# 1

# Retrieving Data Using the SQL `SELECT` Statement

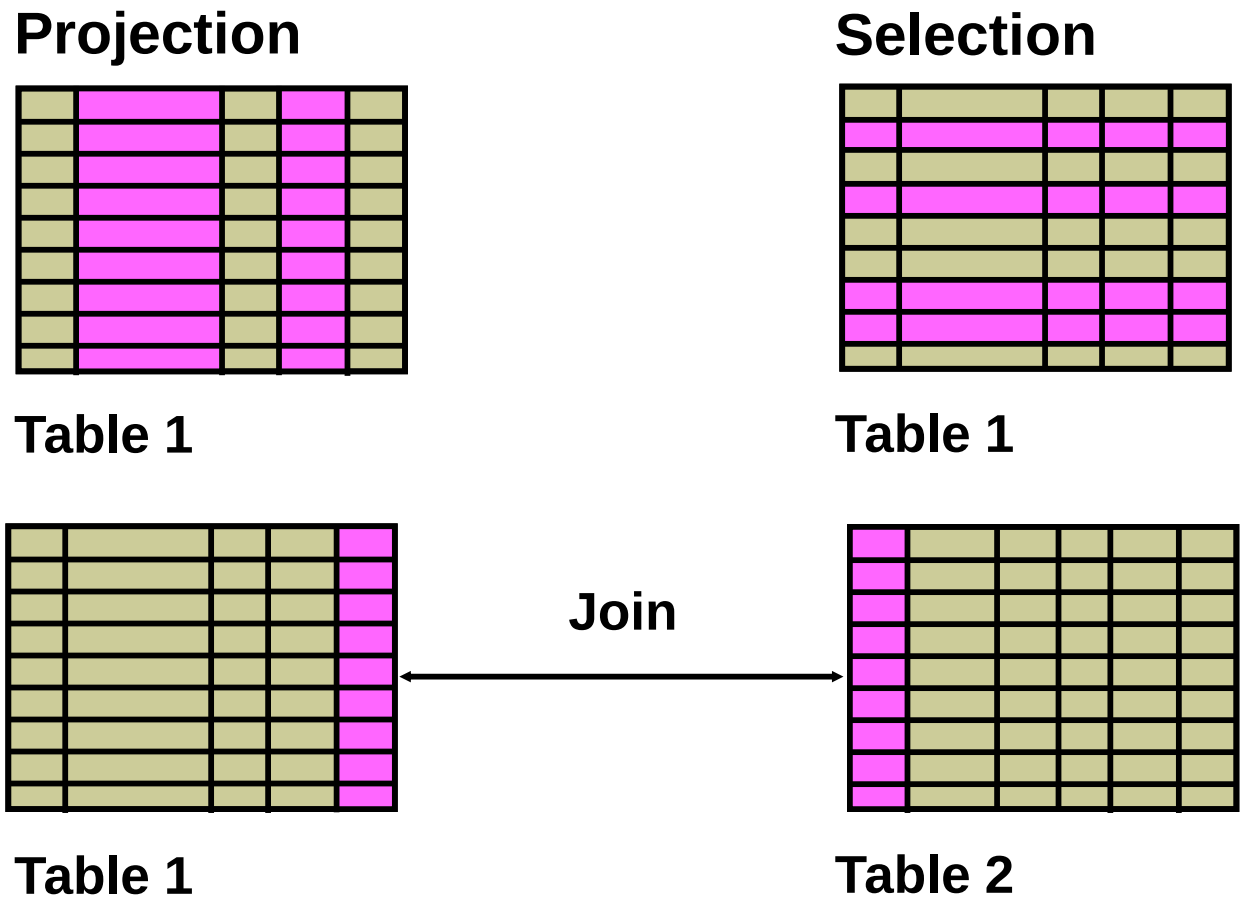
Oracle SQL Programming: Rebecca Gottlieb

ORACLE

# Lecture 1 Review

- Start SQL*Plus – username?  Schema?
- Executing a script in SQL*Plus?
- login.sql script
- ABC Company Solution
- Primary Key
- Foreign Key
- Not Null value
- 1:1, 1:M, M:M
- Table Structure – column data types

# Capabilities of SQL `SELECT` Statements



# 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 |
|---|---|---|---|---|
| 1 | 10 | Administration | 200 | 1700 |
| 2 | 20 | Marketing | 201 | 1800 |
| 3 | 50 | Shipping | 124 | 1500 |
| 4 | 60 | IT | 103 | 1400 |
| 5 | 80 | Sales | 149 | 2500 |
| 6 | 90 | Executive | 100 | 1700 |
| 7 | 110 | Accounting | 205 | 1700 |
| 8 | 190 | Contracting | (null) | 1700 |

# Selecting Specific Columns

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

| | DEPARTMENT_ID | LOCATION_ID |
|---|---|---|
| 1 | 10 | 1700 |
| 2 | 20 | 1800 |
| 3 | 50 | 1500 |
| 4 | 60 | 1400 |
| 5 | 80 | 2500 |
| 6 | 90 | 1700 |
| 7 | 110 | 1700 |
| 8 | 190 | 1700 |

# Writing SQL Statements

- SQL statements are not case-sensitive.
- SQL statements can be entered 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 Developer, SQL statements can optionally be terminated by a semicolon (;). Semicolons are required when you execute multiple SQL statements.
- In SQL*Plus, you are required to end each SQL statement with a semicolon (;).

# Column Heading Defaults

- SQL Developer:
  - Default heading alignment: Left-aligned
  - Default heading display: Uppercase
- SQL*Plus:
  - Character and Date column headings are left-aligned.
  - Number column headings are right-aligned.
  - Default heading display: Uppercase

# 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 |
|---|---|---|---|
| 1 | King | 24000 | 24300 |
| 2 | Kochhar | 17000 | 17300 |
| 3 | De Haan | 17000 | 17300 |
| 4 | Hunold | 9000 | 9300 |
| 5 | Ernst | 6000 | 6300 |
| 6 | Lorentz | 4200 | 4500 |
| 7 | Mourgos | 5800 | 6100 |
| 8 | Rajs | 3500 | 3800 |
| 9 | Davies | 3100 | 3400 |
| 10 | Matos | 2600 | 2900 |

...

# Operator Precedence

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

1

| | LAST_NAME | SALARY | 12*SALARY+100 |
|---|---|---|---|
| 1 | King | 24000 | 288100 |
| 2 | Kochhar | 17000 | 204100 |
| 3 | De Haan | 17000 | 204100 |

...

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

2

| | LAST_NAME | SALARY | 12*(SALARY+100) |
|---|---|---|---|
| 1 | King | 24000 | 289200 |
| 2 | Kochhar | 17000 | 205200 |
| 3 | De Haan | 17000 | 205200 |

...

# Defining a Null Value

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

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

| | LAST_NAME | JOB_ID | SALARY | COMMISSION_PCT |
|---|---|---|---|---|
| 1 | King | AD_PRES | 24000 | (null) |
| 2 | Kochhar | AD_VP | 17000 | (null) |
| ... | | | | |
| 12 | Zlotkey | SA_MAN | 10500 | 0.2 |
| 13 | Abel | SA_REP | 11000 | 0.3 |
| 14 | Taylor | SA_REP | 8600 | 0.2 |
| ... | | | | |
| 19 | Higgins | AC_MGR | 12000 | (null) |
| 20 | Gietz | AC_ACCOUNT | 8300 | (null) |

# 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 |
|---|---|---|
| 1 | King | (null) |
| 2 | Kochhar | (null) |
| ... | | |
| 12 | Zlotkey | 25200 |
| 13 | Abel | 39600 |
| 14 | Taylor | 20640 |
| ... | | |
| 19 | Higgins | (null) |
| 20 | Gietz | (null) |

# 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 |
|---|---|---|
| 1 | King | (null) |
| 2 | Kochhar | (null) |
| 3 | De Haan | (null) |

...

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

| | Name | Annual Salary |
|---|---|---|
| 1 | King | 288000 |
| 2 | Kochhar | 204000 |
| 3 | De Haan | 204000 |

...

# 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 |
|---|---|
| 1 | AbelSA_REP |
| 2 | DaviesST_CLERK |
| 3 | De HaanAD_VP |
| 4 | ErnstIT_PROG |
| 5 | FayMK_REP |

...

# 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 within 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 |
|---|---|
| 1 | Abel is a SA_REP |
| 2 | Davies is a ST_CLERK |
| 3 | De Haan is a AD_VP |
| 4 | Ernst is a IT_PROG |
| 5 | Fay is a MK_REP |

...

| | |
|---|---|
| 18 | Vargas is a ST_CLERK |
| 19 | Whalen is a AD_ASST |
| 20 | Zlotkey is a SA_MAN |

# Alternative Quote (q) Operator

- Specify your own quotation mark delimiter.
- Select any delimiter.
- Increase readability and usability.

```
SELECT department_name || q'[ Department's Manager Id: ]'
       || manager_id
       AS "Department and Manager"
FROM departments;
```

| | Department and Manager |
|---|---|
| 1 | Administration Department's Manager Id:200 |
| 2 | Marketing Department's Manager Id:201 |
| 3 | Shipping Department's Manager Id:124 |
| 4 | IT Department's Manager Id:103 |
| 5 | Sales Department's Manager Id:149 |
| 6 | Executive Department's Manager Id:100 |
| 7 | Accounting Department's Manager Id:205 |
| 8 | Contracting Department's Manager Id: |

# Duplicate Rows

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

```
SELECT department_id
FROM   employees;
```

1

| | DEPARTMENT_ID |
|---|---|
| 1 | 90 |
| 2 | 90 |
| 3 | 90 |
| 4 | 60 |
| 5 | 60 |

...

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

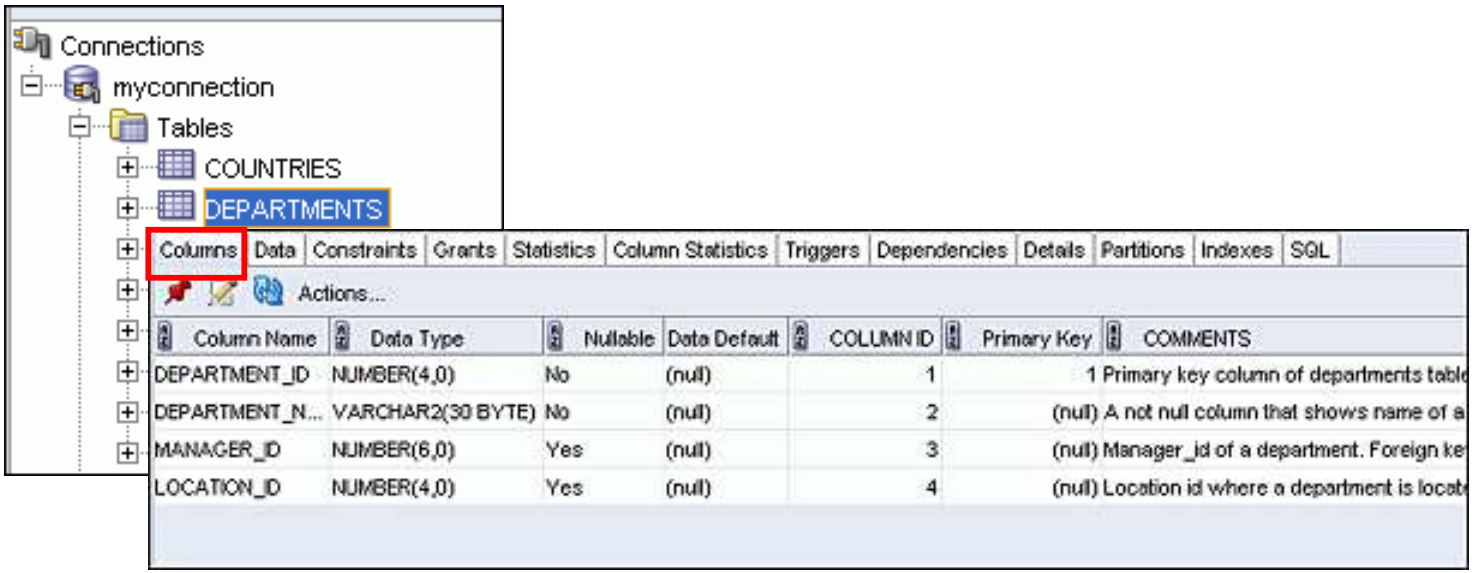
2

| | DEPARTMENT_ID |
|---|---|
| 1 | (null) |
| 2 | 90 |
| 3 | 20 |
| ... 4 | 110 |

# Displaying the Table Structure

- Use the `DESCRIBE` command to display the structure of a table.
- Or, select the table in the Connections tree and use the Columns tab to view the table structure.

```
DESC[RIBE] tablename
```



# Using the DESCRIBE Command

```
DESCRIBE employees
```

```
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)

11 rows selected
```

# Quiz

Identify the `SELECT` statements that execute successfully.

1.
```
SELECT first_name, last_name, job_id, salary*12
 AS Yearly Sal
FROM   employees;
```

2.
```
SELECT first_name, last_name, job_id, salary*12
 yearly sal
FROM   employees;
```

3.
```
SELECT first_name, last_name, job_id, salary AS
 yearly sal
FROM   employees;
```

4.
```
SELECT first_name+last_name AS name, job_Id,
 salary*12 yearly sal
FROM   employees;
```

# Summary

In this lesson, you should have learned how to:

- Write a `SELECT` statement that:
  - Returns all rows and columns from a table
  - Returns specified columns from a table
  - Uses column aliases to display more descriptive column headings

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