**Markdown report**

**# SQL Assignment - Basic Queries**

### 1. List all customers with their full name and city.

```sql

SELECT

CONCAT(first\_name, ' ', second\_name) AS full\_name, city

FROM customers;

2. Show all books priced above 2000.

sql

Copy

Edit

SELECT title, price

FROM books

WHERE price > 2000;

3. List customers who live in 'Nairobi'.

sql

Copy

Edit

SELECT \*

FROM customers

WHERE city = 'Nairobi';

4. Retrieve all book titles that were published in 2023.

sql

Copy

Edit

SELECT title, published\_date

FROM books

WHERE EXTRACT(YEAR FROM published\_date) = 2023;

Filtering and Sorting

5. Show all orders placed after March 1st, 2025 (NO Data).

sql

Copy

Edit

SELECT order\_id, order\_date

FROM orders

WHERE order\_date > '2025-03-01';

6. List all books ordered, sorted by price (descending).

sql

Copy

Edit

SELECT title, price

FROM books

ORDER BY price DESC;

7. Show all customers whose names start with 'J'.

sql

Copy

Edit

SELECT \*

FROM customers

WHERE first\_name LIKE 'J%';

8. List books with prices between 1500 and 3000.

sql

Copy

Edit

SELECT title, price

FROM books

WHERE price BETWEEN 1500 AND 3000;

Aggregate Functions and Grouping

9. Count the number of customers in each city.

sql

Copy

Edit

SELECT city, COUNT(\*) AS total\_customers

FROM customers

GROUP BY city;

10. Show the total number of orders per customer.

sql

Copy

Edit

SELECT customer\_id, COUNT(\*) AS total\_orders

FROM orders

GROUP BY customer\_id;

11. Find the average price of books in the store.

sql

Copy

Edit

SELECT AVG(price) AS average\_book\_price

FROM books;

12. List the book title and total quantity ordered for each book.

sql

Copy

Edit

SELECT b.title,

SUM(o.quantity::INTEGER) AS total\_quantity\_ordered

FROM books b

JOIN orders o ON b.book\_id = o.book\_id

GROUP BY b.title;

Subqueries

13. Show customers who have placed more orders than the customer with ID = 1.

sql

Copy

Edit

SELECT customer\_id, COUNT(\*) AS total\_orders

FROM orders

GROUP BY customer\_id

HAVING COUNT(\*) > (SELECT COUNT(\*)

FROM orders

WHERE customer\_id = 1);

14. List books that are more expensive than the average book price.

sql

Copy

Edit

SELECT title, price

FROM books

WHERE price > (

SELECT AVG(price)

FROM books

);

15. Show each customer and the number of orders they placed using a subquery in SELECT.

sql

Copy

Edit

SELECT

CONCAT(first\_name, ' ', second\_name) AS full\_name,

(SELECT COUNT(\*)

FROM orders o

WHERE o.customer\_id = c.customer\_id) AS total\_orders

FROM customers c;

**JOINS:**

16. Show the full name of each customer and the titles of books they ordered.

sql

Copy

Edit

SELECT c.first\_name || ' ' || c.second\_name AS full\_name,

b.title

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN books b ON o.book\_id = b.book\_id;

17. List all orders including book title, quantity, and total cost (price × quantity).

sql

Copy

Edit

SELECT

b.title,

o.quantity,

CAST(b.price AS NUMERIC) \* CAST(o.quantity AS INTEGER) AS total\_cost

FROM orders o

JOIN books b ON o.book\_id = b.book\_id;

18. Show customers who haven't placed any orders (LEFT JOIN).

sql

Copy

Edit

SELECT

c.first\_name || ' ' || c.second\_name AS full\_name

FROM customers c

LEFT JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.order\_id IS NULL;

19. List all books and the names of customers who ordered them, if any (LEFT JOIN).

sql

Copy

Edit

SELECT

b.title AS book\_title,

c.first\_name || ' ' || c.second\_name AS customer\_name

FROM books b

LEFT JOIN orders o ON b.book\_id = o.book\_id

LEFT JOIN customers c ON o.customer\_id = c.customer\_id;

20. Show customers who live in the same city (SELF JOIN).

sql

Copy

Edit

SELECT

c1.first\_name || ' ' || c1.second\_name AS customer\_1,

c2.first\_name || ' ' || c2.second\_name AS customer\_2,

c1.city

FROM customers c1

JOIN customers c2 ON c1.city = c2.city AND c1.customer\_id <> c2.customer\_id;

**Combined Logic**

21. Show all customers who placed more than 2 orders for books priced over 2000.

sql

Copy

Edit

SELECT b.price, b.title, COUNT(o.order\_id) AS total\_orders

FROM books b

JOIN orders o ON o.book\_id = b.book\_id

WHERE b.price > 2000

GROUP BY b.price, b.title

HAVING COUNT(o.order\_id) > 2;

22. List customers who ordered the same book more than once.

sql

Copy

Edit

SELECT c.first\_name, COUNT(o.order\_id) AS total\_orders

FROM orders o

JOIN customers c ON c.customer\_id = o.customer\_id

GROUP BY c.first\_name

HAVING COUNT(o.order\_id) > 1;

23. Show each customer's full name, total quantity of books ordered, and total amount spent.

sql

Copy

Edit

SELECT

c.first\_name || ' ' || c.second\_name AS full\_name,

SUM(CAST(o.quantity AS INTEGER)) AS total\_quantity\_ordered,

SUM(CAST(o.quantity AS INTEGER) \* CAST(b.price AS NUMERIC)) AS total\_amount\_spent

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN books b ON o.book\_id = b.book\_id

GROUP BY c.customer\_id, c.first\_name, c.second\_name;

24. List books that have never been ordered.

sql

Copy

Edit

SELECT b.book\_id, b.title

FROM books b

LEFT JOIN orders o ON b.book\_id = o.book\_id

WHERE o.book\_id IS NULL;

25. Find the customer who has spent the most in total (JOIN + GROUP BY + ORDER BY + LIMIT).

sql

Copy

Edit

SELECT

c.first\_name || ' ' || c.second\_name AS full\_name,

COUNT(o.order\_id) AS total\_orders

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

GROUP BY c.first\_name, c.second\_name

ORDER BY total\_orders DESC

LIMIT 1;

26. Write a query that shows, for each book, the number of different customers who have ordered it.

sql

Copy

Edit

SELECT

b.title,

COUNT(DISTINCT o.customer\_id) AS different\_customers

FROM books b

JOIN orders o ON b.book\_id = o.book\_id

GROUP BY b.title;

27. Using a subquery, list books whose total order quantity is above the average order quantity.

sql

Copy

Edit

SELECT b.title,

SUM(CAST(o.quantity AS INTEGER)) AS total\_order\_quantity

FROM books b

JOIN orders o ON b.book\_id = o.book\_id

GROUP BY b.title

HAVING SUM(CAST(o.quantity AS INTEGER)) > (

SELECT AVG(total\_quantity)

FROM (

SELECT SUM(CAST(o.quantity AS INTEGER)) AS total\_quantity

FROM books b

JOIN orders o ON b.book\_id = o.book\_id

GROUP BY b.title

) AS subquery

);

28. Show the top 3 customers with the highest number of orders and the total amount they spent.

sql

Copy

Edit

SELECT

c.first\_name || ' ' || c.second\_name AS full\_name,

COUNT(o.order\_id) AS total\_orders,

SUM(CAST(o.quantity AS INTEGER) \* CAST(b.price AS NUMERIC)) AS total\_amount\_spent

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN books b ON o.book\_id = b.book\_id

GROUP BY c.customer\_id, c.first\_name, c.second\_name

ORDER BY total\_orders DESC

LIMIT 3;

**SQL QUERIES**

**Basic Queries**

**1. List all customers with their full name and city.**

**SELECT**

**CONCAT(first\_name, ' ', second\_name) AS full\_name, city**

**FROM customers;**

**2. Show all books priced above 2000.**

**select title,price from books**

**where price >2000;**

**3. List customers who live in 'Nairobi'.**

**SELECT \***

**FROM customers**

**WHERE city = 'Nairobi';**

**SELECT customer\_id ,city FROM customers**

**where city ='Nairobi';**

**4. Retrieve all book titles that were published in 2023**

**SELECT title, published\_date**

**FROM books**

**WHERE EXTRACT(YEAR FROM published\_date) = 2023;**

**Filtering and Sorting**

**5. Show all orders placed after March 1st, 2025= NO Data**

**SELECT order\_id, order\_date**

**FROM orders**

**WHERE order\_date > '2025-03-01';**

**6.List all books ordered, sorted by price (descending).**

**select title, price**

**from books**

**order by price DESC;**

**7.Show all customers whose names start with 'J'**

**SELECT \***

**FROM customers**

**WHERE first\_name LIKE 'J%';**

**8. List books with prices between 1500 and 3000.**

**SELECT title, price**

**FROM books**

**where price between 1500 and 3000;**

**Aggregate Functions and Grouping**

**9-Count the number of customers in each city.**

**SELECT city, COUNT(\*) AS total\_customers**

**FROM customers**

**Group by city;**

**10.Show the total number of orders per customer.**

**select customer\_id,COUNT(\*) as total\_orders**

**from orders**

**GROUP BY customer\_id;**

**11,Find the average price of books in the store.**

**select AVG (price) as average\_book\_price**

**from books;**

**12.List the book title and total quantity ordered for each book**

**SELECT b.title,**

**SUM(o.quantity::INTEGER) AS total\_quantity\_ordered**

**FROM books b**

**JOIN orders o ON b.book\_id = o.book\_id**

**GROUP BY b.title;**

**SUBQUERIES:**

**13.Show customers who have placed more orders than customer with ID = 1**

**SELECT customer\_1d, COUNT(\*) AS total\_orders**

**from orders**

**GROUP BY customer\_id**

**HAVING COUNT(\*) > (SELECT COUNT(\*)**

**FROM orders**

**where customer\_id => 1**

**);**

**14.--List books that are more expensive than the average book price**

**SELECT title, price**

**FROM books**

**WHERE price > (**

**SELECT AVG(price)**

**FROM books**

**);**

**15--Show each customer and the number of orders they placed using a subquery in SELECT.**

**SELECT**

**CONCAT(first\_name, ' ', second\_name) AS full\_name,**

**(SELECT COUNT(\*)**

**FROM orders o**

**WHERE o.customer\_id = c.customer\_id) AS total\_orders**

**FROM customers c;**

**----JOINS**

**16- Show full name of each customer and the titles of books they ordered**

**SELECT c.first\_name || ' ' || c.second\_name AS full\_name,**

**b.title**

**FROM customers c**

**JOIN orders o ON c.customer\_id = o.customer\_id**

**JOIN books b ON o.book\_id = b.book\_id;**

**17-- List all orders including book title, quantity, and total cost (price × quantity)**

**SELECT**

**b.title,**

**o.quantity,**

**CAST(b.price AS NUMERIC) \* CAST(o.quantity AS INTEGER) AS total\_cost**

**FROM orders o**

**JOIN books b ON o.book\_id = b.book\_id;**

**18,Show customers who haven't placed any orders (LEFT JOIN)**

**SELECT**

**c.first\_name || ' ' || c.second\_name AS full\_name**

**FROM customers c**

**LEFT JOIN orders o ON c.customer\_id = o.customer\_id**

**WHERE o.order\_id IS NULL;**

**19.--List all books and the names of customers who ordered them, if any (LEFT JOIN)**

**SELECT**

**b.title AS book\_title,**

**c.first\_name || ' ' || c.second\_name AS customer\_name**

**FROM books b**

**LEFT JOIN orders o ON b.book\_id = o.book\_id**

**LEFT JOIN customers c ON o.customer\_id = c.customer\_id;**

**20.---Show customers who live in the same city (SELF JOIN)**

**SELECT**

**c1.first\_name || ' ' || c1.second\_name AS customer\_1,**

**c2.first\_name || ' ' || c2.second\_name AS customer\_2,**

**c1.city**

**FROM customers c1**

**JOIN customers c2 ON c1.city = c2.city AND c1.customer\_id <> c2.customer\_id;**

**Combined Logic**

**21. Show all customers who placed more than 2 orders for books priced over**

**SELECT b.price, b.title, COUNT(o.order\_id) AS total\_orders**

**FROM books b**

**JOIN orders o on o.book\_id = b.book\_id**

**WHERE b.price>2000**

**GROUP BY b.price, b.title**

**HAVING COUNT(o.order\_id)> 2;**

**22\*. List customers who ordered the same book more than once.**

**SELECT c.first\_name, COUNT(o.order\_id) AS total\_orders**

**FROM orders o**

**JOIN customers c ON c.customer\_id = o.customer\_id**

**GROUP BY c.first\_name**

**HAVING COUNT(o.order\_id) > 1;**

**23. Show each customer's full name, total quantity of books ordered, and total amount**

**spent.**

**SELECT**

**c.first\_name || ' ' || c.second\_name AS full\_name,**

**SUM(CAST(o.quantity AS INTEGER)) AS total\_quantity\_ordered,**

**SUM(CAST(o.quantity AS INTEGER) \* CAST(b.price AS NUMERIC)) AS total\_amount\_spent**

**FROM customers c**

**JOIN orders o ON c.customer\_id = o.customer\_id**

**JOIN books b ON o.book\_id = b.book\_id**

**GROUP BY c.customer\_id, c.first\_name, c.second\_name;**

**24.List books that have never been ordered.**

**SELECT b.book\_id, b.title**

**FROM books b**

**LEFT JOIN orders o ON b.book\_id = o.book\_id**

**WHERE o.book\_id IS NULL;**

**25.Find the customer who has spent the most in total (JOIN + GROUP BY + ORDER BY +**

**LIMIT)**

**SELECT**

**c.first\_name || ' ' || c.second\_name AS full\_name,**

**COUNT(o.order\_id) AS total\_orders**

**FROM customers c**

**JOIN orders o ON c.customer\_id = o.customer\_id**

**GROUP BY c.first\_name, c.second\_name**

**ORDER BY total\_orders DESC**

**LIMIT 1;**

**26.Write a query that shows, for each book, the number of different customers who have**

**ordered it.**

**SELECT**

**b.title,**

**COUNT(DISTINCT o.customer\_id) AS different\_customers**

**FROM books b**

**JOIN orders o ON b.book\_id = o.book\_id**

**GROUP BY b.title;**

**27.Using a subquery, list books whose total order quantity is above the average order**

**quantity**

**SELECT b.title,**

**SUM(CAST(o.quantity AS INTEGER)) AS total\_order\_quantity**

**FROM books b**

**JOIN orders o ON b.book\_id = o.book\_id**

**GROUP BY b.title**

**HAVING SUM(CAST(o.quantity AS INTEGER)) > (**

**SELECT AVG(total\_quantity)**

**FROM (**

**SELECT SUM(CAST(o.quantity AS INTEGER)) AS total\_quantity**

**FROM books b**

**JOIN orders o ON b.book\_id = o.book\_id**

**GROUP BY b.title**

**) AS subquery**

**);**

**28.. Show the top 3 customers with the highest number of orders and the total amount they**

**spent**

**SELECT**

**c.first\_name || ' ' || c.second\_name AS full\_name,**

**COUNT(o.order\_id) AS total\_orders,**

**SUM(CAST(o.quantity AS INTEGER) \* CAST(b.price AS NUMERIC)) AS total\_amount\_spent**

**FROM customers c**

**JOIN orders o ON c.customer\_id = o.customer\_id**

**JOIN books b ON o.book\_id = b.book\_id**

**GROUP BY c.customer\_id, c.first\_name, c.second\_name**

**ORDER BY total\_orders DESC**

**LIMIT 3;**