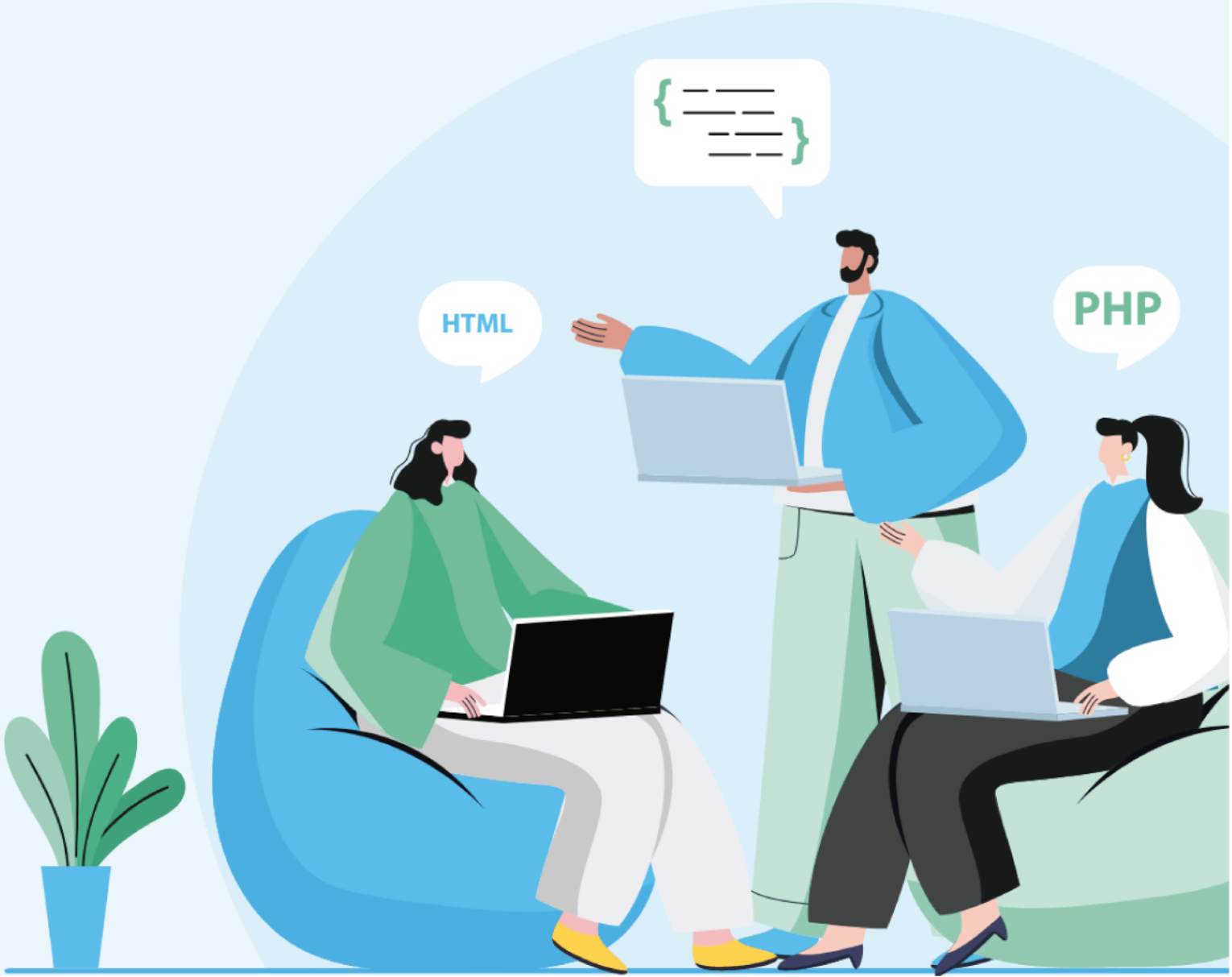
alkemy

SQL II



Introducción

Bienvenido al módulo de SQL II. Durante el desarrollo de este módulo, abordaremos características más avanzadas que en el módulo SQL I. Para ello, buscaremos hacer foco en conceptos claves como “funciones agregadas”, “agrupación de datos” e “índices”. Queremos que, luego de este recorrido, puedas hacer consultas de mayor dificultad.

Tema 1: Funciones agregadas

COUNT

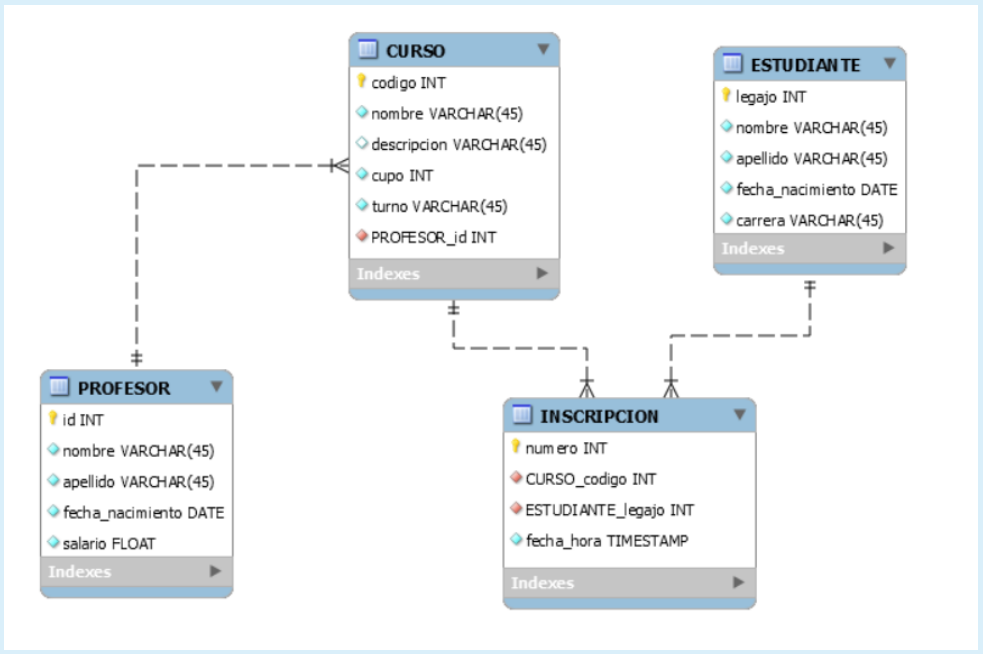
La función de agregación **COUNT** es útil para contar la cantidad de registros que devuelve una consulta a una tabla de la base de datos.

Sintaxis:

SELECT **COUNT**(*column_name*)
FROM *table*
WHERE *condiciones*;
Otra posibilidad es reemplazar el nombre de la columna, por el selector *.
SELECT **COUNT**(*)
FROM *table*
WHERE *condiciones*;

En la siguiente figura, podemos ver un ejemplo de modelo relacional:

Figura 1: Modelo relacional



Fuente: [Imagen sin título sobre modelo relacional]. (s.f.).

A continuación, se muestran algunos registros de cada tabla, donde podemos ver las relaciones de los datos:

Tabla 1: Registro ESTUDIANTE

legajo	nombre	apellido	fecha_nacimiento	carrera
36485	Romina	Nieva	1999-11-26	Mecánica
39685	Brenda	Medrano	2000-09-25	Sistemas
41258	Ramiro	Ríos	1994-12-06	Sistemas
43651	Cristian	Gómez	1995-03-19	Mecánica
47521	María	Velazq...	1998-01-02	Sistemas
47961	Alexis	Reinoso	1994-12-17	Sistemas
48952	Gabriel	Morales	1996-10-03	Sistemas
51200	Lourdes	Martinez	2001-12-13	Sistemas

Fuente: elaboración propia.

Tabla 2: Registro CURSO

codigo	nombre	descripcion	cupo	turno	PROFESOR_id
101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
102	Matemática Discreta	NULL	20	Tarde	2
103	Programación Java	POO en Java	35	Noche	4
104	Programación Web	NULL	35	Noche	5
105	Programación C#	.NET, Visual Studio 2019	30	Noche	6

Fuente: elaboración propia.

Tabla 3: Registro INSCRIPCION

	numero	CURSO_codigo	ESTUDIANTE_legajo	fecha_hora
▶	1	101	41258	2012-05-02 18:45:00
	2	102	41258	2012-04-02 18:45:00
	3	102	47961	2012-01-02 20:01:00
	4	103	47961	2012-04-28 18:45:00
	5	101	39685	2012-04-12 18:45:00
	6	103	36485	2012-04-28 18:45:00
	7	103	43651	2012-04-28 18:45:00
	8	101	47961	2012-04-28 18:45:00
	11	101	43651	2012-04-21 18:45:00
	13	102	47521	2012-04-03 18:45:00
	14	102	51200	2012-05-02 18:45:00

Fuente: elaboración propia.

Para contar la cantidad de estudiantes inscriptos al curso Algoritmos, sabiendo que su código es 101, podemos ejecutar lo siguiente:

```
SELECT COUNT(ESTUDIANTE_legajo)
FROM INSCRIPCION
where CURSO_codigo = 101;
```

Resultado:

Tabla 4. Resultado

	COUNT(ESTUDIANTE_legajo)
▶	4

Fuente: elaboración propia.

Sin embargo, puede ocurrir que no sepamos el código del curso Algoritmos. En ese caso, es posible resolver esta situación utilizando INNER JOIN:

```
Select count(ESTUDIANTE_legajo)
FROM INSCRIPCION I INNER JOIN CURSO C ON I.CURSO_codigo
= C.codigo
WHERE C.nombre = 'Algoritmos';
```

Resultado:

Tabla 5. Resultado

	count(ESTUDIANTE_legajo)
▶	4

Fuente: elaboración propia.

Lo que hará esa consulta será lo siguiente:

JOIN entre las tablas:

Tabla 6: Registro JOIN entre tablas

	numero	CURSO_codigo	ESTUDIANTE_legajo	fecha_hora	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	1	101	41258	2012-05-02 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	5	101	39685	2012-04-12 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	8	101	47961	2012-04-28 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	11	101	43651	2012-04-21 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	2	102	41258	2012-04-02 18:45:00	102	Matemática Discreta	NULL	20	Tarde	2
	3	102	47961	2012-01-02 20:01:00	102	Matemática Discreta	NULL	20	Tarde	2
	13	102	47521	2012-04-03 18:45:00	102	Matemática Discreta	NULL	20	Tarde	2
	14	102	51200	2012-05-02 18:45:00	102	Matemática Discreta	NULL	20	Tarde	2
	4	103	47961	2012-04-28 18:45:00	103	Programación Java	POO en Java	35	Noche	1
	6	103	36485	2012-04-28 18:45:00	103	Programación Java	POO en Java	35	Noche	1
	7	103	43651	2012-04-28 18:45:00	103	Programación Java	POO en Java	35	Noche	1

Fuente: elaboración propia.

Filtrar por nombre del curso igual a “Algoritmos”:

Tabla 7: Filtro del curso Algoritmos

	numero	CURSO_codigo	ESTUDIANTE_legajo	fecha_hora	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	1	101	41258	2012-05-02 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	5	101	39685	2012-04-12 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	8	101	47961	2012-04-28 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	11	101	43651	2012-04-21 18:45:00	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1

Fuente: elaboración propia.

Finalmente, contará los registros del resultado.

SUM

Con la función de agregación **SUM,** es posible sumar los valores de una columna numérica.

Sintaxis:

SELECT **SUM**(*column_name*)
FROM *table*
WHERE *condiciones*;

Tomemos como ejemplo la tabla “Profesor”:

Tabla 8: Tabla PROFESOR

	id	nombre	apellido	fecha_nacimiento	salario
▶	1	Juan	Pérez	1990-06-06	55000
	2	María Emilia	Paz	1984-07-15	72000
	3	Martín	Correa	1987-12-07	63000
	4	Lucía	Díaz	1991-02-24	45000
	5	Raúl	Martínez	1980-10-15	85000
	6	Mabel	Ríos	1982-06-12	83000

Fuente: elaboración propia.

Para conocer la cantidad de dinero que se destina a los salarios, se debería ejecutar lo siguiente:

SELECT SUM(Salario)
FROM PROFESOR;

Esta consulta devolverá un valor numérico que será el resultado de la suma de todos los valores del campo “Salario” para cada uno de los empleados.

Tabla 9. Salario

	SUM(Salario)
▶	403000

Fuente: elaboración propia.

MAX/MIN

Para saber el máximo o mínimo valor de una columna determinada en el resultado de una consulta, las funciones de agregación **MAX** y **MIN** son las indicadas.

Sintaxis con función MAX:

SELECT **MAX**(*column_name*)
FROM *table*
WHERE *condiciones*;

Sintaxis con función MIN:

SELECT **MIN**(*column_name*)
FROM *table*
WHERE *condiciones*;

Siguiendo el ejemplo de la tabla “Profesor” del apartado anterior, supongamos que se necesita saber el mínimo y el máximo salario de los profesores. Para eso, es posible ejecutar lo siguiente:

SELECT MAX(Salario) as “Máximo Salario”, MIN(Salario) as “Mínimo Salario”
FROM PROFESOR;

Resultado:

Tabla 10. Resultado

	Máximo Salario	Mínimo Salario
▶	85000	45000

Fuente: elaboración propia.

AVERAGE

La palabra average significa “promedio” en español. Por lo tanto, la función de agregación **AVG** en SQL sirve para calcular el promedio de los valores de una columna del resultado de una consulta.

Sintaxis:

SELECT AVG(columna)
FROM table
WHERE condiciones;

Se puede usar la siguiente consulta para calcular el promedio de salarios de los profesores:

SELECT AVG(Salario)
FROM PROFESOR;

Resultado:

Tabla 11. Resultado

	avg(Salario)
▶	67166.66666666667

Fuente: elaboración propia.

ROUND

Con la función de agregación ROUND(), es posible redondear un valor con una cantidad determinada de decimales.

Sintaxis:

SELECT ROUND(numero_decimal o campo_decimal, cantidad_decimales)
FROM tabla;

En el ejemplo anterior, se puede redondear el resultado usando la función ROUND:

SELECT ROUND(AVG(Salario),2)
FROM PROFESOR;

Resultado:

Tabla 12. Resultado

	ROUND(AVG(Salario),2)
▶	67166.67

Fuente: elaboración propia.

GROUP BY I

La sentencia GROUP BY sirve para agrupar registros que tengan el mismo valor en las columnas de agrupación indicadas. Suele

utilizarse para aplicar las funciones de agregación como MAX, MIN, COUNT, entre otras, agrupando registros. Veamos dos opciones de sintaxis:

Sintaxis 1:

SELECT *funcion(nombre_columna)*
FROM *tabla*
GROUP BY *columna_agrupacion*

Sintaxis 2:

SELECT *columna_agrupacion* , *funcion(nombre_columna)*
FROM *tabla*
GROUP BY *columna_agrupacion*

GROUP BY puede incluir una o varias columnas de agrupación.

Ejemplo de consulta con GROUP BY con una columna de agrupación a partir de la tabla ESTUDIANTE:

Tabla 13: Tabla ESTUDIANTE

	legajo	nombre	apellido	fecha_nacimiento	carrera
▶	36485	Romina	Nieva	1999-11-26	Mecánica
	39685	Brenda	Medrano	2000-09-25	Sistemas
	41258	Ramiro	Ríos	1994-12-06	Sistemas
	43651	Cristian	Gómez	1995-03-19	Mecánica
	47521	María	Velazquez	1998-01-02	Sistemas
	47961	Alexis	Reinoso	1994-12-17	Sistemas
	48952	Gabriel	Morales	1996-10-03	Sistemas
	51200	Lourdes	Martinez	2001-12-13	Sistemas

Fuente: elaboración propia.

Si queremos saber la cantidad de estudiantes por carrera, la sintaxis sería la siguiente:

SELECT carrera, COUNT(*)
FROM estudiante
GROUP BY carrera;

Esta consulta agrupará los estudiantes de la misma carrera y contará cuántos son.

Resultado:

Tabla 14. Resultado

	carrera	COUNT(*)
▶	Mecánica	2
	Sistemas	6

Fuente: elaboración propia.

Un detalle importante a tener en cuenta es que solo las columnas incluidas en el GROUP BY pueden incluirse en la selección de columnas para mostrar.

GROUP BY II

El GROUP BY también puede aplicarse en consultas donde hay UNA o más tablas relacionadas.

Por ejemplo, dadas las dos tablas CURSO e INSCRIPCIÓN, obtenemos lo siguiente:

Tabla 15: Tablas CURSO e INSCRIPCIÓN

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	102	Matemática Discreta	NULL	20	Tarde	2
	103	Programación Java	POO en Java	35	Noche	4
	104	Programación Web	NULL	35	Noche	5
	105	Programación C#	.NET, Visual Studio 2019	30	Noche	6

	numero	CURSO_codigo	ESTUDIANTE_legajo	fecha_hora
▶	1	101	41258	2012-05-02 18:45:00
	2	102	41258	2012-04-02 18:45:00
	3	102	47961	2012-01-02 20:01:00
	4	103	47961	2012-04-28 18:45:00
	5	101	39685	2012-04-12 18:45:00
	6	103	36485	2012-04-28 18:45:00
	7	103	43651	2012-04-28 18:45:00
	8	101	47961	2012-04-28 18:45:00
	11	101	43651	2012-04-21 18:45:00
	13	102	47521	2012-04-03 18:45:00
	14	102	51200	2012-05-02 18:45:00

Fuente: elaboración propia.

Si queremos saber la cantidad de estudiantes de cada profesor, podemos ejecutar la siguiente sintaxis:

```
SELECT p.id, count(*) as 'Cantidad de estudiantes'
from PROFESOR p inner join CURSO c on p.id = c.PROFESOR_id
                inner join INSCRIPCION I on c.codigo =
I.CURSO_codigo
group by p.id;
```

Resultado:

Tabla 16. Resultado

	id	Cantidad de estudiantes
▶	1	4
	2	4
	4	3

Fuente: elaboración propia.

HAVING

La cláusula HAVING es opcional y se utiliza con GROUP BY. Su función es agregar condiciones o filtros para que el resultado dé las funciones de agregación utilizadas. En términos simples, el HAVING es al GROUP BY lo que el WHERE es al SELECT.

Tomemos como ejemplo la tabla CURSO:

Tabla 17: Tabla CURSO

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
	102	Matemática Discreta	NULL	20	Tarde	2
	103	Programación Java	POO en Java	35	Noche	4
	104	Programación Web	NULL	35	Noche	5
	105	Programación C#	.NET, Visual Studio 2019	30	Noche	6

Fuente: elaboración propia.

Si queremos saber qué cantidad de cursos se dictan en cada turno, pero, a su vez, queremos filtrar los turnos con más de dos cursos, podemos utilizar la siguiente consulta:

```
SELECT turno, COUNT(*)
FROM alkemy.curso
GROUP BY turno HAVING COUNT(*) > 2;
```

Resultado:

Tabla 18. Resultado

	turno	COUNT(*)
▶	Noche	3

Fuente: elaboración propia.

Documentación

Funciones de agregación MAX y MIN:

Fuente: W 3 Schools (s. f.). *SQL MIN() and MAX() Functions.* Recuperado de https://www.w3schools.com/sql/sql_min_max.asp

Funciones de agregación COUNT, AVG, y SUM:

Fuente: W 3 Schools (s. f. a). *SQL COUNT(), AVG() and SUM() Functions.* Recuperado de https://www.w3schools.com/sql/sql_count_avg_sum.asp

GROUP BY y HAVING:

Fuente: W 3 Schools (s. f. b). SQL GROUP BY Statement. Recuperado de https://www.w3schools.com/sql/sql_groupby.asp

Fuente: W 3 Schools (s. f. c). SQL HAVING Clause. Recuperado de https://www.w3schools.com/sql/sql_having.asp

Videos

Funciones de agregación:

Fuente: **deividcoptero Programación** [deividcoptero Programación], (27 de enero de 2014). *Tutoriales SQL Server #15 Funciones de Agregado* [YouTube]. Recuperado de https://www.youtube.com/watch?v=iAcv1jxEuGs

GROUP BY:

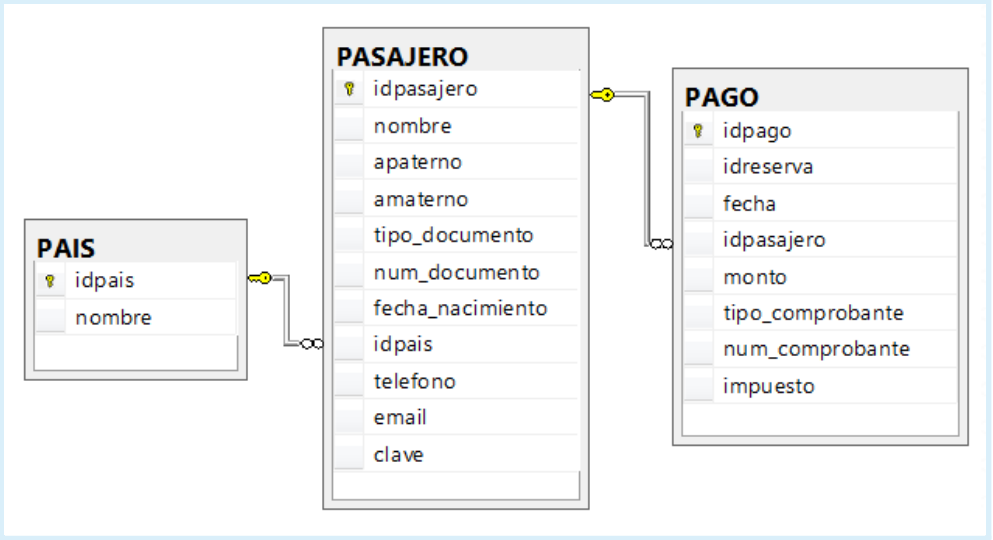
Fuente: **Tecnología Binaria** [Tecnología Binaria;]. (03 de agosto de 2016). *Cláusula GROUP BY | Curso SQL Server - #58* [YouTube]. Recuperado de https://www.youtube.com/watch?v=renHFr2jeqk

Ejercitación 1

Objetivo

El objetivo de este ejercicio es poder hacer consultas que obtengan datos en forma agregada.

Figura 2: Modelo relacional



Fuente: [imagen sin título sobre modelo relacional], s. f., https://bit.ly/3kFSUS4

Para este modelo, escribe las siguientes consultas:

1) Cantidad de pasajeros por país
2) Suma de todos los pagos realizados
3) Suma de todos los pagos que realizó un pasajero (el monto debe aparecer con dos decimales)
4) Promedio de los pagos que realizó un pasajero

Formato de entrega: la actividad se entrega a través de una URL correspondiente al repositorio sobre el que se haya trabajado.