PREPARACIÓN ULTIMO EXAMEN bD

Trabajo de Alejandro Sainz Sainz

BD-ACTIVIDAD 4.4

CREACIÓN DE USUARIOS 4

DROPEAR USUARIOS 5

GARANTIZAR PRIVILEGIOS 5

CREACIÓN DE ROL 5

ASIGNACIÓN DE ROL A UN USUARIO 5

DESIGNAR UN ROL POR DEFECTO 5

ASIGNAR PRIVILEGIOS A UN ROL 6

TRIGGERS 6

TRIGGER CON VARIABLES CONSULTA 7

DISTINTOS TIPOS DE TRIGGERS 9

TRIGGER CON MENSAJES DE ERROR 10

TIMESTAMP Y MENSAJE 10

PROCEDIMIENTOS 11

PROCEDIMIENTOS SENCILLOS Y VARIABLES, LLAMADA Y OBTENCIÓN DE RESULTADO 11

PROCEDIMIENTO Y BUCLE FOR 12

FUNCIONES 14

FUNCIONES SIMPLES 14

FUNCION CON UN SELECT CASE 15

FUNCION SELECT CASE RETORNA UN ENUM CON SELECT CASE 16

CONSULTAS 17

EJERCICIOS EXAMEN TIPO A SEGUNDA EVAL 17

EJERCICIOS EXAMEN TIPO B SEGUNDA EVAL 19

TIPOS DE CONSULTAS VARIADOS 21

MIS EJERCICIOS DE CONSULTAS MI EMPRESA FICTICIA 25

ACTUALIZACIÓN DE VALORES DE UNA TABLA SEGÚN DOS CONDICIONES 25

UPDATE BASADO EN EL RESULTADO DE UNA CONSULTA 25

ACTUALIZACIÓN DE DOS TABLAS AL MISMO TIEMPO BASADAS EN EL MISMO PARÁMETRO 26

UN UPDATE BASADO EN MODIFICAR UNA FECHA CON ADDDATE 26

CONSULTA CON COUNT 26

CONSULTA Y SUCONSULTA CON UN MAX 26

CONSULTA CON UN SUM 27

CONSULTA Y ANIDADA CON COUNT 27

CONSULTA CON SUBCONSULTA WHERE NOT EXISTS 28

[1 PRIMER PROCEDIMIENTO 4](#_Toc198493528)

[2 LLAMADA AL PROCEDIMIENTO 4](#_Toc198493529)

[3 SELECT @RESULTADO 4](#_Toc198493530)

[4 CREACION DEL PROCEDIMIENTO 5](#_Toc198493531)

[5 DEFINIMOS UNA VARIABLE EXTERNA 5](#_Toc198493532)

[6 LLAMADA AL PROCEDIMIENTO 5](#_Toc198493533)

[7 RESULTADO DEL PROCEDIMIENTO 6](#_Toc198493534)

[8 RESULTADO 6](#_Toc198493535)

[9 CODIGO DE LA FUNCION 6](#_Toc198493536)

[10 EJECUCIÓN DE LA FUNCION 7](#_Toc198493537)

[11 RESULTADO DE LA FUNCIÓN 7](#_Toc198493538)

[12 CODIGO DEL PROCEDIMIENTO 8](#_Toc198493539)

[13 EJECUCIÓN DEL PROCEDIMIENTO 8](#_Toc198493540)

[14 OBTENEMOS EL RESULTADO 8](#_Toc198493541)

[15 RESULTADO 8](#_Toc198493542)

[16 CODIGO DE LA FUNCIÓN 9](#_Toc198493543)

[17 SELECT DIA\_SEMANA(3) 9](#_Toc198493544)

[18 CODIGO DE LA FUNCION 10](#_Toc198493545)

[19 FUNCIONES A EJECUTAR 10](#_Toc198493546)

[20 CODIGO DEL PROCEDIMIENTO 12](#_Toc198493547)

[21 LLAMAMOS AL PROCEDIMIENTO 13](#_Toc198493548)

[22 SELECT @VALORFACTORIAL 13](#_Toc198493549)

[23 LAS TABLAS 14](#_Toc198493550)

[24 CODIGO 15](#_Toc198493551)

[25 LLAMADA A LA FUNCION 15](#_Toc198493552)

[26 OBTENEMOS LAS TABLAS 15](#_Toc198493553)

[27 RESULTADO 16](#_Toc198493554)

# CREACIÓN DE USUARIOS

**CREATE USER** 'alumno' **IDENTIFIED BY** 'alumno';

**CREATE USER** 'decroly' **IDENTIFIED BY** 'decroly';

**CREATE USER** developer@localhost **IDENTIFIED BY** '1234';

# DROPEAR USUARIOS

**DROP USER** developer@localhost;

# GARANTIZAR PRIVILEGIOS

**GRANT ALL PRIVILEGES** **ON** ParkingAlquiler.\* **TO** developer@localhost;

También nos sirve simplemente grant all.

GRANT select,insert,update,delete ON BD.\* TO USER@localhost IDENTIFIED BY "PASSWORD";

Para asignar diferente privilegios dependiendo de las acciones que queramos realizar.

Si hay tablas específicas o views específicas insertar el nombre de la tabla o la view en lugar del asterisco.

**GRANT SELECT ON** antiguedadTrabajadores **TO** deptrrhh@localhost;

Este es un ejemplo de privilegio sobre una vista.

**GRANT SELECT ON** Resumen\_Salarios **TO** alejandro@localhost;

**GRANT SELECT** on Jefes\_Departamento **TO** alejandro@localhost;

**GRANT SELECT** on Contratados95 **TO** alejandro@localhost;

Más ejemplos de privilegios y garantías.

# GARANTIZAR PRIVILEGIOS CON GRAN OPTION

GRANT privilegios ON base/tabla TO usuario [IDENTIFIED by ´contraseña´] [WITH GRANT OPTION];

With grant option nos permite darle a este usuario la opción de otorgar privilegios el a otros usuarios.

# ELIMINACIÓN DE PRIVILEGIOS

REVOKE privilegios ON base/tabla FROM usuario

Comando básico.

REVOKE ALL ON BD.\* from USER@localhost

Ejemplo de comando.

# CREACIÓN DE ROL

**CREATE ROLE** administracion;

# ELIMINACIÓN DE UN ROL

DROP ROLE 'nombre\_del\_rol';

# ASIGNACIÓN DE ROL A UN USUARIO

**GRANT** administracion **TO** deptrrhh@localhost, deptconta@localhost;

# ELIMINACIÓN DE ROL A UN USUARIO

REVOKE ROLE 'nombre\_rol' FROM 'nombre\_usuario'@'host';

# DESIGNAR UN ROL POR DEFECTO

**SET DEFAULT ROLE** administracion **TO** deptrrhh@localhost, deptconta@localhost;

# ASIGNAR PRIVILEGIOS A UN ROL

**GRANT SELECT ON** evolucionSalarioTrabajadores **TO** administracion;

# TRIGGERS

**DELIMITER //**

**CREATE TRIGGER** cambioDeNombre

**AFTER UPDATE ON** departments

**FOR EACH ROW**

**BEGIN**

**INSERT INTO** cambioDeLook

**VALUES** (old.dept\_no, old.dept\_name, new.dept\_name, **current\_date()**);

**END//**

Ejemplo de trigger after update.

## TRIGGER CON VARIABLES CONSULTA

**DELIMITER //**

**CREATE TRIGGER** finalDelDeber

**BEFORE UPDATE ON** dept\_manager

**FOR EACH ROW**

**BEGIN**

**DECLARE** id **INT**;

**DECLARE** nombre, apellido, departamento **VARCHAR(50)**;

**SELECT**

e.emp\_no,

e.first\_name,

e.last\_name,

d.dept\_name

**INTO**

id,

nombre,

apellido,

departamento

**FROM**

employees e

**JOIN**

dept\_manager dem on e.emp\_no = dem.emp\_no

**JOIN**

dept\_emp de on e.emp\_no = de.emp\_no

**JOIN**

departments d on de.dept\_no = d.dept\_no

**WHERE**

e.emp\_no = old.emp\_no;

**IF**

new.to\_date < '9999-01-01'

**THEN**

**INSERT INTO** finDeGuardia **VALUES** (id, nombre, apellido, departamento, new.to\_date);

**END IF**;

**END//**

Ejemplo de trigger en el que creamos variables y almacenamos los resultados de una consulta en esas variables para luego usarlos haciendo otras cosas.

delimiter $$

create trigger mejorPagados

after insert on salaries

for each row

begin

declare id, sueldo int;

declare nombre, apellido varchar(50);

declare f\_nac, f\_cont date;

declare gen char(1);

select e.emp\_no, e.birth\_date, e.first\_name, e.last\_name, e.gender, e.hire\_date

into id, f\_nac, nombre, apellido, gen, f\_cont

from employees e where e.emp\_no = new.emp\_no;

if new.salary >= 100000 then

insert into millonetis values(id, nombre, apellido, f\_nac,gen, f\_cont, new.salary);

end if;

end$$

Otro trigger con variables

## DISTINTOS TIPOS DE TRIGGERS

delimiter $$

create trigger antiguedadJefes

before insert on dept\_manager

for each row

begin

declare t int;

select timestampdiff(year, e.hire\_date, current\_date()) into t

from employees e

where e.emp\_no = new.emp\_no;

if t < 10 then

signal sqlstate '50001' set message\_text = 'Para ser jefe de departamento la antiguedad mínima son 10 años';

end if;

end$$

Trigger con diferentes funciones

## TRIGGER CON MENSAJES DE ERROR

delimiter $$

create trigger fechas

before insert on salaries

for each row

begin

if new.to\_date < new.from\_date

then signal sqlstate '50001' set message\_text = 'La fecha de hasta de un salario no puede ser menor a la fecha desde';

end if;

end$$

## TIMESTAMP Y MENSAJE

delimiter $$

create trigger chaval

before insert on employees

for each row

begin

if timestampdiff(year, new.birth\_date, current\_date()) < 16

then signal sqlstate '50001' set message\_text = 'Vuelve al colegio pollo, que estás muy verde';

end if;

end$$

# PROCEDIMIENTOS

## PROCEDIMIENTOS SENCILLOS Y VARIABLES, LLAMADA Y OBTENCIÓN DE RESULTADO

delimiter //

create procedure suma\_numeros(in num1 int, in num2 int, out resultado int)

begin

set resultado = num1 + num2;

end//

call suma\_numeros(3,5,@resultado);

select @resultado;

drop procedure if exists suma\_producto;

delimiter //

create procedure suma\_producto(in num1 int, inout num2 int, out res\_suma int)

begin

set res\_suma = num1 + num2;

set num2 = num1 \* num2;

end//

set @num2 = 5;

call suma\_producto(3, @num2, @res\_suma);

select @num2,@res\_suma;

## PROCEDIMIENTO Y BUCLE FOR

delimiter //

create procedure factorial(in n int, out valorFactorial int)

begin

declare nInicial int;

set nInicial = 2;

set valorFactorial = 1;

bucle : loop

set valorFactorial=valorFactorial \* nInicial;

set nInicial = nInicial + 1;

if nInicial > n then

leave bucle;

end if;

end loop bucle;

end //

call factorial(5,@valorFactorial);

select @valorFactorial;

delimiter //

create procedure impares()

begin

declare contador int;

set contador = 0;

bucle\_impares : loop

set contador = contador + 1;

if mod(contador,2)<>0 then

insert into resultado (resultado) values (contador);

insert into resultado2 (resultado) values (contador);

end if;

if contador > 50 then

leave bucle\_impares;

end if;

end loop bucle\_impares;

end //

call impares();

select \* from resultado;

# FUNCIONES

## FUNCIONES SIMPLES

delimiter //

create function historico\_gerentes(cod\_empleado int)

returns boolean

deterministic

begin

return exists(select emp\_no from dept\_manager where emp\_no = cod\_empleado);

end //

select \* from dept\_manager;

select historico\_gerentes(10010);

Función simple que devuelve un boolean gracias a la función exists.

## FUNCION CON UN SELECT CASE

delimiter //

create function calculadora(numero1 decimal(5,2), numero2 decimal(5,2), operacion int)

returns decimal(10,2)

deterministic

begin

return case

when operacion = 1 then numero1 + numero2

when operacion = 2 then numero1 - numero2

when operacion = 3 then numero1 \* numero2

when operacion = 4 then numero1/numero2

end;

end //

select calculadora(100.99,100.99,1);

select calculadora(100.99,100.99,2);

select calculadora(100.99,100.99,3);

select calculadora(100.99,100.99,4);

## FUNCION SELECT CASE RETORNA UN ENUM CON SELECT CASE

delimiter //

create function dia\_semana(numero int)

returns enum('Lunes','Martes','Miercoles','Jueves','Viernes','Sabado','Domingo')

deterministic

begin

return case

when numero = 1 then 'Lunes'

when numero = 2 then 'Martes'

when numero = 3 then 'Miercoles'

when numero = 4 then 'Jueves'

when numero = 5 then 'Viernes'

when numero = 6 then 'Sabado'

when numero = 7 then 'Domingo'

end;

end //

# CONSULTAS

## EJERCICIOS EXAMEN TIPO A SEGUNDA EVAL

-- Seleccionar todos los empleados que han recibido un homenaje y su regalo

select e.first\_name, e.last\_name, e.hire\_date, h.regalo

from employees e

join homenaje h on e.emp\_no = h.cod\_empleado;

-- Indica el gerente de cada departamento y su salario actual

select e.first\_name, e.last\_name, d.dept\_name, s.salary

from employees e

join dept\_manager dem on e.emp\_no = dem.emp\_no

join salaries s on e.emp\_no = e.emp\_no

join dept\_emp de on de.emp\_no = e.emp\_no

join departments d on d.dept\_no = de.dept\_no

where dem.to\_date = '9999-01-01' and s.to\_date = '9999-01-01';

-- Muestra la evolución del primer empleado contratado

SELECT e.first\_name, e.last\_name, s.salary, d.dept\_name

FROM employees e

join dept\_emp de on e.emp\_no = de.emp\_no

join departments d on d.dept\_no = de.dept\_no

join salaries s on s.emp\_no = e.emp\_no

where e.hire\_date = (select min(e.hire\_date)

from employees e);

-- Salario medio de todos los empleados en cuyo título aparece la palabra senior

select avg(s.salary) as 'Salario medio Seniors'

from departments d

join dept\_emp de on d.dept\_no = de.dept\_no

join employees e on de.emp\_no = e.emp\_no

join salaries s on e.emp\_no = s.emp\_no

join titles t on e.emp\_no = t.emp\_no

where s.to\_date = '9999-01-01' and t.title like '%Senior%' and t.to\_date = '9999-01-01';

-- Se muestran los datos de todos los empleados, y los datos del regalo y del presentador en el caso de que lo hayan tenido

select e.first\_name, e.last\_name, h.presentador, h.regalo

from employees e

left join

homenaje h on e.emp\_no = h.cod\_empleado;

-- Muestra los datos de todos los empleados y los de su jefe de departamente en caso de que lo tuviesen

select e.first\_name, e.last\_name, j.first\_name, j.last\_name

from employees AS e

join dept\_emp de on e.emp\_no = de.emp\_no

left join

(select e.first\_name, e.last\_name,dem.dept\_no

from employees e

join dept\_manager dem on e.emp\_no = dem.emp\_no

where dem.to\_date = '9999-01-01') AS j

on de.dept\_no = j.dept\_no

where de.to\_date = '9999-01-01'

order by de.dept\_no;

## EJERCICIOS EXAMEN TIPO B SEGUNDA EVAL

select \* from induccion;

-- Mostrar datos de empleados que hayan pasado por la induccion, su fecha de contratación y su regalo.

select e.first\_name, e.last\_name, e.hire\_date, i.regalo

from employees e

join induccion i on e.emp\_no = i.organizador;

-- Obtener el salario medio de los empleados cuyo título actual contenga la palabra engineer

select avg(s.salary)

from salaries s

join employees e on s.emp\_no = e.emp\_no

join titles t on e.emp\_no = t.emp\_no

where t.title like '%Engineer%' and t.to\_date = '9999-01-01';

-- Obtener los datos de todos aquellos empleados que ya no son gerentes

select e.first\_name, e.last\_name, s.salary

from employees e

join salaries s on e.emp\_no = s.emp\_no

join dept\_manager dem on e.emp\_no = dem.emp\_no

where dem.to\_date < '9999-01-01' and s.to\_date = '9999-01-01';

-- Muestra el salario y la evolución del departamento del último empleado contratado

select e.first\_name, e.last\_name, s.salary, d.dept\_name

from employees e

join dept\_emp de on e.emp\_no = de.emp\_no

join departments d on d.dept\_no = de.dept\_no

join salaries s on e.emp\_no = s.emp\_no

where e.hire\_date = (select max(e.hire\_date)

from employees e);

-- Muestra el nombre y el salario actual de aquellos empleados que lleven más de dos años con el mismo salario

select e.first\_name, e.last\_name, s.salary

from employees e

join salaries s on e.emp\_no = s.emp\_no

where datediff(s.to\_date, s.from\_date) > 730 and datediff(s.to\_date, curdate()) > 730;

-- Enumera los nombres y salarios de los empleados que han tenido más de un salario

select e.first\_name, e.last\_name, s.salary

from employees e

join salaries s on e.emp\_no = s.emp\_no;

-- Implementar cambios en la base de datos

alter table induccion add facilitador int;

select \* from induccion;

alter table induccion add constraint facilitador foreign key (facilitador) references employees (emp\_no);

update induccion set facilitador =

(select e.emp\_no

from employees e

join dept\_emp de on e.emp\_no = de.emp\_no

where e.birth\_date = (select max(e.birth\_date) from employees e) and de.to\_date = '9999-01-01');

select \* from induccion;

select e.first\_name, e.last\_name, i.regalo, i.facilitador

from employees e

left join

induccion i on e.emp\_no = i.organizador;

## TIPOS DE CONSULTAS VARIADOS

#Salario medio por departamento entre los empleados que actualmente trabajan en el

select d.dept\_name, avg(s.salary)

from employees e

join salaries s on e.emp\_no = s.emp\_no

join dept\_emp de on e.emp\_no = de.emp\_no

join departments d on de.dept\_no = d.dept\_no

where de.to\_date = '9999-01-01'

group by d.dept\_no;

#Muestra el trabajador mejor pagado por detartamento de los que trabajan actualmente

select e.first\_name, e.last\_name, s.salary, d.dept\_name

from employees e

join salaries s on e.emp\_no = s.emp\_no

join dept\_emp de on e.emp\_no = de.emp\_no

join departments d on de.dept\_no = d.dept\_no

where (s.salary in (select max(s.salary) from salaries s

join dept\_emp de on s.emp\_no = de.emp\_no

group by de.dept\_no))

and de.to\_date = '9999-01-01' and s.to\_date = '9999-01-01';

#Muestra los datos de jefe de departamento actual con el salario más bajo

select e.emp\_no, e.birth\_date, e.first\_name, e.last\_name, e.gender, s.salary, d.dept\_name

from employees e

join salaries s on e.emp\_no = s.emp\_no

join dept\_manager dem on e.emp\_no = dem.emp\_no

join departments d on dem.dept\_no = d.dept\_no

where dem.to\_date = '9999-01-01' and s.to\_date = '9999-01-01'

order by s.salary asc limit 1;

#Muestra los detalles del primer empleado contratado por la empresa

select s.salary, t.title,e.emp\_no, e.first\_name, e.last\_name, e.hire\_date

from titles t

join employees e on t.emp\_no = e.emp\_no

join salaries s on e.emp\_no = s.emp\_no

where s.to\_date = '9999-01-01' and t.to\_date = '9999-01-01'

order by e.hire\_date asc limit 1;

select \* from salaries where emp\_no = '10008';

#Ejercicio adicional

#Nombre y apellidos del primer empleado del departamento de ventas

select e.first\_name, e.last\_name, de.from\_date, d.dept\_name from employees e

join dept\_emp de on e.emp\_no = de.emp\_no join departments d on de.dept\_no = d.dept\_no

where d.dept\_name = 'Sales' order by de.from\_date asc limit 1;

#Titulo de la mujer que más gana actualmente

select t.title, s.salary from titles t join employees e on t.emp\_no = e.emp\_no

join salaries s on e.emp\_no = s.emp\_no where s.to\_date = '9999-01-01' and e.gender = 'F'

order by s.salary desc limit 1;

#Fecha de contratación del jefe actual de Marketing

select d.dept\_name, de.from\_date from employees e join dept\_manager dem on

e.emp\_no = dem.emp\_no join departments d on d.dept\_no = dem.dept\_no join

dept\_emp de on d.dept\_no = de.dept\_no where d.dept\_name = 'Marketing' and

dem.to\_date = '9999-01-01';

#Otro ejercicio adicional

select avg(s.salary) as 'Salario Medio'

from salaries s

join employees e on s.emp\_no = e.emp\_no

join dept\_manager dem on e.emp\_no = dem.emp\_no

where dem.to\_date < '9999-01-01';

select t.title, count(t.emp\_no) as 'Empleados/titulo'

from titles t

join employees e on t.emp\_no = e.emp\_no

join dept\_emp de on e.emp\_no = de.emp\_no

join departments d on de.dept\_no = d.dept\_no

where d.dept\_name = 'Sales'

group by t.title

order by count(t.emp\_no) desc

limit 1;

select sum(s.salary) as 'Suma Salarios', d.dept\_name as 'Departamento'

from salaries s

join employees e on s.emp\_no = e.emp\_no

join dept\_emp de on e.emp\_no = de.emp\_no

join departments d on de.dept\_no = d.dept\_no

where de.to\_date < '9999-01-01'

group by d.dept\_no

order by sum(s.salary) asc

limit 1;

## MIS EJERCICIOS DE CONSULTAS MI EMPRESA FICTICIA

### ACTUALIZACIÓN DE VALORES DE UNA TABLA SEGÚN DOS CONDICIONES

-- Este update es para actualizar los precios según el coste de la vida

select \* from paquetes;

update paquetes p, paquetes p1 set p.precio = p.precio \* 1.1, p1.precio = p1.precio \* 1.05 where p.precio <= 600.00 and p1.precio > 600;

select \* from paquetes;

-- Este update sirve para lo mismo pero con el rango de precio de los paquetes más caros

### UPDATE BASADO EN EL RESULTADO DE UNA CONSULTA

update reservas r

set r.precio =

(select sum(listado.precio) as precio\_final from

(select r.cod\_reserva,p.precio, p.cod\_paquete

from paquetes p

join actividad a on p.cod\_paquete = a.cod\_paquete

join reservas r on a.cod\_reserva = r.cod\_reserva) as listado

where listado.cod\_reserva = 1

group by listado.cod\_reserva)

where r.cod\_reserva = 1;

### ACTUALIZACIÓN DE DOS TABLAS AL MISMO TIEMPO BASADAS EN EL MISMO PARÁMETRO

update departamento d, empleados e

set d.departamento = 'Dep. Técnico', e.especialidad = 'Tecnico Mantenimiento'

where d.departamento = 'Mantenimiento' and e.especialidad in ('Mantenimiento' , 'Guía de Senderismo');

### UN UPDATE BASADO EN MODIFICAR UNA FECHA CON ADDDATE

update reservas r set r.fecha\_fin\_reserva = adddate(r.fecha\_fin\_reserva,interval 5 day), r.precio = precio \* 1.25 where r.cod\_reserva = 4;

### CONSULTA CON COUNT

-- Contar sedes por provincia

select p.nombre, count(s.cod\_sede) as 'Sucursales'

from provincia p

join localidad l on p.cp = l.cp

join sedes s on l.cp\_completo = s.cp\_completo

group by p.nombre;

### CONSULTA Y SUCONSULTA CON UN MAX

-- Calcular que agencia ha facturado más

select a.nombre, r.cod\_reserva, r.precio

from agencias a

join reservas\_agencias re on a.cod\_agencia = re.cod\_agencia

join reservas r on r.cod\_reserva = re.cod\_reserva

where r.precio in(select max(r.precio)

from reservas r

join reservas\_agencias re on r.cod\_reserva = re.cod\_reserva);

### CONSULTA CON UN SUM

-- Calcular la suma de los paquetes que componen cada reserva

select listado.cod\_reserva,sum(listado.precio) from

(select r.cod\_reserva,p.precio, p.cod\_paquete

from paquetes p

join actividad a on p.cod\_paquete = a.cod\_paquete

join reservas r on a.cod\_reserva = r.cod\_reserva) as listado

group by listado.cod\_reserva;

### CONSULTA Y ANIDADA CON COUNT

-- Queremos conocer cuantos clientes han contratado cada paquete ordenados de mayor a menor

select listado.nombre\_paquete as 'Paquete', count(listado.nombre\_completo) as 'Clientes' from

(select p.nombre\_paquete, c.nombre\_completo from clientes c

join componentes co on c.dni = co.dni

join grupos g on co.id\_grupo = g.id\_grupo

join reservas r on r.id\_grupo = g.id\_grupo

join actividad a on r.cod\_reserva = a.cod\_reserva

join paquetes p on p.cod\_paquete = a.cod\_paquete)as listado

group by listado.nombre\_paquete

order by count(listado.nombre\_completo) desc;

### CONSULTA CON SUBCONSULTA WHERE NOT EXISTS

-- Seleccionar los paquetes que no se han venido nunca

select p.cod\_paquete, p.nombre\_paquete from paquetes p

where not exists

(select a.cod\_paquete

from actividad a

where p.cod\_paquete = a.cod\_paquete);

# VISTAS

create or replace view evolucionTitulos as

select e.first\_name, e.last\_name, t.title, t.from\_date as 'Fecha Desde', t.to\_date as 'Fecha Hasta'

from employees e

join titles t on e.emp\_no = t.emp\_no

order by e.emp\_no;

create or replace view jefesDepartamentoActual as

select e.first\_name, e.last\_name, d.dept\_name

from employees e

join dept\_manager dem on e.emp\_no = dem.emp\_no

join dept\_emp de on e.emp\_no = de.emp\_no

join departments d on de.dept\_no = d.dept\_no

where dem.to\_date = '9999-01-01';

create or replace view evolucionSalarioTrabajadores as

select e.emp\_no, e.first\_name, e.last\_name, s.salary, s.from\_date, s.to\_date

from employees e

join salaries s on e.emp\_no = s.emp\_no

order by e.emp\_no;