**SIMPLE DATABASE PROJECT**

A new client, Bubs Glover, the owner of Bubs’ Bigtime Baby Booties, has reached out to you for help building his business a database from the ground up.

Use Your MySQL Database Administration skills to:

Design a database from scratch, which will capture information about Bubs’ customers, the purchases they make, his products, and his employees.

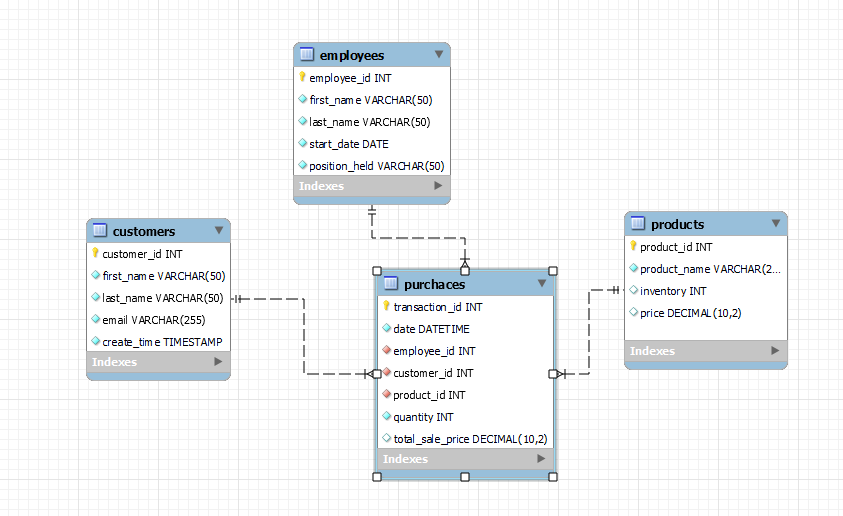
Bubs wants you to track the following information:

1. customers
   1. Customer id
   2. first name,
   3. last name,
   4. Email
   5. create\_time
2. employees
   1. Employee id
   2. first name,
   3. last name,
   4. start date,
   5. position held
3. Products
   1. Product id
   2. Product name
   3. Inventory
   4. price
4. purchases customers make
   1. Transaction id
   2. when it was purchased
   3. Employee who sold product
   4. which customer,
   5. What was purchases
   6. quantity
   7. for how much money

\*Bobs request

\*My Suggestions

Given the database design I came up with, I used Workbench to create an EER diagram of the database, making sure to include things like; primary keys and foreign keys, reasonable data types for each column, any constraints like which columns need to be unique, which ones are allowed to have NULL values, etc.



Now it's time to create the schema bubsbooties. Within that schema, I created the tables that I had previously diagramed out, making sure they relate to each other, and that they had the same data types and constraints I selected.

CREATE SCHEMA bubsbooties DEFAULT CHARACTER SET utf8mb4;

CREATE TABLE customers

(

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

email VARCHAR (255) NOT NULL UNIQUE,

create\_time TIMESTAMP

);

For the customers table I made the customer\_id the primary key and allowed it to auto increment to avoid duplication. I also made email unique to prevent repeat clients from creating multiple accounts. All other fields were giving the NOT NULL constraint to avoid missing data.

CREATE TABLE employees

(

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

start\_date DATE NOT NULL,

position\_held VARCHAR(50) NOT NULL

);

For the employee table I made the employee\_id the primary key and allowed it to auto increment to avoid duplication. All other fields were giving the NOT NULL constraint to avoid missing data.

CREATE TABLE products

(

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR (255) NOT NULL,

inventory INT NOT NULL,

price DECIMAL(10,2) NOT NULL

);

For the products table I made the product\_id the primary key and allowed it to auto increment to avoid duplication. All other fields were giving the NOT NULL constraint to avoid missing data.

CREATE TABLE purchases

(

transaction\_id INT AUTO\_INCREMENT PRIMARY KEY,

date DATE NOT NULL,

employee\_id INT,

customer\_id INT,

product\_id INT,

quantity INT,

total\_sale\_price DECIMAL(10,2),

FOREIGN KEY (employee\_id)

REFERENCES employees(employee\_id),

FOREIGN KEY (customer\_id)

REFERENCES customers(customer\_id),

FOREIGN KEY (product\_id)

REFERENCES products(product\_id)

);

For the purchases table I made the product\_id the primary key and allowed it to auto increment to avoid duplication. I created foreign key relationships for the %\_id columns back to their home table.

I used Mockaroo to generate data for the customers, employees and products tables.

Here are some example from each table:

-- customer data

insert into customers (first\_name, last\_name, email, create\_time)

values ('Trudi', 'Grigorushkin', 'tgrigorushkin0@intel.com', '2022-03-09');

Note - For this exercise, I manually entered the TIMESTAMP data in the customer table to avoid having them all be the same day.

-- employee data

insert into employees (first\_name, last\_name, start\_date, position\_held)

values ('Northrop', 'Maleby', '2023-03-19', 'manager');

-- products data

insert into products (product\_name, inventory, price)

values ('Yellow Necklacepod', 5, '18.16');

Then I combined some of the data from each table to accurately mimic transactions. For the initial insert I used excel to create a table with a formula to calculate the total sale price. (This will be done by SQL in the future…see below). For simplicity, only one type of product was purchased during each transaction.

-- purchases data

insert into purchases (date, employee\_id, customer\_id, product\_id, quantity, total\_sale\_price)

values ('2022-12-23', 4, 26, 13, 5, '83.35');

The next step is to create some triggers to maintain proper relationships between tables.

The first trigger I created was to have SQL generate the total amount of the purchase based on the quantity of products purchased and the price of the product.

CREATE TRIGGER purchase\_price

BEFORE INSERT ON purchases

FOR EACH ROW

SET NEW.total\_sale\_price :=

(SELECT NEW.quantity \* products.price

FROM products

WHERE products.product\_id = NEW.product\_id);

Then I created a trigger to update the inventory on the products page every time a purchase was made.

CREATE TRIGGER purchase\_update\_inventory

AFTER INSERT ON purchases

FOR EACH ROW

UPDATE products

SET inventory = inventory-1

WHERE product\_id = NEW.product\_id;

I tested the triggers with several more inserts until I was sure they were working properly.