Evaluaciones de Aprobación Directa 2022

Parcial AD

Enunciado 1

****Sugerencias de tours.**** Para una promoción se desea sugerir a cada asistente otros tours que sean de su interés. Para cada asistente que haya realizado más de un tour en alguna temática, identificar la temática más frecuente elegida por el asistente y sugerirle todos los tours futuros con la misma temática. Indicar dni, nombre, apellido y teléfono del asistente y de los tours, número, temática, fecha y hora de salida y precio unitario. Ordenar por apellido y nombre alfabético y fecha y hora de salida descendente.

Resolución 1

```sql
#### Opción A)

with asist_tema  as (
      select dni_asistente, t.tematica, count(*) cant
      from asistente_contrato ac
```

```
      inner join tour t
            on ac.nro_tour=t.nro
      group by dni_asistente, t.tematica
    having cant > 1
    order by count(*) desc
),
max_asist as (
      select dni_asistente, max(cant) max_cant
    from asist_tema
    group by dni_asistente
)
select a.dni, a.apellido, a.nombre, a.telefono
          , t.nro, t.tematica, t.fecha_hora_salida,
t.precio_unitario_sugerido
from asist_tema ate
inner join max_asist ma
      on ate.dni_asistente=ma.dni_asistente
    and ate.cant=ma.max_cant
left join tour t #se perdona un inner
      on t.tematica=ate.tematica
    and t.fecha_hora_salida>now()
inner join asistente a
      on ate.dni_asistente=a.dni
order by a.apellido, a.nombre, t.fecha_hora_salida desc;
```

```sql
#### Opción B)

select a.dni, a.apellido, a.nombre, a.telefono
          , t.nro, t.tematica, t.fecha_hora_salida,
t.precio_unitario_sugerido
from asistente a
left join tour t #se perdona un inner
on t.tematica=(
    select tour.tematica
    from asistente_contrato ac
    inner join tour
           on ac.nro_tour=tour.nro
    where ac.dni_asistente=a.dni
    group by tour.tematica
    having count(*) > 1
```

```
    order by count(*) desc
    limit 1
)
and  t.fecha_hora_salida > now()
order by a.apellido, a.nombre, t.fecha_hora_salida desc;
```

Enunciado 2

****Determinar el margen de ganancia por temática**.** Crear una función que dado un tour devuelva ~~la~~ el costo total de todas sus actividades a la fecha de salida y otra función que dado un tour devuelva el ingreso utilizando los contratos. Finalmente utilizando dichas funciones calcular el margen de ganancia total de cada temática para los cursos ya iniciados. Indicar, temática, cantidad de tours y margen de ganancia. Ordenar por margen de ganancia descendente.

Margen de ganancia: (ingresos - costos)/costos

Resolución 2

```sql
USE `role_play_events`;
DROP function IF EXISTS `costo_actividades_tour`;

USE `role_play_events`;
DROP function IF EXISTS `role_play_events`.`costo_actividades_tour`;
;

DELIMITER $$
USE `role_play_events`$$
CREATE DEFINER=`gestiondatos`@`%` FUNCTION
`costo_actividades_tour`(nro_t int unsigned) RETURNS decimal(10,3)
    READS SQL DATA
BEGIN
declare costo_total decimal(10,3);
with val_fec as (
     select c.codigo_locacion, c.nro_actividad, max(c.fecha_desde)
fec_val
    from costo c
    inner join escala e
```

```
           on e.codigo_locacion=c.codigo_locacion
        and e.nro_actividad=c.nro_actividad
    where c.fecha_desde ⩽(select t.fecha_hora_salida from tour t
where t.nro=nro_t)
    and e.nro_tour=nro_t
    group by c.codigo_locacion, c.nro_actividad
)
select sum(c.valor) into costo_total
from val_fec
inner join costo c
  on c.codigo_locacion=val_fec.codigo_locacion
  and c.nro_actividad=val_fec.nro_actividad;

RETURN costo_total;
END$$


CREATE FUNCTION `ingreso_contrato_tour` (nro_t int unsigned)
RETURNS decimal(10,3)
reads sql data
BEGIN
declare ingreso_tot decimal (10,3);

select sum(c.importe) into ingreso_tot
from contrata c
where c.nro_tour=nro_t;

RETURN ingreso_tot;
END$$

DELIMITER ;

select t.tematica,
sum((coalesce(ingreso_contrato_tour(t.nro),0)-coalesce(costo_activid
ades_tour(t.nro),0))/coalesce(costo_actividades_tour(t.nro),1))
ganancia
from tour t
where t.fecha_hora_salida<now()
group by t.tematica;
```

Enunciado 3

****Consistencia de datos.**** La empresa notó que tanto la temática del tour como la ambientación de locación son en realidad lo mismo. Se decidió crear la tabla tematica (identificada por un código autoincremental y con una descripción), cargar las descripciones con los datos de tour y ambientación y modificar las tablas tour y locación para migrar las viejas columnas por la correspondiente clave foránea a la tabla temática.

Resolución 3

```sql
```sql
use role_play_events;
create table tematica (
  codigo int unsigned not null auto_increment,
  descripcion varchar(255) not null,
  primary key (codigo));

alter table tour
add column cod_tematica int unsigned null,
add constraint fk_tour_tematica foreign key(cod_tematica) references
tematica(codigo) on delete restrict on update cascade;

alter table locacion
add column cod_tematica int unsigned null,
add constraint fk_locacion_tematica foreign key(cod_tematica)
references tematica(codigo) on delete restrict on update cascade;

begin;

insert into tematica(descripcion)
select distinct tematica from tour
union
select distinct ambientacion from locacion;

update tour
inner join tematica
     on tour.tematica=tematica.descripcion
set tour.cod_tematica=tematica.codigo;

update locacion
inner join tematica
     on locacion.ambientacion=tematica.descripcion
```
```

```
set locacion.cod_tematica=tematica.codigo;

commit;

alter table tour
modify column cod_tematica int unsigned not null,
drop column tematica;


alter table locacion
modify column cod_tematica int unsigned not null,
drop column ambientacion;
```

Recuperatorio AD

Enunciado 1

****Sugerir encargados a promover para guía.**** Debido al incremento en la demanda se deben nombrar nuevos guías para los tours de entre los encargados. Mostrar para cada temática el encargado que más escalas ya finalizadas ha coordinado en tours de dicha temática. Solo se deberán sugerir encargados que no hayan sido asignados como guías a ningún tour hasta ahora. Indicar temática, cuil, nombre y apellido del empleado y cantidad de escalas coordinadas. Ordenar por cantidad de escalas coordinadas descendente y temática alfabéticamente.

Resolución 1

```sql
with cant_enc as (
    select t.tematica, e.cuil_encargado, count(e.cuil_encargado)
cant
    from tour t
    inner join escala e
        on t.nro=e.nro_tour
    where e.fecha_hora_fin < now()
    and e.cuil_encargado not in (
        select tr.cuil_guia
        from tour tr
    )
    group by t.tematica, e.cuil_encargado
),
max_cant as (
    select ce.tematica, max(cant) max_cant
    from cant_enc ce
    group by ce.tematica
)
select ce.tematica, ce.cuil_encargado
    , e.nombre, e.apellido, ce.cant
from max_cant mc
inner join cant_enc ce
    on mc.tematica=ce.tematica
    and mc.max_cant=ce.cant
inner join empleado e
    on ce.cuil_encargado=e.cuil
order by ce.cant desc, ce.tematica;
```

Enunciado 2

****Cálculo de costo del personal por tour.**** Crear la función costo_hora_empleado que reciba el cuil y una fecha y devuelva el salario por hora a esa fecha.
Luego crear un SP costo_tours que reciba una fecha y hora y liste todos los tours que ya sucedieron (fecha y hora de regreso menor o igual dicha fecha y hora) indicando los costos del guía y la sumatoria de los costos de los encargados invocando a la función antes creada. Indicar número, temática, fecha y hora de salida y regreso del tour, costo del guía a la fecha y hora de salida, sumatoria del costo de los encargados a la fecha y hora de inicio de cada escala y la suma de ambos costos. Ordenar por costo total descendente.
Finalmente invocar el SP con la fecha y hora actual.

Costo guia: horas_trabajadas_tour * salario_fecha_salida_tour
horas_trabajadas_tour: diferencia en horas entre fecha_hora_salida y fecha_hora_llegada.

Costo encargado: horas_trabajadas_escala* salario_fecha_hora_ini_escala
horas_trabajadas_escala: diferencia en horas entre fecha_hora_ini y fecha_hora_fin.

****Nota:**** Si no conoce otra función o forma que le resulte familiar para los cálculos de tiempo, puede usar la función `**timestampdiff(time_unit, datetime_since, datetime_until)**` que permite calcular la diferencia de tiempo en una unidad por ejemplo:
select timestampdiff(hour,'2022-01-01',now());

Resolución 2

```sql
DROP function IF EXISTS `costo_hora_empleado`;
DROP procedure IF EXISTS `costo_tours`;

DELIMITER $$

CREATE DEFINER=`gestiondatos`@`%` FUNCTION
`costo_hora_empleado`(cuil_guia bigint, fecha_valor datetime)
RETURNS int
    READS SQL DATA
BEGIN

declare valor_hora decimal(10,3);

select s.valor into valor_hora
from salario_hora s
where s.cuil_empleado=cuil_guia
```

```
and s.fecha_desde=(
    select max(sal.fecha_desde)
    from salario_hora sal
    where sal.cuil_empleado=cuil_guia
    and fecha_desde ⩽ fecha_valor
);

RETURN valor_hora;
END$$

DELIMITER $$
USE `role_play_events`$$
CREATE PROCEDURE `costo_tours` (limite datetime)
BEGIN
select t.nro, t.tematica, t.cuil_guia, t.fecha_hora_salida,
t.fecha_hora_regreso
    , costo_hora_empleado(t.cuil_guia, t.fecha_hora_salida)
        *
timestampdiff(hour,t.fecha_hora_salida,t.fecha_hora_regreso)
costo_guia
    , sum(costo_hora_empleado(e.cuil_encargado,e.fecha_hora_ini)
        * timestampdiff(hour,e.fecha_hora_ini,e.fecha_hora_fin))
costo_encargados
    , (costo_hora_empleado(t.cuil_guia, t.fecha_hora_salida) *
timestampdiff(hour,t.fecha_hora_salida,t.fecha_hora_regreso))
        +
(sum(costo_hora_empleado(e.cuil_encargado,e.fecha_hora_ini) *
timestampdiff(hour,e.fecha_hora_ini,e.fecha_hora_fin))) costo_total
from tour t
inner join escala e
    on t.nro=e.nro_tour
where t.fecha_hora_regreso ⩽ limite
group by t.nro, t.tematica, t.cuil_guia, t.fecha_hora_salida,
t.fecha_hora_regreso
order by costo_total desc;
END$$

DELIMITER ;

call costo_tours(now());
```

Enunciado 3

****Salarios a categoría**La empresa decidió cambiar los salarios de los empleados de ser por empleado a por categoría y también deberá llevarse un registro histórico. Se debe:
a) Crear una entidad categoría, la misma debe identificarse por un código autoincremental y tener una descripción.
b) Crear una tabla salario_hora_categoria débil de categoría con una fecha desde y un valor.
c) Migrar los datos de las categorías de los empleados a la descripción de categoría y reemplazar la descripción actual con una clave foránea.
d) Cargar los salario_hora_categoria de cada categoría desde hoy, con el máximo salario actual de los empleados de dicha categoría
e) Limpiar los datos que ya no son necesarios.

Resolución 3

```sql
CREATE TABLE `categoria` (
  `codigo` int unsigned NOT NULL AUTO_INCREMENT,
  `descripcion` varchar(255) NOT NULL,
  PRIMARY KEY (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `salario_categoria_hora` (
  `codigo_categoria` int unsigned NOT NULL,
  `fecha_desde` date NOT NULL,
  `valor` decimal(10,3) NOT NULL,
  PRIMARY KEY (`codigo_categoria`,`fecha_desde`),
  CONSTRAINT `fk_salario_categoria_hora_categoria` FOREIGN KEY
(`codigo_categoria`) REFERENCES `categoria` (`codigo`) ON DELETE
RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

alter table empleado
add column codigo_categoria int unsigned null;

begin;

insert into categoria(descripcion)
select distinct categoria
from empleado;
```

```
update empleado e
inner join categoria c
    on e.categoria=c.descripcion
set e.codigo_categoria=c.codigo;

insert into salario_categoria_hora
with fec_val as (
    select sal.cuil_empleado, max(sal.fecha_desde) ult_fec
    from salario_hora sal
    where sal.fecha_desde ⩽ now()
    group by sal.cuil_empleado
)
select e.codigo_categoria, now(), max(sh.valor)
from salario_hora sh
inner join fec_val
    on sh.cuil_empleado=fec_val.cuil_empleado
    and sh.fecha_desde=fec_val.ult_fec
inner join empleado e
    on sh.cuil_empleado=e.cuil
group by e.codigo_categoria;

commit;

alter table empleado drop column categoria;
drop table salario_hora;

```