# Retail Summary

Retail management is the process in which customers (buyers) can obtain a desired good from a retail store. The goal of this application is to create an effective and efficient database that would allow customers to surf through millions of products and be able to purchase items off the internet. The database will generate account reports, transaction reports, and order reports. These reports further include order details such as the shipping date, order date and the status of the product. The database will also give the customer all the products that they have bought over the past six months. The database will prioritize the customers and the transaction details. It does this because these two details are important for a seemingly efficient database application. The database application will store:

* the information of employees
* the product information
* the buyer’s information
* the transaction details
* the suppliers’ information
* the information of the product

# Modelling assumptions

The Modelling Assumptions made for this database include:

* One supplier has many products, but a product has only one supplier
* Many products have many orders.
* One customer has many orders, but an order has only one customer.
* One customer can have many transactions, but a transaction has only one customer.
* One employee works with many customers, but many customers can get help from one employee.
* One transaction has one orders and an order has only one transaction.
* An employee may have many employees under him/her.

# Keys

The database application keys:

* Supplier Id: Uniquely identifies each supplier.
  + Foreign key in product
* Product id: Uniquely identifies each product
  + Foreign key in order detail entity.
* Customer Id: Uniquely identifies each customer
  + Foreign key in transaction entity
  + Foreign key in order entity
* Transaction Id: Uniquely identifies each transaction
  + Foreign key in order entity
* Employee ID: Uniquely identifies each employee
  + Foreign key in customer entity
  + Foreign key in employee entity
* Order ID: Uniquely identifies each order
  + Foreign key in order detail entity

# Checks and Triggers

## Checks

* Checks if the phone number of both the customer and supplier are in the “240-885-6547” format.
* Checks if the email of the customer, employee, and supplier are in the “tacocat@yahoo.com” format.

## Triggers

* Delete Triggers
  + A trigger that deletes the account id in transaction and order where it matches the deleted account id in customer.
  + A trigger that deletes the Order number from transaction where it matches the deleted Order Number in Order table.
* Insert Triggers
  + A trigger that will not allow one to insert a tuple in order detail unless the product id and order number exist in the product and order table respectively.
  + A trigger will not let you insert a tuple in order unless the account ID exists in the customer table.
  + A trigger that will not allow one to insert a tuple in transaction unless the account ID and order number exist in the customer and order table respectively.
* Update Triggers
  + Each Foreign key has an update Cascade so placing an update trigger would be redundant.

# Error Handling

For our error handling, we used:

* A JavaScript code that will check if the user in the deleteAccount.php has inputted something or a number.
* A SQL Check that sees if phone in supplier is in a certain format.
* A SQL Check to see if email in employee is in a certain format.
* A SQL Check to see if email in customer is in a certain format.
* A SQL Check to see if email in supplier is in a certain format.
* A HTML code to validate the right phone number in in the createAccount.PHP.

We used JavaScript because it is real programming language which makes it easier, quicker, and more efficient to handle errors because it is on the client side. We chose the SQL checks because we are most familiar with and will make sure to handle errors on the server side but it may be slower.

# Challenges and Changes

## Challenges

* Triggers did not execute properly. We figured out all the problems were related with the Order table, so we dropped it and created a new one. Now, they all seem to work fine.
* Our forms worked but the PHP files did not execute as would have liked too.

## Changes

* We fixed up all three of the PHP code.
* We fixed up all the triggers.
* We created test cases for each trigger.
* We validated our phone attribute (Using HTML) to insure it was in the correct pattern.