Part B:

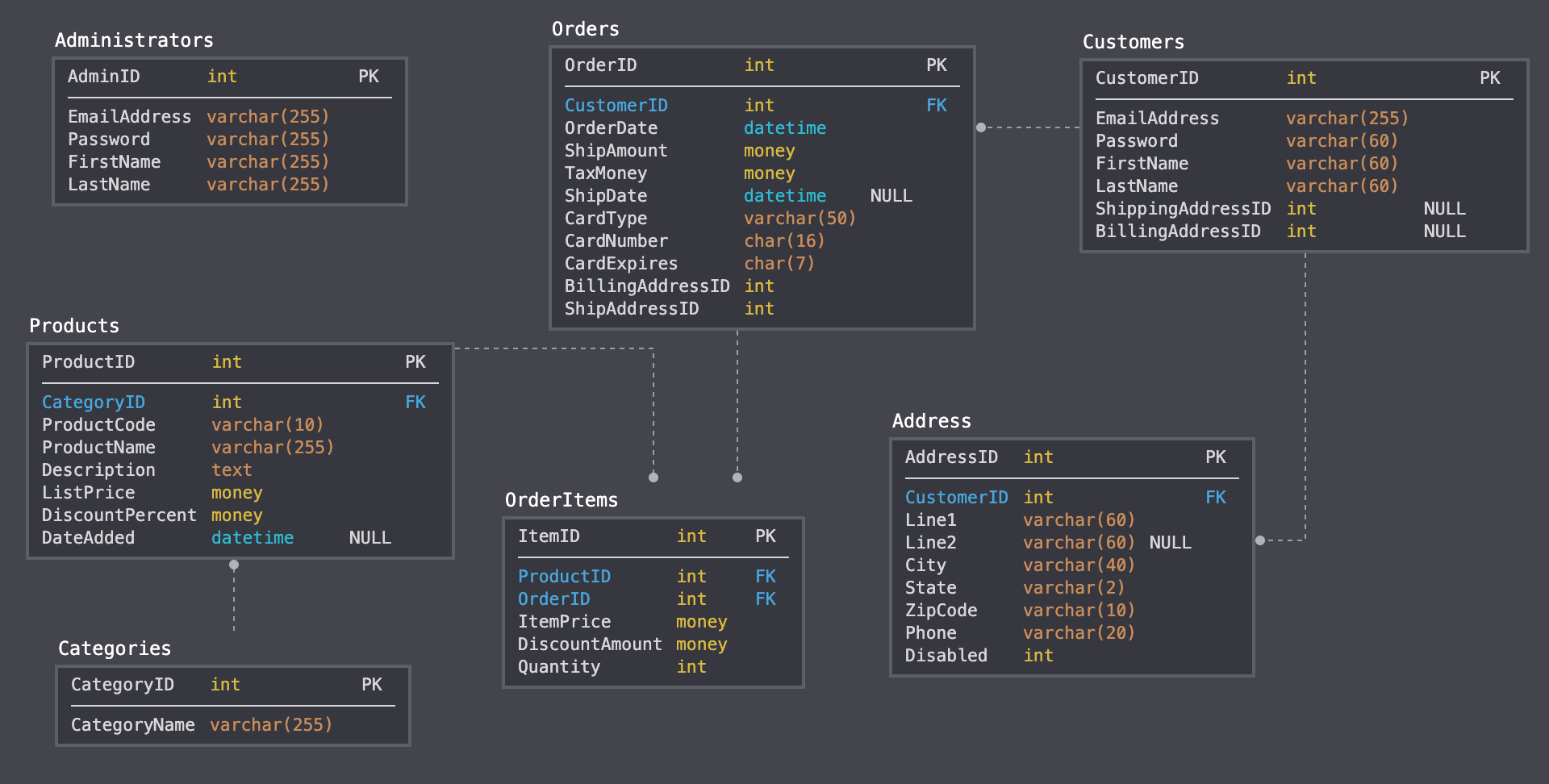
Q1. Create a database diagram that shows the relationships between the seven tables in the MyGuitarShop database.

The MyGuitarShop includes seven tables, one of which is independent – the Administrators table. The other six tables include the Products, Orders, Customers, Categories, OrderItems, and Addresses. For the most part, the database is straight forward; however, there are a coupld of relationships that I want to identify specifically.

The Products and Orders table has a many-to-many relationship. The databases uses the OrderItems table to link the two tables together. Also the Customers table makes use of an Addresses table to keep track of the Customer’s shipping and billing address. Using a separate table serves two primary purposes:

1. If the customer’s shipping and billing address is the same, only the AddressID is duplicated on the Customer’s table. Without the use of the Addresses table the address would have to be logged in both the shipping and the billing address columns. This would unnecessarily increase the size of the table, and expose the database to data consistency problems.
2. It could increase the performance of some Customers table queries that do not need the customer’s address information.

The last table relationship I want to highlight is the relationship between the Products and the Categories table. Instead of storing the CategoryName directly on the Products table itself, it is abstracted to a separate table. In the scenario in which a CategoryName changed, this would allow the database to be updated in a single spot, the relevant row/instance on the Categories table. Nothing would need to change on the Products table since that table is only storing the CategoryID.



Q2. A database diagram for a database that stores information about the downloads that users make.

I decided to create three different tables in order to meet the requirements: Users, Downloads, and Products tables. The Downloads table is where the magic happens; it links each download to the user via the UserEmail foreign key, and each download is also linked to a product via the ProductId foreign key. This setup allows each user and product to relate to more than one download.

