Задание 1

Ссылка на страницу, которую необходимо протестировать:
https://widgets.inssmart.ru/contract/mortgage/?appId=226ff66a-3108-5157-9d69-997b59f77bb1&secret=39919a81-fc16-5afd-9ff1-d0f6c9602608

Задача:

1. Составьте перечень проверок (чек лист) длывя первой страницы, который по Вашему мнению будет достаточным для выпуска продукта в продакшен.
2. Сгруппируйте проверки в форме иерархического списка. В этом вам поможет https://checkvist.com/ (должен получиться полноценный чек лист, с понятной группировкой. Документ должен соответствовать вашему представлению о том, что можно использовать в реальной работе, не забывайте про ожидаемый результат)
3. Укажите приоритеты проверок.
4. Откройте доступ к списку (кнопка Share)
5. Ссылку на список отправьте на почту

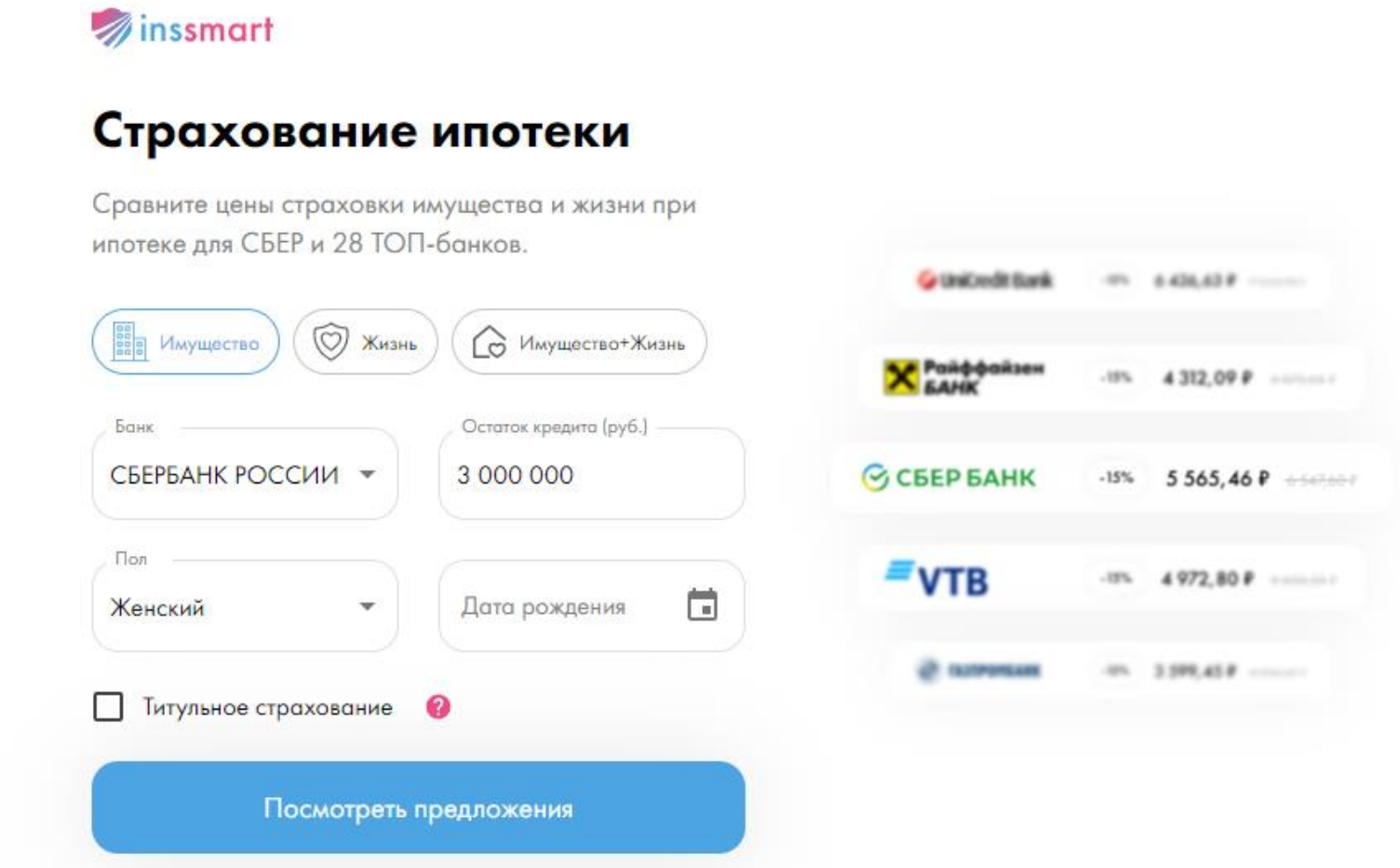
Условия:

Проверку необходимо сделать только для первого экрана, экран с предложениями и последующие тестировать не нужно.

Критерии оценки по приоритетам:

1. Структура и приоритезация проверок.
2. Полнота проверок
3. Правильно расставленные приоритеты

ГОТОВОЕ РЕШЕНИЕ: https://checkvist.com/p/ksgCkxgH1kNwpum6Jn4KuB



Страница с выбором условий страхования ипотеки */ read-only*

Шапка экрана

Логика

- Иконка и текст некликабельны, при наведении стрелка курсора не меняется на палец

Верстка

- Состоит из иконки и текста "inssmart", распложенных в одну линию друг за другом (иконка, а затем текст)
- Иконка в виде щита розово-голубого цвета с тремя белыми полосками

- Текст "inssmart" выделен жирным шрифтор и состоит из двух цветов: "ins" - голубого, "smart" - розового

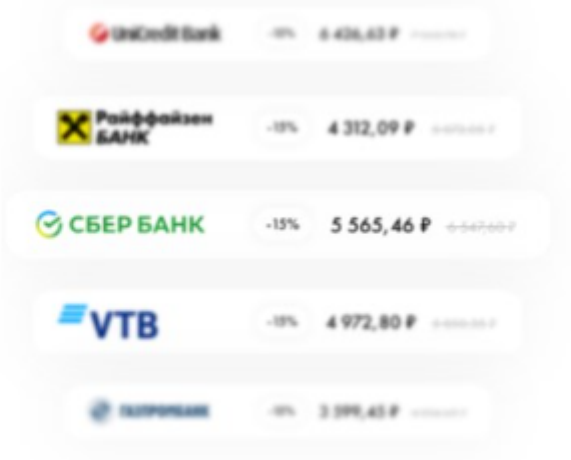
Основная часть экрана

- Основная часть экрана состоит из двух блоков, расположенных в двух колонках выровненных по центру страницы.
- Левый блок экрана содержит условия страхования ипотеки, а правый блок - картинку с названиями и процентными ставками банков

Правый блок - картинка

Верстка

- Картинка в правом блоке содержит названия пяти банков в пять строчек и их процентные ставки напротив названий, средняя строчка с названием банка четкая, остальные размываются к приближению к верхнему и нижнему краям



- Названия банков на картинке сверху вниз: "Unicredit Bank", "Раффайзен Банк", "Сбербанк", "VTB", "Газпромбанк"

- **Левый блок - выбор условий страхования**
  - **1. Название блока**
    - **Верстка**
      - Название некликабельно, при наведении стрелка мышки не меняется на палец
      - Название левого блока - "Страхование ипотеки", шрифт черного цвета, выделен жирным, текст выровнен по левому краю блока
      - Шрифт текста названия: 'Futura PT', serif, 32 px
  - **2. Надпись под названием блока**
    - **Верстка**
      - Надпись некликабельна, при наведении стрелка мышки не меняется на палец
      - Надпись расположена под названием блока и содержит текст: "Сравните цены страховки имущества и жизни при ипотеке для СБЕР и 28 ТОП-банков."
      - Текст надписи меньше по размеру, чем текст названия, серого цвета и не выделен жирным
      - Шрифт текста надписи: Futura PT Book, 18 px
  - **3. Кнопки**
    - **Логика**
      - Кнопки кликабельны, при наведении появляется палец
      - Активной можно сделать только одну кнопку
    - **Верстка**
      - Под надписью расположены три кнопки в один ряд, выровнены по ширине блока
      - Кнопки белого цвета, контур кнопок полукруглый и выделен черной линией
      - Кнопка с названием **"Имущество"** расположена слева, внутри кнопки по центру выровнены иконка с пиктограммой дома и название кнопки

      - Кнопка с названием **"Жизнь"** расположена посередине, внутри кнопки по центру выровнены иконка с пиктограммой сердца в щите и название кнопки

      - При наведении на неактивную кнопку она меняет цвет фона на светло-серый
      - Кнопка с названием **"Имущество+Жизнь"** расположена слева, внутри кнопки по центру выровнены иконка с пиктограммой сердца в доме с треугольной крышей и название кнопки

- Шрифт текста кнопок: Futura PT Book, 13 px
- Кнопки одной высоты, разнозначны по стилю оформления
- При наведении на активную кнопку она меняет цвет фона на светло-голубой
- В активном состоянии контур кнопки, иконка и текст внутри синего цвета

4. Поля для заполнения

Верстка для всех полей

- Состоят из четырех полей, расположенных в две строки, по два поля на каждую строку, выровнены по центру левого блока экрана
- Каждое поле имеет форму прямоугольника с закругленными углами, в неактивном состоянии выледены серой линией, высота и ширина полей равнозначны
- При наведении и в активном состоянии контур полей становится синего цвета
- Шрифт текста полей: Futura PT Book, 16 px
- В выбранном состоянии надписи на полях становятся синего цвета
- Раскрывающиеся списки в полях "Банк" и "Пол" одинаковые по цвету поля и шрифта, а также по размеру
- Раскрывающийся список пропадает при клике на значение из него или при клике на лубую другую часть поля
- Текст значений в раскрывающихся списках в полей "Банк" и "Пол" черного цвета, при наведении значение подсвечивается светло-серым цветом, в выбранном состоянии значение подсвечивается темно-серым цветом
- Размер и цвет шрифта в названии полей одного шрифта и цвета - черного, текст выровнен по левому краю поля
- Размер и цвет шрифта у надписей на полях одного шрифта и цвета - серого, текст выровнен по левому краю поля и его обрамляет серая линия выделения поля

1. Поле "Банк"

Логика

- Поле кликабельное, при наведении появляется палец
- При клике на любую часть поля появляется раскрывающийся список с названиями банков
- Можно выбрать только один банк
- Можно ввести первую букву банка, после чего подсветится название банка, содержащее эту букву в качестве первой в названии
- Выбирать значения можно как мышкой, так и кнопками на клавиатуре

Верстка

- Поле имеет надпись "Банк" в верхнем левом углу, шрифт меньше шрифта текста самого поля
- Значение в раскрывающемся списке не отсортированы в алфавитном порядке
- Первые пять названий банков в раскрывающемся списке совпадают с названиями пяти банков на картинке в правом блоке

2. Поле "Остаток Кредита"

Логика

- Поле кликабельное, при наведении появляется курсор
- Поле принимает значение из 8 цифр: 1000000
- Поле принимает значение из 7 цифр: 100000
- Поле принимает значение из 6 цифр: 10000
- При вводе 9 цифр (100000000) поле сохраняет значение из первых 8 цифр, а следующие вводить не разрешает
- Значения проставляются в поле в формате nn nnn nnn
- При вводе нуля (0) и 5 цифр (99999) появляется сообщение об ошибке: "Значение поля "Остаток долга по ипотеке" не должно быть меньше 100000""
- Поле принимает значение "100001"
- В случае отсутствия заполнения поля появляется сообщение об ошибке

Верстка

- Поле имеет надпись "Остаток кредита (руб.)" в верхнем левом углу, шрифт меньше шрифта текста самого поля
- Название ошибки в случае **пустого поля**: "Поле "Остаток долга по ипотеке" обязательно для заполнения", текст красного цвета, расположен пол рамкой поля
- При появлении сообщения об ошибке в случае пустого поля надпись над полем перемещается внутрь поля по центру и становится красного цвета, как и линия вокруг поля
- Название ошибки в случае ввода **некорректного значения**: "Значение поля "Остаток долга по ипотеке" не должно быть меньше 100000", текст красного цвета, расположен под рамкой поля
- При появлении сообщения об ошибке в случае ввода некорректного значения надпись над полем остается слева над полем и становится красного цвета, как и линия вокруг поля

3. Поле "Пол"

Логика

- Поле кликабельное, при наведении появляется палец
- При клике на любую часть поля появляется раскрывающийся список с выбором пола, состоящий из двух вариантов
- Можно выбрать только один из имеющихся вариантов

Верстка

- Поле имеет надпись "Пол" в верхнем левом углу, шрифт меньше шрифта текста самого поля
- В раскрывающемся списке два варианта выбора с названиями: "Мужской", "Женский"

- **4. Поле "Дата рождения"**
  - **Логика**
    - Поле кликабельное, при наведении появляется курсор, при клике появляется выпадающий календарь
    - В выпадающем календаре можно выбрать дату рождения, равную возрасту = ровно 18 лет от текущей даты
    - В выпадающем календаре можно выбрать дату рождения, равную возрасту = 18 лет + 1 день от текущей даты
    - При выборе в выпадающем календаре нельзя выбрать дату рождения = 18 лет - 1 день от текущей даты появится сообщение об ошибке
    - В выпадающем календаре можно выбрать дату рождения = 75 лет от текущей даты
    - В выпадающем календаре нельзя выбрать дату рождения = 75 лет + 1 день от текущей даты
    - Поле нельзя оставить пустым, появится сообщение об ошибке
  - **Верстка**
    - Поле имеет надпись "Дата рождения" в верхнем левом углу, шрифт меньше шрифта текста самого поля
    - При выборе **некорректной даты рождения** появляется сообщение об ошибке с текстом: "Возраст страхователя должен быть больше или равен 18 годам", текст красного цвета, расположен под рамкой поля
    - Название ошибки в случае **пустого поля**: "Поле "Дата рождения страхователя" обязательно для заполнения", текст красного цвета, расположен пол рамкой поля
    - При появлении сообщения об ошибке в случае пустого поля надпись над полем перемещается внутрь поля по центру и становится красного цвета, как и линия вокруг поля
- **5. Чекбокс "Титульное страхование"**
  - **Верстка**
    - Чекбокс кликабельный, при наведении появляется палец, можно поставить галочку на чекбокс
    - Чекбокс расположен под полями "Пол" и "Дата рождения", выровнен по левому краю блока
    - В неактивном состоянии чекбокс представляет собой белый квадрат с жирной черной рамкой
    - В выбранном состоянии чекбокс представляет собой синий квадрат без рамки с белой галочкой внутри
- **6. Кнопка знака вопроса**
  - **Верстка**
    - Кликабельна, при клике появляется всплывающая подсказка
    - При наведении появляется палец и контур вокруг кнопки подсвечивается розовым вертикальным овалом
    - После клика появляется текст подсказки, расположенный в белом всплывающем окне прямоугольной формы с голубым краем с левой стороны
    - Текст всплывающей подсказки: "Титульное стразование защищает собственника недвижимого имущества от признания сделки по приобретению недвижимости недействительной или требований бывшего собственника о возврате имущества, выбывшего у него помимо его воли. Включите чекбокс, если ваш банк требует данный вид стразования".
    - Клик по любой зоне внутри или вне всплывающей подсказки закрывает ее

- **7. Кнопка "Посмотреть предложения"**
  - **Логика**
    - При клике на кнопку происходит переход на страницу выбора предложения
  - **Верстка**
    - Кликабельна, при наведении появляется палец
    - Кнопка прямоугольной форма синего цвета с загругленными углами, выровнена по центру левого блока
    - Текст на кнопке белого цвета: "Посмотреть предложения", выровнен по центру кнопки
    - Шрифт текста кнопки: Futura PT Book, 18 px
    - При наведении на кнопку ее цвет становится темно-синим, а вокруг нее подсвечивается серая тень

Задание 2

Коллекция в swagger, с которой предстоит работать: https://petstore.swagger.io/

Вам необходимо на основе этой коллекции, создать коллекцию в Postman

Нужно будет работать с запросами:

1. Создать нового питомца (**метод POST /pet**)
Прописать скрипт, который берет из тела ответа id созданного питомца и записывает в окружение.

Подсказка: скрипт прописывается во вкладке «tests», для записи переменной в окружение можно использовать pm.environment
Если Вы не можете решить проблему с записью id в окружения целиком - можете задать id изначально при создании питомца

2. Запросить питомца по id (**метод GET /pet/{petId}**) используя в запросе ранее записанную переменную из окружения
Прописать скрипт, который будет брать тело ответа, менять параметр «status» на «sold» (продан) и перенаправлять его в запрос на изменение питомца (**метод PUT /pet**)

Подсказка, скрипт прописывается во вкладке «tests», для перенаправления запроса можно использовать pm.sendRequest.

3. Коллекцию необходимо экспортировать (Export/ Export Collection) сюда: Petstore.postman_collection.json

В итоге должно быть два запроса, при отправке которых должен создаваться новый питомец и он же продаваться

```
    "info": {
        "_postman_id": "6de77aca-7ccd-464f-b5d3-ae74c14bbb68",
        "name": "Petstore",
        "schema": "https://schema.getpostman.com/json/collection/v2.0.0/collection.json",
        "_exporter_id": "22183380"
    },
    "item": [
        {
            "name": "Добавление нового питомца",
            /*
            Создать нового питомца (метод POST /pet)
            Прописать скрипт, который берет из тела ответа id созданного питомца и записывает в
            окружение. Скрипт прописан  во вкладке «tests», для записи переменной в окружение
            использовался скрипт:  pm.environment
             */

            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200\", function () {\r",
                            "    pm.response.to.have.status(200);\r",
                            "});\r",
                            "\r",
                            "console.log(pm.test);\r",
                            "\r",
                            "var jsonData = pm.response.json();\r",
                            "pm.environment.set(\"petId\", jsonData.id);"
                        ],
                        "type": "text/javascript"
                    }
                }
            ],
            "request": {
                "method": "POST",
                "header": [],
                "body": {
                    "mode": "raw",
                    "raw": "{\r\n  \"id\": {{nameId}},\r\n  \"category\": {\r\n    \"id\": 0,\r\n    \"
                    "options": {
                        "raw": {
                            "language": "json"
                        }
                    }
                },
                "url": "https://petstore.swagger.io/v2/pet"
            },
```

```
        {
            "name": "Получение данных о питомце",
            /*
            Запросить питомца по id (метод GET /pet/{petId}) используя в запросе ранее записанную переменную из окружения
            Прописать скрипт, который будет брать тело ответа, менять параметр «status» на
            «sold» (продан) и перенаправлять его в запрос на изменение питомца (метод PUT/pet).
            Скрипт прописан во вкладке «tests», для перенаправления запроса использовался pm.sendRequest
             */
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "var jsonData = pm.response.json();\r",
                            "jsonData.status = 'sold';\r",
                            "\r",
                            "const statusSold = {\r",
                            "    url: 'https://petstore.swagger.io/v2/pet/',\r",
                            "    header: 'Content-Type: application/json',\r",
                            "    method: 'PUT',\r",
                            "    body: {\r",
                            "        mode: \"raw\",\r",
                            "        raw: JSON.stringify(jsonData)\r",
                            "    }\r",
                            "}\r",
                            "\r",
                            "pm.sendRequest(statusSold, function (err, response) {\r",
                            "    console.log('Успех');\r",
                            "});"
                        ],
                        "type": "text/javascript"
                    }
                }
            ],
            "request": {
                "method": "GET",
                "header": [],
                "url": "https://petstore.swagger.io/v2/pet/{{petId}}"
            },
            "response": []
        }
    ]
}
```

Follow link (ctrl + click)

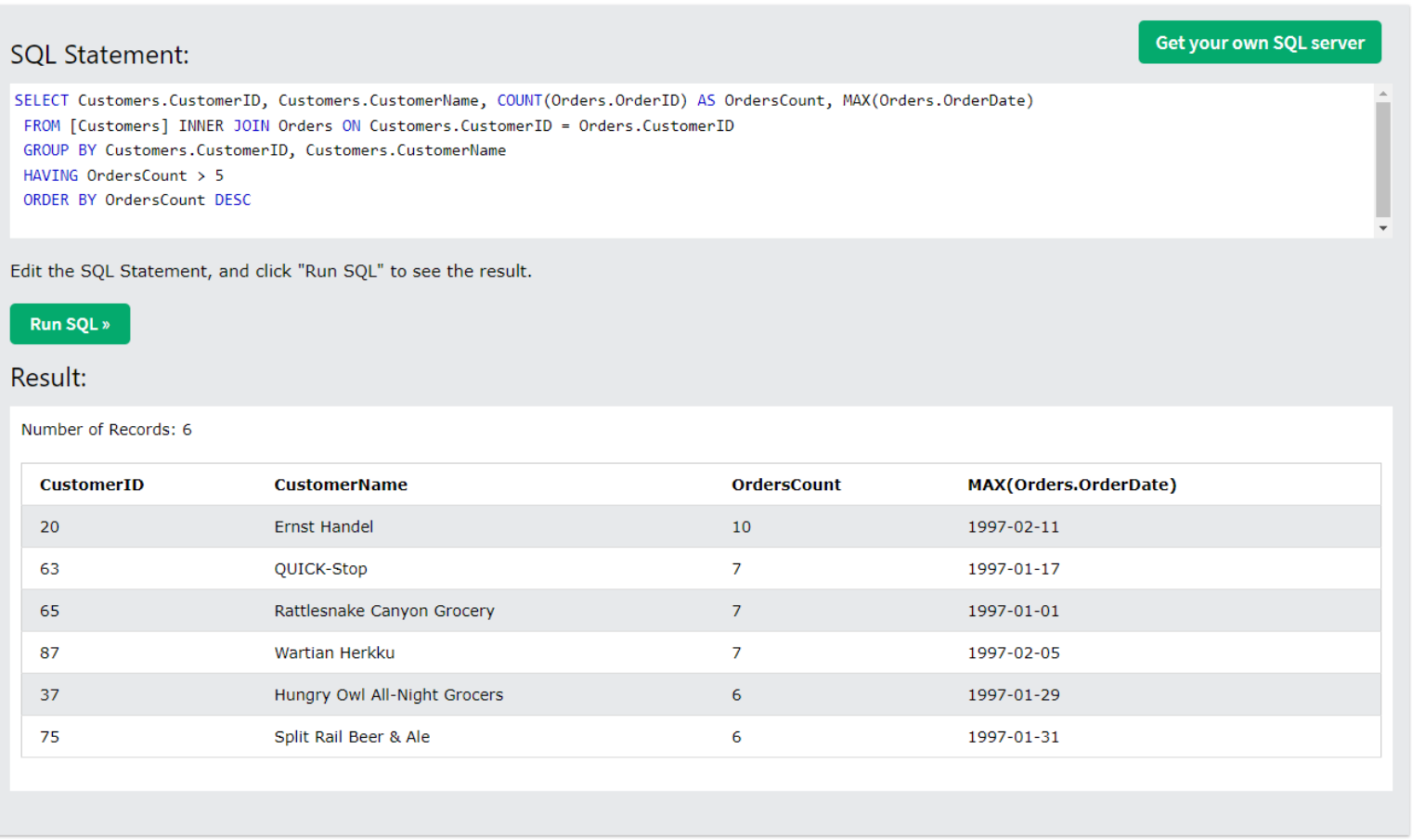
Задание 3

Таблицы, с которыми предстоит работать:: заказы (**Orders**) и клиенты (**Customers**), найти их можно тут: https://www.w3schools.com/sql/trysql.asp?filename=trysql_asc

Необходимо вывести id клиента и имя клиента, сделавших больше 5-ти заказов, а также количество заказов и дату последнего заказа. Список должен идти по убыванию от клиентов с большим количеством заказов к клиентам с меньшим количеством заказов

ГОТОВОЕ РЕШЕНИЕ:

SELECT Customers.CustomerID, Customers.CustomerName, COUNT(Orders.OrderID) AS OrdersCount, MAX(Orders.OrderDate)
FROM [Customers] INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerID, Customers.CustomerName
HAVING OrdersCount > 5
ORDER BY OrdersCount DESC



SQL Statement:

Get your own SQL server

```
SELECT Customers.CustomerID, Customers.CustomerName, COUNT(Orders.OrderID) AS OrdersCount, MAX(Orders.OrderDate)
 FROM [Customers] INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID
 GROUP BY Customers.CustomerID, Customers.CustomerName
 HAVING OrdersCount > 5
 ORDER BY OrdersCount DESC
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 6

CustomerID	CustomerName	OrdersCount	MAX(Orders.OrderDate)
20	Ernst Handel	10	1997-02-11
63	QUICK-Stop	7	1997-01-17
65	Rattlesnake Canyon Grocery	7	1997-01-01
87	Wartian Herkku	7	1997-02-05
37	Hungry Owl All-Night Grocers	6	1997-01-29
75	Split Rail Beer & Ale	6	1997-01-31