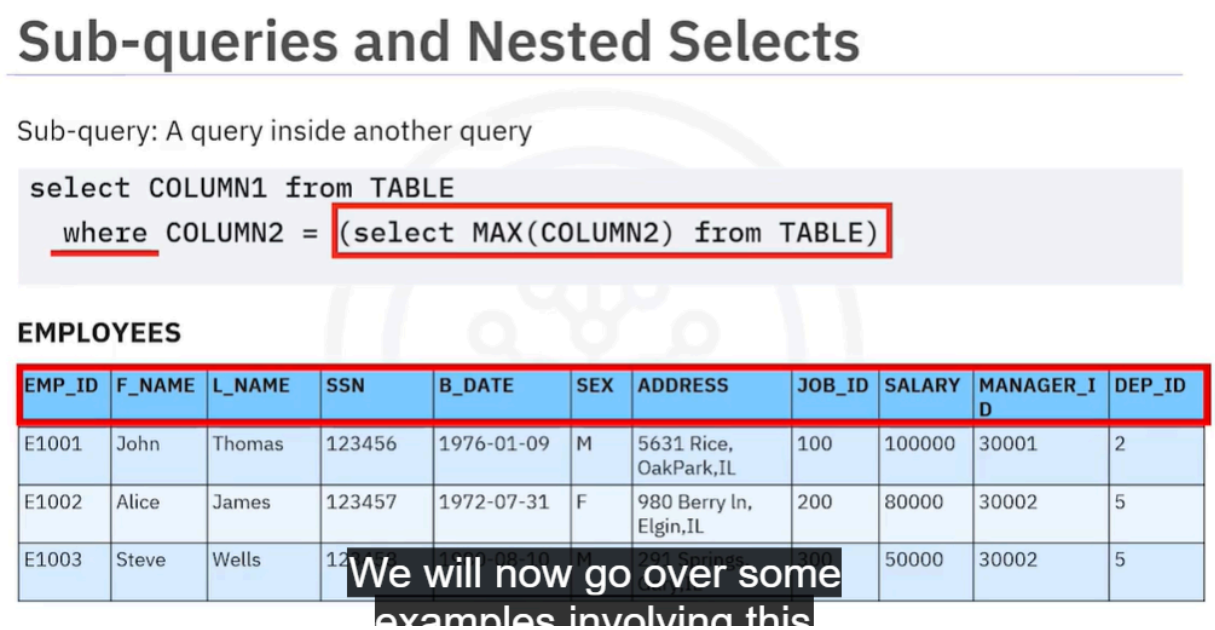
Sub-queries



- Sub queries or sub selects are like regular queries but placed within parentheses and nested inside another query. This allows you to form more powerful queries than would have been otherwise possible. An example of a nested query is shown in this example the sub query is inside the Where clause of another query. Consider the employees table from the previous video. The first few rows of data are shown here. The table contains several columns, including an employee ID first name, last name, salary, etc. We will now go over some examples involving this table.

Why use sub-queries?

To retrieve the list of employees who earn more than the average salary:

```
select * from employees
       where salary > AVG(salary)
```

This query will result in error:

```
SQL0120N Invalid use of an aggregate function or OLAP
function.SQLCODE=-120, SQLSTATE=42903
```

- Let's consider a scenario which may necessitate the use of sub queries. Let's say we want to retrieve the list of employees who earn more than the average salary. To do so we could try this code: select star from employees where salary is greater than average salary. However, running this query will result in an error like the one shown, indicating an invalid use of the aggregate function

Sub-queries to evaluate Aggregate functions

- Cannot evaluate Aggregate functions like AVG() in the WHERE clause –
- Therefore, use a sub-Select expression:

```
select EMP_ID, F_NAME, L_NAME, SALARY
       from employees
       where SALARY <
       (select AVG(SALARY) from employees);
```

Sub-queries to evaluate Aggregate functions

Result:

EMP_ID	F_NAME	L_NAME	SALARY
E1003	Steve	Wells	50000.00
E1004	Santosh	Kumar	60000.00
E1007	Mary	Thomas	65000.00

- One of the limitations of built-in aggregate functions like the average function is that they cannot always be evaluated in the where clause. So, to evaluate a function like average in the Where clause, we could make use of a sub-select expression like the one shown here: select employee ID, first name, last name, salary from employees where salary is less than, open parenthesis, select average salary from employees, close parenthesis. Notice that the average function is evaluated in the first part of the sub query, allowing us to circumvent the limitation of evaluating it directly in the where clause

Sub-queries in list of columns

- Substitute column name with a sub-query
- Called Column Expressions

```
select EMP_ID, SALARY, AVG(SALARY) AS AVG_SALARY
       from employees
```

Sub-queries in list of columns

- Substitute column name with a sub-query
- Called Column Expressions

```
select EMP_ID, SALARY, AVG(SALARY) AS AVG_SALARY
       from employees ;
```

```
select EMP_ID, SALARY,
       ( select AVG(SALARY) from employees )
                AS AVG_SALARY
       from employees ;
```

For example select employee ID

Sub-queries in list of columns

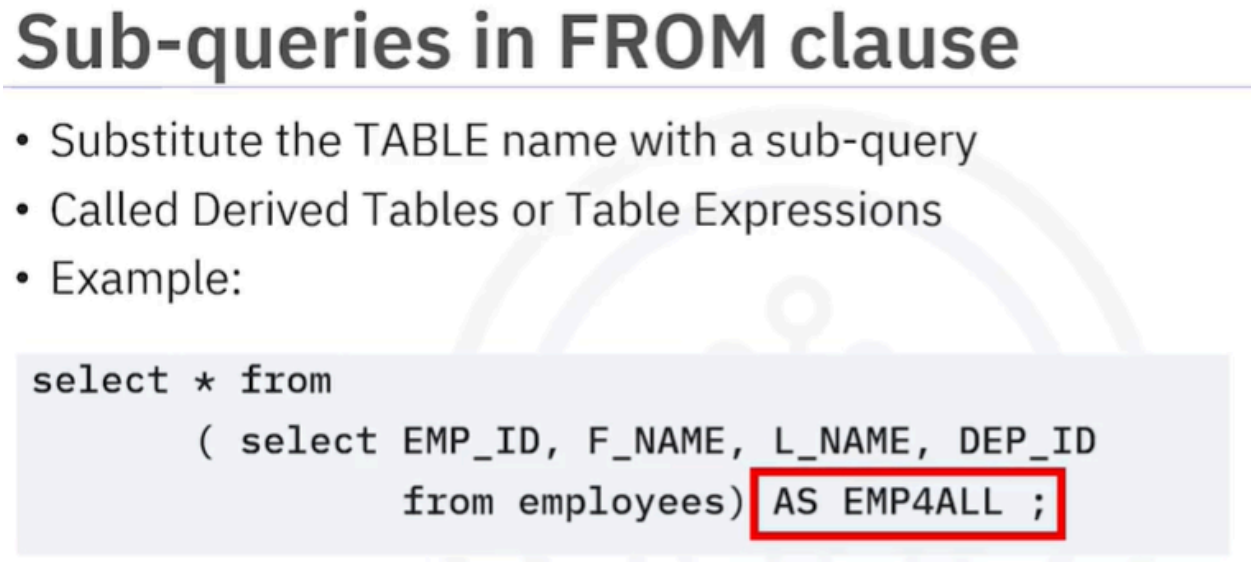
Result:

EMP_ID	SALARY	AVG_SALARY
E1002	80000.00	68888.888888888888888888888888
E1003	50000.00	68888.888888888888888888888888
E1004	60000.00	68888.888888888888888888888888
E1005	70000.00	68888.888888888888888888888888
E1006	90000.00	68888.888888888888888888888888
E1007	65000.00	68888.888888888888888888888888
E1008	65000.00	68888.888888888888888888888888
E1009	70000.00	68888.888888888888888888888888
E1010	[illegible]	[illegible]888888888888888888

parenthesis, as average salary from employees

- The subselect doesn't just have to go in the WHERE clause, it can also go in other parts of the query such as in the list of columns to be selected. Such sub queries are called Column Expressions. Now let's look at a scenario where we might want to use a Column Expression. Say we wanted to compare the salary

of each employee with the average salary. We could try a query like: select employee ID, salary, average salary, as average salary from employees. Running this query will result in an error, indicating that no Group by Clause is specified. We can circumvent this error by using the average function in a sub query placed in the list of the columns. For example select employee ID, salary, open left parenthesis, select average salary from employees, close right parenthesis, as average salary from employees.



Sub-queries in FROM clause

Result:

EMP_ID	F_NAME	L_NAME	DEP_ID
E1002	Alice	James	5
E1003	Steve	Wells	5
E1004	Santosh	Kumar	5
E1005	Ahmed	Hussain	2
E1006	Nancy	Allen	2
E1007	Mary	Thomas	7
E1008	Bharath	Gupta	7
E1009	Andrea	Jones	7
E1010	Ann	Jacob	5

- Another option is to make the sub query be part of the From clause. Sub queries like these are sometimes called Derived Tables or Table Expressions because the outer query uses the results of the sub query as a data source. Let's look at an example to create a table expression that contains non-sensitive employee information select star from select employee ID, first name, last name, Department ID from employees, as employee for all. The derived table in the sub query does not include sensitive fields like date of birth or salary. This example is a trivial one and we could just as easily have included the columns in the outer query, however such derived tables can prove to be powerful in more complex situations, such as when working with multiple tables and doing joins.