# PROJECT DESIGN

## Overview

This documentation provides a complete description of the underlying design of the project. This includes the data model, the tables and constraints, as well as stored functions that are not part of the external API.

## Data model

We aimed to develop data models that can represent the entities and relationships in an order management system. The data model for the project is based on the concept of:

- *products that can be purchased*
- *inventories of products available for purchase*
- *customers who purchase products*
- *orders for products by customers*

Therefore, we built five data models and we created corresponding java class for each model. The following tables will show the field, type and description for each class representing the data model. In addition to the basic fields, we defined a static String called NOT_VALID_ARGUMENT to show if any argument is not valid, this field will not be shown in the following tables.

All the java files related to build models can be found in the **/code/model** directory.

- **Model 1: Product**

  The Product Model represents a product that can be purchased.  It includes the name of the product, a product description, a vendor product SKU (Stock Keeping Unit) that identifies the product

| Fields | Type | Description |
|---|---|---|
| **name** | String | Name of the product |
| **description** | String | Description of the product |
| **sku** | String | Stock Keeping unit that identifies the product |

- **Model 2: InventoryRecord**

  The InventoryRecord Model represents the number of units available for purchase and the price per unit for the current inventory. We keep 2 digits after the decimal place to record the price per unit.

  | Fields | Type | Description |
  |---|---|---|
  | **quantityInStock** | int | number of units available for purchase |
  | **unitPrice** | double | price per unit |
  | **sku** | String | Stock Keeping unit that identifies the product. |

- **Model 3: Customer**

  The Customer Model has the information about the customer, including name, address, city, state, country, postal code. The customer also has a customer id that is a numeric gensym. We made reasonable assumptions about the sizes of the fields, and we chose String to represent state and country instead of enumerated values. We will not include payment information for this model.

  | Fields | Type | Description |
  |---|---|---|
  | **customerId** | int | The customer's Id |
  | **name** | String | The customer's name |
  | **address** | String | The customer's address |
  | **city** | String | The customer's city |
  | **state** | String | The customer's state |
  | **country** | String | The customer's country |
  | **postalCode** | String | The customer's postal code |

- **Model 4: Order**

  This Order Model represents an order for a set of products. It includes a customer ID, an order ID gensym, the order date, and shipment date indicating when the order was shipped. If shipment date is null, the order has not yet shipped. All items must be available in a single transaction to place an order.

  | Fields | Type | Description |
  |---|---|---|
  | **customerId** | int | The customer's Id |
  | **orderId** | int | The order's Id |
  | **orderDate** | Date | The order's date |
  | **shipDate** | Date | The date when the order was shipped |

- **Model 5: OrderRecord**

  This OrderRecord Model keeps the record for an item in the order. It includes the order ID, the number of units, and the unit price. The item must be available, and the inventory is automatically reduced when an order record is created for an order.

  | Fields | Type | Description |
  |--------|------|-------------|
  | **numUnits** | int | The number of units |
  | **orderId** | int | The order's Id |
  | **unitPrice** | double | price per unit |
  | **sku** | String | Stock Keeping unit that identifies the product. |

## Tables and Constraints

Based on the data model we built, we generated five corresponding tables that we will use in our database. You can find java files in the **/code/service** directory that provide services to operate different tables. The following tables show the basic information of each table as well as possible constraints we designed to make our project more accurate.

### Table 1: Product

| Fields | Type | Description | Constraints & other notes |
|--------|------|-------------|---------------------------|
| **name** | varchar(255) | Name of the product | NOT NULL |
| **description** | varchar(255) | Description of the product | NOT NULL |
| **SKU** | varchar(16) | Stock Keeping unit that identifies the product | *PRIMARY KEY<br>SKU is a 12-character value of the form AA-NNNNNN-CC where A is an upper-case letter, N is a digit from 0-9, and C is either a digit or an uppercase letter. For example, "AB-123456-0N". |

### Table 2: InventoryRecord

| Fields | Type | Description | Constraints & other notes |
|--------|------|-------------|---------------------------|
| **QuantityInStock** | int | number of units available for purchase | NOT NULL,<br>QuantityInStock >= 0 |
| **UnitBuyPrice** | decimal(19,2) | price per unit | NOT NULL |
| **ProductSKU** | varchar(16) | Stock Keeping unit that identifies the product. | *PRIMARY KEY<br>*FOREIGN KEY (ProductSKU) references Product(SKU) on delete cascade |

## Table 3: Customer

| Fields | Type | Description | Constraints & other notes |
|---|---|---|---|
| CustomerId | int | The customer's Id | *PRIMARY KEY |
| Name | varchar(16) | The customer's name | NOT NULL |
| Address | varchar(225) | The customer's address | NOT NULL |
| City | varchar(16) | The customer's city | NOT NULL |
| State | varchar(16) | The customer's state | NOT NULL |
| Country | varchar(16) | The customer's country | NOT NULL |
| PostalCode | varchar(32) | The customer's postal code | NOT NULL |

## Table 4: OrderTable

| Fields | Type | Description | Constraints & other notes |
|---|---|---|---|
| CustomerId | int | The customer's Id | *FOREIGN KEY (CustomerId) references Customer (CustomerId) on delete cascade |
| OrderId | int | The order's Id | OrderId > 0 *PRIMARY KEY |
| OrderDate | date | The order's date | NOT NULL |
| ShipmentDate | date | The date when the order was shipped | *If shipment date is null, the order has not yet shipped |

## Table 5: OrderRecord

| Fields | Type | Description | Constraints & other notes |
|---|---|---|---|
| Quantity | int | The number of units | NOT NULL, Quantity >= 0 |
| OrderId | int | The order's Id | *PRIMARY KEY *FOREIGN KEY(OrderId) references OrderTable (OrderId) on delete cascade |
| UnitSellPrice | decimal(19,2) | price per unit | NOT NULL |
| ProductSKU | varchar(16) | Stock Keeping unit that identifies the product. | *PRIMARY KEY *FOREIGN KEY (ProductSKU) references Product (SKU) on delete cascade |

## Stored Functions:

### isSku

This function will test whether the format of sku is valid. The input will be a String represent the product's sku, the function will return a Boolean to indicate whether the input is valid.

A valid sku format should follow the rules:

*SKU is a 12-character value of the form AA-NNNNNN-CC where A is an upper-case letter, N is a digit from 0-9, and C is either a digit or an uppercase letter. For example, "AB-123456-0N".*

```java
public static boolean isSku(String sku) {
    return sku.matches( regex: "([A-Z]{2})-([0-9]{6})-([0-9A-Z]{2})");
}
```