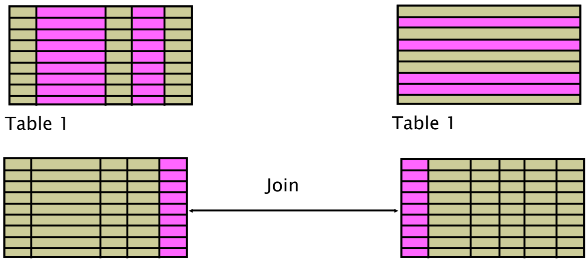
Retrieving Data Using the SQL SELECT Statement

Objectives

After completing this , you should be able to do the following:

- List the capabilities of SQL SELECT statements
- Execute a basic SELECT statement

Capabilities of SQL SELECT Statement



Projection

Selection

Table 1

Table 1

Join

Table 1

Table 2

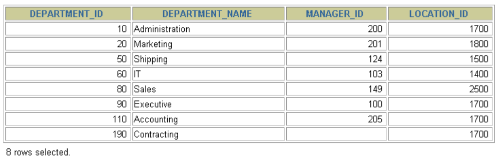
Basic SELECT Statement

```
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;
```

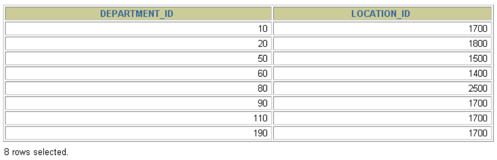


DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

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.
- In SQL Developer, SQL statements can optionally be terminated by a semicolon (;). Semicolons are required if 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: Center
  - Default heading display: Uppercase
- SQL*Plus:
  - Character and Date column headings are left-aligned
  - Number column headings are right-aligned
  - Default heading display: Uppercase

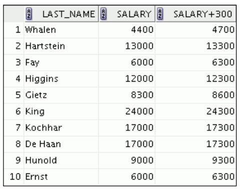
Arithmetic Expressions

- 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;
```

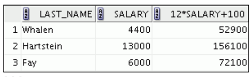


	LAST_NAME	SALARY	SALARY+300
1	Whalen	4400	4700
2	Hartstein	13000	13300
3	Fay	6000	6300
4	Higgins	12000	12300
5	Gietz	8300	8600
6	King	24000	24300
7	Kochhar	17000	17300
8	De Haan	17000	17300
9	Hunold	9000	9300
10	Ernst	6000	6300

…

Operator Precedence

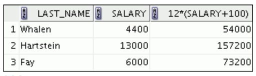
```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```



	LAST_NAME	SALARY	12*SALARY+100
1	Whalen	4400	52900
2	Hartstein	13000	156100
3	Fay	6000	72100

…

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



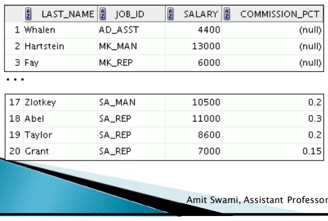
	LAST_NAME	SALARY	12*(SALARY+100)
1	Whalen	4400	54000
2	Hartstein	13000	157200
3	Fay	6000	73200

…

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

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



	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Whalen	AD_ASST	4400	(null)
2	Hartstein	MK_MAN	13000	(null)
3	Fay	MK_REP	6000	(null)

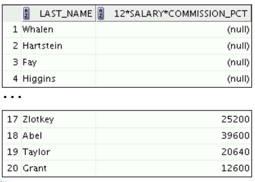
…

17	Zlotkey	SA_MAN	10500	0.2
18	Abel	SA_REP	11000	0.3
19	Taylor	SA_REP	8600	0.2
20	Grant	SA_REP	7000	0.15

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	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)
4	Higgins	(null)

…

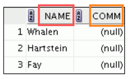
17	Zlotkey	25200
18	Abel	39600
19	Taylor	20640
20	Grant	12600

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

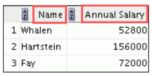
Using Column Aliases

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



	NAME	COMM
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)

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



	Name	Annual Salary
1	Whalen	52800
2	Hartstein	156000
3	Fay	72000

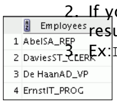
Concatenation Operator

- 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;
```

Note:

1. The AS keyword before the alias name makes the SELECT clause easier to read.
2. If you concatenate a null value with a character string, the result is a character string.
3. Ex: LAST_NAME || NULL results in LAST_NAME.



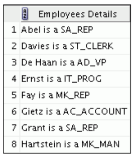
	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG

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

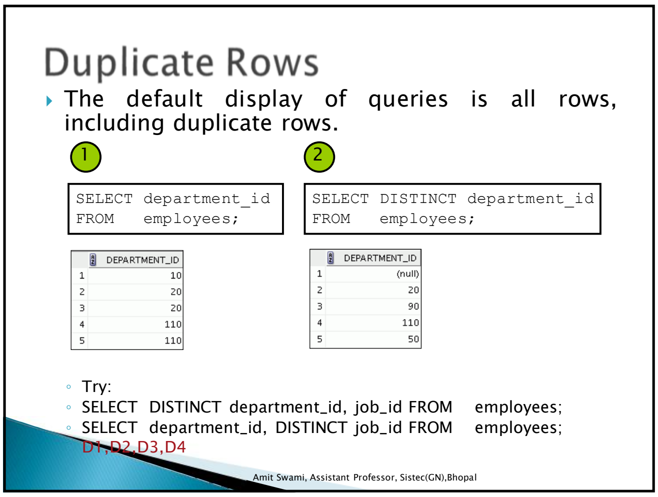
```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```



	Employees 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
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN

…

- Try:
  SELECT last_name ||': 1 Month salary = '||salary Monthly
  FROM   employees;



Duplicate Rows

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

1

```
SELECT department_id
FROM   employees;
```

	DEPARTMENT_ID
1	10
2	20
3	20
4	110
5	110

2

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

	DEPARTMENT_ID
1	(null)
2	20
3	90
4	110
5	50

- Try:
- SELECT DISTINCT department_id, job_id FROM employees;
- SELECT department_id, DISTINCT job_id FROM employees;

D1,D2,D3,D4