	8	1
5	2b3c4d567	6	2
6	2b3c4d567	10	1
7	3c4d5e678	7	1
8	3c4d5e678	3	4
9	4d5e6f789	4	2
10	4d5e6f789	12	1
11	5e6f7g890	9	1
12	5e6f7g890	13	1
13	6f7g8h901	15	1
14	7g8h9i012	14	1
15	8h9i0j123	11	1
16	9i0j1k234	1	6
17	9i0j1k234	5	1
18	8h9i0j123	16	11
19	0j1k2l345	17	10000
20	8h9i0j123	18	100
21	0j1k2l345	18	500

3. **Update**

Query | Query History

```
1  UPDATE "IN_INVENTORY"
2  SET quantity = 10
3  WHERE character_id = '0j1k2l345';
4
5  SELECT * FROM "IN_INVENTORY"
6  WHERE character_id = '0j1k2l345';
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	item_id [PK] integer	quantity integer
1	0j1k2l345	1	10
2	0j1k2l345	5	10
3	0j1k2l345	17	10
4	0j1k2l345	18	10

4. **Delete**

Query | Query History

```
1  DELETE FROM "IN_INVENTORY"
2  WHERE character_id = '0j1k2l345' and item_id = 18;
3
4  SELECT * FROM "IN_INVENTORY"
5  WHERE character_id = '0j1k2l345';
```

Data Output | Messages | Notifications

	character_id [PK] character varying (36)	item_id [PK] integer	quantity integer
1	0j1k2l345	1	10
2	0j1k2l345	5	10
3	0j1k2l345	17	10

TRANSACTION Table

1. Query

Query | Query History

```
1  SELECT * FROM "TRANSACTION";
```

Data Output | Messages | Notifications

	transaction_id [PK] character varying (36)	listing_id integer	seller_id character varying (36)	buyer_id character varying (36)	quantity integer	total_price integer	transaction_date date
1	7lgij2an48	2	2b3c4d567	1a2b3c456	5	1750	2024-10-28
2	tiu0p1pmon	3	3c4d5e678	0j1k2l345	4	4000	2024-11-04
3	vt4ix91tle	4	4d5e6f789	5e6f7g890	6	7200	2024-11-06
4	urgd1d5h6i	5	5e6f7g890	1a2b3c456	8	4000	2024-11-12
5	o3jc0zpf5w	6	6f7g8h901	5e6f7g890	10	2500	2024-11-23
6	nTdbGTf8Mz	7	7g8h9i012	6f7g8h901	2	400	2024-11-30
7	46onzox4ew	8	8h9i0j123	0j1k2l345	1	300	2024-12-14
8	bf9ky0a20c	9	9i0j1k234	5e6f7g890	1	950	2024-12-21
9	6UFAVp7Tok	10	0j1k2l345	7g8h9i012	4	200	2024-12-24
10	sdf4w5rwer2	8	1a2b3c456	2b3c4d567	9	2700	2024-11-16

2. Insert

Query | Query History

```
1  INSERT INTO
2      "TRANSACTION" (transaction_id, listing_id, seller_id, buyer_id, quantity, transaction_date)
3  VALUES
4      ('asq3e3q4324', 5, '1a2b3c456', '2b3c4d567', 200, '2024-11-16');
5
6  SELECT * FROM "TRANSACTION";
```

Data Output | Messages | Notifications

	transaction_id [PK] character varying (36)	listing_id integer	seller_id character varying (36)	buyer_id character varying (36)	quantity integer	total_price integer	transaction_date date
1	7lgij2an48	2	2b3c4d567	1a2b3c456	5	1750	2024-10-28
2	tiu0p1pmon	3	3c4d5e678	0j1k2l345	4	4000	2024-11-04
3	vt4ix91tle	4	4d5e6f789	5e6f7g890	6	7200	2024-11-06
4	urgd1d5h6i	5	5e6f7g890	1a2b3c456	8	4000	2024-11-12
5	o3jc0zpf5w	6	6f7g8h901	5e6f7g890	10	2500	2024-11-23
6	nTdbGTf8Mz	7	7g8h9i012	6f7g8h901	2	400	2024-11-30
7	46onzox4ew	8	8h9i0j123	0j1k2l345	1	300	2024-12-14
8	bf9ky0a20c	9	9i0j1k234	5e6f7g890	1	950	2024-12-21
9	6UFAVp7Tok	10	0j1k2l345	7g8h9i012	4	200	2024-12-24
10	sdf4w5rwer2	8	1a2b3c456	2b3c4d567	9	2700	2024-11-16
11	asq3e3q4324	5	1a2b3c456	2b3c4d567	200	100000	2024-11-16

3. Update

Query | Query History

```
1  UPDATE "TRANSACTION"
2  SET quantity = 10
3  WHERE transaction_id = 'asq3e3q4324';
4
5  SELECT * FROM "TRANSACTION";
```

Data Output | Messages | Notifications

	transaction_id [PK] character varying (36)	listing_id integer	seller_id character varying (36)	buyer_id character varying (36)	quantity integer	total_price integer	transaction_date date
1	7lgij2an48	2	2b3c4d567	1a2b3c456	5	1750	2024-10-28
2	tiu0p1pmon	3	3c4d5e678	0j1k2l345	4	4000	2024-11-04
3	vt4ix91tle	4	4d5e6f789	5e6f7g890	6	7200	2024-11-06
4	urgd1d5h6i	5	5e6f7g890	1a2b3c456	8	4000	2024-11-12
5	o3jc0zpf5w	6	6f7g8h901	5e6f7g890	10	2500	2024-11-23
6	nTdbGTf8Mz	7	7g8h9i012	6f7g8h901	2	400	2024-11-30
7	46onzox4ew	8	8h9i0j123	0j1k2l345	1	300	2024-12-14
8	bf9ky0a20c	9	9i0j1k234	5e6f7g890	1	950	2024-12-21
9	6UFAVp7Tok	10	0j1k2l345	7g8h9i012	4	200	2024-12-24
10	sdf4w5rwer2	8	1a2b3c456	2b3c4d567	9	2700	2024-11-16
11	asq3e3q4324	5	1a2b3c456	2b3c4d567	10	5000	2024-11-16

4. Delete

Query | Query History

```
1  DELETE FROM "TRANSACTION"
2  WHERE transaction_id = 'asq3e3q4324';
3
4  SELECT * FROM "TRANSACTION";
```

Data Output | Messages | Notifications

	transaction_id [PK] character varying (36)	listing_id integer	seller_id character varying (36)	buyer_id character varying (36)	quantity integer	total_price integer	transaction_date date
1	7lgij2an48	2	2b3c4d567	1a2b3c456	5	1750	2024-10-28
2	tiu0p1pmon	3	3c4d5e678	0j1k2l345	4	4000	2024-11-04
3	vt4ix91tle	4	4d5e6f789	5e6f7g890	6	7200	2024-11-06
4	urgd1d5h6i	5	5e6f7g890	1a2b3c456	8	4000	2024-11-12
5	o3jc0zpf5w	6	6f7g8h901	5e6f7g890	10	2500	2024-11-23
6	nTdbGTf8Mz	7	7g8h9i012	6f7g8h901	2	400	2024-11-30
7	46onzox4ew	8	8h9i0j123	0j1k2l345	1	300	2024-12-14
8	bf9ky0a20c	9	9i0j1k234	5e6f7g890	1	950	2024-12-21
9	6UFAVp7Tok	10	0j1k2l345	7g8h9i012	4	200	2024-12-24
10	sdf4w5rwer2	8	1a2b3c456	2b3c4d567	9	2700	2024-11-16

3.5. Create View: Use minimum one CREATE VIEW statement in your normalized database to implement a view based on your specific database design. Please indicate what this view is for. Provide a screenshot of your view(s), as well as each resulting table in your report exactly in this section.

CHARACTER_TRANSACTION View

- Purpose: Shows the purchase history of the characters, i.e. the item they purchased, the amount purchased, the selling price, the total price they paid, and the date purchased

```
-- Create a view of character transaction showing
-- the item a character bought and the date
-- it was purchased.
CREATE OR REPLACE VIEW "CHARACTER_TRANSACTION"
AS SELECT
    C.character_name AS character,
    I.item_name AS purchased_item,
    T.quantity AS amount_purchased,
    L.sale_price AS selling_price,
    T.total_price AS total_price,
    T.transaction_date AS date_purchased
FROM
    "CHARACTER" AS C
    JOIN "TRANSACTION" AS T ON C.character_id = T.buyer_id
    JOIN "LISTING" AS L ON T.listing_id = L.listing_id
    JOIN "ITEM" AS I ON L.item_id = I.item_id;
```

-

Query | Query History

```
1  SELECT * FROM "CHARACTER_TRANSACTION";
```

Data Output | Messages | Notifications

	character character varying (25)	purchased_item character varying (50)	amount_purchased integer	selling_price numeric (10)	total_price integer	date_purchased date
1	Alduin	Greater Health Potion	5	350	1750	2024-10-28
2	Kael	Mana Potion	4	1000	4000	2024-11-04
3	Theron	Elixir of Fortitude	6	1200	7200	2024-11-06
4	Alduin	Steel Sword	8	500	4000	2024-11-12
5	Theron	Iron Dagger	10	250	2500	2024-11-23
6	Miraak	Fire Staff	2	200	400	2024-11-30
7	Kael	Leather Armor	1	300	300	2024-12-14
8	Theron	Plate Armor	1	950	950	2024-12-21
9	Valdrin	Ring of Protection	4	50	200	2024-12-24
10	Lyria	Leather Armor	9	300	2700	2024-11-16

CHARACTER_INVENTORY View

- Purpose: Shows the items that are inside a character's inventory, as well as their category and rarity

```
-- Create a view of character inventory
-- that displays the character name, their
-- inventory, and the quantity of each item
CREATE OR REPLACE VIEW "CHARACTER_INVENTORY"
AS SELECT
    C.character_name AS character,
    I.item_name AS item,
    V.quantity AS quantity,
    IC.item_category AS category,
    IR.item_rarity as rarity
FROM
    "CHARACTER" AS C
    JOIN "IN_INVENTORY" AS V ON C.character_id = V.character_id
    JOIN "ITEM" I ON V.item_id = I.item_id
    JOIN "ITEM_CATEGORY" IC ON I.category_id = IC.category_id
    JOIN "ITEM_RARITY" IR ON I.rarity_id = IR.rarity_id;
```

-

```
Query  Query History
1  SELECT * FROM "CHARACTER_INVENTORY";
```

Data Output | Messages | Notifications

	character character varying (25)	item character varying (50)	quantity integer	category character varying (25)	rarity character varying (25)
1	Alduin	Greater Health Potion	3	Consumable	Uncommon
2	Alduin	Leather Armor	1	Armor	Common
3	Lyria	Iron Dagger	2	Weapon	Uncommon
4	Lyria	Ring of Protection	1	Accessory	Rare
5	Gorath	Fire Staff	1	Weapon	Epic
6	Gorath	Mana Potion	4	Consumable	Common
7	Serana	Elixir of Fortitude	2	Consumable	Rare
8	Serana	Wooden Shield	1	Shield	Common
9	Theron	Plate Armor	1	Armor	Rare
10	Theron	Divine Shield	1	Shield	Legendary
11	Miraak	Crown of the Bard King	1	Headgear	Legendary
12	Valdrin	Wizard Hat	1	Headgear	Rare
13	Nalya	Necklace of Wisdom	1	Accessory	Epic
14	Druin	Health Potion	6	Consumable	Common
15	Druin	Steel Sword	1	Weapon	Rare
16	Nalya	Warhammer of the Iron Titan	11	Weapon	Epic
17	Nalya	Dragonbane Wand	100	Weapon	Rare
18	Kael	Health Potion	10	Consumable	Common
19	Kael	Steel Sword	10	Weapon	Rare
20	Kael	Sigil of the Saint	10	Weapon	Legendary

-

USER_CHARACTER_INFO View

- Purpose: Displays all users and their associated characters they own. It will also display the class, experience level, and balance for each character they own

```
-- Create a view of user and their
-- associated characters they own
CREATE OR REPLACE VIEW "USER_CHARACTER_INFO"
AS SELECT
    U.username AS username,
    U.email AS email,
    C.character_name AS character,
    CL.class_name AS class,
    C.exp_level AS level,
    C.gold_balance AS balance
FROM
    "USER" U
    INNER JOIN "CHARACTER" C ON U.user_id = C.owner_id
    INNER JOIN "CLASS" CL ON C.character_class = CL.class_id;
```

-

Query | Query History

```
1  SELECT * FROM "USER_CHARACTER_INFO";
```

Data Output | Messages | Notifications

	username character varying (25)	email character varying (25)	character character varying (25)	class character varying (25)	level integer	balance integer
1	dragonlord	dragonlord@example.com	Alduin	Warrior	5	300
2	skywalker	skywalker@example.com	Lyria	Rogue	3	150
3	ironfist	ironfist@example.com	Gorath	Mage	7	500
4	shadowblade	shadowblade@example.c...	Serana	Warrior	6	250
5	stormrider	stormrider@example.com	Theron	Rogue	10	800
6	firemage	firemage@example.com	Miraak	Mage	4	200
7	windrunner	windrunner@example.com	Valdrin	Cleric	12	1000
8	earthwarden	earthwarden@example.co...	Nalya	Warrior	2	100
9	waterseer	waterseer@example.com	Druin	Cleric	9	750
10	lightbringer	lightbringer@example.com	Kael	Paladin	8	600

-

ITEM_INFO View

- Purpose: Displays the item information, such as their category, rarity, and classes that are allowed to use that item

```sql
-- Create a view detailing the item info
-- such as name, category, rarity, and
-- usable classes
CREATE OR REPLACE VIEW "ITEM_INFO"
AS SELECT
    I.item_name AS item,
    IC.item_category AS category,
    IR.item_rarity AS rarity,
    STRING_AGG(C.class_name, ', ') AS equipable_classes
FROM
    "ITEM" I
    JOIN "ITEM_CATEGORY" IC ON I.category_id = IC.category_id
    JOIN "ITEM_RARITY" IR ON I.rarity_id = IR.rarity_id
    JOIN "ITEM_CLASS" ICN ON I.item_id = ICN.item_id
    JOIN "CLASS" C ON ICN.class_id = C.class_id
    GROUP BY I.item_name, IC.item_category, IR.item_rarity;
```

- Query | Query History

```sql
SELECT * FROM "ITEM_INFO";
```

Data Output | Messages | Notifications

	item character varying (50)	category character varying (25)	rarity character varying (25)	equipable_classes text
1	Steel Sword	Weapon	Rare	Warrior, Rogue, Paladin
2	Iron Dagger	Weapon	Uncommon	Warrior, Rogue
3	Fire Staff	Weapon	Epic	Mage, Druid
4	Divine Shield	Shield	Legendary	Cleric, Paladin
5	Leather Armor	Armor	Common	Warrior, Paladin
6	Plate Armor	Armor	Rare	Warrior, Paladin
7	Greater Health Potion	Consumable	Uncommon	Necromancer, Warrior, Rogue, Mage, Cleric, Paladin, Ranger, Druid, Monk, B...
8	Elixir of Fortitude	Consumable	Rare	Necromancer, Warrior, Rogue, Mage, Cleric, Paladin, Ranger, Druid, Monk, B...
9	Sigil of the Saint	Weapon	Legendary	Mage, Paladin
10	Wooden Shield	Shield	Common	Warrior, Paladin
11	Dragonbane Wand	Weapon	Rare	Mage, Necromancer
12	Warhammer of the Iron Titan	Weapon	Epic	Paladin
13	Necklace of Wisdom	Accessory	Epic	Mage, Cleric
14	Wizard Hat	Headgear	Rare	Mage, Druid
15	Health Potion	Consumable	Common	Necromancer, Warrior, Rogue, Mage, Cleric, Paladin, Ranger, Druid, Monk, B...
16	Ring of Protection	Accessory	Rare	Necromancer, Cleric, Druid
17	Crown of the Bard King	Headgear	Legendary	Bard
18	Mana Potion	Consumable	Common	Necromancer, Warrior, Rogue, Mage, Cleric, Paladin, Ranger, Druid, Monk, B...

ITEM_MARKET View

- Purpose: For all active items in the LISTING table, display the item, its default price from the ITEM table, its quantity and selling price from the LISTING table, its unit price calculated as $\lfloor \frac{sale\ price}{quantity} \rfloor$, and the availability status of that item

```sql
-- Create a view of the item market
-- displaying active item, default price,
-- quantity, selling price, and unit price
CREATE OR REPLACE VIEW "ITEM_MARKET"
AS SELECT
    I.item_name AS item,
    I.item_price AS default_price,
    L.quantity AS quantity,
    L.sale_price AS market_price,
    FLOOR(L.sale_price / L.quantity) AS unit_price,
    L.is_active AS status
FROM "ITEM" I
JOIN "LISTING" L ON I.item_id = L.item_id AND L.is_active = TRUE;
```

Query | Query History

```sql
SELECT * FROM "ITEM_MARKET";
```

Data Output | Messages | Notifications

	item character varying (50)	default_price numeric (10)	quantity integer	market_price numeric (10)	unit_price numeric	status boolean
1	Greater Health Potion	100	5	350	70	true
2	Elixir of Fortitude	200	2	1200	600	true
3	Iron Dagger	250	7	250	35	true
4	Fire Staff	700	4	200	50	true
5	Plate Armor	600	6	950	158	true
6	Divine Shield	1200	8	600	75	true
7	Wizard Hat	500	9	1000	111	true
8	Titanic Experience Bottle	50000	1	1000	1000	true

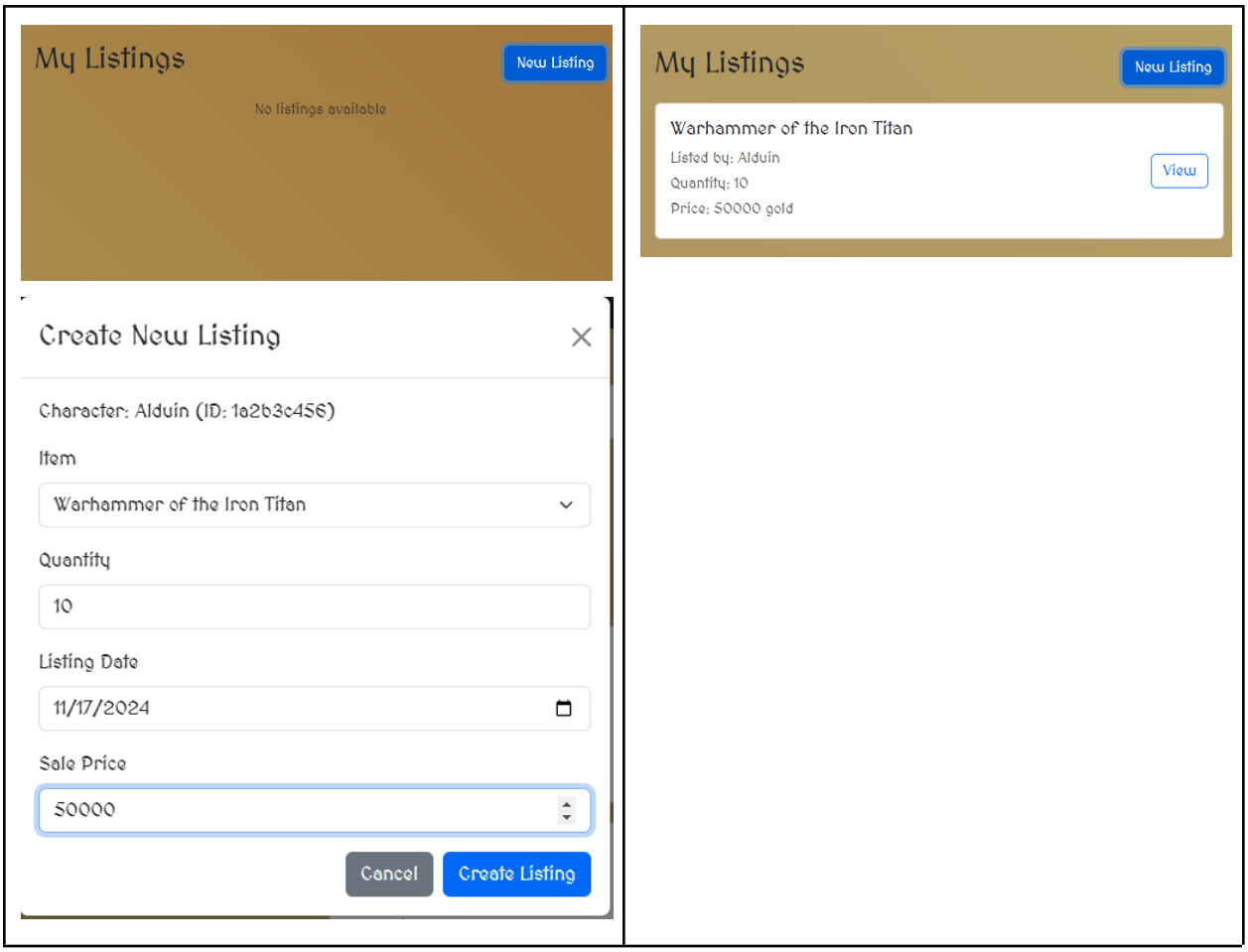
Front End User Interface

[25 POINTS] 4. Front End User Interface: In this part of your project, you are required to use a programming platform to present a front end that will communicate with your normalized backend database you already created earlier. Please note that you are free to choose a web-based or a non-web-based design. One has no (dis)advantage over the other. If you choose to use a web-based design, please note that you will need to have a web server to store and maintain your database. It will be your responsibility to design, install the necessary tools (such as an ER diagram drawing tool, DBMS, web server, compiler, etc.), and implement the database of your

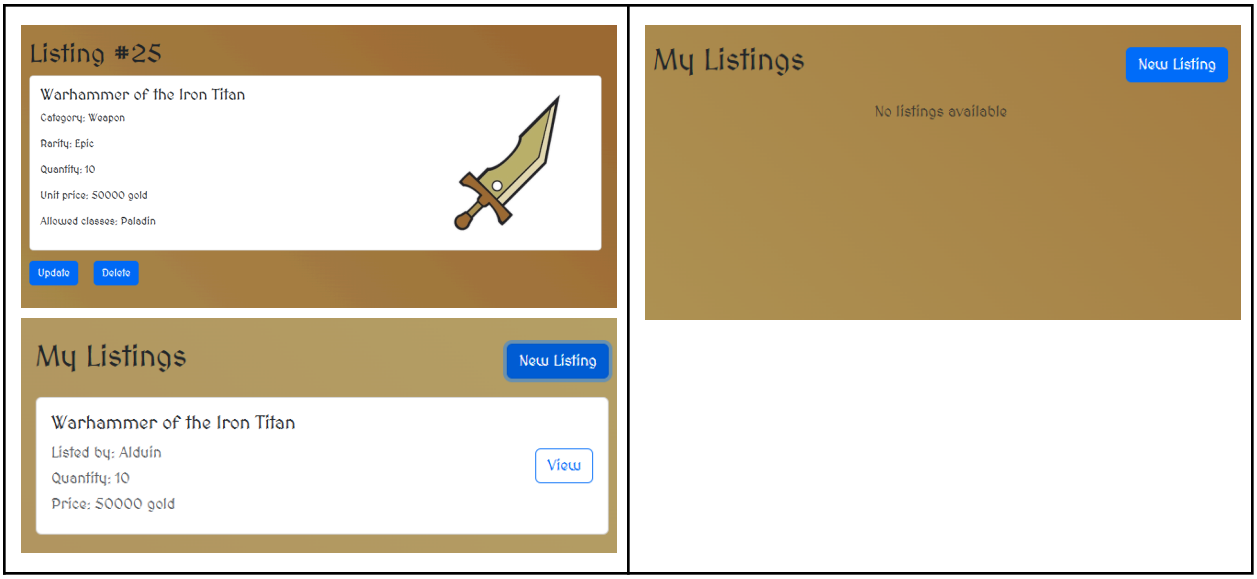
design. Using a programming platform of your choice, implement a code to provide the following: Display a menu with the listed options:

1. Query – to perform operations such as list employees earning more than 150K per year

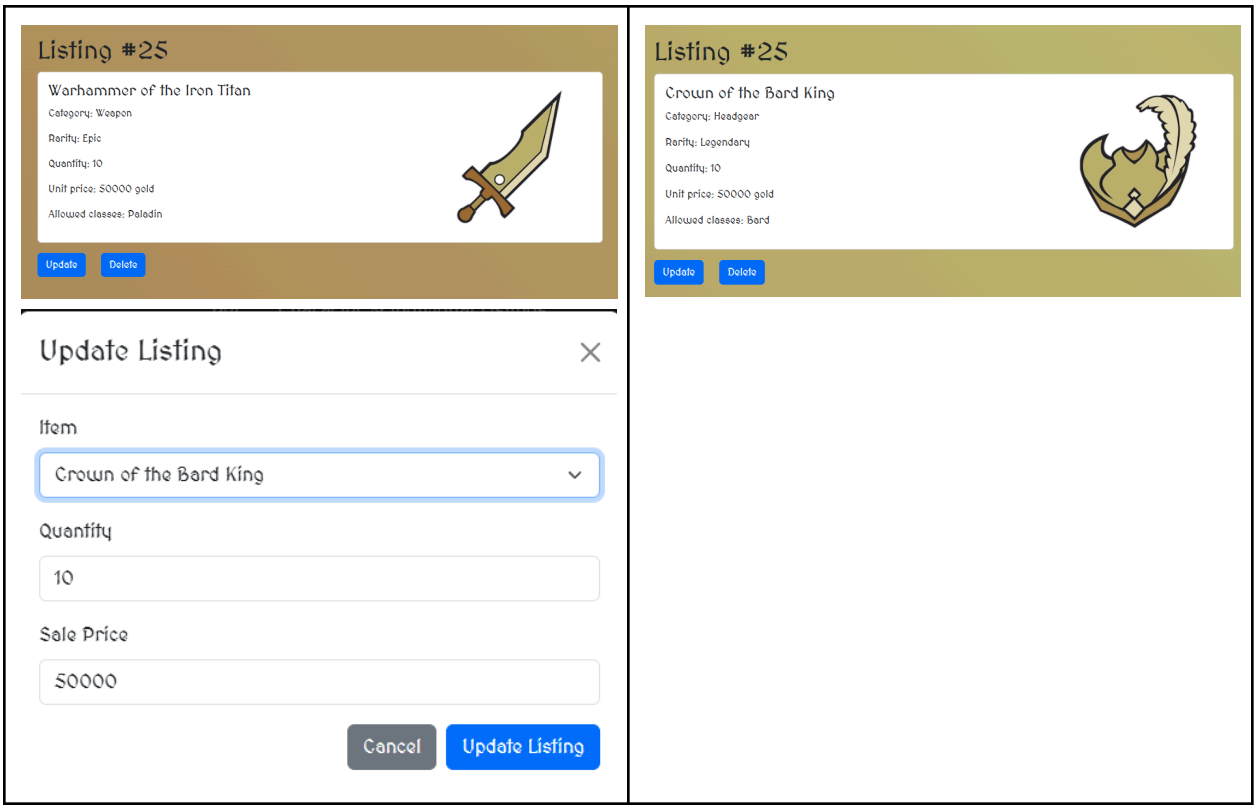
2. Insert – to add new tuple(s), and/or field(s) to your table(s)



3. Delete - to remove tuple(s), and/or field(s) from your table(s)



4. Update - to modify tuple(s), and/or field(s) from your table(s)



5. Quit

Once the user selects an option, perform the selected option. **Provide a screenshot of each of the above 5 menu options, as well as the resulting output displayed by your program in your report exactly in this section.** Offer some variety by including more complex queries such as subqueries, aggregates, set operators, etc.

Conclusion and Future Work

[3 POINTS] 5. Conclusion and Future Work: Provide a summary of your work here. Also provide any changes that you need to make (if any), if things have deviated from what you had originally planned for, and try to give justification for such changes. Suggest future expansion ideas for your project to promote its validity

Our project was centered around creating a virtual marketplace for an MMO game, especially concerning the database. We created 5 different operations to use in the database: insert operations to create new listings and register new users, queries for viewing all listings and

logging into an existing account, delete operations to delete listings, update operations to edit listings, and quit operations to log out of existing accounts.

One of the changes we made compared to our original design was restructuring the item table which was not normalized. The allowed_classes column violated 1NF by storing a set/array of data. To address this, we created a new item_class table with a composite primary key (item_id and class_id), referencing the item and class tables, to store the association between items and classes. This resolved the normalization issue for allowed_classes. We then identified that the item_category and item_rarity columns were only in 2NF due to transitive dependencies (item_id → item_name → item_category and item_id → item_name → item_rarity). We created item_category and item_rarity tables with primary keys (category_id and rarity_id) and corresponding names to fix this. We also replaced item_category and item_rarity in the item table with foreign keys referencing these new tables. These changes ensured our database adhered to 3NF, eliminating redundancy and improving integrity.

Another change we made was to change the table creation string length from “25” to “50” since the app was blocking registrations/login for longer emails. We also made usernames and emails unique IDs to avoid any mixups.

For future expansion, we could include more features for implementation, such as adding direct trading amongst players and having a seasonal shop that rotates monthly. Auction houses are also a viable feature that could be worked into the virtual marketplace, given the compatibility of the main game. We can also promote the marketplace to more games, allowing them to easily implement an important feature of any MMO game.

References

[1] "FIFA Marketplace," FIFA Collect, https://collect.fifa.com/marketplace (accessed Oct. 8, 2024).

[2] "Official site: Second life - virtual worlds, virtual reality, VR, avatars, free 3D chat," Second Life Marketplace, https://marketplace.secondlife.com/en-US (accessed Oct. 5, 2024).

[3] "Unity Asset Store," Unity Asset Store 2024 version, https://assetstore.unity.com/ (accessed Oct. 4, 2024).

[4] "Steam community :: Steam community market," Steam Community :: Steam Community Market, https://steamcommunity.com/market/ (accessed Oct. 6, 2024).

[5] S. E. Inc., "Eorzea database," FINAL FANTASY XIV, The Lodestone, https://na.finalfantasyxiv.com/lodestone/playguide/db/ (accessed Oct. 6, 2024).

[6] BitSkins, "DOTA2 market: Buy & Sell DOTA2 skins," BitSkins, https://bitskins.com/market/dota2 (accessed Oct. 5, 2024).

[7] Larvalabs, "Larvalabs/cryptopunks: Collectible 8-bit characters on the Ethereum blockchain.," GitHub, https://github.com/larvalabs/cryptopunks (accessed Oct. 6, 2024).

[8] "Marketplace," RuneScape Wiki, https://runescape.wiki/w/Marketplace (accessed Oct. 5, 2024).