Data Base

Assignment 2

إسلام محمد عطية محمد

سكشن 1

الكود : 20220126

Task 1

Note: Tables are generated using SQL Server

1- Consider the following tables, the course table and the instructor table. State the name of the join for the following output and **explain** it

The following join is **Full Join**

A full outer join combines all records from both tables being joined,

2- **Create and manage** advanced query to produce the above output.

Create Course Table

```
create table tbl_course(
    course_id INT PRIMARY KEY,
    course VARCHAR(50)
);
```

Insert the Values

```
INSERT INTO tbl_course
VALUES
    (111,'Science'),
    (222,'Finance'),
    (333,'History'),
    (444,'Art'),
    (555,'Music'),
    (777,'physics');
```

```
SELECT * FROM tbl_course;
```

Results | Messages

	course_id	course
1	111	Science
2	222	Finance
3	333	History
4	444	Art
5	555	Music
6	777	physics

Create **instructor** table

```
CREATE TABLE tbl_instructor(
    instructor_id int PRIMARY KEY,
    Name VARCHAR(50),
    Place VARCHAR(50),
    Course_id int
);
SELECT * FROM tbl_instructor;
```

Results Messages

instructor_id	Name	Place	Course_id

Inserting the values

```
INSERT INTO
    tbl_instructor
VALUES
    (1, 'Lisa','california',111 ),
    (2, 'Roy','Bangalore',222),
    (3, 'Mathew','Mumbai',333),
    (4, 'Lorna','Boston',444),
    (5, 'Marcelo','Texax',555),
    (6, 'Pinto','Delhi',666);
SELECT * FROM tbl_instructor;
```

Results Messages

	instructor_id	Name	Place	Course_id
1	1	Lisa	california	111
2	2	Roy	Bangalore	222
3	3	Mathew	Mumbai	333
4	4	Lorna	Boston	444
5	5	Marcelo	Texax	555
6	6	Pinto	Delhi	666

Full Outer Join

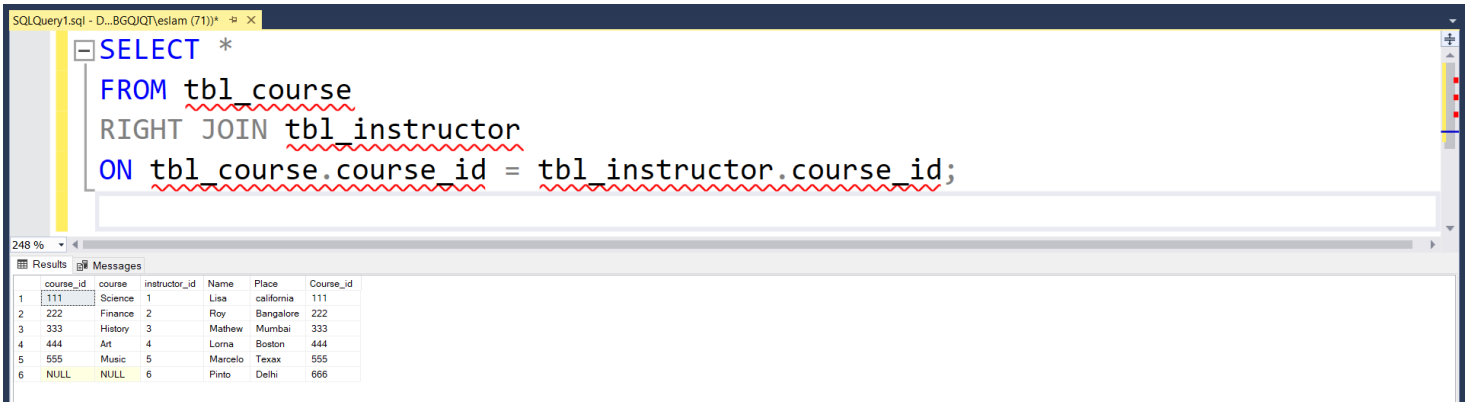
```
SELECT *
FROM tbl_course
FULL OUTER JOIN tbl_instructor
ON tbl_instructor.Course_id = tbl_course.course_id;
```

Results Messages

	course_id	course	instructor_id	Name	Place	Course_id
1	111	Science	1	Lisa	california	111
2	222	Finance	2	Roy	Bangalore	222
3	333	History	3	Mathew	Mumbai	333
4	444	Art	4	Lorna	Boston	444
5	555	Music	5	Marcelo	Texax	555
6	777	physics	NULL	NULL	NULL	NULL
7	NULL	NULL	6	Pinto	Delhi	666

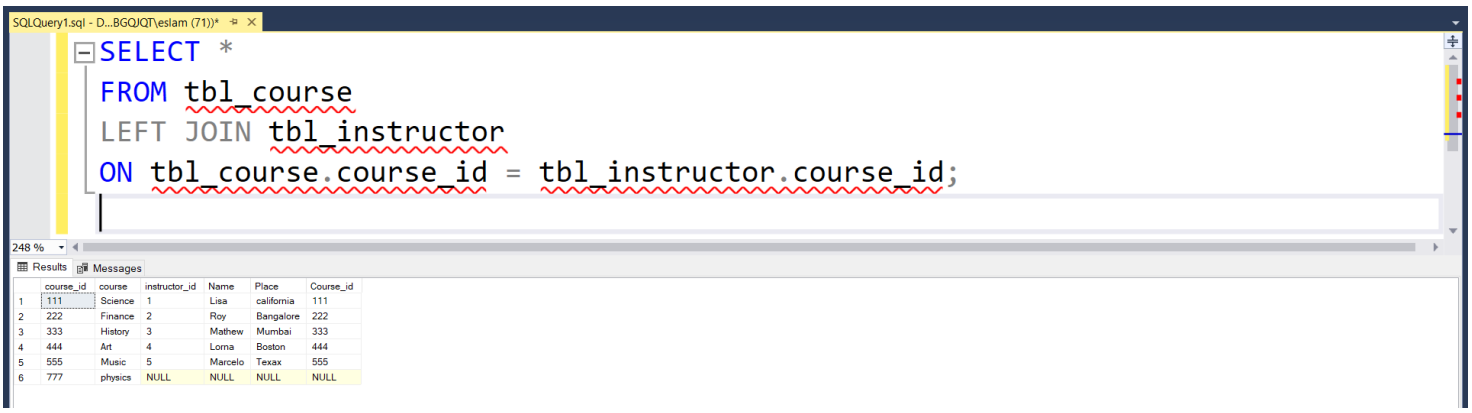
3- **Write** MySQL query to perform Left join, Right join on the above tables

Right Join



	course_id	course	instructor_id	Name	Place	Course_id
1	111	Science	1	Lisa	california	111
2	222	Finance	2	Roy	Bangalore	222
3	333	History	3	Mathew	Mumbai	333
4	444	Art	4	Lorna	Boston	444
5	555	Music	5	Marcelo	Texax	555
6	NULL	NULL	6	Pinto	Delhi	666

Left Join



	course_id	course	instructor_id	Name	Place	Course_id
1	111	Science	1	Lisa	california	111
2	222	Finance	2	Roy	Bangalore	222
3	333	History	3	Mathew	Mumbai	333
4	444	Art	4	Lorna	Boston	444
5	555	Music	5	Marcelo	Texax	555
6	777	physics	NULL	NULL	NULL	NULL

4- The following query is producing an error. Identify the error and also create the correct core SQL query:

a) GRANT TO peter@localhost ALL PRIVILEGES on *.*;

Error: <> privileges should have come before the user.

```
GRANT ALL PRIVILEGES on *.* peter;
```

b) CREATE VIEW AS SELECT name, age FROM CUSTOMERS;

Error: <syntax error> the view name wasn't exist

```
CREATE VIEW VIEW_NAME AS SELECT name, age FROM CUSTOMERS;
```

c) CREATE USER IDENTIFIED BY 'password' AHMED;

Error: <syntax error> the user name should be after USER

In MySQL: `CREATE USER AHMED IDENTIFIED BY 'password';`

5- **Describe** the SQL commands which are used to:

a. Create a virtual table based on the result-set of an SQL statement.

Command: `CREATE VIEW VIEW_NAME AS SELECT name, age FROM CUSTOMERS;`

b. Enables you to improve the faster retrieval of records on a database table. It creates an entry for each value of the indexed columns

Command: `CREATE INDEX <index_name> ON <table_name> (<column_name(s)>)`

c. A special type of stored procedure that is invoked automatically in response to an event.

Command: `CREATE TRIGGER <trigger_name> <trigger_event> ON <table_name> FOR EACH ROW <trigger_action>`

d. A collection of pre-compiled SQL statements stored inside the database. It always contains a name, parameter lists, and SQL statements

Command: `CREATE PROCEDURE <procedure_name> (<parameter_list>)`

e. It is used to combine rows from two or more tables, based on a related column between them

Command: `JOIN` (various types like `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, `FULL JOIN`)

f. It is used to specify the rule that allows or restricts what values/data will be stored in the table.

Command: `CREATE TABLE table_name (column_name data_type [, column_name data_type]..., [CONSTRAINT constraint_name CHECK (condition)])` (Constraints can define rules like NOT NULL, PRIMARY KEY, FOREIGN KEY etc.)

Task 2

6- **Explain** NoSQL database and its characteristics

NoSQL databases are a type of database management system (DBMS) that diverge from traditional relational databases (SQL) in how they store and retrieve data.

NoSQL Characteristics

- Non-relational
- Flexible schema
- Other or additional query languages than SQL
- Distributed – horizontal scaling (Scaling Out)
- Less structured data
- Supports big data

7- **Describe** CAP theorem

GIVEN:

- Many nodes
- Nodes contain replicas of partitions of the data

• Consistency

- All replicas contain the same version of data
- Client always has the same view of the data (no matter what node)

• Availability

- System remains operational on failing nodes
- All clients can always read and write

• Partition tolerance

- multiple entry points
- System remains operational on system split (communication malfunction)
- System works well across physical network partitions

8- **Compare** between SQL database and NoSQL database.

SQL	NoSQL
Relational Databases (RDBMS)	Non-relational or distributed database
Vertically scalable	Horizontally scalable
Table based databases	Document based, key-value pairs, graph databases or wide- column stores.
Supports predefined schema	Supports dynamic schema
SQL (structured query language) for defining and manipulating the data	Uses unstructured Query Language
Standard interface for executing complex query	Not good for executing complex query
Best suited for huge load and complex transactional applications	Not suited for huge load and complex transactional type applications
SQL databases maintains on ACID properties Atomicity, Consistency, Isolation and Durability)	NoSQL database follows the Brewers CAP theorem/BASE properties
Synchronous Inserts & Updates	Asynchronous Inserts & Updates

9- **Critically evaluate** NoSQL database types

1. **Document Stores:** These databases store data in flexible, JSON-like documents, allowing for dynamic schemas and nested data structures. Examples include MongoDB and CouchDB.
2. **Key-Value Stores:** Key-value databases store data as a collection of key-value pairs, offering high-speed access and simple data models. Examples include Redis, Amazon DynamoDB, and Riak.
3. **Column-Family Stores:** Also known as wide-column stores, these databases organize data into columns rather than rows, allowing for efficient storage and retrieval of large volumes of data. Examples include Apache Cassandra and HBase.
4. **Graph Databases:** Graph databases are designed for storing and querying highly connected data, making them ideal for applications with complex relationships. Examples include Neo4j, Amazon Neptune, and TigerGraph.