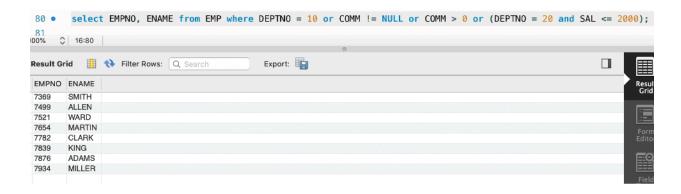
## **CSCD 327 Lab #3 (13 points)**

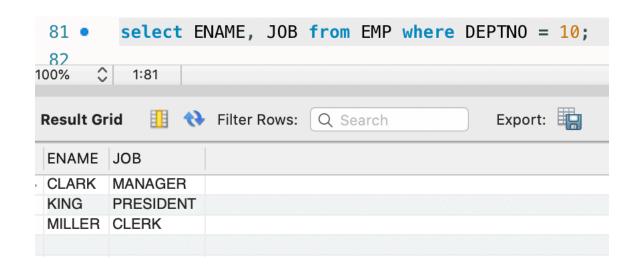
## Include the SQL statements AND the query results in your submission.

## Section 1: Use database *YourUsername\_1* to complete the following queries in SQL.

1. Find all the employees in department 10, along with any employees who earn a commission (i.e., comm isn't null), along with any employees in department 20 whose salary are at most \$2000.

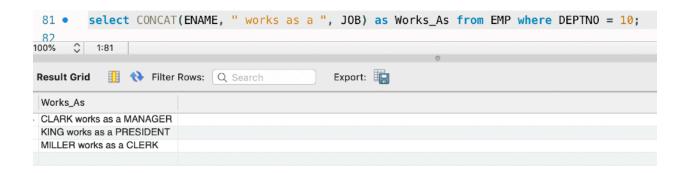


2. ListtheENAMEandJOBofemployeesassignedtodepartmentnumber 10.



 Can you display the query result from Question 2 as the following? (Hint: MySQL supports a function called CONCAT to concatenate values from multiple columns.) Works As

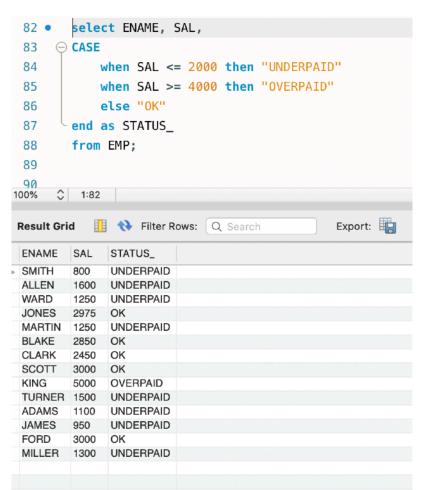
CLARK WORKS AS A MANAGER KING WORKS AS A PRESIDENT MILLER WORKS AS A CLERK



4. Sometimes you want to perform IF-ELSE operations on values in your SELECT statement. For example, you would like to produce a result set such that, if an employee is paid \$2000 or less,

a message

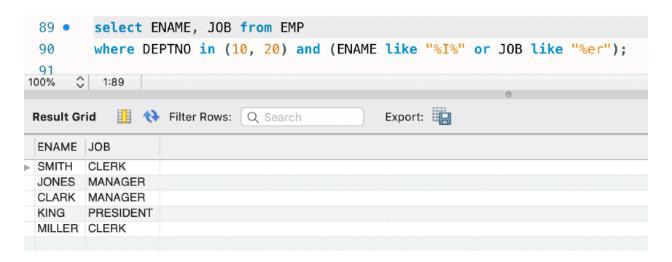
of



"UNDERPAID" is returned, if an employee is paid \$4000 or more, a message of "OVERPAID" is returned, if they make somewhere in between, then "OK" is returned. The result set should look like this:
3000 OK

**Hint**: Use the **CASE** expression to perform conditional logic directly in the SELECT statement. CASE is combined with WHEN and THEN to specify the condition.

5. Find all the employees in departments 10 and 20, and return only those that have either an "I" somewhere in their name or a job title ending with "ER".

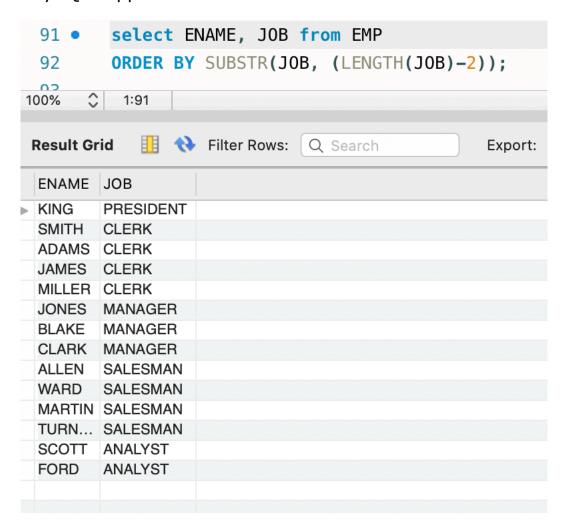


6. Return employee names and jobs from table EMP and sort by the last THREE characters in the job field. The result set should look like the following:

ENAME ----- KING SMITH ADAMS JAMES MILLER JONES CLARK BLAKE ALLEN MARTIN WARD TURNER SCOTT FORD

JOB ----- PRESIDENT CLERK CLERK CLERK CLERK MANAGER MANAGER MANAGER SALESMAN SALESMAN SALESMAN SALESMAN ANALYST ANALYST

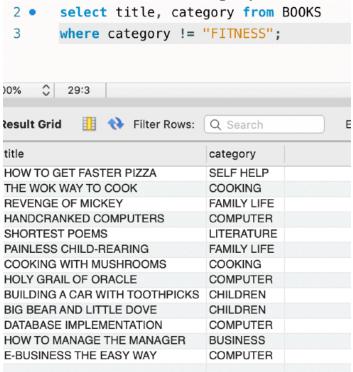
Hint: MySQL supports SUBSTR function and LENGTH function.



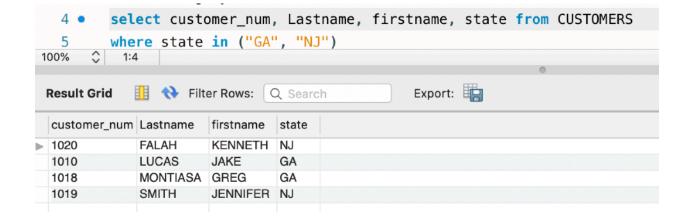
**SUBSTR(str,pos)**: Select all characters from <str> starting with position <pos>. **LENGTH(str)**: Return the length of <str>.

Section 2: Use database *YourUsername\_3* to complete the following queries in SQL.

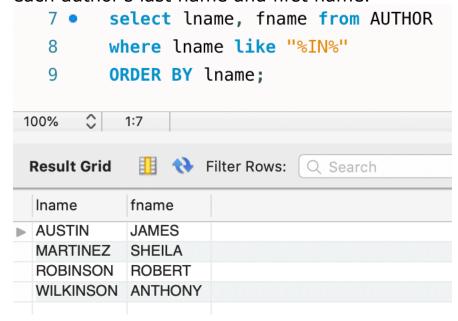
7. Find all the books that are **NOT** in the Fitness category. List each book title and category.



8. Find all the customers who live in Georgia or New Jersey. Put the results in ascending order by last name. List each customer's customer number, last name, and state.

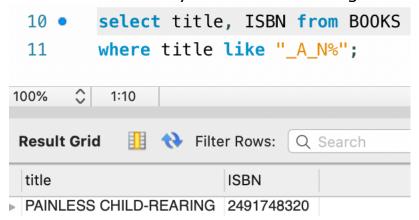


9. List all authors whose last name contains the letter pattern "IN". Put the results in order of last name, then first name. List each author's last name and first name.



2

10. Use a search pattern to find any book title with "A" for the second letter and "N" for the fourth letter. List each book's ISBN and title. Sort the list by title in descending order.



Section 3: Use database *YourUsername\_5* to complete the following queries in SQL.

11. Return one column from the Customers table named full\_name that joins the last\_name and first\_name columns.

Format this column with the last name, a comma, a space, and the first name like this:

## Doe, John

Sort the result set by last name in ascending order and return only the customers whose last name begins with letters from M to Z.



12. Return these column names and data from the Products table: product\_name list\_price discount\_percent discount\_amount discount\_price

The product\_name column

The list\_price column

The discount\_percent column

A column that's calculated from the previous two columns

A column that's calculated from the previous three columns

Round the discount\_amount and discount\_price columns to 2 decimal places. Sort the result set by discount price in descending order.

- 1					
5 • select product_nam	ne, list_	_price, disco	unt_percent,		
<pre>6 Round(concat(list_</pre>	price *	(0.01 * disc	ount_percent))	, 2) as discount_amount,	
7 Round(concat(list	price -	(list price :	* (0.01 * disc	ount_percent))), 2) as discount_p	pri
	.p. 230	,	, , , , , , , , , , , , , , , , , , , ,		
, p ,					
9					
100% 🗘 1:5			0		
Result Grid 🏥 💎 Filter Rows:	Q Search	Exp	ort: 📳		
	[		1		
product_name	list_price	discount_percent	discount_amount	discount_price ^	
Rodriguez Caballero 11	415.00	39.00	161.85	253.15	
Washburn D10S	299.00	0.00	0.00	299.00	
Yamaha FG700S	489.99	38.00	186.20	303.79	
Hofner Icon	499.99	25.00	125.00	374.99	
Fender Stratocaster	699.00	30.00	209.70	489.30	
Ludwig 5-piece Drum Set with Cymbals	699.99	30.00	210.00	489.99	
Fender Precision	799.99	30.00	240.00	559.99	
Tama 5-Piece Drum Set with Cymbals	799.99	15.00	120.00	679.99	
Gibson Les Paul	1199.00	30.00	359.70	839.30	
Gibson SG	2517.00	52.00	1308.84	1208.16	

13. Write a SELECT statement that returns these column names and data from the Order Items table:

item\_id item\_price discount\_amount quantity price\_total

discount\_total item\_total

The item\_id column

The item\_price column

The discount\_amount column

The quantity column

A column that's calculated by multiplying the item price by the quantity

A column that's calculated by multiplying the discount amount by the quantity

A column that's calculated by subtracting the discount amount from the item price and then multiplying by the quantity

Only return rows where the item\_total is greater than 500. Sort the result set by item total in descending order.

```
9 •
       select item_id, item_price, discount_amount, quantity,
        concat(item_price * quantity) as price_total, concat(discount_amount * quantity) as discount_total,
 10
 11
        concat((item_price - discount_amount) * quantity) as item_total
 12
        from order_items
 13
        where ((item_price - discount_amount) * quantity) > 500
 14
        order by item_total DESC;
 15
100% 🗘 1:9
Export:
item_id item_price discount_amou... quantity price_total discount_to... item_total
      1199.00
              359.70
                                 1199.00
                                         359.70
                                                   839.30
11
      799.99
                                         120.00
                                                   679.99
              120.00
                                 799.99
9
      799.99
              240.00
                                 799.99
                                         240.00
                                                   559.99
5
      1199.00
              359.70
                                 2398.00
                                         719.40
                                                   1678.60
3
      2517.00
                                 2517.00
                                         1308.84
                                                   1208.16
              1308.84
```