Лабораторная работа № 2

студента группы ИТз-221

Дмитриева Дмитрия Анатольевича

*Выполнение:\_\_\_\_\_\_\_\_\_ Защита:\_\_\_\_\_\_\_\_\_\_\_*

Язык SQL. Выборка данных

*Цель работы***:** изучить язык запросов на выборку данных из БД.

Содержание работы

1. Напишите запрос для вывода названия, автора и цен книг из таблицы Книги.

2. Напишите запрос для вывода всех уникальных имен авторов книг в текущем порядке из таблицы Книги.

3. Напишите запрос для вывода кодов всех книг и их цен, поставленных из Москвы.

4. Напишите запрос для выборки всех заказчиков, чьи имена начинаются с  
буквы A

5. Напишите запрос для выборки всех книг, в названии которых есть словосочетание «база данных» или «базы данных» или «баз данных».

6. Напишите запрос для подсчета суммы всех заказов на 31 января 2018 года.

7. Напишите запрос для подсчета числа различных не NULL-значений полей Адрес и Телефон в таблице Поставщики.

8. Напишите запрос для подсчета общей суммы заказов за каждый день с 01.02.2018 по 01.04.2018.

9. Напишите запрос к таблице Заказы для определения средней суммы предоплаты за каждый месяц.

10. Напишите запрос, который бы выводил все заказы с величиной предоплаты выше средней. Выведите код заказа, дату, имя заказчика и величину предоплаты.

11. Напишите запрос, который вычислил бы суммарные остатки количества книг, поставленных каждым поставщиком. Выведите код поставщика, количество видов книг и суммарный остаток.

12. Напишите запрос, который бы вывел все заказы, сделанные за первый квартал 2001 года заказчиками из г. Бреста, и подсчитал количество оплаченных и неоплаченных заказов.

13. Напишите запрос для получения списка книг, цены которых превысят 10000 руб. в случае повышения цен на 12%.

14. Напишите запрос, который бы выбрал общую сумму всех приобретений для каждого заказчика.

15. Напишите запрос для получения списка книг, которые не были ни разу заказаны.

16. Напишите аналогичные запросы для индивидуальной предметной области.

**Ход работы:**

***Вариант 8***

1) SELECT name, author, price FROM books

2) SELECT DISTINCT author FROM books

3) SELECT id, price FROM books WHERE publish\_table = 'Москва'

4) SELECT \* FROM customers WHERE LEFT(name, 1) = 'A'

5) SELECT \* FROM books WHERE name like '%баз% данных%'

6) SELECT SUM(books.price) FROM orders

INNER JOIN books ON books.id = orders.book\_id

WHERE orders.date = '2018-01-31'

7) SELECT COUNT(\*) FROM suppliers as S

INNER JOIN (

SELECT address, phone FROM suppliers GROUP BY address, phone

HAVING COUNT(\*) = 1

) T

ON S.address = T.address AND S.phone = T.phone

8) SELECT SUM(books.price) FROM orders

INNER JOIN books ON books.id = orders.book\_id

WHERE orders.date BETWEEN '2018-02-01' AND '2018-04-01'

9) SELECT DATE\_TRUNC('month', date)::date as month, AVG(books.price::NUMERIC)

FROM orders

INNER JOIN books ON books.id = orders.book\_id

GROUP BY month

10) SELECT orders.id, orders.date, customers.name FROM orders

INNER JOIN customers ON customers.id = orders.customer\_id

INNER JOIN books ON books.id = orders.book\_id

WHERE orders.pre\_payed > books.price::NUMERIC

11) SELECT suppliers.id, COUNT(supplies.id) AS "Количество видов книг", COUNT(books.rest) AS "Суммарный остаток" FROM suppliers

INNER JOIN supplies ON supplies.supplier\_id = suppliers.id

INNER JOIN books ON books.id = supplies.book\_id

GROUP BY suppliers.id

12) 1) SELECT orders.\* FROM orders

INNER JOIN customers ON customers.id = orders.customer\_id

WHERE EXTRACT(month FROM date) BETWEEN 1 AND 3

AND EXTRACT(year FROM date) = 2001

AND customers.address = 'Брест'

2) WITH orders\_in\_2001 AS

(

SELECT customers.id as customer\_id, orders.paid

FROM orders

INNER JOIN customers ON customers.id = orders.customer\_id

WHERE EXTRACT(month FROM date) BETWEEN 1 AND 3

AND EXTRACT(year FROM date) = 2001

AND customers.address = 'Брест'

)

SELECT orders\_in\_2001.customer\_id,

(SELECT COUNT(\*) FROM orders\_in\_2001 WHERE paid = 'yes') AS "Количество оплаченных заказов",

(SELECT COUNT(\*) FROM orders\_in\_2001 WHERE paid = 'no') AS "Количество неоплаченных заказов"

FROM orders\_in\_2001

GROUP BY orders\_in\_2001.customer\_id

13) SELECT \* FROM books

WHERE price::NUMERIC \* 1.12 > 10000

14) SELECT customers.id, SUM(books.price) FROM customers

INNER JOIN orders ON orders.customer\_id = customers.id

INNER JOIN books ON books.id = orders.book\_id

GROUP BY customers.id

15) SELECT \* FROM books

WHERE books.id NOT IN (

SELECT books.id FROM books

INNER JOIN orders ON orders.book\_id = books.id

)

### INDIVIDUAL TASK

16) SELECT number, category, rent\_per\_day FROM rooms

16.1) SELECT DISTINCT category FROM rooms

16.2) SELECT id, rent\_per\_day FROM rooms WHERE category = 'Люкс'

16.3) SELECT \* FROM administrators WHERE LEFT(first\_name, 1) = 'Д'

16.4) SELECT SUM(bookings.price) FROM bookings

WHERE date = '2023-03-13'

16.5) SELECT clients.id, clients.first\_name, SUM(bookings.total\_cost) FROM bookings

JOIN clients ON bookings.client\_id = clients.id

GROUP BY clients.id

16.6) SELECT rooms.number, rooms.category

FROM bookings

JOIN rooms ON bookings.room\_id = rooms.id

WHERE bookings.date = '2024-04-17';

16.7) SELECT clients.first\_name, clients.last\_name, AVG(bookings.total\_cost::NUMERIC) AS average\_booking\_cost

FROM clients

JOIN bookings ON clients.id = bookings.client\_id

GROUP BY clients.first\_name, clients.last\_name;

16.8) SELECT room\_categories.category, SUM(bookings.total\_cost) AS revenue, COUNT(bookings.id) AS bookings\_count

FROM (

SELECT rooms.id, rooms.category

FROM rooms

GROUP BY rooms.id, rooms.category

) AS room\_categories

JOIN bookings ON room\_categories.id = bookings.room\_id

GROUP BY room\_categories.category

ORDER BY SUM(bookings.total\_cost) DESC

**Контрольные вопросы:**

1. Операторы в SQL:

- BETWEEN: Используется для выбора значений в пределах заданного диапазона.

- IN: Позволяет проверить, содержится ли значение в списке значений.

- LIKE: Используется для сравнения значений с шаблоном, используя специальные символы.

2. Агрегирующие функции в SQL выполняют вычисления над набором значений и возвращают одно обобщенное значение. Некоторые из них включают SUM (сумма), AVG (среднее), MAX (максимальное значение), MIN (минимальное значение) и COUNT (количество записей).

3. Функции SUM, AVG, MAX, MIN, COUNT называются агрегирующими функциями, потому что они агрегируют (объединяют) данные из нескольких строк в одно значение. Каждая из этих функций выполняет определенное вычисление над набором значений.

4. Агрегирующие функции требуют группировки данных, чтобы работать корректно. Без использования модификатора GROUP BY, SQL не знает, как группировать данные, и поэтому не может применять агрегирующие функции к полям.

5. COUNT(\*) возвращает количество строк в результате запроса, в то время как COUNT(DISTINCT <Имя>) возвращает количество уникальных значений в указанном столбце.

6. Предложение WHERE используется для фильтрации строк, которые должны быть включены в результат запроса, в то время как предложение HAVING используется для фильтрации результатов агрегированных функций, которые включены в результат запроса.

7. Типы соединений таблиц в SQL:

- INNER JOIN: Возвращает только строки, имеющие совпадающие значения в обеих таблицах.

- LEFT JOIN (или LEFT OUTER JOIN): Возвращает все строки из левой таблицы и соответствующие строки из правой таблицы. Если нет совпадений, для правой таблицы возвращается NULL.

- RIGHT JOIN (или RIGHT OUTER JOIN): Возвращает все строки из правой таблицы и соответствующие строки из левой таблицы. Если нет совпадений, для левой таблицы возвращается NULL.

- FULL JOIN (или FULL OUTER JOIN): Возвращает строки, когда есть совпадение в любой из таблиц. Все строки из обеих таблиц возвращаются, где нет совпадений, добавляются NULL значения в столбцы другой таблицы.

- CROSS JOIN: Возвращает декартово произведение двух таблиц, т.е. все возможные комбинации строк из двух таблиц.

**Вывод:** изучил язык запросов на выборку данных из БД.