Base de Datos I - Parcial 2 (2021C2)

Dado el siguiente MR de una empresa de reparacion de lavarropas, donde las PK están en **negrita** y las FK subrayadas, realice las operaciones que se detallan a continuación utilizando SQL.

Reparacion(**nro**, fecha, importe, id_cliente, cuil_empleado)
Cliente(**id**, nombre, cod_loc, fecha_nac)
Localidad(**cod**, descripcion)
Empleado (**cuil**, nombre, cod_loc, sueldo)
Insumo(**nro**, descripcion, costo, cuit_proveedor)
Proveedor(**cuit**, razon_social, telefono)
Reparacion_Insumo(**nro_reparacion**, **nro_insumo**)

Nota: el campo fecha es de tipo DATE (no es DATETIME)

Puntos: 5/10

1. Listar la descripción de aquellos insumos que se han utilizado en todas las reparaciones.
(0/1 puntos)

```
select i.descripcion as 'Insumos'
        from insumo i join reparacion_insumo ri on i.nro =
        ri.nro_insumo
        group by i.descripcion
        having count(*)=(select max(total)
                                from (select count(*)as 'total'
                                                from reparacion_insumo ri2
                        group by ri2.nro_insumo)alias);
```

"El count debe tener un distinct por reparación. Subconsulta incorrecta"

2. Modificar los empleados que comienzan con la letra “A” de manera que tengan el mismo sueldo del empleado de CUIL 123456789
(1/1 puntos)

```
update empleado e
set e.sueldo=(select e2.sueldo
                        from empleado e2
                        where e2.cuil=123456789)
where e.nombre like 'A%';
```

3. Eliminar a todas las reparaciones que se hayan efectuado con un empleado que viva en “San Justo”
(0.5/1 puntos)

```
delete from reparacion r
where r.cuil_empleado in ( select e.cuil
                                    from empleado e join localidad l on e.cod_ciu=l.cod_ciu
                    where l.descripcion like "San Justo"
                                    );
```

“No utilizar LIKE si no se compara con un patrón”

4. Listar el nombre de aquellos clientes que hayan efectuado más de 5 reparaciones superiores a $1000.
(0/1 puntos)

```
select distinct c.nombre
      from cliente c join reparacion on c.id=r.id_cliente
   where r.importe>1000
   and 5<(select count(*)
    from cliente c2 join reparacion r2 on c2.id=r2.id_cliente
   );
```

“Al juntar cliente con reparación en la consulta principal hace que los resultados obtenidos sean invalidos”

5. Indicar el nombre y sueldo de aquellos empleados que NO hayan realizado reparación alguna en todo el año 2014.
(0/1 puntos)

```
 select e.nombre, e.sueldo
from empleado join reparacion r on e.cuil=r.cuil
where  r.fecha not exists(select 1
                                    from reparacion r2
                  where r2.fecha >="2014-01-01" and r.fecha<="2014-12-31"
                  and e.cuil=r2.cuil);
```

"Al juntar la tabla Empleado con Reparacion en la consulta principal hace que no funcione"

6. Agregar un campo nuevo a la tabla Proveedor, que indique la localidad donde el mismo vive, utilizando el tipo de dato y las restricciones de integridad que correspondan.
(1/1 puntos)

```
alter table proveedor add column cod_localidad  smallint not null;
alter table proveedor add foreign key (cod_localidad) references localidad(cod);
-- Aunq tambien lo pense en un solo paso
alter table proveedor add column cod_localidad  smallint not null, add foreign key
(cod_localidad) references localidad(cod);
```

7. Agregar un nuevo cliente de nombre "Juan Perez", id 15, fecha de nacimiento 21 de mayo de 1956 y código de localidad 34.
(1/1 puntos)

```
insert into cliente, (id, nombre, cod_loc, fecha_nac)
values (15,"Juan Perez",34,"1956-04-21");
```

8. Listar las reparaciones realizadas en el año 2012. Por cada una de ellas mostrar: fecha, importe, nombre de cliente y descripción de su localidad y finalmente el nombre del empleado que realizó la reparación. Ordenar los resultados por fecha (las más recientes primero).
(0.5/1 puntos)

```
select r.nro, r.fecha, r.importe, c.nombre as'Cliente', l.descripcion, e.nombre as 'Empleado'
	from reparacion r    join cliente c on r.id_cliente=c.id
                         join empleado e on r.cuil_empleado=e.cuil
                          join localidad l on c.cod_loc=l.cod
	where r.fecha>='2012-01-01' and r.fecha <='2012-12-31'
	order by r.fecha
```

"Sentido incorrecto de ordenamiento"

9. Indicar las tres posibles operaciones que intervienen en una transacción de base de datos y explique brevemente para qué sirve cada una de ellas.
(1/1 puntos)

Hay 3 posibles, llamadas Commit, Begin y RollBack
El BEGING va antes de comenzar la transaccion, hara como un "Boceto" de lo que se efectuara luego, pero no estara oficialmente dado hasta llegar al Commit
El COMMIT viene luego del "Boceto" cuando por fin se ha efectuado todo, se podra hacer la transaccion oficalmente y como dijo el profesor "Pasarla a tinta" cumpliendo exitoso
El ROLLBACK es cuando entre el Beging y el Commit paso algo y la transaccion se ha quedado estancada por un tiempo, el ROLLBACK luego de dicho tiempo sin tener aviso borrara ese "Boceto" y dejara todo como estaba antes. Es una manera de asegurar (Tanto

10. Indicar el sueldo máximo de los empleados por cada localidad (descripción).
(0/1 puntos)

```
select l.descripcion, max(e.sueldo)
from empleado e join localidad l ON EMP.cod_loc = LOC.cod
group by l.cod;
```

“Falta agrupar por descripción de localidad si se quiere mostrar ese campo”

Este contenido lo creó el propietario del formulario. Los datos que envíes se enviarán al propietario del formulario. Microsoft no es responsable de las prácticas de privacidad o seguridad de sus clientes, incluidas las que adopte el propietario de este formulario. Nunca des tu contraseña.

Con tecnología de Microsoft Forms | Privacidad y cookies (https://go.microsoft.com/fwlink/p/?linkid=857875) | Términos de uso (https://go.microsoft.com/fwlink/p/?LinkId=2083423)