About the tables:

- The **Users** document should contain the user's name, email, role, sizes of upper garments, lower garments and other garments, his shipping addresses and date_created_at.

- The **Orders** document should be able to contain the order's id, the user's id, its status, shipping_address and total_price, date_created_at and date_of_arrival. It should be linked to the **OrderItems** entity which will contain the order_item_id, product_id, user_id and quantity. The Users document will be linked to the **CartItems** entity which will contain cart_item_id**,** cart_item_image, cart_item_price and cart_item_quantity.

- The **Products** document should contain information on each product such as its name, images, color, garment_material, fit, specifications, supply_type and date_created_at. It should be linked to the **ProductVariants** entity that contains the sku, (inseam_length, OR upper_size_number and upper_size_letter OR other_size_number and other_size_letter), current_stock, total_stock and quantity_sold. The **Products** document should also be linked to the **ProductImages** entity that contains the product_image_ids and URIs of the respective product's images. The **Products** document should also be linked to the **ProductReviews** entity that contains the product_review_id, product_review_comment, date_created_at and product_review_rating of all the reviews that exist for that product. The **Products** document should also be linked to the **CartItems** document which will contain cart_item_id**,** cart_item_image, cart_item_price and cart_item_quantity.

- The **Categories** and **Subcategories** entities aren't going to have their own documents as these are single-valued attributes. So, even though they have a 1:N relationship with products (A category / subcategory can be had by N products), they aren't getting their own documents. A color can also be shared by many products. This doesn't mean that it gets its own document.
- The **UpperSizes**, **LowerSizes** and **OtherSizes** entities won't have their own documents since they are easier to store through embedding.

Entity	**Attribute**	**Relationship**
Users	user_id	1:1, A user has 1 user_id
	email	1:1, A user has 1 email.
	password	1:1, A user has 1 password.
	phone_number	1:1, A user has 1 phone_number.
	first_name	1:1. A user has one firstname
	last_name	1:1, A user has has one lastname

	user_role	1:1, A user has one user_role.
	UPPER_SIZE	**1:1**, A user has one upper_size.
	lower_size	1:1, A user has one lower_size.
	others_size	1:1, A user has 1 others_size.
	email_comms_type	1:1, A user has one email_comms_type.
	sms_comms	1:1, A user has one sms_comms.
	SHIPPING_ADDRESS	**1: 0 or N**, A user can have 0 or N shipping addresses. An address belongs to 1 user.
	ORDER	**1:N**, A user can have 1 or N orders. An order belongs to 1 user.
Orders	order_id	1:1. An order has only 1 order_id.
	PRODUCT	**M:N**, An order can have 1 or N products. A product can belong to 0 or M orders.
	order_status	1:1, An order can have one order_status.
	shipping_address	1:1, An order can have 1 shipping address.
Products	product_id	1:1, A product has 1 product_id.
	product_name	1:1, A product has 1 product_name.
	product_type	1:1, A product has 1 product_type.
	UPPER_SIZE	**1: 0 or 1**, A product has 0 or 1 upper_size.
	lower_size	1: 0 or 1, A product has 0 or 1 lower_size.

	inseam_length	1: 0 or 1, A product has 0 or 1 inseam_length.
	product_color	1:1, A product has 1 product_color.
	product_description	1:1, A product has 1 product_description.
	REVIEW	**1:N**, A product has N reviews. Each review belongs to 1 product.
	product_total_stock	1:1, A product has 1 product_total_stock.
	product_current_stock	1:1, A product has 1 product_current_stock.
	product_status	1:1, A product has one product_status.
	IMAGE	**1:N**, A product has N images. An image belongs to 1 product.
	product_category	1:1, A product has 1 product_category.
	product_subcategory	1:1, A product has 1 product_subcategory.
	product_fit	1:1, A product has 1 product_fit.
	product_specifications	1:1, A product has only 1 set of product_specifications.
	product_garment_weight	1:1, A product has only 1 product_garment_weight.
	product_material	1:1, A product has only 1 product_material.
	product_supply_type	1:1, A product has only 1 supply_type.
UpperSizes	upper_size_id	1:1, An upper_size has one upper_size_id.
	upper_size_number	1:1, An upper_size has one upper_size_number.
	upper_size_letter	1:1, An upper_size has one upper_size_letter.

OtherSizes	other_size_id	1:1, An other_size has one other_size_id.
	other_size_number	1:1, An other_size has one other_size_number.
	other_size_letter	1:1, An other_size has one other_size_letter.
ProductImages	image_id	1:1, A product image has only 1 image_id.
	image_uri	1:1, A product_image has only 1 image_uri.
	main_image	1:1, A product_image has 1 main_image.
Categories	*category_id*	*1:1, A category has 1 category_id.*
	category_name	*1:1, A category has 1 category_name.*
	product	*1: 1 or N, A category can have 1 or N products.*
Subcategories	*subcategory_id*	*1:1, A subcategory has 1 subcategory_id.*
	subcategory_name	*1:1, A subcategory has 1 subcategory_name.*
	product	*1:1 or N, A subcategory can have 1 or N products.*
Carts	cart_id	1:1, A cart has 1 cart_id.
	product	**M:N**, A cart has 0 or N products. Each product can belong to M carts.
OrderHistory	order_id	1:1, An order has only 1 order_id.
	product	M:N, An order may have 1 or N products in it. A product may belong to 0 or M orders.
	product_name	1:1, Each product in an order has 1 product_name.
	product_price	1:1, Each product in an order has 1 product_price.

	product_category	1:1, Each product in an order has 1 product_category.
	product_subcategory	1:1, Each product in an order has 1 product_subcategory.
	image_uri	1:1, Each product in an order has 1 image_uri.

The Entity-to-Entity relationships are described below:

Entities	**Relationship**
Users - UpperSizes	1: 0 or 1 A user can have 0 or 1 upper size.
Users - ShippingAddresses	1:0 or N, A user can have 0 or N shipping addresses.
Users - Carts	1:0 or 1, A user can have 0 (empty) or 1 cart.
Users - Orders	1:0 or N, A user can have 0 or N orders.
Orders - Products	M:N, An order can have N products and each product in it can belong to M other orders.
Products - UpperSizes	1: 0 or 1, A product can have 0 or 1 upper size.
Products - Images	1:1 or N, A product can have 1 or N images.
Carts - Products	M:N, A cart can have 1 or N products and each product in it can belong to M other carts.

Relationship	**Questions**	**Answers**
Users - UpperSizes	Would keeping the entities together lead to a simpler data model and simpler code?	Yes -> EMB
	Do the entities have a “has / contains” relationship?	Yes -> EMB
	Does the application query	Yes -> EMB

	the entities contents together?	
	Are the entities updated together?	No -> REF -> Info related to Users and UpperSizes aren't always updated together.
	Do the entities need to be archived together?	Yes -> EMB -> Since they're related.
	Is there high cardinality on the child side of the relationship?	No -> EMB -> Since limited data on the child side.
	Is data duplication too complex to manage and is undesired?	No -> EMB -> Since no data duplication.
	Would the 2 entities take up too much space or too much bandwidth?	No -> EMB -> Since both entities are of manageable size.
	Would the embedded piece grow without bound?	No -> EMB -> Since the embedded piece contains only 2 attributes.
	Are the 2 entities written at different times in a write-heavy workload?	No -> EMB -> Since they're related.
	For the child side of the relationship, can the pieces exist by themselves, without a parent?	Yes -> REF -> Since other entities have relations with UpperSizes.
Score	EMB (09); REF (02)	
Decision	EMB	

Relationship	**Questions**	**Answers**
Users - ShippingAddresses	Would keeping the entities together lead to a simpler data model and simpler code?	Yes -> EMB
	Do the entities have a "has / contains" relationship?	Yes -> EMB
	Does the application query	Yes -> EMB -> Since

	the entities contents together?	user_id and shipping_addresses will be queried together.
	Are the entities updated together?	No -> REF -> Info in Users and ShippingAddresses may not be updated together.
	Do the entities need to be archived together?	Yes -> EMB -> Since they're related.
	Is there a high cardinality on the child side of the relationship?	Yes -> REF -> Since a user can have N number of addresses.
	Is data duplication too complex to manage and is undesired?	Yes -> REF -> Since there may be N number of addresses and N could be large.
	Would the combined size of the entities take up too much memory or too much bandwidth?	Yes -> REF -> Since there may be N number of addresses and N could be large.
	Would the embedded piece grow without bound?	Yes -> REF -> Since there may be N number of addresses and N could be large.
	Would the entities be written at different times in a write-heavy workload?	No -> EMB -> Since they're related.
	For the child side of the relationship, can the pieces exist by themselves, without a parent?	No -> EMB -> Since a shipping address can't exist without a user.
Score	REF (05), EMB (06)	
Decision	(**Score Overridden**) REF (Due to possibility of unbounded array and large RAM & BW needs of each document).	

Relationship	**Questions**	**Answers**
Users - **Carts**	Would keeping the entities together lead to a simpler data model and simpler code?	Yes -> EMB
	Do the entities have a "has / contains" relationship?	Yes -> EMB

	Does the application query the entities contents together?	Yes -> EMB -> This is as each cart belongs to 1 user. So the app has to query a particular cart belonging to a particular user_id.
	Are the entities updated together?	No -> REF
	Do the entities need to be archived together?	Yes -> EMB -> Since they're related.
	Is there a high cardinality on the	