IT2351 / IT2851 / IT2552 / IT2152 / IT2651

Database Management Systems

Structured Query Language (Basic SELECT)

Unit Objectives

- At the end of this unit, you should be able to
  - Use SQL to query databases (SELECT).
  - Use SQL to manipulate data in databases (INSERT, UPDATE, and DELETE).
  - Use SQL to define and create databases.

Topics

- Lesson A:
  - Overview of SQL
  - DML (Querying the database)
    - Basic SELECT statement
- Lesson B:
  - DML (Querying the database)
    - Advanced SELECT statement
- Lesson C:
  - DML (Updating the Database)
    - INSERT statement
    - UPDATE statement
    - DELETE statement
  - DDL (Defining the Database)
    - CREATE, ALTER statement

Overview of SQL

- Background
  - Structured Query Language
  - SQL has become the standard relational database language.
  - In 1986, a standard for SQL was defined by ANSI, which was subsequently adopted in 1987 as an international standard by the ISO.

- It has 2 major components:
  - Data Manipulation Language (DML):
    select, insert, update, delete
  - Data Definition Language (DDL):
    create and manage database and relations structure

Overview of SQL

- It is a non-procedural language.
- SQL does not contain flow control commands.
  - It can be issued interactively or embedded within an application program.
- It can be used by a range of users
- An ISO standard now exists for SQL, making it both the formal and de facto standard language for relational databases

Writing SQL Commands

- SQL statement consists of reserved words and user-defined words.

  - **Reserved words** are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.

  - **User-defined words** are made up by user and represent names of various database objects such as relations, columns, views.

Writing SQL Commands

- Most components of an SQL statement are **case insensitive**, except for literal character data.

- More readable with indentation and lineation:
  - Each clause should begin on a new line.
  - Start of a clause should line up with start of other clauses.
  - If clause has several parts, should each appear on a separate line and be indented under start of clause.

SELECT Statement

- The SELECT Statement
  - **SELECT** is the most important and the most complex SQL statement.

  - It can be used
    - to retrieve and display data from one or more tables.
    - as part of an **INSERT** statement to produce new rows.
    - as part of **UPDATE / DELETE** statement to update/delete data.

SELECT statement

- Syntax :

```
SELECT          [DISTINCT] column_list
FROM            table_name
{[INNER JOIN table_name ON condition]}
[WHERE          condition]
[GROUP BY       column_list]
[HAVING         condition]
[ORDER BY       column_list [DESC]]
```

- Only SELECT & FROM are mandatory
- Order of the clauses cannot be changed

SELECT statement

SELECT	Specifies which columns are to appear in output
FROM	Specifies table to be used
{[INNER JOIN .. ON ..]}	Specifies other table(s) to be joined. Repeats for each additional table.
[WHERE]	Filters rows
[GROUP BY]	Forms groups of rows with same column value
[HAVING]	Filters groups subject to some condition
[ORDER BY]	Specifies the order of the rows in the output

SELECT … FROM clause

```
SELECT          [DISTINCT] column_list
FROM            table_list
```

- Retrieve <u>full details</u> of all customers
  - Use * to denote <u>ALL</u> columns OR specify each column explicitly
    - select * from customer;
    - Select customer_num, fname, lname, address1, zipcode
      from   customer;

customer_num	fname	lname	address1	zipcode
101	Ludwig	Pauli	213 Erstwild Court	94086
102	Carole	Sadler	785 Geary St	94117
103	Philip	Currie	654 Poplar	94303

3 rows selected

SELECT … FROM clause

```
SELECT          [DISTINCT] column_list
FROM            table_list
```

- Retrieve <u>specific columns</u> of all customers
  - select zipcode, fname, lname from customer;

customer_num	fname	lname	address	zipcode
101	Ludwig	Pauli	213 Erstwild Court	94086
102	Carole	Sadler	785 Geary St	94117
103	Philip	Currie	654 Poplar	94303

Customer Table

zipcode	fname	lname
94086	Ludwig	Pauli
94117	Carole	Sadler
94303	Philip	Currie

3 rows selected

Can select in any order regardless of the order of the columns in the table. <u>Data independence</u>.

SELECT … FROM clause

```
SELECT      [DISTINCT] column_list
FROM        table_list
```

- Retrieve distinct column values from the table(s)
  - select **distinct** zipcode from customer;

  OR

  - select **unique** zipcode from customer;

Example

zipcode
123456
123456
654321

Table data

zipcode
123456
654321

2 rows selected

SELECT … FROM clause

```
SELECT        [DISTINCT] column_list
FROM          table_list
```

- You may have <u>calculated (derived) columns</u> in the *column_list* :
  i) By performing <u>arithmetic operations</u> on the base table columns :

     - select   prod_num, *unit_price*1.1* ***new_unit_price***
       from   product ;
     - select   order_num, *datediff(ship_date, order_date)* ***span***
       from   orders ;

You may give an alias to the calculated field (optional)

SELECT … FROM clause

Alias for the calculated field (optional)

- select prod_num, *unit_price*1.1* ***new_unit_price***
  from product ;

prod_num	unit_price
200	150
201	200

Product

prod_num	**new_unit_price**
200	165
201	220

2 rows selected

- select order_num, datediff(*ship_date, order_date*) ***span***
  from orders ;

order_num	ship_date	order_date
1001	1-jun-2007	20-may-2007
1002	26-may-2007	20-may-2007

Orders

order_num	**span**
1001	12
1002	5

2 rows selected

SELECT … FROM clause

You may have <u>calculated (derived) columns</u> in the *column_list* :

ii) By applying <u>round function</u> on the columns :

```
select      prod_num, round(unit_price, 0)
from        product ;
```

Example

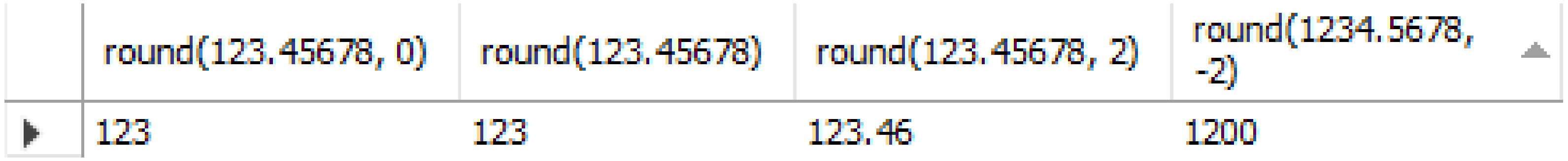
prod_num	unit_price
113	685.7
120	37

Table data

prod_num	Round(unit_price,0)
113	686
120	37

2 rows selected

SELECT round(123.456, 0), round(123.456), round(123.456, 2), round(1234.56, -2);



	round(123.45678, 0)	round(123.45678)	round(123.45678, 2)	round(1234.5678, -2)
▶	123	123	123.46	1200

SELECT … FROM clause

You may have calculated (derived) columns in the *column_list* :

iii) By applying **concat** function on the base table columns :

- select *CONCAT(fname, ' ', lname)* ***cust_name***
  from customer ;

Alias given to the calculated field (optional)

Example:

fname	lname
Ah Kaw	Lim
Jennifer	Tan
Jeffrey	Koh

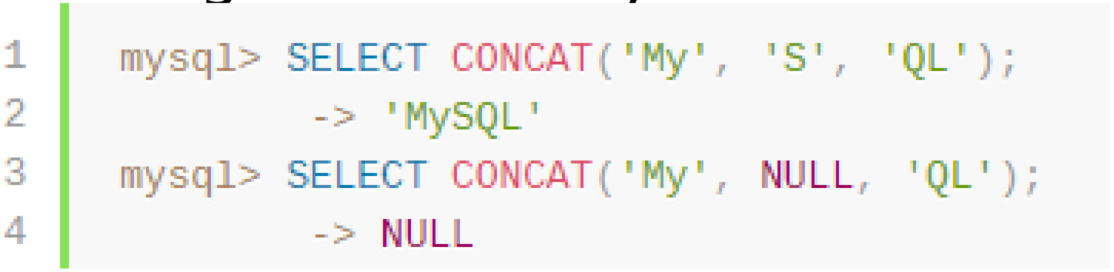
Table data

cust_name
Lim Ah Kaw
Tan Jennifer
Koh Jeffrey

3 rows selected

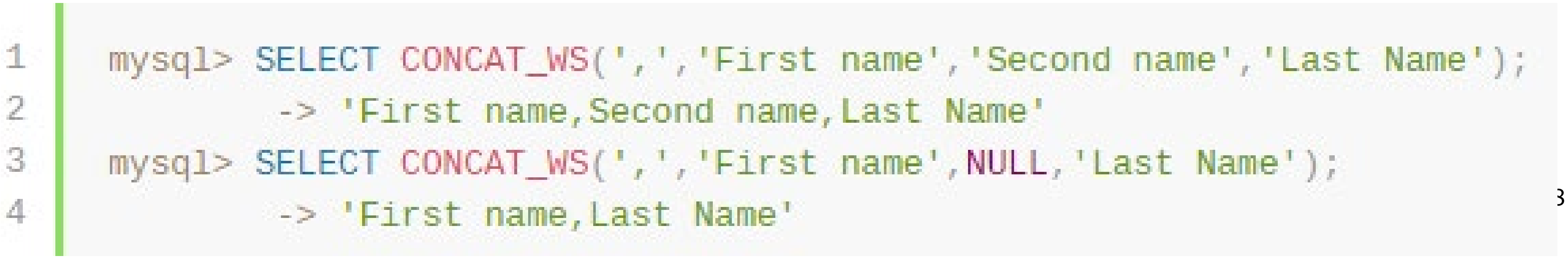
String Concatenation (MySQL specific)

- CONCAT(str1, str2, ….)
  - Returns the string that results from concatenating the arguments. May have one or more arguments.



```
mysql> SELECT CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
        -> NULL
```

- CONCAT_WS(separator,str1,str2,...)
  - stands for Concatenate With Separator and is a special form of CONCAT().



```
mysql> SELECT CONCAT_WS(',','First name','Second name','Last Name');
        -> 'First name,Second name,Last Name'
mysql> SELECT CONCAT_WS(',','First name',NULL,'Last Name');
        -> 'First name,Last Name'
```

SELECT … FROM clause

You may have <u>calculated (derived) columns</u> in the *column_list* :

iv) By applying <u>substr function</u> on the columns

```
select      substr(zipcode,1,3)
from        customer ;
```

Syntax:

substr(str, start_position, length)

Example

Zipcode
123456
123456
654321

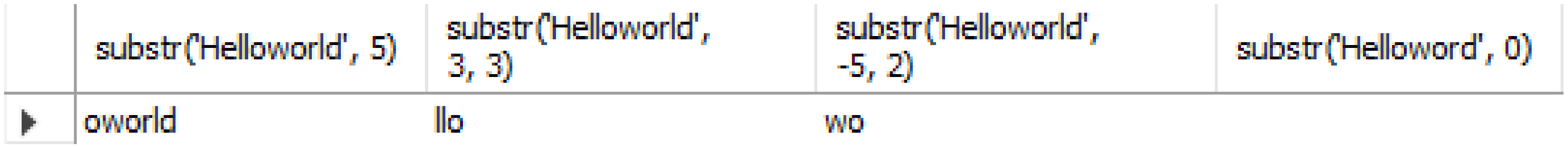
Table data

substr(zipcode,1,3)
123
123
654

3 rows selected

Function SUBSTR more example

select substr('Helloworld', 5), substr('Helloworld', 3, 3), substr('Helloworld', -5, 2), substr('Helloword', 0) ;



	substr('Helloworld', 5)	substr('Helloworld', 3, 3)	substr('Helloworld', -5, 2)	substr('Helloword', 0)
▶	oworld	llo	wo	

SELECT column_list FROM table

- SELECT DISTINCT zipcode FROM customer;
- You may have **calculated (derived)** columns in the column_list:

  -
    ```
    select  prod_num, unit_price*1.1 new_unit_price
    from    product ;
    ```

  -
    ```
    select  prod_num, round(unit_price, 0)
    from    product ;
    ```

  -
    ```
    select   CONCAT(fname, ‘ ‘, lname)   cust_name
    from     customer ;
    ```

  -
    ```
    select  substr(zipcode,1,3)
    from    customer ;
    ```

SELECT statement

- Syntax :

```
SELECT          [DISTINCT] column_list
FROM            table_name
{[INNER JOIN table_name ON condition]}
[WHERE          condition]
[GROUP BY       column_list]
[HAVING         condition]
[ORDER BY       column_list [DESC]]
```

- Only SELECT & FROM are mandatory
- Order of the clauses cannot be changed

WHERE condition clause

- Row Selection, using the WHERE clause
  - To restrict the rows to be retrieved based on the condition(s) specified on the base table columns:

    -
      ```
      select    prod_num, unit_price
      from      product
      where     unit_price > 500 ;
      ```

prod_num	unit_price
113	685.5
120	37

Table data

prod_num	unit_price
113	685.5

1 row selected

WHERE condition clause

- Row Selection, using the WHERE clause
  - conditions can also be specifed on <u>derived columns</u>:
    - select order_num, datediff(*ship_date, order_date) span*

      from orders

      where datediff(**ship_date,order_date) > 14** ;

order_num	ship_date	order_date	span
1004	30-may-2007	22-may-2007	8
1005	09-jun-2007	24-may-2007	16

Table data

order_num	**span**
1005	16

1 row selected

WHERE condition clause

Syntax :

[WHERE *column_name* <operator> *value(s)*]

- 5 basic search conditions that can be used in the WHERE clause :
  - Comparison (=, <, >, <=, >=, <>)

    Where salary > 5000

    Where state_code <> 'CA'

  - Range (BETWEEN, NOT BETWEEN)

    Where salary **BETWEEN** 5000 **and** 10000

    Where order_date **BETWEEN** '1994-07-01' **and** '1994-07-31'

WHERE condition clause

- 5 basic search conditions that can be used in the WHERE clause :
  - Set membership (IN, NOT IN)

    Where position **IN** (‘Manager’, ‘Deputy Manager’)

  - Pattern match (LIKE) with wildcards (%, _)

    Where address **LIKE** ‘Ang Mo Kio%’

    Where state_code **LIKE** ‘_A’

  - Null (IS NULL, IS NOT NULL)

    Where ship_instruct **IS NULL**

    Compare with : where ship_instruct = ‘ ‘, any difference ?

Most components of an SQL statement are **case insensitive**, except for **literal character data**. (Slide 7)

WHERE condition clause

- Two or more conditions can be combined with AND / OR :

```
Where     salary > 5000 AND position = 'Manager'
Where     order_date IS NULL OR ship_date IS NULL
```

ORDER BY clause

- To sort the rows in the query result, in ascending or descending order of a column value or a combination of columns

  Syntax :

  ```
      [order by    column_list [desc]]
  ```

  where column_list :

  - a column name in the *select clause*; or
  - a column number (e.g. 1 : the first element in the *select clause*, 2 : the second element, and so on)

    - order by   1, 2 desc
    - order by   1 desc, 2

ORDER BY clause

- Examples :
  - Sort in descending order of ZIPCODE :

```
SELECT      *
FROM        CUSTOMER
ORDER BY    ZIPCODE   DESC ;
```

  - Sort in ascending order of LNAME

```
SELECT      ZIPCODE,  LNAME,  FNAME
FROM        CUSTOMER
ORDER BY    2 ;
```

  - Sort in ascending order of SUPPL_CODE, followed by descending order of UNIT_PRICE

```
SELECT      *
FROM        PRODUCT
ORDER BY    SUPPL_CODE,  UNIT_PRICE  DESC ;
```

Summary

- Basic SELECT statement

```
  SELECT          [DISTINCT]  column_list
  FROM            table_name
{[INNER JOIN      table_name ON condition]}
  [WHERE          condition]
  [ORDER BY       column_list [DESC]]
```

Multiple Tables Queries

- To obtain information from different tables (e.g. customer table, order table).
- Could use a subquery or a join.
- Example (List all the orders made by customers) :

  -
    ```
    Select       fname, order_num
    From         customer c
    Inner Join   orders o
    On           c.customer_num=o.customer_num;
    ```

Multiple Tables Queries

- Joining Tables

Customer

customer_n	fname
1000	X
1001	Y
1002	Z
1003	A
1004	B

Orders

customer_nu	order_num	order_date	paid_date
1000	1	1/1/2003	
1000	2	2/2/2003	2/28/2003
1001	3	3/3/2003	
1002	4	4/4/2003	4/30/2003
1004	5	5/5/2003	

customer_num in the customer table

c.customer_num = o.customer_num

customer_num in the orders table

Result

c.customer_nu	fname	o.customer_nu	order_num	order_date	paid_date
1000	X	1000	1	1/1/2003	
1000	X	1000	2	2/2/2003	2/28/2003
1001	Y	1001	3	3/3/2003	
1002	Z	1002	4	4/4/2003	4/30/2003
1004	B	1004	5	5/5/2003	

Multiple Tables Queries

- To write a multiple table query :

```
Select       c.customer_num, fname, order_num
From         customer c
Inner Join   orders o On c.customer_num = o.customer_num
```

- Include the table in the FROM clause
- Use the INNER JOIN clause to specify each additional table
- Include a ON clause to specify the column(s) to join, these columns must have compatible data types
- Whenever there is ambiguity in the source of the columns (same column name used in multiple tables), may use an alias for the table to qualify the column name

Multiple Tables Queries Examples

- Example 1: List the customer's first name and name of the state they are in:

  -
    ```
    Select       fname, state_name
    From         customer c
    Inner Join   state s On c.state_code = s.state_code
    ```

- Example 2: List the order_num, order_date and the description of each product in the order 1002:

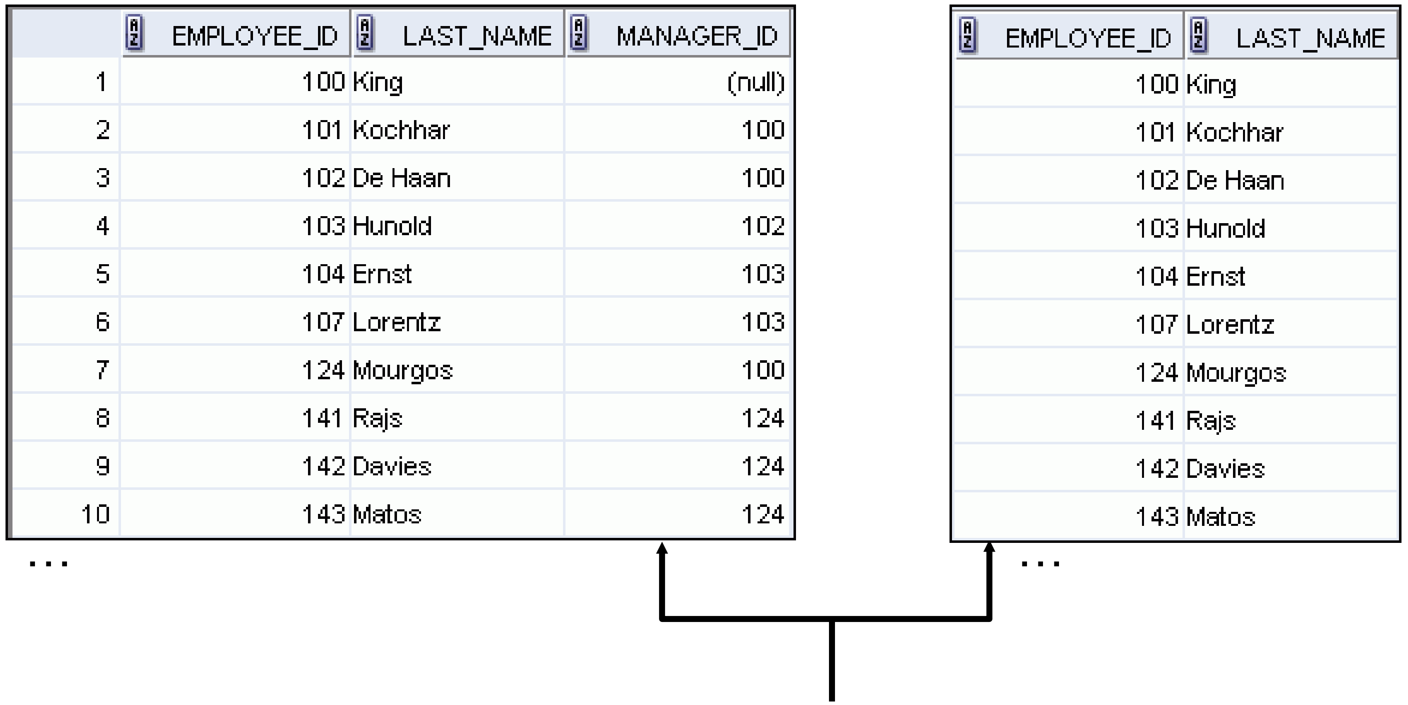
  -
    ```
    Select       o.order_num, order_date, prod_desc
    From         orders o
    Inner Join   order_detail od On o.order_num = od.order_num
    Inner Join   product_desc pd On od.prod_num = pd.prod_num
    Where        o.order_num = 1002;
    ```

Multiple Tables Queries

- Various forms of JOINs available:
  - INNER JOIN
  - OUTER JOIN – LEFT/ RIGHT/ FULL OUTER JOIN
  - SELF JOIN
  - CROSS JOIN – Cartesian product

- For simplicity, we will focus on **<u>INNER JOIN.</u>**

Self Join



EMPLOYEES (WORKER)

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124

…

EMPLOYEES (MANAGER)

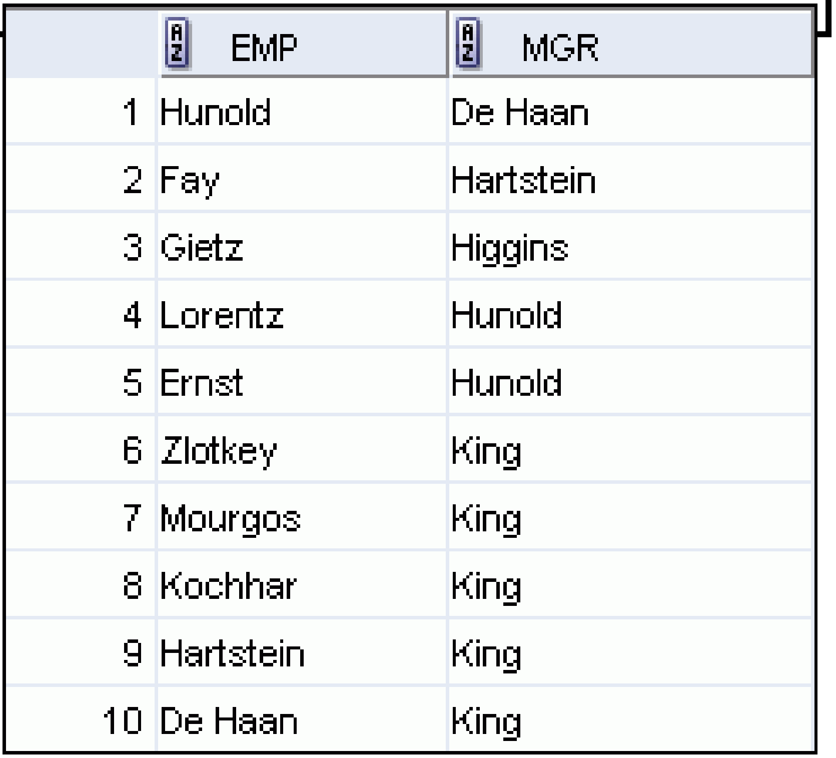
EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos
141	Rajs
142	Davies
143	Matos

…

MANAGER_ID in the WORKER table is equal to EMPLOYEE_ID in the MANAGER table.

Self join - Example

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON    (worker.manager_id = manager.employee_id);
```

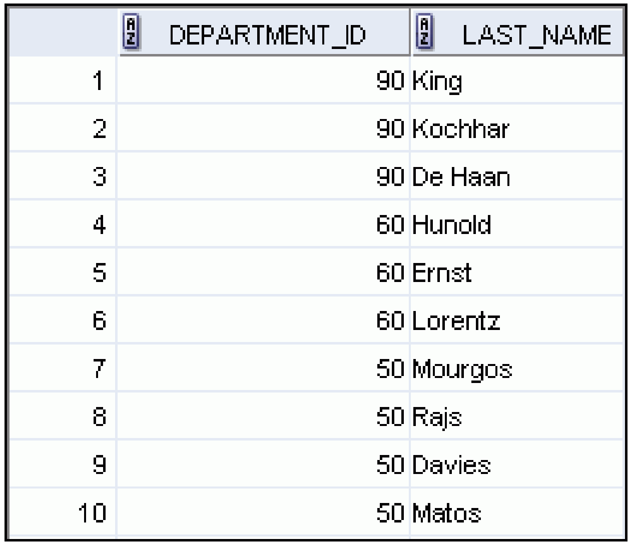


	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
8	Kochhar	King
9	Hartstein	King
10	De Haan	King

Outer Joins

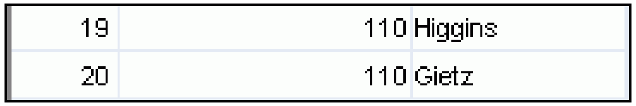
* Returning Records with No Direct Match

EMPLOYEES



	DEPARTMENT_ID	LAST_NAME
1	90	King
2	90	Kochhar
3	90	De Haan
4	60	Hunold
5	60	Ernst
6	60	Lorentz
7	50	Mourgos
8	50	Rajs
9	50	Davies
10	50	Matos

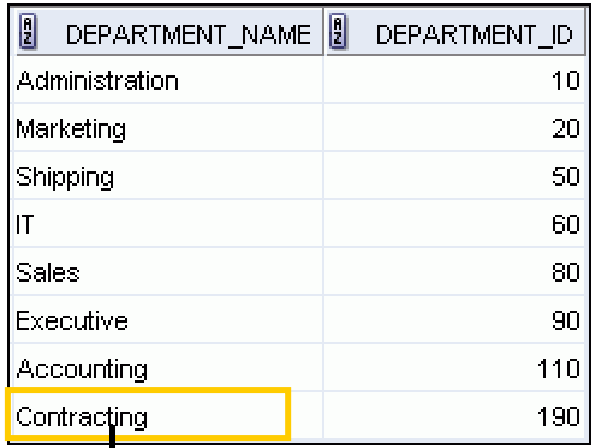
…		
	<null>	Grant



19	110	Higgins
20	110	Gietz

This employee does not have a department_ID

DEPARTMENTS

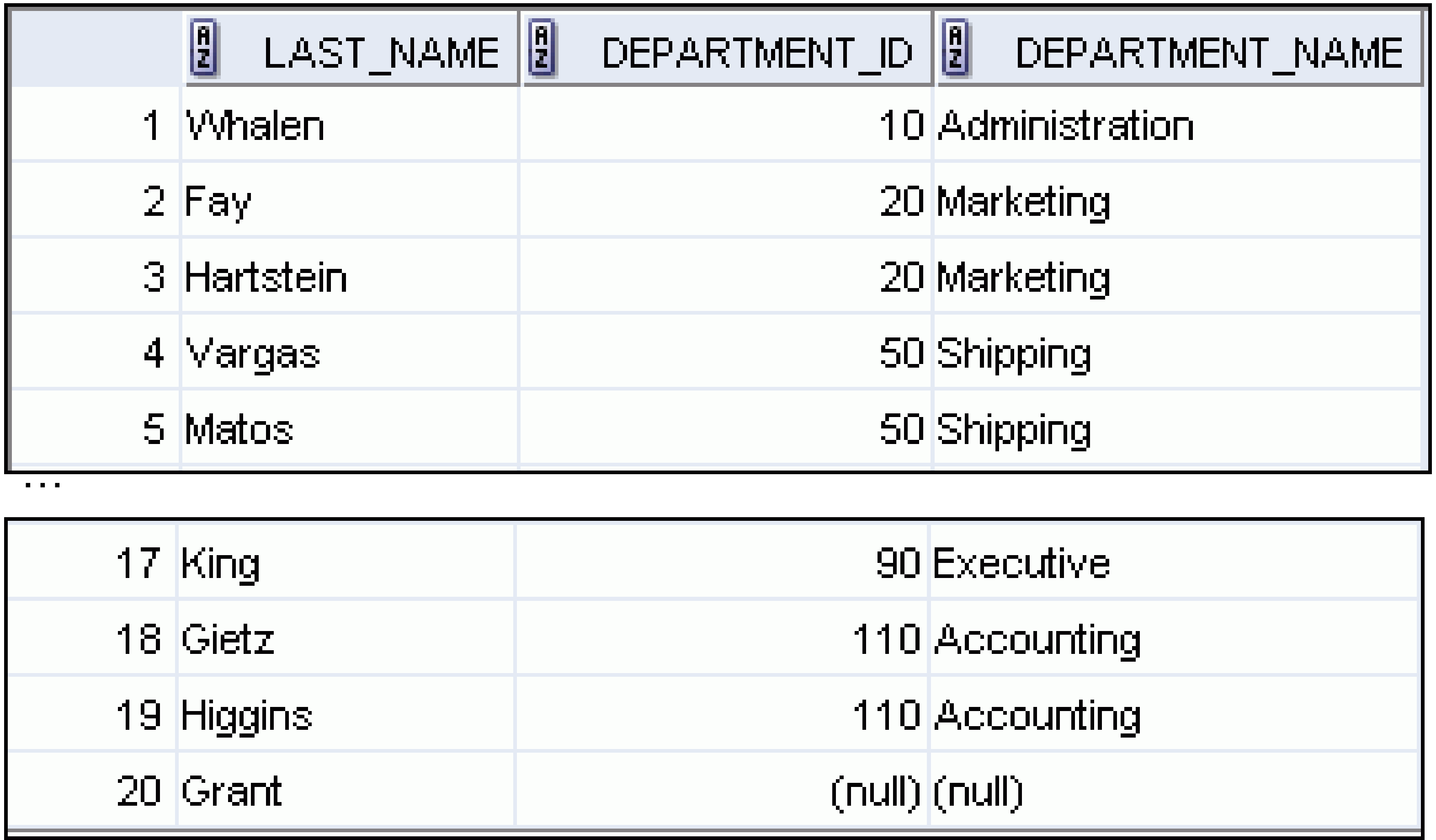


DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

There are no employees in department 190.

Left outer join

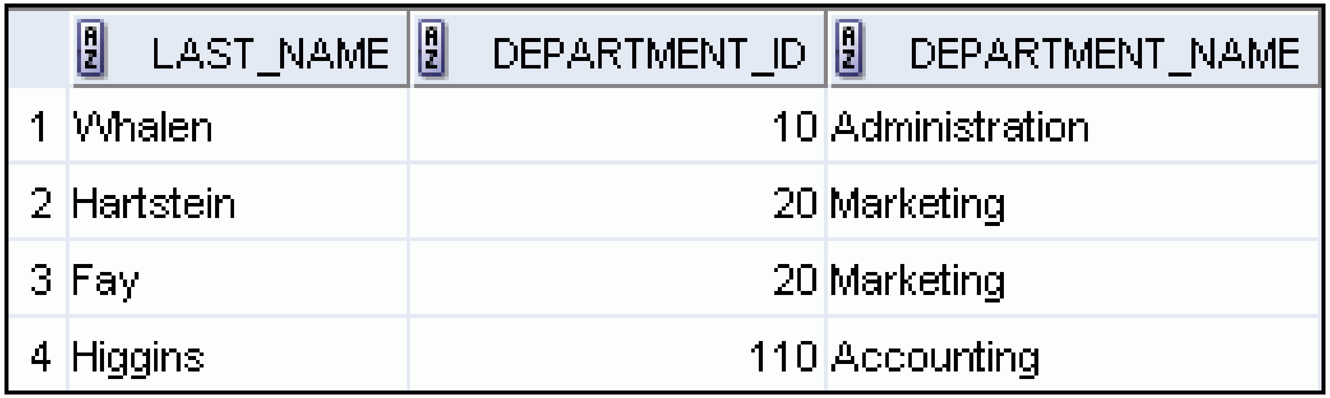
```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON   (e.department_id = d.department_id) ;
```



	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping
…			
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

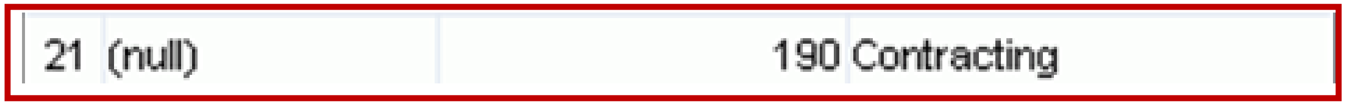
Right outer join

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```



	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

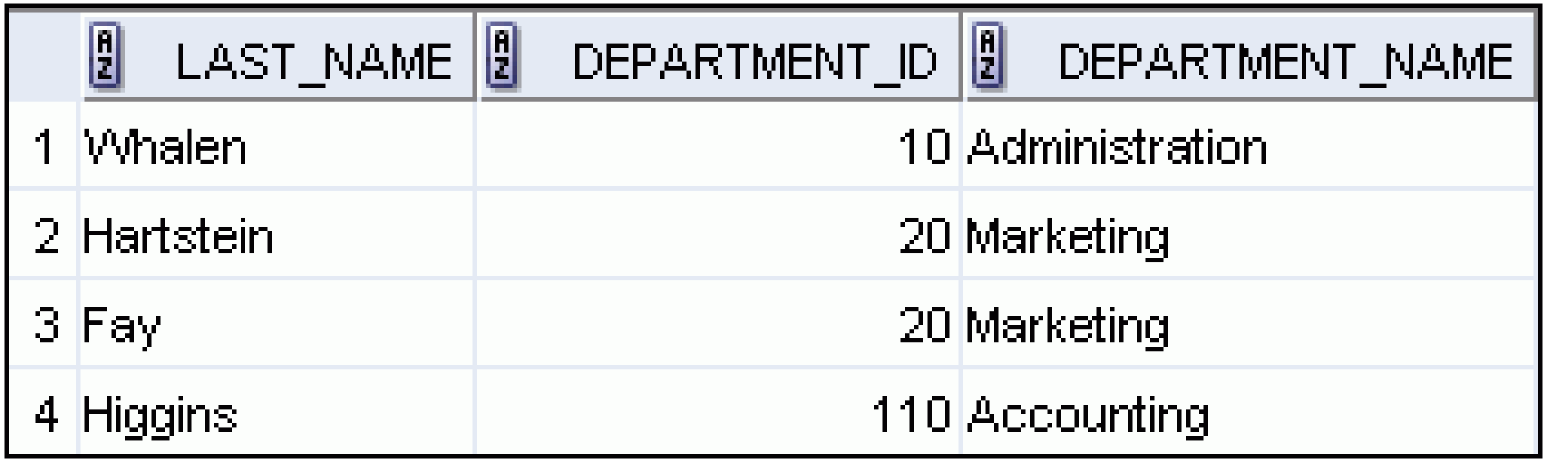
…



21	(null)	190	Contracting

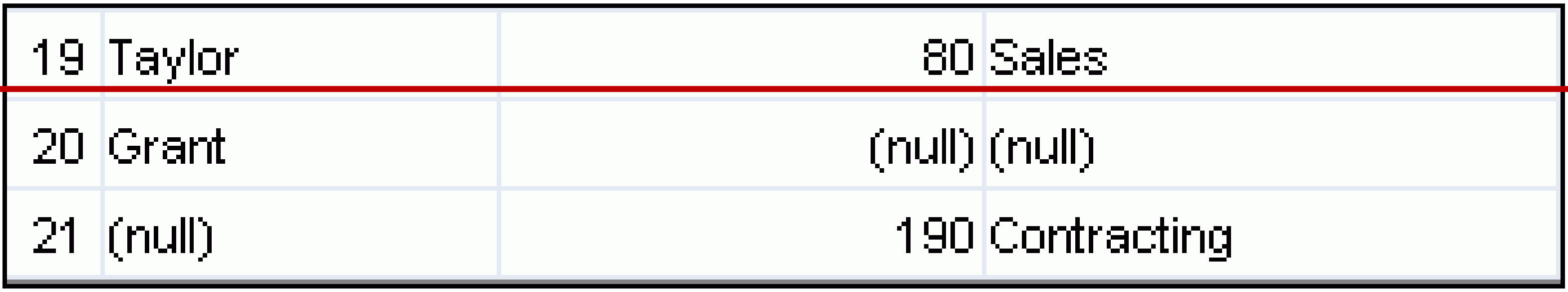
Full outer join

```
SELECT e.last_name, d.department_id, d.department_name
FROM  employees e FULL OUTER JOIN departments d
ON   (e.department_id = d.department_id) ;
```



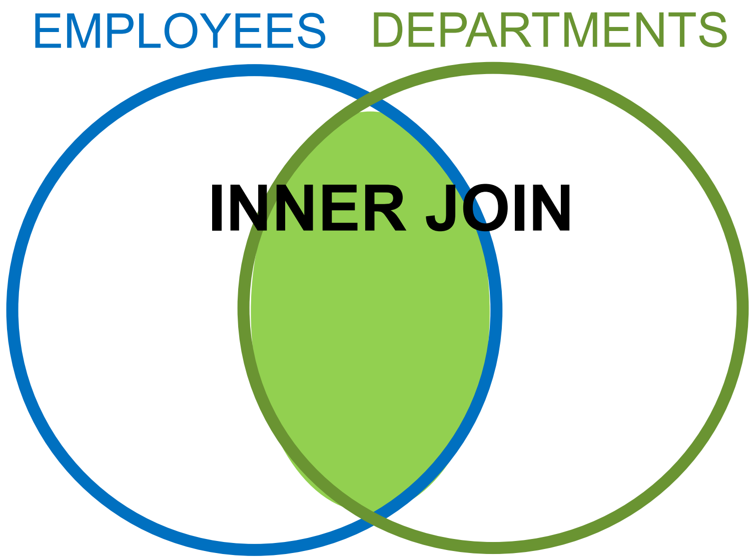
	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

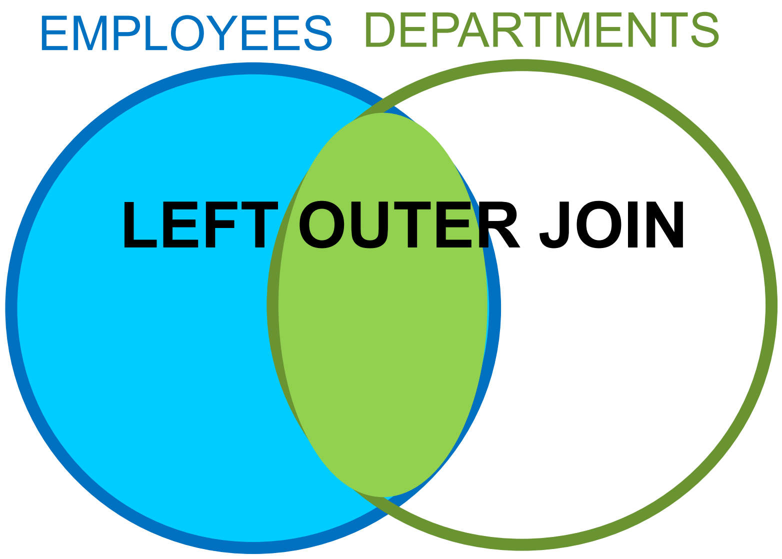
…

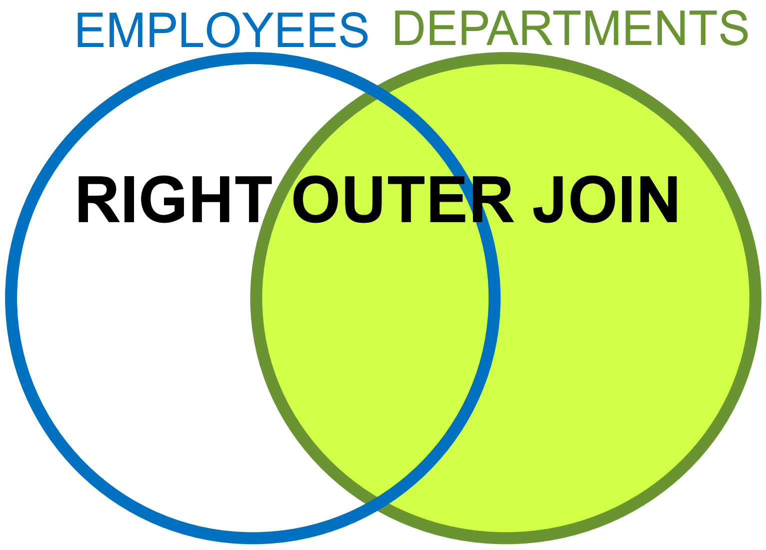


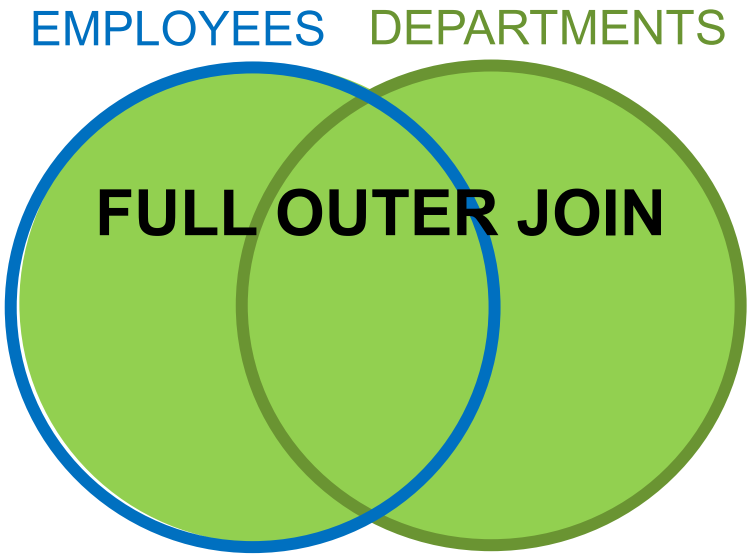
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

Inner Join vs. outer join









SELECT statement

SELECT	**Specifies which columns are to appear in output**
FROM	**Specifies table to be used**
{[INNER JOIN .. ON ..]}	**Specifies other table(s) to be joined. Repeats for each additional table.**
[WHERE]	**Filters rows**
[GROUP BY]	Forms groups of rows with same column value
[HAVING]	Filters groups subject to some condition
[ORDER BY]	**Specifies the order of the rows in the output**

Reference Materials, ELOs

- Reference text : Database Systems, Connolly
  - DML : Ch 6
  - DDL : Ch 7

Quiz

Match the function of the SELECT statement to the correct descriptions.

HAVING	Specifies the order of the output.
FROM	Filters the rows subject to some condition
ORDER BY	Forms groups of rows with the same column value.
SELECT	Specifies the table/s to be used.
GROUP BY	Specifies which columns are to appear in the output.
WHERE	Filters the groups subject to some condition.