**Designing the Storefront Database: Nick Squad Team 1**

| **Class:** | 3432 Database Systems |
| --- | --- |
| **Professor:** | Dr. Jarman |
| **Term / Year:** | Spring 2022 |
| **Effort [hours]** | Estimated 50 hours |
| **Student Name (s)** | Nick Ferrara  Nick Hackett  Nick Istre  Logan Salem  Jonah Walker |
| **Assignment:** | Database Project: Storefront Database |
| **Date of Submission:** | 05/03/2022 |

**Your satisfaction with learning-experience with this assignment**

1. Very Useful
2. Somewhat Useful
3. Useless

**Table of Contents**

[**Team Members**](#_heading=h.gjdgxs) 4

[**Scope**](#_heading=h.30j0zll) 5

[**Functional Requirements**](#_heading=h.1fob9te) 6

[**Style Guide**](#_heading=h.3znysh7) 7

[**10 Questions**](#_heading=h.2et92p0) 8

Nick F [[Authors, Discounts, Stores]](#_heading=h.tyjcwt) 8

Nick H [[Jobs, Employees]](#_heading=h.3dy6vkm) 8

Nick I [[Sales, Titles]](#_heading=h.1t3h5sf) 8

Jonah W [Customer, OrderHist, Cart]9

Logan S [[](#_heading=h.4d34og8)[Reviews, Products](#_heading=h.lnxbz9)[]](#_heading=h.4d34og8) 9

[**10 SQL Queries**](#_heading=h.2s8eyo1) 10

Nick F [[Authors, Discounts, Stores]](#_heading=h.17dp8vu) 10

Nick H [[Jobs, Employees] 1](#_heading=h.3rdcrjn)6

Nick I [[Sales, Titles] 2](#_heading=h.26in1rg)1

Jonah W [Customer, OrderHist, Cart] [2](#_heading=h.lnxbz9)7

Logan S [[Reviews, Products]](#_heading=h.lnxbz9) ??

[**Database Application Scope** 3](#_heading=h.35nkun2)2

**Storefront Initial** [**- Physical Model**](#_heading=h.z337ya) 38

**Storefront Redesign** [**- Physical Model**](#_heading=h.z337ya) 39

[**Tables and Data Types**](#_heading=h.2xcytpi) 40

CUSTOMER [Table](#_heading=h.1ci93xb) 40

CATEGORIES [Table](#_heading=h.3whwml4) 40

SUPPLIERS [Table](#_heading=h.2bn6wsx) 40

WAREHOUSES [Table](#_heading=h.qsh70q) 40

PRODUCTS [Table](#_heading=h.3as4poj) 40

ITEMS [Table](#_heading=h.1pxezwc) 40

ORDERHIST [Table](#_heading=h.49x2ik5) 40

CART [Table](#_heading=h.2p2csry) 40

SALES [Table](#_heading=h.147n2zr) 41

REVIEWS [Table](#_heading=h.3o7alnk) 41

[**Creating The Tables**](#_heading=h.41mghml) 42

[**Referential Integrity and Constraints**](#_heading=h.2grqrue) 44

[**Future Opportunities**](#_heading=h.vx1227) 45

[**Post Mortem**](#_heading=h.3fwokq0) 46

Nick F 46

Nick H 46

Nick I46

Logan S 46

Jonah W 47

# Team Members

**Nick F** - **Categories, Suppliers, Warehouses, Products**

**Nick H** - **Customers, Reviews**

**Nick I** - **Sales, Items**

**Logan S** - **Reviews, Products**

**Jonah W** - **Customer, OrderHist, Car**

# Scope

The purpose in creating this project was to make a functional database for an online tech storefront. By including features such as a shopping cart and a list of inventory users can see what items are in stock and place an order on an item. The items themselves were divided into different categories based on what type of electronic or piece of hardware they were. We intended to create an intuitive user experience that could easily modify aspects of the database as needed, such as adding new categories for new types of items being supplied and sold.

We had the items split into different categories, but they were also divided by which supplier provided them, and which warehouse they were being stored in. These two aspects made it easier to keep track of where the inventory was being sold, and in the future could be used to calculate how long the shipping time on an order would take. The items were also kept track of with individual SKUs and serial numbers, which helped to ensure that no inventory was lost, and everything was accounted for and available for sale to the customers.

Our workload was evenly split between the members of the group for each section, with each member accounting for a small portion of each part of the database. Had we decided to create a GUI for this database we likely would have implemented Java and had the work divided between the project members, similarly to how we had the rest of the project designed.

# Functional Requirements

**Availability**

* As it currently stands our project is running off of a local server, without a large network connection.
* If we were to make use of a server in the implementation of this project we would likely utilize AWS, as they have proven to be fairly reliable, and would likely be the easiest option for the smaller scope of this database.

**Security**

* The two primary security concerns of this project were the user’s information and keeping track of inventory in the storefront.
* By requiring a password for the user the database can help to ensure that only the user has access to their account and shopping cart
* By utilizing SKUs and serial numbers, the database keeps track of the items in its inventory and ensures that there is not lost or missing stock, and also makes sure that the user can only place orders on items that are available

**Reliability**

* Due to the smaller scope of this project, reliability was not a major concern in our initial design. However, we still wanted to make sure that certain aspects of the database only functioned as intended, with little room for user error.
* One major implementation of this was us creating a VerifiedPurchase value under the reviews table. This would ensure that users can see who is confirmed by the website to have purchased a product, and what their review on the product is.
* The customer and the admins both have access to the user’s order history, which keeps track of purchases and can be used to ensure that a customer received the correct product on time.

# Style Guide

Throughout the projects, our team tried to adhere to a standard with redesigning the database to keep names and conventions clean and consistent. The general idea is to avoid any confusion caused by confusing or misleading attribute names.

**General Conventions**

* + Uppercase for table names
  + Lowercase for column names
  + Underscores in place of whitespace
  + Use singular form for table names

**Avoid Using**

* + Abbreviations
  + Plurals
  + Camelcase
  + Numericals

# 10 Questions

Each team member was responsible for coming up with 10 questions in relation to their database that will assist in forming queries. These queries will then be incorporated into the GUI.

## Nick F [Categories, Suppliers, Warehouses, Products]

1. List all, if any warehouses that share a city with a supplier
2. Which warehouse has the most stock?
3. Are there any warehouses shared between suppliers?
4. Which warehouse has the highest value stockpile?
5. Which warehouse has the lowest value stockpile?
6. List all warehouse addresses that have storage devices in them.
7. List all devices stored at Macon, GA
8. Which supplier is located in Savannah, if any?
9. Which warehouse are the ATX motherboards stored in, and which supplier provides them?
10. Which suppliers provide CPUs and motherboards

## 

## Nick H [Customers, Reviews]

1. What are all the reviews from a specific customer?
2. Which customers tend to rate items more highly?
3. Which reviews were from someone who has a verified purchase?
4. Which reviews were not from someone who has a verified purchase?
5. Which SKUs have the highest average rating?
6. Which SKUs have the lowest average rating?
7. Which SKUs are very polarizing, i.e. have many very high and very low ratings?
8. Did any SKUs have a lot of reviews posted on the same date?
9. Are multiple customers posting reviews from the same address?
10. Are any customers posting many ratings at a time?

## Nick I [Sales, Items]

1. What products are on sale?
2. Which product has the best discount?
3. What is the cheapest item in the store after discounts?
4. What is the most expensive item after discounts?
5. How many discounted items have been purchased?
6. Which suppliers have discounted items?
7. Which discounted items are in nistre00’s cart?
8. List all serial numbers for a particular SKU (12345678)
9. What discounted items are at Warehouse1?
10. What are the serial numbers for all discounted items?

## Jonah W [Customer, OrderHist, Cart]

1. What is in Logan Salem’s cart that isn’t in Nick Hackett’s?
2. What is the subtotal of Nick Ferrara’s cart?
3. What has Jonah Walker ordered?
4. What is the most expensive item in Jonah Walker’s cart?
5. How many processors has Nick Istre purchased?
6. Sort the customers by the number of items purchased (from most to least).
7. List the customers with items in their cart.
8. List the customers who have ordered something.
9. How many products have been sold?
10. How many storage devices have been ordered?

**Logan Salem [Reviews, Products]**

1. Which products have a positive review
2. which products have a negative review
3. How many reviews does a specific product have
4. What date was the review left
5. Is a product with a positive review on sale
6. Where is a product with a positive review located
7. What is the current price of the highest reviewed product
8. How many items with a positive review were sold
9. Do we have any products with a high review in stock
10. Which supplier supplies our highest reviewed product

# 10 SQL Queries

Each member was responsible for formulating each of their 10 questions and executing the queries. This section will show the SQL query, results, and execution plan.

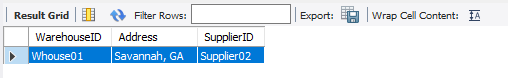
## Nick F.[Categories, Suppliers, Warehouses, Products]

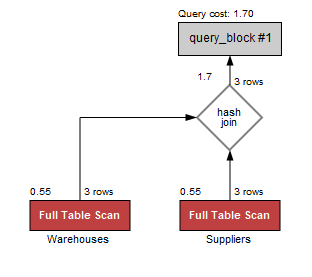
**-- Q1. List all if any warehouses that share an address with a supplier.**

**select WarehouseID, Warehouses.Address, SupplierID**

**from Warehouses, Suppliers**

**where Warehouses.Address=Suppliers.Address**

****

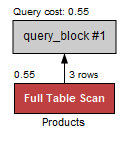
****

**-- Q2. Which warehouse has the most stock?**

select WarehouseID, Max(Stock) as maximumStocked

from Products;





**-- Q3. Are there any warehouses shared between suppliers?**

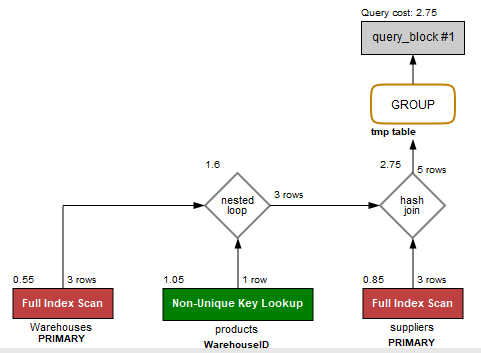
**select Warehouses.WarehouseID**

**from Warehouses,Suppliers,Products**

**where Products.WarehouseID=Warehouses.WarehouseID and Products.SupplierID!=Suppliers.SupplierID**

**group by Warehouses.WarehouseID**

****

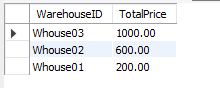
****

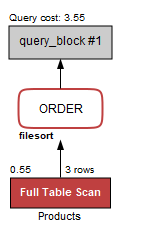
**-- Q4. Which warehouse has the highest value stockpile?**

**select WarehouseID,(stock\*price) as TotalPrice**

**from Products**

**order by TotalPrice desc**

****

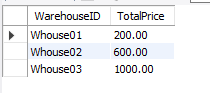
****

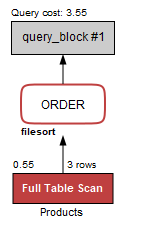
**-- Q5. Which warehouse has the lowest value stockpile?**

select WarehouseID,(stock\*price) as TotalPrice

from Products

order by TotalPrice



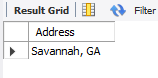


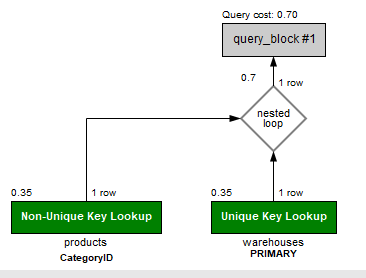
**-- Q6. List all warehouse addresses that have storage devices in them.**

select Warehouses.Address

from Products, Warehouses

where (Products.CategoryID="storage") and (products.WarehouseID=warehouses.WarehouseID)



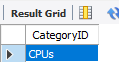


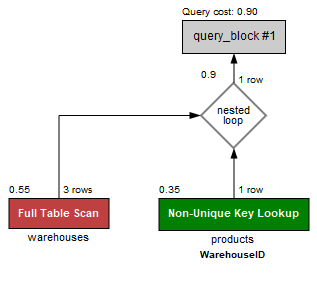
**-- Q7. List all devices stored at Macon, GA**

select Products.CategoryID

from Products, Warehouses

where (Warehouses.Address="Macon, GA") and (Warehouses.WarehouseID=Products.WarehouseID)

****

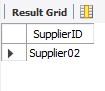
****

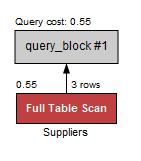
**-- Q8. Which supplier is located in Savannah, if any?**

**select SupplierID**

**from Suppliers**

**where Address="Savannah, GA"**

****

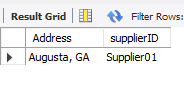
****

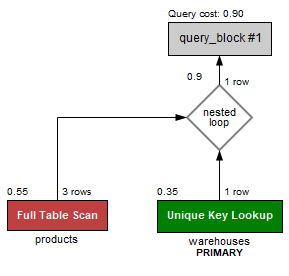
**-- Q9. Which warehouse are the ATX motherboards stored in, and which supplier provides them?**

**select Warehouses.Address, SupplierID**

**from Products,Warehouses**

**where (Name="ATX Motherboard")and(Warehouses.WarehouseID=products.WarehouseID)**

****

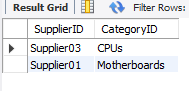


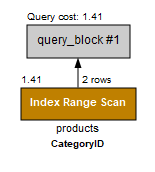
**-- Q10. Which suppliers provide CPUs and Motherboards?**

select SupplierID,CategoryID

from Products

where (CategoryID="CPUs" or CategoryID="Motherboards")



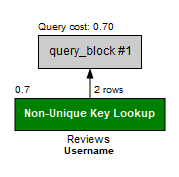


## Nick H[Customers, Reviews]

**-- Q1) What are all the reviews from a specific customer? SELECT SKU, Rating, VerifiedPurchase, Date**

**FROM reviews**

**WHERE username = “nhackett00”**

****

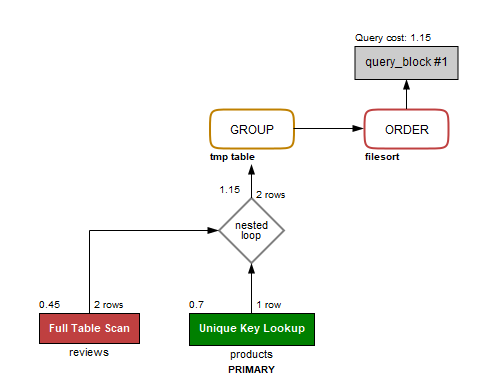
**-- Q2) Which customers tend to rate items more highly? select Username, avg(Rating) average**

**from reviews**

**left join products on products.sku = reviews.sku**

**group by username**

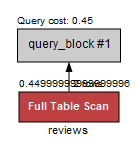
**order by average DESC;**

****

**-- Q3) Which reviews were from someone who has a verified purchase? SELECT \***

**FROM reviews**

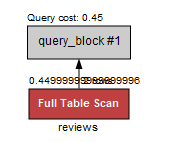
**where VerifiedPurchase = 1**

****

**-- Q4) Which reviews were from someone who does not have a verified purchase? SELECT \***

**FROM reviews**

**where VerifiedPurchase = 0**

****

**-- Q5. Which SKUs have the highest average rating?**

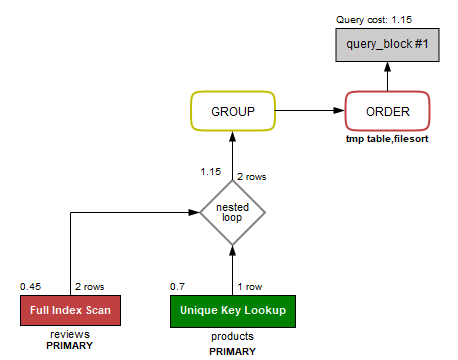
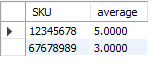
**select reviews.SKU, avg(Rating) average**

**from reviews**

**left join products on products.sku = reviews.sku**

**group by SKU**

**order by average DESC;**

****

**-- Q6. Which SKUs have the lowest average rating?**

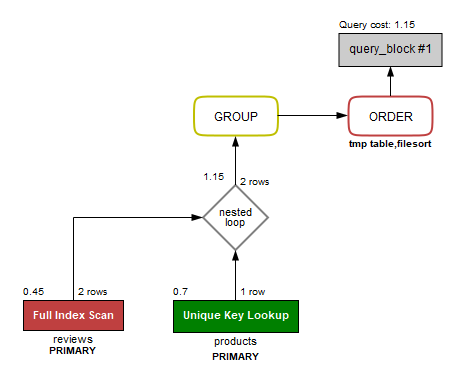
**select reviews.SKU, avg(Rating) average**

**from reviews**

**left join products on products.sku = reviews.sku**

**group by SKU**

**order by average ASC;**

****

**-- Q7. Which SKUs are very polarizing, i.e. have both very high and very low ratings?**

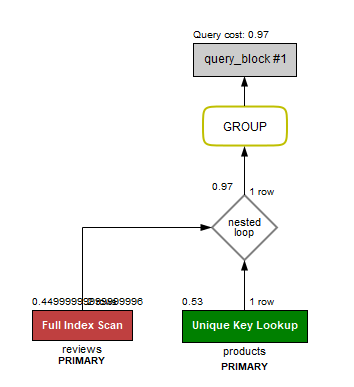
**select reviews.SKU**

**from reviews**

**left join products on products.sku = reviews.sku**

**where rating = 5 or rating = 1**

**group by SKU;**

****

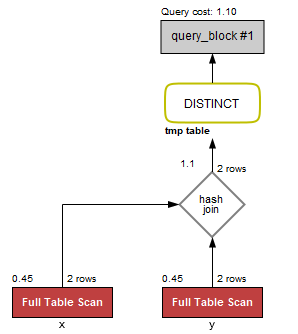
-- Q8. Did any SKUs have a lot of reviews posted on the same date?

**select distinct x.sku, x.date**

**from reviews x**

**join reviews y**

**on y.SKU != x.SKU and y.date = x.date**

****

**-- Q9. Are multiple customers posting reviews from the same address?**

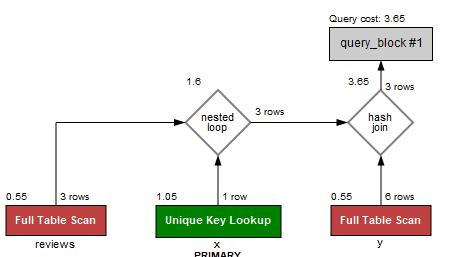
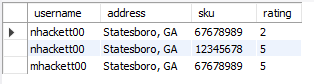
**select x.username, x.address, reviews.sku, reviews.rating**

**from reviews, customer x**

**join customer y**

**on x.address = y.address and x.username != y.username**

**where reviews.username = x.username**

****

**-- Q10. Are any customers posting many ratings at a time?**

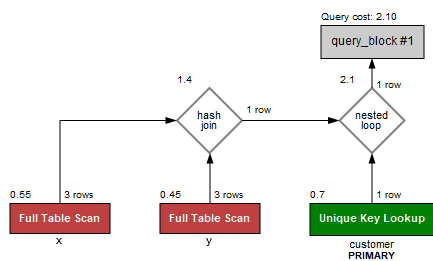
**select customer.username, x.sku, x.Rating, x.date**

**from customer, reviews x**

**join reviews y**

**on x.date = y.date and x.sku != y.sku**

**where customer.username = x.username and x.date**

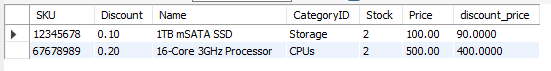
****

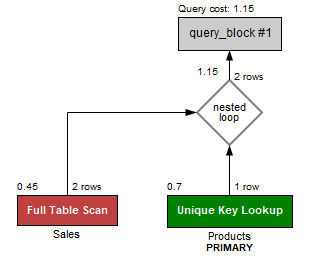
## Nick I. [Sales, Items]

**-- Q1. What products are on sale?**

select Sales.\*, Name, CategoryID, Stock, Price, Price\*(1-Discount)discount\_price from Sales, Products

where Sales.SKU = Products.SKU



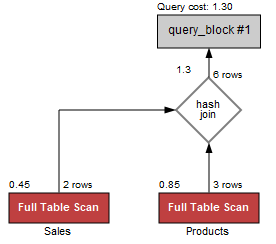
****

**-- Q2. Which product has the best discount?**

**select Name, Max(Discount) as LargestDiscount**

**from Sales, Products;**

****

****

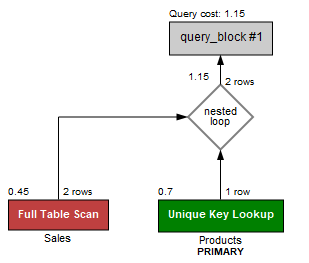
**-- Q3. What is the cheapest item in the store after discounts?**

**select Sales.\*, Name, CategoryID, Stock, Min(Price) as SmallestPrice, Price\*(1-Discount)discount\_price**

**from Sales, Products**

**where Sales.SKU = Products.SKU**

****

****

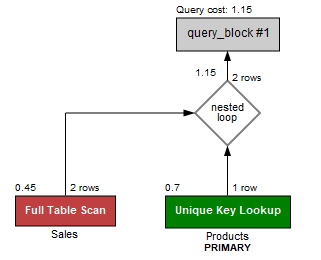
**-- Q4. What is the most expensive item after discounts?**

**select Sales.\*, Name, CategoryID, Stock, Max(Price) as HighestPrice, Price\*(1-Discount)discount\_price**

**from Sales, Products**

**where Sales.SKU = Products.SKU**

****

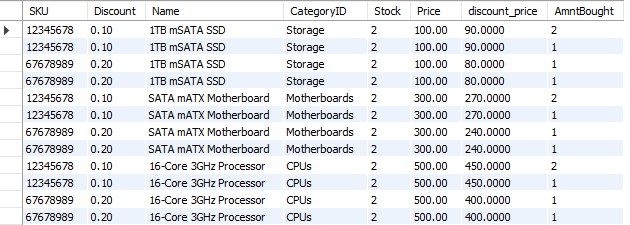
****

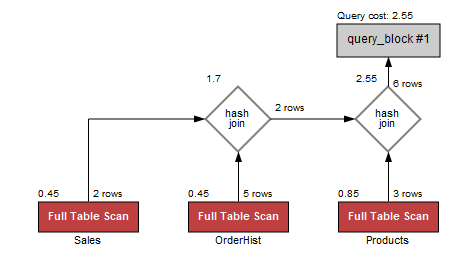
**-- Q5. How many discounted items have been purchased?**

**select Sales.\*, Name, CategoryID, Stock, Price, Price\*(1-Discount)discount\_price, AmntBought**

**from OrderHist, Sales, Products**

**where Sales.SKU = OrderHist.SKU**

****

****

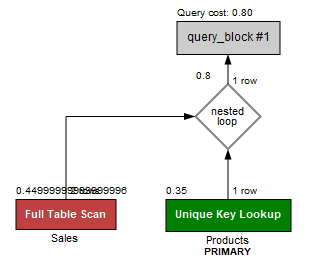
**--Q6. Which suppliers have discounted items?**

**select Name, SupplierID, Discount, Stock**

**from Sales, Products**

**where Sales.SKU = Products.SKU and Discount>0**

****

****

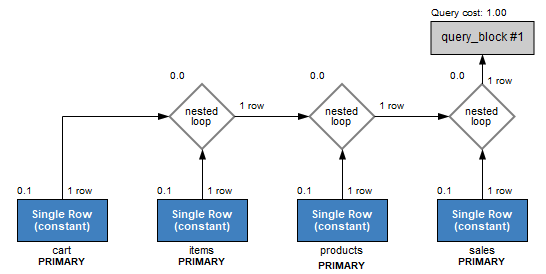
**-- Q7. Which discounted items are in nistre00’s cart?**

**select Name, CategoryID, Stock, Price, Cart.SerialNum, products.SKU, Price\*(1-Discount)discount\_price**

**from cart, items, products, sales**

**where username = 'nistre00' and cart.SerialNum = items.SerialNum and items.sku = products.sku and sales.sku=items.sku**

****

****

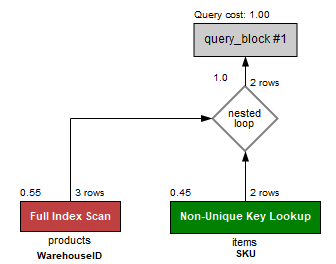
**-- Q8. List all serial numbers for a particular SKU (12345678)**

**select items.SerialNum**

**from items, products**

**where products.SKU=12345678 and items.SKU=products.SKU**

****

****

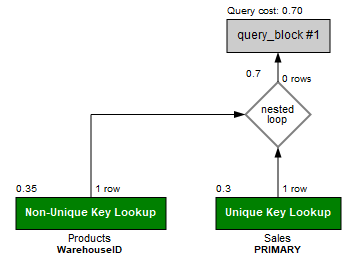
**-- Q9. What discounted items are at Warehouse01?**

**select Name, WarehouseID, Discount, Stock**

**from Sales, Products**

**where Sales.SKU = Products.SKU and Discount>0 and WarehouseID="Warehouse01"**

****

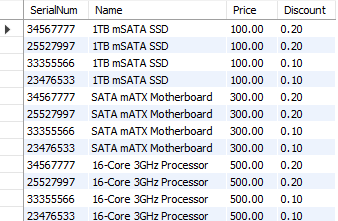
****

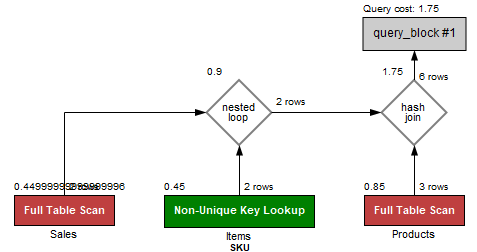
**-- Q10. What are the serial numbers for all discounted items?**

**select SerialNum, Name, Price, Discount**

**from Sales, Products, Items**

**where Sales.SKU = Items.SKU and Discount>0**

****

****

**Jonah W. [Customer, Cart, OrderHist]**

-- Q1. What is in Logan Salem’s cart that isn’t in Nick Hackett’s?

select p.name as Product\_Name

from products p, items i, cart c

where c.username = "lsalem00" and c.serialnum = i.serialnum

and i.sku = p.sku and p.name not in (

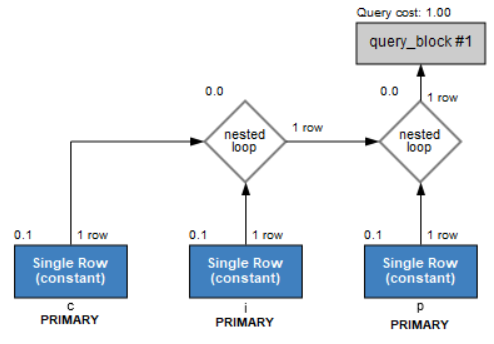
select p.name

from products p, items i, cart c

where c.username = "nhackett00" and c.serialnum

i.serialnum and i.sku = p.sku);





-- Q2. What is the subtotal of Nick Ferrara’s cart?

select c.fname as FirstName, c.lname as LastName,

sum(p.price) as Subtotal

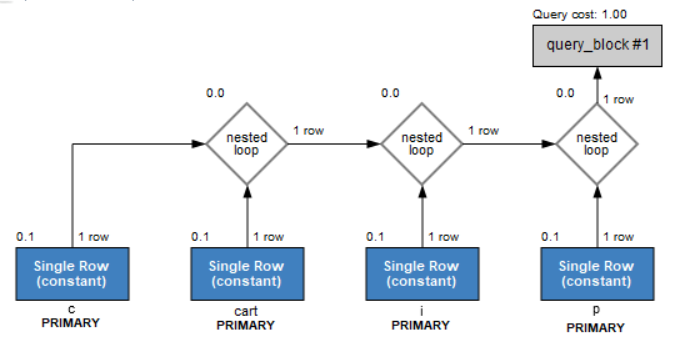
from customer c, cart, products p, items i

where c.username = "nferrara00" and c.username =

cart.username and cart.serialnum = i.serialnum and i.sku =

p.sku;





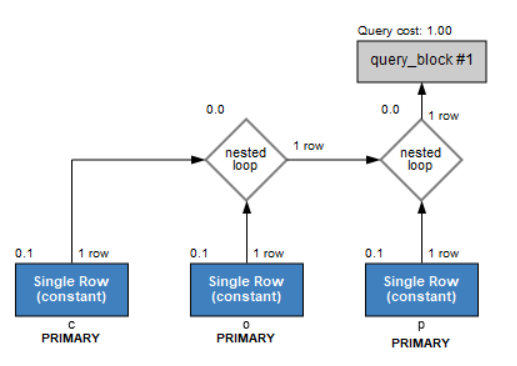
-- Q3. What has Jonah Walker ordered?

select c.fname as FirstName, c.lname as LastName, p.Name as Products

from customer c, products p, orderhist o

where o.username = c.username and c.username = "jwalker00" and o.sku = p.sku;





-- Q4. What is the most expensive item in Jonah Walker’s cart?

select c.fname as FirstName, c.lname as Last Name, p.name as

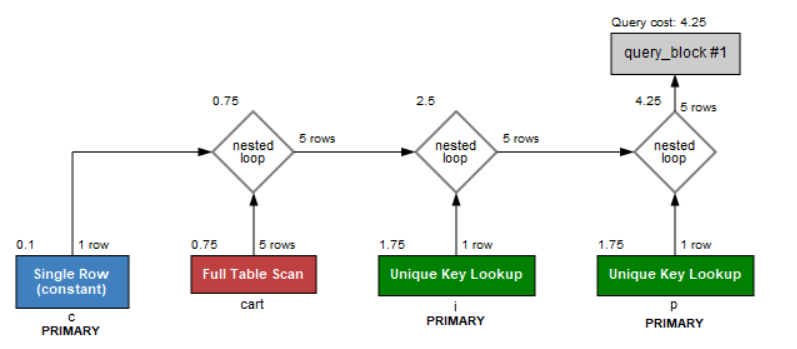
Most\_Expensive\_Item, max(p.price) as Price

from customer c, products p, cart, items i

where c.username = c.username and c.username = "jwalker00"

and cart.serialnum = i.serialnum and i.sku = p.sku;





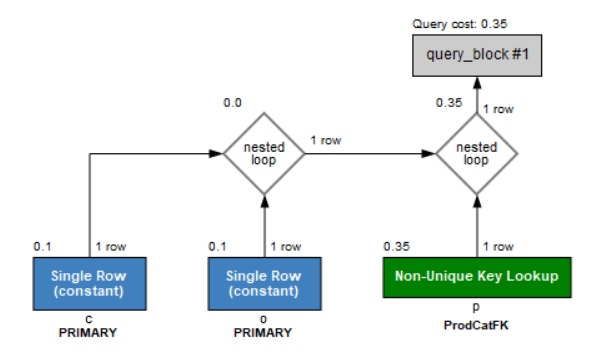
-- Q5. How many processors has Nick Istre purchased?

select c.fname as FirstName, c.lname as LastName, count(\*) as Num\_of\_Processors\_Bought

from customer c, orderhist o, products p

where c.username = o.username and o.username = "nistre00" and p.categoryid = "CPUs";





-- Q6. Sort the customers by the number of items purchased (from most to least).

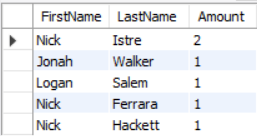
select c.fname as FirstName, c.lname as LastName,

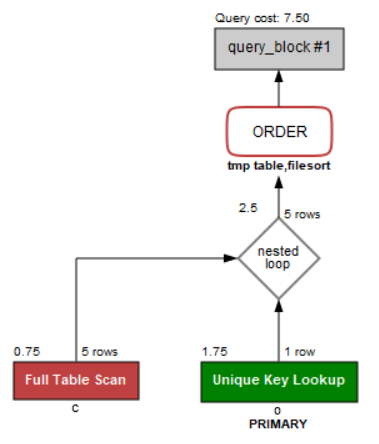
o.amntbought as Amount

from customer c, orderhist o

where o.username = c.username

order by o.amntbought desc;



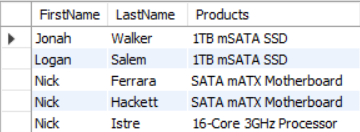


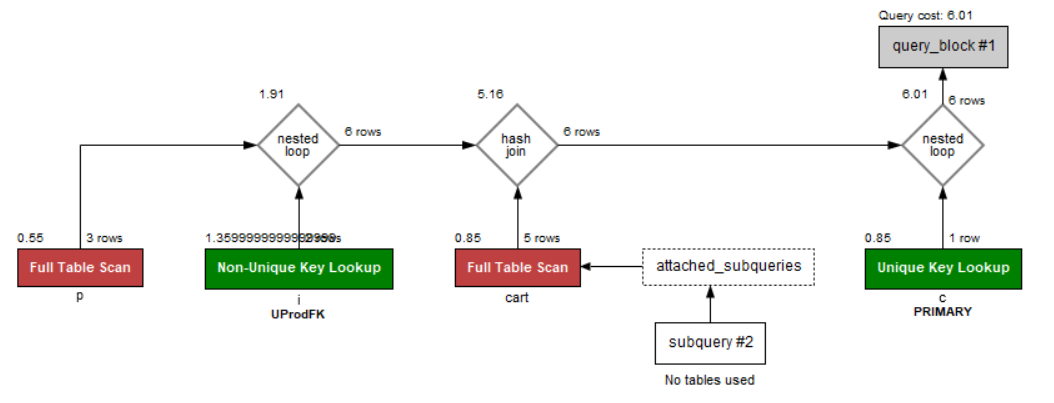
-- Q7. List the customers with items in their cart.

select c.fname as FirstName, c.lname as LastName, p.name as Products

from customer c, cart, products p, items i

where c.username = cart.username and cart.serialnum = i.serialnum and i.sku = p.sku and exists(select cart.username);



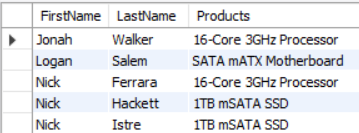


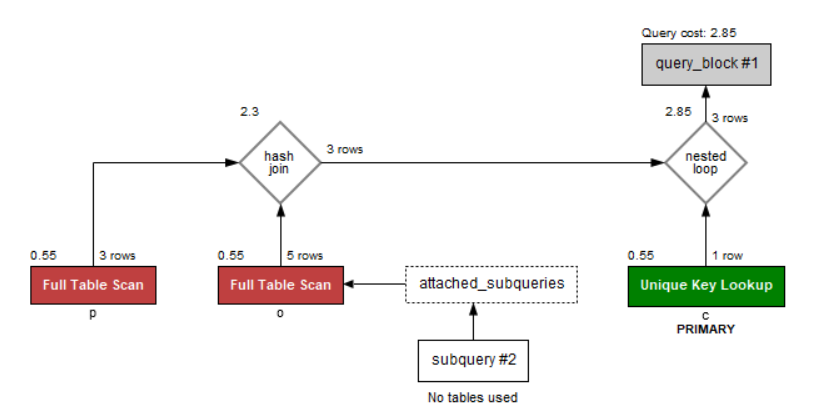
-- Q8. List the customers who have ordered something.

select c.fname as FirstName, c.lname as LastName, p.name as Products

from customer c, orderhist o, products p

where c.username = o.username and o.sku = p.sku and exists(select o.username);



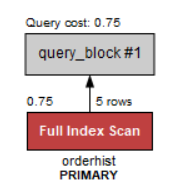


-- Q9. How many products have been sold?

select count(\*) as Num\_of\_Items\_Sold

from orderhist;





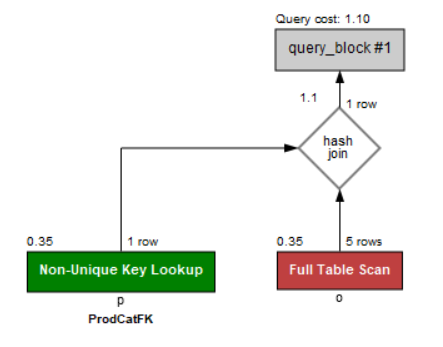
-- Q10. How many storage devices have been ordered?

select count(\*) as Num\_of\_Storage\_Devices\_Sold

from orderHist o, products p

where o.sku = p.sku and p.categoryid = "Storage";





Logan Salem (reviews, Products}

q1 ) Which products have a positive review

SELECT \*

FROM reviews

Group by Positive

q2) which products have a negative review

SELECT \*

FROM reviews

Group by Negative

q3) How many reviews does a specific product have

SELECT \*

FROM reviews

Count \*

q4) What date was the review left

SELECT \* Dates

FROM reviews

q5) Is a product with a positive review on sale

SELECT \* Positive, sale

FROM review

q6) Where is a product with a positive review located

SELECT, products.CategoryID

FROM products, warehouses

WHERE review = Positive

q7) What is the current price of the highest reviewed product

SELECT, Name, price, CategoryID

FROM reviews, products

WHERE review = highest

q8) How many items with a positive review were sold

SELECT count(\*) as num of items Sold with review positive

FROM orderhist;

q9) Do we have any products with a high review in stock

SELECT products, stock

From reviews

Where review = high

q10) Which supplier supplies our highest reviewed product

SELECT \* SupplierID, Suppliers

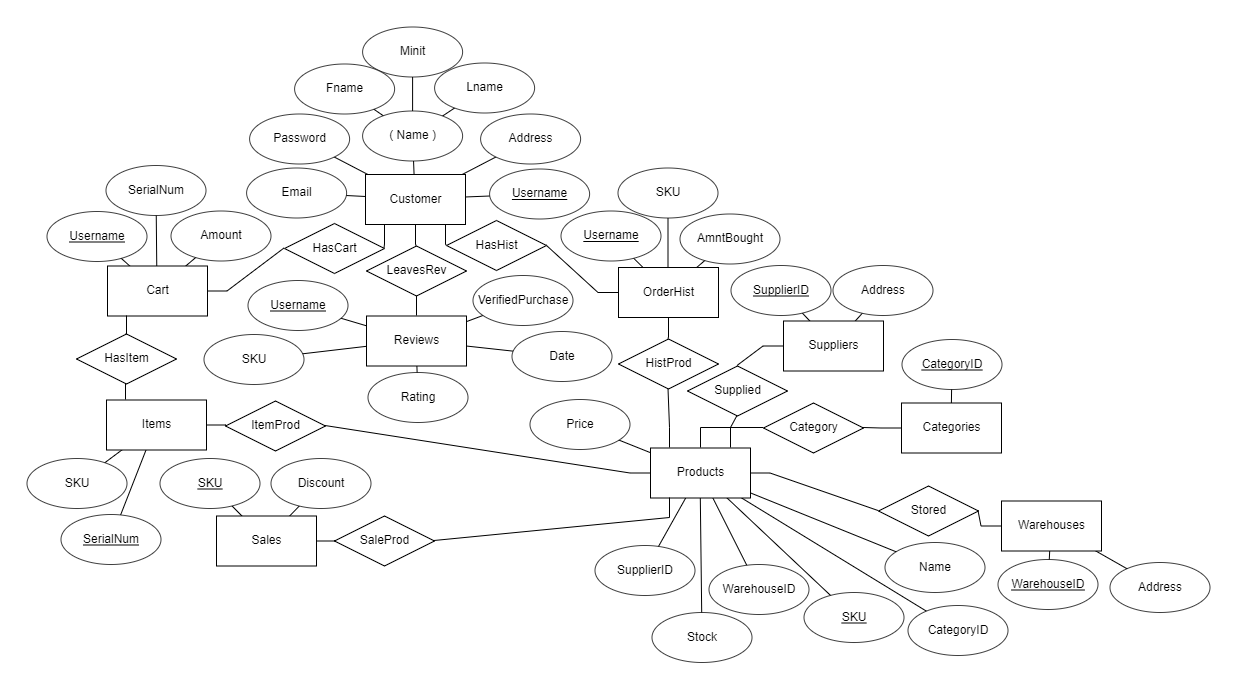
FROM Suppliers

Where product review = High

# Database Application Scope

The scope of this project was fairly small, with the database mostly serving to help keep track of orders and supply. It ensures that orders are accounted for, and all objects and suppliers are in the correct locations. Should we continue to add onto and iterate on this project, we likely would hope to implement a GUI utilizing java, and provide a more user friendly experience for users not familiar with database systems.

**ER Schema Diagram**

****

**1st Normal Form**

**Our database design is in 1st Normal Form as all attributes in all relations depend on the primary key of that relation.**

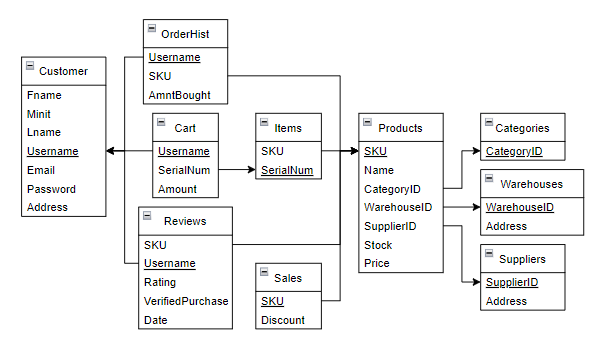
**2nd Normal Form**

**Our database design is in 2nd normal form as we do not have any partial keys nor functional dependencies to partial keys.**

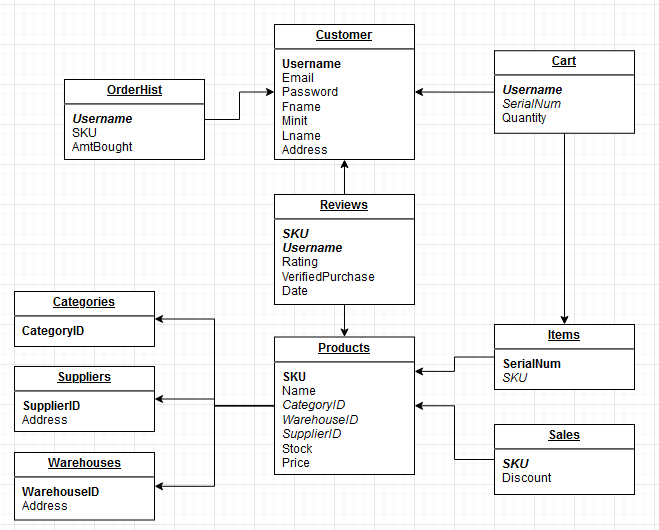
**3rd Normal Form**

**Our database design is in 3rd normal form as it does not contain any transitive dependencies to the key nor any functional dependencies to anything but the key.**

# Storefront Initial - Physical Model

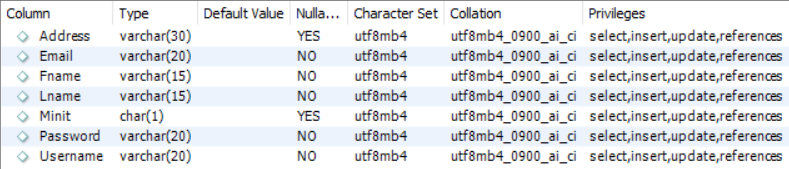


# Storefront Redesign - Physical Model

****

# Tables and Data Types

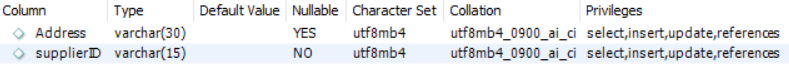
## CUSTOMER Table



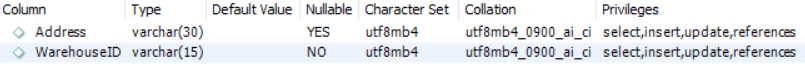
## CATEGORIES Table



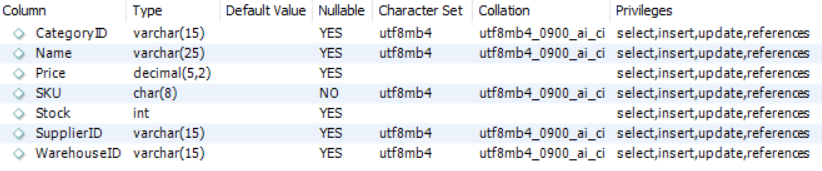
## SUPPLIERS Table



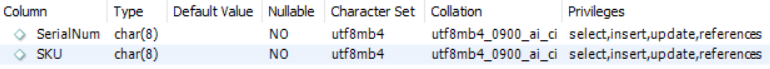
## WAREHOUSE Table



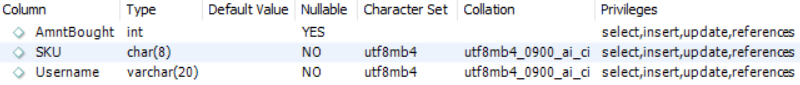
## PRODUCTS Table

****

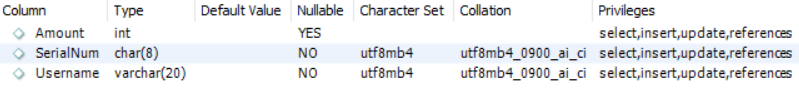
## ITEMS Table

****

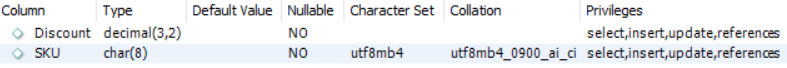
## ORDERHIST Table



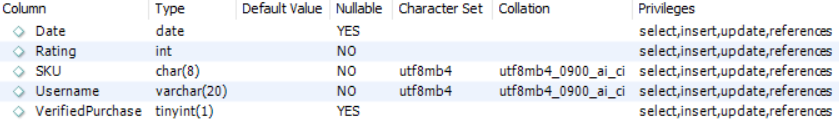
## CART Table



## SALES Table



## REVIEWS Table



# Creating the Tables

create table Customer (

Fname varchar(15) NOT NULL,

Minit char,

Lname varchar(15) NOT NULL,

Username varchar(20) NOT NULL,

Email varchar(20) NOT NULL,

Password varchar(20) NOT NULL,

Address varchar(30),

constraint CustPK primary key (Username));

create table Categories (

CategoryID varchar(15) NOT NULL,

constraint CatPK primary key (CategoryID));

create table Suppliers (

supplierID varchar(15) NOT NULL,

Address varchar(30),

constraint SupplierPK primary key (SupplierID));

create table Warehouses (

WarehouseID varchar(15) NOT NULL,

Address varchar(30),

constraint WarehousePK primary key (WarehouseID));

create table Products (

SKU char(8) NOT NULL,

Name varchar(25),

CategoryID varchar(15),

WarehouseID varchar(15),

SupplierID varchar(15),

Stock int,

Price decimal(5,2),

constraint ProdPK primary key (SKU),

constraint WareHouseFK foreign key (WarehouseID) references Warehouses(WarehouseID),

constraint SupplyFK foreign key (SupplierID) references Suppliers(supplierID),

constraint ProdCatFK foreign key (CategoryID) references Categories (CategoryID)

on update cascade);

create table Items (

SKU char(8) NOT NULL,

SerialNum char(8) NOT NULL,

constraint UProdPK primary key (SerialNum),

constraint UProdFK foreign key (SKU) references Products (SKU)

on update cascade);

create table OrderHist (

Username varchar(20) NOT NULL,

SKU char(8) NOT NULL,

AmntBought int,

constraint OrdersPK primary key (Username),

constraint OrdersFK foreign key (Username) references Customer (Username));

create table Cart (

Username varchar(20) NOT NULL,

SerialNum char(8) NOT NULL,

Amount int,

constraint CartPK primary key (Username),

constraint CartFK foreign key (Username) references Customer (Username));

create table Sales (

SKU char(8) NOT NULL,

Discount decimal(3,2) NOT NULL,

constraint SalesPK primary key (SKU),

constraint SalesFK foreign key (SKU) references Products (SKU));

create table Reviews (

SKU char(8) NOT NULL,

Username varchar(20) NOT NULL,

Rating int NOT NULL,

VerifiedPurchase bool,

Date date,

constraint ratingBounds check (Rating >= 0 and Rating <= 5),

constraint reviewsPK primary key (Rating),

constraint reviewsSKUFK foreign key (SKU) references Products(SKU),

constraint reviewsUNFK foreign key (Username) references Customer(Username));

# Referential Integrity and Constraints

See “Creating the Tables” section for information on the referential integrity and constraints placed on our database.

# Future Opportunities

We believe that this project gave us a lot of insights and practice in creating, modifying and organizing a database’s contents. By having us create a practical example of a database that might be used in a real work setting it helped us to understand the principles behind why certain features and aspects of a database are organized and accessed the way they are. While we do not foresee this particular project being reused for anything in the future, as it would need some modification to be completely useful, it served as an important learning experience for everyone involved on the team.

# Post Mortem

**Nick F.**

I was able to help with some of the initial ideas as far as organizing and adding tables to the database, as well as a lot of the queries used to access the database in the initial project design. It was an interesting learning experience that helped teach me a lot about how to organize a database and where data should be separated and why. I also learned a lot about actually accessing data, and what constraints are needed for a database to function as intended for its proper use case. I am glad to have helped where I could for this project, but would not have been able to do this without the help of the other members, who also helped with designing the database and wrote the tables and declarations needed to implement everything.

**Nick H.**

Overall, this project went very smoothly. Everyone contributed adequately and I am confident that we would all say we learned a lot from this project about writing and maintaining databases. It is clear that SQL is one of those programming languages that can be very obtuse in concept and difficult to understand conceptually but by having an actual database to work with we were able to gain a practical understanding of how the language functions. Writing questions first and then having to formulate queries based on them forced us to learn all the different functions and keywords and stopped us from sticking to really basic usage of SQL. This project definitely helped me understand how a database would function in the real world.

## Nick I.

The project was organized very well and contribution felt equally balanced among all the members. Initially the design of the database and constructing everything together seemed like it would be difficult although it became very simple as we put values in and designed the tables and schemas. I was able to learn a lot about the inner workings of the database and how a system can obtain values depending on what is asked of it which became much easier with a physical database to work with and because of the gradual production of all the separate parts, it was easy to understand everything involving it. Designing queries for each section allowed us to become familiar with every table and the SQL involved with each and it allowed us to develop a much better understanding of database systems and the coding involved.

## Logan S. All of our group members did a wonderful job helping each other. I personally had issues with my sql towards the end of the project so i could not get pictures of my ten questions. Other than that large issue I was able to understand all aspects of the project. The project did a good job at teaching how a database is necessary to a group or team developing any online system in the modern era. I was able to understand the workings of a database and its complexities. The project exposed me and the group to database development as well as the inner workings of mysql. One thing that I will take away from this project is that within tech issues will arise and you will have to be able to resolve them in a timely manner.

**Jonah W.**

Each member of the group proved to be very useful and the workload was reasonably divided among us. This project has allowed us to better understand SQL and database design through the interactivity of designing our own database. Concepts such as referential integrity as well as entity strength would have proven to be much more difficult without having an actual database to work with. Each step along completing this project led to increasing my understanding of database design. Formulating queries in order to obtain specific information from our database incentivized furthering our knowledge of SQL beyond the basics. Having a rather basic idea for a database with a high required number of relations forced us to unknowingly put our database into the three normal forms, and this realization solidified my understanding of said forms.