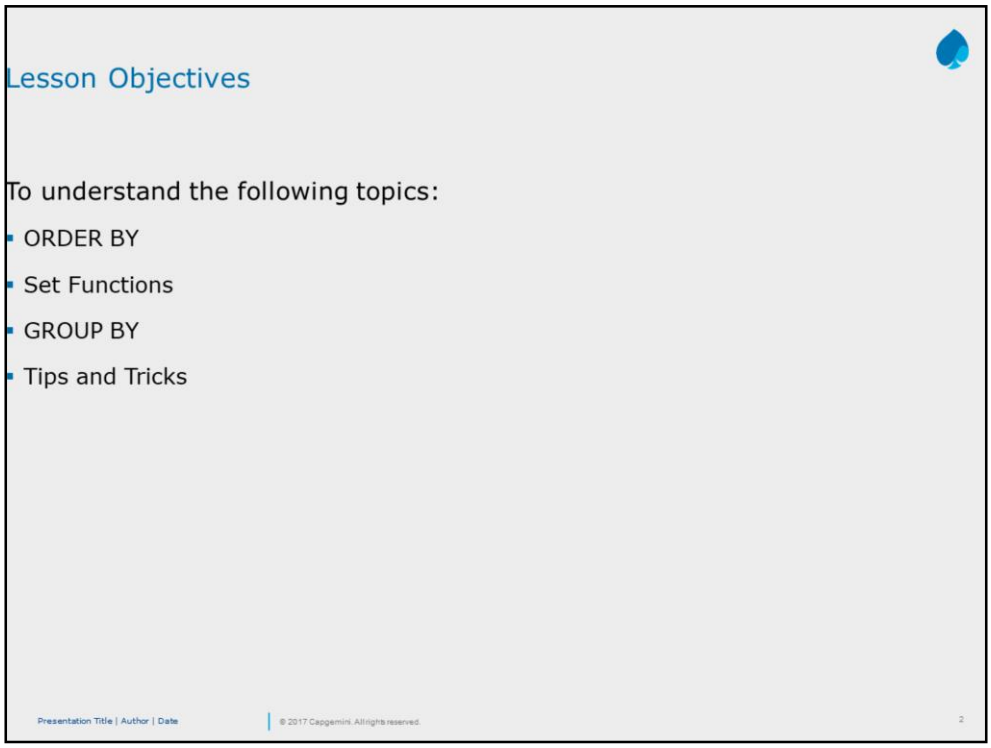
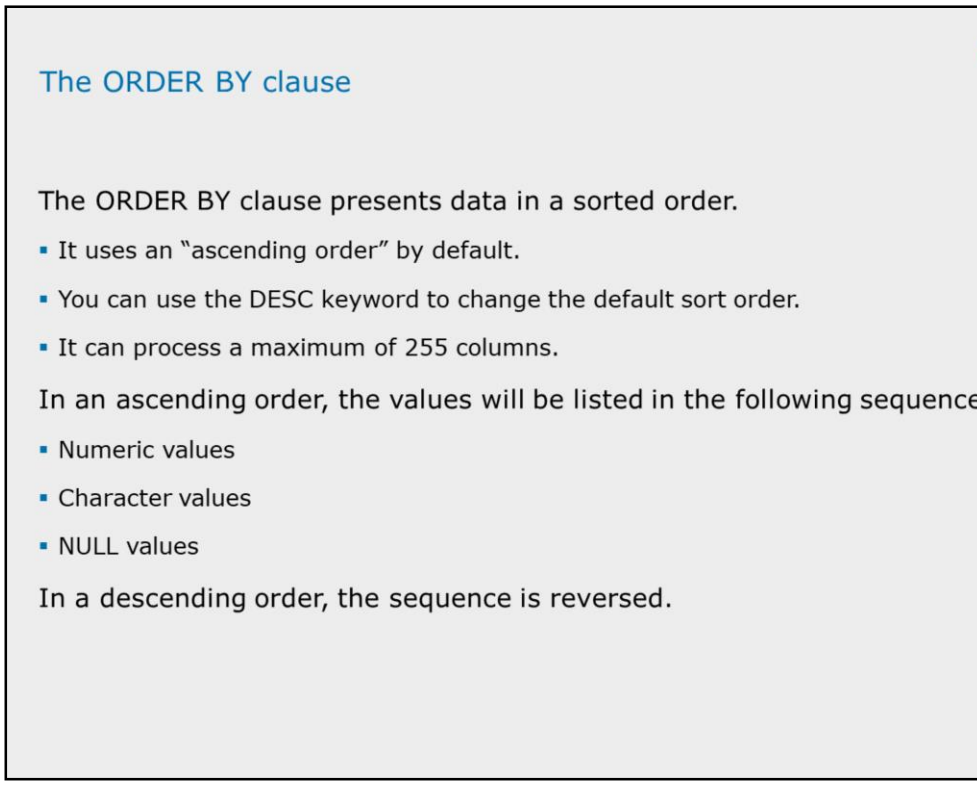
Instructor Notes:

Instructor Notes:

None



Instructor Notes:



The Order By Clause:

- A query with its various clauses (FROM, WHERE, GROUP BY, HAVIN determines the rows to be selected and the columns. The order of rows not fixed unless an ORDER BY clause is given.
- An ORDER BY clause is of the form:

```
ORDER BY < Sort  list> ASC/DESC
```

- The columns to be used for ordering are specified by using the "column names" or by specifying the "serial number" of the column in the SELE list.
- The sort is done on the column in "ascending" or "descending" order. B default the ordering of data is "ascending" order.

contd.

Instructor Notes:

Sorting Data using ORDER BY Clause

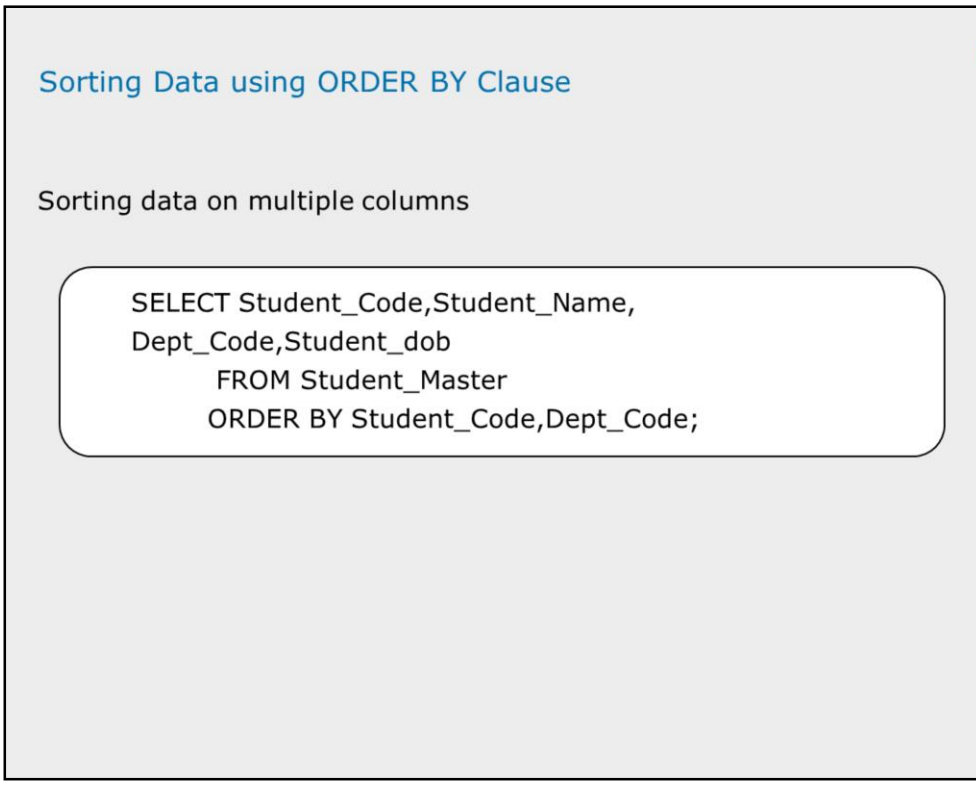
The output of the SELECT statement can be sorted using ORDER BY clau

- ASC : Ascending order, default
- DESC : Descending order

Display student details from student_master table sorted on student_co in descending order.

```
SELECT Student_Code,Student_Name,Dept_Code,
Student_dob
      FROM Student_Master
     ORDER BY Student_Code DESC ;
```

Instructor Notes:



The query on the slide sorts the data on both the columns in ascending order whic is default. But you could also sort the data in different order for the columns.

For Example in the query given below the data is sorted in ascending order on student_code and dept_code is sorted in descending order

```
SELECT
Student_Code,Student_Name,Dept_Code,Student_dob
  FROM Student_Master
 ORDER BY Student_Code,Dept_Code DESC;
```

Instructor Notes:

None

Set Operators:

There are situations when we need to “combine the results” from two or more SELECT statements. SQL enables us to handle these requirements by using “Set operations”.

The result of each SELECT statement can be treated as a set. SQL set operations can be applied on these sets to arrive at a final result.

SQL supports the following four Set operations:

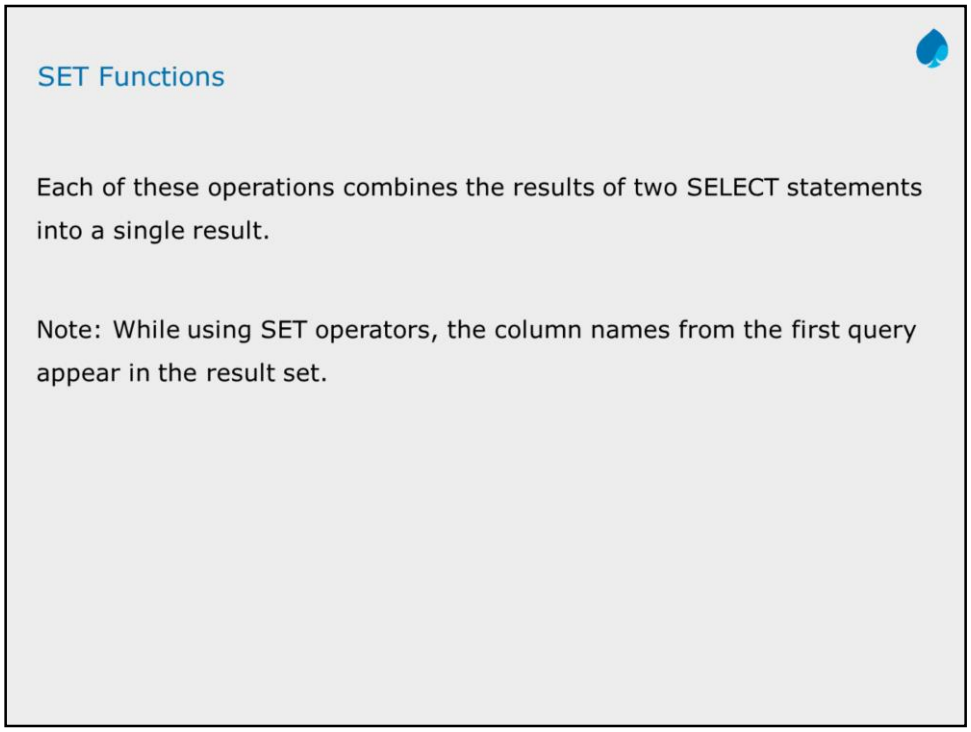
- UNION ALL
- UNION
- MINUS
- INTERSECT

All set operators have equal precedence.

If a SQL statement contains multiple set operators, Oracle evaluates them from the left to right if there are no parentheses explicitly specifying another order.

contd.

Instructor Notes:



SQL statements containing the Set operators are referred to as “compound queries”. Each SELECT statement in a compound query is referred to as a “component query”.

Two SELECT statements can be combined into a compound query by a set operation only if they satisfy the following two conditions:

The “result sets” of both the queries must have the “same number of columns”.

The “datatype” of each column in the “second result set” must match the “datatype” of its corresponding column in the “first result set”.

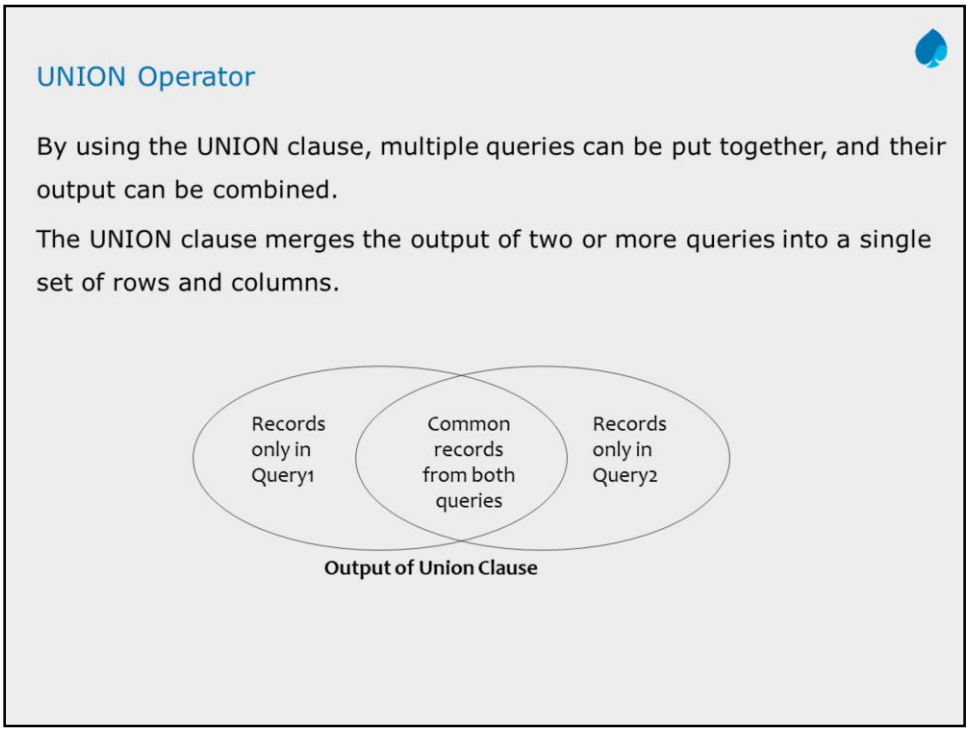
For example: If component queries select character data, then the datatype of the return values are determined as follows:

If both queries select values of datatype CHAR, then the returned values have datatype CHAR.

If either or both of the queries select values of datatype VARCHAR2, then the returned values have datatype VARCHAR2.

Tip: The datatypes do not need to be the same, if those in the second result set can be automatically converted by Oracle (using implicit casting) to types that are compatible with those in the first result set.

Instructor Notes:



UNION Operator:

The UNION operator returns the records retrieved by either of the queries.

- By default, the UNION operator eliminates duplicate records.
- If however we want to retain duplicates, we use UNION ALL instead of UNION.

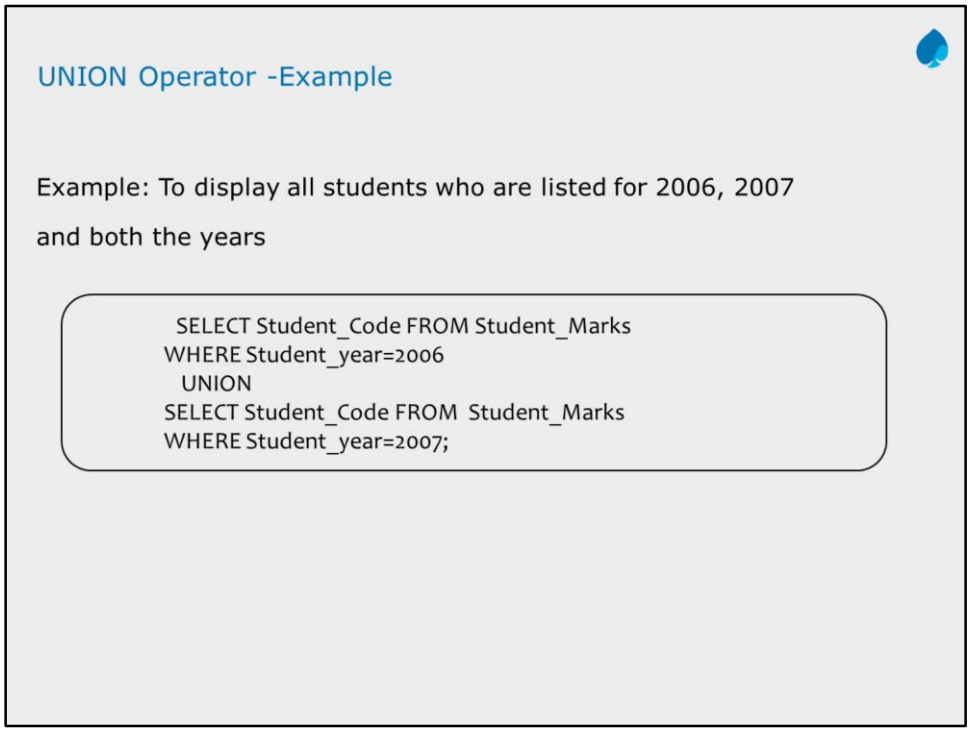
UNION operates over all of the columns being selected.

NULL values are not ignored during duplicate checking.

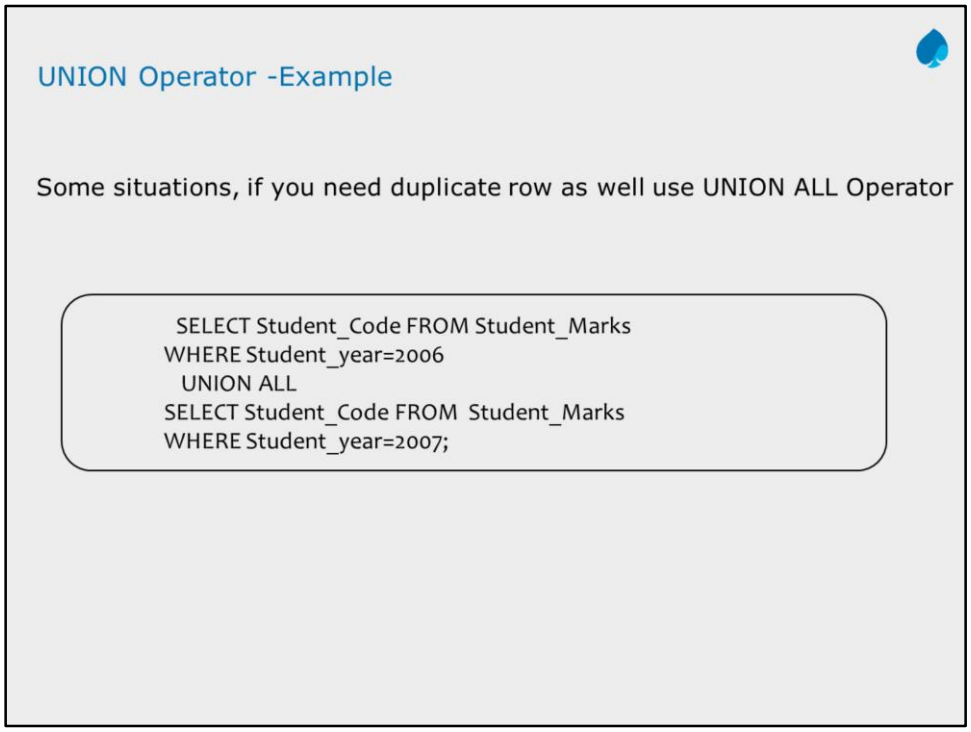
The IN operator has a higher precedence than the UNION operator.

By default, the output is sorted in ascending order of the first column of the SELECT clause.

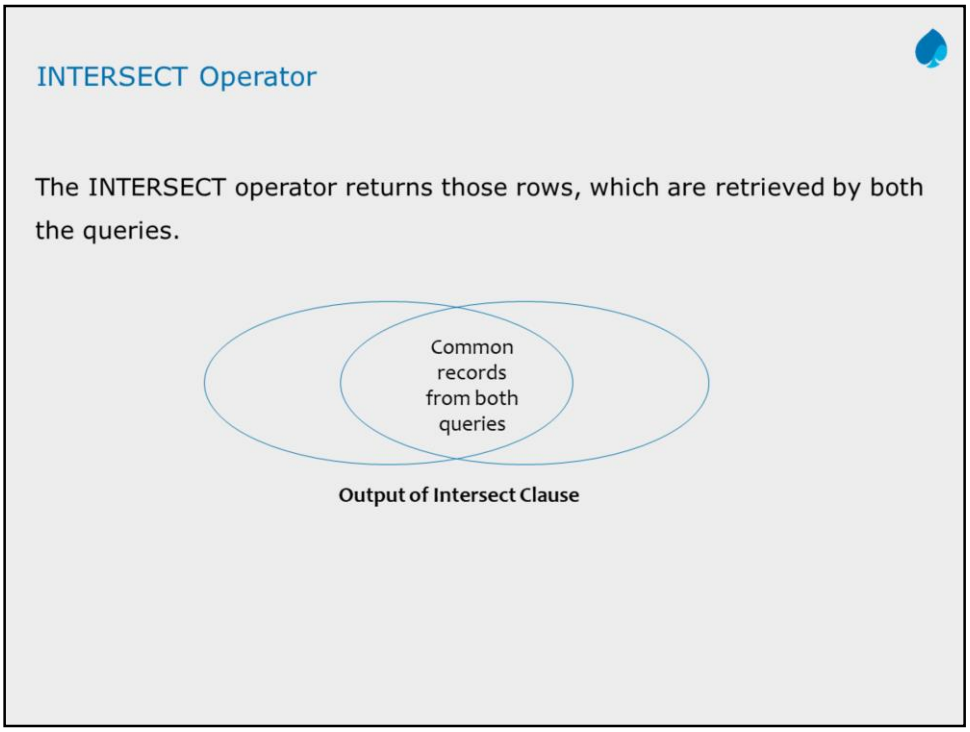
Instructor Notes:



Instructor Notes:



Instructor Notes:

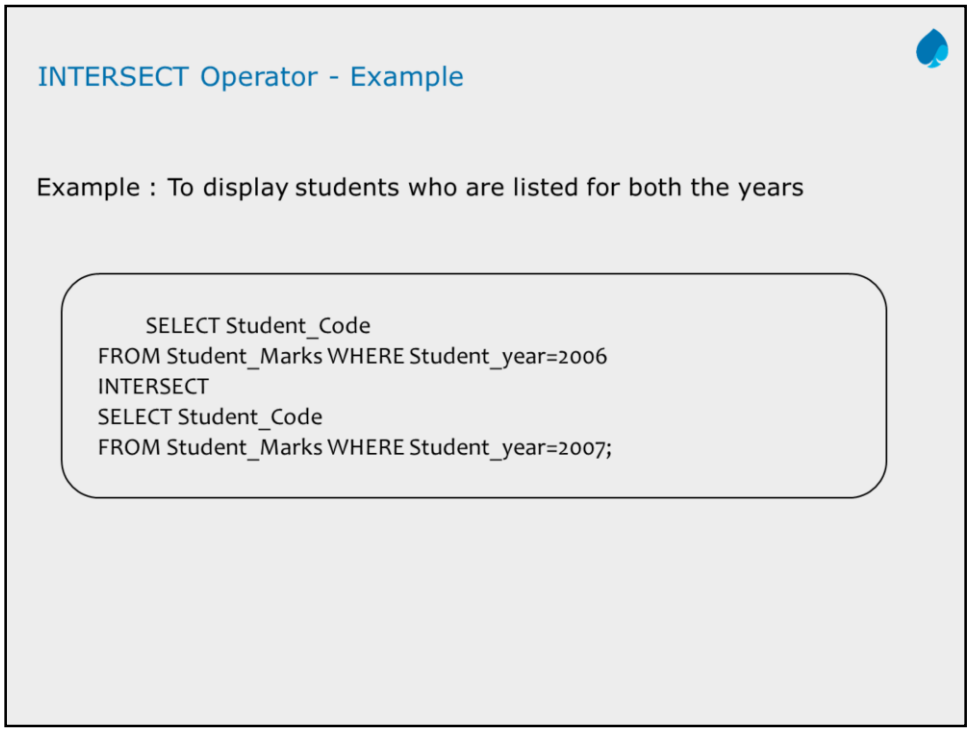


INTERSECT Operator

INTERSECT result is same on reversing the order

INTERSECT does not ignore NULL values

Instructor Notes:



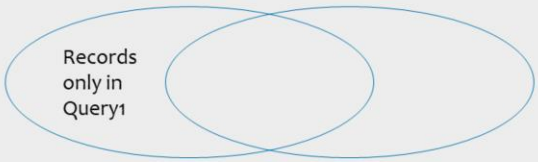
Example of INTERSECT operator:

The example on the slide will only display the common records retrieved by both the queries

Instructor Notes:

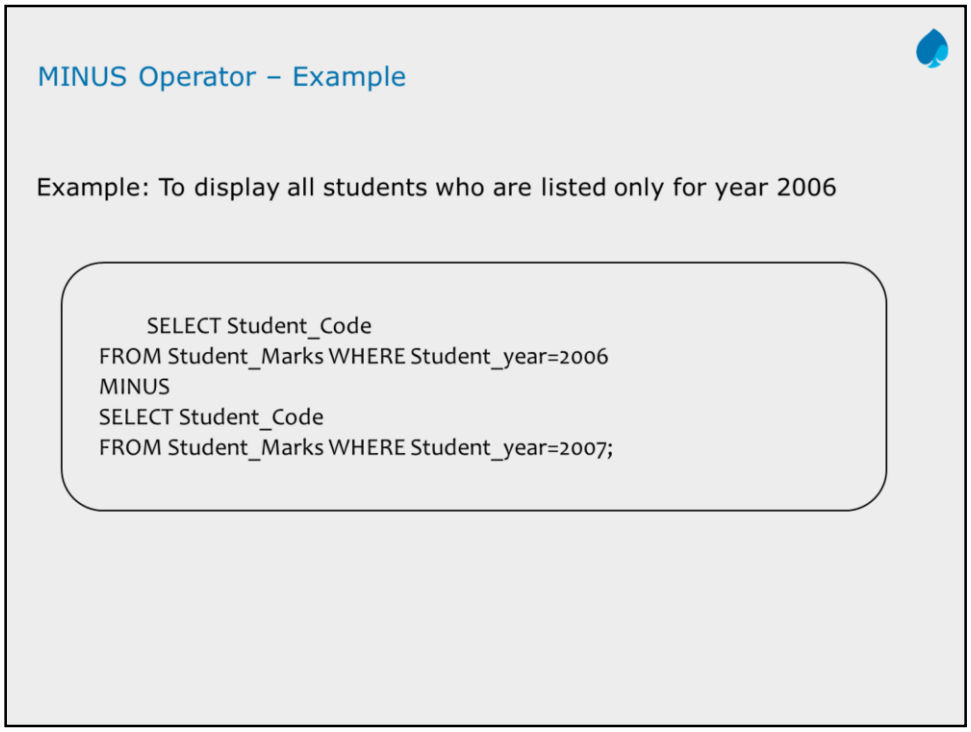
MINUS Operator

The MINUS operator returns all rows retrieved by the first query but not by the second query.



Output of Minus Clause

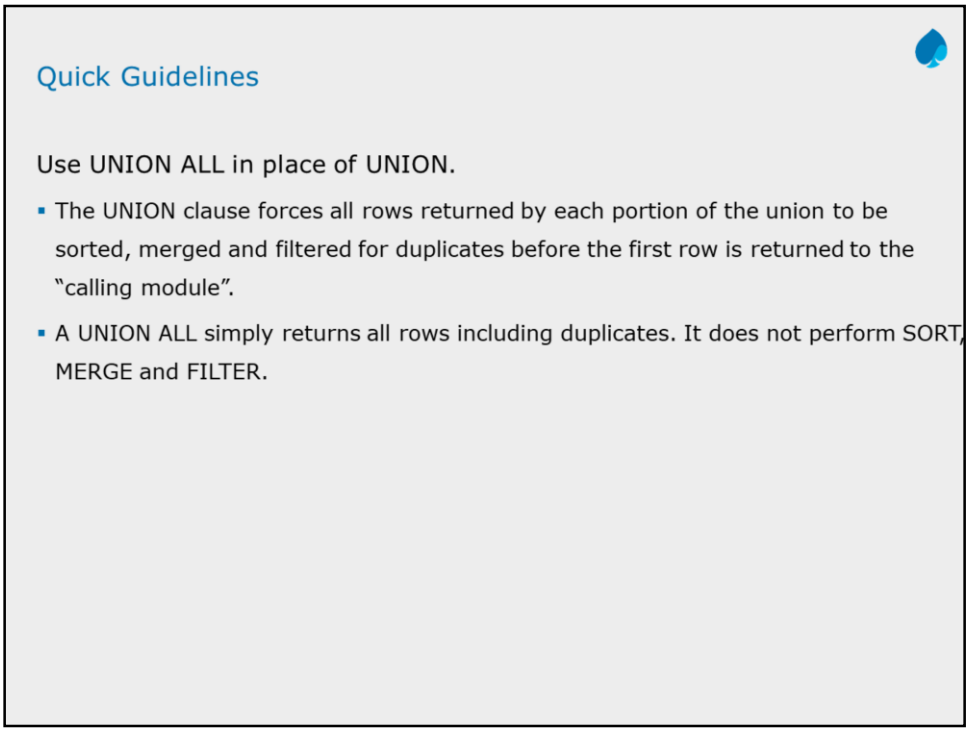
Instructor Notes:



Example of MINUS operator:

The query on the slide will show results which are unique to first query

Instructor Notes:



Tips and Tricks:

When using the UNION statement, keep in mind, that the UNION statement performs the equivalent of a SELECT DISTINCT on the final result set, by default.

In other words, UNION takes the results of two like record sets, combines them, and then performs a SELECT DISTINCT in order to eliminate any duplicate rows.

> This process occurs even if there are no duplicate records in the final record set.

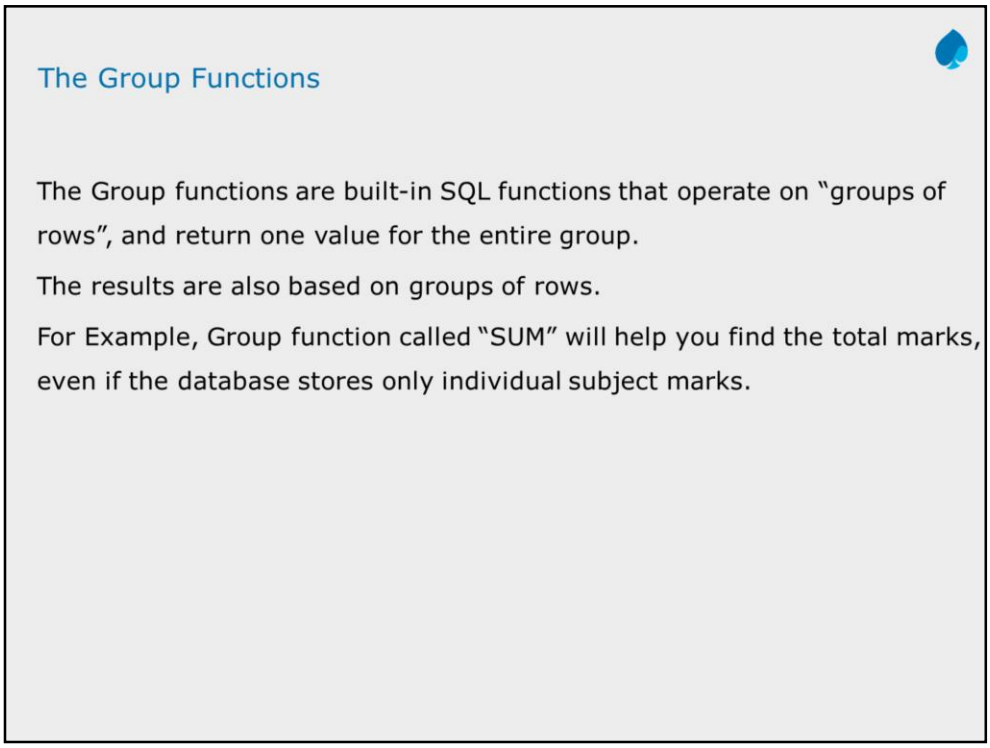
Hence,

> If you know that there are duplicate records, and it creates a problem for your application, then use the UNION statement "to eliminate the duplicate rows".
>
> If you know that there will never be any duplicate rows, or if there are duplicates, and it does not create problems in your application, then you should use the UNION ALL statement instead of the UNION statement.

The advantage of the UNION ALL statement is that is does not perform the SELECT DISTINCT function. This saves a lot of server resources from being unnecessarily used.

Instructor Notes:

None

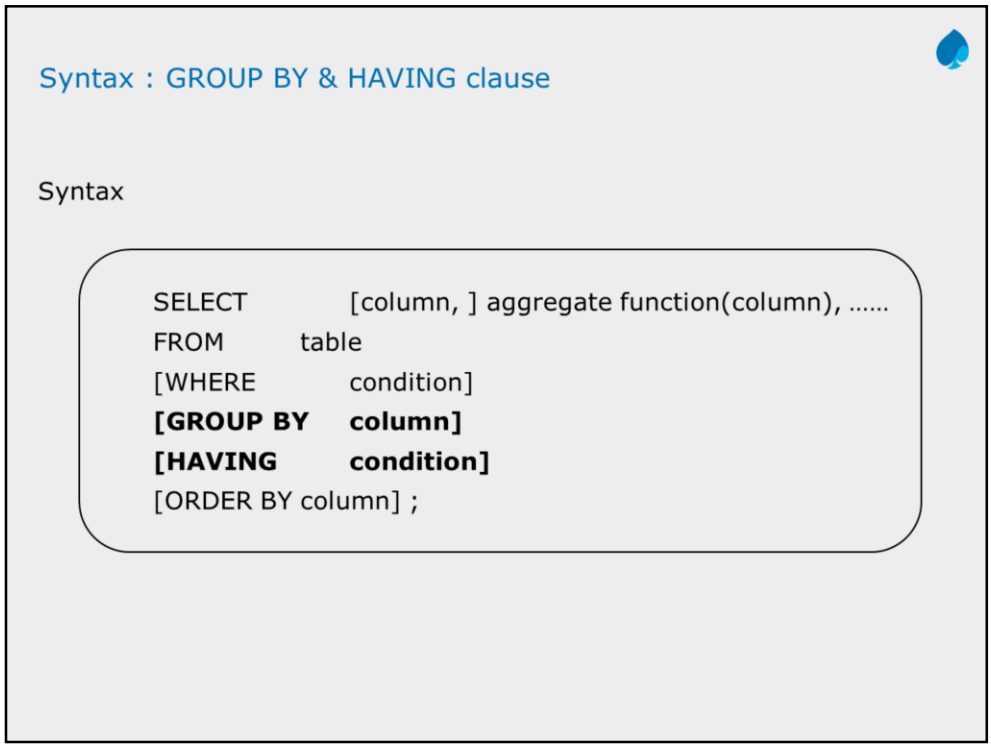


Aggregate (Group) Functions:

- SQL provides a set of "built-in" functions for producing a single value for an entire group. These functions are called as "Set functions" or "Aggregate (Group) functions".
- These functions can work on a "normal result table" or a "grouped result table".
  - If the result is not grouped, then the aggregate will be taken for the whole result table.

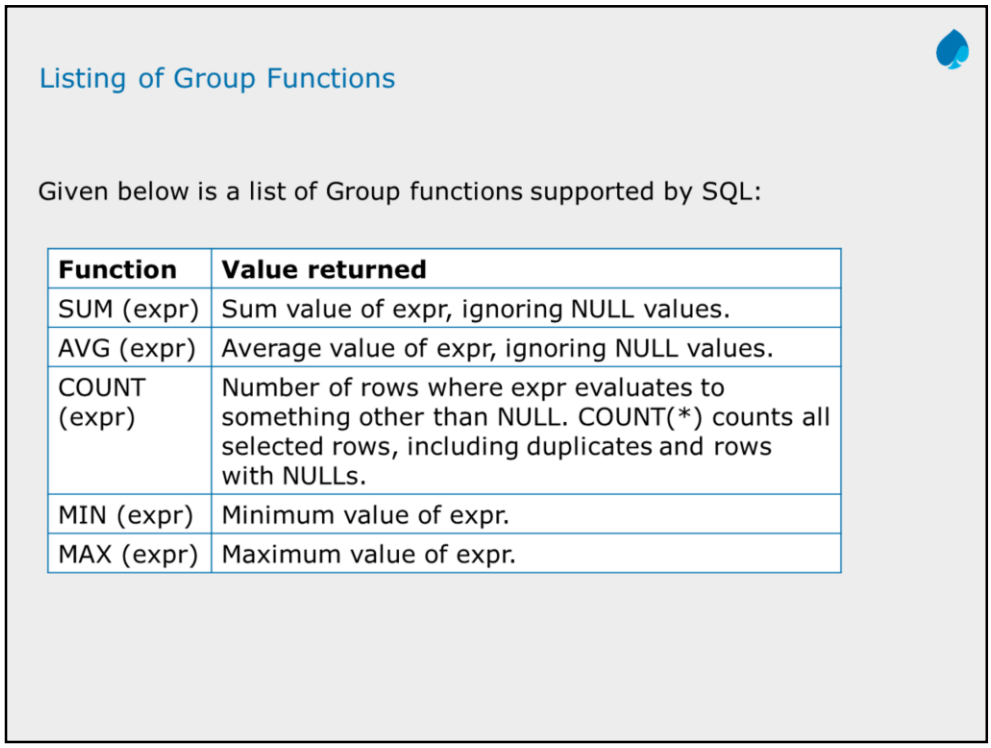
Instructor Notes:

None



Instructor Notes:

None



Aggregate (Group) Functions supported by SQL:

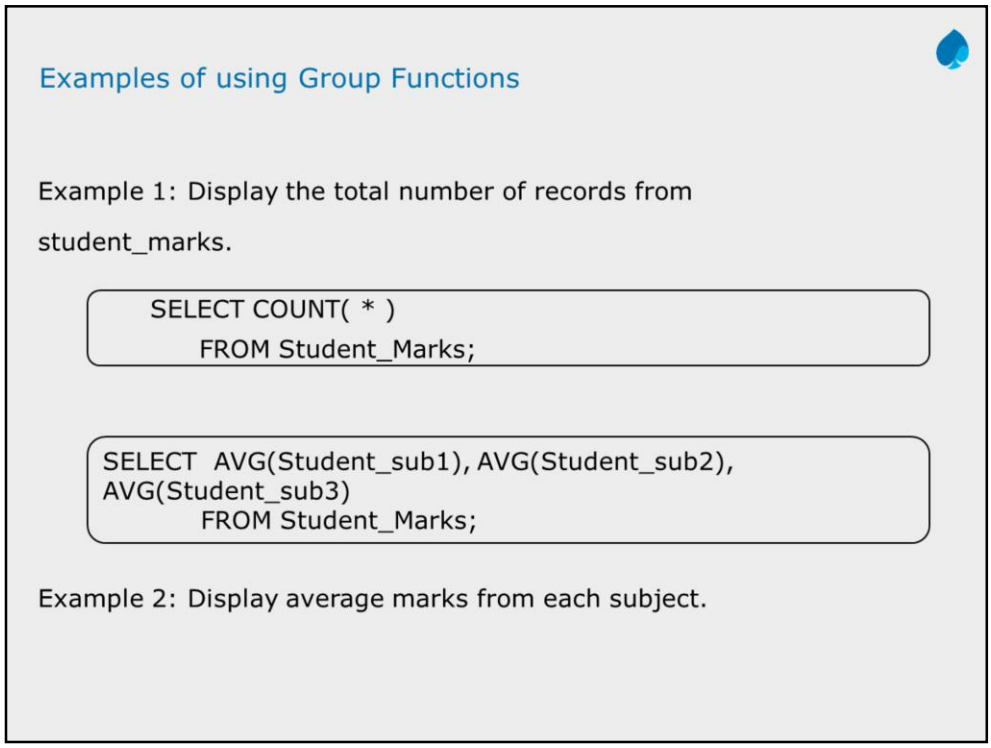
- All the above functions operate on a number of rows (for example, an entire table), and are therefore known as “Group (or Aggregate) functions”.
- A Group function can be used on a subset of the rows in a table by using the WHERE clause.
- The Aggregate functions ignore NULL values in the column.
  - To include NULL values, NVL function can be used with Aggregate functions.

Note :

- **Count(*)**: Returns the number of rows in the table, including duplicates and those with NULLs.
- **Count(<Expression>)**: Returns the number of rows where expression is NOT NULL.

Instructor Notes:

None

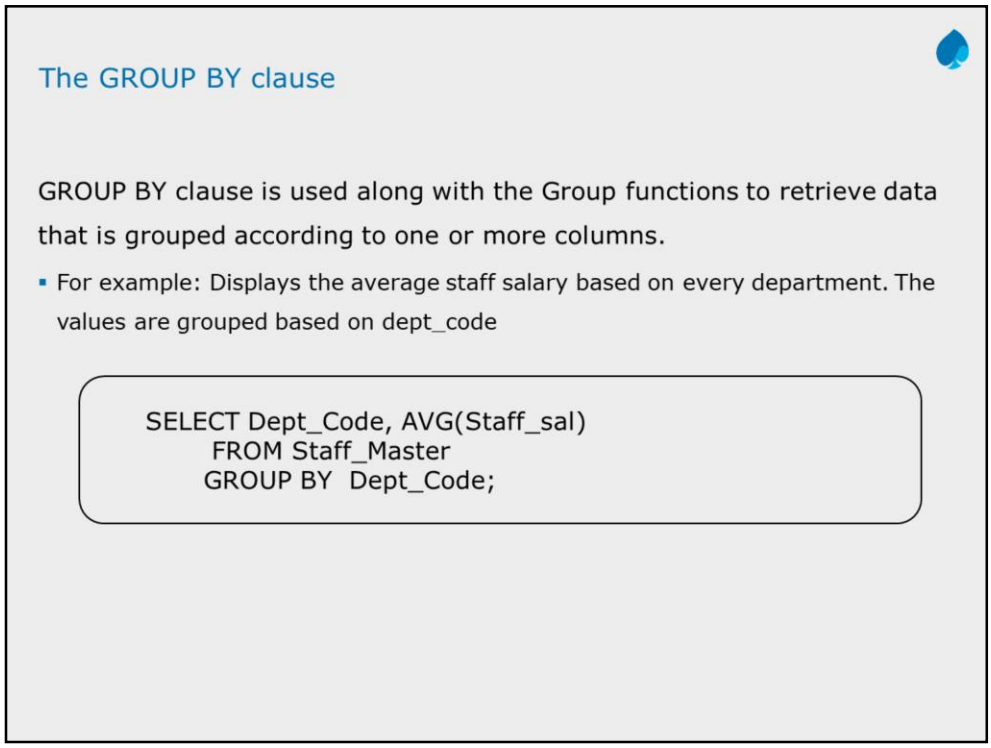


Note:

- The first query returns the value, which counts the number of rows fetched from the student_marks table. All the rows in thetable are treated as one group.Group Functions operate on sets of rows to give one result per group. Can be used on the whole table or certain set of rows

Instructor Notes:

None

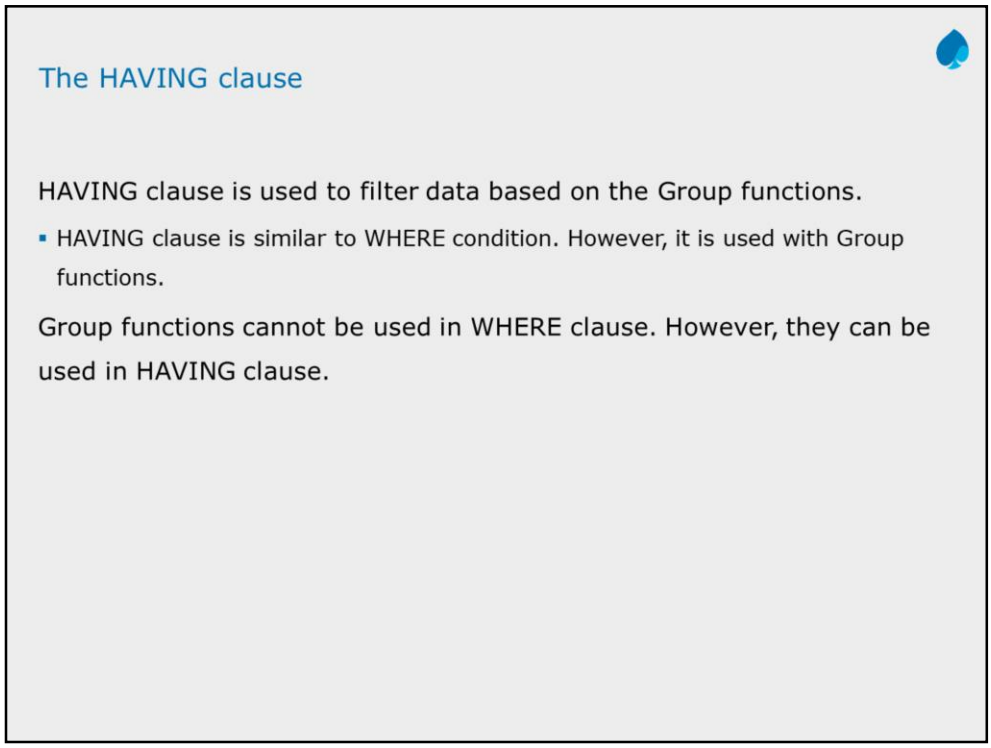


Usage of GROUP BY and HAVING clauses:

- All the SELECT statements we have used until now have acted on data as if the data is in a "single group". But the rows of data in some of the tables can be thought of as being part of "different groups".
  For example: The staff_master table contains staff information allocated to various departments identified by dept_code. If we wish to find the minimum salary of each group of employees in respective department, then none of the clauses that we have seen until now are of any use.
- The GROUP BY clause is used to group the result table derived from earlier FROM and WHERE clauses. Further, a HAVING clause is used to apply search condition on these groups.
  - When a GROUP BY clause is used, each row of the resulting table will represent a group having same values in the column(s) used for grouping.
  - Subsequently, the HAVING clause acts on the resulting grouped table to remove the row that does not satisfy the criteria in the HAVING search condition

Instructor Notes:

None



The HAVING Clause:

- A HAVING clause is of the form:

  HAVING <search condition>

- The HAVING search condition applies to "each group". It can be:
  - formed using various predicates like between, in, like, null, comparison, etc
  - combined with Boolean operators like AND, OR, NOT
- Since the search condition is for a "grouped table". The predicates should be:
  - on a column by which grouping is done.
  - on a set function (Aggregate function) on other columns.
- The aggregate functions can be used in HAVING clause. However, they cannot be used in the WHERE clause.
- When WHERE, GROUP BY, and HAVING clauses are used together in a SELECT statement:
  1. The WHERE clause is processed first in order.
  2. Subsequently, the rows that are returned after the WHERE clause is executed are grouped based on the GROUP BY clause.
  3. Finally, any conditions, on the Group functions in the HAVING clause, are applied to the grouped rows before the final output is displayed.

Instructor Notes:

None
Order of Clause Evaluation:
When included in the same SELECT statement, evaluated in order of:
•WHERE
•GROUP BY
•HAVING

Nesting Functions:
•Inner functions are resolved first

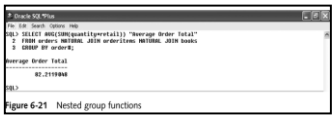


Figure 6-21 Nested group functions

Examples – GROUP BY and HAVING clause

For example: Display all department numbers having more than five employees.

```
SELECT Department_Code, Count(*)
        FROM Staff_Master
        GROUP BY Department_Code
        HAVING Count(*)> 5;
```

The HAVING clause (contd.):

- To find out Average, Maximum, Minimum salary of departments, where average salary is greater than 2000.

```
SELECT dept_code,AVG(staff_sal),MIN(staff_sal),
MAX(staff_sal)
FROM staff_master
GROUP BY dept_code HAVING AVG(staff_sal) >
2000;
```

- To find out average salary of all staff members who belong to department 10 or 20 and their average salary is greater than 10000

```
SELECT design_code,dept_code,avg(staff_sal)
FROM staff_master where dept_code in(10,20)
GROUP BY  design_code,dept_code
HAVING avg(staff_sal) >10000;
```

Instructor Notes:

Quick Guidelines

All group functions except COUNT(*) ignores NULL values.

To substitute a value for NULL values use NVL functions.

DISTINCT clause makes the function consider only non duplicate values.

The AVG and SUM are used with numerica data.

The MIN and MAX functions used with any data type.

NVL function is covered in the next lesson

Instructor Notes:

Quick Guidelines

All individual columns included in the SELECT clause other than group functions must be specified in the GROUP BY clause.

Any column other than selected column can also be placed in GROUP BY clause.

By default rows are sorted by ascending order of the column included in the GROUP BY list.

WHERE clause specifies the rows to be considered for grouping.

NVL function is covered in the next lesson

Instructor Notes:

None

Quick Guidelines

Suppose your SELECT statement contains a HAVING clause. Then write your query such that the WHERE clause does most of the work (removing undesired rows) instead of the HAVING clause doing the work of removing undesired rows.

Use the GROUP BY clause only with an Aggregate function, and not otherwise.

- Since in other cases, you can accomplish the same end result by using the DISTINCT option instead, and it is faster.

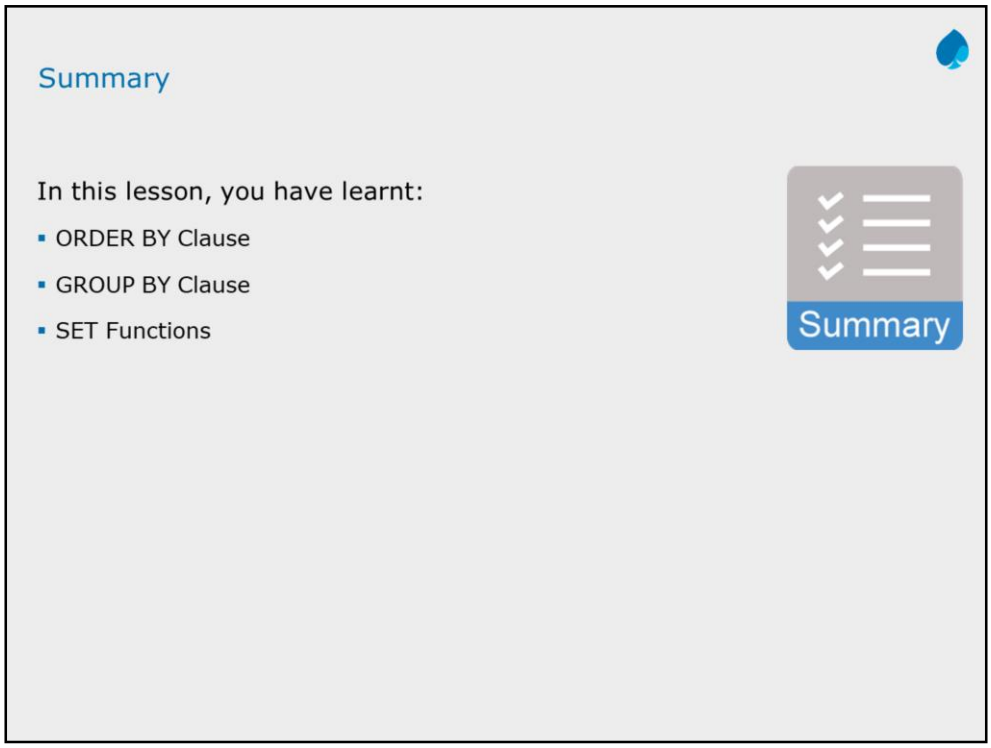
Tips and Tricks:

- By appropriately using the WHERE clause, you can eliminate unnecessary rows before they reach the GROUP BY and HAVING clause. Thus saving some unnecessary work, and boosting performance.
  For example: In a SELECT statement with WHERE, GROUP BY, and HAVING clauses, the query executes in the following sequence.
  - First, the WHERE clause is used to select the appropriate rows that need to be grouped.
  - Next, the GROUP BY clause divides the rows into sets of grouped rows, and then aggregates their values.
  - And last, the HAVING clause then eliminates undesired aggregated groups.
    - If the WHERE clause is used to eliminate as many of the undesired rows as possible, then the GROUP BY and the HAVING clauses will have to do less work. Thus boosting the overall performance of the query.

contd.

Instructor Notes:

None



Instructor Notes:

Answers for Match the Following:

1 - b
2 - a
3 - d
4 - c

Review Questions

Question 1: The Set operation that will show all the rows from both the resultsets including duplicates is ___.

Option 1: Union All
Option 2: Union
Option 3: Intersect
Option 4: Minus

Question 2: The Intersect operator returns ___.

Instructor Notes:

Answers for Review Questions:

Question 1:
Answers: Option 2, 3

Review – Questions

Question 1: Identify the various group functions from the list given below:

- Option 1: maximum
- Option 2: sum
- Option 3: count
- Option 4: minimum

Instructor Notes:

Answers for Review Questions:

Question 2:
Answer: True

Question 3:
Answer: True

Review – Questions

Question 2: The AVG function ignores NULL values in the column.

- True / False

Question 3: Count(*) returns the number of rows in the table, including duplicates and those with NULLs.

- True / False