**Nama Lengkap : Krisna Heru Saputra**

**Latihan Pertemuan : Ke-1**

1. **Front-End Developer** adalah pengembang perangkat lunak yang fokus pada pembuatan dan pengelolaan bagian depan dari sebuah aplikasi web atau situs web.

**Back-End Developer** adalah pengembang perangkat lunak yang fokus pada pengembangan bagian belakang (mengelola dan mengatur data, logika, dan fungsi) dari sebuah aplikasi web atau sistem.

**DevOps Developer** adalah seseorang yang berfokus pada praktik dan metodologi penggabungan antara pengembangan (development) dan operasi (operations) dalam proses pengembangan perangkat lunak atau aplikasi.

**Full-Stack Developer** adalah pengembangan perangkat lunak yang memiliki kemampuan untuk menguasai seluruh bagian dari teknologi dan tumpukan (stack) yang digunakan dalam pengembangan aplikasi atau situs web.

1. Alur Algoritma Serching Algorithms
2. Mulai algoritma.
3. Tentukan data yang ingin dicari (misalnya, kita ingin mencari nilai 10 dalam array).
4. Buat sebuah array (misalnya, data) yang berisi beberapa angka.
5. Inisialisasi variabel cariData dengan nilai yang ingin dicari (misalnya, 10).
6. Mulai perulangan dari index pertama (index 0) hingga index terakhir (index data.length - 1) dari array data.
7. Pada setiap iterasi perulangan, periksa apakah elemen saat ini sama dengan nilai yang dicari (cariData).
8. Jika elemen saat ini sama dengan nilai yang dicari (cariData), maka kembalikan index elemen tersebut dan selesai.
9. Jika tidak ditemukan nilai yang cocok pada setiap iterasi perulangan, kembalikan nilai -1 untuk menandakan bahwa nilai tidak ditemukan.
10. Selesai.

Coding :

function linearSearch(arr, cari) {

for (let i = 0; i < arr.length; i++) {

if (arr[i] === cari) {

return i; // Mengembalikan index jika elemen ditemukan

}

}

return -1; // Mengembalikan -1 jika elemen tidak ditemukan

}

// Contoh penggunaan

const array = [10, 25, 5, 45, 30, 15];

const cariData = 30;

const result = linearSearch(array, cariData);

if (result !== -1) {

console.log(`Elemen ${cariData} ditemukan pada index ${result}.`);

} else {

console.log(`Elemen ${cariData} tidak ditemukan dalam array.`);

}

**Nama Lengkap : Krisna Heru Saputra**

**Latihan Pertemuan : Ke-2**

LINK REPOSITORY GITHUB : <https://github.com/krisna12217016/kelas>

**Nama Lengkap : Krisna Heru Saputra**

**Latihan Pertemuan : Ke-3**

Soal :

1. Show first name, last name, and gender of patients whose gender is 'M'
2. Show unique birth years from patients and order them by ascending.
3. Show unique first names from the patients table which only occurs once in the list.

For example, if two or more people are named 'John' in the first\_name column then don't include their name in the output list. If only 1 person is named 'Leo' then include them in the output.

1. Show patient\_id and first\_name from patients where their first\_name start and ends with 's' and is at least 6 characters long.
2. Display every patient's first\_name.

Order the list by the length of each name and then by alphabetically

1. Show the city and the total number of patients in the city.

Order from most to least patients and then by city name ascending.

1. Show the provinces that has more patients identified as 'M' than 'F'. Must only show full province\_name
2. Show patient\_id, weight, height, isObese from the patients table.

Display isObese as a boolean 0 or 1.

Obese is defined as weight(kg)/(height(m)2) >= 30.

weight is in units kg.

height is in units cm.

1. Sort the province names in ascending order in such a way that the province 'Ontario' is always on top.
2. For each day display the total amount of admissions on that day. Display the amount changed from the previous date.

Jawab :

1. SELECT first\_name, last\_name, gender

FROM patients

WHERE gender = 'M';

1. SELECT DISTINCT YEAR(birth\_date) AS birth\_year

FROM patients

ORDER BY birth\_year ASC;

1. SELECT first\_name

FROM patients

GROUP BY first\_name

HAVING COUNT(\*) = 1;

1. SELECT patient\_id, first\_name

FROM patients

WHERE first\_name LIKE 's%s' AND LENGTH(first\_name) >= 6;

1. SELECT first\_name

FROM patients

ORDER BY LENGTH(first\_name), first\_name;

1. SELECT city, COUNT(\*) as total\_patients

FROM patients

GROUP BY city

ORDER BY total\_patients DESC, city;

1. SELECT province\_names.province\_name

FROM province\_names

JOIN patients ON province\_names.province\_id = patients.province\_id

GROUP BY province\_names.province\_name

HAVING SUM(CASE WHEN patients.gender = 'M' THEN 1 ELSE 0 END) > SUM(CASE WHEN patients.gender = 'F' THEN 1 ELSE 0 END);

1. SELECT

patient\_id,

weight,

height,

CASE WHEN (weight / POWER(height / 100.0, 2)) >= 30 THEN 1 ELSE 0 END AS isObese

FROM patients;

1. SELECT

province\_name

FROM

province\_names

ORDER BY

CASE WHEN province\_name = 'Ontario' THEN 0 ELSE 1 END,

province\_name ASC;

1. SELECT

admission\_date,

admissions\_count,

CASE

WHEN LAG(admissions\_count) OVER (ORDER BY admission\_date) IS NULL THEN NULL

ELSE admissions\_count - LAG(admissions\_count) OVER (ORDER BY admission\_date)

END AS admissions\_diff

FROM

(

SELECT

admission\_date,

COUNT(patient\_id) AS admissions\_count

FROM

admissions

GROUP BY

admission\_date

) AS temp

ORDER BY

admission\_date;