Guardería Gaghiel - Parcial SQL

Parcial AD - Tema A

AD.A1

Enunciado

AD.A1- Embarcación de mayor almacenamiento por tipo de embarcación. La empresa desea identificar a las embarcaciones que llevan más tiempo total almacenadas para cada tipo de embarcación. El total de horas de almacenamiento de cada embarcación se calcula como la sumatoria de las diferencia en horas entre la fecha y hora de fin de contrato y la fecha y hora de contrato (atención: en los contratos que no tienen aún fecha y hora de fin deberá usarse la actual). La embarcación con mayor hs totales será considerada como la de mayor almacenamiento
Se requiere listar para cada tipo de embarcación la embarcación con más horas totales almacenadas, indicando: código y nombre del tipo de embarcación; hin y nombre de la embarcación; número y nombre del propietario, fecha de la primera vez que almacenó la embarcación, cantidad de horas totales de almacenamiento, y de las salidas con dicha embarcación la última fecha de salida en 2024 y la cantidad de salidas durante 2024.
Si una embarcación no tiene tiempo de almacenamiento indicar 0 en las horas totales.
Se deberán listar todos los tipos de embarcación y aún si no tienen ningún almacenamiento indicando 0.
Se deberá mostrar la embarcación con más tiempo de almacenamiento aún si no tuvo salidas indicando 0 en la cantidad de salidas.
Ordenar los datos por horas totales de almacenamiento descendente, fecha del primer almacenamiento ascendente y cantidad de salidas descendente.

Resolución

```
with te_alm as (
    select te.codigo cod_tipo, te.nombre tipo
        , e.hin, e.nombre emb, e.numero_socio
        , s.nombre propietario
```

```
        , min(ec.fecha_hora_contrato) desde
        , sum(coalesce(timestampdiff(hour, fecha_hora_contrato,
coalesce(fecha_hora_baja_contrato,now())),0)) hs_tot
    from tipo_embarcacion te
    left join embarcacion e
        on te.codigo=e.codigo_tipo_embarcacion
    left join embarcacion_cama ec
        on e.hin=ec.hin
    left join socio s
    on e.numero_socio=s.numero
    group by te.codigo, e.hin
), max_alm as (
    select te_alm.cod_tipo, max(hs_tot) max_hs
    from te_alm
    group by te_alm.cod_tipo
)
select te_alm.cod_tipo, te_alm.tipo, te_alm.hin, te_alm.emb
    , te_alm.numero_socio, te_alm.propietario
    , te_alm.desde, te_alm.hs_tot
    , max(sal.fecha_hora_salida) ult_salida,
count(sal.fecha_hora_salida) salidas
from te_alm
inner join max_alm
    on te_alm.cod_tipo=max_alm.cod_tipo
    and te_alm.hs_tot=max_alm.max_hs
left join salida sal
    on te_alm.hin=sal.hin
    and sal.fecha_hora_salida between '20240101' and '20241201'
group by te_alm.cod_tipo, te_alm.tipo, te_alm.hin, te_alm.emb,
te_alm.numero_socio, te_alm.propietario,
te_alm.desde, te_alm.hs_tot
order by te_alm.hs_tot desc, te_alm.desde, salidas desc
```

AD.A2

Enunciado

AD.A2- Estado camas. La empresa ha decidido que necesita más información sobre el estado de las camas. Se decidió separar del estado la situación de uso en una nueva columna y mantener esta información actualizada por medio de triggers.
Se requiere:
Agregar en la tabla cama la columna en_uso (utilizar el tipo de dato apropiado).
Reflejar el valor de dicha columna según esta regla. Si tiene un

contrato en embarcacion_cama sin fecha y hora de baja de contrato está en uso. Caso contrario no lo está.
A través del uso de triggers al registrar un nuevo contrato de una embarcación sin fecha y hora de baja de contrato cambiar el valor de la columna en_uso para reflejar que se encuentra utilizada y al registrar una fecha y hora de baja de contrato reflejar que se encuentra libre.

Usa los siguientes datos para probar

insert into embarcacion_cama values ('ESP010',2,8,now(),null);

##y luego

update embarcacion_cama set fecha_hora_baja_contrato=now() where hin='ESP010' and fecha_hora_baja_contrato is null;

Resolución

```
alter table cama add column en_uso boolean null;

update cama
set en_uso=true
where exists (
    select 1
    from embarcacion_cama ec
    where ec.codigo_sector=cama.codigo_sector
    and ec.numero_cama=cama.numero
    and ec.fecha_hora_baja_contrato is null
);

update cama set en_uso=false where en_uso is null;

alter table cama change column en_uso en_uso boolean not null;

delimiter (◡‿◡)
drop trigger if exists tr_embarcacion_cama_contrata (◡‿◡)
create trigger tr_embarcacion_cama_contrata after insert on
embarcacion_cama
for each row
begin
    update cama set en_uso=true
    where cama.codigo_sector=new.codigo_sector
    and cama.numero=new.numero_cama
    and new.fecha_hora_baja_contrato is null;
end; (◡‿◡)
```

```
drop trigger if exists tr_embarcacion_cama_fin_contrato (◡‿◡)
create trigger tr_embarcacion_cama_fin_contrato after update on
embarcacion_cama
for each row
begin
    update cama set en_uso=false
    where cama.codigo_sector=new.codigo_sector
    and cama.numero=new.numero_cama
    and new.fecha_hora_baja_contrato is not null;
end; (◡‿◡)
delimiter ;
```

Parcial AD - Tema B

AD.B2

Enunciado

AD.B2- Cursos más exitosos. La empresa desea conocer para cada tipo de embarcación información de los cursos más exitosos. Se considera que el curso más exitoso para cada tipo de embarcación es el que tiene mayor cantidad de socios inscriptos dentro de los cursos de dicho tipo de embarcación (que se relaciona según la actividad realizada en el curso). Se requiere listar para cada tipo de embarcación el curso más exitoso indicando: código y nombre del tipo de embarcación; número, nombre y descripción de la actividad del curso, número de curso, cuántos días pasaron desde que se comenzó a dictar dicho curso, cantidad de inscriptos que tuvo y la cantidad de embarcaciones del tipo de embarcación que están actualmente almacenadas en la guardería.
Si para un tipo de embarcación no se dictó ningún curso debe mostrarse igualmente con 0 inscriptos y sin datos de la actividad o curso y si no hay embarcaciones almacenadas actualmente de dicho tipo debe mostrarse con 0.
Ordenar por cantidad de embarcaciones almacenadas descendente, cantidad de inscriptos ascendente y días desde que comenzó el curso ascendente.

Resolución

```
#opción A
with te_ins as (
```

```
    select te.codigo cod_tipo_emb
        , te.nombre tipo
        , a.numero nro_act, a.nombre actividad
        , c.numero curso, c.fecha_inicio
        , count(i.numero_socio) insc
    from tipo_embarcacion te
    left join actividad a
        on te.codigo=a.codigo_tipo_embarcacion
    left join curso c
        on a.numero=c.numero_actividad
    left join inscripcion i
        on c.numero=i.numero_curso
    group by te.codigo, c.numero, a.numero
)
select te_ins.cod_tipo_emb, te_ins.tipo
    , datediff(current_date, te_ins.fecha_inicio) dias_desde_inicio
    , te_ins.nro_act, te_ins.actividad
    , te_ins.curso
    , te_ins.insc
    , count(ec.hin) cant_emb_almac
from te_ins
inner join (
    select ti.cod_tipo_emb, max(ti.insc) max_insc
    from te_ins ti
    group by ti.cod_tipo_emb
) ti
    on te_ins.cod_tipo_emb=ti.cod_tipo_emb
    and te_ins.insc=ti.max_insc
left join embarcacion e
    on te_ins.cod_tipo_emb=e.codigo_tipo_embarcacion
left join embarcacion_cama ec
    on e.hin=ec.hin
    and (ec.fecha_hora_baja_contrato is null or
ec.fecha_hora_baja_contrato > now())
group by te_ins.cod_tipo_emb, te_ins.tipo, te_ins.fecha_inicio,
te_ins.nro_act, te_ins.actividad, te_ins.curso
order by cant_emb_almac desc, insc, dias_desde_inicio


#opción B
with te_ins as (
    select a.codigo_tipo_embarcacion cod_tipo_emb
        , a.numero nro_act, a.nombre actividad
        , c.numero curso, c.fecha_inicio
        , count(i.numero_socio) insc
        , datediff(current_date, c.fecha_inicio) dias_desde_inicio
    from actividad a
```

```
    left join curso c
        on a.numero=c.numero_actividad
    left join inscripcion i
        on c.numero=i.numero_curso
    group by a.codigo_tipo_embarcacion, c.numero, a.numero
), ti as  (
    select ti.cod_tipo_emb, max(ti.insc) max_insc
    from te_ins ti
    group by ti.cod_tipo_emb
), emb_tipo as (
    select e.codigo_tipo_embarcacion cod_tipo_emb, count(*)
cant_emb_almac
    from embarcacion e
    inner join embarcacion_cama ec
        on e.hin=ec.hin
    where ec.fecha_hora_baja_contrato is null or
ec.fecha_hora_baja_contrato > now()
    group by e.codigo_tipo_embarcacion
)
select te.codigo cod_tipo_emb, te.nombre tipo
    , te_ins.dias_desde_inicio
    , te_ins.nro_act, te_ins.actividad
    , te_ins.curso
    , te_ins.insc
    , emb_tipo.cant_emb_almac
from tipo_embarcacion te
left join te_ins
    on te.codigo=te_ins.cod_tipo_emb
left join ti
    on te_ins.cod_tipo_emb=ti.cod_tipo_emb
    and te_ins.insc=ti.max_insc
left join embarcacion e
    on te_ins.cod_tipo_emb=e.codigo_tipo_embarcacion
left join emb_tipo
    on te.codigo=emb_tipo.cod_tipo_emb
order by emb_tipo.cant_emb_almac desc, te_ins.insc,
te_ins.dias_desde_inicio
```

AD.B2

Enunciado

AD.B2- Embarcaciones almacenadas. La empresa ha detectado dificultad para identificar si las embarcaciones se encuentran almacenadas o no a la hora de cierre. Por este motivo se ha decidido agregar una columna “almacenada” en la tabla embarcación

que refleje la situación y automatizar con triggers el estado de dicha columna.
Se requiere
Crear una columna almacenada para reflejar el estado (utilizar el tipo de dato que crea apropiado).
Cargar el valor inicial de la columna. Las embarcaciones que no tengan una salida con fecha y hora de regreso real en null está almacenadas.
A través del uso de triggers al registrar una nueva salida de una embarcación cambiar el valor de la columna almacenada para reflejar que salió y al registrar una fecha y hora de regreso real reflejar que se encuentra almacenada.

Puede usar los siguientes datos para probar:

```
insert into salida
values ('can002',now(),'20241202T200000',null);
```

```
update salida
set fecha_hora_regreso_real=now()
where fecha_hora_regreso_real is null;
```

Resolución

```
alter table embarcacion add column almacenada boolean null;

update embarcacion
set almacenada=true
where hin not in (
    select hin
    from salida
    where fecha_hora_regreso_real is null
);

update embarcacion
set almacenada=false
where almacenada is null;


alter table embarcacion change column almacenada almacenada boolean not
null;

delimiter (◡‿◡)
drop trigger if exists tr_salida_salir (◡‿◡)
create trigger tr_salida_salir after insert on salida
```

```
for each row
begin
    update embarcacion set almacenada=false
    where hin = new.hin and new.fecha_hora_regreso_real is null;
end; (‿ˠ‿)


drop trigger if exists tr_salida_regreso (‿ˠ‿)
create trigger tr_salida_regreso after update on salida
for each row
begin
    update embarcacion set almacenada=true
    where hin = new.hin and new.fecha_hora_regreso_real is not null;
end; (‿ˠ‿)

delimiter ;

insert into salida values ('can002',now(),'20241202T200000',null);

##y luego

update salida set fecha_hora_regreso_real=now() where
fecha_hora_regreso_real is null;
```

Recuperatorio AD

REC.1

Enunciado

REC.1- Contratos y salidas. Para obtener información de forma rápida y concisa sobre los contratos y salidas de embarcaciones se requiere crear una rutina llamada contratos_salidas que reciba como parámetro el código de tipo de embarcación e indique para las embarcaciones de dicho tipo cuantas salidas en total ha realizado, la fecha de su última salida, la mayor cantidad de días que ha salido en una salida y para cada uno de sus contratos la cantidad de salidas durante la duración de dicho contrato y que proporción son sobre el total de salidas de dicha embarcación. Si no realizó salidas debe mostrar la cantidad de salidas en 0. Indicar hin y nombre de la embarcación; nombre del tipo de embarcación; nombre del propietario; cantidad total de salidas; fecha de la última salida, máxima cantidad de días que duró una salida realmente y para cada contrato, la fecha y hora

de contrato, fecha y hora de baja de contrato, cantidad de salidas en dicho contrato y que proporción representan del total. Invocar la rutina para con el código de tipo de embarcación 8 (incluirlo en la resolución).

Resolución

```
delimiter c[_]

drop procedure if exists contratos_salidas c[_]

create procedure contratos_salidas(in tipo int unsigned)
reads sql data
begin
with sal_ag as (
    select sal.hin
        , count(*) tot_salidas, max(sal.fecha_hora_salida) ult_salida
        , max(timestampdiff(day, sal.fecha_hora_salida,
sal.fecha_hora_regreso_real)) largo_max_salida
    from salida sal
    # pueden joinear acá con embarcación
    # y filtrar por tipo
    # y evitar el coalesce en el total de salidas con un count(sal.algo)
    group by sal.hin
)
select e.hin, e.nombre
    , te.nombre
    , so.nombre
    , ec.fecha_hora_contrato, ec.fecha_hora_baja_contrato
    , coalesce(sal_ag.tot_salidas,0), sal_ag.ult_salida
,sal_ag.largo_max_salida
    , count(sa.hin) salidas_durante_contrato, count(sa.hin)
from embarcacion e
inner join tipo_embarcacion te
    on e.codigo_tipo_embarcacion=te.codigo
inner join socio so
    on e.numero_socio=so.numero
left join embarcacion_cama ec
    on e.hin=ec.hin
left join sal_ag
    on e.hin=sal_ag.hin
left join salida sa
    on e.hin=sa.hin
    and sa.fecha_hora_salida between ec.fecha_hora_contrato and
coalesce(ec.fecha_hora_baja_contrato, now())
where e.codigo_tipo_embarcacion=tipo
```

```
group by e.hin, ec.fecha_hora_contrato;
end c[_]

delimiter ;

call contratos_salidas(8);
```

REC.2

Enunciado

REC.2- Mejoras al modelo. Tras haber utilizado nuestro sistema algún tiempo se ha detectado una nueva necesidad de la empresa. Las actividades deben poder asociarse a más de un tipo de embarcación. Actualmente es una relación 1aN y debe ser modificada a una NaM.
Para ello se requiere realizar los siguientes cambios al modelo:
Implementar los cambios necesarios en el modelo para este cambio.
Migrar los datos de la relación actual a la nueva.
Eliminar la relación 1aN existente.

Resolución

```
create table actividad_tipo_embarcacion2(
  numero_actividad int unsigned not null,
  codigo_tipo_embarcacion int unsigned not null,
  primary key (numero_actividad, codigo_tipo_embarcacion),
  constraint fk_actividad_tipo_embarcacion_actividad
  foreign key (numero_actividad) references actividad(numero),
  constraint fk_actividad_tipo_embarcacion_tipo_embarcacion
  foreign key (codigo_tipo_embarcacion) references
  tipo_embarcacion(codigo)
);

insert into actividad_tipo_embarcacion
select numero, codigo_tipo_embarcacion
from actividad;

alter table actividad drop foreign key fk_actividad_tipo_embarcacion,
drop column codigo_tipo_embarcacion;
```

REC.3

Enunciado

REC.3- Consistencia de datos. Se detectó una redundancia entre los campos sector.tipo_operacion con valores: “Manual” y “Automático” y tipo_embarcacion.operacion_requerida con valores: “Manual” y “Automática”. Ambos significan lo mismo pero al utilizarlos con una descripción los datos fueron cargados inconsistentes, se desea resolver esto y evitar que vuelva a suceder creando una entidad tipo_operacion con un código numérico autoincremental y una descripción. A tal efecto se requiere:
Crear una nueva tabla operatoria con CP código autoincremental y una descripción de texto.
Corregir la inconsistencia entre los valores.
Insertar los valores ya consistentes en la nueva tabla.
Reemplazar las columnas descriptivas por una FK a la tabla operatoria con los códigos adecuados
Eliminar las columnas preexistentes.
Ayuda: utilizar character set utf8mb4 y collate utf8_unicode_ci para evitar errores;

Resolución

```
create table operatoria(
    codigo int unsigned not null auto_increment,
    descripcion varchar(255),
    primary key (codigo)
) character set utf8mb4 collate utf8mb4_unicode_ci;

alter table tipo_embarcacion
add column codigo_operatoria int unsigned null,
add constraint fk_tipo_embarcacion_operatoria
    foreign key (codigo_operatoria)
    references operatoria(codigo);

alter table sector
add column codigo_operatoria int unsigned null,
add constraint fk_sector_operatoria
    foreign key (codigo_operatoria)
    references operatoria(codigo);

begin;

update sector
set tipo_operacion='Automática'
where tipo_operacion='Automático';
```

```
insert into operatoria(descripcion)
select distinct tipo_operacion
from sector;

update sector s
inner join operatoria o
on s.tipo_operacion=o.descripcion
set s.codigo_operatoria=o.codigo;

update tipo_embarcacion te
inner join operatoria o
on te.operacion_requerida=o.descripcion
set te.codigo_operatoria=o.codigo;

commit;

alter table tipo_embarcacion
drop column operacion_requerida,
change column codigo_operatoria codigo_operatoria int unsigned not null;

alter table sector
drop column tipo_operacion,
change column codigo_operatoria codigo_operatoria int unsigned not null;
```

Ejercicios Extra (Merciless)

DMD.1

Enunciado

DMD.A1- Reubicar embarcaciones. La empresa quiere realizar un mantenimiento y ampliación del sector Bohemian Junkheap. Por este motivo se necesita reubicar todas las embarcaciones que se encuentran actualmente almacenadas en dicho sector en camas de otro sector que no estén ocupadas. Para ello se debe revisar para cada embarcación almacenada su tipo y en que otros sectores puede almacenarse y cuales camas están disponibles.
Se requiere listar para cada embarcación que se encuentre actualmente almacenada en Bohemian Junkheap todas las camas posibles donde podría almacenarse que estén disponibles y fuera de mantenimiento, en caso de que no haya ninguna cama disponible debería mostrarse igual sin los datos de camas disponibles.
Indicar hin, nombre y descripción de la embarcación; nombre del tipo de embarcación; número de cama actual; y de cada posible

cama número de la misma, código y nombre del sector al que pertenece.
Nota: El atributo estado de las de la cama solo indica si están en mantenimiento o no, no si están ocupadas o no.

Resolución

```
select e.hin, e.nombre nombre_embarcacion, e.descripcion
descripcion_embarcacion
, te.nombre tipo
, ec.numero_cama cama_actual
, cdisp.numero cama_posible , cdisp.codigo_sector codigo_sector_posible
, spos.nombre nombre_sector_posible
from embarcacion_cama ec
inner join sector sact
    on ec.codigo_sector=sact.codigo
inner join embarcacion e
    on ec.hin=e.hin
inner join tipo_embarcacion te
    on e.codigo_tipo_embarcacion=te.codigo
inner join sector_tipo_embarcacion ste
    on te.codigo=ste.codigo_tipo_embarcacion
    and sact.codigo <> ste.codigo_sector
left join cama cdisp
    on ste.codigo_sector=cdisp.codigo_sector
left join sector spos
    on cdisp.codigo_sector=spos.codigo
where (fecha_hora_baja_contrato is null or fecha_hora_baja_contrato >
NOW())
    and sact.nombre='Bohemian Junkheap'
    and not exists( #también se puede con not in pero con 2 atributos
        select 1
        from embarcacion_cama ec
        where ec.codigo_sector=cdisp.codigo_sector
        and ec.numero_cama=cdisp.numero
        and (ec.fecha_hora_baja_contrato is null or
ec.fecha_hora_baja_contrato ⩾ now())
    )
    and cdisp.estado <> 'Disponible'
```

DMD.2

Enunciado

DMD.B2- Recomendación de Actividades. La empresa desea iniciar una política más agresiva de oferta de cursos para embarcaciones no convencionales. Para ello quiere para cada socio que tenga una embarcación no convencional que se encuentre actualmente almacenada indicar todas las posibles actividades que puede realizar según el tipo de embarcación pero que el socio aún no haya realizado ya.
Se requiere listar para cada embarcación no convencional actualmente almacenada, el propietario, y las actividades que puede realizar con ella, según el tipo de embarcación, pero que no haya realizado en alguno de los cursos a los que se inscribió, si con alguna embarcación no hay ninguna actividad posible deberá listarse igualmente. Indicar, número y nombre del socio; hin y nombre de la embarcación; código y nombre del tipo de embarcación; número, nombre y descripción de la actividad; legajo, nombre y apellido de los instructores que podrían dictarla.

Resolución

```
select s.numero, s.nombre
    , e.hin, e.nombre
    , te.codigo codigo_tipo_embarcacion, te.nombre
nombre_tipo_embarcacion
    , a.numero numero_actividad, a.nombre nombre_actividad,
a.descripcion descripcion_actividad
    , i.legajo legajo_instructor, i.nombre nombre_instructor, i.apellido
apellido_instructor
from tipo_embarcacion te
inner join embarcacion e
    on te.codigo=e.codigo_tipo_embarcacion
inner join socio s
    on e.numero_socio=s.numero
inner join embarcacion_cama ec
    on e.hin=ec.hin
left join actividad a
    on a.codigo_tipo_embarcacion=te.codigo
left join instructor_actividad ia
    on a.numero=ia.numero_actividad
left join instructor i
    on ia.legajo_instructor=i.legajo
where (ec.fecha_hora_baja_contrato is null or
ec.fecha_hora_baja_contrato >=now())
    and not exists (
```

```
        select 1
        from inscripcion i
        inner join curso c
            on i.numero_curso=c.numero
        where i.numero_socio=s.numero
            and c.numero_actividad=a.numero
    )
    and te.nombre='No convencional'
```

DMD.3

Enunciado

DMD.B2- Estadísticas cursos para tipo de embarcación. Desde varias aplicaciones se necesita acceder a las estadísticas de los instructores, cursos, actividades e inscriptos a los cursos según el tipo de embarcación para el que se crea la actividad y para evitar discrepancias se decidió utilizar una rutina para ello. Se requiere entonces crear la rutina estadistica_cursos que reciba como parámetro un código de tipo de embarcación y para la misma indique las actividades que se dictan para dicho tipo, los instructores que pueden dictarlas, los cursos dictados y, la cantidad de socios inscriptos a dicho curso, para el instructor la cantidad total de socios que se han inscripto a cursos para dicho tipo de embarcación, la proporción que representan los inscriptos al curso sobre el total y la fecha del primer curso que dictó y la máxima duración de cursos para ese tipo de embarcación. Si el instructor no ha dictado cursos debe mostrarse la cantidad en 0 y si no se ha inscripto nadie también deben mostrarse las cantidades y proporción en 0.
Indicar legajo, nombre y apellido del instructor; número y nombre de la actividad, fecha de inicio del primer curso del instructor, duración máxima de un curso, cantidad total de socios inscriptos total a sus cursos y para cada curso que dictó ese instructor el número, cantidad de inscriptos al curso y cantidad que representa sobre el total de inscriptos a todos los cursos del instructor. (todos los cursos, tener el cuenta tipo de embarcacion)
Invocar la rutina para con el código de tipo de embarcación 6 (incluirlo en la resolución).
Ayuda:para calcular la cantidad de días entre dos fechas utilizar timestampdiff(day, fecha_desde, fecha_hasta)

Resolución

```
delimiter c[_]

drop procedure if exists estadistica_cursos c[_]

create procedure estadistica_cursos(in tipo int unsigned)
reads sql data
begin

with ins_insc as (
    select c.legajo_instructor, count(i.numero_socio) cant_tot
        , min(c.fecha_inicio) inicio_primer_curso
        , max(timestampdiff(day,fecha_inicio,fecha_fin)) durac_maxima
    from curso c
    inner join actividad a
        on a.numero=c.numero_actividad
    left join inscripcion i
        on c.numero=i.numero_curso
    where a.codigo_tipo_embarcacion=tipo
    group by c.legajo_instructor
)
select i.legajo, i.nombre, i.apellido
    , ia.numero_actividad, a.nombre
    , c.numero
    , ii.inicio_primer_curso, ii.durac_maxima, coalesce(ii.cant_tot,0)
cant_tot
    , count(insc.numero_socio) cant
    , coalesce(count(insc.numero_socio)/ii.cant_tot,0) prop
from instructor i
inner join instructor_actividad ia
    on i.legajo=ia.legajo_instructor
inner join actividad a
    on ia.numero_actividad=a.numero
left join ins_insc ii
    on ia.legajo_instructor=ii.legajo_instructor
left join curso c
    on ia.legajo_instructor=c.legajo_instructor
    and ia.numero_actividad=c.numero_actividad
left join inscripcion insc
    on c.numero=insc.numero_curso
where a.codigo_tipo_embarcacion=tipo
group by i.legajo, c.numero, a.numero;

end c[_]
```

```
delimiter ;

call estadistica_cursos(6);
```