

REPORTE DE PRÁCTICA NO. 1.3

**NOMBRE DE LA PRÁCTICA: Álgebra relacional y SQL
(1)**

ALUMNO: Axel Damian Ortiz Simon

Profesor: Dr. Eduardo Cornejo-Velázquez



1. Introducción

En esta práctica, nos enfocaremos en resolver ejercicios mediante consultas utilizando álgebra relacional. El objetivo es aplicar las operaciones fundamentales de este álgebra, como la selección, proyección, inserción y borrado, para extraer y modificar datos de tablas de forma efectiva. Al practicar con estos ejercicios, se busca fortalecer la comprensión de cómo las consultas se estructuran y se ejecutan en un sistema de base de datos relacional, preparándonos para su aplicación en escenarios reales.

El modelado de bases de datos relacionales requiere un enfoque sistemático y metodológico para garantizar una estructura eficiente y organizada. Este proceso involucra la identificación de relaciones entre los datos y la definición de operaciones sobre ellos, utilizando herramientas como el álgebra relacional. A través de un conjunto de operaciones formales, como la selección, proyección y unión, el álgebra relacional permite manipular y consultar datos de manera precisa. Un diseño bien estructurado asegura bases de datos escalables, seguras y fáciles de mantener, fundamentales en aplicaciones tecnológicas y empresariales.

SQL es el lenguaje estándar utilizado para interactuar con bases de datos relacionales, y es compatible con diversos sistemas de gestión de bases de datos. Uno de los sistemas más populares y utilizados es MySQL, una implementación de código abierto que se basa en el modelo relacional y soporta el uso de SQL para realizar consultas, actualizaciones e interacciones con las bases de datos. A lo largo de esta práctica, se explorarán y aplicarán conceptos clave de álgebra relacional, SQL y MySQL, para resolver ejercicios prácticos de consultas, inserciones y modificaciones de bases de datos, utilizando operaciones tanto teóricas como prácticas.

2. Marco teórico

El álgebra relacional es un conjunto de operaciones matemáticas utilizadas para manipular y consultar datos en bases de datos relacionales. Se trata de un lenguaje formal que proporciona una base teórica para realizar consultas de manera eficiente y estructurada, independientemente del sistema de gestión de bases de datos. Aunque SQL es el lenguaje más utilizado para la interacción con bases de datos, se basa en los principios del álgebra relacional.

Operaciones del Álgebra Relacional

El álgebra relacional incluye varias operaciones que permiten manipular y obtener información de las relaciones (tablas) en una base de datos[1]. Estas operaciones se dividen en dos categorías:

2.1.1 Operaciones Binarias

Las operaciones binarias actúan sobre dos relaciones diferentes. Las más comunes son:

- **Unión** : Combina dos relaciones eliminando los duplicados.
- **Diferencia de conjuntos** : Extrae los registros de una relación que no están presentes en otra.
- **Producto cartesiano** : Combina todas las filas de dos relaciones, generando todas las combinaciones posibles.

2.1.2 Operaciones Unarias

Las operaciones unarias actúan sobre una sola relación y permiten realizar operaciones como:

- **Selección** : Filtra los registros de una relación que cumplen con una condición dada.
- **Proyección** : Extrae solo las columnas relevantes de una relación.
- **Renombramiento** : Cambia el nombre de las columnas de una relación.

Importancia del Álgebra Relacional

El álgebra relacional es crucial para el diseño y gestión de bases de datos, ya que permite realizar consultas de manera clara y eficiente, asegurando la integridad de los datos. Proporciona los fundamentos sobre los cuales

se desarrollan lenguajes como SQL, facilitando la manipulación de grandes volúmenes de datos. El conocimiento del álgebra relacional es esencial para optimizar consultas y garantizar la coherencia de los datos en los sistemas de bases de datos [3].

SQL (Structured Query Language)

El **SQL** es un lenguaje estándar utilizado para gestionar y manipular bases de datos relacionales. A diferencia del álgebra relacional, que es un lenguaje procedimental, SQL es un lenguaje declarativo, lo que significa que el usuario especifica lo que quiere obtener sin tener que detallar cómo hacerlo. SQL permite realizar una gran variedad de operaciones, como la creación, modificación y eliminación de tablas, la inserción y eliminación de registros, y la recuperación de datos mediante consultas[4]. Las consultas SQL pueden incluir operaciones como:

- **SELECT**: Recupera datos de una o más tablas según criterios específicos.
- **INSERT**: Inserta nuevos registros en una tabla.
- **UPDATE**: Modifica los registros existentes en una tabla.
- **DELETE**: Elimina registros de una tabla.

MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto, que implementa el modelo relacional y es compatible con el uso de SQL para la gestión de datos [5]. MySQL es ampliamente utilizado en aplicaciones web y en el desarrollo de aplicaciones de software debido a su fiabilidad, facilidad de uso y alta compatibilidad con diversos lenguajes de programación[6]. A través de MySQL, los usuarios pueden realizar las siguientes operaciones:

- Crear bases de datos y tablas.
- Establecer relaciones entre tablas.
- Ejecutar consultas complejas utilizando SQL.
- Gestionar transacciones, seguridad y optimización de consultas.

3. Herramientas empleadas

Describir qué herramientas se han utilizado...

1. DataGrip: Es un entorno de desarrollo integrado (IDE) para bases de datos desarrollado por JetBrains. Permite conectarse a múltiples sistemas de gestión de bases de datos (DBMS), como MySQL, PostgreSQL, SQL Server, entre otros. Ofrece características avanzadas como autocompletado de código SQL, análisis de consultas y administración eficiente de bases de datos.
2. MySQL: Es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto, ampliamente utilizado en aplicaciones web y empresariales. Proporciona un entorno robusto para almacenar, gestionar y recuperar datos de manera eficiente.
3. SQL (Structured Query Language): Es un lenguaje estándar para la manipulación y gestión de bases de datos relacionales. Permite realizar operaciones como la creación de tablas, inserción y eliminación de datos, consultas, actualizaciones y administración de la estructura de la base de datos.

4. Desarrollo

Ejercicios Practicos

El conjunto de ejercicios está basado en las tablas Employee y Reward. Se pide que para cada ejercicio se incluya una captura de pantalla con la sentencia SQL de solución y del resultado de su ejecución.

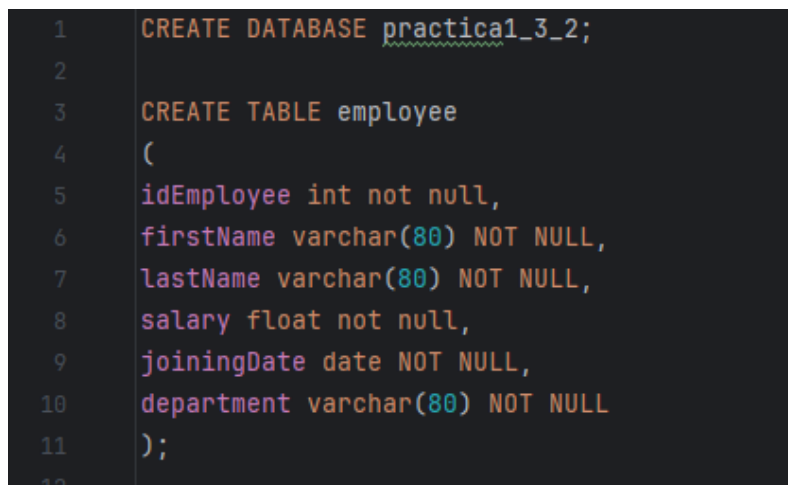
EJERCICIOS

Presentar las sentencias para crear la base de datos y tablas. Además incluir las sentencias para insertar registros.

Sentencias SQL con Capturas.

1. Escribe la sintaxis para crear la tabla “Employee”.

```
CREATE TABLE employee
(
  idEmployee int not null,
  firstName varchar(80) NOT NULL,
  lastName varchar(80) NOT NULL,
  salary float not null,
  joiningDate date NOT NULL,
  department varchar(80) NOT NULL
);
```



```
1 CREATE DATABASE practica1_3_2;
2
3 CREATE TABLE employee
4 (
5   idEmployee int not null,
6   firstName varchar(80) NOT NULL,
7   lastName varchar(80) NOT NULL,
8   salary float not null,
9   joiningDate date NOT NULL,
10  department varchar(80) NOT NULL
11 );
12
```

Figure 1: Crear Tabla Employee.

2. Escribe la sintaxis para insertar 7 registros (de la imagen) a la tabla “Employee”.

INSERT INTO employee (idEmployee, firstName, lastName, salary, joiningDate, department) VALUES

(1, 'Bob', 'Kinto', 1000000, '2019-01-20', 'Finance'),
(2, 'Jerry', 'Kansxo', 6000000, '2019-01-15', 'IT'),
(3, 'Philip', 'Jose', 89000000, '2019-02-05', 'Banking'),
(4, 'John', 'Abraham', 2000000, '2019-02-25', 'Insurance'),
(5, 'Michael', 'Mathew', 22000000, '2019-02-28', 'Finance'),
(6, 'Alex', 'Chreketo', 4000000, '2019-05-10', 'IT'),
(7, 'Yohan', 'Soso', 1230000, '2019-06-20', 'Banking');

```
19
20 INSERT INTO employee (idEmployee, firstName, lastName, salary, joiningDate, department) VALUES
21 ( idEmployee 1, firstName 'Bob', lastName 'Kinto', salary 1000000, joiningDate '2019-01-20', department 'Finance'),
22 ( idEmployee 2, firstName 'Jerry', lastName 'Kansxo', salary 6000000, joiningDate '2019-01-15', department 'IT'),
23 ( idEmployee 3, firstName 'Philip', lastName 'Jose', salary 89000000, joiningDate '2019-02-05', department 'Banking'),
24 ( idEmployee 4, firstName 'John', lastName 'Abraham', salary 2000000, joiningDate '2019-02-25', department 'Insurance'),
25 ( idEmployee 5, firstName 'Michael', lastName 'Mathew', salary 22000000, joiningDate '2019-02-28', department 'Finance'),
26 ( idEmployee 6, firstName 'Alex', lastName 'Chreketo', salary 4000000, joiningDate '2019-05-10', department 'IT'),
27 ( idEmployee 7, firstName 'Yohan', lastName 'Soso', salary 1230000, joiningDate '2019-06-20', department 'Banking');
28
```

Figure 2: Insertar 7 registros en Tabla Employee.

3. Escribe la sintaxis para crear la tabla “Reward”.

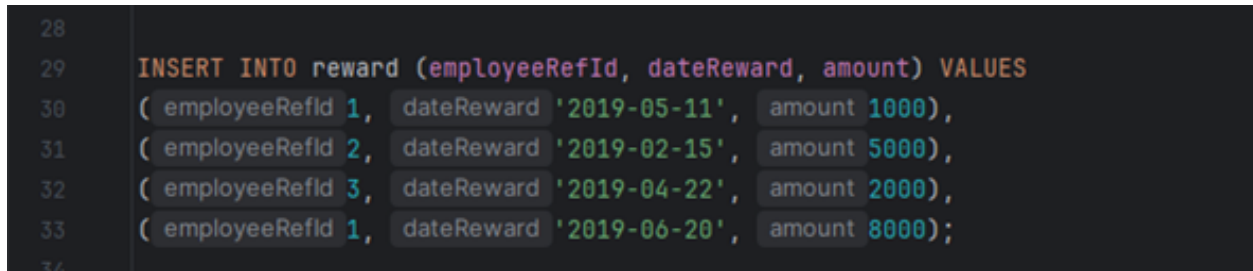
```
CREATE TABLE reward
(
employeeRefId int not null,
dateReward date not null,
amount float not null
);
```

```
12
13 CREATE TABLE reward
14 (
15 employeeRefId int not null,
16 dateReward date not null,
17 amount float not null
18 );
```

Figure 3: Crear Tabla Reward.

4. Escribe la sintaxis para insertar 4 registros (en la imagen) a la tabla “Reward”.

```
INSERT INTO reward (employeeRefId, dateReward, amount) VALUES  
(1, '2019-05-11', 1000),  
(2, '2019-02-15', 5000),  
(3, '2019-04-22', 2000),  
(1, '2019-06-20', 8000);
```

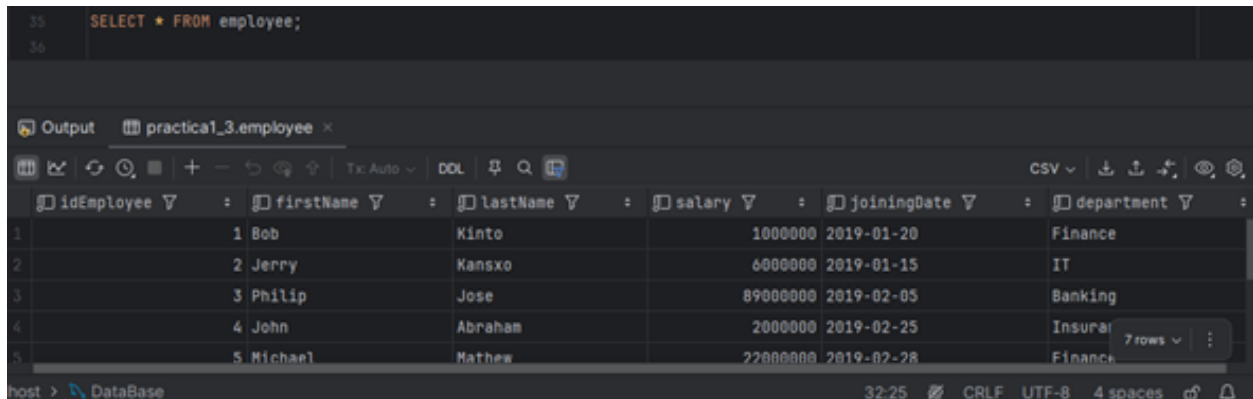


```
28  
29 INSERT INTO reward (employeeRefId, dateReward, amount) VALUES  
30 ( employeeRefId 1, dateReward '2019-05-11', amount 1000),  
31 ( employeeRefId 2, dateReward '2019-02-15', amount 5000),  
32 ( employeeRefId 3, dateReward '2019-04-22', amount 2000),  
33 ( employeeRefId 1, dateReward '2019-06-20', amount 8000);  
34
```

Figure 4: Insertar 4 registros en Tabla Reward.

5. Obtener todos los empleados.

```
SELECT * FROM employee;
```



```
35 SELECT * FROM employee;  
36
```

	idEmployee	firstName	lastName	salary	joiningDate	department
1	1	Bob	Kinto	1000000	2019-01-20	Finance
2	2	Jerry	Kansxo	6000000	2019-01-15	IT
3	3	Philip	Jose	89000000	2019-02-05	Banking
4	4	John	Abraham	2000000	2019-02-25	Insurance
5	5	Michael	Mathew	22000000	2019-02-28	Finance

Figure 5: Obtener todos los empleados.

6. Obtener el primer nombre y apellido de todos los empleados.
SELECT firstName, lastName FROM employee;

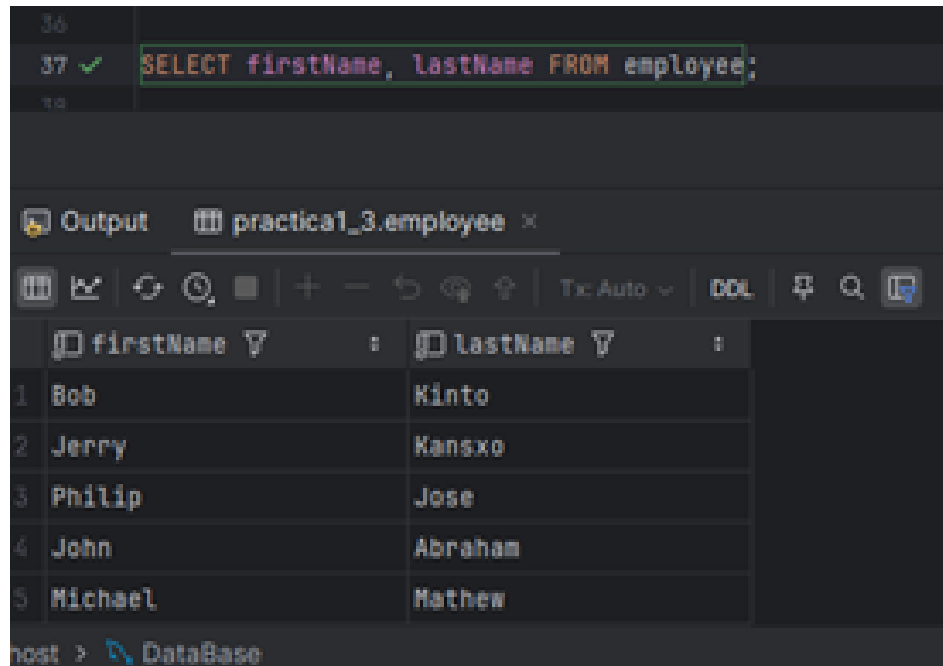


Figure 6: Primer Nombre y apellido Tabla Employee.

7. Obtener todos los valores de la columna "First name" usando el alias "Nombre de empleado".
SELECT firstName AS "Nombre de empleado" FROM employee;

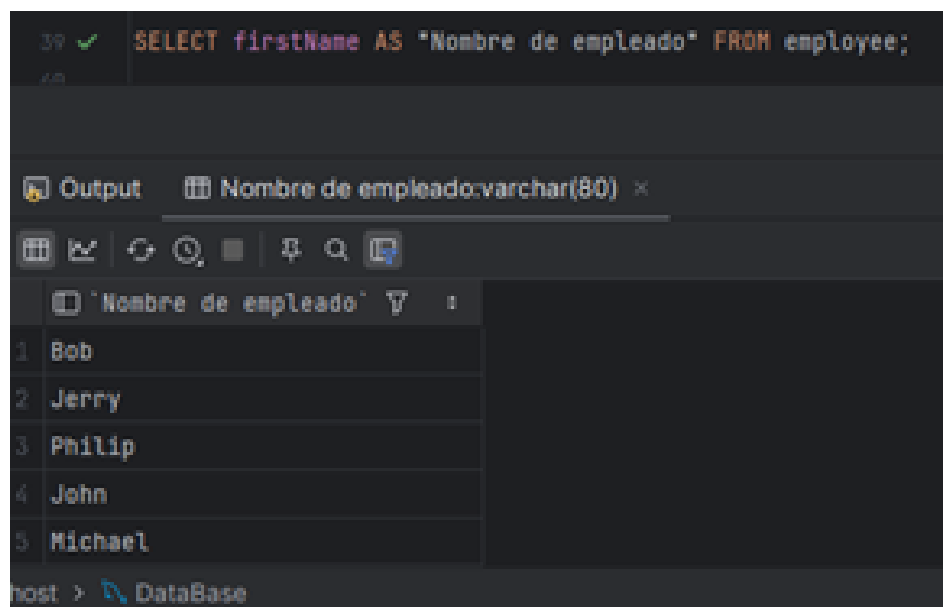
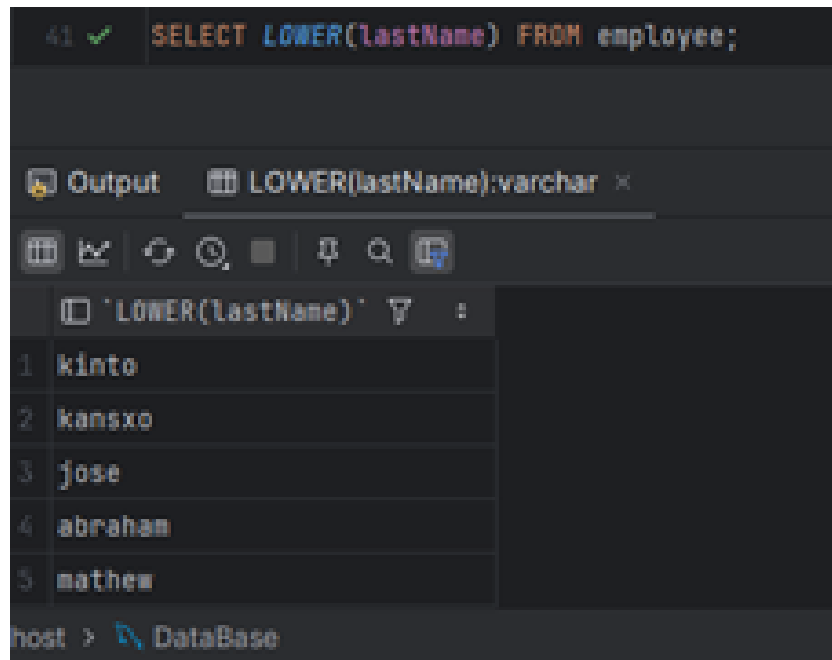


Figure 7: Valores de la columna First Name usando el alias "Nombre Empleado".

8. Obtener todos los valores de la columna “Last name” en minúsculas.
SELECT LOWER(lastName) FROM employee;



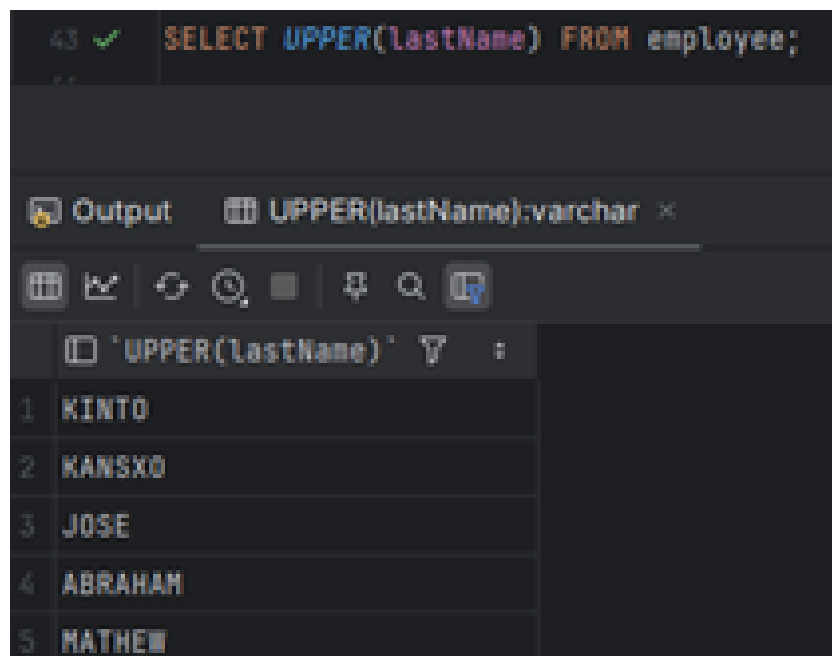
The screenshot shows a SQL query execution interface. At the top, the query `SELECT LOWER(lastName) FROM employee;` is entered. Below it, the 'Output' tab is selected, showing the result set for the column `LOWER(lastName):varchar`. The results are displayed in a table with 5 rows and 1 column:

	LOWER(lastName)
1	kinto
2	kansxo
3	jose
4	abraham
5	matheW

The status bar at the bottom indicates the connection is to 'host' and the database is 'DataBase'.

Figure 8: Valores de la columna Last Name en minusculas.

9. Obtener todos los valores de la columna “Last name” en mayúsculas.
SELECT UPPER(lastName) FROM employee;



The screenshot shows a SQL query execution interface. At the top, the query `SELECT UPPER(lastName) FROM employee;` is entered. Below it, the 'Output' tab is selected, showing the result set for the column `UPPER(lastName):varchar`. The results are displayed in a table with 5 rows and 1 column:

	UPPER(lastName)
1	KINTO
2	KANSXO
3	JOSE
4	ABRAHAM
5	MATHEW

The status bar at the bottom indicates the connection is to 'host' and the database is 'DataBase'.

Figure 9: Valores de la columna Last Name en mayusculas.

10. Obtener los nombre únicos de la columna “Departament”.
SELECT DISTINCT department FROM employee;

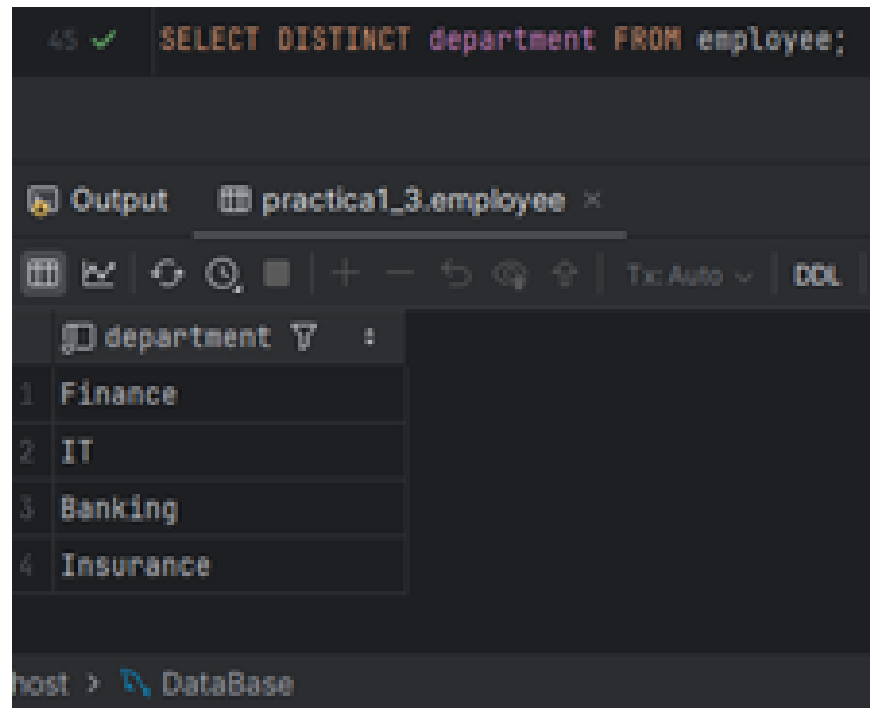


Figure 10: Nombres unicos de la columna Departament.

11. Obtener los primeros 4 caracteres de todos los valores de la columna “First name”.
SELECT LEFT(firstName, 4) AS primeros_4_caracteres FROM employee;

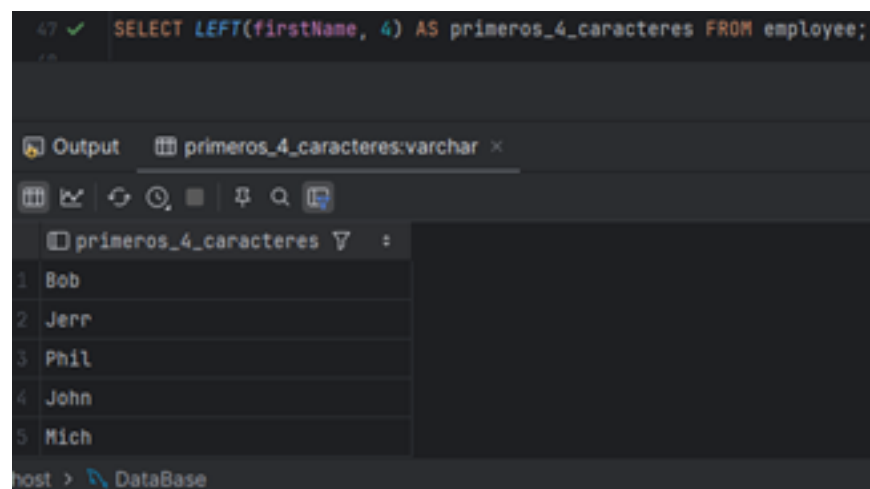


Figure 11: Primero 4 caracteres de la Tabla Employee.

12. Obtener la posición de la letra “h” en el nombre del empleado con First name=“Jhon”.

```
SELECT LOCATE('h', firstName) AS letra_h FROM employee WHERE  
firstName = 'Jhon';
```

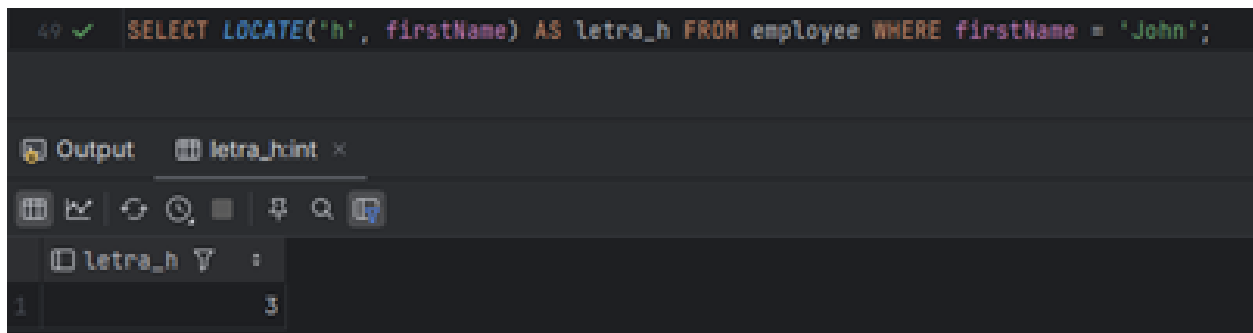


Figure 12: Posicion de letra h en el nombre Jhohn Tabla Employee.

13. Obtener todos los valores de la columna “First name” después de remover los espacios en blanco de la derecha. 1

```
SELECT RTRIM(firstName) AS sin_espacios_derecha FROM employee;  
espacios en blanco de la derecha.
```

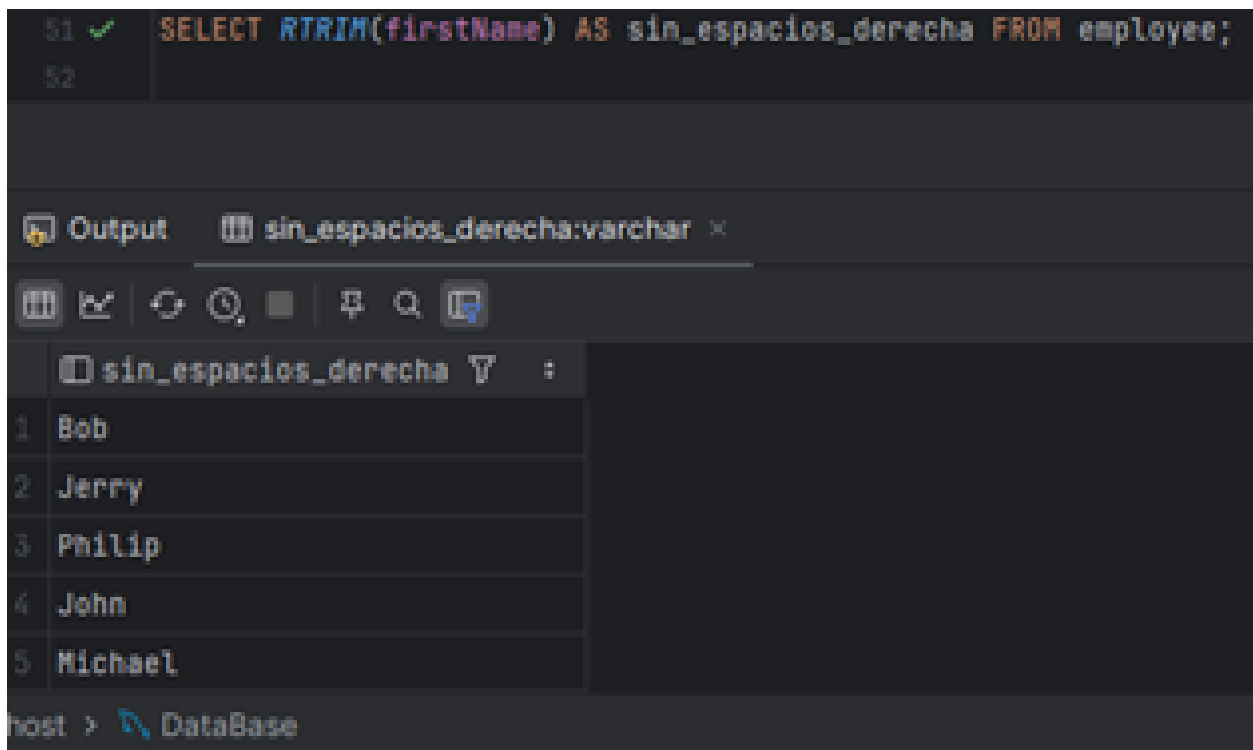
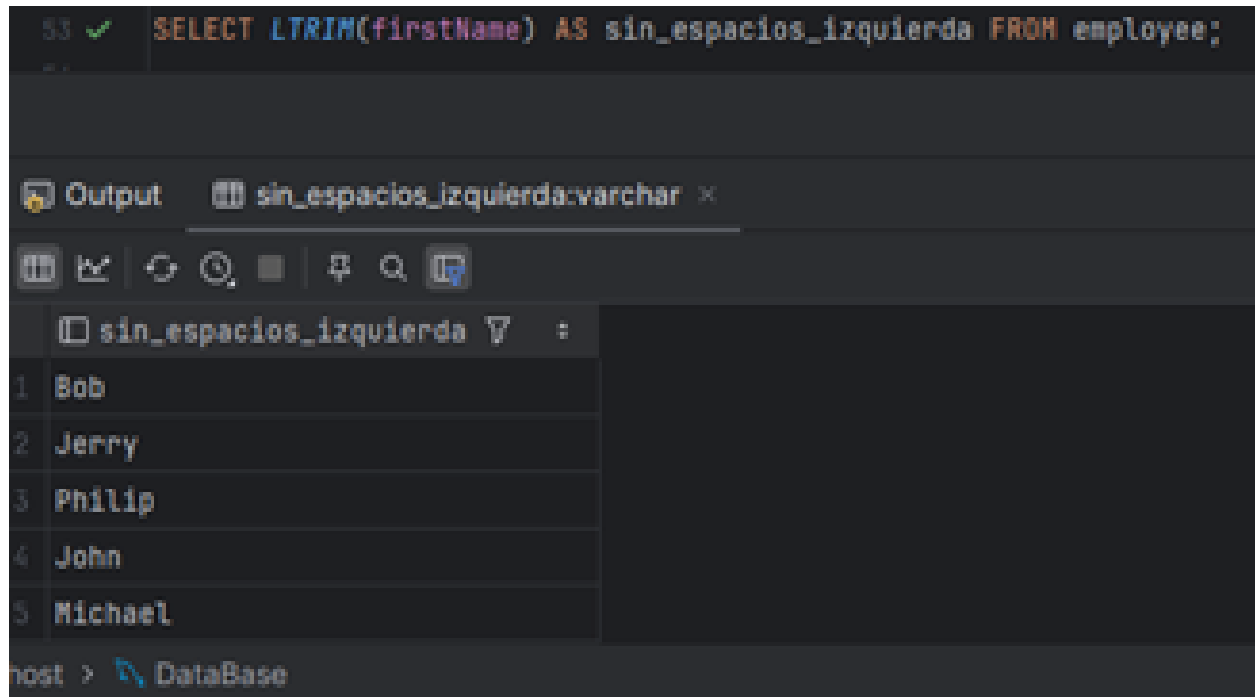


Figure 13: Valores de la columna First Name despues de remover espacios en blanco a la derecha.

14. Obtener todos los valores de la columna “First name” después de remover los espacios en blanco de la izquierda.

SELECT LTRIM(firstName) AS sin espacios izquierda FROM employee;



The screenshot shows a SQL IDE interface. At the top, a query editor displays the command: `SELECT LTRIM(firstName) AS sin_espacios_izquierda FROM employee;`. Below the editor, the 'Output' pane is active, showing the results of the query. The results are displayed in a table with a single column named 'sin_espacios_izquierda'. The table contains five rows of data: 'Bob', 'Jerry', 'Philip', 'John', and 'Michael'. The interface includes standard SQL IDE controls like a toolbar with icons for running, refreshing, and saving, and a status bar at the bottom indicating the current host and database.

	sin_espacios_izquierda
1	Bob
2	Jerry
3	Philip
4	John
5	Michael

Figure 14: Valores de la columna First Name despues de remover espacios en blanco a la derecha.

5. Conclusiones

A lo largo de esta práctica, fortalecí mis habilidades en la gestión y manipulación de bases de datos, lo que me permitió mejorar mi comprensión sobre la sintaxis SQL y la estructuración de consultas para la obtención de datos específicos. Aprendí a utilizar DataGrip, lo que facilitó la interacción con la base de datos y la ejecución de consultas de manera más eficiente. Además, adquirí experiencia en LaTeX, lo que me permitió estructurar y documentar de forma profesional los resultados obtenidos. Asimismo, exploré nuevas funciones SQL para la transformación y manipulación de datos, lo que me brindó una mejor comprensión sobre cómo optimizar consultas y mejorar su legibilidad. En general, esta práctica reforzó mis conocimientos y habilidades en bases de datos, herramientas especializadas y documentación técnica.

Referencias Bibliográficas

References

- [1] Codd, E. F. (**1970**). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377-387.
- [2] Elmasri, R.; Navathe, S. B. (**2015**). *Fundamentals of Database Systems* (7th ed.). Pearson.
- [3] Melton, J.; Simon, A. R. (**2002**). *SQL: 1999 - Understanding Relational Language Components*. Morgan Kaufmann.
- [4] Groff, J. R.; Weinberg, P. N. (**2009**). *SQL: The Complete Reference* (3rd ed.). McGraw-Hill.
- [5] DuBois, P. (**2014**). *MySQL Cookbook* (3rd ed.). O'Reilly Media.
- [6] Schwartz, B.; Zaitsev, P.; Tkachenko, V. (**2012**). *High Performance MySQL* (3rd ed.). O'Reilly Media.