**Упражнение Вартоломеев**

**От задачите за group by:**

*-- [] -- знаем откъде да вземем данните*

*-- {} – НЕ знаем откъде да вземем данните :)*

*-- GROUP BY. Ships. 6. Напишете заявка,*

*-- [която извежда броя на корабите, потънали в битка] според [класа],*

*-- [за тези класове с повече от 2 кораба.]*

*-- Класовете корабит с повече от 2 кораба*

*SELECT class, COUNT(\*) FROM ships*

*GROUP BY class*

*HAVING COUNT(\*) > 2*

*-- Корабите, които са потънали в битка*

*SELECT ship FROM outcomes WHERE result = 'sunk';*

*-- заявка,*

*-- която извежда броя на корабите, потънали в битка според класа,*

*-- за тези класове с повече от 2 кораба.*

*SELECT class, COUNT(\*)*

*FROM ships*

*WHERE name IN (SELECT ship FROM outcomes WHERE result = 'sunk')*

*AND class IN (SELECT class FROM ships GROUP BY class HAVING COUNT(\*) > 2)*

*GROUP BY class;*

**SQL**

--DML =Data Manipulation Language

-- --промяна на на данните

-- --запис(INESRT)

-- --промяна(UPDATE)

-- --изтриване(DELETE)

--DQL = Data Query Language

-- --извличане(SELECT)

--DDL = Data definition language

--дефиниране на структурата на базата – какви бази имаме, те какви колони имат

-- -- CREATE \_TABLE\_ - създава

-- --ALTER\_TABle – променя

-- -- DROP \_TABLE\_ - трие

--DCL = Data control language – управление на потребители, задаване на права и т.н.

-- --INSERT

-- --UPDATE

-- --DELETE

-- За да видите всички таблици:

-- Postgres: \dt

-- MySQL: show tables;

-- MsSQL: SELECT \* FROM INFORMATION\_SCHEMA.TABLES;

-- За да видите структурата на таблицата:

-- Postgres: \d+ table\_name

-- MySQL: DESC table\_name

-- MsSQL: EXEC sp\_columns table\_name

-- INSERT

INSERT INTO SHIPS (NAMES, CLASS, LAUNCHED) VALUES (‘ENTERPRISE’, ‘YAMATO’, 2020)

INSERT INTO SHIPS VALUES (‘ENTERPRISE’, ‘YAMATO’, 2020)

INSERT INTO SHIPS (CLASS, NAMES , LAUNCHED) VALUES (‘ENTERPRISE’, ‘YAMATO’, 2020)

Можем да разменяме местата на колоните и да указваме кое кое е във values.

Винаги предпочитаме да ползваме имената на колоните, тъй като избягваме грешки при добавяне и махане на нови колони в някакъв момент от зареждането на базата.

Ако пропуснем някоя колона и не й дадем value, тя е null по default

Forеign key – в тази колона мога да слагам само съществуващи записи от другата колона

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

INSERT INTO ships (name, class, launched) VALUES ('Enterprise', 'Yamato', 2020);

-- Valid, but don't use it!!!

INSERT INTO ships VALUES ('Enterprise 2', 'Yamato', 2020);

-- rearranging columns

INSERT INTO ships (class, name, launched) VALUES ('Yamato', 'Enterprise 3', 2020);

-- skipping nullalbe column

INSERT INTO ships (class, name) VALUES ('Yamato', 'Enterprise 4');

-- Missing Not-nullalbe column

INSERT INTO ships (name) VALUES ('Enterprise 4');

-- ERROR:  null value in column "class" violates not-null constraint

-- DETAIL:  Failing row contains (Enterprise 4, null, null).

-- Wrong value for Foreign Key column

INSERT INTO ships (name, class) VALUES ('Enterprise 5', 'ala bala');

-- ERROR:  insert or update on table "ships" violates foreign key constraint "fk\_ships\_classes"

-- DETAIL:  Key (class)=(ala bala) is not present in table "classes".

-- INSERT INTO SELECT

INSERT INTO classes (class, type, country, numguns, bore, displacement)

SELECT CONCAT('Mega ', class), type, country, numguns \* 10, bore, displacement

FROM classes;

**-- UPDATE**

Update, SET, WHERE

WHERE e optional, но винаги я искаме, тъй като в противен случай ще ъпдейтне всички редове

-- Structure

-- UPDATE \_table\_

-- SET col1 = 'new value'

-- WHERE ...

--

--

-- UPDATE \_table\_

-- SET col1 = 'new value',

--         ...

--         col2 = 'new value'

-- WHERE ...

-- Step 0: Write SELECT with proper WHERE clause

SELECT \* FROM classes WHERE class LIKE 'Mega%';

-- Step 1: Write invalid with the WHERE clause

UPDATE classes SET ... WHERE class LIKE 'Mega%';

-- Step 2: Write the full UPDATE

UPDATE classes

SET displacement = displacement \* 100

WHERE class LIKE 'Mega%';

-- Using subquery

UPDATE classes

SET displacement = (SELECT MAX(displacement) FROM classes)

WHERE class LIKE 'SUPER Mega%';

-- Corelational subquery on the update

UPDATE classes c

SET numguns = (SELECT MAX(numguns) FROM classes WHERE classes.type = c.type)

WHERE class LIKE 'SUPER Mega%';

**-- DELETE**

-- Structure:

-- DELETE FROM \_TABLE\_ WHERE ...;

-- Step 0. Write SELECT with proper WHERE clause

SELECT \* FROM classes WHERE class LIKE 'SUPER Mega %';

-- Step 1. Change SELECT to DELETE

DELETE FROM classes WHERE class LIKE 'SUPER Mega %';

-- Where trying to delete records referred by a FK

DELETE FROM classes WHERE class = 'Yamato';

-- ERROR:  update or delete on table "classes" violates foreign key constraint "fk\_ships\_classes" on table "ships"

-- DETAIL:  Key (class)=(Yamato) is still referenced from table "ships".

***Задачи:***

-- PC. 4. Използвайте две INSERT заявки. Съхранете в базата данни факта, че персонален компютър

-- модел 1100 е направен от производителя C, има процесор 2400 MHz, RAM 2048 MB, твърд диск

-- 500 GB, 52x оптично дисково устройство и струва $299. Нека новият компютър има код 12.

-- Забележка: модел и CD са от тип низ.

INSERT INTO product (maker, model, type)

VALUES ('C', '1100', 'PC');

INSERT INTO pc (code, model, speed, ram, hd, cd , price)

VALUES (12, '1100', 2400, 2048, 500, '52x', 299);

-- PC. 5. Да се изтрие наличната информация в таблицата PC за компютри модел 1100.

SELECT \* FROM pc WHERE model = '1100';

DELETE FROM pc WHERE model = '1100';

-- PC. 6. Да се изтрият от таблицата Laptop всички лаптопи, направени от производител, който не

-- произвежда принтери.

SELECT \*

FROM product

WHERE maker NOT IN (SELECT maker FROM product WHERE type = 'Printer');

SELECT \*

FROM product

WHERE maker NOT IN (SELECT maker FROM product WHERE type = 'Printer')

AND type = 'Laptop'

SELECT \* FROM laptop

WHERE model IN (SELECT model

FROM product

WHERE maker NOT IN (SELECT maker FROM product WHERE type = 'Printer')

AND type = 'Laptop');

SELECT \* FROM laptop

WHERE model IN (SELECT model

FROM product

WHERE maker NOT IN (SELECT maker FROM product WHERE type = 'Printer'));

DELETE FROM laptop

WHERE model IN (SELECT model

FROM product

WHERE maker NOT IN (SELECT maker FROM product WHERE type = 'Printer')

AND type = 'Laptop');

-- PC. 7. Производител А купува производител B. На всички продукти на В променете производителя да

-- бъде А.

UPDATE product

SET maker = 'A'

WHERE maker = 'B'

-- PC. 8. Да се намали наполовина цената на всеки компютър и да се добавят по 20 GB към всеки твърд

-- диск.

SELECT hd, hd + 20, price, price / 2 FROM pc ;

UPDATE pc

SET hd = hd + 20,

price = price / 2;

-- PC. 9. За всеки лаптоп от производител B добавете по един инч към диагонала на екрана.

SELECT model, screen, screen + 1

FROM laptop

WHERE model IN (SELECT model FROM product WHERE maker = 'B');

UPDATE laptop

SET screen = screen + 1

WHERE model IN (SELECT model FROM product WHERE maker = 'B');

-- Test with existing data

UPDATE laptop

SET screen = screen + 1

WHERE model IN (SELECT model FROM product WHERE maker = 'A');

-- Restore existing data

UPDATE laptop

SET screen = screen - 1

WHERE model IN (SELECT model FROM product WHERE maker = 'A');