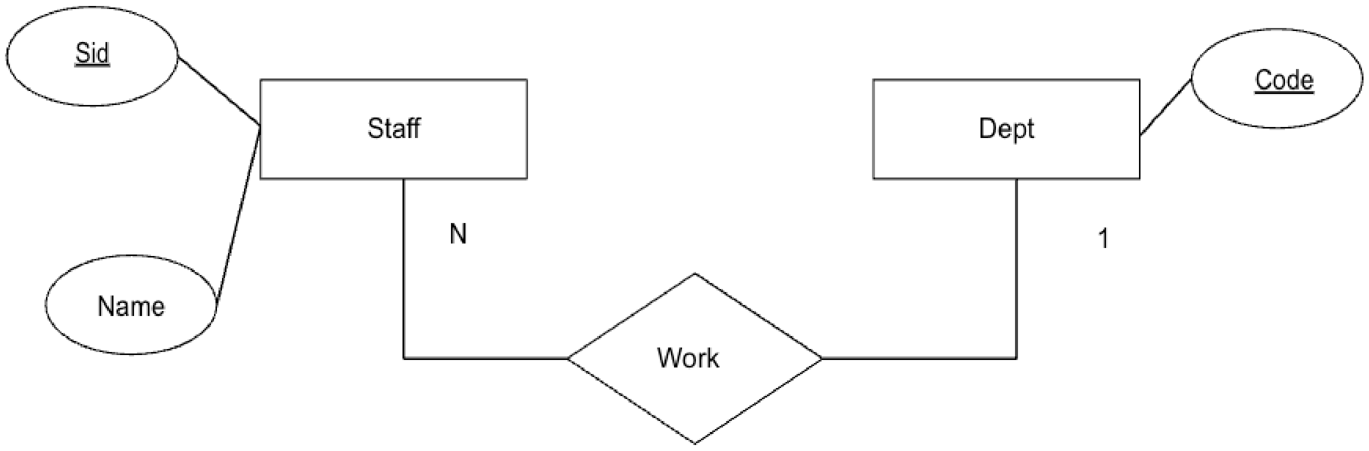
Task 2

Problem Description

Given the ER diagram below,



give the step-by-step explanation on how we can implement the following API in your Express.js webapp which returns the count of staff from each department by using MySQL database:

```
localhost:3000/dept/count
```

yields

```
[
    {
        "count": 2,
        "dept": "HR"
    }
]
```

Solution

The following steps assume that the MongoDB instance set up in the current webapp is no longer valid.

1. Create a new database in MySQL (using client shell) based on the ER diagram.

```
CREATE DATABASE db

CREATE TABLE Dept (
    Code     INT NOT NULL AUTO_INCREMENT,
    PRIMARY KEY (Code)
)

CREATE TABLE Staff (
    Sid      INT NOT NULL AUTO_INCREMENT,
    Name     VARCHAR(255) NOT NULL,
    PRIMARY KEY (Sid),
    FOREIGN KEY (Code) REFERENCES Dept(Code)
)
```

2. Create a new database user in MySQL (using client shell).

```
CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON db.* TO 'user'@'localhost';
FLUSH PRIVILEGES;
```

3. Create a new connection to the MySQL database by replacing models/db.js with the following:

```
import mysql from 'mysql2';

let pool = mysql
  .createPool({
    host: "localhost",
    user: "user",
    database: "db",
    password: "password",
    connectionLimit: 10,
  })
  .promise();

async function cleanup() {
  await pool.end();
}

export { pool, cleanup };
```

If `mysql2` is not installed in the node modules, install it via

```
npm install mysql2
```

4. Perform the query in models/dept.js as follows:

```
import { pool } from './db.js';

// other functions
```

```
async function count() {
    try {
        const [numStaff, deptCode] = await pool.query(
            `SELECT COUNT(*) AS numStaff, Code AS deptCode
            FROM Staff
            GROUP BY deptCode`
        );
        const result = [];
        for (let i of numStaff) {
            result.push({ count: numStaff[i], dept: deptCode[i] });
        }
        return result;
    }
    catch (error) {
        console.error("database connection failed. " + error);
        throw error;
    }
}
```

Note that `COUNT(*)` counts the number of entries. Since the MySQL query returns entries grouped by their department codes, the `COUNT(*)` query serves to count the number of staff members in each department code (and thereby in each department).

5. Finally, create a new API endpoint `/count` in routes/dept.js to call the `count()` function, as follows:

```
/* GET count of staff in each department */
router.get('/count', async function(_req, res, _next) {
    const counts = await count();
    res.json(counts);
});
```

Once the rest of the webapp is configured to use the MySQL database, the new functionality can then be used to produce the required output.