**SQL: агрегирующие функции.**

Некоторые запросы в SQL требуют подсчитать количество элементов или максимальное значение среди элементов и т. д. Для этого существуют агрегирующие функции. Перечислим основные из них:

* **AVG**: среднее значение;
* **MIN**: наименьшее значение;
* **MAX**: наибольшее значение;
* **COUNT**: количество строк;
* **SUM**: сумма значений.

|  |  |
| --- | --- |
| 1 | **SELECT** **AVG**(столбец) **FROM** название\_таблицы; |
| 1 | **SELECT MIN(**столбец**) FROM** название\_таблицы**;** |

**Примеры:**

1. Количество товаров, цена которых выше 50:

|  |  |
| --- | --- |
| 1 | **SELECT** **COUNT**(\*) **FROM** Product **WHERE** Quantity>50; |

https://cdn.discordapp.com/attachments/1110227508290388019/1143543995251826748/image.png

1. Вывести товар с наименьшей ценой и его название:

|  |  |
| --- | --- |
| 1 | **SELECT** **MIN**(Price),Name **FROM** Product; |

https://cdn.discordapp.com/attachments/1110227508290388019/1143545088593641552/image.png

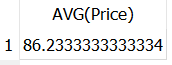
1. Вывести максимальное количество и название товара, цена которого больше 70:

|  |  |
| --- | --- |
| 1 | **SELECT** **MAX**(Quantity),Name **FROM** Product **WHERE** Price>70; |

https://cdn.discordapp.com/attachments/1110227508290388019/1143546077019123823/image.png

1. Вывести среднюю цену товара с названием Barilla:

|  |  |
| --- | --- |
| 1 | **SELECT** **AVG**(Price) **FROM** Product **WHERE** Name = 'Barilla'; |



**SQL: сортировка.**

Для упорядочения данных следует использовать сортировку. Для этого в конце запроса используется оператор **ORDER BY**. Сортировать можно как по возрастанию, так и по убыванию. Сортировать можно по нескольким столбцам.

**Примеры:**

1. Вывести без повторений названия товаров, сортируя по цене (по возрастанию):

|  |  |
| --- | --- |
| 1 | **SELECT** **DISTINCT**(Name) **FROM** Product **ORDER** **BY** Price; |



1. Вывести таблицу "Покупатели", сортируя имена по убыванию:

|  |  |
| --- | --- |
| 1 | **SELECT** \* **FROM** Buyer **ORDER** **BY** FIO **DESC**; |



1. Вывести список товаров, отсортированных сначала по названию (от А до Я), а потом по цене (по убыванию):

|  |  |
| --- | --- |
| 1 | **SELECT** Name, Price **FROM** Product **ORDER** **BY** Name **ASC**, Price **DESC**; |



**SQL: группировка.**

Оператор **GROUP BY** используется для группировки строк. Этот оператор возвращает одну строку для каждой группы, для которых потом можно выполнить агрегирующиефункции.

**Пример:**

1. Вывести количество товара по наименованию:

|  |  |
| --- | --- |
| 1 | **SELECT** Name, **COUNT**(\*) **AS** **Count** **FROM** Product **GROUP** **BY** Name; |



### Для фильтрации групп используют оператор HAVING.

### Пример:

1. Вывести наименования товаров, количество которых превышает 1 по убыванию:

|  |  |
| --- | --- |
| 1  2  3  4  5 | **SELECT** Name, **COUNT**(\*) **AS** **Count**  **FROM** Product  **GROUP** **BY** Name  **HAVING** **COUNT**(\*)>1  **ORDER** **BY** Count **DESC**; |



**SQL: подзапросы.**

Подзапросы в SQL - это запросы, которые вложены в другие запросы. Они используются для выполнения более сложных операций, комбинирования данных из разных таблиц или фильтрации результатов. Подзапросы могут быть использованы в различных частях запроса (SELECT, FROM, WHERE, HAVING и других). Они позволяют выполнять операции с меньшим набором данных, а затем использовать эти результаты в основном запросе.

Подзапросы бывают **коррелирующими** и **некоррелирующими**. Результат коррелирующего запроса зависит от строк, которые выбираются в основном запросе, в то время как у некоррелирующего – не зависит.

**Примеры:**

1. Некоррелирующий подзапрос:

Вывести товары, цена которых превышает среднюю цену.

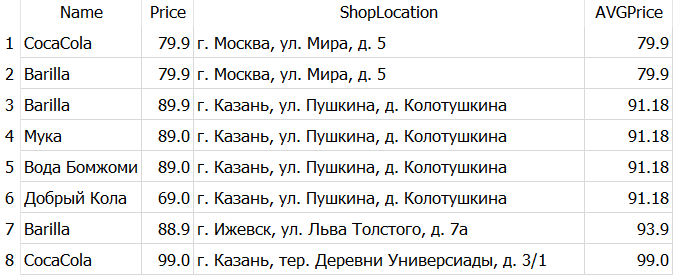
|  |  |
| --- | --- |
| 1  2 | **SELECT** \* **FROM** Product  **WHERE** Price > (**SELECT** **AVG**(Price) **FROM** Product); |



1. Коррелирующий подзапрос:

Вывести наименование товара, его цену, магазин, в котором он находится и среднюю цену по этому магазину. Условие: цена товара должна быть меньше или равна средней цене по магазину, в котором он находится. Результаты запроса отсортировать сначала по средней цене магазина (в порядке возрастания), а потом по цене товара (в порядке убывания).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | **SELECT** Name,Price,  (**SELECT** **Location** **FROM** Shop  **WHERE** Shop.ID = Product.ShopID) **AS** ShopLocation,  (**SELECT** **AVG**(Price)  **FROM** Product **AS** Sproduct  **WHERE** Sproduct.ShopID = Product.ShopID) **AS** AVGPrice  **FROM** Product  **WHERE** Price <= (**SELECT** **AVG**(Price)  **FROM** Product **AS** Sproduct  **WHERE** Sproduct.ShopID=Product.ShopID)  **ORDER** **BY** AVGPrice, Price **DESC**; |



1. Подзапросы в INSERT:

Добавить товар в магазин по адресу г. Казань, ул. Пушкина, д. Колотушкина.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | **INSERT** **INTO** [Product]  (ShopID, UnitOfMessure, Price, Quantity, Availability, Date, Name)  **VALUES**  ((**SELECT** ID **FROM** Shop  **WHERE** **Location** = 'г. Казань, ул. Пушкина, д. Колотушкина'),  'л','89.0',15,1,'22.08.2023','Вода Бомжоми'); |

1. Подзапросы в UPDATE:

Повысить цены на все товары в магазине по адресу г. Казань, тер. Деревни Универсиады, д. 3/1.

|  |  |
| --- | --- |
| 1  2  3 | **UPDATE** [Product] **SET** Price = Price+10  **WHERE** ShopID = (**SELECT** ID **FROM** Shop  **WHERE** **Location** = 'г. Казань, тер. Деревни Универсиады, д. 3/1'); |

1. Подзапросы в DELETE:

Удалить все товары из магазина по адресу г. Казань, ул. Кремлевская, д. 35.

|  |  |
| --- | --- |
| 1  2  3 | **DELETE** **FROM** Product  **WHERE** ShopID = (**SELECT** ID **FROM** Shop  **WHERE** **Location** = 'г. Казань, ул. Кремлевская, д. 35'); |