DBMS Lec 7 (SQL) :

These notes contains extra knowledge I gained while watching tutorial.

- MySQL command-Line client is 2 tier architecture where as the MySQL workbench is 3 tier architecture.
- to state a foreign key in a table : Worker is parent class and bonus is child class.

```
CREATE TABLE Bonus(
    WORKER_ REF_ID INT,
    BONUS_AMOUNT INT(10),
    BONUS_DATE DATETIME,
    FOREIGN KEY (WORKER_ REF_ID)
        REFERENCES WORKER (WORKER_ID)
        ON DELETE CASCADE
);
```

Interview Question → Will SELECT query run without FROM keyword????

- If we have made some database with some tables in our workbench then we can convert it into its equivalent ER diagram by the *REVERSE ENGINEER OPTION* in workbench.
- SQL uses DUAL TABLE (Dummy) for answering
    i. SELECT 44 + 11; will give → 55 in a table named 44 + 11
    ii. SELECT lcase('Ram') will give → ram in a table named lcase('Ram').

Wildcards:

- Used with 'like' keyword
- % → any number of characters
- _ → only one

Sorting:

- By default order is ascending.
- SELECT * FROM worker order by salary ASC;
- SELECT * FROM worker order by salary DESC;

Data Grouping:

- To achieve aggregation (count, sum, avg, max, min).
- SELECT department, COUNT(department) FROM worker GROUP BY department; (returns distinct departments with their counts).

- As we use WHERE with SELECT for filtering.
- We use HAVING with GROUP BY or HAVING is used before GROUP BY only.

Constraints:

- PRIMARY KEY.
- Foreign KEY.
- UNIQUE.
- CHECK (e.g. age > 12).
- DEFAULT (sets default value of column).

ALTER KEYWORD:

- Used to change ADD/DROP/RENAME/MODIFY columns of table.

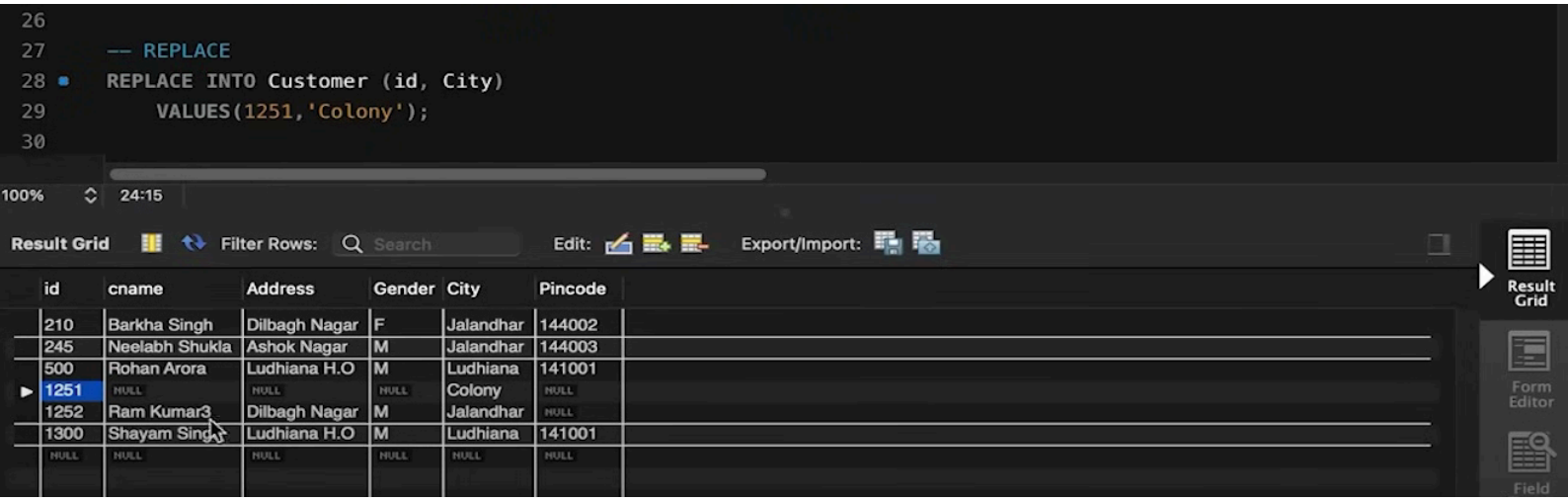
ON UPDATE CASCADE / ON DELETE CASCADE:

- In situation where we have 2 tables and one tables contains primary key of other table in itself (foreign key (parent-child relationship)), then there may be cases when we have to update/delete values in one table.
- using ON UPDATE CASCADE / ON DELETE CASCADE, changes are reflected in parent table too if made in child and visca barca.

Extra knowledge:

- MySQL has a default feature of safety whenever we try to change all rows.
- We can first off it by → SET SQL_SAFE_UPDATES = 0;

REPLACE vs INSERT:



- From above image, we had a existing data at id 1251 but on using replace query it gets replaced. We only specified the city that's why other fields are null.
- If we didn't had any data existing with id 1251 then this REPLACE query would have worked like INSERT QUERY.

REPLACE vs UPDATE:

- If row is not present then REPLACE will add a new row while UPDATE will do nothing.

JOINS:

- To apply joins, there should be an common attribute.
- Suppose we have two tables (customer and order) and they have key attribute common in them.

INNER JOIN:

- Returns a table contaning intersecting column of left and right table.
- SELECT c.* , o.* FROM customer AS c INNER JOIN order AS ON c.id = o.cust_id;

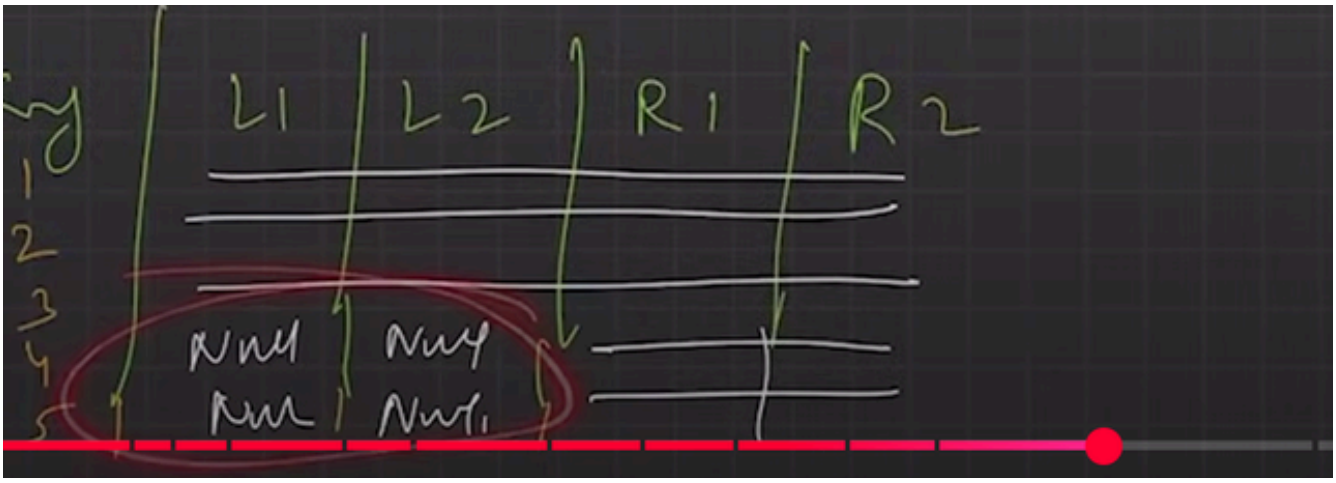
LEFT JOIN:

- Returns a table contaning all left table's data and matching data from right table.
- And it has null values for column of right table (suppose we get matching data of left and right table but here we are not only getting matching but also the whole left table data which will have null values in right columns value).

- SELECT c.* , o.* FROM customer AS c LEFT JOIN order AS ON c.id = o.cust_id;

RIGHT JOIN:

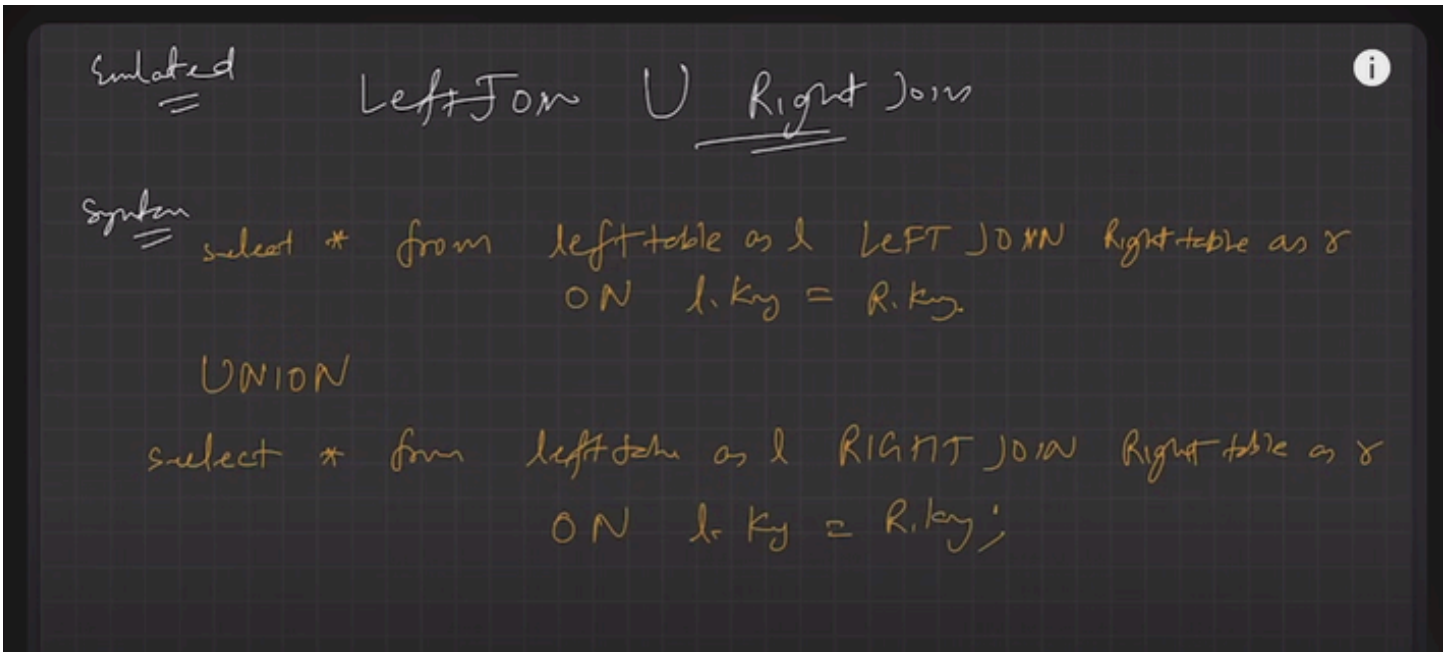
- Returns a table contaning intersecting column of left and right table.



- SELECT c.* , o.* FROM customer AS c RIGHT JOIN order AS ON c.id = o.cust_id;

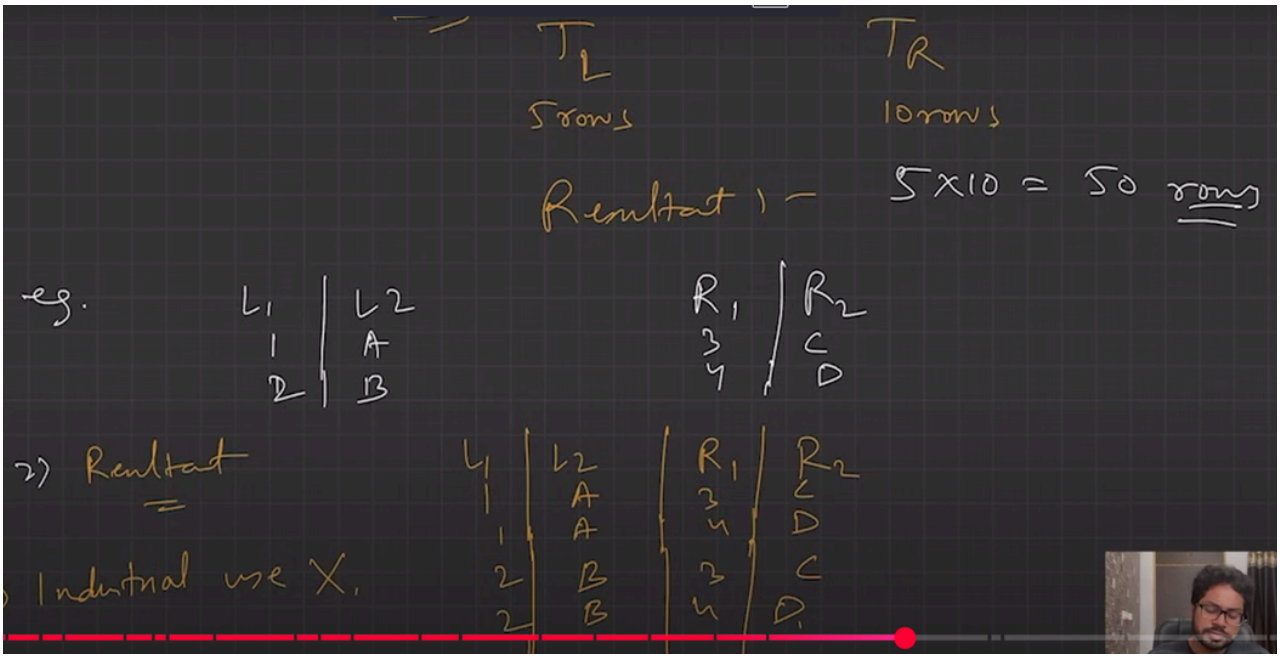
FULL JOIN:

- Returns a resulting table that contains all data when there is a match on left or right table data.
- *There is no FULL JOIN keyword in MYSQL.*
- LEFT JOIN U RIGHT JOIN.



CROSS JOIN (Cartesian Product):

- **No Industrial use.**



SELF JOIN:

- Used for Unary relationship.
- No keyword for SELF JOIN.

- SELECT e1.id, e2.id, e2.name

  FROM employee AS e1

  INNER JOIN employee AS e2

  ON e1.id = e2.id;

Can we use joins without ‘join’ keyword?

- Yes.
- SELECT * FROM left_table, right_table WHERE left_table_id = right_table_id;

SET OPERATIONS:

- Set contains unique elements.
- Joins vs Union

SET OPERATIONS

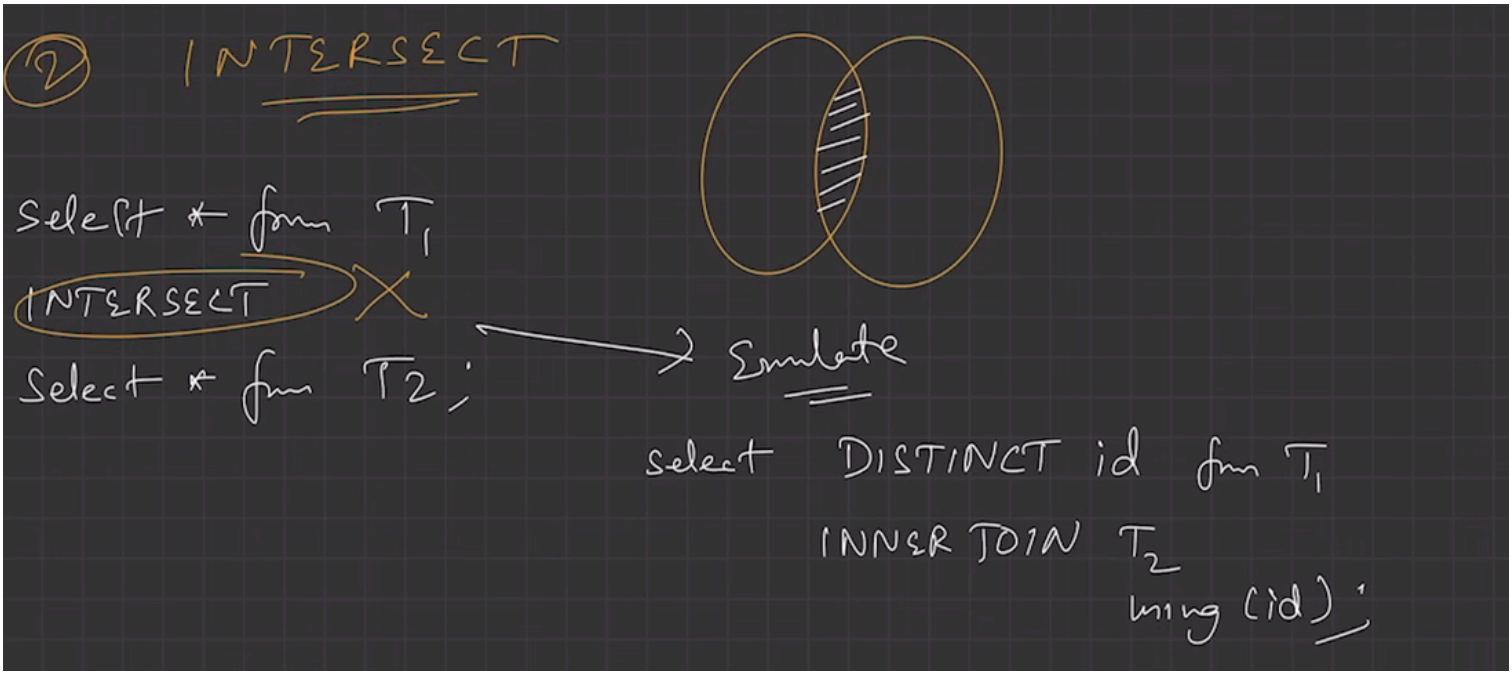
1. Used to combine multiple select statements.
2. Always gives distinct rows.

JOIN	SET Operations
Combines multiple tables based on matching condition.	Combination is resulting set from two or more SELECT statements.
Column wise combination.	Row wise combination.
Data types of two tables can be different.	Datatypes of corresponding columns from each table should be the same.
Can generate both distinct or duplicate rows.	Generate distinct rows.
The number of column(s) selected may or may not be the same from each table.	The number of column(s) selected must be the same from each table.
Combines results horizontally.	Combines results vertically.

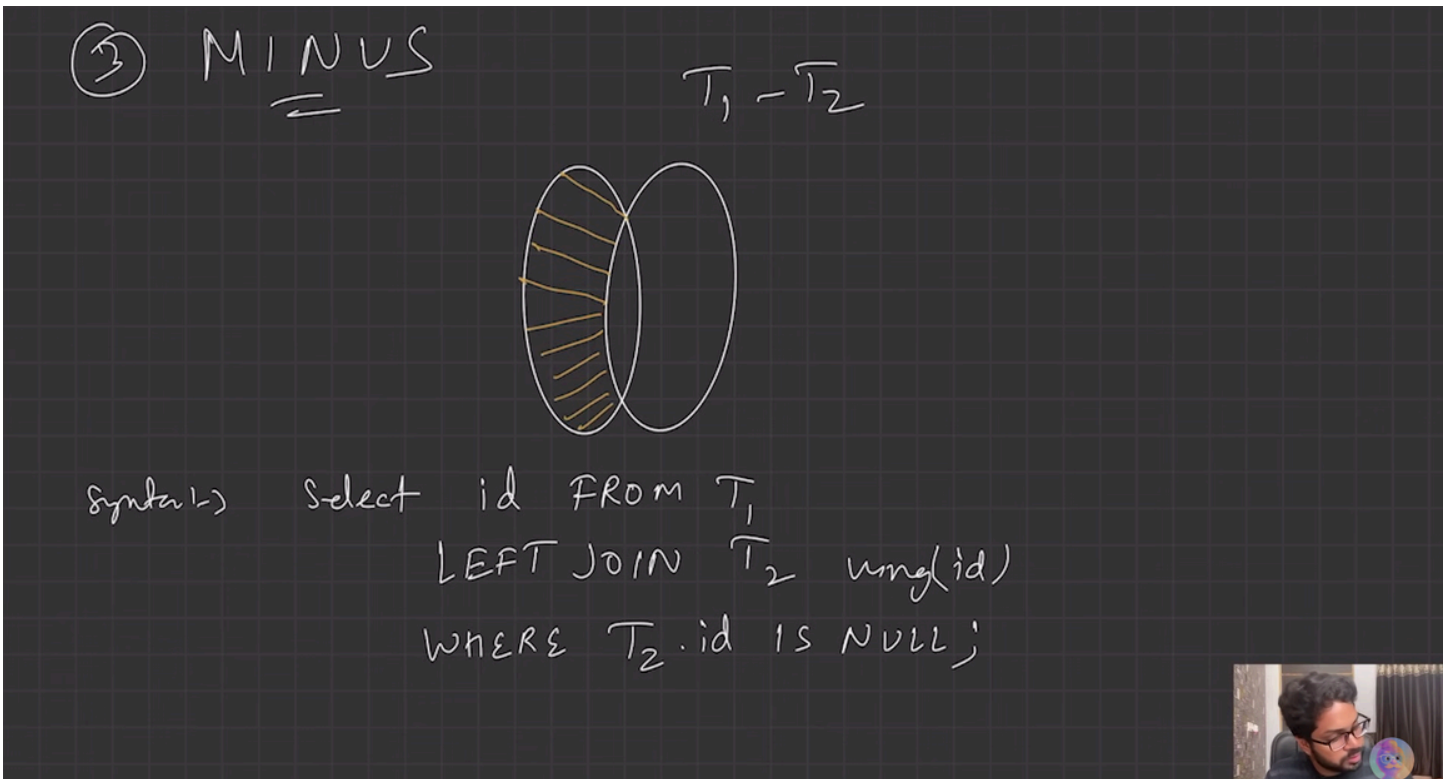
3. **UNION**
    1. Combines two or more SELECT statements.
    2. SELECT * FROM table1
       UNION
       SELECT * FROM table2;
    3. Number of column, order of column must be same for table1 and table2.
4. **INTERSECT**
    1. Returns common values of the tables.
    2. Emulated.
    3. SELECT DISTINCT column-list FROM table-1 INNER JOIN table-2 USING(join_cond);
    4. SELECT DISTINCT * FROM table1 INNER JOIN table2 ON USING(id);

5. MINUS

- **INTERSECT:**



- **MINUS:**

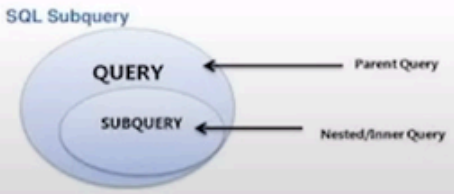


SQL subqueries:

- **Alternate to joins.**
- **Contains outer query and inner query.**
- **Generally outer query depends on inner query.**

SUB QUERIES

1. Outer query depends on inner query.
2. Alternative to joins.
3. Nested queries.
4. SELECT column_list (s) FROM table_name WHERE column_name OPERATOR (SELECT column_list (s) FROM table_name [WHERE]);
5. e.g., SELECT * FROM table1 WHERE col1 IN (SELECT col1 FROM table1);
6. Sub queries exist mainly in 3 clauses
    1. Inside a WHERE clause.



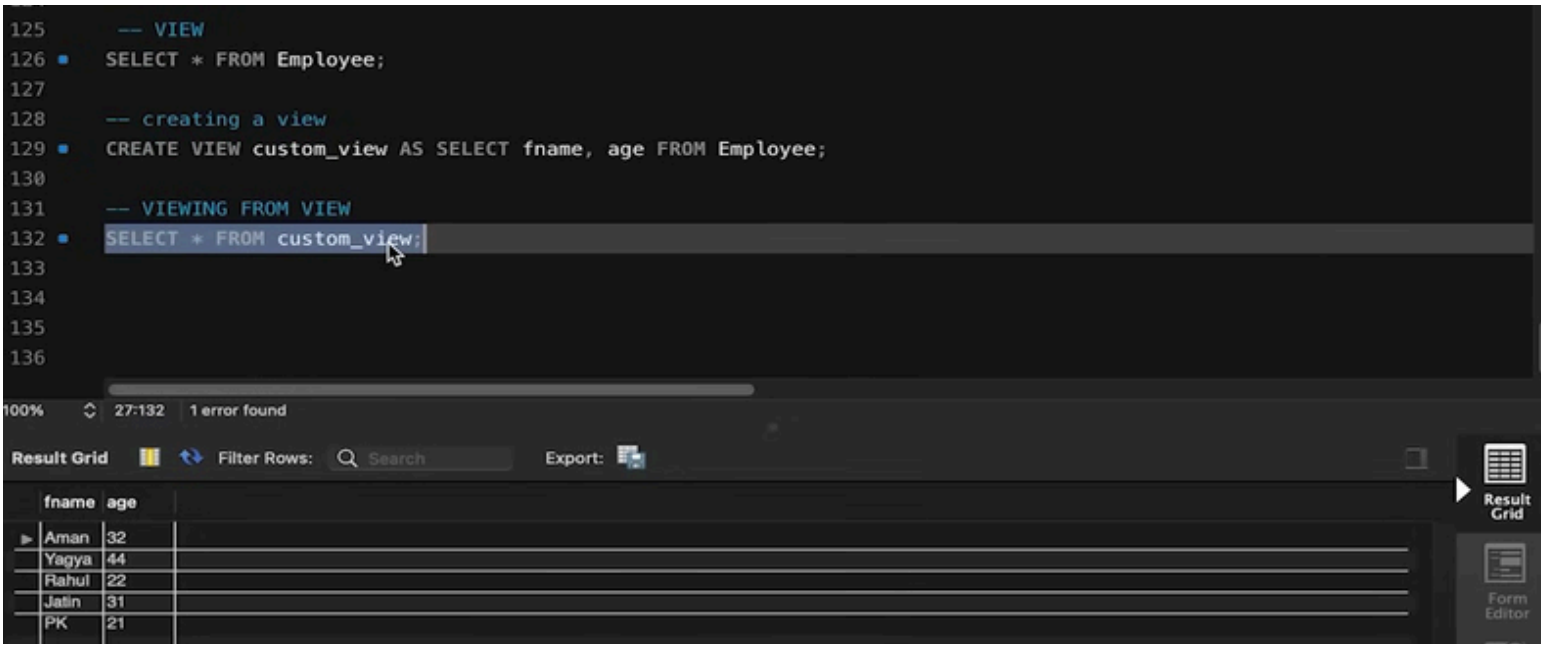
    2. Inside a FROM clause.
    3. Inside a SELECT clause.
7. **Subquery using FROM clause**
    1. SELECT MAX(rating) FROM (SELECT * FROM movie WHERE country = 'India') as temp;
8. **Subquery using SELECT**
    1. SELECT (SELECT column_list(s) FROM T_name WHERE condition), columnList(s) FROM T2_name WHERE condition;
9. **Derived Subquery**
    1. SELECT columnLists(s) FROM (SELECT columnLists(s) FROM table_name WHERE [condition]) as new_table_name;
10. **Co-related sub-queries**
    1. With a normal nested subquery, the inner SELECT query runs first and executes once, returning values to be used by the main query. A correlated subquery, however, executes once for each candidate row considered by the outer query. In other words, the inner query is driven by the outer query.

```
SELECT column1, column2, ....
FROM table1 as outer
WHERE column1 operator
                    (SELECT column1, column2
                     FROM table2
                     WHERE expr1 =
                                 outer.expr2);
```

JOIN VS SUB-QUERIES

JOINS	SUBQUERIES
Faster	Slower
Joins maximise calculation burden on DBMS	Keeps responsibility of calculation on user.
Complex, difficult to understand and implement	Comparatively easy to understand and implement.
Choosing optimal join for optimal use case is difficult	Easy.

SQL VIEWS:


125 -- VIEW
126 SELECT * FROM Employee;
127
128 -- creating a view
129 CREATE VIEW custom_view AS SELECT fname, age FROM Employee;
130
131 -- VIEWING FROM VIEW
132 SELECT * FROM custom_view;
133
134
135
136
100%
27:132
1 error found
Result Grid
Filter Rows:
Search
Export:
fname age
Aman 32
Yagya 44
Rahul 22
Jatin 31
PK 21
Result Grid
Form Editor