Step 1 - https://www.sql-practice.com/
Step 2 - Select Database hospital.db
Step 3 - Select Easy Level Questions

Ques 1. Show first name, last name, and gender of patients whose gender is 'M'

SELECT first_name, last_name, gender FROM patients WHERE gender == "M"

- *SELECT first_name, last_name, gender: This part of the query specifies the columns to be retrieved from the database. It asks for the first name, last name, and gender of some entries in a table. In this context, it means you're interested only in these three pieces of information for each entry that matches the criteria you're about to specify.*
- *FROM patients: This specifies the table from which you want to retrieve the data. The table in question is named patients, which implies that it likely contains information about individuals who have received medical care or consultation.*
- *WHERE gender == "M": This is the condition that entries must meet to be included in the results of the query. It filters the entries in the patients table to only include those whose gender column is equal to "M". The double equals sign (==) is a comparison operator that checks for equality. However, it's worth noting that in standard SQL, the equality operator is a single equals sign (=), so the correct syntax should be WHERE gender = "M".*

Ques 2. Show first name and last name of patients who do not have allergies. (null)

SELECT first_name, last_name FROM patients WHERE allergies IS NULL

- *This SQL query is designed to select and display the first names and last names of patients from a table named `patients`, specifically targeting those records where the `allergies` column has a null value, indicating that no allergies have been recorded for these patients. The use of `IS NULL` in the `WHERE` clause is key for filtering out any*

records where the `allergies` field is not populated, allowing the query to return only the desired subset of patients without allergies information.

Ques 3. Show first name of patients that start with the letter 'C'

SELECT first_name FROM patients WHERE first_name LIKE 'c%'

- *This SQL query selects the first names of patients from the `patients` table, filtering the results to only include those names that start with the letter 'C'. The `LIKE` operator is used in the `WHERE` clause to match a specific pattern, where ``'c%'`` signifies any string that starts with 'c', followed by zero or more characters. The `%` symbol acts as a wildcard for any sequence of characters. This approach is case-insensitive in many SQL databases, effectively capturing names that start with either uppercase 'C' or lowercase 'c'.*

Ques 4. Show first name and last name of patients that weight within the range of 100 to 120 (inclusive)

SELECT first_name, last_name FROM patients WHERE weight BETWEEN 100 AND 120

- *This SQL query selects the first names of patients from the `patients` table, filtering the results to only include those names that start with the letter 'C'. The `LIKE` operator is used in the `WHERE` clause to match a specific pattern, where ``'c%'`` signifies any string that starts with 'c', followed by zero or more characters. The `%` symbol acts as a wildcard for any sequence of characters. This approach is case-insensitive in many SQL databases, effectively capturing names that start with either uppercase 'C' or lowercase 'c'.*

Ques 5. Update the patients table for the allergies column. If the patient's allergies is null then replace it with 'NKA'

UPDATE patients SET allergies = "NKA" WHERE allergies IS NULL

- *This SQL statement updates the patients table, specifically targeting the allergies column. It sets the allergies column to "NKA" for all records where the allergies column currently has a null value. "NKA" typically stands for "No Known Allergies." The WHERE clause specifies that this update should only apply to records where the allergies column is null, ensuring that patients with existing allergy information remain unaffected. This operation is useful for maintaining cleaner, more informative database records by replacing null values with a meaningful default that indicates a patient has no known allergies.*

Ques 6. Show first name and last name concatinated into one column to show their full name.

SELECT CONCAT(first_name, " ", last_name) AS full_name from patients

- *This SQL query uses the CONCAT function to combine the first_name and last_name columns into a single column, creating a full name for each patient in the patients table. The inclusion of " " (a space) between the two names ensures that the first and last names will be properly separated in the resulting full name. The AS full_name part of the query assigns an alias to the resulting column, so in the output of this query, the combined names will appear under the column header full_name. This approach is useful for situations where displaying the full name as a single piece of information is more convenient or necessary for reporting, user interfaces, or data processing tasks.*

Ques 7. Show first name, last name, and the full province name of each patient.

SELECT first_name, last_name, province_names.province_name FROM patients INNER JOIN province_names ON patients.province_id = province_names.province_id

- *This SQL query effectively combines data from two tables to display a comprehensive list of patient names along with the full names of their associated provinces. The `INNER JOIN` clause is crucial here, as it links the `patients` table with the `province_names` table using the common key `province_id`. This operation ensures that only those records from both tables that have matching `province_id` values are included in the result set, allowing for the accurate pairing of patients with their respective province names based on the relational database's structured relationships.*

Ques 8. Show how many patients have a birth_date with 2010 as the birth year.

SELECT COUNT(patient_id) FROM patients WHERE birth_date LIKE '2010%'

- *This query counts the patients in the patients table who were born in the year 2010. The LIKE '2010%' condition filters the birth_date to those starting with "2010", effectively selecting dates within that year.*

Ques 9. Show the first_name, last_name, and height of the patient with the greatest height.

SELECT first_name, last_name, MAX(height) FROM patients

- *The MAX() function is an aggregate function that returns the maximum value of a column across all rows that match the query criteria. However, SQL standards generally require that any column in the SELECT list that is not an aggregate function (like MAX()) must be included in a GROUP BY clause, unless you are using specific database functionalities that allow for such a query without explicit grouping.*

Ques 10. Show all columns for patients who have one of the following patient_ids: 1,45,534,879,1000

SELECT * FROM patients WHERE patient_id IN (1,45,534,879,1000)

- *This SQL query retrieves all columns for patients within the patients table who have a patient_id matching any of the specified values: 1, 45, 534, 879, or 1000. The IN clause is used to filter the results to include only rows where the patient_id column matches one of the values listed in the parentheses. This approach is efficient for querying multiple specific values in a single operation, eliminating the need for multiple OR conditions.*

Ques 11. Show the total number of admissions

SELECT COUNT(patient_id) FROM admissions

- *This SQL query calculates the total number of admissions by counting the entries in the admissions table, using the patient_id column as the basis for the count. The COUNT(patient_id) function counts the number of rows where patient_id is not null, effectively giving the total number of admissions recorded in the table. If patient_id is a unique identifier for each admission, this will provide the total count of admissions. If patients can have multiple admissions, this count will include all instances of admissions, potentially counting some patients multiple times if they were admitted on multiple occasions.*

Ques 12. Show all the columns from admissions where the patient was admitted and discharged on the same day.

SELECT * from admissions WHERE (admission_date == discharge_date)

- *The concept highlighted in this query involves comparing two date columns within the same table to filter records based on a specific temporal relationship: the admission_date and discharge_date being identical. This approach is useful in scenarios where the duration of an event is relevant to the analysis or reporting requirements, such as assessing hospital efficiency, patient turnaround time, or the necessity for short-term medical interventions. The comparison underscores the ability of SQL to perform row-level operations that evaluate the relationship between different columns within the same dataset, providing a powerful tool for data analysis directly within the database through conditional logic.*

Ques 13. Show the total number of admissions for patient_id 579.

select patient_id, COUNT(*) from admissions where patient_id == 579

- *In the context of SQL, using COUNT(*) alongside a specific column in the SELECT clause without including a GROUP BY clause is a common point of confusion. COUNT(*) is an aggregate function that returns the number of rows matching the query criteria. When it's used with another column (patient_id in this case), without a GROUP BY clause, it can lead to misunderstandings or errors in some SQL dialects, because the query seems to be asking for both aggregated and non-aggregated data simultaneously.*
- *However, in the specific scenario where the WHERE clause restricts the results to a single patient_id, the inclusion of patient_id in the SELECT list alongside COUNT(*) without a GROUP BY clause might not raise an error in certain SQL environments because the query implicitly concerns a single group. Yet, this practice can be confusing and is generally not recommended without clear intent or understanding.*
- *The main takeaway here is the importance of distinguishing between aggregated and non-aggregated data in a SELECT statement and knowing when to use GROUP BY to ensure clarity and correctness in SQL queries.*

Ques 14. Based on the cities that our patients live in, show unique cities that are in province_id 'NS'?

SELECT DISTINCT city FROM patients WHERE province_id == "NS"

- *The concept of DISTINCT in SQL plays a crucial role when you need to eliminate duplicate entries in your result set. By using SELECT DISTINCT, the query ensures that each city name returned is unique, effectively removing any repetition. This is particularly useful in scenarios where you want to analyze or list distinct categories, locations, or other non-repeated values within a dataset. In this case, it's applied to city names within a specified province, highlighting its utility in geographical or demographic data analysis to provide a concise list of locations.*

Ques 15. Write a query to find the first_name, last name and birth date of patients who have height more than 160 and weight more than 70

SELECT first_name, last_name, birth_date FROM patients WHERE (height > 160) AND (weight > 70)

- *The query combines conditions using the AND logical operator within the WHERE clause. This operator ensures that for a record to be included in the result set, it must satisfy all the conditions listed—specifically, a height greater than 160 and a weight greater than 70. This use of AND is fundamental in SQL for defining queries that require multiple criteria to be met simultaneously, allowing for precise filtering based on several attributes.*

Ques 16. Write a query to find list of patients first_name, last_name, and allergies from Hamilton where allergies are not null

SELECT first_name, last_name, allergies FROM patients WHERE (city == "Hamilton") and allergies IS NOT NULL

- *The query introduces the IS NOT NULL condition, which is used to filter out rows where a specified column (in this case, allergies) does not contain a null value. This is essential for scenarios where you're interested in records that have specific, non-empty data in a certain field, ensuring that the query results exclude any entries lacking information in the allergies column.*

Ques 17. Based on cities where our patient lives in, write a query to display the list of unique city starting with a vowel (a, e, i, o, u).

SELECT DISTINCT(city) FROM patients WHERE (city LIKE 'a%' OR city LIKE 'e%' OR city LIKE 'i%' OR city LIKE 'o%' OR city LIKE 'u%') ORDER by city ASC

- *The query uses the LIKE operator with wildcard character % for pattern matching, combined with the OR logical operator to select cities starting with any vowel. This pattern matching technique is pivotal when querying data based on specific text criteria, allowing for flexible and inclusive data retrieval. Additionally, the ORDER BY clause with ASC ensures the results are sorted in ascending alphabetical order, facilitating an organized presentation of the unique city names that meet the specified condition. This ordered and distinct listing is particularly useful for data analysis and reporting purposes, ensuring clarity and ease of reading.*