Group Name:

IDK

Group Members:

Tammy Tran, Phuc Huynh, Sesario Imanputra, Tom Blanchard

1. Introduction

Team IDK intends to create a retail database that stores environmentally friendly apparel. The database will allow clients who utilize it to identify eco-friendly clothing products that use clean energy and recycled materials and compare them to other apparels.

2. Application

The application is developed as an inventory for existing eco-friendly, and non-eco-friendly, clothing products. This allows retail platforms to oversee their existing inventory of such products; giving customers the ability to identify and compare all existing types of products. The functionalities of the application include:

- Product Information
- Manufacturer Practice
- Customer Review
- Material
- Shipping information
- Stock Inventory

With these functionalities, users will have access to queries such as:

- Percentage of Renewable Material in Product
- Customer Review of Product
- Retailer that supplies Product

3. Entities & Queries

The database will be defined by the following entities:

- Customers: Name, Age, Customer ID, Gender, Address
- Product: Type, Production Date, Inventory Number, Dimensions, price, fabric, chemicals
- Retailers: Location, Name,
- Manufacturer: Name, Manufacturer ID, Manufacturer Practice, Type,
- Location: State, City, St. Number, Apt Number.
- Vendors: name, address, number
- Dimensions: width, weight, length, weight
- Brand: ID, Name
- Sustainable Process Index: Area Consumption, Non-renewable Consumption, Emission in Air, Emission in Water, Emission in Soil

Although most of these entities are self-explanatory, the sustainable process index is a unique. The index is used to measure the carbon footprint of an individual's everyday activities. Since its metrics (Area Consumption, Non-renewable Consumption, Emission in Air, Emission in Water, and Emission in Soil) is generalized, it can also be used to measure the carbon footprint the lifecycle of an apparel, including the environmental cost of creating them and its decomposition.

<u>Iteration 3 Update:</u>

Originally, the environmental footprint of the manufacturers was planned to be included. However, we realized that the scope of this task is too large. Thus, we reviewed and decided to remain focus on the environmental footprint of products only.

With the database, you can retrieve relevant information using the following queries:

- Access the manufacturing process with respect to regulatory agencies
- Access a list of the product materials
- Access data on the types of chemicals applied to the product

4. Schedule *(Deadlines)*

1. **Team Setup** *(10/8/2019)*
    1. Team Contract
    2. Project Proposal
2. **Diagram** *(10/18/2019)*
    1. Entity Relationship Diagram
    2. Relational Schema Diagram
3. **Tools** (*11/08/2019)*
    1. Choose Web Hosting Service
    2. Programming Language: Start with SQL
4. **Development Process** *(11/25/2019)*
    1. Set up Web Server
    2. Write Code
    3. Populate Database
5. **Finalization & Presentation** *(12/09/2019)*
    1. Develop a Website
    2. Finalize Report
    3. Create Poster

5. Distribution of Work

While working on this project, we will equally contribute to the project but we will be setting leading roles for each individual person in the group. Phuc (Billy) Huynh will be the project manager and programmer. Sesario (Rio) Imanputra will be the lead researcher and programmer. Tom Blanchard will be the lead UI Designer & Data Researcher. Tammy Tran will be the lead web designer and developer.

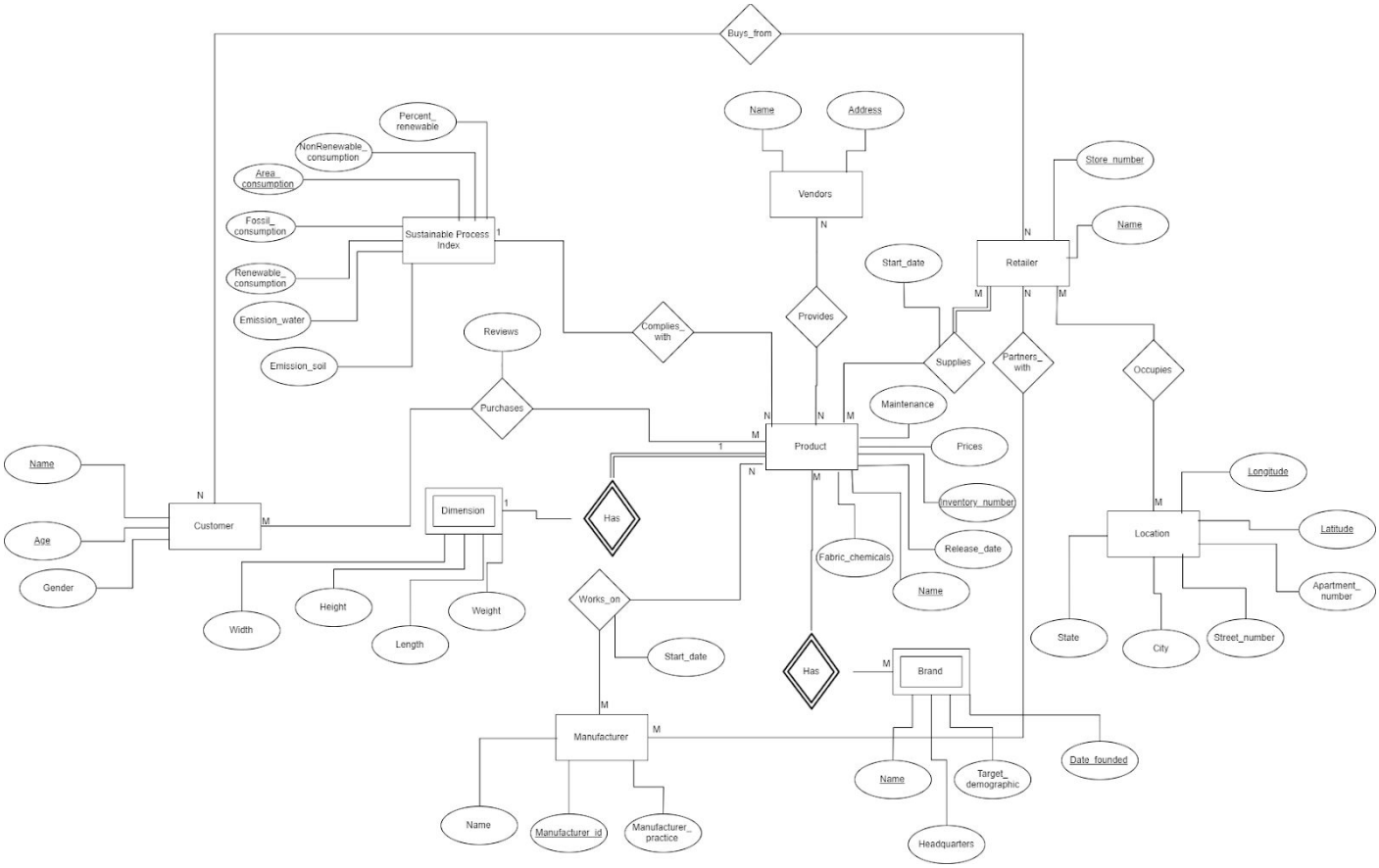
6. Design

6.1 Assumptions

- HQ is any location managing a brand
- Retailer can be either physical or online

6.2 ER Diagram

The following provides a detailed insight on all entities and relations of the database.

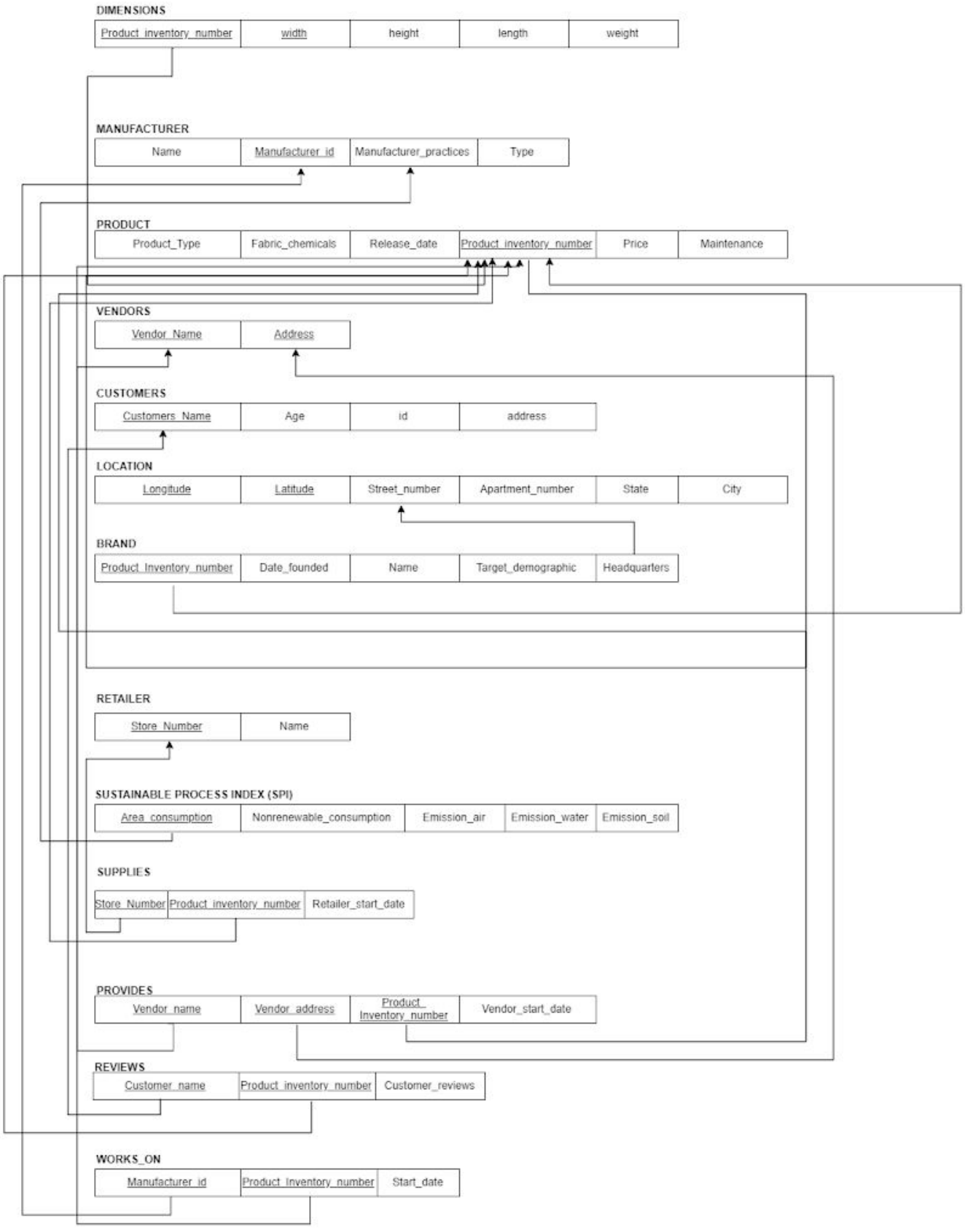


`

Note that above the entity “vendors”, the following attributes are name and address.. Although they can be difficult to see, they are all clearly mentioned in the RM Schema.

6.3 RM Schema

The following provides a detailed overview on the relations between all existing entities. This logic is to be used during the implementation stages. Thus, it is crucial to finalize all logic beforehand.


DIMENSIONS
Product_inventory_number
width
height
length
weight
MANUFACTURER
Name
Manufacturer_id
Manufacturer_practices
Type
PRODUCT
Product_Type
Fabric_chemicals
Release_date
Product_inventory_number
Price
Maintenance
VENDORS
Vendor_Name
Address
CUSTOMERS
Customers_Name
Age
id
address
LOCATION
Longitude
Latitude
Street_number
Apartment_number
State
City
BRAND
Product_Inventory_number
Date_founded
Name
Target_demographic
Headquarters
RETAILER
Store_Number
Name
SUSTAINABLE PROCESS INDEX (SPI)
Area_consumption
Nonrenewable_consumption
Emission_air
Emission_water
Emission_soil
SUPPLIES
Store_Number
Product_inventory_number
Retailer_start_date
PROVIDES
Vendor_name
Vendor_address
Product_Inventory_number
Vendor_start_date
REVIEWS
Customer_name
Product_inventory_number
Customer_reviews
WORKS_ON
Manufacturer_id
Product_Inventory_number
Start_date

As seen from the figure above, the relations are heavily based on the “product” entity. This is true, and expected, since the purpose of this database is product centric.

6.4 Domain Restrictions

The following section details restrictions for all domains. Note that ∈ means “belongs to” and “⊂” means subset to.

Did: all string with the maximum length of 10.
Dnumber: Integer (8)
Dname: all alphanumetic string with the maximum length of 50
Dcoutries: {Italy, America, Brazil….}
Ddate: all integer of length 10 formatted in dd/mm/yyyy
Dprice: all double (8)
Dage: Integer(3)
Dmeasure: all decimal(8,2)
Dcoordinate: {1:1, -111:1333,.....}
Dtext: all alphanumeric string with the maximum length of 200

WORKS_ON
Manufacturer_id ⊂ Dnumber, cannot be Null
Inventory_number ⊂ Dnumber, cannot be Null
Start_date: Ddate, Cannot be Null

REVIEW
Costumer_Name ⊂ Dname
Product_Inventory_Number ⊂ Dnumber
Customer_Reviews ⊂ Dtext, can be Null

PROVIDES
Vendor_Name ⊂Dname
Vendor_Address ∈ Location
Inventory_number ⊂ Dnumber
Vendor_start_date ⊂ Ddate

SUPPLIES
Store_Number ⊂ Dnumber
Product_Inventory_number ⊂ Dnumber
Retailer_start_date ⊂ Ddate

SUSTAINABLE PROCESS INDEX (SPI)
Area_consumption ⊂ Dtext
Non_renewable_Consumption ⊂ Dtext
Emission_in_air ⊂ Dtext

Emission_in_water ⊂ Dtext
Emission_in_soil ⊂ Dtext

RETAILER
Store_Number ⊂ Dnumber
Store_Name ⊂ Dname

BRAND
Product_inventory_number ⊂ Dnumber
Date_founded ⊂ Ddate
Brand_Name ⊂ Dname
Target_demographic ⊂ Dtext
Headquarters ⊂ Dname

LOCATION
Longitude ∈ Dcoordinate
Latidtude ∈ Dcoordinate
Country ∈ Dcountry

CUSTOMERS
Customers_Name ⊂ Dname
Age ⊂ Dage
Id ⊂ Did
Address ∈ Location {Dcoordinate, Dcity, Dcoutries, Dname, Dnumber)

VENDORS
Name ⊂ Dname
Address ∈ Location {Dcoordinate, Dcity, Dcoutries, Dname, Dnumber)

PRODUCT
Type ⊂ Dname
Fabric_chemicals ⊂ Dname
Release_date ⊂ Ddate
Product_Inventory_number ⊂ Dnumber
Price ⊂ Dprice
Maintenance ⊂ Dtext

MANUFACTURER
Name ⊂ Dname
Manufacturer_id ⊂ Did
Manufacturer_practices ⊂ Dtext
Type ⊂ Dname

DIMENSIONS

Product_Inventory_Number ⊂ Dnumber
Width ⊂ Dmeasure
Height ⊂ Dmeasure
Length ⊂ Dmeasure
Weight ⊂ Dmeasure

Iteration 3 Update

Note that the list presented below has been updated. This is mainly because an integer can't have a bigger size than 10. Thus, columns with this attribute, which are manufacturer and store ID, acquire an integer size of 8.Another update occurs for the domain constraints for cities, state and apartment number. Since creating a constraints for every possible combination of cities, states, and apartement is out of our scope, they are combined to street address. To verify any location, latitude and longitude is still used. To add an additional method of verification for location, countries are used since the scope of including all countries is within our scope:

- Did: all string with the maximum length of 8.

7. Tooling

The following details all the developing tools used to create the database:

- DBMS: MySQL
- DB Host: Microsoft Azure
- UI Tools: HTML, CSS, JavaScript
- Version Control: GIT

With the tools, the approach used is definitely web-based. This is approach is used as team members are better equipped to handle web development. Since there will be multiple developers, GIT will be used to oversee version control. The combination of MySQL and Microsoft Azure is used as Azure directly supports MySQL. For accessibility, there is also the benefit of using a trial version of Azure or creating a free student account. To generate random data, we will use Mockaroo.

Iteration 3 Updates

Currently, these are the tools that we use:

- DBMS: MySQL
- DB Host: AWS
- UI Tools: HTML, CSS, JavaScript
- Version Control: GIT

The use of azure has discontinued as it does not support the latest version of MySQL.. Thus, using CONSTRAINTS and CHECK cannot be used with Azure.

To connect to the database, use the following specification

- Connection Method: Standard (TCP/IP)
- Hostname: ecodb.cqtxjyiq283g.us-east-2.rds.amazonaws.com
- Username: TEAMIDK
- Password: BillyTheBomb123
- Set SSN USE as NO

8. Create Database

The following section details creation for all of our database entities including keys and constraints.

BRAND

```
CREATE TABLE `brand` (
`Product_inventory_number` int(8) NOT NULL,
`Date_founded` date DEFAULT NULL,
`Name` varchar(50) NOT NULL,
`Target_demographic` varchar(200) DEFAULT NULL,
`Location_headquarters` varchar(50) DEFAULT NULL,
PRIMARY KEY (`Product_inventory_number`),
UNIQUE KEY `Name` (`Name`),
KEY `Location_headquarters` (`Location_headquarters`),
CONSTRAINT `brand_ibfk_1` FOREIGN KEY (`Location_headquarters`) REFERENCES
`location` (`Street_address`),
CONSTRAINT `brand_ibfk_2` FOREIGN KEY (`Product_inventory_number`) REFERENCES
`product` (`Inventory_number`))
```

CUSTOMERS

```
CREATE TABLE `customers` (
`Name` varchar(50) NOT NULL,
`Id` varchar(15) NOT NULL,
`Age` int(3) NOT NULL,
`Street_address` varchar(50) NOT NULL,
PRIMARY KEY (`Name`),
UNIQUE KEY `Id` (`Id`),
UNIQUE KEY `Street_address` (`Street_address`))
```

DIMENSIONS

```
CREATE TABLE `dimensions` (
`Product_inventory_number` int(8) NOT NULL,
`Width` decimal(5,2) NOT NULL,
`Height` decimal(5,2) NOT NULL,
`Length` decimal(5,2) NOT NULL,
`Weight` decimal(5,2) NOT NULL,
PRIMARY KEY (`Product_inventory_number`,`Width`),
CONSTRAINT `dimensions_ibfk_1` FOREIGN KEY (`Product_inventory_number`)
REFERENCES `product` (`Inventory_number`))
```

LOCATION

```
CREATE TABLE `location` (
`Longitude` decimal(11,8) NOT NULL,
`Latitude` decimal(10,8) NOT NULL,
`Street_address` varchar(50) NOT NULL,
`Country` varchar(50) DEFAULT NULL,
PRIMARY KEY (`Longitude`,`Latitude`),
UNIQUE KEY `Longitude` (`Longitude`),
UNIQUE KEY `Latitude` (`Latitude`),
UNIQUE KEY `Street_address` (`Street_address`))
```

MANUFACTURER

```
CREATE TABLE `manufacturer` (
`Type` varchar(50) DEFAULT NULL,
`Name` varchar(50) NOT NULL,
`Manufacturer_practices` varchar(200) DEFAULT NULL,
`id` int(8) NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `id_UNIQUE` (`id`))
```

PRODUCT

```
CREATE TABLE `product` (
```

```
`Fabric_chemicals` varchar(50) DEFAULT NULL,
`Release_date` date NOT NULL,
`Price` decimal(8,2) NOT NULL,
`Maintenance` varchar(200) DEFAULT NULL,
`Inventory_number` int(8) NOT NULL,
`Product_Type` varchar(50) NOT NULL,
PRIMARY KEY (`Inventory_number`),
UNIQUE KEY `Product_inventory_number` (`Inventory_number`))
```

PROVIDES

```
CREATE TABLE `provides` (
`Vendor_name` varchar(50) DEFAULT NULL,
`Vendor_Street_address` varchar(50) DEFAULT NULL,
`Product_inventory_number` int(8) NOT NULL,
`Vendor_Start_date` date DEFAULT NULL,
PRIMARY KEY (`Product_inventory_number`),
KEY `Vendor_name` (`Vendor_name`),
KEY `Vendor_Street_address` (`Vendor_Street_address`),
CONSTRAINT `provides_ibfk_1` FOREIGN KEY (`Vendor_name`) REFERENCES `vendor`
(`Name`),
CONSTRAINT `provides_ibfk_2` FOREIGN KEY (`Vendor_Street_address`) REFERENCES
`vendor` (`Street_address`),
CONSTRAINT `provides_ibfk_3` FOREIGN KEY (`Product_inventory_number`) REFERENCES
`product` (`Inventory_number`))
```

RETAILER

```
CREATE TABLE `retailer` (
`Name` varchar(50) NOT NULL,
`Store_number` int(10) NOT NULL,
PRIMARY KEY (`Store_number`),
UNIQUE KEY `Store_number` (`Store_number`))
```

REVIEW

```
CREATE TABLE `review` (
`Review` varchar(200) DEFAULT NULL,
`Customer_name` varchar(50) NOT NULL,
`Product_inventory_number` int(8) NOT NULL,
PRIMARY KEY (`Product_inventory_number`,`Customer_name`),
KEY `Customer_name` (`Customer_name`),
```

```
CONSTRAINT `review_ibfk_1` FOREIGN KEY (`Customer_name`) REFERENCES `customers`
(`Name`),
CONSTRAINT `review_ibfk_2` FOREIGN KEY (`Product_inventory_number`) REFERENCES
`product` (`Inventory_number`))
```

SUPPLIES

```
CREATE TABLE `supplies` (
`Store_number` int(10) NOT NULL,
`Product_inventory_number` int(8) NOT NULL,
`Retailer_Start_date` date DEFAULT NULL,
PRIMARY KEY (`Product_inventory_number`,`Store_number`),
KEY `Store_number` (`Store_number`),
CONSTRAINT `supplies_ibfk_1` FOREIGN KEY (`Store_number`) REFERENCES `retailer`
(`Store_number`),
CONSTRAINT `supplies_ibfk_2` FOREIGN KEY (`Product_inventory_number`) REFERENCES
`product` (`Inventory_number`)
```

SUSTAINABLEPROCESSINDEX

```
CREATE TABLE `sustainableprocessindex` (
`Product_inventory_number` int(8) NOT NULL,
`Area_consumption`decimal(8,2) NOT NULL,
`Non_renewable_consumption` varchar(200) NOT NULL,
`Emission_in_air` decimal(8,2) NOT NULL,
`Emission_in_water` decimal(8,2) NOT NULL,
`Emission_in_soil` decimal(8,2) NOT NULL,
PRIMARY KEY (`Product_inventory_number`),
CONSTRAINT `sustainableprocessindex_ibfk_1` FOREIGN KEY (`Product_inventory_number`)
REFERENCES `product` (`Inventory_number`))
```

VENDOR

```
CREATE TABLE `vendor` (
`Name` varchar(50) NOT NULL,
`Street_address` varchar(50) NOT NULL,
PRIMARY KEY (`Name`),
UNIQUE KEY `Street_address` (`Street_address`))
```

WORKS_ON

```
CREATE TABLE `works_on` (
`Product_Release_date` date NOT NULL,
`Product_inventory_number` int(8) NOT NULL,
`Manufacturer_id` int(10) NOT NULL,
PRIMARY KEY (`Product_inventory_number`,`Manufacturer_id`),
KEY `Manufacturer_id` (`Manufacturer_id`),
CONSTRAINT `works_on_ibfk_1` FOREIGN KEY (`Manufacturer_id`) REFERENCES
`manufacturer` (`id`),
CONSTRAINT `works_on_ibfk_2` FOREIGN KEY (`Product_inventory_number`)
REFERENCES `product` (`Inventory_number`))9
```

9. Populate Database

To populate the database, data must be generated for the following tables:

- Brand
- Customers
- Dimensions
- Location
- Manufacturer
- Product
- Provides
- Retailer
- Review
- Supplies
- Sustainable process index
- Vendor
- Works_on

The tables that will be inserted first is location, manufacturer, product, retailer, and vendor. This order must be respected as inserting data from a weak entity or an entity with a foreign key first will cause referential integrity constraint violation.

9.1 Data insert samples

We imported our data into our database using excel sheets, but here are some examples of data insertion if we inserted our data manually:

- insert into customer values ('Allan Seary', '880-89-8454', 32,'075 Ryan Avenue');

- Insert into brand values (12621022, '2003-11-02', 'EchoStar Corporation','in tempor turpis nec euismod scelerisque quam turpis adipiscing lorem', '16747 Rowland Hill');
- Insert into dimensions values (12621022,367.06,47.92,60.74,92.67);
- Insert into location values(-81.7375526, 30.3121498, '16747 Rowland Hill', 'United States');
- Insert into manufacturer values('Books', 'Jetwire','Excision of muscle or fascia of hand for graft',10900000);
- Insert into product values('Yankee Oscar Zulu Golf Charlie Bravo Echo India Ki', '2019-4-23',224.38,'Etiam pretium iaculis justo. In hac habitasse platea dictumst.',12621022,'shirt');
- Insert into retailer values('Jabberbean', 11558693);

As of the current iteration, most of the data inserted is data to test the constraints of each attribute with a few meant for analysis. An example of the few is the NMD_R1 shoes. In our database, it is registered as product ID 93314351. It is important to node that product ID is relative to our database, not adidas. To ensure the following output respects the actual real life instance of the product please compare the output with the following website:
https://www.adidas.com/us/nmd_r1-shoes/B42200.html

Using the following query, we would obtain:

- Product:
  Insert into product values(' textile', '2019-12-20',130,'liquid detergent or dish soap with a brush',93314351,shoe)

- Brand
  Insert into brand values (93314351, '1946-08-18', 'Adidas','young adults, adults as well as children who have passion for fitness & sports. Although it targets customers in the age group of 13-40 years but majority of its customers are of 15-30 years of age who hail from upper middle class or the luxury class of customers', 'Herogenaurach Germany');

- Customers
  insert into customer values ('Billy Huynh', '773-23-2341', 18,'Plum Street');

- Location
  Insert into location values (49.5683, 10.8829, '91074 Herzogenaurach', 'Germany');

- Manufacturer

Insert into manufacturer values('Footwear', 'Dass Suarez S.a.','Sweatshop',89900000);

- Retailer
  Insert into retailer values('Champs Sports', 11552693);

- Vendor
  Insert into vendor values('Advanced Sporting Goods Co. Ltd',' Longyan Industrial Zone Humen Town Dongguan');

- Review
  Insert into review values('Great product! So comfortable to walk on'. 'Billy Huynh', 93314351);

- Works_on
  Insert into works_on values('2019-12-20',93314351,89900000);

- Dimensions
  Insert into Dimensions values(93314351, 5,3.2,10.5, 12);

- Sustainable process index
  Insert into sustainableprocessindex values ('93314351',19.5,8.5,25,5.2,22.59);

- Supplies
  Insert into supplies values(11552693,93314351,'2019-12-23');

- Provides
  Insert into provides values(' Advanced Sporting Goods Co. Ltd', 'Longyan Industrial Zone Humen Town Dongguan',93314351,'2019-08-23');

The result of the test on the product id 93314351

Inventory_number	Release_date	Price	Maintenance	Product_Type
93314351	2019-12-20	130.00	liquid detergent or dish soap with a brush	shoe

9.2 Data Generation

As the previous section has mentioned, all the data in the current iteration of the database is from Mockaroo. Mockaroo is a web application, which generates data to test an application. In this case, mockaroo generated data to test the constraints and integrity of our database. In terms inserting our data, we chose the option to import data through excel rather than using manual commands. This is done so that

inserting data won't take as much time as generating data. To import data to a database, an excel workbook must use .CSV format. Thankfully, Mockaroo does support the option to generate data in a .CSV format. Thus the whole process of inserting data can be described as simple as "drag and drop".

10. Normalization

This following section indicates which normal form each of our relations are in and shows how the normalized version of our database could improve our database design.

Brand

Product_inventory_number	Date_founded	Name	Target_demographic	Headquarters

Product_invenory_number → Name
Product_invenory_number → Target_demographic
1NF
It's in the 1NF since the row is uniquely identified and we have Name as a unique key to prevent value duplicates and Product_invenory_number as a primary key

2NF
Assumption: Each product line will target different demographic target and Name is uniquely identified since each brand has its own trademark brand name.

It's not in the second form since Date_founded and Headquarters doesn't depend on the primary key
Normalized version

Product_inventory_number	Target_demographic	Name (FK)

Product_invenory_number → Name
Product_invenory_number → Target_demographic

Name	Date_founded	Headquarters

Name → Date_founded
Name → Headquarters

Now It's in the 3NF since all columns can be determined by only the key in the table and doesn't contain any transitive dependencies

Customers

Customers_Name	Age	id	address

Customers_Name → Age
Customers_Name → id
Customers_Name → address

Assumption: Customers_name is a unique username that customers choose
It's in the NF since the row are uniquely identified and depends on the key (Customers_Name)

Dimensions

Product_Inventory_Number	width	height	length	weight

Product_Inventory_Number, width → height
Product_Inventory_Number, width → length
Product_Inventory_Number, width → weight

It's in the 3NF since the row are uniquely identified and all the attributes depends on the key (Product_Inventory_Number and width) and there is no transitive dependencies in the table.

Location

Longitude	Latitude	Street_number	Apartment_number	State	City

Longitude, Latitude → Street_number
Longitude, Latitude → Apartment_number
Longitude, Latitude → State
Longitude, Latitude → City

It's in the 3NF since the row are uniquely identified and all the attributes depends on the key (Longitude and Latitude) and there is no transitive dependencies in the table. Each of the coordination of longitude and latitude has it own unique address so it won't be duplicated

Manufacturer

Name	Manufacturer_id	Manufacturer_practices	Type

Manufacturer_id → Name
Manufacturer_id → Manufacturer_practices
Manufacturer_id → Type

It's in the 3NF since the row are uniquely identified and all the attributes depends on the key (Manufacturer_id) and there is no transitive dependencies in the table.

Product

Name	Fabric_chemicals	Release_date	Product_Inventory_number	Price	Maintenance

Product_Inventory_number → Name
Product_Inventory_number → Fabric_chemicals
Product_Inventory_number → Released_date
Product_Inventory_number → Price
Product_Inventory_number → Maintenance

It's in the 3NF since the row are uniquely identified and all the attributes depends on the key (Product_Inventory_number) and there is no transitive dependencies in the table.

Provides

Vendor_Name	Vendor_Address	Inventory_number	Vendor_start_date

Vendor_Name, Vendor_Address, Invemtory_number → Vendor_start_date

This table is used to communicate between Product and Vendor table and each column in this table is uniquely identified and there is no transitive dependencies in the table. It's in the 3NF

Retailer

Store_Number	Name

Store_Number → Name

It's in the 3NF since the row are uniquely identified and all the attributes depends on the key (Store_number) and there is no transitive dependencies in the table.

Review

Customer_Name	Product_Inventory_Number	Customer_Reviews

Since each customer has their own unique review about each product so the composited key for the review table is combined by all attributes in the table including Customer_name, Product_Inventory_Number, and Customer_Reviews. Since there is no non key attributes existing in the table so this relation is in the 3NF

Supplies

Store_Number	Product_Inventory_number	Retailer_start_date

Store_Number, Product_Inventory_number → Retailer_start_date

This table is used to communicate between Retailers and Product. Store_number and Product_Inventory_number is a composite key for the supplies table. Each column in the table is uniquely identified and the non key attribute (Retailer_start_date) is dependent on the key; plus, there is no transitive dependencies in table so this relation in 3NF

Sustainable process index

Area_consumption	Non_renewable_Consumption	Emission_in_air	Emission_in_water	Emission_in_soil

Area_comsumption → Non_renewable_Consumption
Area_comsumption → Emission_in_air
Area_comsumption → Emission_in_water
Area_comsumption → Emission_ in_soil

The row is uniquely identified where non of attribute is repeated. All of non key attributes are dependent on the key (Area_consumption). There is also no transitive dependencies. It's in 3NF

Vendor

Name	Address

The row is uniquely identified. The composite key of Name and Address is used as primary key since vendor names can be the same. Beside the primary keys there is no non key attributes existing in the table, so this relation is in 3NF.

Works_on

Manufacturer_id	Inventory_number	Start_date

Manfacturer_id, Inventory_number → Start_date

This table is used to communicate between Manufacturer and Product. The non key attribute (Start_date) is dependent on the key (Manfacturer_id and Inventory_number). It's in the 3NF

11. Queries

The following examples detail the types of queries that will be implemented in the clothing database:

1. Find a specific product's emissions into the soil.
2. What product has the lowest sustainable process index from a specific manufacturer.
3. What product has the lowest price.
4. What is the specific vendor start date for a specific product.
5. Access the fabric chemicals of a product
6. Access the location of a specific manufacturer.
7. What is the most affordable and quality eco friendly products released this year?

Query Implementation

1.

```
Select product_name, Emission_in_soil
From ecodb.product, ecodb.sustainableprocessindex
Where ecodb.product.inventory_number =
ecodb.sustainableprocessindex.Product_inventory_number
AND ecodb.product.Product_Name = 'Nikes Air 2770';
```

2.

```
Select Product_name, Inventory_number, Manufacturer_id, Emission_in_air
FROM ecodb.product, ecodb.manufacturer, ecodb.sustainableprocessindex,
ecodb.works_on
WHERE works_on.Manufacturer_id = manufacturer.id
AND sustainableprocessindex.Product_inventory_number =
product.inventory_number
AND works_on.Product_inventory_number = product.Inventory_number
AND Emission_in_air IN(
Select MIN(Emission_in_air)
FROM ecodb.sustainableprocessindex);
```

3.

```
Select Product_name, Inventory_number, Price
From ecodb.product
Where price = ( Select Min(Price)
From ecodb.product);
```

4.

```
Select Inventory_number, Fabric_chemicals
From ecodb.product
Where Inventory_number = 93314351;
```

5.

```
Select Inventory_number, Vendor_start_date
From ecodb.product JOIN ecodb.provides
Where Product_inventory_number = Inventory_number;
```

6.

```
Select Manufacturer_id, manufacturer.Name, Location_headquarters
FROM ecodb.product, ecodb.manufacturer, ecodb.location, ecodb.brand, works_on
WHERE ecodb.brand.Product_inventory_number =
ecodb.product.Inventory_number
AND ecodb.works_on.Product_inventory_number =
ecodb.product.inventory_number
```

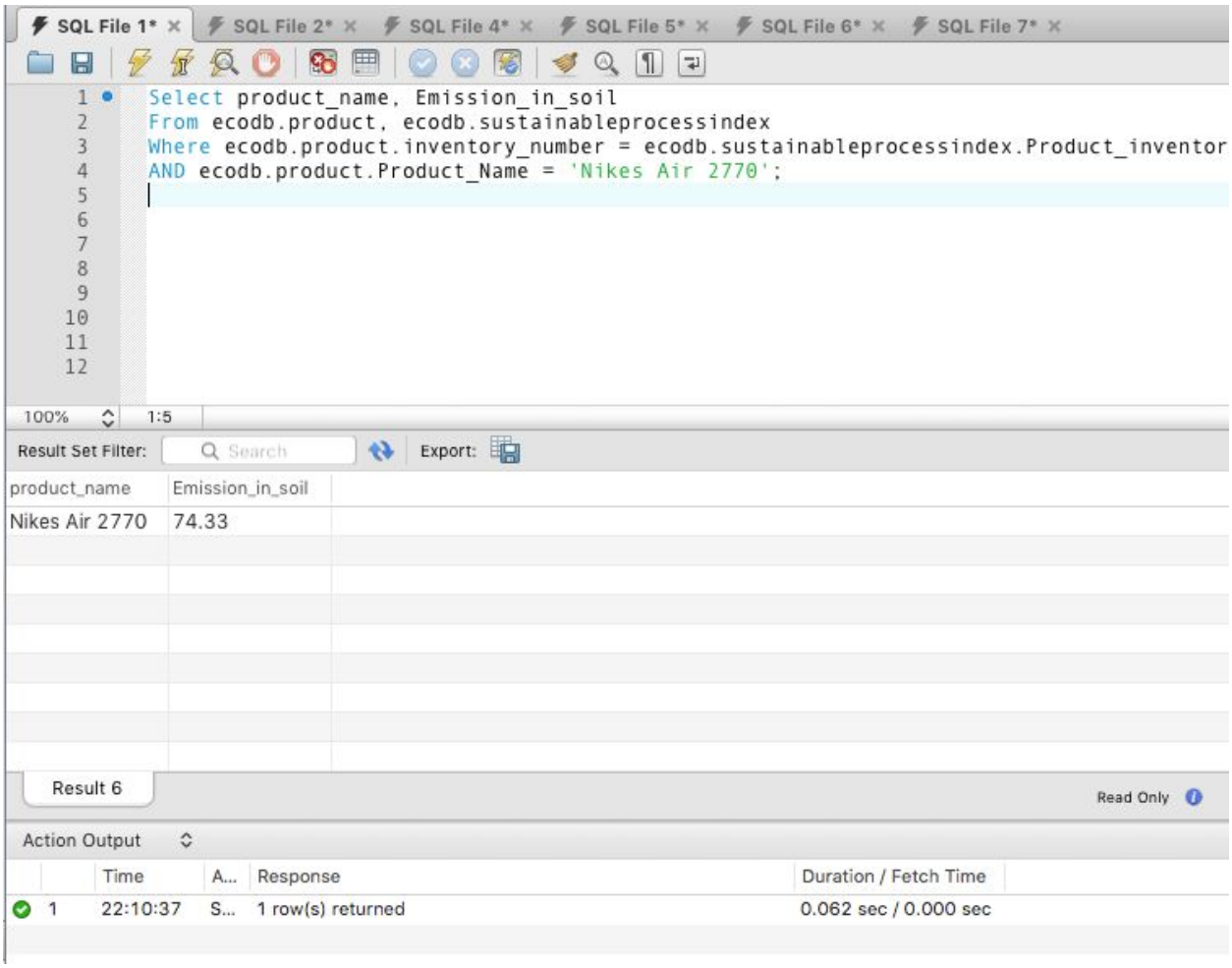
```
AND ecodb.works_on.Product_inventory_number =
ecodb.brand.Product_inventory_number
AND manufacturer.Name = 'Nlounge'
GROUP BY manufacturer.Name;
```

7.

```
SELECT
review.Product_inventory_number,Customer_name,Review,Product_Release_date
FROM ecodb.review
INNER JOIN ecodb.sustainableprocessindex ON
sustainableprocessindex.Product_inventory_number =
review.Product_inventory_number AND Area_consumption > 20
INNER JOIN ecodb.product ON product.Inventory_number =
review.Product_inventory_number AND Price > 50
INNER JOIN ecodb.works_on ON works_on.Product_inventory_number =
review.Product_inventory_number AND Product_Release_date > "2018-01-01" AND
Product_Release_date < "2019-01-01"
GROUP BY Product_Release_date;
```

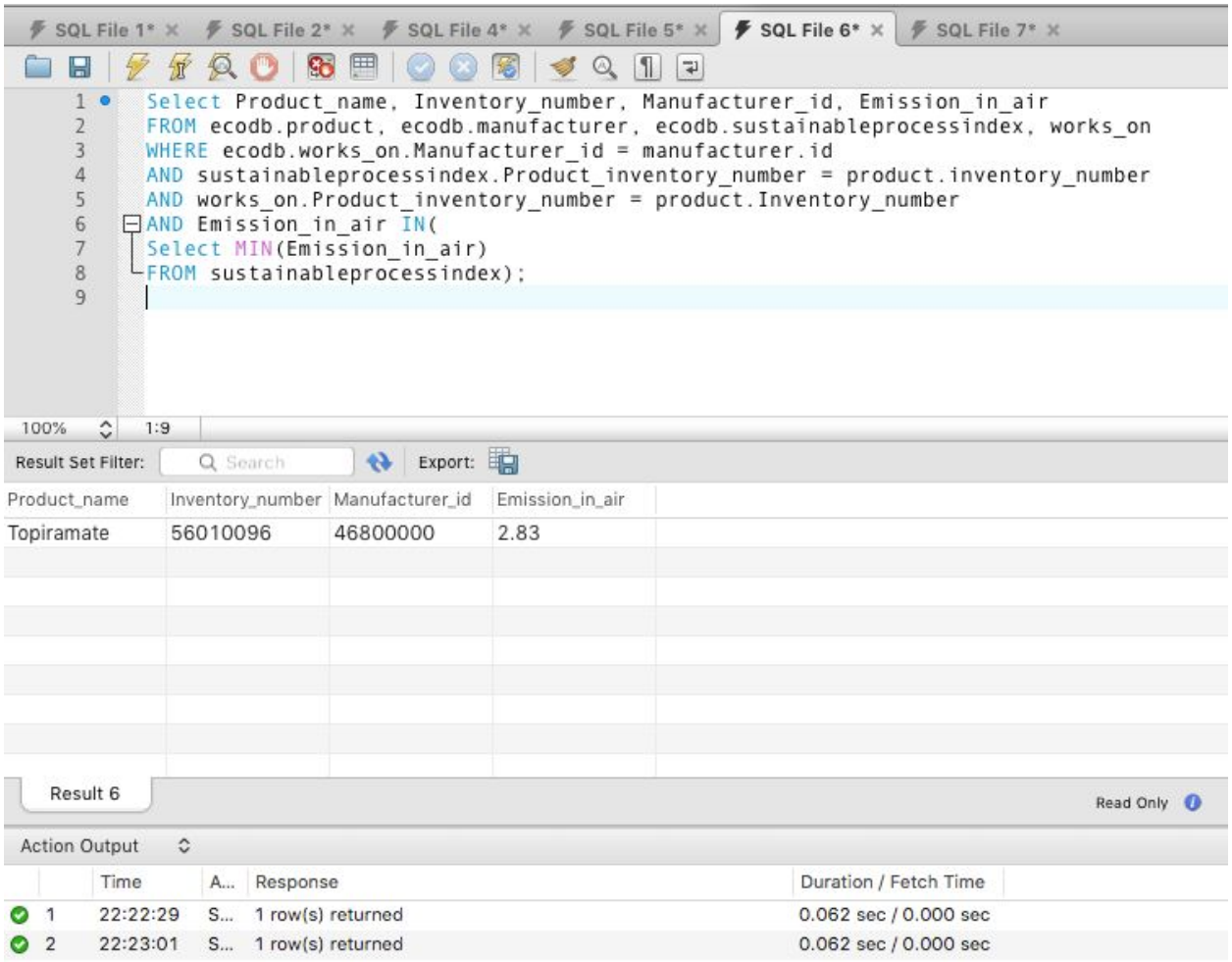
12. Testing

Test 1.

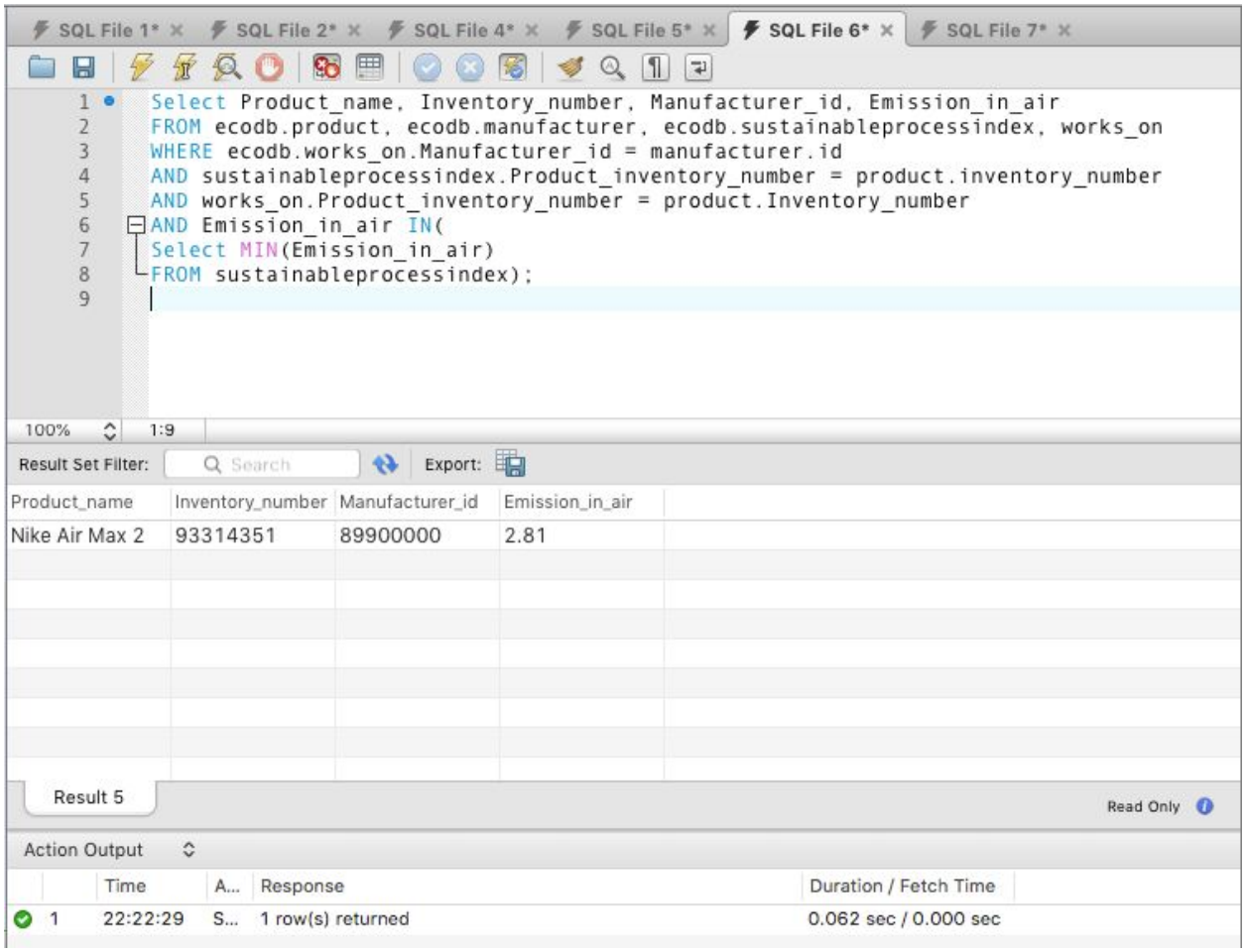


Test 2.
For this test we entered the query as is to retrieve the product with the lowest emission in air, and then we inserted a new row with a lower emission in air to verify our results.

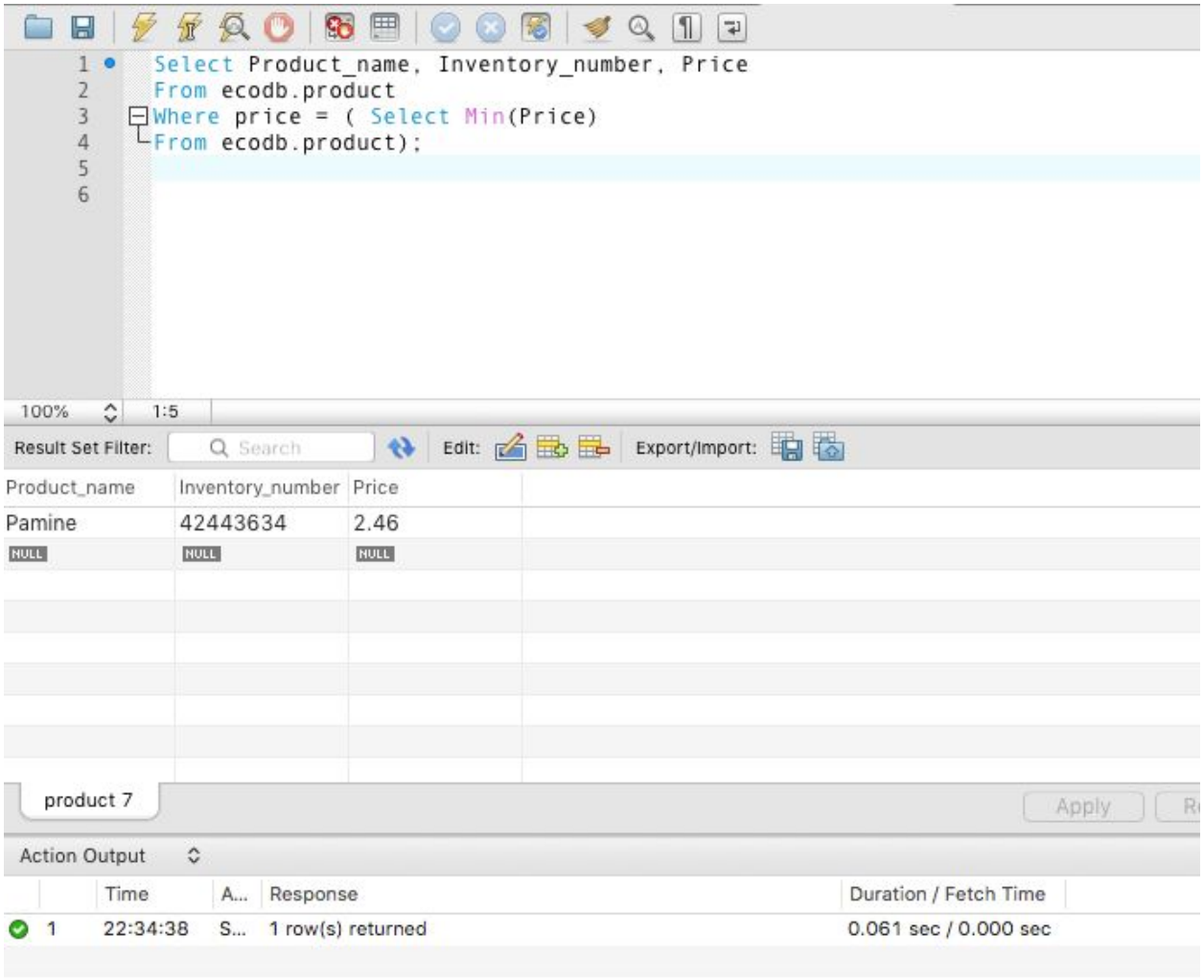
Before:


SQL File 1*
SQL File 2*
SQL File 4*
SQL File 5*
SQL File 6*
SQL File 7*
Select Product_name, Inventory_number, Manufacturer_id, Emission_in_air
FROM ecodb.product, ecodb.manufacturer, ecodb.sustainableprocessindex, works_on
WHERE ecodb.works_on.Manufacturer_id = manufacturer.id
AND sustainableprocessindex.Product_inventory_number = product.inventory_number
AND works_on.Product_inventory_number = product.Inventory_number
AND Emission_in_air IN(
Select MIN(Emission_in_air)
FROM sustainableprocessindex);
100%
1:9
Result Set Filter:
Search
Export:
Product_name
Inventory_number
Manufacturer_id
Emission_in_air
Topiramate
56010096
46800000
2.83
Result 6
Read Only
Action Output
Time
A...
Response
Duration / Fetch Time
1
22:22:29
S...
1 row(s) returned
0.062 sec / 0.000 sec
2
22:23:01
S...
1 row(s) returned
0.062 sec / 0.000 sec

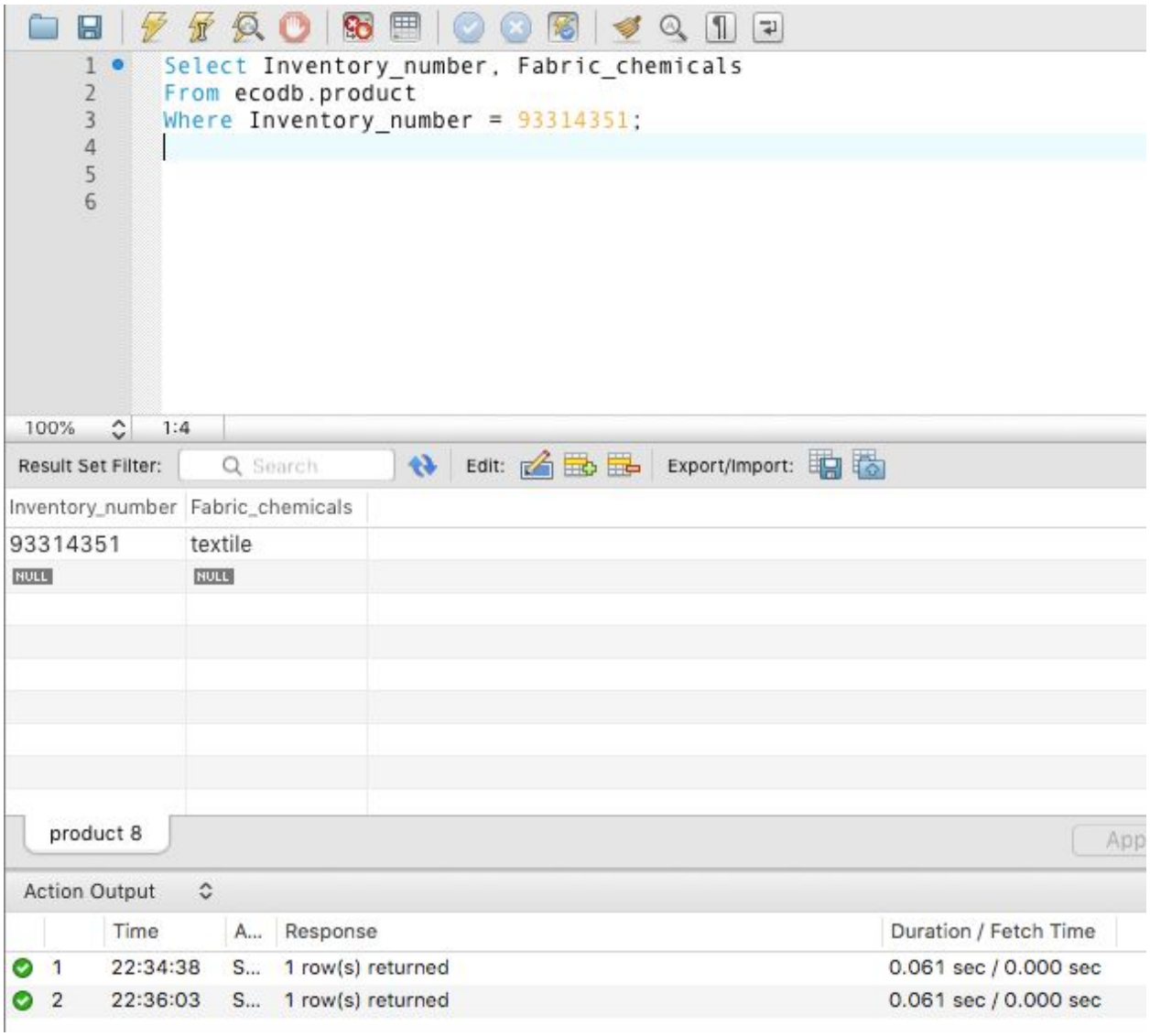
After:


SQL File 1*
SQL File 2*
SQL File 4*
SQL File 5*
SQL File 6*
SQL File 7*
1 Select Product_name, Inventory_number, Manufacturer_id, Emission_in_air
2 FROM ecodb.product, ecodb.manufacturer, ecodb.sustainableprocessindex, works_on
3 WHERE ecodb.works_on.Manufacturer_id = manufacturer.id
4 AND sustainableprocessindex.Product_inventory_number = product.inventory_number
5 AND works_on.Product_inventory_number = product.Inventory_number
6 AND Emission_in_air IN(
7 Select MIN(Emission_in_air)
8 FROM sustainableprocessindex);
9
100%
1:9
Result Set Filter:
Search
Export:
Product_name
Inventory_number
Manufacturer_id
Emission_in_air
Nike Air Max 2
93314351
89900000
2.81
Result 5
Read Only
Action Output
Time
A...
Response
Duration / Fetch Time
1
22:22:29
S...
1 row(s) returned
0.062 sec / 0.000 sec

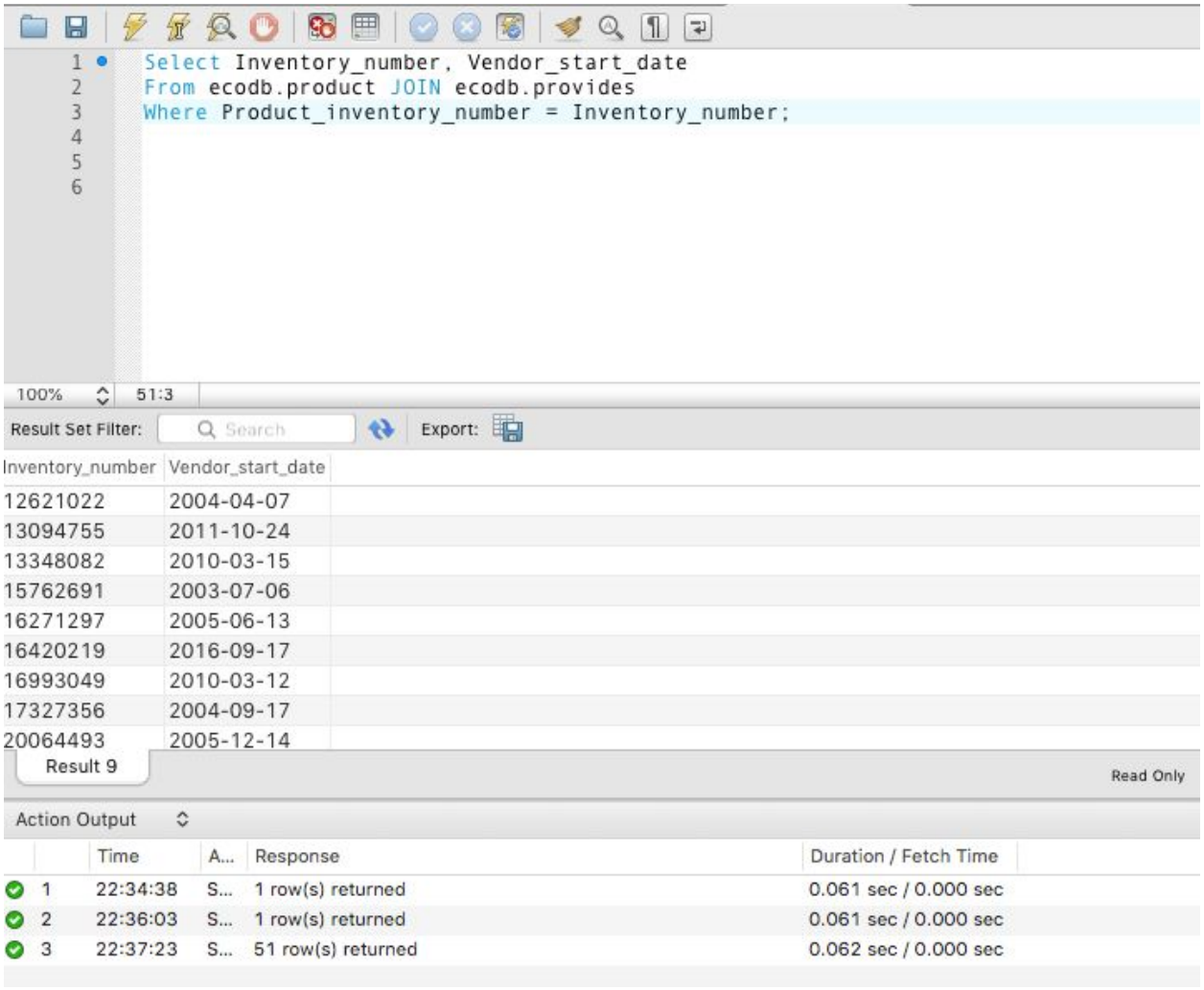
Test 3.


Select Product_name, Inventory_number, Price
From ecodb.product
Where price = (Select Min(Price)
From ecodb.product);
100%
1:5
Result Set Filter:
Search
Edit:
Export/Import:
Product_name
Inventory_number
Price
Pamine
42443634
2.46
NULL
NULL
NULL
product 7
Apply
Action Output
Time
A...
Response
Duration / Fetch Time
1
22:34:38
S...
1 row(s) returned
0.061 sec / 0.000 sec

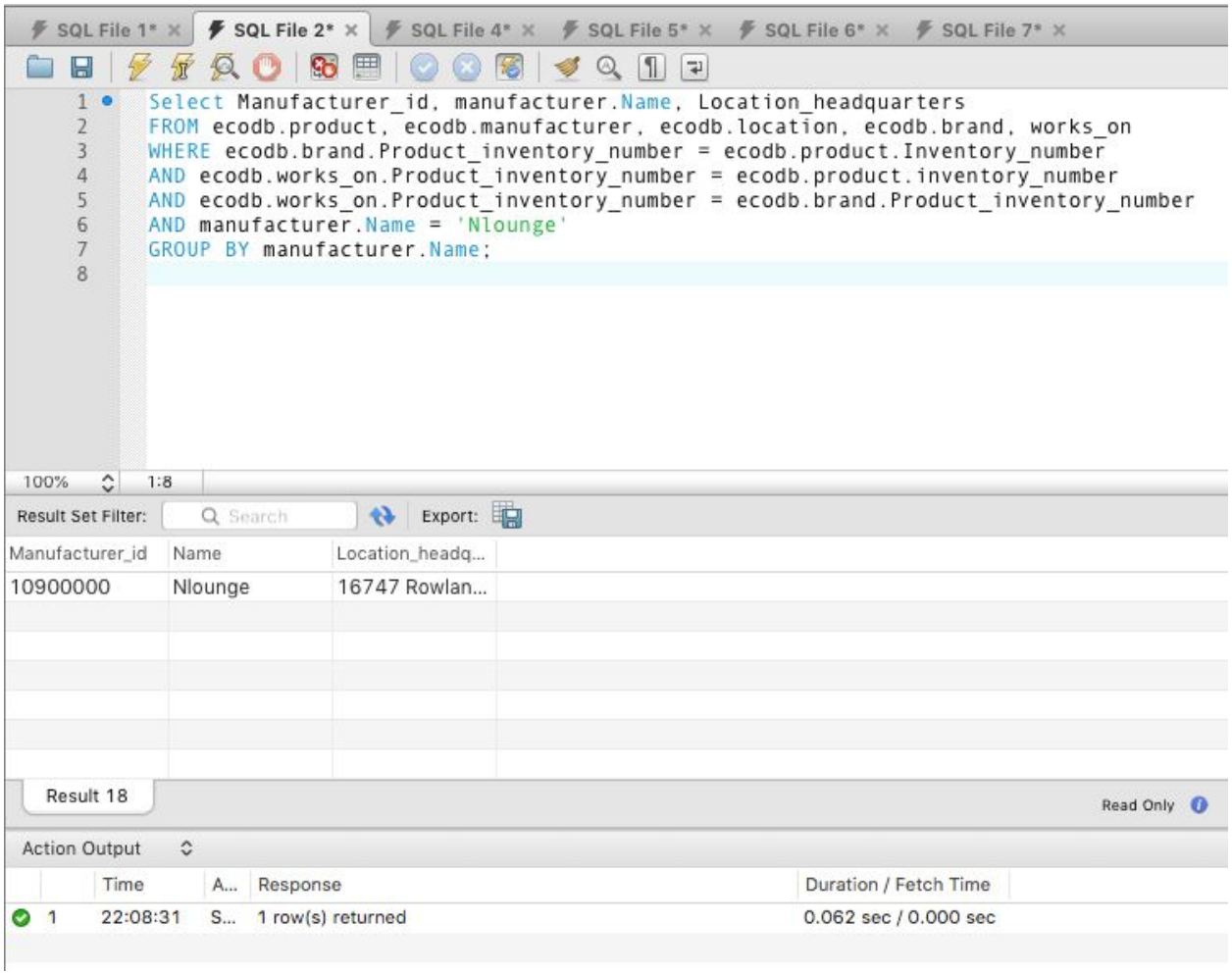
Test 4.


Select Inventory_number, Fabric_chemicals
From ecodb.product
Where Inventory_number = 93314351;
100%
1:4
Result Set Filter:
Search
Edit:
Export/Import:
Inventory_number
Fabric_chemicals
93314351
textile
NULL
NULL
product 8
Action Output
Time
A...
Response
Duration / Fetch Time
1
22:34:38
S...
1 row(s) returned
0.061 sec / 0.000 sec
2
22:36:03
S...
1 row(s) returned
0.061 sec / 0.000 sec

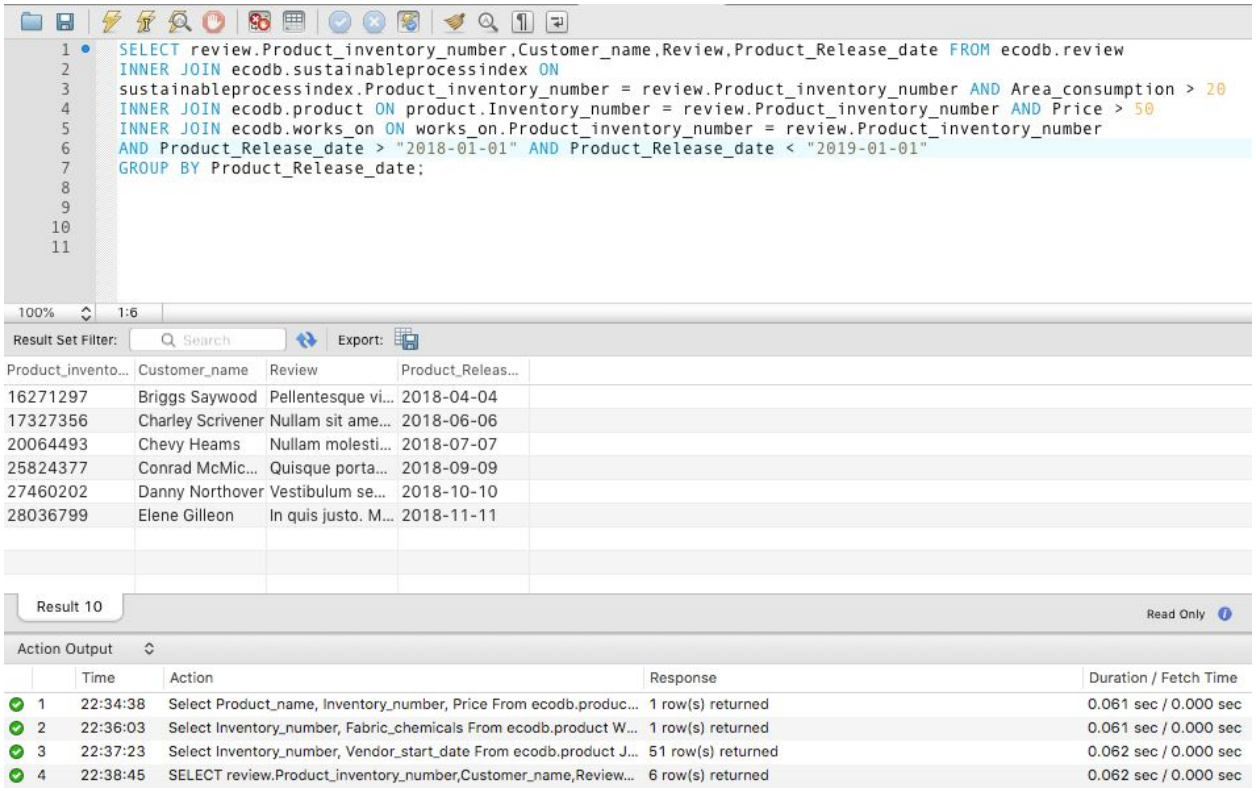
Test 5.


Select Inventory_number, Vendor_start_date
From ecodb.product JOIN ecodb.provides
Where Product_inventory_number = Inventory_number;
100%
51:3
Result Set Filter:
Search
Export:
Inventory_number
Vendor_start_date
12621022 2004-04-07
13094755 2011-10-24
13348082 2010-03-15
15762691 2003-07-06
16271297 2005-06-13
16420219 2016-09-17
16993049 2010-03-12
17327356 2004-09-17
20064493 2005-12-14
Result 9
Read Only
Action Output
Time
A...
Response
Duration / Fetch Time
1 22:34:38 S... 1 row(s) returned 0.061 sec / 0.000 sec
2 22:36:03 S... 1 row(s) returned 0.061 sec / 0.000 sec
3 22:37:23 S... 51 row(s) returned 0.062 sec / 0.000 sec

Test 6.


SQL File 1*
SQL File 2*
SQL File 4*
SQL File 5*
SQL File 6*
SQL File 7*
1 Select Manufacturer_id, manufacturer.Name, Location_headquarters
2 FROM ecodb.product, ecodb.manufacturer, ecodb.location, ecodb.brand, works_on
3 WHERE ecodb.brand.Product_inventory_number = ecodb.product.Inventory_number
4 AND ecodb.works_on.Product_inventory_number = ecodb.product.inventory_number
5 AND ecodb.works_on.Product_inventory_number = ecodb.brand.Product_inventory_number
6 AND manufacturer.Name = 'Nlounge'
7 GROUP BY manufacturer.Name;
8
100%
1:8
Result Set Filter:
Search
Export:
Manufacturer_id
Name
Location_headq...
10900000
Nlounge
16747 Rowlan...
Result 18
Read Only
Action Output
Time
A...
Response
Duration / Fetch Time
1
22:08:31
S...
1 row(s) returned
0.062 sec / 0.000 sec

Test 7.



```
SELECT review.Product_inventory_number,Customer_name,Review,Product_Release_date FROM ecodb.review
INNER JOIN ecodb.sustainableprocessindex ON
sustainableprocessindex.Product_inventory_number = review.Product_inventory_number AND Area_consumption > 20
INNER JOIN ecodb.product ON product.Inventory_number = review.Product_inventory_number AND Price > 50
INNER JOIN ecodb.works_on ON works_on.Product_inventory_number = review.Product_inventory_number
AND Product_Release_date > "2018-01-01" AND Product_Release_date < "2019-01-01"
GROUP BY Product_Release_date;
```

100% 1:6

Result Set Filter: Search Export:

Product_invento...	Customer_name	Review	Product_Releas...
16271297	Briggs Saywood	Pellentesque vi...	2018-04-04
17327356	Charley Scrivener	Nullam sit ame...	2018-06-06
20064493	Chevy Heams	Nullam molesti...	2018-07-07
25824377	Conrad McMic...	Quisque porta...	2018-09-09
27460202	Danny Northover	Vestibulum se...	2018-10-10
28036799	Elene Gilleon	In quis justo. M...	2018-11-11

Result 10 Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
1	22:34:38	Select Product_name, Inventory_number, Price From ecodb.produc...	1 row(s) returned	0.061 sec / 0.000 sec
2	22:36:03	Select Inventory_number, Fabric_chemicals From ecodb.product W...	1 row(s) returned	0.061 sec / 0.000 sec
3	22:37:23	Select Inventory_number, Vendor_start_date From ecodb.product J...	51 row(s) returned	0.062 sec / 0.000 sec
4	22:38:45	SELECT review.Product_inventory_number,Customer_name,Review...	6 row(s) returned	0.062 sec / 0.000 sec

13. Discussion

Evaluation of Project

Overall, we put a lot of effort into the design and implementation process. Through each iteration, we revised and improved our design, i.e. from iteration 1 to iteration 2, we focused on adding relations to make our database more robust and/or fixing relations that were weak or had other issues. The things that went right: our team dynamic was conducive to thoughtful and productive work, we were able to produce our deliverable on time, we implemented feedback to the best of our ability.

If we had more time, we would certainly focus more on polishing our tables to make sure each and every attribute is needed, simplify dependencies between relations to improve testing, and possibly expand the functionality of our database to products other than just clothing. We would also focus more on rigorously normalizing our tables to strive for a more elegant and simplistic interaction. With more time we would also focus more on the user interface.

Iteration Improvement

Throughout the iteration process, we were able to make improvements by adding necessary information and taking feedback into consideration and changing what needed to be changed.

Database Population
We used Mockaroo to generate dummy data to populate our tables.

Testing
We were able to retrieve data from different tables from our eco database using queries.

UI
The current status of the UI is that its functional. The remaining issues are that the data displayed MySQL are not designed. Thus, they appear scattered. However, all the data displayed are correct. Use the following link to connect: http://ecodbinstance.3zm8mec7mp.us-east-2.elasticbeanstalk.com/

If the following link does not work, a local instance of the UI is attached to this document. To run it, XAMPP is required.

14. References

AWS Elastic Beanstalk Documentation. AWS, https://docs.aws.amazon.com/elastic-beanstalk/index.html. Accessed 22 November 2019.

Mockaroo, https://mockaroo.com/ Accessed 7 October 2019.

Normalization in DBMS: 1NF, 2NF, 3NF and BCNF with Examples. Hackr.io, https://hackr.io/blog/dbms-normalization. Accessed 6 December 2019.