

HOMEWORK WEEK 2

TASK 1 (SQL)

- **MySQL Index:**

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

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

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

(TASK 1) - SQL

ANSWERS:

- SINGLE COLUMN AND MULTI COLUMN INDEX:

SINGLE COLUMN - (source: <https://dev.mysql.com/doc/refman/8.0/en/column-indexes.html>)

The most common type of index involves a single column, storing copies of the values from that column in a data structure, allowing fast lookups for the rows with the corresponding column values. The data structure lets the index quickly find a specific value, a set of values, or a range of values, corresponding to operators such as =, >, ≤, BETWEEN, IN, and so on, in a WHERE clause.

Syntax:

```
CREATE INDEX <index_name>
```

```
ON <table_name>(column_list);
```

Example:

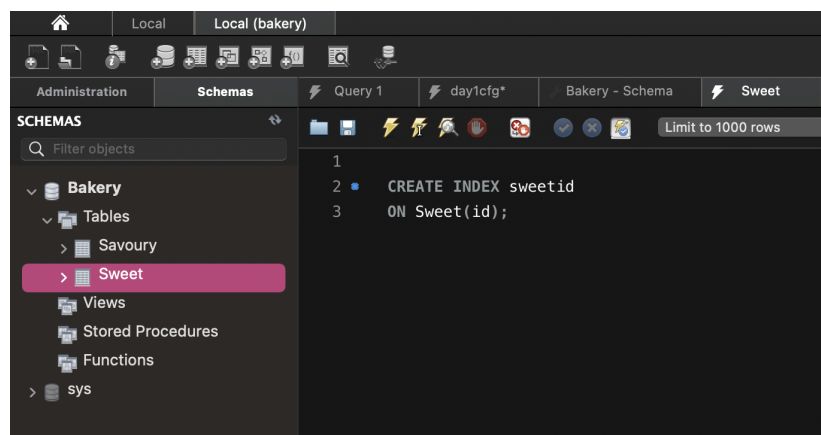
```
CREATE INDEX idx_lastname
```

```
ON Person (Lastname);
```

*CREATE INDEX cannot be used to create a PRIMARY KEY.

(source: <https://www.w3resource.com/mysql/creating-table-advance/create-index.php>)

- Add a new index to the 'Sweet' table in Bakery database:



Explanation:

The above MySQL statement will create an INDEX on 'sweetid' column for 'Sweet' table.

MULTICOLUMN INDEXES - (source: <https://dataschool.com/sql-optimization/multicolumn-indexes/>)

Multicolumn indexes (also known as composite indexes) are similar to standard indexes. They both store a sorted “table” of pointers to the main table. Multicolumn indexes however can store additional sorted pointers to other columns.

Standard indexes on a column can lead to substantial decreases in query execution times on optimizing queries. Multi-column indexes can achieve even greater decreases in query time due to its ability to move through the data quicker.

SYNTAX:

```
CREATE INDEX <index_name>
```

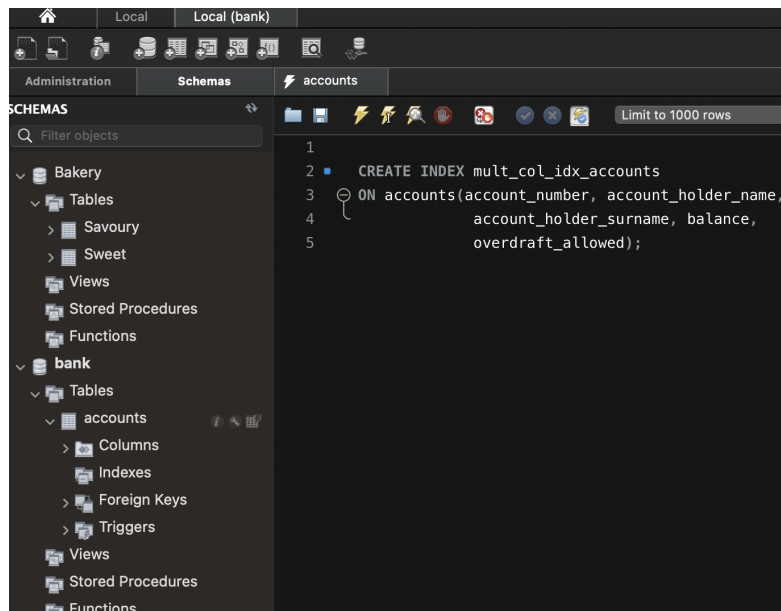
```
ON <table_name>([column1, column2, column3,...]);
```

EXAMPLE:

```
CREATE INDEX mult_col_idx_vehicle
```

```
ON traffic_data(year, make, model);
```

- Add a new index (multi-column) to the table 'Accounts' in the Bank database.



Explanation:

CREATE INDEX mult_col_idx_accounts

ON accounts (account_number, account_holder_name, account_holder_surname, balance,
overdraft_allowed);

- Perform much better once additional Columns are added to the query.
- Column order is very important.
 - The second column in a multicolumn index can never be accessed without accessing the first column as well.

- Composite index

A composite index is an index on multiple columns. MySQL allows you to create a composite index that consists of up to 16 columns.

A composite index is also known as a multiple-column index.

The query optimizer uses the composite indexes for queries that test all columns in the index, or queries that test the first columns, the first two columns, and so on.

If you specify the columns in the right order in the index definition, a single composite index can speed up these kinds of queries on the same table.

you can add a composite index to an existing table by using the CREATE INDEX statement:

SYNTAX:

```
CREATE INDEX <index_name>
```

```
ON <table_name>([column1, column2, column3,...]);
```

EXAMPLE:

The following statement creates a composite index over the lastName and firstName columns:

```
CREATE INDEX name  
ON employees(lastName, firstName);
```

First, the name index can be used for lookups in the queries that specify a lastName value because the lastName column is the leftmost prefix of the index.

Second, the name index can be used for queries that specify values for the combination of the lastName and firstName values.

The name index, therefore, is used for lookups in the following queries:

1) Find employees whose last name is Patterson.

```
SELECT firstName, lastName, email  
FROM employees  
WHERE lastName = 'Patterson';
```

This query uses the name index because the leftmost prefix of the index, which is the lastName column, is used for lookups.

You can verify this by adding the EXPLAIN clause to the query:

```
EXPLAIN SELECT firstName, lastName, email  
FROM employees  
WHERE lastName = 'Patterson';
```

HERE IS THE OUTPUT:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees	NULL	ref	name	name	52	const	3	100.00	NULL

- Clustered index

Cluster index is a type of index which sorts the data rows in the table on their key values.

In the Database, there is only one clustered index per table.

A clustered index defines the order in which data is stored in the table which can be sorted in only one way. So, there can be only a single clustered index for every table.

In an RDBMS, usually, the primary key allows you to create a clustered index based on that specific column.

Characteristic of Clustered Index:


- Default and sorted data storage
- Use just one or more than one columns for an index
- Helps you to store Data and index together
- Fragmentation
- Operations
- Clustered index scan and index seek
- Key Lookup

EXAMPLE OF CLUSTERED INDEX:

In the example below, SalesOrderDetailID is the clustered index. Sample query to retrieve data

```
SELECT CarrierTrackingNumber, UnitPrice
FROM SalesData
WHERE SalesOrderDetailID = 6;
```

SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice
43659	1	4911-403C-98	1	776	1	2024.994
43659	2	4911-403C-98	3	777	1	2024.994
43659	3	4911-403C-98	1	778	1	2024.994
43659	4	4911-403C-98	1	771	1	2039.994
43659	5	4911-403C-98	1	772	1	2039.994
43659	6	4911-403C-98	2	773	1	2039.994
43659	7	4911-403C-98	1	774	1	2039.994
43659	8	4911-403C-98	1	775	1	28.8404
43659	9	4911-403C-98	1	776	1	28.8404
43659	10	4911-403C-98	1	777	1	5.70
43659	11	4911-403C-98	2	712	1	5.1865
43659	12	4911-403C-98	4	711	1	20.1865
43660	13	6431-4D57-83	1	762	1	419.4589
43660	14	6431-4D57-83	1	758	1	874.794
43661	15	4E0A-4F89-AE	1	745	1	809.76



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:

```
chairs = '15'
```

```
nails = 4
```

```
total_nails = int(chairs) * nails
```

```
message = 'I need to buy {} nails'
```

```
print(message.format(total_nails))
```

==> chairs is a string type, do casting to convert to integer. use int().

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:

```
my_name = 'Penelope'
```

```
my_age = 29
```

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

```
print(message)
```

"Penelope class undefined, it should be enclosed in a quote to declare as a string.

Correct => my_name = 'Penelope'

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"

ANSWER:

```
boxes = input("Enter number of boxes: ")
omellete = int(boxes)*6/4
message = "You can make {} omelettes with {} boxes of eggs.".format(omellete,boxes)
print(message)
```

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)

Task 1 - ANSWER - I love coding!

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

Task 2 - Reassign str so that all of its characters are lowercase.
my_str_1 = "EVERY Exercise Brings Me Closer to Completing my GOALS."

Task 2 - ANSWER - every exercise brings me closer to completing my goals.

```
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 - Output whether this string starts with an A or not

my_str_2 = "We enjoy travelling" # Type your code here.

ans_1 =

print(ans_1)

Task 3 - ANSWER - The string does not starts with 'A'.

```
my_str_2 = "We enjoy travelling"
ans_1 = (my_str_2[0])
    if my_str_2[0]=="A":
        ans_1 = "The string starts with 'A'."
    else:
        ans_1 = "The string does not starts with 'A'."
print(ans_1)
```

Task4 - What is the length of the given string? my_str_3="1.458.001"

Task4 - ANSWER - 9

my_str_3="1.458.001"

ans_2=len(my_str_3)

print(ans_2)

Complete a series of tasks to slice strings

Task 1 - Slice the word so that you get "thon".

wrd="Python"

ANSWER: thon

wrd="Python"

ans_1= wrd[2:]

print(ans_1)

Question 5

Task 2 - Slice the word until "o". (Pyth)

wrd="Python"

Type your answer here. ans_1=

print(ans_1)

ANSWER: Pyth

wrd="Python"

ans_1= wrd[4:5]

print(ans_1)

#Task3-Nowtrytoget"th"only.

wrd="Python"

Type your answer here. ans_1=

print(ans_1)

ANSWER: th

```
wrd="Python"
```

```
ans_1= wrd[2:4]
```

```
print(ans_1)
```

Task 4 - Now slice the word with steps of 2, excluding first and last characters

```
wrd="Python"
```

Type your answer here. ans_1=

```
print(ans_1)
```

ANSWER: Pn

```
wrd="Python"
```

```
ans_1= wrd[:-5] + wrd[-1:]
```

```
print(ans_1)
```

Question 6

Explain what this program does:

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

ANSWER:

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

==>It will print letter 'o' multiple times that corresponds to the value of number, as the number iterates from 1 to 100.

For example, number = 3, it will print 'ooo', number = 4, it prints 'oooo' and so on.

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.

```
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:

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

==>Although a parameter of value 100 was passed as an argument to the function `calculate_vat`, inside the function the calculated value was not returned.

Insert the keyword 'return' in front of "amount * 1.2."

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

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

ANSWER:

```
Item_1_name = input('Item_1_name = ')
Item_1_price = float(input('Item_1_price = '))
Item_2_name = input('Item_2_name = ')
Item_2_price = float(input('Item_2_price = '))
Item_3_name = input('Item_3_name = ')
Item_3_price = float(input('Item_3_price = '))
Total = float(Item_1_price + Item_2_price + Item_3_price)
```

Receipt = ""

```

_____  

Receipt  

_____  

1. {}      {:.2f}  

2. {}      {:.2f}  

3. {}      {:.2f}  

_____  

Total      {:.2f}  

""
```

```
print(Receipt.format(Item_1_name,Item_1_price,Item_2_name,Item_2_price,Item_3_name,Item_3_price,Total))
```