Statement and Confirmation of Own Work

***A signed copy of this form must be submitted with every assignment.***

***If the statement is missing your work may not be marked.***

**Student Declaration**

I confirm the following details:

|  |  |
| --- | --- |
| **Candidate Name:** | SHIHAB MIRZA |
| **Candidate ID Number:** | P00190603 |
| **Qualification:** | NCC L5DC |
| **Unit:** | DDD (Database’s design and development) (20 Credits) |
| **Centre:** | ZCAS University |
| I have read and understood both NCC Education’s *Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.  I confirm that this is my own work and that I have not colluded or plagiarised any part of it. | |
| **Candidate Signature:** |  |
| **Date:** | 12/10/2022 |



**OPS020\_Candidate Statement of Own Work**

**TASK (1) (Business Description)-**

Zambia Commodity Exchange (ZAMACE) is a commodity trading platform which involves an ecosystem of players that partake in it.

It enables the trade of many of the commodities that are grown and harvested in Zambia, like maize, soy bean, rice etc.

The players of this ecosystem involve, brokers, traders, farmers and warehouse operators.

A warehouse operator is simply a company that stores commodities under the provision of ZAMACE. The warehouse operator can have many storage sites (Warehouses) situated at various locations across the country. The storage sites have a certain storage capacity for storing commodities.

Farmers deposit their harvest by the warehouse operators through brokers in exchange for money or a receipt if they want to keep it as a security. Obviously, there is a fee for storing it. But this process is not only limited for farmers, but any other people who want to store and secure their commodities. ZAMACE is also responsible for the act of grading the commodities present in a certain warehouse.

ZAMACE also keeps records about its “hot” warehouse locations. Where warehouses are located the most.

Each of the storage sites for all warehouse operators registered with ZAMCE are graded and tested by ZAMACE to check for the quality of their infrastructure and the suitability for storing commodities. Factors such as cleanliness levels, moisture levels, and so on, are measured by ZAMACE operatives.

Brokers, are simply the middlemen or organizations that are present for allowing traders to execute trades. ZAMACE itself also has its own brokers.

ZAMACE takes a 1(one) % fee on each trade. And the brokers can have their own fees set. However, if one decides to choose a broker other than ZAMACE, he will not pay the 1% fee set by ZAMACE, rather the brokers will pay that. (Farmers can also be traders, but they are given a discount on storage fees if they are subsistence farmer hence ZAMACE keeps separate records for them). Commodity trades are only allowed in (metric tons). A broker cannot make more than 10000 trades on his account.

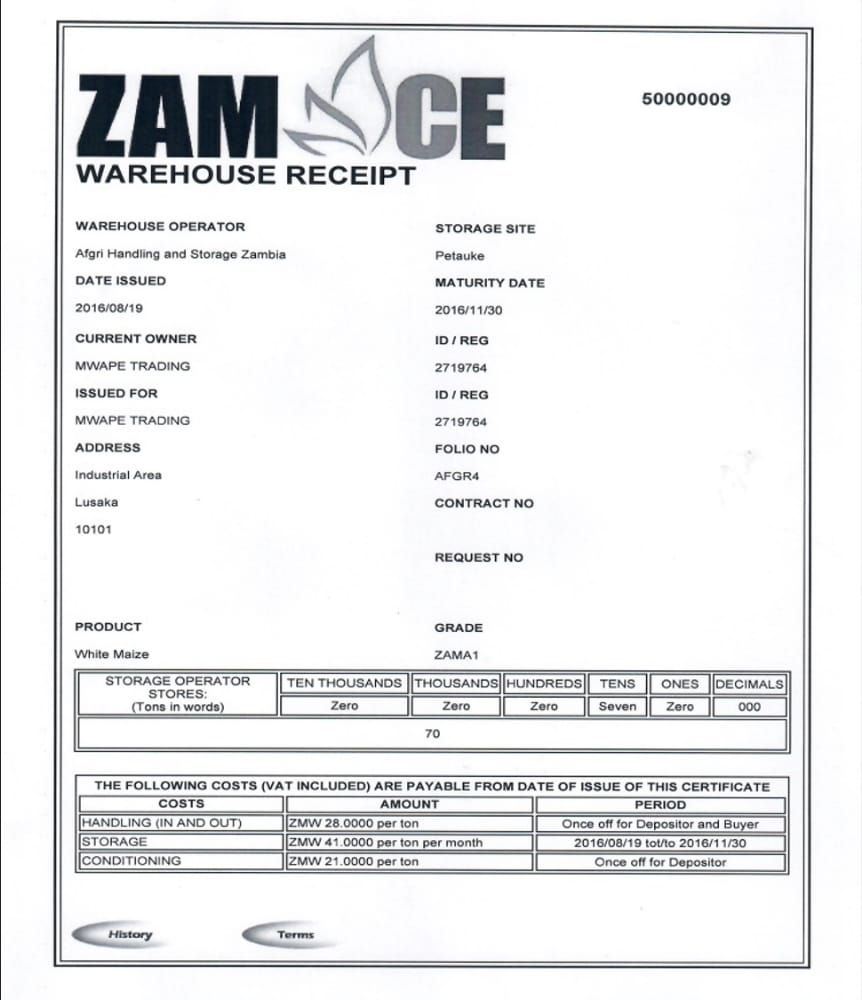
If a trader decides to buy a certain commodity, he will have the option to store it with a warehouse operator, for a certain fee per month. Until he finally decides to sell.

The constant daily buying and selling dictates the market price of each commodity.

The scope of this database project will critically consider data issues related to ZAMACE. Some of the entities are suggested below.

* Farmers
* Brokers
* Traders
* Warehouse Operators
* Commodities
* Storage sites
* Commodity grades.
* Locations
* Trades
* Storage site grades.

Below is a document from ZAMACE (a warehouse Receipt)



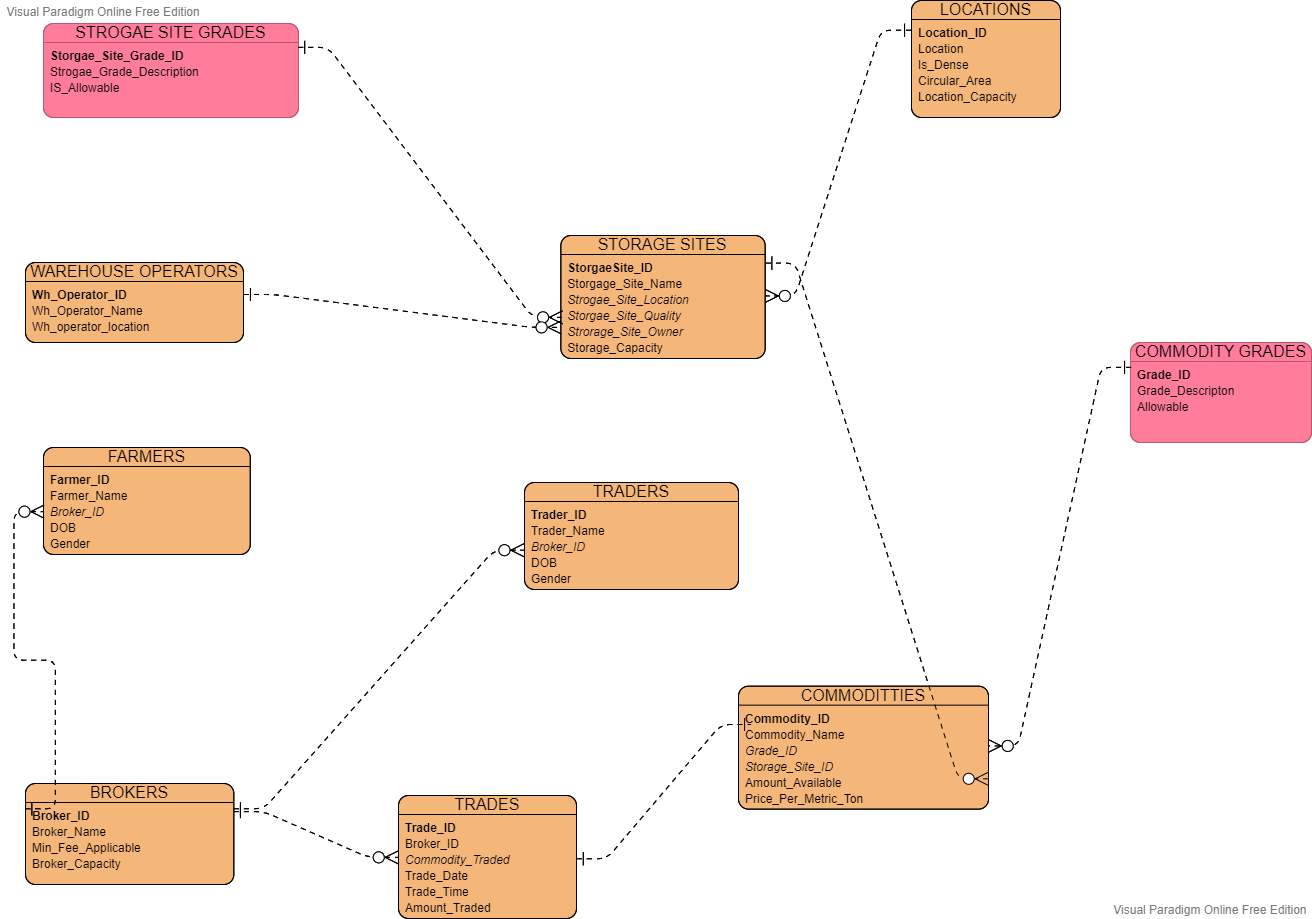
***Note:*** It is important to know that not all of the attributes mentioned in this document might make it to the entity attributes. And also, that some new attributes might be added as well

ZAMACE has a complex set of mechanisms and data assets in its ecosystem which cannot be fully expressed through this project. Hence, the scope of this project will only include all attributes and pieces of data related to the entities mentioned above. All else will be ignored. For example, the trades that the brokers will place will be kept record of, but not what specific trader each trade belongs to. That is the duty of the brokers to handle.

**TASK (2)-**

ERD: (This image can be found in the assignment folder).

Hint: Primary keys are bold. Foreign keys are italic.



Some important constraints to note, just off the top:

A farmer or trader should be above the age of 18.

The valid values for gender are either male, female or other.

A Broker record on update cascade. On delete no action. In farmer and trader tables.

The valid values for the allowable fields in the pink tables are either True or False or Unknown.

DATA DICTIONARY:

All primary keys maintain entity integrity and foreign keys maintain relational and referential integrity…

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TABLE** | **KEY** | **ATTRIBUTE NAME** | **DATA TYPE** | **DATA SIZE** | **CONSTRAINTS** | **DESCRIPTION** |
| **FARMERS** | Primary Key | Farmer ID | VARCHAR | 20 | Primary Key, Not Null | Uniquely identifies each farmer |
|  |  | Farmer Name | VARCHAR | 40 | Should be more than 1 char long | Name |
|  |  | DOB | DATE | \_ | Should be in YYYY//MM//DD format | DOB |
|  |  | Gender | VARCHAR | 10 | Should be male, female or other. | Gender |
|  | Foreign key | Broker ID | VARCHAR | 20 | Foreign key, on delete set null, on update cascade | Tells us which broker the farmer is associated with |
| **TRADERS** | Primary Key | Trader ID | VARCHAR | 20 | Primary Key, Not Null | Uniquely identifies each Trader |
|  |  | Trader Name | VARCHAR | 40 | Should be more than 1 char long | Name |
|  |  | DOB | DATE |  | Should be in YYYY//MM//DD format | DOB |
|  |  | Gender | VARCHAR | 10 | Should be male, female or other. | Gender |
|  | Foreign key | Broker ID | VARCHAR | 20 | Foreign key, on delete set null, on update cascade | Tells us which broker the trader is associated with |
| **BROKERS** | Primary Key | Broker ID | VARCHAR | 20 | Primary Key, Not Null | Uniquely identifies each broker |
|  |  | Broker Name | VARCHAR | 30 | Should be more than 1 char long, not null | Broker name |
|  |  | Min\_fee\_applicable | FLOAT or DOUBLE | 8 bytes | Should be more than 1 | The minimum fee (in %) to pay to ZAMACE on each trade. |
|  |  | Broker Capacity | INT | \_ | shouldn’t be more than 10000 | The maximum number of trades a broker can have |
| **TRADES** | Primary Key | Trade ID | VARCHAR | 20 | Primary Key, Not Null | Uniquely iidentifies each trade |
|  | Foreign key | Broker ID | VARCHAR | 20 | Foreign key, on delete no action, on update cascade | Identifies which broker placed trade. |
|  | Foreign key | Commodity Traded | VARCHAR | 20 | Foreign key, on delete no action, on update cascade | Identifies which commodity was traded. |
|  |  | Trade date | DATE | \_ | not null, should be in dd//mm//YY format | Date of trade |
|  |  | Trade time | TIME | \_ | not null, should be in HH//MM//SS format | Time of trade |
|  |  | Amount traded | INT | \_ | not null. Should be more than 10, less than 500, but correct to 0 DP and an int | amount traded in metric tons |
| **COMMODITIES** | Primary Key | Commodity ID | VARCHAR | 20 | Primary Key, Not Null | Uniquely identifies each commodity |
|  |  | Commodity name | VARCHAR | 30 | Should be valid commodity name | name of commodity |
|  | Foreign key | Grade ID | VARCHAR | 1 | Foreign key, on delete no action, on update cascade | The grade given to this commodity |
|  | Foreign key | Storage site ID | VARCHAR | 20 | Foreign key, on delete no action, on update cascade | The warehouse in which this commodity is present |
|  |  | Amount\_Available | INT | \_ | Should be more than 10000 (mt) | The amount of commodity present in (mt) |
|  |  | Price\_Per\_Metric\_Ton | DOUBLE | \_ | not null, should be correct to at least 2 DP | The price per metric ton for that commodity |
| **COMMODITY GRADES** | Primary Key | Grade ID | VARCHAR | 1 | Primary Key, Not Null, | Uniquely identifies each type of grading |
|  |  | Grade Description | TEXT | 100 | Should be less than 100 chars | description |
|  |  | Allowable | BOOLEAN | \_ | Allows only true, false or unknown value | If the grade given is acceptable or not. |
| **WAREHOUSE OPERATORS** | Primary Key | WH\_operator ID | VARCHAR | 20 | Primary Key, Not Null | Uniquely identifies each warehouse operator company |
|  |  | WH\_operator name | VARCHAR | 30 | not null | operator name |
|  |  | Wh\_operator location | VARCHAR | 60 | not null | the location of main office |
| **STROAGE SITES** | Primary Key | Storage site ID | VARCHAR | 20 | Primary Key, Not Null, | Uniquely identifies each storage site |
|  |  | Storage site name | VARCHAR | 30 | Should be small caps | The name of the storage site |
|  | Foreign key | Location ID | VARCHAR | 20 | Foreign key, on delete no action, on update cascade | The location of the storage site |
|  | Foreign key | Quality | VARCHAR | 1 | Foreign key, on delete no action, on update cascade | The quality of the storage site |
|  | Foreign key | Owner | VARCHAR | 20 | Foreign key, on delete set null, on update cascade | The owner (wwarehouse operator) of the storage site |
|  |  | Capacity | INT | \_ | Not null, should be more than 500 | The capacity in metric tons |
| **STORAGE SITE GRADES** | Primary Key | Grade ID | INT | 11 | Primary Key, Not Null | Uniquely identifies each type of grading |
|  |  | Description | TEXT | 100 | Can be null | description |
|  |  | Allowable | BOOLEAN | \_ | Allows only true, false or unknown value | If the grade given is acceptable or not. |
| **LOCATIONS** | Primary Key | Location ID | VARCHAR | 20 | Primary Key, Not Null | Uniquely identifies each location. |
|  |  | Location | VARCHAR | 60 | Should be valid location | The name of the location |
|  |  | is Dense | BOOLEAN | \_ | Allows only true, false or unknown value, not null | Checks if area is dense by checking the amount of storage sites present |
|  |  | Circular area | DOUBLE | \_ | Should be more than 50 km squared | The circular area of the location in km squared |
|  |  | Location capacity | DOUBLE | \_ | not null | The maximum number of warehouses the area can have. |

**TASK 3 (Normalization)**

The method used for organizing data in a database is known as normalization. This method includes creating of tables and establishing of relationships between them in accordance with the rules that are meant to both safeguard the data and make the database overall more adaptable by removing of redundant and inconsistent dependencies.

removing data that is redundant (storing the same data in multiple tables) and making sure that data dependencies make sense (only storing related data in a table) are two important purposes.

There are 3 mainly know stages of normalization:

* 1-NF (Removing of repeating groups.)
* 2-NF (Removing of partial (non-key) dependencies).
* 3-NF (Removing of transitive dependencies.

All my entities are normalized to 3NF because due to the absence of insertion, update, and deletion anomalies in the majority of 3NF tables, Third Normal Form (3NF) is considered adequate for normal relational database design. Furthermore, 3NF always guarantees lossless and functional dependency preservation.

How I Verified that my tables are well structured using normalization:

* In my database, each of my tables has a single (atomic) value in each of its columns.
* The values stored in a column of each table are of the same domain; every table has its own name to avoid confusion; the order in which the data is stored is irrelevant; and normalization gives each column in each table a unique name.
* Each record is uniquely identified in my tables by the primary key because, there is functional dependence A -> B, C
* Update, deletion and insertion anomalies do not occur.

What causes an update anomaly?

If we changed data in one tuple and made it inconsistent with the same data in another tuple, we might get an update anomaly. Normalization solved this problem because, the data is structured by the database designer so that there are no duplicates, if duplication occurs, that will become another entity on its own. Given the example below. Let’s say the commodities table was structured like this.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Commodity ID | Commodity Name | Grade Code | Grade Description | Allowable | Storage Site ID | Price ($) | Amount (mt) |
| 1 | Maize | A | Good | TRUE | 1 | 200 | 1000 |
| 2 | Soy Bean | C | Bad | FALSE | 1 | 100 | 10000 |
| 3 | Rice | C | Bad | FALSE | 2 | 250 | 200000 |

Say now we want to change the Allowable values to TRUE and the Description to “consumable” for each commodity that has a C grade, we will have to change for all the records, which is redundant and brings about update anomalies.

So, we make our life easier by splitting the grading of the commodity into a separate table like so and the change is only made once;

|  |  |  |
| --- | --- | --- |
| Grade ID | Description | Allowable |
| A | GOOD | TRUE |
| B | OK | TRUE |
| C | CONSUMABLE | TRUE |
| D | BAD | FALSE |

And we can just introduce a foreign key field in the commodity table like so:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Commodity ID | Commodity Name | Storage Site ID | Price ($) | Amount (mt) | GRADE ID |
| 1 | Maize | 1 | 200 | 1000 | A |
| 2 | Soy Bean | 1 | 100 | 10000 | C |
| 3 | Rice | 2 | 250 | 200000 | C |

FOREIGN KEY FIELD

We can also take the storage sites table and storage site grades table as an example for this. Though for storage sites we use INT and we consider the higher number the better

From 1 to 10.

**TASK (4) (Assessment of design)**

**Logical Design**

A conceptual, abstract design is a logical design. You haven't talked about the specifics of the actual implementation yet; Choose only the kinds of information you need.

The organization of data into a series of logical relationships known as entities and attributes is one aspect of logical design. A means of representing a piece of information is an entity. An entity and a table frequently correspond in relational databases. An attribute is a feature of an entity that makes it distinctive. A column is mapped to an attribute in relational databases.

The logical design can be created with a pen and paper or with a design tool like Oracle Designer or Oracle Warehouse Builder.

**To physical design**

This is what needs to be done to put the physical design into action, assuming that the logical data model is finished. By transforming the logical data model into a physical implementation based on our DBMS, we must establish an initial physical data model. We must accomplish the following in order to successfully create a physical data model:

Attributes must be transformed into columns, relationships into foreign keys, and domains into constraints and data types, entities must all be transformed into tables.

**How the tables have been designed:**

In order to avoid data redundancies and anomalies, the designed database has been normalized to the third normal form. As a result, I have created ten tables, each with its own name and characteristics. Additionally, it has special keys like the foreign and primary keys. While the foreign key is used to connect one table to another and adheres to the referential integrity rules, the primary key is in charge of uniquely identifying each row in the table.

**Derived Data:**

New data is derived when existing raw data are combined and processed to produce derived data. Observational, experimental, and simulation data can be used to create derived data, but not previously derived data. In my case, the location table has a field called location capacity, the value for this column is derived from calculating the following, circular-area / average storage site land area. This column tells us the maximum number of warehouses possible to fit it that location.

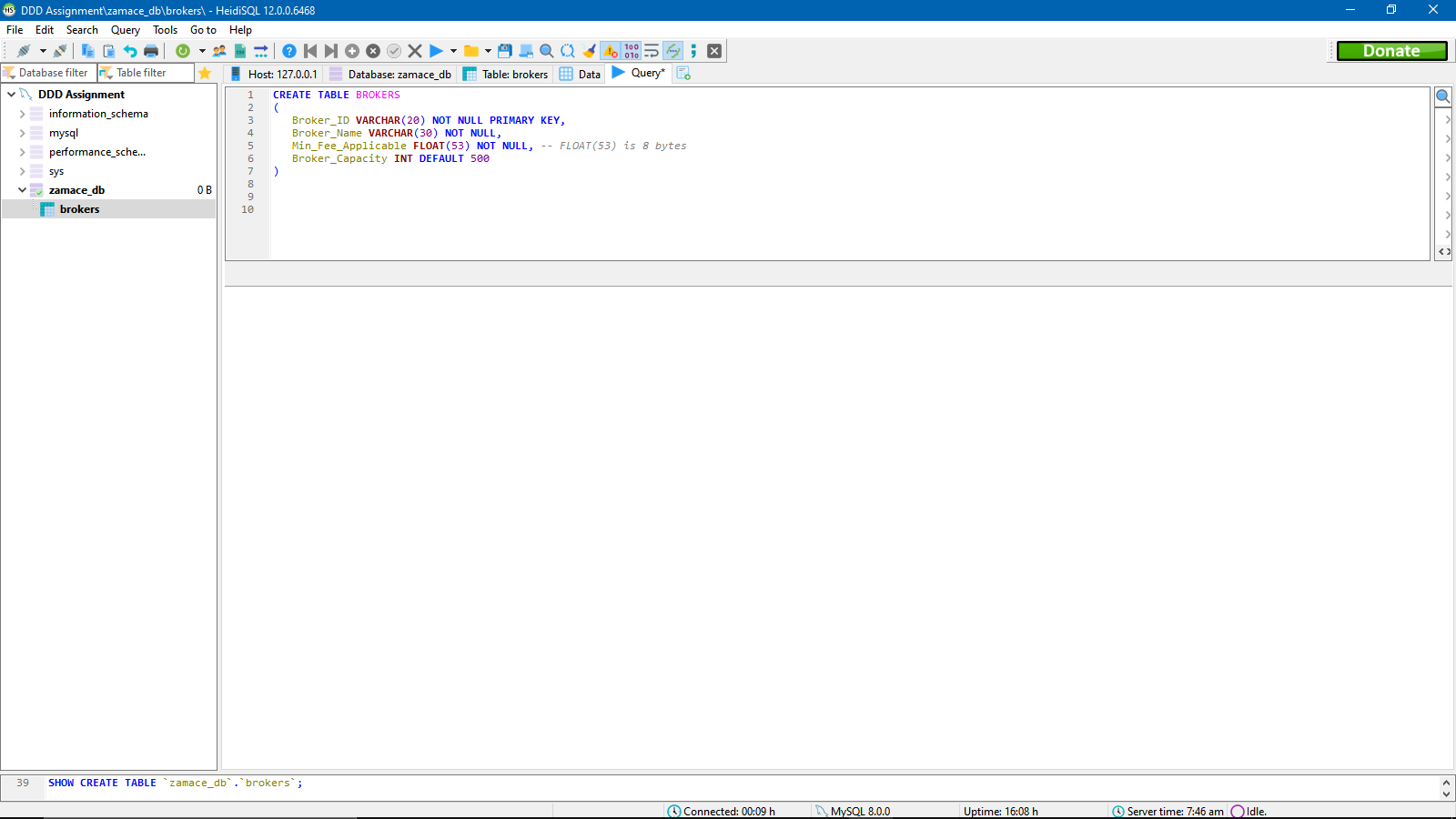
**Denormalization:**

Denormalization is a technique for improving performance in a normalized database. Denormalization is the process of trying to improve a database's read performance while sacrificing some write performance by grouping or adding redundant copies of data.

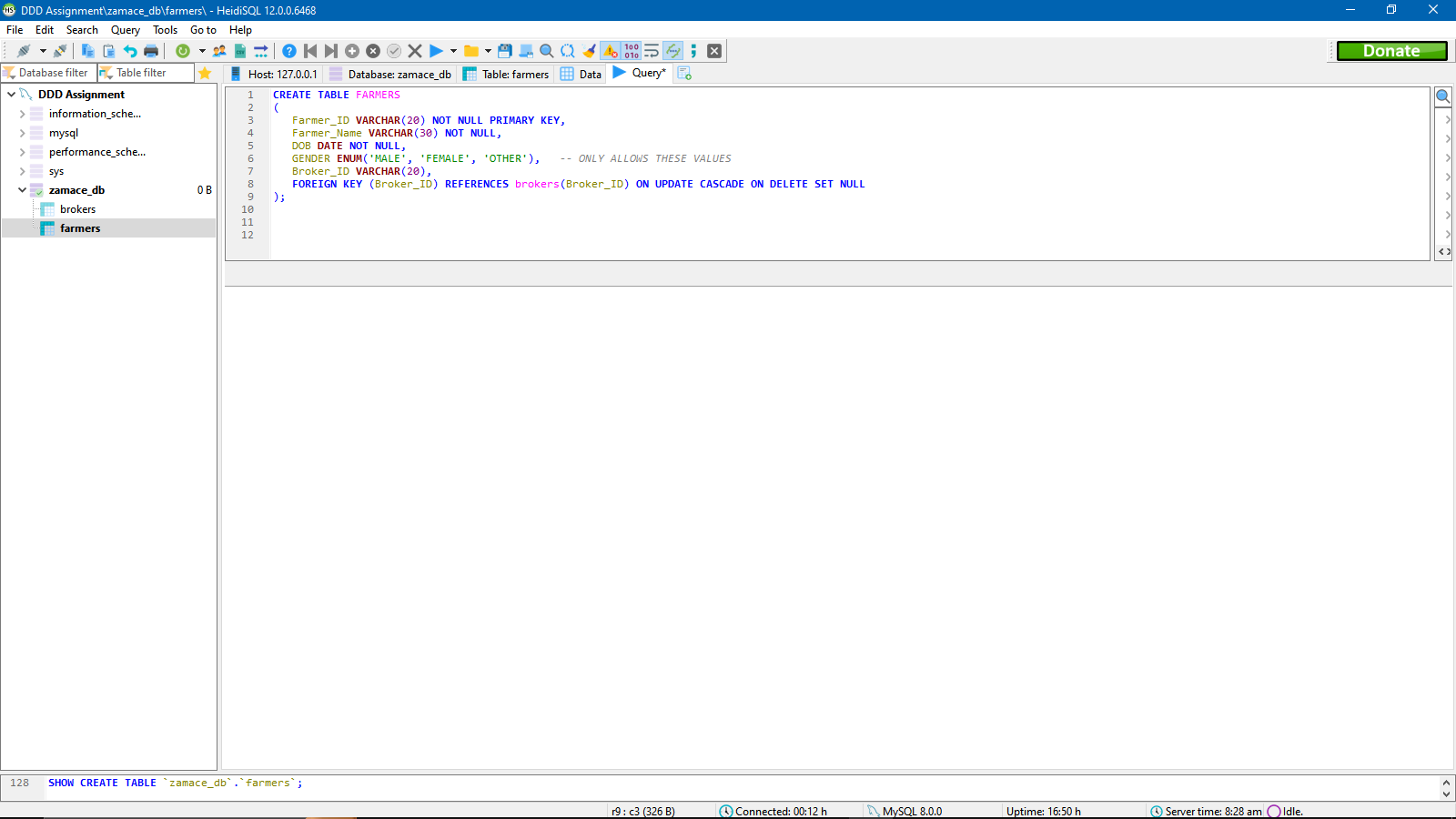
In my case, I will not be using any denormalization on any of my tables.

**TASK (5) (Creating of tables)**

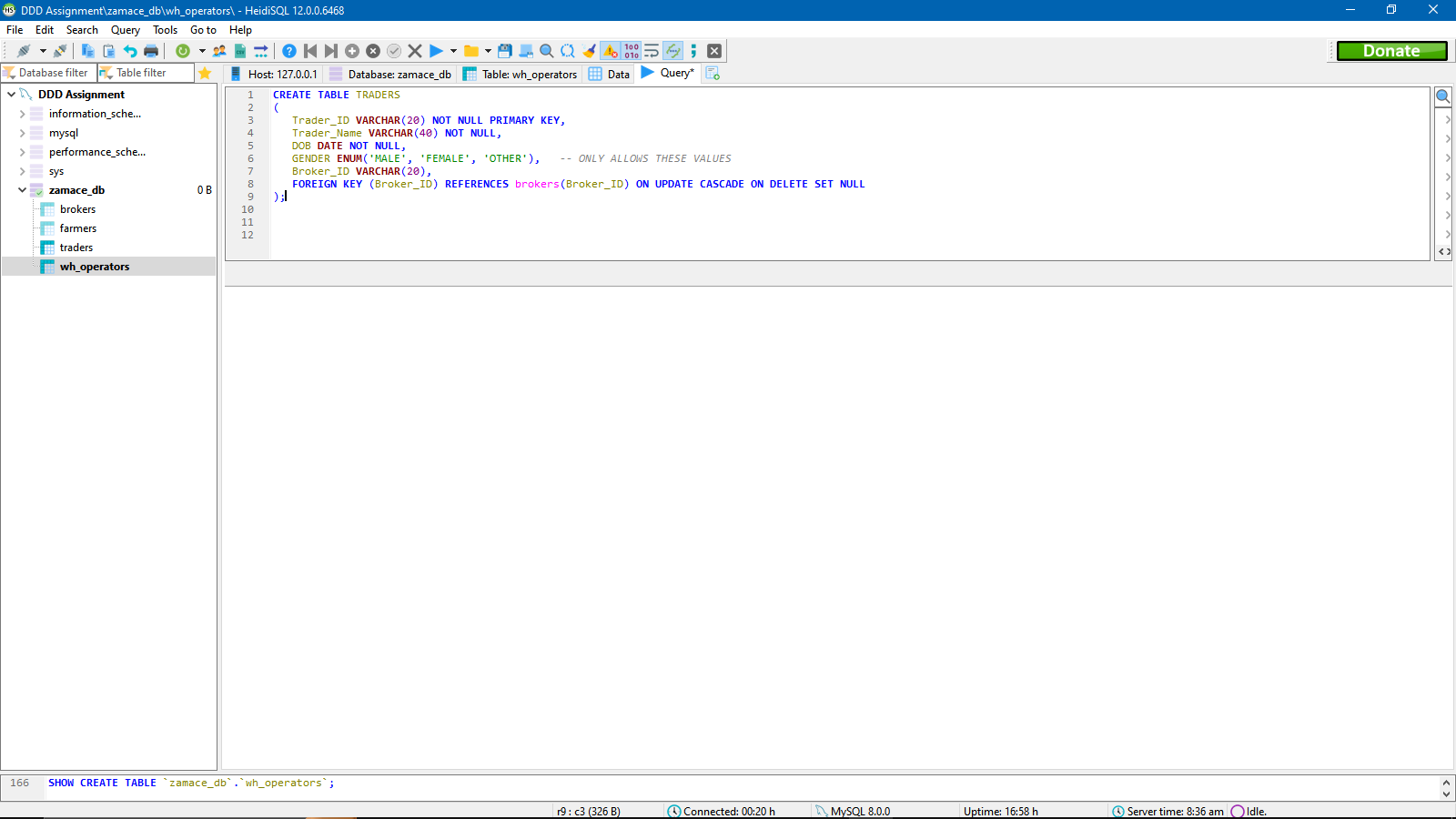
BROKERS TABLE



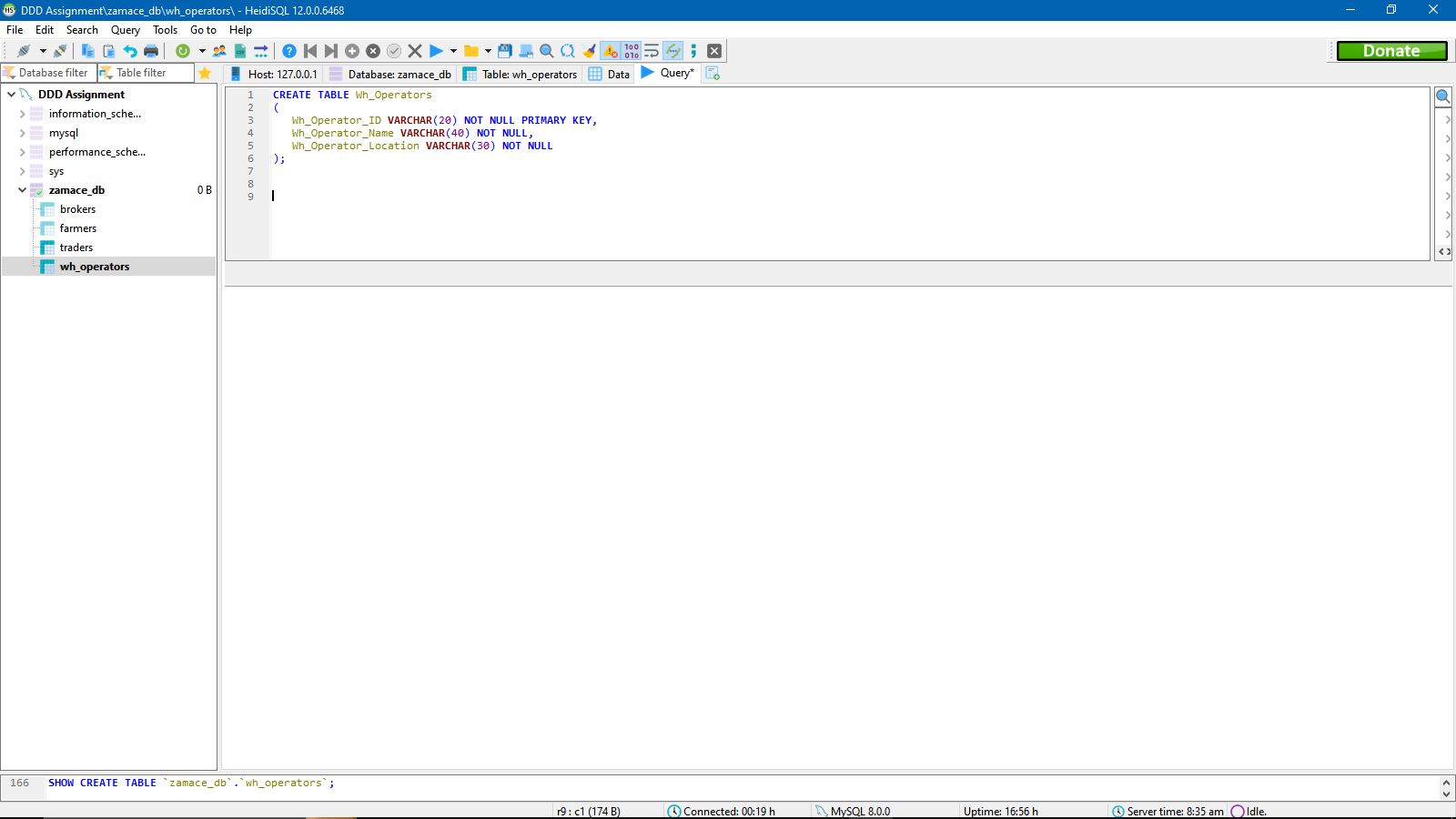
FARMERS TABLE



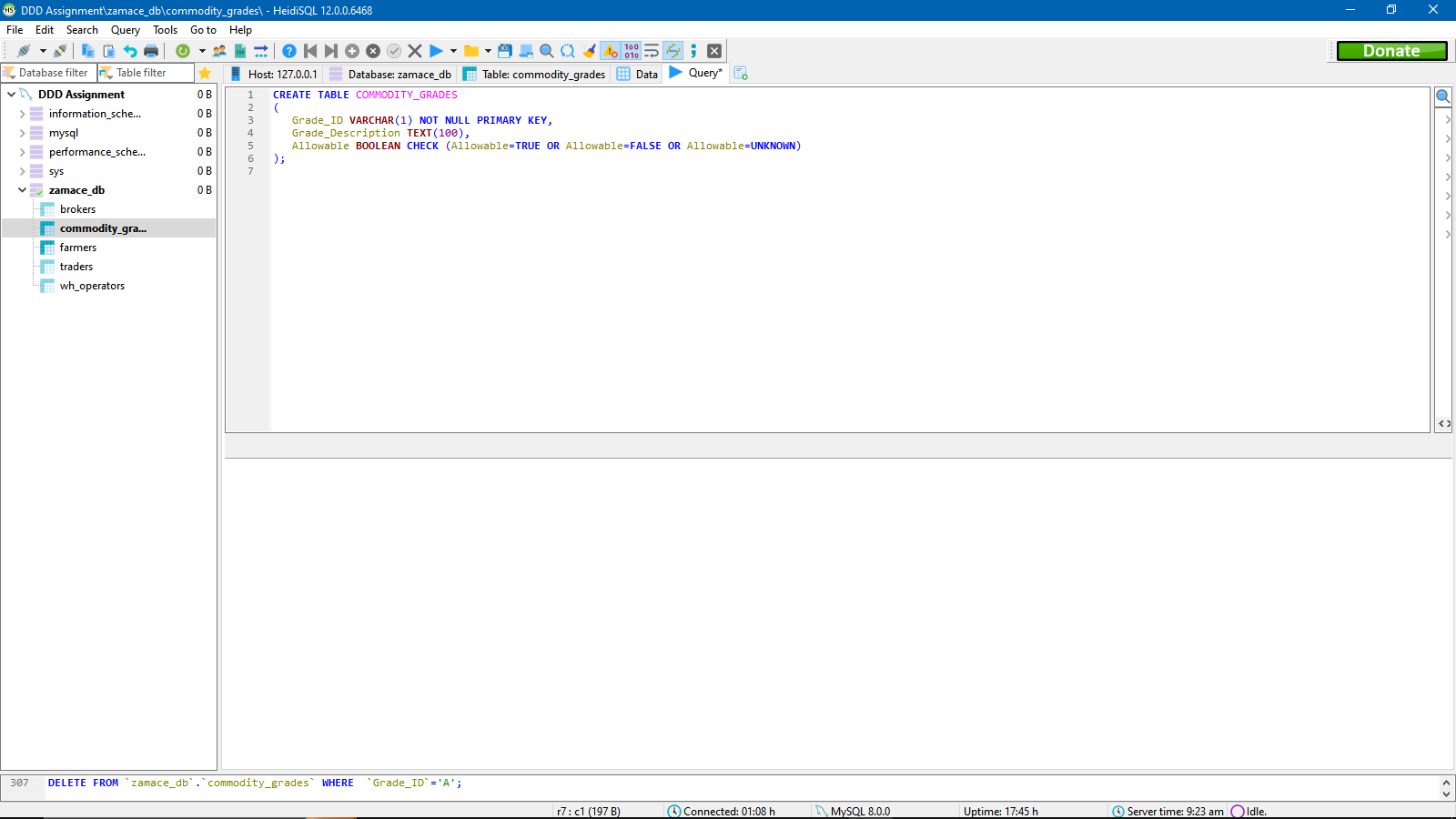
TRADERS TABLE



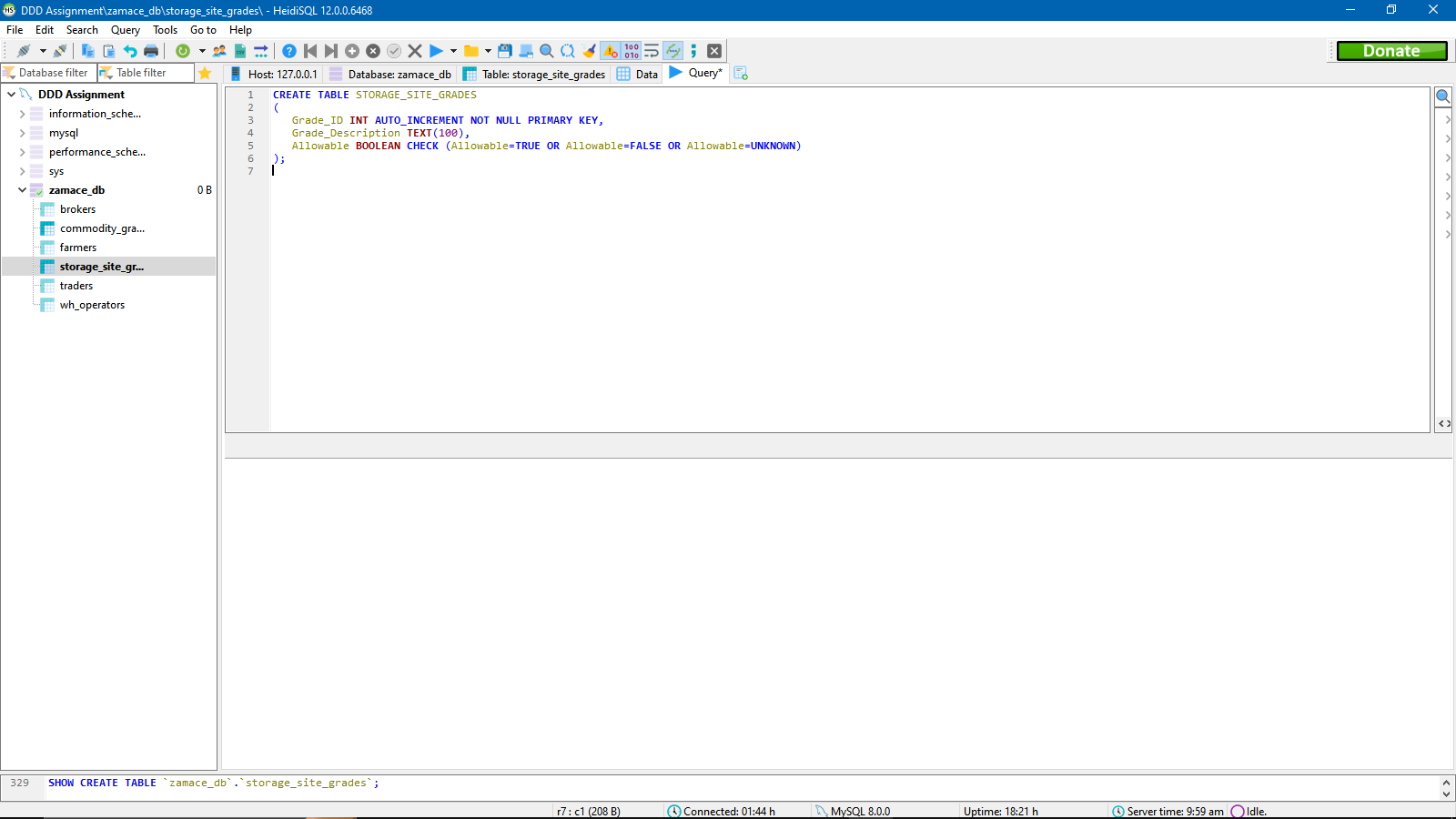
WAREHOUSE OPERATORS TABLE



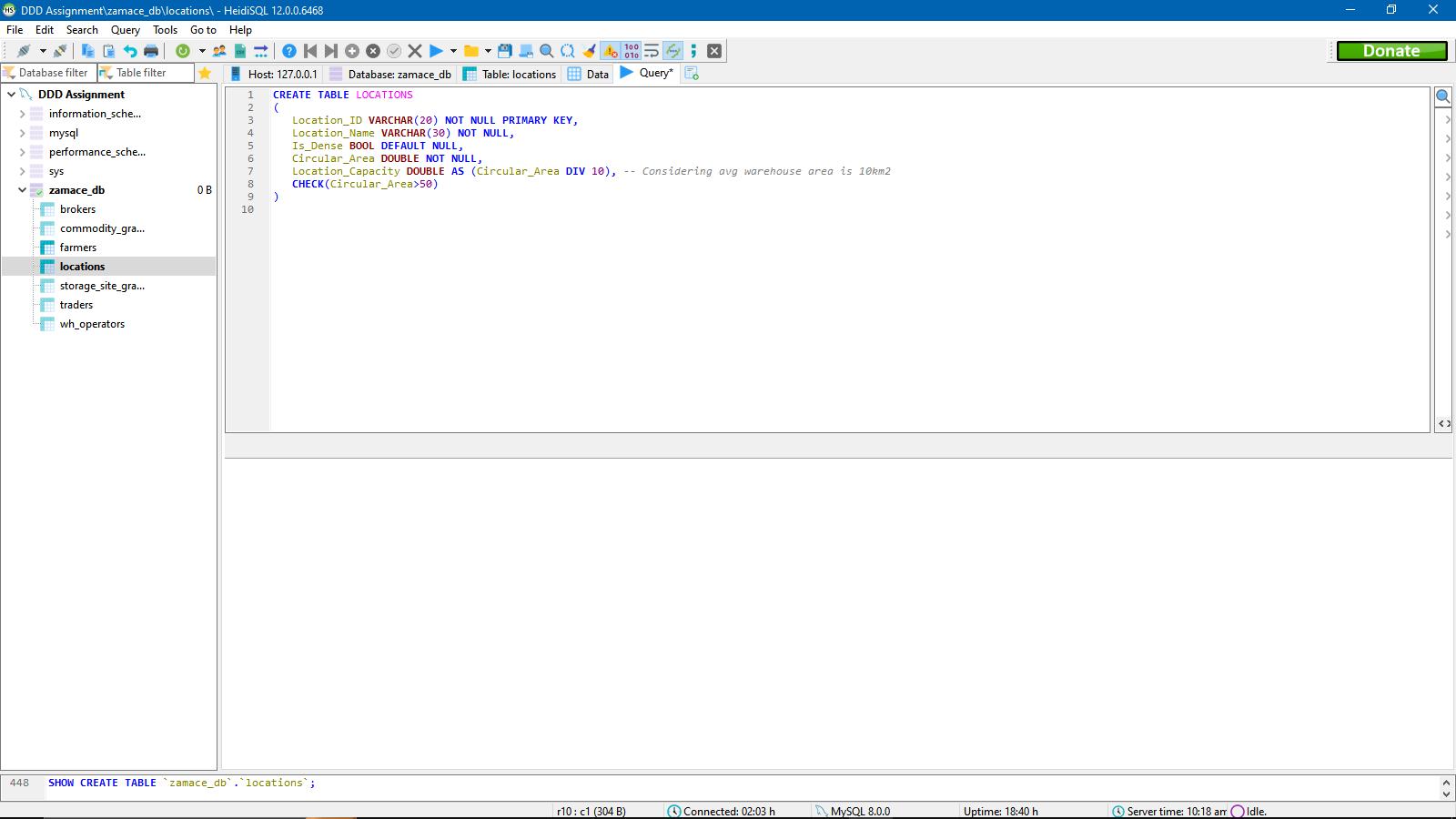
COMMODITY GRADES TABLE



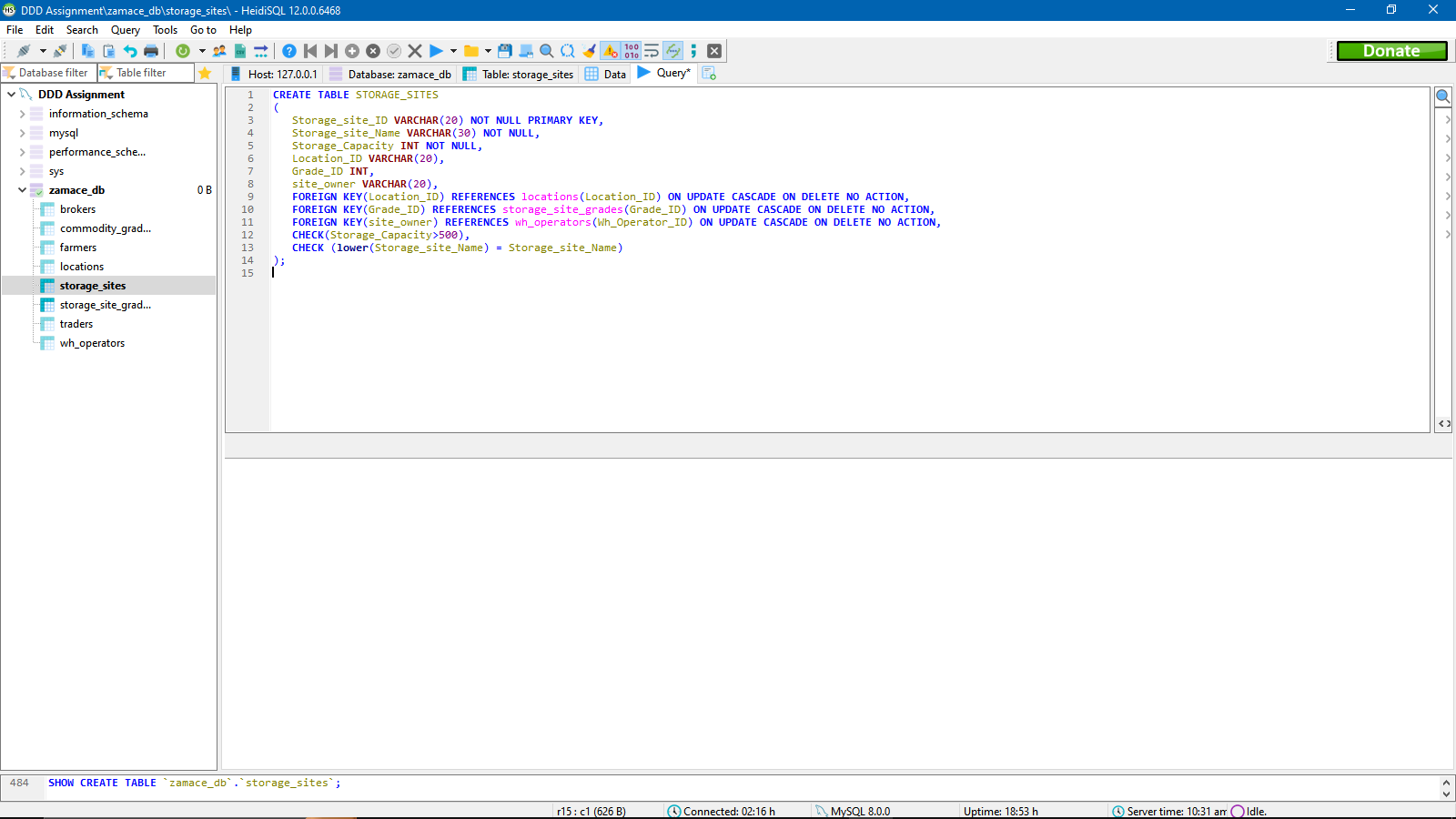
STORAGE SITE GRADES TABLE



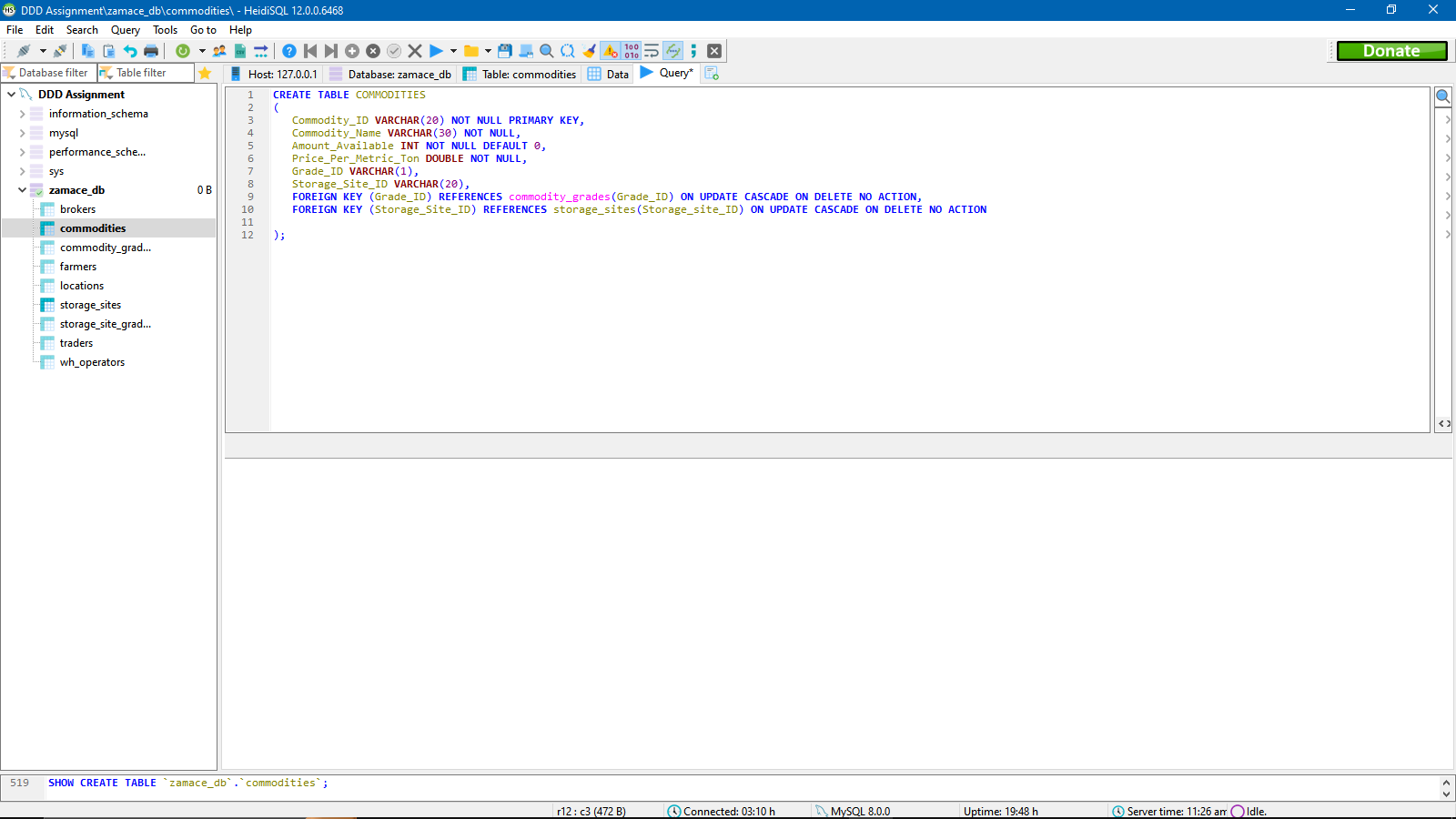
LOCATIONS TABLE



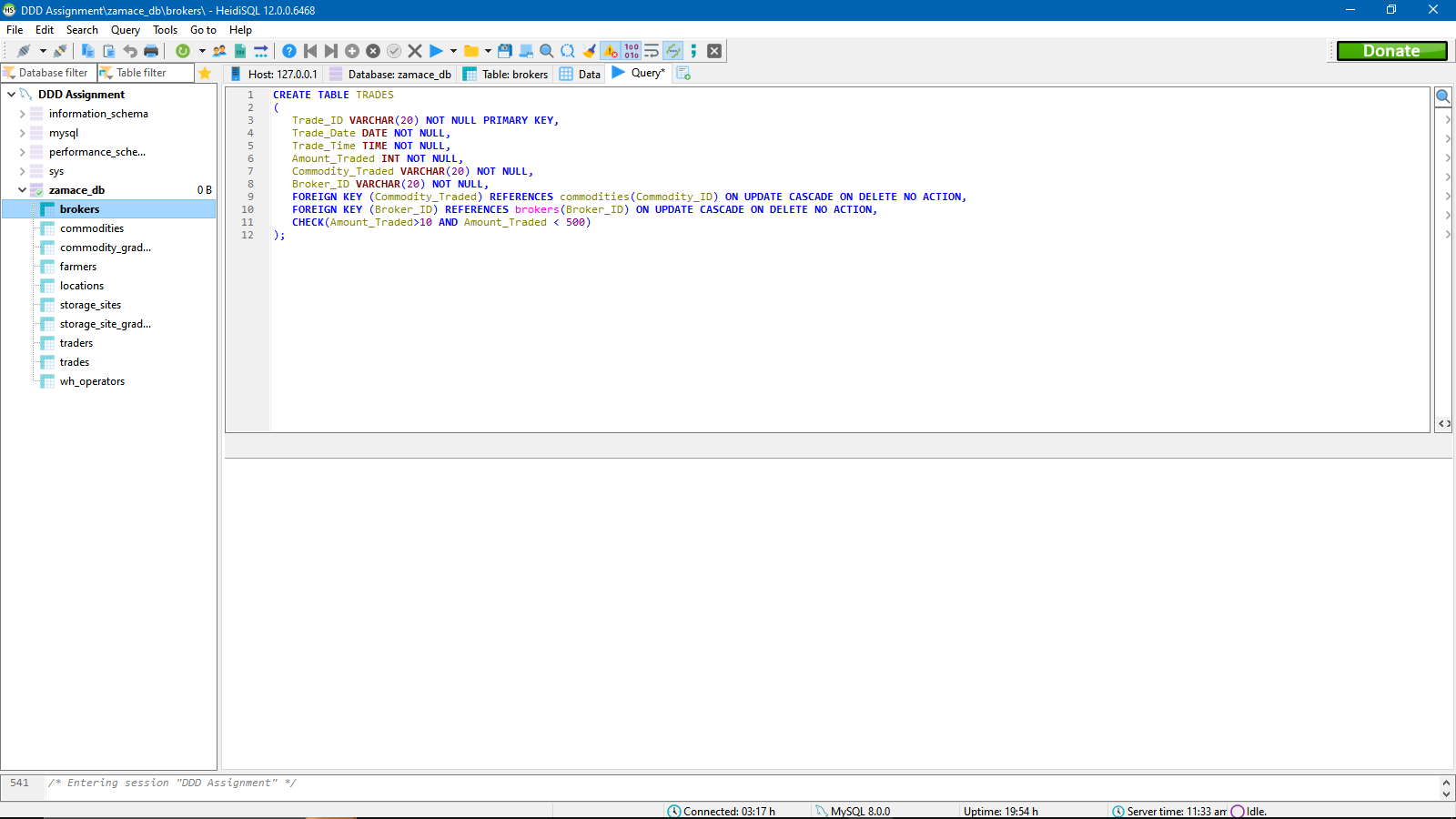
STORAGE SITES TABLE



COMMODITIES TABLE



**TRADES** TABLE

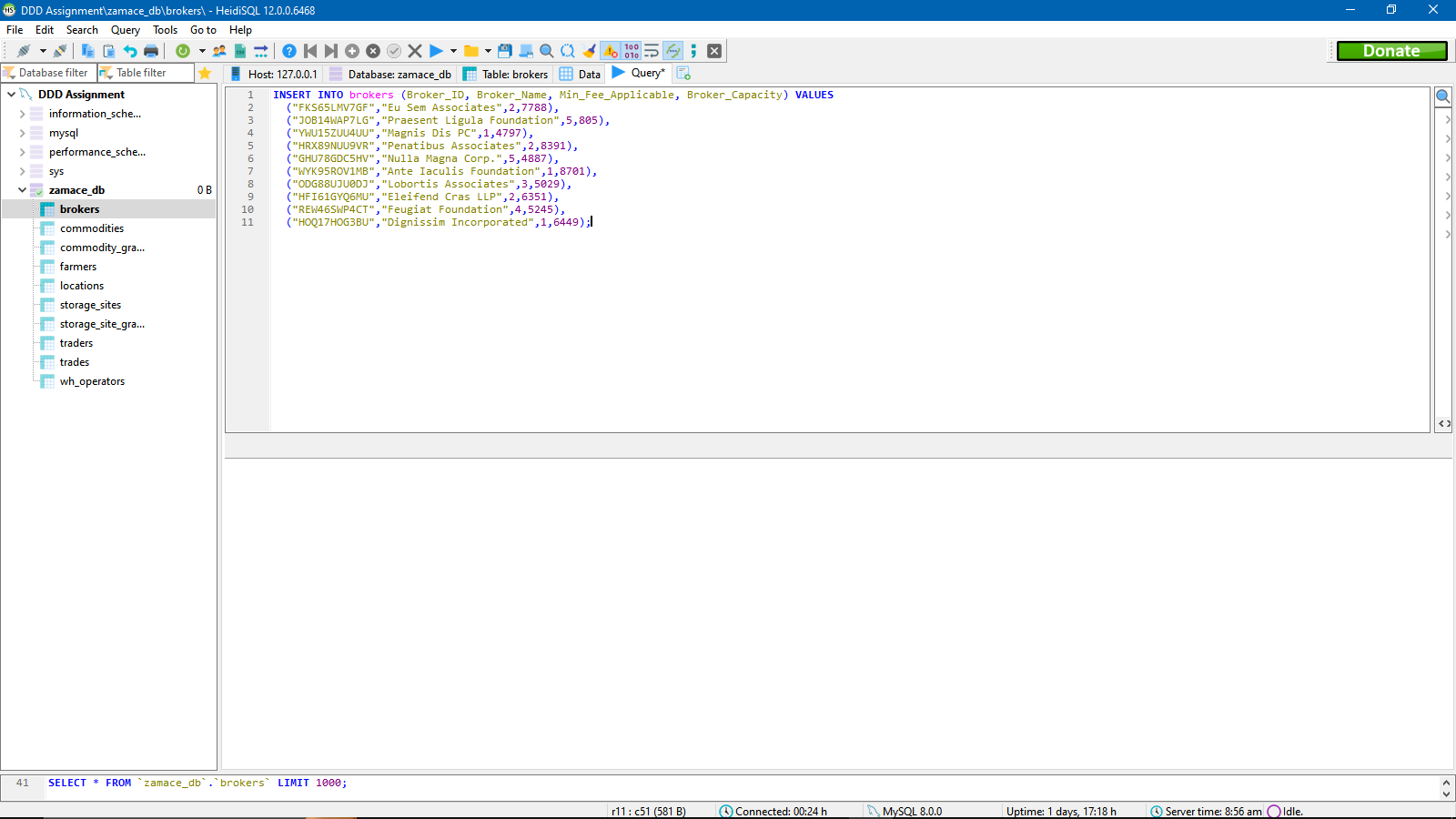


**How I developed the scripts, order or running and errors I encountered:**

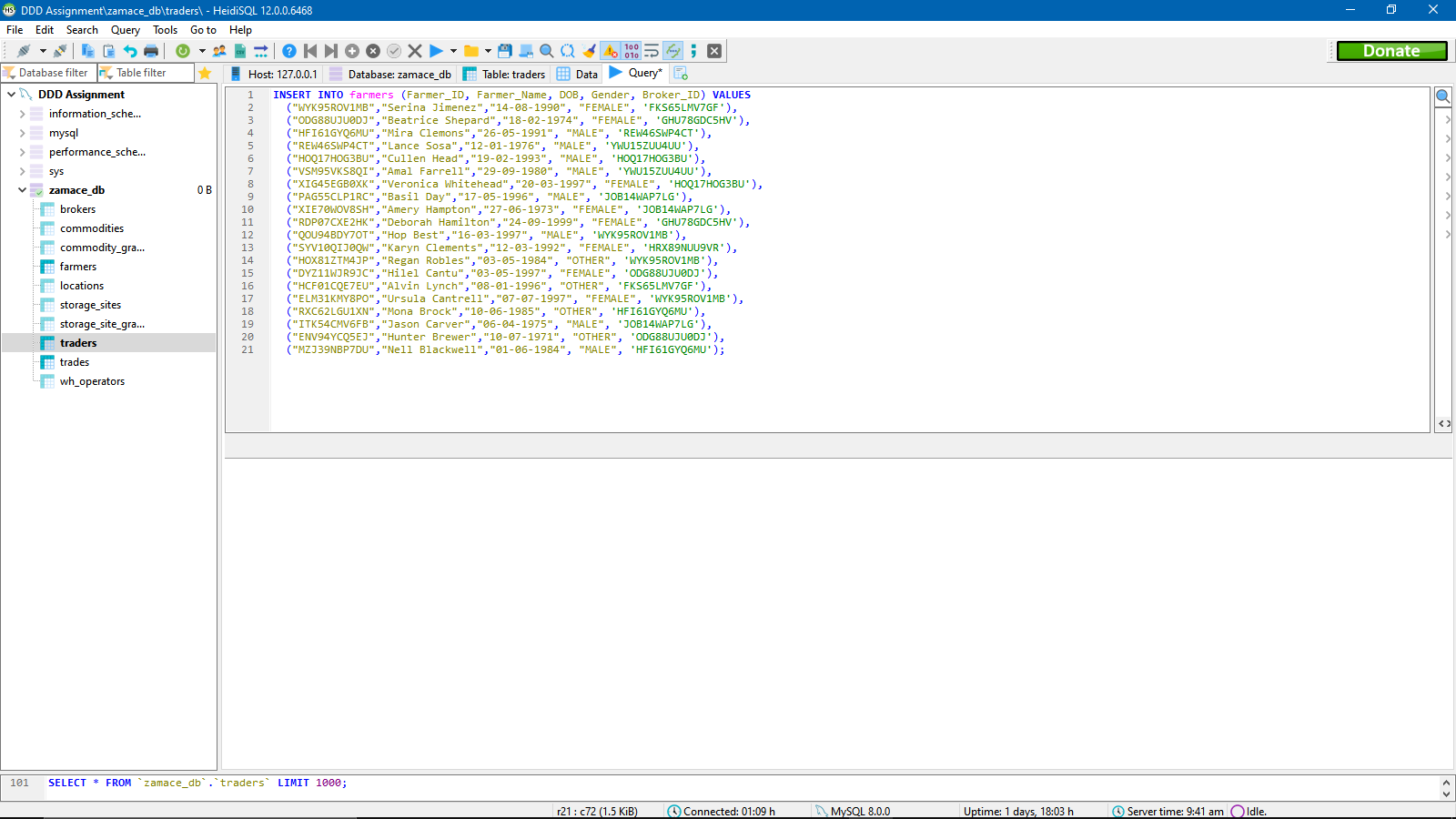
I developed the scripts mainly with the help of the data dictionary and the Entity relational diagram. I checked for constraints, data types etc. and I translated them into SQL code, The order I which I created the tables was such that, I created tables that had no foreign keys, first, then I created the tables with foreign keys to avoid referencing errors. The errors I encountered were that some of the domain constraints were not supported in the SQL version I was using so I had to look for workarounds. Note that there might still be changes to be made in the future using ALTER TABLE.

**TASK (6) (Data Population)**

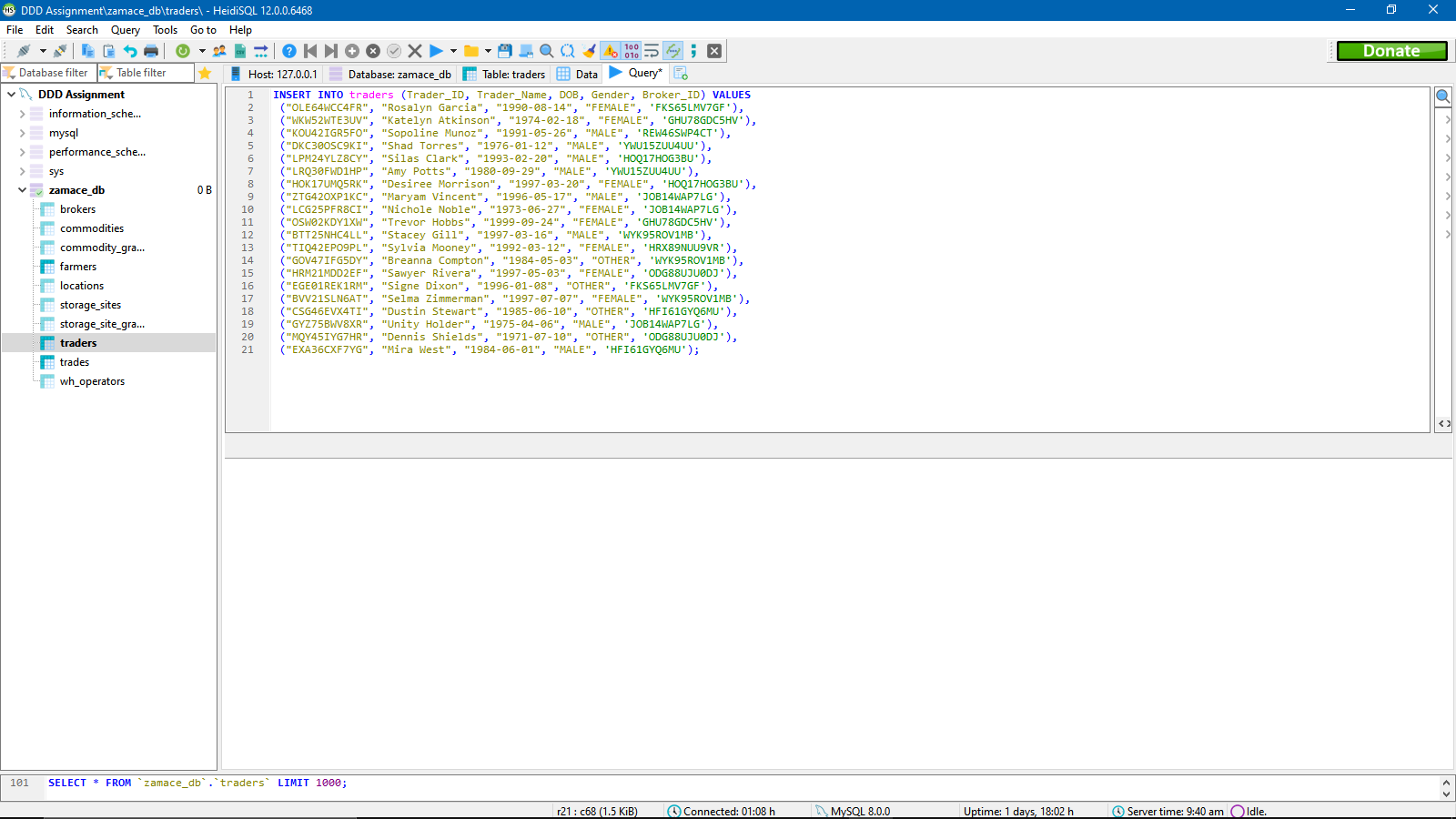
BROKERS TABLE



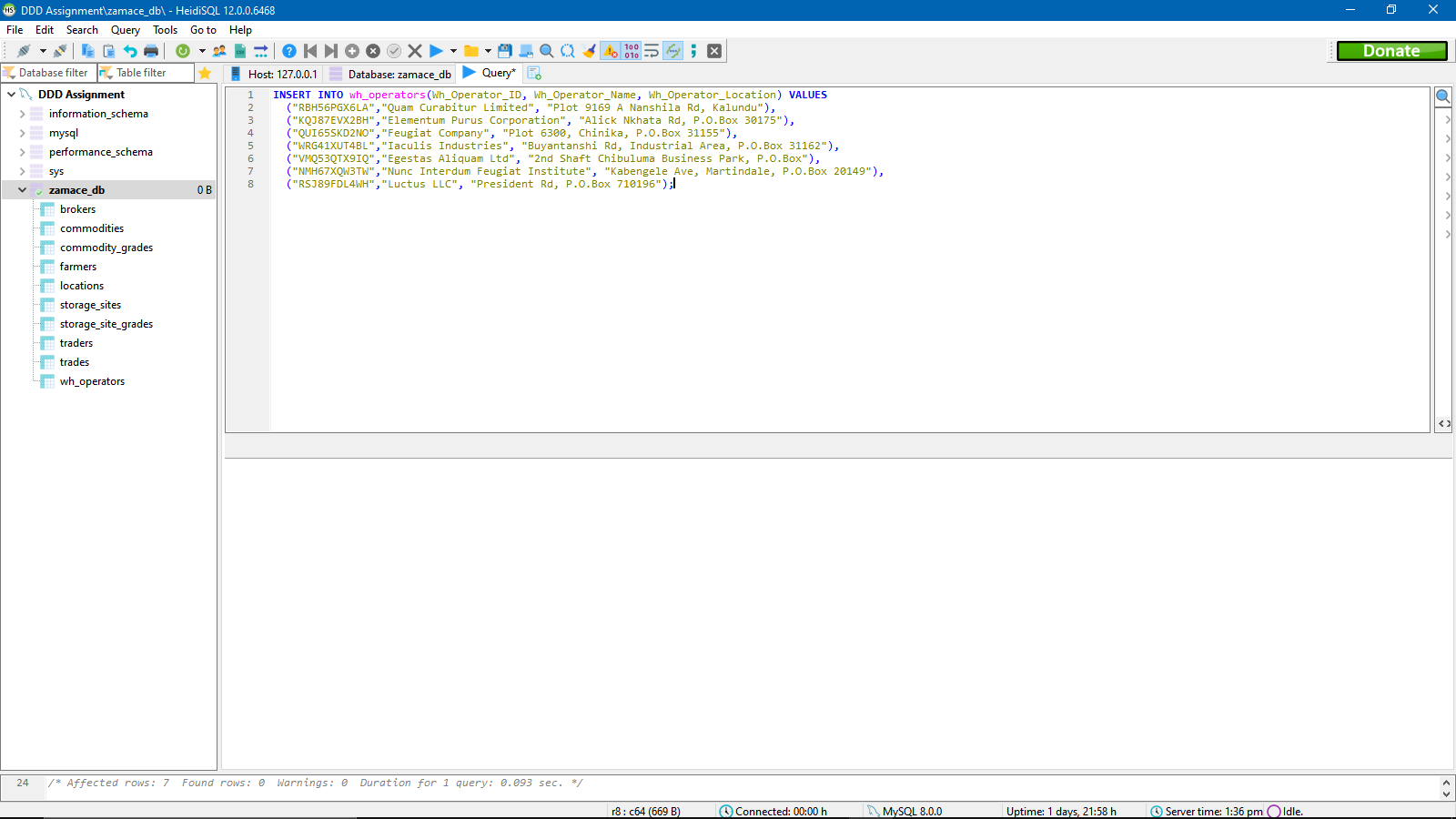
FARMERS TABLE



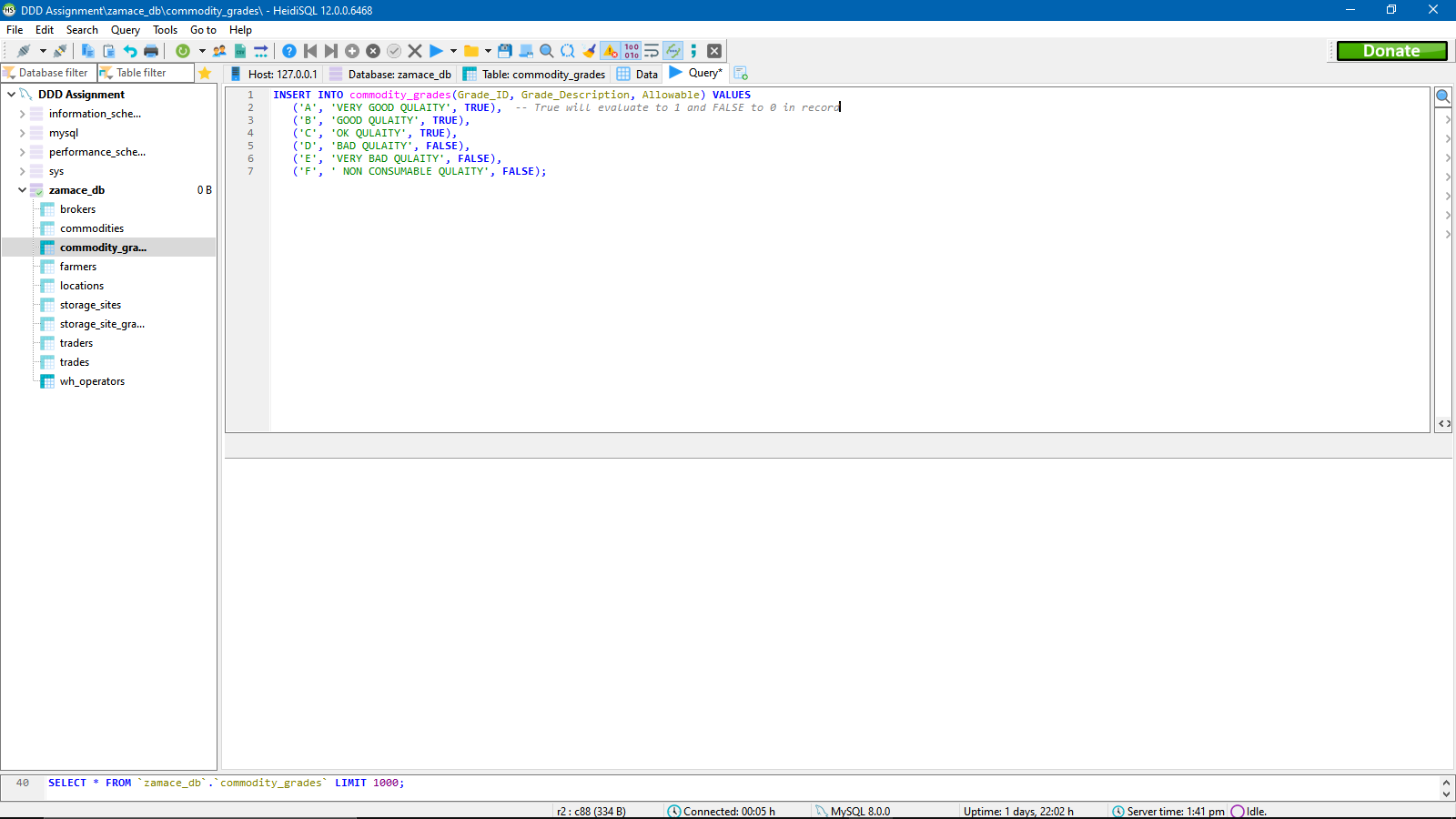
TRADERS TABLE



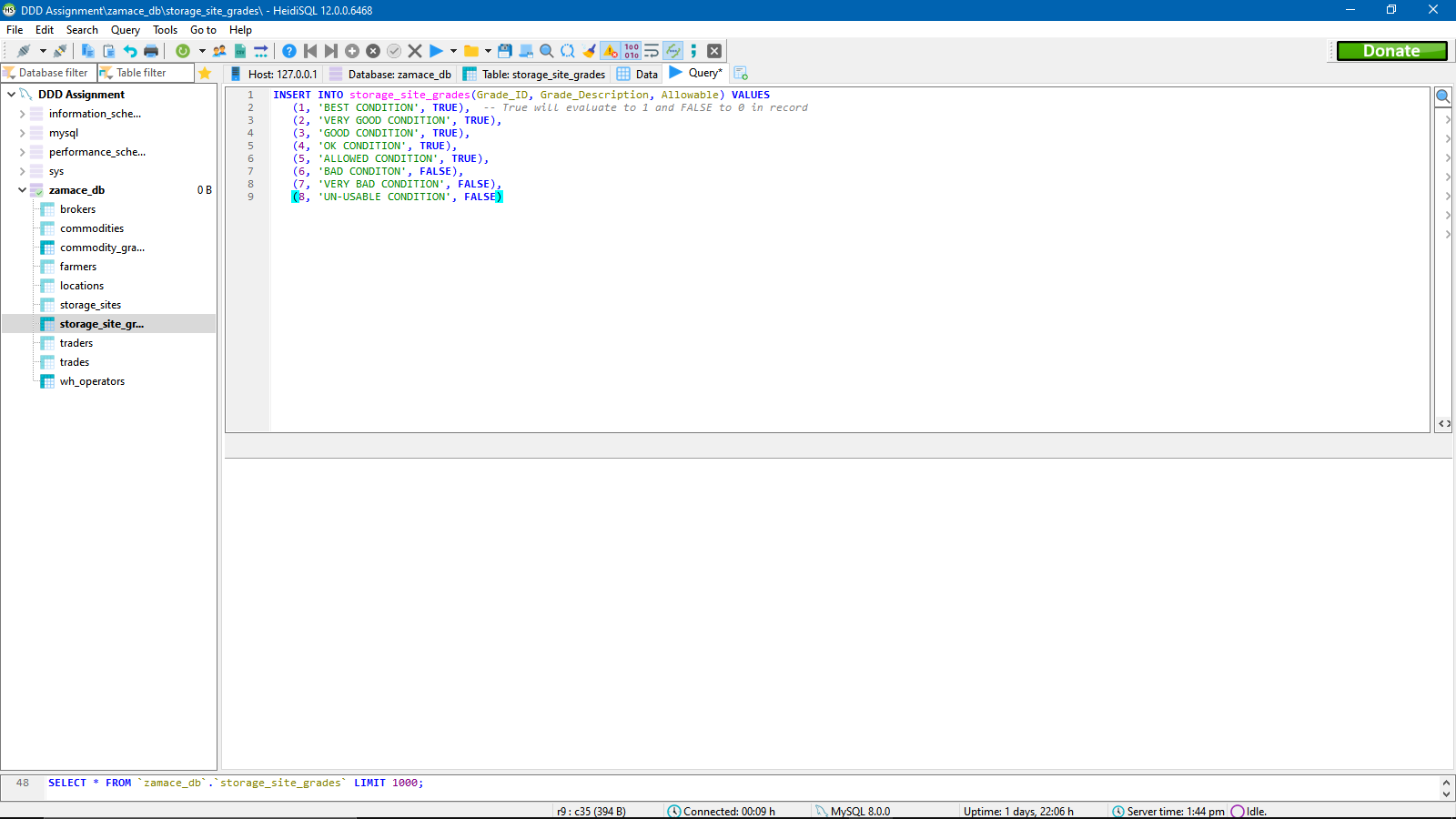
WAREHOUSE OPERATORS TABLE



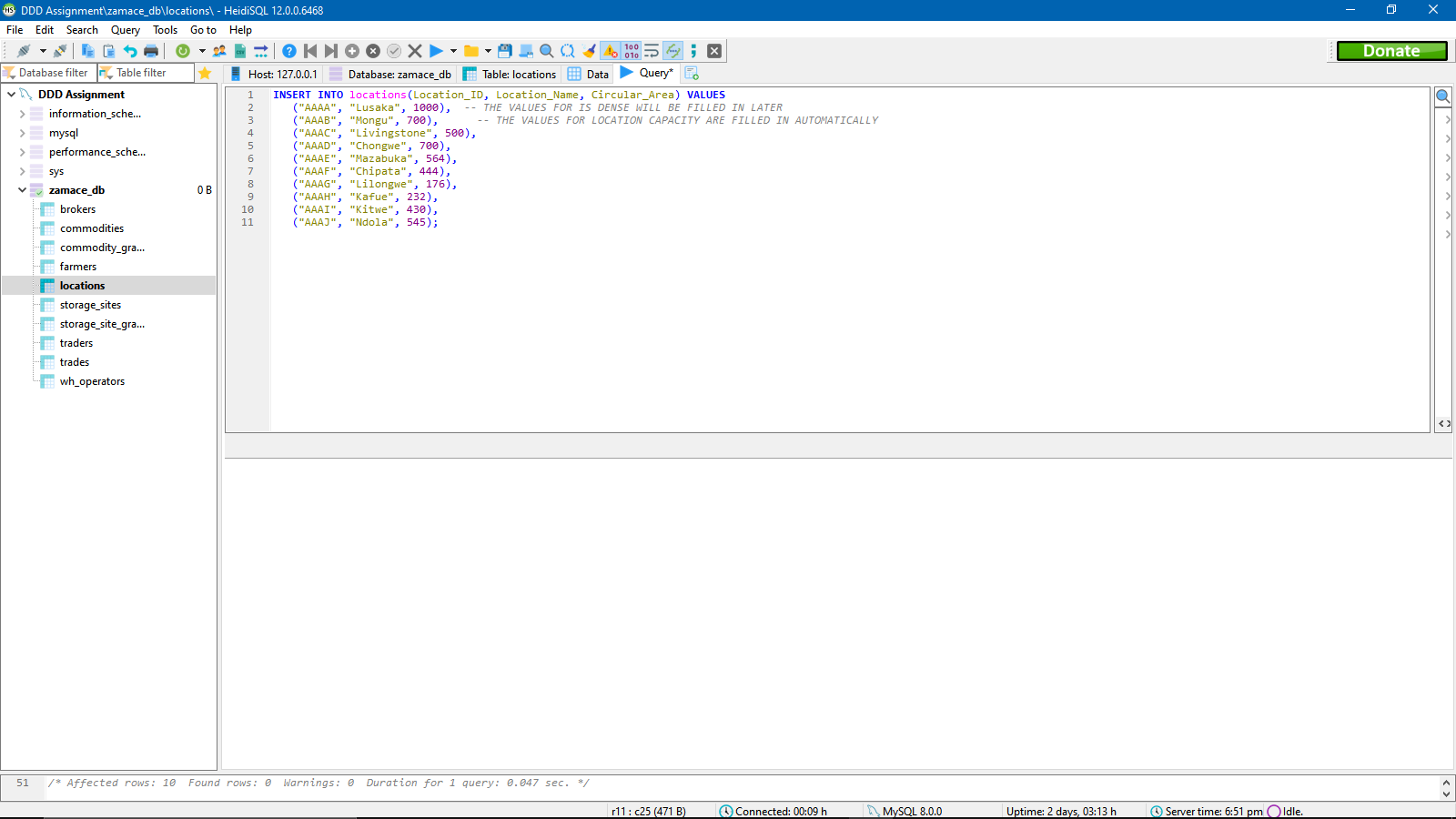
COMMODITY GRADES TABLE



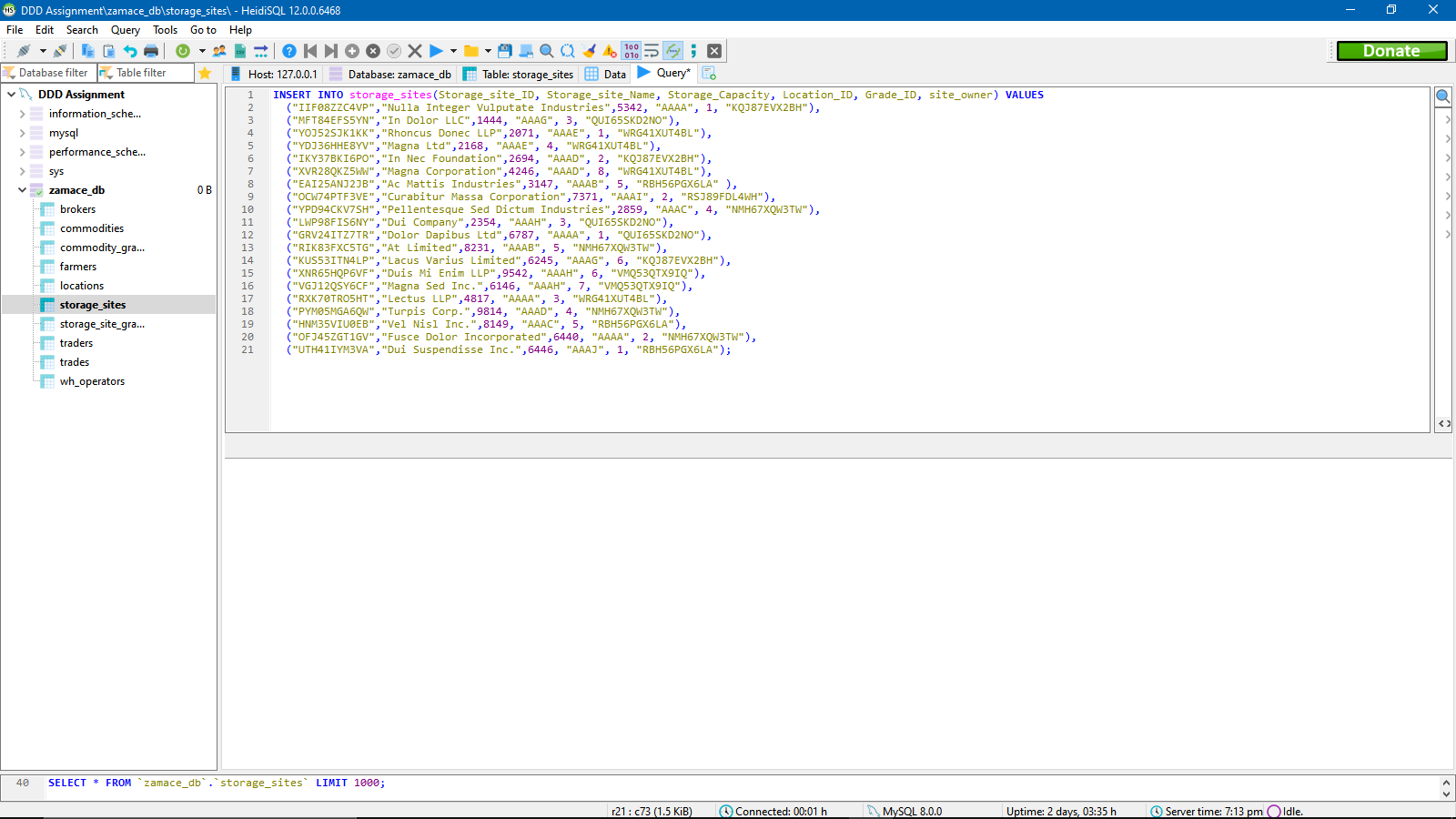
STORAGE SITE GRADES TABLE



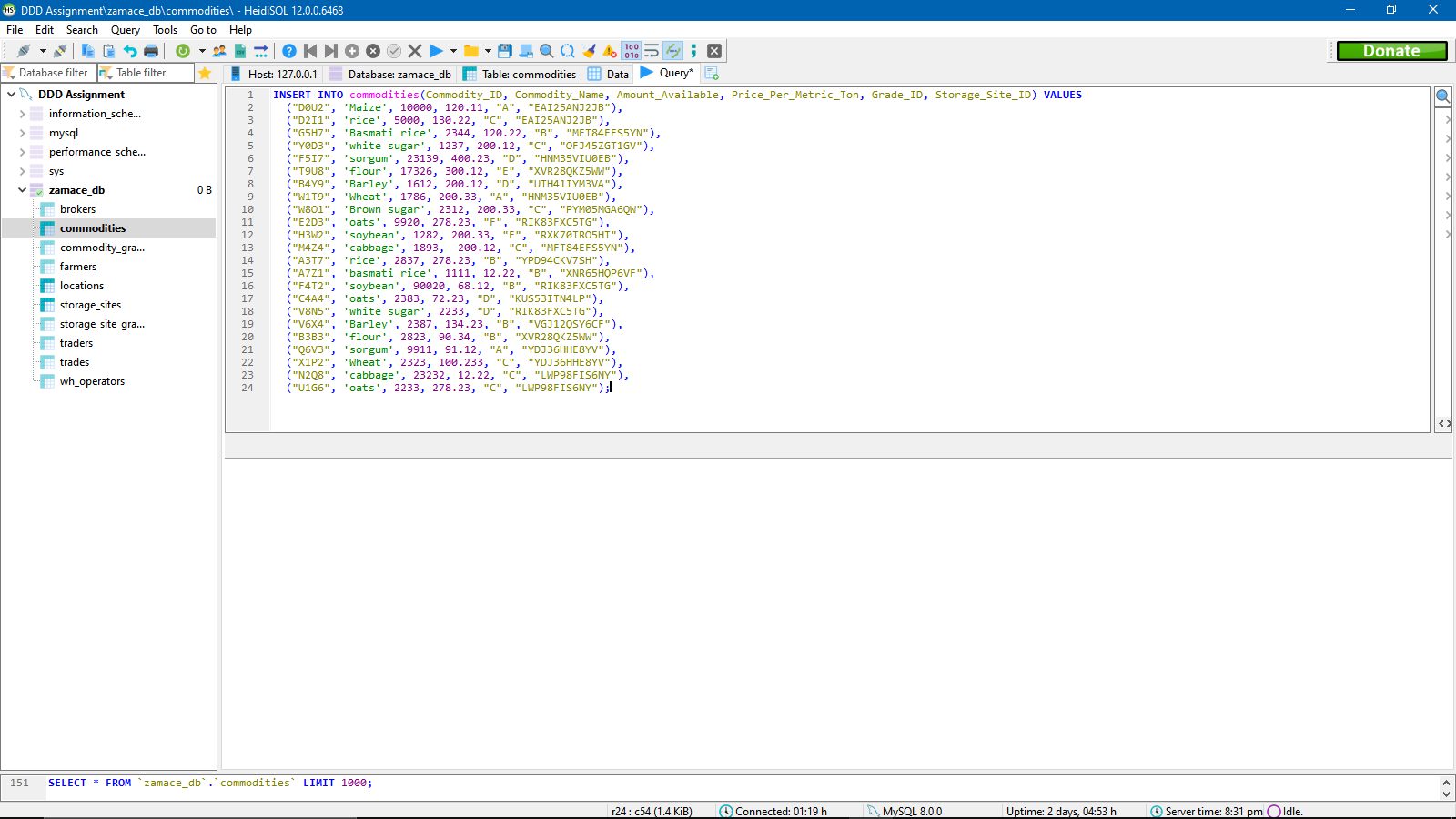
LOCATIONS TABLE



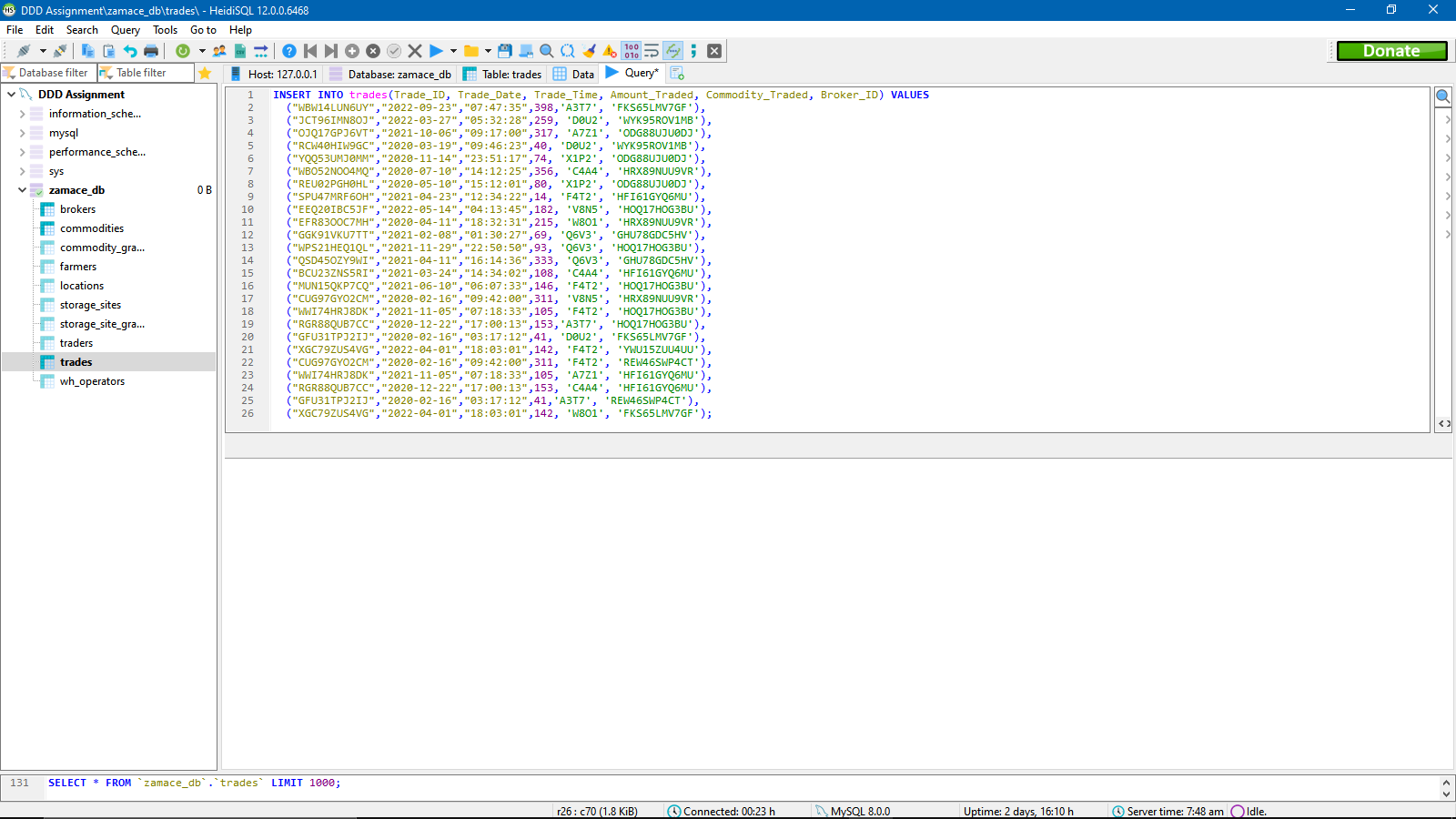
STORAGE SITES TABLE



COMMODITIES TABLE



TRADES TABLE



**How I developed the scripts, order or running and errors I encountered:**

I developed the scripts using relevant data, I used online data generators to help me come up with large sums of relevant data in order to ease the process of inserting data. Then I simply took the data and placed it into the insert statement.

The order of running the scripts was the same as the order of creating the tables. I entered data in tables with no foreign keys first and then went to the other tables.

The issues I encountered with that I had to copy foreign keys in the foreign key columns and I had to make sure I copied each foreign key correctly otherwise the record would not get entered.

**TASK (7) (Enhancement using SQL)**

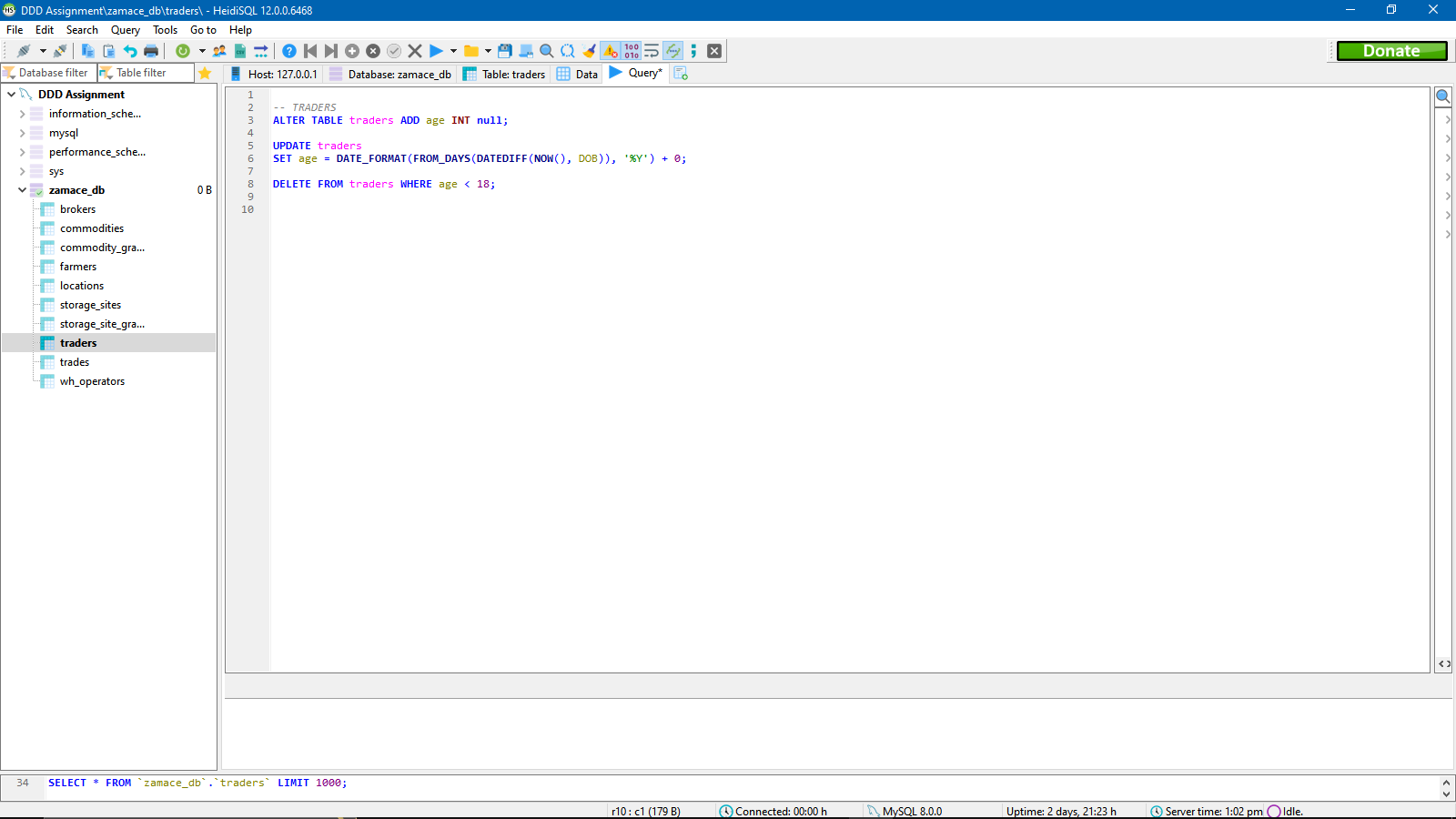
There a few potential changes that could be made to the original database design. Namely;

* We can add an AGE column to the farmers and trader’s table
* We can add a check constraint to the trades table on the amount traded column so that the amount traded is not greater than the amount available in the commodities table.
* Add a column in location table which gets number of storage sites in that location.

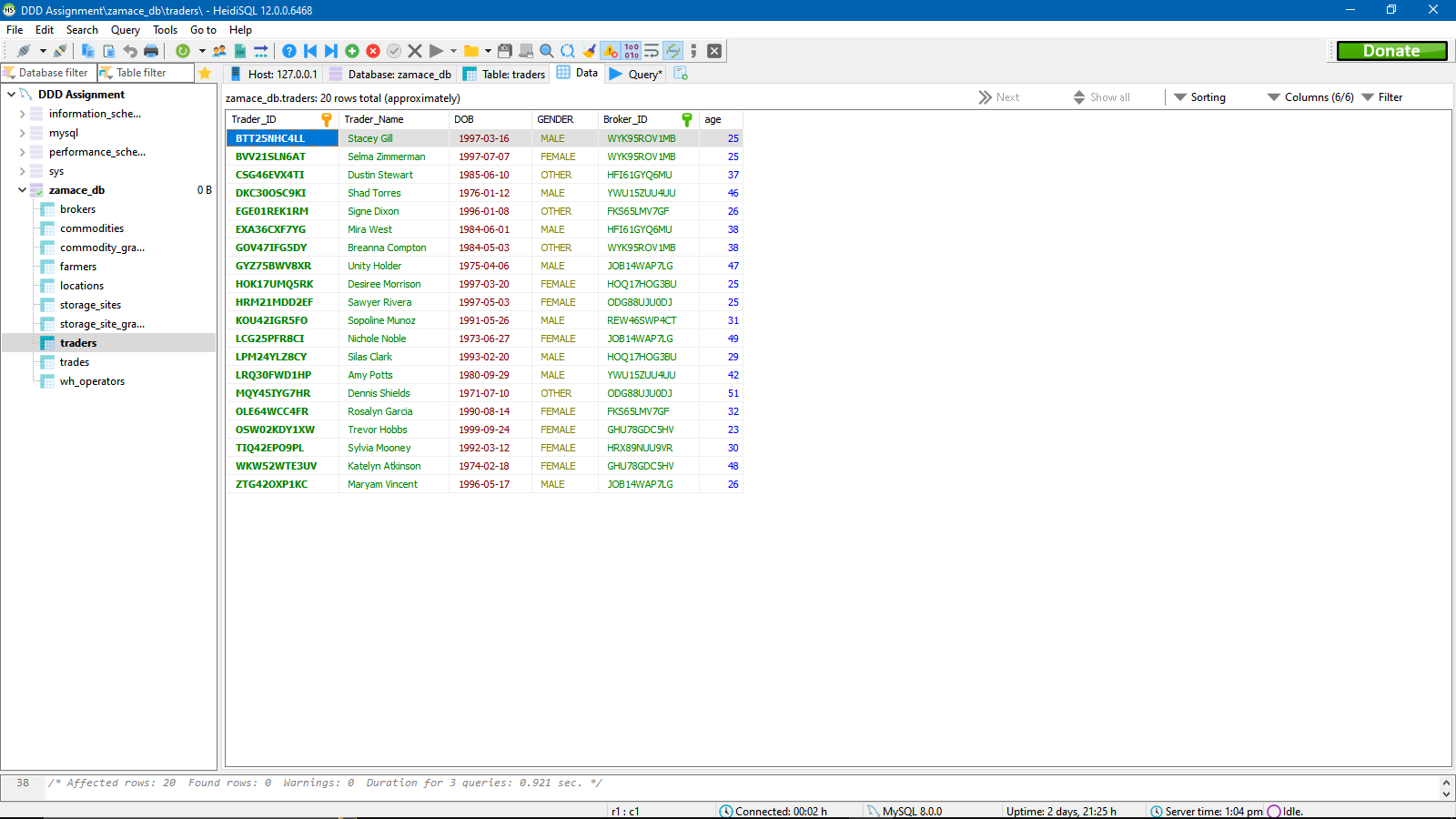
The reason for the first change is that it will become easier for companies to check the age of their users and only allow users above age 18.

The reason for the second change is brokers will only be able to trade less than the amount of a certain commodity that is available in the storage site otherwise it will give an error.

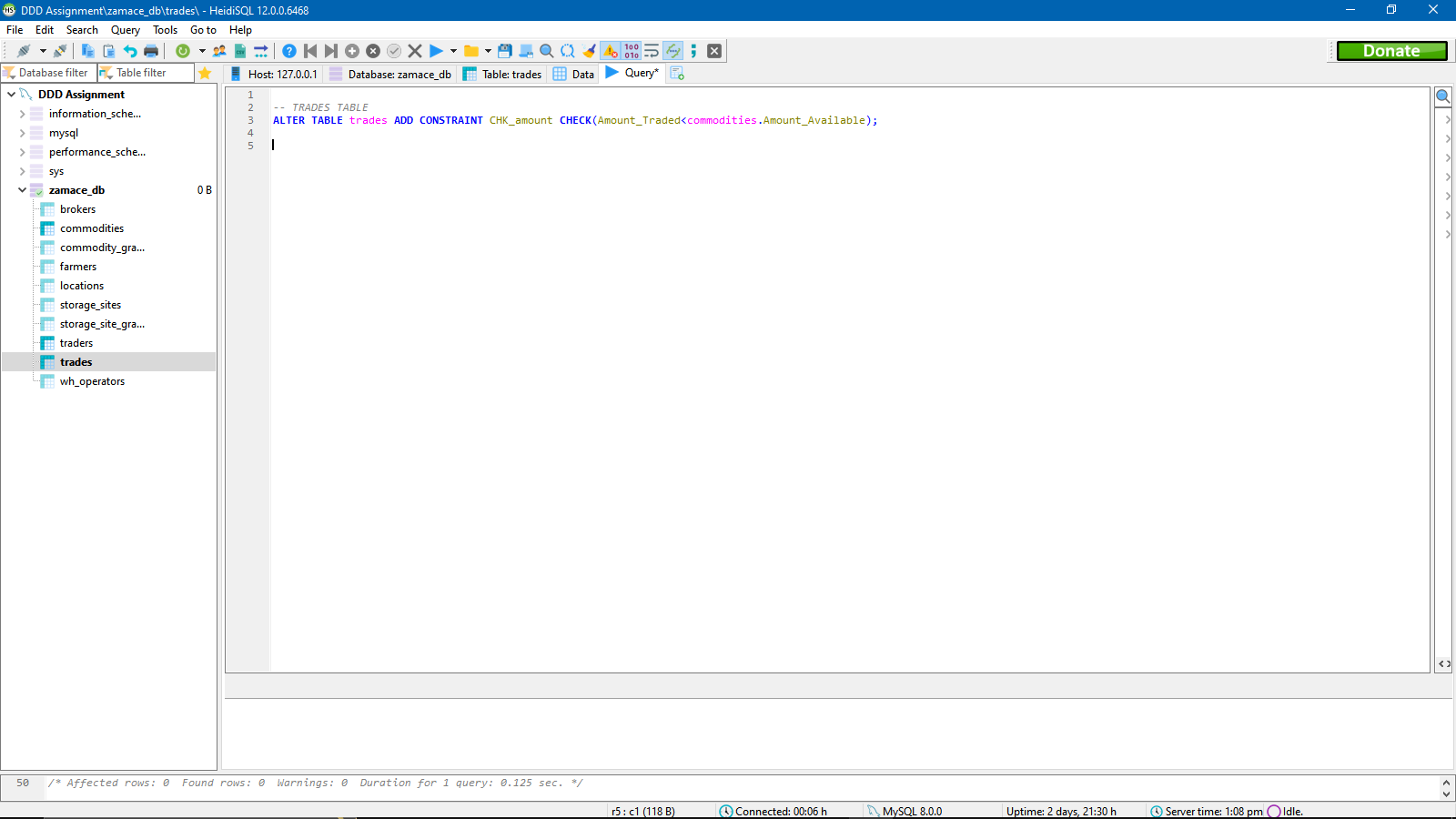
Using SQL to implement first change: (Same done for farmers table).



RESULTS OF FIRST CHANGE (Similar results for farmers table)

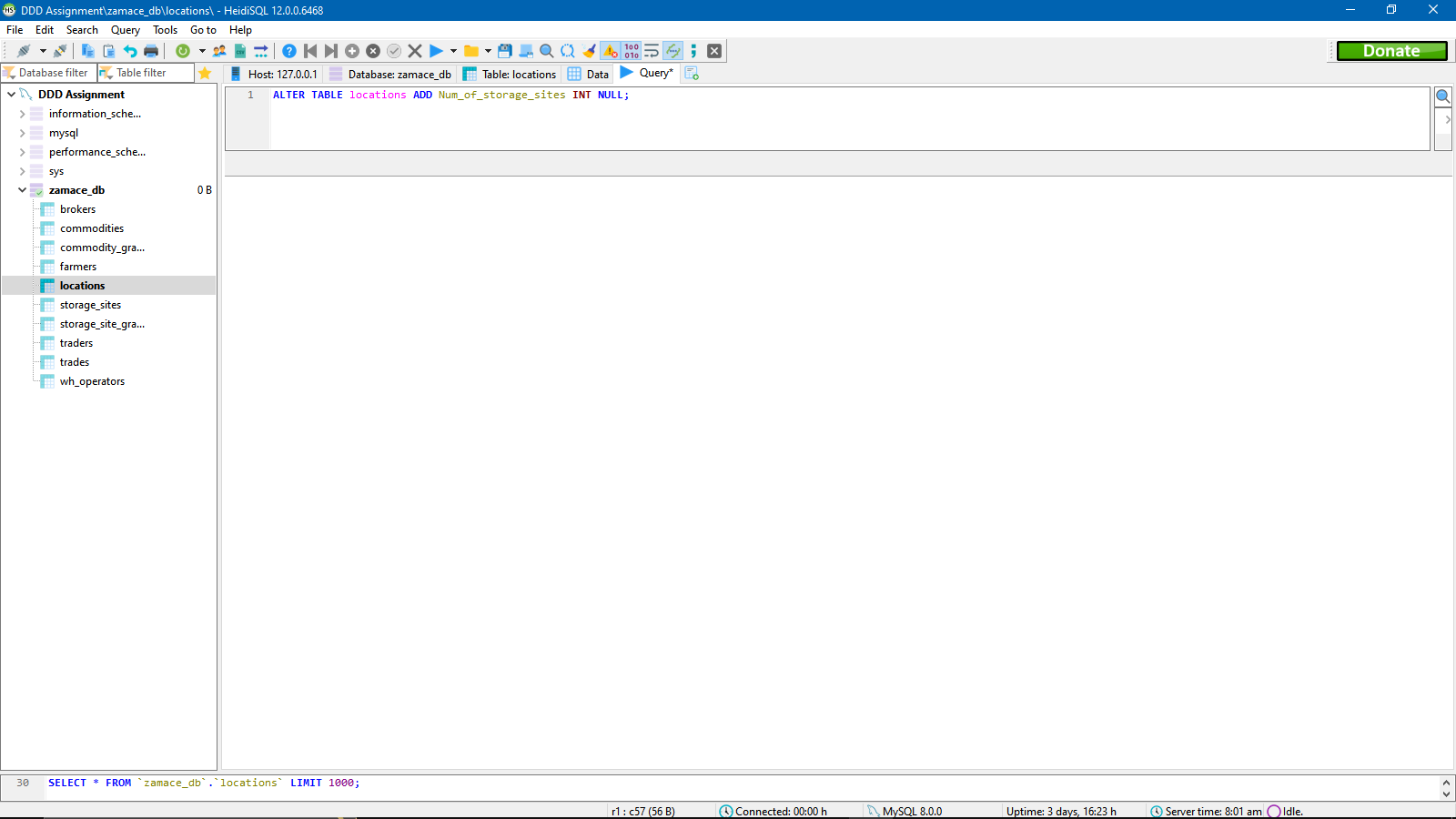


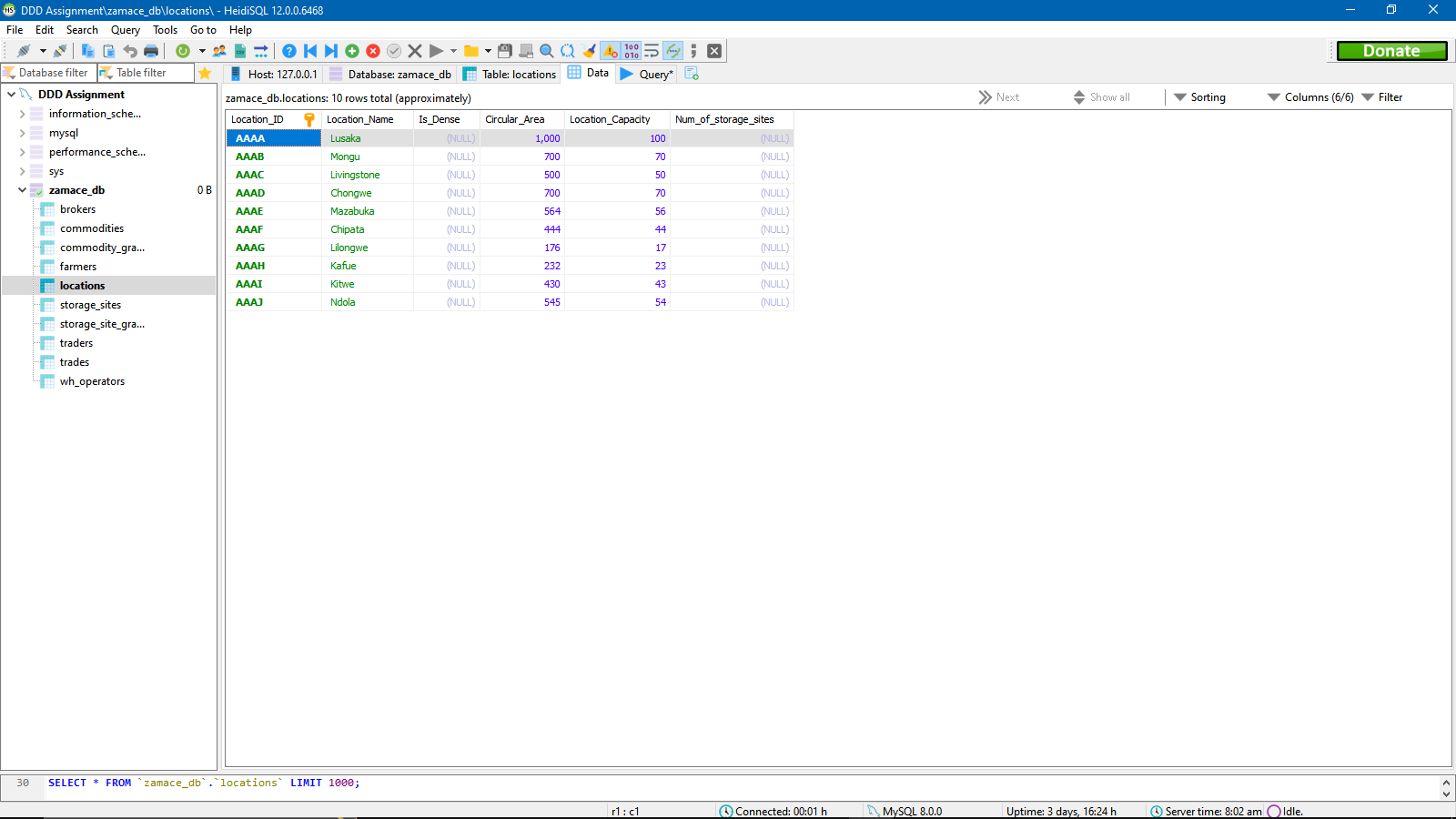
Using SQL To implement second change;



By adding this constraint to the trades table, we can now only allow trades that have the amount traded less than the amount of that commodity available,

Using SQL to implement third change.



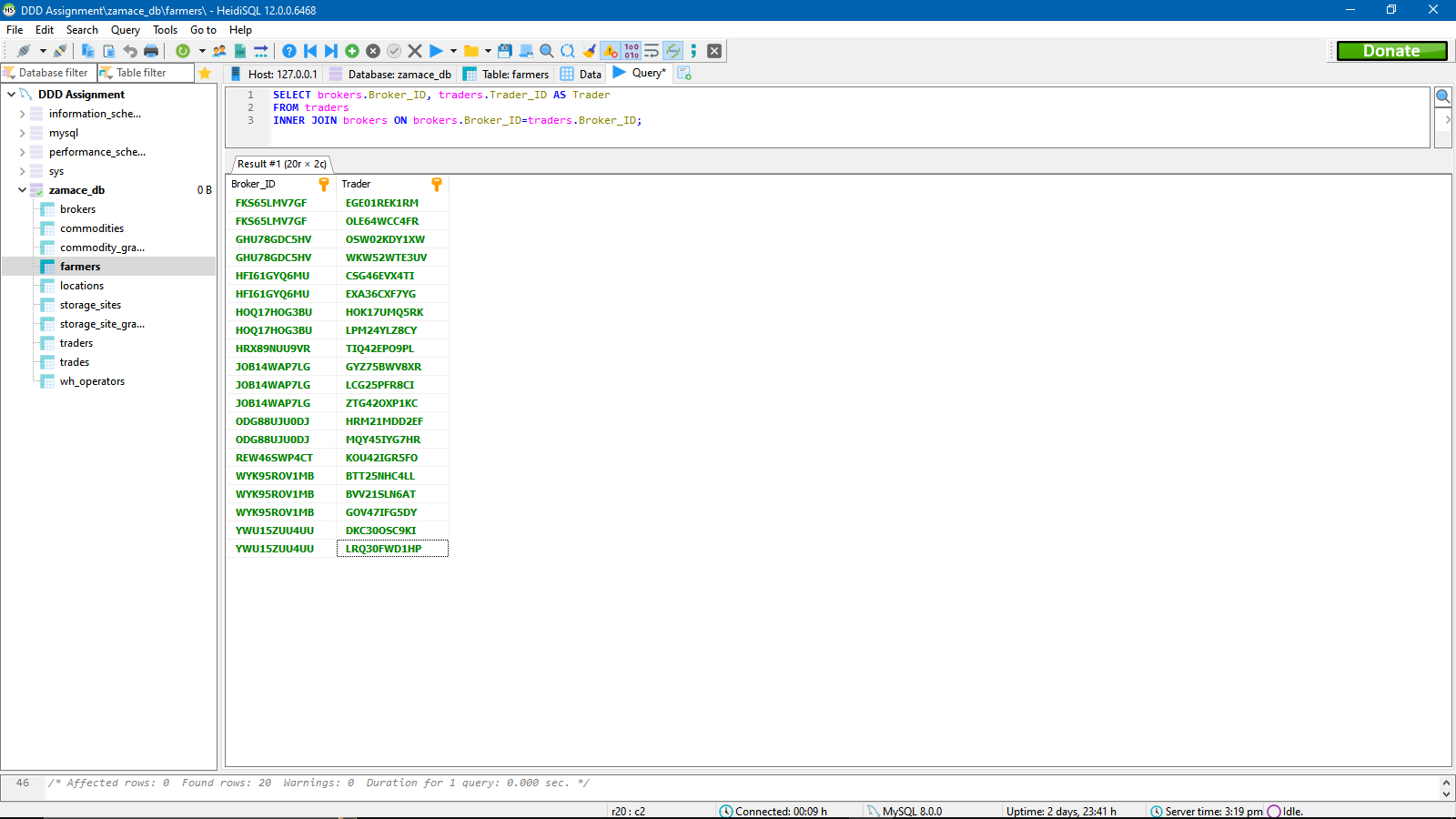


**TASK (8) (SQL REPORTS)**

**Query 1:**

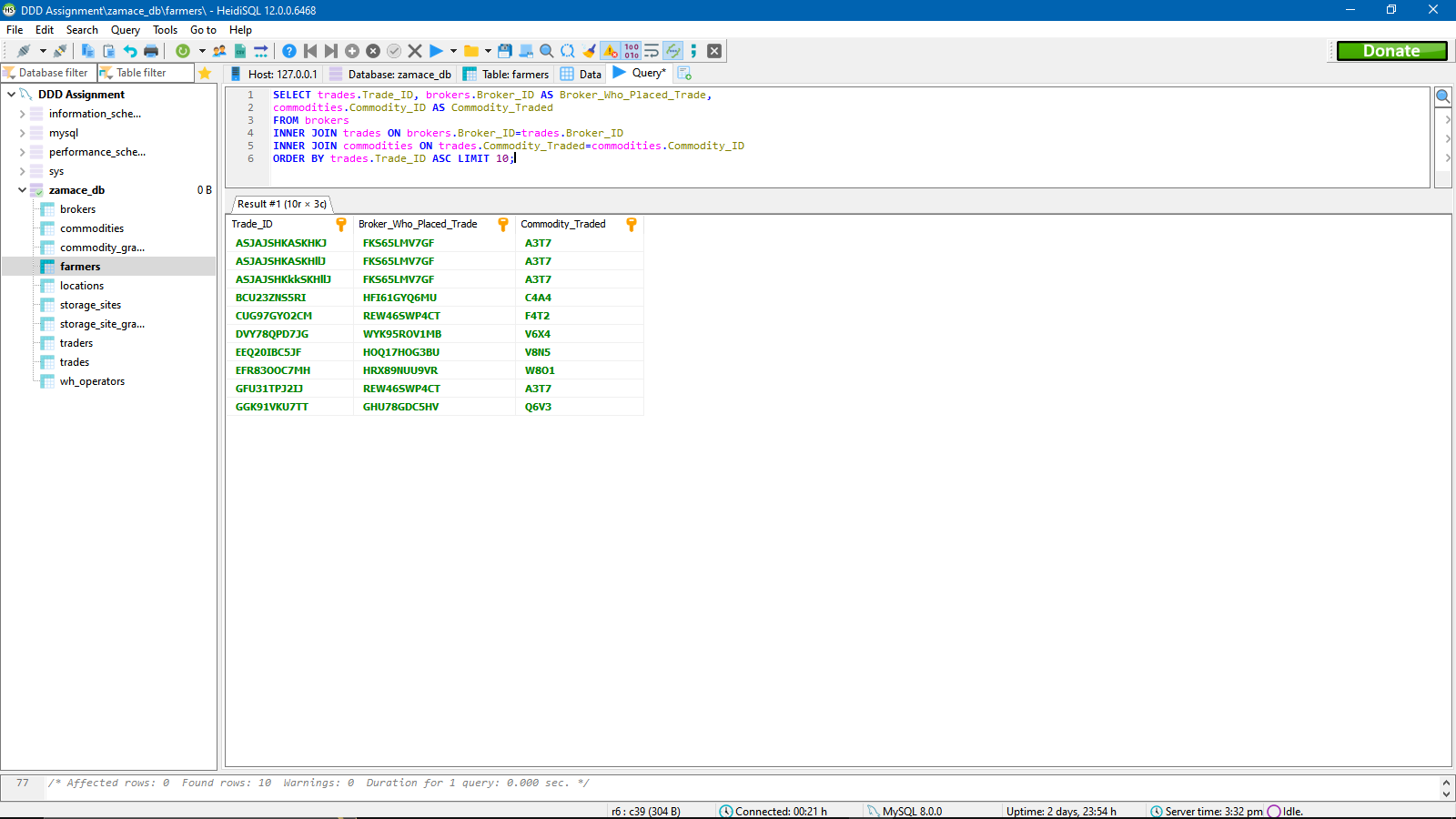
This query will inner join the brokers and traders table on condition that the traders broker id is = brokers broker id We want to know the brokers each trader use to place trades on ZAMACE, this way we can catch people who are using brokers who are not registered with ZAMACE. This same query can be done on farmers table as well.

**SQL QUERY WITH ITS RUNTIME RESULTS ON THE BOTTOM IN GREEN:**



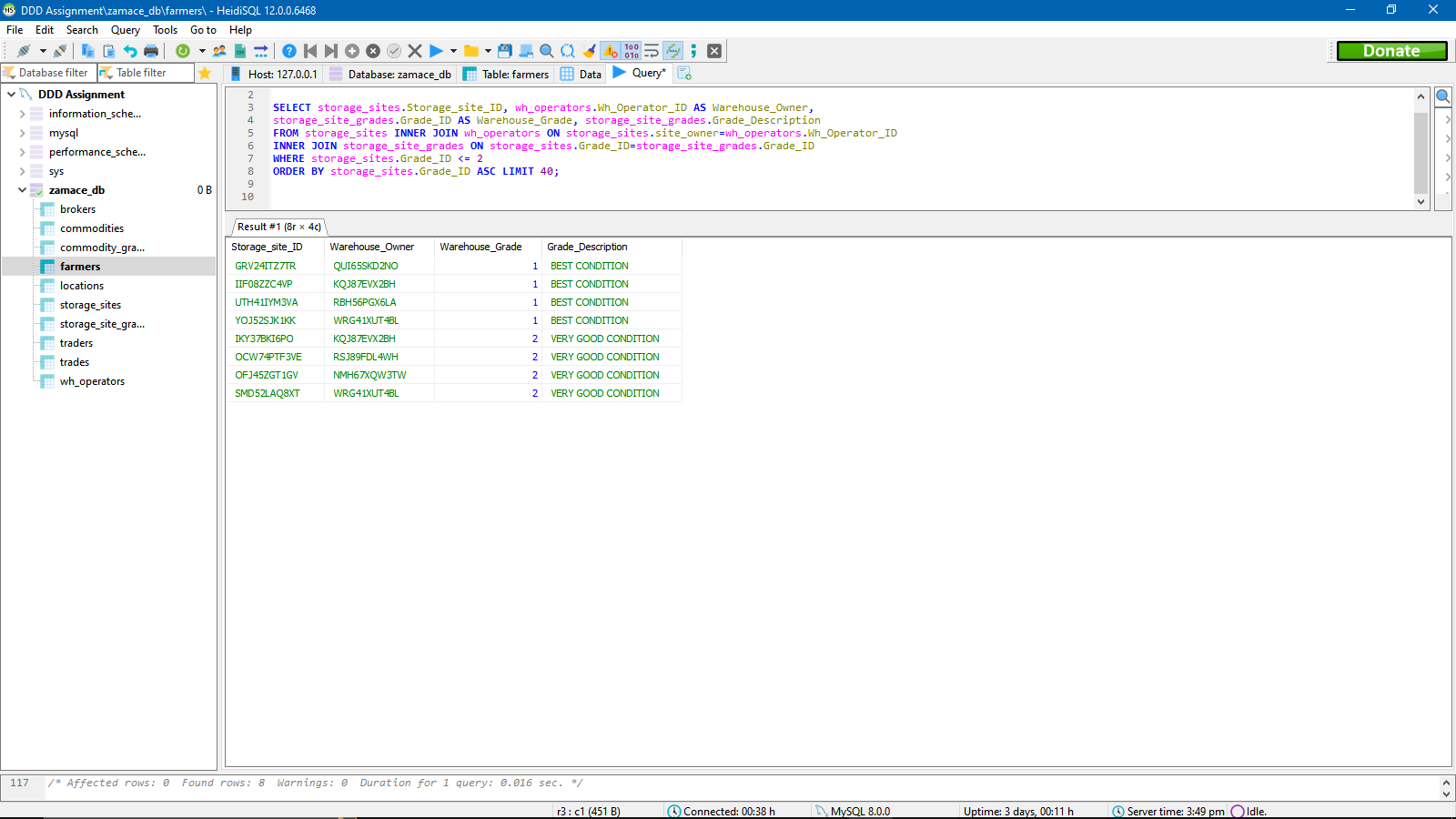
**Query2:**

In this query we will retrieve the first 10 trades that were placed by brokers, and what commodity was traded in each trade. This can be useful if we want to know what was traded by a certain broker.



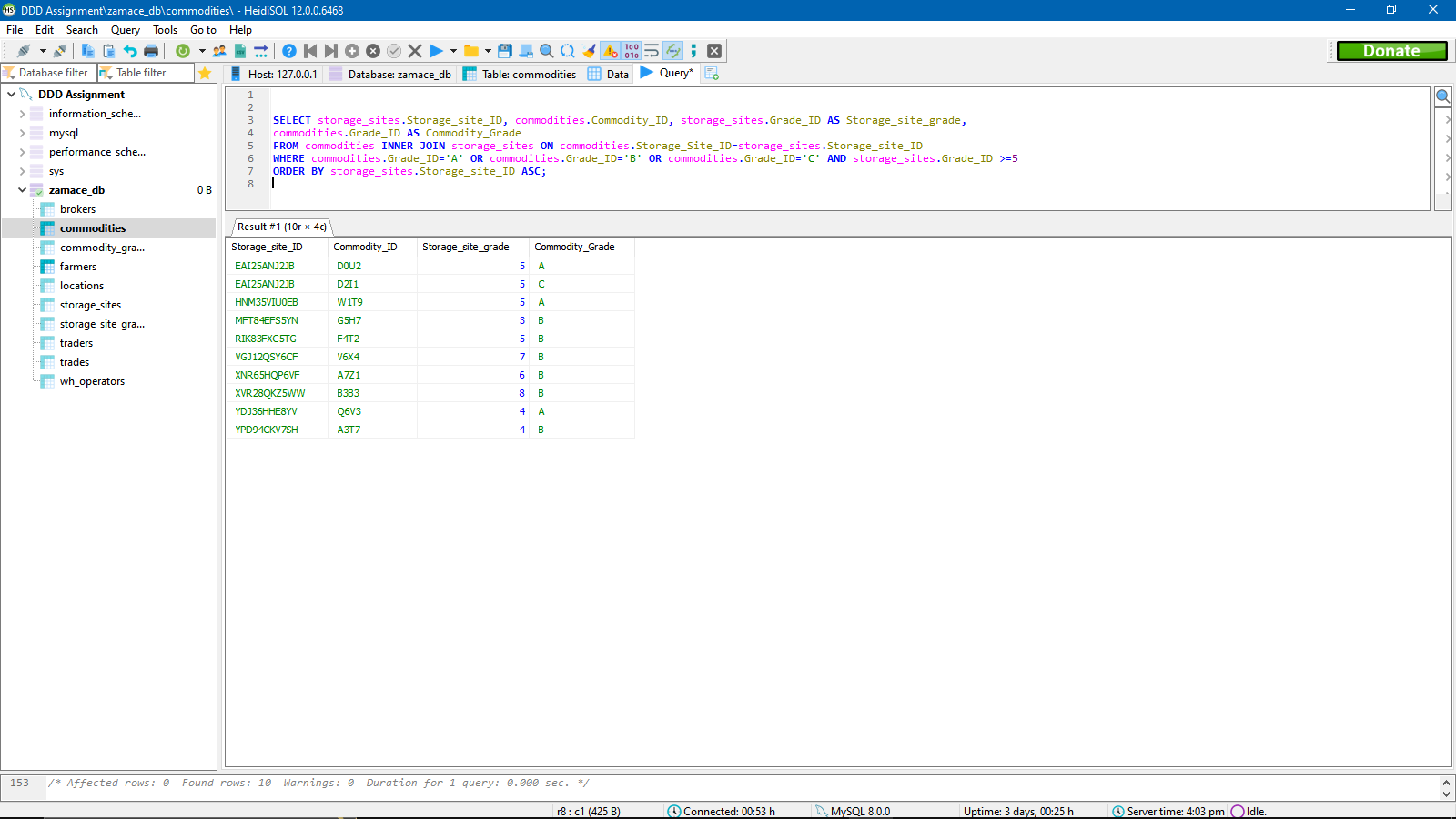
**Query 3:**

In this query we will retrieve the storage sites along with their owners and grades, but we only want those with grades that are 1 or 2. This query can be useful to find out the best storage sites that the ZAMACE ecosystem has to offer.



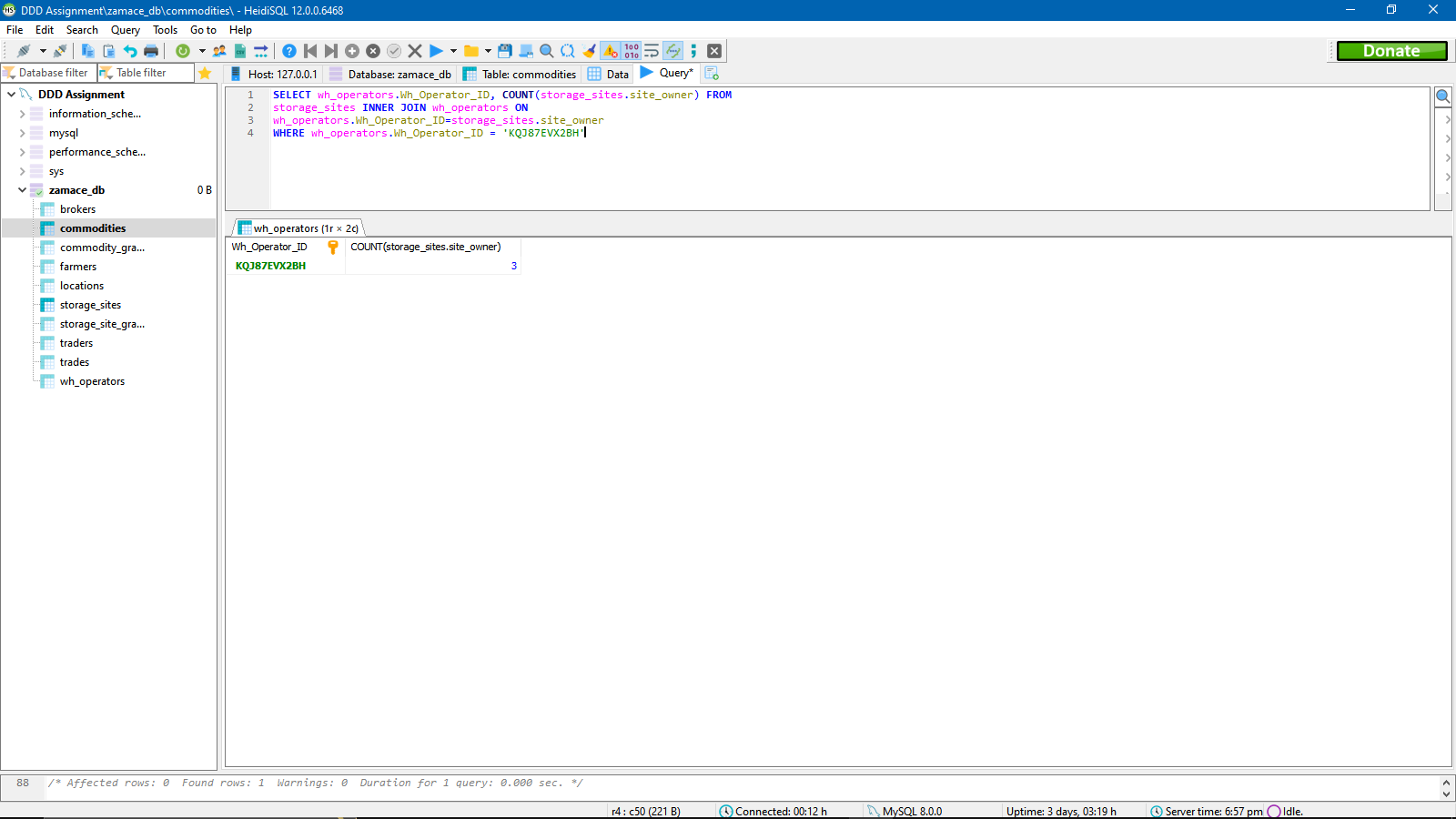
**Query 4:**

In this query we want to retrieve the commodities present in a storage site that have a grade of A, B or C, and the storage site must be above **grade 5 (which is a bad grade).** **Only Grade 1 to 4 are good grades**. This can be helpful is the company wants to move all the good grade commodities present in the bad warehouses to the good warehouses.



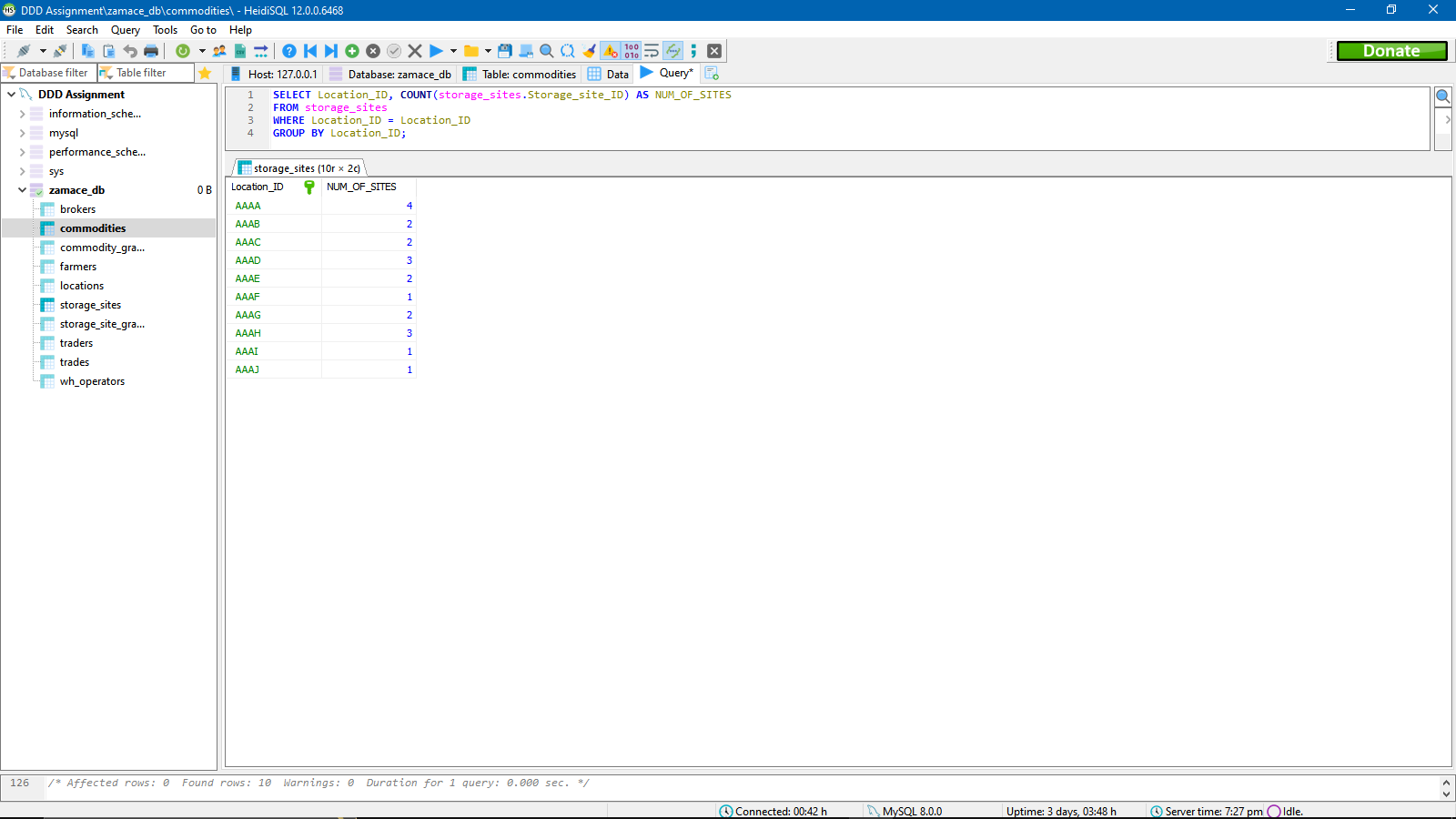
**Query 5:**

We want Count how many storage sites a certain warehouse operator owns.



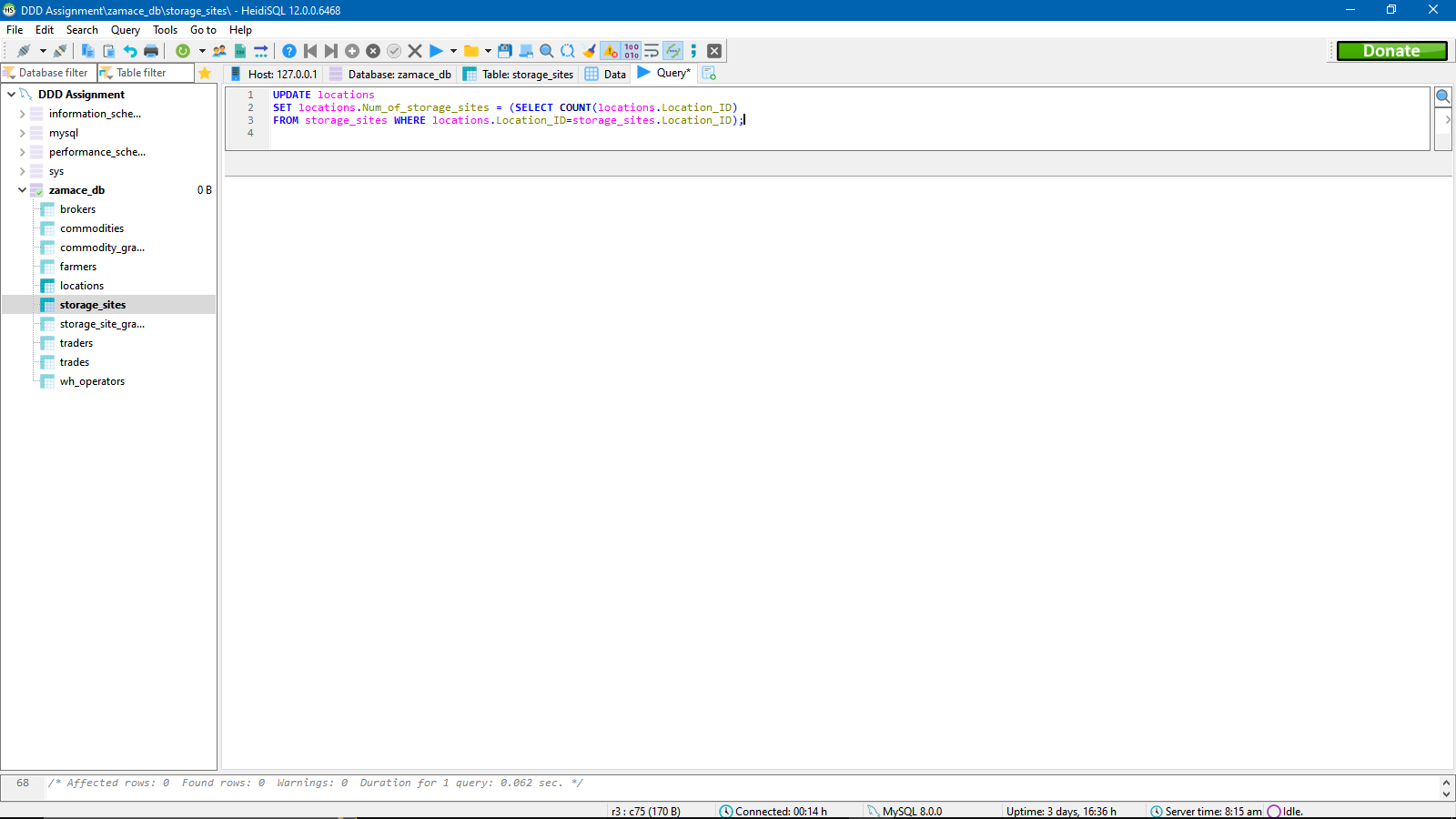
**Query 6:**

We want to get the number of sites in each location present in our database;

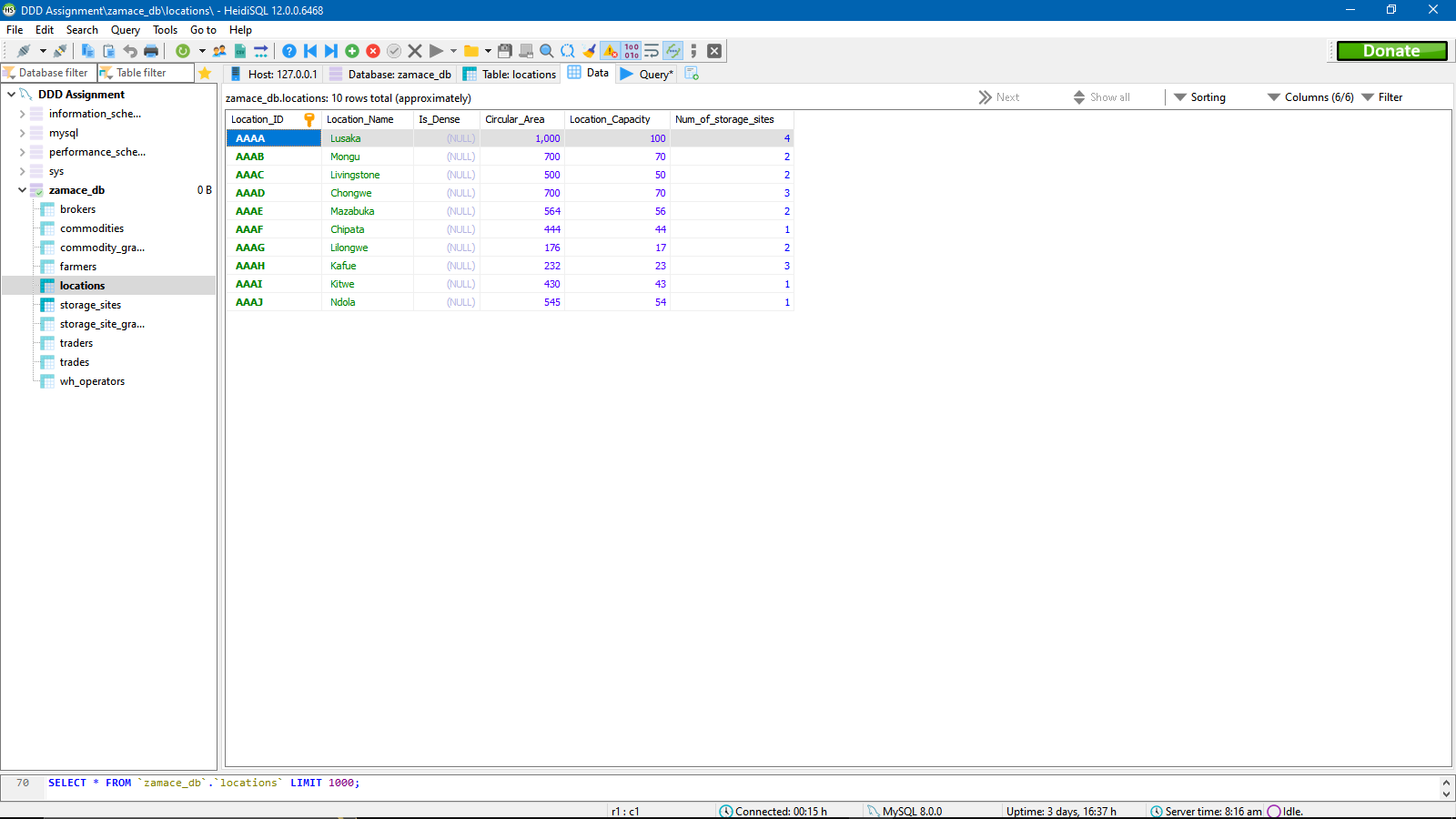


**Query 7:**

Since the “Num of storage sites column in the locations table had null values) we want to set those values to the actual number of storage sites in that location. This has a sub query

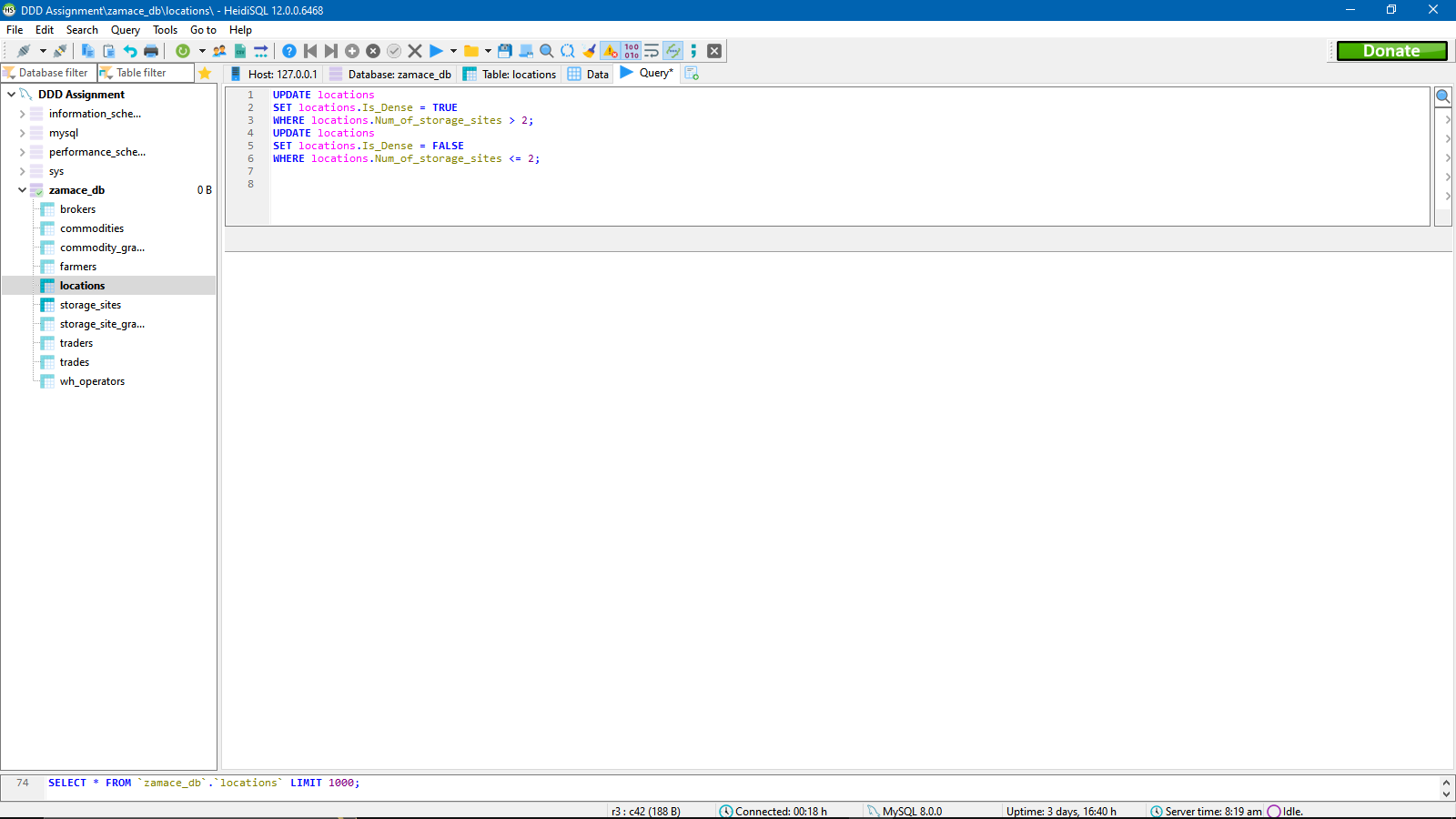


**Query 7 results:**

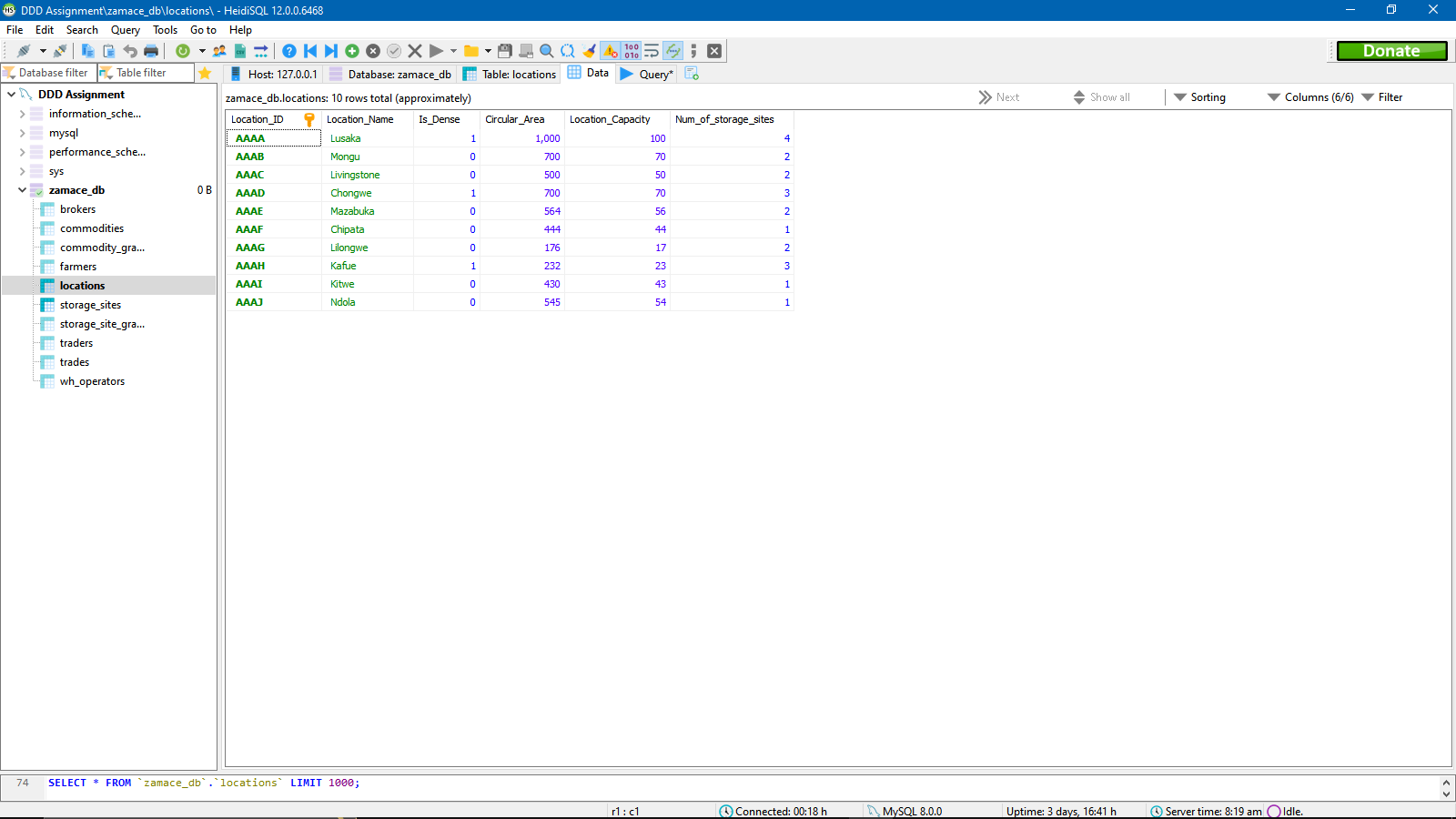


**Query 8:**

We want to also set the IS Dense column to TRUE if num of storage sites is greater than 2 otherwise FALSE.



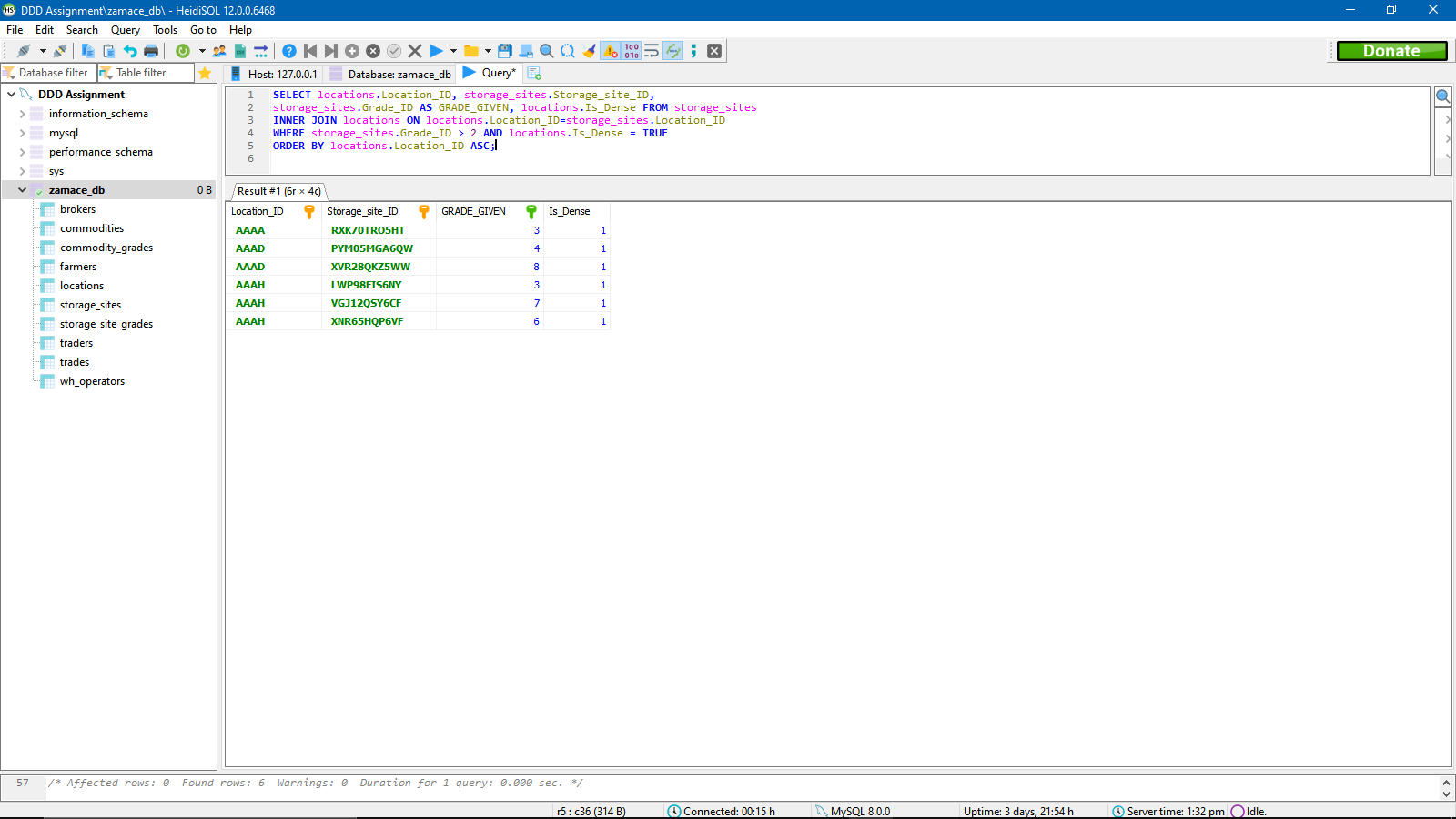
**Results:**



***TRUE evaluates to 1, FALSE evaluates to 0.***

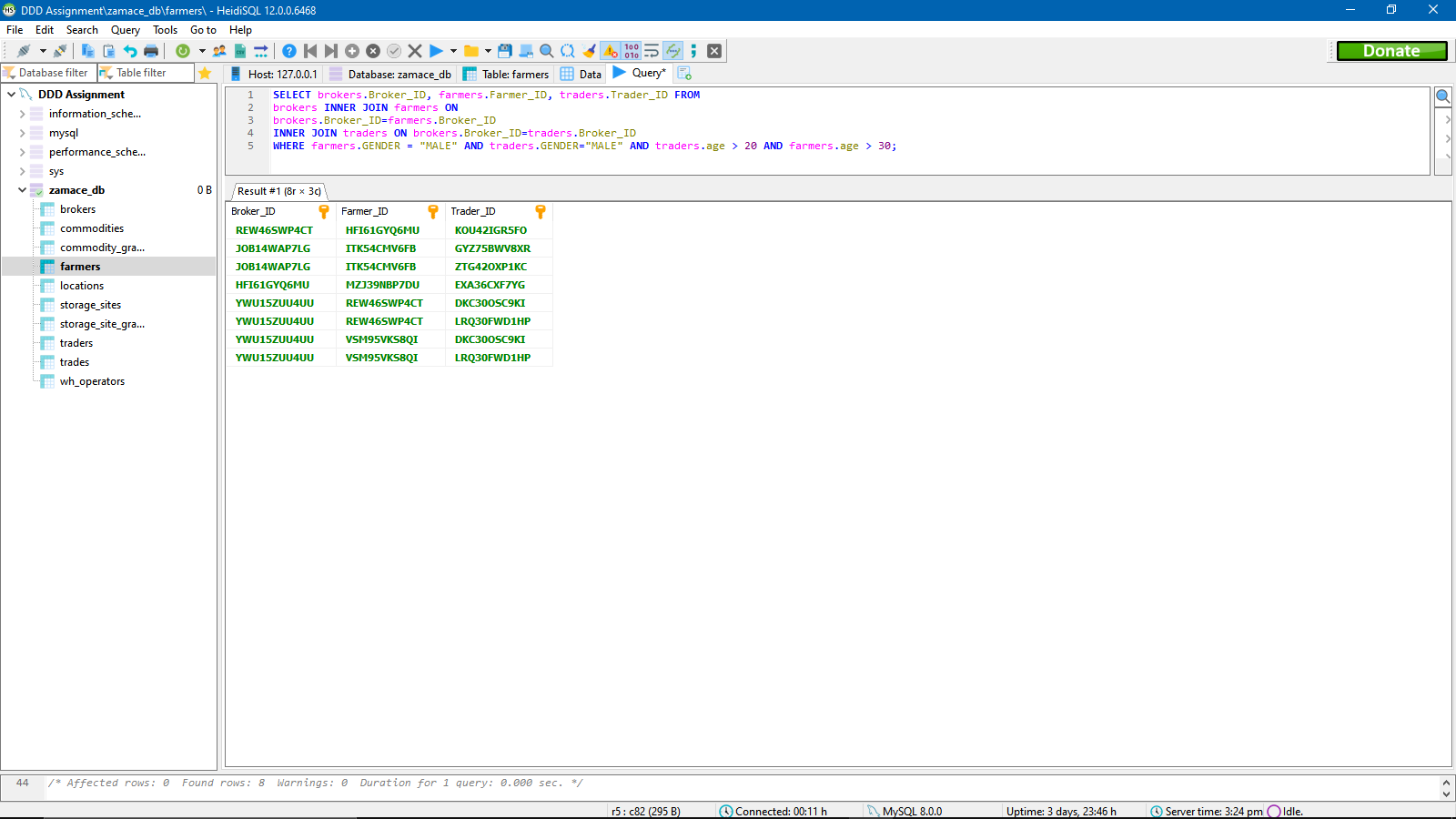
**Query 9:**

We want to list all storage sites in each location with the storage site grade which is above 2 and that location is dense.



**Query 10:**

We want to query only those brokers with their farmer clients and trader clients whose gender is male and above age 20.



**TASK (9) (Data Warehouse)**

A data warehouse, also known as an enterprise data warehouse, is a system used for data analysis and reporting in computing and is regarded as an essential part of business intelligence.

Factors:

We need a Data warehouse to keep historical records, even if the source transaction system does not. It will allow us to create and store metadata that helps data warehouse users understand data. Retrieving data from multiple data sources takes time;

therefore, in order to save time, we need to integrate the data from multiple sources and put it in a single location.

It prevents report users from misusing or corrupting transaction data; it can be used to store clean data that can be accessed by data analysts, businesses, team members, and data scientists; it will allow us to quickly run analysis on huge datasets; the complexity of queries will rise, and our users will require faster query processing; we will need a Data warehouse; users will be able to create their own reports without involving an IT specialist.

Common uses:

* Predicting customer churn based on sales data from the past ten years;
* dividing customers into distinct groups based on previous purchases to provide them with more detailed online content;
* and assisting businesses in the creation of demand and sales forecasts to determine which areas to focus on.

Our company will be able to develop efficient plans with the assistance of data warehouses if it is able to comprehend past data performances. It will provide our company with historical data, which we can use later to guide our business decisions. By utilizing a data warehouse, we will be able to precisely determine where our company currently stands and establish measurable branch marks that can drive long-term growth.

We will be able to check the most profitable commodities and warehouse operators in a certain amount of time in the past. Through this we can make useful business decisions to drive more profit to the company. We can find out the most profitable brokers to the company also and so on and so forth.

**TASK (10) (Reflective Commentary)**

**Analysis and Description (what came and so what)**

The assignment and course overall brought forward a variety of concepts about database design and development. This made me research and get a better understanding about how these concepts work. Concepts such as table constraints, propagation constraint’s, the importance of normalization etc.

It is important to know these things because they save a whole lot of time and avoid confusion in designing a database for a company. I learnt how to create table with constraints, how to do join queries and how to alter tables. I have also noticed that I have a challenge in normalizing sometimes. I also found it difficult to follow some the SQL syntax because my DBMS had a different version. I also learnt that different DBMS can have different syntax sometimes to get a certain thing done.

I have also noticed that creating a database report is a challenge for me.

**Future Plan (Now what)**

I will learn more about data warehousing and put it into practice with a small-scale company by explaining to them how it works and its benefits.

I will practice more SQL queries and learn more about Triggers and more advanced concepts about SQL in the future.

I have to spend more time understanding the different variations of DBMS that support SQL i.e., MySQL, PostgreSQL, SQLite and so on.

I will learn how to integrate a database system into a Realtime application for real life use cases. A real time application such as a website, a bespoke software etc.

Finally, I will learn how to create more storage and performance efficient entities and performance efficient queries.

**(REFERENCES)**

**Data warehousing and the factors that cause its need**

<https://aws.amazon.com/data-warehouse/#:~:text=A%20data%20warehouse%20is%20a,typically%20on%20a%20regular%20cadence>.

Accessed on: 15/10/2022

**Normalization and its importance**

<https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description#:~:text=Normalization%20is%20the%20process%20of,eliminating%20redundancy%20and%20inconsistent%20dependency>.

Accessed on: 15/10/2022