|  |  |  |  |
| --- | --- | --- | --- |
| **Profesor:** | María Dolores Cuadra Fernández | **Grupo** | 80 |
| **Alumno/a:** | Sofía Najarro Almaraz | **NIA:** | 100363545 |
| **Alumno/a:** | Álvaro González de la Vega | **NIA:** | 100363686 |
| **Alumno/a:** | Jaime de Esteban Uranga   |  | | --- | |  | | **NIA:** | 100363590 |

# Introducción

Una introducción que sea el punto de partida del trabajo y sirva para analizar el problema que se va a resolver, fije los objetivos que se persiguen, y describa la estructura del documento.

# Consultas

Para cada consulta, describe:

1. su diseño en álgebra relacional
2. su implementación en SQL (indica el número de filas del resultado)
3. las pruebas realizadas para demostrar que funciona correctamente

Adjunta un script (query.sql) con las sentencias SQL.

-----------------------------------------------------------------------------------------------------

# *Los 10 vehículos más 'observados' en el transcurso del día de hoy.*

1. *Álgebra Relacional*

σ ROWNUM<11(π nplate as top\_observed\_vehicles (σ fecha\_odatetime=fecha\_SYSDATE GROUP BYnPlate

ORDER BY COUNT(nPlate) DESC (OBSERVATIONS ) ))

**Nota:**

Para obtener la fecha de hoy usamos SYSDATE y sabemos que tiene el siguiente formato:

 "YYYY-MM-DD HH:MM:SS". Como la consulta tiene que ser en el transcurso de hoy, solo necesitamos la fecha. Se obtiene de la siguiente forma: SUBSTR((TO\_TIMESTAMP(SYSDATE,'YYYY-MM-DDHH24:MI:SS.FF2')), 1, 8). Hemos convertido SYSDATE en el tipo TIMESTAMP para luego compararlo con odatetime de la tabla OBSERVATIONS. Lo mismo ocurre con las fechas (odatetime) de las observaciones (OBSERVATIONS), es decir, haremos SUBSTR(odatetime, 1, 8).

b) *Implementación en SQL.*

*|SELECT \* FROM (*

*|SELECT DISTINCT nPlate AS top\_observed\_vehicles FROM OBSERVATIONS*

*|WHERE SUBSTR(odatetime, 1, 8)=SUBSTR((TO\_TIMESTAMP(SYSDATE,'YYYY-MM-|DDHH24:MI:SS.FF2')), 1, 8)*

*|GROUP BY nPlate ORDER BY COUNT(nPlate) DESC)*

*|WHERE ROWNUM<11*;

Si se quiere ver más visual cuántas veces se repiten el avistamiento de un mismo coche, implemente esto:

*|SELECT \* FROM (*

*|SELECT DISTINCT nPlate AS top\_observed\_vehicles,COUNT(nPlate) AS times\_seen |FROM OBSERVATIONS*

*|WHERE SUBSTR(odatetime, 1, 8)=SUBSTR((TO\_TIMESTAMP(SYSDATE,'YYYY-|MM-DDHH24:MI:SS.FF2')), 1, 8)*

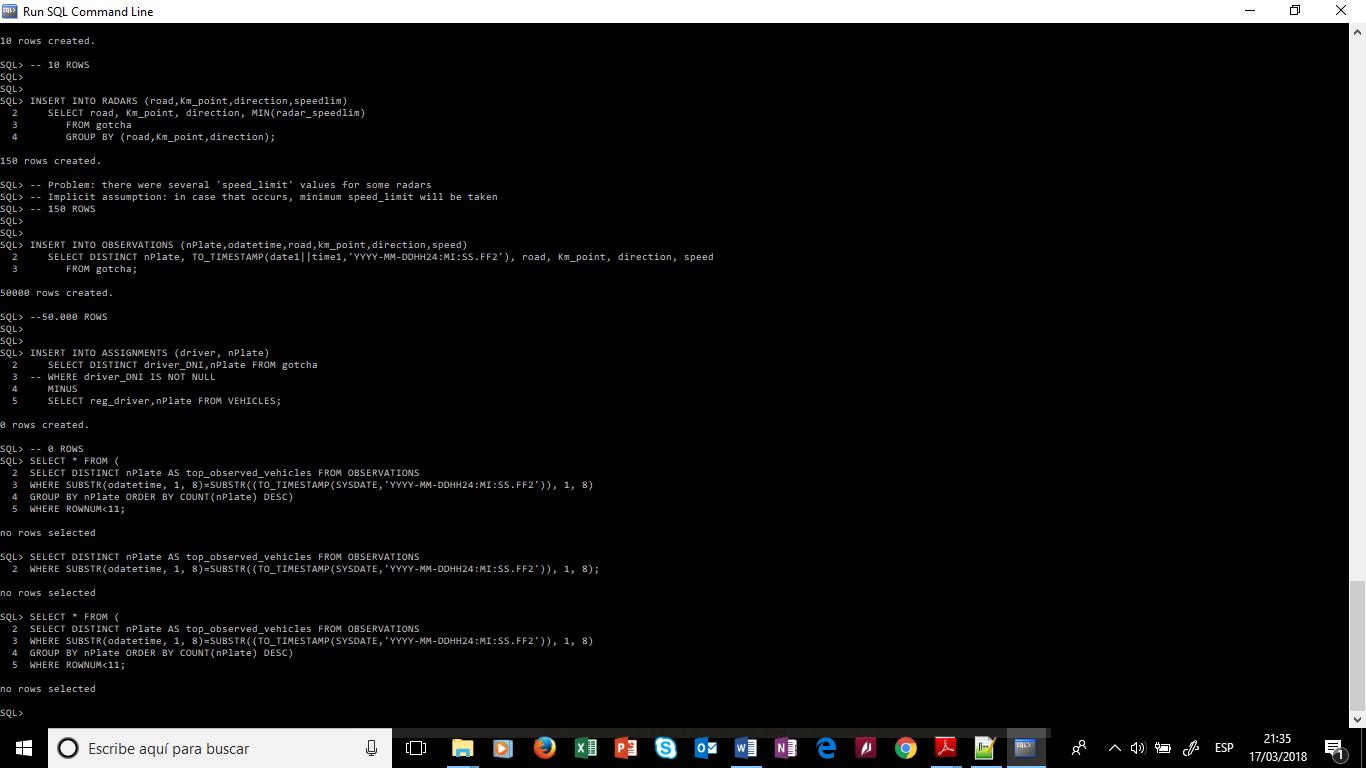
*|GROUP BY nPlate ORDER BY COUNT(nPlate) DESC)*

*|WHERE ROWNUM<11;*

**Resultado**: Ninguna fila seleccionada.

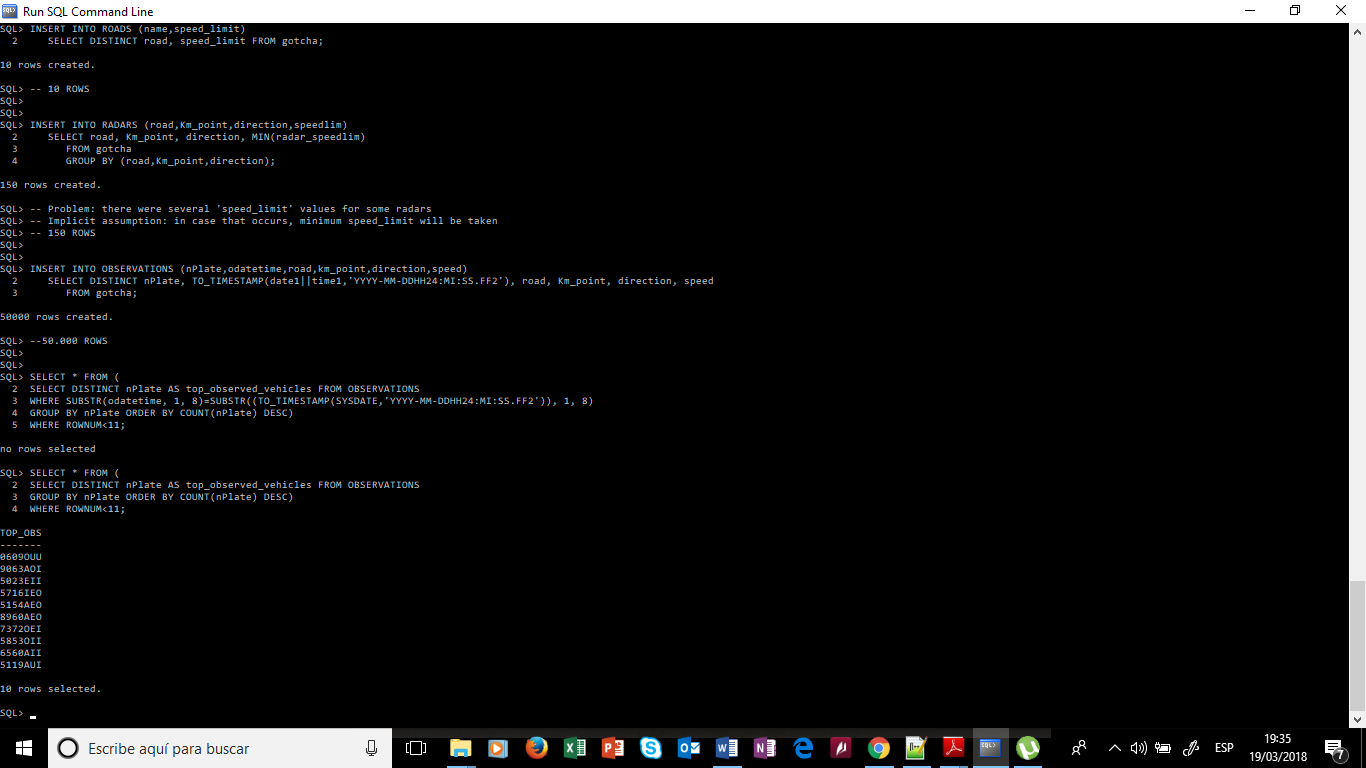
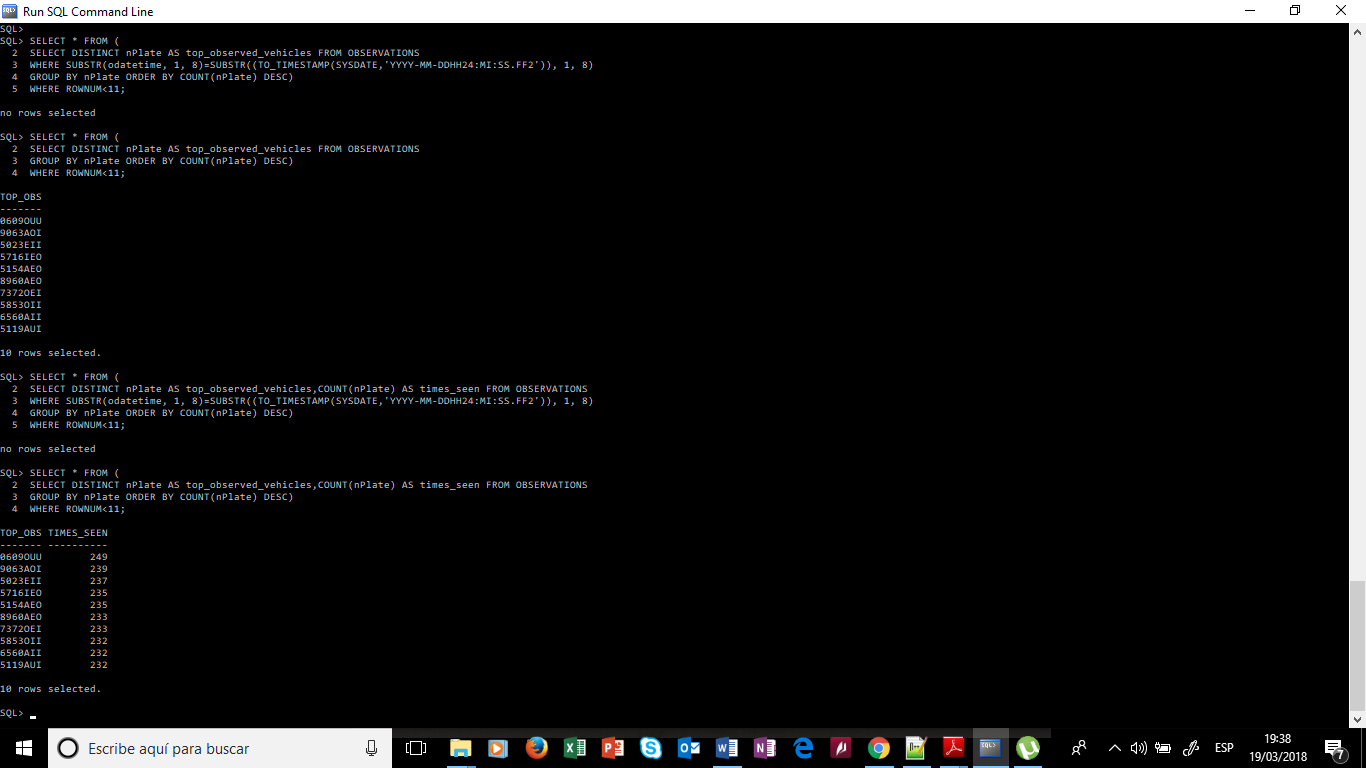
c) *Pruebas*

Cargamos los archivos dados (corrección Práctica 1) en este orden: creation-megatable-english.sql, NEWCreation-Eng.sql, gotcha.sql, NEWload-Eng-FROM-GOTCHA.sql y query.sql.



Efectivamente, son datos de otros años, no hay ningún vehículo observado hoy y menos podemos consultar los tops 10 más observados.

Si quitamos la condición de fecha, obtendríamos los 10 vehículos más observados:

# *Listado de carreteras y su valor de velocidad promedio establecida, ordenado de mayor a menor velocidad en primera instancia y por orden alfabético de carreteras en segunda, contando ambos sentidos.*

*a) Álgebra Relacional*

π name as Roads, speed\_limit (GROUP BY speed\_limit, name

ORDER BY speed\_limit DESC, name ASC (ROADS))

**Nota:**

Lo que nos dicen con ese enunciado es que se ordenan de mayor a menor velocidad y si hay carreteras que tienen las mismas velocidades se hace por orden alfabético.

b) *Implementación en SQL.*

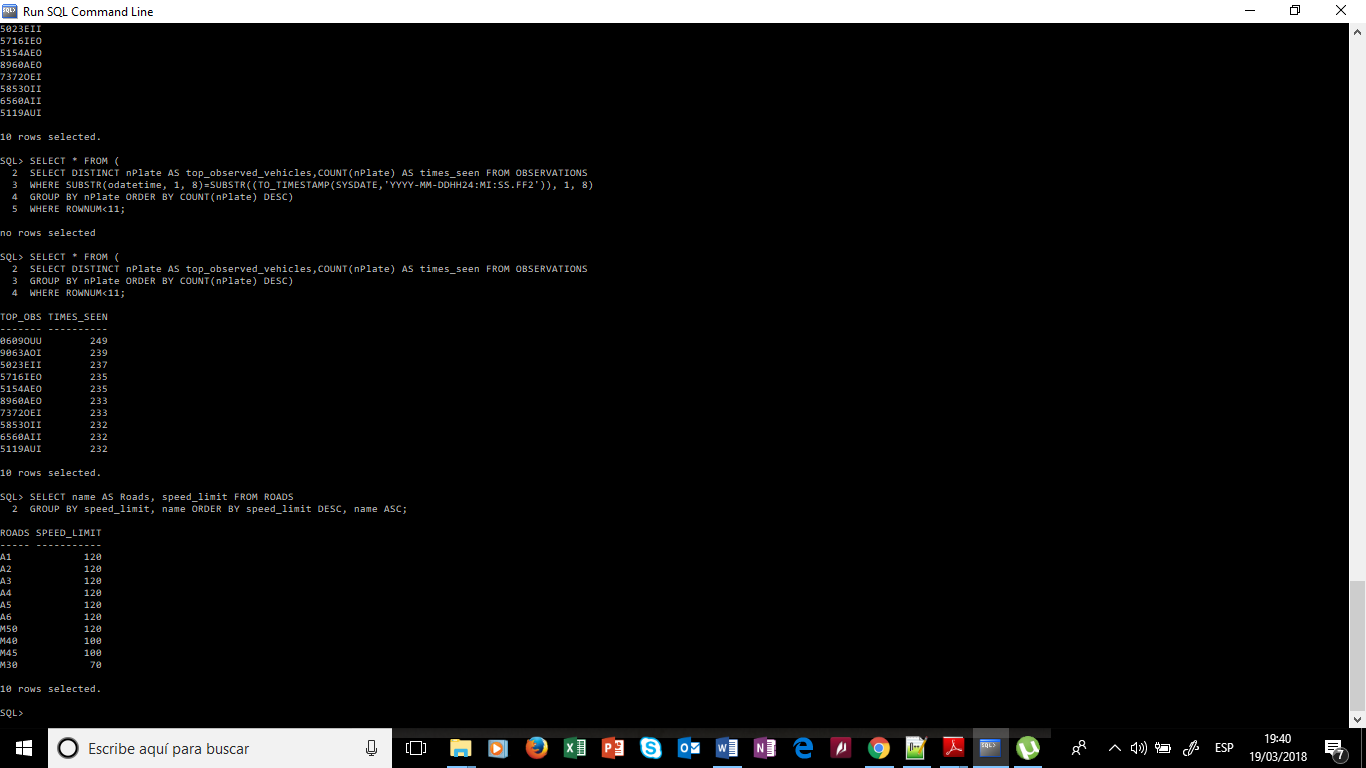
*|SELECT name AS Roads, speed\_limit FROM ROADS*

*|GROUP BY speed\_limit, name ORDER BY speed\_limit DESC, name ASC;*

**Resultado**: 10 filas seleccionadas.

c) *Prueba*

Cargamos los archivos en el orden explicado en la anterior consulta.

Observamos que lo hace correctamente.

# *Personas que no conducen ninguno de sus vehículos (ni como conductor habitual ni como conductor adicional).*

*a) Álgebra Relacional*

π owner (GROUP BY owner (π owner (VEHICLES A ▷B.reg\_driver σ A. nPlate=B.nPlate(VEHICLES B) ∩ VEHICLES C ▷D.driver σ C. nPlate=D.nPlate(ASSIGNMENTS D) )))

**Nota:**

Hemos hecho una intersección de dos consultas. La primera abarca los dueños que no son conductores habituales de sus vehículos y la segunda, dueños que no son conductores adicionales de sus vehículos.

b) *Implementación en SQL.*

*|SELECT owner FROM (*

*|(SELECT owner FROM VEHICLES A*

*|WHERE A.owner NOT IN (SELECT B.reg\_driver FROM VEHICLES B*

*|WHERE A.nPlate= B.nPlate))*

*|INTERSECT (SELECT owner FROM VEHICLES C*

*|WHERE C.owner NOT IN (SELECT D.driver FROM ASSIGNMENTS D*

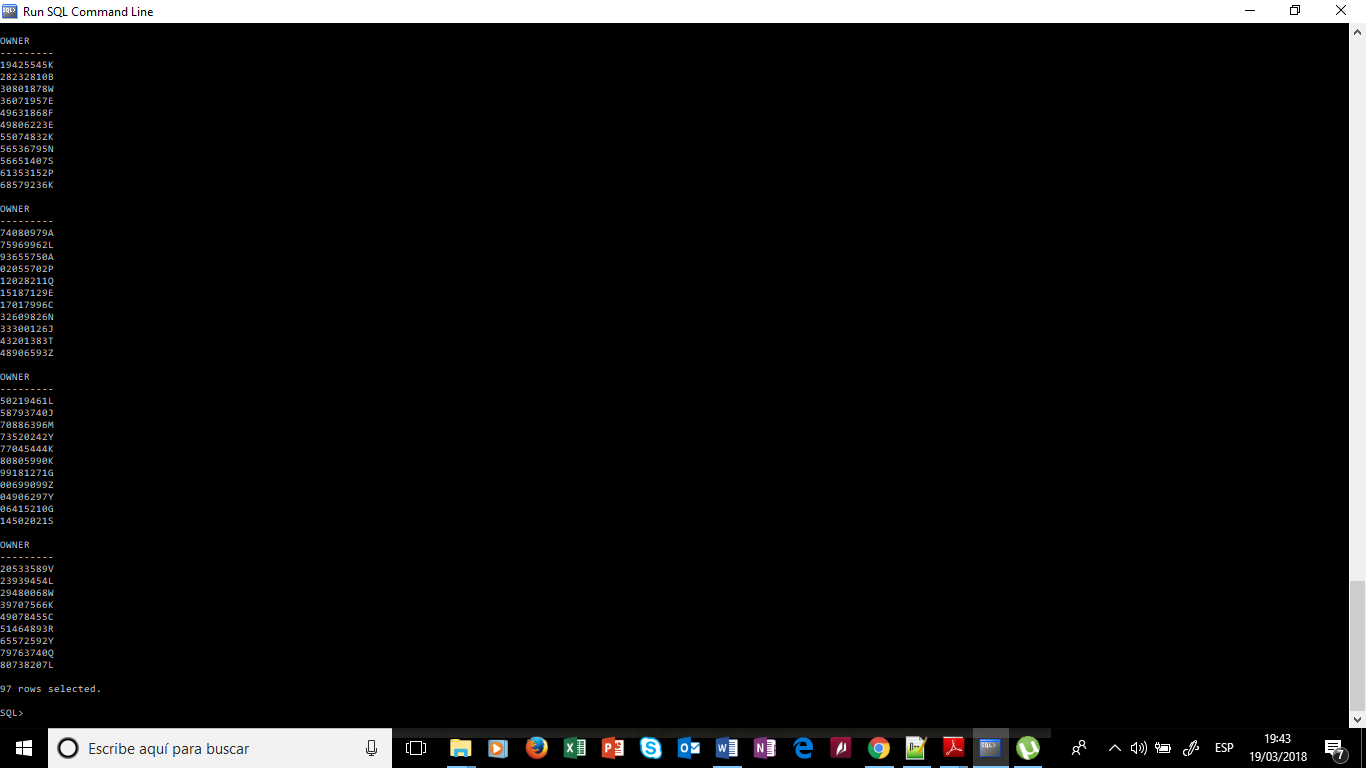
*|WHERE C.nPlate= D.nPlate))*

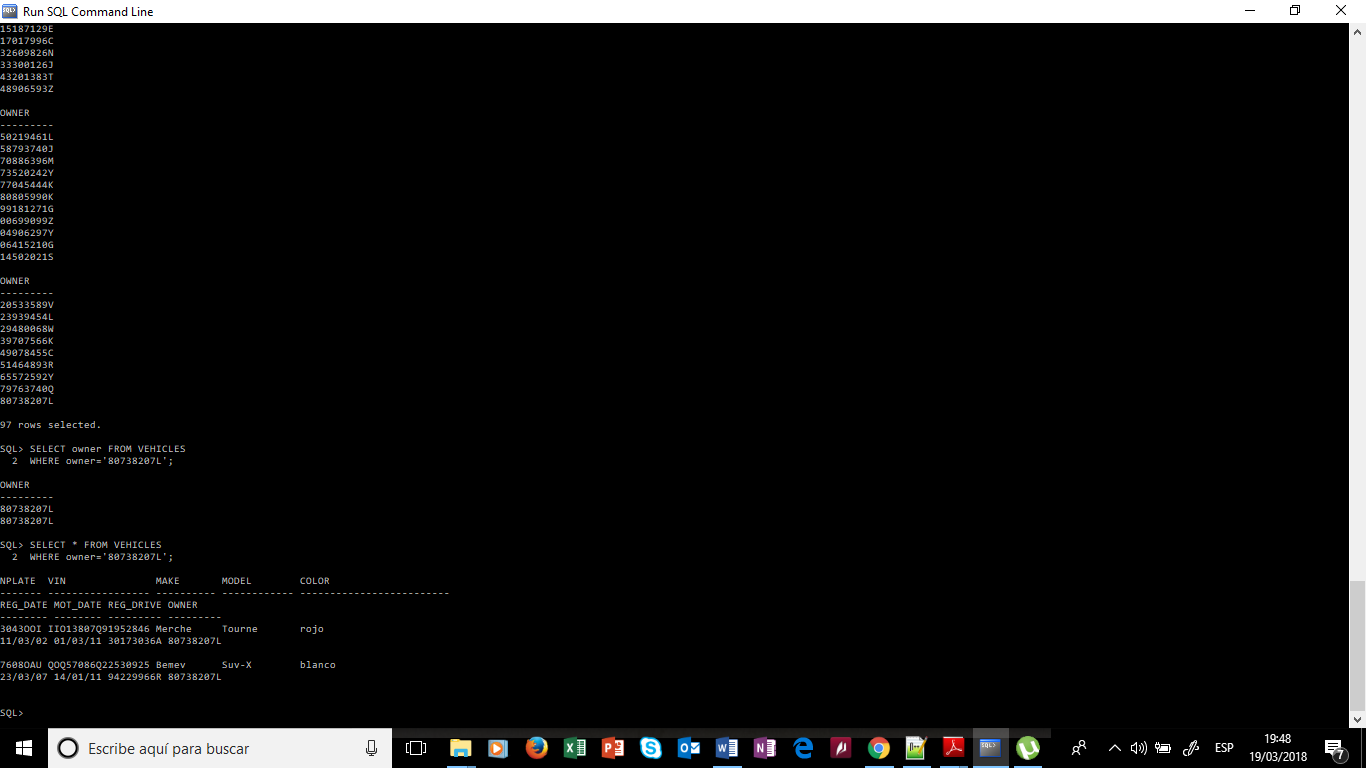
*|)GROUP BY owner;*

**Resultado**: 97 filas seleccionadas.

c) *Prueba*

Cargamos los archivos en el orden explicado en la anterior consulta.

Vamos a observar con algunos de estos dni si los dueños que han salido cumplen que no conducen ninguno de sus vehículos.

Cogiendo el dueño con dni= '80738207L', observamos que los conducen otras personas. Si hacemos esto con las 97 filas seleccionadas, comprobaremos que la consulta hace lo que tiene que hacer.

# *Jefazo: dueños de al menos tres coches que no son conductores.*

# Funciones

Para cada función, describe:

1. explicación del código implementado
2. prueba realizada para la comprobación de su correcto comportamiento

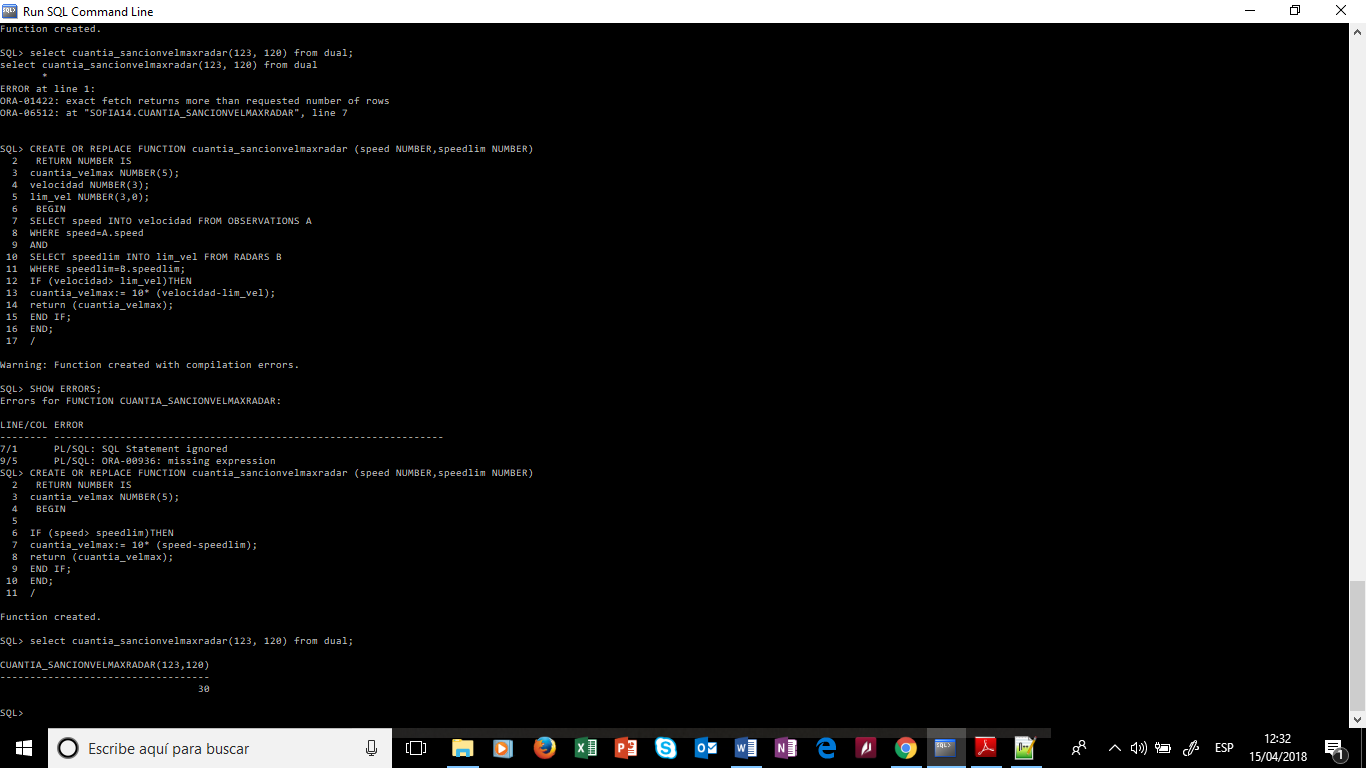
Adjunta un script (function.sql) con las sentencias SQL.

En este apartado, vamos a desarrollar funciones dentro de un paquete. Serán muy útiles para otros apartados. No vamos a seguir el orden riguroso, ya que, hay funciones que nos interesan más hacerlas antes.

1. ***Cuantía para una sanción de velocidad máxima de radar.***
2. El enunciado dice que el importe de las sanciones de velocidad es de 10€ por cada km/h que supere lo permitido. Simplemente por parámetro tenemos la velocidad a la que va el vehículo (*speed*) y la velocidad máxima del radar (*speedlim*). Declaramos una variable que almacenará el resultado(*cuantia\_velmax*). Si *speed* es mayor que *speedlim* significará que se produce una sanción de velocidad máxima de radar. La sanción es la diferencia entre los valores de los parámetros multiplicado por 10.
3. Para probar la función hemos ejecutado la siguiente línea:

*|SELECT cuantia\_sancionvelmaxradar(123, 120) FROM dual;*

Y nos ha salido lo siguiente:



Solución: 30

1. ***Observación inmediatamente anterior a otra observación (del mismo radar).***

a) Como parámetros tendremos los primary keys de OBSERVATIONS (matricula y tiempo) y los de RADARS (carretera, Punto\_km y dirección). La idea es devolver la fila con todos los atributos de la observación inmediatamente anterior, es decir, de tipo %ROWTYPE. Para ello, lo que hemos hecho es seleccionar las observaciones del mismo radar, ordenadas y agrupadas por el tiempo de manera descendente, es decir, de mayor a menor teniendo las más recientes arriba y limitándolo teniendo como primer elemento de la fila el inmediatamente anterior.

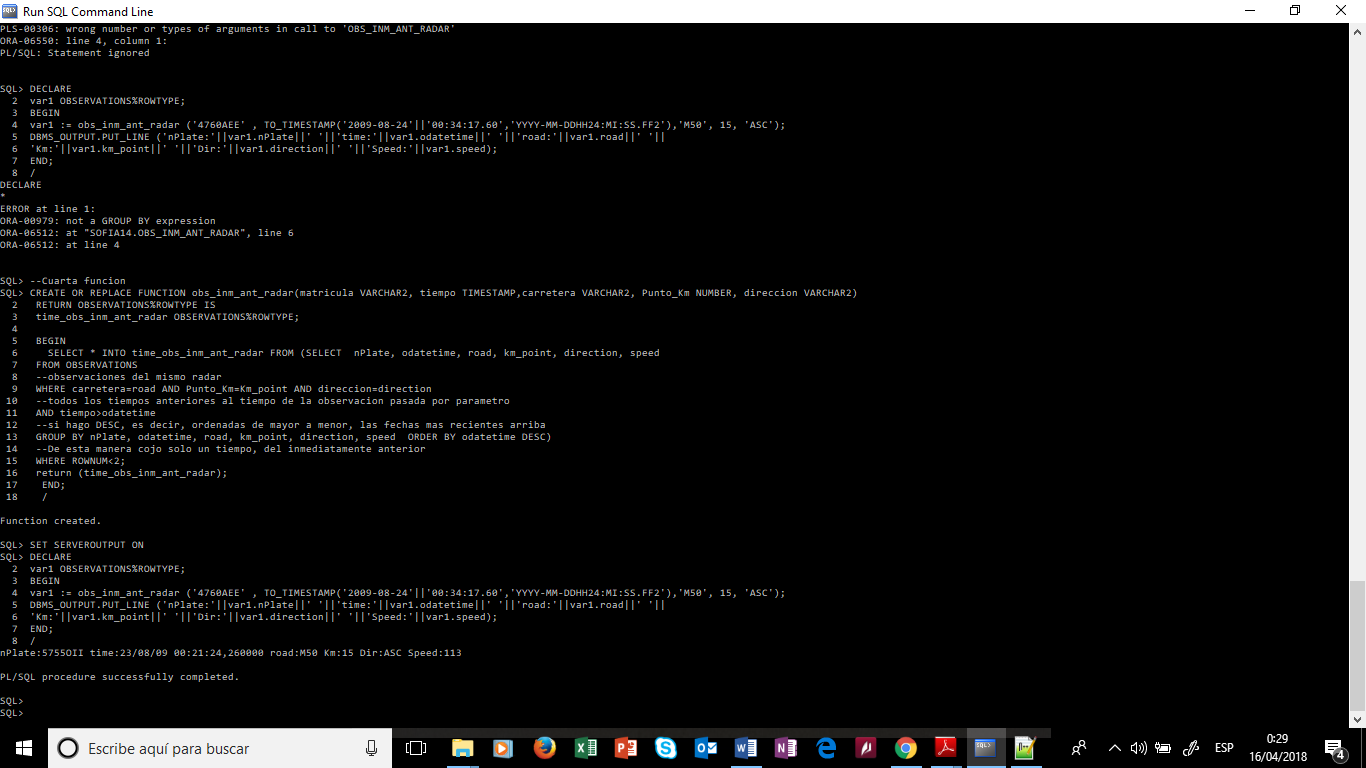
b) Para probar funciones a las que se les pasa por parametro una rowtype, hay que declarar una variable de este tipo y llamarla dentro de un procedimiento no nominado:

*|SET SERVEROUTPUT ON*

|--Hay que darle a ENTER y aparte en otra línea

…

Código en la captura:



1. ***Observación inmediatamente anterior a otra observación (del mismo vehículo).***

a) Como parámetros tendremos los primary keys de OBSERVATIONS (matricula y tiempo). La idea es devolver la fila con todos los atributos de la observación inmediatamente anterior, es decir, de tipo %ROWTYPE. Para ello, lo que hemos hecho es seleccionar las observaciones del mismo vehículo, ordenadas y agrupadas por el tiempo de manera descendente, es decir, de mayor a menor teniendo las más recientes arriba y limitándolo teniendo como primer elemento de la fila el inmediatamente anterior.

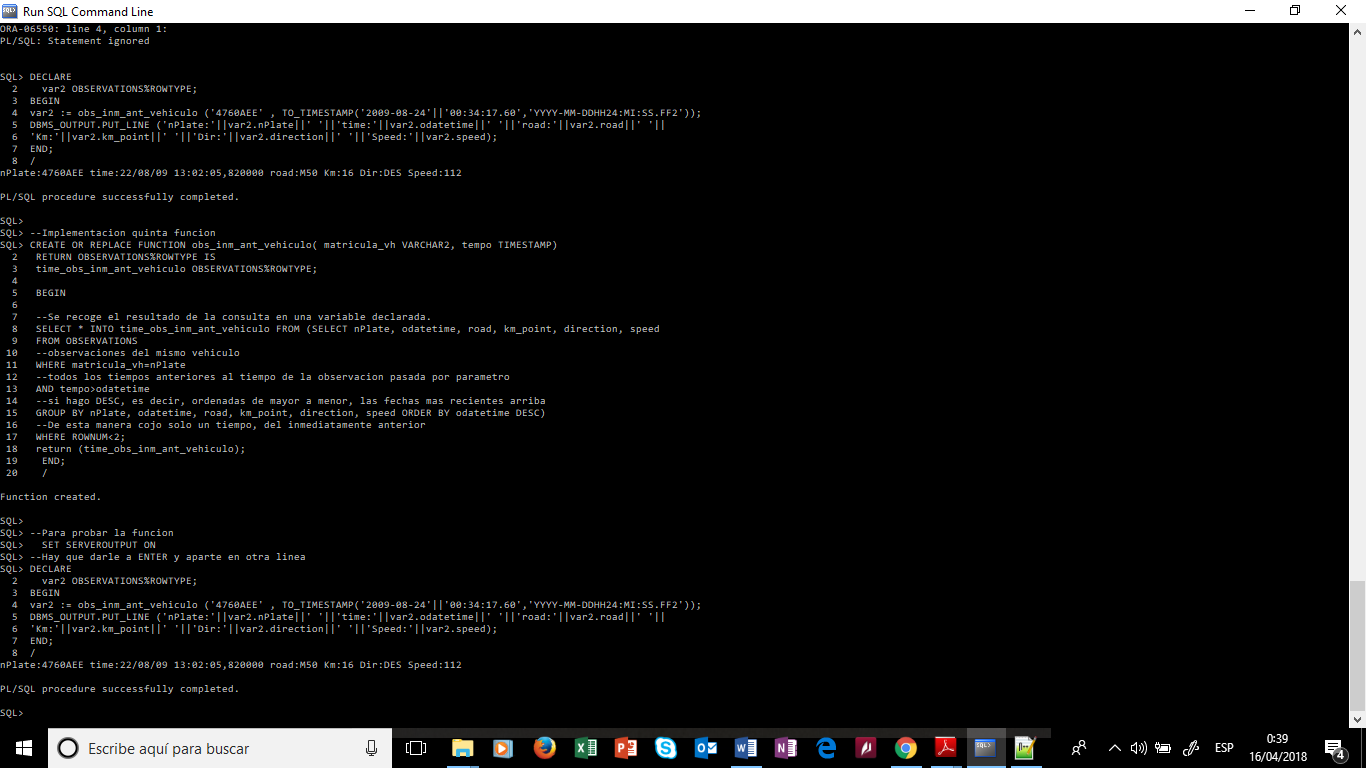
b) Para probar funciones a las que se les pasa por parametro una rowtype, hay que declarar una variable de este tipo y llamarla dentro de un procedimiento no nominado:

*|SET SERVEROUTPUT ON*

|--Hay que darle a ENTER y aparte en otra línea

…

Código en la captura:



1. ***Cuantía para una sanción de velocidad de tramo***

a)

# Vistas

Para cada vista, debes exponer:

1. su implementación en SQL
2. la funcionalidad de la vista: consulta (índica el número de filas del resultado), inserción, borrado y modificación (no es necesario completar la funcionalidad; sólo establecer qué se puede hacer). Incluye las pruebas realizadas para demostrar que la vista está bien definida, y que permite las operaciones indicadas.

Adjunta un script (view.sql) con las sentencias SQL.

# Disparadores

**Diseño**

Tabla Asociada: Alumno

Evento: UPDATE (NIA)

Temporalidad: AFTER

Granuralidad: FOR EACH ROW

Acción: modificar en matricula todos los :OLD.NIA a :NEW.NIA

Nota: new para insert y update y old para delete y update

Describe los disparadores pedidos en la práctica (al menos aquellos que son obligatorios).

Para cada uno de ellos, debes incluir una descripción que contenga:

Tabla a la que está asociado, Evento o eventos en los que se dispara, Temporalidad (antes, después o en vez de), Granularidad (por fila o sentencia), Condición (si la tiene) y Acción (el código o pseudocódigo).

En el/los disparador/es que hayas implementado, adjunta también las pruebas realizadas para comprobar su correcto funcionamiento.

Adjunta un script (trigger.sql) con las sentencias SQL.

# Conclusiones

En primer lugar, debéis defender el resultado que hayáis alcanzado, haciendo hincapié en la cobertura semántica y potencia de vuestra implementación.

Después, comentad vuestro desempeño en estas dos primeras prácticas: esfuerzo requerido, organización de vuestro equipo de trabajo, progreso en vuestros conocimientos, etc. También podéis proponer mejoras para otros años (tamaño del problema, elementos que se piden, valoración, plazos, material de apoyo, etc).