2020-Final_19-02-2020

PRACTICA

3.a)

```
table Numeros(
      clave int primary key
      valor int
)
```

Sean las consultas

```
select count(valor) from Numeros
select count(clave) from Numeros
```

El resultado de las consultas son N y M respectivamente, tal que

0<N<M<5000

Entre que rangos se encuentra el resultado de la siguiente consulta:

```
select count(distinct valor) from Numeros
```

Entre 0 y N, o igual a N.

3.b)Realizar una consulta que de como maximo 10 resultados, mostrando las 5 claves positivas mayores, y las 5 claves positivas menores

```
Select TOP 5 IdProd from productos
                Where IdProd > 0
                Order by IdProd desc
UNION ALL
Select TOP 5 IdProd from productos
                Where IdProd > 0
                Order by IdProd asc
```

2019-Final_26_de_Febrero

a) Se realiza la siguiente consulta de datos:

```
SELECT t1.campoA, COUNT(*)
FROM tabla1 t1, tabla2
GROUP BY t1.campoA
```

Sabiendo que la tabla1 tiene 500 registros y la tabla2 no posee datos indicar la respuesta correcta:

i) La consulta da error al ejecutarse

ii) Devulve 500 filas con t1.campoA y COUNT(*)=1

iii) Devuelve 1 fila con COUNT(*)=0

iv) Devuelve un conjunto vacio de datos

Respuesta: Devuelve un conjunto vacío de datos. Al determinar la relación como "tabla1, tabla2" es el producto cartesiano de las dos tablas. Al no

poseer datos (tampoco datos nulos) en la segunda tabla, el producto cartesiano es un conjunto vacío de datos.

b) Se dispone de la siguiente información:

```
CREATE TABLE [dbo].[Persona](
      [PersonID] [int] NOT NULL,
      [Apellido] [varchar](255) NOT NULL,
      [Nombre] [varchar](255) NOT NULL,
      [Direccion] [varchar](255) NULL,
      [Ciudad] [varchar](255) NULL,
```

Sabiendo que la tabla no posee ningun constrainte, implementar el/los objetos necesarios para emular que el campo ciudad sea una FK que referencie a la tabla ciudades (asuma que dicha tabla existe y su campo PK se llama idCiudad)

3.b)

```
CREATE TRIGGER tg_fkCiudades ON Personas INSTEAD OF INSERT, UPDATE
AS
BEGIN
    DECLARE @PersonID int, @Apellido varchar(255), @Nombre varchar(255), @Direccion varchar(255), @Ciudad varchar(255)
    DECLARE cursor_fk CURSOR FOR
    SELECT PersonID, Apellido, Nombre, Direccion, Ciudad FROM inserted

    OPEN cursor_fk
        FETCH NEXT FROM inserted INTO @PersonID, @Apellido, @Nombre, @Direccion, @Ciudad

        WHILE(@@FETCH_STATUS = 0)
        BEGIN
            IF(@Ciudad IS NOT NULL AND @Ciudad IN (SELECT idCiudad FROM Ciudades))
            BEGIN
                IF(NOT EXISTS (SELECT * FROM deleted) )
                    INSERT INTO Personas (PersonID, Apellido, Nombre, Direccion, Ciudad)
                                VALUES(@PersonID, @Apellido, @Nombre, @Direccion, @Ciudad)
                ELSE
                    UPDATE Personas SET Apellido = @Apellido, Nombre = @Nombre, Direccion = @Direccion, Ciudad = @Ciudad
                                WHERE PersonID = @PersonID
            END
            ELSE
            BEGIN
                IF(NOT EXISTS (SELECT * FROM deleted) )
                    INSERT INTO Personas (PersonID, Apellido, Nombre, Direccion, Ciudad)
                                VALUES(@PersonID, @Apellido, @Nombre, @Direccion, @Ciudad)
                ELSE
                    UPDATE Personas SET Apellido = @Apellido, Nombre = @Nombre, Direccion = @Direccion, Ciudad = @Ciudad
                                WHERE PersonID = @PersonID
            END

            FETCH NEXT FROM inserted INTO @PersonID, @Apellido, @Nombre, @Direccion, @Ciudad
        END

    CLOSE cursor_fk
    DEALLOCATE cursor_fk

END
```

OTRA MANERA SIN CURSORES

```
CREATE TRIGGER tg_fcCiudades ON Persona INSTEAD OF INSERT, UPDATE
AS
```

```
BEGIN
	IF EXISTS (SELECT * FROM deleted)
	BEGIN --ES UN UPDATE
		UPDATE A
		SET
			A.Apellido = B.Apellido,
			A.Nombre = B.Nombre,
			A.Direccion = B.Direccion,
			A.Ciudad = B.Ciudad
		FROM Persona A
		INNER JOIN inserted B
			ON A.PersonaID = B.PersonaID
		INNER JOIN Ciudades C
			ON A.Ciudad = C.Ciudad
	END --END UPDATE
	ELSE
	BEGIN --ES UN INSERT
		INSERT INTO Persona
		SELECT PersonaId, Apellido, Nombre, Direccion, Ciudad
		FROM inserted A
		INNER JOIN Ciudades B
			ON A.Ciudad = B.IdCiudad
		UNION
		SELECT PersonaId, Apellido, Nombre, Direccion, Ciudad
		FROM inserted A
		WHERE A.Ciudad IN NULL
	END --END INSERT
END --END TRIGGER
```

2019-Final_25_de_Septiembre

Dada las siguientes tablas

CARRERAS:
IdCarrera int (pk)
Fecha date not null
NumeroCarrera int not null

CABALLOS:
IdCarrera int (fk)
NumeroCaballo int not null
NombreCaballo varchar(20) not null
TotalApostado int not null
Puesto int

Se sabe que por cada carrera un solo caballo puede tener el campo puesto con el valor 1 e indica que es el ganador, si la carrera aun no se corrio

ese campo se mantiene en nulo, los registros de ambas tablas nunca pueden borrarse y al insertar nuevo registros el campo Puesto siempre esta en nulo

a) Realizar una consulta ANSI-SQL que para la fecha del dia devuelva el/los nombre/s del caballo ganador que menos dinero han recibido en apuestas hasta el momento

```
select TOP 1 cb.NumeroCaballo, cb.NombreCaballo, sum(cb.totalApostado) as
'Total Apostado' from CABALLOS cb
      join CARRERAS cr on cb.IdCarrera = cr.IdCarrera
      where cr.Fecha = '2020-02-24' and cb.Puesto = 1
      group by cb.NumeroCaballo, cb.NombreCaballo
      order by sum(cb.TotalApostado) asc
```

OTRA “que no entiendo”

```
SELECT cb.nombreCaballo
FROM Caballos cb JOIN Carreras cr ON (cb.idCarrera = cr.idCarrera)
WHERE cb.puesto = 1 AND DAY (cr.fecha) = DAY ( GETDATE() )
GROUP BY cb.nombreCaballo
HAVING (SELECT SUM(cb3.totalApostado)
        FROM Caballos cb3
        WHERE cb3.nombreCaballo = cb.nombreCaballo
        GROUP BY cb3.nombreCaballo ) =
                                (SELECT TOP 1 SUM(cb2.totalApostado)
                                FROM Caballos cb2
                                GROUP BY cb2.nombreCaballo
                                ORDER BY SUM(cb2.totalApostado) asc)
ORDER BY SUM (cb.totalApostado) asc
```

b) Cree el/los objetos de base de datos necesarios para implantar la siguiente regla "Por cada carrera el contenido no nulo del campo PUESTO debe tener valores consecutivos iniciando en 1". Se sabe que en la actualidad dicha regla se cumple y que la base de datos es accedida por n aplicaciones de diferentes tipos y tecnologías.

2019-Final_25_de_Septiembre

Dado el siguiente modelo resuelva:

PERSONA:

id
fecha_nacimiento
apellido
nombre

su_padre_es

a) Escriba la sentencia SQL necesaria para que se muestren las personas con sus abuelos paternos. Se debe mostrar nombre y apellido de la persona y nombre y apellido de su abuelo

```
SELECT h.Apellido, h.Nombre, a.Apellido, a.Nombre FROM PERSONA h
                             JOIN PERSONA p ON h.su_padre_es = p.id
                             JOIN PERSONA a ON p.su_padre_es = a.id
```

b) Escriba la sentencia SQL necesaria para que se muestren los padres y cuantos hijos tienen. Se debe mostrar nombre y apellido del padre y cuantos hijos tienen.

```
SELECT p.Nombre, p.Apellido, COUNT(h.su_padre_es) as 'CANT. HIJOS'
      FROM PERSONA p
      JOIN PERSONA h ON h.su_padre_es = p.id
      GROUP BY p.Nombre, p.Apellido, h.su_padre_es
```

2019-Final_04_de_Diciembre

a) Se realiza la siguiente consulta de datos

```
SELECT p.prod_codigo, COUNT(*) FROM PRODUCTOS p, DEPOSITO d
GROUP BY p.prod_codigo HAVING COUNT(*)>0
```

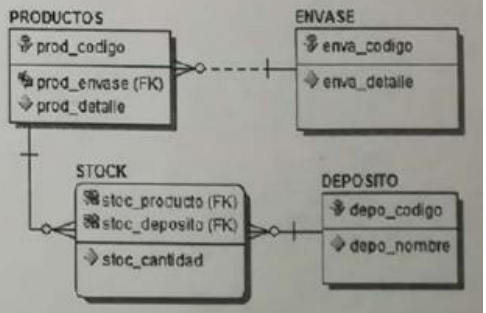
Sabiendo que la tabla PRODUCTOS tiene 150000 registros y la tabla de DEPOSITO no posee datos, que ocurre al ejecutar la misma, justificar la respuesta:

1)Devuelve 15000 filas con prod_codigo con valor y COUNT(*) = 1
2)Da error al intentar ejecutar
3)Devuelve un conjunto vacio de datos
4)Devuelve una fila con prod_codigo vacio y COUNT(*) = 0
5)Ninguna de las anteriores (especificar que ocurre)

La respuesta correcta es la 3: devuelve un conjunto vacío de datos. En el from al tomar las filas del producto cartesiano entre una tabla con datos y una tabla vacía, siempre el resultado va a ser un conjunto de datos vacíos ya que el producto cartesiano es vacío.

3b Dado el siguiente DER

Se le solicita a un programador realizar una tarea de reasignación de depósitos para los productos, de la siguiente manera: Aquellos productos que no posean envase deberán ser asignados al depósito cuyo nombre comience con "CABA" (en caso de haber más de un deposito se deberá considerar el de menor código), caso contrario los productos deberán ser reasignados al depósito cuyo código es 19.
El programador presenta la siguiente solución:




```
create function FN_REASIGNAR_PRODUCTOS ()
returns bit
as
    update stock
    set stoc_deposito = (select top 1 depo_codigo
                         from deposito
                         where depo_nombre like 'CABA%'
                         ORDER BY depo_codigo asc)
    where stoc_producto in (select prod_codigo
                            from producto
                            where prod_envase is null)
    update stock
    set stoc_deposito = 19
    where stoc_producto not in (select prod_codigo
                                from producto
                                where prod_envase is null)

    RETURN 1
END
```

Responda si la solución brindada es correcta o incorrecta. En caso de ser incorrecta realice las correcciones necesarias para que la solución sea válida. **Justifique su respuesta en ambos casos.**

Se puede notar rápidamente que la respuesta es incorrecta. El programador realizó bien la lógica dentro de la función pero olvidó el hecho de que las funciones no pueden modificar los datos de tablas por lo que tendría que haber creado un stored procedure.
Los cambios serían cambiar el "create function" por "create procedure" y eliminar las 2 líneas que tienen "return" para que la sintaxis coincida con la de un procedimiento.

2018-Final_17_de_Julio
Dato el siguiente modelo de entidad relacion y la consulta asociada:

EMPRESA:
id_empresa: char(5)
razon_social: char(40)
presidente: char(60)

AREA:
id_area: char(18)
id_empresa: char(5)
detalle: varchar(60)

EMPLEADO:
id_area: char(18)
id_empresa: char(5)

cuit: char(11)

```
SELECT e.id_empresa, count(distinct a.id_area), count(em.id_empresa)
      FROM empresa e JOIN area a on e.id_empresa = a.id_empresa LEITH JOIN
empleado em ON (a.id_area=em.id_area and a.id_empresa = em.id_empresa)
      GROUP by e.id_empresa
```

Opción	Filas	Columna 1	Columna 2	Columna 3
A	1 por c/ empresa	Id_empresa	Cantidad de áreas	Cantidad de empleados
B	1 por c/ empresa con al menos 1 área	Id_empresa	Cantidad de áreas	Cantidad de empleados
C	1 por c/ empresa con al menos 1 área y un empleado	Id_empresa	Cantidad de áreas por cantidad de empleados del área	Cantidad de empleados
D	1 por c/ empresa con al menos 1 área	Id_empresa	Cantidad de áreas por cantidad de empleados	Cantidad de áreas por cantidad de empleados

a)Determine cual de las siguientes opciones coincide con el resultado que arrojaria la consulta, en caso de no ser ninguna de estas explique claramente cual seria el resultado:

La respuesta más acertada sería la C. Lo que aclararía es que en la columna 2 no veríamos la cantidad de áreas total sino que la cantidad de áreas con empleados de la empresa. Y en la columna 3 veríamos la cantidad de empleados de la empresa de la columna 1.

b)Desarrolle una consutla que retorne, un renglon por cada empresa, que contenga al menos 1 area y que ninguna de las areas tenga mas de 10 empleados con las siguientes columnas: id_empresa, cantidad de areas y cantidad de empleados

```
SELECT e1.id_empresa, COUNT(distinct a1.id_area) AS cant_Areas,
COUNT(em1.id_area) AS cant_Empleados
      FROM EMPRESA e1 LEFT JOIN AREA a1 ON e1.id_empresa = a1.id_empresa
                  JOIN EMPLEADO em1 ON (em1.id_area = a1.id_area and
em1.id_empresa = e1.id_empresa)
      WHERE a1.id_area IN (SELECT a2.id_area FROM AREA a2
                        JOIN EMPLEADO e2 ON a2.id_area = e2.id_area
                        GROUP BY a2.id_area
                        HAVING COUNT(e2.cuit) > 10)
      GROUP BY e1.id_empresa
```

2018-Final_12_de_Marzo

Dada las siguientes tablas

USUARIOS:

idUsuario int (pk)
Nombre char(100) not null
Apellido char(100) not null
FechaAlta date not null

INGRESOS:
IdUsuario int (fk)
Fecha date not null

La primer tabla contiene todos los usuarios de un aplicativo, la segunda los logueos al mismo, el campo Fecha no contiene hora por lo cual si un usuario ingresa mas de una vez en el dia solo se inserta un registro en la tabla. La tabla usuarios no acepta borrado de datos.

a)La empresa solicita que se cree una vista que obtenga el ultimo ingreso al aplicativo de cada usuario mostrando nombre, apellido y la fecha, en caso de que un usuario nunca haya accedido debe mostrarse la fecha de alta del usuario como ultimo ingreso, un programador lo resulve de la siguiente manera.

```
CREATE VIEW vw_final(nombre, apellido,ultimoingreso) as
      SELECT nombre, apellido, max(fecha) from INGRESOS i, USUARIOS
            WHERE i.idUsuario = usuarios.IdUsuario
            GROUP BY usuarios.IdUsuario, apellido, nombre
            UNION
            SELECT nombre, apellido, fechaAlta from USUARIOS
```

Seleccione la opcion correcta, solo en el caso de elegir la opciones II a IV justifique la respuesta y reescriba la vista con las correcciones mencionadas en la justificacion.

I. La vista es correcta y devuelve exactamente lo solicitado
II. La vista se crea pero al consultarla no devuelve lo solicitado
III. La vista se crea pero da error al consultarla
IV. La vista no puede crearse

Respuesta correcta (II)

```
CREATE VIEW vw_final (nombre, apellido,ultimoingreso) AS
      SELECT nombre, apellido, max(fecha) from INGRESOS i, USUARIOS
            WHERE i.idUsuario = usuarios.IdUsuario
            GROUP BY usuarios.IdUsuario, apellido, nombre
            UNION
            SELECT nombre, apellido, fechaAlta from USUARIOS u2
            LEFT JOIN INGRESOS i2 ON u2.idUsuario = i2.idUsuario
            WHERE i2.Fecha IS NULL
```

b) Se toma la decision de comenzar a depurar la tabla de usuarios, se ejecuta la siguiente instruccion y da error.
DELETE FROM usuarios WHERE idUsuarios = 8
Explique el motivo y luego implemente y desarrolle el/los objetos necesarios para que se puedan eliminar los usuarios a traves de sentencias como la anterior.

```
CREATE TRIGGER tr_BorrarUsuarios ON USUARIOS INSTEAD OF DELETE
AS
set nocount on
BEGIN
      IF (EXISTS (SELECT * FROM INGRESOS, DELETED WHERE INGRESOS.idUsuario
= deleted.idUsuario))
            DELETE FROM INGRESOS WHERE ingresos.idUsuario = (SELECT
idUsuario FROM DELETED)
      DELETE FROM USUARIOS WHERE idUsuario = (SELECT idUsuario FROM
DELETED)
END
```

2017-Final_11_de_Julio

Tiempo	Transacción A	Transacción B
0	Begin Transaccion	Begin Transaccion
1	Select * from product where id = 1	
2		Select * from product where id = 1
3	Update product set detalle = 'X' where id = 1	
4		Update product set detalle = 'Y'
5	commit	rollback

a) Las transacciones A y B son ejecutadas ambas con un nivel de aislamiento SERIALIZABLE. Responder cual es la respuesta correcta.
1) Se modifica solamente el producto cuto id = 1, con un valor X en el atributo detalle.
2) Se genera un interbloqueo.
3) La transaccion B queda desestimada porque aplica un rollback al final, dejando que la transaccion A realice la modificacion.
4) Ninguna de las anteriores

1

Hago un paso a paso:
Tiempo 1 => TA toma lockeo de la tabla product con clave id = 1
Tiempo 2 => TB queda bloqueado ya que, por estar en SERIALIZABLE, quiere hacer un select el cual implica lockear la tabla. Como ya existe un lock (lo tiene TA) se queda bloqueado esperando.
Tiempo 3 => TA hace el update sin problemas ya que posee el lock
Tiempo 5 => commitea la transaccion
Tiempo 6 => se libera el lock, lo toma TB, hace el select, hace el update y rollbackea.

Resultado => la tabla product con id=1 queda con detalle = X.

b) Dada la siguiente tabla
CREATE TABLE empleados

(cod_empleado INT PRIMARY KEY,
des_empleado VARCHAR(40),
cod_jefe INT REFERENCES empleados)

Se desea crear una Vista "Estructura_Jefes" que devuelva los siguientes valores: Codigo de Jefe, Descripcion del Jefe, Codigo de Empleado, Descripcion del Empleado, para todos los jefes que tengan mas de 4 empleados a cargo.
La vista debera mostrar como titulo de cada valor el siguiente literal "Codigo_Jefe, Nombre_Jefe, Codigo_Empleado, Nombre_Empleado"

```
ALTER VIEW vw_JefesyEmpleados
(Codigo_Jefe,Nombre_Jefe,Codigo_Empleado,Nombre_Empleado)
AS
SELECT j1.cod_empleado, j1.des_empleado, e1.cod_empleado, e1.des_empleado
FROM EMPLEADOS j1
      JOIN EMPLEADOS e1 ON e1.cod_jefe = j1.cod_empleado
      WHERE j1.cod_empleado IN (SELECT j2.cod_empleado FROM EMPLEADOS j2
JOIN EMPLEADOS e2 ON j2.cod_empleado = e2.cod_jefe
      GROUP BY j2.cod_empleado
      HAVING COUNT(e2.cod_jefe) >3)
```

2017-Final_12_de_Mayo

a) Se realiza la siguiente consulta de datos:

```
select e.cod_emp, count(*)
from empleados e, perfiles
group by e.cod_emp
```

Sabiendo que la tabla de empleados tiene 100 registros y la tabla de perfiles no posee datos, que ocurre al ejecutar la misma, justificar la respuesta.

1) Devuelve 100 filas con cod_emp con valor y COUNT(*) = 1

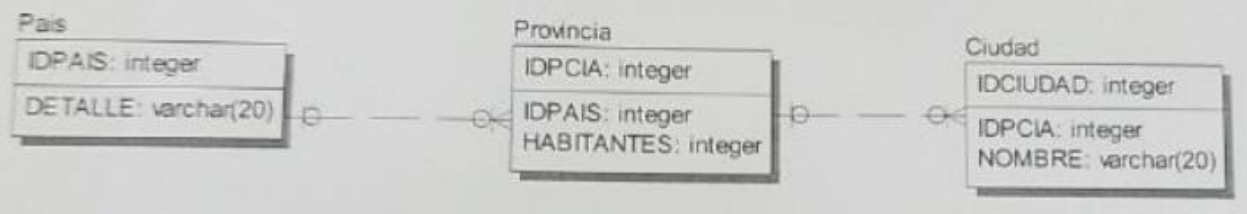
2) Devuelve un conjunto vacío de datos.

3) Cancela al intentar ejecutarla.

4)Devuelve una fila con cod_emp vacío y count(*) = 0.

La respuesta correcta es la 2. Hacer un FROM <Tabla1>, <Tabla2> es lo mismo que hacer el producto cartesiano entre dos tablas. Como en la consulta tenemos FROM empleados e, perfiles y perfiles es una tabla sin datos, el producto cartesiano da un conjunto vacío de datos.

2017-Final_25_de_Julio



a)Cree el/los objetos de base de datos necesarios para implantar la siguiente regla de negocio. "Si una provincia no posee ciudades, el campo Habitantes debe ser 0 o nulo, en caso contrario debe ser mayor o igual a 0". Se sabe que en la actualidad dicha regla se cumple y que la base de datos es accesible por n aplicaciones de diferentes tipos y tecnolocias.

```
CREATE TRIGGER tr_HabitantesProv ON CIUDAD INSTEAD OF INSERT, UPDATE
AS
BEGIN
IF ((NOT ISNULL(SELECT idPcia FROM inserted)and (ISNULL (SELECT habitantes
FROM Provincia,inserted WHERE provincia.idPcia = inserted.idPcia)
UPDATE Provincia SET habitantes = 0 WHERE inserted.idPcia =
Provincia.idPcia
```

b)Determinar cual de las siguientes opciones coincide con el resultado que arroja la consulta, en caso de no ser ninguna de estas explique claramente cual seria el resultado.

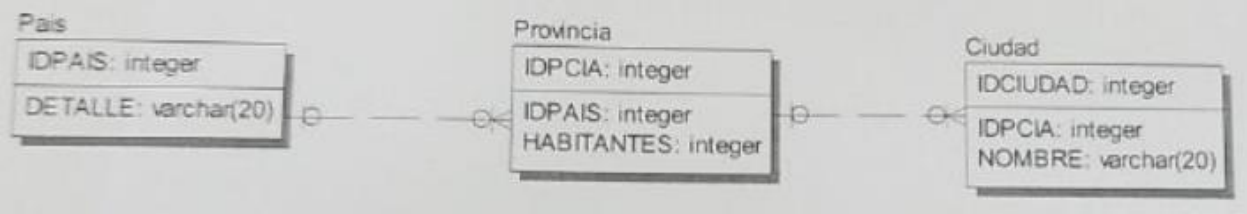
```
SELECT pro.idPcia, isnull(pro.habitantes,1), pa.detalle
FROM provincias pro inner join PAIS pa on pro.idPais = pa.idPais
WHERE pro.Habitantes = (SELECT max(pro2.Habitantes) FROM provincias pro2
where pa.idPais = pro2.idPais)
ORDER BY pro.Habitantes
```

Opción	Filas	Columna 1	Columna 2	Ordenamiento
A	Como mínimo 1 por c/ país con al menos 1 provincia	Id de la provincia	Cantidad de habitantes de la provincia o 1 si los habitantes son nulos	Por cantidad de habitantes, quedando los nulls arriba
B	Como mínimo 1 por c/ país con al menos 1 provincia	Id de la provincia	Cantidad de habitantes de la provincia o 1 si los habitantes son nulos	Por cantidad de habitantes, quedando los nulls abajo
C	Como mínimo 1 por c/ país con al menos 1 provincia que tiene al menos 1 ciudad	Id de la provincia	Cantidad de habitantes de la provincia	Por cantidad de habitantes
D	1 por c/ país con al menos 1 provincia que tiene al menos 1 ciudad	Id de la provincia	Cantidad de habitantes de la provincia	Por cantidad de habitantes

RESPUESTA (C)

2017-Final_11_de_Julio

Dado el siguiente modelo de datos resuelva:



a)Escriba una consulta ANSI SQL que retorne todas las filas de la tabla Pais. La columna a mostrar son: FILA (numero de fila que se esta mostrando), ID (id del pais), DETALLE (detalle del pais). El resultado debe ser ordenado por ID descendente. No se pueden utilizar ningun tipo de funcion ni variable auxiliar propietaria de ningun motor en particular (por ej, TOP, ROWIND, ROWNUM, etc.).

Ejemplo de resultado

FILA	IDPAIS	DETALLE
1	23	Peru
2	11	Argentina
3	2	Chile

```
SELECT (SELECT COUNT(p2.id_pais) + 1 AS 'FILA' FROM pais p2 WHERE
p1.id_pais < p2.id_pais) , p1.id_pais, p1.detalle
FROM pais p1
ORDER BY p1.id_pais DESC
```

b) Determine cuál de las siguientes opciones coincide con el resultado que arrojaría la consulta, en caso de no ser ninguna de estas explique claramente cuál sería el resultado.

```
SELECT pa.IDPAIS, sum(pro.HABITANTES), count(*)
      FROM PAIS pa, PROVINCIAS pro, CIUDAD c
      WHERE pa.IDPAIS = pro.IDPAIS and pro.IDPCIA = c.IDPCIA
      GROUP BY pa.IDPAIS
```

Opción	Filas	Columna 1	Columna 2	Columna 3
	1 por c/ pais con al menos 1 provincia y que dicha provincia tenga al menos una ciudad	Idpais	Cantidad de habitantes del pais	Cantidad de ciudades del pais.
	1 por c/ pais	Idpais	Cantidad de habitantes del pais	Cantidad de provincias del pais
	1 por c/ pais con al menos 1 provincia	Idpais	Cantidad de habitantes de la provincia	Cantidad de ciudades de la provincia
	1 por c/ pais con al menos 1 provincia	Idpais	Cantidad de habitantes del pais	Cantidad de áreas por cantidad de empleados

OPCION A.

2016-Final_20_de_Diciembre

a)Dado el siguiente query escrito en pl/sql, reescribirlo en ansi sql sin utilizar outer join. Se aclara que la tabla de facturas posee nulos en el campo codcli cuando la misma esta anulada.

SELECT nrofactura, importe, NVL(t2.nombre,'Sin Cliente') FROM facturas t1, clientes t2
WHERE t1.codcli = t2.codcli (+)
AND t2.fecha > (sysdate - 365)

```
select nombre, importe, cli_id
      from cliente c, factura f
      where f.cli_id = c.cli_id and f.anio > (sysdatetime - 365) and
cli_id is not null
union
select nombre, importe, isnull(cli_id, 'sin cliente')
      from cliente c, factura f
      where f.cli_id = c.cli_id and f.anio > (sysdatetime - 365) and
cli_id is null
```

b)Se solicita obtener un listado de las facturas no anuladas del ultimo año. Un programador lo resuelve de la siguiente forma:

SELECT f.* FROM Clientes c, Factura f WHERE c.codcli = f.codcli AND f.fecha > (SYSDATE - 365) AND f.codcli IS NOT NULL

Responde que sucede al ejecutar y justifique la respuesta:
1) La consulta se ejecuta correctamente.
2) La consulta se ejecuta, pero genera un producto cartesiano
3) La consulta se ejecuta correctamente, pero existe un filtro redundante
4) La consulta se ejecuta, pero el resultado obtenido no es el solicitado.
5) La consulta se ejecuta, pero existe un filtro redundante y ademas el resultado no es el solicitado.
6) La consulta cancela al ejecutarse.

En teoria es la respuesta 3 pero para mi tambien puede ser la 5 porque me estoy trayendo todas las facturas del ultimo año pero no hay una condicion que me diga que son las no anuladas.

2016-Final_14_de_Mayo

a)La siguiente sentencia SQL se ejecuta sobre una tabla que posee al menos 100 registros.
El campo clave no posee nulos y tiene un solo valor duplicado.
Indicar si dicha sentencia da error al ejecutar, en caso afirmativo detallar el inconveniente, en caso negativo indicar que informacion devuelve.

```
SELECT clave FROM Secuencias
      WHERE clave IN (SELECT TOP 5 clave FROM Secuencias
                        ORDER BY clave)
```

```
UNION ALL
SELECT clave FROM Secuencias
      WHERE clave IN (SELECT TOP 5 clave FROM Secuencias
                        ORDER BY clave DESC)
ORDER BY 1
```

Trae las 5 claves de la tabla secuencia que coinciden con las 5 primeras claves ordenadas de manera ascendente.
unidas con las 5 claves de la tabla secuencia que coinciden con las 5 primeras claves ordenadas de manera descendente
ordena todo ascendente.

NOTA: La consulta va a traer claves duplicadas, triplicadas, etc en caso de que haya, ya que el union all no elimina duplicados.

b)En una unica instruccion DML,implemente una solucion para eliminar los registros duplicados en el campo clave. Si se volviera a ejecutar la sentencia anterior luego de eliminar los duplicados, la cantidad de filas resultantes puede variar?

```
DELETE FROM Secuencias WHERE clave IN (SELECT clave FROM Secuencias
                                       GROUP BY CLAVE
                                       HAVING COUNT(clave) > 1)
```

No varia ya que posee 100 registros y solo me quedo con 10 de ellos en la consulta; y de duplicados elimine solo uno.

2016-Final_01_de_Marzo

Dada la siguiente tabla:
PERSONA:
id int PK
NOMBRE varchar(50)
Fecha_Nac date
Padre int FK

a)Describa claramente que retorna la siguiente consulta. En caso que de error detalle linea por linea los errores encontrados.

```
SELECT a.nombre, a.nombre
FROM persona a join persona h on h.padre=a.ID join persona a on
a.padre=h.ID
AND not exists(select 1 from persona a2 join persona h2 on a2.padre=h2.ID
and h2.padre=a.ID and a.Fecha_nac<a2.Fecha_nac)
```

La consulta compila bien, anda joya. Y devuelve los Abuelos con los nietos menores

Abuelo1 - NietoMenor1
Abuelo2 - NietoMenor2

b)Desarrolle el codigo de como implementaria la siguiente regla de negocio SIN ALTERAR LA ESTRUCTURA DE LA TABLA: "Dos hermano no pueden llamarse igual"

```
CREATE TRIGGER ReglaHermanos on Persona
Instead of INSERT
AS
BEGIN
IF Exits (select 1 FROM PERSONA p inner join inserted i on p.idpadre =
i.idpadre and p.nombre = i.nombre)
PRINT 'No se puede registrar a una persona con un hermano del mismo
nombre'
ELSE
INSERT INTO Persona select * from INSERTED

END
```

---

https://www.utnianos.com.ar/foro/tema-final-gdd-4-12-2018
https://www.utnianos.com.ar/foro/tema-aporte-final-gdd-30-07-18
https://www.utnianos.com.ar/foro/tema-gdd-final-14-02-2018
https://www.utnianos.com.ar/foro/tema-aporte-final-gesti%C3%B3n-de-datos-21-02-17
https://www.utnianos.com.ar/foro/tema-aporte-final-gesti%C3%B3n-de-datos-14-02-17
https://www.utnianos.com.ar/foro/tema-aporte-gestion-de-datos-final-13-12-2016
https://www.utnianos.com.ar/foro/tema-aporte-final-gdd-06-12-16
https://www.utnianos.com.ar/foro/tema-aporte-final-gestion-de-datos-06-10-2016
https://www.utnianos.com.ar/foro/tema-pedido-gdd-final-del-23-02-2016
https://www.utnianos.com.ar/foro/tema-aporte-final-gdd-16-02-16
https://www.utnianos.com.ar/foro/tema-aporte-final-gdd-01-10-15
https://www.utnianos.com.ar/foro/tema-aporte-final-gdd-15-12-15
https://www.utnianos.com.ar/foro/tema-aporte-gesti%C3%B3n-de-datos-final-01-12-2015
https://www.utnianos.com.ar/foro/tema-aporte-final-gestion-de-datos-28-07-2015
https://www.utnianos.com.ar/foro/tema-aporte-final-de-gestion-de-datos-16-12-2014
https://www.utnianos.com.ar/foro/tema-pedido-final-de-gestion-de-datos-02-12-2014
https://www.utnianos.com.ar/foro/tema-aporte-final-gesti%C3%B3n-de-datos-26-05-2014

https://www.utnianos.com.ar/foro/tema-pedido-final-gesti%C3%B3n-de-datos-11-02-2014
https://www.utnianos.com.ar/foro/tema-pedido-final-gestion-de-datos-06-08-2013
https://www.utnianos.com.ar/foro/tema-aporte-gesti%C3%B3n-de-datos-final-26-02-2013
https://www.utnianos.com.ar/foro/tema-pedido-gestion-de-datos-final-04-12-2012