# Tehtävä 2: Verkkokauppa

Tehtäväsi on suunnitella tietokanta, jota voitaisiin käyttää verkkokaupan taustalla. Suunnittele tietokanta niin, että siihen voidaan tallentaa tietosisältö seuraavia toimintoja varten:

1. Verkkokaupassa on tuotteita, joilla on nimi, kuvaus ja hinta.
2. Käyttäjä pystyy etsimään tuotteita ja hänellä on ostoskori, johon voi lisätä ja poistaa tuotteita.
3. Tuotteita voidaan luokitella tuoteryhmiin. Sama tuote voi kuulua useaan tuoteryhmään.
4. Kun tilaus on valmis, käyttäjä pystyy antamaan yhteystietonsa ja pankkikorttinsa tiedot, minkä jälkeen tilaus siirtyy käsittelyyn.
5. Käyttäjä pystyy arvostelemaan tuotteita ja lukemaan muiden käyttäjien antamia arvosteluja.
6. Tuotteista voidaan muodostaa tarjouspaketteja, joissa tietyt tuotteet saa tiettyyn hintaan pakettina.
7. Käyttäjä näkee tuotteen kuvauksen yhteydessä, paljonko tuotetta on jäljellä missäkin verkkokaupan varastossa.
8. Käyttäjä voi saada alennuskoodin, jonka voi syöttää tilauksen yhteydessä. Alennuskoodin voi käyttää vain kerran.
9. Käyttäjä näkee tuotteen yhteydessä esimerkkejä, mitä tuotteita muut tämän tuotteen ostaneen asiakkaat ovat ostaneet.

Tehtäväsi on suunnitella, mitä tauluja tietokannassa on, mitä sarakkeita kussakin taulussa on ja miten taulut viittaavat toisiinsa. Jos yllä oleva kuvaus ei kerro jotain yksityiskohtaa, tee jokin järkevä päätös tämän asian suhteen.

Suunnittele tietokanta luvun 6 periaatteiden mukaisesti ja esitä tietokannan rakenne SQL-skeemana ja tietokantakaaviona. Selosta lisäksi jokaisesta yllä olevasta vaatimuksesta (1–9), mitä tietoa tietokantaan tallennetaan, jotta kyseinen vaatimus saadaan toteutettua.

## SQL schema

```
Lomakkeen alareuna
CREATE TABLE `orders` (
  `id` int PRIMARY KEY,
  `customer_id` int UNIQUE NOT NULL,
  `added_product_id` int,
  `removed_product_id` int,
  `quantity` int DEFAULT 1,
  `price` int,
  `total_payment` int,
  `discount_id` int,
  `order_status` ENUM ('failed', 'accepted', 'in_process', 'shipped',
'received')
);

CREATE TABLE `other_customer_buy` (
  `id` int PRIMARY KEY,
  `order_id` int UNIQUE NOT NULL,
  `product_name` varchar(255),
  `price` int
);

CREATE TABLE `review` (
  `id` int PRIMARY KEY,
  `order_id` int UNIQUE NOT NULL,
  `rate` int,
  `review_body` varchar(255)
);
```

```
CREATE TABLE `search` (
  `id` int PRIMARY KEY,
  `search_name` varchar(255),
  `product_name` varchar(255),
  `description` varchar(255),
  `price` int
);

CREATE TABLE `customers` (
  `id` int PRIMARY KEY,
  `full_name` varchar(255),
  `email` varchar(255) UNIQUE,
  `address` varchar(255),
  `phone` int,
  `card_name` varchar(255),
  `card_number` int,
  `security_code` int
);

CREATE TABLE `products` (
  `id` int PRIMARY KEY,
  `products_name` varchar(255),
  `category_id` int NOT NULL,
  `price` int,
  `description` varchar(255),
  `review_id` int,
  `search_id` int,
  `combo_id` int,
  `stock_id` int,
  `other_customer_buy_id` int NOT NULL,
  `status` ENUM ('out_of_stock', 'in_stock', 'running_low')
);

CREATE TABLE `stocks` (
  `id` int PRIMARY KEY,
  `product_name` varchar(255),
  `quantity` int,
  `product_status` ENUM ('out_of_stock', 'in_stock', 'running_low')
);

CREATE TABLE `combo` (
  `id` int PRIMARY KEY,
  `product_combo_name` varchar(255),
  `quantity` int NOT NULL,
  `product_status` ENUM ('out_of_stock', 'in_stock', 'running_low'),
  `combo_price` int
);

CREATE TABLE `discount` (
  `id` int PRIMARY KEY,
  `discount_code` int,
  `discount_status` ENUM ('expired', 'in_use')
);

CREATE TABLE `categories` (
  `id` int PRIMARY KEY,
  `product_name` varchar(255),
  `cat_name` varchar(255),
  `parent_id` int
);
```

```
ALTER TABLE `orders` ADD FOREIGN KEY (`customer_id`) REFERENCES `customers`
(`id`);

ALTER TABLE `orders` ADD FOREIGN KEY (`discount_id`) REFERENCES `discount`
(`id`);

ALTER TABLE `orders` ADD FOREIGN KEY (`added_product_id`) REFERENCES `products`
(`id`);

ALTER TABLE `orders` ADD FOREIGN KEY (`removed_product_id`) REFERENCES
`products` (`id`);

ALTER TABLE `other_customer_buy` ADD FOREIGN KEY (`order_id`) REFERENCES
`orders` (`id`);

ALTER TABLE `products` ADD FOREIGN KEY (`category_id`) REFERENCES `categories`
(`id`);

ALTER TABLE `products` ADD FOREIGN KEY (`review_id`) REFERENCES `review` (`id`);

ALTER TABLE `products` ADD FOREIGN KEY (`search_id`) REFERENCES `search` (`id`);

ALTER TABLE `products` ADD FOREIGN KEY (`combo_id`) REFERENCES `combo` (`id`);

ALTER TABLE `products` ADD FOREIGN KEY (`stock_id`) REFERENCES `stocks` (`id`);

ALTER TABLE `products` ADD FOREIGN KEY (`other_customer_buy_id`) REFERENCES
`other_customer_buy` (`id`);

ALTER TABLE `categories` ADD FOREIGN KEY (`parent_id`) REFERENCES `categories`
(`id`);

ALTER TABLE `review` ADD FOREIGN KEY (`order_id`) REFERENCES `orders` (`id`);
```

## Database diagram

Link to an online image (eg dbdiagram.io "Share" or observoard.co)

https://dbdiagram.io/d/6043f378fcdcb6230b22e904

## How are requirements 1-9 implemented?

1. Create 'Product' table including 'products name','description' and 'price'. The table also includes id, category_id, review_id, search_id, combo_id, stock_id, other_customer_buy_id, product_status. 'Product_status' will show, if the product is 'in_stock', 'out_of_stock' or 'running_low' (if product items are less than 20)

2. Create 'Search' table including id, search_name, product_name, description and price. The 'id' of 'search' table is connected to 'search_id' of 'Product' table.

Create 'Orders' table (aka. shopping cart) including id, customer_id, added_product_id, removed_product_id, quantity, price, total_payment, discount_id, order_status.

'Customer_id' of 'Orders' is connected to 'id' of 'Customer' table

'Added_product_id', 'removed_product_id' are connected to 'id' of 'Product' table
The 'order_status' will show, if the orders are failed, accepted, in_process, shipped or received.

3. Create 'category' table including id, product_name, cat_name, parent_id. Parent id is connected to category name in the same table. 'id' of 'Category' is connected to 'category_id' of 'Product' table.

4. Create 'Customer' table including id, full_name, email, address, phone, card_name, card_number, security_code in order to store customer detail information. 'id' of 'Customer' table is connected to 'customer_id' of 'Order' table.

5. Create 'Review' table including id, order_id, rate, review_body in order to store review and product rating submitted by other users. 'id' of 'review' table is connected to 'review_id' of 'Product' table. 'id' of 'Orders' table is connected to 'order_id' of 'Review' table

6. Create 'Combo' table including id, product_combo_name, quantity, product_status, combo_price in order to perform available packages at a certain price. 'id' of 'Combo' table is connected to 'combo_id' of 'Product' table.

7. Create 'Stocks' table including id, product_name, quantity, product_status in order to inform customer the amount of available products in storage in the description page. 'id' of 'Stocks' table is connected to 'stock_id' on 'Product' table.

8. Create 'discount' table including id, discount_code, discount_status in order to store the discount offers and codes. The 'id' of 'discount' table is connected to 'discount_id' of 'Orders' table. The 'discount status' will also clarify, if the discount code is still 'in use' or 'expired'. if customer has used the code, it will be automatically 'expired'.

9. Create 'Other_customer_buy' including  id, order_id and product_name in order to show customers what other customers purchased previously. The 'id' of 'other_customer_buy' table is connected to 'other_customer_buy_id' of 'Profuct' table. 'Id' of 'Orders' table is connected to 'order_id' of 'other_customer_buy' table.