1. List all countries.

SELECT country FROM country;

2. Show the number of countries.

SELECT COUNT(country\_id) AS number\_of\_countries FROM country;

3. Find United States in the country table.

SELECT \* FROM country WHERE country = 'United States';

4. List all payments with an amount of either 1.99, 2.99, 3.99 or 4.99

 SELECT \* FROM payment

WHERE amount IN (1.99, 2.99, 3.99, 4.99);

Suppose the country table was created using the following statement:

CREATE TABLE country (  
    country\_id serial primary key,  
    country character varying(50) NOT NULL,  
    last\_update timestamp without time zone DEFAULT now() NOT NULL  
);

Check [https://www.postgresql.org/docs/10/static/datatype-numeric.htmlLinks to an external site.](https://www.postgresql.org/docs/10/static/datatype-numeric.html)) for more information on the data type "serial".

5. Insert a new record named utopia into the country table.

INSERT INTO country (country) VALUES ('Utopia');

6. Can this query be executed successfully: insert into country(country\_id, country) values (1, 'Utopia');

 It depends on the current state of the table. If there’s no record with country\_id = 1, then it will execute successfully. If a record already exists the query will fail because of the primary key constraint violation.

7. Order countries by id asc, then show the 12th to 17th rows.

SELECT \* FROM country

ORDER BY country\_id ASC

LIMIT 6 OFFSET 11;

8. List the first name of all customers. The list should NOT have any duplicates (meaning, if multiple customers have the same first name, it should appear in the result only once).

SELECT DISTINCT first\_name FROM customer;

9. List stores with more than 200 customers.

SELECT store\_id

FROM customer

GROUP BY store\_id

HAVING COUNT(customer\_id) > 200;

10. Find all duplicated first names in the customer table.

SELECT first\_name

FROM customer

GROUP BY first\_name

HAVING COUNT(customer\_id) > 1;

11. List all addresses in a city whose name starts with 'A'.

SELECT address

FROM address

JOIN city ON address.city\_id = city.city\_id

WHERE city.city LIKE 'A%';

12. Why this query doesn't work? select \* from address natural join city where city like 'A%'

The NATURAL JOIN operation joins two tables using all columns with the same name in both tables. It can lead to potential ambiguities if the tables being joined have multiple columns with the same name but for different purposes. Also a secondary issue might be the WHERE clasue, it's ambiguous doesn’t know if you're referring to the city table or the city column in that table.

Corrected Query- explicit JOIN on the columns and also specifies which city where is referring to:  
SELECT \* FROM address

JOIN city ON address.city\_id = city.city\_id

WHERE city.city LIKE 'A%';

13. Display the average amount paid by each customer, along with the customer's first and last name.

SELECT customer.first\_name, customer.last\_name, AVG(payment.amount) as avg\_amount

FROM customer

JOIN payment ON customer.customer\_id = payment.customer\_id

GROUP BY customer.customer\_id;

14. List all customers' first name, last name and the city they live in.

SELECT customer.first\_name, customer.last\_name, city.city

FROM customer

JOIN address ON customer.address\_id = address.address\_id

JOIN city ON address.city\_id = city.city\_id;

15. Assume there're n film categories. Let L = Min(L1, L2, .. Ln)  where Li = the length of the longest film in the ith category.

Please write a single SQL query that finds all films whose lengths are greater than L.

 SELECT \*

FROM film

WHERE length > (

SELECT MIN(max\_length)

FROM (

SELECT category\_id, MAX(length) AS max\_length

FROM film

JOIN film\_category ON film.film\_id = film\_category.film\_id

GROUP BY category\_id

) AS subquery

);

16. Find all customers with at least one payment whose amount is greater than 11 dollars.

SELECT DISTINCT customer\_id, first\_name, last\_name

FROM customer

JOIN payment ON customer.customer\_id = payment.customer\_id

WHERE payment.amount > 11;

17. Find all customers with at least three payments whose amount is greater than 9 dollars.

SELECT customer\_id, first\_name, last\_name

FROM customer

JOIN payment ON customer.customer\_id = payment.customer\_id

WHERE payment.amount > 9

GROUP BY customer.customer\_id

HAVING COUNT(payment.payment\_id) >= 3;