HOMEWORK 2

Task 1,1

SINGLE COLUMN INDEX :
Single index is a lookup tables created based on one table column as the key that allows to speed up data retrieval

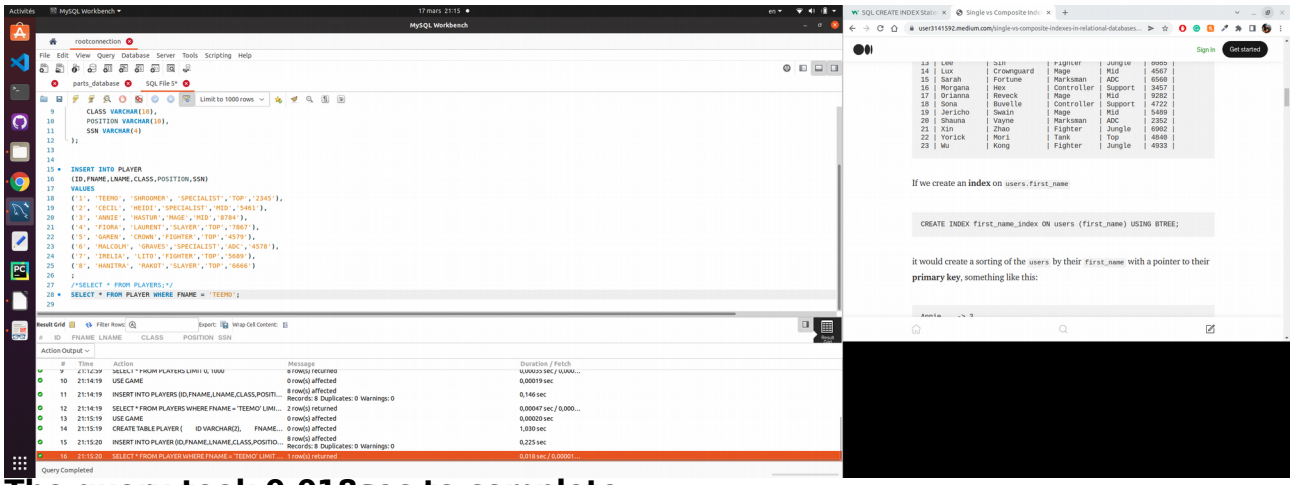
Consider this example: we are dooing a review of the beta players applications of our new adventure MMORPG game. 6000 beta players and we are adding records to the table as they come in.

Result: at this stage we have an unsorted gamers list that simply stores the first names, last names, class,position etc.

Task: we need to find a participant called Teemo which could require l*ooking at every single entry in the list, It will take time to find the player,*

*SELECT * FROM PLAYER WHERE FNAME = 'TEEMO';*

A query like the one above reads all first,name records (aka a full table scan) to find all matches which is a slow process,

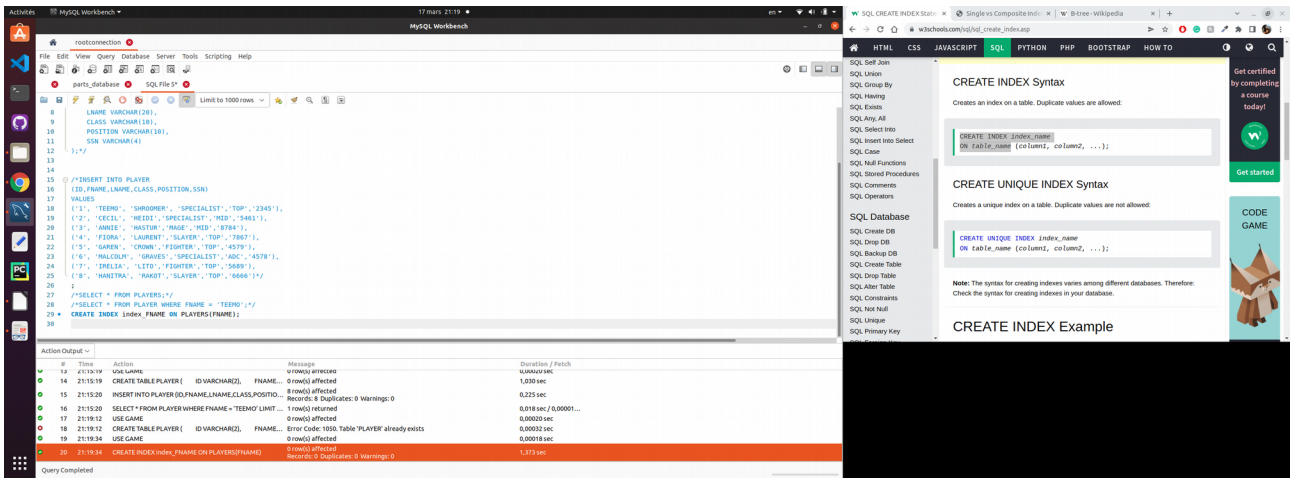


The query took 0,018sec to complete

WITH INDEX

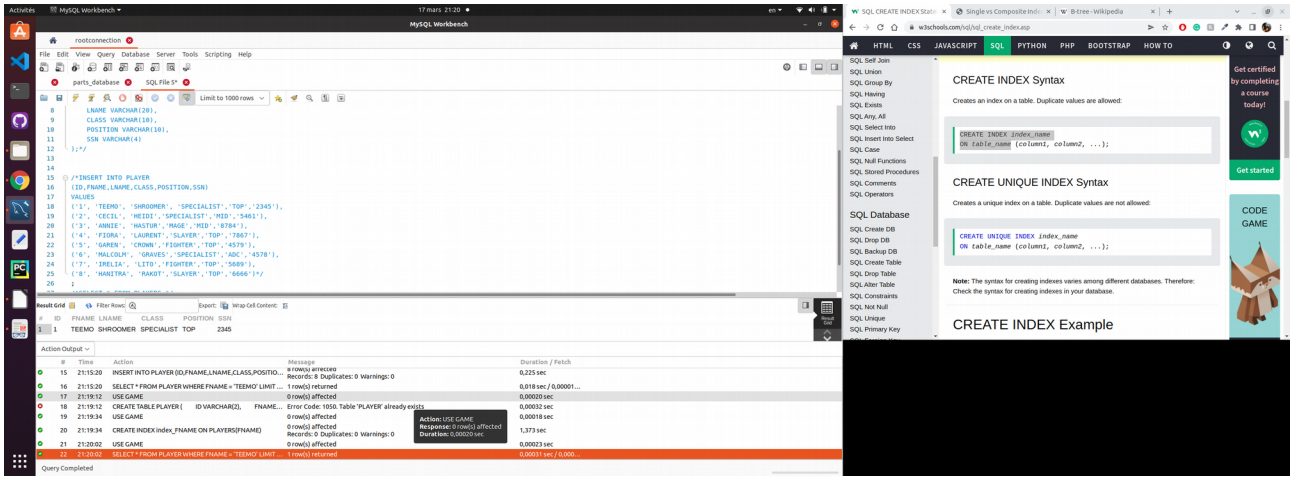
Adding a single column index would store underlying data in some logical order for example in our case for firstname

CREATE INDEX index_FNAME on PLAYER(FNAME) ;



And after creating the index, you can send the same query

*SELECT * FROM applicants WHERE first,name = 'Teemo';*



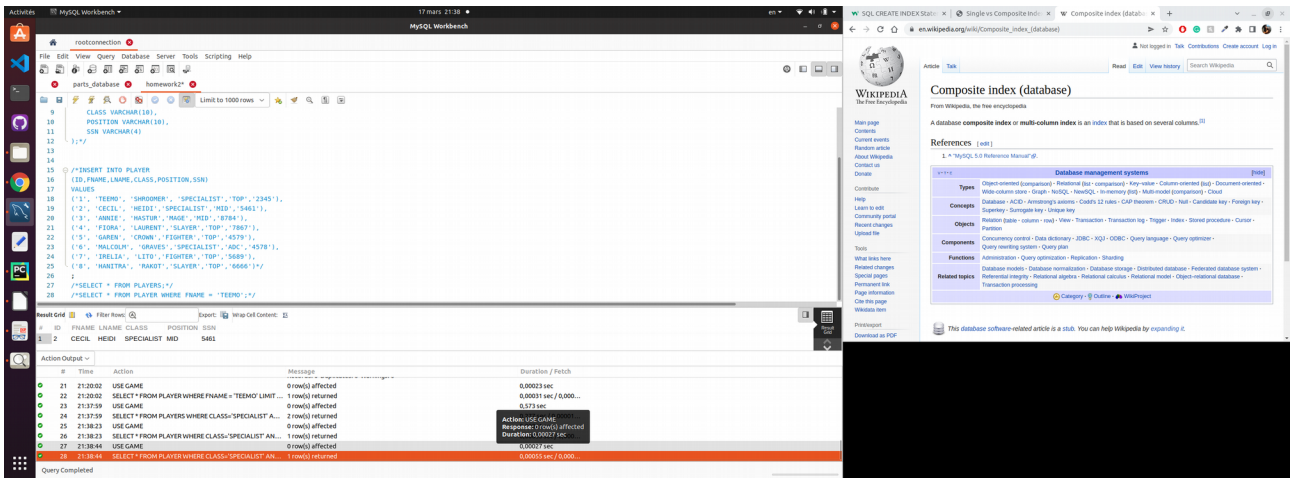
The query took 0,00031sec to complete this time.

COMPOSITES/ MULTI COLUMN INDEX :

Composites and multi column indexes is a lookup tables created based on multiples table columns that allows to speed up data retrieval

Consider the same example than before :there is probably several person who plays as 'SPECIALIST' class so let's search for people playing SPECIALIST class and TOP position

*SELECT * FROM PLAYERS WHERE class = 'SPECIALIST' and position='TOP';*



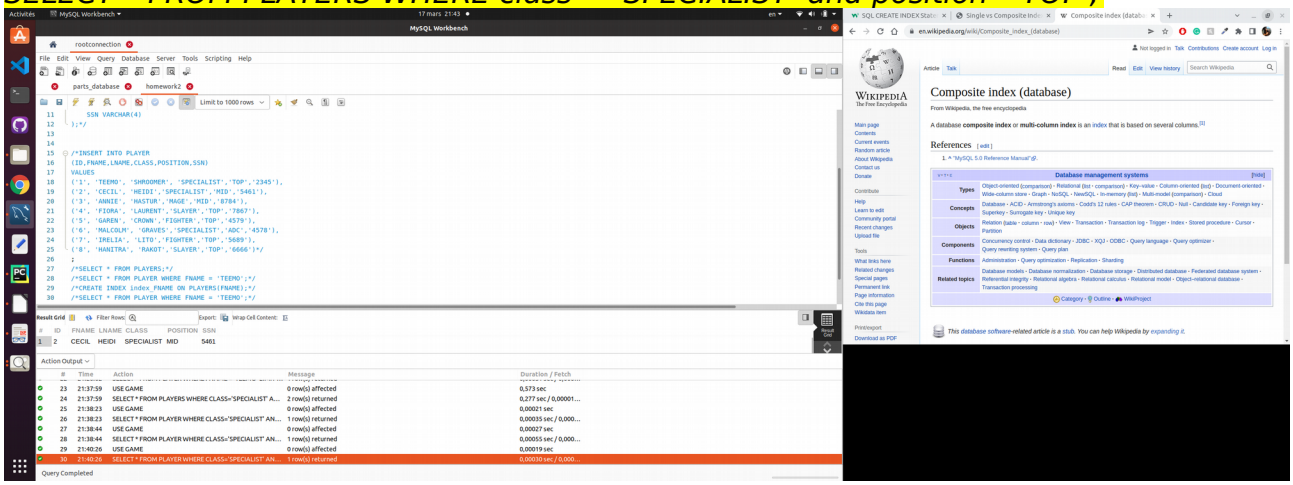
The query took 0,00055sec to complete

```
CREATE INDEX class_pos_index ON PLAYER (SPECIALIST, POSITION);
```

```
SELECT * FROM PLAYER
WHERE
CLASS= 'SPECIALIST'
AND
POSITION = 'TOP';
```

After creating the index, you can send the query below again

*SELECT * FROM PLAYERS WHERE class = 'SPECIALIST' and position='TOP';*



The query took 0,00030sec to complete

<u>Cluster INDEX :</u>

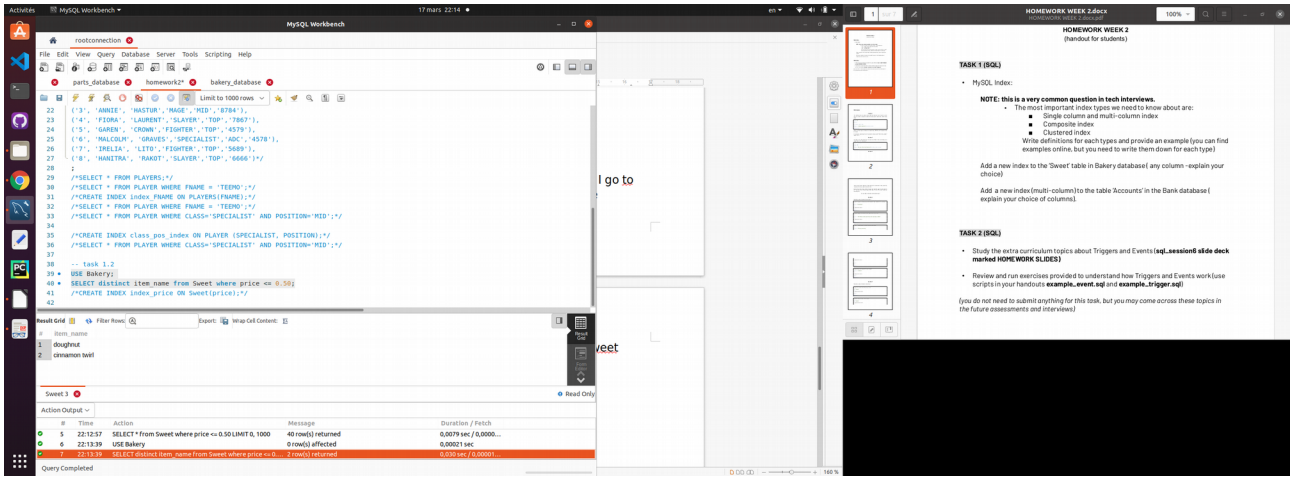
A cluster index is the table itself, which enforces the order of the rows in the table.

Task 1,2

I decided to have an single index on the price of Sweet because when I go to the Bakery , I am usually looking for a cheap snack before going home therefore, I am not searching for a particular type of Sweet but at a Sweet which will cost me less than 0,50£ for example,

USE Bakery;

SELECT distinct item_name from Sweet where price <= 0.50;



Without the index, the query took 0,030sec,

CREATE INDEX index_price ON Sweet(price);

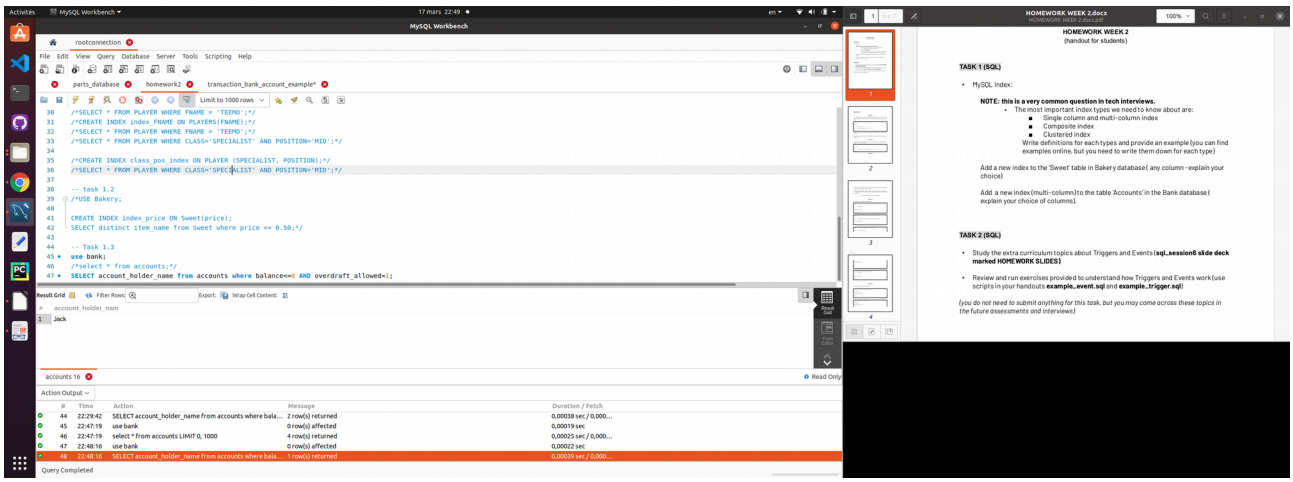
SELECT distinct item_name from Sweet where price <= 0.50;

With the index, the query took 0,00045sec

Task 1.3

As a banker, i would like to keep an eye on my client who have their balance under 0 to check if they have a overdraft allowed or not,

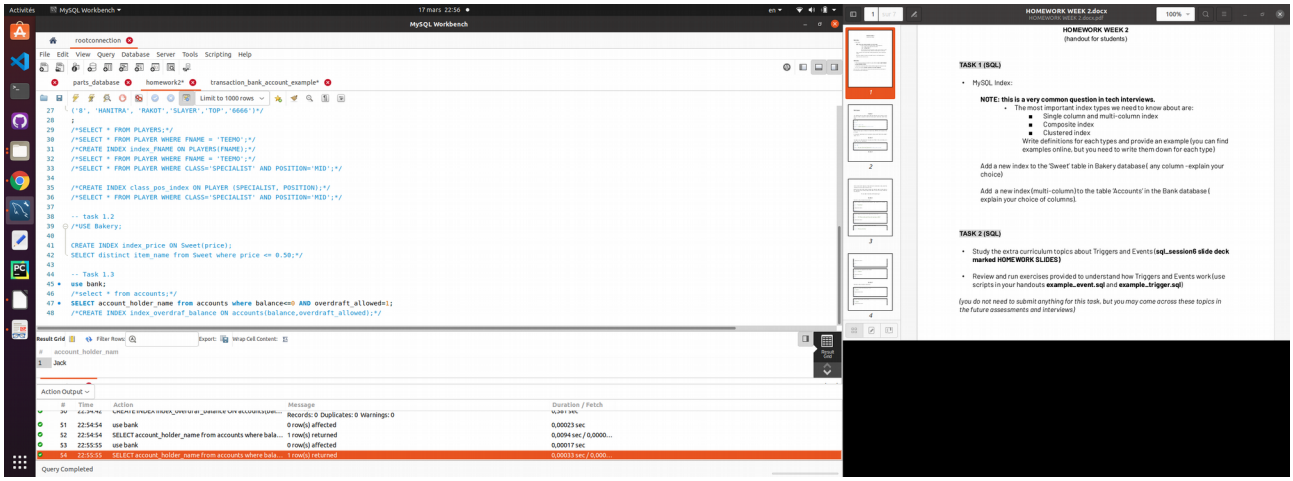
SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1;



The query took 0,00039sec to complete.

With index CREATE INDEX index_overdraf_balance ON accounts(balance,overdraft_allowed); and then sending the same query again

SELECT account_holder_name from accounts where balance<=0 AND overdraft_allowed=1;



The query took 0,00033sec to complete.

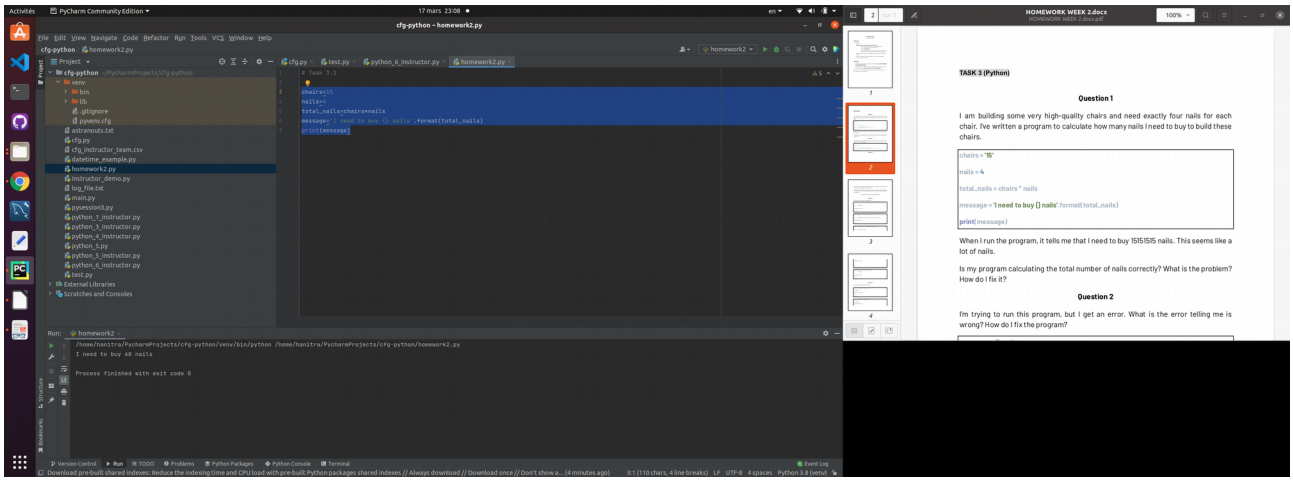
Creating indexes are sometimes storage consuming and have some effeciency improvement.

Task 3Q1

The problem is 15 which is not consider as a number but as a word

The fix is to remove the ' before and after 15 as below

```
chairs=15
nails=4
total_nails=chairs*nails
message='I need to buy {} nails'.format(total_nails)
print(message)
```



Task 3.Q2

The problem is Penelope which should be between '' as Penelope is a word. If Penelope is not between '' , the program considers as a variable which we didn't define. See the fixed code below

```
my_name='Penelope'
my_age=29
message='My name is {} and I am {} years old'.format(my_name,my_age)
print(message)
```

Task 3.Q3

```
box=input("how many boxes of eggs do you have ? ")
eggs=6
box_eggs = float(box) * float(eggs)
omelettes= float(box_eggs)/4
message='you can make {} omelettes with {} boxes of eggs'.format(omelettes,box)
print(message)
```

Task 3.Q4.1

```
my_str="I love coding."
my_str=my_str.replace(".","!")
print(my_str)
```

Task 3.Q4.2

```
my_str_1="EVERY Exercise Brings Me CLoser to Completing my GOALS"
```

```
my_str_1=my_str_1.lower()
print(my_str_1)
```

Task 3.Q4.3

```
my_str_2="We enjoy travelling"
ans_1=my_str_2.startswith('A')
print(ans_1)
```

Task 3.Q4.4

```
my_str_3="1.458.001"
ans_2=len(my_str_3)
print(ans_2)
```

The lenght of the string is 9

Task 3.Q5.1

```
wrd="Python"
ans_1=wrd[2:6]
print(ans_1)
```

Task 3.Q5.2

```
wrd="Python"
ans_1=wrd[0:4]
print(ans_1)
```

Task 3.Q5.3

```
wrd="Python"
ans_1=wrd[2:4]
print(ans_1)
```

Task 3.Q5.4

```
wrd="Python"
ans_1=wrd[2:5:2]
print(ans_1)
```

Task 3.6.

it is taking each number between 0 to 100 and multiply to ‘o’. As you can’t multiply a string with a number without type/format conversion, it wrote 100 times ‘o’

Task 3.Q7

The function doesn’t return a value,

the fix is as below

```
def calculate_vat(amount):
    q=amount * 1.2
    return q
#outside function
total=calculate_vat(100)
print(total)
```

Task 3 .Q8

```
item_1_name=input("what is the first article ? ")
item_1_price = input("what is the price of first article ? ")
item_2_name=input("what is the first article ? ")
item_2_price = input("what is the price of second article ? ")
item_3_name=input("what is the third article ? ")
item_3_price = input("what is the price of third article ? ")


total = float(item_1_price)+float(item_2_price)+float(item_3_price)
message1 ='{}.....{}'.format(item_1_name,item_1_price)
message2 ='{}.....{}'.format(item_2_name,item_2_price)
message3 ='{}.....{}'.format(item_3_name,item_3_price)
message4 ='Total.....{}'.format(total)
print(message1)
print(message2)
print(message3)
print(message4)
```