|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ** |
| **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** |
| **«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»** |
|  |
| Институт информационных технологий и управления в технических системах |
| (полное название института) |
|  |
| кафедра «Информационные системы» |
| (полное название кафедры) |

**Расчётно-графическая работа**

по дисциплине **«**Теория баз данных»

Вариант 17

Выполнил

студент ИИТУТС

группы ИС/б-18-2-о

Радыгина Екатерина

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | | | | | | |
|  | (должность, учёная степень преподавателя) | | | | | | | | |
|  |  | | | | | | | | |
|  | (ФИО преподавателя) | | | | | | | | |
|  | « |  | » |  |  |  | 20 | 20 | г. |
|  |  | | | | | | | | |
|  | (оценка) | | | | | | | | |
|  |  | | | | | | | | |
|  |  | | | | | | | | |

Севастополь 2020

**1 Цель работы**

Выработка и закрепление у учащихся практических навыков по работе с реляционными базами данных, изучить основы языка запросов SQL, научиться создавать таблицы и осуществлять элементарные выборки.

**2 Ход работы**

В ходе выполнения работы необходимо выполнить три задания:

**2.1 Реализовать запросы на языке SQL.**

Дана реляционная модель:

S (№\_поставщика, фамилия, состояние, город)

P (№\_детали, название, цвет, вес, город)

SP (№\_поставщика, №\_детали, количество)

S (id, FIO, money, city)

|  |  |  |  |
| --- | --- | --- | --- |
| id | FIO | money | city |
| 1 | Иванов И.И. | 50000 | Лондон |
| 2 | Петров П.П. | 20000 | Москва |
| 3 | Сидоров С.С. | 150000 | Париж |
| 4 | Васильев В.В. | 300000 | Лондон |
| 5 | Семенов С.С. | 70000 | Киев |

P (id, name, color, wieght, city)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | name | color | wieght | city |
| 1 | Болт | #000000 | 34 | Киев |
| 2 | Винт | #FF0000 | 67 | Киев |
| 3 | Гайка\_мал | #FFFF00 | 18 | Лондон |
| 4 | Гвоздь | #FFFFFF | 56 | Париж |
| 5 | Петля | #00FF00 | 90 | Москва |

SP (id\_post, id\_det, quantity)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id\_post | id\_det | quantity |  | id\_post | id\_det | quantity |  | id\_post | id\_det | quantity |
| 2 | 3 | 120 |  | 1 | 5 | 100 |  | 1 | 3 | 14 |
| 2 | 4 | 190 |  | 2 | 2 | 20 |  | 4 | 5 | 22 |
| 2 | 5 | 15 |  | 3 | 1 | 150 |  | 1 | 2 | 67 |
| 1 | 4 | 18 |  | 4 | 3 | 30 |  | 3 | 3 | 15 |
| 3 | 5 | 190 |  | 5 | 4 | 18 |  | 4 | 1 | 18 |
| 5 | 5 | 200 |  | 2 | 1 | 25 |  | 5 | 1 | 217 |

1. Создать БД firma.fdb.

create database 'D:\firma.fdb'

user 'SYSDBA' password 'masterkey'

page\_size = 4096

default character set win1251;

2. Создать таблицы и заполнить их значениями, представленными в таблицах S (рисунок 1), P (рисунок 2), SP (рисунок 3).

CREATE TABLE s(

id INTEGER NOT NULL PRIMARY KEY,

fio VARCHAR(20),

money INTEGER,

city VARCHAR(15)

);

CREATE TABLE p(

id INTEGER NOT NULL PRIMARY KEY,

name VARCHAR(20),

color VARCHAR(15),

weight INTEGER,

city VARCHAR(20)

);

CREATE TABLE sp(

id\_post INTEGER,

id\_det INTEGER,

quantity INTEGER

);

INSERT INTO s

VALUES(1, 'Ivanov I.I.', 50000, 'London');

INSERT INTO s

VALUES(2, 'Petrov P.P.', 20000, 'Moskva');

INSERT INTO s

VALUES(3, 'Sidorov S.S.', 150000, 'Paris');

INSERT INTO s

VALUES(4, 'Vasilev V.V.', 300000, 'London');

INSERT INTO s

VALUES(5, 'Semenov S.S.', 70000, 'Kiev');

INSERT INTO p

VALUES(1, 'Bolt', '#000000', 34, 'Kiev');

INSERT INTO p

VALUES(2, 'Vint', '#ff0000', 67, 'Kiev');

INSERT INTO p

VALUES(3, 'Gayka\_mal', '#ffff00', 18, 'London');

INSERT INTO p

VALUES(4, 'Gvozd', '#ffffff', 56, 'Paris');

INSERT INTO p

VALUES(5, 'Petlya', '#00ff00', 90, 'Moskva');

INSERT INTO sp VALUES(2,3,120);

INSERT INTO sp VALUES(2,4,190);

INSERT INTO sp VALUES(2,5,15);

INSERT INTO sp VALUES(1,4,18);

INSERT INTO sp VALUES(3,5,190);

INSERT INTO sp VALUES(5,5,200);

INSERT INTO sp VALUES(1,5,100);

INSERT INTO sp VALUES(2,2,20);

INSERT INTO sp VALUES(3,1,150);

INSERT INTO sp VALUES(4,3,30);

INSERT INTO sp VALUES(5,4,18);

INSERT INTO sp VALUES(2,1,25);

INSERT INTO sp VALUES(1,3,14);

INSERT INTO sp VALUES(4,5,22);

INSERT INTO sp VALUES(1,2,67);

INSERT INTO sp VALUES(3,3,15);

INSERT INTO sp VALUES(4,1,18);

INSERT INTO sp VALUES(5,1,217);

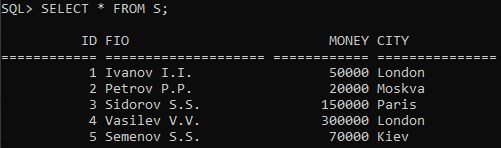


Рисунок 1 - Таблица S

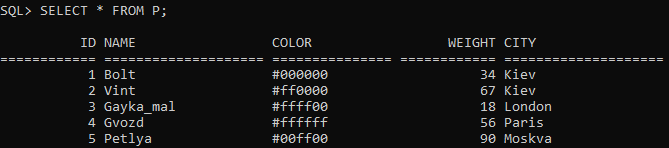


Рисунок 2 – Таблица P

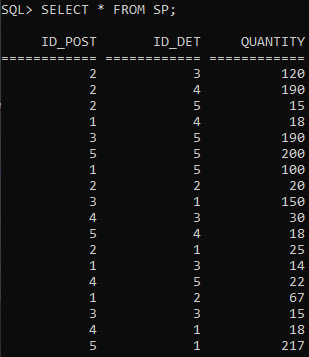


Рисунок 3 – Таблица SP

3. Выдать номера всех поставляемых деталей (рисунок 4).

SELECT id\_det FROM sp;

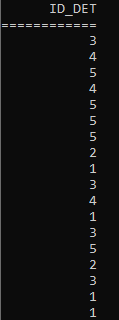


Рисунок 4 – Номера деталей

4. Добавить столбец «счёт в банке» в таблицу P (рисунок 5).

ALTER TABLE p ADD bank\_account INTEGER;

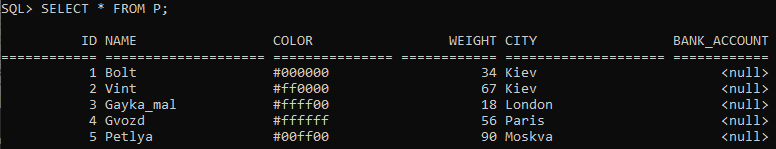


Рисунок 5 – Добавление bank\_account в таблицу P

5. Выделить всех поставщиков, не проживающих в Лондоне (рисунок 6).

SELECT \* FROM s

WHERE 'London' NOT LIKE city;

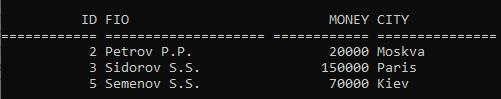


Рисунок 6 ­– Поставщики не из Лондона

6. Выдать номера и состояния для поставщиков, находящихся в Москве (рисунок 7).

SELECT id, money FROM s

WHERE city = 'Moskva';



Рисунок 7 – Выполнение запроса

7. Выдать номера и состояния для поставщиков, проживающих в Париже, и состояние которых меньше 50 тыс.$ (в порядке убывания состояний) (рисунок 8).

SELECT id, money FROM s

WHERE 'Paris' LIKE city AND

money < 50000

ORDER BY money DESC;

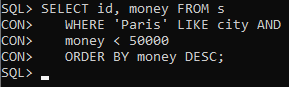


Рисунок ­8 – Таких поставщиков нет

8. Выдать полные характеристики всех поставщиков (рисунок 9).

SELECT \* FROM s;

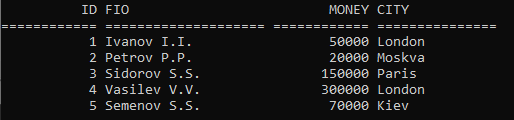


Рисунок 9 – Вывод таблицы s

9. Выдать сведения о деталях, вес которых в диапазоне от 30 до 60 (рисунок 10).

SELECT \* FROM p

WHERE weight >= 30 AND

weight <= 60;

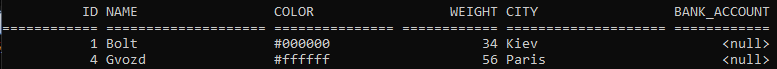


Рисунок 10 – Сведения о деталях по условию

10. Выдать номер и вес каждой детали в граммах для всех деталей, если вес указан в фунтах (454 гр.) (рисунок 11).

SELECT id, weight\*1000 AS weight\_g, 'gr.' FROM p;

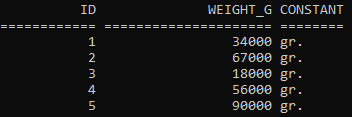


Рисунок 11 – Вывод деталей по условию

11. Выдать номера деталей, вес которых 12, 18, 34 (рисунок 12).

SELECT id FROM p

WHERE weight = 12 OR

weight = 18 OR

weight = 34;

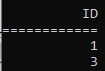


Рисунок 12 ­– Вывод номеров деталей по условию

12. Выдать все детали, название которых начинается с буквы «Г» (рисунок 13).

SELECT \* FROM p

WHERE LEFT(name,1) = 'G';

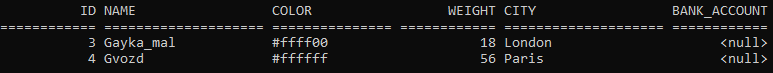


Рисунок 13 – Вывод деталей по условию

13. Выдать все детали, название которых заканчивается на букву «T» (рисунок 14).

SELECT \* FROM p

WHERE RIGHT(name,1) = 't';

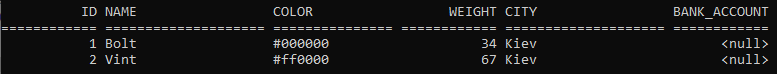


Рисунок 14 – Вывод деталей по условию

14. Выдать все детали, название которых содержит в середине букву «о» (рисунок 15).

SELECT \* FROM p

WHERE POSITION('o',

SUBSTRING(name

FROM 1 FOR (CHAR\_LENGTH(name)-1))) <> 0;

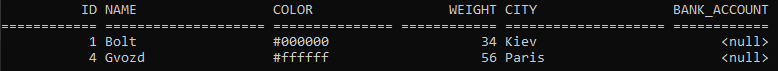


Рисунок 15 – Вывод деталей по условию

15. Выдать все детали, название которых содержит третью букву «л» (рисунок 16).

SELECT \* FROM p

WHERE POSITION('l', name) = 3;



Рисунок 16 – Вывод деталей по условию

16. Выдать все детали, название которых содержит «\_» (рисунок 17).

SELECT \* FROM p

WHERE POSITION('\_', name) <> 0;



Рисунок 17 – Вывод деталей по условию

17. Выдать все комбинации информации о таких поставщиках и деталях, которые размещены в одном и том же городе (рисунки 18 – 19).

SELECT \* FROM p JOIN s

ON p.city = s.city;

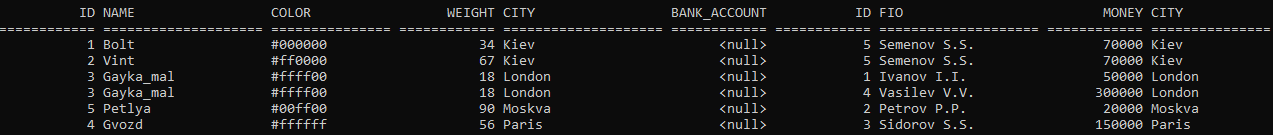


Рисунок 18 – Выполнение запроса с использованием JOIN

SELECT \* FROM p, s

WHERE p.city = s.city;

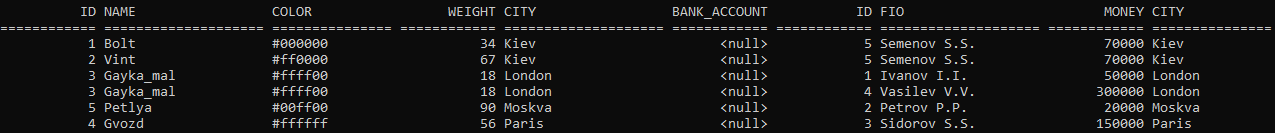


Рисунок 19 – Выполнение запроса без использования JOIN

18. Изменить цвет детали 2 на жёлтый (#FFFF00), увеличить её вес на пять и установить значение города – неизвестен (рисунок 20).

UPDATE p

SET color = '#ffff00',

weight = weight + 5,

city = NULL

WHERE id = 2;

SELECT \* FROM p

WHERE id = 2;



Рисунок 20 – Выполнение запроса

19. Удалить столбец «счёт в банке» из таблицы P (рисунок 21).

ALTER TABLE p DROP bank\_account;

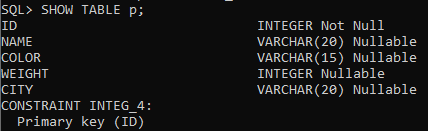


Рисунок 21 – Удаление столбца из таблицы p

20. Удвоить состояние всех поставщиков, находящихся в Лондоне (рисунок 22).

UPDATE s

SET money = money \* 2

WHERE city = 'London';

SELECT \* FROM s

WHERE city = 'London';

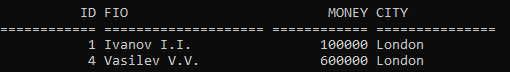


Рисунок 22 – Выполнение запроса

21. Удалить всех поставщиков из Лондона (рисунок 23).

DELETE FROM s

WHERE city = 'London';

SELECT \* FROM s;

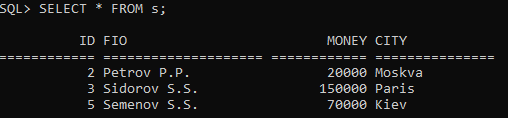


Рисунок 23 – Удаление поставщиков из Лондона

22. Добавить в таблицу Р запись: деталь – 6, город – Минск, цвет – #FF9900, название – лента, вес – неизвестен (рисунок 24).

INSERT INTO p

VALUES(6,'Lenta','#ff9900',NULL,'Minsk');

SELECT \* FROM p;

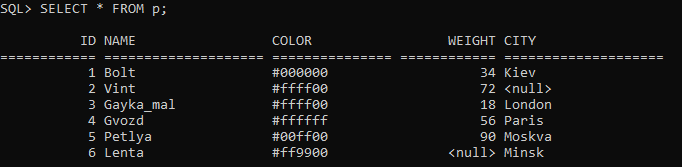


Рисунок 24 – Добавление записи в таблицу p

23. Выведите номера деталей, для которых не определено значение веса (рисунок 25).

SELECT id FROM p

WHERE weight IS NULL;



Рисунок 25 – Выполнение запроса

24. Выдать общее количество поставщиков, поставляющих в настоящее время детали (рисунок 26).

SELECT COUNT(DISTINCT id\_post)

FROM sp;



Рисунок 26 – Выполнение запроса

25. Для каждой поставляемой детали выдать её номер и объем поставок (рисунок 27).

SELECT id\_det, SUM(quantity)

FROM sp

GROUP BY id\_det;

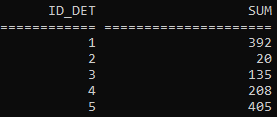


Рисунок 27 – Выполнение запроса

26. Для каждой детали выдать её номер и объем поставок за исключением поставщика под номером 3 (рисунок 28).

SELECT id\_det, SUM(quantity)

FROM sp

WHERE id\_post <> 3

GROUP BY id\_det;

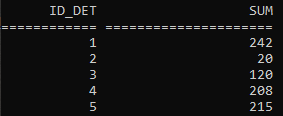


Рисунок 28 – Выполнение запроса

27. Выдать номера деталей для всех деталей, поставляемых более чем одним поставщиком (рисунок 29).

SELECT DISTINCT id\_det

FROM sp AS o

WHERE EXISTS(

SELECT \* FROM sp AS i

WHERE i.id\_det = o.id\_det

AND i.id\_post <> o.id\_post);

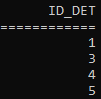


Рисунок 29 – Выполнение запроса

28. Установить связи между таблицами S, P и SP (внешние ключи) (рисунки 30 – 32).

ALTER TABLE sp

ADD CONSTRAINT id\_post FOREIGN KEY(id\_post)

REFERENCES s(id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE sp

ADD CONSTRAINT id\_det FOREIGN KEY(id\_det)

REFERENCES p(id)

ON DELETE CASCADE

ON UPDATE CASCADE;

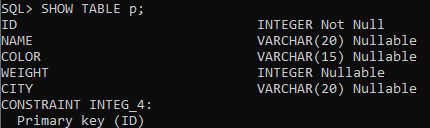


Рисунок 30 – Таблица p

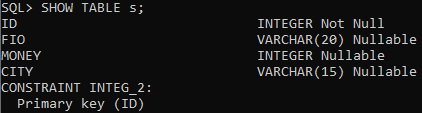


Рисунок 31 – Таблица s

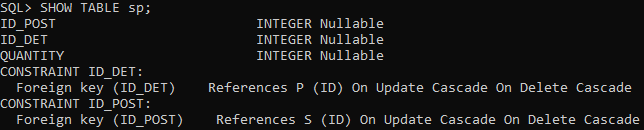


Рисунок 32 – Таблица sp

29. Вывести информацию об именах поставщиков и о том, в каком количестве и какие детали они поставляют (рисунки 33 – 34).

SELECT s.fio, sp.id\_det, sp.quantity

FROM s JOIN sp

ON sp.id\_post = s.id

ORDER BY s.id;

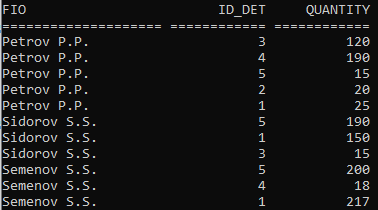


Рисунок 33 – Выполнение запроса с использованием JOIN

SELECT s.fio, sp.id\_det, sp.quantity

FROM s,sp

WHERE sp.id\_post = s.id

ORDER BY s.id;

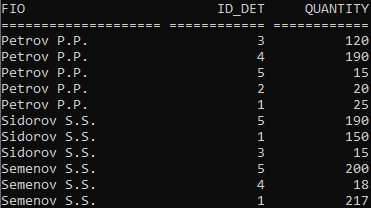


Рисунок 34 – Выполнение запроса без использования JOIN

30. Вывести информацию об имени поставщика и названии деталей, которые он поставляет (рисунки 35 – 36).

SELECT s.fio, p.name

FROM sp JOIN p

ON sp.id\_det = p.id

JOIN s

ON sp.id\_post = s.id

ORDER BY s.id;



Рисунок 35 – Выполнение запроса с использованием JOIN

SELECT s.fio, p.name

FROM s,p,sp

WHERE sp.id\_det = p.id

AND sp.id\_post = s.id

ORDER BY s.id;

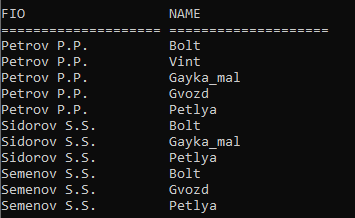


Рисунок 36 – Выполнение запроса без использования JOIN

31. Вывести все пары поставщиков, живущих в одном городе. Исключить комбинации продавцов с ними же, а также дубликаты строк, выводимые в обратном порядке (рисунок 37 – 38).

Добавим двух поставщиков из Лондона, которые были удалены в задании 21.

SELECT DISTINCT a.fio

FROM s AS a JOIN s AS b

ON a.city = b.city

AND NOT(a.fio = b.fio);

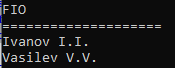


Рисунок 37 – Выполнение запроса с использованием JOIN

SELECT DISTINCT a.fio

FROM s a, s b

WHERE a.city = b.city

AND NOT(a.fio = b.fio);

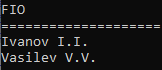


Рисунок 38 – Выполнение запроса без использования JOIN

32. Вывести фамилии всех поставщиков, которые поставляют детали с номером 1 (рисунок 39).

SELECT fio

FROM s

WHERE id IN (

SELECT id\_post

FROM sp WHERE id\_det = 1);

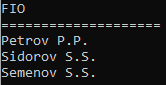


Рисунок 39 – Вывод поставщиков по условию

33. Вывести фамилии поставщиков, которые поставляют, по крайней мере, одну красную деталь (рисунок 40).

SELECT fio FROM s

WHERE id IN (

SELECT id\_post FROM sp

WHERE id\_det IN (

SELECT id FROM p

WHERE color = '#FF0000'));

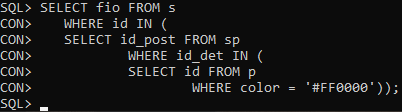


Рисунок 40 – Таких деталей нет

34. Вывести номера и фамилии поставщиков, которые поставляют по крайней мере одну деталь, поставляемую поставщиком 3 (рисунки 41 – 42).

SELECT DISTINCT sp.id\_post, s.fio

FROM sp JOIN s

ON sp.id\_post = s.id

WHERE sp.id\_det IN

(SELECT sp.id\_det FROM sp

WHERE sp.id\_post = 3);

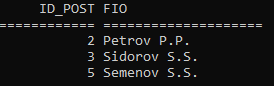


Рисунок 41 – Выполнение запроса с использованием JOIN

SELECT DISTINCT sp.id\_post, s.fio

FROM sp, s

WHERE sp.id\_post = s.id

AND sp.id\_det IN

(SELECT sp.id\_det

FROM sp

WHERE sp.id\_post = 3);

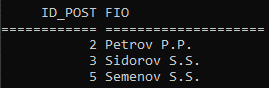


Рисунок 42 – Выполнение запроса без использования JOIN

35. Вывести номера поставщиков, для которых в базе существует информация о номерах поставляемых деталей (рисунок 43).

SELECT id FROM s

WHERE EXISTS(SELECT \*

FROM sp

WHERE s.id = sp.id\_post);

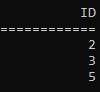


Рисунок 43 – Вывод поставщиков по условию

36. Вывести номера поставщиков, для которых не существует информация о номерах поставляемых деталей (рисунок 44).

SELECT id FROM s

WHERE NOT EXISTS(

SELECT \* FROM sp

WHERE s.id = sp.id\_post);

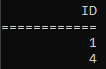


Рисунок 44 – Вывод поставщиков по условию

37. Вывести номера поставщиков, которые поставляют все детали (рисунок 45).

SELECT id, fio FROM s

WHERE id IN (

SELECT id\_post

FROM sp

GROUP BY id\_post

HAVING COUNT(DISTINCT id\_det) = (

SELECT COUNT(id) FROM p));

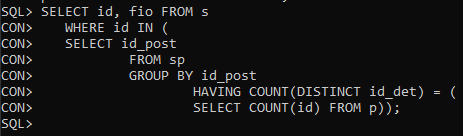


Рисунок 45 – Таких поставщиков нет

38. Вывести номера только тех деталей, которые не поставляет поставщик 1 (рисунок 46).

Так как этот поставщик был удален в задании 21, все связанные с ним данные были удалены из таблицы sp.

SELECT id FROM p

WHERE id IN(

SELECT id\_det FROM sp

WHERE NOT(id\_post = 1));

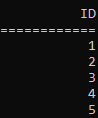


Рисунок 46 – Выполнение запроса

39. Вывести номера деталей, которые имеют вес больше 34 или поставляются поставщиком 3 (рисунок 47).

SELECT id FROM P

WHERE weight> 34 UNION

SELECT id\_det FROM SP

WHERE id\_post = 3;

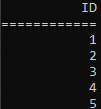


Рисунок 47 – Выполнение запроса

40. Вывести названия деталей, которые поставляет поставщик с номером 2 (рисунок 48).

SELECT name FROM p

WHERE id IN(

SELECT id\_det FROM sp

WHERE id\_post = 2);

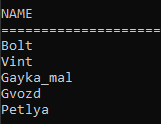


Рисунок 48 – Выполнение запроса

41. Вывести номера поставщиков с состоянием меньшим, чем текущее максимальное состояние в таблице поставщиков (рисунок 49).

SELECT id FROM s

WHERE money < (

SELECT MAX(money) FROM s);



Рисунок 49 – Выполнение запроса

42. Выдать номера, состояние и город для всех поставщиков, у которых состояние равно или больше среднего по городу (рисунок 50).

SELECT id, money, city FROM s a

WHERE money >= (

SELECT AVG (money) FROM s b

WHERE b.city = a.city);

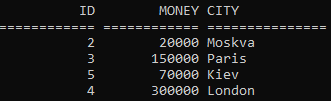


Рисунок 50 – Выполнение запроса

43. Для каждой поставляемой детали получить её номер и общий объём поставок. Сохранить результат в новой таблице (рисунок 51).

CREATE TABLE for\_time(

id\_det CHAR (6),

amount INTEGER);

INSERT INTO for\_time (id\_det, amount)

SELECT id\_det, SUM(quantity)

FROM sp

GROUP BY id\_det;

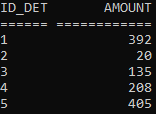


Рисунок 51 – Выполнение запроса

44. Вывести имена поставщиков и названия деталей, которые поставляются в наибольшем объеме (рисунки 52 – 53).

SELECT p.name, s.fio, MAX(sp.quantity)

FROM sp JOIN p

ON p.id = sp.id\_det JOIN s

ON s.id = sp.id\_post

GROUP BY p.name, s.fio, quantity

ORDER BY quantity;

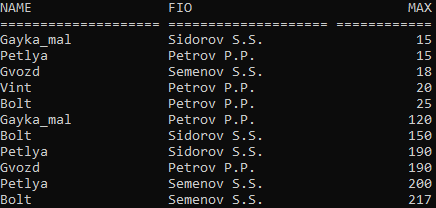


Рисунок 52 – Выполнение запроса с использованием JOIN

SELECT p.name, s.fio, MAX(sp.quantity)

FROM sp, p, s

WHERE p.id = sp.id\_det AND s.id = sp.id\_post

GROUP BY p.name, s.fio, sp.quantity

ORDER BY sp.quantity;

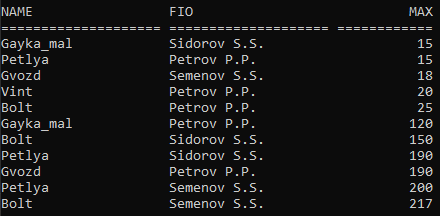


Рисунок 53 – Выполнение запроса без использования JOIN

**2.2 Реализовать запросы на языке SQL.**

1. Выдать цвета всех поставляемых деталей, сгруппировав их с поставщиками из Керчи (рисунок 54).

Создадим исходную таблицу заново. Так как в исходной таблице нет поставщиков из Керчи, изменим город на этот у поставщика из Парижа и Киева.

SELECT s.fio, p.color FROM s, p

WHERE s.city = 'Kerch'

GROUP BY s.fio, p.color;

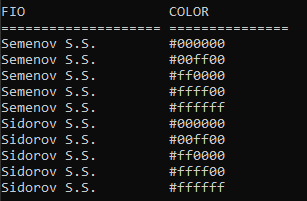


Рисунок 54 – Выполнение запроса

2. Для каждой поставляемой детали получить ее номер и общий объем поставок без учета детали «втулка» (рисунок 55).

Так как в исходной таблице нет детали «втулка», заменим на нее деталь «Петля».

SELECT id\_det, SUM(quantity) FROM sp

WHERE id\_det IN(

SELECT id FROM p

WHERE NOT(p.name = 'Vtulka'))

GROUP BY id\_det;

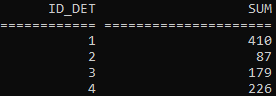


Рисунок 55 – Выполнение запроса

3. Вывести номера поставщиков с состоянием меньшим, чем текущее максимальное состояние в таблице S (рисунок 56).

SELECT id FROM s

WHERE money < (

SELECT MAX(money) FROM s);

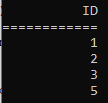


Рисунок 56 – Выполнение запроса

4. Для каждой детали выдать её поставщика, город производства в порядке возрастания количества (рисунок 57).

SELECT fio, city, COUNT(id\_det) FROM s, sp

WHERE id = id\_post

GROUP BY fio, city

ORDER BY COUNT(DISTINCT id\_det) ASC;



Рисунок 57 – Выполнение запроса

**2.3 Реализовать запросы на языке SQL.**

Р1 (N\_аквариума, общее\_количество\_рыб, температура, тип\_аэрации, тип\_флоры);

Р2 (вид\_рыбы, вид\_корма, код\_содержания, страна\_происхождения);

Р3 (код\_содержания, минимальная\_температура, максимальная\_температура, тип\_флоры, тип\_аэрации);

Р4 (N\_аквариума, вид\_рыбы, количество).

1. Получить список аквариумов, температур и количества рыб в каждом из них в порядке возрастания температур

2. Скопировать таблицу Р1.

**Вывод**

В результате выполнения расчётно-графической работы были улучшены практические навыки работы с БД и таблицами, навыки создания и использования запросов на языке SQL.