Pharmacy Database

Outline:

Users:

* Cashiers
  + See price of product given its ID

-SELECT Name, SKU FROM Product WHERE SKU = inputSKU

* + Record the sale of something, this will add to income statement and reduce the stock of that something.

bool invalid = false;

-foreach(Product p in productsForSale)

{  
 int stock = SELECT Stock FROM Product WHERE Name = p.name;

If (stock < p.AmountRequested) invalid = true; //ERROR: Not enough stock.

Bool isPerscription = SELECT isPerscription FROM Product WHERE Name = p.name;

If(isPerscription) //WARNING: THIS SHIT IS REAL DUDE

}

If (invalid)

{

System.out.println(“Java error: you’ve requested a ridiculous amount of stock.”);

Echo Console.WriteLine(“System.out.println(\”Debug.Log(\”Log.l(“…”)));

}

INSERT INTO Sale VALUES(…new Sale values…..)

foreach(Product p in productsForSale)

{

INSERT INTO SP VALUES(Sale.id, p.id); //this causes a trigger in SP to reduce stock in Product

}

* + Cashiers must be warned of prescription products.

-done in error checking loop

* Pharmacists (Same as cashiers):
  + Queue of prescriptions to be picked.

SELECT \* FROM Prescription WHERE isFilled = true ORDER BY DateFilled FETCH FIRST 5 ROWS ONLY

* + - Remove and add prescriptions.

DELETE FROM Prescription WHERE Name = InputName

INSERT INTO Prescription VALUES (…..values…..)

* + When picked up remove from Queue and add to income

-have a trigger flag when adding or removing prescriptions

* Administrators:
  + Query income
    - Date

SELECT ID FROM Sale WHERE Date >= StartDate && Date <= StartDate (imcomplete)

* + - Department
    - Product
    - Employees (Attribute)
* Customers:
  + Customers want see products organized by a variety of attributes:
    - Price

SELECT \* FROM Product ORDER BY Price

* + - Department

SELECT \* FROM Product ORDER BY Department

* + - Stock level

SELECT \* FROM Product ORDER BY Stock

* + Customers can see if their prescription is ready.

SELECT DateDelivered FROM Prescription WHERE Customer = InputName

Console.printf(“%s %s %s %s\n”, “It”, “has”, DateDelivered == NULL ? “not been” : “been”, “delivered.”);

* + Submit prescriptions

INSERT INTO Prescriptions VALUES (…….prescriptions values……)

* + See prescription they picked up (prescription history).

SELECT \* FROM Prescription WHERE Customer = InputName

Advanced Queries:

Find all products that have ever been sold.

SELECT \* FROM Product WHERE EXIST (SELECT \* FROM SaleProduct WHERE Product.ID = SaleProduct.Product);

Find all products that have at least one other product with the same price.

SELECT \* FROM Product x WHERE Price = ANY (SELECT Price FROM Product y WHERE x.Name <> y.Name);

Find all products with the greatest price.

SELECT \* FROM Product x WHERE Price >= ALL (SELECT Price FROM Product y);

Find all products with stock 2, 4 or 6.

SELECT \* FROM Product WHERE Stock IN (2, 4, 6);

Find all products in the Drugs department with a stock level between 5 and 10

SELECT \* FROM Product WHERE Department = ‘Drugs’ AND Stock BETWEEN 5 AND 10;