```sql
--INNER JOIN
--Example:write a query to fetch artist name and album title they belong to;
--We can use KEYWORD Inner join or JOIN
--Find common column btwn the tables. It doesnt matter the name of the column but the value stored in the column.
 SELECT * FROM albums;
 SELECT * FROM artists;

SELECT a.Name, b.Title
FROM artists a
INNER JOIN albums b
ON a.ArtistId=b.AlbumId;--also called join condition. The most important thing.

--LEFT JOIN(ALSO LEFT OUTER JOIN) will do an inner join and add the remaining values on the left column that dont have a mat
--Example: Fetch ALL artist names and their album titles.
SELECT a.Name, b.Title
FROM artists a
LEFT JOIN albums b
ON a.ArtistId=b.AlbumId;

--RIGHT JOIN(ALSO RIGHT OUTER JOIN) will first do an inner join then add the remaining values on the right column that dont
--Using same example as above:
SELECT a.Name, b.Title
FROM artists a
RIGHT JOIN albums b
ON a.ArtistId=b.AlbumId;

--COMBINATION  of JOINS
--Eample: Write a query to fetch details of ALL track names, their artists, their genre names and albums titles
SELECT * FROM tracks;
SELECT * FROM genres;
SELECT * FROM artists;
SELECT * FROM albums;

SELECT t.Name, a.Name, g.Name, b.Title
FROM tracks t
LEFT JOIN artists a
on a.ArtistId=t.TrackId
INNER JOIN genres g
ON g.GenreId=t.GenreId
LEFT JOIN albums b
ON b.AlbumId=t.TrackId;


--FULL OUTER JOIN(ALSO FULL JOIN)
--Full join=Inner Join+Left Join+ Right Join
SELECT * FROM albums
SELECT * FROM artists;

SELECT a.Name, B.Title
FROM artists a
FULL OUTER JOIN albums b
on a.ArtistId=b.AlbumId

--CROSS JOIN
--Dont need to specify join condition('on')
--CROSS JOIN RETURNS CARTESIAN PRODUCT(total rows displayed will be a multtiple of the two rows in the tables)
SELECT a.Name , b.Title
FROM albums b
CROSS JOIN artists a;

--you can use it with an INNER JOIN
--Write a query to fetch the artists name, album title and make sure to display the invoice date and corresponding billing c
SELECT a.Name, b.Title, i.InvoiceDate, i.BillingCountry
FROM artists a
INNER JOIN albums b
ON a.ArtistId=b.AlbumId
CROSS JOIN invoices i;


--NATURAL JOIN
```

```sql
--Don't specify join condition('on')
--Joins on basis of similar column names.
--If no similar columns are present, it will perform a CROSS JOIN
--Since dataset playlists and playlist_track have a similar column 'PlaylistId' we can perform a Natural Join succesfully.
SELECT * FROM playlists
SELECT * FROM playlist_track

SELECT *
FROM playlists p
NATURAL JOIN playlist_track q;


CREATE TABLE family ( FirstName VarChar(20),
  age VarChar(20),
  member_id VarChar(20),
  parent_id VarChar(20))

SELECT * FROM family

INSERT INTO family(FirstName,age,member_id,parent_id)
VALUES('Chandler','32','F1','F5'),('Monicah','23','F2','F5'),('Phoebe','15','F3','F5');

INSERT INTO family(FirstName,age,member_id,parent_id)
VALUES('David','20','F4',''),('Carol','50','F5','F6'),('Stewart','70','F6','');

INSERT INTO family(FirstName,age,member_id,parent_id)
VALUES('Rowan','35','F7','F4'),('Asha','8','F8','F6');

--SELF JOIN is used when you need to match record of the SAME table
--Write a query to fetch the child name and their age corresponding to their parent name and parent age.
SELECT child.FirstName as child_name,
child.age as child_age,
parent.age as parent_age,
parent.FirstName as parent_name
FROM family as child
JOIN family as parent
ON child.parent_id=parent.member_id;

--introducing a left join;
SELECT child.FirstName as child_name,
child.age as child_age,
parent.age as parent_age,
parent.FirstName as parent_name
FROM family as child
LEFT JOIN family as parent
ON child.parent_id=parent.member_id;
```