Database Design

Anatomy of a SQL Statement

ORACLE® ACADEMY

Objectives

This lesson covers the following objectives:

- Match projection, selection, and join with their correct functions/ capabilities
- Create a basic SELECT statement
- Use the correct syntax to display all rows in a table
- Use the correct syntax to select specific columns in a table, modify the way data is displayed, and perform calculations using arithmetic expressions and operators

Objectives (cont.)

This lesson covers the following objectives:

- Formulate queries using correct operator precedence to display desired results
- Define a null value
- Demonstrate the effect null values create in arithmetic expressions
- Construct a query using a column alias

SELECT Keyword

SELECT is one of the most important, if not the most important keyword in SQL. You use SELECT to retrieve information from the database. When you learn how to use SELECT, you've opened the door to the database.

Imagine a database containing information about movies such as title, genre, studio, producer, release date, series, country, language, rating, running time, and so on. What if you only wanted the titles of movies created in India? The SELECT statement allows you to search for specific data.

SELECT Statement

The SELECT statement retrieves information from the database. The syntax for a SELECT statement is as follows:

```
SELECT <column_name(s)>
FROM <table_name>;
```

In its simplest form, a SELECT statement must include the following:

- A SELECT clause, which specifies the columns to be displayed
- A FROM clause, which specifies the table containing the columns listed in the SELECT clause

Conventions

```
SELECT title

FROM d_songs;
```

Throughout this course, the following will be used:

- A keyword refers to an individual SQL command. For example, SELECT and FROM are keywords.
- A clause is a part of a SQL statement. For example, SELECT title is a clause.
- A statement is a combination of two or more clauses. For example, SELECT title FROM d_songs is a statement.

Capabilities of SELECT Statements

Projection: Used to choose columns in a table

Selection: Used to choose rows in a table

Table 2: Projection

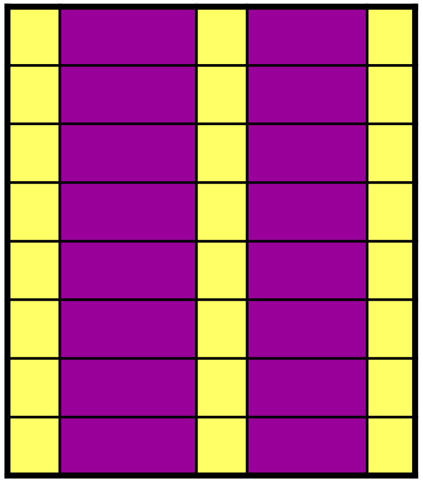
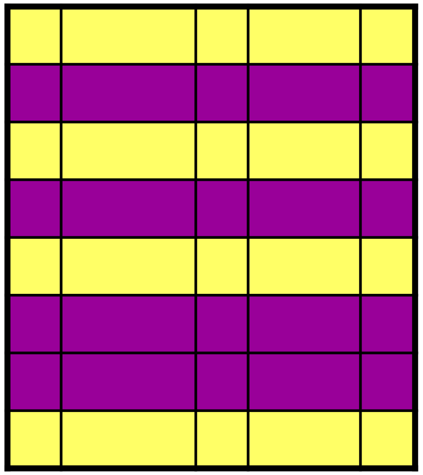
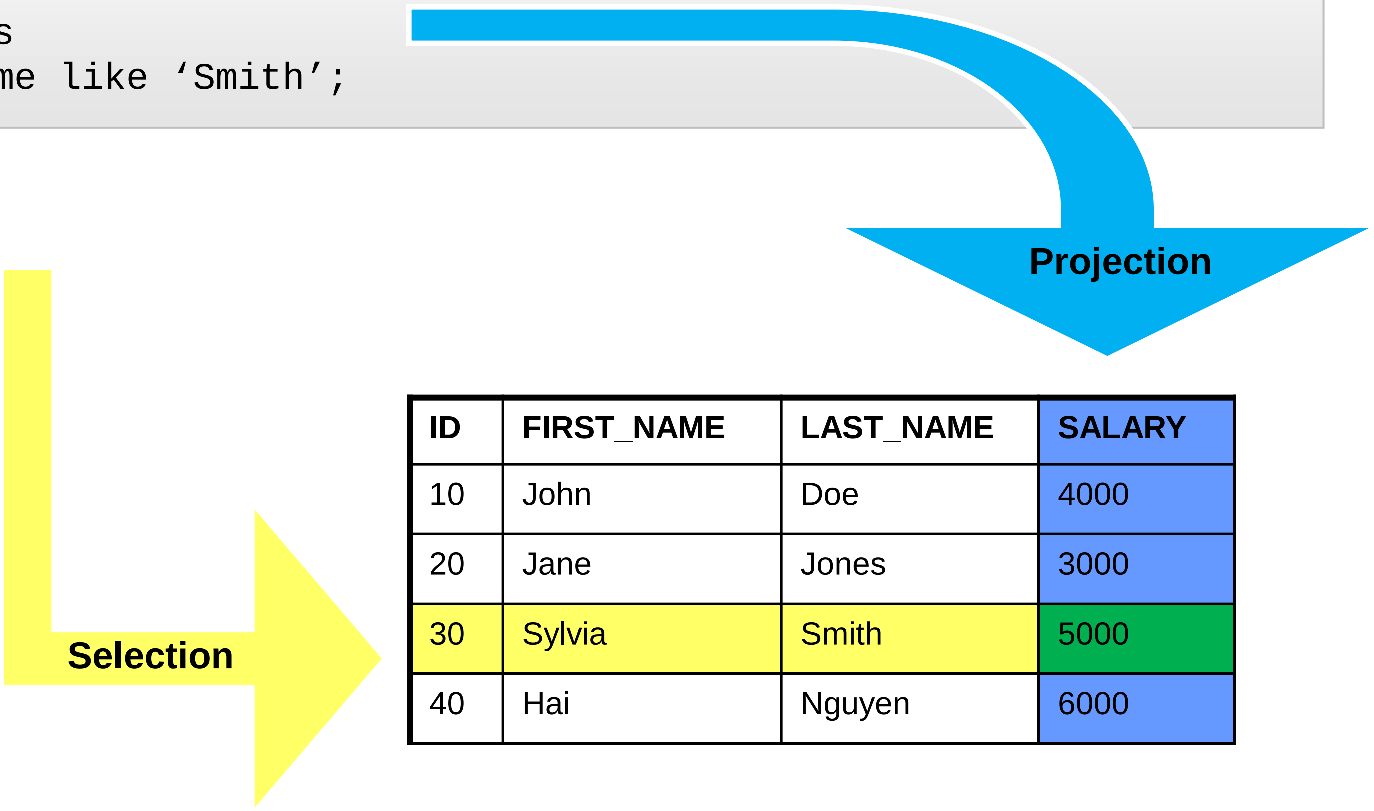


Table 2: Selection



Projection and Selection

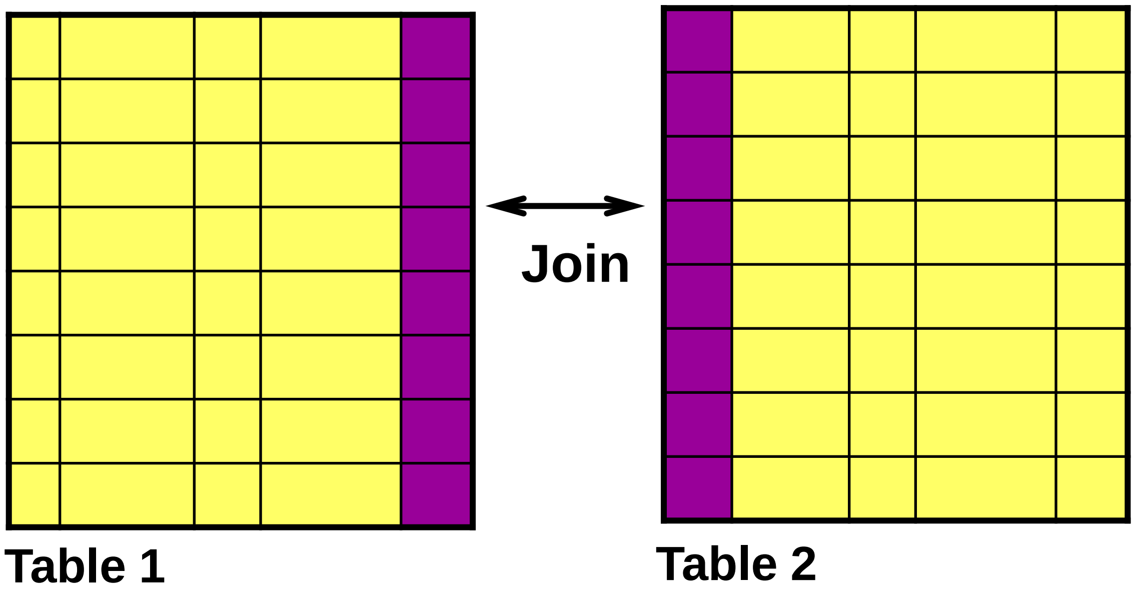
```
SELECT salary
FROM employees
WHERE last_name like ‘Smith’;
```



ID	FIRST_NAME	LAST_NAME	SALARY
10	John	Doe	4000
20	Jane	Jones	3000
30	Sylvia	Smith	5000
40	Hai	Nguyen	6000

Joining Tables Together

Join: Used to bring together data that is stored in different tables by creating a link between them. You will learn about joins later in the course.



Selecting All Columns

You can display all the columns of data in a table by using an asterisk symbol (*) instead of a column name in the SELECT clause.

```
SELECT *
FROM d_songs;
```

In the following example, all of the columns in the d_songs table are selected.

ID	TITLE	DURATION	ARTIST	TYPE_CODE
45	Its Finally Over	5 min	The Hobbits	12
46	I'm Going to Miss My Teacher	2 min	Jane Pop	12
47	Hurrah for Today	3 min	The Jubilant Trio	77
48	Meet Me at the Altar	6 min	Bobby West	1
49	Lets Celebrate	8 min	The Celebrants	77
50	All These Years	10 min	Diana Crooner	88

Selecting All Columns (cont.)

You can also display all the columns in a table by listing them individually.

```
SELECT id, title, duration, artist, type_code
FROM d_songs;
```

ID	TITLE	DURATION	ARTIST	TYPE_CODE
45	Its Finally Over	5 min	The Hobbits	12
46	I'm Going to Miss My Teacher	2 min	Jane Pop	12
47	Hurrah for Today	3 min	The Jubilant Trio	77
48	Meet Me at the Altar	6 min	Bobby West	1
49	Lets Celebrate	8 min	The Celebrants	77
50	All These Years	10 min	Diana Crooner	88

Projecting Specific Columns

If you want to PROJECT only specific columns from a table to be displayed, simply list each of the column names you want and separate each name with a comma in the SELECT clause.

```
SELECT id, title, artist
FROM d_songs;
```

ID	TITLE	ARTIST
45	Its Finally Over	The Hobbits
46	I'm Going to Miss My Teacher	Jane Pop
47	Hurrah for Today	The Jubilant Trio
48	Meet Me at the Altar	Bobby West
49	Lets Celebrate	The Celebrants
50	All These Years	Diana Crooner

Using Arithmetic Operators

Using a few simple rules and guidelines, you can construct SQL statements that are both easy to read and easy to edit. Knowing the rules will make learning SQL easy.

You may need to modify the way in which data is displayed, perform calculations, or look at what-if scenarios. For example, "What if every employee was given a 5% raise? How would that affect our yearly profit figures?"

Using Arithmetic Operators (cont.)

These types of calculations are all possible using arithmetic expressions. You are already familiar with arithmetic expressions in mathematics:

add (+), subtract (-) , multiply (*) and divide (/).

Note that this example does not create new columns in the tables or change the actual data values. The results of the calculations will appear only in the output.

Using Arithmetic Operators (cont.)

The example shown uses the addition operator to calculate a salary increase of 300 for all employees and displays a new SALARY + 300 column in the output.

```
SELECT last_name, salary,
salary + 300
FROM employees;
```

LAST_NAME	SALARY	SALARY + 300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Whalen	4400	4700
Higgins	12000	12300
Gletz	8300	8600
Zlotkey	10500	10800
Abel	11000	11300
Taylor	8600	8900
Grant	7000	7300

Putting in blank spaces before and after an arithmetic operator will not affect the output.

Precedence in Arithmetic Operators

Precedence is the order in which Oracle evaluates different operators in the same expression. When evaluating an expression containing multiple operators, Oracle evaluates operators with higher precedence before evaluating those with lower precedence. Oracle evaluates operators with equal precedence from left to right within an expression.

Precedence in Arithmetic Operators (cont.)

Arithmetic operators perform the mathematical operations of Multiplication, Division, Addition, and Subtraction. If these operators appear together in an expression, multiplication and division are evaluated first. So the order is: * / + -.

An easy way to remember their operator precedence is the mnemonic device:

My **D**ear **A**unt **S**ally

Precedence in Arithmetic Operators (cont.)

If operators within an expression are of the same priority, then evaluation is done from left to right. You can always use parentheses to force the expression within parentheses to be evaluated first.

In the example tables shown on the next slide, what are the differences in the output between the query that used parentheses and the one that didn't?

Precedence in Arithmetic Operators (cont.)

Operator Precedence

LAST_NAME	SALARY	12 * SALARY + 100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100
Whalen	4400	52900
Higgins	12000	144100
Gietz	8300	99700
…	…	…

Using Parentheses

LAST_NAME	SALARY	12 * (SALARY + 100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Whalen	4400	54000
Higgins	12000	145200
Gietz	8300	100800
…	…	…

NULL Values

In SQL, NULL is an interesting word. To understand NULL, you have to know what NULL is and what NULL is not. NULL is a value that is unavailable, unassigned, unknown, or inapplicable. NULL is not the same as a zero or a space. In SQL, a zero is a number, and a space is a character.

Sometimes, you don't know the value for a column. In a database, you can store unknowns in your databases. Relational databases use a placeholder called NULL or null to represent these unknown values.

NULL Values (cont.)

If any column value in an arithmetic expression is null, the result is null or unknown.

If you try to divide by null, the result is null or unknown. However, if you try to divide by zero, you get an error.

Salaries and Commissions

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
De Haan	AD_VP	17000	
Whalen	AD_ASST	4400	
Higgins	AC_MGR	12000	.2
Gietz	AC_ACCOUNT	8300	.3
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.15
…	…	…	…

NULL Values (cont.)

```
SELECT last_name, job_id, salary,
commission_pct, salary*commission_pct
FROM employees;
```

Salaries and Commissions

LAST_NAME	JOB_ID	SALARY	COMMISSION_ PCT	SALARY* COMMISSION_PCT
King	AD_PRES	24000		
Kochhar	AD_VP	17000		
De Haan	AD_VP	17000		
Whalen	AD_ASST	4400		
Higgins	AC_MGR	12000	.2	2400
Gietz	AC_ACCOUNT	8300	.3	2490
Zlotkey	SA_MAN	10500	.2	2100
Abel	SA_REP	11000	.15	1650
…	…	…	…	

Aliases

An Alias is a way of renaming a column heading in the output. Without aliases, when the result of a SQL statement is displayed, the name of the columns displayed will be the same as the column names in the table or a name showing a arithmetic operation such as 12*(SALARY + 100).

You probably want your output to display a name that is easier to understand, a more "friendly" name. Column aliases let you rename columns in the output.

Aliases (cont.)

There are several rules when using column aliases to format output.

A column alias:

- Renames a column heading
- Is useful with calculations
- Immediately follows the column name
- May have the optional AS keyword between the column name and alias
- Requires double quotation marks if the alias contains spaces or special characters, or is case-sensitive

Using Column Aliases

The syntax for aliases is:

```
SELECT * |column|expr [ AS alias], .....
FROM      table;
```

Examples:

```
SELECT last_name AS name,
commission_pct AS comm
FROM   employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

```
SELECT last_name "Name",
salary*12 AS "Annual Salary"
FROM   employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

Terminology

Key terms used in this lesson included:

- Arithmetic expression
- Arithmetic operator
- Clause
- Column
- Column alias
- From clause
- Join
- NULL

Terminology (cont.)

Key terms used in this lesson included:

- Projection
- Select clause
- Selection
- Select statement
- Statement

Summary

In this lesson, you should have learned how to:

- Match projection, selection, and join with their correct functions/ capabilities
- Create a basic SELECT statement
- Use the correct syntax to display all rows in a table
- Use the correct syntax to select specific columns in a table, modify the way data is displayed, and perform calculations using arithmetic expressions and operators

Summary (cont.)

In this lesson, you should have learned how to:

- Formulate queries using correct operator precedence to display desired results
- Define a null value
- Demonstrate the effect null values create in arithmetic expressions
- Construct a query using a column alias