

PROGRAMACION

EJERCICIO

ORACLE®
PL/SQL



ABRIL 2019

BASE DE DATOS

Indice

EJERCICIO DE PL/SQL Oracle.	3
TABLAS	4
EJERCICIO	5
PROCEDURE KLEVER_UYANA_NOMINA	6
PROCEDURE COMPROBARMES	7
FUNCTION TOTAL_EMPLEADOS	8
FUNCTION NOMBRE	8
FUNCTION SUELDO_BASE	9
FUNCTION COMPLEMENTO_KM	10
FUNCTION COMPLEMENTO_HIJOS	11
FUNCTION COMPLEMENTO_KM	12
PROCEDURE KLEVER_UYANA_NOMINA	13
SALIDA PROCEDURE KLEVER_UYANA_NOMINA	14
REFLEXIONES	15
BIOGRAFÍA	16

BASE DE DATOS

EJERCICIO DE PL/SQL Oracle.

[Dadas las tablas]

Realizar un procedimiento PL/SQL llamado 'xxxx_nomina' (donde xxxx es el nombre del alumno) de forma que saque un informe con el siguiente formato, ordenado ascendente mente por nombre de empleado.

- a) El mes de la nómina se introducirá como parámetro al procedimiento que se ha de programar y debe ser un mes válido (entre 1 y 12)
- b) El complemento km ha de realizarse añadiendo 3 céntimos de € por cada KM recorrido en el mes si es conductor.
- c) Por cada hijo que tenga el empleado se añade un 5% del sueldo base y se coloca en el campo 'Complemento Hijos'
- d) El total de la nómina es 'Sueldo Base' + 'Complemento KM' + 'Complemento Hijos'

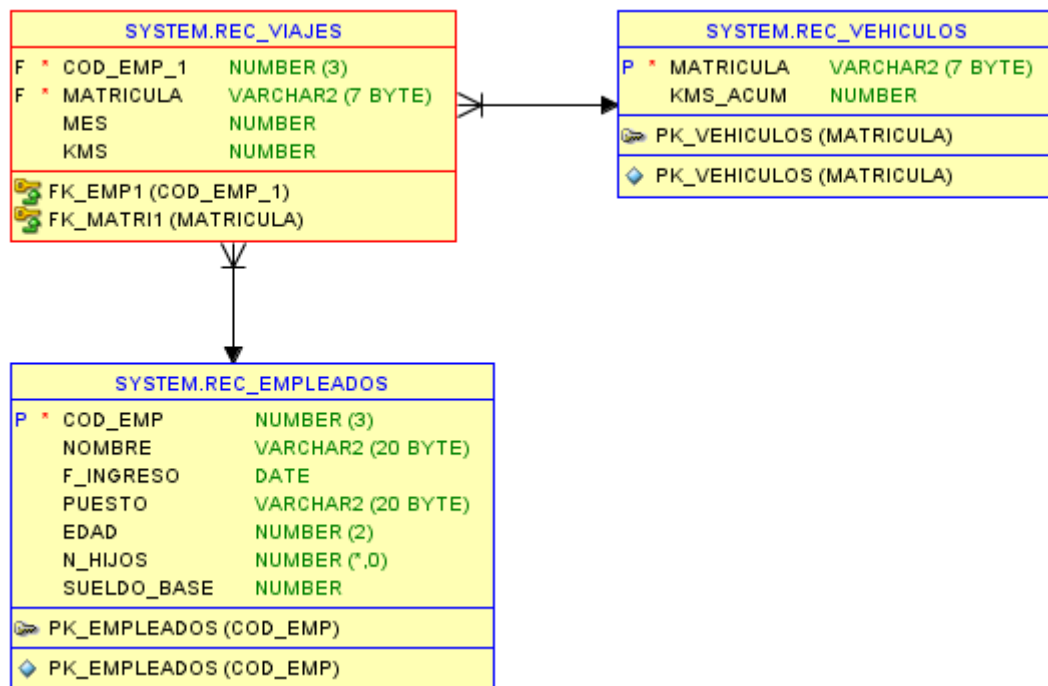
[Ejemplo de ejecución muestra :]

```
SQL> execute alumno_nomina(5);
*****
****Total Empleados tratados 3 *****
****Total Nómina 3644,5 *****
*****
Nómina del mes 5
Nombre del empleado .....Primer Empleado
Sueldo Base .....1000
Complemento KM.....4,5
Complemento Hijos .....0
Total Nómina .....1004,5
*****
Nómina del mes 5
Nombre del empleado .....Segundo Empleado
Sueldo Base .....1100
Complemento KM.....0
Complemento Hijos .....110
Total Nómina .....1210
*****
Nómina del mes 5
Nombre del empleado .....Tercer Empleado
Sueldo Base .....1300
Complemento KM.....0
Complemento Hijos .....130
Total Nómina .....1430
*****
Procedimiento PL/SQL terminado correctamente.
```

BASE DE DATOS

Tablas

Para la realización de este ejercicio se han creado tablas con datos previamente expresos para la ejecución de cada punto .



rec_empleados

klever REC_EMPLEADOS x							
Columnas Datos Model Restricciones Permisos Estadísticas Disparadores Flashback Dependencias De							
	COD_EMP	NOMBRE	F_INGRESO	PUESTO	EDAD	N_HIJOS	SUELDO_BASE
1	1	1*empleado	10/10/10	conductor	28	0	1000
2	2	2*empleado	01/01/11	conductor	45	2	1100
3	3	3*empleado	20/02/05	director	53	3	1300

rec_vehiculos

	MATRICULA	KMS_ACUM
1	2586png	10000
2	5113ycb	105000
3	7884tbc	50000

rec_viajes

	COD_EMP_1	MATRI...	MES	KMS
1	1	2586png	5	3000
2	2	7884tbc	5	2000
3	3	5113ycb	5	1000

BASE DE DATOS

Ejercicio

Realizar un procedimiento PL/SQL llamado 'xxxx_nomina' (donde xxxx es el nombre del alumno) de forma que saque un informe con el siguiente formato, ordenado ascendente mente por nombre de empleado.

Para que el informe se ejecute de forma ascendente se a modificado el orden la de la columna con ayuda de una pestaña que esta en datos (ordenar), se elige la columna y la forma de ordenar.

rec_empleados

	COD_EMP	NOMBRE	F_INGRESO	PUESTO	EDAD	N_HIJOS	SUELDO_BASE
1	1	1*empleado	10/10/10	conductor	28	0	1000
2	2	2*empleado	01/01/11	conductor	45	2	1100
3	3	3*empleado	20/02/05	director	53	3	1300

Ordenar Columnas

Ordenar Columnas

Columnas Disponibles

COD_EMP
F_INGRESO
PUESTO
EDAD
N_HIJOS
SUELDO_BASE

Columnas Seleccionadas

NOMBRE (asc)

Ascendente

Descendente

Nulos al Principio

Nulos al Final

↑

↑

↓

↓

Ordenar Historial...

Suprimir

Suprimir Todo

Seleccionar

Ayuda

Aceptar

Cancelar

BASE DE DATOS

PROCEDURE KLEVER_UYANA_NOMINA

Propuesta de procedimiento .

```
klever | REC_EMPLEADOS | KLEVER_UYANA_NOMINA x-
Código | Permisos | Detalles | Errores | Dependencias | Perfiles | Referencias
[+] [edit] [run] [test] [debug] [refresh] [undo] [redo] [find] [replace] [help]

create or replace PROCEDURE KLEVER_UYANA_NOMINA
(
    ME IN VARCHAR2
) AS
m number:=5;
cursor c_codemp is
select cod_emp from rec_empleados inner join rec_viajes on rec_empleados.cod_emp=rec_viajes.cod_emp_l where mes=me;
emp c_codemp%rowtype;
e emp.cod_emp%type;
tn rec_empleados.sueldo_base%type;
cont rec_empleados.sueldo_base%type;
error_mes exception;
BEGIN
comprobarmes(me); a llamada a un procedimiento.
cont:=0;
open c_codemp;
FETCH c_codemp into emp;
while c_codemp%FOUND loop
e:=emp.cod_emp;
tn:=total_nomina(e);
cont:=cont+tn;
FETCH c_codemp into emp;
end loop;
DBMS_OUTPUT.put_line('*****');
DBMS_OUTPUT.put_line('**** Total Empleados tratados ' || total_empleados() || ' *****')//total_empleados(me)
DBMS_OUTPUT.put_line('**** Total Nóminas ' || cont || ' *****');
DBMS_OUTPUT.put_line('*****');
close c_codemp;
open c_codemp;
FETCH c_codemp into emp;
while c_codemp%FOUND loop
e:=emp.cod_emp;
DBMS_OUTPUT.put_line('Nómina del mes '||me);
DBMS_OUTPUT.put_line('Nombre del empleado ..... ||nombre(e));
DBMS_OUTPUT.put_line('Sueldo Base ..... ||sueldo_base(e));
DBMS_OUTPUT.put_line('Complemento KM ..... ||complemento_km(e));
DBMS_OUTPUT.put_line('Complemento Hijos ..... ||complemento_hijos(e));
DBMS_OUTPUT.put_line('Total Nómina ..... ||total_nomina(e));
DBMS_OUTPUT.put_line('*****');
FETCH c_codemp into emp;
end loop;
close c_codemp;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.put_line('ERROR EN LA APLICACION');
ROLLBACK;
RAISE;
END KLEVER_UYANA_NOMINA;
```

PROCEDURE KLEVER_UYANA_NOMINA BEGIN while loop

b,c,d. llamada a funciones.

El procedimiento realiza el informe mediante la llamada a funciones y a un procedimiento que se han realizado previamente y que realizan ciertas operaciones de cada apartado del ejercicio, así como otras operaciones como nombre() que retorna el nombre del empleado entre otras.

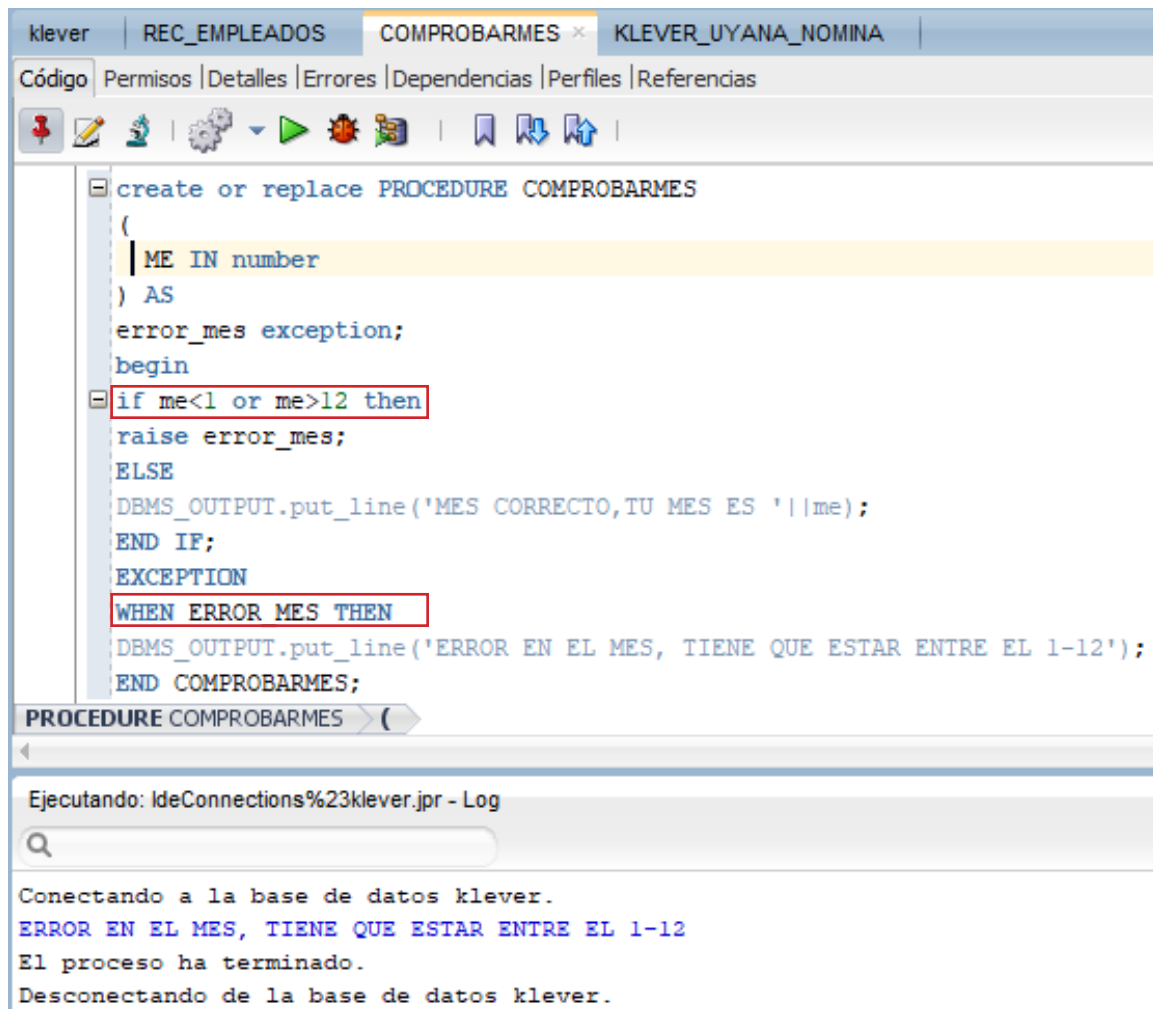
- a) El mes de la nómina se introducirá como parámetro al procedimiento que se ha de programar y debe ser un mes válido (entre 1 y 12)
- b) El complemento km ha de realizarse añadiendo 3 céntimos de € por cada KM recorrido en el mes si es conductor.
- c) Por cada hijo que tenga el empleado se añade un 5% del sueldo base y se coloca en el campo 'Complemento Hijos'
- d) El total de la nómina es 'Sueldo Base' + 'Complemento KM' + 'Complemento Hijos'

BASE DE DATOS

En este apartado iremos viendo cada una de las funciones y procedimientos que llama el procedimiento (KLEVER_UYANA_NOMINA)

PROCEDURE COMPROBARMES

a) El mes de la nómina se introducirá como parámetro al procedimiento que se ha de programar y debe ser un mes válido (entre 1 y 12)

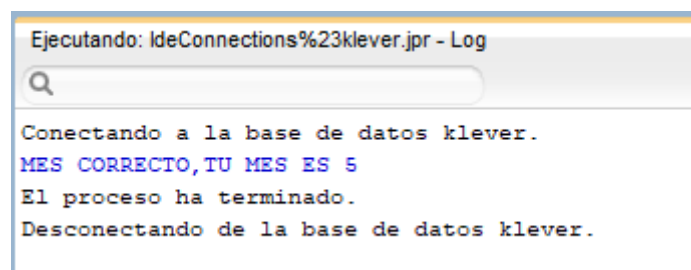


```
create or replace PROCEDURE COMPROBARMES
(
  ME IN number
) AS
error_mes exception;
begin
  if me<1 or me>12 then
    raise error_mes;
  ELSE
    DBMS_OUTPUT.put_line('MES CORRECTO,TU MES ES '||me);
  END IF;
EXCEPTION
  WHEN ERROR MES THEN
    DBMS_OUTPUT.put_line('ERROR EN EL MES, TIENE QUE ESTAR ENTRE EL 1-12');
END COMPROBARMES;
```

PROCEDURE COMPROBARMES (

Ejecutando: ldeConnections%23klever.jpr - Log

Conectando a la base de datos klever.
ERROR EN EL MES, TIENE QUE ESTAR ENTRE EL 1-12
El proceso ha terminado.
Desconectando de la base de datos klever.



```
Ejecutando: ldeConnections%23klever.jpr - Log
```

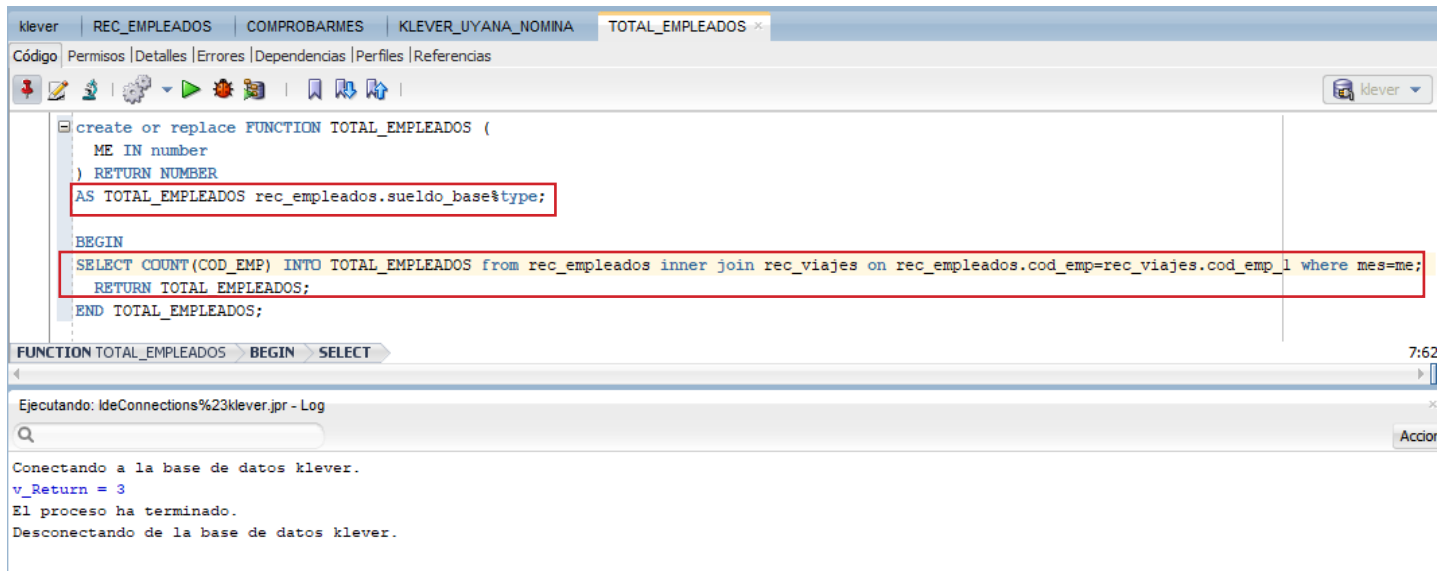
Conectando a la base de datos klever.
MES CORRECTO,TU MES ES 5
El proceso ha terminado.
Desconectando de la base de datos klever.

El procedimiento comprueba que el mes (parametro) sea el correcto , si no lo es , se a capturado una EXCEPTION con el nombre ERROR_MES, y devuelve ('ERROR EN EL MES, TIENE QUE ESTAR ENTRE EL 1-12')

BASE DE DATOS

FUNCTION TOTAL_EMPLEADOS

Esta función devuelve el numero total de empleados pasando como parámetro el mes que se ingreso, se realizo con ayuda de %TYPE Y UNA SELECT.



The screenshot shows the SQL Developer interface with the 'TOTAL_EMPLEADOS' function being edited. The function is defined as follows:

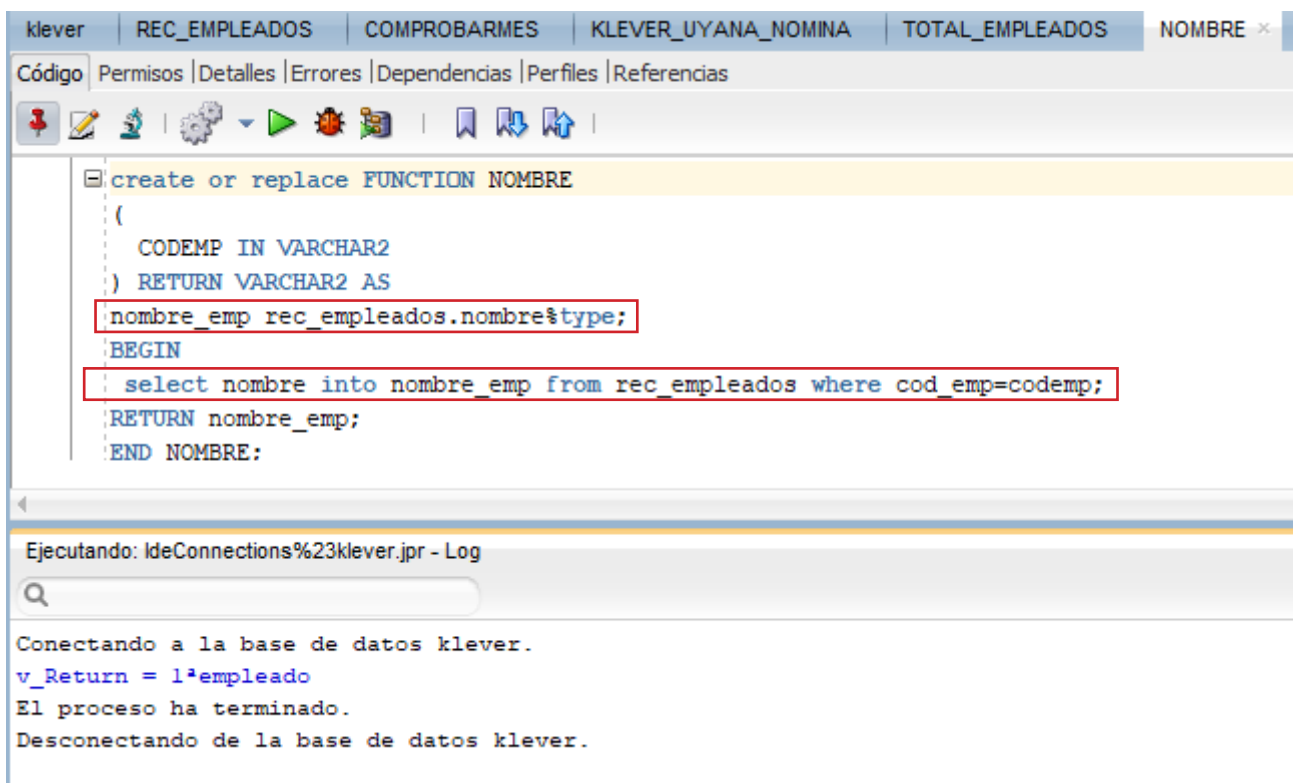
```
create or replace FUNCTION TOTAL_EMPLEADOS (  
    ME IN number  
) RETURN NUMBER  
AS TOTAL_EMPLEADOS rec_empleados.sueldo_base%type;  
  
BEGIN  
    SELECT COUNT(COD_EMP) INTO TOTAL_EMPLEADOS from rec_empleados inner join rec_viajes on rec_empleados.cod_emp=rec_viajes.cod_emp_l where mes=me;  
    RETURN TOTAL_EMPLEADOS;  
END TOTAL_EMPLEADOS;
```

The execution log at the bottom shows the following output:

```
Ejecutando: ldeConnections%23klever.jpr - Log  
Conectando a la base de datos klever.  
v_Return = 3  
El proceso ha terminado.  
Desconectando de la base de datos klever.
```

FUNCTION NOMBRE

Esta función devuelve nombre del empleado pasando como parámetro el cod_emp que se ingresa, se realizo con ayuda de %TYPE Y UNA SELECT.



The screenshot shows the SQL Developer interface with the 'NOMBRE' function being edited. The function is defined as follows:

```
create or replace FUNCTION NOMBRE  
(  
    CODEMP IN VARCHAR2  
) RETURN VARCHAR2 AS  
nombre_emp rec_empleados.nombre%type;  
  
BEGIN  
    select nombre into nombre_emp from rec_empleados where cod_emp=codemp;  
    RETURN nombre_emp;  
END NOMBRE;
```

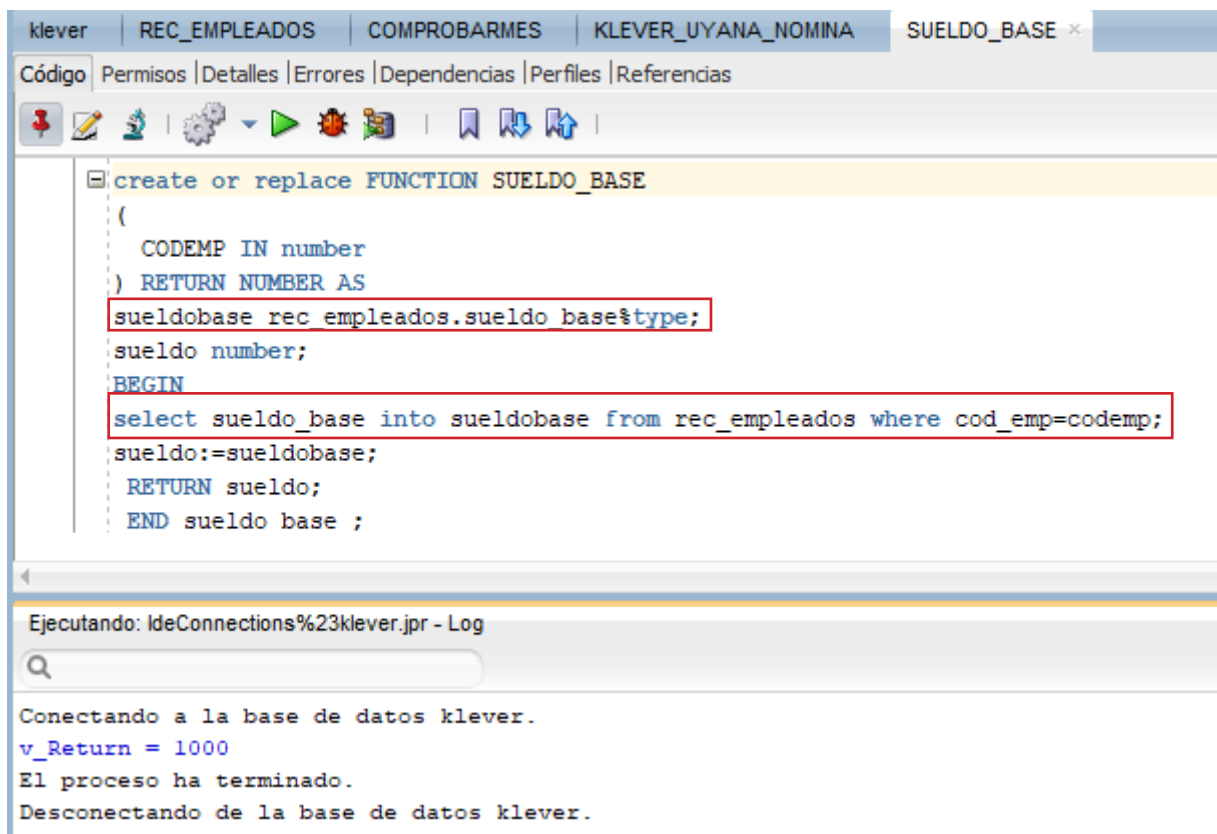
The execution log at the bottom shows the following output:

```
Ejecutando: ldeConnections%23klever.jpr - Log  
Conectando a la base de datos klever.  
v_Return = 1*empleado  
El proceso ha terminado.  
Desconectando de la base de datos klever.
```


BASE DE DATOS

FUNCTION SUELDO_BASE

Esta función devuelve el SUELDO BASE del empleado pasando como parámetro el cod_emp, se realiza con ayuda de %TYPE Y UNA SELECT.



```
create or replace FUNCTION SUELDO_BASE
(
  CODEMP IN number
) RETURN NUMBER AS
sueldobase rec_empleados.sueldo_base%type;
sueldo number;
BEGIN
select sueldo_base into sueldobase from rec_empleados where cod_emp=codemp;
sueldo:=sueldobase;
RETURN sueldo;
END sueldo base ;
```

Ejecutando: IdeConnections%23klever.jpr - Log

Conectando a la base de datos klever.
v_Return = 1000
El proceso ha terminado.
Desconectando de la base de datos klever.

BASE DE DATOS

FUNCTION COMPLEMENTO_KM

Esta función devuelve el complemento km de los empleados que son conductores pasando como parámetro el cod_emp, se realiza con ayuda de %TYPE, IF Y UNA SELECT.

- b) El complemento KM ha de realizarse añadiendo 3 céntimos de € por cada KM recorrido en el mes si el empleado es un conductor.

```
create or replace FUNCTION COMPLEMENTO_KM
(
  CODEMP IN VARCHAR2
) RETURN NUMBER AS
complemento_KM number;
kilometros rec_viajes.kms%type;
puesto rec_empleados.puesto%type;
BEGIN
select puesto into puesto from rec_empleados WHERE cod_emp=codemp;
if puesto='conductor' then
select (kms*0.03) into kilometros from rec_viajes where cod_emp_l=codemp;
complemento_km:=kilometros;
else
kilometros:=0;
RETURN kilometros;
end if;
RETURN kilometros;
END COMPLEMENTO_KM;
```

Ejecutando: IdeConnections%23klever.jpr - Log

Conectando a la base de datos klever.
v_Return = 90
El proceso ha terminado.
Desconectando de la base de datos klever.

```
Ejecutando: IdeConnections%23klever.jpr - Log
```

Conectando a la base de datos klever.
v_Return = 0
El proceso ha terminado.
Desconectando de la base de datos klever.

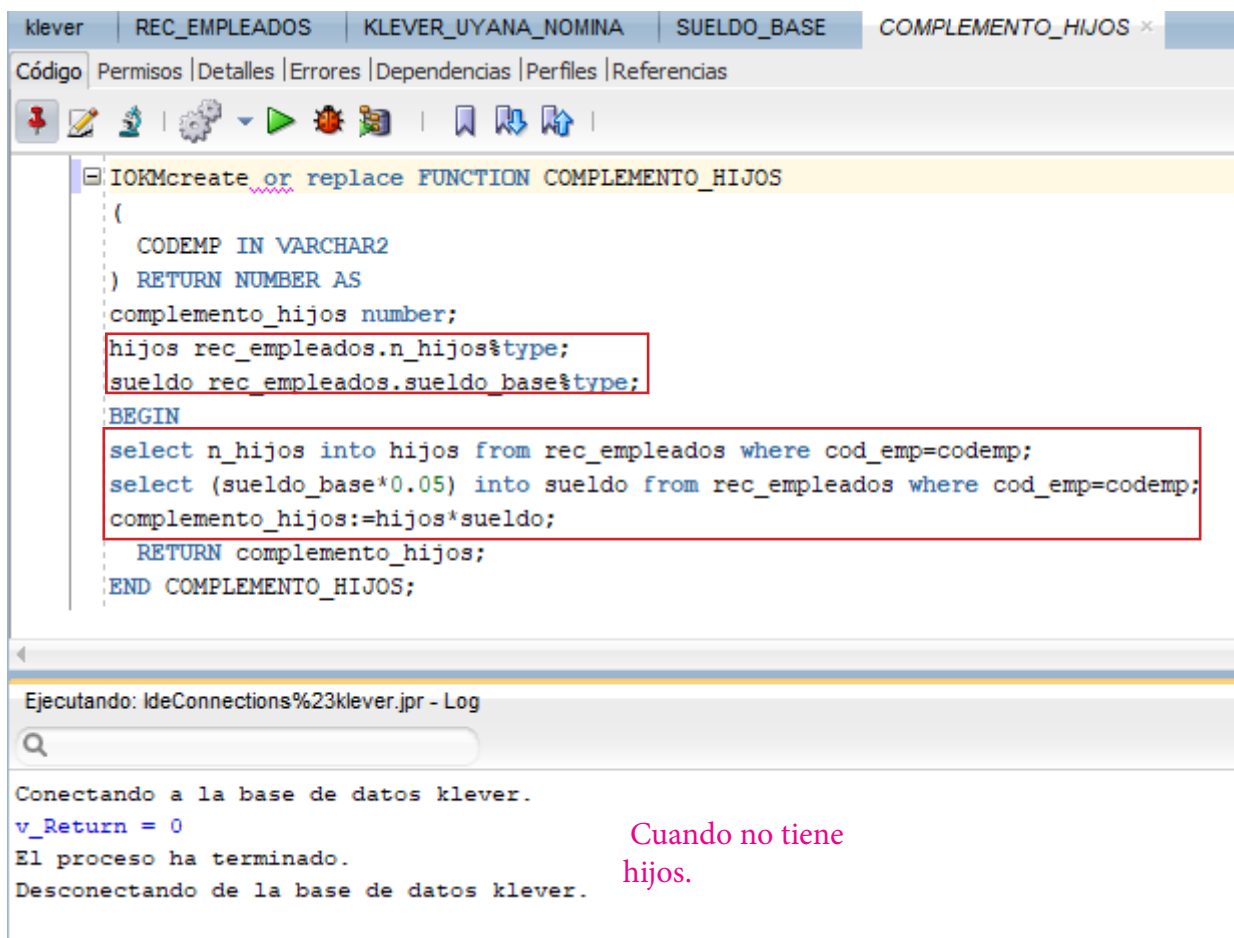
si no es conductor.

BASE DE DATOS

FUNCTION COMPLEMENTO_HIJOS

Esta función devuelve el complemento hijos de los empleados que tienen hijos pasando como parámetro el `cod_emp`, se realiza con ayuda de `%TYPE`, `SELECT`, y una operación $\text{complemento_hijos} = \text{hijos} * (\text{sueldo_base} * 0,05)$.

- c) Por cada hijo que tenga el empleado se añade un 5% del sueldo base y se coloca en el campo 'Complemento Hijos'



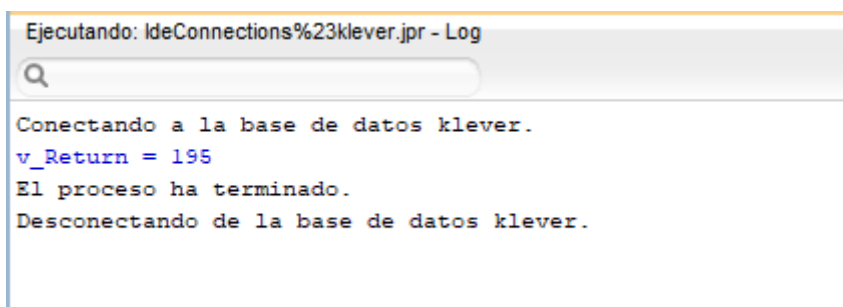
The screenshot shows an IDE window with a tab titled 'COMPLEMENTO_HIJOS'. The code editor displays the following PL/SQL code:

```
10KMccreate or replace FUNCTION COMPLEMENTO_HIJOS
(
  CODEMP IN VARCHAR2
) RETURN NUMBER AS
complemento_hijos number;
hijos rec_empleados.n_hijos%type;
sueldo rec_empleados.sueldo_base%type;
BEGIN
select n_hijos into hijos from rec_empleados where cod_emp=codemp;
select (sueldo_base*0.05) into sueldo from rec_empleados where cod_emp=codemp;
complemento_hijos:=hijos*sueldo;
RETURN complemento_hijos;
END COMPLEMENTO_HIJOS;
```

Below the code editor is a log window titled 'Ejecutando: ldeConnections%23klever.jpr - Log'. It contains the following text:

```
Conectando a la base de datos klever.
v_Return = 0
El proceso ha terminado.
Desconectando de la base de datos klever.
```

Cuando no tiene hijos.



The screenshot shows a log window titled 'Ejecutando: ldeConnections%23klever.jpr - Log'. It contains the following text:

```
Conectando a la base de datos klever.
v_Return = 195
El proceso ha terminado.
Desconectando de la base de datos klever.
```

si tiene hijos.

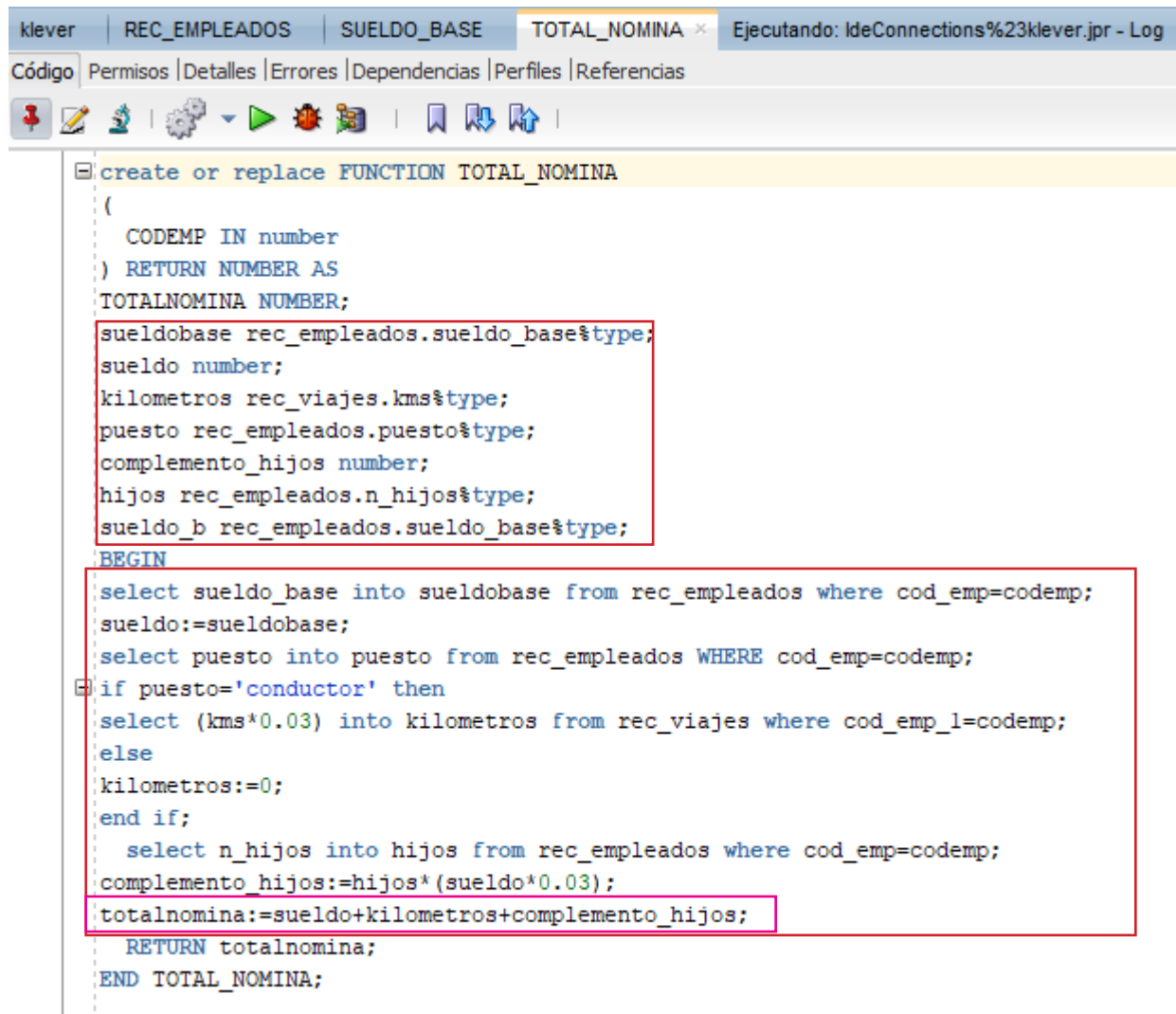
BASE DE DATOS

FUNCTION TOTAL_NOMINA

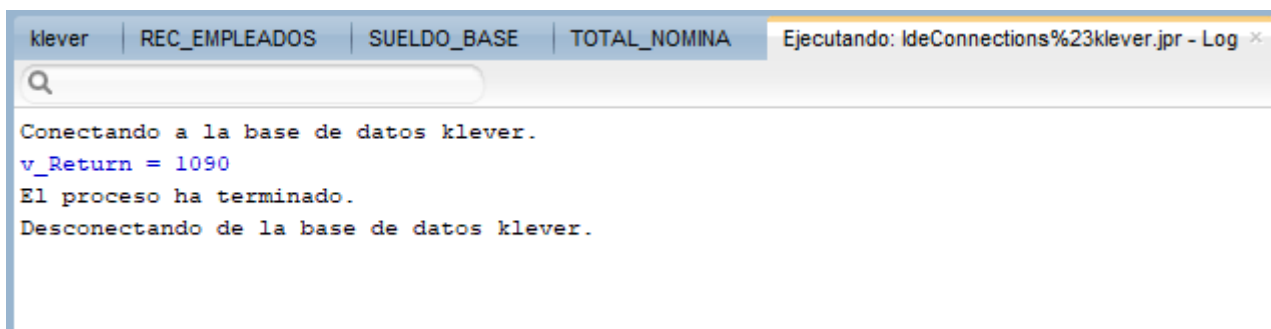
Esta función devuelve el total de la nómina de los empleados pasando como parámetro el cod_emp, se realiza con ayuda de %TYPE, SELECT IF, y una operación
 $\text{totalnomina} = \text{sueldo} + \text{kilometros} + \text{complemento_hijos}$;

En esta función se han combinado algunos códigos de las anteriores funciones

d) El total de la nómina es 'Sueldo Base' + 'Complemento KM' + 'Complemento Hijos'



```
create or replace FUNCTION TOTAL_NOMINA
(
    CODEMP IN number
) RETURN NUMBER AS
TOTALNOMINA NUMBER;
sueldobase rec_empleados.sueldo_base%type;
sueldo number;
kilometros rec_viajes.kms%type;
puesto rec_empleados.puesto%type;
complemento_hijos number;
hijos rec_empleados.n_hijos%type;
sueldo_b rec_empleados.sueldo_base%type;
BEGIN
select sueldo_base into sueldobase from rec_empleados where cod_emp=codemp;
sueldo:=sueldobase;
select puesto into puesto from rec_empleados WHERE cod_emp=codemp;
if puesto='conductor' then
select (kms*0.03) into kilometros from rec_viajes where cod_emp_l=codemp;
else
kilometros:=0;
end if;
select n_hijos into hijos from rec_empleados where cod_emp=codemp;
complemento_hijos:=hijos*(sueldo*0.03);
totalnomina:=sueldo+kilometros+complemento_hijos;
RETURN totalnomina;
END TOTAL_NOMINA;
```



```
Conectando a la base de datos klever.
v_Return = 1090
El proceso ha terminado.
Desconectando de la base de datos klever.
```

BASE DE DATOS

PROCEDURE KLEVER_UYANA_NOMINA

Propuesta de procedimiento .

```
klever | REC_EMPLEADOS | KLEVER_UYANA_NOMINA x-
Código | Permisos | Detalles | Errores | Dependencias | Perfiles | Referencias
[+] [Pencil] [Bug] [Gears] [Play] [Stop] [Refresh] [Print] [Share]
[+] create or replace PROCEDURE KLEVER_UYANA_NOMINA
(
  ME IN VARCHAR2
) AS
m number:=5;
cursor c_codemp is
select cod_emp from rec_empleados inner join rec_viajes on rec_empleados.cod_emp=rec_viajes.cod_emp_l where mes=me;
emp c_codemp%rowtype;
e emp.cod_emp%type;
tn rec_empleados.sueldo_base%type;
cont rec_empleados.sueldo_base%type;
error_mes exception;
BEGIN
comprobarmes(me);
cont:=0;
open c_codemp;
FETCH c_codemp into emp;
while c_codemp%FOUND loop
e:=emp.cod_emp;
tn:=total_nomina(e);
cont:=cont+tn;
FETCH c_codemp into emp;
end loop;
DBMS_OUTPUT.put_line('*****');
DBMS_OUTPUT.put_line('**** Total Empleados tratados '|| total_empleados() ||' *****');
DBMS_OUTPUT.put_line('**** Total Nóminas ' ||cont||' *****');
DBMS_OUTPUT.put_line('*****');
close c_codemp;
open c_codemp;
FETCH c_codemp into emp;
while c_codemp%FOUND loop
e:=emp.cod_emp;
DBMS_OUTPUT.put_line('Nómina del mes '||me);
DBMS_OUTPUT.put_line('Nombre del empleado ..... '||nombre(e));
DBMS_OUTPUT.put_line('Sueldo Base ..... '||sueldo_base(e));
DBMS_OUTPUT.put_line('Complemento KM..... '||complemento_km(e));
DBMS_OUTPUT.put_line('Complemento Hijos ..... '||complemento_hijos(e));
DBMS_OUTPUT.put_line('Total Nómina ..... '||total_nomina(e));
DBMS_OUTPUT.put_line('*****');
FETCH c_codemp into emp;
end loop;
close c_codemp;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.put_line('ERROR EN LA APLICACION');
ROLLBACK;
RAISE;
END KLEVER_UYANA_NOMINA;
```

En el procedimiento se a utilizado un cursor en donde se realizo una select que recoge los cod_emp.

Con ayuda de un WHILE, LOOP FETCH se a calculado la suma de todas los nominas de los empleados del mes ingresado (cont = total de las nominas) Cont se imprimirá en el encabezado del informe.

También nos a servido el WHILE y el cursor para hacer la impresión de la información de cada uno de los empleados.

Y para finalizar se le a añadido una EXCEPTION por si sucede algún inconveniente.

BASE DE DATOS

SALIDA PROCEDURE KLEVER_UYANA_NOMINA

Resultado poniendo el mes=5 y mes=0.

```
klever | REC_EMPLEADOS | KLEVER_UYANA_NOMINA | Ejecutando: IdeConnections%23klever.jpr - Log x
Q
Conectando a la base de datos klever.
MES CORRECTO,TU MES ES 5
*****
**** Total Empleados tratados 3 ****
**** Total Nóminas 3733 ****
*****
Nómina del mes 5
Nombre del empleado ..... 1ºempleado
Sueldo Base ..... 1000
Complemento KM..... 90
Complemento Hijos ..... 0
Total Nómina ..... 1090
*****
Nómina del mes 5
Nombre del empleado ..... 2ºempleado
Sueldo Base ..... 1100
Complemento KM..... 60
Complemento Hijos ..... 110
Total Nómina ..... 1226
*****
Nómina del mes 5
Nombre del empleado ..... 3ºempleado
Sueldo Base ..... 1300
Complemento KM..... 0
Complemento Hijos ..... 195
Total Nómina ..... 1417
*****
El proceso ha terminado.
Desconectando de la base de datos klever.
```

```
klever | REC_EMPLEADOS | KLEVER_UYANA_NOMINA | Ejecutando: IdeConnections%23klever.jpr - Log x
Q
Conectando a la base de datos klever.
ERROR EN EL MES, TIENE QUE ESTAR ENTRE EL 1-12
*****
**** Total Empleados tratados 0 ****
**** Total Nóminas 0 ****
*****
El proceso ha terminado.
Desconectando de la base de datos klever.
```

BASE DE DATOS

Reflexiones

Quando me dispuse a instalar el programa Oracle EX 18c y el Oracle SQL Developer me daba problemas a la hora de realizar una conexión, buscando en youtube encontré muy pocas soluciones, al parecer es un problema muy frecuente, al parecer el Oracle SQL Developer no se sincroniza automáticamente, hay que entrar con ayuda de Ejecutar a services.msc y cambiar todo lo relacionado a Oracle en en Manual o automático.

Otro problema que tuve es en que yo intentaba realizar llamadas a funciones desde otra , busque por foros y al parece si se puede.

También me resulto complicado implementar un cursor, pero al final lo pude realizar, aunque solo sea de un solo campo.

Encontré en la pagina oficial Oracle llamada <https://livesql.oracle.com/> un wedapp en donde se pueden realizar pruebas si tener que instalar nada en tu ordenador , ademas de realizarlo todo de una forma mas interactiva y gráfica, esto puede ser muy util para gente que quiere aprender ya que lo complementa con tutoriales de la comunidad Oracle.

La realización de tablas lo realice con el lenguaje SQL, y la inserción de forma gráfica con Oracle Developer

```

create table rec_empleados(
Cod_emp number(3) not null,
Nombre varchar2(20) null,
F_ingreso date null,
Puesto varchar(2) null,
Edad number(2) null,
N_hijos Smallint null,
Sueldo_Base number null,
constraint pk_empleados primary key (Cod_emp)
)

create table rec_vehiculos(
Matricula varchar2(7) not null,
Kms_acum number null,
constraint pk_vehiculos primary key (Matricula)
)

create table rec_viaje(
Cod_emp number(3) not null,
Matricula varchar(7) not null,
Mes number null,
Kms number null,
constraint fk_empl foreign key (Cod_emp) references rec_empleados(Cod_emp),
constraint fk_matril foreign key (Matricula) references rec_vehiculos(Matricula))

```

	MATRICULA	KMS_ACUM
1	2586png	10000
2	5113ycb	105000
3	7884tbc	50000
+4	(null)	(null)

BASE DE DATOS

Biografía

En estos link encontré la información necesaria para poder realizar este Ejercicio.

TUTORIAL PL/SQL

<https://www.tutorialspoint.com/plsql/>

PRACTICAR EN ORACLE LIVE SQL

<https://livesql.oracle.com/>

Oracle PLSQL Basico - Introduccion Curso Youtube

<https://www.youtube.com/watch?v=7SZ7slB1kyw&list=PL7E27A95750BB1C8B>

Inta

Clases PL/SQL PROFESOR PEDRO CAMACHO UNIVERSIDAD EUROPEA DE MADRID

https://www.youtube.com/watch?v=bwwhM8Un20A&list=PLTt_5WPJqE-l90KDcxe1njAvDqwhrvbLV

Descargar e Instalar Oracle Database XE 11g + SQL Developer- Windows 64 Bits

<https://www.youtube.com/watch?v=GAsCwlGlGAw&t=643s>

REFLEXIÓN FINAL

Aunque el ejercicio se a realizado de forma exitosa, PL/SQL a sido desde un principio un verdadero reto, he tenido que echarle tiempo para resolver algunos de los apartados de una forma que se viera bien planteado a nivel estructural.

Sin duda he aprendido mucho sobre las bases de datos de Oracle así como su lenguaje procedural PL/SQL.

He creado un repositorio en GITHUB para este trabajo, en el se encuentran archivos SQL para poder comprobar el código.

<https://github.com/klipsmf4/Base-de-datos>