Preguntas Examen Final

create table students(

id int identity(1,1) not null,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

email varchar(100) not null,

mobile nvarchar(9) not null,

birthdate date not null,

country varchar(50) not null,

master\_id int not null,

adviser\_id int not null

constraint students\_pk primary key (id),

constraint students\_fk foreign key (master\_id) references masters(id),

constraint students\_fk2 foreign key (adviser\_id) references advisers(id)

);

create table masters (

id int identity(1,1) not null,

master\_name varchar(50) not null,

master\_description varchar(255) not null,

master\_version int not null,

constraint masters\_pk primary key (id)

);

create table advisers (

id int identity(1,1) not null,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

nationality varchar(50) not null,

inscription\_date date not null,

constraint advisers\_pk primary key (id),

);

create table invoices (

id int identity(1,1) not null,

amount int not null,

deadline date not null,

tax int not null,

student\_id int not null,

constraint invoices\_pk primary key (id),

constraint invoices\_fk foreign key (student\_id) references students(id)

);

create table study\_groups (

id int identity(1,1) not null,

study\_group\_name varchar(50) not null,

study\_group\_start\_date date not null,

study\_group\_end\_date date not null,

constraint study\_groups\_pk primary key (id),

);

create table students\_by\_groups (

student\_id int not null,

study\_group\_id int not null,

constraint students\_by\_groups\_pk primary key (student\_id, study\_group\_id),

constraint students\_by\_groups\_fk foreign key (student\_id) references students(id),

constraint students\_by\_groups\_fk2 foreign key (study\_group\_id) references study\_groups(id)

);

create table courses (

id int identity(1,1) not null,

course\_name varchar(50) not null,

course\_description varchar(255) not null,

course\_version int not null,

sumary\_link varchar(100) not null,

constraint courses\_pk primary key (id)

);

create table courses\_by\_master (

master\_id int not null,

course\_id int not null,

constraint courses\_by\_master\_pk primary key (master\_id, course\_id),

constraint courses\_by\_master\_fk foreign key (master\_id) references masters(id),

constraint courses\_by\_master\_fk2 foreign key (course\_id) references courses(id)

);

create table courses\_by\_study\_group (

study\_group\_id int not null,

course\_id int not null,

constraint courses\_by\_study\_group\_pk primary key (study\_group\_id, course\_id),

constraint courses\_by\_study\_group\_fk foreign key (study\_group\_id) references masters(id),

constraint courses\_by\_study\_group\_fk2 foreign key (course\_id) references courses(id)

);

create table conferences (

id int identity(1,1) not null,

seats int not null,

deadline date,

course\_id int not null,

constraint conferences\_pk primary key (id),

constraint conferences\_fk foreign key (course\_id) references courses(id),

);

create table keynotes (

id int identity(1,1) not null,

keynote\_name varchar(100) not null,

session\_link varchar(100) not null,

grade int not null,

course\_id int not null,

constraint keynotes\_pk primary key (id),

constraint keynotes\_fk foreign key (course\_id) references courses(id),

);

drop table exams

create table exams (

id int identity(1,1) not null,

exam\_name varchar(100) not null,

exam\_start\_date date not null,

exam\_end\_date date not null,

grade int not null,

course\_id int not null,

constraint exams\_pk primary key (id),

constraint exams\_fk foreign key (course\_id) references courses(id),

);

create table questions\_by\_exam (

exam\_id int not null,

question\_id int not null,

constraint questions\_by\_exam\_pk primary key (exam\_id, question\_id),

constraint questions\_by\_exam\_fk foreign key (exam\_id) references exams(id),

constraint questions\_by\_exam\_fk2 foreign key (question\_id) references questions(id)

);

create table questions (

id int identity(1,1) not null,

question\_name varchar(100) not null,

constraint question\_pk primary key (id),

);

drop table essays

create table essays (

id int identity(1,1) not null,

deadline date,

content varchar(255) not null,

course\_id int not null,

constraint essays\_pk primary key (id),

constraint essays\_fk foreign key (course\_id) references courses(id),

);

create table reviews (

id int identity(1,1) not null,

review\_date date not null,

comment varchar(255) not null,

essay\_id int not null,

adviser\_id int not null,

constraint reviews\_pk primary key (id),

constraint reviews\_fk foreign key (essay\_id) references essays(id),

constraint reviews\_fk2 foreign key (adviser\_id) references advisers(id),

);

-- Insertar datos en la tabla masters

INSERT INTO masters (master\_name, master\_description, master\_version)

VALUES ('Master1', 'Descripción del Master 1', 1),

('Master2', 'Descripción del Master 2', 1),

('Master3', 'Descripción del Master 3', 1);

-- Insertar datos en la tabla advisers

INSERT INTO advisers (first\_name, last\_name, nationality, inscription\_date)

VALUES ('Asesor1', 'Apellido1', 'Nacionalidad1', '2023-01-01'),

('Asesor2', 'Apellido2', 'Nacionalidad2', '2023-02-01'),

('Asesor3', 'Apellido3', 'Nacionalidad3', '2023-03-01');

-- Insertar datos en la tabla students

INSERT INTO students (first\_name, last\_name, email, mobile, birthdate, country, master\_id, adviser\_id)

VALUES ('Estudiante1', 'Apellido1', 'estudiante1@email.com', '123456789', '2000-01-01', 'País1', 1, 1),

('Estudiante2', 'Apellido2', 'estudiante2@email.com', '987654321', '2000-02-01', 'País2', 2, 2),

('Estudiante3', 'Apellido3', 'estudiante3@email.com', '123789456', '2000-03-01', 'País3', 3, 3);

INSERT INTO students (first\_name, last\_name, email, mobile, birthdate, country, master\_id, adviser\_id)

VALUES ('Estudiante4', 'Apellido4', 'estudiante4@email.com', '123456780', '2000-04-01', 'País4', 1, 2),

('Estudiante5', 'Apellido5', 'estudiante5@email.com', '987654320', '2000-05-01', 'País5', 1, 2),

('Estudiante6', 'Apellido6', 'estudiante6@email.com', '123789450', '2000-06-01', 'País6', 3, 3),

('Estudiante7', 'Apellido7', 'estudiante7@email.com', '123789400', '2000-07-01', 'País7', 3, 3);

-- Insertar datos en la tabla study\_groups

INSERT INTO study\_groups (study\_group\_name, study\_group\_start\_date, study\_group\_end\_date)

VALUES ('Grupo1', '2023-01-01', '2023-05-01'),

('Grupo2', '2023-02-01', '2023-06-01'),

('Grupo3', '2023-03-01', '2023-07-01');

-- Insertar datos en la tabla students\_by\_groups

INSERT INTO students\_by\_groups (student\_id, study\_group\_id)

VALUES (1, 1),

(2, 1),

(3, 2);

-- Insertar datos en la tabla courses

INSERT INTO courses (course\_name, course\_description, course\_version, sumary\_link)

VALUES ('Curso1', 'Descripción del Curso 1', 1, 'Link1'),

('Curso2', 'Descripción del Curso 2', 1, 'Link2'),

('Curso3', 'Descripción del Curso 3', 1, 'Link3');

-- Insertar datos en la tabla courses\_by\_master

INSERT INTO courses\_by\_master (master\_id, course\_id)

VALUES (1, 1),

(2, 2),

(3, 3);

-- Insertar datos en la tabla courses\_by\_study\_group

INSERT INTO courses\_by\_study\_group (study\_group\_id, course\_id)

VALUES (1, 1),

(1, 2),

(2, 3);

-- Insertar datos en la tabla conferences

INSERT INTO conferences (seats, deadline, course\_id)

VALUES (100, '2023-04-01', 1),

(150, '2023-05-01', 2),

(200, '2023-06-01', 3);

INSERT INTO conferences (seats, deadline, course\_id)

VALUES (20, '2023-08-01', 3);

INSERT INTO conferences (seats, deadline, course\_id)

VALUES (20, null, 3);

-- Insertar datos en la tabla invoices

INSERT INTO invoices (amount, deadline, tax, student\_id)

VALUES (1000, '2023-04-01', 50, 1),

(1200, '2023-05-01', 60, 2),

(800, '2023-06-01', 40, 3);

-- Insertar datos en la tabla keynotes

INSERT INTO keynotes (keynote\_name, session\_link, grade, course\_id)

VALUES ('Keynote1', 'Link1', 90, 1),

('Keynote2', 'Link2', 85, 2),

('Keynote3', 'Link3', 88, 3);

-- Insertar datos en la tabla exams

INSERT INTO exams (exam\_name, exam\_start\_date, exam\_end\_date, grade, course\_id)

VALUES ('Examen1', '2023-04-01', '2023-04-15', 92, 1),

('Examen2', '2023-05-01', '2023-05-15', 87, 2),

('Examen3', '2023-06-01', '2023-06-15', 90, 3);

-- Insertar datos en la tabla questions

INSERT INTO questions (question\_name)

VALUES ('Pregunta1'),

('Pregunta2'),

('Pregunta3');

-- Insertar datos en la tabla questions\_by\_exam

INSERT INTO questions\_by\_exam (exam\_id, question\_id)

VALUES (1, 1),

(1, 2),

(2, 2),

(3, 3);

-- Insertar datos en la tabla essays

INSERT INTO essays (deadline, content, course\_id)

VALUES ('2023-04-01', 'Contenido del Ensayo 1', 1),

('2023-05-01', 'Contenido del Ensayo 2', 2),

('2023-06-01', 'Contenido del Ensayo 3', 3);

INSERT INTO essays (deadline, content, course\_id)

VALUES (null, 'Contenido del Ensayo 4', 3),

(null, 'Contenido del Ensayo 5', 2),

(null, 'Contenido del Ensayo 6', 3);

-- Insertar datos en la tabla reviews

INSERT INTO reviews (review\_date, comment, essay\_id, adviser\_id)

VALUES ('2023-04-15', 'Comentario del Asesor 1', 1, 1),

('2023-05-15', 'Comentario del Asesor 2', 2, 2),

('2023-06-15', 'Comentario del Asesor 3', 3, 3);

go

--1. Crear un procedimiento almacenado o funcion que retorne los nombres

--de los asesores con la mayor cantidad de alumnos asignados

alter procedure USPEjercicio1

as

begin

declare @MaxTotal int

select @MaxTotal = max(Total)

from (

select COUNT(\*) as Total

from students as S

join advisers as A on A.id = S.adviser\_id

group by A.first\_name ) as Total

select count(S.adviser\_id) as Total, A.first\_name

from students as S

join advisers as A on A.id = S.adviser\_id

group by A.first\_name

having count(S.adviser\_id) = @MaxTotal

end

go

exec USPEjercicio1

go

create view VEjercicio1

as

select count(S.adviser\_id) as Total, A.first\_name

from students as S

join advisers as A on A.id = S.adviser\_id

group by A.first\_name

select \* from VEjercicio1

create function FEjercicio1()

returns table

return

select \* from VEjercicio1

where Total = (select max(Total) from VEjercicio1)

select \* from dbo.FEjercicio1()

--2. Crear un procedimiento almacenado o funcion que retorne los nombres

-- y versiones de las maestrias con la mayor cantidad de alumnos.

create procedure USPEjercicio2

as

begin

declare @MaxTotal int

select @MaxTotal = max(Total)

from ( select count(\*) as Total, master\_name, master\_version

from masters as M

join students as S on S.master\_id = M.id

group by master\_name, master\_version) as Total

select count(master\_id) as Total, master\_name, master\_version

from masters as M

join students as S on S.master\_id = M.id

group by master\_name, master\_version

having count(master\_id) = @MaxTotal

end

go

exec USPEjercicio2

go

create view VEjercicio2

as

select count(master\_id) as Total, master\_name, master\_version

from masters as M

join students as S on M.id = S.master\_id

group by master\_name, master\_version

select \* from VEjercicio2

create function FEjercicio2()

returns table

return

select \*

from VEjercicio2

where Total = (select max(total) from VEjercicio2)

select \* from dbo.FEjercicio2()

--3. Crear un procedimiento almacenado o funcion que retorne los nombres

--completos de los estudiantes que no estan en ningun grupo de estudio.

alter function FEjercicio3 ()

returns table

return

select concat(first\_name, ' ', last\_name) as Alumno, study\_group\_id, student\_id

from students as S

left join students\_by\_groups as SG on SG.student\_id = S.id

where student\_id is null;

select \* from dbo.FEjercicio3()

--4. Crear un procedimiento almacenado o funcion que retorne los nombres

--de los cursos con la mayor cantidad de conferencias.

create view VEjercicio4

as

select course\_name, count(course\_id) as Total

from conferences as CF

join courses as CR on CR.id = CF.course\_id

group by course\_name

go

select \* from VEjercicio4

create function FEjercicio4()

returns table

return

select \*

from VEjercicio4

where Total = (select max(Total)

from VEjercicio4)

select \* from dbo.FEjercicio4()

--5. Crear un procedimiento almacenado o funcion que retorne la cantidad

--consolidada de actividades (examenes, ensayos y presentaciones) para cada curso.

select count(course\_id) as Total, course\_name

from courses as C

join conferences as CF on CF.course\_id = C.id

join exams as E on E.course\_id = C.id

join essays as ES on ES.course\_id = C.id

where deadline is not null and exam\_end\_date is not null

group by course\_name

/\*Lista:

- tipo\_plan: object (nombre: string, precio: decimal)

- canciones: object (nombre: string, artista: string, duracion: timestamp, album: object, listas\_reproduccion: array)

- lista\_reproduccion: object (nombre: string, usuario: string, fecha\_creacion: date, relacion\_canciones: array)

- relacion\_canciones: object(nombre:string, fecha\_agregacion: date)

- artistas: object (nombre: string, descripcion\_corta: string, descripcion\_detallada: string, albumes: array)

- albumes: object (artista: string minItems:1 maxItems:1, nombre: string, genero: string, fecha: date, relacion\_canciones: array)

- usuarios: object (nombre: string, fecha\_refistro: date, plan: string, relacion\_listas: array)

\*/

db.createCollection('lista',{

validator: {

$jsonSchema:{

bsonType: 'object',

required: ['nombre', 'usuario', 'fecha\_creacion', 'relacion\_canciones'],

properties:{

nombre: {

bsonType: 'string'

},

usuario: {

bsonType: 'string'

},

fecha\_creacion: {

bsonType: 'timestamp'

},

relacion\_canciones: {

bsonType: 'array',

required: ['nombre', 'arista', 'duracion', 'album'],

properties: {

nombre: {

bsonType: 'string'

},

artista: {

bsonType: 'string'

},

duracion: {

bsonType: 'timestamp',

minimum : 0

},

album: {

bsonType: 'string'

},

fecha\_agregado: {

bsonType : 'date'

}

}

}

}

}

}

})

use ('si35')

db.lista.insertOne(

{

nombre: "Jorge Mayta",

usuario: "jmayt",

fecha\_creacion: "2015-03-23",

relacion\_canciones: {

nombre: "Base de batos",

artista: "Luis",

duracion: 3,

fecha\_agregado: "2023-11-29"

}

}

)

use ('si35')

db.lista.find()

//Patrones de modelado: Embedded document pattern tanto para relacion de uno a uno como para relacion de uno a muchos porque? porque el json schema permite la incrustacion de campos

//conformados por otros campos, por ejemplo en relacion\_canciones es un arreglo pero que a la vez tiene

//como propiedades obligatorias otros campos como nombre de la cancion y la fecha de agregacion. Y cuando hay mas canciones, permite

//agregar mas canciones formando el arreglo

use ('si35')

db.lista.find()

use ('si35')

db.sales.findOne({'customer.email': "cauho@witwuta.sv"})

/\*1. Escriba una consulta que permita mostrar la cantidad de ventas

realizadas en cada ciudad. Considerar solo

aquellas ventas en las cuales se haya utilizado un cupon de descuento\*/

use('si35')

db.sales.find()

use ('si35')

db.sales.aggregate([

{

$match: {

couponUsed: true

}

},

{

$group: {

\_id: '$storeLocation',

'number': {$count: {}}

}

}

])

/\*2. Escribir una consulta que permita mostrar la cantidad

de ventas realizadas por cada metodo de compra.

Considerar solo aquellas ventas en las cuales la satisfaccion

del cliente haya sido mayor o igual a 4\*/

use ('si35')

db.sales.aggregate([

{

$match: {

'customer.satisfaction': {

$gt: 3

}

}

},

{

$group:{

\_id: '$purchaseMethod',

'number': {$count:{}}

}

}

])

use ('si35')

db.sales.count({

'customer.satisfaction':{

$gt: 3

}

})

use ('si35')

db.sales.find({

'customer.satisfaction':{

$gt: 3

}

}).count()