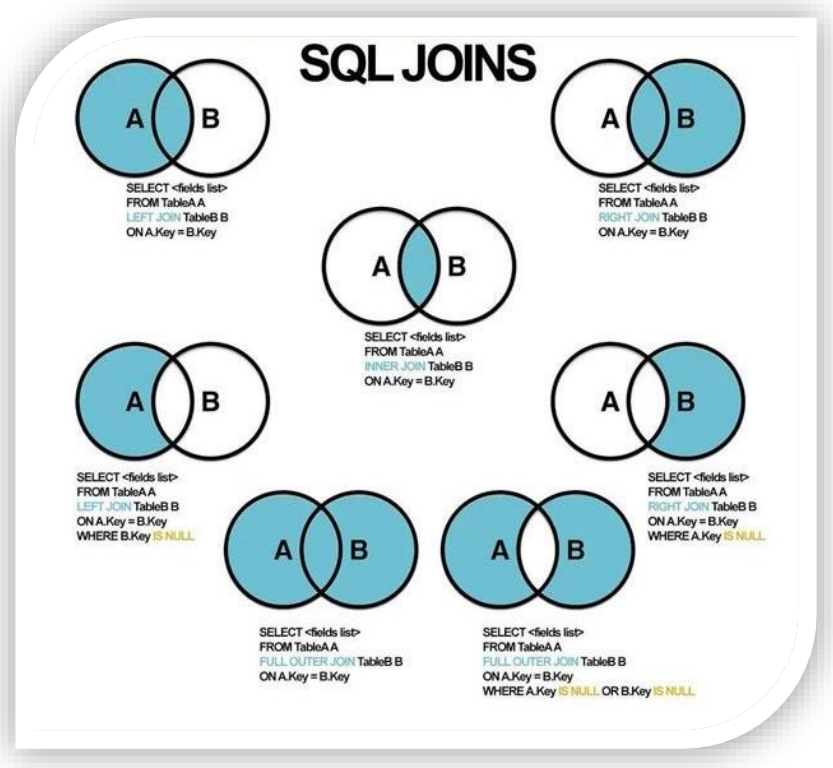
Lab No 6 CLO-2

Topic: JOINS

A `JOIN` clause is used to combine rows from two or more tables, based on a related column between them. We can retrieve data from more than one tables using the JOIN statement.



We will discuss the 4 of SQL Server types of joins:

- INNER JOIN/simple join
- LEFT OUTER JOIN/LEFT JOIN
- RIGHT OUTER JOIN/RIGHT JOIN
- FULL OUTER JOIN INNER JOIN

INNER JOIN/Simple Join

This type of JOIN returns rows from all tables in which the join condition is true. It takes the following syntax:

```
SELECT columns FROM
table_1 INNER JOIN
table_2

ON table_1.column = table_2.column;
```

We will use the following two tables to demonstrate this:

Students Table:

	admission	firstName	lastName	age
1	3420	Nicholas	Samuel	14
2	3380	Joel	John	15
3	3410	Japheth	Becky	16
4	3398	George	Joshua	14
5	3386	John	Lucky	15
6	3403	Simon	Dan	13
7	3400	Calton	Becham	16

Fee table:

	admission	course	amount_paid
1	3380	Electrical	20000
2	3420	ICT	15000
3	3398	Commerce	13000
4	3410	HR	12000

The following command demonstrates an INNER JOIN:

```
SELECT Students.admission, Students.firstName, Students.lastName, Fee.amount_paid FROM Students

INNER JOIN Fee

ON Students.admission = Fee.admission
```

The command returns the following:

	admission	firstName	lastName	amount_paid
1	3420	Nicholas	Samuel	15000
2	3380	Joel	John	20000
3	3410	Japheth	Becky	12000
4	3398	George	Joshua	13000

We can tell the students who have paid their fee. We used the column with common values in both tables, which is the admission column.

LEFT OUTER JOIN/Left Join

This type of join will return all rows from the left-hand table plus records in the right-hand table with matching values. For example:

```
SELECT Students.admission, Students.firstName, Students.lastName, Fee.amount_paid FROM
Students
LEFT OUTER JOIN Fee
ON Students.admission = Fee.admission
```

The code returns the following:

	admission	firstName	lastName	amount_paid
1	3420	Nicholas	Samuel	15000
2	3380	Joel	John	20000
3	3410	Japheth	Becky	12000
4	3398	George	Joshua	13000
5	3386	John	Lucky	NULL
6	3403	Simon	Dan	NULL
7	3400	Calton	Becham	NULL

The records without matching values are replaced with NULLs in the respective columns.

RIGHT OUTER JOIN

This type of join returns all rows from the right-hand table and only those with matching values in the left-hand table. For example:

```
SELECT Students.admission, Students.firstName, Students.lastName, Fee.amount_paid FROM
Students
RIGHT OUTER JOIN Fee
ON Students.admission = Fee.admission
```

The statement returns the following:

	admission	firstName	lastName	amount_paid
1	3380	Joel	John	20000
2	3420	Nicholas	Samuel	15000
3	3398	George	Joshua	13000
4	3410	Japheth	Becky	12000

The reason for the above output is that all rows in the Fee table are available in the Students table when matched on the admission column.

FULL OUTER JOIN

This type of join returns all rows from both tables with NULL values where the JOIN condition is not true. For example:

```
SELECT Students.admission, Students.firstName, Students.lastName, Fee.amount_paid FROM Students

FULL OUTER JOIN Fee

ON Students.admission = Fee.admission
```

The code returns the following result:

	admission	firstName	lastName	amount_paid
1	3420	Nicholas	Samuel	15000
2	3380	Joel	John	20000
3	3410	Japheth	Becky	12000
4	3398	George	Joshua	13000
5	3386	John	Lucky	NULL
6	3403	Simon	Dan	NULL
7	3400	Calton	Becham	NULL

Tasks: *Sample Table:* Salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001

Task 1:

Write a SQL query to find the salesperson and customer who belongs to same city. Return Salesman name, cust_name and city.

Task 2:

Write a SQL query to find those orders where order amount exists between 500 and 2000. Return ord_no, purch_amt, cust_name, city.

Task 3:

Write a SQL query to find the salesperson(s) and the customer(s) he handles. Return Customer Name, city, Salesman, commission.

Task 4:

From the following tables write a SQL query to find those salespersons who received a commission from the company more than 12%. Return Customer Name, customer city, Salesman, commission.

Task 5:

Write a SQL query to display the cust_name, customer city, grade, Salesman, salesman city.

Task 6:

Write a SQL query to find those customers whose grade less than 300. Return cust_name, customer city, grade, Salesman, saleman city.

Task 7:

Write a SQL statement to make a list in ascending order for the salesmen who works either for one or more customer or not yet join under any of the customers.

Task 8:

Write a SQL statement to make a report with customer name, city, order no., order date, purchase amount for those customers from the existing list who placed one or more orders or which order(s) have been placed by the customer who is not on the list.