APELLIDOS _____ NOMBRE _____

Instrucciones

- El examen consta de dos partes. Esta parte dura media hora. La segunda parte durará dos horas y media.
- Escribe **en esta hoja** tu nombre y la respuesta al test.
- Puedes utilizar lápiz.

PARTE 1: Test de cuestiones teórico-prácticas

1. *(2 puntos)* **SOBRE LA SIGUIENTE CUADRÍCULA, escribe para cada una de las siguientes preguntas la ÚNICA afirmación correcta (a, b, c ó d)**. Una pregunta mal contestada NO puntúa negativo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A														
puntos según nº aciertos →			0.0	0.2	0.4	0.6	0.8	1,0	1,2	1,4	1,6	1.8	2.0	2.0

APELLIDOS _____ NOMBRE _____

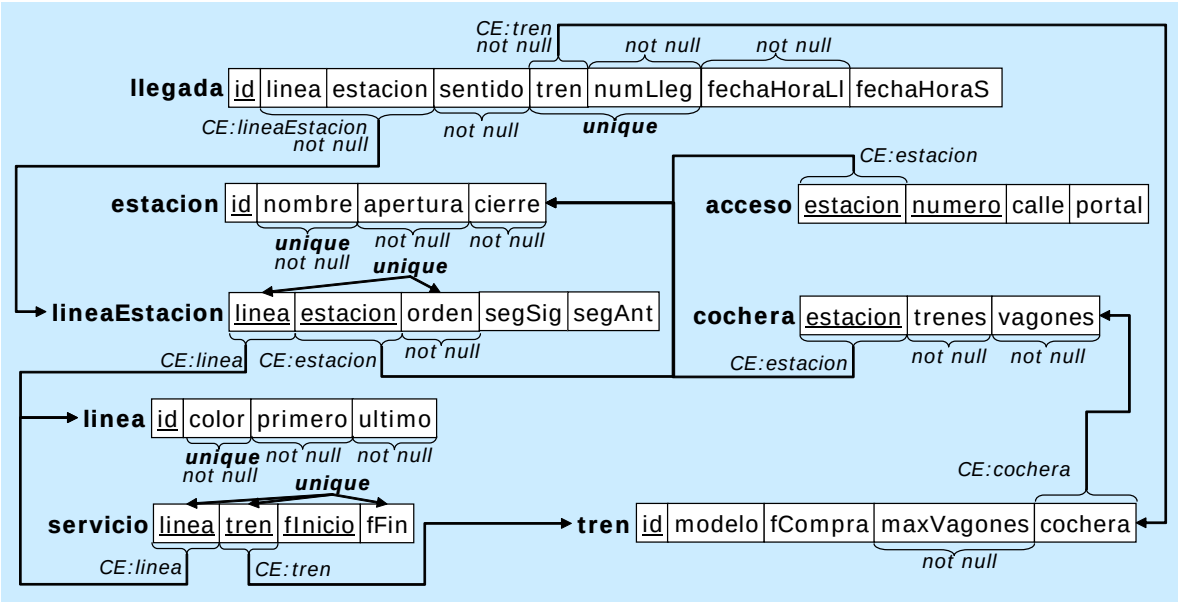
Instrucciones

- Tiempo de esta parte: 2 ½ horas. Puedes entregar los ejercicios en **lápiz**.
- Escribe tu nombre **en todas las hojas** que uses.
- Escribe las respuestas a los ejercicios **siguiendo el mismo orden** que en el enunciado.
- **No incluyas** en las consultas **ni tablas ni *outer join* innecesarios.**
- **Recuadra** cada **subconsulta**. **Subraya** y **comenta** cada correlación ***(-- correlación).***
- En **join,** pon el nombre de **cada tabla** a principio de **línea.**
- **Alinea** el contenido de **select**, **from**, **where**, etc. de forma que quede un poco más a la derecha de la palabra select, from o where, etc
- **Elimina explícitamente duplicados únicamente cuando sea necesario**

PARTE 2: Ejercicios

Se ha desarrollado un sistema informático para gestionar las ***líneas de metro de una ciudad***. La BD recoge información sobre las líneas de metro, sus estaciones y accesos desde el exterior y sus cocheras. De los trenes se guardan las líneas en las que han prestado servicio, su cochera y la información de las entradas y salidas a y desde las estaciones.

Esquema BD (en cada tabla, el conjunto de atributos subrayados forma su clave primaria)



estacion

- Contiene el identificador numérico de la estación, el nombre no nulo y único de la estación y los horarios no nulos de apertura y de cierre (cadenas de caracteres en formato 'hh:mm:ss').
- Por cada estación pasa al menos una línea de metro.

linea

- En la ciudad existen varias líneas de metro. Cada una se identifica por un identificador numérico o por un color no nulo (cadena de caracteres).
- Los atributos no nulos "primero" y "ultimo" son cadenas de caracteres (formato 'hh:mm:ss') que contienen el horario de partida de los primeros y últimos trenes de la línea. En cada uno de estos horarios parten dos trenes en la línea desde cada uno de sus extremos (y en sentido contrario).

lineaEstacion

- Cada línea pasa por un conjunto de estaciones. En las estaciones que pasan varias líneas, los pasajeros pueden cambiar de línea.
- Cada fila de esta tabla corresponde a una estación de una línea. Por lo tanto, en las estaciones donde pasan varias líneas, el identificador de la estación aparecerá en esta tabla tantas veces como líneas de metro pasan por ella.
- "Orden" es un número no nulo que indica el número de orden de la estación en la línea (recorrida en sentido 'ascendente' según la columna 'sentido' de la tabla llegada). No puede repetirse el mismo número de orden en una línea (la pareja orden-linea es unique).
- "Estacion" y "linea" son dos claves extranjeras a la tabla estación y línea respectivamente.
- "SegAnt" y "segSig" son dos enteros positivos que indican el número de segundos que debería tardar el tren en ir a la siguiente estación de la línea en sentido descendente (segAnt) y ascendente (segSig) respecto al valor del atributo "orden" (si no hay siguiente o anterior vale nulo).

acceso

- Cada acceso desde el exterior (la calle) a una estación de metro cuenta con una fila en esta tabla. "Estacion" es clave extranjera que referencia a la tabla estación y "numero" es un entero positivo que sirve para distinguir los diferentes accesos a una misma estación. Toda estación cuenta al menos con un acceso.
- Los atributos opcionales "calle" y "portal", indican la dirección de la ciudad donde se encuentra situada la puerta de acceso. Pueden ser nulos.

tren

- Cada fila recoge información de una *máquina de tren*. "Id" es un identificador numérico.
- El atributo opcional "modelo" es una cadena de caracteres que contiene el modelo de la máquina de tren. El atributo opcional "fCompra" contiene la fecha de adquisición de la máquina (tipo date).
- "MaxVagones" contiene el número máximo de vagones que puede arrastrar y es obligatorio.
- "Cochera" es un atributo opcional. Si contiene el **valor nulo** significa que el tren **está en ese momento de servicio**. En caso contrario, contiene el identificador de la cochera donde se encuentra estacionado el tren (clave extranjera a cochera).

cochera

- Una cochera está situada en una estación y cada estación sólo puede tener una cochera. "Estacion" es clave extranjera que referencia a la tabla estación.
- Los atributos obligatorios "trenes" y "vagones" son sendos números que recogen el número máximo de máquinas de tren y de vagones que pueden guardarse en la cochera.

servicio

- Periodos continuados de servicio de un determinado tren a una de las líneas. Un mismo tren **puede prestar servicio a varias líneas** a la vez.
- "Linea" y "tren" son claves extranjeras que hacen referencia, una a la tabla línea, y la otra a tren.
- "fInicio" y "fFin" son las fechas (date) de inicio y fin del periodo de servicio del tren en la línea. **El tren presta sus servicios *actualmente* en la línea cuando "fFin" vale null.**
- "Línea", "tren" y "fFin" forman una clave (unique). Eso supone que no puede haber dos servicios actuales (fFin is null) del mismo tren en la misma línea.

llegada

- Cada fila contiene datos de la entrada de un tren a un andén. Un andén pertenece a una estación y a una línea y pertenece a un sentido del recorrido. Se mantienen datos de los últimos meses.
- "Id" es un número que distingue una fila de la tabla llegada de las demás.
- "Linea" y "estacion" conforman una clave extranjera que referencia a la tabla lineaEstacion.
- "Sentido" es el sentido de recorrido del tren. Es una cadena de caracteres no nula que contiene bien el valor 'ascendente' o bien el valor 'descendente' según el orden en el que se visiten las estaciones de la línea (en relación al atributo "orden" de lineaEstación).
- "Tren" es una clave extranjera a la tabla tren.

(continúa en la siguiente cara)

- "Tren" y "numLleg" forman una clave. Ninguno de estos atributos valdrá nulo. "NumLleg" es un número positivo y correlativo que sirve para distinguir las diferentes paradas de un determinado tren en algún andén. La primera parada que hizo el tren 100 en un andén tendrá "numLleg"=1, la segunda "numLleg"=2, etc.
- "FechaHoraLl" contiene la fecha y hora (date) no nulas de entrada del tren a la estación.
- "FechaHoraS" contiene la fecha y hora de salida del tren tras recoger y dejar pasajeros. Si vale nulo, significa que el tren todavía está detenido en el andén.
- Si "fechaHoraS" tiene un valor no nulo, éste será mayor que "fechaHoraLl". Ambos valores tienen bien la misma fecha, o bien "fechaHoraLl" es un día anterior a "fechaHoraS".
- Dos llegadas consecutivas al mismo andén (igual línea, estación y sentido) cumplen que la "fechaHoraS" de la primera es menor que "fechaHoraLl" de la segunda.

2. *(0,5 puntos)* Se va a incorporar a la BD anterior información sobre los conductores de trenes y el trabajo que realizan. Crea mediante una sola **instrucción SQL** la tabla de nombre "**conduce**" que contiene el identificador numérico del conductor (**conductor**), el del tren conducido (**tren**), el de la línea por la que circulará el tren (**linea**), la fecha y hora en la que el conductor empieza a trabajar conduciendo el tren (**fechaHoraInicio**) y la fecha y hora en la que termina (**fechaHoraFin**). Se va a asignar un identificador numérico a esta tabla como clave primaria (**id**).
   Hay que garantizar
   - Que no se repita la terna: *(conductor, tren, fechaHoraInicio)*.
   - Que el identificador de tren sea uno de los existentes en la tabla tren y que el de línea sea uno de los existentes en la tabla línea.
   - Que fechaHoraInicio sea menor que fechaHoraFin.
   - Los servicios de los conductores se van a insertar en la tabla a su finalización y por lo tanto ninguno de los valores de las filas de esta tabla debería ser nulo.

3. *(0,5 puntos)* Se acaba de insertar en la BD la **estación** (777, 'Paz', null, null). Se trata de una estación intermedia en la línea de color 'azul' que ocupará el número de orden 7 (las estaciones posteriores deberán incrementar su orden en la línea). Los segundos estimados a la estación siguiente son 120 (segSig) y a la anterior 90 (segAnt). También se tardan 90 segundos en venir desde la estación anterior y 120 en ir a la estación siguiente. Escribe las operaciones necesarias, **sobre la tabla lineaEstacion**, para incorporar estos cambios disponiéndolas en un orden adecuado.

4. *(1 punto)* Se desea cambiar la clave primaria de la tabla "linea" (id) por la columna color (de tipo varchar(15)) y viceversa (definir id como clave candidata). ¿Qué **instrucciones sql** son necesarias sobre las **tablas "linea" y "servicio"** (escríbelas en un orden adecuado) manteniendo la relación entre ambas? Ten en cuenta que la restricción de clave primaria de "linea" se llama pk_linea, la de "servicio" pk_servicio y la de unicidad de "linea" u_color. Además, las claves extranjeras de la tabla "servicio", que hacen referencia a las tablas "linea" y "tren" respectivamente, se llaman fk_linea y fk_tren.

5. *(0,5 puntos)* Escribe una **consulta sql** que obtenga, sin repeticiones de filas, 3 columnas: (1) el identificador de tren, (2) el identificador de línea y (3) el año, de aquellos trenes para los que figura alguna llegada en esa línea y durante ese año ***en la tabla llegada*** (tomado de fechaHoraLl), pero que **no** estuvieran en servicio en el momento de la llegada (fechaHoraLl) para esa línea según ***la tabla servicio***.

   *Nota: recuerda que **fFin** de la tabla servicio **puede valer nulo** y significa que el tren está actualmente en servicio en la línea.*

6. *(0,5 puntos)* Crea una **vista** llamada **trenesCochera** que devuelva el identificador de cada cochera que tenga algún tren estacionado en ella, junto al número de trenes estacionados en este momento en la misma y el número máximo de trenes que entran en la cochera.

7. *(0,5 puntos)* Crea una **tabla** llamada **infoCochera** que tenga 3 columnas: (1) la columna 'cochera' contendrá el identificador de una cochera, (2) la columna numTrenes contendrá el número de trenes estacionados en la cochera (cero si no hay ninguno), y (3) la columna maxTrenes contendrá el número máximo de trenes que entran en la cochera. En la tabla tiene que haber una fila para cada cochera existente en la BD. Nos interesa únicamente una solución que haga uso de la vista **trenesCochera** y una sola instrucción SQL.

8. *(1 punto)* Escribe una **consulta sql** que obtenga 3 columnas: (1) identificador de estación, (2) número de accesos a la estación y (3) número de líneas que pasan por la estación. Sólo deben figurar en el resultado estaciones con **3 o más** trenes **detenidos** en sus andenes. Cuando un tren está detenido en un andén (línea-estación-sentido) existe una fila del tren en la tabla llegada donde fechaHoraS contiene el valor nulo.

9. *(1 punto) (1 punto)* Escribe una **consulta sql** que obtenga dos columnas (1) identificador de línea *(de todas para las existan llegadas en la BD)* y (2) retraso medio en segundos de los trenes de esa línea*.* El retraso se calcula restando los segundos estimados de viaje (segAnt o segSig), a los transcurridos entre la salida de un tren de un andén y la entrada del mismo tren al siguiente andén.

   En el cómputo del retraso **hay que descartar** los segundos que pasan entre la última llegada de una jornada y la primera de la siguiente. Para simplificar, entenderemos que toda estación cierra antes de las 0:00 y abre a partir de las 06:00. Por tanto, la **hora** (hora-minutos-segundos) de la última llegada de la jornada a un andén será **mayor** que la hora de la primera salida de la siguiente jornada (a diferencia del resto la jornada).

   *Dispones de la función **segundos_entre(**date1**,** date2**)**, que calcula cuántos segundos hay entre date1 y date2. Si date1≤date2 ese número es positivo o cero y si no negativo.*

10. *(1 punto)* Escribe una **consulta sql** que obtenga dos columnas: (1) el identificador de una estación de la línea 'verde' y (2) el *tiempo medio de detención* de los trenes (para recoger y dejar pasajeros) en los andenes de esa estación y línea. Dichos andenes deben cumplir que su *tiempo medio de detención* (columna 2 del resultado) sea **inferior o igual** al *tiempo medio de detención* en los andenes de **una línea diferente** a la 'verde' de la **misma estación**. *Puedes utilizar la función **segundos_entre** del ejercicio anterior.*

11. *(1 punto)* Escribe una **consulta sql** que obtenga el identificador de los trenes que hayan visitado **<u>todas</u>** las estaciones pertenecientes a líneas que ***pasan*** por la estación de nombre 'Esperanza' en los últimos 10 días (considerar la llegada como fecha de visita).

    ***No confundir** lo anterior con "visitar todas las estaciones de nombre Esperanza", ya que sólo hay una estación 'Esperanza' (el nombre de estación es clave).*

12. *(0,5 puntos)* Escribe una **consulta en álgebra relacional** que muestre el identificador de los trenes para los que figure en la BD que han prestado servicio en la línea de color 'verde', bien porque figura dicha información en la **tabla servicio**, o bien en la **tabla llegada**.

ALGUNAS FUNCIONES DE ORACLE (puede que no hagan falta todas)

Nombre(argumentos)	Significado	Ejemplos
sysdate	Devuelve la fecha actual	**sysdate** → '09-sept-2009'
to_char(<fecha>, <formato>)	Convierte la fecha a cadena según el formato especificado	**to_char(sysdate,'YYYY')** → '2009' **to_char(sysdate,'YYYY/MM/DD')** → '2009/09/01'
to_date(<cadena>,<formato>)	Convierte la cadena dada según el formato especificado a fecha	**to_date('01-09-09','DD-MM-YY')** → [fecha de hoy]
substr(<cadena>,<inic>,<long>)	Extrae una subcadena de <cadena> de longitud <long> empezando en el carácter <inic>.	**substr('mi cadena', 5, 3)** → 'ade'

2. *(0,5 puntos)* Se va a incorporar a la BD anterior información sobre los conductores de trenes y el trabajo que realizan. Crea mediante una sola **instrucción SQL** la tabla de nombre "**conduce**" que contiene el identificador numérico del conductor (**conductor**), el del tren conducido (**tren**), el de la línea por la que circulará el tren (**linea**), la fecha y hora en la que el conductor empieza a trabajar conduciendo el tren (**fechaHoraInicio**) y la fecha y hora en la que termina (**fechaHoraFin**). Se va a asignar un identificador numérico a esta tabla como clave primaria (**id**).

   Hay que garantizar
   - Que no haya varias filas con el mismo valor para la terna: *(conductor, tren, fechaHoraInicio)*.
   - Que el identificador de tren sea uno de los existentes en la tabla tren y que el de línea sea uno de los existentes en la tabla línea.
   - Que la fechaHoraInicio sea menor que fechaHoraFin.
   - Los servicios de los conductores se van a insertar en la tabla a su finalización y por lo tanto ninguno de los valores de las filas de esta tabla debería ser nulo.

```
CREATE TABLE conduce (
id               int not null,
conductor        int not null,
tren             int not null,
linea            int not null,
fechaHoraInicio  date not null,
fechaHoraFin     date not null,

constraint pk_conduce
            primary key (id),
constraint u_conduce
            unique(conductor, tren, fechaHoraInicio),
constraint fk_tren
            foreign key (tren)
            references tren(id),
constraint fk_linea
            foreign key (linea)
            references linea(id),
constraint inicio_fin
                 check(fechaHoraInicio < fechaHoraFin)
);
```

3. *(0,5 puntos)* Se acaba de insertar en la BD la **estación** (777, "Paz", null, null). Se trata de una estación intermedia en la línea de color "azul" que ocupará el número de orden 7 (las estaciones posteriores deberán incrementar su orden en la línea). Los segundos estimados a la estación siguiente son 120 (segSig) y a la anterior 90 (segAnt). También se tardan 90 segundos en venir desde la estación anterior y 120 en venir desde la estación siguiente. Escribe las operaciones necesarias, **sobre la tabla lineaEstacion**, para incorporar estos cambios disponiéndolas en un orden adecuado.

```
update lineaEstacion
  set orden=orden + 1
  where linea=(select id from linea where color="azul") and orden>=7;
insert into lineaEstacion
  values ((select id from linea where color="azul"), 777, 7,120, 90);
update lineaEstacion
  set segSig=90
  where linea=(select id from linea where color="azul") and orden=6;
update lineaEstacion
  set segAnt=120
  where linea=(select id from linea where color="azul") and orden=8;
```

4. *(1 punto)* Se desea cambiar la clave primaria de la tabla línea (id) por la columna color (de tipo varchar(15)) y viceversa (definir id como clave candidata). ¿Qué **instrucciones sql** son necesarias sobre las **tablas línea y servicio** (escríbelas en un orden adecuado) manteniendo la relación entre ambas? Ten en cuenta que la restricción de clave primaria de línea se llama pk_linea, la de servicio pk_servicio y la de unicidad de línea u_color. Además, las claves extranjeras de la tabla servicio, que hacen referencia a la tabla línea y a la tabla tren respectivamente, se llaman fk_linea y fk_tren.

alter table *servicio* ***add*** *color varchar(15);*
update *servicio S* ***set*** *color=(select L.color from linea L where L.id=S.linea --correlación);*
alter table *servicio* ***drop constraint*** *fk_linea;*
alter table *servicio* ***drop column*** *linea;*
alter table *servicio* ***rename column*** *color* ***to*** *linea;*
alter table *linea* ***drop constraint*** *pk_linea;*
alter table *linea* ***drop constraint*** *u_color;*
alter table *linea* ***add constraint*** *pk_linea* ***primary key*** *(color);*
alter table *linea* ***add constraint*** *u_id* ***unique*** *(id);*
alter table *servicio* ***add constraint*** *fk_linea* ***foreign key*** *(linea)* ***references to*** *linea(color);*

5. *(0,5 puntos)* Escribe una **consulta sql** que obtenga, sin repeticiones de filas, 3 columnas: (1) el identificador de tren, (2) el identificador de línea y (3) el año, de aquellos trenes para los que figura alguna llegada en esa línea y durante ese año ***en la tabla llegada*** (tomado de fechaHoraLl), pero que **no** estuvieran en servicio en el momento de la llegada (fechaHoraLl) para esa línea según ***la tabla servicio***.

Nota: recuerda que ***fFin*** *de la tabla servicio* ***puede valer nulo*** *y significa que el tren está actualmente en servicio en la línea.*

```
select  distinct LL.tren, LL.linea, to_char(LL.fechaHoraLl, 'yyyy')
from  llegada LL left join
        servicio S on LL.tren=S.tren and LL.linea=S.linea and S.fInicio<=LL.fechaHoraLl
                    and (LL.fechaHoraLl<=S.fFin or S.fFin is null)
where S.linea is null -- el tren no estaba en servicio de la línea
```

Otra forma*:*

```
select  LL.tren, LL.linea, to_char(LL.fechaHoraLl, 'yyyy')
from  llegada LL
except
select  LL.tren, LL.linea, to_char(LL.fechaHoraLl, 'yyyy')
from  llegada LL join
        servicio S on LL.tren=S.tren and LL.linea=S.linea
where S.fInicio<=LL.fechaHoraLl and (LL.fechaHoraLl<=S.fFin or S.fFin is null)
```

6. *(0,5 puntos)* Crea una **vista** llamada **trenesCochera** que devuelva el identificador de cada cochera que tenga algún tren estacionado en ella, junto al número de trenes estacionados en este momento en la misma y al número máximo de trenes que entran en la cochera.

```
create view trenesCochera as
   select  C.estacion as cochera, count(T.id) as numTrenes, C.trenes as maximoTrenes
   from  tren T join
           cochera C on T.cochera=C.estacion
   group by C.estacion, C.trenes
```

7. *(0,5 puntos)* Crea una **tabla** llamada **infoCochera** que tenga 3 columnas: (1) la columna 'cochera' contendrá el identificador de una cochera, (2) la columna numTrenes contendrá el número de trenes estacionados en la cochera (cero si no hay ninguno), y (3) la columna maxTrenes contendrá el número máximo de trenes que entran en la cochera. En la tabla tiene que haber una fila para cada cochera existente en la BD. Nos interesa únicamente una solución que haga uso de la vista **trenesCochera** y una sola instrucción SQL.

```
create table infoCochera as
   select  C.estacion as cochera, coalesce(TC.numTrenes, 0) as numTrenes,
           C.trenes as maxTrenes
   from  cochera C left join trenesCochera TC on C.estacion=TC.cochera
```

8. *(1 punto)* Escribe una **consulta sql** que obtenga 3 columnas: (1) identificador de estación, (2) número de accesos a la estación y (3) número de líneas que pasan por la estación. Sólo deben figurar en el resultado estaciones con **3 o más** trenes **detenidos** en sus andenes. Cuando un tren está detenido en un andén (línea-estación-sentido) existe una fila del tren en la tabla llegada donde fechaHoraS contiene el valor nulo.

```
select      LE.estacion, count(distinct A.numero) as "andenes",
            count(distinct LE.linea) as "lineas"
from lineaEstacion LE join
       acceso A on A.estacion=LE.estacion
group by LE.estacion
having 3=<
        ( select count(LL.id)
          from llegada LL
          where LL.fechaHoraS is null and LL.estacion=LE.estacion -- correlacion )
```

***Otra forma**:*

```
select      LE.estacion, count(distinct A.numero) as "andenes",
            count(distinct LE.linea) as "lineas"
from  ( lineaEstacion LE join
         acceso A on A.estacion=LE.estacion ) left join
        llegada LL on LL.linea=LE.linea and LL.estacion=LE.estacion
                      and LL.fechaHoraS is null
group by LE.estacion
having count(distinct LL.id) >= 3
```

9. *(1 punto)* Escribe una **consulta sql** que obtenga dos columnas (1) identificador de línea (de todas para las existan llegadas en la BD) y (2) retraso medio en segundos de los trenes de esa línea. El retraso se calcula restando los segundos estimados de viaje (segAnt o segSig), a los transcurridos entre la salida de un tren de un andén y la entrada del mismo tren al siguiente andén.

   En el cómputo del retraso **hay que descartar** los segundos que pasan entre la última llegada de una jornada y la primera de la siguiente. Para simplificar, entenderemos que toda estación cierra antes de las 0:00 y abre a partir de las 06:00. Por tanto, la **hora** (hora-minutos-segundos) de la última llegada de la jornada a un andén será **mayor** que la hora de la primera salida de la siguiente jornada (a diferencia del resto la jornada).

   *Dispones de la función **segundos_entre(**date1**,** date2**)**, que calcula cuántos segundos hay entre date1 y date2. Si date1≤date2 ese número es positivo o cero y si no negativo.*

```
select  L1.linea as "línea",
        avg( case L1.sentido
               when 'ascendente' then
                  segundos_entre(L1.fechaHoraS, L2.fechaHoraLl) - LE.segSig
               else
                  segundos_entre(L2.fechaHoraS, L1.fechaHoraLl) - LE.segAnt
               end
        ) as "retraso"
from  llegada L1 join
      llegada L2 on L1.tren=L2.tren and L1.numLleg+1=L2.numLleg join
      lineaEstacion LE on L1.linea=LE.linea and L1.estacion=LE.estacion
where  -- descartar el periodo entre dos jornadas:
       to_char(L1.fechaHoraS, 'hh:mm:ss') < to_char(L2.fechaHoraLl, 'hh:mm:ss')
group by L1.linea
```

10. *(1 punto)* Escribe una **consulta sql** que obtenga dos columnas: (1) el identificador de una estación de la línea 'verde' y (2) el *tiempo medio de detención* de los trenes (para recoger y dejar pasajeros) en los andenes de esa estación y línea. Dichos andenes deben cumplir que su *tiempo medio de detención* (columna 2 del resultado) sea **inferior o igual** al *tiempo medio de detención* en los andenes de **una línea diferente** a la 'verde' de la **misma estación**. *Puedes utilizar la función **segundos_entre** del ejercicio anterior.*

```
select  LL.estacion, avg(segundos_entre(LL.fechaHoraLl, LL.fechaHoraS))
from  linea L join
      llegada LL on L.linea=LL.linea
where L.color='verde'
group by LL.estacion, LL.linea -- solo es una línea (verde): está aquí para la correlación
having avg(segundos_entre(LL.fechaHoraLl, LL.fechaHoraS)) <=any
    ( select avg(segundos_entre(LL2.fechaHoraLl, LL2.fechaHoraS))
      from llegada LL2
      where LL2.estacion = LL.estacion -- correlacion
            and LL2.linea <> LL.linea -- correlación
      group by LL2.linea
    )
```

11. *(1 punto)* Escribe una **consulta sql** que obtenga el identificador de los trenes que hayan visitado **todas** las estaciones pertenecientes a líneas que ***pasan*** por la estación de nombre 'Esperanza' en los últimos 10 días (considerar sólo la hora de llegada como fecha de visita).

    No confundir *lo anterior con "visitar todas las estaciones de nombre Esperanza", ya que sólo hay una estación 'Esperanza' (el nombre de estación es clave).*

```
select  T.id
from  tren T
where not exists
    ( select LE.estacion
      from   estacion E join
             lineaEstacion LEsp on LEsp.estacion=E.id join
             lineaEstacion LE  on LEsp.linea=LE.linea
      where E.nombre='Esperanza'
      except
      select L.estacion
      from llegada L
      where L.fechaHoraLl>=sysdate-10 and L.tren=T.id -- correlacion
    )
```

12. *(0,5 puntos)* Escribe una **consulta en álgebra relacional** que muestre el identificador de los trenes para los que figure en la BD que han prestado servicio en la línea verde bien porque figura dicha información en la **tabla servicio** o bien en la **tabla llegada**.

    En sql se escribiría, por ejemplo, así:

```
select  S.tren
from  servicio S join
        linea L on S.linea=L.id
where L.color ='verde'
union
select LL.tren
from llegada LL join
       linea L on LL.linea=L.id
where L.color='verde'
```

    Nos piden su escritura en álgebra relacional:

LineaVerde ← $\sigma_{color='verde'}$(linea)
ServiciosVerde ← π_{tren} (servicio $|X|_{linea=id}$ LineaVerde)
LlegadasVerde ← π_{tren} (llegada $|X|_{linea=id}$ LineaVerde)
Resultado ← ServiciosVerde $\cup$ LlegadasVerde