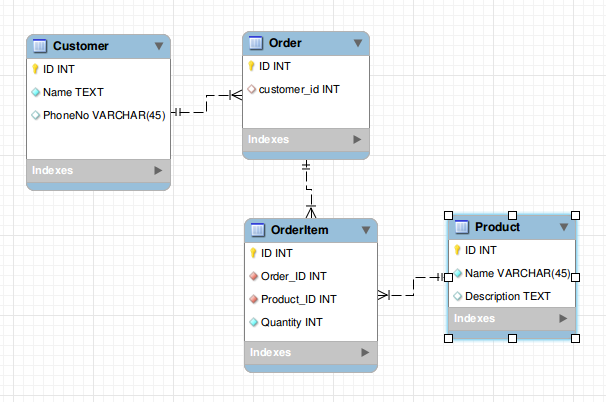
You need four tables, something like this:



Customers

Contains a list of customers. One row per Customer. Would contain all the customers information - their contact details, etc...

Orders

Contains a list of orders. One row per order. Each order is placed by a customer and has a Customer\_ID - which can be used to link back to the customer record. Might also store the delivery address, if different from the customers address from their record - or store addresses in separate tables.

OrderItems

Contains a list of order items. One row for each item on an order - so each Order can generate multiple rows in this table. Each item ordered is a product from your inventory, so each row has a product\_id, which links to the products table.

Products

Contains a list of products. One row per product. Similar to the customers table, but for products - contains all the product details.

Here's the SQL code that you could use to create this structure - it will create a database for itself called mydb:

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`Customer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (

`ID` INT NOT NULL ,

`Name` TEXT NOT NULL ,

`PhoneNo` VARCHAR(45) NULL ,

PRIMARY KEY (`ID`) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Order`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Order` (

`ID` INT NOT NULL ,

`customer\_id` INT NULL ,

PRIMARY KEY (`ID`) ,

INDEX `fk\_Order\_1\_idx` (`customer\_id` ASC) ,

CONSTRAINT `fk\_Order\_1`

FOREIGN KEY (`customer\_id` )

REFERENCES `mydb`.`Customer` (`ID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Product`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Product` (

`ID` INT NOT NULL ,

`Name` VARCHAR(45) NOT NULL ,

`Description` TEXT NULL ,

PRIMARY KEY (`ID`) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`OrderItem`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`OrderItem` (

`ID` INT NOT NULL ,

`Order\_ID` INT NOT NULL ,

`Product\_ID` INT NOT NULL ,

`Quantity` INT NOT NULL ,

PRIMARY KEY (`ID`) ,

INDEX `fk\_OrderItem\_1\_idx` (`Order\_ID` ASC) ,

INDEX `fk\_OrderItem\_2\_idx` (`Product\_ID` ASC) ,

CONSTRAINT `fk\_OrderItem\_1`

FOREIGN KEY (`Order\_ID` )

REFERENCES `mydb`.`Order` (`ID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_OrderItem\_2`

FOREIGN KEY (`Product\_ID` )

REFERENCES `mydb`.`Product` (`ID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

USE `mydb` ;

[share](https://stackoverflow.com/a/17711375)[improve this answer](https://stackoverflow.com/posts/17711375/edit)

[edited Jul 17 '13 at 22:43](https://stackoverflow.com/posts/17711375/revisions)

answered Jul 17 '13 at 22:28

[[https://www.gravatar.com/avatar/ca1ac9c04fbcaae624e80e6aa6c7ec52?s=32&d=identicon&r=PG](https://stackoverflow.com/users/259698/duncan-lock)](https://stackoverflow.com/users/259698/duncan-lock)

[Duncan Lock](https://stackoverflow.com/users/259698/duncan-lock)

**7,124**52538

* Was pretty much typing exactly that:) – [null](https://stackoverflow.com/users/2508465/null) [Jul 17 '13 at 22:29](https://stackoverflow.com/questions/17711324/database-structure-for-customer-table-having-many-orders-per-customer-and-many/17711375#comment25813272_17711375)
* 1

And probably an Items Table. – [Hart CO](https://stackoverflow.com/users/1130801/hart-co) [Jul 17 '13 at 22:30](https://stackoverflow.com/questions/17711324/database-structure-for-customer-table-having-many-orders-per-customer-and-many/17711375#comment25813288_17711375)

* 1

Great answer, not so hard to understand even for a beginner like me. The relationships diagram was pretty useful. Thanks man. – [edferda](https://stackoverflow.com/users/1956586/edferda) [Jul 18 '13 at 0:57](https://stackoverflow.com/questions/17711324/database-structure-for-customer-table-having-many-orders-per-customer-and-many/17711375#comment25815714_17711375)

* 1

@HartCO Wouldn't that be the same thing as the Product table in the answer? – [eabates](https://stackoverflow.com/users/5451315/eabates) [Aug 9 '16 at 13:29](https://stackoverflow.com/questions/17711324/database-structure-for-customer-table-having-many-orders-per-customer-and-many/17711375#comment65068362_17711375)

* 1

@bteague To add a payments table would just be more of the same. A table called payments, with an ID, a customer\_id, and an order\_id (if you want to tie a payment to an order), an amount - and any other information you want to collect about the payment. – [Duncan Lock](https://stackoverflow.com/users/259698/duncan-lock) [Aug 22 at 19:51](https://stackoverflow.com/questions/17711324/database-structure-for-customer-table-having-many-orders-per-customer-and-many/17711375#comment90898755_17711375)

[show **4** more comments](https://stackoverflow.com/questions/17711324/database-structure-for-customer-table-having-many-orders-per-customer-and-many/17711375)

up vote2down vote

There's no sense in creating a table per order. Don't do that. It's not practical, not maintainable. You won't be able to normally query your data. For starters all you need just four tables like this

* customers
* orders
* order\_items
* products (or items)

Here is oversimplified [**SQLFiddle**](http://sqlfiddle.com/#!2/6ed32/3) demo

[share](https://stackoverflow.com/a/17711432)[improve this answer](https://stackoverflow.com/posts/17711432/edit)

[edited Jul 17 '13 at 22:47](https://stackoverflow.com/posts/17711432/revisions)

answered Jul 17 '13 at 22:32

[[https://i.stack.imgur.com/iUA8y.jpg?s=32&g=1](https://stackoverflow.com/users/1920232/peterm)](https://stackoverflow.com/users/1920232/peterm)

[peterm](https://stackoverflow.com/users/1920232/peterm)