# Question 1

### Problem: [https://leetkode.com/problems/intersection-of-two-arrays/](https://leetcode.com/problems/intersection-of-two-arrays/)

### Your Own Solution:

class Solution:

def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:

def inter(nums1,nums2):

inter = []

for i in nums1:

if i in nums2:

if i not in inter:

inter.append(i)

return inter

if len(nums1) <= len(nums2):

unique = inter(nums1,nums2)

else:

unique = inter(nums2,nums1)

return unique

### Recommended Solution: [https://leetkode.com/problems/intersection-of-two-arrays/solutions/500206/python3-dictionary-with-comments-faster-than-90-88-less-than-100/](https://leetcode.com/problems/intersection-of-two-arrays/solutions/500206/python3-dictionary-with-comments-faster-than-90-88-less-than-100/)

### Write in your own words, in Bahasa Indonesia, how does the recommended solution work?

**dictionary = {}**

kode di atas adalah dictionary kosong yang mana akan diisi.

**for index in nums1:**

**if index not in dictionary:**

**dictionary[index] = 1**

kode ini melakukan perulangan terhadap isi yang ada pada nums1. Perulangan ini bertujuan untuk memeriksa apakah ada angka yang sama pada key dalam dictionary yang diinisialisasi diawal. Jika index tidak ada pada dictionary maka dictionary diawal akan diisi, dengan keynya adalah index dan valuenya adalah 1.

**for index in nums2:**

**if index in dictionary and dictionary[index] != 2:**

**dictionary[index] += 1**

kode ini melakukan perulangan melalui setiap elemen dalam list nums2, dan jika elemen tersebut juga ada dalam dictionary dan nilainya bukan 2, maka nilai tersebut akan ditambahkan 1. Ini adalah cara sederhana untuk menghitung berapa kali masing-masing elemen dalam nums2 muncul dalam kamus dictionary, kecuali jika nilainya sudah 2, karena elemen dengan nilai 2 tidak akan diubah dalam dictionary.

**newList = []**

**for key, value in dictionary.items():**

**if value == 2:**

**newList.append(key)**

kode ini melakukan perulangan melalui semua pasangan key dan values dalam dictionary, dan jika nilai value adalah 2, maka key tersebut akan ditambahkan ke dalam list newList. Sebagai hasilnya, newList akan berisi semua kunci yang memiliki nilai 2 dalam dictionary.

# Question 2

### Problem: [https://leetkode.com/problems/product-sales-analysis-iii/description/?envType=study-plan-v2&envId=top-sql-50](https://leetcode.com/problems/product-sales-analysis-iii/description/?envType=study-plan-v2&envId=top-sql-50)

### Your Own Solution:

# Write your MySQL query statement below

select

product\_id,

year as first\_year,

quantity,

price

from Sales

where (product\_id, year)in(

select product\_id, min(year) as year

from Sales

group by 1

)

### Recommended Solution: [https://leetkode.com/problems/product-sales-analysis-iii/solutions/984255/mysql-simple-solution-beats-90-w-o-window-functions/?envType=study-plan-v2&envId=top-sql-50](https://leetcode.com/problems/product-sales-analysis-iii/solutions/984255/mysql-simple-solution-beats-90-w-o-window-functions/?envType=study-plan-v2&envId=top-sql-50)

### Write in your own words, in Bahasa Indonesia, how does the recommended solution work?

**select**

**product\_id,**

**year as first\_year,**

**quantity,**

**price**

**from Sales**

Memilih 4 kolom dari tabel Sales "product\_id", "year" yang dinamai ulang menjadi "first\_year", "quantity", dan "price".

**where (product\_id, year) in (select product\_id, min(year) from Sales group by 1)**

Kondisi untuk memfilter data menggunakan subquery.

**select product\_id, min(year) from Sales group by 1 ->** digunakan untuk mengambil pasangan "product\_id" dan tahun ("year") di mana tahun merupakan tahun minimum (terkecil) untuk setiap produk. Dalam subquery ini, data dikelompokkan berdasarkan "product\_id" dan kemudian mengambil tahun minimum untuk setiap kelompok menggunakan fungsi agregat min(year)**.**

**(product\_id, year) in ->** akan membandingkan setiap baris dalam tabel "Sales" dengan pasangan "product\_id" dan tahun yang dihasilkan dari subquery. Ini akan menghasilkan baris-baris di mana pasangan ini ada dalam hasil subquery, yang berarti hanya akan mendapatkan data untuk produk pada tahun pertama penjualan (tahun minimum) untuk setiap produk.

# Question 3

### Problem: [https://leetkode.com/problems/power-of-three/description/](https://leetcode.com/problems/power-of-three/description/)

class Solution:

def isPowerOfThree(self, n: int) -> bool:

if n==1:

return True

elif n <= 0 or n%3 != 0:

return False

else:

check = True

while check:

n=n/3

if n==1:

return True

check = False

elif n%3 != 0:

return False

check = False

### Answer the question:

**Why does the kode below return False?**

from math import log

log(243,3) == 5

**Your Answer:**

Kode log(243, 3) tidak mengembalikan nilai yang sama persis dengan 5 karena masalah presisi floating point pada komputer. Ketika menggunakan math.log dalam Python, hasil yang dikembalikan adalah dalam format floating point (bilangan pecahan), dan hasilnya mungkin tidak selalu akurat hingga digit terakhir.

Dalam kasus ini, hasil sebenarnya dari log(243, 3) adalah hampir 5, tepatnya 4.999999999999999. Hal ini terjadi karena komputer menggunakan representasi floating point dalam sistem biner dan kadang-kadang tidak dapat merepresentasikan bilangan desimal dengan presisi yang sempurna. Dalam hal ini, hasil yang mendekati 5.

Presisi floating point pada komputer merujuk pada sejauh mana komputer dapat merepresentasikan dan memanipulasi bilangan pecahan dengan akurasi tertentu dalam sistem floating point. Presisi ini mengacu pada jumlah digit (bit) yang digunakan untuk merepresentasikan bilangan pecahan dalam format floating point.

### Recommended Solution: [https://leetkode.com/problems/power-of-three/solutions/77977/math-1-liner-no-log-with-explanation/](https://leetcode.com/problems/power-of-three/solutions/77977/math-1-liner-no-log-with-explanation/)

### Write in your own words, in Bahasa Indonesia, how does the recommended solution work?

def isPowerOfThree(self, n):

return n > 0 == 3\*\*19 % n

Kode ini mengembalikan True jika n adalah bilangan positif dan memiliki sisa 0 ketika dibagi dengan 3 pangkat 19. Angka 3^19 dipilih dalam kode ini karena itu adalah pangkat tertinggi dari tiga yang dapat diwakili dalam variabel 32-bit yang umum digunakan dalam komputasi. Range variabel 32 bit yaitu **-2.147.483.648** hingga **2.147.483.647.** Nilai 3^19 adalah **1.162.261.467** dan 3^20 adalah **3.486.784.401** sehingga dipilihlah 3^19 karena nilai 3^19 batas atas dari perpangkatan 3 yang bisa digunakan pada variabel 32-bit dan 3^20 telah melewati range dari variabel 32bit.

Namun ada sedikit catatan, menurut saya kode ini agak susah dipahami karena membandingkan 2 buah kondisi dengan tipe data yang dihasilkan berbeda untuk setiap kondisinya. n>0 jelas ini akan menghasilkan nilai boolean (True/False) sedangkan 3\*\*9%n sebenarnya tidak menghasilkan nilai True atau false melainkan nilai numerik, dan membandingkan nilai boolean dengan numerik menimbulkan ketidakjelasan. Menurut saya lebih baik menuliskannya seperti ini:  
return n>0 and 3\*\*19%n == 0.