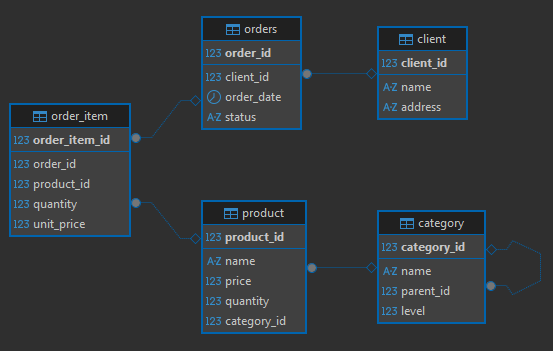
1. 

2.1. Запрос:

select

c.name as "Наименование клиента",

round(coalesce(sum(oi.quantity \* oi.unit\_price), 0), 2) as "Сумма"

from client c

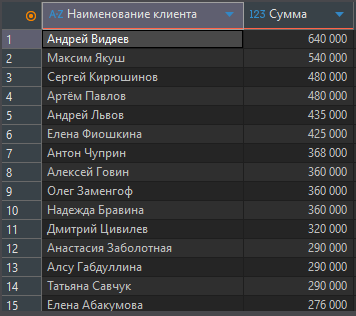
left join orders o on c.client\_id = o.client\_id

left join order\_item oi on o.order\_id = oi.order\_id

group by c.client\_id, c.name

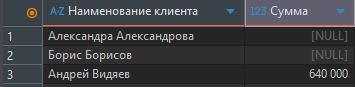
order by "Сумма" desc;

Результат:



left join делаем чтобы включить даже тех, у кого заказов нет. При join, как и при right join, этих клиентов не было бы (однако inner join можно использовать, если нас интересуют только активные клиенты). И чтобы у тех клиентов, кто еще не сделал заказ показывало 0, а не NULL, используем coalesce(…, 0) Вот результат:  


Иначе было бы:



2.2. Запрос:  
select

*c*.category\_id,

*c*.name as *"Категория"*,

count(*child*.category\_id) as *"Кол-во дочерних элементов"*

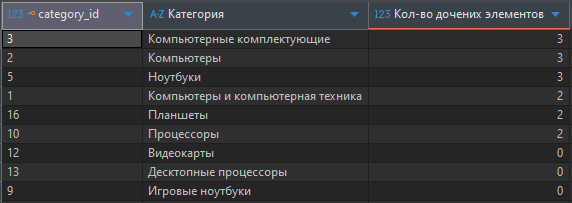
from category c

left join category *child* on *c*.category\_id = *child*.parent\_id

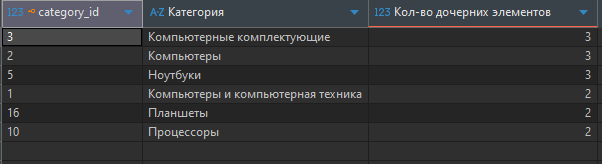
group by c.category\_id, c.name

order by " *Кол-во дочерних элементов*" desc, c.name;

Результат:



По условию мы должны брать только прямых потомков (*c*.category\_id = *child*.parent\_id) и считать только тех, кто непосредственно вложенный (count(*child*.category\_id)). Также делается left join для того, чтобы считать категории и без детей, но если они нам не нужны, то делаем просто join. Результат получится таким:



2.3.1. Запрос:

**create or replace** view top\_5\_best\_selling\_products\_last\_month **as**

**with recursive** category\_tree **as** (

**select**

category\_id,

**name** **as** category\_name,

parent\_id,

**level**,

category\_id **as** root\_id,

**name** **as** root\_name

**from** category

**where** parent\_id **is null**

**union all**

**select**

c.category\_id,

c.**name**,

c.parent\_id,

c.**level**,

ct.root\_id,

ct.root\_name

**from** category c

**join** category\_tree ct **on** c.parent\_id = ct.category\_id

)

**select**

p.**name** **as** **"Наименование товара"**,

ct.root\_name **as** **"Категория 1-го уровня"**,

**sum**(oi.quantity) **as** **"Общее количество проданных штук"**

**from** order\_item oi

**join** orders o **on** oi.order\_id = o.order\_id

**join** product p **on** oi.product\_id = p.product\_id

**join** category\_tree ct **on** p.category\_id = ct.category\_id

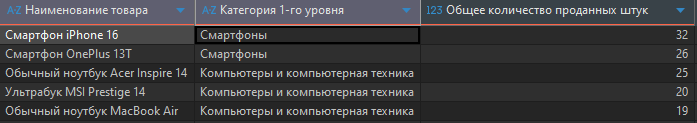
**where** o.order\_date >= CURRENT\_DATE - **interval** '1 month'

**group by** p.**name**, ct.root\_name

**order by** **"Общее количество проданных штук"** **desc**

**limit** 5;

Результат:



Сперва мы в запросе создаем или перезаписываем (если существует) представление. После создаем СТЕ, временную таблицу, там уже пишем первый select, базовый запрос, затем пишем рекурсивную часть, чтобы искать родителя и наследовать руты. Таким образом мы «протягиваем» имя корня вниз по всем уровням.

После создания вьюшки мы выполняем уже основной запрос по заданным условиям (в данном случае выявить топ-5 продаваемых товаров за последний месяц).

2.3.2. Пройдемся по запросу для начала. Тот запрос выполняет следующее:

* Находит топ-5 самых продаваемых товаров за последний месяц по количеству
* Использует with recursive для получения корневой категории 1-го уровня

В чем может быть проблема?

* Рекурсивный СТЕ будет выполняться каждый раз, когда используется view, а если категории будет много, то запрос может выполняться медленно
* Отсутствие индексов, тоже влияет на производительность.

Индексы можно принять так:

create index idx\_orders\_order\_date on orders(order\_date);

create index idx\_order\_item\_order\_id on order\_item(order\_id);

create index idx\_order\_item\_product\_id on order\_item(product\_id);

create index idx\_product\_category\_id on product(category\_id);

create index idx\_product\_name on product(name);

Так запрос не будет выполнять полное сканирование таблиц и, следовательно, будет выполняться быстрее.

Вместо рекурсии можно использовать столбец root\_category\_id в таблице category:

alter table category add column root\_category\_id integer;

update category set root\_category\_id = category\_id where parent\_id is null;

И можно будет писать запрос так:

join category c on p.category\_id = c.category\_id

join category root on root.category\_id = c.root\_category\_id

Можно еще разделить таблицу order\_item по месяцам и годам:

CREATE TABLE order\_item\_2025\_09 PARTITION OF order\_item FOR VALUES IN (2025, 9);