ITMD 523	*Advanced Database Management*	Homework 5
Student Name	**Harshal Sawant**	Due Date 11/07/23
Instructor	**Luke Papademas**	Section 05

Part	1	2	3	4	Total
maximum	**25** points	**25** points	**25** points	**25** points	**100** points
Your Score					

Topic: Introduction to the Structured Query Language

Reading Assignment: Thoroughly read Chapter 7 in the course textbook.

Part 1 Glossary Terms

Review, in detail, each of these glossary terms from the realm of computer database systems and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes and / or course textbook(s) as references.

(a) Between

Between:
The SQL comparison operator "BETWEEN" is used to ascertain if a value is inside a certain range. You may use a database query to filter data according to a range of values for a certain attribute. For the operator to function properly, it is crucial to first supply the lower range value.
For example:
If we want to find a price of a certain product from a PRODUCTS table where the PRICE of the product we want to find should be between 250 to 300; then the query for that will be like following:

```
SELECT *
FROM PRODUCTS
WHERE PRICE BETWEEN 250.00 AND 300.00;
```

In the above scenario:
The table "PRODUCTS" is where you may find details about every product in your business.
The property called "PRICE" is used to store each product's pricing.
The goods whose prices fall within the designated range of $250.00 to $300.00 are filtered using the "BETWEEN" operator.
A list of all the goods in your database that fit the pricing requirements will be the output of this query. This may be helpful for many things, such creating reports, revising prices, or looking at the stock of goods that fall within a certain price range.

(b) Cross Join

Cross Join:
In relational databases, a cross join, also called a cartesian product, is a kind of join operation that joins all the rows from one table with all the rows from another, creating a new table in which every row from the first table is matched with every row from the second. It creates a result set whose size is equal to the multiplication of the number of rows in each of the two tables involved, thereby acting as the product of the two tables.
Example:

SELECT Students.Student_Name, Courses.Course_Name
FROM Students
CROSS JOIN Courses;
In the above scenario:
There are two database tables with information about students and courses, they are called "Students" with “Student_Name” column and "Courses" with “Course_Name” column, containing 3 records in each table respectively, in this example. The SQL query creates a comprehensive list of all potential student-course pairings by utilizing a Cross Join to join every student from the "Students" dataset with every course from the "Courses" table. As a consequence, a new table is created in which the names of all the students enrolled in the courses are linked. The output will give nine rows overall.

(c) `EXISTS`

EXISTS:
"EXISTS" is a SQL operator used in computer database systems that determines whether a subquery returns any results. The "EXISTS" condition is met if the subquery yields one or more rows, in which case the main query is executed. On the other hand, the execution of the main query is ignored if the subquery returns no results, indicating that the condition is false. When paired with correlated subqueries, "EXISTS" may assess correlations between the subquery and the main query. It's a useful tool that's frequently used in database queries to filter data according to particular criteria.
Example:
SELECT EmployeeName
FROM Employees
WHERE EXISTS (SELECT * FROM TrainingHistory WHERE TrainingHistory.EmployeeID = Employees.EmployeeID);
The "EXISTS" operator in the above SQL query is used to find out which workers have finished their training. The subquery verifies whether the "EmployeeID" values in the primary "Employees" table and the "TrainingHistory" table match. The "EXISTS" condition is satisfied and the employee's name is obtained from the result set if at least one match is discovered. Using keywords like "SELECT," "WHERE," and "EXISTS" to filter data based on related records in a linked subquery, this query effectively finds workers with training records.

(d) `Is Null`

IS NULL:
The comparison operator "IS NULL" in SQL is used to ascertain whether a field or attribute in a database table is null, or more specifically, if it has a value. It is important to distinguish between a null value, which denotes the lack of any data or value, and a specific value, such as 0 or an empty string. When "IS NULL" is used in a query, it evaluates to false if an attribute has a value and to true otherwise.
Example:
SELECT Order_ID, Order_Date, Customer_ID
FROM Orders
WHERE Order_Status IS NULL;

In order to filter and obtain entries in this SQL query when the "Order_Status" property is null—a hint that the order is pending and has not yet been assigned a status—the expression "IS NULL" is used. This is a useful use of "IS NULL" that helps companies effectively manage their order fulfillment process by identifying orders that require attention or further processing.

(e) Recursive Query

Recursive Query:
A kind of database query called a recursive query joins a table to itself. When working with hierarchical or self-referential data structures, including network graphs, family trees, or organizational charts, where data items have associations with other pieces inside the same dataset, this is often utilized. Recursive queries are helpful because they allow you to traverse relationships inside a table to retrieve data from these kinds of structures.
Example:
A recursive query iterates through a table containing the following columns: Dept_Name (department name), Parent_Dept_ID (parent department ID), and Dept_ID (department ID) in the department hierarchy of a university. Starting with the upper level "University," all departments are recursively extracted and categorized according to levels by the query. The output that is produced shows the university's departments' hierarchical structure.

Part 2 Completion: True / False Exercises

For each of these exercises, enter either True or False.

(1) **False** Comparison operators cannot be used to place restrictions on character-based attributes.

(2) **True** The COUNT function is designed to tally the number of non-null "values" of an attribute and is often used in conjunction with the DISTINCT clause.

(3) **True** The SQL COUNT function gives the number of rows containing non-null values for a given column.

(4) **False** Rows can be grouped into smaller collections quickly and easily using the COUNT clause within the SELECT statement.

(5) **True** Although SQL commands can be grouped together on a single line, complex command sequences are best shown on separate lines, with space between the SQL command and the command's components.

Part 3 Multiple Choice Exercises

Select the correct answer.

(1) Consider the following table

Table name: **TENANT**
ID	NAME	RENT
18	Alex	219
31	Rose	250
51	Alex	300
77	Mary	280
81	Rose	240

How many rows are in the following query?

SELECT DISTINCT NAME FROM TENANT;

a. 5 **b**. **3**
c. 1 d. 0

(2) Consider the following table

Table name: TENANT
ID	NAME	RENT
18	Alex	219
31	Rose	250
77	Alex	300
41	Mary	280
81	Rose	240

Which query returns the following?

ID	NAME	RENT
18	Alex	219
81	Rose	240
31	Rose	250
41	Mary	280
77	Alex	300

a. **SELECT * FROM TENANT ORDER BY RENT**
b. SELECT ORDER BY RENT FROM TENANT *
c. SELECT * ORDER BY RENT FROM TENANT

d. SELECT RENT FROM TENANT ORDER BY RENT

(3) After requesting a service, a client must pay 30 days after the service is required. In the SERVICE table, there exists the attribute DATE that holds the day a service was required, but there is no attribute for the due date. Which of the following queries shows the date a service was required and when the payment is due as DUE_DATE.

a. SELECT SID, DATE, DUE_DATE FROM SERVICE
b. SELECT SID, DATE, DATE+30 FROM SERVICE
c. SELECT SID, DATE, DUE+30 AS DUE_DATE FROM SERVICE
d. SELECT SID, DATE, DUE_DATE FROM SERVICE

(4) Consider the following tables

Name: CLIENTS
Primary Key: CLIENT_ID

CLIENT_ID	NAME
19283	Jhon
19281	Mary
19272	Elizabeth

Name: SALES
Primary Key: (SALE_ID, PROD_ID)
Foreign Key: PROD_ID, CLIENT_ID

SALE_ID	PROD_ID	CLIENT_ID	DATE
FGS12301	12930182	19281	2022-07-10
DEF28358	19283104	19272	2022-07-11
GHM3920	19283123	17913	2021-06-22

How many rows produce the following query?

SELECT *
FROM CLIENTS RIGHT JOIN SALES ON CLIENTS.CLIENT_ID = SALES.CLIENT_ID;

a. 0
b. 2
c. 3
d. 6

(5) Which of the following better describes the result of SELECT * FROM table1 LEFT JOIN table2 ON table1.id = table2.id2?

a. Only the values of table1 with id matching id2 from table2.

b. Rows match the join condition and rows in table1 with unmatched id on table2.
c. Rows match the join condition and rows in table2 with unmatched id2 on table1.
d. Rows in table1 with unmatched id on table2.

Part 4 Essays Exercises

Write a brief but complete answer to each of these questions / exercises.

(1) Below are some rows of the table INVOICE

COD	PROV_COD	DATE	TYPE	LOC	TOTAL
2910	192	2022-03-11	90	TX	1928
9301	384	2022-05-03	90	NY	2800

Overdue invoices are those whose date plus TYPE days have passed. Explain why the following query is incorrect when we wish to show all invoices with overdue dates.

```
SELECT * FROM INVOICE
WHERE CURDATE() - DATE > TYPE;
```

According to the inquiry, past-due bills are as follows: Current date()>Date + TYPE
Thus, the updated inquiry will be:
```
SELECT * FROM INVOICE
 WHERE CURDATE()>Date + TYPE;
```
However, there is a distinction in data type between Type, which is numerical, and Date, which is a time stamp.
Additionally, SYSDATE is used by Oracle SQL Developers to obtain the current date, ensuring that the right query is
```
Select * FROM INVOICE
WHERE sysdate > Date + type;
```

(2) Consider the STUDENT table given below:

CODE	NAME	GPA	YEAR
291	ALEX	3.1	2
938	MICHELE	2.3	1
931	JHON	3.3	1

182	JOE	3.4	2
190	REY	2.0	2
330	RON	3.9	3

Explain which records will be returned by the following query.

SELECT YEAR, MAX(GPA) FROM STUDENT GROUP BY YEAR ORDER BY MAX(GPA);

Output will be :

Year	MAX(GPA)
1	3.3
2	3.4
3	3.9

(3) Suppose that a mattress store has a TABLE1 with sales representatives with an attribute vcode as primary key. It also has a TABLE2 with every mattress sold, the table has an attribute vcode to indicate which sales representative made the sale, and a mt_code to indicate which mattress was sold. Which of the following returns all the sales representatives that haven't closed a sale?

a. SELECT * FROM TABLE1 RIGHT JOIN TABLE2 ON TABLE1.vcode = TABLE2.vcode WHERE mt_code IS NULL.

b. SELECT * FROM TABLE1 LEFT JOIN TABLE2 ON TABLE1.vcode = TABLE2.vcode WHERE mt_code IS NULL.

c. SELECT * FROM TABLE1 RIGHT JOIN TABLE2 ON TABLE1.vcode = TABLE2.vcode WHERE mt_code IS NOT NULL.

d. SELECT * FROM TABLE1 JOIN TABLE2 ON TABLE1.vcode = TABLE2.vcode WHERE mt_code IS NULL.

b. SELECT * FROM TABLE1 LEFT JOIN TABLE2 ON TABLE1.vcode = TABLE2.vcode WHERE mt_code IS NULL

(4) Consider the following tables:

table1

vcode	name
7	Alex

8	Tony
9	Charles
11	Mary

table2

scode	vcode	total
341	7	102
213	9	59
312	7	89
712	3	301

How many rows will comprise the results yielded by the following query?

SELECT * FROM table1 NATURAL JOIN table2;

Vcode may function as a join key between two tables according to the table structure, therefore if we do a natural join, it will return the data in which VCODE data matched between the two tables, i.e.

VCODE	NAME	SCODE	Total
7	Alex	341	102
9	Charles	213	59
7	Alex	312	89

When two tables naturally connect, the result will show which table has more comparable data and why the tables joined or used a join key.

(5) Consider the following tables:

Name: table1

cod1	val1	val2

10	10	8
12	10	6
21	11	15
33	10	2
41	9	11
8	10	6
14	9	5
11	11	4

What is the result of the query below?

```
SELECT MIN(cod1)
FROM (SELECT val1, MAX(val2) max_val2
 FROM table1
 GROUP BY val1) max_table
 JOIN table1
 ON max_table.val1 = table1.val1
WHERE val2 = max_val2;
```

MIN(COD1)
10
Finding the lowest cod1 value from the rows where val2 equals the maximum val2 for each unique val1 yields the query's final result, MIN(COD1): 10. The minimum cod1 value in this instance is 10.