Въведение в SQL

Основни понятия

* SQL - език за съставяне на заявки към сървъри за бази данни (БД)
* всяка БД се състои от множество таблици, в които се намират даните
* всяка таблица съответства на определен обект от модела на БД
* обект в модела на БД е нещо, за което събираме данни: Продукти, Коментари, Потребители, Плащания и т.н.

* **първичен ключ** (PK - Primary Key) е уникалният за дадена таблица идентификатор на редовете. НЕ СЕ ПОВТАРЯТ СТ-СТИТЕ ПО РЕДОВЕТЕ.
СТ-СТИТЕ PK И FK по принцип не съвпадат. Те се генерират автоматично
PF - композитен/съставен първичен ключ.Идентифицира уникално редовете от таблицата и гарантира уникалност на комбинацията. Може да е комбинация от повече от 2 колони/ символа
FK - означава, че ст-стите на ключовете се генерират в таблицата първоизточник/ собствените таблици
Връзката на таблица към себе си (виж employees) се нарича рекурсивна връзка из изразява йерархия. По принцип се избягват такива връзки

```
  (pk)  user                    (pk)     posts
+-----+-----------+           +----+----------------+
| UID | UserName  |           | PID| Post           |
+-----+-----------+           +----+----------------+
|  1  |  John     |           | 1  |Hello ...       |
+-----+-----------+           +----+----------------+
|  2  |  Maria    |           | 2  |MySQL is...     |
+-----+-----------+           +----+----------------+
|  3  |  Peter    |           | 3  |SQL query...    |
+-----+-----------+           +----+----------------+
|  4  |  Anna     |           | 4  |Oracle driver...|
+-----+-----------+           +----+----------------+
```

* **външен ключ** (FK - Foreign Key) е идентификатор, който задава съответствието на редовете от една таблица към друга таблица. ст-стите ***може*** да се повтарят
Неписано правило е колните за PK И FK да се кръщават с едно и също име. Страна много се отбелязва с чертички <

```
  (pk)  user                    (pk)     posts         (fk)
+-----+-----------+           +----+----------------+------+
| UID | UserName  |           | PID| Post           | UID  |
+-----+-----------+           +----+----------------+------+
|  1  |  John     |           | 1  |Hello ...       |  1   |
+-----+-----------+           +----+----------------+------+
|  2  |  Maria    |           | 2  |MySQL is...     |  2   |
+-----+-----------+----------<+----+----------------+------+
|  3  |  Peter    |           | 3  |SQL query...    |  1   |
+-----+-----------+           +----+----------------+------+
|  4  |  Anna     |           | 4  |Oracle driver...|  2   |
```

Връзки между таблиците

* 1:M -> на един ред от А съответстват един или повече реда от В
* 1:1 -> на един ред от А съответства точно един ред от В. Частен случай! Трябва да мислим, като за една таблица с 2 части
* M:N -> на един ред от А съответстват един или повече реда от В, обратното
  също е вярно; обикновено се преработва до 1:M и N:1. Води до аномалии

в SQL не се води борба за намаляване на броя на редовете, дори напротив. От БД не се трият данни!
Води се за намаляване броя колони.

Референциална цялост на БД (интегритет)

С данните могат да бъдат извършвани четири основни опрации:

* ЧЕТЕНЕ на **данни**
* ДОБАВЯНЕ на **редове**
* ПРОМЯНА на **данни** (рестриктивно/каскадно). Не засяга интегритета, освен ако не пипаме специалните колони (тези с ключ).

Обичайно промените се правят каскадно, тоест при промяна от едната страна се променя и другата.

* ИЗТРИВАНЕ на **редове** (рестриктивно/каскадно) обичайно е рестриктивно

Рестриктивно -проверява, дали имаме записи с FK от страна 1 (ПК) и ако има не може да изтрием. Това могат да правят само хора със спец. Права
Каскадно - ако изтрием запис от 1, трие и съответстващите записи от страна много
ВАЖНО! ВСЯКА ТАБЛИЦА ИМА РК, ВТОРИЧЕН МОЖЕ ДА ИМА ИЛ ДА НЯМА
ЧЕТЕНЕТО на данни няма отношение към референциалната цялост, тъй като при тази операция не се променя съдържанието на таблиците

При ДОБАВЯНЕ определя последователността на действията (1. добавя се ред от страната 1; 2. добавя се ред от страната М). Първо се добавя страна 1. После страна много. Обратното не може.

За ПРОМЯНА/ИЗТРИВАНЕ интегритета може да се поддържа по два начина: рестриктивно или каскадно.

Страна 1 (pk) user — страна много (pk) posts (fk)

UID	UserName
1	John
2	Maria
3	Peter
4	Anna

PID	Post	UID
1	Hello ...	1
2	MySQL is...	2
3	SQL query...	1
4	Oracle driver...	2

Модел на базата данни Northwind

(чувствителност към регистър на три нива: 1, ниво сесия, 2. ниво таблица
настройва се от tools-pref-db-nls, като somparesion чувствително, linguistic - не чувст.

ЛИНКОВЕ ЗА ИЗТЕГЛЯНЕ

```
СЪРВЪР ЗА БД – СТАРА ВЕРСИЯ
Express edition – версия 11,2. ЗАПИШИ ПАРОЛА SYSTEM!!! РАЗАРХИВИРАМ
```

1. https://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/xe-prior-releases-5172097.html

2. http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html - - 64BIT С ВКЛ.JDK. ТРЯБВА ДА СЕ РАЗАРХИВИРА И ДА СЕ СЪЗДАДЕ ПРЯК ПЪТ. Трябва да си направим профил в оракъл и е много важно да запишем паролите за admin И system

2.1. http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html ако се наложи https://notepad-plus-plus.org/download/v7.7.1.html

НАСТРОЙКИ

TOOLS - PREF- ENV.. – UTF-8
TOOLS - PREF-CODE EDITOR – LINE GUTTER – SHOW LINE NUMBERS - ПОКАЗВА НОМЕР НА РЕДОВЕТЕ
TOOLS – PREF – DB – ADVANCE – НАСТРОЙВА NULL

ПЪРВА ВРЪЗКА - АДМИН

ОТ + се прави връзка със system(admin)
Name: Admin
User: system
Pass: system
✓ /save pass/

TEST (АКО ИЗПИСВА SUCCESS ПОДЪЛЖАВАМЕ НАПРЕД –-→ SAVE–-→CONNECT
КОПИРА СЕ ТОЧКА 2 И 3 И СЕ СЛАГАТ В SQL DEV. 1 СЕ ИЗПОЛЗВА ЗА ПЪЛНО ИЗТРИВАНЕ

```
-- 1. Run to drop schema
-- DROP USER northwind CASCADE;

-- 2. Run to create user and schema
CREATE USER northwind IDENTIFIED BY northwind
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  QUOTA UNLIMITED ON users;

-- 3. Run to grant permissions
GRANT "CONNECT" TO northwind
```

```
/
GRANT DBA TO northwind
/
GRANT "RESOURCE" TO northwind
/
ALTER USER northwind DEFAULT ROLE "CONNECT", DBA,"RESOURCE"
```

ВТОРА ВРЪЗКА - Northwind

ОТ + се прави връзка със nothwind. Важно да не е admin!!!!!

Name: Northwind
User: northwind
Pass: northwind

✓ /save pass/

TEST (АКО ИЗПИСВА SUCCESS ПОДЪЛЖАВАМЕ НАПРЕД –-→ SAVE–-→CONNECT

ВЪВ ВТОРАТА ВРЪЗКА СЕ КОПИРА ТЕКСТА (ВИЖ ПРИЛОЖЕНИЕТО – ДЪЛГИЯ СКРИПТ)

SQL - Structured Query Language

* езикът SQL се състои от две групи изрази: DDL и DML.

DDL - Data Definition Language

* CREATE ... ->създава обект (таблица, изглед, процедура, индекс и т.н.)
* ALTER ... ->променя съществуващ обект
* DROP .... ->премахва обект

DML - Data Manipulation Language

* SELECT ... ->чете данни от таблиците (много таблици)
* INSERT ... ->добавя редове към таблица (върху 1 таблица или ограничен брой редове)
* UPDATE ... ->променя данни от таблица (върху 1 таблица или ограничен брой редове)
* DELETE ... ->изтрива редове от таблица (върху 1 таблица или ограничен брой редове)

SELECT заявки

```sql
синтактична диаграма
SELECT * | col1, col2,... какво да се върне, като колони. Изброяваме в реда, в които искаме да ги видим
FROM  table(s) от коя таблица избираме
[WHERE col1 operator value1 ... ] ограничения на които трябва да отговаря, критерии на които да отговарят
```
| изключващо или
[] незадължит. елементи
звезда * значи всички
колоните с малки букви или кемъл кейс. Select, FROM i т.н. с големи (добра практика)
SQL не е чувст. към регистър. Binary - чувствително/ anis I linguistic - не е чувствително
Run statment = CTRL и Enter
RUN SCRIPT Е ЗА ПОВЕЧЕ ЗАЯВКИ
F8 - вади история
-- коментар
Думите се разделят с _
Запетая в началото на реда
```sql
SELECT *
FROM customers
---
SELECT country
, company_name
, contact_name
```

```
, phone
FROM customers
```

WHERE клауза

```
* филтрира данните от таблиците включени в заявката
* при повече от едно условие, отделните условия се свързват с AND или OR
```

WHERE	Действие
[1] AND [2]	Задължаващо И. трябва да са изпълнени и двете, ако едното не е изпълнено, не се връща нищо
[1] OR [2]	Или! може да са изпълнени [1] или [2] или и двете заедно, важното е поне едно да е изпълнено
[1] OR [2] AND [3]	Трябва да се изпълнени 1 и 3 или 2 и 3
([1] OR [2]) AND [3]	[1],[2]- едно от двете или двете заедно, [3]-винаги

Кои редове ще се върнат в резултата?

```sql
WHERE (1) OR (2) AND (3)  -- по подобие на 5 + 6 * 3
--     a      b       c
```

a	b	c	v
d	b	c	v
d	e	c	x
g	b	c	v
a	b	f	v

??????????????

```sql
WHERE ( (1) OR (2) ) AND (3)  -- по подобие на ( 5 + 6 ) * 3 => 5 * 3 + 6 * 3
--       a      b         c   -- или ((1) OR (2) ) AND (3) е като
                              --      (1) AND (3) OR (2) AND (3)
```

```
+----+---+---+---+
| a  | b | c | v |
+----+---+---+---+
| d  | b | c | v |
+----+---+---+---+
| d  | e | c | x |
+----+---+---+---+
| g  | b | c | v |
+----+---+---+---+
| a  | b | f | x |
+----+---+---+---+
```

за различните видове данни се използват различни оператори

* **операторите за филтриране на числа,** дати и символи (char - ЕГН,Банкова сметка,..)

не са подходящи за свободен текст, а за фиксиран формат

Оператори	Пример
>, <, >=, <=, =, <> различно	WHERE price > 20 AND price <> 50
[NOT] IN ('val1','val2',....)	WHERE itemID IN ('AA123','AB456','BD213')
[NOT] BETWEEN 'val1' AND 'val2'	WHERE price BETWEEN 5 AND 30

подходящо за ДАТИ И ЧИСЛА. ЗАТВОРЕН ИНТЕРВАЛ. ВКЛЮЧВА СЕ 5 И 30

[NOT] да не е/ да не съвпада

IN се използва за данни с дискретен характер. Те са фиксирани ст-сти/ номер поръчка, код, клиент, продукт, не се изискава да има всичко зададено, може само едното да присъства.
IN Позволява задаване на множество от ст-ст, може да съвпада с част.
Текстови данни:
- определен фиксиран формат
- свободен текст

Да се внимава с колони, които се срещат в повече от една таблица

```sql
SELECT order_id
, order_date
, ship_country
, ship_city
FROM orders
```

```
WHERE order_date >= '2015-07-01' AND order_date <= '2015-07-19'
не е нужно col по която филтрираме да присъства в резултата
Всичко което не е число се отделя в ед. кавички ' Съвръра прави разлика м/у ' и "
при различните сървъри датите се задават различно. При някои формата се задава
още при създаване на таблицата.
Оракъл съхранява датите, като числа. Настройва се от tools-pref-nls
ВАЖНО! РЕД НА ПИСАНЕ НА ЗАЯВКИ. ТАКЪВ Е И РЕДА НА ИЗПЪЛНЕНИЕ
1.FROM
2.WHERE
3.SELECT

SELECT customer_id
, order_id
, order_date
, ship_country
FROM orders
WHERE customer_id IN (1,8,57)
---
SELECT customer_id
, customer_code
, company_name
, country
, city
FROM customers
WHERE customer_code IN ('ALFKI','BOLID','PARIS') --AND ...
-----
-- Същото като
SELECT customer_id
, customer_code
, company_name
, country
, city
FROM customers
WHERE customer_code = 'ALFKI' OR customer_code = 'BOLID' OR customer_code = 'PARIS' --AND ...
----
SELECT product_name
, quantity_per_unit
, unit_price
FROM products
WHERE unit_price BETWEEN 10 AND 20
---
SELECT product_name
, quantity_per_unit
, unit_price
FROM products
```

```
WHERE unit_price NOT BETWEEN 10 AND 20
---
-- Същото като
SELECT product_name
, quantity_per_unit
, unit_price
FROM products
WHERE unit_price >= 10 AND unit_price <= 20
---
-- заместване на NOT BETWEEN
SELECT product_name
, quantity_per_unit
, unit_price
FROM products
WHERE unit_price < 10 OR unit_price > 20
```

* операторите за филтриране на текст (varchar, text)

ВАЖНО !!! REGEXP_LIKE И LIKE СЕ ИЗПОЛЗВАТ ЗА СВОБОДЕН ТЕКСТ, НЯМА ИНТЕРВАЛ ПРЕДИ СКОБИТЕ

Оператори	Пример
[NOT] LIKE 'pattern'	```WHERE title LIKE 'The blue %'```

Може да задаваме част от текста използва се за прости търсения

REGEXP_LIKE(column, pattern[, match])	```WHERE REGEXP_LIKE(company_name, '^A.*[ae]$')```

функция (колона, шаблон, [не е задължително, допълнит. настройка]/
тук не се използват символите % и _
column -колона , pattern - шаблон [, match] - незадължителна допълнителна настройка
ВАЖНО! В REGEXP_LIKE НЕ СЕ ИЗПОЛЗВАТ % И _ !!!!!!!!!!!!!!!!!!!!

ВАЖНО!
% 0 ИЛИ ПОВЕЧЕ СИМВОЛА ОТ СТРАНАТА В КОЯТО Е ПОСТАВЕН _ **ТОЧНО 1 СИМВОЛ. НЕ МОЖЕ ДА СА 0**
ИНТЕРВАЛА СЪЩО СЕ БРОИ СА СИМВОЛ

```sql
col = 'Maria' трябва да отговаря точно, няма да вкл. Мариана, Мария Попова, Марио и т.н.

'Mario'

col LIKE 'M%' => 'M', 'Ma', 'Mab', ...
```

col LIKE 'M_' => 'Ma', 'Mb', ..., 'M ', но не 'M' или 'Mab'
```sql
SELECT customer_code
, contact_name
FROM customers
WHERE contact_name LIKE 'M%'
----
SELECT customer_code
, contact_name
FROM customers
WHERE contact_name LIKE 'M_r%'
```

```sql
'Xyz Abcd' ТЪРСИМ ФАМИЛИЯ, КОЯТО започва с А и някакво име
'__% A__%'ДА

' Асен Петров'
'Мария Петров а'

LIKE ''

1) A% -НЕ
2) _% A% -НЕ
3) __ A% -НЕ
4) % A% -НЕ
```

```sql
SELECT customer_code
, contact_name
FROM customers
WHERE contact_name LIKE '__% A__%'
```
композитна колна- съсржа повече от едно нещо
според теорията на БД всяка колона трябва да съдържа единична ст-ст.
напр. код на тел е една колна, а самия тел - друга
ако ще се търси по дадена кол. то е добре тя да не е композитна
---+

REGEXP LIKE

основни оператори за съставяне на регулярни изрази (pattern)

Оператор	Действие	Пример
^	редът да започва с онова което е след човката	```'^Maria'```
$	редът да завършва с това преди долар	```'son$'```
.	замества точно 1 символ, но всеки специален символ, замества само 1символ, но той може да всеки	```'.*'```
{n}	n - пъти предходното ПОСЛЕДОВАТЕЛНО????	
{n,m}	поне n пъти, но не повече от m пъти	```'^.n{1,2}'```
{,m}	не повече от m пъти	
{n,}	поне n пъти, не по-малко от n пъти	
[списък]	един от символите в списъка	```'^Mar[yt]'```
*	0 или повече срещания на преходното	
?	0 или 1 срещане срещания на преходното	
+	1 или повече срещания срещания на преходното	
()	символ за групиране ^(A|M)	
|	или	
[^]	тук ^ си сменя смисъла и значи да не е. ИГРАЕ РОЛЯ НА ОТРИЦАНИЕ	
[]	МОЖЕ ДА СЕ СЛОЖИ ПОРЕДИЦА ОТ СИМВОЛИ, НО ЗАМЕСТВА 1 [cty]- една от тези	
\d	цифра	
\d	не е цифра	
\w	дума	
\W	не е дума	

```
           +----------+-------------------------------------+------------------+
\s интервал \S не е интервал

допълнителни настройки на процеса на търсене (match)
В MATCH МОЖЕ ДА СЕ ИЗПБРОЯВАТ ПАРАМЕТРИ

+-------+----------------------------------+
| match | действие                         |
+-------+----------------------------------+
| i     | да НЕ е чувствително към аА регистър
+-------+----------------------------------+
| c     | да Е чувствително към аА         |
+-------+----------------------------------+
| n     | да третира . като част от текста |
+-------+----------------------------------+
| m     | текстът е многоредов. Да търси по всички редове
+-------+----------------------------------+

```sql
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^A')
Да започва с А
'ana' някъде в текста да има ana
---
не!
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^A|^M')
да започва с А или с М, но това не е удачен вариант.
Трябва да се обедини със скоби. Виж Долния пример
---
да!
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^(A|M)')
---
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, 'er$')
ВСИЧКО което завършва на er
---
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^An.')
```
```

```
---


SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, 'n{2}|a{2}|t{2}') една от трите букви да се повтаря ТОЧНО два пъти ПОСЛЕДОВАТЕЛНО ЛИ?!
---
-- 'Xyz Bennett'
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, 'n{2}.t{2}$') Два пъти n, . замества всеки символ и 2 пъти t
----
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^Mar[ity]') Да започва с MAR и после да има една от трите букви i,t или y
----
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^Mar[a-zA-Z0-9 ]')
Да започва с MAR и после може да има от a до я с малки и големи,всички числа и интервал
-----
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^Mar[^ty]') Да започва с MAR и следващата да НЕ Е t или y
----
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, ' A[a-z]+$')
итревал A, едно или повече срещания (+) на всички букви и символ за край. Връща всички фамилии, които започват с A
----
SELECT contact_name
FROM customers
WHERE REGEXP_LIKE(contact_name, '^\w+\s\w+\s\w+$')
Търсим хора с три имена, w е дума, + за един или повече срещания, s за спейс и накрая $ за край
/w дума
/W не е дума
/s спейс
/S не е спейс
/d цифра
/D не е цифра
```

1. **https://docs.oracle.com/cd/E11882_01/server.112/e41084/ap_posix003.htm#SQLRF55544**
2. **https://regexr.com/ тренер.Да сменя настройката от JS на PCRE**

```sql
ALTER SESSION SET NLS_COMP=LINGUISTIC;

ALTER SESSION SET NLS_SORT=BINARY_CI;
```

Свързване на таблици в SELECT

* свързвайки таблици в SELECT можем да комбинираме редовете и колоните от 2 или повече таблици

Видове свързвания

* **INNER JOIN** (Вътрешно съединение) - връща само данните за които има движение. Peter i Anna няма да се върнат

В резултата ще се върнат всички редове от таблица A и таблица B, за които е изпълнено **B.FK = A.PK**

```
  (pk)  user                    (pk)     posts     (fk)
+-----+-----------+           +----+-------------+------+
| UID | UserName  |           | PID| Post        | UID  |
+-----+-----------+           +----+-------------+------+
|  1  |  John     |           | 1  |Hello ...    |  1   |
+-----+-----------+           +----+-------------+------+
|  2  |  Maria    |           | 2  |MySQL is...  |  2   |
+-----+-----------+---------< +----+-------------+------+
|  3  |  Peter    |           | 3  |SQL query... |  1   |
+-----+-----------+           +----+-------------+------+
|  4  |  Anna     |           | 4  |PHP driver...|  2   |
+-----+-----------+           +----+-------------+------+
```

Резултат:

```
+-----------+-------------+------+
| UserName  | Post        | UID  |
+-----------+-------------+------+
|  John     |Hello ...    |  1   |
+-----------+-------------+------+
|  Maria    |MySQL is...  |  2   |
+-----------+-------------+------+
```

```
|  John     |SQL query... |  1   |
+-----------+-------------+------+
|  Maria    |PHP driver...|  2   |* псевдоними на таблици
```sql
SELECT table1.column1, table2.column2, table2.column5
FROM table1
      INNER JOIN
     table2
     ON table2.fk_column = table1.pk_column
```
```sql
FROM BANKSTATEMENTLINEBANKDOCUMENTINFORMATION
```
```sql
SELECT t1.column1, t2.column2, t2.column5
FROM table1 t1
      INNER JOIN
     table2 t2
     ON t2.fk_column = t1.pk_column     - t1 е псевдоним и е валиден само докато заявката се изпълнява. Да не се бърка със синоним
```
```sql
SELECT t1.customer_code
, t1.company_name
, t2.order_id
, t2.order_date
, t2.ship_country
, t1.country
FROM customers t1
        INNER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
WHERE t1.customer_code IN ('ALFKI','BOLID','PARIS')
---
SELECT t1.customer_id
, t1.order_id
, t1.order_date
, t3.product_name
, t2.quantity
, t2.unit_price
FROM orders t1
        INNER JOIN
     order_details t2
        ON t2.order_id = t1.order_id
        INNER JOIN
     products t3
```
```

```
        ON t2.product_id = t3.product_id
WHERE t1.customer_id = 1 --ALFKI
ORDER BY t1.order_id
Когато свързваме таблици, трябва да минем през всички свързващи,, които стоят по пътя, като търсим мин. брой таблици
Не могат да се прескачат свързващи и да се свързват таблици без обща колона.
ВАЖНО!Всяка следваща таблица се свързва с множеството от предходните, които сме свързали.
3 се свързва с множеството от 1 и 2, а не само с 2.
---
SELECT t1.customer_id
, t1.order_id invoiceNo
, t1.order_date
, t3.product_name
, t2.quantity
, t2.unit_price
, t2.quantity * t2.unit_price total -- може и така: as total
, t2.quantity * t2.unit_price * 1.2 totalVAT
FROM orders t1
        INNER JOIN
     order_details t2
        ON t2.order_id = t1.order_id
        INNER JOIN
     products t3
        ON t2.product_id = t3.product_id
WHERE t1.customer_id = 1 --ALFKI
ORDER BY t1.order_id
Могат да се налагат ограничения във сяка таблица вкл. в FROM, дори в/у колони, които не се виждат в момента
---
SELECT t1.customer_id
, t1.order_id invoiceNo
, t1.order_date
, t3.product_name
, t2.quantity
, t2.unit_price
, t2.quantity * t2.unit_price total -- може и така: as total
, t2.quantity * t2.unit_price * 1.2 totalVAT
FROM orders t1
        INNER JOIN
     order_details t2
        ON t2.order_id = t1.order_id
        INNER JOIN
     products t3
        ON t2.product_id = t3.product_id
WHERE t1.customer_id = 1 --ALFKI
        AND
      t2.quantity * t2.unit_price < 100
```

```
ORDER BY total DESC


Псевдонимите на колони имат мн. по-ограничено действие от тези на таблици
as се изсиква само при някои сървъри
total е изчислима колона, получва се от ед. цена и количество, като може тези две колони да не са извикани
 Което по принцип не е прието, но се налага понякога
 ВАЖНО! ПОСЛЕДНОВАТЕЛНОСТ НА ИЗПЪЛНЕНИЕ НА ЗАЯВКА!
 1. FROM
 2. WHERE
 3. SELECT
 4. ВСИЧКО ПОД WHERE СЕ ИЗПЪЛНЯВА СЛЕД СЕЛЕКТ И ТОГАВА МОЖЕ ДА ИЗВИКВАМЕ ПСЕВДОНИМИ НА КОЛОНИ,
 ЗАЩОТО ТЕ ВЕЧЕ СА ДЕФИНИРАНИ И ПРОЧЕТЕНИ
 ORDER Е ЗА СОРТИРАНЕ
```

```sql
ПО ПОДРАЗБИРАНЕ РЕДА Е asc - възходящ/ desc- нискодящ
ORDER BY col1 [ASC]|DESC, col2 [ASC]|DESC, ...
--или
извикване според поредния номер в селекст частта
ORDER BY 1 [ASC]|DESC, 2 [ASC]|DESC, ...
```

ВАЖНО! ВИНАГИ СЕ ЗАПОЧВА ОТ КОЛОНА 1 ИЛИ АКО СА ДВЕ, ТО ЗАПОЧВАМЕ ОТ ТОВА, ЗА КОЕТО НИ ПИТАТ

ЛЕКЦИЯ 4

* OUTER JOIN (Външно съвединение) – Дава възможност да се извадят всички данни, но без да го гарантира
  + **LEFT OUTER JOIN** - ляво външно съединение

  + **RIGHT OUTER JOIN** - дясно външно съединение. Взема всички данни от лявата таблица, без да гледа за съвпадение. Там където няма съвпадение с дясната таблица излизат празни клетки / страна много/

  + **FULL OUTER JOIN** - ляво + дясно външно съединение – не се поддържа от всички сървъри. Напр. в MySQL го няма

NULL – състояние състоящо се в липса на данни. Настройва се от Tools – Pref – DB - Advance
* **LEFT** или **RIGHT** се определя от мястото на таблицата, която е страна 1
във връзката между таблиците

* **LEFT**: В резултата се връщат всички данни от A (без значение дали имат
свързани записи в B) със съответстващите данни от B като за редовете от A, които
нямат свързани записи в B, клетките в колоните от B са празни.

NULL- означава, че няма данни, не значи 0
ctrl + / - коментар

```
  (pk)  user                    (pk)     posts     (fk)
+-----+-----------+           +----+-------------+------+
| UID | UserName  |           | PID| Post        | UID  |
+-----+-----------+           +----+-------------+------+
|  1  |  John     |           | 1  |Hello ...    |  1   |
+-----+-----------+           +----+-------------+------+
|  2  |  Maria    |           | 2  |MySQL is...  |  2   |
+-----+-----------+---------< +----+-------------+------+
|  3  |  Peter    |           | 3  |SQL query... |  1   |
+-----+-----------+           +----+-------------+------+
|  4  |  Anna     |           | 4  |PHP driver...|  2   |
+-----+-----------+           +----+-------------+------+
```

Резултат:

```
  (pk)                          (fk)
+-----+-----------+-------------+------+
| UID | UserName  | Post        | UID  |
+-----+-----------+-------------+------+
|  1  |  John     |Hello ...    |  1   |
+-----+-----------+-------------+------+
|  2  |  Maria    |MySQL is...  |  2   |
+-----+-----------+-------------+------+
|  1  |  John     |SQL query... |  1   |
+-----+-----------+-------------+------+
|  2  |  Maria    |PHP driver...|  2   |
+-----+-----------+-------------+------+
|  3  |  Peter    |             |      |
+-----+-----------+-------------+------+
|  4  |  Anna     |             |      |
+-----+-----------+-------------+------+
```

* ляво или дясно външно съединение?

```sql
       1   LEFT                  M
FROM tab1  ....     OUTER JOIN tab2
       M   RIGHT                 1
```

```sql
SELECT t1.customer_code
, t1.country
```

```
, t2.order_id
, t2.order_date
, t2.ship_country
FROM customers t1
        LEFT OUTER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
WHERE t1.customer_code IN ('ALFKI','BOLID','PARIS')
ORDER BY 1
---
```

DESCRIBE orders – дава всички имена на колони и тип данни

NULL – NOT NULL – задължителн колони. В сървъра на майкрософт го няма, има sp help

```
----
-- Грешен тип на свързване
SELECT t1.customer_code
, t1.country
, t2.order_id
, t2.order_date
, t2.ship_country
FROM customers t1
        RIGHT OUTER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
WHERE t1.customer_code IN ('ALFKI','BOLID','PARIS')
ORDER BY 1
----
-- Грешен начин на филтриране. ЛИПСВА КЛИЕНТ ЛАРИС
SELECT t1.customer_code
, t1.country
, t2.order_id
, t2.order_date
, t2.ship_country
FROM customers t1
        LEFT OUTER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
WHERE t1.customer_code IN ('ALFKI','BOLID','PARIS')
        AND
      t2.order_date BETWEEN '2016-01-01' AND '2016-12-31'

ORDER BY 1
```

```
---
SELECT t1.customer_code
, t1.country
, t2.order_id
, t2.order_date
, t2.ship_country
FROM customers t1
        LEFT OUTER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
WHERE t1.customer_code IN ('ALFKI','BOLID','PARIS')
        AND
      (
        t2.order_date BETWEEN '2016-01-01' AND '2016-12-31'
        OR
        t2.order_date IS NULL)
ORDER BY 1
```

*** изключение от правилото за LEFT/RIGHT. КОГАТО ВЪВ ВЪНШИЯ КЛЮЧ НЯМА СТ-СТ ВЪВ ВРЪЗКА С ИНТЕГРИТЕТ**

```
  (pk)  user                    (pk)     posts      (fk)
+-----+-----------+           +----+-------------+------+
| UID | UserName  |           | PID| Post        | UID  |
+-----+-----------+           +----+-------------+------+
|  1  |  John     |           | 1  |Hello ...    |  1   |
+-----+-----------+           +----+-------------+------+
|  2  |  Maria    |           | 2  |MySQL is...  |  2   |
+-----+-----------+---------< +----+-------------+------+
|  3  |  Peter    |           | 3  |SQL query... |  1   |
+-----+-----------+           +----+-------------+------+
|  4  |  Anna     |           | 4  |PHP driver...|      |
+-----+-----------+           +----+-------------+------+
```

Резултат:

```
 (pk)
+-----+---------------+----------+
| PID | Post          | UserName |
+-----+---------------+----------+
| 1   | Hello ...     | John     |
+-----+---------------+----------+
| 2   | MySQL is...   | Maria    |
+-----+---------------+----------+
| 3   | SQL query...  | John     |
+-----+---------------+----------+
```

```
| 4   | PHP driver... |          |
```

* FULL OUTER JOIN

```
  (pk)  user                    (pk)     posts      (fk)
+-----+-----------+           +----+-------------+------+
| UID | UserName  |           | PID| Post        | UID  |
+-----+-----------+           +----+-------------+------+
|  1  |  John     |           | 1  |Hello ...    |  1   |
+-----+-----------+           +----+-------------+------+
|  2  |  Maria    |           | 2  |MySQL is...  |  2   |
+-----+-----------+---------< +----+-------------+------+
|  3  |  Peter    |           | 3  |SQL query... |  1   |
+-----+-----------+           +----+-------------+------+
|  4  |  Anna     |           | 4  |PHP driver...|      |
+-----+-----------+           +----+-------------+------+
```

Резултат:

```
+-----+---------------+----------+
| UID | Post          | UserName |
+-----+---------------+----------+
| 1   | Hello ...     | John     |
+-----+---------------+----------+
| 2   | MySQL is...   | Maria    |
+-----+---------------+----------+
| 1   | SQL query...  | John     |
+-----+---------------+----------+
|     | PHP driver... |          |
+-----+---------------+----------+
| 3   |               | Peter    |
+-----+---------------+----------+
| 4   |               | Anna     |
+-----+---------------+----------+
```

Изчисления в SELECT

* независимо от това какво и как изчисляваме, данните в таблиците **не се променят**

Оператори

* аритметични: +, -, * , / + понякога има странични ефекти, може да направи конкатенация
* конкатенация: ||

```sql
MS SQL, Sybase: SELECT firstname + ' ' + lastname ...
MySQL: SELECT concat(firstname, ' ', lastname) ...
MySQL: SELECT concat_ws(' ', firstname, lastname може да се използва,когато сме сигурни,че е използван един разделител.Напр.интервал
```

```sql
SELECT firstname || ' ' || lastname Employee – псевдоним на колона
FROM employees
----
SELECT company_name || ' (' || country || ', ' || city || ')' Customer – скобите са сложени, за да излезе резултата в скоби
FROM customers

```

https://docs.oracle.com/cd/E11882_01/server.112/e41084/toc.htm

Функции (Single-row functions)

Действат на всеки ред от заявката/ селект, where, order by, from/ недостатък е,че са различни при различните сървъри

* математически

ROUND(n[,m])	закръглява n до m-я знак. Традиционно закръгляване, но 35,20 се изписва 35,2
TRUNC(n[,m])	"реже" от n m десетични знака. Нищо не закръгля, директно реже. Може да се ползва за дати
CEIL(n)	закръглява до нагоре до цяло число. Закръгля до най-малко цяло число, което е по-голямо от аргумента. 2,1 на 3
FLOOR(n)	закръглява надолу до цяло число. Закръгля до най-малко цяло число, което е по-малко от аргумента. 3,9 на 3

ФУНЦИИТЕ МОГАТ ДА СЕ ВЛАГАТ ЕДНА В ДРУГА

```sql
SELECT product_name
, unit_price
, unit_price * 1.95583 priceBGN
, ROUND(unit_price * 1.95583, 2) priceRnd
, TRUNC(unit_price * 1.95583, 2) priceTrn
, CEIL(unit_price * 1.95583) priceC
, FLOOR(unit_price * 1.95583) priceF
```

FROM products

* текст

SUBSTR(txt, p[, l])	връща от позиция p в txt l символа.**ВАДИ ЧАСТ ОТ ТЕКСТА (текст, начална позиция,колко символа да върне)ВАЖНО!!!ПРИ ОТРИЦАТЕЛЕН ЗНАК НА НАЧАЛНА ПОЗИЦИЯ БРОИ ОТЗАД НАПРЕД**
INSTR(txt, s[, p])	връща позицията на s в txt, започвайки от позиция p
UPPER(txt)/LOWER(txt)	преобразува в главни/малки букви txt
REPLACE(txt,v,s)	замества v в txt
REGEXP_REPLACE(txt,'reg','with')	замества с помощта на регулярен израз

REGEXP_COUNTH	Брои срещания на шаблона
REGEXP_INSTR	Текст в които търсим, какво търсим, от коя позиция да започне да брои, след кое срещане. При отрицателна позиция започва да брои от края
LENGTH	ВРЪЩА ДЪЛЖИНА НА ТЕКСТ
LOWER	Връща малки букви
UPPER	Големи букви
TRIM	В класически вид реже интервали, но може да реже и други символи. Какво да реже - Колона, leading (отпред)/ traling (отзад)/ both (двете страни)/, символ FROM откъде да реже
RTRIM реже от дясно. LTRIM (n[,m] реже от ляво	

```sql
-- Maria Anders -> M. Anders
--Maria Anders -> M. Anders
SELECT contact_name
, SUBSTR(contact_name,1,1)--връща първа буква от името, SUBSTR(contact_name,1,M) търсим имена, които започват с M
 || '.' -- конкатенираме, влагаме фукция в полето което търсим
 || SUBSTR(contact_name, INSTR(TRIM(contact_name), ' ', -1)) Contact -- -1 за да брой отзад напред, ' ' - първи интервал. Търсим първия интервал отзад напред
FROM customers

---
--Maria Anders -> M. Anders
SELECT contact_name
, REGEXP_REPLACE(contact_name, '^(\w).+\s(\w+)\s?$', '\1. \2') Contact -- '\1. \2' обръщаме се към 1-ви и 2-ри кръгли скоби
FROM customers
      -- s? - 0 или 1 интервали/ +. И +* Замества всички символи, поглъща всичко и можем да го спрем със \s (интервал)
```

```

# ЛЕКЦИЯ 5

* дати – могат да се ползват на всеки ред от заявката. Служат за трансформация на данни

| INTERVAL 'expr' qualifier | използва се в изрази с дати |
|---|---|
| NEXT_DAY(date,weekday ) | връща първата дата на деня от седмицата след date |
| LAST_DAY(date) | последният ден от месеца |
| MONTHS_BETWEEN(date1,date2) | брой месеци между date1 и date2 |
| ADD_MONTHS(date,n) | добавя n месеца към датата date |

| EXTRACT(какво FROM коя таблица) | вади отделни части от дати,год, месец, ден, час и т.н. |
|---|---|
| dateif | Връща брой дни между две дати |

**ROWNUM – Връща номера на редовете**
**SYSDATE –псевдо колона.** Специален тип колони, дефинирани на глобално ниво и са част от всяка една таблица

```sql
MS SQL, Sybase: SELECT YEAR(order_date), MONTH(order_date), DAY(order_date) --вади отделни части от дати, в други сървъри, където няма тези функции
```

Не всички сървъри позволяват да извършваме аритметични операции с дати

```sql
 ● ВАДИМ МЕСЕЦА И ГОДИНАТА ПРЕЗ 2015-ТА
SELECT order_id
, order_date
, EXTRACT(YEAR FROM order_date) Year
, EXTRACT(MONTH FROM order_date) Month
FROM orders
WHERE EXTRACT(YEAR FROM order_date) = 2015 --вадим годината, като задаваме и критерии
```
```

- За изчисляване на дни. не е добър вариант, защото не работи на всички сървъри. ROWNUM
 Връща номер на реда, но има особености

```
SELECT ROWNUM
, order_id
, order_date
, SYSDATE
, TRUNC(SYSDATE - order_date) Days -режем датата, за да се получи цяло число
FROM orders
WHERE EXTRACT(YEAR FROM order_date) = 2015 AND ROWNUM <= 2 с = дава грешки
```

- на каква възраст са били служителите към момента на постъпване на работа

```
SELECT firstname || ' ' || lastname Employee
, FLOOR(MONTHS_BETWEEN(hiredate, birthdate)/12) Age
FROM employees
```

* функции за конвертиране от един тип в друг

TO_CHAR(v[, fmt])	конвертира до текст, но не и символ (ст-ст, формат) формата не е задължителен
TO_DATE(v[, fmt])	конвертира до дата,(ст-ст, формат) формата не е задължителен
TO_NUMBER(v[, fmt])	конвертира до число (ст-ст, формат) формата не е задължителен

[FMT](https://docs.oracle.com/en/database/oracle/oracle-database/18/sqlrf/Format-Models.html#GUID-24E16D8D-25E4-4BD3-A38D-CE1399F2897C)

```sql
SYSDATE е датата от компютъра/сървъра. Тази функция е много полезна, ако колоната с датата е зададена като текст. Това може да се провери с describe
DUAL - за екперимент. Въвеждаме нещо и искаме да видим какво се връща

SELECT SYSDATE - TO_DATE('12.08.2019','DD.MM.RRRR')
FROM DUAL
---
SELECT order_id
, order_date
, TO_CHAR(order_date,'Q') Qtr
FROM orders
```

● В КОЕ ТРИМЕСЕЧИЕ ОТ ГОДИНАТА СА НАПРАВЕНИ ПОРЪЧКИТЕ/Q (QUARTER) – ЧЕТВЪРТ ЗА ТРИМЕСЕЧИЕ. Трябва да има достатъчно 9-ки за цялата част, иначе излизат #

Настройват се от tools -→ pref -→ nls -→ currency

tools -→ pref -→ nls -→ iso currency

L – лв.
C- bgn
$ - долар
0 – слага 0 отпред/ отзад

DO ден от месец
DDD ден от година
DY абревиатура за ден
● вадим цената закръглена до втори знак

```
SELECT product_name
, unit_price
, TO_CHAR( unit_price * 1.95583,'9999.99L') PriceBGN
FROM products
```

Агрегатни функции

* изискват групиране на данните за да работят правилно

**При групиране на данни, сървърът сравнява данните в съответните колони
по редове и ако открие два или повече реда със съвпадащи стойности, представя
редовете като група, оставяйки в резултат само един от съвпадащите редове.**ю
Когато напишем заявка и С (сървър) вади данните повлияни от заявката, но преди да ги покаже в резултата започва да сравнява ст-стите в колоната по съответните редове, ако открие два или повече реда, при които ст-стите съвпадат ги групира, като 1 ред. След като ги групира се включва функцията, която извършва изчисленията .

Как да групираме данните? е най-важният въпрос, на който трябва да отговорим,
преди да използваме която и да е от агрегатните функции.
MySql е сървър, които е спец. Създаден да работи в интернет, но не се използва в банки
Пример: В колона [3] всички стойности по редовете са уникални т.е. групиране не може да се
получи => използването, на която и да е от агрегатните функции няма да върне полезен

резултат.

[1]	[2]	[3]
a	b	d
a	b	e
a	c	f
a	c	g

!!!!! В GROUP ТРЯБВА ДА ВКЛЮЧИМ МИН. ОНЕЗИ РЕДОВЕ, КОИТО СА ВЛЕЗЛИ В SELECT, НО БЕЗ АГРЕГИРАЩИТЕ ФУНКЦИИ. АКО ИМА КОЛОНА С ДР. ВИД ФУНКЦИЯ, ТЯ СЪЩО ТРЯБВА ДА ВЛЕЗЕ В GROUP . АКО НЯМА ГРУПИРАНЕ РЕЗУЛТАТА НЕ Е ПОЛЕЗЕН, ГРУПА ОТ 1 Е БЕЗМИСЛЕНА
ОТ ГРУПИРАНЕТО ЗАВИСИ РЕЗУЛТАТА И СЕ ПРОМЕНЯ СМИСЪЛА НА ЗАЯВКАТА
РК – ИМА У ИКАЛНА СТ-СТ И ПРЕЧИ ПРИ ГРУПИРАНЕ

```sql
SELECT [1],[2],[3], COUNT(*)
FROM ...
GROUP BY [1],[2],[3]
```

GROUP СТОИ СЛЕД WHERE ИЛИ FROM, АКО НЯМА WHERE КЛАУЗА

Пример: Първите два реда и последните два реда имат еднакви стойности => имаме две групи с по два реда.

[1]	[2]
a	b
a	b
a	c
a	c

```sql
SELECT [1],[2], COUNT(*)
FROM ...
GROUP BY [1],[2]
```

```
+----+---+---+
| a  | b | 2 |
+----+---+---+
| a  | c | 2 |
+----+---+---+
```

Пример: Всички редове съдържат една и съща стойност => имаме 1 група от 4 реда

```
+-----+
| [1] |
+-----+
| a   |
+-----+
| a   |
+-----+
| a   |
+-----+
| a   |
+-----+
```

```sql
SELECT [1], COUNT(*)
FROM ...
GROUP BY [1]
```

```
+----+---+
| a  | 4 |
+----+---+
```

* основни АГРЕГАТНИ функции

COUNT()	брои редове
AVG()	изчислява средно аритметично
SUM()	сумира
MIN()	минимум
MAX()	максимум

● _____

БРОЙ ПОРЪЧКИ НА ALFKI

```sql
SELECT customer_id
, COUNT(*) NOrders --* БРОИ РЕДОВЕ. В СЛУЧАЯ БРОЯ ПОВТОРЕНИЯ СЪОТВЕТСТВА НА БРОЯ ПОРЪЧКИ
FROM orders
WHERE customer_id = 1 -- ALFKI МОЖЕМ ДА СЛАГАМЕ ОГРАНИЧЕНИЯ ПО КОЛОНА, КОЯТО НЕ ПРИСЪСТВА В СЕЛЕКТ,ВАЖИ И ЗА ГРУПИРАНЕТО
GROUP BY customer_id
```

```sql
● БРОЙ ПОРЪЧКИ НА ВСЕКИ ЕДИН КЛИЕНТ, В ПОВЕЧЕТО СЛУЧАИ РК ПРЕЧИ НА ГРУПИРАНЕТО
SELECT customer_id --КОЙ КОЛКО ПОРЪЧКИ ИМА
, COUNT(*) NOrders -- ПСЕВДОНИМ
FROM orders
--WHERE customer_id = 1 -- ALFKI ПРАВИМ ПРОВЕРКА С 1 КЛИЕНТ. БОЛИД ИМА 8 ПОРЪЧКИ И 8 ПРОДУКТА
GROUP BY customer_id
-- МОЖЕМ ДА ГРУПИРАМЕ ПО КОЛОНА, КОЯТНО НЕ ПРИСТСЪВА В СЕЛЕКТ, ЩЕ ВИЖДАМЕ РЕЗУЛТАТА, НО НЯМА ДА ВИЖДАМЕ СТ-СТА В САМАТА КОЛ.
```

* филтриране на резултати от агрегираща функция

ЗА АГРЕГИРАЩИТЕ ФУНКЦИИ НЕ СЕ ИЗПОЛЗВА WHERE, ЗА ТЯХ ИМА СПЕЦИАЛНА КЛАУЗА - HAVING, КОЯТО СЕ ИЗПОЛЗВА САМО ЗА ТЯХ, ЗАЩОТО ФИЛРИРАНЕТО СТАВА НА МНОГО ПО-КЪСЕН ЕТАП.
HAVING НЕ ПОЗВОЛЯВА ДА ВИКАМЕ С ПСЕВДОНИМИ, ТРЯБВА ДА КОПИРАМЕ ФУНКЦИЯТА ОТ SELECT

●
```sql
A { 'a','b',NULL,NULL,'c'} A- КОЛОНА/ { 'a','b',NULL,NULL,'c' - СТ-СТИ В КОЛОНА

COUNT(*) -> 5 БРОИ РЕДОВЕ, НЕЗАВИСМО, ДАЛИ СА ПРАЗНИ ИЛИ НЕ!!!! АКО КЛИЕНТА ИМА ДВА НУЛЕВИ РЕДА, COUNT ЩЕ ГИ ПРЕБРОИ
COUNT(A) -> 3 ( брои <> NULL) НО АКО СЛОЖИМ ИМЕТО НА КОЛОНОТА, ТО ЩЕ ПРЕБРОИ САМО КОЛОНИТЕ СЪС СТ-СТ, БЕЗ НУЛЕВИТЕ
ТОЕСТ БРОИ РАЗЛИЧНИТЕ ОТ NULL СТ-СТИ В КОЛОНАТА
ВАЖНО!!!! ВАЖНО Е КОЛОНАТА В АРГУМЕНТА ДА Е ЗАДЪЛЖИТЕЛНА И ДА НЕ ЗАБРАВИМ ДА ПРОМЕНИМ АРГУМЕНТА НА COUNT
```

● ВАДИМ ВСИЧКИ КЛИЕНТИ, КОИТО ИМАТ ПО-МАЛКО ОТ 5 ПОРЪЧКИ.ТУК СЪЩО НЕ ИЗЛИЗАТ ОНЕЗИ С НУЛЕВИ ПОРЪЧКИ
```sql
SELECT [1],[2], COUNT(*)
FROM ...
GROUP BY [1], [2] HAVING COUNT(*) < 5
```

●
```sql ВАДИМ ВСИЧКИ КЛИЕНТИ,КОИТО ИМАТ ПО-МАЛКО ОТ 3 ПОРЪЧКИ,НЕ ВРЪЩА ТЕЗИ БЕЗ ПОР-КА,ЗАЩОТО В ТАЗИ ТАБЛИЦА НЯМА ДАННИ ЗА ТЯХ
SELECT customer_id
, COUNT(*) NOrders
FROM orders
--WHERE customer_id = 1 -- ALFKI
GROUP BY customer_id HAVING COUNT(*) < 3 -- HAVING COUNT СЕ ПОЛЗВА САМО ЗА ФИЛТРИРАНЕ  НА РЕЗУЛТАТИ ОТ АГРЕГ. ФУНКЦИИ.
НЕ СЕ ПОДДЪРЖА ПСЕВДОНИМ, СЛАГАМЕ ФУНКЦИЯТА
```

● _____

ПРАВИЛЕН НАЧИН ЗА ВАДЕНЕ ВКЛ. НА КЛИЕНТИ БЕЗ ПОРЪЧКА. ВАДИМ ВСИЧКИ КЛИЕНТИ, ВКЛ. ОНЕЗИ БЕЗ ПОРЪЧКА, В СЛУЧАЯ С ПО-МАЛКО ОТ 3 ПОРЪЧКИ

АКО В COUNT ОСТАНЕ (*) ЗА КЛИЕНТИТЕ БЕЗ ПОРЪЧКА ЩЕ ВЪРНЕ 1, ЗАЩОТО COUNT БРОИ РЕДОВЕ, А ТЕИМАТ ПО 1 РЕД ЗАПИС И НЕ ОТЧИТА, ЧЕ ТОЙ Е NULL, ЗАТОВА Е ВАЖНО ДА СЛОЖИМ ИМЕТО НА КОЛОНАТА В АРГУМЕНТ

```sql
SELECT t1.customer_id
, t1.customer_code
, COUNT(t2.order_id) NOrders --МОЖЕ ДА СЕ СЛОЖИ PK
FROM
    customers t1
        LEFT OUTER JOIN
    orders t2
        ON t2.customer_id = t1.customer_id
--WHERE customer_id = 1 -- ALFKI
GROUP BY t1.customer_id --ВАЖНО Е ДА НЕ ЗАБРАВИМ ДА СЛОЖИМ В ГРУПИРАНЕТО СЪЩИТЕ КОЛОНИ ОТ СЕЛЕКТ
        ,t1.customer_code
HAVING COUNT(t2.order_id) < 3 ---НЕ ПОДДЪРЖА ПСЕВДОНИМ
```

● ГРЕШЕН НАЧИН ЗА ФИЛТРИРАНЕ - COUNT в SELECT се различава от COUNT в HAVING !!!!.

НЕ ИЗВЕЖДА НИЩО ЗА КЛИЕНТИ С 0 ПОРЪЧКИ, ЗАЩОТО БРОИ РЕДОВЕ

```sql
SELECT t1.customer_id
, t1.customer_code
, COUNT(t2.order_id) NOrders
FROM
    customers t1
        LEFT OUTER JOIN
    orders t2
        ON t2.customer_id = t1.customer_id
--WHERE customer_id = 1 -- ALFKI
GROUP BY t1.customer_id
        ,t1.customer_code
HAVING COUNT(*) = 0 -- COUNT в SELECT се различава от COUNT в HAVING !!!!
```

АКО СМЕ МАХНАЛИ ВСИЧКИ ПРЕЧЕЩИ КОЛОНИ И НЯМА НИТО ЕДНА КОЛОНА ОТ ТАБЛИЦАТА, ТО ТРЯБВА ДА МАХНЕМ И ТАБЛИЦАТА, ЗАЩОТО ТОВАРИ

● ВАДИМ ВСИЧКИ ПОРЪЧКИ И НА КАКВА ОБЩА СТ-СТ Е ВСЯКА ПОРЪЧКА И БРОЙ ПРОДУКТИ

```
SELECT t1.customer_id --НА КАКВА ОБЩА СТ-СТ Е ВСЯКА ПОРЪЧКА
, t1.order_id
, t1.order_date
, SUM(t2.unit_price * t2.quantity) total --СУМА НА ВСЯКА ПОРЪЧКА
, COUNT(*) NProducts --БРОЙ ПРОДУКТИ В ПОРЪЧКАТА
FROM orders t1 --- ЗАПОЧВАМЕ С ОРДЕРС ЗАЩОТО НЯМА ДА ВАДИМ  ДАННИ ЗА САМИЯ КЛИЕНТ
        INNER JOIN
     order_details t2
        ON t2.order_id = t1.order_id
WHERE t1.customer_id = 1 - ALFKI --- АКО ЗАКОМЕНТИРАМЕ АЛКФКИ ЩЕ ВИЖДАМЕ ОБЩА СТ-СТ ЗА ВСЕКИ КЛИЕНТ
GROUP BY  t1.customer_id
        , t1.order_id
        , t1.order_date
ORDER BY 2
```

● ВАДИМ ВСИЧКИ ПОРЪЧКИ НА АЛКФКИ И ТЯХНАТА ОБЩА СТ-СТ И БРОЙ ПРОДУКТИ, НО ЗАКРЪГЛЕНА ДО 2-РИ ЗНАК СЛЕД ЗАПЕТАЯ И ЛЕВА НАКРАЯ

ТЕЗИ ФОРМАТИРАНИЯ ЧЕСТО ПРЕЧАТ!

```
SELECT t1.customer_id
, t1.order_id
, t1.order_date
, TO_CHAR(SUM(t2.unit_price * t2.quantity),'9999.99L') total
, COUNT(*) NProducts
FROM orders t1
        INNER JOIN
     order_details t2
        ON t2.order_id = t1.order_id
WHERE t1.customer_id = 1 -- ALFKI
GROUP BY  t1.customer_id
        , t1.order_id
        , t1.order_date
ORDER BY 2
```

```
ОБЩА СТ-СТ НА ПОРЪЧКИТЕ НА ALFKI И КОЛКО НА БРОЙ ПОРЪЧКИ ИМА
SELECT t1.customer_id
, SUM(t2.unit_price * t2.quantity) total
, COUNT(*) NProducts
FROM orders t1
        INNER JOIN
     order_details t2
        ON t2.order_id = t1.order_id
WHERE t1.customer_id = 1 -- ALFKI
GROUP BY  t1.customer_id
ORDER BY 2
```

ЛЕКЦИЯ 6

Вложени SQL заявки

Вложените SQL заявки представляват мощно средство за извличане на данни

* FROM клауза
* WHERE клауза
* SELECT клауза

1. ### Вложени заявки във FROM (inline view)

*** ограничения:**
+ ВЛОЖЕНИТЕ задължително трябва да имат псевдоним НА САМАТА ТАБЛИЦА, КАКТО И НА ИЗЧИСЛИМИТЕ КОЛОНИ
+ в основната заявка могат да се достъпват само колоните изброени в SELECT

!!!! ВЪНШНАТА ЗАЯВКА ВЗИМА ДАННИ ОТ ВЛОЖЕНАТА!!!!!!!!!!!!!!!!
ВЪВ ВЪНШЕН СЕЛЕКТ МОГАТ ДА СЕ ВИКАТ САМО ОНЕЗИ КОЛОНИ, КОИТО ПРИСЪСТВАТ В СЕЛЕКТ НА ВЛОЖЕНАТА ЗАЯВКА

ВЛОЖЕНИТЕ ВЪВ ФРОМ ЗАЯВКИ ДАВАТ НАЙ-ГОЛЯМА СВОБОДА, ЗАЩОТО ПЪРВАО СЕ ИЗПЪЛНЯВА ФРОМ КЛАУЗАТА И СЪОТВЕТНО ВЛОЖЕНАТА ЗАЯВКА
Вложените във FROM заявки са подобни на изгледите (view) с тази разлика, че изгледът е обект в базата данни и се създава с `CREATE VIEW` заявка. Кода директно е забит в завката. Изгледите се правят за компактност и защото адмиите не дават права за цяла таблица, а до изглед. ИЗГЛЕДИТЕ НЕ СЪДЪРЖАТ ДАННИТЕ. ТОЙ ВСЕКИ ПЪТ СЕ ОБРЪЩА КЪМ ТАБЛИЦАТА НА СЪРВЪРА.
(northwind -→ views) sql заявка запомнена на сървъра
Материализираните изгледи съчетават данни от няколко таблици и съдържат данни, които рядко се променят. Те съдържат в себе си данните ик се рефрешват през определен период от време, това ускорява извличането на данните
РЕД НА ИЗПЪЛНЕНИЕ
1. **ПЪРВО СЕ ИЗПЪЛНЯВА ВЛОЖЕНАТА ЗАЯВКА**
2. **WHERE**
3. **SELECT**

● _____

```sql
SELECT ...
FROM (
        SELECT ...
        FROM ...
      ) alias
```

● **ДИРЕКТНО ИНФО ЗА ПОРЪЧКИТЕ НЯМА, ЗАТОВА С ВЛОЖЕНА ТАБЛИЦА ВАДИМ ВСИЧКИ ПОРЪЧКИ И ПРОДУКТИ** НА АЛФКИ

```
Round, order by, to char задължително се слагат във външната заявка.НЕ СЕ СЛАГАТ ВЪВ ВЪТРЕШНИТЕ
 Вложено може да се СВЪРЗВАТ/JOIN таблици

SELECT t2.customer_code --самостоятелна колона, задължително се групира
, MIN(vt.total) MinT --мин. ст-ст на поръчка. . Изчислима колона. Задължителен псевдоним
, TO_CHAR(AVG(vt.total),'9999.99') AvgT . --средно е пазарувал на ст-ст/ Изчислима колона. Задължителен псевдоним
, MAX(vt.total) MaxT --макс. ст-ст на поръчка. Изчислима колона. Задължителен псевдоним
, MIN(vt.nproducts) MinNP --купувал най-малко 1 продукт. Изчислима колона. Задължителен псевдоним
, TO_CHAR(AVG(vt.nproducts),'9999.99') AvgNP --Средно е купувал по 2 продукта.Изчислима колона. Задължителен псевдоним
, MAX(vt.nproducts) MaxNP --Най-много е купувал 3 продукта. Изчислима колона. Задължителен псевдоним
, SUM(vt.total) SumT --Обща ст-ст на която е пазарувал. Изчислима колона. Задължителен псевдоним
, SUM(vt.nproducts) SumNP --Общ брой закупени продукти.Изчислима колона. Задължителен псевдоним
, COUNT(*) NOrders -- Общ брой поръчки. Изчислима колона. Задължителен псевдоним. Брои по онова, което сме групирали
FROM ( -- вложена заявка. ВАДИМ ПОРЪЧКИТЕ НА АЛФКИ, ТЯХНАТА СТ-СТ И КОЛКО ПРОДУКТА ИМА В ТЯХ
        SELECT t1.customer_id -- във външната колона са достъпни само тези колони в селект тук
        , t1.order_id
        , t1.order_date
        , SUM(t2.unit_price * t2.quantity) total --ЗАДЪЛЖИТЕЛЕН ПСЕВДОНИМ НА ИЗЧИСЛИМИТЕ КОЛОНИ
        , COUNT(*) NProducts
        FROM orders t1
                INNER JOIN
             order_details t2 -ПСЕВДОНИМИТЕ НА ТАБЛИЦИТЕ НЕ СЕ ВИЖДАТ ОТВЪН, ВИЖДА СЕ КРАЙНИЯ ПСЕВДОНИИМ - VT
                ON t2.order_id = t1.order_id
        --WHERE t1.customer_id = 1 -- ALFKI --- ЗА ДА ВИЖДАМЕ ВСИЧКИ КЛЕИНТИ, ТОВА Е ЗА ПРОВЕРКА
        GROUP BY  t1.customer_id
                , t1.order_id
                , t1.order_date
     ) vt - задължителен псевдоним на вложените заявки
        INNER JOIN --- САМО ЗА ДА ВИДИМ ИМЕТО НА КЛИЕНТА
     customers t2
        ON vt.customer_id = t2.customer_id
GROUP BY t2.customer_code
ORDER BY 1
```

дублиране на редове и местене се настройва от tools-pref..-keys

SHIFT + TAB – ОБРАТНА ТАБУЛАЦИЯ

ctrl + / – коментар

СЪЗДАВАНЕ НА ИЗГЛЕД

customers_orders_summary -- Е ИМЕТО КОЕТО ЗАДАВАМЕ НА ИЗГЛЕДА
SQL ЗАЯВКИ В ИЗГЛЕДИТЕ МОГАТ ДА БЪДАТ САМО В SELECT.ИЗГЛЕДИТЕ СЕ СЪХРАНЯВАТ В VIEWS И ВИНАГИ ВРЪЩАТ АКТУАЛНИ ДАННИ, ЗАЩОТО ЗАЯВКАТА Е ЗАПИСАНА НА СЪРВЪРА. КЪМ ИЗГЛЕДИТЕ МОЖЕМ ДА СВЪРЗВАМЕ ДРУГИ ТАБЛИЦИ, АКО ИМАТ ОБЩА КОЛОНА
НАЙ-УДОБНО Е ДА ИЗПОЛЗВА CREATE OR REPLACE VIEW, ЗАЩОТО АКО СЛОЖИМ САМО CREATE И СЕ НАЛОЖИ ПРОМЯНА ТРЯБВА ДА ЗАМЕНЯМЕ CREATE С OUTHER VIEW, ИМЕТО И НОВИТЕ НЕЩА, А ТАКА СЪЗДАВА И ЗАМЕСТВА.ПРИЛОЖЕНИЯТА РАБОТЯТ С ИЗГЛЕДИ, НО АКО ДОБАВИМ НОВА В КОЛОНА В ТАБЛИЦАТА, ИЗГЛЕДА НЯМА ДА Я ВИЖДА

●

```sql
CREATE OR REPLACE VIEW customers_orders_summary
AS
    SELECT t2.customer_code
    , MIN(vt.total) MinT
    , TO_CHAR(AVG(vt.total),'9999.99') AvgT
    , MAX(vt.total) MaxT
    , MIN(vt.nproducts) MinNP
    , TO_CHAR(AVG(vt.nproducts),'9999.99') AvgNP
    , MAX(vt.nproducts) MaxNP
    , SUM(vt.total) SumT
    , SUM(vt.nproducts) SumNP
    , COUNT(*) NOrders
    FROM (
            SELECT t1.customer_id
            , t1.order_id
            , t1.order_date
            , SUM(t2.unit_price * t2.quantity) total
            , COUNT(*) NProducts
            FROM orders t1
                    INNER JOIN
                order_details t2
                    ON t2.order_id = t1.order_id
            --WHERE t1.customer_id = 1 -- ALFKI
            GROUP BY  t1.customer_id
                    , t1.order_id
                    , t1.order_date
        ) vt
            INNER JOIN
        customers t2
            ON vt.customer_id = t2.customer_id
    GROUP BY t2.customer_code
    ORDER BY 1
```

ВИКАНЕ НА ИЗГЛЕД

ТЪРСИМ ВСИЧКИ КЛИЕНТИ,ПРИ КОИТО ИМА ПО-МАЛКО ОТ 10 ПРОДУКТА ОБЩО В ПОРЪЧКИТЕ СИ

```
● SELECT *
FROM customers_orders_summary -- викаме изглед ОТ ГОРНАТА ЗАДАЧА
WHERE sumnp < 10
ORDER BY 1
```

* **WITH клауза** КЪМ ВЛОЖЕНИТЕ ВЪВ FROM ЗАЯВКИ

- позволява основната заявка да стане по-прегледна. CTE - Common Table Expressions

ВАРИАНТ 1: ДВЕ ВЛОЖЕНИ ЗАЯВКИ, НЕЗАВИСИМИ ЕДНА ОТ ДРУГА

```sql
WITH q1 AS ( SELECT ... FROM ... )
,    q2 AS ( SELECT ... FROM ...  )
SELECT ...
FROM q1 , q2

```

ВАРИАНТ 2: ВТОРАТА ВЛОЖЕНА ЗАЯВКА ЧЕРПИ ИНФОРМАЦИЯ ОТ ПЪРВАТА

```sql
WITH q1 AS (SELECT ... FROM ... )
,    q2 AS (SELECT ... FROM q1  )
,    q3 AS (SELECT ... FROM q2  )
SELECT ...
FROM q3
```

```
-- същото като ВАРИАНТ 2, НО НЕУДОБНО И НЕПРИГЛЕДНО
SELECT ...
FROM (
        SELECT ...
        FROM (
                SELECT ...
                FROM (
                      SELECT ...
                      FROM ...
                    ) q1
             )  q2
     ) q3
```

ЦЯЛОТО ВЛОЖЕНО СЪДЪРЖАНИЕ ОТ FROM СЕ ИЗТЕГЛЯ ВЪВ WITH И МУ СЕ ДАВА ПСЕВДОНИМ, КОИТО ПОЛСЕ СЕ СЛАГА ВЪВ FROM, ЗА ДА СЕ ИЗВИКА ВЛОЖЕНАТА ЗАЯВКА АБСОЛЮТНО СЪЩОТО Е, КАТО ДА СЕ ЗАПИШЕ ВЪВ FROM (ДА СЕ ВЛОЖИ ЗАЯВКА ВЪВ ФРОМ), НО Е МНОГО ПО-УДОБНО И ПРИГЛЕДНО

ОТНОВО ВАДИ ВСИЧКИ ПОРЪЧКИ И ПРОДУКТИ НА ВСИЧКИ КИЕНТИ

```
WITH vt AS (
            SELECT t1.customer_id
            , t1.order_id
            , t1.order_date
            , SUM(t2.unit_price * t2.quantity) total
            , COUNT(*) NProducts
            FROM orders t1
                    INNER JOIN
                 order_details t2
                    ON t2.order_id = t1.order_id
            --WHERE t1.customer_id = 1 -- ALFKI
            GROUP BY  t1.customer_id
                    , t1.order_id
                    , t1.order_date)
SELECT t2.customer_code
, MIN(vt.total) MinT
, TO_CHAR(AVG(vt.total),'9999.99') AvgT
, MAX(vt.total) MaxT
, MIN(vt.nproducts) MinNP
, TO_CHAR(AVG(vt.nproducts),'9999.99') AvgNP
, MAX(vt.nproducts) MaxNP
, SUM(vt.total) SumT
, SUM(vt.nproducts) SumNP
, COUNT(*) NOrders
FROM vt ---ПСЕВДОНИМ НА ВЛОЖЕНАТА ЗАЯВКА
        INNER JOIN
     customers t2
        ON vt.customer_id = t2.customer_id
GROUP BY t2.customer_code
ORDER BY 1
```

---

2. ### Вложени заявки в WHERE

ТУК ДЕЙСТВАТ ПОВЕЧЕ ОГРАНИЧЕНИЯ, ИЗИСКВАНИЯТА КЪМ ВЛОЖЕНАТА ЗАЯВКА ПРОИЗТИЧАТ ОТ ОПЕРАТОРА, КОИТО ИЗПОЛЗВАМЕ
АКО ОПЕРАТОРА ОЧАКВА 1 СТ-СТ, WHERE КЛАУЗАТА ТРЯБВА ДА БЪДЕ ТАКА НАПИСАНА, ЧЕ ТОЗИ ВЛОЖЕН СЕЛЕКТ ДА ВРЪЩА 1 СТ-СТ
НАПР. ПРИ < И > В СЕЛЕКТ ТРЯБВА ДА ИМА 1 СТ-СТ, КОЯТО ДА БЪДЕ ИЗПОЛЗВАНА ЗА ТЕЗИ СРАВНЕНИЯ.
АКО ОПЕРАТОРА Е **IN СЕ ДОПУСКА МНОЖЕСТВО СТ-СТИ И WHERE ВРЪЩА МНОГО РЕДОВЕ, НО ПАК ТРЯБВА ДА БЪДЕ ПО ЕДНА КОЛОНА**
НЕ СЕ ИЗПОЛЗВАТ LIKE И BETWEEN
!!!! ОБРАТНО НА FROM!!! ВЛОЖЕНАТА ЗАЯВКА МОЖЕ ДА ВЗЕМА ДАННИ ВЪНШНАТА (SUBQUERY FACTORING), НО НЕ И ОБРАТНОТО, ВЪНШНАТА НЕ МОЖЕ ДИРЕКТНО ДА ВЗЕМА ДАННИ ОТ ВЛОЖЕНАТА

*** ограниченния**
Данните във вложената да са съвместими по тип и смисъл с тези отвън
+ SELECT по една колона (с малки изключения)
+ брой редове - според оператора (IN (), >, < и т.н.)

СИНТАКСИС
```sql
SELECT ...
FROM ...
WHERE column operator (SELECT ... FROM ... WHERE ...)
                              --(Вложена заявка)--
-- WHERE колона оператор (SELECT 1 колона FROM няма ограничение WHERE няма ограничение)
```

●_____
ВАДИМ ВСИЧКО ОНЕЗИ ПРОДУКТИ, ЧИИТО ЕДИНИЧНА ЦЕНА Е ПОД СРЕДНАТА ЗА ЦЯЛАТА БАЗА ДАННИ
ОТ ВСИЧКИ ЕД. ЦЕНИ (UNIT PRICE)ТРЯБВА ДА НАМЕРИМ СРЕДНОТО И ПОСЛЕ ДА ИЗВАДИМ СПИСЪК НА ОНЕЗИ ПРОДУКТИ, НА КОИТО ЕД. ЦЕНА Е ПОД СРЕДНАТА, ТЪЙ КАТО ВЛОЖЕНАТА ЗАЯВКА ВРЪЩА 1 СТ-СТ И КОЛОНАТА Е 1 МОЖЕМ ДА ВЛОЖИМ В СЕЛЕКТ
```sql
SELECT product_id
, product_name
, unit_price
FROM products
WHERE unit_price <= ( SELECT AVG(unit_price) -- 1, ВЛОЖЕНАТА ЗАЯВКА Е ПЪРВАТА СТЪПКА,ТЪРСИМ СРЕДНА ЦЕНА (28,86).
                      FROM products )        -- В SELECT ИМА САМО АГРЕГИРАЩА ФУНКЦИЯ ЗАТОВА НЕ Е НУЖНО ГРУПИРАНЕ
ORDER BY 3 DESC
-- ГРЕШНО Е ДА ЗАПИШЕН ЧИСЛО, ВМЕСТО ФУНКЦИЯ СЪС СРЕДНО, ЗАЩОТО СРЕДНАТА ЦЕНА Е НЕЩО, КОЕТО СЕ ПРОМЕНЯ
```

- **ВАДИМ ПОРЪЧКИТЕ САМО НА ВЛИЕНТИТЕ ОТ ГЕРМАНИЯ**

```
SELECT customer_id ---ИД НА КЛИЕНТ
, order_id
, order_date
, ship_country --КЪДЕ Е ДОСТАВЕНА ПОРЪЧКА
FROM orders
WHERE customer_id IN (SELECT customer_id -- СТЪПКА 1, ПОЛУЧВАМЕ ИД-ТАТА НА КЛИЕНТИТЕ ОТ ГЕРМАНИЯ
                      FROM customers   --!!!ЗАБЕЛЕЖИ, ЧЕ ВЗИМАМЕ ИНФО ОТ РАЗЛИЧНИ ТАБЛИЦИ, БЕЗ СВЪРЗВАНЕ!!!!
                      WHERE country = 'Germany')
ORDER BY 1
```

- **СПИСЪК ПРОДУКТИ С ЦЕНА ПОД СРЕДНАТА ЗА КАТЕГОРИЯТА**

ЦРЕДАТА ЦЕНА Е НЕЩО КОВАРНО, ЗАЩОТО СЕ ВЛИАЕ ОТ НЕОБИЧАЙНО ВИСОКИ И НИСКИ ЦЕНИ/НАПР. БАТЕРИИ И ПЛАЗМЕН ТЕЛЕВИЗОР. МНОГО ПРОДУКТИ НЯМА ДА ВЛЯЗАТ, ЗАРАДИ ТЕЗИ ДВЕ НЕОБИЧНО ВИСОИ И НИСКИ ЦЕМИ, ЗАТОВА Е КОРЕКТНО ДА ИЗВАДИМ СРЕДНАТА ЦЕНА НЕ ЗА ЦЯЛАТА БАЗА ДАННИ, А ПО КАТЕГОРИИ

```
SELECT t1.category_id
, t1.category_name
, t2.product_name
, t2.unit_price
FROM categories t1
        INNER JOIN
     products t2
        ON t2.category_id = t1.category_id
WHERE t2.unit_price <= ( SELECT AVG(ns.unit_price)
                         FROM products ns --ДАВАМЕ ПСЕВДОНИМ НА ТАБЛИЦАТА
                         WHERE ns.category_id = t1.category_id)--ДАВАМЕ СТ-СТ НА КАТЕГОРИ ИД, ЧИЯТО СТ-СТ ДА ПОЛЗВАМ ОТВЪН
ORDER BY 1,4 DESC
```

- **СПИСЪК ПРОДУКТИ С ЦЕНА ПОД СРЕДНАТА ЗА КАТЕГОРИЯТА, НО ИЗЧИСЛЕНО С МЕДИАНА**

МЕДИАНАТА - РАЗДЕЛЯ СПИСЪКА НА ПОЛОВИНА И СЕ ПОЛЗВА, ЗА ДА СЕ ЕЛИМИНБИРАТ НЕОБИЧАЙНО ВИСОКИ И НЕОБИЧАЙНО НИСКИ СТ-СТИ

```
SELECT t1.category_id
, t1.category_name
, t2.product_name
, t2.unit_price
FROM categories t1
        INNER JOIN
     products t2
        ON t2.category_id = t1.category_id
WHERE t2.unit_price <= ( SELECT MEDIAN(ns.unit_price)
                         FROM products ns
                         WHERE ns.category_id = t1.category_id)
```

3. ### Вложени заявки в SELECT

```sql
SELECT column1
, column2
, (SELECT column FROM table_a WHERE conditions )
--(SELECT 1 КОЛОНА И ДА ВРЪЩА 1 СТ-СТ FROM няма ограничение WHERE няма ограничение )
FROM ...
```

*** ограничения**
+ вложената заявка трябва да връща само 1 ред и да съдържа само 1 колона

ПОДОБНО НА ПИВОТ ТАБЛИЦА, ПОЗВОЛЯВА НИ ДА ДОБАВИМ НОВИ КОЛОНИ, ДА СЪДАВАМЕ НОВИ КОЛОНИ С ИЗЧИСЛИМИ ДАННИ
ИСКАМЕ ДА ИЗВАДИМ КОЙ КЛИЕНТ, КОЛКО ПОРЪЧКИ ИМА В ГОДИНА

ТУК СЪЩО ВАЖИ SUBSQUERY FACTORING - **ВЛОЖЕНАТА ЗАЯВКА ВЗИМА ДАННИ ОТ ВЪНШНАТА**, НО ОБРАТНОТО НЕ ВАЖИ

COUNT(*) БРОИ ПОРЪЧКИТЕ НА КЛИЕНТ, НО ЗА ЦЕЛИЯ ПЕРИОД ОТ ВРЕМЕ, АКО ИСКАМЕ ДА Е ЗА ОПРЕДЕЛЕН ПЕРИОД, ТРЯБВА ДОПЪЛНИТЕЛНО ДА ГО ЗАДАДЕМ,В СЛУЧАЯ С ТАЗИ ЧАСТ EXTRACT(YEAR FROM ns.order_date) = 2014) "2014"

2014 НЕ Е ВАЛИДЕН ПСЕВДОНИМ,ЗА ДА Е ВАЛИДЕН ТРЯБВА ДА ИМА БУКВА ОТПРЕД,НЕ ТРЯБВА ДА ИМА ИНТЕРВАЛИ, ТИРЕТА, СКОБИ, КИРИЛИЦИ И Т.Н.АКО ДЪРЖИМ ДА Е ТАКА, ТРЯБВА ДА Е В ДВОЙНИ КАВИЧКИ, ТАКИВА НЕЩА СЕ ПРАВЯТ В НАЙ-ВЪНШНАТА ЗАЯВКА, НАЙ-ГОРНОТО НИВО, АКО ИМА ВЛОЖЕНИ, ИНАЧЕ АКО СЕ ОБРЪЩАМЕ КЪМ ТАЗИ КОЛОНА ТРБЯВА ДА БЪДЕ ПСЕВДОНИМ, ДВОЙНИ КАВИЧКИ И НЕВАЛИДЕН ПСЕВДОНИМ 2014, КОЕТО НЕ Е УДОБНО
NS Е ПСЕВДОНИМ НА ТАБЛИЦА ORDERS

ЗАДАЧА 9, ВЛОЖЕНИ ЗАЯВКИ В СЕЛЕКТ

```sql
SELECT t1.company_name -- долу сме му присъединили ст-ст, като сме написали, че е равно на ns.customer_id
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id AND EXTRACT(YEAR FROM ns.order_date) = 2014) "2014"
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id AND EXTRACT(YEAR FROM ns.order_date) = 2015) "2015"
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id AND EXTRACT(YEAR FROM ns.order_date) = 2016) "2016"
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id ) Total
--ХОРИЗОНТАЛЕН ТОТАЛ.АКО ИМАХМЕ ПОВЕЧЕ ГОДИНИ ТРЯБВА ДА Е IN 2014,2015,2016 ИЛИ BETWEEN АКО СА ПОСЛЕДОВАТЕЛНИ ГОДИНИ,
--МОЖЕ ДА СВЕРИМ С АПРИС, КОЙТО НЯМА ПОРЪЧКИ
FROM customers t1
```

UNION

ПОЗВОЛЯВА НИ ДА **ОБЕДИНИЯВАМЕ/ КОМБИНИРАМЕ РЕЗУЛТАТИ** ОТ 2 ИЛИ ПОВЕЧЕ ЗАЯВКИ

С UNION МОЖЕМ ДА ПОСТЪГНЕМ FULL OUTER JOIN, ИМАМЕ ЕДНА ЗАЯВКА, КОЯТО СВЪРЗВА ДВЕ ТАБЛИЦИ С LEFT OUTER JOIN, ПИШЕМ UNION, КОПИРАМЕ Я И БЕЗ ДА ПИПАМЕ НИЩО ДРУГО, САМО ПРОМЕНЯМЕ НА RIGHT

НЯМА ЗНАЧЕНИЕ ОТ КАКВА ТАБЛИЦА ВАДИМ ДАННИ, НЯМА ОГРАНИЧЕНИЕ И ЗА WHERE КЛАУЗАТА

ОГРАНИЧЕНИЯ:
ВЪВ ВЪТРЕШНАТА ЗАЯВКА НЕ МОЖЕ ДА ИМА ORDER BY, МОЖЕ ДА ИМА ORDER BY В НАЙ-ВЪНШНАТА ЗАЯВКА
ЗАЯВКИТЕ ТРЯБВА ДА ИМАТ ЕДНАКЪВ БРОЙ КОЛОНИ, ТИП И СМИСЪЛ, НО МОГАТ ДА НОСЯТ РАЗЛИЧНИ ИМЕНА

КРАЙНИЯ РЕЗУЛТАТА **ВЗИМА ПСЕВДОНИМА НА ПЪРВАТА ТАБЛИЦА** ОТ ЗАЯВКАТА, ЗАТОВА НА 2-РА НЯМА СМИСЪЛ ДА СЕ СЛАГАТ ПСВДОНИМИ ИМА ГО ВЪВ ВСИЧКИ СЪРВЪРИ

```sql
SQL1
UNION [ALL]
SQL2
...
UNION [ALL]
SQLN
```

```
КОЛОНА A СЪС СТ-СТИ {'a','b','c'}
КОЛОНА B СЪС СТ-СТИ {'a', 'd', 'e'}

A UNION B     =>{'a', 'b', 'c', 'd', 'e'} - ОБЕДИНЯВА КОЛОНИ, КАТО ПРЕМАХВА ПОВТОРЕНИЯТА
A UNION ALL B =>{'a', 'a', 'b', 'c', 'd', 'e'} - ОБЕДИНЯВА КОЛОНИ, КАТО ЗАПАЗВА ПОВТОРЕНИЯТА


A INTERSECT B  =>{'a'} ОБЩИТЕ ЕЛЕМЕНТИ,ТЕЗИ КОИТО ГИ ИМА И В ЕДНОТО И В ДРУГОТО
A MINUS B      =>{'b', 'c'} C1 - C2 ГОРНОТО МИНУС ДОЛНОТО, ОНЕЗИ ОТ ГОРНОТО, КОИТО ГИ НЯМА В ДОЛНОТО
```

--ТРЕНАЖОР

```
WITH customers1(cid, firstname, lastname) AS ( --ПОТРЕБИТЕЛСКИ К-Т
    SELECT 11, 'John', 'Doe' FROM DUAL UNION
    SELECT 12, 'Anna', 'Smith' FROM DUAL UNION
    SELECT 13, 'Maria', 'Anders' FROM DUAL UNION
    SELECT 14, 'Markus', 'Handke' FROM DUAL UNION
    SELECT 18, 'Micheal', 'Rotger' FROM DUAL )
, customers2(cid,firstname, lastname) AS ( -- ФИРМЕН КРЕДИТ
    SELECT 11, 'John', 'Doe' FROM DUAL UNION
    SELECT 12, 'Anna', 'Smith' FROM DUAL UNION
    SELECT 16, 'Jane', 'Geiss' FROM DUAL UNION
    SELECT 17, 'Jorg', 'Orlowski' FROM DUAL )
SELECT c1.*
FROM customers1 c1
MINUS
SELECT c2.*
FROM customers2 c2
ORDER BY 1
```

UNION ЩЕ ВЪРНЕ ВСИЧКИ/ UNION ALL - ЩЕ СЕ ПОВТОРЯТ ДЖОН И АННА/

АКО НАПРИМЕР НАПРАВИМ ЗАВВКА ЗА ПОТРЕБИТЕЛИ ТЕГЛИЛИ ПОТРЕБИТ. КРЕДИТ И ТЕГЛИЛИ ФИРМЕН, АКО ГИ ИЗВАДИМ С UNION ALL
И ПОТРЕБИТЕЛЯ СЕ ПОВТОРИ ДВА ПЪТИ, ЗНАЧИ Е ТЕГЛИЛ И ДВАТА

INTERSECT-СЕЧЕНИЕТО НА ДВЕТЕ МНОЖЕСТВА.ТОВА СА САМО ОБЩИТЕ ЕЛЕМЕНТИ,ТЕЗИ КОИТО ГИ ИМА И В ЕДНОТО И В ДРУГОТО,
INTERSECT - ПОДХОДЯЩО ДА ИЗВАДИМ ТЕГЛИЛИТЕ ДВАТА ВИДА КРЕДИТ

MINUS - C1 - C2 ГОРНОТО МИНУС ДОЛНОТО, ОНЕЗИ ОТ ГОРНОТО, КОИТО ГИ НЯМА В ДОЛНОТО
MINUS - АКО ГОРЕ СА ПОТРЕБИТ. ТЕГЛИЛИ ПОТРЕБИТР К-Т, ДОЛУ ТЕГЛИЛИ ФИРМЕНЕН К-Т,
C1 - C2 ЩЕ ИЗВАДИМ ОНЕЗИ КОИТО СА ТЕГЛИЛИ ПОТРЕБИТ, НО НЕ СА ТЕГЛИЛИ ФИРМЕН
C2 - 1ИЗВАДИМ ОНЕЗИ КОИТО СА ТЕГЛИЛИ ФИРМЕН К-Т , НО НЕ СА ТЕГЛИЛИ ПОТРЕБИТЕЛСКИ

MINUS I INTERSECT НЯМА ВЪВ ВСИСКИ СЪРВЪРИ

ПРИМЕР КАК ДА ПОЛУЧИМ MINUS: select ot customers_1 WHERE sid not in select customers2
intersect ..WHERE in select customers2

ЗАДАЧА 10

```
SELECT ROWNUM -- СЛАГАМЕ  ROWNUM, ЗА ДА ПОДРЕДИМ МАЛКО ЗАЯВКАТА, ЗА НОМЕРИРАНЕ НА РЕДОВЕ
, t1.company_name
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id AND EXTRACT(YEAR FROM ns.order_date) = 2014) "2014"
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id AND EXTRACT(YEAR FROM ns.order_date) = 2015) "2015"
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id AND EXTRACT(YEAR FROM ns.order_date) = 2016) "2016"
, (SELECT COUNT(*) FROM orders ns WHERE ns.customer_id = t1.customer_id ) Total --ПОРЪЧКИТЕ НА КЛИЕНТ ЗА ЦЕЛИЯ ПЕРИОД,АКО НЕ УТОЧНИМ
FROM customers t1
UNION
SELECT NULL --СЛЕД КАТО ГОРЕ СМЕ ВКАРАЛИ ROWNUM ЗА ПОДРЕДБА, ТРЯБВА ДА ИЗРАВНИМ БРОЯ КОЛОНИ И ЗАТОВА ДОБАВЯМЕ И ТУК.
----НЕ МОЖЕ ДА Е ПРАЗНА КОЛОНА, ЗАЩОТО ТИПОВЕТЕ ДАННИ ЩЕ БЪДАТ РАЗЛИЧНИ.ROWNUM Е ЧИСЛОВА И ТУК ТРЯБВА ДА Е ТАКАВА
-- МОЖЕ ДА БЪДЕ ИЗЧИСЛИМА КОЛОНА, КОЯТО ДА СМЯТА БРОЯ НА КЛИЕНТИТЕ +1
, 'GTOTAL' --НЕ ИСКАМЕ ДА ИЗЛИЗА ИМЕТО НА ФИРМАТА,НО НЕ МОЖЕМ ДА МАХНЕМ КОЛОНАТА COMPANY NAME,ЗАТОВА ЗАМЕСТВАМЕ С ИНТЕРВАЛ ИЛИ GRAND TOTAL
--НАС НЕ НИ ИНТЕРЕДУВА ВСЕКИ КЛИЕНТ, А ОБЩО ЗА ЦЯЛАТА ГОДИНА, ЗАТОВА МАХАМЕ ПЪРВА ЧАСТ НА WHERE КЛАУЗА(ns.customer_id =
t1.customer_id)
--АКО ИМА ДРУГИ ГОДИНИ, В НАЙ-ПОСЛЕДНАТА ЧАСТ(ТОТАЛ) МОЖЕ ДА ОСТАНЕ WHERE КЛАУЗАТА И ДА ИЗБРОИМ ГОДИНИТЕ
--НЕ ОСТАНА НИЩО, КОЕТО ДА ПОЛЗВЕАМЕ ОТ CUSTOMERS,САМО ОТ ORDERS,ЗАТОВА В ПОСЛЕДНАТА ТОЧКА
, (SELECT COUNT(*) FROM orders ns WHERE EXTRACT(YEAR FROM ns.order_date) = 2014) "2014"
, (SELECT COUNT(*) FROM orders ns WHERE EXTRACT(YEAR FROM ns.order_date) = 2015) "2015"
, (SELECT COUNT(*) FROM orders ns WHERE EXTRACT(YEAR FROM ns.order_date) = 2016) "2016"
, (SELECT COUNT(*) FROM orders ns  ) Total
FROM DUAL --ИЗПОЛЗВА СЕ КОГАТО ТРЯБВА ДА НАПИШЕМ СЕЛЕКТ БЕЗ ФРОМ
ORDER BY 1 NULLS LAST --ДА СОРТИРА ПО ПЪРВА КОЛОНА И НУЛЛС ЛАСТ, ЗА ДА ОСТАНЕ ГРАНД ТОТАЛ НАЙ-ДОЛУ ИЛИ FIRST АКО ИСКАМЕ ДА Е НАЙ-
ГОРЕ
--АКО ЗАЯВКАТА Е МНОГО СЛОЖНА СИ СТРУВА ДА НАПИШЕМ ФУНКЦИЯ И ДА СЪБЕРЕМ ВЪТРЕ СЛОЖНИТЕ НЕЩА. ФУНКЦИИ ОБАЧЕ СЕ ПИШАТ НА ДОПЪЛНИТЕЛЕН
ЕЗИК
-- PL SQL ЗА ОРАКЪЛ, НО МОЖЕ ДА ПОМОЛИМ ИТ ОРДЕЛА, ВИЖ В ПРОФИЛА МУ, КАК ДА СИ НАПРАВИМ СОБСТСВЕНА АНАЛИТИЧНА ИЛИ АГРЕГИРАЩА
ФУНКИЯ(КЛИП)
-- СРАВНЕНИЕ НА ВЛОЖЕНИТЕ ЗАЯВКИ
-- FROM - ИЗПЪЛНЯВА СЕ ВЕДНЪЖ, ВРЪЩА СЕ РЕЗУЛТАТ И СЛЕД ТОВА СЕ ПОЛЗВА
-- SELECT - нАЙ-НЕИКОНОМИЧЕН,ИЗЧИСЛЕНИЯТА ЩЕ ПРОТЕКЪТ ТОЛКОВА ПЪТИ, КОЛКОТО СА ТРЕДОВЕТЕ ВЪВ ВЪНШНАТА ЗАЯВКА 91 *4 В СЛУЧАЯ
-- WHERE - ЗАВИСИ, АКО ИМА SUBQUIRY FACTORING ЗА ВСЕКИ РЕД/ КАТЕГОРИЯ СЕ ИЗПЪЛНЯВА
```

CASE... WHEN ...THEN ...ELSE

подръжа се от всички сървъри

CASE ИЗРАЗА МОЖЕ ДА БЪДЕ КОЛОНА, МОЖЕ ДА Е ИЗЧИСЛИМА

WHEN ПРОВЕРКИ, КОГАТО ИЗРАЗА (ОТ CASE) ИМА СТ-СТ 1, ИЗПЪЛНЯВА THEN 1, КОГАТО МИНЕ ПРЕЗ ВСИЧКИ WHEN И НЕ НАМЕРИ СЪВПАДЕНИЕ ИЗПЪЛНЯВА ELSE
THEN ОНОВА КОЕТО ЩЕ ИЗКАРА, АКО Е СЪВПАДНАЛО С WHEN

ПЪРВИ ФОРМАТ.
ТУК ИМАМЕ СТ-СТ, ФИКСИОРАНА (1.5.10)**ИЗПОЛЗВА СЕ ЗА КОЛОНИ, КОИТО ИМАТ ДИСКРЕТНИ СТ-СТИ** (НАПР. МЕСЕЦ, ТЕ СА ЦЯЛО ЧИСЛО)
МОЖЕ ДА СЕ ИЗВАДЯТ МЕСЕЦИТЕ CASE EXTRACT MOUNT, WHEN 1, THEN ЯНУАРИ И Т.Н. 12 ПЪТИ + ЕДМНО ПРАЗНО, АКО ИМА ПРАЗНА ДАТА. ИЛИ ЗА ПОЛ,
-- WHEN 1, THEN FEMALE

```
CASE израз
  WHEN стойност1 THEN израз1
  WHEN стойност2 THEN израз2
  WHEN стойност3 THEN израз3
  ...
  ELSE
    изразN
END
```

ВТОРИ ФОРМАТ.

В WHEN СТОИ УСЛОВИЕ, МОЖЕ ДА ГО СЪСТАВИМ ПО ВСИЧ1КИ НАЧИНИ LIKE, IN, BETWEEN И Т.Т.ПО-СЛОЖНИ ПРОВЕРКИ
ИЗПОЛЗВА СЕ ЗА НЕПРЕКЪСНАТИ СТ-СТИ, ОТ 0 ДО + БЕЗКРАЙНОСТ
(напр. СУМА НА ПОРЪЧКИТЕ)ВИЖ ПРОФИКЛА, КАК ОТ ЕГН ДА РАЗБИЕМ МЪЖЕ, ЖЕНИ И ВЪЗ. КАТЕГОРИИ

```
CASE
  WHEN условие1 THEN израз1
  WHEN условие2 THEN израз2
  WHEN условие3 THEN израз3
  ...
  ELSE
    изразN
END
```

ЗАДАЧА 11 - СТЪПКИ - CASE

```
-- От коя държава колко поръчки има в диапазона до 1000 $| от 1000 до 4000 $| над 4000 $
--1 От коя държава каква е стойността на всяка поръчки?
SELECT t1.country
, t2.order_id
, SUM(t3.unit_price * t3.quantity) total
FROM customers t1
        INNER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
        INNER JOIN
     order_details t3
        ON t3.order_id = t2.order_id
GROUP BY t1.country
      ,  t2.order_id
ORDER BY 1, 3
----------------------------------------------------------РЕШЕНИЕ НА ЗАДАЧА 11
WITH q1 AS (SELECT t1.country
            , t2.order_id
            , SUM(t3.unit_price * t3.quantity) total
            FROM customers t1
                    INNER JOIN
                 orders t2
                    ON t2.customer_id = t1.customer_id
                    INNER JOIN
                 order_details t3
                    ON t3.order_id = t2.order_id
            GROUP BY t1.country
                  ,  t2.order_id)
SELECT q1.country --БРОИМ ЕДИЦИЦИТЕ,КАКТО ЗАТВОРНИЦИТЕ ДРАСКАТ  ЧЕРТИ
, SUM(CASE WHEN q1.total < 1000 THEN 1 ELSE 0 END) "LT 1000" -- ТРЯБВА МНОГО ДА ВНИМАВАМЕ ДА НЕ ЗАСЕЧЕМ ДИАПАЗОНТЕ
, SUM(CASE WHEN q1.total < 1000 THEN q1.total ELSE 0 END) "Total < 1000" -- ИСКАМЕ ДА НАМЕРИМ ОБЩАТА СТ-СТ НА ТЕЗИ ПОРЪЧКИ, КОИТО СА
ПОД ХИЛЯДА
, SUM(CASE WHEN q1.total BETWEEN 1000 AND 4000 THEN 1 ELSE 0 END) "BTW 1000-4000"
, SUM(CASE WHEN q1.total BETWEEN 1000 AND 4000 THEN q1.total ELSE 0 END) "BTW 1000-4000"-- ИСКАМЕ ДА НАМЕРИМ ОБЩАТА СТ-СТ НА ТЕЗИ
ПОРЪЧКИ,КОИТО СА М/У 1000-400
, SUM(CASE WHEN q1.total > 4000 THEN 1 ELSE 0 END) "GT 4000"
, SUM(CASE WHEN q1.total > 4000 THEN q1.total ELSE 0 END) "GT 4000"-- ИСКАМЕ ДА НАМЕРИМ ОБЩАТА СТ-СТ НА ТЕЗИ ПОРЪЧКИ, КОИТО СА НАД
4000
, COUNT(*) NOrders --ДА ПРЕБРОИМ ОБЩО ВСИЧКИ ПОРЪЧКИ ОТ ДЪРЖАВА
, SUM(q1.total) GTotal -- СУМА НА АБСОЛЮТНО ВСИЧКИ ПОРЪЧКИ ПО КАТЕГОРИИ
FROM q1
GROUP BY q1.country
ORDER BY 1
```

СТЪПКА 1 ЗАДАЧА 12

```
-------------------------------------------------------------------------
--  СТЪПКА 1 ЗАДАЧА 12
-- От коя държава по колко поръчки има и по колко продукта са продадени? НЕ МОЖЕ ДА СЕ ПОЛУЧИ С ЕДНА ЗАЯВКА
-------------------------------------------------------------------------
--1 От коя държава колко продукта са закупени
-- CUSTOMERS I ORDERS - БРОИ ПОРЪЧКИ, КОГАТО ДОБАВИМ ORDER DETAILS ВЕЧЕ БРОИ ПРОДУКТИ
SELECT t1.country
, COUNT(*) NProducts
FROM customers t1
        INNER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
        INNER JOIN
     order_details t3
        ON t3.order_id = t2.order_id
GROUP BY t1.country
ORDER BY 1
------------
--  СТЪПКА 2 ЗАДАЧА 12
--2 От коя държава колко поръчки има? ТУК ВЕЧЕ БРОИ ПОРЪЧКИ
SELECT t1.country
, COUNT(*) NOrders
FROM customers t1
        INNER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
GROUP BY t1.country
ORDER BY 1 -- НЕ МОЖЕМ ДА ОБЕДИНИМ ВЕДНАГА С UNION,ЗАЩОТО ИМАМЕ РАЗЛИЧЕН БРОЙ КОЛОНИ, КОИТО НОСЯТ РАЗЛИЧЕН СМИСЪЛ

-- ПЪРВАТА КОЛОНА СЪВПАДА, НО В 1-ВА ЗАЯВКА, 2-РА КОЛОНА Е ПОРЪЧКИ, А ВЪВ 2-РА ПРОДУКТИ
--ЗАТОВА КЪМ ПЪРВАТА ЗАЯВКА ДОБАВЯМЕ 3-ТА КОЛОНА С НУЛА (0 Nproducts) И ГРУПИРАМЕ 0
-- ВЪВ ВТОРАТА ДОБАВЯМЕ НА СВОЙ РЕД, КАТО ВТОРА КОЛОНА 0 ПОРЪЧКИ И ГРУПИРАМЕ ПО 0
-- ВЛАГАМЕ ЗАЯВКАТА, ЗА ДА РАЗКАРАМЕ 0-ТЕ И ДА ГИ ОБЕДИНИМ
-- КОГАТО ДОБАВЯМЕ ФИКТИВНИ КОЛОНИ, ЪТРЯБВА ДА ВНИМАВАМЕ С НУЛЛ, НО МОЖЕ ДА ГО ПОЛЗВАМЕ ЗСА ВСИЧКИ ВИДОВЕ КОЛОНИ'
```

РЕШЕНИЕ НА ЗАДАЧА 12 - ТРИК

```
WITH q1 AS (SELECT t1.country
        , COUNT(*) NOrders
        , 0 Nproducts --0 ПРОДУКТИ, ЗА ДА ИЗРАВНИМ КОЛОНИТЕ,БЕЗ ДА ПОВЛИАЕМ НА РЕЗУЛТАТА, АКО Е ТЕКСТОВА КОЛОНА,МОЖЕ Е ПРАЗЕН ТЕКС
        FROM customers t1
                INNER JOIN
             orders t2
                ON t2.customer_id = t1.customer_id
        GROUP BY t1.country, 0 -- НЕ ЗАБРАВЯМЕ ДА ДОБАВИМ НУЛАТА В ГРУПИРАНЕТО
        UNION
        SELECT t1.country
        , 0  -- НУЛА ПОРЪЧКИ, ЗА ДА БЪДАТ КОЛОНИТЕ С ЕДНАКЪВ БРОЙ И СМИСЪЛ, НО И ДА НЕ ИЗКРИВИМ РЕЗУЛТАТА
        , COUNT(*) NProducts
        FROM customers t1
                INNER JOIN
             orders t2
                ON t2.customer_id = t1.customer_id
                INNER JOIN
             order_details t3
                ON t3.order_id = t2.order_id
        GROUP BY t1.country, 0) -- НЕ ЗАБРАВЯМЕ ДА ДОБАВИМ НУЛАТА В ГРУПИРАНЕТО
SELECT q1.country
, SUM(q1.norders) NOrders -- СУМИРАМЕ, ЗА ДА ОБЕДИМ ДВАТА РЕДА, КОИТО ИЗЛИЗАТ ЗА ВСЯКА ДЪРЖАВА, КАТО ЕДНАТА ЗАДЪЛЖИТ Е НУЛА 0 + ...
, SUM(q1.nproducts) NProducts -- СУМИРА ДВЕТЕ КОЛОНИ НА ПРОДУКТИ 0 + ...
FROM q1
GROUP BY q1.country
ORDER BY 1
```

* SELECT с или без FROM

ЧАТ ОТ СЪРВЪРИТЕ ПОЗВОЛЯВАТ SELECT БЕЗ FROM, НАПРИМЕР My SQL, НО ORACLE НЕ ПОЗВОЛЯВА
ЗА ДА СЕ НАПИШЕ СЕЛЕКТ БЕЗ ФРОМ СЕ ИЗПОЛЗВА СПЕЦИАЛНА ТАБЛИЦА, КОЯТО Е ЧАСТ ОТ ВСЯКА БАЗА ДАННИ, НО НОСИ РАЗЛИЧНО ИМЕ
ЗА ОРАКЪЛ Е DUAL (ВИЖ ПО-ДОЛУ)
СЕЛЕКТ БЕЗ ФРОМ СЕ ИЗПОЛЗВА ЗА ПРОВЕРКИ, ИЗЧИСЛЕНИЯ,REGEXP LIKE И ДРУГИ
МОЖЕМ И САМИ ДА СЪЗДАДЕМ ТАКАВА (CREATE TABLE ИМЕ, ДОБАВЯМЕ КОЛОНА И 1 РЕД)

```sql
MS SQL, MySQL, PostgreSQL: SELECT SYSDATE


SYBASE: SELECT SYSDATE FROM DUMMY


Oracle: SELECT SYSDATE FROM DUAL
```

ВИЖ ЗАДАЧА 10

ЛЕКЦИЯ 8

CROSS JOIN -> Декартовото произведение на редовете в табл. **ВСЯКА СТ-СТ ОТ ПЪРВАТА ТАБЛИЦА С ВСЯКА ОТ ВТОРАТА ПОДХОДЯЩО Е, КОГАТО 1-та ТАБЛИЦА ВРЪЩА ЕДИН РЕД (**
A {a,b} B {c,d,e} = A x B -> {a, c}, {a,d}, {a,e}, {b, c}, {b,d}, {b,e}

Кой служител колко поръчки е обслужил и какъв % от общият брой представлява това

```
--1. Общ брой поръчки
SELECT COUNT(*) AllOrds
FROM orders
-----
--2. Кой служител колко поръчки е обсл.
SELECT t1.firstname || ' ' || t1.lastname Employee
, COUNT(*) NOrders
FROM
    employees t1
        INNER JOIN
    orders t2
        ON t2.employee_id = t1.employee_id
GROUP BY t1.firstname || ' ' || t1.lastname
--- РЕШЕНИЕ НА ЗАДАЧАТА Кой служител колко поръчки е обслужил и какъв % от общият брой представлява това

WITH emps AS (SELECT t1.firstname || ' ' || t1.lastname Employee
                , COUNT(*) NOrders
                FROM
                    employees t1
                        INNER JOIN
                    orders t2
                        ON t2.employee_id = t1.employee_id
                GROUP BY t1.firstname || ' ' || t1.lastname)
, ords AS  ( SELECT COUNT(*) AllOrds
             FROM orders)
SELECT e.employee
, e.norders
, ROUND(e.norders / r.allords * 100,2) PercOfAll --ПРОЦЕНТ ОТ ОБЩИЯ БРОЙ,АКО ЩЕ ХОДИ В ЕКСЕЛ ПО-ДОБРЕ ДА НЯМА РАУНД
FROM emps e CROSS JOIN ords r --алтернативен синтаксис, не пишем cross join, а само запетая FROM emps e, ords r
ORDER BY 2 DESC
```

ЛЕКЦИЯ 9

● Алтернативен синтаксис на CROSS JOIN

```
WITH emps AS (SELECT t1.firstname || ' ' || t1.lastname Employee
                , COUNT(*) NOrders
                FROM
                    employees t1
                        INNER JOIN
                    orders t2
                        ON t2.employee_id = t1.employee_id
                GROUP BY t1.firstname || ' ' || t1.lastname)
, ords AS  ( SELECT COUNT(*) AllOrds
             FROM orders)
SELECT e.employee
, e.norders
, ROUND(e.norders / r.allords * 100,2) PercOfAll
FROM emps e, ords r -- също е CROSS JOIN, ХУБАВО Е ДА ГО ИЗПИСВАМЕ, ЗАЩОТО ПРИЧЛИА НА АЛТЕНБАТИВЕН СИНТАКСИС НА INNER JOIN
ORDER BY 2 DESC
```

●-- Алтернативен синтаксис на INNER JOIN

```
●
SELECT t1.firstname || ' ' || t1.lastname Employee
, COUNT(*) NOrders
FROM
      employees t1
    , orders t2
-- ВМЕСТО
--FROM
--employees t1
--      INNER JOIN липсва и се слага запетая
--orders t2
--      ON t2.employee_id = t1.employee_id това без On се мести в where клаузата
WHERE t2.employee_id = t1.employee_id
GROUP BY t1.firstname || ' ' || t1.lastname


Диалектни изписвания
--WHERE t2.employee_id += t1.employee_id left outer join
```

ТЕЗИ ВРЪЗКИ М/У ТАБЛИЦИ ТРЯБВА ДА СТПЯТ НА ПЪРВО МЯСТО, ЗАЩОТО,WHERE ЧЕТЕ ОТ 1 КЪМ 3, НО НАЛАГА ОГРАНИЧЕНИЯ ОТ ПОСЛЕДНО КЪМ ПЪРВО. ТОВА СЕ ОТРАЗЯВА НА СКРОСТТА НА ИЗПЪЛНЕНИЕ

```
WHERE (1) --връзките м/у таблиците СА НАЙ-МАЛКО ОГРАНИЧАВАЩИ, ЗАЩОТО СТОЯТ НА ПЪРВО МЯСТО
        AND
      (2)
        AND
      (3) -- PK,FK. Реда на изпълнение е отзад напред, затова най-долу се пишат най-ограничаващите критерии,като PK I FK. ЦЕЛТА Е
ОЩЕ С ПЪРВИЯ КРИТЕРИИ ДА ОТПАДНАТ ПО-ГОЛЯМА ЧАСТ ОТ ДАННИТЕ

Изпълнение: (3) -> (2) -> (1)
```

- Кой служител колко поръчки е обслужил и какъв % от общият брой представлява това, НО ВЪВ ВАРИАНТ, В КОИТО СЪРВЪРА ЩЕ ИЗЧИСЛИ ПО-БЪРЗО

```
WITH emps AS (SELECT t1.firstname || ' ' || t1.lastname Employee
                , COUNT(*) NOrders
                FROM
                    employees t1
                        INNER JOIN
                    orders t2
                        ON t2.employee_id = t1.employee_id
                GROUP BY t1.firstname || ' ' || t1.lastname)
, ords AS  ( SELECT SUM(emps.NOrders) AllOrds -- горе сме извадили,кои служител,колко поръчки е обслужил и вместо отново да брои в/у
--цялата таблица ордерс,колко да поръчките,да променим заявката така, защото тук сумира 9 числа, а не брои 830 реда
             FROM emps)
--SELECT * FROM ords     --за да тестваме само горната част
SELECT e.employee
, e.norders
, ROUND(e.norders / r.allords * 100,2) PercOfAll
FROM emps e, ords r -- също е CROSS JOIN
ORDER BY 2 DESC


1+1+1+0+0+0/6
```

Предишната задача, но с **категория "Други" – CASE..**

- Кой служител колко поръчки е обслужил и какъв % от общият брой представлява това, КАТО В КАТЕГОРИЯ ДРУГИ ДА БЪДАТ ТЕЗИ С ПОРЪЧКИ ПОД 10%

```
WITH emps AS (SELECT t1.firstname || ' ' || t1.lastname Employee
                , COUNT(*) NOrders
                FROM
                    employees t1
                        INNER JOIN
                    orders t2
                        ON t2.employee_id = t1.employee_id
                GROUP BY t1.firstname || ' ' || t1.lastname)
, ords AS  ( SELECT SUM(emps.NOrders) AllOrds
             FROM emps)
, prc  AS  (SELECT e.employee
             , e.norders
             , ROUND(e.norders / r.allords * 100,2) PercOfAll
             FROM emps e CROSS JOIN ords r )
SELECT (CASE WHEN p.PercOfAll >= 10 THEN p.employee ELSE 'Others' END) Employee
--когато процента от общото е по-голям или равен на другото, тогава човека да си излезе със името, else да ходи в други
, SUM(p.norders) Norders
, SUM(p.percofall) PercOfAll --не е проблем, че съвпада псевдонима
FROM prc p
GROUP BY (CASE WHEN p.PercOfAll >= 10 THEN p.employee ELSE 'Others' END) --сумираме за да съберем категория други в едно
--като не забравяме, че GROUP BY не поддръжа псевдоними и трябва да изпишем цялата функция
ORDER BY 2 ASC
```

СЪС CASE МОГАТ ДА СЕ ИЗЧИСЛЯВАТ И ПРОЦЕНТИ, НО С AVG. НАПРИМЕР ПРОЦЕНТ НА ГЕРМАНИЯ
ЗАДАВАМЕ ЗА ГЕРМАНИЯ – 1, А ЗА ELSE – 0, ВЗИМАМЕ AVG, КОЕТО СУМИРА ЕДИНИЦИТЕ И НЕ ВЗИМА ПОД ВНИМАНИЕ НУЛИТЕ, И НАКРАЯ ДЕЛИ ОБЩОТО НА СБОРА ОТ ЕДИНИЦИТЕ

АКО ГЕРМАНИЯ Е 1, А ЕЛС 0
1,1,1,0,0,0
AVG ЩЕ СМЕТНЕ СРЕДНО АРИТМЕТИЧНО, ТОЕСТ ЩЕ РАЗДЕЛИ 3 (1+1+1) НА 6 (1+1+1+0+0+0)
ТОЗИ МЕТОД Е УДОБЕН ЗА БИНАРНИ ИЗЧИСЛЕНИЯ (ДА, НЕ) (МЪЖ, ЖЕНА, КАТО НАПР. 1 Е МЪЖ, ДВЕ ЖЕНА И ГИ СУМИРА)

АНАЛИТИЧНИ ФУНКЦИИ

●-- ---

Изчисляване на ранг (ТОП 10 напр.)

НЕ ИЗИСКВАТ ГРУПИРАНЕ И МОГАТ ДА ИЗВЪРШВАТ ОПЕРАЦИИ М/У РЕДОВЕТЕ НА ИЗВАДКАТА И СА ИЗКЛЮЧИТЕЛНО ПОЛЕЗНИ. ИМА ГИ В ОРАКЪЛ, МАЙКРОСОФТ И ОТ СКОРО В MY SQL, post..ДРУГОТО ИМ ИМ Е ПРОЗОРЪЧНИ ФУКНЦИИ / WINDOW FUNCTIONS< ТЪЙ КАТО ДЕЙСТВАТ В/У ПРОЗОРЕЦ ОТ ДАННИ В ОРЪКЪЛ И ПОСТ.. ,МОЖЕМ САМИ ДА НАПИШЕМ ТАКАВА ФУКЦИЯ. ПРИ АНАЛИТИЧНИТЕ ФУНКЦИИ НАКРАЯ ВИНАГИ ИМА OVER

● ЗА ВСЕКИ СЛУЖИТЕЛ, КОЙТО ИЗВЕЖДАМЕ КОЛКО НА БРОЙ НА СЛУЖИТЕЛИТЕ КОИТО ИМАТ NORDERS, ПО-ГОЛЯМО ОТ ТОВА, КОЕТО В МОМЕНТА ИЗВЕЖДАМЕ НАПРИМЕР АНДРЮ ИМА 96, КОЛКО СА СЛУЖИТЕЛИТЕ, КОИТО ИМАТ ПОВЕЧЕ ОТ 96

Класическо решение – ПЪЛЕН ТАШАК! НЕ СЕ ЗАНИМАВАЙ!

```
WITH emps AS (SELECT t1.firstname ||' '||t1.lastname Employee
                  , COUNT(*) NOrders
                  FROM
                        employees t1
                        INNER JOIN
                        orders t2
                        ON t2.employee_id = t1.employee_id
                  GROUP BY t1.firstname ||' '||t1.lastname)
SELECT e.employee
, e.norders
, (SELECT COUNT(*) FROM emps ns WHERE ns.norders >= e.norders) Rank --COUNT Броя колко от цялото множество имат броя поръчки по-голям или равен на лицето, което е стигнало в момента
FROM emps e
ORDER BY 3
```

●-- Изчисляване на ранг (ТОП 10 напр.) Аналитична функция

● ИЗЧИСЛЯВА НЕ НА РАНГ. НОРМАЛЕН ЧОВЕШКИ НАЧИН!!!!!

```
WITH emps AS (SELECT t1.firstname || ' ' || t1.lastname Employee
                , COUNT(*) NOrders
                FROM
                    employees t1
                        INNER JOIN
                    orders t2
                        ON t2.employee_id = t1.employee_id
                GROUP BY t1.firstname || ' ' || t1.lastname)
SELECT e.employee
, e.norders
, RANK() OVER( ORDER BY e.norders DESC) Rank --ТРЯБВА ДА СЕ ВНИМАВА ЗА АРГУМЕНТА В ORDER BY, АКО СЛОЖИМ EMPLOYEE ЩЕ НАПРАВИ РАНГА ПО
--АЗБУЧЕН РЕД, ПРИ DESC – СТИВЪН КИНГ ЩЕ ДОЙДЕ НА ПЪРВО МЯСТО, ЗАЩОТО S е на последно място по азбучен ред
FROM emps e
```

ORDER BY В АНАЛИТИЧНА Ф-Я (КАТО ИМЕ НА КОЛОНА) УПРАВЛЯВА КАК ЩЕ ПРОТИЧАТ ИЗЧИСЛЕНИЯТА ОТ ВИСКО КЪМ НИСКИ СТ-СТИ ИЛИ ОБРАТНО(ASC I DESC)

●-- Филтриране на стойностите

●

```
WITH emps AS (SELECT t1.firstname || ' ' || t1.lastname Employee
                , COUNT(*) NOrders
                FROM
                    employees t1
                        INNER JOIN
                    orders t2
                        ON t2.employee_id = t1.employee_id
                GROUP BY t1.firstname || ' ' || t1.lastname)
, ranks AS (SELECT e.employee
            , e.norders
            , RANK() OVER( ORDER BY e.norders DESC) Rank
            FROM emps e)
SELECT rs.employee
, rs.norders
, rs.rank
FROM ranks rs --може да остане навсякъде с псевдоним rank
WHERE rs.rank <= 3
--ORDER BY 3
```

Exetution order - кога се изчислява аналитичната функция.

!!!!!!! Аналитичните ф-ии се изчисляват накрая, след всичко останало. Не може да сложим RANG В WHERE

ФИЛТРИРАНЕТО СТАВА С ВЛАГАНЕ

PARTITION BY използва се за разделяне на резултата от заявката, зададен на групи въз основа на една или повече *value_expr*. Ако пропуснем тази клауза, функцията третира всички редове от резултата от заявката, зададени като една група.

-

```
SELECT t1.department_name
, t2.last_name
, t2.salary
, t2.job_id
, RANK() OVER (ORDER BY t2.salary DESC) CompanyRank
, RANK() OVER ( PARTITION BY t1.department_name
                ORDER BY t2.salary DESC) DepRank
FROM departments t1
        INNER JOIN
     employees t2
        ON t2.department_id = t1.department_id
ORDER BY 1,5
```

-

```
SELECT t1.department_name
, t2.last_name
, t2.salary
, t2.job_id
, DENSE_RANK() OVER (ORDER BY t2.salary DESC) CompanyRank
, DENSE_RANK() OVER ( PARTITION BY t1.department_name
                ORDER BY t2.salary DESC) DepRank
FROM departments t1
        INNER JOIN
     employees t2
        ON t2.department_id = t1.department_id
ORDER BY 5
```

```
-------------------------------------------------------------
●-- Кои държави формират 30% от продажбите (комулативна диаграма)
-------------------------------------------------------------
```

●

```
SELECT t1.country
, COUNT(*) NOrders
FROM customers t1
        INNER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
GROUP BY t1.country
ORDER BY 2 DESC
```

ЛЕКЦИЯ 10

```
--1
SELECT t1.country
, EXTRACT(YEAR FROM t2.order_date) Year
, EXTRACT(Month FROM t2.order_date) Month
, EXTRACT(DAY FROM t2.order_date) Day
, SUM(t3.unit_price * t3.quantity) Total
FROM customers t1
        INNER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
        INNER JOIN
     order_details t3
        ON t3.order_id = t2.order_id
GROUP BY t1.country
        , EXTRACT(YEAR FROM t2.order_date)
        , EXTRACT(Month FROM t2.order_date)
        , EXTRACT(DAY FROM t2.order_date)
ORDER BY 1,2,3,4
```

```sql
--2


WITH qry AS (SELECT t1.country
             , EXTRACT(YEAR FROM t2.order_date) Year
             , EXTRACT(Month FROM t2.order_date) Month
             , EXTRACT(DAY FROM t2.order_date) Day
             , SUM(t3.unit_price * t3.quantity) Total
             FROM customers t1
                     INNER JOIN
                  orders t2
                     ON t2.customer_id = t1.customer_id
                     INNER JOIN
                  order_details t3
                     ON t3.order_id = t2.order_id
             GROUP BY t1.country
                     , EXTRACT(YEAR FROM t2.order_date)
                     , EXTRACT(Month FROM t2.order_date)
                     , EXTRACT(DAY FROM t2.order_date))
SELECT q.country
, q.year
, q.month
, q.day
, q.total
, SUM(q.total) OVER( PARTITION BY q.country
                     ORDER BY q.year, q.month, q.day
                     ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) TotalCountry
, SUM(q.total) OVER( PARTITION BY q.country, q.year, q.month
                     ORDER BY q.year, q.month, q.day
                     ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) TotalYearMonth
FROM qry q
```

```
                    ------------------------------------------------------
          -- От коя държава по колко продукта са закупени и колко са уникални (%)
                    ------------------------------------------------------
```

```
COUNT(*) -> брои редове (INNER JOIN)
COUNT(col) -> брои <> NULL в колоната ( ... OUTER JOIN)
COUNT( DISTINCT col ) -> брои уникални стойности
```

```
---
SELECT DISTINCT
  country
, city
FROM customers
ORDER BY 1,2
---
WITH prods AS (SELECT t1.country
               , COUNT(*) NProducts
               , COUNT(DISTINCT t3.product_id) NUnique
               FROM customers t1
                    INNER JOIN
                 orders t2
                    ON t2.customer_id = t1.customer_id
                    INNER JOIN
                 order_details t3
                    ON t3.order_id = t2.order_id
               GROUP BY t1.country)
SELECT ps.country
, ps.nproducts
, ps.nunique
, ROUND(ps.nunique / ps.nproducts * 100,2) PercUnique
FROM prods ps
ORDER BY 4 DESC
```

```
-------------------------------------------------------------
-- Поръчки, продукти
-------------------------------------------------------------
```

```sql
SELECT t1.country
, COUNT(*) NProducts
, COUNT(DISTINCT t2.order_id) NOrders
, SUM(t3.unit_price * t3.quantity) Total
FROM customers t1
        INNER JOIN
     orders t2
        ON t2.customer_id = t1.customer_id
        INNER JOIN
     order_details t3
        ON t3.order_id = t2.order_id
GROUP BY t1.country
ORDER BY 1
```

Разпределение на поръчки според броя на цифрите в сумата на поръчката

```
1-9
10-99
100-999
1000-9999
```

```
-- LOG(y,x) = n  y^n=x
SELECT CEIL(LOG(10,1236.45))
FROM DUAL
```

```
SELECT POWER(10, 3.092176558899383968248254485320662690 58 )
FROM DUAL
```

```
SELECT POWER(10, 4-1) LowerBound
, POWER(10,4)-1 UpperBound
FROM DUAL
```

```sql
--1
SELECT order_id
, SUM(unit_price*quantity) total
, COUNT(*) NProducts
FROM order_details
GROUP BY order_id



------------------------



WITH ords AS (SELECT order_id
                , SUM(unit_price*quantity) total
                , COUNT(*) NProducts
                FROM order_details
                GROUP BY order_id)
, dgts AS  (SELECT rs.order_id
             , rs.total
             , rs.nproducts
             , CEIL(LOG(10,rs.total)) ndigits
             FROM ords rs)
SELECT POWER(10, dg.ndigits-1) LBound
, POWER(10,dg.ndigits)-1 UBound
, COUNT(*) NOrders
, SUM(dg.nproducts) NProducts
, SUM(dg.total) GTotal
, MIN(dg.total) MinTotal
, ROUND(AVG(dg.total),2) AvgTotal
, MAX(dg.total) MaxTotal
FROM dgts dg
GROUP BY POWER(10, dg.ndigits-1)
       , POWER(10,dg.ndigits)-1
ORDER BY 1
```