## Задача №1

Имеется база со следующими таблицами:

CREATE TABLE `users` (

    `id`         INT(11) NOT NULL AUTO\_INCREMENT,

    `name`       VARCHAR(255) DEFAULT NULL,

    `gender`     INT(11) NOT NULL COMMENT '0 - не указан, 1 - мужчина, 2 - женщина.',

    `birth\_date` INT(11) NOT NULL COMMENT 'Дата в unixtime.',

    PRIMARY KEY (`id`)

);

CREATE TABLE `phone\_numbers` (

    `id`      INT(11) NOT NULL AUTO\_INCREMENT,

    `user\_id` INT(11) NOT NULL,

    `phone`   VARCHAR(255) DEFAULT NULL,

    PRIMARY KEY (`id`)

);

Напишите запрос, возвращающий имя и число указанных телефонных номеров девушек в возрасте от 18 до 22 лет.  
Оптимизируйте таблицы и запрос при необходимости.

## Ответ по задаче №1

Запрос, возвращающий имя и число указанных телефонных номеров девушек в возрасте от 18 до 22 лет:

SELECT u.name, count(p.id) as coutn\_phones

FROM `users` as u

LEFT JOIN phone\_numbers as p on u.id = p.user\_id

WHERE u.gender = 2 AND u.birth\_date BETWEEN UNIX\_TIMESTAMP((NOW() - INTERVAL 22 YEAR)) AND UNIX\_TIMESTAMP((NOW() - INTERVAL 18 YEAR))

GROUP BY p.user\_id

\* если пользователь не сохранял телефоны, запрос покажет, что их 0.

Для оптимизации таблиц, я бы убрал стандартное значение поля «**phone**», в таблице «**phone\_numbers**». При указании пустого телефона, создавать новую запись не имеет смысла, а в запросе будут учитываться пустые значения если их не исключить.

|  |  |
| --- | --- |
| Было | CREATE TABLE `phone\_numbers` (      `id`      INT(11) NOT NULL AUTO\_INCREMENT,      `user\_id` INT(11) NOT NULL,      `phone`   VARCHAR(255) DEFAULT NULL,      PRIMARY KEY (`id`)  ); |
| Стало | CREATE TABLE `phone\_numbers` (      `id`      INT(11) NOT NULL AUTO\_INCREMENT,      `user\_id` INT(11) NOT NULL,      `phone`   VARCHAR(255) NOT NULL,      PRIMARY KEY (`id`)  ); |

Тоже самое со стандартным пустым значением поля «**name**» в таблице «**users**». Пользователи могут не задавать имени, но при запросе получается, что неизвестно кто имеет сколько-то номеров. В таком случае лучше делать статистику не по имени, а по id пользователя.

## Задача №2

Проведите рефакторинг, исправьте баги и продокументируйте в стиле PHPDoc код, приведённый ниже (таблица users здесь аналогична таблице users из задачи №1).  
Примечание: код написан исключительно в тестовых целях, это не "жизненный пример" :)

function load\_users\_data($user\_ids) {

    $user\_ids = explode(',', $user\_ids);

    foreach ($user\_ids as $user\_id) {

        $db = mysqli\_connect("localhost", "root", "123123", "database");

        $sql = mysqli\_query($db, "SELECT \* FROM users WHERE id=$user\_id");

        while($obj = $sql->fetch\_object()){

            $data[$user\_id] = $obj->name;

        }

        mysqli\_close($db);

    }

    return $data;

}

// Как правило, в $\_GET['user\_ids'] должна приходить строка

// с номерами пользователей через запятую, например: 1,2,17,48

$data = load\_users\_data($\_GET['user\_ids']);

foreach ($data as $user\_id=>$name) {

    echo "<a href=\"/show\_user.php?id=$user\_id\">$name</a>";

}

Плюсом будет, если укажете, какие именно уязвимости присутствуют в исходном варианте (если таковые, на ваш взгляд, имеются), и приведёте примеры их проявления.

## Ответ по задаче №2

## Новый код для второй задачи:

/\*\*

         \* проверка на приходящие данные

         \* Как правило все должно быть хорошо, но лучше проверить, особенно если ввод проиходит вручную

         \* @param string id пользователей,  формата "1,2,17,48"

         \* @return string проверенные и только числовые id пользователей,  формата "1,2,17,48"

         \*/

        function validate (string $string\_users\_id) : string

        {

            $array\_users\_id = explode(',', $string\_users\_id);

            $verified\_ids = [];

            foreach($array\_users\_id as $user\_id){

                if(is\_int((int) ($user\_id)) && $user\_id > 0){

                    $verified\_ids[] = $user\_id;

                }

            }

            return implode(',', $verified\_ids);

        }

        /\*\*

         \* запрос для данных пользователя

         \*

         \* @param string id пользователей,  формата "1,2,17,48"

         \* @return array массив пользователей формата user\_id => user\_name

         \*/

        function load\_users\_data (string $user\_ids) : array

        {

            if(!$user\_ids){

                return [];

            }

            $db = mysqli\_connect(("localhost", "root", "123123", "database");

            $sql = mysqli\_query($db, "SELECT id, name FROM users WHERE id IN ($user\_ids)");

            $data = [];

            while($obj = $sql->fetch\_object()){

                $data[$obj->id] = $obj->name;

            }

            mysqli\_close($db);

            return $data;

        }

        /\*\*если данные переданы совсем не так, ничего не делаем \*/

        if(!is\_string($\_GET['user\_ids'])){

           return;

        }

        $verified\_ids = validate($\_GET['user\_ids']);

        $data = load\_users\_data($verified\_ids);

        foreach ($data as $user\_id=>$name) {

            /\*\* таблица создана так, что имени может и не быть\*/

            if(!$name){

                $name = "Неопознанный субъект №$user\_id";

            }

            echo "<a href=\"/show\_user.php?id=$user\_id\">$name</a>";

        }

В процессе рефакторинга были добавлены несколько проверок на приходящие данные, а также изменен запрос для базы данный. Запрос теперь уходит один, возвращая параметры для всех искомых пользователей. Помимо одного запроса, решена проблема запроса повторяющихся пользователей. Исходный код мог обрабатывать повторно один и тот же id, хотя записывает он их в один массив.

Подводя итог, в исходном варианте можно отметить несколько вещей:

1. Открытые данные для доступа к базе данных. Это не только уязвимость, но и повторяющийся код. Следует вынести адрес, логин и пароль, куда-нибудь за пределы кода, и ограничить их доступ.
2. Отсутствие пагинации для вывода данных. Ну тут нужно работать с самой страницей, в запросе можно добавить только лимит для искомых id.
3. Отсутствие проверки прав для запроса. Вероятно, этот код мог бы использоваться для администраторов или менеджеров сайта, соответственно нужно проверять пользователя сессии на доступ к этому функционалу.
4. Отсутствие валидации приходящих данных. Хотя по условию задачи указано, что входная строка должна иметь перечень номеров пользователей, разделенных запятой, не очень понятно будет ли так на самом деле.