**AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)**

FACULTY OF SCIENCE & TECHNOLOGY

A picture containing calendar

Description automatically generated

**Course Title**

INTRODUCTION TO DATABASE

**Summer 2022-2023**

**Section: G**

**TITLE**

Car Shop Management System

**Supervised By**

MD Sajid Bin Faisal

**Submitted By: Group no:**12

|  |  |
| --- | --- |
| Name | ID |
| MD. Nazmus Sadat Numan | 22-48497-3 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| SL | Topic | page |
| 01 | Introduction | 02 |
| 02 | Case study | 02 |
| 03 | ER Diagram | 03 |
| 04 | Normalization | 04 - 05 |
| 05 | Finalization and final tables | 06 |
| 06 | Table creation | 07 - 14 |
| 07 | Value insertion | 15 - 22 |
| 08 | Query test | 23 - 31 |
| 09 | Conclusion | 32 |

**Introduction**

This initiative brings together elements, in the car shop to create a harmonious environment for managers, employees and customers. Built on Oracle Live, it empowers managers with insights based on data analysis. Employees benefit from workflows and tools that prioritize customer satisfaction. As for the customers themselves they can expect a buying experience, with support and flexible payment options. This project isn't just software, it's an economic engine, driving customer satisfaction, xemployee productivity, and informed decisions. Whether you're running the shop playing your part or simply enjoying the journey this system takes your car shop experience to a level.

**Case Study**

In a Car shop management system, the Car shop has one Manager. The system is responsible for keeping the manager's name, ID, salary, address, and phone number. The manager manages the employees. There is only one manager responsible for managing multiple employees. The system also stores the employee's information, including their employee ID, name, job type, salary, address, and phone number. Employee assists customers. Multiple employees can assist multiple customers at the same time. Customer buys cars. Here, one customer can buy multiple cars, and one car can be bought by multiple customers. The system stores the customer's ID, name, email, phone number, address, and car information such as car ID, car name, car color, car brand, and car price. After buying cars, the customer makes payment using any payment method. Payment information such as payment date, payment ID, and payment type are stored by the system.

**ER-Diagram**



Fig3.1:ER-Diagram

**Normalization**

**Manages Relation:** (1..\*)

UNF: (m\_id, m\_name, m\_phone, m\_address, m\_salary, e\_id, e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype)

1NF: (m\_id, m\_name, m\_phone, m\_address, m\_salary, e\_id, e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype)

2NF:

1. m\_id (PK), m\_name, m\_phone, m\_address, m\_salary
2. e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype, m\_id (FK)

3NF:

1. m\_id (PK), m\_name, m\_phone, m\_address, m\_salary
2. e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype, m\_id (FK)

**Assists Relation:** (\*..\*)

UNF: ( e\_id, e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype, c\_id, c\_name, c\_address, c\_phone, c\_ email)

1NF: (e\_id, e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype, c\_id, c\_name, c\_address, c\_phone, c\_ email)

2NF:

1. e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype
2. c\_id (PK), c\_name, c\_address, c\_phone, c\_ email
3. e\_id (PK), c\_id (FK)

3NF:

1. e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype
2. c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email
3. e\_id (PK), c\_id (FK)

**Buys Relation:** (\*..\*)

UNF: (c\_id, c\_name, c\_address, c\_phone, c\_ email, car\_id, car\_name, car\_brand, car\_color, car\_price)

1NF: (c\_id, c\_name, c\_address, c\_phone, c\_ email, car\_id, car\_name, car\_brand, car\_color, car\_price)

2NF:

1. c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email
2. car\_id (PK), car\_name, car\_brand, car\_color, car\_price
3. c\_id (PK), car\_id (FK)

3NF:

1. c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email
2. car\_id (PK), car\_name, car\_brand, car\_color, car\_price
3. c\_id (PK), car\_id (FK)

**Pays Relation:** (1..\*)

UNF: (c\_id, c\_name, c\_address, c\_phone, c\_email, p\_id, p\_date, p\_type)

1NF: (c\_id, c\_name, c\_address, c\_phone, c\_email, p\_id, p\_date, p\_type)

2NF:

1. c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email
2. p\_id (PK), p\_date, p\_type, c\_id(FK)

3NF:

1. c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email
2. p\_id (PK), p\_date, p\_type, c\_id(FK)

**Finalization:**

1. m\_id (PK), m\_name, m\_phone, m\_address, m\_salary
2. e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype, m\_id (FK)
3. e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype
4. c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email
5. e\_id (PK), c\_id (FK)
6. ~~c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email~~
7. car\_id (PK), car\_name, car\_brand, car\_color, car\_price
8. c\_id (PK), car\_id (FK)
9. ~~c\_id (PK), c\_name, c\_address,c\_ phone, c\_ email~~
10. p\_id (PK), p\_date, p\_type, c\_id (FK)

**Final Table**

* 1. **Manager Table:** (m\_id (PK), m\_name, m\_phone, m\_address, m\_salary)
  2. **Employee Table:** (e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype)
  3. **Customer Table:** (c\_id (PK), c\_name, c\_address, c\_phone, c\_ email)
  4. **Car Table:** (car\_id (PK), car\_name, car\_brand, car\_color, car\_price)
  5. **Manage Table:** (e\_id (PK), e\_name, e\_ phone, e\_address, e\_ salary, e\_ jobtype, m\_id (FK))
  6. **Assist Table:** (e\_id (PK), c\_id (FK))
  7. **Buy Table:** (c\_id (PK), car\_id (FK))
  8. **Pay Table:** (p\_id (PK), p\_date, p\_type, c\_id (FK))

**Table creation**

1. **Manager Table:**

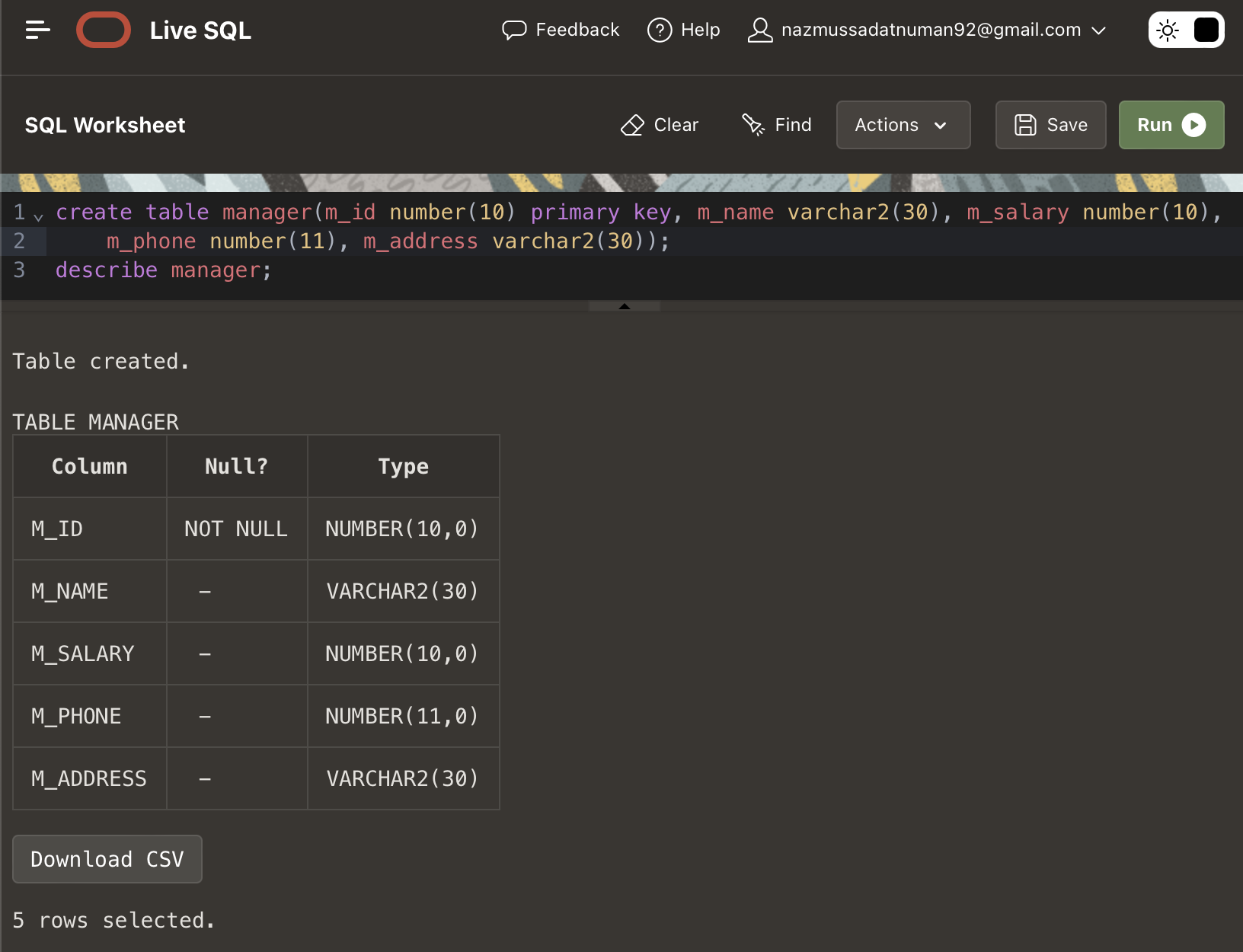


Fig6.1: creation and description of the manager table.

1. **Employee Table:**

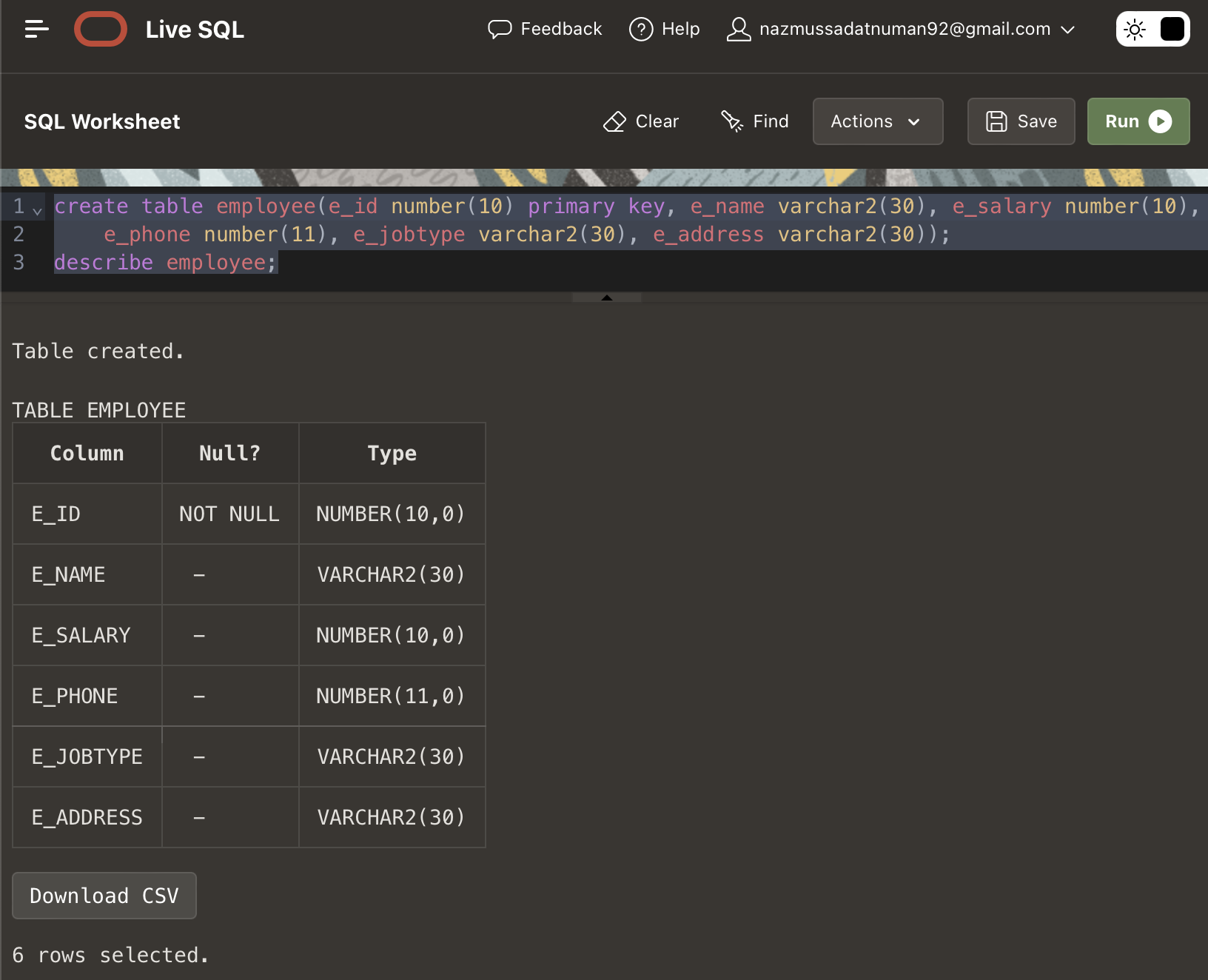


Fig6.2: creation and description of the employee table.

1. **Customer Table:**

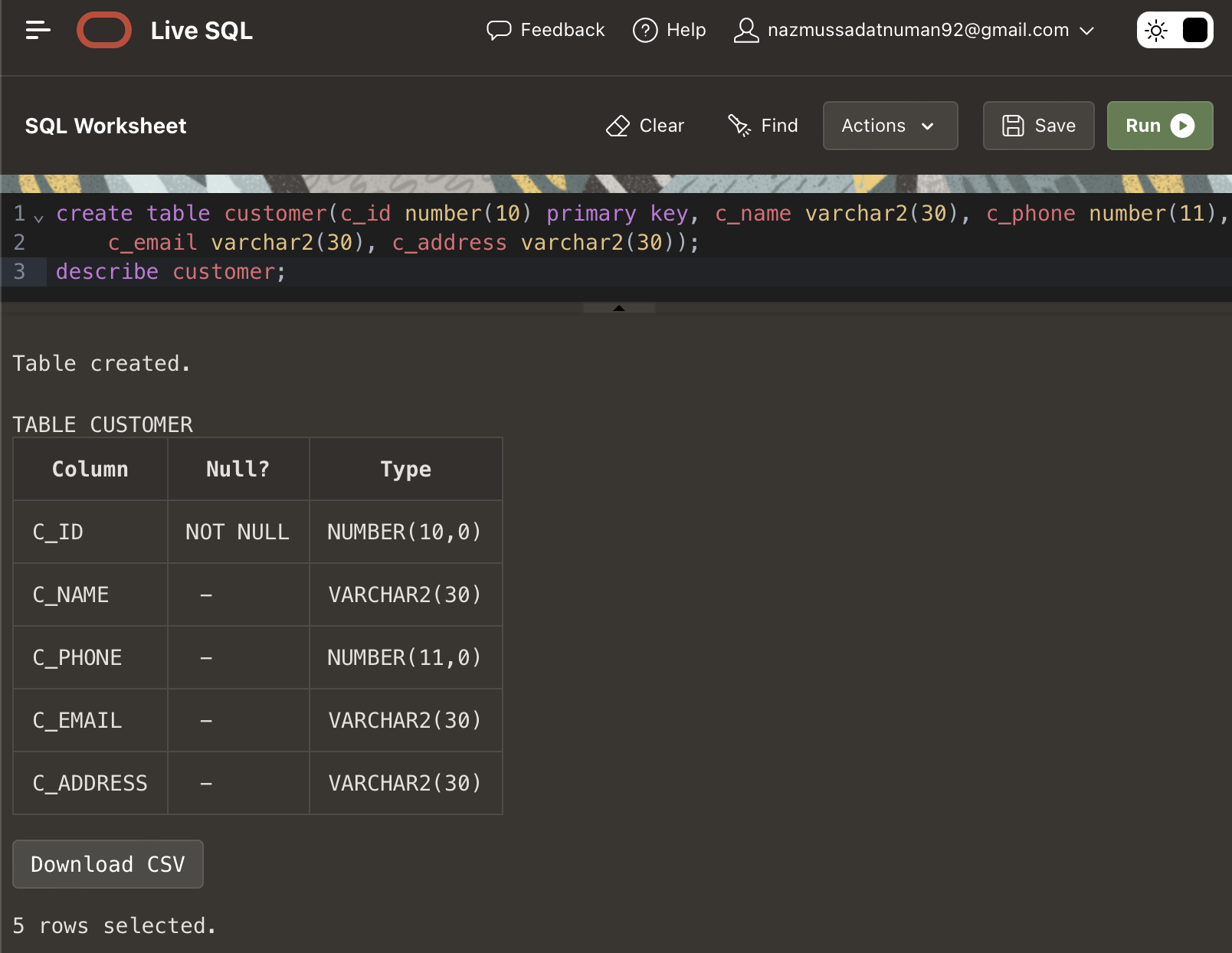


Fig6.3: creation and description of the customer table.

1. **Car Table:**

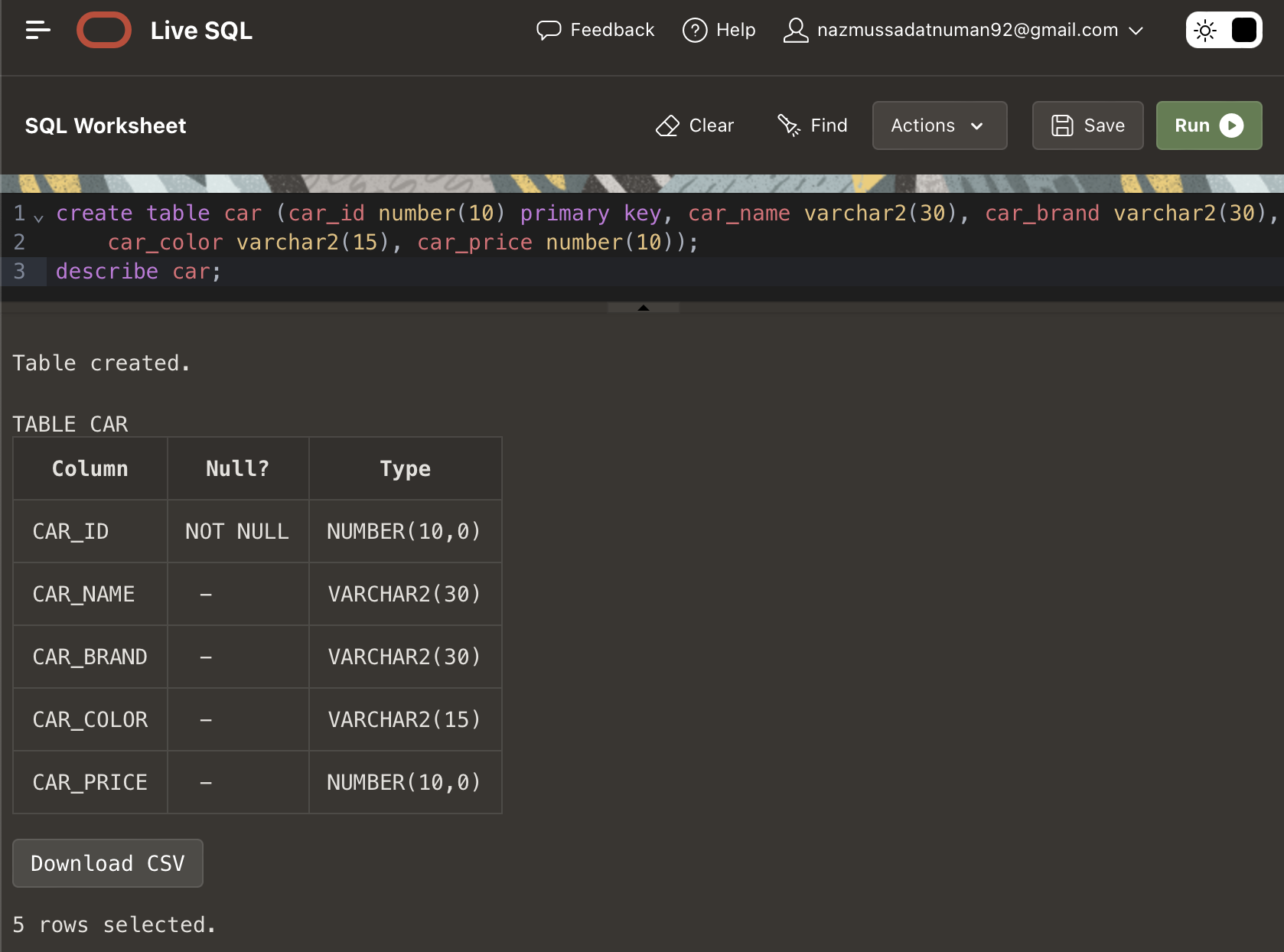


Fig6.4: creation and description of the table.

1. **Pay Table:**

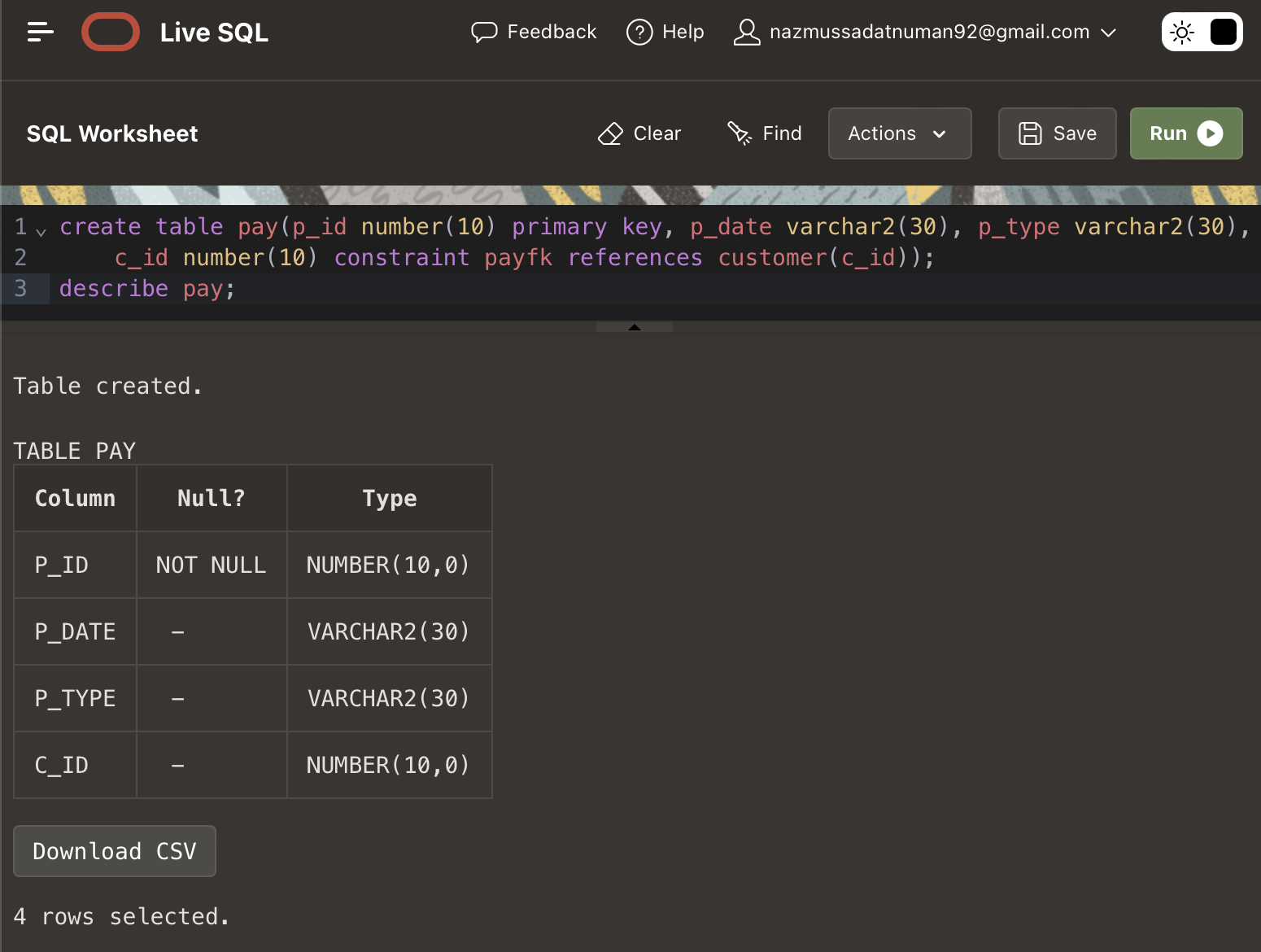


Fig6.5: creation and description of the pay table.

1. **Mange Table:**

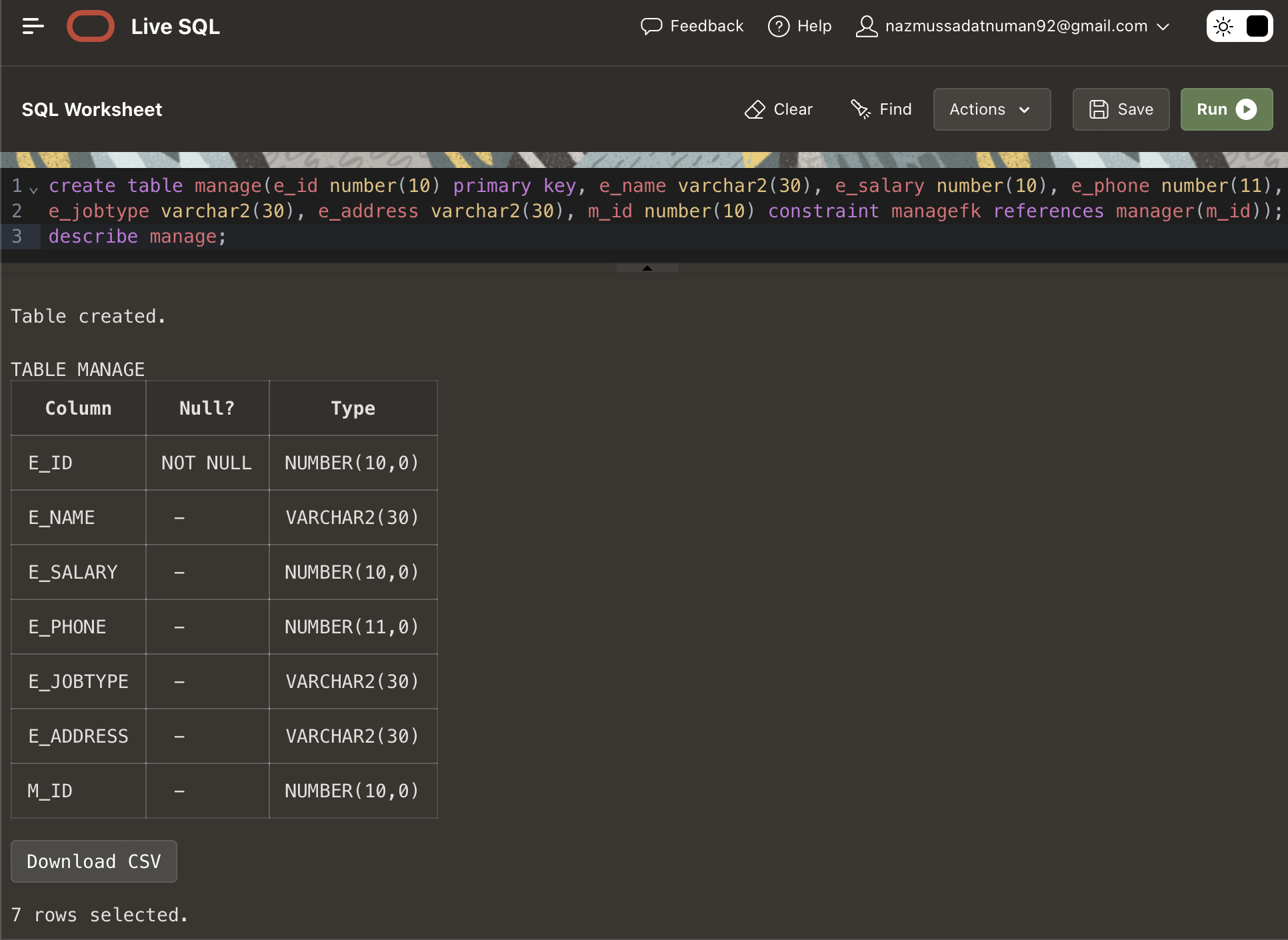


Fig6.6: creation and description of the manage table.

1. **Assist Table:**

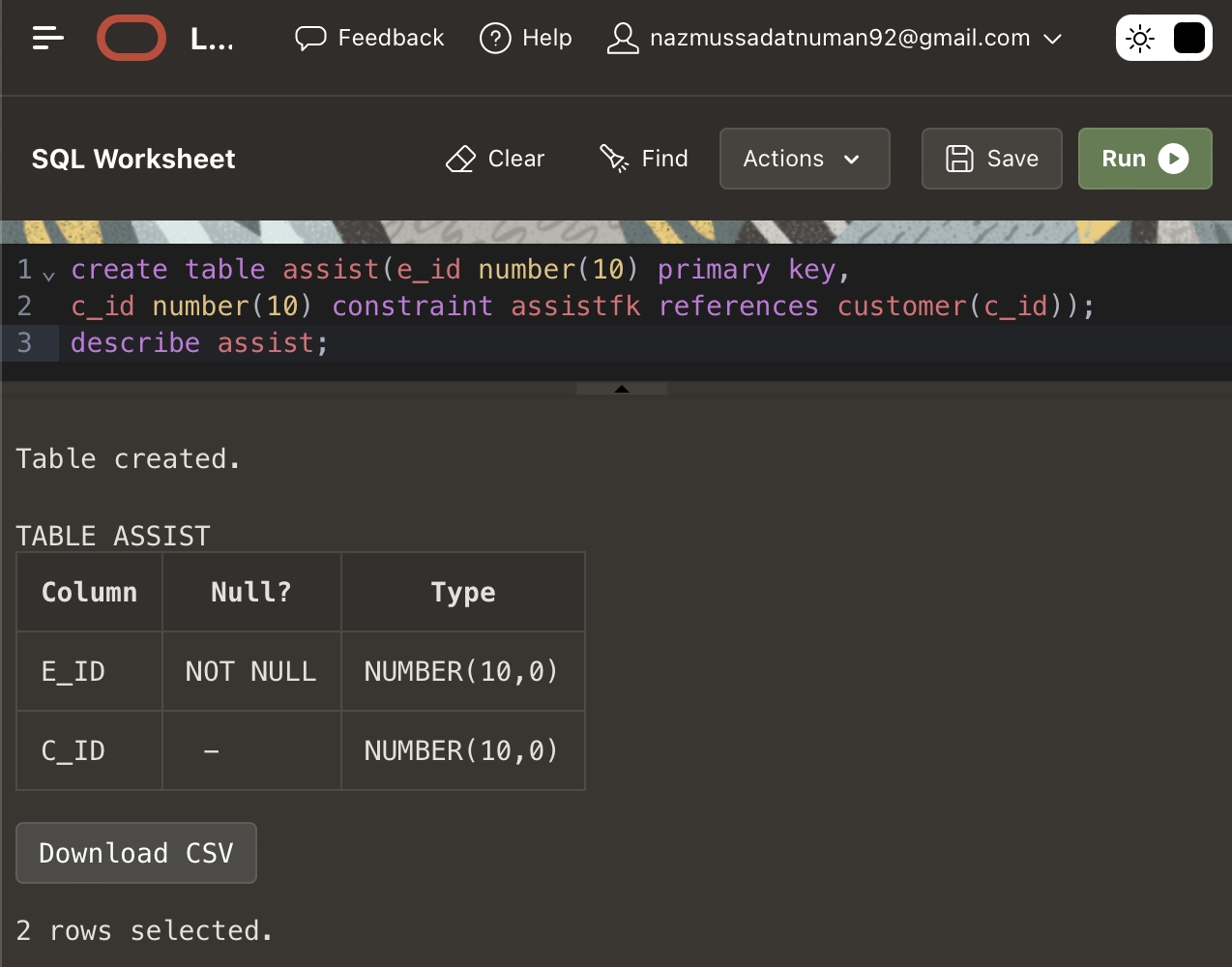


Fig6.7: creation and description of the assist table.

1. **Buy Table:**

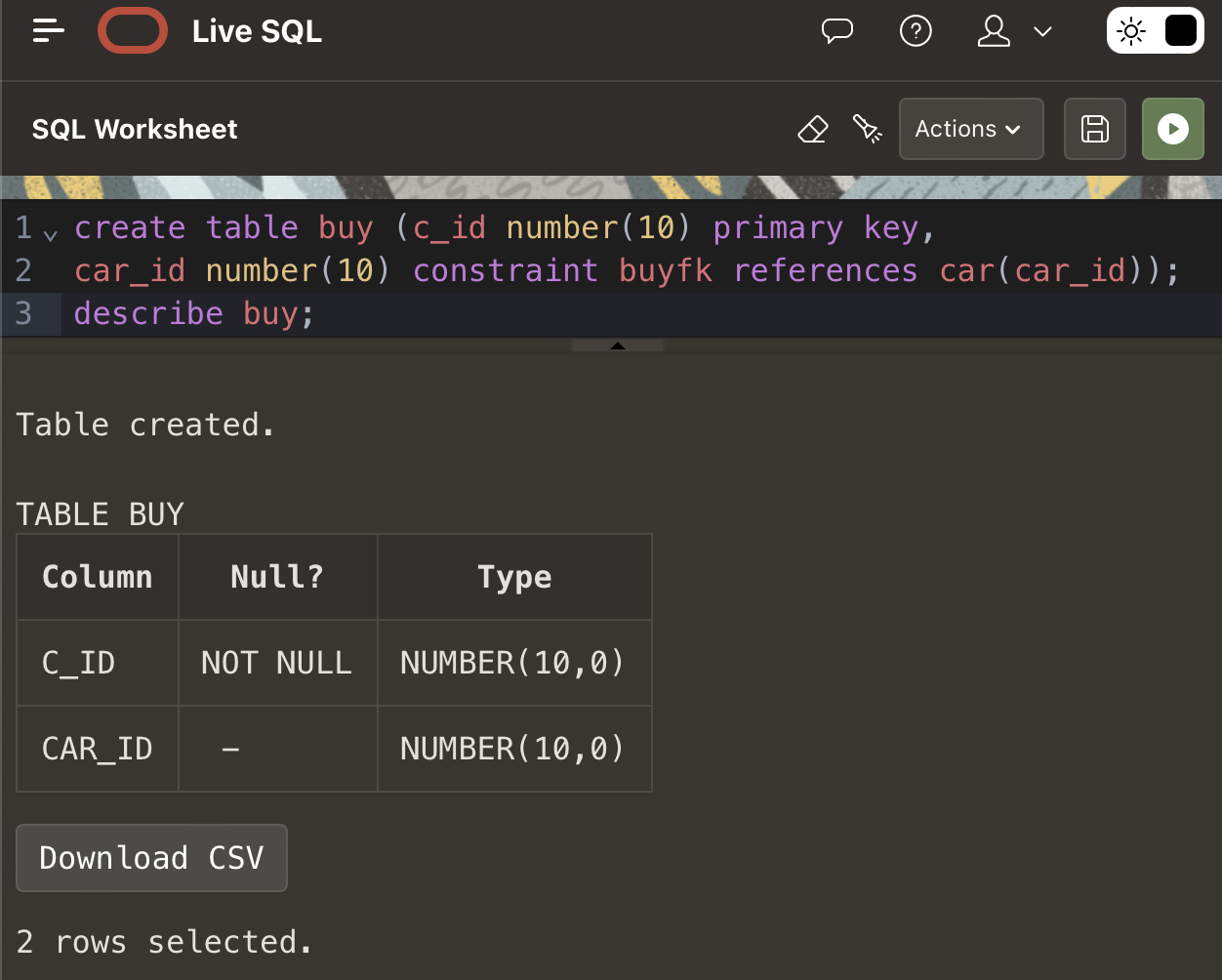


Fig6.8: creation and description of the buy table.

**Value Insertion**

1.**Manager:**

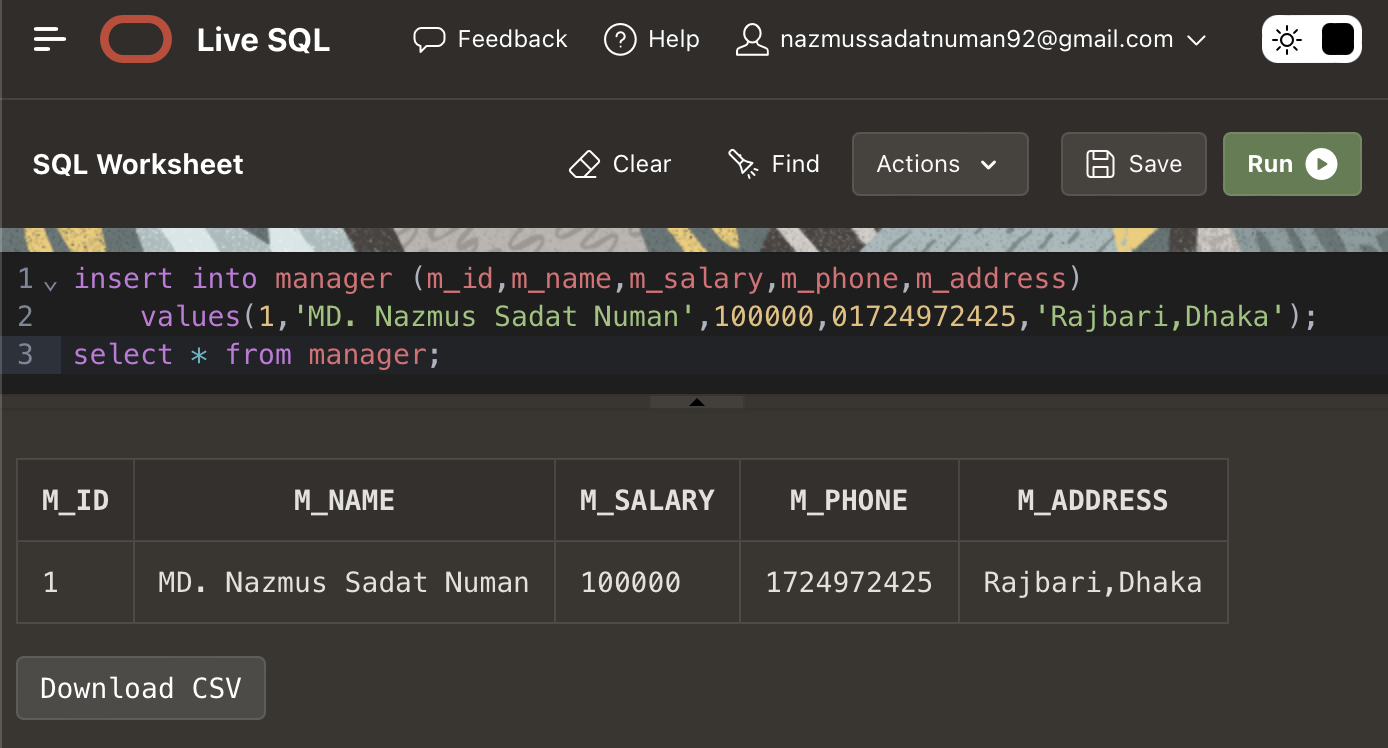


Fig7.1: Input values for the manager table and all its data.

2.**Employee:**

A screenshot of a computer

Description automatically generated

Fig7.2: Input values for the employee table and all its data.

3.**Customer:**

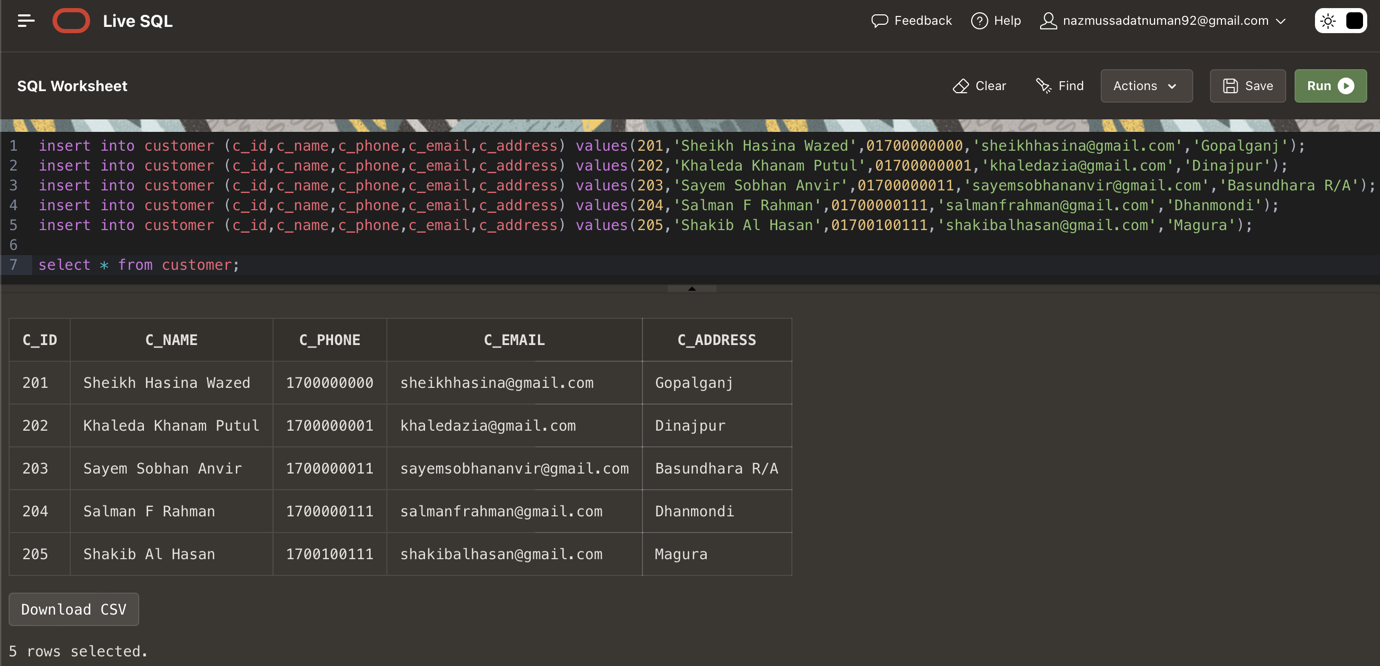


Fig7.3: Input values for the customer table and all its data.

4.**Car:**

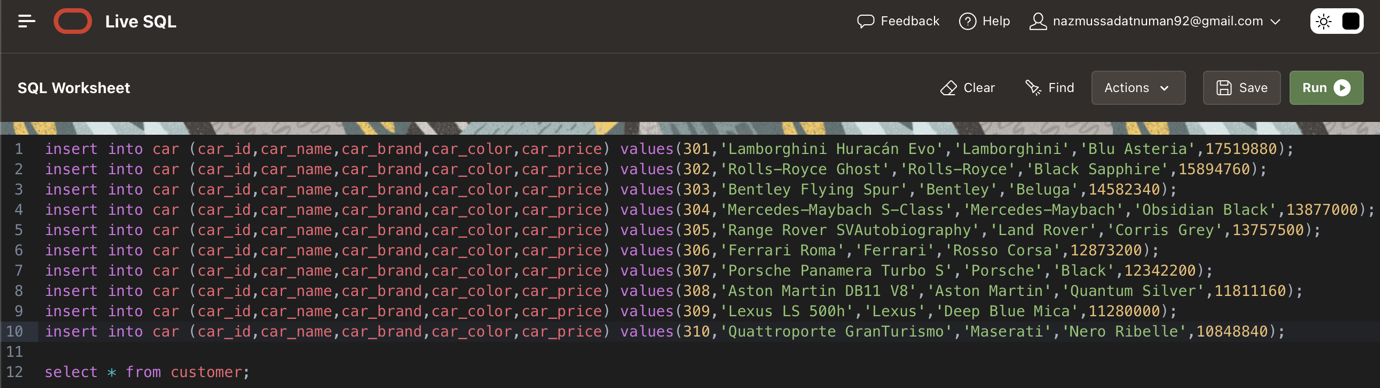


Fig7.4: Input values for the car table.



Fig7.5: All data of car table.

5.**Pay:**

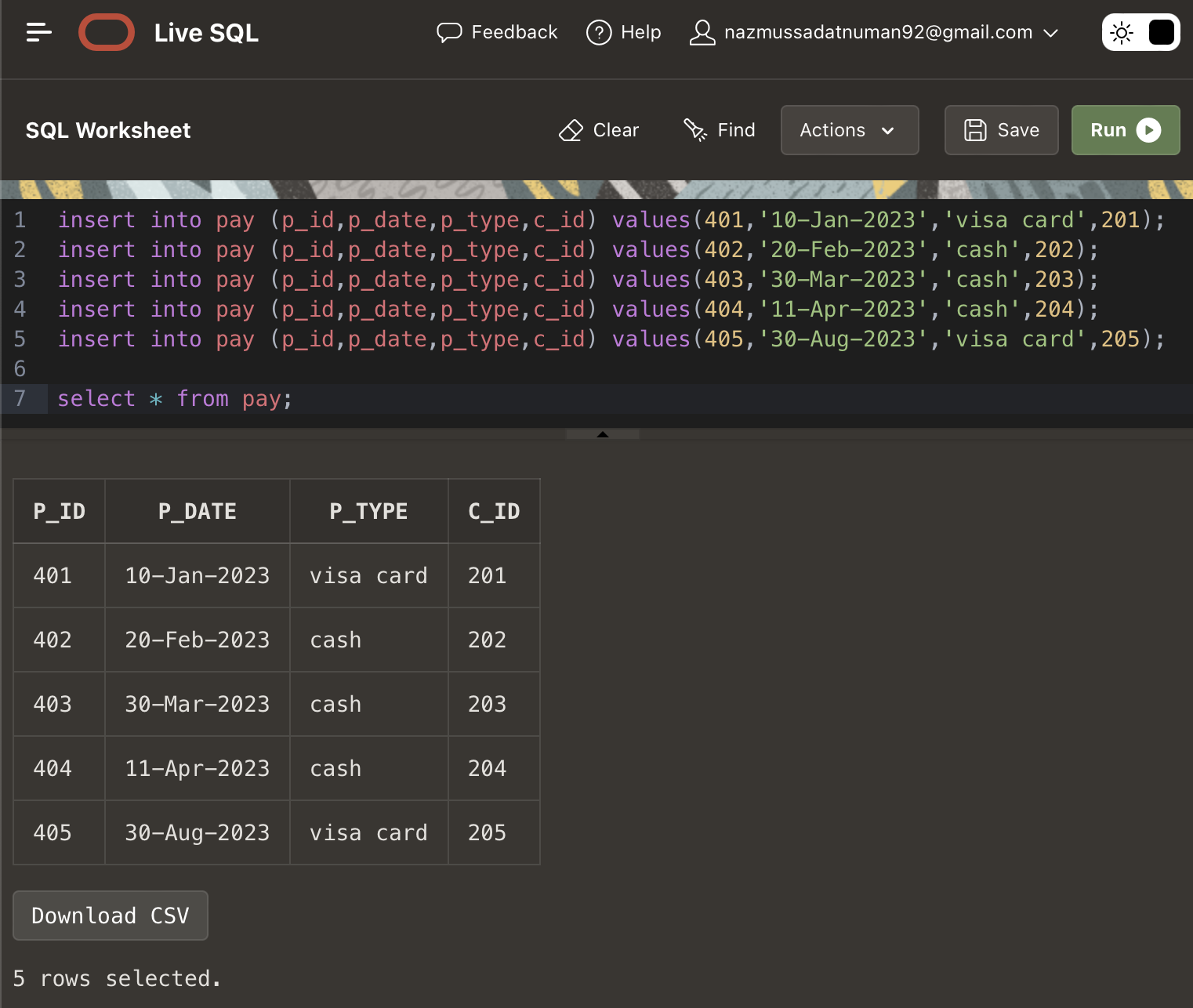


Fig7.6: Input values for the pay table and all its data.

6.**Manage:**

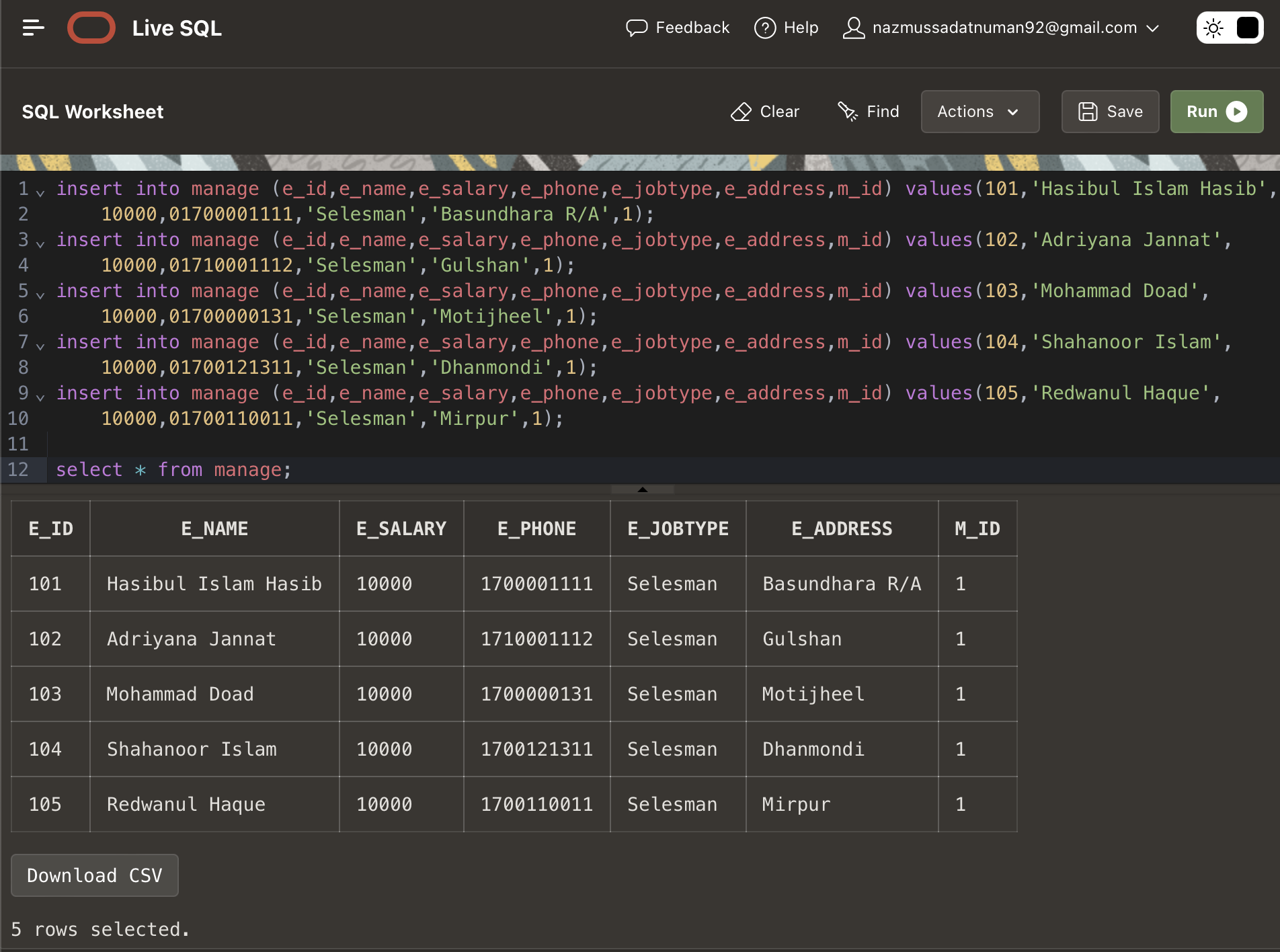


Fig7.7: Input values for the manage table and all its data.

7.**Assist:**

A screenshot of a computer

Description automatically generated

Fig7.8: Input values for the assist table and all its data.

8.**Buy:**

A screenshot of a computer

Description automatically generated

Fig7.9: Input values for the buy table and all its data.

**Query Test**

**Simple Query:**

Showing the new salary with 5% increment of all employees with their NAME, SALARY, 5% INCREMENT.

SELECT e\_name "NAME", e\_salary "SALARY", e\_salary\*0.05 "5% INCREMENT", (e\_salary\*0.05)+e\_salary "NEW SALARY" FROM employee;

A screenshot of a computer

Description automatically generated

Fig8.1: Simple Query command and full table as a result

**Single row function:**

Showing the id, name, job, salary of the employees by changing the salary for the people whose id is 202, 203, 204, 205 respectively 5%, 7%, 9% and 10% increment as “New salary”.

SELECT e\_id AS "ID", e\_name AS "Name", e\_jobtype AS "JOB", e\_salary AS "SALARY", DECODE( e\_id, 102, (e\_salary \* 0.05)+e\_salary, 103, (e\_salary \* 0.07)+e\_salary, 104, (e\_salary \* 0.09)+e\_salary, 105, (e\_salary \* 0.1)+e\_salary, e\_salary ) AS "NEW SALARY" FROM employee;

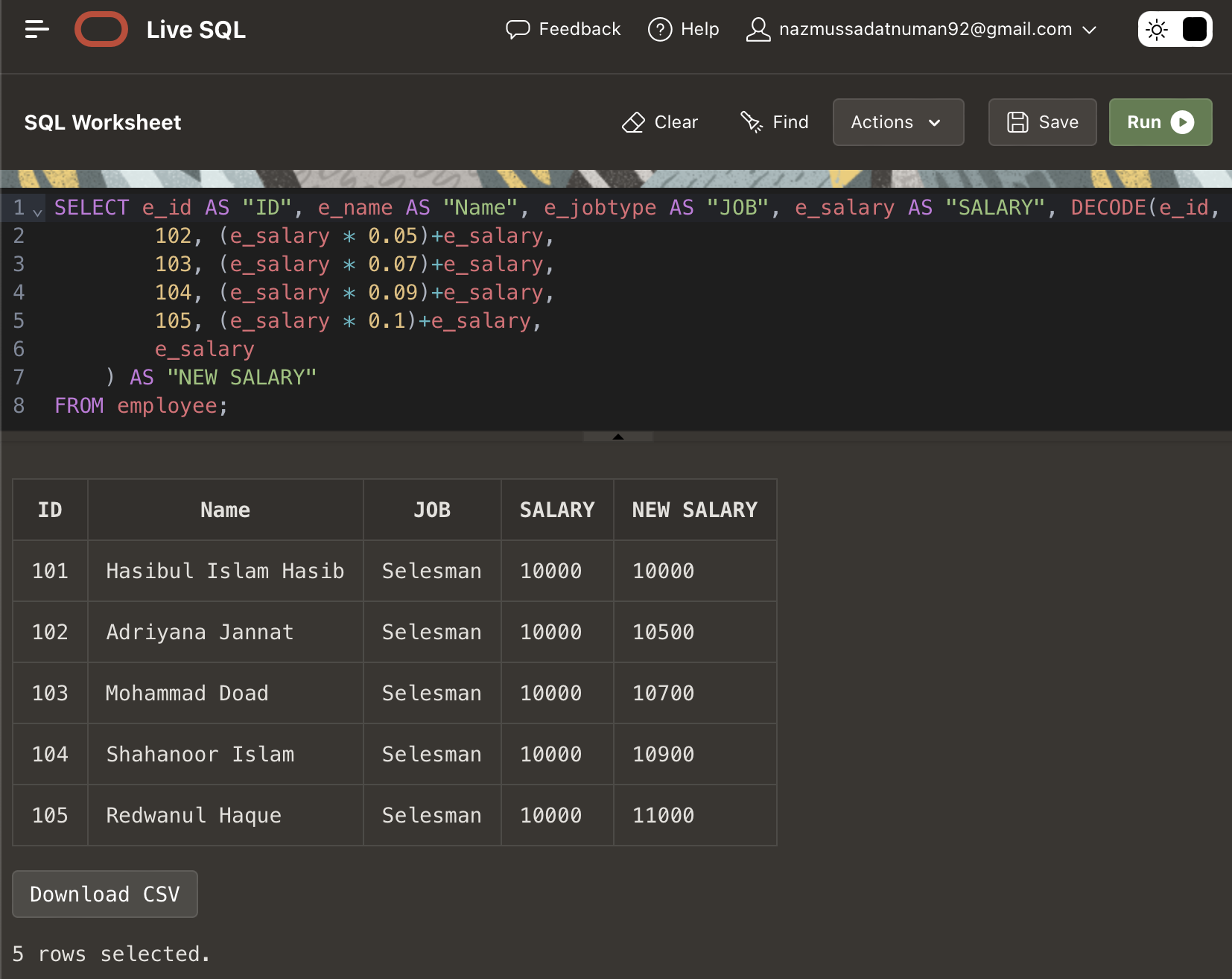


Fig8.2: Single row function command and full table as a result.

Multiple row function:

Calculating the average salary, sum of total salary, and count of employees for the manager from the manage table.

SELECT m\_id, AVG(e\_salary) AS avg\_salary, SUM(e\_salary) AS SUM\_Salary, COUNT(\*) AS employee\_count FROM manage GROUP BY m\_id;

A screenshot of a computer

Description automatically generated

Fig8.3: Multiple row function command and full table as a result.

**Single row subquery:**

Showing the name, ID, and brand of the car that has a price less than the average.

SELECT car\_id, car\_name, car\_brand FROM car WHERE car\_price < (SELECT AVG(car\_price) FROM car);

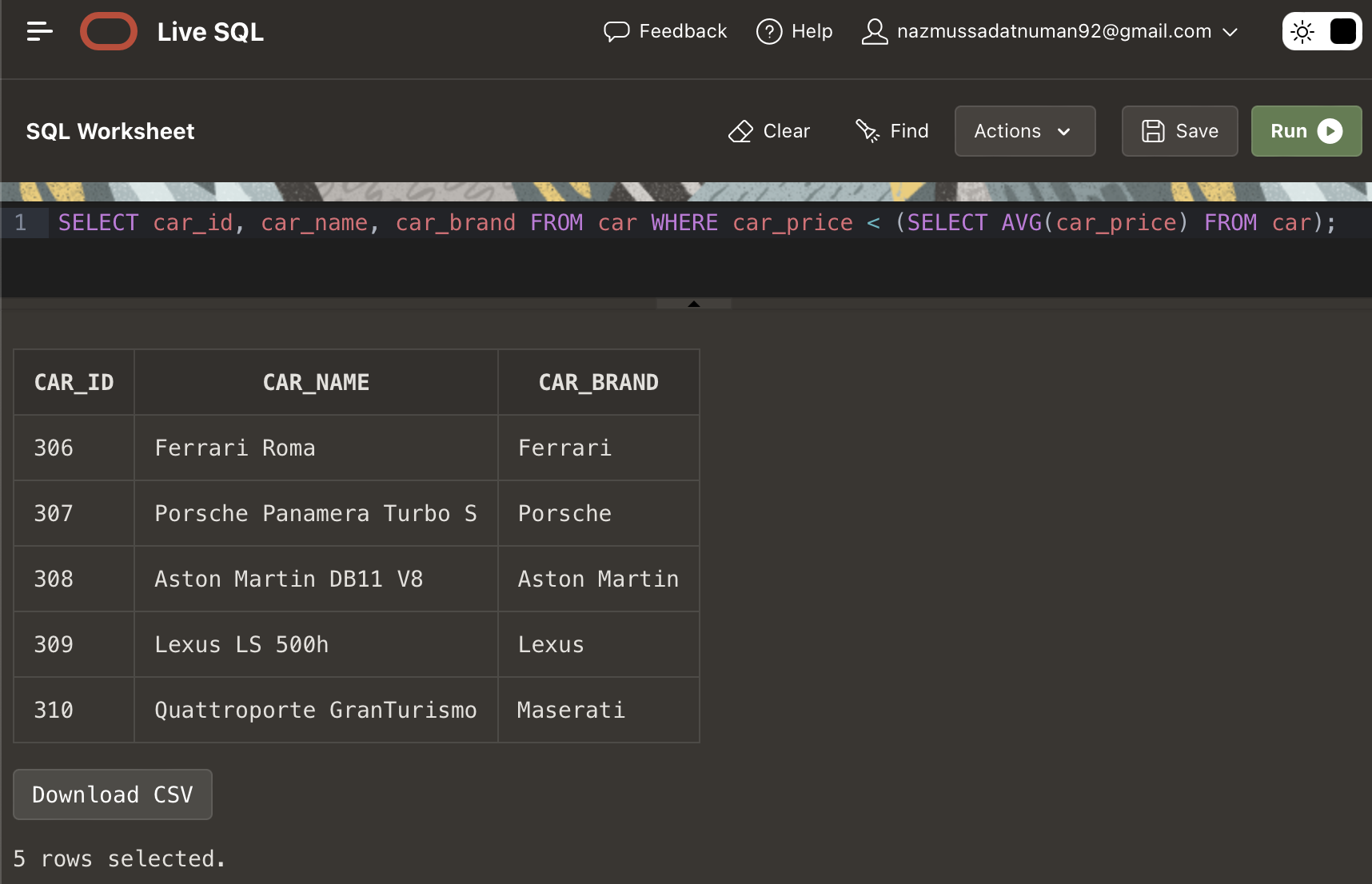


Fig8.4: Single row subquery command and full table as a result.

**Multiple row subquery:**

Showing the name, ID, and address of employees whose names are longer than 'Mohammed Doad'.

SELECT e\_id, e\_name, e\_address FROM employee WHERE e\_id = ANY (SELECT e\_id FROM employee WHERE LENGTH(e\_name)>LENGTH('Mohammad Doad'));

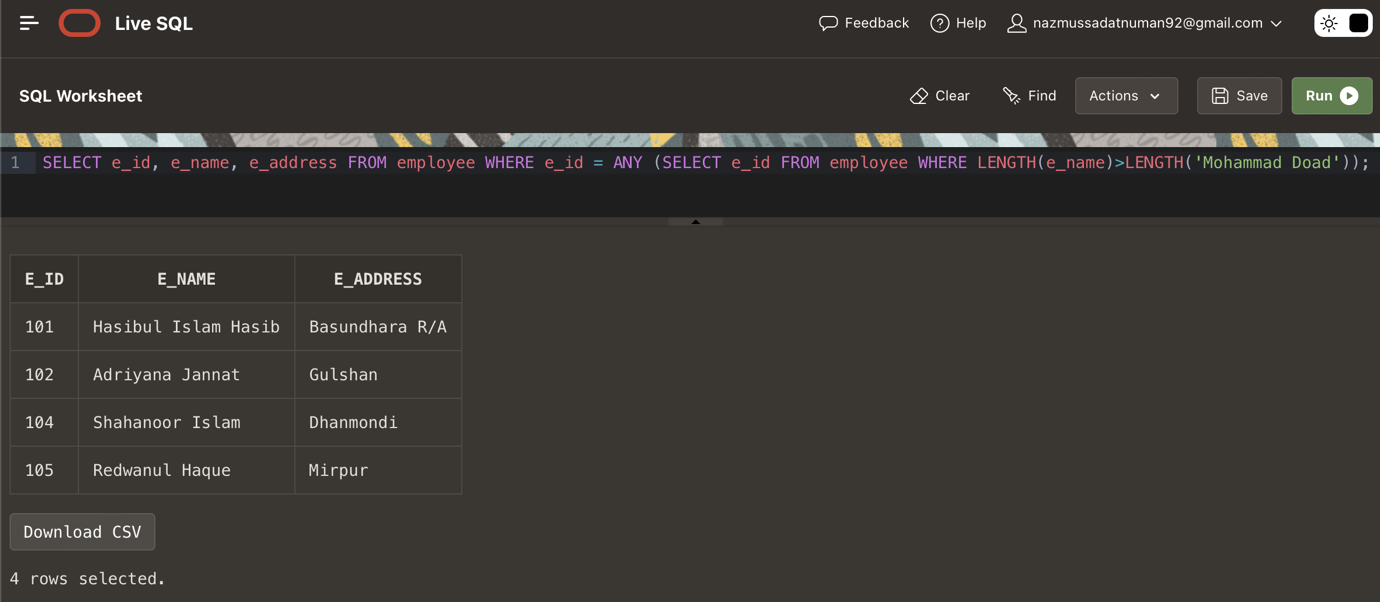


Fig8.5: Multiple row subquery command and full table as a result.

**Right outer join:**

List of all cars and their details, along with the corresponding customer ID of those who bought them. If a car has not been bought, display NULL for the customer ID.

SELECT c.car\_id, c.car\_name, c.car\_brand, c.car\_color, b.c\_id "Customer ID" FROM car c, buy b WHERE c.car\_id = b.car\_id(+);

A screenshot of a computer

Description automatically generated

Fig8.6: Right outer join creation command and full table as a result.

**Self-join:**

Self join of car table. Car table divided into 2 part: (a.car\_id, a.car\_name) and (b.car\_id, b.car\_brand, b.car\_price)

SELECT a.car\_id, a.car\_name, b.car\_id, b.car\_brand, b.car\_price FROM car a, car b WHERE a.car\_id = b.car\_id;

A screenshot of a computer

Description automatically generated

Fig8.7: Self join creation command and full table as a result.

**Simple view:**

Create a view named as customer\_view where car name, brand, color and price will be shown to the customer.

CREATE VIEW customer\_view AS SELECT car\_name,car\_brand,car\_color,car\_price FROM car;

A screenshot of a computer

Description automatically generated

Fig8.8: Simple view creation command.

DESCRIBE customer\_view;

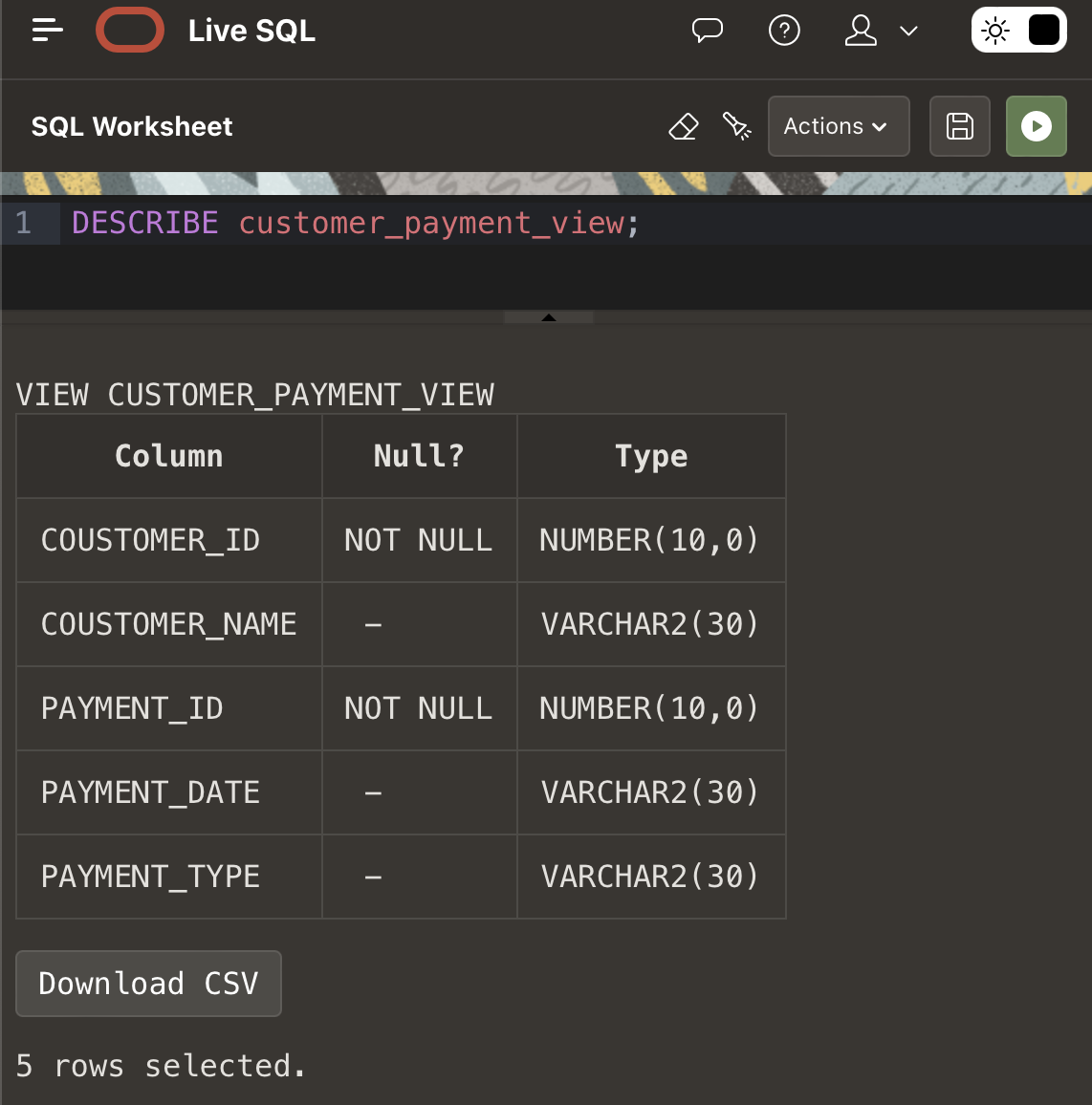


Fig8.9: Description of the simple view.

SELECT \* FROM customer\_view;

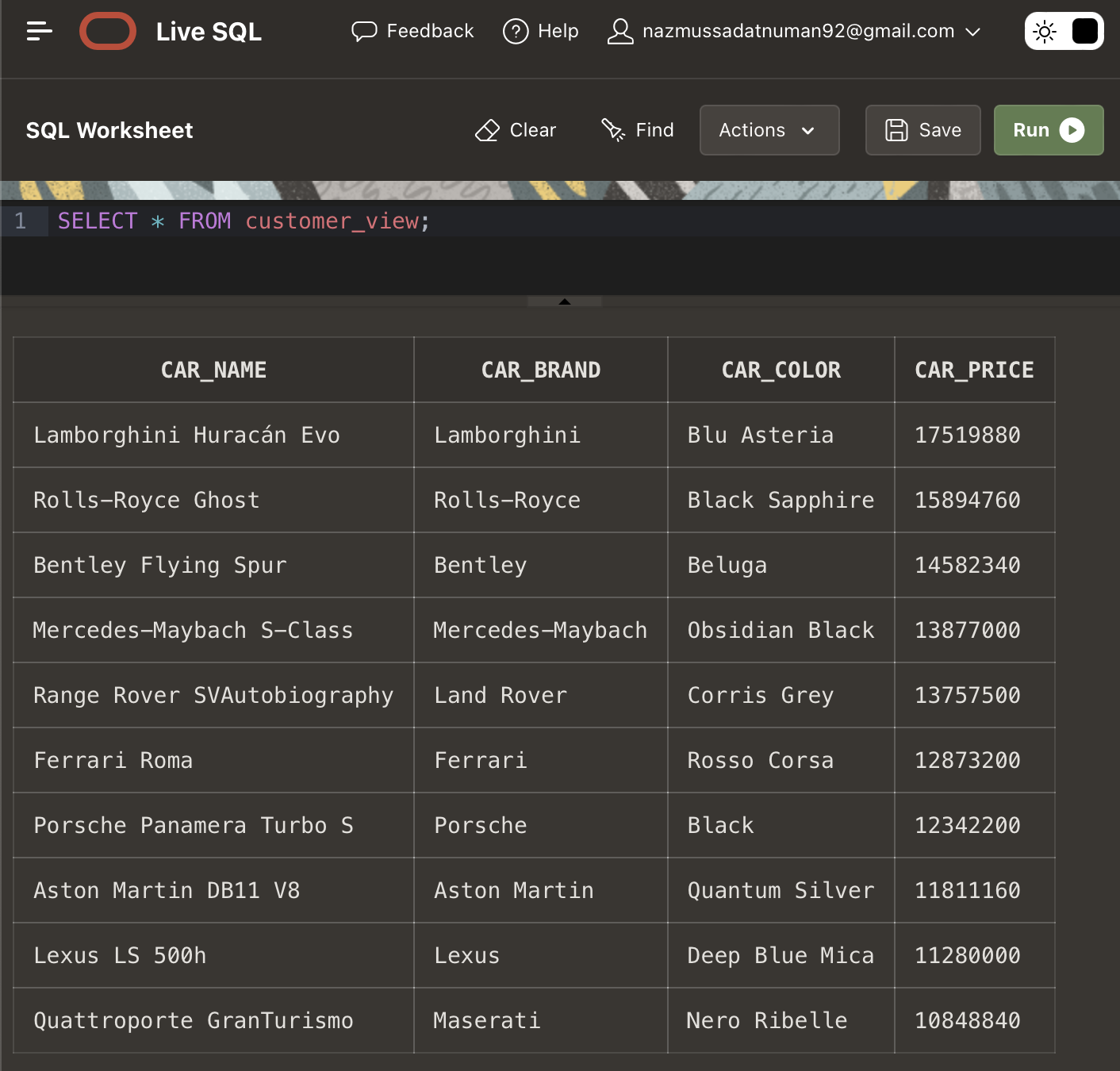


Fig8.10: Result of the simple view as a whole table.

**Complex view:**

Create a view named as customer\_payment\_view where Coustomer ID, Name and Payment ID, Date, Type will be shown.

CREATE VIEW customer\_payment\_view (Coustomer\_ID, Coustomer\_Name, Payment\_ID, Payment\_Date, Payment\_Type) AS SELECT c.c\_id, c.c\_name, p.p\_id, p.p\_date, p.p\_type FROM customer c, pay p WHERE c.c\_id = p.c\_id;

A screenshot of a computer

Description automatically generated

Fig8.11: Complex view creation command.

DESCRIBE customer\_payment\_view;

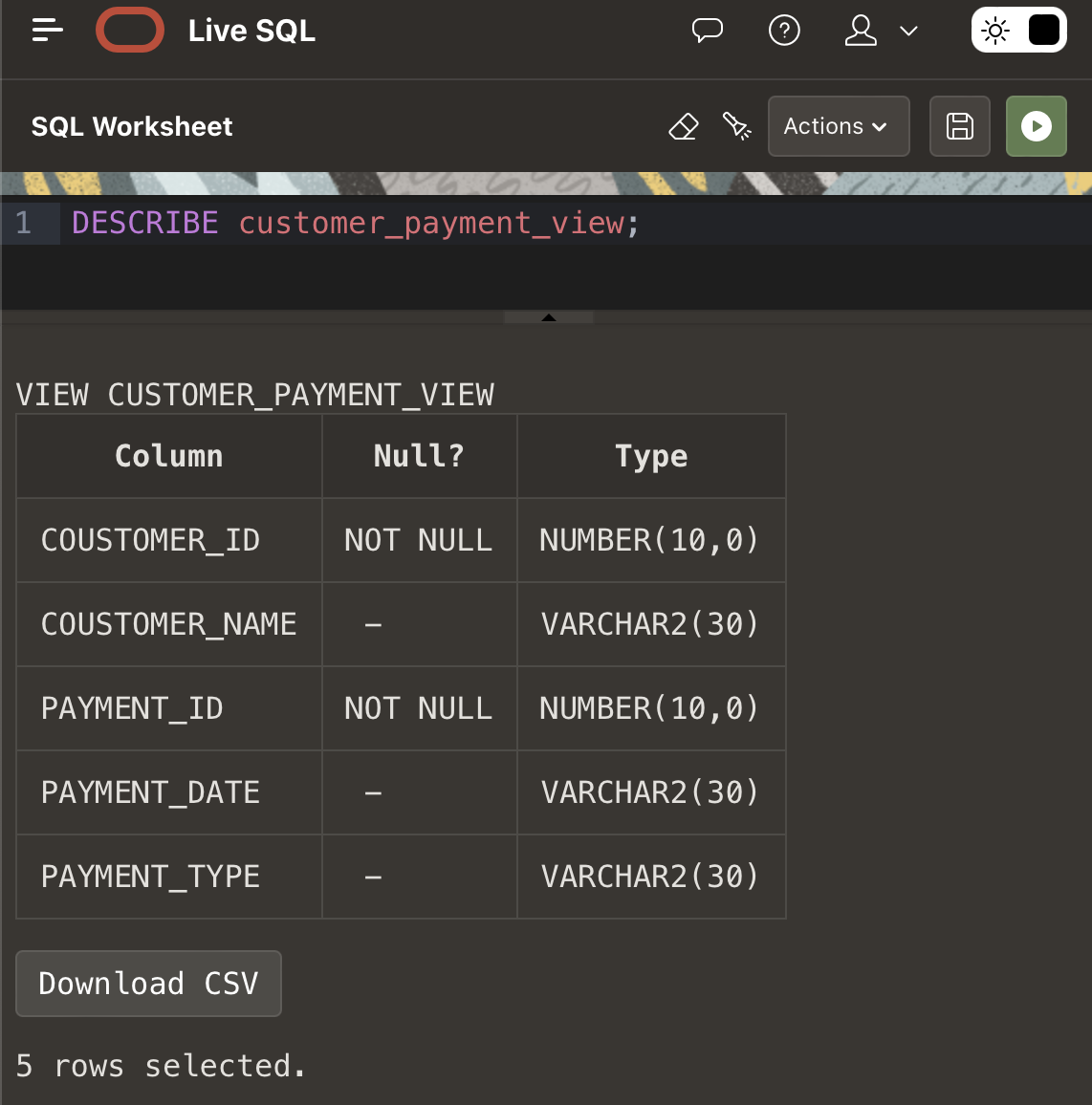


Fig8.12: Description of the complex view.

SELECT \* FROM customer\_payment\_view;

A screenshot of a computer

Description automatically generated

Fig8.13: Result of the complex view as a whole table.

**Conclusion**

I have shown all the queries to create the tables in Live Oracle. Also, I showed the queries to insert the values and tested multiple queries then took screenshots of them. Here, I have established four different relationships between the entities. Due to the normalization process, my work has become easier.