

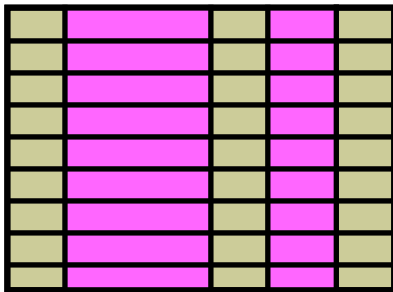
Retrieving Data Using the SQL `SELECT` Statement

Objectives

- After completing this lesson, you should be able to do the following:
 - List the capabilities of SQL `SELECT` statements
 - Execute a basic `SELECT` statement
 - Differentiate between SQL statements and SQL*Plus commands

Capabilities of SQL `SELECT` Statements

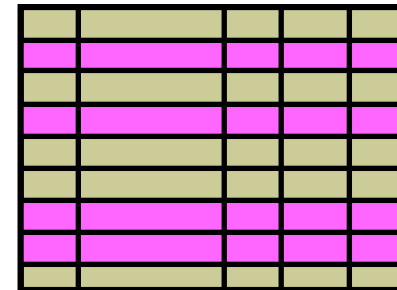
Projection



A 10x5 grid representing a table. The second, third, and fourth columns are highlighted in pink, illustrating the result of a projection operation that selects specific columns from the original table.

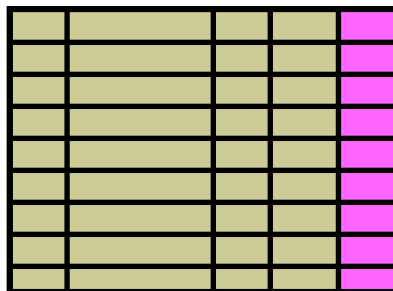
Table 1

Selection



A 10x5 grid representing a table. The first, third, fourth, and fifth rows are highlighted in pink, illustrating the result of a selection operation that filters specific rows from the original table.

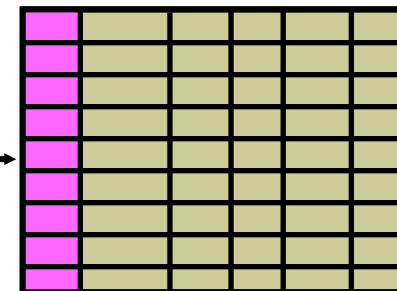
Table 1



A 10x5 grid representing a table. The fifth column is highlighted in pink, representing the result of a join operation where a specific column from the first table is joined with the result of another operation.

Table 1

Join



A 10x5 grid representing a table. The first column is highlighted in pink, representing the result of a join operation where a specific column from the second table is joined with the result of another operation.

Table 2

Basic SELECT Statement

```
SELECT * | { [DISTINCT] column | expression [alias], ... }  
FROM      table;
```

- **SELECT** identifies the **columns** to be displayed
- **FROM** identifies the **table** containing those columns

Selecting All Columns

```
SELECT *  
FROM departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

Selecting Specific Columns

```
SELECT department_id, location_id  
FROM departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Writing SQL Statements

- SQL statements are **not case-sensitive**.
- SQL **statements** can be **on one or more lines**.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- In SQL*Plus, SQL statements can optionally be terminated by a semicolon (;). Semicolons are required if you execute multiple SQL statements.

Arithmetic Expressions

- Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators

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

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...

20 rows selected.

Operator Precedence

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

1

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100

...

20 rows selected.

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

2

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200

...

20 rows selected.

Defining a Null Value

- A **null** is a value that is **unavailable**, **unassigned**, **unknown**, or **inapplicable**.
- A null is not the same as a zero or a blank space.

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

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

Null Values in Arithmetic Expressions

- Arithmetic expressions containing a null value **evaluate to null**.

```
SELECT last_name, 12*salary*commission_pct  
FROM employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

Defining a Column Alias

- A column alias:
 - Renames a column heading
 - Is useful with calculations
 - Immediately follows the column name (There can also be the optional `AS` keyword between the column name and alias.)
 - Requires double quotation marks if it contains spaces or special characters or if it is case-sensitive

Using Column Aliases

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

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

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

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

20 rows selected.

Concatenation Operator

- A concatenation operator:
 - Links columns or character strings to other columns
 - Is represented by two vertical bars (||)
 - Creates a resultant column that is a character expression

```
SELECT    last_name||job_id AS "Employees"  
FROM      employees;
```

Employees	
KingAD_PRES	
KochharAD_VP	
De HaanAD_VP	

...

20 rows selected.

Literal Character Strings

- A literal is a character, a number, or a date that is included in the `SELECT` statement.
- Date and character literal values must be enclosed by single quotation marks.
- Each character string is output once for each row returned.

Using Literal Character Strings

```
SELECT last_name || ' is a ' || job_id  
       AS "Employee Details"  
FROM   employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...

20 rows selected.

Alternative Quote (q) Operator

- Specify your own quotation mark delimiter
- Choose any delimiter (or {}, [], <> pairs)
- Increase readability and usability

```
SELECT department name ||  
       q'[, it's assigned Manager Id: ]'  
       || manager_id  
       AS "Department and Manager"  
FROM departments;
```

Department and Manager
Administration, it's assigned manager ID: 200
Marketing, it's assigned manager ID: 201
Shipping, it's assigned manager ID: 124

...

8 rows selected.

Duplicate Rows

- The **default** display of queries is all rows, **including duplicate rows**.

```
SELECT department_id  
FROM employees;
```

1

DEPARTMENT_ID	
	90
	90
	90

...

20 rows selected.

```
SELECT DISTINCT department_id  
FROM employees;
```

2

DEPARTMENT_ID	
	10
	20
	50

...

8 rows selected.

SQL Statements Versus SQL*Plus Commands

SQL

- A language
- ANSI standard
- Keyword cannot be abbreviated
- Statements manipulate data and table definitions in the database
- Runs in database

SQL
statements

SQL*Plus

- An environment
- Oracle-proprietary
- Keywords can be abbreviated
- Commands do not allow manipulation of values in the database
- Runs on a client machine

SQL*Plus
commands

Displaying Table Structure

- Use the SQL*Plus DESCRIBE command to display the structure of a table:

```
DESC[RIBE] tablename
```

Displaying Table Structure

```
DESCRIBE employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)