HOMEWORK WEEK 2

TASK 1 (SQL)

• MySQL Index:

NOTE: this is a very common question in tech interviews.

• The most important index types we need to know about are:

■ Single column and multi-column index

■ Composite index

■ Clustered index

Write definitions for each type and provide an example (you can find examples online, but you need to write them down for each type)

- Single column: a single column index is an index based on values of one column of a table

```
use bakery;

CREATE INDEX alphabetical_sweets
ON sweet (item_name);

select * from sweet where item_name like 'c%';
```

Result Grid | Filter Rows: | Export: | Wrap Cel

id	item_name	price
5	cannoli	0.88
456	Caramel slice	0.9
11	cherry_cake	5
4	cinnamon twirl	0.45
2	croissant	0.75

There can be single indexes applied to multiple columns, and it can work efficiently if each column has many distinct values that are not repeated many times in a database, and if the columns are used often separately in queries.

```
use bakery;

CREATE INDEX alphabetical_sweets
ON sweet(item_name);

CREATE INDEX price_index
ON sweet(price);

select * from sweet where item_name LIKE "c%";
select * from sweet where price > 0.5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	item_name	price
5	cannoli	0.88
456	Caramel slice	0.9
11	cherry_cake	5
4	cinnamon twirl	0.45
2	croissant	0.75

id	item_name	price
2	croissant	0.75
3	painauchocolat	0.55
5	cannoli	0.88
6	apple tart	1.12
11	cherry_cake	5
123	Apple_pie	1.2
456	Caramel slice	0.9
789	Yum yum	0.65

(Curiously, I think in the second instance my price index has not been used, potentially because of https://prologika.com/why-is-sql-server-not-using-my-index/ ! It did use it when I added second condition: select * from sweet where price > 0.5 and price < 1; Very cool!).

```
use bakery;

select * from sweet where price > 0.5 and price < 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

id	item_name	price
3	painauchocolat	0.55
789	Yum yum	0.65
2	croissant	0.75
5	cannoli	0.88
456	Caramel slice	0.9

- Multi-column (composite): is based on values of multiple columns.

```
use bank;

select * from accounts;

CREATE INDEX name_idx on accounts(account_holder_name,account_holder_surname);

select * from accounts where account_holder_name = 'Julie' AND account_holder_surname = 'Smith';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

account_number	account_holder_name	account_holder_surname	balance	overdraft_allowed
111112	Julie	Smith	150	1

- Clustered index: It's an index that reorganises data in a way that is easy to use. There can be only one clustered index per table. In MySQL primary key will be used as clustered index.

```
use bank;

ALTER TABLE accounts
ADD PRIMARY KEY (account_number);

select * from accounts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

account_number	account_holder_name	account_holder_surname	balance	overdraft_allowed
111112	Julie	Smith	150	1
111113	James	Andrews	20	0
111114	Jack	Oakes	-70	1
111115	Jasper	Wolf	200	1
NULL	NULL	NULL	NULL	NULL

- After adding clustered index (primary key in this example), each new account added will be sorted using account_number:

```
insert into accounts(account_number, account_holder_name, account_holder_surname, balance)
values ('111111', 'Lily', 'Marks', '400');

select * from accounts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

account_number	account_holder_name	account_holder_surname	balance	overdraft_allowed
111111	Lily	Marks	400	NULL
111112	Julie	Smith	150	1
111113	James	Andrews	20	0
111114	Jack	Oakes	-70	1
111115	Jasper	Wolf	200	1
NULL	NULL	NULL	NULL	NULL

Add a new index to the 'Sweet' table in Bakery database (any column -explain your choice)

Answer:

Bakery wants to create a marketing event called "Alphabetical sweets" where through each day they present their goods in alphabetical order (each day is a different letter) and they want to quickly check what they have in stock and what order should the sweets be presented in, without having to sort the items after fetching the data.

```
use bakery;

CREATE INDEX alphabetical_sweets
ON sweet (item_name);

select * from sweet where item_name like 'c%';
```

Result Grid | Filter Rows: | Export: | Wrap Cel

id	item_name	price
5	cannoli	0.88
456	Caramel slice	0.9
11	cherry_cake	5
4	cinnamon twirl	0.45
2	croissant	0.75

Add a new index (multi-column) to the table 'Accounts' in the Bank database (explain your choice of columns).

Answer:

```
use bank;

select * from accounts;

CREATE INDEX name_idx on accounts(account_holder_name,account_holder_surname);

select * from accounts where account_holder_name = 'Julie' AND account_holder_surname = 'Smith';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

account_number	account_holder_name	account_holder_surname	balance	overdraft_allowed
111112	Julie	Smith	150	1

Adding index on first name and last name creates a lookup that will allow us to perform various queries about bank customers. We now don't have to go through the whole database when searching for the specific name/surname combo. It will speed up the process significantly for large quantities of data.

TASK 3 (Python)

Question 1

I am building some very high-quality chairs and need exactly four nails for each chair. I've written a program to calculate how many nails I need to buy to build these chairs.

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

When I run the program, it tells me that I need to buy 15151515 nails. This seems like a lot of nails.

Is my program calculating the total number of nails correctly? What is the problem? How do I fix it? Write your explanation, along with the code to fix this, below.

Answer:

Program is running correctly and does what has been asked of it – it repeats the string '15' four times. To be able to change it so that it calculates the number of needed nails, we have some options:

<u>Use integer for chairs, i.e.:</u>

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

```
I need to buy 60 nails
```

<u>If the number of chairs is for example a string from an input, we can swap it later in the code:</u>

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

```
I need to buy 60 nails
```

Question 2

I'm trying to run this program, but I get an error. What is the error telling me is wrong? How do I fix the program?

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

Answer:

In this example we tried to use the variable name that is not valid. We have to make Penelope a string, and we can do it by:

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

```
My name is Penelope and I am 29 years old
```

Question 3

I have a lot of boxes of eggs in my fridge and I want to calculate how many omelettes I can make. Write a program to calculate this.
Assume that a box of eggs contains six eggs and I need four eggs for each omelette, but I should be able to easily change these values if I want. The output should say something like:
"*You can make 9 omelettes with 6 boxes of eggs*"

```
amount_of_boxes = 6
eggs_in_a_box_of_eggs = 6
eggs_needed_for_omlette = 4
making_omlette = (amount_of_boxes*eggs_in_a_box_of_eggs)//eggs_needed_for_omlette
# Use conditions to format text correcly
if making_omlette == 1 and amount_of_boxes == 1:
    print(f'You can make {making_omlette} omlette with {amount_of_boxes} box of eggs.')
elif making_omlette > 1 and amount_of_boxes == 1:
    print(f'You can make {making_omlette} omlettes with {amount_of_boxes} box of eggs.')
elif making_omlette == 1 and amount_of_boxes > 1:
    print(f'You can make {making_omlette} omlette with {amount_of_boxes} boxes of eggs.')
elif making_omlette > 1 and amount_of_boxes > 1:
    print(f'You can make {making_omlette} omlettes with {amount_of_boxes} boxes of eggs.')
elif making_omlette < 1:
    print('No omlette for you today!')
```

```
You can make 9 omlettes with 6 boxes of eggs.
```

Question 4

Complete a series of tasks to format strings

```
# Task 1 - Replace the (.) character with (!) instead. Output should be "I love coding!"
my_str = "I love coding."
# Type your code here.
```

```
print(my_str)
```

Answer:

```
# Task 1 - Replace the (.) character with (!) instead. Output should be “I love coding!”
my_str = "I love coding."
# Type your code here.
my_str = my_str.replace('.','!')
print(my_str)
```

```
I love coding!
```

```
# Task 2 - Reassign str so that all of its characters are lowercase.
my_str_1 = "EVERY Exercise Brings Me Closer to Completing my GOALS."
# Type your code here.
my_str_1 = my_str_1.lower()
print(my_str_1)
```

```
every exercise brings me closer to completing my goals.
```

```
# Task 3 - Output whether this string starts with an A or not
my_str_2 = "We enjoy travelling"
# Type your code here.
ans_1 = my_str_2.startswith('A')
print(ans_1)
```

```
False
```

```
# Task4 - What is the length of the given string?
my_str_3 = "1.458.001"
# Type your code here:
ans_2 = len(my_str_3)
print(ans_2)
```

```
9
```

Question 5

Complete a series of tasks to slice strings

Answer:

```
# Task 1 - Slice the word so that you get "thon".
wrd = "Python"
# Type your answer here.
ans_1 = wrd[2:]
print(ans_1)
```

```
thon
```

```
# Task 2 - Slice the word until "o". (Pyth)
wrd = "Python"
# Type your answer here.
ans_1 = wrd[:4]
print(ans_1)
```

```
Pyth
```

```python
# Task 3 - Now try to get "th" only.
wrd = "Python"
# Type your answer here.
ans_1 = wrd[2:4]
print(ans_1)
```

```
th
```

```python
# Task 4 - Now slice the word with steps of 2, excluding first and last characters
wrd = "Python"
# Type your answer here.
ans_1 = wrd[1:-1:2]
print(ans_1)
```

```
yh
```

Question 6

Explain what this program does:

```python
for number in range(100):
        output = 'o' * number
print(output)
```

Answer:

For each number in range of 100 (99 numbers, as zero is included), program will repeat ‘o’ as many times as the number represent. It will run through all of the numbers, and then print out the last result, which is a string of 99 ‘o’s. If the ‘print’ was included within the ‘for’ loop, it would print the corresponding strings for each number.

Question 7

Your boss really likes calculating VAT on their purchases. It is their favourite hobby. They've written this code to calculate VAT and need your help to fix it.

```python
def calculate_vat(amount):
        amount * 1.2
total = calculate_vat(100)
print(total)
```

When your boss runs the program they get the following output:

```
None
```

Your boss expects the program to output the value 120 . What is wrong? How do you fix it?

Answer:

They forgot to return anything from that function. It can be fixed by adding:

```python
def calculate_vat(amount):
    return amount * 1.2
total = calculate_vat(100)
print(total)
```

```
120.0
```

Question 8

Write a new function to print a 'cashier receipt' output for a shop (any shop – clothes, food, events etc).
It should accept 3 items, then sum them up and print out a detailed receipt with TOTAL.
For example:

Input:

Item_1_name = 'Trainers'
Item_1_price = 50.45
Item_2_name = 'T-shirt
Item_2_price = 12

Output:

Trainers 50.45
T-shirt 12.00
TOTAL 62.45

Answer:

```python
# Create an empty dictionary to store values for later
items = {}

# Request information from user
Item_1_name = input('What is the name of the first item? ')
Item_1_price = float(input("First items price: "))
Item_2_name = input('What is the name of the second item? ')
Item_2_price = float(input("Second items price: "))
Item_3_name = input('What is the name of the third item? ')
Item_3_price = float(input("Third items price: "))

# Add keys and values to the dictionary
items = {Item_1_name: Item_1_price,
        Item_2_name: Item_2_price,
        Item_3_name: Item_3_price}

# Define function
def cashier_receipt(d):
    total_sum = 0
    for key in d:
        print(key, d[key])
        total_sum = total_sum + d[key]
    print('TOTAL')
    print(total_sum)

# Run the function with the input from user
cashier_receipt(items)
```

```
What is the name of the first item? Cinnamon bun
First items price: 7.20
What is the name of the second item? Raspberry croissant
Second items price: 6.60
What is the name of the third item? Coffee
Third items price: 2.50
Cinnamon bun 7.2
Raspberry croissant 6.6
Coffee 2.5
TOTAL
16.3
```