

# Tutoriales Oracle SQL

---

**Formación Oracle SQL**

Bloque Oracle SQL Básico

---

## Destinatario(s)



## Historial

Versión	Fecha	Origen de la actualización	Redactado por	Validado por
1.0	27/03/2018			

# Índice

---

<b>1.</b>	<b>Diseño de bases de datos relacionales con Oracle</b>	<b>5</b>
1.1.	<b>Tutorial 1 - Investigar la base de datos y entorno de aplicación</b>	<b>5</b>
1.2.	<b>Tutorial 2 - Diseño de un diagrama de Entidad-Relación para el escenario de los núcleos geológicos</b>	<b>5</b>
1.3.	<b>Tutorial 3 - Global</b>	<b>7</b>
<b>2.</b>	<b>Recuperación de datos utilizando la sentencia SQL SELECT</b>	<b>8</b>
2.1.	<b>Tutorial 1 - Describiendo el esquema de HR</b>	<b>8</b>
2.2.	<b>Tutorial 2 - Contestando las primeras preguntas con SQL</b>	<b>10</b>
2.3.	<b>Tutorial 3 - Experimentar con las expresiones y la tabla DUAL</b>	<b>13</b>
2.4.	<b>Tutorial 4 - Global</b>	<b>16</b>
<b>3.</b>	<b>Restringiendo y ordenando datos</b>	<b>21</b>
3.1.	<b>Tutorial 1 - Utilizando el operador LIKE</b>	<b>21</b>
3.2.	<b>Tutorial 2 - Ordenando datos utilizando la cláusula ORDER BY</b>	<b>22</b>
3.3.	<b>Tutorial 3 - Utilizando la sustitución ampersand</b>	<b>24</b>
3.4.	<b>Tutorial 4 - Global</b>	<b>26</b>
<b>4.</b>	<b>Funciones de registro único</b>	<b>28</b>
4.1.	<b>Tutorial 1 - Utilizando funciones de conversión de mayúsculas/minúsculas</b>	<b>28</b>
4.2.	<b>Tutorial 2 - Uso de las funciones de manipulación de caracteres</b>	<b>30</b>
4.3.	<b>Tutorial 3 - Utilizando las funciones de fecha</b>	<b>31</b>
4.4.	<b>Tutorial 4 - Global</b>	<b>32</b>
<b>5.</b>	<b>Utilizando funciones de conversión y expresiones condicionales</b>	<b>34</b>
5.1.	<b>Tutorial 1 - Convirtiendo Fechas en Caracteres Utilizando la Función TO_CHAR</b>	<b>34</b>
5.2.	<b>Tutorial 2 - Utilizando NULLIF Y NVL2 Para Lógica Condicional Simple</b>	<b>36</b>
5.3.	<b>Tutorial 3 - Usando la función DECODE</b>	<b>38</b>
5.4.	<b>Tutorial 4 - Global</b>	<b>39</b>
<b>6.</b>	<b>Presentando datos agregados utilizando funciones agregadas</b>	<b>42</b>
6.1.	<b>Tutorial 1 - Utilizando las funciones agregadas</b>	<b>42</b>
6.2.	<b>Tutorial 2 - Agrupando datos basados en múltiples columnas</b>	<b>42</b>
6.3.	<b>Tutorial 3 - La cláusula HAVING</b>	<b>44</b>
6.4.	<b>Tutorial 4 - Global</b>	<b>45</b>
<b>7.</b>	<b>Visualización de datos de múltiples tablas</b>	<b>47</b>
7.1.	<b>Tutorial 1 - Utilizando NATURAL JOIN</b>	<b>47</b>
7.2.	<b>Tutorial 2 - Utilizando la cláusula NATURAL JOIN...ON</b>	<b>48</b>

---

7.3.	<b>Tutorial 3 - Realizando un SELF-JOIN</b>	50
7.4.	<b>Tutorial 4 - Realizando un OUTER JOIN</b>	51
7.5.	<b>Tutorial 5 - Realizando un CROSS JOIN</b>	52
7.6.	<b>Tutorial 6 - Global</b>	54
8.	<b>Utilizando subconsultas para resolver problemas</b>	55
8.1.	<b>Tutorial 1 - Tipos de subconsultas</b>	55
8.2.	<b>Tutorial 2 - Subconsultas mas complejas</b>	57
8.3.	<b>Tutorial 3 - Investigar los diferentes tipos de subconsultas</b>	59
8.4.	<b>Tutorial 4 - Crear una consulta que sea fiable y fácil de interpretar</b>	61
8.5.	<b>Tutorial 5 - Global</b>	64
9.	<b>Utilizando los operadores de Conjunto</b>	66
9.1.	<b>Tutorial 1 - Describir los operadores de conjunto</b>	66
9.2.	<b>Tutorial 2- Utilizando los Operadores Conjunto</b>	67
9.3.	<b>Tutorial 3 - Controlar el orden de los registros devueltos</b>	70
9.4.	<b>Tutorial 4 - Global</b>	74
10.	<b>Manipulando datos</b>	75
10.1.	<b>Tutorial 1 - Uso del comando INSERT</b>	75
10.2.	<b>Tutorial 2 - Utilizar el comando UPDATE</b>	77
10.3.	<b>Tutorial 3 - Uso del comando DELETE</b>	79
10.4.	<b>Tutorial 4 - Uso de comandos COMMIT y ROLLBACK</b>	80
10.5.	<b>Tutorial 5 - Global</b>	82
11.	<b>Utilizando sentencias DDL para crear y administrar tablas</b>	85
11.1.	<b>Tutorial 1 - Determinar qué objetos son accesibles desde la sesión de usuario</b>	85
11.2.	<b>Tutorial 2 - Investigar la Estructura de Tablas</b>	86
11.3.	<b>Tutorial 3 - Investigar los Tipos de Datos en el Esquema HR</b>	86
11.4.	<b>Tutorial 4 - Crear Tablas</b>	87
11.5.	<b>Tutorial 5 - Trabajar con Restricciones</b>	90
11.6.	<b>Tutorial 6 - Global</b>	91

# 1. Diseño de bases de datos relacionales con Oracle

## 1.1. Tutorial 1 - Investigar la base de datos y entorno de aplicación

Se trata de un ejercicio en papel, sin solución específica.

Identificar los procesos de usuario, los servidores de aplicación y los servidores de base de datos utilizado en el entorno que se está utilizando. Intentar averiguar dónde se está generando el SQL y dónde se está ejecutando. Se debe tener en cuenta que normalmente los procesos de usuario utilizados por los usuarios finales serán gráficos y pasarán con frecuencia por los servidores de aplicaciones; el personal de administración y desarrollo de bases de datos a menudo preferirá usar cliente-servidor que se conectan directamente al servidor de base de datos.

- [Grade essays](#)

## 1.2. Tutorial 2 - Diseño de un diagrama de Entidad-Relación para el escenario de los núcleos geológicos

En este ejercicio, se familiarizará con el modelado relacional básico al completar un diagrama de entidad-relación para el escenario de Núcleo Geológico introducido anteriormente. Para recapitular: Se han recolectado muestras de la tierra por la agencia geológica local. Para asegurar el rigor científico, los desarrolladores de GeoCore han determinado que el sistema debe rastrear la ubicación geográfica exacta, el contenido elemental de las muestras del núcleo y la fecha de recolección.

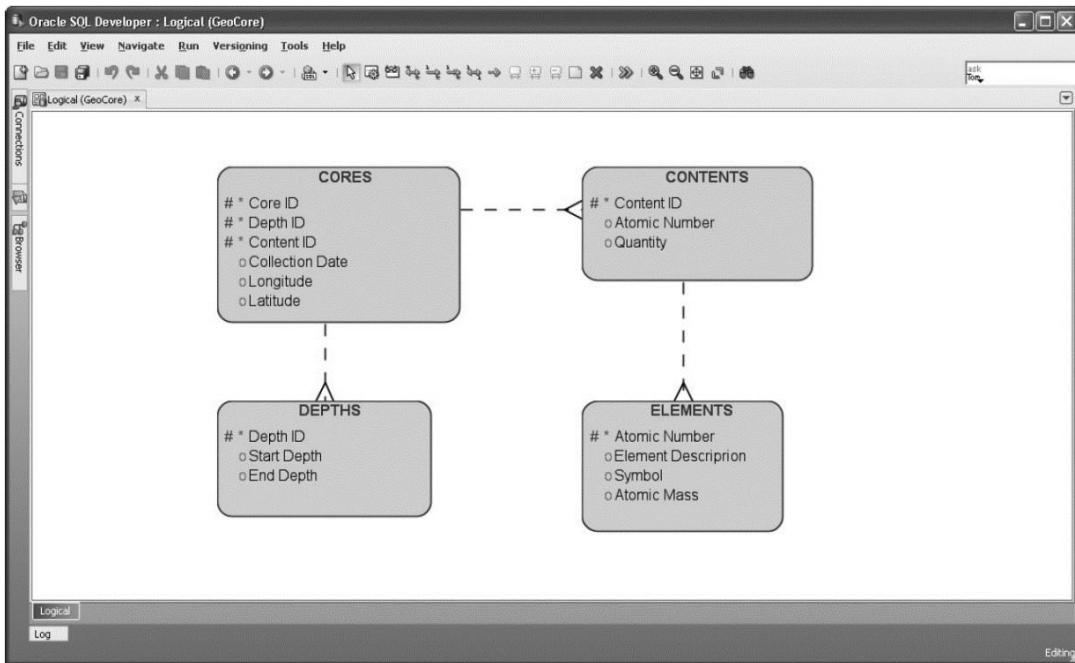
1. Los desarrolladores de GeoCore han propuesto cuatro entidades con sus respectivos atributos según la siguiente tabla. Se dibujará un diagrama con estas entidades reflejando las claves primarias.

Entidad	Atributo
Elements	Atomic Number (Primary Key)
	Element Description
	Symbol
	Atomic Mass
Contents	Content ID (Primary Key)
	Atomic Number

	Quantity
Depth	Depth ID (Primary Key)
	Start Depth
	End Depth
Cores	Core ID (Primary Key)
	Depth ID
	Collection Date
	Longitude
	Latitude
	Content ID

2. Las entidades Elements y Contents comparten una relación de many-to-one a través del atributo Atomic Number. La entidad Cores comparte una relación de many-to-one con la entidad Depth a través del atributo Depth ID y una relación de many-to-one con la entidad Contents a través del atributo Content ID.

El objetivo es actualizar el diagrama de entidad para reflejar estas relaciones. Su diagrama de Entidad-Relación completo debe estar muy cerca del que aparece en la siguiente ilustración.



### 1.3. Tutorial 3 - Global

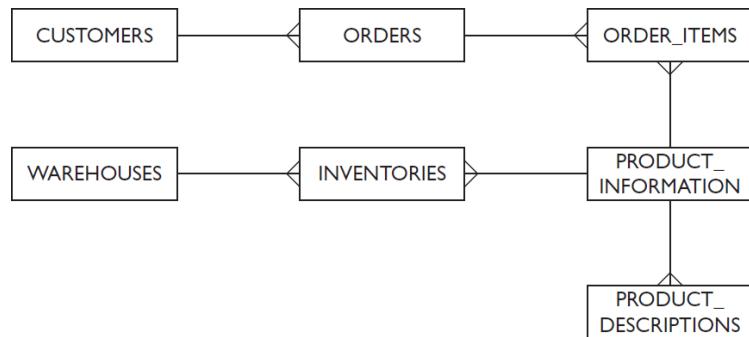
El esquema OE incluye estas tablas:

- CUSTOMERS
- INVENTORIES
- ORDERS
- ORDER\_ITEMS
- PRODUCT\_DESCRIPTIONS
- PRODUCT\_INFORMATION
- WAREHOUSES

Un CUSTOMER (cliente) puede realizar muchos ORDERS (pedidos), y un pedido puede tener muchos ORDER\_ITEMS (artículos del pedido). Cada ítem será un producto, descrito por su PRODUCT\_INFORMATION (información del producto), y cada producto tiene varias PRODUCT\_DESCRIPTIONS (descripciones del producto), en distintos idiomas. Hay un número de WAREHOUSES (almacenes de datos), cada uno puede almacenar muchos productos; un producto puede ser almacenado en muchos almacenes de datos. Una entrada INVENTORIES (inventarios) relaciona productos a almacenes de datos, mostrando cuantos de cada producto hay en cada almacén de datos.

Se requiere este esquema como un diagrama de relación de entidad, mostrando las relaciones many-to-one entre las tablas y asegurándose de que no haya relaciones many-to-many.

Un diagrama de entidad relación describiendo el esquema OE:



## 2. Recuperación de datos utilizando la sentencia SQL SELECT

### 2.1. Tutorial 1 - Describiendo el esquema de HR

El esquema HR contiene siete tablas que representan un modelo de datos de un departamento de Recursos Humanos ficticio.

- La tabla EMPLOYEES almacena los datos del personal.
- La tabla DEPARTMENTS contiene los datos de los departamentos de la organización.
- La tabla JOBS contiene los diferentes tipos de trabajo disponibles en la organización.
- La tabla JOB\_HISTORY contiene un seguimiento de los detalles del trabajo de los empleados que cambiaron de trabajo, pero permanecieron a la organización.
- También están las tablas LOCATIONS, COUNTRIES y REGIONS, que contienen la información geográfica relativa a la organización.

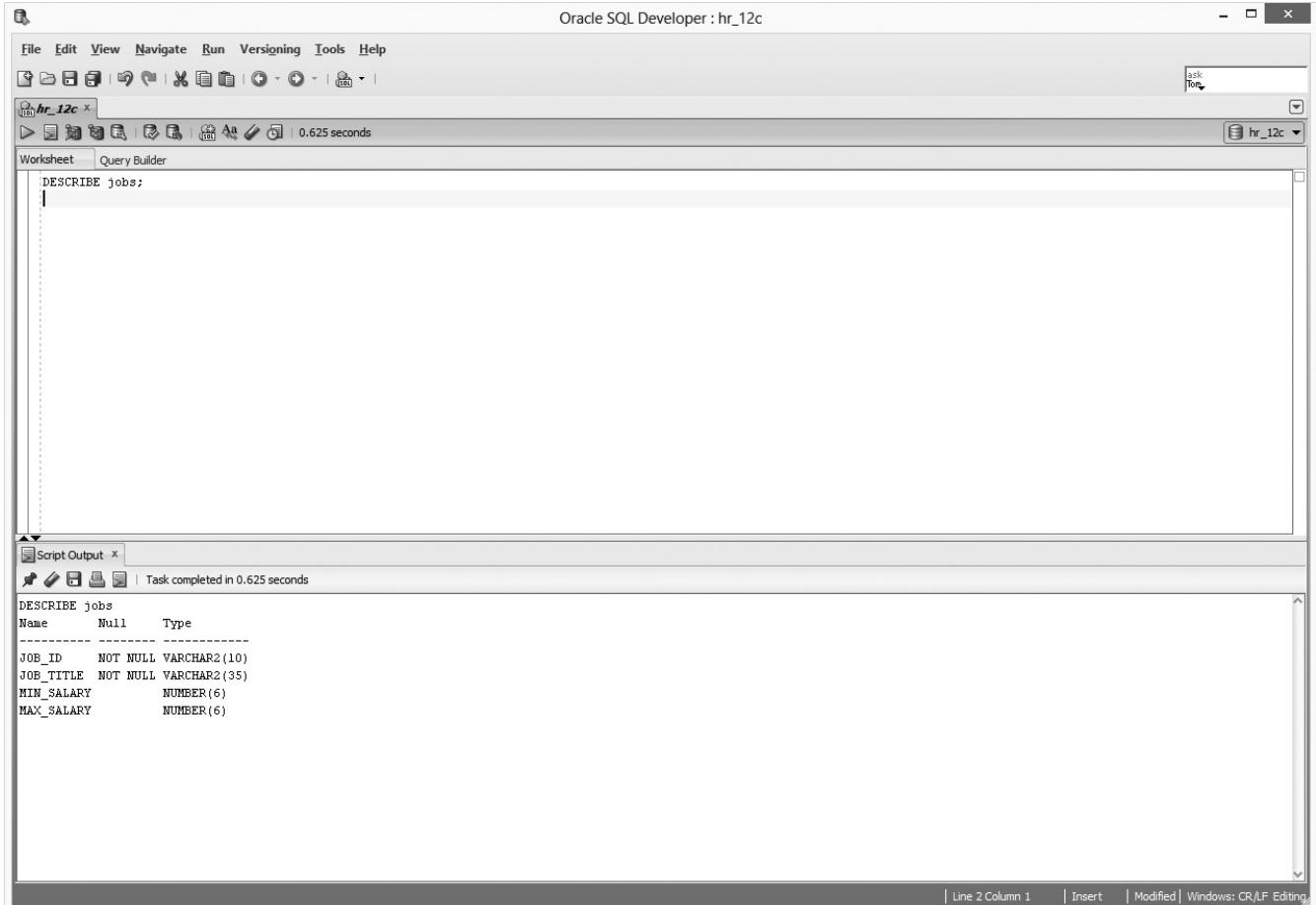
En este tutorial se pretende explorar algunas de estas tablas, y para ello se seguirán los siguientes pasos:

1. Iniciar SQL Developer y elegir Nuevo en el menú Archivo. Seleccionar Conexión de base de datos. Si esta es la primera vez que se conecta a la base de datos desde SQL Developer, se debe crear una conexión. Se deberá proporcionar un nombre de conexión descriptivo y HR como nombre de usuario. Los detalles de conexión restantes deben obtenerse del administrador de la base de datos. Una vez se guarde la conexión, se deberá hacer clic en el botón Conectar.
2. Navegar hasta la hoja de trabajo SQL, que es la pestaña titulada Hoja de Trabajo.
3. Escribir el comando:

```
DESCRIBE jobs
```

La terminación de este comando con punto y coma es opcional.

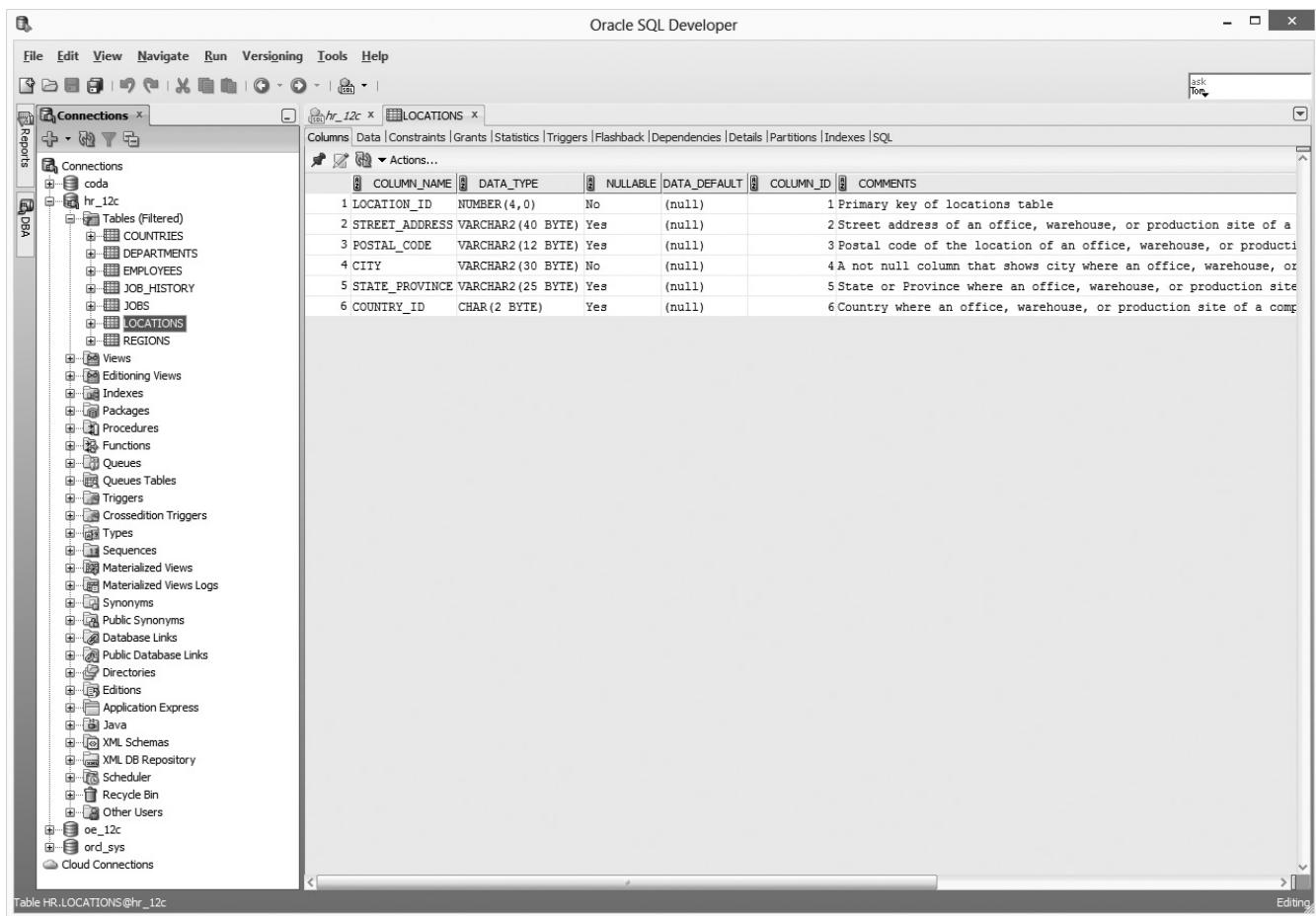
4. Ejecutar el comando DESCRIBE, ya sea presionando la tecla f5 o haciendo clic en el ícono de la flecha triangular verde ubicada en la barra de herramientas encima del Editor SQL.
5. La descripción de la tabla JOBS aparece en el marco Resultados como se muestra en la siguiente imagen.



The screenshot shows the Oracle SQL Developer interface. The top window is titled "Oracle SQL Developer : hr\_12c". The "Worksheet" tab is active, displaying the SQL command "DESCRIBE jobs;". Below it, the "Script Output" tab shows the results of the command:

```
Task completed in 0.625 seconds
DESCRIBE jobs
Name      Null    Type
-----  -----
JOB_ID    NOT NULL VARCHAR2(10)
JOB_TITLE NOT NULL VARCHAR2(35)
MIN_SALARY      NUMBER(6)
MAX_SALARY      NUMBER(6)
```

6. Los pasos 3 a 5 se pueden repetir para mostrar las tablas JOB\_HISTORY, LOCATIONS, COUNTRIES, y REGIONS.
7. *SQL Developer proporciona una alternativa al comando DESCRIBE a la hora de obtener la información estructural de las tablas.*
8. Navegar a la tabla LOCATIONS usando el Navegador de Árbol ubicado en el marco izquierdo debajo del nombre de la conexión.
9. SQL Developer describe la tabla automáticamente en el lado derecho de la herramienta como se muestra en la siguiente ilustración.



## 2.2. Tutorial 2 - Contestando las primeras preguntas con SQL

En este tutorial paso a paso, se establecerá una conexión utilizando SQL \* Plus como el usuario HR para responder dos preguntas utilizando la instrucción SELECT.

**Pregunta 1: ¿Cuántos departamentos únicos tienen empleados trabajando actualmente en ellos?**

1. Se inicia SQL \* Plus y se conecta al esquema HR.
2. Inicialmente, existe la posibilidad de poder encontrar la respuesta en las tablas DEPARTMENTS. Un examen cuidadoso revela que la pregunta requiere información sobre empleados. Esta información está contenida en la tabla EMPLOYEES.
3. El enunciado pide obtener los departamentos **únicos**, por lo que esta palabra podría servir como pista para saber que hay que hacer uso de DISTINCT.
4. Combinando los pasos 2 y 3, se puede construir la siguiente instrucción SQL:

```
SELECT distinct department_id
FROM employees;
```

5. Como se muestra en la imagen, esta consulta devuelve 12 registros. Se observa que el tercer registro está vacío. Este es un valor nulo en la columna DEPARTMENT\_ID.



```
SQL> SELECT distinct department_id
  2  FROM employees;

DEPARTMENT_ID
-----
100
30

90
20
70
110
50
80
40
60
10

12 rows selected.

SQL>
```

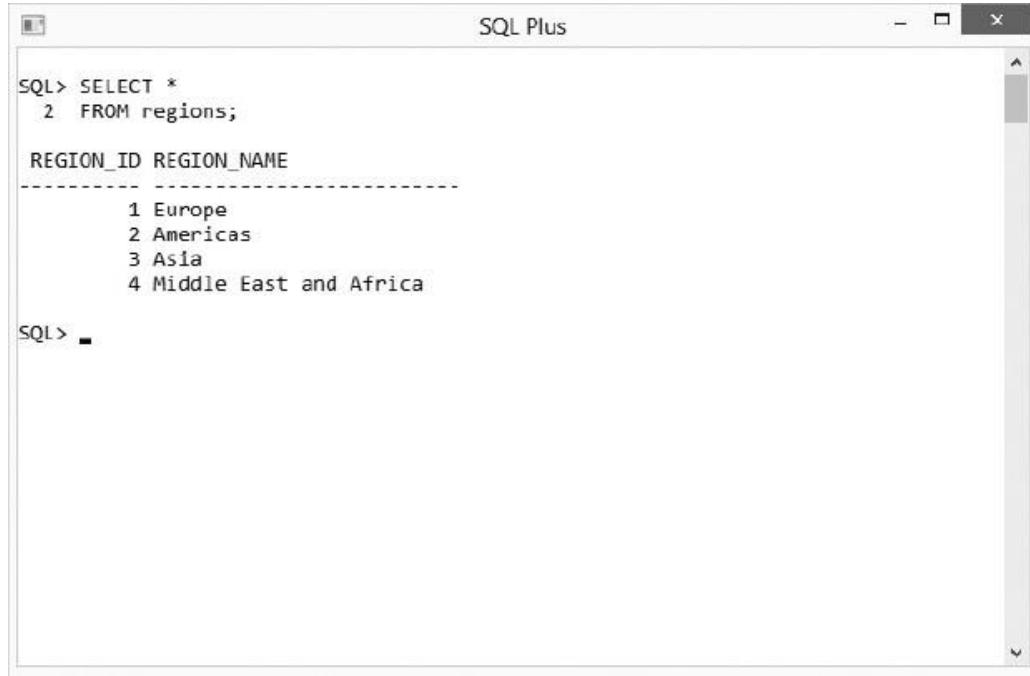
La respuesta a la primera pregunta es por lo tanto: once departamentos únicos tienen empleados trabajando en ellos, pero al menos un empleado no ha sido asignado a un departamento.

### Pregunta 2: ¿Cuántos países hay en Europa?

1. Esta pregunta comprende dos partes. Se considera la tabla REGIONS, que contiene cuatro regiones identificadas de forma única por un valor REGION\_ID, y la tabla COUNTRIES, que tiene una columna REGION\_ID que indica la región a la que pertenece un país.
2. La primera consulta debe identificar el REGION\_ID de la región de Europa. Esta se logra mediante la consulta de SQL:

```
SELECT * FROM regions;
```

3. La siguiente ilustración muestra que la región de Europa tiene un REGION\_ID de valor 1



The screenshot shows an Oracle SQL Plus window titled "SQL Plus". The query executed is:

```
SQL> SELECT *
  2  FROM regions;
```

The output displays the following data:

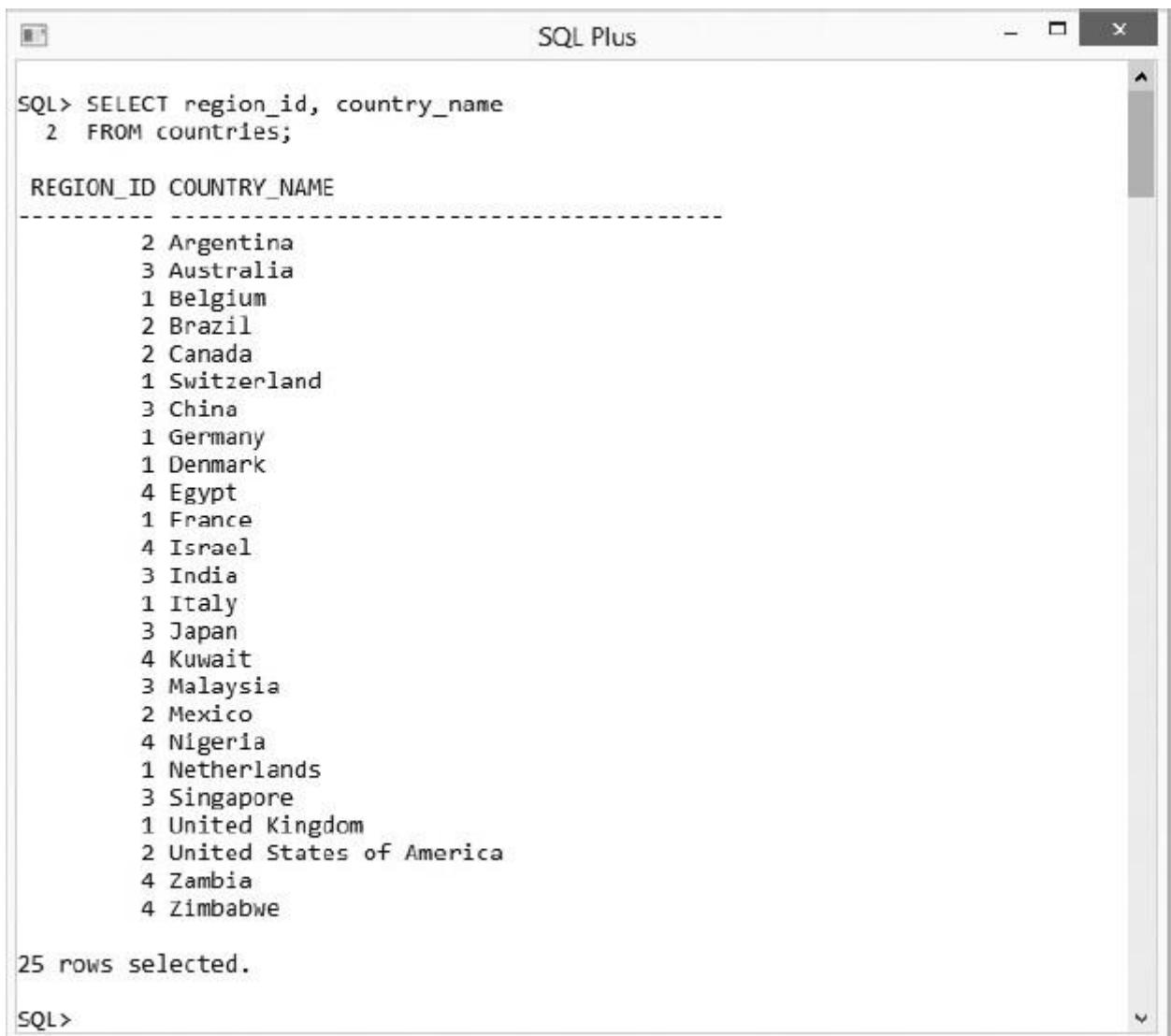
REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

SQL> ■

4. Para identificar qué países tienen 1 como su REGION\_ID, se debe ejecutar la siguiente consulta SQL:

```
SELECT region_id, country_name FROM countries;
```

5. Contando manualmente las filas del país con un REGION\_ID de 1, en la siguiente imagen se puede a responder la segunda pregunta:



The screenshot shows a window titled "SQL Plus". Inside, a SQL command is executed:

```
SQL> SELECT region_id, country_name
  2  FROM countries;
```

The output displays the results of the query:

REGION_ID	COUNTRY_NAME
2	Argentina
3	Australia
1	Belgium
2	Brazil
2	Canada
1	Switzerland
3	China
1	Germany
1	Denmark
4	Egypt
1	France
4	Israel
3	India
1	Italy
3	Japan
4	Kuwait
3	Malaysia
2	Mexico
4	Nigeria
1	Netherlands
3	Singapore
1	United Kingdom
2	United States of America
4	Zambia
4	Zimbabwe

Below the table, the message "25 rows selected." is displayed. At the bottom of the window, there is a prompt "SQL>".

La respuesta a la segunda pregunta es, por lo tanto: hay ocho países en la región de Europa en lo que respecta al modelo de datos HR.

### 2.3. Tutorial 3 - Experimentar con las expresiones y la tabla DUAL

En este tutorial paso a paso, se establecerá una conexión utilizando SQL Developer como el usuario HR. Hay que utilizar expresiones y operadores para responder a tres preguntas relacionadas con la sentencia SELECT:

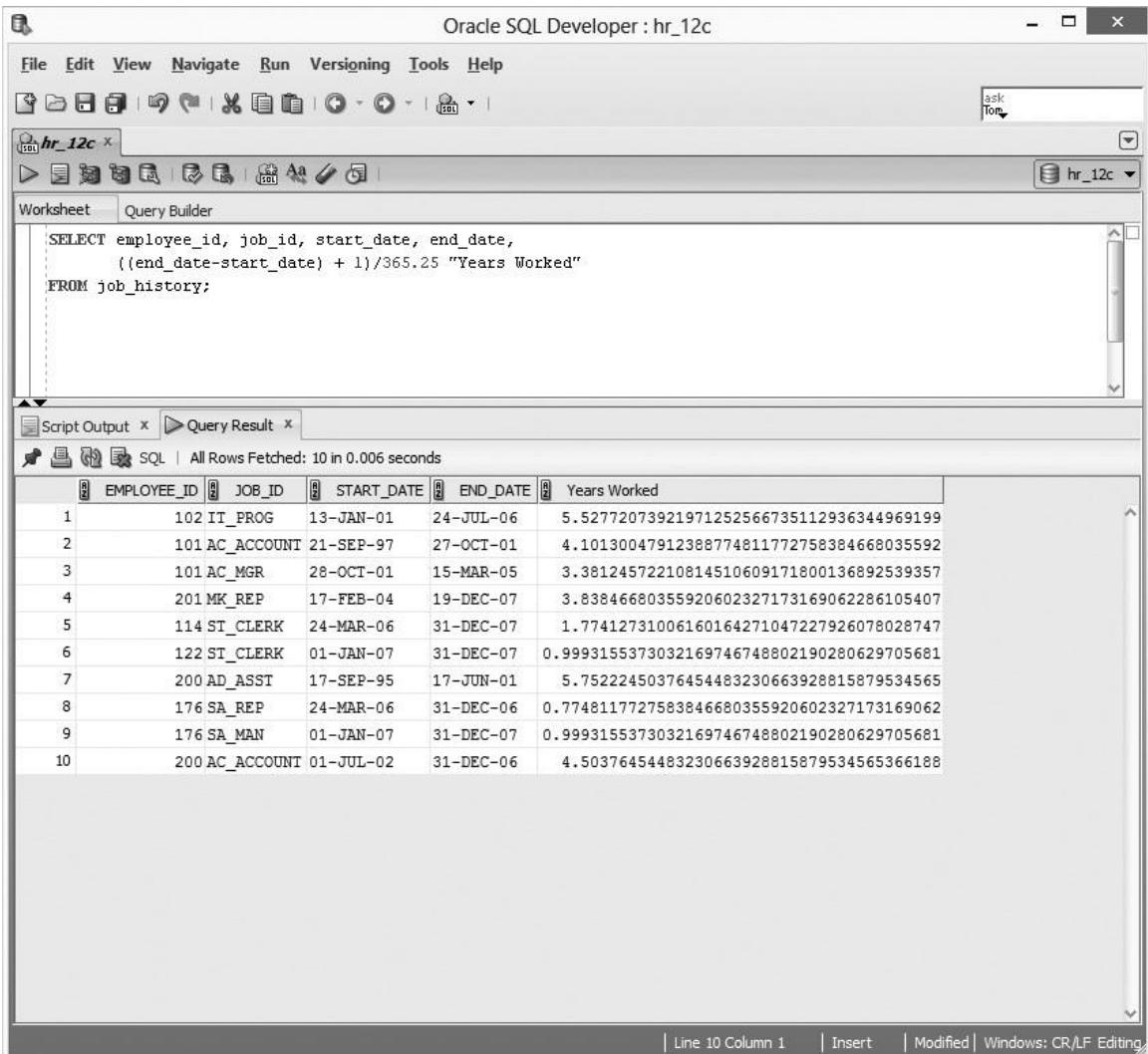
**Pregunta 1: ¿Durante cuántos años ha estado un trabajador ejerciendo cierto puesto de trabajo y cuáles son su EMPLOYEE\_ID, JOB\_ID, START\_DATE y END\_DATE? Se deberá asignar el alias Years Worked a la expresión que defina la columna. Asúmase que un año tiene 365.25 días.**

Los pasos a seguir serán los siguientes:

1. Iniciar SQL Developer y conectarse al esquema HR.
2. La proyección de las columnas necesarias incluye EMPLOYEE\_ID, JOB\_ID, START\_DATE, END\_DATE, y una expresión llamada Years Worked de la tabla JOB\_HISTORY.
3. La expresión se puede calcular dividiendo la diferencia entre END\_DATE y START\_DATE más 1 entre 365.25 días, como se muestra a continuación:

```
SELECT employee_id, job_id, start_date, end_date, ((end_date-start_date) + 1)/365.25 "YearsWorked"
FROM job_history;
```

Al ejecutar la instrucción SELECT anterior se obtienen los resultados que se muestran en la siguiente imagen:



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : hr\_12c". The menu bar includes File, Edit, View, Navigate, Run, Versioning, Tools, and Help. The toolbar has various icons for file operations like Open, Save, and Print. A search bar at the top right contains the text "ask Tom". Below the toolbar is a tab bar with "hr\_12c" selected. The main workspace is divided into two panes: "Worksheet" and "Query Result". The "Worksheet" pane contains the SQL query:

```
SELECT employee_id, job_id, start_date, end_date,
       ((end_date-start_date) + 1)/365.25 "Years Worked"
FROM job_history;
```

The "Query Result" pane displays the output of the query, which is a table with 10 rows of data:

	EMPLOYEE_ID	JOB_ID	START_DATE	END_DATE	Years Worked
1	102	IT_PROG	13-JAN-01	24-JUL-06	5.52772073921971252566735112936344969199
2	101	AC_ACCOUNT	21-SEP-97	27-OCT-01	4.10130047912388774811772758384668035592
3	101	AC_MGR	28-OCT-01	15-MAR-05	3.38124572210814510609171800136892539357
4	201	MK_REP	17-FEB-04	19-DEC-07	3.83846680355920602327173169062286105407
5	114	ST_CLERK	24-MAR-06	31-DEC-07	1.77412731006160164271047227926078028747
6	122	ST_CLERK	01-JAN-07	31-DEC-07	0.9993155373032169746748802190280629705681
7	200	AD_ASST	17-SEP-95	17-JUN-01	5.75222450376454483230663928815879534565
8	176	SA REP	24-MAR-06	31-DEC-06	0.7748117727583846680355920602327173169062
9	176	SA MAN	01-JAN-07	31-DEC-07	0.9993155373032169746748802190280629705681
10	200	AC_ACCOUNT	01-JUL-02	31-DEC-06	4.50376454483230663928815879534565366188

At the bottom of the "Query Result" pane, there are status messages: "Line 10 Column 1 | Insert | Modified | Windows: CR/LF Editing".

**Pregunta 2: Se deberá consultar la tabla JOBS de forma que se obtenga un resultado con el formato siguiente: "The Job Id for the <job\_title's> job is: <job\_id>".**

Hay que tener en cuenta que el job\_title debe tener un apóstrofo y una letra "s" puesto que el texto devuelto está en inglés. Una muestra de este resultado para el presidente de la organización es:

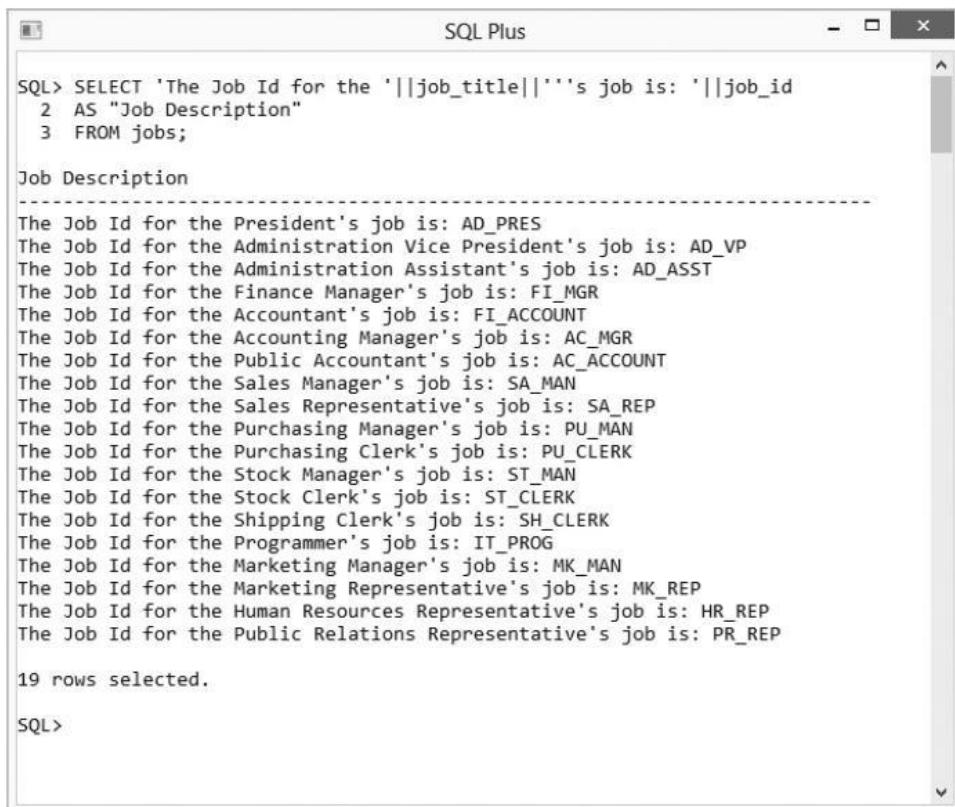
"The Job Id for the President's job is: AD\_PRES".

Se debe asignar el alias "Job Description" a la expresión que nos permita obtener este resultado usando la palabra clave AS.

1. Hay múltiples soluciones a este problema. El enfoque elegido aquí es manejar las comillas simples naturales con una comilla simple adicional.
2. Se requiere una expresión con alias Job Description. Este alias lo constituye la cadena de caracteres "The Job Id for the" concatenada con la columna JOB\_TITLE. Esta cadena se concatena entonces con "'s job is: " que se concatena con la columna JOB\_ID. Se agrega una comilla adicional para obtener la instrucción SELECT que se muestra a continuación:

```
SELECT 'The Job Id for the'||job_title||''s job is: ||job_id
AS "Job Description"
FROM jobs;
```

Los resultados de esta consulta SQL se muestran en la siguiente imagen:



The screenshot shows an Oracle SQL Plus window with the following content:

```
SQL> SELECT 'The Job Id for the'||job_title||''s job is: ||job_id
  2  AS "Job Description"
  3  FROM jobs;

Job Description
-----
The Job Id for the President's job is: AD_PRES
The Job Id for the Administration Vice President's job is: AD_VP
The Job Id for the Administration Assistant's job is: AD_ASST
The Job Id for the Finance Manager's job is: FI_MGR
The Job Id for the Accountant's job is: FI_ACCOUNT
The Job Id for the Accounting Manager's job is: AC_MGR
The Job Id for the Public Accountant's job is: AC_ACCOUNT
The Job Id for the Sales Manager's job is: SA_MAN
The Job Id for the Sales Representative's job is: SA_REP
The Job Id for the Purchasing Manager's job is: PU_MAN
The Job Id for the Purchasing Clerk's job is: PU_CLERK
The Job Id for the Stock Manager's job is: ST_MAN
The Job Id for the Stock Clerk's job is: ST_CLERK
The Job Id for the Shipping Clerk's job is: SH_CLERK
The Job Id for the Programmer's job is: IT_PROG
The Job Id for the Marketing Manager's job is: MK_MAN
The Job Id for the Marketing Representative's job is: MK_REP
The Job Id for the Human Resources Representative's job is: HR_REP
The Job Id for the Public Relations Representative's job is: PR_REP

19 rows selected.

SQL>
```

**Pregunta 3: Utilizando la tabla DUAL, hay que calcular el área de un círculo con un radio de 6.00 unidades, siendo pi aproximadamente 22/7. Se usará la fórmula: Área = pi × radio × radio. Se debe asignar un alias "Area" al resultado.**

1. Trabajar con la tabla DUAL puede parecer inicialmente curioso. A medida que se va usando, su funcionalidad se hace más evidente. Esta pregunta implica seleccionar una expresión aritmética literal de la tabla para obtener una respuesta calculada en un solo registro que no se base en los valores de las columnas de ninguna tabla.
2. La expresión puede calcularse utilizando la siguiente sentencia SQL (hay que tener en cuenta el uso de paréntesis para la prioridad de los operandos):

```
SELECT (22/7) * (6000 * 6000) Area
FROM dual;
```

Los resultados devueltos muestran el área aproximada del círculo como 113,142,857 unidades cuadradas.

## 2.4. Tutorial 4 - Global

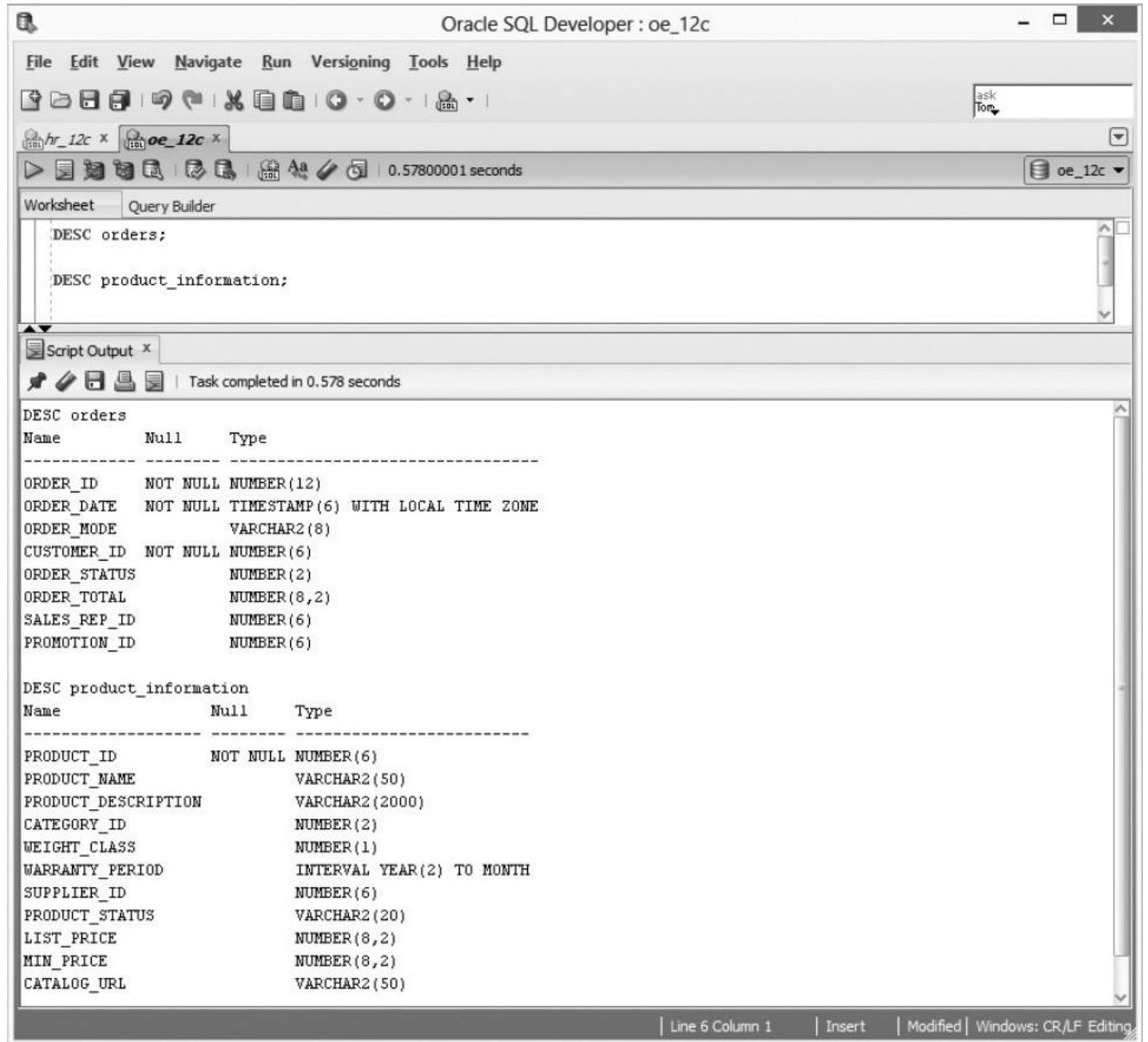
En este capítulo se trabajaron ejemplos en el esquema HR. Oracle proporciona una serie de esquemas de ejemplo con los que se puede experimentar y aprender diferentes conceptos. En este tutorial se utilizará el esquema de **Order Entry** (Entrada de pedidos), u OE. Las soluciones para estos tutoriales globales se proporcionarán más adelante utilizando SQL Developer. Utilizando SQL Developer o SQL\*Plus. Es necesario completar las siguientes tareas.

1. Obtener información estructural para las tablas PRODUCT\_INFORMATION y ORDERS.
2. Seleccionar los valores únicos SALES\_REP\_ID de la tabla ORDERS. ¿Cuántos representantes de ventas diferentes se han asignado a los pedidos en la tabla ORDERS?
3. Devolver un conjunto de resultados basado en la tabla ORDERS que incluya las columnas ORDER\_ID, ORDER\_DATE y ORDER\_TOTAL. Observar cómo se formatea la salida ORDER\_DATE de forma diferente a las columnas START\_DATE y END\_DATE en la tabla HR.JOB\_HISTORY.
4. La tabla PRODUCT\_INFORMATION almacena los datos relativos a los productos disponibles para la venta en tienda de productos hardware de informática ficticia. Se deberá extraer la información del producto en el formato < PRODUCT\_NAME > with code: <PRODUCT\_ID> has status of: < PRODUCT\_STATUS >. Alias la expresión " Product ". Los resultados deben proporcionar el LIST\_PRICE, el MIN\_PRICE, la diferencia entre LIST\_PRICE, y el MIN\_PRICE alias "Max Actual Savings", junto con una expresión adicional que es la diferencia entre LIST\_PRICE y MIN\_PRICE y la división por el LIST\_PRICE y multiplica por 100 el total. Esta última expresión debe escribirse como "Max Discount %".
5. Calcular la superficie de la tierra utilizando la tabla DUAL. Ponerle el alias: "Earth's Area". La fórmula para calcular el área de una esfera es:  $4\pi r^2$ . Suponer, por ejemplo, que la tierra es una esfera simple con un radio de 3,958.759 millas y que  $\pi$  es 22/7.

## Solución:

Se asume que una base de datos Oracle está disponible para poder practicar.

1. El comando DESCRIBE nos da la descripción estructural de una tabla. La siguiente ilustración muestra la descripción de estas dos tablas:



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : oe\_12c". The menu bar includes File, Edit, View, Navigate, Run, Versioning, Tools, and Help. The toolbar has various icons for file operations like Open, Save, Print, and Run. The connection status shows "oe\_12c" is connected. The main area has tabs for Worksheet and Query Builder, with "Worksheet" selected. The worksheet contains the following SQL code:

```
DESC orders;
DESC product_information;
```

Below the worksheet, the "Script Output" tab shows the results of the DESCRIBE commands:

```
DESC orders
Name      Null      Type
-----
ORDER_ID  NOT NULL NUMBER(12)
ORDER_DATE NOT NULL TIMESTAMP(6) WITH LOCAL TIME ZONE
ORDER_MODE  VARCHAR2(8)
CUSTOMER_ID NOT NULL NUMBER(6)
ORDER_STATUS    NUMBER(2)
ORDER_TOTAL     NUMBER(8,2)
SALES REP_ID   NUMBER(6)
PROMOTION_ID   NUMBER(6)

DESC product_information
Name      Null      Type
-----
PRODUCT_ID    NOT NULL NUMBER(6)
PRODUCT_NAME   VARCHAR2(50)
PRODUCT_DESCRIPTION  VARCHAR2(2000)
CATEGORY_ID    NUMBER(2)
WEIGHT_CLASS    NUMBER(1)
WARRANTY_PERIOD INTERVAL YEAR(2) TO MONTH
SUPPLIER_ID    NUMBER(6)
PRODUCT_STATUS  VARCHAR2(20)
LIST_PRICE      NUMBER(8,2)
MIN_PRICE       NUMBER(8,2)
CATALOG_URL    VARCHAR2(50)
```

2. La solicitud de valores únicos normalmente implica el uso de la palabra clave DISTINCT como parte de la instrucción SELECT. Los dos componentes de la declaración incluyen la cláusula SELECT y la cláusula FROM. Se pidieron valores SALES REP\_ID únicos de la tabla ORDERS. Es sencillo traducir esta solicitud en la siguiente instrucción SELECT:

```
SELECT DISTINCT sales_rep_id FROM orders;
```

A partir de los resultados de la figura, se puede responder a la pregunta original: Hay nueve representantes de ventas diferentes responsables de los pedidos listados en la tabla ORDERS, pero hay por lo menos un pedido que contiene valores nulos en sus campos SALESREP\_ID.

SALES REP ID
153
(null)
155
161
163
154
158
156
159
160

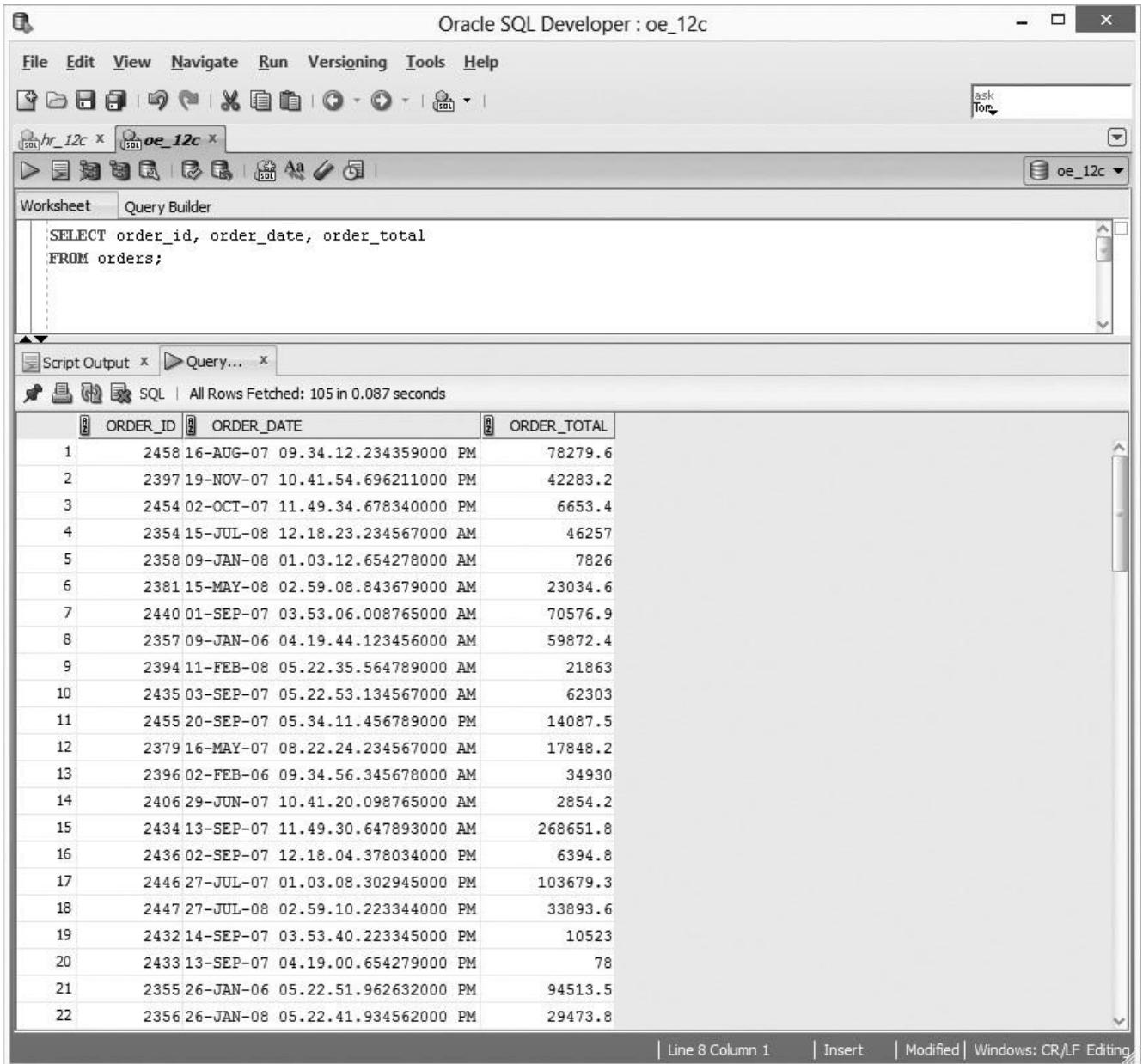
3. Cuando se pide que se cree un conjunto de resultados, se traduce a "SELECT una o más columnas de una tabla". En este caso, la cláusula SELECT se construye a partir de las tres columnas solicitadas. No hay petición de valores únicos, así que no hay necesidad de poner la palabra clave DISTINCT. La cláusula FROM sólo necesita incluir la tabla ORDERS para construir la siguiente sentencia SELECT:

```
SELECT order_id, order_date, order_total FROM orders;
```

Se puede observar el resultado en la siguiente imagen, específicamente la columna ORDER\_DATE. Esta columna contiene el día, el mes, el año, las horas, los minutos, los segundos y los segundos fraccionarios con hasta seis decimales o con una precisión de hasta una millonésima de segundo. La descripción de la tabla ORDERS expone ORDER\_DATE como un TIMESTAMP(6) con la columna LOCAL TIMEZONE. Esto significa que los datos de esta columna pueden almacenarse con una precisión fraccionaria de hasta seis

decimales y que los datos son sensibles a la zona horaria. Básicamente, los datos pueden ser usados por personas en diferentes zonas horarias. Por lo tanto, Oracle proporciona un tipo de datos que normaliza la hora local a la zona horaria de la base de datos para evitar confusiones.

En comparación con las columnas START\_DATE y END\_DATE de la tabla HR.JOB\_HISTORY, el tipo de datos de la columna ORDER\_DATE es mucho más sofisticado. Sin embargo, ambos tipos de datos almacenan información de fecha y hora, pero con diferentes grados de precisión.



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : oe\_12c". The menu bar includes File, Edit, View, Navigate, Run, Versioning, Tools, and Help. The toolbar has various icons for file operations like Open, Save, Print, and Database connections. Below the toolbar, there are two tabs: "Worksheet" and "Query Builder", with "Worksheet" selected. The main workspace contains a SQL query:

```
SELECT order_id, order_date, order_total
FROM orders;
```

Below the query, the "Script Output" tab shows the results of the execution:

All Rows Fetched: 105 in 0.087 seconds

	ORDER_ID	ORDER_DATE	ORDER_TOTAL
1	2458	16-AUG-07 09.34.12.234359000 PM	78279.6
2	2397	19-NOV-07 10.41.54.696211000 PM	42283.2
3	2454	02-OCT-07 11.49.34.678340000 PM	6653.4
4	2354	15-JUL-08 12.18.23.234567000 AM	46257
5	2358	09-JAN-08 01.03.12.654278000 AM	7826
6	2381	15-MAY-08 02.59.08.843679000 AM	23034.6
7	2440	01-SEP-07 03.53.06.008765000 AM	70576.9
8	2357	09-JAN-06 04.19.44.123456000 AM	59872.4
9	2394	11-FEB-08 05.22.35.564789000 AM	21863
10	2435	03-SEP-07 05.22.53.134567000 AM	62303
11	2455	20-SEP-07 05.34.11.456789000 PM	14087.5
12	2379	16-MAY-07 08.22.24.234567000 AM	17848.2
13	2396	02-FEB-06 09.34.56.345678000 AM	34930
14	2406	29-JUN-07 10.41.20.098765000 AM	2854.2
15	2434	13-SEP-07 11.49.30.647893000 AM	268651.8
16	2436	02-SEP-07 12.18.04.378034000 PM	6394.8
17	2446	27-JUL-07 01.03.08.302945000 PM	103679.3
18	2447	27-JUL-08 02.59.10.223344000 PM	33893.6
19	2432	14-SEP-07 03.53.40.223345000 PM	10523
20	2433	13-SEP-07 04.19.00.654279000 PM	78
21	2355	26-JAN-06 05.22.51.962632000 PM	94513.5
22	2356	26-JAN-08 05.22.41.934562000 PM	29473.8

At the bottom of the interface, there are status indicators: "Line 8 Column 1", "Insert", "Modified", and "Windows: CR/LF Editing".

4. La cláusula SELECT para responder a esta pregunta debe contener un alias " Product " compuesta por concatenaciones de cadenas de caracteres con las columnas PRODUCT\_NAME, PRODUCT\_ID y PRODUCT\_STATUS. Además, la cláusula SELECT debe contener las columnas LIST\_PRICE y MIN\_PRICE, así como otras dos expresiones aritméticas con los alias "Max Actual Savings" y



5. La tabla DUAL forma claramente la cláusula FROM. La cláusula SELECT es más interesante, ya que no se seleccionan columnas reales, sólo una expresión aritmética. Una posible sentencia SELECT para realizar este cálculo podría ser:

6.

```
SELECT (4 * (22/7) * (3958.759 * 3958.759)) AS "Earth's Area" FROM dual;
```

Este cálculo de la superficie del planeta Tierra es de 197.016.573 millas cuadradas aproximadamente.

## 3. Restringiendo y ordenando datos

### 3.1. Tutorial 1 - Utilizando el operador LIKE

En este tutorial se deberá devolver una lista de valores DEPARTMENT\_NAME que terminen con las tres letras “ing” de la tabla DEPARTMENTS (esquema HR). Los pasos a seguir serán los siguientes:

1. Arrancar SQL\*Plus y conectarse al esquema HR.
2. La cláusula SELECT será:

```
SELECT DEPARTMENT_NAME
```

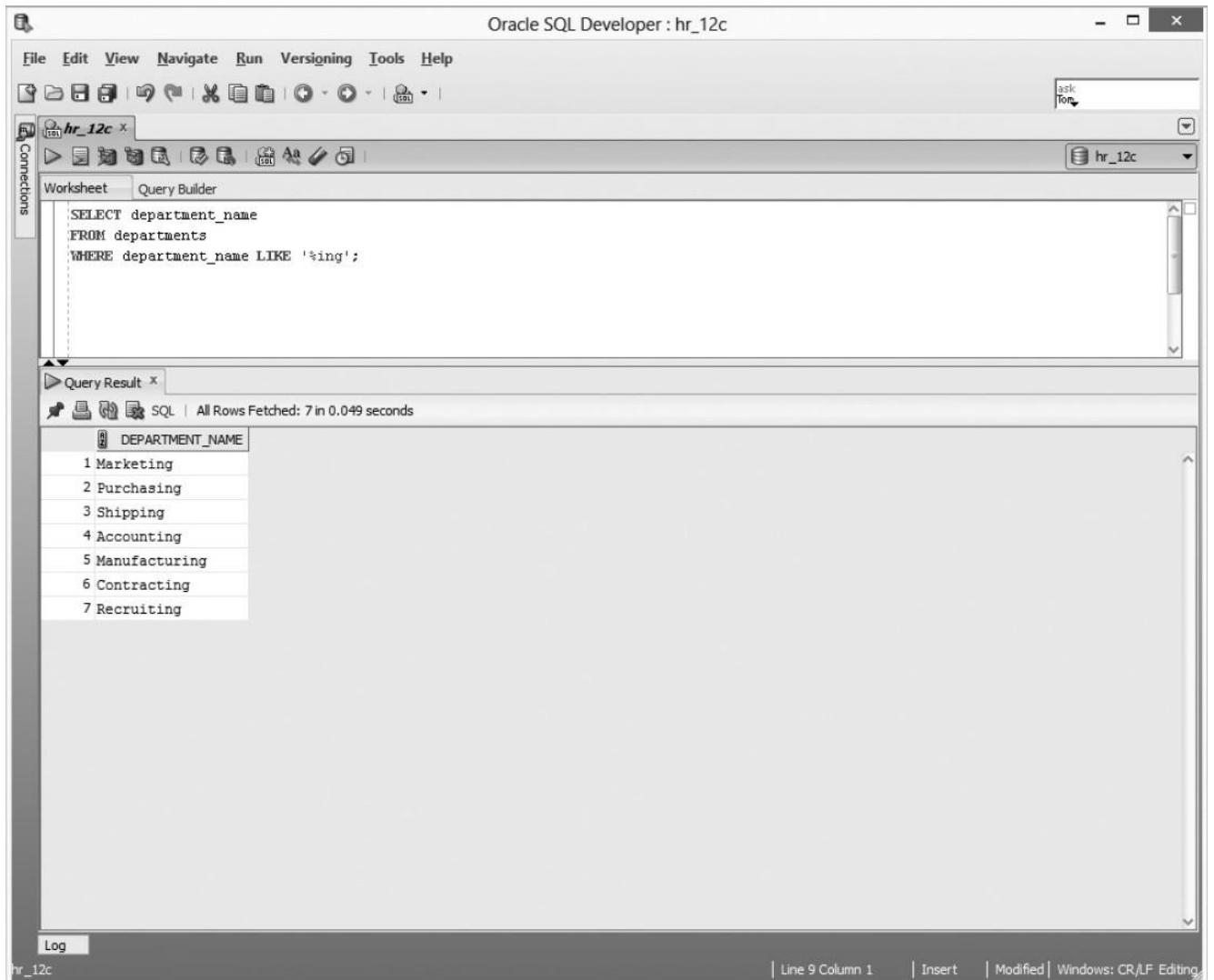
3. La cláusula FROM será:

```
FROM DEPARTMENTS
```

4. La cláusula WHERE debe realizar la comparación entre los valores de la columna DEPARTMENT\_NAME y el patrón de caracteres que comiencen con cero o más caracteres y terminando con tres caracteres específicos, “ing”.
5. El operador que permite establecer si un patrón se corresponde con una secuencia de caracteres es el operador LIKE. El patrón de la columna DEPARTMENT\_NAME debe ser '%ing'. El símbolo comodín de porcentaje indica que cero o más caracteres deben preceder a “ing” en la cadena de caracteres.
6. Así, la cláusula WHERE será:

```
WHERE DEPARTMENT_NAME LIKE '%ing'
```

Ejecutar la sentencia devuelve el conjunto de resultados coincidentes con el patrón que se muestra en la siguiente imagen:



### 3.2. Tutorial 2 - Ordenando datos utilizando la cláusula ORDER BY

La tabla JOBS del esquema HR contiene la descripción de varios tipos de trabajo que un empleado en la organización puede ocupar. Contiene las columnas JOB\_ID, JOB\_TITLE, MIN\_SALARY y MAX\_SALARY.

En este tutorial se requiere escribir una consulta que extraiga las columnas de JOB\_TITLE, MIN\_SALARY, MAX\_SALARY y una expresión con alias VARIANCE que sea la diferencia entre MAX\_SALARY y MIN\_SALARY para cada registro. Los resultados deben solo incluir los valores JOB\_TITLE que contienen las palabras "President" o "Manager". Se debe ordenar la lista en orden descendente basándose en la expresión VARIANCE. Si más de un registro tiene el mismo valor para este campo, se debe ordenar esos registros por JOB\_TITLE en orden alfabético descendente.

Los pasos a seguir serán los siguientes:

1. Iniciar el entorno de desarrollo SQL y conectarse al esquema HR.
2. La cláusula SELECT será:

```
SELECT JOB_TITLE, MIN_SALARY, MAX_SALARY, (MAX_SALARY - MIN_SALARY) VARIANCE
```

3. La cláusula FROM será:

```
FROM JOBS
```

4. Las condiciones WHERE deben permitir solo aquellos registros de JOB\_TITLE que contienen en String "President" or el String "Manager".
5. La cláusula WHERE será:

```
WHERE JOB_TITLE LIKE '%President%' OR JOB_TITLE LIKE '%Manager%'
```

6. La ordenación se consigue usando la cláusula ORDER BY. Se necesitará utilizar la ordenación compuesta para la expresión VARIANCE y la columna JOB\_TITLE en orden descendente.
7. La cláusula ORDER BY será:

```
ORDER BY VARIANCE DESC, JOB_TITLE DESC
```

Se puede especificar alternativamente la expresión en la cláusula ORDER BY en vez de usar el alias VARIANCE.

Al ejecutar la sentencia, se recupera un conjunto de resultados que concuerdan con el patrón como se muestra en la siguiente imagen.

The screenshot shows the Oracle SQL Developer interface. In the top menu bar, the title is "Oracle SQL Developer : hr\_12c". Below the menu is a toolbar with various icons. On the left, there's a "Connections" sidebar with a single entry "hr\_12c". The main area has two tabs: "Worksheet" and "Query Builder", with "Worksheet" selected. The worksheet contains the following SQL query:

```

SELECT job_title, min_salary, max_salary, (max_salary - min_salary) variance
FROM jobs
WHERE job_title LIKE '%President%'
OR job_title LIKE '%Manager%'
ORDER BY variance DESC, job_title DESC;

```

Below the query, the "Script Output" tab is active, showing the results of the query execution:

JOB_TITLE	MIN_SALARY	MAX_SALARY	VARIANCE
1 President	20080	40000	19920
2 Administration Vice President	15000	30000	15000
3 Sales Manager	10000	20080	10080
4 Finance Manager	8200	16000	7800
5 Accounting Manager	8200	16000	7800
6 Purchasing Manager	8000	15000	7000
7 Marketing Manager	9000	15000	6000
8 Stock Manager	5500	8500	3000

At the bottom of the interface, there are status bars for "Log", "hr\_12c", "Line 14 Column 1", "Insert", "Modified", and "Windows: CR/LF Editing".

### 3.3. Tutorial 3 - Utilizando la sustitución ampersand

Un cálculo común que realizan los departamentos de Recursos Humanos es el cálculo de los impuestos que le corresponden a cada empleado. Aunque este cálculo se realiza para todos los empleados, siempre hay miembros de la compañía que discuten los impuestos deducidos de sus ingresos. Los impuestos deducidos se obtienen a partir de los ingresos anuales del empleado multiplicado por la tasa impositiva actual, que puede variar año a año.

En este tutorial se requiere crear una consulta reutilizable a través de la tasa actual y el EMPLOYEE\_ID como entrada y recuperar las siguientes informaciones: EMPLOYEE\_ID, FIRST\_NAME, SALARY, ANNUAL SALARY (SALARY \* 12), TAX\_RATE, y TAX (TAX\_RATE \* ANNUAL SALARY).

Los pasos a seguir serán los siguientes:

1. Iniciar SQL\*Plus y conectarse al esquema HR.
2. Para poder lanzar peticiones de entrada de datos se debe introducir la siguiente sentencia:

```
SET DEFINE ON
```

3. La lista SELECT debe incluir las cuatro columnas especificadas y las dos expresiones. La primera expresión denominada ANNUAL SALARY es un cálculo simple, mientras que la segunda expresión TAX, depende del TAX\_RATE. Como el TAX\_RATE puede variar, esta variable debe ser sustituida en tiempo de ejecución.
4. La cláusula SELECT será:

```
SELECT &&EMPLOYEE_ID EMPLOYEE_ID, FIRST_NAME, SALARY, SALARY* 12
AS "ANNUAL SALARY", &&TAX_RATE "TAX RATE", (&TAX_RATE *(SALARY * 12)) AS TAX
```

5. El doble ampersand que precede a los campos EMPLOYEE y TAX\_RATE le indican a Oracle que cuando se ejecute la sentencia, el usuario deberá introducir un valor por cada una de las variables que se necesite y serán usadas donde estén referenciadas como &EMPLOYEE\_ID y &TAX\_RATE, respectivamente.
6. La cláusula FROM será:

```
FROM EMPLOYEES
```

7. La cláusula WHERE deberá incluir solamente el registro que contenga el EMPLOYEE\_ID indicado en tiempo de ejecución.
8. La cláusula WHERE será entonces:

```
WHERE EMPLOYEE_ID = &EMPLOYEE_ID
```

Al ejecutar la sentencia se recuperará el conjunto de resultados que se ven a continuación

```

SQL> SELECT &&EMPLOYEE_ID employee_id, first_name, salary,
2      salary * 12 as "ANNUAL SALARY",
3      &&TAX_RATE "TAX RATE", (&&TAX_RATE * (salary * 12)) as TAX
4  FROM employees
5 WHERE employee_id = &EMPLOYEE_ID;
Enter value for employee_id: 101
old  1: SELECT &&EMPLOYEE_ID employee_id, first_name, salary,
new  1: SELECT 101 employee_id, first_name, salary,
Enter value for tax_rate: 0.1
old  3:      &&TAX_RATE "TAX RATE", (&&TAX_RATE * (salary * 12)) as TAX
new  3:      0.1 "TAX RATE", (0.1 * (salary * 12)) as TAX
old  5: WHERE employee_id = &EMPLOYEE_ID
new  5: WHERE employee_id = 101

EMPLOYEE_ID FIRST_NAME          SALARY ANNUAL SALARY    TAX RATE      TAX
-----  -----
        101 Neena                17000      204000       .1      20400

SQL>

```

### 3.4. Tutorial 4 - Global

Teniendo en cuenta el esquema OE, véase el siguiente caso: un cliente necesita una unidad de disco duro y una tarjeta gráfica para su ordenador personal. El cliente está dispuesto a gastarse entre \$500 y \$800 en la unidad de disco duro, pero no está seguro del coste de una tarjeta gráfica. Su único requisito es que la resolución soportada por la tarjeta gráfica debe ser de  $1024 \times 768$  o  $1280 \times 1024$ .

La finalidad de este tutorial es escribir una consulta que busque en la tabla PRODUCT\_INFORMATION el valor de PRODUCT\_NAME que comienza con HD (disco duro) o GP (procesador gráfico) y su lista de precios. Los precios de la lista de discos duros deben estar entre 500\$ y 800\$ y los procesadores gráficos tienen que ser de  $1024 \times 768$  o  $1280 \times 1024$ . Los resultados de LIST\_PRICE deberán clasificarse en orden descendente.

#### Solución:

Utilizando SQL Developer o SQL\*Plus, se deberán seguir los siguientes pasos:

Hay que tener en cuenta que se pide que se consulte la tabla PRODUCT\_INFORMATION en el esquema OE para las columnas PRODUCT\_NAME y LIST\_PRICE. Los registros seleccionados deber pertenecer a una de las dos condiciones siguientes:

- La primera condición es que el PRODUCT\_NAME debe empezar por los caracteres "HD" y su LIST\_PRICE debe estar comprendido en el rango entre \$500 y \$800.
- Alternativamente, el registro puede cumplir como segunda condición que el PRODUCT\_NAME empiece por los caracteres "GP" y contenga además las cadenas "1024x768" o "1280x1024". Finalmente, los resultados deben ser ordenados de manera descendente por LIST\_PRICE.

1. Iniciar el SQL Developer y conectese al esquema OE.
2. La cláusula SELECT será:

```
SELECT PRODUCT_NAME, LIST_PRICE
```

3. La cláusula FROM será:

```
FROM PRODUCT_INFORMATION
```

4. La primera condición será:

```
PRODUCT_NAME LIKE 'HD%' AND LIST_PRICE BETWEEN 500 AND 800
```

5. La segunda condición será:

```
PRODUCT_NAME LIKE 'GP%1024x768%' OR  
PRODUCT_NAME LIKE 'GP%1280x1024%'
```

6. Dado que o bien la primera condición o bien la segunda se deben cumplir para que un registro se incluya en el conjunto de resultados, ambas condiciones deben ser unidas por un operador Booleano OR.

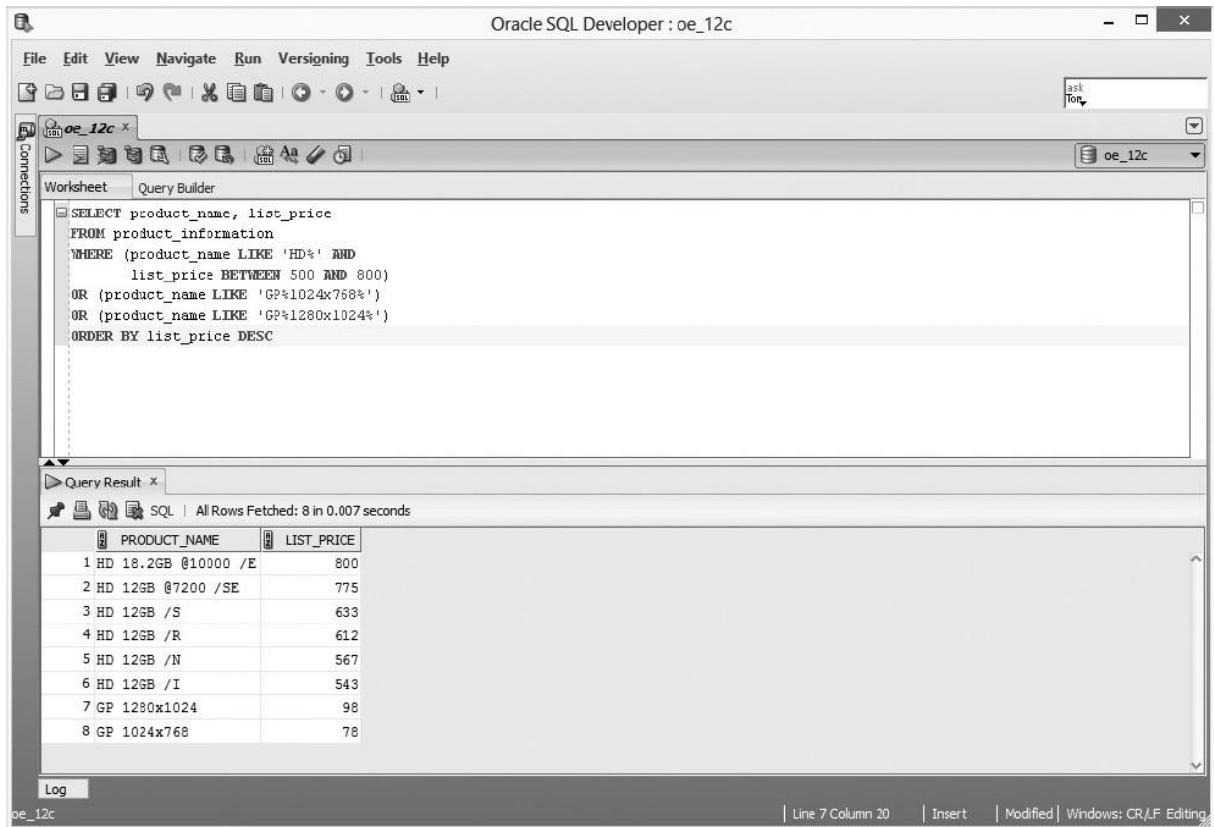
7. La cláusula WHERE será:

```
WHERE (PRODUCT_NAME LIKE 'HD%' AND LIST_PRICE BETWEEN 500 AND 800)  
OR (PRODUCT_NAME LIKE 'GP%1024x768%')  
OR (PRODUCT_NAME LIKE 'GP%1280x1024%')
```

8. La cláusula ORDER BY será:

```
ORDER BY LIST_PRICE DESC
```

Al ejecutar la consulta, se obtiene un conjunto de resultados que coinciden con el patrón establecido tal y como se muestra en la siguiente imagen:



## 4. Funciones de registro único

### 4.1. Tutorial 1 - Utilizando funciones de conversión de mayúsculas/minúsculas

En este tutorial se recuperará una lista de todos los valores FIRST\_NAME que contengan la cadena "li" y su LAST\_NAME de la tabla EMPLOYEES del esquema HR. Los pasos a seguir serán los siguientes:

1. Arrancar SQL Developer y conectarse al esquema HR.
2. La cláusula SELECT será:

```
SELECT FIRST_NAME, LAST_NAME
```

3. La cláusula FROM será:

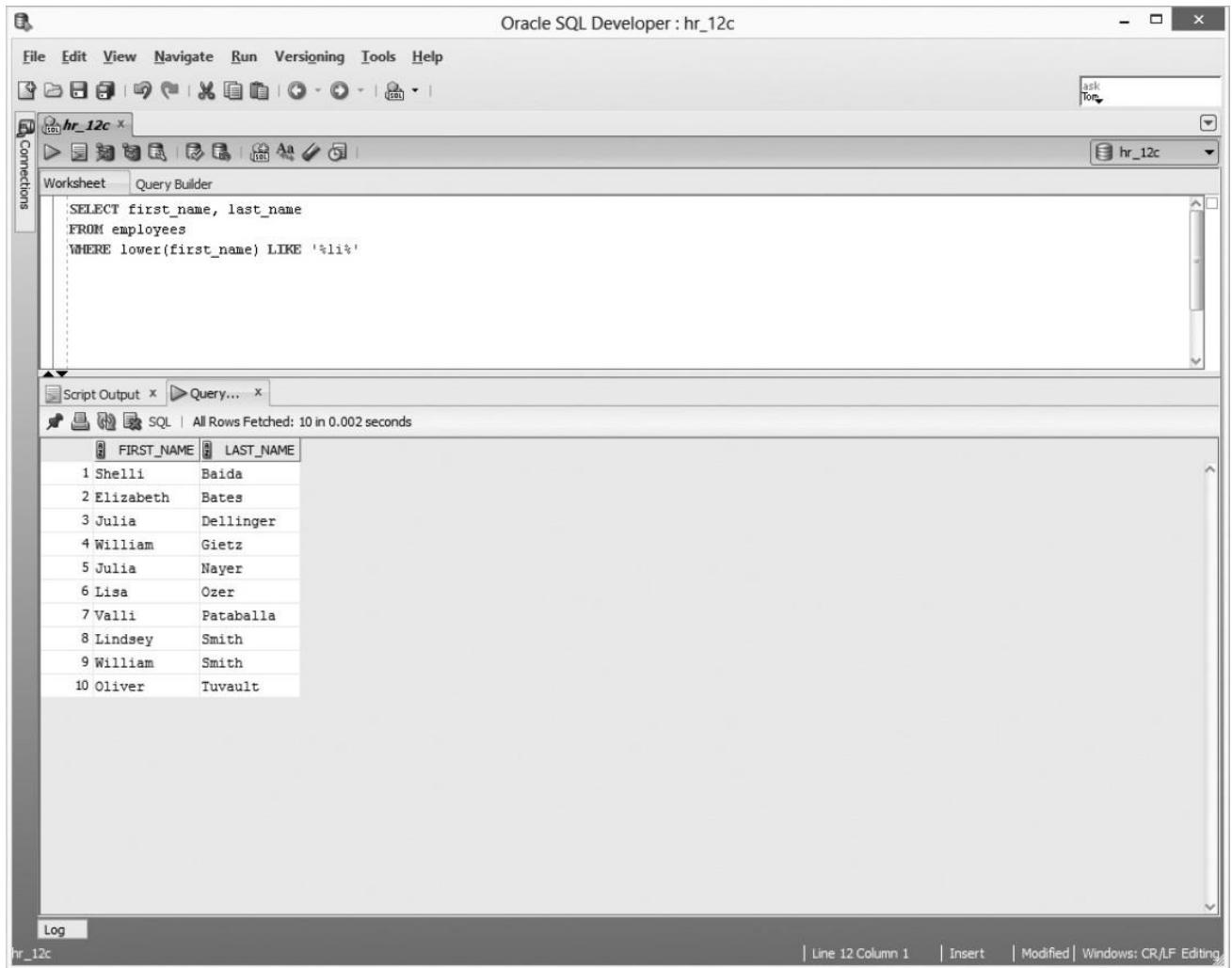
```
FROM EMPLOYEES
```

4. La cláusula WHERE tiene que comparar los valores de la columna FIRST\_NAME con el patrón de caracteres introducido "li". Además, si FIRST\_NAME contiene las cadenas "LI", "Li", "lI" o "li" el valor se tendrá que recuperar.

5. El operador LIKE se utiliza para comparar caracteres y las cuatro combinaciones se pueden recuperar con cuatro WHERE separadas por el enlace OR. Las funciones de mayúscula/minúscula pueden simplificar la condición. Si la función LOWER se utiliza en la columna FIRST\_NAME, la comparación puede hacerse con una única cláusula WHERE. Las funciones UPPER o INITCAP también pueden ser utilizadas.
6. La cláusula WHERE será:

```
WHERE LOWER (FIRST_NAME) LIKE '%li%'
```

Ejecutando esta consulta se devuelven los empleados que contienen los caracteres "li" como se muestra en la siguiente imagen.



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : hr\_12c". The main area displays a query in the Worksheet tab:

```
SELECT first_name, last_name
FROM employees
WHERE lower(first_name) LIKE '%li%'
```

Below the query, the Script Output tab shows the results of the execution:

SQL | All Rows Fetched: 10 in 0.002 seconds

	FIRST_NAME	LAST_NAME
1	Shelli	Baida
2	Elizabeth	Bates
3	Julia	Dellinger
4	William	Gietz
5	Julia	Nayer
6	Lisa	Ozer
7	Valli	Pataballa
8	Lindsey	Smith
9	William	Smith
10	Oliver	Tuvault

The Log tab at the bottom shows the status: "Line 12 Column 1 | Insert | Modified | Windows: CR/LF Editing".

## 4.2. Tutorial 2 - Uso de las funciones de manipulación de caracteres

Teniendo como base el esquema HR, considérese lo siguiente: la impresión de sobres restringe el campo del destinatario a 16 caracteres. Lo ideal sería que el campo destinatario contenga los valores FIRST\_NAME y LAST\_NAME de los empleados separados por un solo espacio. Cuando la longitud combinada de FIRST\_NAME de un empleado y LAST\_NAME excede los 15 caracteres, el campo del destinatario debe contener su nombre formal. El nombre formal de un empleado se compone de la primera letra de su FIRST\_NAME y los primeros 14 caracteres de su LAST\_NAME.

Se pide recuperar una lista de los valores FIRST\_NAME, LAST\_NAME y los nombres formales para empleados donde la longitud combinada de FIRST\_NAME y LAST\_NAME excede los 15 caracteres. Para ello se deberán seguir los siguientes pasos:

1. Iniciar SQL\*Plus y conectarse al esquema HR.
2. El nombre formal se construye concatenando el primer carácter en el campo FIRST\_NAME con un espacio y los primeros 14 caracteres del campo LAST\_NAME para devolver una cadena de 16 caracteres. Se utilizará SUBSTR para extraer la parte inicial y la parte del apellido.
3. La cláusula SELECT será:

```
SELECT FIRST_NAME, LAST_NAME, SUBSTR (FIRST_NAME,1,1) || ' ' || SUBSTR (LAST_NAME,1,14) FORMAL_NAME
```

4. La cláusula FROM será:

```
FROM EMPLOYEES
```

5. La cláusula WHERE debe limitar los registros devueltos sólo a aquellos donde las longitudes combinadas de su NOMBRE y APELLIDOS exceda los 15 caracteres.
6. La cláusula WHERE será:

```
WHERE LENGTH (FIRST_NAME) + LENGTH (LAST_NAME) > 15
```

La ejecución de esta sentencia devuelve el siguiente conjunto de resultados:

```

SQL> select first_name, last_name,
2 substr(first_name,1,1)||' '||substr(last_name,1,14) formal_name
3 from employees
4 where length(first_name)+length(last_name)>15;

FIRST_NAME      LAST_NAME      FORMAL_NAME
-----          -----          -----
Nanette         Cambrault     N Cambrault
Alberto         Errazuriz    A Errazuriz
Michael         Hartstein    M Hartstein
Irene           Mikkilineni I Mikkilineni
Christopher     Olsen        C Olsen
Jose Manuel    Urman       J Urman

6 rows selected.

SQL> =

```

### 4.3. Tutorial 3 - Utilizando las funciones de fecha

En este tutorial se debe obtener una lista de EMPLOYEE\_ID, LAST\_NAME y HIRE\_ID (esquema HR) para los empleados que han trabajado más de 100 meses entre la fecha en que fueron contratados y el 01-JAN-2012. Los pasos a seguir serán los siguientes:

1. Iniciar SQL Developer y conectarse al esquema HR.
2. La cláusula SELECT será:

```
SELECT EMPLOYEE_ID, LAST_NAME, HIRE_DATE, MONTHS_BETWEEN ('01-jan-2012',HIRE_DATE) MONTHS_WORKED
```

3. La cláusula FROM será:

```
FROM EMPLOYEES
```

4. La cláusula WHERE debe comparar los meses entre la fecha dada y el valor HIRE\_DATE con el literal numérico 100.
5. La función MONTHS\_BETWEEN se puede utilizar en la cláusula WHERE.
6. La cláusula WHERE será:

```
WHERE MONTHS_BETWEEN ('01-JAN-2012', HIRE_DATE) > 100
```

La ejecución de esta sentencia devuelve el conjunto de resultados que se muestran a continuación:

The screenshot shows a window titled "SQL Plus". Inside, a SQL query is executed:

```
SQL> SELECT employee_id, last_name, hire_date,
2      months_between('01-jan-2012',hire_date) months_worked
3  FROM employees
4 WHERE months_between('01-jan-2012',hire_date) >100;
```

The results are displayed in a tabular format:

EMPLOYEE_ID	LAST_NAME	HIRE_DATE	MONTHS_WORKED
100	King	17-JUN-03	102.483871
102	De Haan	13-JAN-01	131.612903
108	Greenberg	17-AUG-02	112.483871
109	Faviet	16-AUG-02	112.516129
114	Raphaely	07-DEC-02	108.806452
115	Khoo	18-MAY-03	103.451613
122	Kaufling	01-MAY-03	104
137	Ladwig	14-JUL-03	101.580645
203	Mavris	07-JUN-02	114.806452
204	Baer	07-JUN-02	114.806452
205	Higgins	07-JUN-02	114.806452
206	Gietz	07-JUN-02	114.806452

12 rows selected.

SQL>

#### 4.4. Tutorial 4 - Global

En este tutorial se hará uso del esquema OE, y se tendrá en cuenta el escenario siguiente: se solicitaron varios presupuestos para los precios de las impresoras en color. La información del proveedor no está disponible desde la fuente habitual, pero se sabe que el número de identificación del proveedor está en la columna CATALOG\_URL de la tabla PRODUCT\_INFORMATION.

Se quieren recuperar los valores PRODUCT\_NAME y CATALOG\_URL y extraer el número de proveedor de la columna CATALOG\_URL para todos los productos que tengan las palabras COLOR y PRINTER en la columna PRODUCT\_DESCRIPTION almacenadas con cualquier formato.

##### Solución:

Utilizando SQL Developer o SQL\*Plus se deberán seguir los pasos siguientes:

1. Iniciar SQL Developer y conectarse al esquema OE.
2. Una entrada típica de CATALOG\_URL tiene el siguiente aspecto:

[www.supp-102094.com/cat/hw/p1797.html](http://www.supp-102094.com/cat/hw/p1797.html)

El número de identificación del proveedor tiene una longitud constante de seis caracteres y comienza a partir del decimoséptimo carácter de CATALOG\_URL. La función SUBSTR se utiliza para extraer este valor.

3. Por lo tanto, la cláusula SELECT será :

```
SELECT PRODUCT_NAME, CATALOG_URL, SUBSTR (CATALOG_URL, 17, 6) SUPPLIER
```

4. La cláusula FROM será:

```
FROM PRODUCT_INFORMATION
```

5. Los registros recuperados deben limitarse a aquellos que contengan las palabras COLOR y PRINTER. Estas palabras pueden aparecer en cualquier orden y pueden estar presentes en mayúsculas, minúsculas, o ambas. Cualquiera de las funciones de conversión de mayúsculas-minúsculas puede utilizarse para resolver problemas en el que las palabras están en mayúsculas o minúsculas, pero como las dos palabras pueden ocurrir en cualquier orden, se necesitan dos condiciones. UPPER se utilizará para la conversión a mayúsculas de la comparación.
6. La primera condición será:

```
UPPER (PRODUCT_DESCRIPTION) LIKE '%COLOR%'
```

7. La segunda condición será:

```
UPPER (PRODUCT_DESCRIPTION) LIKE '%PRINTER%'
```

8. La cláusula WHERE será:

```
WHERE UPPER (PRODUCT_DESCRIPTION) LIKE '%COLOR%' AND UPPER (PRODUCT_DESCRIPTION) LIKE '%PRINTER%'
```

La ejecución de la sentencia devuelve el conjunto de resultados que coinciden con este patrón, tal y como se muestra en la siguiente imagen:

```

select product_name, catalog_url, substr(catalog_url,17,6) supplier
from product_information
where upper(product_description) like '%PRINTER%'
  AND upper(product_description) LIKE '%COLOR%';

```

PRODUCT_NAME	CATALOG_URL	SUPPLIER
1 Industrial 600/DQ	http://www.supp-102088.com/cat/hw/p1792.html	102088
2 Inkjet C/4	http://www.supp-102090.com/cat/hw/p2453.html	102090
3 Inkjet C/8/HQ	http://www.supp-102094.com/cat/hw/p1797.html	102094

## 5. Utilizando funciones de conversión y expresiones condicionales

### 5.1. Tutorial 1 - Convirtiendo Fechas en Caracteres Utilizando la Función TO\_CHAR

En este tutorial se pide recuperar una lista de los valores para FIRST\_NAME y para LAST\_NAME, así como una expresión basada en HIRE\_DATE para los empleados contratados en sábado. La expresión recibirá un alias: START\_DATE. Dicha expresión se deberá construir de forma que al introducir un valor HIRE\_DATE de 17-FEB-1996, se recupere la siguiente cadena de valores:

"Saturday, the 13<sup>th</sup> of January, Two Thousand One".

Los pasos a seguir serán los siguientes:

1. Iniciar SQL Developer y conectarse al esquema HR.

2. La cláusula SELECT será:

```
SELECT FIRST_NAME, LAST_NAME,
```

3. La expresión START\_DATE será:

```
TO_CHAR(HIRE_DATE, 'fmDay, "the" Ddth "of" Month, Yyyysp.')START_DATE
```

El modificador 'fm' es necesario para eliminar los espacios que pueda haber alrededor de la palabra dado que se está haciendo una búsqueda por variable carácter y se requiere una coincidencia exacta de valores.

*Recordar que se debe cambiar la configuración de NLS\_DATE\_LANGUAGE a ENGLISH para que el formato se corresponda con el de la solución, tal y como se explica en la guía de configuración.*

4. La cláusula FROM será:

```
FROM EMPLOYEES
```

5. La cláusula WHERE será:

```
WHERE TO_CHAR(HIRE_DATE, 'fmDay') = 'Saturday';
```

Al ejecutar esta sentencia, se recuperarán los nombres de los empleados y la expresión START\_DATE tal y como se ve en la siguiente imagen:

The screenshot shows the Oracle SQL Developer interface with a connection to 'hr\_12c'. In the Worksheet tab, a query is run:

```
SELECT first_name, last_name, to_char(hire_date,'fmDay, "the "ddth "of" Month, YyyySp.') start_date
FROM employees
WHERE to_char(hire_date,'fmDay')='Saturday';
```

The results show 19 rows of employees hired on Saturday, with columns FIRST\_NAME, LAST\_NAME, and START\_DATE.

	FIRST_NAME	LAST_NAME	START_DATE
1	Lex	De Haan	Saturday, the 13th of January, Two Thousand One.
2	David	Austin	Saturday, the 25th of June, Two Thousand Five.
3	Nancy	Greenberg	Saturday, the 17th of August, Two Thousand Two.
4	Den	Raphaely	Saturday, the 7th of December, Two Thousand Two.
5	Shelli	Baida	Saturday, the 24th of December, Two Thousand Five.
6	Julia	Nayer	Saturday, the 16th of July, Two Thousand Five.
7	Steven	Markle	Saturday, the 8th of March, Two Thousand Eight.
8	Laura	Bissot	Saturday, the 20th of August, Two Thousand Five.
9	Michael	Rogers	Saturday, the 26th of August, Two Thousand Six.
10	Curtis	Davies	Saturday, the 29th of January, Two Thousand Five.
11	Peter	Hall	Saturday, the 20th of August, Two Thousand Five.
12	Nanette	Cambrault	Saturday, the 9th of December, Two Thousand Six.
13	David	Lee	Saturday, the 23rd of February, Two Thousand Eight.
14	Elizabeth	Bates	Saturday, the 24th of March, Two Thousand Seven.
15	Alyssa	Hutton	Saturday, the 19th of March, Two Thousand Five.
16	Julia	Dellinger	Saturday, the 24th of June, Two Thousand Six.
17	Jennifer	Dilly	Saturday, the 13th of August, Two Thousand Five.
18	Samuel	McCain	Saturday, the 1st of July, Two Thousand Six.
19	Vance	Jones	Saturday, the 17th of March, Two Thousand Seven.

## 5.2. Tutorial 2 - Utilizando NULLIF Y NVL2 Para Lógica Condicional Simple

En este tutorial se pretende devolver un conjunto de registros de la tabla EMPLOYEES (del esquema HR) con valores de DEPARTMENT\_ID iguales a 100. El conjunto de registros devueltos debe contener los valores de FIRST\_NAME y de LAST\_NAME y una expresión llamada NAME\_LENGTHS. Esta expresión debe devolver una cadena 'Different Length' si la longitud de FIRST\_NAME difiere de la de LAST\_NAME o la cadena 'Same Length', en caso contrario.

Los pasos a seguir serán los siguientes:

1. Ejecutar SQL Developer y conectarse al esquema HR.

*La expresión NAME\_LENGTHS puede ser calculada de varias formas. La solución proporcionada utiliza la función NULLIF para probar si el valor LENGTH devuelto para las columnas FIRST\_NAME y LAST\_NAME es el mismo. Si es así, se devolverá NULL; en caso contrario, se devolverá el valor LENGTH de*

*LAST\_NAME*. Si la otra función (*NVL2*) recibe un parámetro *NULL*, se devolverá la cadena 'Same Length'; en caso contrario, se devolverá la cadena 'Different Length'.

2. La cláusula SELECT es entonces:

```
SELECT FIRST_NAME,  
LAST_NAME, NVL2(NULLIF(LENGTH(LAST_NAME),  
LENGTH(FIRST_NAME)),  
'Different Length', 'Same Length') NAME_LENGTHS
```

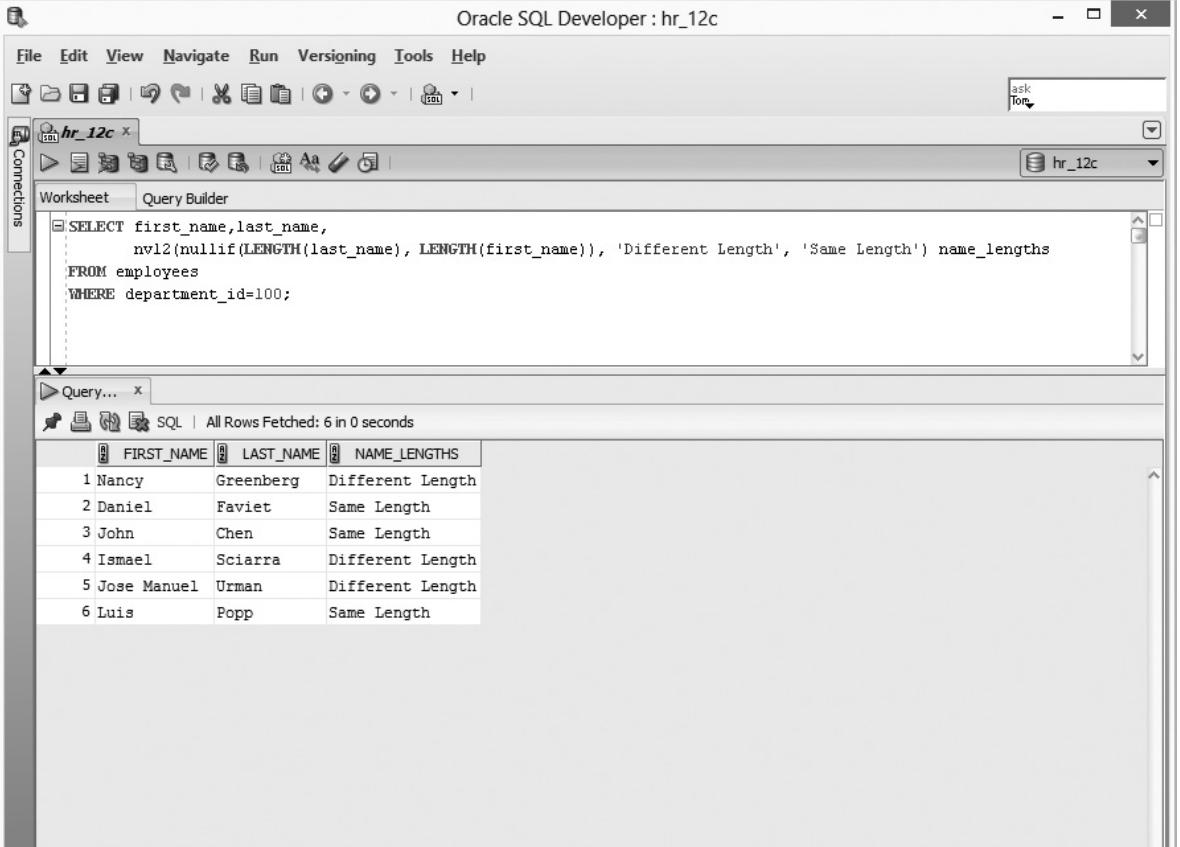
3. La cláusula FROM será:

```
FROM EMPLOYEES
```

4. La cláusula WHERE será:

```
WHERE DEPARTMENT_ID=100
```

Al ejecutar esta sentencia se devuelve el nombre de los empleados y la expresión NAME\_LENGTHS como se muestra en la siguiente imagen:



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : hr\_12c". The menu bar includes File, Edit, View, Navigate, Run, Versioning, Tools, and Help. The toolbar has various icons for file operations like Open, Save, Print, and Run. The Connections sidebar shows a connection named "hr\_12c". The main area has two tabs: "Worksheet" and "Query Builder". The "Worksheet" tab contains the following SQL query:

```
SELECT first_name, last_name,  
       nvl2(nullif(LENGTH(last_name), LENGTH(first_name)), 'Different Length', 'Same Length') name_lengths  
  FROM employees  
 WHERE department_id=100;
```

Below the query, the results are displayed in a table titled "Query...". The table has three columns: FIRST\_NAME, LAST\_NAME, and NAME\_LENGTHS. The data is as follows:

	FIRST_NAME	LAST_NAME	NAME_LENGTHS
1	Nancy	Greenberg	Different Length
2	Daniel	Faviet	Same Length
3	John	Chen	Same Length
4	Ismael	Sciarra	Different Length
5	Jose Manuel	Urman	Different Length
6	Luis	Popp	Same Length

### 5.3. Tutorial 3 - Usando la función DECODE

En este tutorial se pide que se construya una consulta para la tabla LOCATIONS del esquema HR que recupere aquellos registros que tengan el valor US en la columna COUNTRY\_ID. Para ello, se construirá una expresión llamada LOCATION\_INFO para evaluar la columna STATE\_PROVINCE y se recuperará distinta información tal y como se observa en la siguiente tabla.

Se deberán ordenar los registros recuperados como resultado basándose en la expresión LOCATION\_INFO.

<b>Si STATE_PROVINCE es</b>	<b>El valor devuelto es</b>
Washington	Cadena de caracteres "Headquarters"
Texas	Cadena de caracteres "Oil Wells"
California	El valor de la columna CITY
New Jersey	El valor de la columna STREET_ADDRESS

Los pasos a seguir serán:

1. Iniciar SQL Developer y conectarse al esquema HR.

La expresión LOCATION\_ID se puede calcular de varias formas. Esto incluye utilizar una expresión CASE o una función DECODE. La siguiente solución está pensada para usar DECODE.

2. La cláusula SELECT será:

```
SELECT DECODE(STATE_PROVINCE,'Washington','Headquarters','Texas','Oil Wells','California',
CITY,'New Jersey',STREET_ADDRESS) LOCATION_INFO
STATE_PROVINCE, CITY, STREET_ADDRESS,COUNTRY_ID
```

Nótese la mezcla de variables de caracteres con columnas especificados como parámetros dentro de la función DECODE.

3. La cláusula FROM será:

```
FROM LOCATIONS
```

4. La cláusula WHERE será:

```
WHERE COUNTRY_ID='US'
```

5. La cláusula ORDER BY será:

```
ORDER BY LOCATION_INFO
```

El resultado de ejecutar dicha consulta se muestra en la siguiente imagen:

The screenshot shows the Oracle SQL Developer interface with a connection named 'hr\_12c'. In the 'Worksheet' tab, a SQL query is displayed:

```
SELECT decode(state_province,'Washington','Headquarters',
              'Texas','Oil Wells',
              'California',city,
              'New Jersey',street_address) location_info,
        state_province, city, street_address, country_id
   FROM locations
  WHERE country_id='US'
 ORDER BY location_info;
```

In the 'Query Result' tab, the output is shown in a grid:

LOCATION_INFO	STATE_PROVINCE	CITY	STREET_ADDRESS	COUNTRY_ID
1 2007 Zagora St	New Jersey	South Brunswick	2007 Zagora St	US
2 Headquarters	Washington	Seattle	2004 Charade Rd	US
3 Oil Wells	Texas	Southlake	2014 Jabberwocky Rd	US
4 South San Francisco	California	South San Francisco	2011 Interiors Blvd	US

#### 5.4. Tutorial 4 - Global

Supóngase, teniendo en cuenta el esquema OE, que: como parte de una nueva iniciativa de marketing, es necesario obtener una lista con los cumpleaños de los clientes que tienen lugar entre dos días antes y siete días después de cualquier fecha que se pida en el año en curso.

Este listado debe devolver registros de la tabla CUSTOMERS que incluyan las columnas CUST\_FIRST\_NAME, CUST\_LAST\_NAME, CUST\_EMAIL y DATE\_OF\_BIRTH, en orden ascendente, basándose en el día y mes del valor DATE\_OF\_BIRTH. Se requiere una expresión adicional que se llamará

---

BIRTHDAY para devolver un mensaje descriptivo. Hay varias formas de resolver la tarea, por lo que otra solución puede ser correcta aunque difiera de la propuesta a continuación.

### **Solución:**

Utilizando SQL Developer o SQL\*Plus, se deberán seguir los siguientes pasos:

1. Iniciar SQL Developer y conectarse al esquema OE.

El conjunto de datos se debe restringir en registros de la tabla CUSTOMERS donde el componente día de DATE\_OF\_BIRTH esté entre dos días antes y siete días después de cualquier fecha que se pida en el año en curso. La fecha de referencia puede que se entregue usando sustitución ampersand durante la ejecución de la aplicación. Esta solución utiliza la máscara de formado DDD que proporciona el día del año de una fecha.

2. La condición de la cláusula WHERE es:

```
WHERE TO_CHAR(DATE_OF_BIRTH, 'DDD') - TO_CHAR(TO_DATE('&REFDATE'), 'DDD') BETWEEN -2 AND 7
```

3. La expresión BIRTHDAY se puede calcular de varias formas. Teniendo en cuenta la anterior cláusula WHERE se puede asegurar que el conjunto de datos devuelto es correcto. La cláusula SELECT debe recuperar y manipular estos registros para mostrarlos adecuadamente. La expresión CASE proporciona la capacidad de usar expresiones condicionales y se adapta bien a esta tarea.
4. El caso con el que se trabaja está basado en la expresión WHERE que se presentó más arriba. El día del año de DATE\_OF\_BIRTH menos el día del año de la fecha de referencia es evaluado. Por ejemplo, si la diferencia fuese -2, la cadena de caracteres "Day before yesterday" es devuelta. Las diferentes condiciones CASE se comprueban de forma muy similar. El componente ELSE captura cualquier registro que no coincide con las condiciones establecidas en la expresión CASE y devuelve la cadena de caracteres "Later this week".
5. La cláusula SELECT será:

```
SELECT  
CUST_FIRST_NAME, CUST_LAST_NAME, CUST_EMAIL, DATE_OF_BIRTH,  
CASE  
TO_CHAR(DATE_OF_BIRTH, 'DDD') - TO_CHAR(TO_DATE('&REFDATE'), 'DDD')  
WHEN -2 THEN 'Day before yesterday'  
WHEN -1 THEN 'Yesterday'  
WHEN 0 THEN 'Today'  
WHEN 1 THEN 'Tomorrow'  
WHEN 2 THEN 'Day after  
tomorrow'  
ELSE 'Later this week' END BIRTHDAY
```

6. La cláusula FROM será:

FROM CUSTOMERS

7. La cláusula ORDER BY será:

ORDER BY TO\_CHAR(DATE\_OF\_BIRTH, 'MM-DD')

Al ejecutar la consulta, se obtiene un conjunto de resultados que coinciden con el patrón establecido tal y como se muestra en la siguiente imagen. El valor que se introducirá para para *refdate* es 08-jan- 2014, se introducirá las dos veces que se pide:

```
1| SELECT
2| CUST_FIRST_NAME, CUST_LAST_NAME, CUST_EMAIL, DATE_OF_BIRTH,
3| CASE
4| TO_CHAR(DATE_OF_BIRTH, 'DDD') - TO_CHAR(TO_DATE('&REFDATE'), 'DDD')
5| WHEN -2 THEN 'Day before yesterday'
6| WHEN -1 THEN 'Yesterday'
7| WHEN 0 THEN 'Today'
8| WHEN 1 THEN 'Tomorrow'
9| WHEN 2 THEN 'Day after
10| tomorrow'
11| ELSE 'Later this week'
12| END BIRTHDAY
13| FROM CUSTOMERS
14| WHERE TO_CHAR(DATE_OF_BIRTH, 'DDD') - TO_CHAR(TO_DATE('&REFDATE'), 'DDD') BETWEEN -2 AND 7
15| ORDER BY TO_CHAR(DATE_OF_BIRTH, 'MM-DD');
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 10 en 0,003 segundos

	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_EMAIL	DATE_OF_BIRTH	BIRTHDAY
1	Sally	Edwards	Sally.Edwards@TURNSTONE.COM	06/01/80	Day before yesterday
2	Bo	Hitchcock	Bo.Hitchcock@ANHINGA.COM	09/01/54	Tomorrow
3	Bob	Sharif	Bob.Sharif@TEAL.COM	10/01/85	Day aftertomorrow
4	Charlotte	Buckley	Charlotte.Buckley@PINTAIL.COM	10/01/49	Day aftertomorrow
5	Ridley	Hackman	Ridley.Hackman@ANHINGA.COM	11/01/50	Later this week
6	Sachin	Spielberg	Sachin.Spielberg@GADWALL.COM	11/01/71	Later this week
7	Marilou	Landis	Marilou.Landis@TATTER.COM	13/01/33	Later this week
8	Sally	Bogart	Sally.Bogart@WILLET.COM	14/01/85	Later this week
9	Alexander	Stanton	Alexander.Stanton@AUKLET.COM	15/01/65	Later this week
10	Dianne	Sen	Dianne.Sen@TATTER.COM	15/01/53	Later this week

## 6. Presentando datos agregados utilizando funciones agregadas

### 6.1. Tutorial 1 - Utilizando las funciones agregadas

La tabla COUNTRIES almacena una lista de valores COUNTRY\_NAME. En este tutorial se deberá calcular la longitud media de todos los nombres de los países. Cualquier componente fraccionario debe redondearse al número entero más cercano. Los pasos a seguir serán los siguientes:

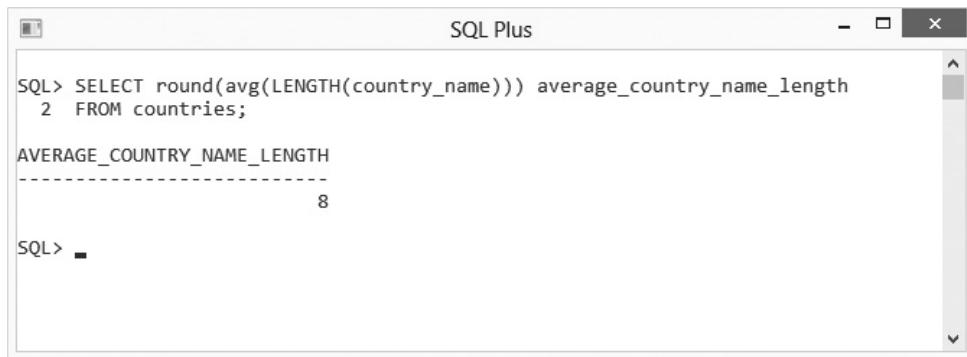
1. Iniciar SQL\*Plus y se conectarse al esquema HR.  
La longitud del valor del nombre del país para cada registro se calcula utilizando la función LENGTH. La longitud promedio se puede determinar con la función AVG. Se puede redondear al número entero más cercano utilizando la función ROUND.
2. La cláusula SELECT, que utiliza el alias AVERAGE\_COUNTRY\_NAME\_LENGTH, será:

```
SELECT ROUND(AVG(LENGTH(COUNTRY_NAME))) AVERAGE_COUNTRY_NAME_LENGTH
```

3. La cláusula FROM será:

```
FROM COUNTRIES;
```

4. La ejecución de esta sentencia devuelve un solo registro que representa la longitud media de todos los nombres de los países en la tabla de COUNTRIES, como se muestra en la siguiente imagen:



The screenshot shows an Oracle SQL Plus window with the following content:

```
SQL> SELECT round(avg(LENGTH(country_name))) average_country_name_length
  2  FROM countries;
AVERAGE_COUNTRY_NAME_LENGTH
-----
8
SQL> -
```

### 6.2. Tutorial 2 - Agrupando datos basados en múltiples columnas

En este tutorial se va a analizar la rotación del personal que aparece en los datos del esquema HR. Se va a crear un informe que contenga el número de empleados que se dieron de baja, agrupados por el año en el que lo hicieron. También se requieren los empleos que realizaron dichos empleados. Los resultados deben ordenarse de forma descendente según el número de empleados en cada grupo. El informe debe listar el año, el JOB\_ID y el número de empleados que dejaron un determinado trabajo en ese año.

---

Los pasos a seguir serán los siguientes:

1. Ejecutar SQL Developer y conectarse al esquema HR. La tabla JOB\_HISTORY contiene las columnas END\_DATE y JOB\_ID, que constituyen la fuente de datos para este informe. El componente del año se extraerá utilizando la función TO\_CHAR. El número de empleados que dejaron un determinado empleo en cada año se obtiene utilizando la función COUNT(\*) .
2. La cláusula SELECT será:

```
SELECT TO_CHAR(END_DATE, 'YYYY') "Quitting Year", JOB_ID, COUNT(*) "Number of Employees"
```

3. La cláusula FROM será:

```
FROM JOB_HISTORY
```

4. No existe cláusula WHERE.
5. Dado que el informe requiere que los empleados sean listados por año y JOB\_ID, estos dos elementos deben aparecer en la cláusula GROUP BY, tal que:

```
GROUP BY TO_CHAR(END_DATE, 'YYYY'), JOB_ID
```

6. La ordenación se realiza con:

```
ORDER BY COUNT(*) DESC
```

Ejecutar esta sentencia devuelve el informe de rotación de personal tal y como muestra la siguiente imagen:

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a 'Connections' panel with a single connection named 'hr\_12c'. The main area has a 'Worksheet' tab active, displaying the following SQL query:

```

SELECT to_char(end_date,'YYYY') "Quitting year",job_id,
       count(*) "Number of Employees"
  FROM job_history
 GROUP BY to_char(end_date,'YYYY'), job_id
 ORDER BY count(*) DESC

```

Below the worksheet, the 'Query Result' pane displays the output of the query:

Quitting year	JOB_ID	Number of Employees
1 2007	ST_CLERK	2
2 2001	AC_ACCOUNT	1
3 2005	AC_MGR	1
4 2006	AC_ACCOUNT	1
5 2006	SA REP	1
6 2001	AD_ASST	1
7 2006	IT_PROG	1
8 2007	MK REP	1
9 2007	SA_MAN	1

At the bottom of the interface, there are status messages: 'Saved: hr\_12c', 'Line 13 Column 1', 'Insert', 'Modified', and 'Windows: CR/LF Editing'.

### 6.3. Tutorial 3 - La cláusula HAVING

Supóngase para este tutorial que la compañía está planeando una campaña de reclutamiento y quiere identificar los días de la semana en los que se contrató a 18 o más miembros del personal. El informe obtenido debe enumerar los días y el número de empleados contratados en cada uno de estos. Los pasos a seguir serán:

1. Iniciar SQL\*Plus y conectarse al esquema HR.

Los registros EMPLOYEES deben dividirse en grupos según el componente día de la columna HIRE\_DATE. El número de empleados por grupo puede obtenerse utilizando la función COUNT.

2. La cláusula SELECT será:

```
SELECT TO_CHAR(HIRE_DATE, 'DAY'), COUNT(*)
```

3. No se requiere la cláusula WHERE ya que se consideran todos los registros físicos de la tabla EMPLOYEES.
4. La cláusula GROUP BY será:

```
GROUP BY TO_CHAR(HIRE_DATE, 'DAY')
```

Esta cláusula GROUP BY crea siete registros a nivel de grupo, uno para cada día de la semana.

La función COUNT de la cláusula SELECT enumera la cantidad de miembros del personal empleados cada día. La cláusula HAVING debe utilizarse para restringir estas siete filas a sólo aquellas en las que el recuento sea mayor o igual a 18.

5. La cláusula HAVING será:

```
HAVING COUNT(*) >= 18;
```

6. La cláusula FROM será:

```
FROM EMPLOYEES
```

La ejecución de esta sentencia devuelve los días de la semana en los que se contrataron 18 o más empleados, como se muestra en la siguiente imagen:

The screenshot shows a window titled "SQL Plus". Inside, a SQL command is being run:

```
SQL> SELECT to_char(hire_date, 'DAY') hire_day, count(*)  
2  FROM employees  
3  GROUP BY to_char(hire_date, 'DAY')  
4  HAVING count(*) >= 18;
```

The output displays two rows of data:

HIRE_DAY	COUNT(*)
SATURDAY	19
FRIDAY	19

SQL> -

## 6.4. Tutorial 4 - Global

En este tutorial se trabajará con el esquema OE. La tabla PRODUCT\_INFORMATION lista los elementos que están pedidos y los que están planeados, obsoletos o en desarrollo. Se requiere que se prepare un informe que agrupe los productos “no disponibles” dependiendo de PRODUCT\_STATUS y que muestre el número de productos en cada grupo y la suma de la columna LIST\_PRICE de los productos por grupo. Además, solo deben mostrarse los registros a nivel de grupo donde la suma de la columna LIST\_PRICE sea mayor que 4000. Un producto no está disponible si su PRODUCT\_STATUS no es igual a la cadena de texto “ordenable”. Hay varias formas de resolver este problema además de la que se muestra a continuación.

### Solución:

Se deberán seguir los siguientes pasos:

1. Iniciar SQL Developer y conectarse al esquema OE. El conjunto de datos debe restringirse para los registros de la tabla PRODUCT\_INFORMATION donde el PRODUCT\_STATUS sea diferente de la cadena de texto "orderable". Dado que esta variable puede tener distinta tipología de letras, se hará una conversión con la función UPPER.

2. La cláusula WHERE será:

```
WHERE UPPER(PRODUCT_STATUS) <> 'ORDERABLE'
```

3. Dado que el conjunto de soluciones se debe segmentar en grupos basándose en su columna PRODUCT\_STATUS, la sentencia GROUP BY será:

```
GROUP BY PRODUCT_STATUS
```

El conjunto de datos está dividido en diferentes grupos basados en los valores de PRODUCT\_STATUS. Por lo tanto, la función COUNT(\*) debe usarse para obtener el número de productos de cada grupo. La función agregada SUM(LIST\_PRICE) puede usarse para calcular la suma de los valores de LIST\_PRICE para todos los registros de cada grupo.

4. La cláusula SELECT será:

```
SELECT COUNT(*), SUM(LIST_PRICE), PRODUCT_STATUS
```

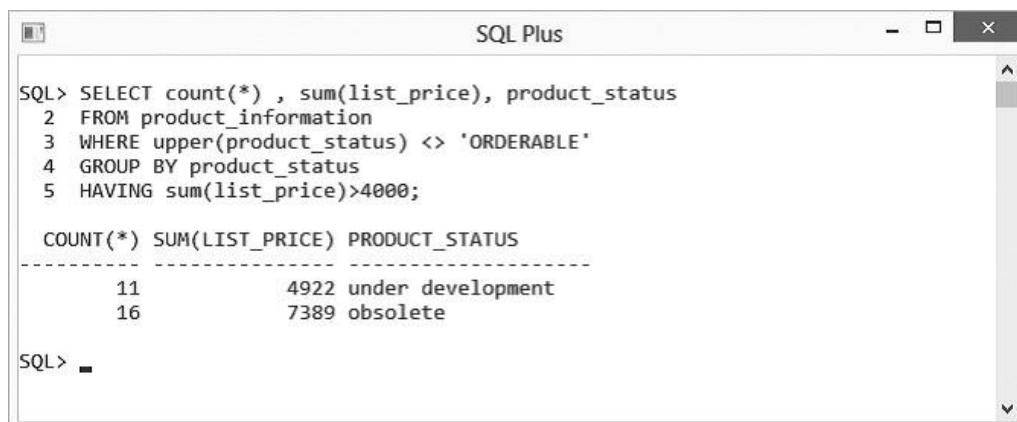
5. La cláusula HAVING que restringe los registros a nivel de grupo será:

```
HAVING SUM(LIST_PRICE) > 4000;
```

6. La cláusula FROM será:

```
FROM PRODUCT_INFORMATION
```

Al ejecutar esta consulta se devuelve el informe requerido como se muestra en la siguiente imagen:



The screenshot shows an Oracle SQL Plus window with the following content:

```
SQL> SELECT count(*), sum(list_price), product_status
  2  FROM product_information
  3 WHERE upper(product_status) <> 'ORDERABLE'
  4 GROUP BY product_status
  5 HAVING sum(list_price)>4000;

COUNT(*) SUM(LIST_PRICE) PRODUCT_STATUS
----- -----
      11          4922 under development
      16          7389 obsolete

SQL> -
```

The query selects the count of products, the total sum of list prices, and the product status for rows where the status is not 'orderable'. It groups by product status and filters using a HAVING clause to include only groups where the total price is greater than 4000. The results show two groups: one for 'under development' products with a count of 11 and a total price of 4922, and another for 'obsolete' products with a count of 16 and a total price of 7389.

---

## 7. Visualización de datos de múltiples tablas

---

### 7.1. Tutorial 1 - Utilizando NATURAL JOIN

La tabla JOB\_HISTORY comparte tres nombres de columnas idénticos con la tabla EMPLOYEES: EMPLOYEE\_ID, JOB\_ID y DEPARTMENT\_ID. Se precisa describir las tablas y extraer los valores de EMPLOYEE\_ID, JOB\_ID, DEPARTMENT\_ID LAST\_NAME, HIRE\_DATE y END\_DATE para todos los registros recuperados utilizando NATURAL JOIN. El alias de la tabla EMPLOYEES será EMP y el de la tabla JOB\_HISTORY será JH. Se deberán seguir los siguientes pasos:

1. Ejecutar SQL\*Plus y conectarse al esquema HR.
2. Las tablas se detallan utilizando los comandos DESC EMPLOYEES y DESC JOB\_HISTORY, y las columnas con nombres idénticos y sus tipos de datos deben ser examinados.
3. La cláusula SELECT será:

```
SELECT EMPLOYEE_ID, JOB_ID, DEPARTMENT_ID, EMP.LAST_NAME, EMP.HIRE_DATE, JH.END_DATE
```

4. La cláusula FROM será:

```
FROM JOB_HISTORY JH
```

5. La cláusula JOIN será:

```
NATURAL JOIN EMPLOYEES EMP
```

Al ejecutar esta sentencia, se devuelve un único registro con el mismo valor de EMPLOYEE\_ID, JOB\_ID y DEPARTMENT\_ID en ambas tablas como se muestra en la siguiente ilustración:

```

SQL> DESC 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)

SQL>
SQL> DESC job_history;
Name          Null?    Type
-----
EMPLOYEE_ID      NOT NULL NUMBER(6)
START_DATE        NOT NULL DATE
END_DATE          NOT NULL DATE
JOB_ID            NOT NULL VARCHAR2(10)
DEPARTMENT_ID      NUMBER(4)

SQL>
SQL> SELECT employee_id, job_id, department_id,
  2     emp.last_name, emp.hire_date, jh.end_date
  3   FROM job_history jh
  4 NATURAL JOIN employees emp;

EMPLOYEE_ID JOB_ID      DEPARTMENT_ID LAST_NAME    HIRE_DATE END_DATE
-----  -----
      176  SA_REP           80 Taylor    24-MAR-06 31-DEC-06

SQL> -

```

## 7.2. Tutorial 2 - Utilizando la cláusula NATURAL JOIN...ON

Cada registro en la tabla DEPARTMENTS tiene una columna MANAGER\_ID que coincide en valor con la columna EMPLOYEE\_ID de la tabla EMPLOYEES. En este tutorial se deberá producir un informe con una columna llamada MANAGERS. Cada registro debe contener una sentencia con el formato '*FIRST\_NAME LAST\_NAME is manager of the DEPARTMENT\_NAME department*'. El alias de la tabla EMPLOYEES será E y el de la tabla DEPARTMENTS será D. Se deberán seguir los siguientes pasos:

1. Ejecutar SQL Developer y conectarse al esquema HR.
2. La columna MANAGERS puede construirse concatenando los elementos requeridos y separados por un espacio.
3. La cláusula SELECT será:

```
SELECT E.FIRST_NAME || ' ' || E.LAST_NAME || ' is manager of the ' || D.DEPARTMENT_NAME || ' department.' "Managers"
```

4. La cláusula FROM será:

```
FROM EMPLOYEES E
```

5. La cláusula JOIN...ON será:

```
JOIN DEPARTMENTS D ON (E. EMPLOYEE_ID=D.MANAGER_ID)
```

6. Al ejecutar la sentencia, se devuelven 11 registros describiendo los directores de cada departamento tal y como se muestra en la siguiente ilustración:

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : hr\_12c". The menu bar includes File, Edit, View, Navigate, Run, Versioning, Tools, and Help. The toolbar has various icons for file operations like Open, Save, Print, and Database. The Connections sidebar shows a connection named "hr\_12c". The main area has two tabs: "Worksheet" and "Query Builder". The "Worksheet" tab contains the following SQL code:

```
SELECT e.first_name||' '||e.last_name||' is manager of the '||  
d.department_name||' department.' "Managers"  
FROM employees e  
JOIN departments d  
ON (e.employee_id=d.manager_id);
```

The "Query Result" tab shows the output of the query:

Managers
1 Steven King is manager of the Executive department.
2 Alexander Hunold is manager of the IT department.
3 Nancy Greenberg is manager of the Finance department.
4 Den Raphaely is manager of the Purchasing department.
5 Adam Fripp is manager of the Shipping department.
6 John Russell is manager of the Sales department.
7 Jennifer Whalen is manager of the Administration department.
8 Michael Hartstein is manager of the Marketing department.
9 Susan Mavris is manager of the Human Resources department.
10 Hermann Baer is manager of the Public Relations department.
11 Shelley Higgins is manager of the Accounting department.

At the bottom, the status bar shows "hr\_12c", "Line 10 Column 1", "Insert", "Modified", and "Windows: CR/LF Editing".

### 7.3. Tutorial 3 - Realizando un SELF-JOIN

En el esquema HR hay una relación jerárquica entre los empleados y sus *managers*. Para cada registro de la tabla EMPLOYEES, la columna MANAGER\_ID guarda el EMPLOYEE\_ID de cada *manager*. Utilizando un SELF-JOIN sobre la tabla EMPLOYEES, se requiere recuperar el LAST\_NAME, EMPLOYEE\_ID y MANAGER\_ID del empleado, así como el LAST\_NAME del *manager* y el DEPARTMENT\_ID del empleado para los registros con valores 10, 20 y 30 de DEPARTMENT\_ID. Se deberán renombrar la tabla EMPLOYEES como E y la segunda utilización de esta tabla como M. Se requiere ordenar los resultados basándose en la columna DEPARTMENT\_ID.

Deberán seguirse los siguientes pasos:

1. Iniciar SQL Developer y conectarse al esquema HR.
2. La cláusula SELECT será:

```
SELECT E.LAST_NAME EMPLOYEE, E.EMPLOYEE_ID, E.MANAGER_ID, M.LAST_NAME MANAGER, E.DEPARTMENT_ID
```

3. La cláusula FROM con la tabla fuente y su alias será:

```
FROM EMPLOYEES E
```

4. La cláusula JOIN ... ON con la tabla objetivo renombrada será:

```
JOIN EMPLOYEES M ON (E.MANAGER_ID=M.EMPLOYEE_ID)
```

5. La cláusula WHERE será:

```
WHERE E.DEPARTMENT_ID IN (10,20,30)
```

6. La cláusula ORDER BY será:

```
ORDER BY E.DEPARTMENT_ID
```

Al ejecutar esta consulta, se devuelven 9 registros que describen a los *managers* de cada empleado de los departamentos seleccionados, tal y como se muestra en la siguiente imagen:

```

SELECT e.last_name employee, e.employee_id, e.manager_id, m.last_name manager, e.department_id
FROM employees e
JOIN employees m
ON (e.manager_id=m.employee_id)
WHERE e.department_id IN (10,20,30)
ORDER BY e.department_id;

```

EMPLOYEE	EMPLOYEE_ID	MANAGER_ID	MANAGER	DEPARTMENT_ID
1 Whalen	200	101 Kochhar		10
2 Fay	202	201 Hartstein		20
3 Hartstein	201	100 King		20
4 Tobias	117	114 Raphaely		30
5 Baida	116	114 Raphaely		30
6 Himuro	118	114 Raphaely		30
7 Khoo	115	114 Raphaely		30
8 Raphaely	114	100 King		30
9 Colmenares	119	114 Raphaely		30

## 7.4. Tutorial 4 - Realizando un OUTER JOIN

La tabla DEPARTMENTS contiene detalles de todos los departamentos de la organización. En este tutorial se pretenden recuperar los valores de DEPARTMENT\_NAME y DEPARTMENT\_ID para esos departamentos que no tienen empleados asignados actualmente. Se deberán seguir los siguientes pasos:

1. Ejecutar SQL\*Plus y conectarse al esquema HR.
2. La cláusula SELECT será:

```
SELECT D.DEPARTMENT_NAME, D.DEPARTMENT_ID
```

3. La cláusula FROM con la tabla origen y el alias será:

```
FROM DEPARTMENTS D
```

4. La cláusula LEFT OUTER JOIN con el alias de la tabla origen será:

```
LEFT OUTER JOIN EMPLOYEES E ON E.DEPARTMENT_ID=D.DEPARTMENT_ID
```

5. La cláusula WHERE será:

```
WHERE E.DEPARTMENT_ID IS NULL
```

6. Ejecutarse, esta sentencia devuelve 16 registros que describen los departamentos los cuales no tienen empleados asignados actualmente tal y como se muestra en la siguiente imagen.

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, 'File', 'Edit', 'View', 'Navigate', 'Run', 'Versioning', 'Tools', and 'Help' are visible. Below the menu is a toolbar with various icons. On the left, there's a 'Connections' panel showing a connection named 'hr\_12c'. The main area has tabs for 'Worksheet' and 'Query Builder', with 'Worksheet' selected. A SQL query is entered in the worksheet:

```
SELECT department_name, d.department_id
FROM departments d
LEFT OUTER JOIN employees e
ON e.department_id = d.department_id
WHERE e.department_id IS NULL;
```

Below the worksheet, a 'Query Result' tab is open, showing the results of the query. The results are displayed in a table with two columns: 'DEPARTMENT\_NAME' and 'DEPARTMENT\_ID'. The data is as follows:

DEPARTMENT_NAME	DEPARTMENT_ID
1 Treasury	120
2 Corporate Tax	130
3 Control And Credit	140
4 Shareholder Services	150
5 Benefits	160
6 Manufacturing	170
7 Construction	180
8 Contracting	190
9 Operations	200
10 IT Support	210
11 NOC	220
12 IT Helpdesk	230
13 Government Sales	240
14 Retail Sales	250
15 Recruiting	260
16 Payroll	270

At the bottom of the interface, there are buttons for 'Log', 'Line 10 Column 1', 'Insert', 'Modified', and 'Windows: CR/LF Editing'.

## 7.5. Tutorial 5 - Realizando un CROSS JOIN

En este tutorial se pretende obtener el número de registros de las tablas EMPLOYEES y DEPARTMENTS realizando un producto cartesiano entre ellas. Se deben confirmar los resultados obteniendo de manera explícita dicho producto tras contar y multiplicar el número de registros en ambas tablas. Los pasos a seguir serán:

1. Iniciar el SQL\*Plus y conectarse al esquema HR
2. La cláusula SELECT será:

```
SELECT COUNT(*)
```

3. La cláusula FROM será:

```
FROM EMPLOYEES
```

4. El producto cartesiano se realiza utilizando:

```
CROSS JOIN DEPARTMENTS
```

5. Los recuentos explícitos de registros se obtienen a través de:

```
SELECT COUNT(*) FROM EMPLOYEES;
```

```
SELECT COUNT(*) FROM DEPARTMENTS;
```

6. La multiplicación explícita de los valores resultantes de las consultas anteriores se puede hacer haciendo uso de una consulta a la tabla DUAL.  
7. Al ejecutar esta consulta, se observa que hay 107 registros en la tabla EMPLOYEES, y 27 en la tabla DEPARTMENTS. El producto Cartesiano devuelve 2889 registros de ambos conjuntos, tal y como se muestra en la siguiente ilustración:

The screenshot shows a window titled "SQL Plus" with the following content:

```
SQL> SELECT count(*)
  2  FROM employees;
COUNT(*)
-----
 107

SQL>
SQL> SELECT count(*)
  2  FROM departments;
COUNT(*)
-----
 27

SQL>
SQL>
SQL> SELECT count(*)
  2  FROM employees
  3  CROSS JOIN departments;
COUNT(*)
-----
 2889

SQL>
SQL>
SQL> SELECT 107 * 27
  2  FROM dual;
 107*27
-----
 2889

SQL> -
```

## 7.6. Tutorial 6 - Global

En este tutorial se pretende hacer uso del esquema OE para crear un informe de los clientes que compraron productos con precios de catálogo superiores a 1.000 USD. El informe debe contener los nombres y apellidos del cliente, así como los nombres de los productos y sus precios en el catálogo. La información del cliente se almacena en la tabla CUSTOMERS, que tiene como clave principal la columna CUSTOMER\_ID. El nombre del producto y los detalles del precio del catálogo se almacenan en la tabla PRODUCT\_INFORMATION con la columna PRODUCT\_ID como clave primaria. Otras dos tablas relacionadas pueden ayudar a generar el informe requerido: la tabla ORDERS, que almacena la información CUSTOMER\_ID y ORDER\_ID, y la tabla ORDER\_ITEMS, que almacena los valores PRODUCT\_ID asociados con cada ORDER\_ID.

Hay varios enfoques para resolver esta cuestión. El enfoque elegido puede diferir de la solución indicada.

### Solución:

Utilizando SQL Developer o SQL\*Plus, es necesario conectarse al esquema OE para seguir los siguientes pasos.

1. La cláusula SELECT consta de cuatro columnas de dos tablas, que se asociarán entre sí mediante varias uniones.

```
SELECT CUST_FIRST_NAME, CUST_LAST_NAME, PRODUCT_NAME, LIST_PRICE
```

2. La cláusula FROM será:

```
FROM CUSTOMERS
```

3. La cláusula WHERE será:

```
WHERE LIST_PRICE > 1000
```

Las cláusulas JOIN son interesantes ya que las tablas PRODUCT\_INFORMATION y CLIENTES no están directamente relacionadas. Se relacionan a través de otras dos tablas.

4. La tabla ORDERS se debe unir en primer lugar a la tabla CUSTOMERS basándose en los valores de CUSTOMER\_ID. La primera cláusula join que sigue a la cláusula FROM CUSTOMERS es la siguiente:

```
JOIN ORDERS USING (CUSTOMER_ID)
```

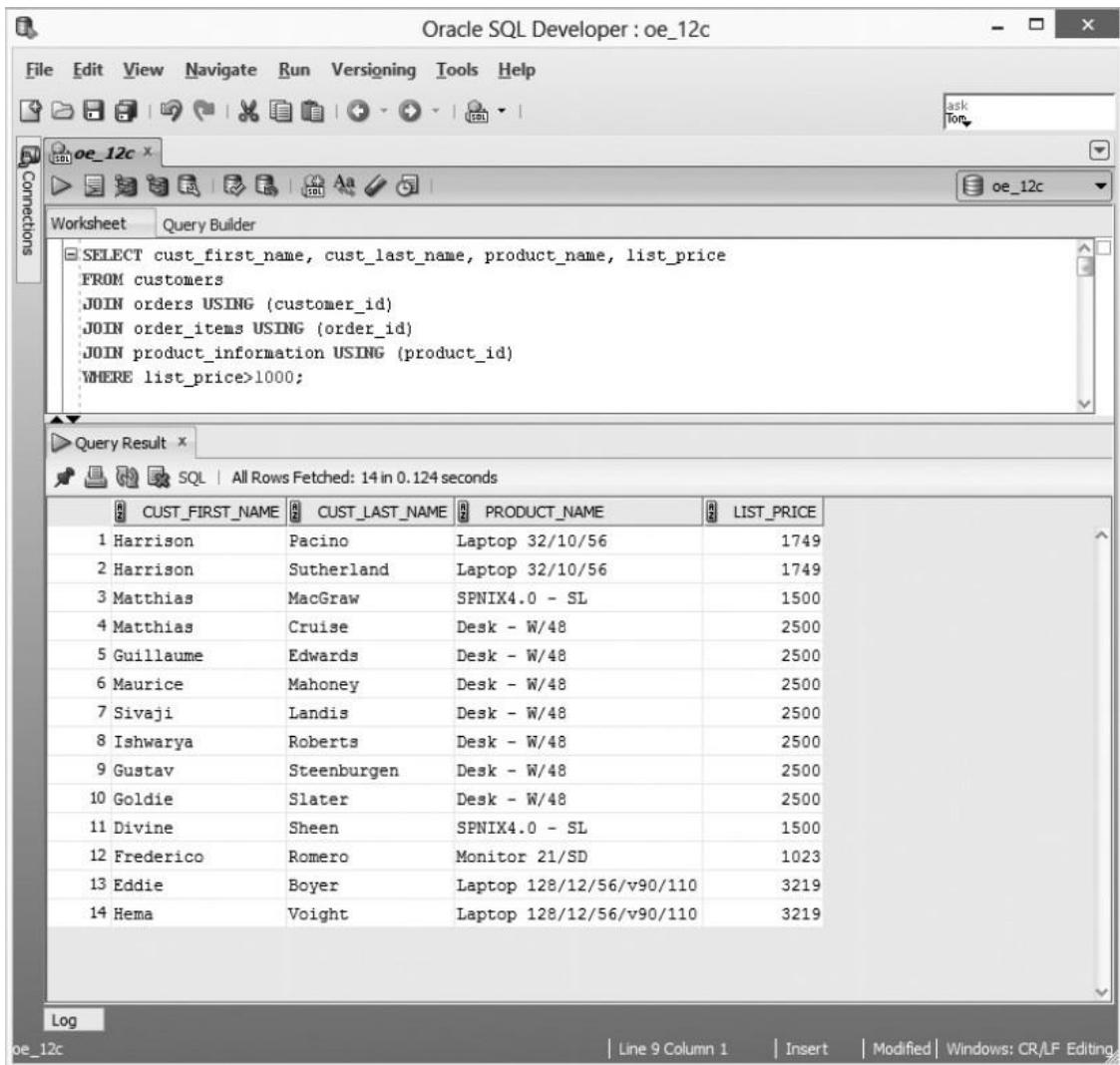
5. Este conjunto debe ahora unirse a la tabla ORDER\_ITEMS basándose en valores comunes de ORDER\_ID, ya que la tabla ORDER\_ITEMS puede enlazarse finalmente con la tabla PRODUCT\_INFORMATION. La segunda cláusula join es por lo tanto la siguiente:

```
JOIN ORDER_ITEMS USING (ORDER_ID)
```

6. Ya está disponible el eslabón que falta para unirse a la tabla PRODUCT\_INFORMATION basada en los valores comunes de las columnas PRODUCT\_ID. La tercera cláusula JOIN es la siguiente:

```
JOIN PRODUCT_INFORMATION USING (PRODUCT_ID)
```

La ejecución de esta expresión devuelve el informe requerido como se muestra en la siguiente imagen (dado que no se ha ordenado el resultado, es posible que la disposición de los registros no sea la misma):



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Versioning, Tools, and Help. A toolbar below the menu has various icons for file operations like Open, Save, and Print. The main workspace is titled 'oe\_12c' and contains a 'Worksheet' tab where the following SQL query is entered:

```
SELECT cust_first_name, cust_last_name, product_name, list_price
FROM customers
JOIN orders USING (customer_id)
JOIN order_items USING (order_id)
JOIN product_information USING (product_id)
WHERE list_price > 1000;
```

Below the worksheet is a 'Query Result' tab showing the output of the query. The results are presented in a table with the following data:

	CUST_FIRST_NAME	CUST_LAST_NAME	PRODUCT_NAME	LIST_PRICE
1	Harrison	Pacino	Laptop 32/10/56	1749
2	Harrison	Sutherland	Laptop 32/10/56	1749
3	Matthias	MacGraw	SPNIX4.0 - SL	1500
4	Matthias	Cruise	Desk - W/48	2500
5	Guillaume	Edwards	Desk - W/48	2500
6	Maurice	Mahoney	Desk - W/48	2500
7	Sivaji	Landis	Desk - W/48	2500
8	Ishwarya	Roberts	Desk - W/48	2500
9	Gustav	Steenburgen	Desk - W/48	2500
10	Goldie	Slater	Desk - W/48	2500
11	Divine	Sheen	SPNIX4.0 - SL	1500
12	Frederico	Romero	Monitor 21/SD	1023
13	Eddie	Boyer	Laptop 128/12/56/v90/110	3219
14	Hema	Voight	Laptop 128/12/56/v90/110	3219

## 8. Utilizando subconsultas para resolver problemas

### 8.1. Tutorial 1 - Tipos de subconsultas

En este tutorial se pretende ilustrar dónde y cómo se pueden utilizar las subconsultas. Se usará SQL\*Plus o SQL Developer. Deberán seguirse los siguientes pasos, tras conectarse al esquema HR:

1. Acceder a la base de datos como usuario HR.
2. Escribir una petición que utilice subconsultas en la lista de proyección de columnas. La consulta informará sobre el número actual de departamentos y de personal:

```
SELECT sysdate Today,  
(SELECT count(*) FROM departments) Dept_count,  
(SELECT count(*) FROM employees) Emp_count  
FROM dual;
```

3. Escribir una consulta para identificar a todos los empleados que son gerentes (*managers*). Esto requerirá el uso de una subconsulta en la cláusula WHERE para seleccionar todos los empleados cuyo EMPLOYEE\_ID aparece como MANAGER\_ID:

```
SELECT last_name  
FROM employees  
  
WHERE employee_id IN  
(SELECT manager_id  
FROM employees);
```

4. Escribir una consulta para identificar el salario más alto pagado en cada país. Esto requerirá el uso de una subconsulta en la cláusula FROM:

```
SELECT max(salary),country_id  
FROM (SELECT salary, country_id  
FROM employees  
NATURAL JOIN departments  
NATURAL JOIN locations)  
GROUP BY country_id;
```

El resultado se puede ver en la siguiente figura:

```
SQL> SELECT sysdate Today,
  2  (SELECT count(*) FROM departments) Dept_count,
  3  (SELECT count(*) FROM employees) Emp_count
  4 FROM dual;

TODAY      DEPT_COUNT   EMP_COUNT
-----      -----
01-OCT-13          27        107

SQL> SELECT last_name
  2  FROM employees
  3  WHERE employee_id IN
  4  (SELECT manager_id
  5    FROM employees);

LAST_NAME
-----
Cambrault
De Haan
Errazuriz
Fripp
Greenberg
Hartstein
Higgins
Hunold
Kaufling
King
Kochhar
Mourgos
Partners
Raphaely
Russell
Vollman
Weiss
Zlotkey

18 rows selected.

SQL> SELECT max(salary),country_id
  2  FROM (SELECT salary, country_id
  3         FROM employees
  4         NATURAL JOIN departments
  5         NATURAL JOIN locations)
  6 GROUP BY country_id;

MAX(SALARY) CO
----- --
      17000 US
       6000 CA
      10000 UK

SQL> -
```

## 8.2. Tutorial 2 - Subconsultas más complejas

En este tutorial se escribirán subconsultas más complejas sobre el esquema HR. Se podrá usar tanto SQL\*Plus o SQL Developer. Se deberán seguir los siguientes pasos:

1. Conectarse a la base de datos como usuario HR.
2. Escribir una consulta que identifique a todos los empleados que trabajan en departamentos ubicados en el Reino Unido. Esto requerirá tres niveles de subconsultas anidadas:

```

SELECT last_name
FROM employees

WHERE department_id IN
  (SELECT
    department_id
    FROM
    departments
    WHERE
    location_id IN
      (SELECT
        location_id
        FROM locations
        WHERE country_id =
          (SELECT country_id
          FROM countries WHERE country_name='United Kingdom')
      )
  );

```

3. Verificar que el resultado del paso 2 sea correcto ejecutando las subconsultas independientemente. Primero, se encontrará el COUNTRY\_ID para el Reino Unido:

```

SELECT country_id
FROM countries

WHERE country_name='United Kingdom';

```

- a. El resultado será el Reino Unido. A continuación, se buscará las ubicaciones correspondientes:

```

SELECT location_id
FROM locations

WHERE country_id = 'UK';

```

- b. El LOCATION\_ID devolverá 2400, 2500 y 2600. Luego se buscará el DEPARTMENT\_ID de estos departamentos en esta ubicación:

```

SELECT department_id
FROM departments
WHERE location_id IN (2400,2500,2600);

```

- c. 3.3 El resultado serán dos departamentos, 40 y 80. Por último, se buscará a los empleados correspondientes:

```
SELECT last_name
FROM employees
WHERE department_id IN (40,80);
```

4. Se escribirá una consulta para identificar a todos los empleados que ganan más que la media y que trabajan en cualquiera de los departamentos de IT. Esto requerirá dos subconsultas no anidadas:

```
SELECT last_name
FROM employees
WHERE department_id IN
  (SELECT department_id
   FROM departments
   WHERE department_name LIKE 'IT%')
AND salary > (SELECT avg(salary) FROM employees);
```

### 8.3. Tutorial 3 - Investigar los diferentes tipos de subconsultas

En este tutorial se demostrarán los problemas que pueden ocurrir con diferentes tipos de subconsultas. Se utilizará SQL\*Plus o SQL Developer. Se supone que la tabla EMPLOYEES tiene los registros por defecto. Se deberán seguir los siguientes pasos:

1. Conectarse a la base de datos con el usuario HR.
2. Escribir una consulta para determinar quién gana más que Tobias:

```
SELECT last_name
FROM employees
WHERE salary >
  (SELECT salary FROM employees WHERE last_name='Tobias') ORDER BY last_name;
```

Esta consulta devolverá 86 nombres, en orden alfabético.

3. Escribir una consulta para determinar quién gana más que Taylor:

```
SELECT last_name
FROM employees
WHERE salary >
  (SELECT salary FROM employees WHERE last_name='Taylor') ORDER BY last_name;
```

Esto fallará con el error "ORA-01427: single-row subquery returns morethan one row." La siguiente imagen muestra los últimos registros de la salida del paso 2 seguido del paso 3 y el error, ejecutado con SQL\*Plus:

The screenshot shows a window titled "SQL Plus". The output pane contains the following text:

```
LAST_NAME
-----
Rogers
Russell
Sarchand
Sciarra
Sewall
Smith
Smith
Stiles
Sully
Taylor
Taylor
Tucker
Tuvault
Urman
Vishney
Vollman
Walsh
Weiss
Whalen
Zlotkey

86 rows selected.

SQL> SELECT last_name
  2  FROM employees
  3  WHERE salary >
  4    (SELECT salary
  5     FROM employees
  6     WHERE last_name='Taylor')
  7  ORDER by last_name;
(SELECT salary
 *
ERROR at line 4:
ORA-01427: single-row subquery returns more than one row

SQL>
```

4. Determinar por qué la consulta del paso 2 tuvo éxito pero falló en el paso 3. La respuesta está en el estado de los datos:

```
SELECT count(last_name)
FROM employees
WHERE last_name='Tobias';
```

```
SELECT count(last_name)
FROM employees
WHERE last_name='Taylor';
```

El uso del operador "mayor que" en las consultas de los pasos 2 y 3 requiere una subconsulta que devuelva un solo registro, pero la subconsulta utilizada puede devolver cualquier número de registros, dependiendo del predicado de búsqueda utilizado.

5. Solucionar el código en los pasos 2 y 3 para que las sentencias tengan éxito sin importar el LAST\_NAME que se use. Hay dos soluciones posibles: una utiliza un operador de comparación diferente que puede gestionar una subconsulta de varios registros; la otra utiliza una subconsulta que siempre revolverá un solo registro.

La primera solución:

```
SELECT last_name
FROM employees
WHERE salary > ALL
(SELECT salary
FROM employees
WHERE last_name='Taylor')
ORDER BY last_name;
```

La segunda solución:

```
SELECT last_name
FROM employees
WHERE salary >
(SELECT max(salary)
FROM employees
WHERE last_name='Taylor')
ORDER BY last_name;
```

#### 8.4. Tutorial 4 - Crear una consulta que sea fiable y fácil de interpretar

En este tutorial se desarrollará una *subconsulta de registros múltiples* que solicite datos de entrada del usuario. Se podrá utilizar SQL\*Plus o SQL Developer. Se supone que los datos de las tablas del esquema HR tienen los valores por defecto. Se deberá seguir los siguientes pasos:

1. Conectarse a la base de datos como usuario HR.
2. Diseñar una consulta que solicite el nombre de un departamento y el apellido de cada empleado

```
SELECT last_name
FROM employees
WHERE department_id =
(SELECT department_id FROM departments
WHERE department_name = '&Department_name');
```

3. Ejecutar la consulta en el paso 2 tres veces, cuando se pida que proporcione estos valores:

Primera vez, Executive

Segunda vez, executive

Tercera vez, Executiv

La siguiente imagen muestra el resultado, utilizando SQL\*Plus:

The screenshot shows the SQL\*Plus interface with the title bar "SQL Plus". The window contains the following SQL code and its execution results:

```
SQL> SELECT last_name
  2  FROM employees
  3  WHERE department_id =
  4    (SELECT department_id
  5     FROM departments
  6     WHERE department_name = '&Department_name');
Enter value for department_name: Executive
old  6:   WHERE department_name = '&Department_name')
new  6:   WHERE department_name = 'Executive')

LAST_NAME
-----
King
Kochhar
De Haan

SQL> /
Enter value for department_name: executive
old  6:   WHERE department_name = '&Department_name')
new  6:   WHERE department_name = 'executive')

no rows selected

SQL> /
Enter value for department_name: Executiv
old  6:   WHERE department_name = '&Department_name')
new  6:   WHERE department_name = 'Executiv')

no rows selected

SQL>
```

4. Obsérvense los resultados del paso 3. La primera ejecución tuvo éxito porque el valor introducido coincidió exactamente, pero la otra falló. Se ajustará la consulta para que sea más fácil de usar, de modo que acepte pequeñas variaciones entre mayúscula y minúscula o la ortografía.

SELECT last\_name

```

FROM employees

WHERE department_id =
  (SELECT department_id
   FROM departments
   WHERE upper(department_name) LIKE
upper('%%&Department_name%'));

```

5. Se ejecuta la consulta en el paso 4 tres veces, utilizando los mismos valores que en el paso 3. Esta vez, la consulta se ejecutará con éxito en todos los casos.
6. Se ejecuta de nuevo la consulta en el paso 4 y esta vez se introduce el valor Pu. La consulta fallará, con un error "ORA-01427: single-row subquery returns more than one row", porque el intento de hacerla más fácil de usar significa que ya no se garantiza que la subconsulta sea una subconsulta de un solo registro. La cadena Pu coincide con dos departamentos.
7. Se modifica la consulta para que no muestre el error ORA-01427, y la salida evite cualquier confusión posible.

```

SELECT last_name,department_name

FROM employees

JOIN departments

ON employees.department_id = departments.department_id

WHERE departments.department_id IN

  (SELECT department_id
   FROM departments
   WHERE upper(department_name) LIKE
upper('%%&Department_name%'));

```

La siguiente imagen muestra este paso final. Este código es mucho más robusto que los códigos anteriores y más fácil de usar:

SQL> SELECT last\_name, department\_name  
 2 FROM employees  
 3 JOIN departments  
 4 ON employees.department\_id=departments.department\_id  
 5 WHERE departments.department\_id IN  
 6 (SELECT department\_id  
 7 FROM departments  
 8 WHERE upper(department\_name) LIKE upper('%&Department\_name%'));  
 Enter value for department\_name: Pu  
 old 8: WHERE upper(department\_name) LIKE upper('%&Department\_name%'))  
 new 8: WHERE upper(department\_name) LIKE upper('%Pu%'))

LAST_NAME	DEPARTMENT_NAME
Raphaely	Purchasing
Khoo	Purchasing
Baida	Purchasing
Tobias	Purchasing
Himuro	Purchasing
Colmenares	Purchasing
Baer	Public Relations

7 rows selected.

SQL> -

## 8.5. Tutorial 5 - Global

El Tutorial 3 incluye esta consulta que intenta encontrar a todos los empleados cuyo salario es mayor que el empleado nombrado:

```
SELECT last_name
FROM employees
WHERE salary >
(SELECT salary FROM employees WHERE last_name='Taylor') ORDER BY last_name;
```

La consulta se ejecuta de manera correcta si el last\_name es único. Hay dos variaciones que se ejecutarán sin error, sin importar el valor que sea provisto.

La primera solución era:

```
SELECT last_name
FROM employees
WHERE salary > ALL
(SELECT salary FROM employees WHERE last_name='Taylor') ORDER BY last_name;
```

La segunda solución era:

```
SELECT last_name
FROM employees
WHERE salary >
(SELECT max(salary) FROM employees WHERE last_name='Taylor')ORDER BY last_name;
```

Existen otras consultas que se ejecutarían de manera correcta.

La finalidad de este tutorial es formular otras dos soluciones, una usando el operador de comparación ANY y otra usando la función de agregación MIN. Toda estas “soluciones” son, de hecho, solo formas de evitar el error. No devuelven necesariamente el resultado que el usuario quiere y quizás no son consistentes. ¿Qué cambios son necesarios para dar un resultado consistente e inequívoco?

### Solución:

Estas son dos posibles soluciones usando ANY y MIN:

```
SELECT last_name
FROM employees
WHERE salary > ANY (SELECT salary
FROM employees
WHERE last_name='Taylor')
ORDER BY last_name;
```

```
SELECT last_name
FROM employees
WHERE salary > (SELECT min(salary)
FROM employees
WHERE last_name='Taylor')
ORDER BY last_name;
```

Estas son soluciones igual de validas que las anteriores que usaban ALL y MAX, pero no devuelven el mismo resultado. No hay forma de decir que estas soluciones son mejores o peores que las anteriores. La peculiaridad de este ejercicio es que la subconsulta está basada en una columna que no es la clave primaria. No sería irracional decir que todas las soluciones son incorrectas y la consulta original es la mejor; da un resultado que es correcto si la columna LAST\_NAME es único y si LAST\_NAME no es único, lanza un error en lugar de dar una respuesta cuestionable. La respuesta correcta es la que la subconsulta debería estar basada en EMPLOYEE\_ID, no en LAST\_NAME.

## 9. Utilizando los operadores de Conjunto

### 9.1. Tutorial 1 - Describir los operadores de conjunto

En este tutorial se verá cómo utilizar los operadores de conjunto. Se podrá trabajar en SQL\*Plus o SQL Developer. Los pasos a seguir serán los siguientes:

1. Conectarse a la base de datos como usuario HR.
2. Ejecutar la consulta:

```
SELECT region_name  
FROM regions;
```

Obsérvese el resultado, en particular el orden de los registros. Si la tabla es como se creó originalmente, se devolverán cuatro registros. El orden será 1.Europa, 2.América, 3.Asia, 4.Oriente Medio y África.

3. Consúltese la tabla Regiones dos veces, usando UNION:

```
SELECT region_name  
FROM regions  
UNION  
SELECT region_name  
FROM regions;
```

Los registros devueltos serán como en el paso 1 pero ordenadas alfabéticamente.

4. Esta vez, se usará UNION ALL:

```
SELECT region_name  
FROM regions  
UNION ALL  
SELECT region_name  
FROM regions;
```

Habrá el doble de registros y no se ordenarán.

5. Usando INTERSECTION se recuperará los registros comunes a las dos consultas:

```
SELECT region_name  
FROM regions  
INTERSECT  
SELECT region_name  
FROM regions;
```

Los cuatro registros son comunes, y el resultado está ordenado.

6. Un MINUS eliminará los registros comunes:

```
SELECT region_name  
FROM regions
```

```
MINUS
SELECT region_name
FROM regions;
```

La segunda consulta eliminará todos los registros de la primera consulta. Resultado: no quedan registros.

## 9.2. Tutorial 2- Utilizando los Operadores Conjunto

En este tutorial se usarán consultas compuestas más complejas. Para ello se seguirán los siguientes pasos:

1. Conectarse a la base de datos como usuario HR.
2. Ejecutar la siguiente consulta para contar el número de empleados en tres departamentos:

```
SELECT department_id, count(1)
FROM employees
WHERE department_id IN (20,30,40)
GROUP BY department_id;
```

3. Se obtiene el mismo resultado con una consulta compuesta:

```
SELECT 20,count(1)
FROM employees
WHERE department_id=20
UNION ALL
SELECT 30,count(1)
FROM employees
WHERE department_id=30
UNION ALL
SELECT 40,count(1)
FROM employees
WHERE department_id=40;
```

4. Buscar si algún *manager* gestiona a los trabajadores en los departamentos 20 y 30 y se excluye a los *managers* con trabajadores en el departamento 40 (siguiente imagen):

```
SELECT manager_id
FROM employees
WHERE department_id=20
INTERSECT
SELECT manager_id
FROM employees
WHERE department_id=30
MINUS
SELECT manager_id
```

```
FROM employees  
WHERE department_id=40;
```

The screenshot shows a SQL Plus window with three distinct SQL statements and their results:

- Statement 1:** A query to count employees by department ID (20, 30, 40). The results are:

DEPARTMENT_ID	COUNT(1)
20	2
30	6
40	1

- Statement 2:** A compound SELECT statement that counts employees for department IDs 20, 30, and 40 separately. The results are:

20	COUNT(1)
20	2
30	6
40	1

- Statement 3:** A compound SELECT statement that finds manager IDs for departments 20, 30, and 40. The results are:

MANAGER_ID
100

5. Utilizar una consulta compuesta para mostrar los subtotales de los salarios por departamento, por *manager* y el total (siguiente imagen):

```
SELECT department_id, NULL,sum(salary)  
FROM employees  
GROUP BY department_id  
UNION  
SELECT NULL, manager_id, sum(salary)
```

```
FROM employees
GROUP BY manager_id
UNION ALL
SELECT NULL, NULL,sum(salary)
FROM
employees;
```

The screenshot shows a SQL Plus window with the following content:

```
SQL> SELECT department_id, NULL,sum(salary)
  2  FROM employees
  3  GROUP BY department_id
  4  UNION
  5  SELECT NULL, manager_id, sum(salary)
  6  FROM employees
  7  GROUP BY manager_id
  8  UNION ALL
  9  SELECT NULL, NULL,sum(salary)
 10  FROM employees;
```

DEPARTMENT_ID	NULL	SUM(SALARY)
10		4400
20		19000
30		24900
40		6500
50		156400
60		28800
70		10000
80		304500
90		58000
100		51608
110		20308
	100	155400
	101	11916
	102	9000
	103	19800
	108	39600
	114	13900
	120	22100
	121	25400
	122	23600
	123	25900
	124	23000
	145	51000
	146	51000
	147	46600
	148	51900
	149	50000
	201	6000
	205	8300
		7000
		24000
		691416

```
32 rows selected.

SQL>
```

---

### 9.3. Tutorial 3 - Controlar el orden de los registros devueltos

En este tutorial se ordenará el resultado del paso final del tutorial anterior. Este paso produjo un listado de los salarios totales por departamento y luego por manager, pero los resultados no estaban muy bien formateados. Se seguirán para ello los siguientes pasos:

1. Conectarse a la base de datos como usuario HR.
2. Generar mejores encabezados de columna para la consulta (ver la siguiente imagen)

```
SELECT department_id dept, NULL mgr, sum(salary)
FROM employees
GROUP BY department_id
UNION ALL
SELECT NULL, manager_id, sum(salary)
FROM employees
GROUP BY manager_id
UNION ALL
SELECT NULL, NULL, sum(salary)
FROM employees;
```

SQL> SELECT department\_id dept, NULL mgr, sum(salary)  
 2 FROM employees  
 3 GROUP BY department\_id  
 4 UNION ALL  
 5 SELECT NULL, manager\_id, sum(salary)  
 6 FROM employees  
 7 GROUP BY manager\_id  
 8 UNION ALL  
 9 SELECT NULL, NULL, sum(salary)  
 10 FROM employees;

DEPT	MGR	SUM(SALARY)
100		51608
30		24900
		7000
90		58000
20		19000
70		10000
110		20308
50		156400
80		304500
40		6500
60		28800
10		4400
		21000
100		155400
123		25900
120		22100
121		25400
147		46600
108		39600
148		51900
149		50000
205		8300
102		9000
201		6000
101		44916
114		13900
124		23000
145		51000
146		51000
103		19800
122		23600
		691416

32 rows selected.

SQL> ■

3. A continuación, se ordenarán los resultados de las subconsultas usando UNION en lugar de UNION ALL.

```
SELECT department_id dept, NULL mgr,  

sum(salary)  

FROM employees  

GROUP BY department_id
```

```

UNION
SELECT NULL, manager_id,
sum(salary)
FROM employees
GROUP BY manager_id
UNION
SELECT NULL, NULL, sum(salary)
FROM employees;

```

SQL> SELECT department\_id, NULL,sum(salary)  
2 FROM employees  
3 GROUP BY department\_id  
4 UNION  
5 SELECT NULL, manager\_id, sum(salary)  
6 FROM employees  
7 GROUP BY manager\_id  
8 UNION ALL  
9 SELECT NULL, NULL,sum(salary)  
10 FROM employees;

DEPARTMENT_ID	NULL	SUM(SALARY)
10		4400
20		19000
30		24900
40		6500
50		156400
60		28800
70		10000
80		304500
90		58000
100		51608
110		20308
	100	155400
	101	11916
	102	9000
	103	19800
	108	39600
	114	13900
	120	22100
	121	25400
	122	23600
	123	25900
	124	23000
	145	51000
	146	51000
	147	46600
	148	51900
	149	50000
	201	6000
	205	8300
		7000
		24000
		691416

32 rows selected.

SQL>

Este procedimiento sería correcto, sin embargo, se puede ver que el salario para el personal sin departamento o *manager* se coloca en la parte inferior del resultado, por encima del total general, no dentro de las secciones correspondientes a los departamentos y administradores. Hay maneras de mejorar este resultado. Por ejemplo, la función NVL puede usarse para poner a cero los valores nulos de DEPARTMENT\_ID y MANAGER\_ID, obteniéndose así, una salida más ordenada (ver siguiente imagen).

SQL> SELECT 'DEP: ' || to\_char(nvl(department\_id, 0)) dept, 'MAN: 0' mgr, sum(salary)  
 2 FROM employees  
 3 GROUP BY department\_id  
 4 UNION  
 5 SELECT 'DEP: 0', 'MAN: ' || to\_char(nvl(manager\_id, 0)), sum(salary)  
 6 FROM employees  
 7 GROUP BY manager\_id  
 8 UNION  
 9 SELECT 'Overall ', 'Sum: ', sum(salary)  
 10 FROM employees;

DEPT	MGR	SUM(SALARY)
DEP: 0	MAN: 0	7000
DEP: 0	MAN: 0	24000
DEP: 0	MAN: 100	155400
DEP: 0	MAN: 101	44916
DEP: 0	MAN: 102	9000
DEP: 0	MAN: 103	19800
DEP: 0	MAN: 108	39600
DEP: 0	MAN: 114	13900
DEP: 0	MAN: 120	22100
DEP: 0	MAN: 121	25400
DEP: 0	MAN: 122	23600
DEP: 0	MAN: 123	25900
DEP: 0	MAN: 124	23000
DEP: 0	MAN: 145	51000
DEP: 0	MAN: 146	51000
DEP: 0	MAN: 147	46600
DEP: 0	MAN: 148	51900
DEP: 0	MAN: 149	50000
DEP: 0	MAN: 201	6000
DEP: 0	MAN: 205	8300
DEP: 10	MAN: 0	4400
DEP: 100	MAN: 0	51608
DEP: 110	MAN: 0	20308
DEP: 20	MAN: 0	19000
DEP: 30	MAN: 0	24900
DEP: 40	MAN: 0	6500
DEP: 50	MAN: 0	156400
DEP: 60	MAN: 0	28800
DEP: 70	MAN: 0	10000
DEP: 80	MAN: 0	304500
DEP: 90	MAN: 0	58000
Overall	Sum:	691416

32 rows selected.

SQL> -

## 9.4. Tutorial 4 - Global

Trabajando en el esquema de HR, se pide formular consultas que generen resultados usando los operadores de consultas. Los resultados que se buscan son los siguientes:

1. Los empleados tienen su trabajo actual (identificado por JOB\_ID) guardado en su registro de EMPLOYEES. Los trabajos que han tenido anteriormente (pero no su trabajo actual) están guardados en JOB\_HISTORY. ¿Qué empleados no han cambiado nunca de trabajo? El listado deberá incluir únicamente los valores de EMPLOYEE\_ID y LAST\_NAME del empleado.
2. ¿Qué empleados fueron contratados en un trabajo, cambiaron a un trabajo diferente, pero trabajan actualmente en el trabajo anterior? De nuevo, se necesitará formular una consulta que compare EMPLOYEES con JOB\_HISTORY. Los resultados deberán mostrar el nombre del empleado y del trabajo. El nombre del trabajo está almacenado en la tabla JOBS.
3. ¿Cuáles son los trabajos que ha tenido un empleado? Esto será el JOB\_ID para el trabajo actual del empleado (en EMPLOYEES) y todos los trabajos previos (en JOB\_HISTORY). Si el empleado ha tenido un trabajo varias veces, no hay necesidad de mostrarlo más de una vez. Se usará una variable de sustitución para preguntar por EMPLOYEE\_ID y mostrar el nombre de todos los trabajos que ha tenido. Los empleados 101 y 200 serán los idóneos para probar.

### Solución

1. Para identificar a todos los empleados que no han cambiado de trabajo; se consulta la tabla EMPLOYEES y se quitan todos aquello que tienen algún registro en JOB\_HISTORY.

```
SELECT employee_id, last_name
FROM employees
MINUS
SELECT employee_id, last_name
FROM job_history
JOIN employees USING (employee_id);
```

2. Todos los empleados que han cambiado al menos una vez de trabajo tendrán al menos un registro en JOB\_HISTORY; para esos quienes han vuelto a un trabajo que han tenido antes, el valor de JOB\_ID en EMPLOYEES será el mismo que el JOB\_ID en uno de sus registros en JOB\_HISTORY.

```
SELECT last_name, job_title
FROM employees
JOIN jobs USING (job_id)
INTERSECT
SELECT last_name, job_title
FROM job_history h
JOIN jobs j ON (h.job_id=j.job_id)
JOIN employees e ON (h.employee_id=e.employee_id);
```

3. Esta consulta compuesta preguntará por un valor de EMPLOYEE\_ID y mostrará el trabajo actual y trabajos previos del empleado.

```
SELECT job_title
FROM jobs
JOIN employees USING (job_id)
WHERE employee_id=&&Who
UNION
SELECT job_title
FROM jobs
JOIN job_history USING (job_id)
WHERE
employee_id=&&Who;
```

## 10. Manipulando datos

### 10.1. Tutorial 1 - Uso del comando INSERT

En este tutorial se utilizarán varias técnicas para insertar registros en una tabla. Se deberán para ello seguir los siguientes pasos:

1. Conectarse al esquema HR, ya sea con SQL Developer o SQL\*Plus.
2. Consultar la tabla REGIONS para verificar los valores actuales para la columna REGION\_ID:

```
SELECT *
FROM regions;
```

En este tutorial se asume que todos los valores están por debajo de 100. En caso de que haya valores por encima de 100, se deberán ajustar para evitar conflictos de claves primarias.

3. Insertar un registro en la tabla REGIONS, proporcionando los valores que se muestran a continuación:

```
INSERT INTO regions
VALUES (101, 'Great Britain');
```

4. Insertar un registro en la tabla REGIONS, proporcionando los valores como variables de sustitución:

```
INSERT INTO regions
VALUES (&Region_number, '&Region_name');
```

Cuando se solicite, se deben introducir los valores 102 para el número y Australasia para el nombre.

5. Insertar un registro en la tabla REGIONS, calculando el REGION\_ID para que sea mayor que el valor mayor actual. Esto necesitará una subconsulta escalar (subconsulta que devuelva un único valor):

```
INSERT INTO regions  
VALUES ((SELECT max(region_id)+1 FROM regions), 'Oceania');
```

6. Confirmar que los registros se han insertado.

```
SELECT *  
FROM regions;
```

7. Hacer efectivos los cambios:

```
COMMIT;
```

La siguiente ilustración muestra los resultados del tutorial, utilizando SQL\*Plus:

The screenshot shows a Windows application window titled "SQL Plus". Inside, a SQL session is running. The user has connected to the "hr/hr" database. They first select all rows from the "regions" table, which shows four existing regions: Europe, Americas, Asia, and Middle East and Africa. Then, they insert a new row with region\_id 101 and name 'Great Britain'. After committing, they insert another row with region\_id 102 and name 'Australasia'. Finally, they select all rows again to verify the changes, and the output shows the new regions along with the previously inserted ones.

```
SQL> conn hr/hr  
Connected.  
SQL> SELECT *  
  2  FROM regions;  
  
REGION_ID REGION_NAME  
-----  
      1 Europe  
      2 Americas  
      3 Asia  
      4 Middle East and Africa  
  
SQL> INSERT INTO regions  
  2  VALUES (101,'Great Britain');  
  
1 row created.  
  
SQL> INSERT INTO regions  
  2  VALUES (&Region_number,'&Region_name');  
Enter value for region_number: 102  
Enter value for region_name: Australasia  
old   2: VALUES (&Region_number,'&Region_name')  
new   2: VALUES (102,'Australasia')  
  
1 row created.  
  
SQL> INSERT INTO regions  
  2  VALUES  
  3  ((SELECT max(region_id)+1  
  4    FROM regions),  
  5  'Oceania');  
  
1 row created.  
  
SQL> SELECT *  
  2  FROM regions;  
  
REGION_ID REGION_NAME  
-----  
      1 Europe  
      2 Americas  
      3 Asia  
      4 Middle East and Africa  
    101 Great Britain  
    102 Australasia  
    103 Oceania  
  
7 rows selected.  
  
SQL> COMMIT;  
Commit complete.  
SQL>
```

## 10.2. Tutorial 2 - Utilizar el comando UPDATE

En este tutorial se utilizarán varias técnicas para actualizar los registros en una tabla. Se asume que la tabla HR.REGIONS está en el estado que muestra la siguiente imagen. De no ser así, se deberán ajustar los valores según sea necesario.

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa
101	Great Britain
102	Australasia
103	Oceania

Se deberán seguir los siguientes pasos:

1. Conectarse al esquema HR utilizando SQL Developer o SQL\*Plus.
2. Actualizar un solo registro, identificado por la clave primaria:

```
UPDATE regions
SET region_name='Scandinavia'
WHERE region_id=101;
```

Esta declaración debe devolver el mensaje "1 fila actualizadas."

3. Actualizar un conjunto de registros, usando una condición de mayor que:

```
UPDATE regions
SET region_name='Iberia'
WHERE region_id > 100;
```

Esta declaración debe devolver el mensaje "3 filas actualizadas."

4. Actualizar un conjunto de registros, utilizando una subconsulta para seleccionar los registros y otra para proporcionar valores de dichos registros:

```
UPDATE regions
SET region_id=
  (region_id +
  (SELECT max(region_id)
   FROM regions))
WHERE region_id IN
```

```
(SELECT region_id  
FROM regions  
WHERE region_id > 100);
```

Esta declaración debe devolver el mensaje "3 filas actualizadas."

5. Confirmar el estado de los registros:

```
SELECT *  
FROM regions;
```

6. Hacer los cambios permanentes en la base de datos:

```
COMMIT;
```

La siguiente ilustración muestra el ejercicio desde SQL\*Plus:

The screenshot shows the SQL\*Plus window with the following session history:

```
SQL> UPDATE regions  
  2  SET region_name='Scandinavia'  
  3  WHERE region_id=101;  
1 row updated.  
  
SQL> UPDATE regions  
  2  SET region_name='Iberia'  
  3  WHERE region_id > 100;  
3 rows updated.  
  
SQL> UPDATE regions  
  2  SET region_id=  
  3  (region_id +  
  4  (SELECT max(region_id)  
  5    FROM regions))  
  6  WHERE region_id IN  
  7  (SELECT region_id  
  8    FROM regions  
  9    WHERE region_id > 100);  
3 rows updated.  
  
SQL> SELECT *  
  2  FROM regions;  
REGION_ID REGION_NAME  
-----  
      1 Europe  
      2 Americas  
      3 Asia  
      4 Middle East and Africa  
204 Iberia  
205 Iberia  
206 Iberia  
7 rows selected.  
  
SQL> COMMIT;  
Commit complete.  
SQL> -
```

### 10.3. Tutorial 3 - Uso del comando DELETE

En este tutorial se utilizarán varias técnicas para borrar registros de una tabla. Se asume que la tabla HR.REGIONS es como se muestra en la siguiente ilustración. Si no es así, se deberán ajustar los valores según sea necesario.

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa
204	Iberia
205	Iberia
206	Iberia

Los pasos a seguir serán los siguientes:

1. Conectarse al esquema HR utilizando SQL Developer o SQL\*Plus.
2. Eliminar un registro, usando una condición por clave primaria:

```
DELETE FROM regions  
WHERE region_id=204;
```

3. Eliminar cada registro de la tabla omitiendo para ello la cláusula WHERE de la sentencia:

```
DELETE FROM regions;
```

La sentencia **fallará** debido a una violación de las restricciones de la base de datos.

4. Eliminar los registros, seleccionándolos con una subconsulta:

```
DELETE  
FROM regions  
WHERE region_id IN  
(SELECT region_id  
  FROM regions  
 WHERE region_name='Iberia');
```

Esto devolverá el mensaje "2 rows deleted."

5. Confirmar que la tabla REGIONS contiene sus 4 registros originales:

```
SELECT *  
FROM regions;
```

6. Hacer los cambios permanentes en la base de datos:

```
COMMIT;
```

La siguiente ilustración muestra los pasos seguidos desde SQL\*Plus:

```
SQL> DELETE FROM regions
  2 WHERE region_id=204;

1 row deleted.

SQL> DELETE FROM regions;
DELETE FROM regions
*
ERROR at line 1:
ORA-02292: integrity constraint (HR.COUNTR_REG_FK) violated - child record
found

SQL> DELETE FROM regions
  2 WHERE region_id IN
  3   (SELECT region_id
  4    FROM regions
  5    WHERE region_name='Iberia');

2 rows deleted.

SQL> SELECT *
  2  FROM regions;

REGION_ID REGION_NAME
----- -----
      1 Europe
      2 Americas
      3 Asia
      4 Middle East and Africa

SQL> COMMIT;

Commit complete.

SQL>
```

#### 10.4. Tutorial 4 - Uso de comandos COMMIT y ROLLBACK

En este ejercicio, se demuestra el uso de sentencias de control de transacciones y aislamiento de transacciones. Se asume que la tabla HR.REGIONS se presenta tal y como se muestra en la siguiente ilustración. En caso contrario, se ajustarán los valores como sea necesario.

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

Para comenzar, se deberán realizar dos conexiones al esquema HR. Estas pueden ser dos sesiones de SQL\*Plus o de SQL Developer o una de cada. La siguiente tabla muestra la lista de pasos a seguir en cada sesión:

<b>PASOS</b>	<b>EN LA PRIMERA SESIÓN</b>	<b>EN LA SEGUNDA SESIÓN</b>
1	SELECT * FROM regions;	SELECT * FROM regions;
Ambas sesiones ven los mismos datos		
2	INSERT INTO regions VALUES (100,'UK');	INSERT INTO regions VALUES (101,'GB');
3	SELECT * FROM regions;	SELECT * FROM regions;
Las dos sesiones ven datos diferentes: los datos originales, mas sus propios cambios		
4	COMMIT;	
5	SELECT * FROM regions;	SELECT * FROM regions;
Una transacción ha sido publicada al resto, la otra está todavía visible solo para una sesión		
6	ROLLBACK;	ROLLBACK;
7	SELECT * FROM regions;	SELECT * FROM regions;
La transacción confirmada no se pudo anular porque ya ha sido confirmada, pero la no confirmada ya ha desaparecido por completo tras haber sido cancelada mediante la anulación de la modificación.		
8	DELETE FROM regions WHERE region_id=100;	INSERT INTO regions VALUES (101,'GB');
8		COMMIT;

9		DELETE FROM REGIONS WHERE region_id=101;
10	SELECT * FROM regions;	SELECT * FROM regions;
Cada registro eliminado sigue siendo visible en la sesión que no lo ha borrado, hasta que se haga lo siguiente:		
10	COMMIT;	COMMIT;
11	SELECT * FROM regions;	SELECT * FROM regions;
Con todas las transacciones terminadas, ambas sesiones tienen una visión consistente de la tabla.		

## 10.5. Tutorial 5 - Global

**Se deberá llevar a cabo un ejercicio en el esquema OE (Atención al cambio de esquema) que siga los siguientes pasos:**

1. Insertar un cliente en CUSTOMER, usando una función para generar un número de cliente único:

```
INSERT INTO customers
  (customer_id,cust_first_name,cust_last_name)
VALUES (
  (SELECT max(customer_id)+1
   FROM customers),
  'John','Watson');
```

2. Dar un límite de crédito igual al promedio de límite de crédito:

```
UPDATE customers
SET credit_limit=
  (SELECT avg(credit_limit)
   FROM customers)
 WHERE cust_last_name='Watson';
```

3. Crear otro cliente utilizando el que se ha generado inicialmente, pero comprobando que CUSTOMER\_ID es único:

```
INSERT INTO customers
(customer_id,cust_first_name,cust_last_name,credit_limit)
SELECT customer_id+1,cust_first_name,cust_last_name,credit_limit
FROM customers
WHERE cust_last_name='Watson';
```

4. Cambiar el nombre del segundo cliente:

```
UPDATE customers
SET cust_last_name='Ramklass',cust_first_name='Roopesh'
WHERE customer_id=
(SELECT max(customer_id)
 FROM customers);
```

5. Realizar una acción *COMMIT* para guardar la transacción:

```
COMMIT;
```

6. Determinar los CUSTOMER \_ID de los dos nuevos clientes e imposibilitar la modificación de los registros:

```
SELECT customer_id,cust_last_name
FROM customers
WHERE cust_last_name IN ('Watson','Ramklass') FOR UPDATE;
```

7. Intentar modificar uno de estos registros bloqueados **desde otra sesión**:

```
UPDATE customers
SET credit_limit=0
WHERE cust_last_name='Ramklass';
```

8. Dicho comando se colgará. En la primera sesión, liberar el bloqueo al realizar una acción de guardado:

```
COMMIT;
```

9. La segunda sesión completará ahora la acción de actualización. En la segunda sesión, eliminar ambos registros.

```
DELETE FROM customers
WHERE cust_last_name IN ('Watson','Ramklass');
```

10. En la primera sesión intentar realizar una acción de TRUNCATE sobre la tabla COSTUMERS:

```
TRUNCATE TABLE customers;
```

11. Esta acción fallará dado que hay una transacción en progreso sobre la tabla que bloqueará todos los comandos DDL. En la segunda sesión, guardar la transacción:

```
COMMIT;
```

12. La tabla COSTUMERS volverá a su estado inicial. Confirmar esta afirmación al comprobar el valor mayor de CUSTOMER \_ID:

```
SELECT max(customer_id)
FROM customers;
```

### Solución:

En la siguiente figura se ven los primeros 5 pasos del ejercicio.



The screenshot shows a window titled "SQL Plus" with a scroll bar on the right. It contains the following SQL commands and their results:

```
SQL> CONNECT oe/oe
Connected.
SQL> INSERT INTO customers(customer_id, cust_first_name,cust_last_name)
  2 VALUES(
  3   (SELECT max(customer_id) + 1
  4    FROM customers),
  5   'John','Watson');

1 row created.

SQL> UPDATE customers
  2 SET credit_limit=
  3   (SELECT avg(credit_limit)
  4    FROM customers)
  5 WHERE cust_last_name='Watson';

1 row updated.

SQL> INSERT INTO customers(customer_id, cust_first_name,cust_last_name,credit_limit)
  2 SELECT customer_id+1,cust_first_name,cust_last_name,credit_limit
  3   FROM customers
  4 WHERE cust_last_name='Watson';

1 row created.

SQL> UPDATE customers
  2 SET cust_last_name='Ramklass',
  3   cust_first_name='Roopesh'
  4 WHERE customer_id=
  5   (SELECT max(customer_id)
  6    FROM customers);

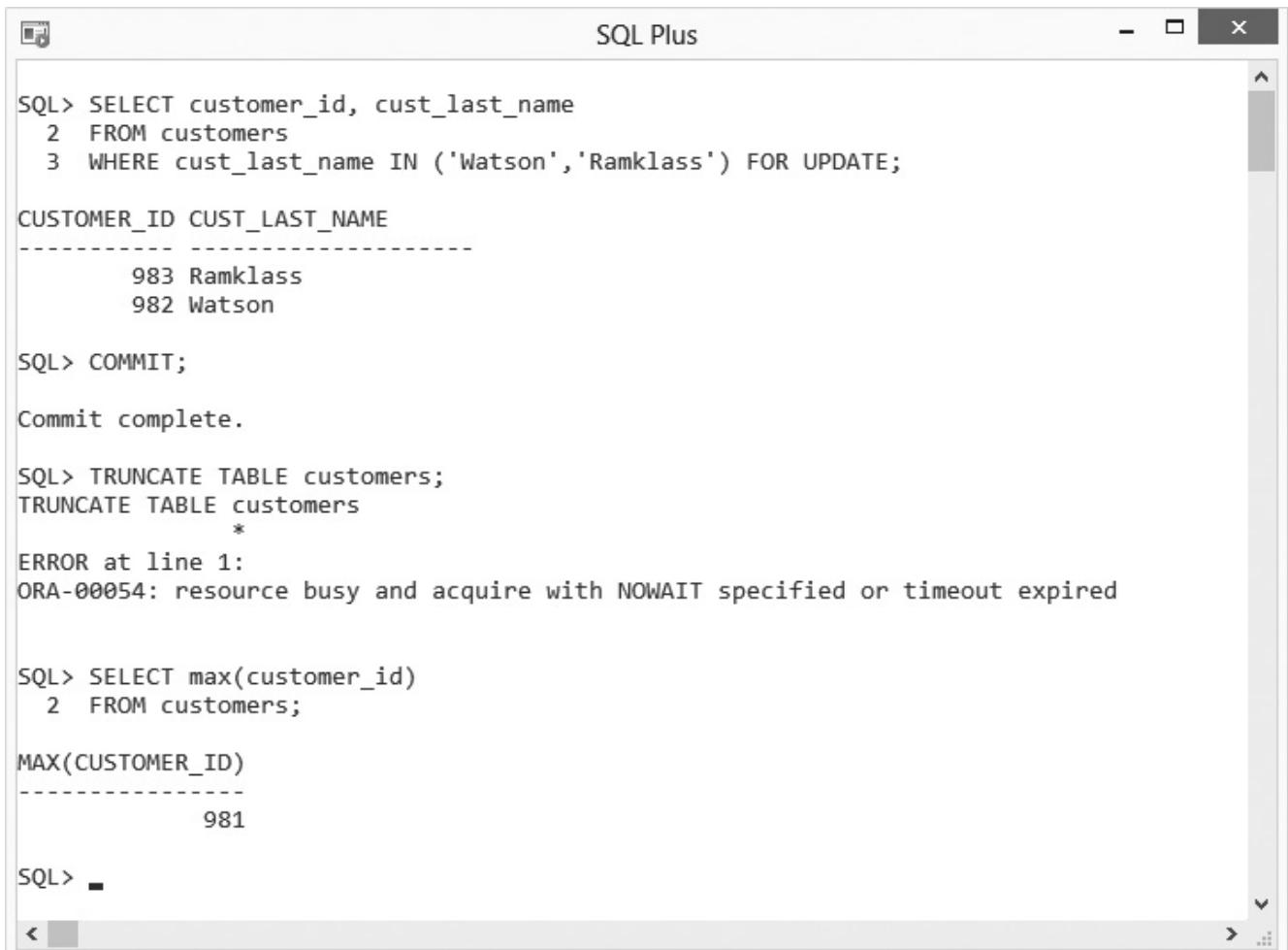
1 row updated.

SQL> COMMIT;

Commit complete.

SQL>
```

Y finalmente, en la figura que se observa a continuación, se pueden ver los últimos 7 pasos del ejercicio, desde el punto de vista de la primera sesión.



The screenshot shows a window titled "SQL Plus". Inside, the following SQL commands are displayed:

```
SQL> SELECT customer_id, cust_last_name
  2  FROM customers
  3 WHERE cust_last_name IN ('Watson','Ramklass') FOR UPDATE;
CUSTOMER_ID CUST_LAST_NAME
-----
 983 Ramklass
 982 Watson

SQL> COMMIT;
Commit complete.

SQL> TRUNCATE TABLE customers;
TRUNCATE TABLE customers
*
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired

SQL> SELECT max(customer_id)
  2  FROM customers;

MAX(CUSTOMER_ID)
-----
 981

SQL> -
```

## 11. Utilizando sentencias DDL para crear y administrar tablas

### 11.1. Tutorial 1 - Determinar qué objetos son accesibles desde la sesión de usuario

En este tutorial se pretende consultar una serie de datos con la finalidad de determinar qué objetos están en el esquema HR y a qué objetos de otros esquemas tiene acceso HR. Se deberán para ello seguir los siguientes pasos:

1. Conectarse a la base de datos con SQL\*Plus o SQL Developer como usuario HR.
2. Determinar cuántos objetos de cada tipo hay en el esquema HR:

```
SELECT object_type, count(*)
FROM user_objects
GROUP BY object_type;
```

La vista USER\_OBJECTS lista todos los objetos que pertenecen al esquema a los cuales la sesión actual está conectada, en este caso HR.

3. Determinar cuántos objetos del total de HR son accesibles:

```
SELECT object_type, count(*)
FROM all_objects
GROUP BY object_type;
```

La vista ALL\_OBJECTS lista todos los objetos para los cuales el usuario tiene algún tipo de acceso.

4. Determinar a quién pertenecen los objetos que HR puede ver:

```
SELECT DISTINCT owner
FROM all_objects;
```

## 11.2. Tutorial 2 - Investigar la Estructura de Tablas

En este tutorial se pretende consultar varias vistas del diccionario de datos como usuario HR para determinar la estructura de las tablas que lo componen. Se deberán seguir los siguientes pasos:

1. Conectarse a la base de datos con SQL\*Plus o SQL Developer como usuario HR.
2. Seleccionar los nombres y tipos de las tablas que existen en el esquema HR:

```
SELECT table_name, cluster_name, iot_type
FROM user_tables;
```

Las tablas agrupadas y las tablas organizadas por índices (IOT) son tablas de estructura avanzada. En el esquema HR, todas las tablas son tablas en pila estándar excepto COUNTRIES, que es un IOT.

3. Hacer uso del comando DESCRIBE para visualizar la estructura de una tabla:

```
DESCRIBE regions;
```

4. Recuperar información similar consultando una vista de diccionario de datos:

```
SELECT column_name, data_type, nullable
FROM user_tab_columns
WHERE table_name='REGIONS';
```

## 11.3. Tutorial 3 - Investigar los Tipos de Datos en el Esquema HR

En este ejercicio, hay que averiguar qué tipos de datos se emplean en las tablas del esquema HR, utilizando dos técnicas.

1. Conectarse a la base de datos como usuario de recursos humanos con SQL\*Plus o SQL Developer.
2. Utilizar el comando DESCRIBE para mostrar los tipos de datos en algunas tablas:

```
DESCRIBE employees;
DESCRIBE departments;
```

3. Escribir una consulta en una vista de diccionario de datos para mostrar qué columnas componen la tabla EMPLOYEES, como lo haría el comando DESCRIBE:

```
SELECT column_name, data_type, nullable, data_length, data_precision, data_scale
FROM user_tab_columns
WHERE table_name= 'EMPLOYEES' ;
```

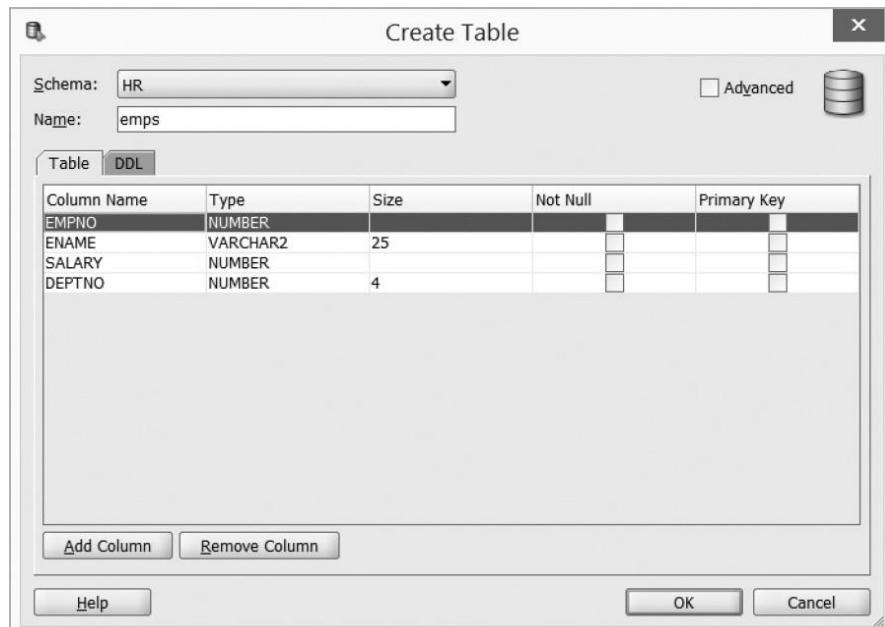
La vista USER\_TAB\_COLUMNS muestra el detalle de cada columna en cada tabla del esquema del usuario actual.

#### 11.4. Tutorial 4 - Crear Tablas

En este tutorial se utilizará SQL Developer para crear una tabla HEAP, se insertarán algunos registros con un comando y se modificará la tabla. Posteriormente, se realizarán algunas modificaciones más con SQL\*Plus y por último se borrará la tabla. Los pasos a seguir serán los siguientes:

1. Conectarse a la base de datos como usuario HR con SQL Developer.
2. Hacer clic con el botón derecho en la rama de Tablas del árbol de navegación y luego en Nueva tabla.

Se nombra la tabla como emps, y se utiliza el botón Add Column para configurarlo como en la siguiente ilustración:



3. Seleccionar la pestaña DDL para ver si la sentencia ha sido construida. Debería verse así:

```
CREATE TABLE EMPS
(
  EMPNO NUMBER,
  ENAME VARCHAR2(25),
  SALARY NUMBER,
  DEPTNO NUMBER(4, 0) );
```

En la pestaña Tabla (como en la ilustración anterior), hacer clic en Aceptar para crear la tabla.

4. Ejecutar la sentencia:

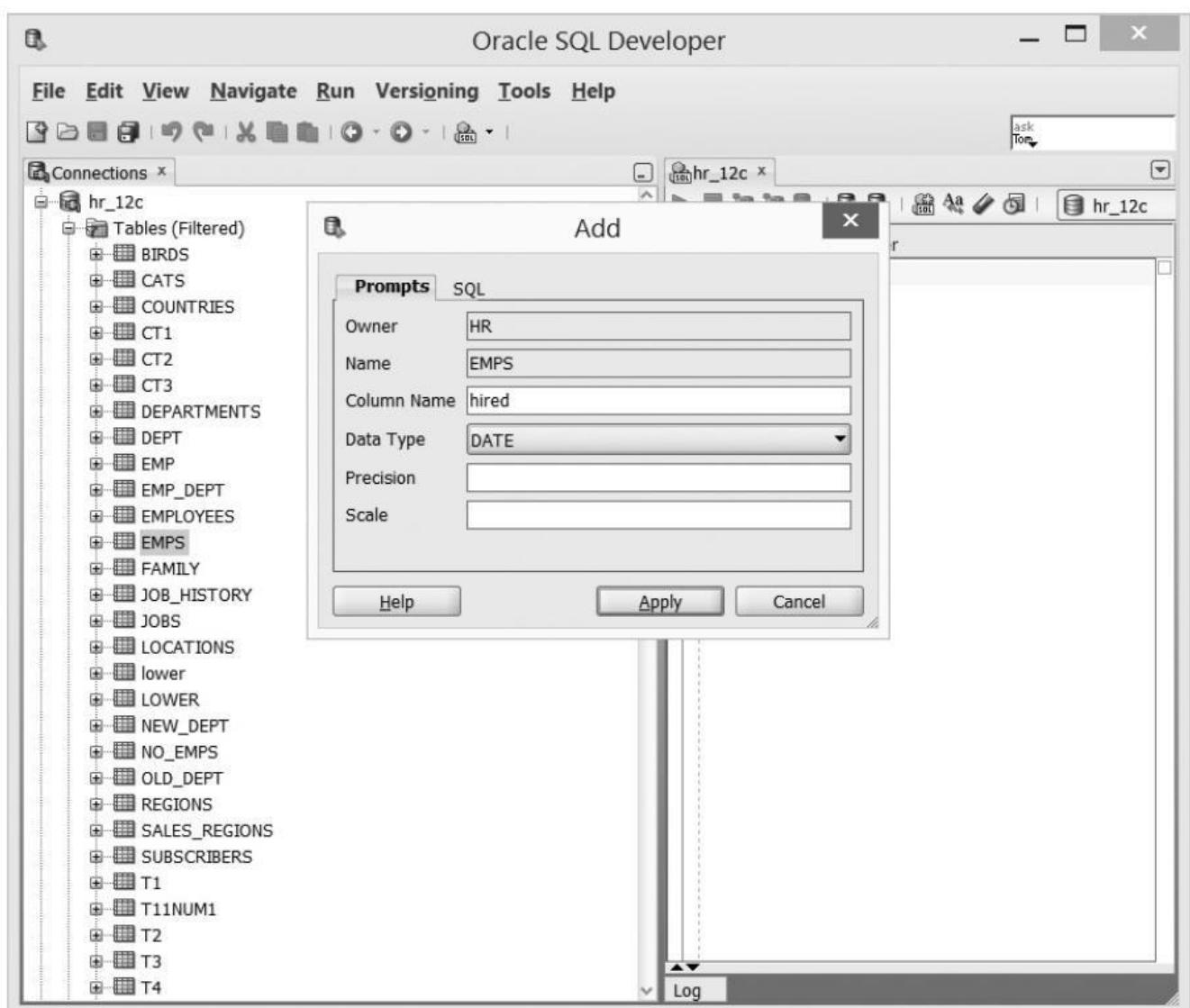
```
INSERT INTO emps
SELECT employee_id, last_name, salary, department_id
FROM employees;
```

5. Confirmar la inserción:

```
COMMIT;
```

6. Con el botón derecho del ratón, clicar en la tabla EMPS en el navegador SQL Developer, luego en Columna y en Agregar.

Se define una nueva columna HIRED, tipo DATE, como en la siguiente ilustración y clicar en Aplicar para crear la columna.



7. Definir la columna HIRED como por defecto en la tabla EMPS:

```
ALTER TABLE emps
MODIFY (hired DEFAULT sysdate);
```

8. Insertar un registro sin especificar un valor para HIRED y verificar que tenga una fecha HIRED, pero los otros registros no:

```
INSERT INTO emps (empno, ename)
VALUES (99, 'Newman');
SELECT hired, count(1)
FROM emps
```

```
GROUP BY hired;
```

9. Finalmente, borrar la tabla nueva:

```
DROP TABLE emps;
```

## 11.5. Tutorial 5 - Trabajar con Restricciones

En este tutorial se hará uso de SQL\*Plus o SQL Developer para crear tablas, añadir restricciones y demostrar el uso de estas. Los pasos a seguir serán los siguientes:

1. Conectarse a la base de datos como usuario HR.
2. Crear una tabla EMP como copia de algunas columnas de EMPLOYEES:

```
CREATE TABLE emp AS
SELECT employee_id empno, last_name ename,
department_id deptno FROM employees;
```

3. Crear una tabla DEPT como copia de algunas columnas de DEPARTMENTS:

```
CREATE TABLE dept AS
SELECT department_id deptno, department_name dname
FROM departments;
```

4. Utilizar DESCRIBE para describir la estructura de las nuevas tablas. Téngase en cuenta que la restricción no nula de ENAME y DNAME se ha transferido de las tablas fuente.
5. Añadir una restricción de clave primaria al EMP y al DEPT y una restricción de clave ajena que vincule las tablas:

```
ALTER TABLE emp
ADD CONSTRAINT emp_pk PRIMARY KEY (empno);

ALTER TABLE dept
ADD CONSTRAINT dept_pk PRIMARY KEY (deptno);

ALTER TABLE emp
ADD CONSTRAINT dept_fk FOREIGN KEY (deptno) REFERENCES dept ON
DELETE SET NULL;
```

La última restricción anterior no especifica a qué columna de DEPT hacer referencia; esto se hará por defecto en la columna clave primaria.

5. Demostrar la eficacia de las restricciones intentando insertar datos que las violen:

```
INSERT INTO dept
VALUES (10, 'New Department');
INSERT INTO emp
```

```
VALUES (9999,'New emp',99);
TRUNCATE TABLE dept;
```

6. Borrar las tablas. Téngase en cuenta que esto debe hacerse en el **orden correcto**:

```
DROP TABLE emp;
DROP TABLE dept;
```

## 11.6. Tutorial 6 - Global

Se considerará para este tutorial un análisis simple de un sistema de facturación telefónica:

- Un cliente se identifica por un número de cliente, un nombre y uno o más teléfonos.
- Un teléfono se identifica por su número, que debe ser un número entero de 7 dígitos que comience por 2 o 3. También tiene una marca de fábrica, una fecha de activación y un indicador para saber si está activo. Los teléfonos inactivos no están asignados a ningún cliente; los teléfonos activos sí.
- Para cada llamada, es necesario registrar la hora a la que comenzó y la hora a la que terminó.

La finalidad del tutorial es crear las tablas correspondientes, con restricciones y valores por defecto que puedan utilizarse para implementar este sistema.

### Solución:

Una posible solución:

- La tabla SUBSCRIBERS:

```
CREATE TABLE subscribers
(id NUMBER(4,0) CONSTRAINT sub_id_pk PRIMARY KEY,
name VARCHAR2(20) CONSTRAINT sub_name_nn NOT NULL);
```

- La tabla TELEPHONES:

```
CREATE TABLE telephones
(telno NUMBER(7,0)

CONSTRAINT tel_telno_pk PRIMARY KEY
CONSTRAINT tel_telno_ck CHECK (telno BETWEEN 2000000 AND 3999999),
activated DATE DEFAULT sysdate,
active VARCHAR2(1) CONSTRAINT tel_active_nn NOT NULL
CONSTRAINT tel_active_ck CHECK(active='Y' OR active='N'),
subscriber NUMBER(4,0) CONSTRAINT tel_sub_fk REFERENCES subscribers,
CONSTRAINT tel_active_yn CHECK((active='Y' AND subscriber IS NOT NULL)
OR (active='N' AND subscriber IS NULL))
);
```

Esta tabla tiene una restricción en la Columna ACTIVE que solo permite Y o N, dependiendo de si la columna SUBSCRIBER contiene un valor o no. Esta restricción es compleja de definir a la vez que la columna ya que referencia a otras columnas. SUBSCRIBER, si no es nula, debe coincidir con un registro de SUBSCRIBERS.

- La tabla CALLS:

```
CREATE TABLE calls
(telno NUMBER (7,0) CONSTRAINT calls_telno_fk REFERENCES telephones,
starttime DATE CONSTRAINT calls_start_nn NOT NULL,
endtime DATE CONSTRAINT calls_end_nn NOT NULL,
CONSTRAINT calls_pk PRIMARY KEY(telno, starttime),
CONSTRAINT calls_endtime_ck CHECK (endtime >= starttime));
```

Se definen dos restricciones a nivel de tabla, no a nivel de columna. La clave primaria no se puede definir en línea con una columna porque se basa en dos columnas: un teléfono puede hacer muchas llamadas, pero no dos que empiecen al mismo tiempo (al menos, no con la tecnología actual). La restricción final compara las horas de inicio y fin de la llamada y, por lo tanto, no puede definirse en línea tampoco.