ФГБОУ ВО

«Заполярный государственный университет им. Н.М. Федоровского»

КафедраИО Специальность	
ОТЧЕТ	
о выполнении лабораторной работы №4	
	Выполнил:
	Быков В. В.
	Дата:
	<u>«17» апреля 2023 г.</u>

Лабораторная работа №4

Tema: SQL. Запросы.

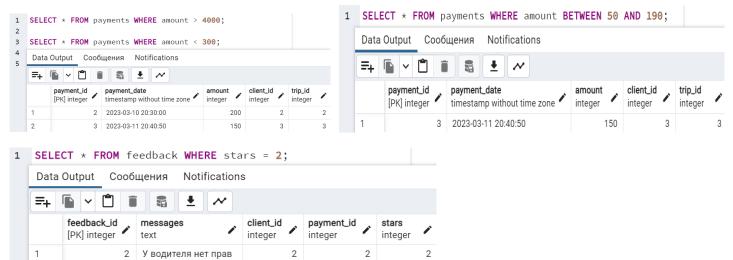
Цель работы: Получение практических навыков работы с СУБД и языком SQL (оператор SELECT).

Задание:

- 1. разработать запросы к базе данных, созданной и заполненной на предыдущих лабораторных работах, следующих видов:
 - **а.** запрос с условием на числовые данные (>,<,=, between);
 - **b.** запрос с условием на текстовые данные (LIKE, IN);
 - с. запрос с вычисляемым полем;
 - **d.** запрос к нескольким таблицам (без явного указания JOIN);
 - **e.** запрос с агрегирующей функцией (AVG, SUM, COUNT, MIN, MAX);
 - **f.** запрос с группировкой (GROUP BY);
 - **g.** запрос с сортировкой (ORDER BY);
 - **h.** запрос с вложенным подзапросом (не менее 3 видов);
 - i. запрос с оператором UNION;
 - j. запрос с оператором INTERSECT;
 - к. запрос с оператором ЕХСЕРТ;
 - **1.** запрос с выражением CASE;
 - **т.** запрос с оператором JOIN (пять видов);
 - **п.** иерархический запрос.
- 2. Для каждого запроса подписать, что именно он возвращает с учетом предметной области (запросы со смыслом, а не только синтаксически правильные операторы).

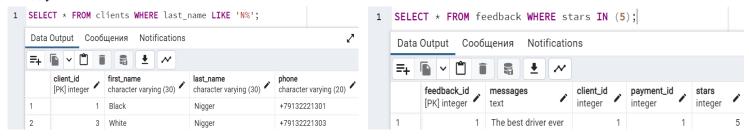
Коды запросов:

- ${f a.}$ SELECT * FROM payments WHERE amount > 4000; запрос вернет все строки из таблицы payments, где плата за поездку больше 4000
- SELECT * FROM payments WHERE amount < 300; запрос вернет все строки из таблицы payments, где плата за поездку меньше 300
- SELECT * FROM feedback WHERE stars = 2; запрос вернет все строки из таблицы feedback, где поставили 2 звезды за поездку
- SELECT * FROM payments WHERE amount BETWEEN 50 AND 190; запрос вернет все строки из таблицы payments, где плата за поездку была между 50 и 190

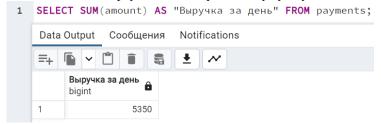


b. SELECT * FROM clients WHERE last_name LIKE 'N%'; - запрос вернет все строки из таблицы clients, где фамилия клиента начинается с буквы Н

SELECT * FROM feedback WHERE stars IN (5); - запрос вернет все строки из таблицы feedback, где поставили 5 звезд за поездку



C. SELECT SUM(amount) AS "Выручка за день" FROM payments; - запрос вычисляет сумму всей выручки за день в столбце amount из таблицы payments и возвращает эту сумму в виде одной строки результата



d. SELECT * FROM clients, feedback WHERE feedback.client_id = clients.client_id; - запрос связывает две таблицы по столбцу "client_id" и возвращает все строки, в которых есть соответствие.



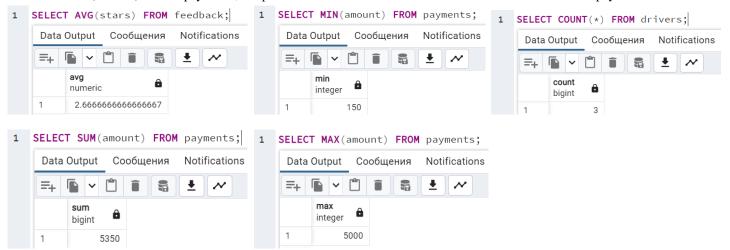
e. SELECT AVG(stars) FROM feedback; - вернет единственное значение, представляющее среднее значение всех значений столбца "stars".

SELECT SUM(amount) FROM payments; - вернет единственное значение, представляющее сумму всех значений столбца "amount".

SELECT COUNT(*) FROM drivers; - вернет количество строк (записей) в таблице "drivers".

SELECT MIN(amount) FROM payments; - вернет минимальное значение столбца "amount" в таблице "payments".

SELECT MAX(amount) FROM payments; - вернет максимальное значение столбца "amount" в таблице "payments".



f. Создание дополнительного поля age к таблице d rivers:

ALTER TABLE drivers ADD COLUMN age int;

UPDATE drivers

SET age = 23 WHERE driver_id = '1';

UPDATE drivers

SET age = 31 WHERE driver_id = '2';

UPDATE drivers

SET age = 50 WHERE driver_id = '3';

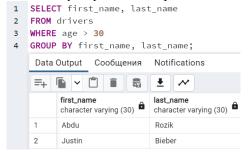
Группировка водителей старше 30 лет:

SELECT first name, last name

FROM drivers

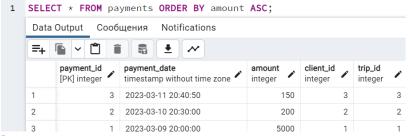
WHERE age > 30

GROUP BY first_name, last_name; - будут возвращены только уникальные комбинации имени и фамилии водителей, которые соответствуют условию возраста.



g• Сортировка полученной суммы по возрастанию

SELECT * FROM payments ORDER BY amount ASC; - вернет все строки (записи) из таблицы "payments", отсортированные в порядке возрастания (ASC) значения столбца "amount".



h. Создание дополнительной таблицы:

```
CREATE TABLE business drivers
(
      driver_id int PRIMARY KEY,
      first_name character varying(30)NOT NULL,
      last_name character varying(30)NOT NULL,
      phone character varying(20) NOT NULL,
      CONSTRAINT business_driver_phone UNIQUE (phone)
);
ALTER TABLE business drivers RENAME COLUMN driver id TO business driver id;
ALTER TABLE cars ADD COLUMN business driver id int;
ALTER TABLE cars ADD FOREIGN KEY (business driver id) REFERENCES business drivers(business driver id);
INSERT INTO business drivers
VALUES
('4','Bob','Willson','+19132221301'),
('5', 'Tony', 'Montana', '+19132221302'),
('6','Vito','Scaletta','+19132221303');
INSERT INTO cars (car_id, model, business_driver_id)
VALUES
('4', 'Bentley Continental', 4),
('5', 'Rolls-Royce Cullinan', 5),
('6', 'Mercedes-Benz Maybach GLS 600', 4),
('7','Велосипед четырехколесный SPORT',4),
('8','AUDI S5',6),
('9','Porsche 911',6);
```

SELECT COUNT(*) FROM trips WHERE client_id = clients.client_id

AS "Кол-во поездок" FROM clients;- вернет список всех клиентов в таблице "clients" с дополнительным столбцом "Кол-во поездок", который показывает, сколько поездок каждый клиент совершил.

```
SELECT first_name, last_name, age FROM drivers WHERE age > (
```

SELECT first name, last name, (

SELECT AVG(age) FROM drivers

); - вернет список всех водителей, у которых возраст выше среднего возраста всех водителей в таблице "drivers".

SELECT * FROM clients

WHERE client id IN (

SELECT payment_id FROM feedback WHERE stars < 5

); - вернет список всех клиентов в таблице "clients", у которых есть отзыв (запись в таблице "feedback"), связанный с платежом (записью в таблице "payments"), и в этом отзыве была оценка меньше 5 звезд.

SELECT first_name, last_name, (

character varying (30)

Data Output

first_name

Black

Rocky

White

2

AS "Кол-во поездок" FROM clients;

Сообщения

SELECT COUNT(*) FROM trips WHERE client_id = clien

Notifications

character varying (30)

Кол-во поездок

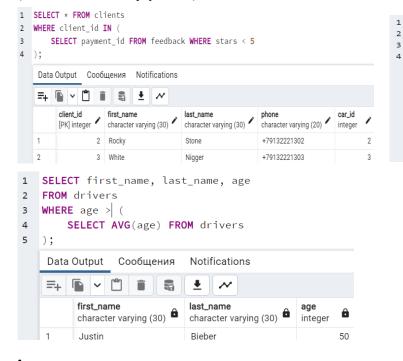
+ ~

last_name

Nigger

Stone

Nigger



1. Создание дополнительного поля age к таблицам business_drivers и clients:

ALTER TABLE clients ADD COLUMN age int;

UPDATE clients

SET age = 29 WHERE client_id = '1';

UPDATE clients

SET age = 72 WHERE client_id = '2';

UPDATE clients

SET age = 19 WHERE client_id = '3';

ALTER TABLE business_drivers ADD COLUMN age int;

UPDATE business_drivers

SET age = 31 WHERE driver_id = '4';

UPDATE business_drivers

SET age = 65 WHERE driver_id = '5';

UPDATE business_drivers

SET age = 23 WHERE driver_id = '6';

SELECT first_name, last_name, age

FROM drivers

WHERE age > 40

UNION

SELECT first_name, last_name, age

FROM business_drivers

WHERE age > 40 - вернет список всех водителей из таблиц "drivers" и "business drivers", у которых возраст больше 40 лет.



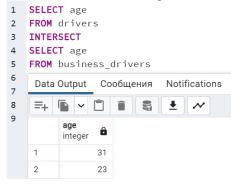
j. SELECT age

FROM drivers

INTERSECT

SELECT age

FROM business_drivers - вернет список возрастов, которые присутствуют и в таблице "drivers", и в таблице "business drivers".



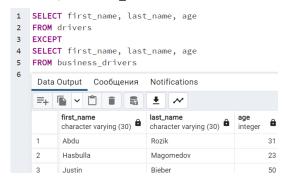
k. SELECT first_name, last_name, age

FROM drivers

EXCEPT

SELECT first_name, last_name, age

FROM business_drivers - вернет список водителей из таблицы "drivers", чьи данные (имя, фамилия, возраст) не встречаются в таблице "business drivers".



l. SELECT first_name, last_name, age,

CASE

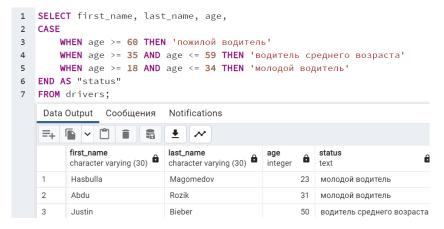
WHEN age >= 60 THEN 'пожилой водитель'

WHEN age >= 35 AND age <= 59 THEN 'водитель среднего возраста'

WHEN age >= 18 AND age <= 34 THEN 'молодой водитель'

END AS "status"

FROM drivers; - вернет список водителей из таблицы "drivers" с добавлением столбца "status", который будет содержать информацию о возрасте водителя в виде текстового описания ("молодой водитель", "водитель среднего возраста", "пожилой водитель").



m. SELECT *

FROM drivers

INNER JOIN business_drivers ON drivers.age = business_drivers.age; - вернет только те строки, в которых возраст водителя из таблицы "drivers" соответствует возрасту водителя из таблицы "business_drivers".

SELECT *

FROM drivers

LEFT JOIN business_drivers ON drivers.age = business_drivers.age; - вернет все строки из таблицы "drivers", а также соответствующие строки из таблицы "business_drivers", при этом если нет соответствующей строки в таблице "business_drivers", то значения возвращаются как NULL.

SELECT *

FROM drivers

RIGHT JOIN business_drivers ON drivers.age = business_drivers.age; - вернет все строки из таблицы "business_drivers", а также соответствующие строки из таблицы "drivers", при этом если нет соответствующей строки в таблице "drivers", то значения возвращаются как NULL.

SELECT *

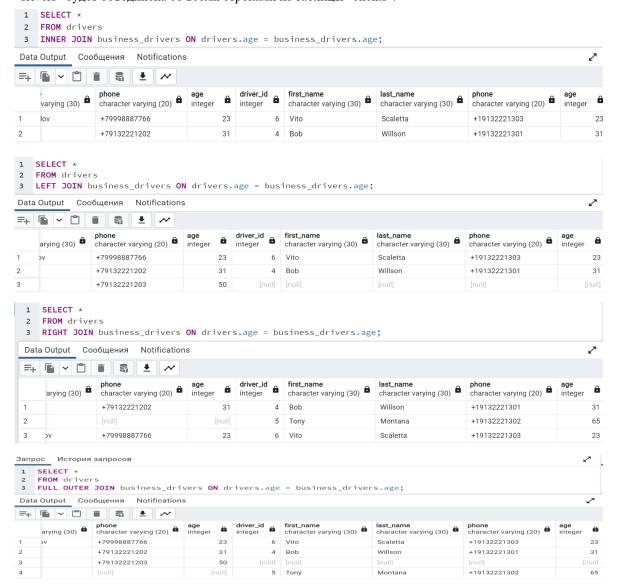
FROM drivers

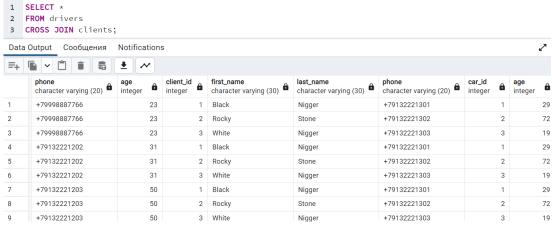
FULL OUTER JOIN business_drivers ON drivers.age = business_drivers.age; - вернет все строки из таблицы "drivers" и все строки из таблицы "business_drivers", при этом если нет соответствующей строки в одной из таблиц, то значения возвращаются как NULL.

SELECT *

FROM drivers

CROSS JOIN clients; - вернет все возможные комбинации строк из таблиц "drivers" и "clients", т.е. каждая строка из таблицы "drivers" будет объединена со всеми строками из таблицы "clients".





(10, 'Vodila', 2);

WITH RECURSIVE managers AS (
 SELECT structure_id, employee_name, manager
 FROM taxi_structure
 WHERE employee_name = 'Chief Keef'
 UNION ALL
 SELECT i.structure_id, i.employee_name, i.manager
 FROM taxi_structure AS i
 INNER JOIN managers AS e ON i.structure_id = e.manager

SELECT structure_id, employee_name, manager

(9, 'Chief Keef', 6),

FROM managers; - запрос используется для построения иерархии руководства, начиная с сотрудника "Chief Keef". запрос вернет столбцы "structure_id", "employee_name" и "manager" для всех руководителей в иерархии данного сотрудника.

