Java Script Task

task 1.1

```
function daysDifferent(d1, d2) {
   return Math.round((d2 - d1) / (1000 * 60 * 60 * 24));
}

const d1 = new Date("2024-07-01");
const d2 = new Date("2024-07-10");
console.log(daysDifferent(d1, d2));
```

task 1.2

```
function getSortedSales(sales) {
   const salesWithTotal = sales.map(sale => ({
      ...sale,
      Total: sale.amount * sale.quantity
   }));

   return salesWithTotal.sort((a, b) => b.Total - a.Total);
}

const sales = [
   { amount: 10000, quantity: 10 },
   { amount: 5000, quantity: 8 },
   { amount: 7000, quantity: 12 }
];

const sortedSales = getSortedSales(sales);
```

```
console.log("Original Sales:", sales);

console.log("Sorted Sales:", sortedSales);
```

task 1.3

```
function projectObject(src, proto) {
    if (typeof src !== "object" || src === null || typeof proto !== "object" || proto === null) {
        return src;
    }

    return Object.keys(proto).reduce((acc, key) => {
        if (key in src) {
            acc[key] = projectObject(src[key], proto[key]);
        }
        return acc;
    }, {});
}

const src = {
    prop11: {
        prop21: 21,
        prop22: {
            prop31: 31,
            prop32: 32
        }
    },
    prop12: 12
};
```

```
const proto = {
   prop11: {
      prop22: null
   }
};

const res = projectObject(src, proto);

console.log(res);
```

Rest Api Task

```
const { google } = require("googleapis");

async function getBusyIntervals(calendarId, startTime, endTime, apiKey) {
   const calendar = google.calendar({ version: "v3", auth: apiKey });

   const response = await calendar.freebusy.query({
      requestBody: {
         timeMin: startTime,
         timeMax: endTime,
         items: [{ id: calendarId }]
      }
   });

   return response.data.calendars[calendarId].busy;
}
```

```
const calendarId = "kavihansi98@gmail.com";
const apiKey = "AIzaSyDi9sEf1Bpo3QFkaq80f59uzOL6r5kfOGA";
const startTime = new Date().toISOString();
const endTime = new Date(Date.now() + 7 * 24 * 60 * 60 * 1000).toISOString();

getBusyIntervals(calendarId, startTime, endTime, apiKey)
  .then(busyIntervals => console.log("Busy Intervals:", busyIntervals))
  .catch(error => console.error("Error:", error));
```

SQL Task

```
CREATE TABLE user (
  id INT PRIMARY KEY,
  firstName VARCHAR(255),
  lastName VARCHAR(255),
  email VARCHAR(255),
  cultureID INT,
  deleted BIT,
  country VARCHAR(255),
  isRevokeAccess BIT,
  created DATETIME
);

CREATE TABLE `group` (
  id INT PRIMARY KEY,
  name VARCHAR(255),
```

```
    created DATETIME
);

CREATE TABLE groupMembership (
    id INT PRIMARY KEY,
    userID INT,
    groupID INT,
    created DATETIME,
    FOREIGN KEY (userID) REFERENCES user(id),
    FOREIGN KEY (groupID) REFERENCES `group`(id)
);

INSERT INTO user (id, firstName, lastName, email, cultureID, deleted, country, isRevokeAccess, created)
VALUES
(1, 'Victor', 'Shevchenko', 'vs@gmail.com', 1033, 1, 'US', 0, '2011-04-05'),
(2, 'Oleksandr', 'Petrenko', 'op@gmail.com', 1034, 0, 'UA', 0, '2014-05-01'),
(3, 'Victor', 'Tarasenko', 'vt@gmail.com', 1033, 1, 'US', 1, '2015-07-03'),
(4, 'Sergiy', 'Ivanenko', 'sergiy@gmail.com', 1046, 0, 'UA', 1, '2010-02-02'),
(5, 'Vitalii', 'Danilchenko', 'shumko@gmail.com', 1031, 0, 'UA', 1, '2014-05-01'),
(6, 'Joe', 'Dou', 'joe@gmail.com', 1032, 0, 'US', 1, '2009-01-01'),
(7, 'Marko', 'Polo', 'marko@gmail.com', 1033, 1, 'UA', 1, '2015-07-03');

INSERT INTO `group` (id, name, created) VALUES
(10, 'Support', '2010-02-02'),
(12, 'Dev team', '2010-02-03'),
(13, 'Apps team', '2011-05-06'),
```

```
(14, 'TEST - dev team', '2013-05-06'),

(15, 'Guest', '2014-02-02'),

(16, 'TEST-QA-team', '2014-02-02'),

(17, 'TEST-team', '2011-01-07');


INSERT INTO groupMembership (id, userID, groupID, created) VALUES

(110, 2, 10, '2010-02-02'),

(112, 3, 15, '2010-02-03'),

(114, 1, 10, '2014-02-02'),

(115, 1, 17, '2011-05-02'),

(117, 4, 12, '2014-07-13'),

(120, 5, 15, '2014-06-15');
```

3.2. Select names of all empty test groups (group name starts with “TEST-”).

```
SELECT name

FROM "group"

WHERE name LIKE 'TEST-%'

AND id NOT IN (SELECT groupID FROM groupMembership);
```

3.3. Select user first names and last names for the users that have Victor as a first name and are not members of any test groups (they may be members of other groups or have no membership in any groups at all).

```
SELECT firstName, lastName

FROM "user"

WHERE firstName = 'Victor'

AND id NOT IN (
```

```
    SELECT gm.userID
    FROM groupMembership gm
    JOIN "group" g ON gm.groupID = g.id
    WHERE g.name LIKE 'TEST-%'
);
```

3.4. Select users and groups for which user was created before the group for which he(she) is member of.

```
SELECT u.firstName, u.lastName, g.name AS groupName
FROM "user" u
JOIN groupMembership gm ON u.id = gm.userID
JOIN "group" g ON gm.groupID = g.id
WHERE u.created < g.created;
```