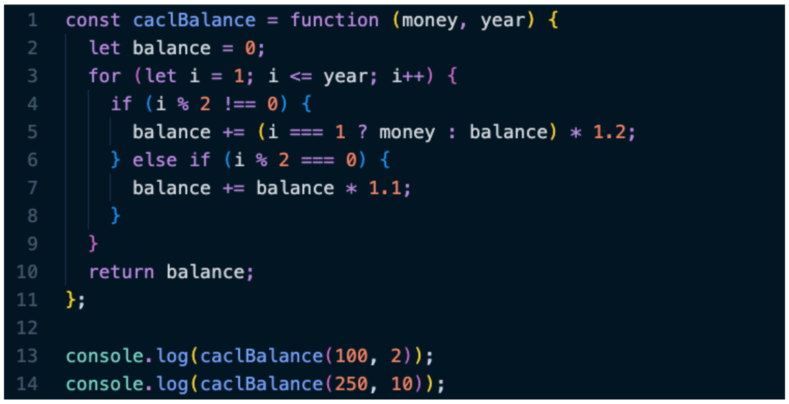
Tran Minh Huy – Backend Technical Assessment

-*link design database*-

Section 1

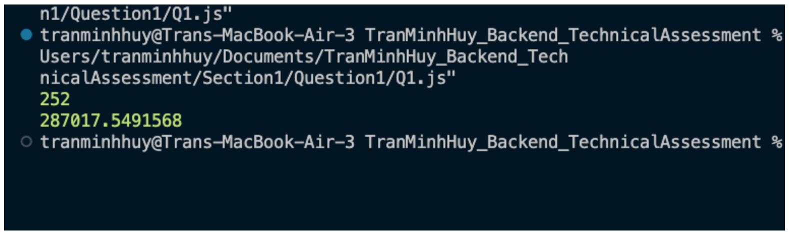
Question 1:

- Source code



```
const caclBalance = function (money, year) {
  let balance = 0;
  for (let i = 1; i <= year; i++) {
    if (i % 2 !== 0) {
      balance += (i === 1 ? money : balance) * 1.2;
    } else if (i % 2 === 0) {
      balance += balance * 1.1;
    }
  }
  return balance;
};

console.log(caclBalance(100, 2));
console.log(caclBalance(250, 10));
```

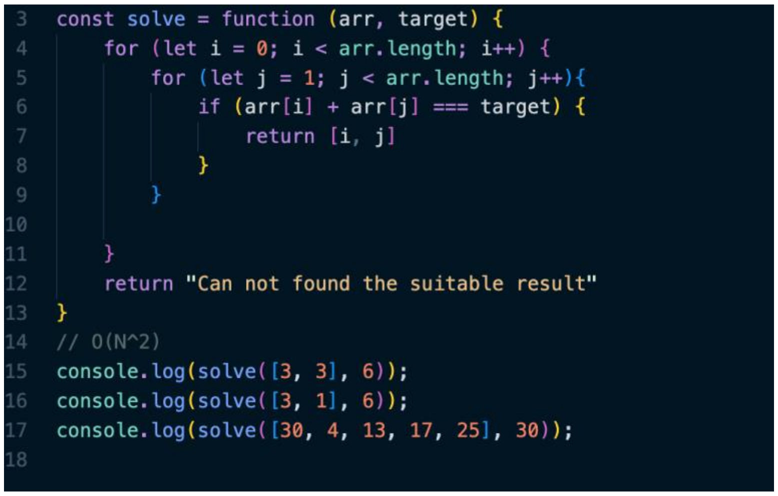
- Result



```
n1/Question1/Q1.js"
● tranminhhuy@Trans-MacBook-Air-3 TranMinhHuy_Backend_TechnicalAssessment %
Users/tranminhhuy/Documents/TranMinhHuy_Backend_Tech
nicalAssessment/Section1/Question1/Q1.js"
252
287017.5491568
○ tranminhhuy@Trans-MacBook-Air-3 TranMinhHuy_Backend_TechnicalAssessment %
```

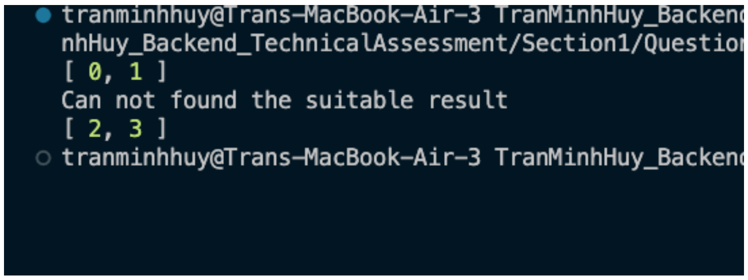
Question 2:

- Source code



```
const solve = function (arr, target) {
    for (let i = 0; i < arr.length; i++) {
        for (let j = 1; j < arr.length; j++){
            if (arr[i] + arr[j] === target) {
                return [i, j]
            }
        }

    }
    return "Can not found the suitable result"
}
// O(N^2)
console.log(solve([3, 3], 6));
console.log(solve([3, 1], 6));
console.log(solve([30, 4, 13, 17, 25], 30));
```

- Result



```
● tranminhhuy@Trans-MacBook-Air-3 TranMinhHuy_Backen
  nhHuy_Backend_TechnicalAssessment/Section1/Questio
  [ 0, 1 ]
  Can not found the suitable result
  [ 2, 3 ]
○ tranminhhuy@Trans-MacBook-Air-3 TranMinhHuy_Backen
```

Question 3:
This is about the web application architecture concepts

1) *DNS*
   a. DNS stands for Domain Name System.
   b. It's a main and backbone technology that make the WWW possible.
   c. At basic level, DNS provides us a key/value pair look up from a domain name to an IP address.
   d. It is necessary to look up the IP address for a domain via DNS.

2) *Load Balancer*
   a. It makes **scaling horizontally** – means you scale by adding more machines into your pool of resources.
   b. Routing incoming requests to one of many application servers that are clone images of each other. Then sending the response from app server back to client.
   c. All of them should process the request the same way, so it is just a matter of distributing the request across the bunch of servers so none of them are overloaded

3) *Web app servers*
   a. They are the backbone behind the whole setup of web applications.
   b. Web app servers handle the actual request sent by the client, process it to receive any incoming data, query the dbs, and return back to the client.
   c. They do their job via communicating with various backend infrastructures such as databases, caching layers, job queues, search services, other microservices, data/ logging queues, so on…
   d. The web servers may be grouped into two main categories, broadly interpreted to include concurrent servers and iterative servers.
      i. Concurrent servers can spawn new instances of themselves to handle a sizeable incoming load.
      ii. Iterative servers are designed to handle requests one by one and consume fewer resources.

4) *Database*
    a. Database is where the web applications store all their useful data.
    b. Databases provide ways of defining your data structures. That includes inserting, finding, updating or deleting, performing computations across the data.
    c. There are two types of databases.
        i. SQL:
            1. Provide a standard way of querying relational data sets
            2. SQL databases store data in tables that are linked together
        ii. No-SQL:
            1. It has emerged to handle the massive amounts of data that can be produced by large scale web application

5) *Caching service*
    a. Caching services help to provide a short yet fast storage for users to load web application data
    b. Applications typically leverage caching services to save the results of expensive computations so that it's possible to retrieve the results from the cache instead of recomputing them the next time they're needed.

6) *Job Queue & Job Server*
    a. Job queues store a list of jobs that need to be run asynchronously.
    b. It consists of two components: a queue of "jobs" that need to be run and one or more job servers (often called "workers") that run the jobs in the queue.
    c. The queue maintains a list of jobs that must be processed, and these jobs go to the servers based on the job schedule and server availability.

7) *Full text search service*
    a. This component allows users to search through an application using keywords.
    b. It is useful in applications such as social media app,… where searching and retrieving are very important.

8) *Service*
    a. Services are independent sections of a web application
    b. Services come in handy in situations where a single app must do a variety of jobs.
    c. Separating these in the form of services with their own life cycles makes coding and maintaining them easier.
    d. Several operational and planned services:
        i. Account service stores user data across all sites
        ii. Content service stores metadata for all of video, audio, image content
        iii. Payment service provides the way for billing customer credit cards
        iv. HTML PDF service accepts HTML and return PDF document

9) *Data "firehose" & Copy of data & Data warehouse*
    a. The app sends data, typically events about user interactions, to the data "firehose" which provides a streaming interface to ingest and process the data.
    b. The final transformed data are saved to cloud storage.
    c. The transformed data is loaded into a data warehouse for analysis.

10) *Cloud storage*
    a. Cloud storage lets us store data with multiple redundancy options, virtually anywhere.
    b. It is easy, simple and scalable way to store, access and share data over the Internet.
    c. It can be used for storing and accessing anything stored on a local file system.
    d. Moreover, we can interact with it via RESTfull API over HTTP.

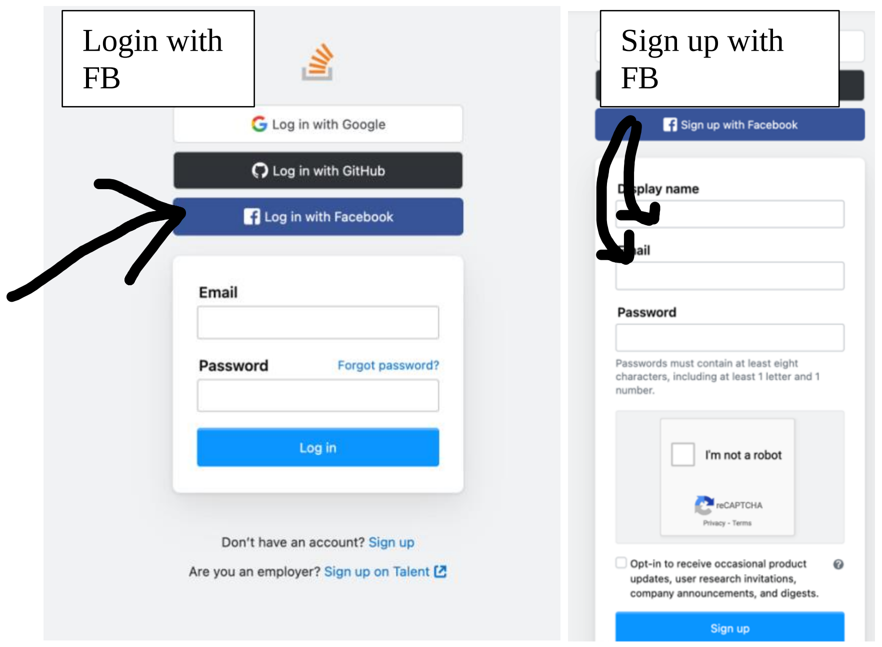
11) *CDN*
    a. CDN stands for Content Delivery Network
    b. CDN systems help distribute content such as HTML files, CSS files, JS files, images, and other types of files to the end-users scattered worldwide.
    c. It works by distributing the content across many "edge" servers around the world so that users end up downloading assets from the "edge" servers instead of the origin server.
    d. It is faster than serving them from a single origin server.

Question 4: Explain Linux command *sudo chmod -R 777 /var/www/html/*
- This means we set access rights to the folder var/www/html (read, write, execute)
- It is for 3 groups: owner, group, others
- Moreover, it means we set full access, and anyone has permission to access everything in var/www/html
- And all of these are the reasons why this command is bad for security reasons.

Question 5: How does OAuth 2.0 work? And give an example of its application
- Using OAuth 2.0, access requests are initiated by the Client, e.g., a mobile app, website, smart TV app, desktop application, etc. The token request, exchange, and response follow this general flow:
-
    o The Client requests authorization (authorization request) from the Authorization server, supplying the client id and secret to as identification; it also provides the scopes and an endpoint URI (redirect URI) to send the Access Token or the Authorization Code to.
    o The Authorization server authenticates the Client and verifies that the requested scopes are permitted.
    o The Resource owner interacts with the Authorization server to grant access.

- The Authorization server redirects back to the Client with either an Authorization Code or Access Token, depending on the grant type, as it will be explained in the next section. A Refresh Token may also be returned.
- With the Access Token, the Client requests access to the resource from the Resource server.

- **Example**:


Login with FB
Sign up with FB
Log in with Google
Log in with GitHub
Log in with Facebook
Email
Password
Forgot password?
Log in
Don't have an account? Sign up
Are you an employer? Sign up on Talent
Sign up with Facebook
Display name
Email
Password
Passwords must contain at least eight characters, including at least 1 letter and 1 number.
I'm not a robot
reCAPTCHA
Privacy - Terms
Opt-in to receive occasional product updates, user research invitations, company announcements, and digests.
Sign up

- OAuth is the underlying technology used for website authentication by sites that let users register or login using their account with another website such as **Facebook, Twitter, LinkedIn, Google, GitHub or Bitbucket**.
- For example, a user **clicks on the Facebook login** option when **logging into another website**, **Facebook authenticates** them, and **the original website logs** them in using permission obtained from Facebook.

Huy's Facebook password	
Main password	abctuihehe123
GitHub	djasd012klasjd0212
Spotify	1923yhadajksdlpqjw
Instagram	fkasjdoiu90123lkjals

$\}\Rightarrow$ access token

- Every generated access token is unique.
- OAuth 2.0 allows to share access between websites or apps.
- **With the sign up**, it is a side effect, website would ask for access to your account with basic permissions such as: name, email, avatar, …. Based on that it would create an account in their system - > you do not need to fill the form to sign up

Question 6: What does the command *git checkout feature_foo_bar* do ?

- We use git checkout <branch-name> to switch branches.
- It does the exact same thing as git switch <branch-name>
- Command git checkout feature_foo_bar means switches us to branch-name: feature_foo_bar
- If the given branch name does not exist, git checkout will create a new branch named feature_foo_bar based on the current branch.

Question 7:

a) Write SQL to find the year in which have most customers were born

```
SELECT EXTRACT(YEAR FROM DOB) as year, COUNT(*) as customer_born
FROM customer
GROUP BY year
ORDER BY customer_born DESC
LIMIT 1;
```

b) Wrire SQL summary all topics with total customers who have the same interest

```
SELECT topics.subject, COUNT(*) as customer
FROM interests
INNER JOIN topics
ON interests.topic_id = topics_id
GROUP BY topic_id;
```

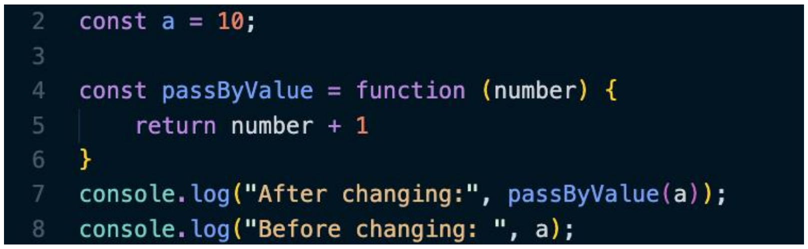
Section 2

Question 1: Describe and give examples of differences between POST and GET methods in the context of HTTP

- GET:
  - Used to retrieve information
  - Data is sent via query string
  - Information is plainly visible in the URL
- POST:
  - Used to post data to the server
  - Used to write/ create/ update
  - Can send any sort of data (JSON)
- Example with comments
  - GET/comments - Get all the comments
  - GET/comments/:id – Get a current comment via comment id
  - POST/comments - Create a new comment
  - POST/comments/:id – Update specific comment on server

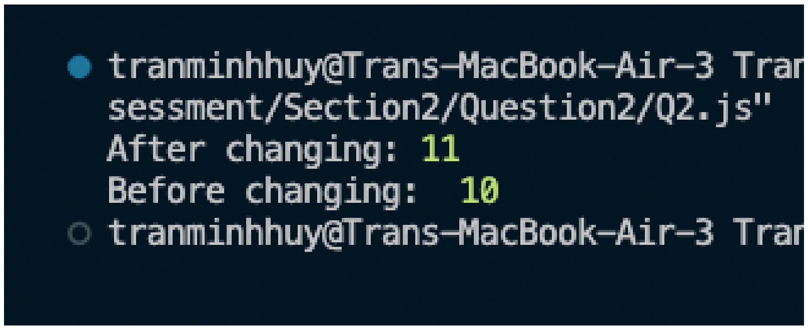
Question 2: Describe and give examples of differences between Pass by Pointer and Pass by Value in programming languages (***JavaScript is the language used to discuss about this topic***)

- **Pass by Value**:
  - Function is called by directly passing the value of the variable as an argument. So any changes made inside the function does not affect the original value. Parameters passed as arguments create its own copy.
  - In JavaScript, pass by value happen with primitive types:
    - String
    - Number
    - Null
    - Undefined
    - bigInt
    - Symbol
    - Boolean
  - The reason for that is primitive value will be stored into call stack. Variable in primitive type will point to address
  - Example
    - Source code:



```
const a = 10;

const passByValue = function (number) {
    return number + 1
}
console.log("After changing:", passByValue(a));
console.log("Before changing: ", a);
```

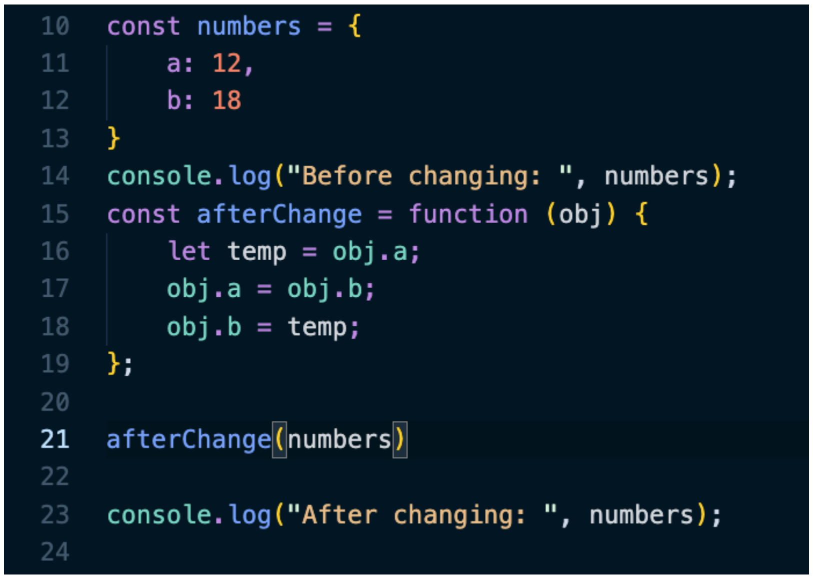
- Result:



```
● tranminhhuy@Trans-MacBook-Air-3 Trar
  sessment/Section2/Question2/Q2.js"
  After changing: 11
  Before changing:  10
○ tranminhhuy@Trans-MacBook-Air-3 Trar
```

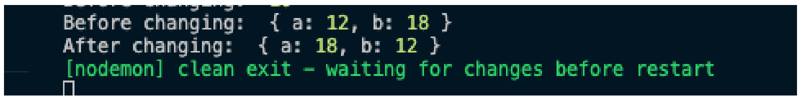
- **Pass by Pointer**:
  - Function is called by directly passing the reference/address of the variable as an argument. Changing the value inside the function also change the original value. Parameters passed as arguments does not create its own copy, it refers to the original value, so changes made inside function affect the original value.
  - In JavaScript, pass by pointer / reference happen with objects:
    - Object literal
    - Array
    - Function

- The reason for that is reference type will be stored into the Heap and the memory value will be pointed to memory address in Heap.
- Example
    - Source code:



```
const numbers = {
    a: 12,
    b: 18
}
console.log("Before changing: ", numbers);
const afterChange = function (obj) {
    let temp = obj.a;
    obj.a = obj.b;
    obj.b = temp;
};

afterChange(numbers)

console.log("After changing: ", numbers);
```

    - Result:



```
Before changing:  { a: 12, b: 18 }
After changing:  { a: 18, b: 12 }
[nodemon] clean exit - waiting for changes before restart
```

Question 3:

a) Write SQL to find 3 employees who have the highest current salaries

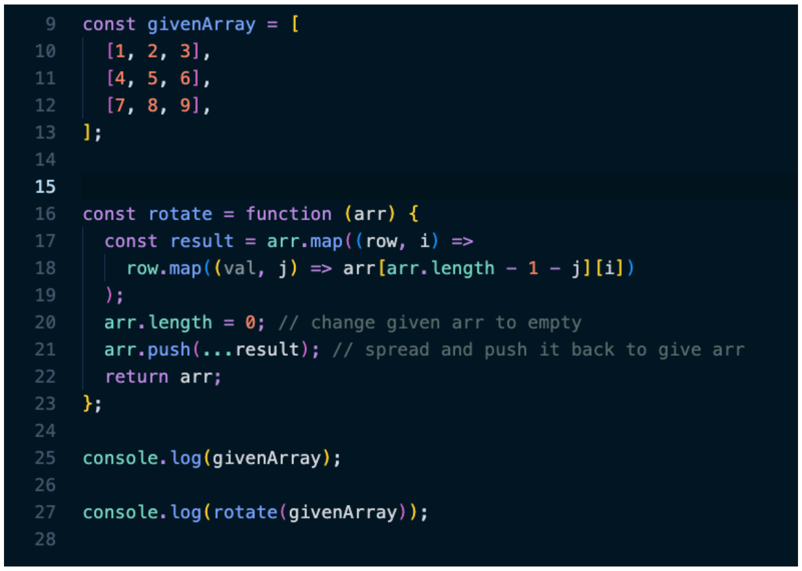
SELECT employees.first_name, salaries.salary
FROM employees
INNER JOIN salaries
ON employees.emp_no = salaries.emp_no
ORDER BY salaries.salary DESC
LIMIT 3;

b) Write SQL to get the department that has no employees in it

SELECT departments.dept_name
FROM departments
LEFT JOIN dept_emp
ON departments.dept_no = dept_emp.dept_no
WHERE dept_emp IS NULL;

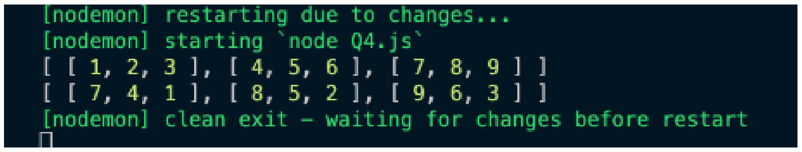
Question 4:

- Source code



```
const givenArray = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9],
];


const rotate = function (arr) {
  const result = arr.map((row, i) =>
    row.map((val, j) => arr[arr.length - 1 - j][i])
  );
  arr.length = 0; // change given arr to empty
  arr.push(...result); // spread and push it back to give arr
  return arr;
};

console.log(givenArray);

console.log(rotate(givenArray));

```

- Result



```
[nodemon] restarting due to changes...
[nodemon] starting `node Q4.js`
[ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ]
[ [ 7, 4, 1 ], [ 8, 5, 2 ], [ 9, 6, 3 ] ]
[nodemon] clean exit - waiting for changes before restart
```

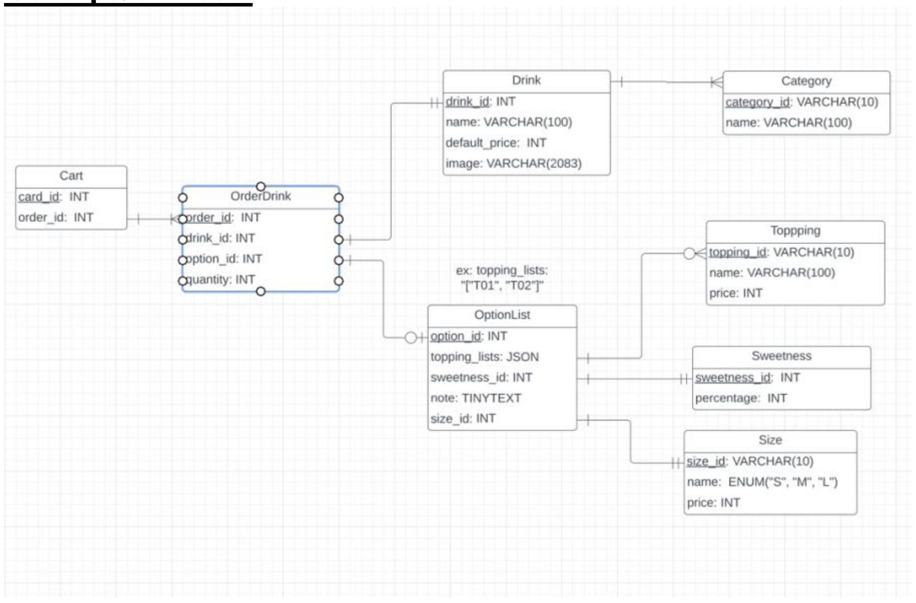
Section 3:

Question 1:

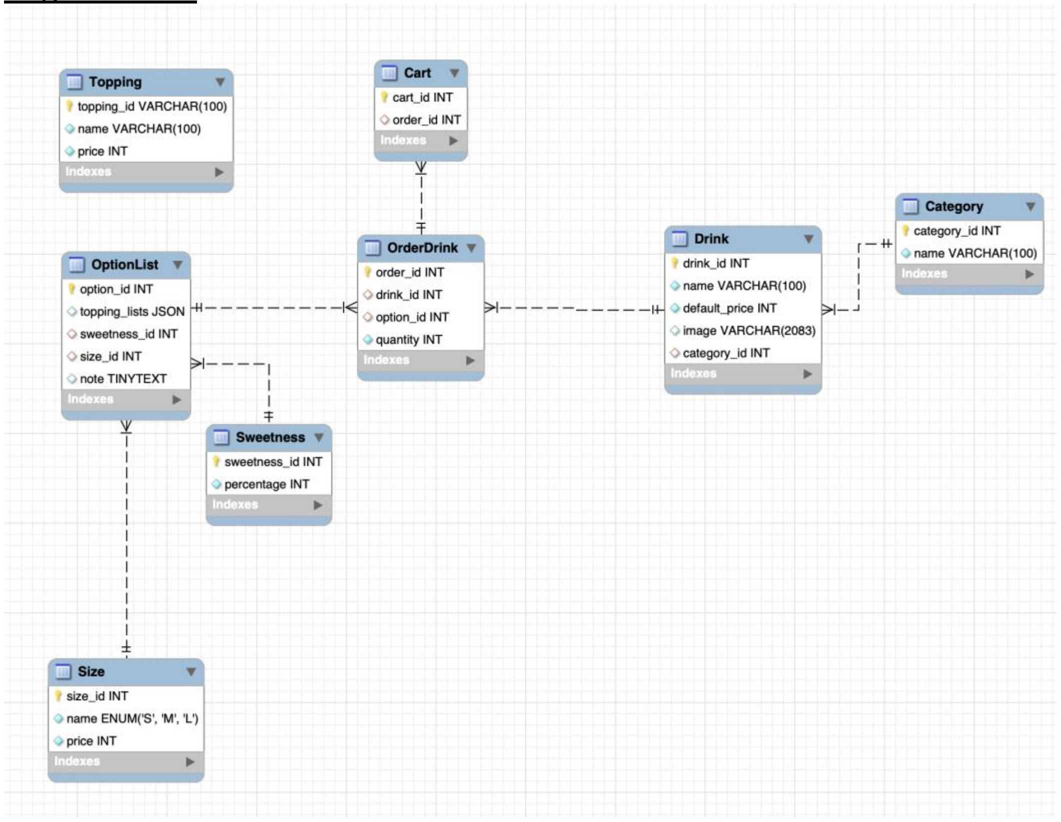
- The cost is spent to buy a computing server with 2vCPU and 2Gb Ram on Digital Ocean will depend on the type of chosen server and the deployed region
- Now, the cheapest option for a server is the standard droplet plan, which starts at $10/month.
- Moreover, you will receive a discount up to 17% when you choose to pay for the whole year. So that its cost will be about $100/year

Question 2:

Conceptual model



Logical model



<u>Physical model</u>

```
CREATE TABLE Category (
	category_id INT AUTO_INCREMENT PRIMARY KEY,
	name VARCHAR(100) NOT NULL
);

CREATE TABLE Drink (
	drink_id INT AUTO_INCREMENT PRIMARY KEY,
	name VARCHAR(100) NOT NULL,
	default_price INT NOT NULL,
	image VARCHAR(2083),
	category_id INT,
	FOREIGN KEY (category_id) REFERENCES Category(category_id)
);

CREATE TABLE Topping(
	topping_id VARCHAR(100) NOT NULL PRIMARY KEY,
	name VARCHAR(100) NOT NULL,
	price INT NOT NULL
);



CREATE TABLE Sweetness (
	sweetness_id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
	percentage INT NOT NULL
);

CREATE TABLE Size(
	size_id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
	 name ENUM("S", "M", "L") NOT NULL,
	 price INT NOT NULL
);

CREATE TABLE OptionList (
	option_id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
	topping_lists JSON,
	sweetness_id INT,
	size_id INT,
		note TINYTEXT,
	FOREIGN KEY (sweetness_id) REFERENCES Sweetness(sweetness_id),
	FOREIGN KEY (size_id) REFERENCES Size(size_id)
);
```

```
CREATE TABLE OrderDrink (
	order_id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
	drink_id INT,
	option_id INT,
	quantity INT NOT NULL,
	FOREIGN KEY (drink_id) REFERENCES Drink(drink_id),
	FOREIGN KEY (option_id) REFERENCES OptionList(option_id)
);

CREATE TABLE Cart(
	cart_id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
	order_id INT,
	FOREIGN KEY (order_id) REFERENCES OrderDrink(order_id)
);
```