AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

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

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

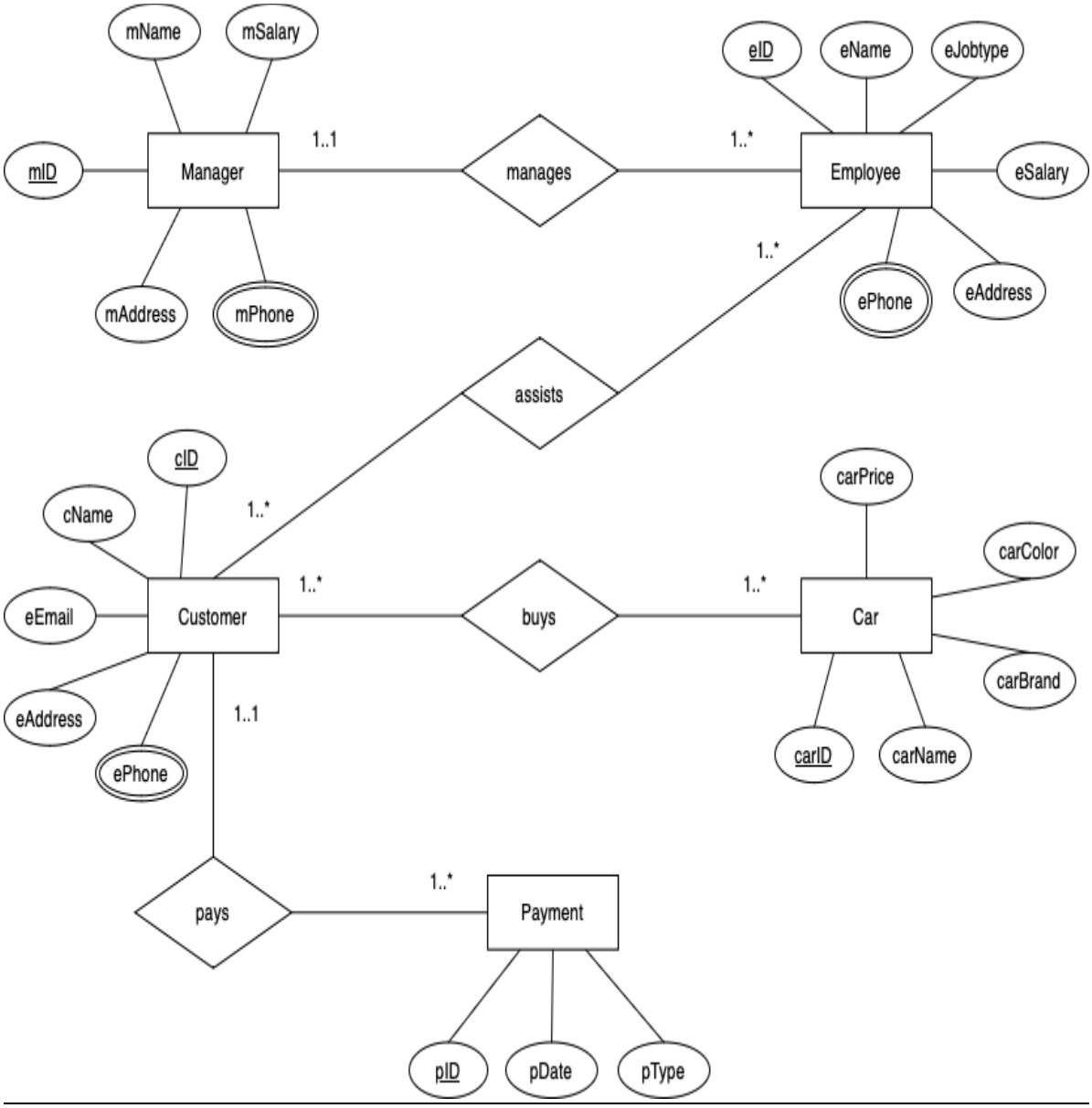


Fig3.1:ER-Diagram

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:**

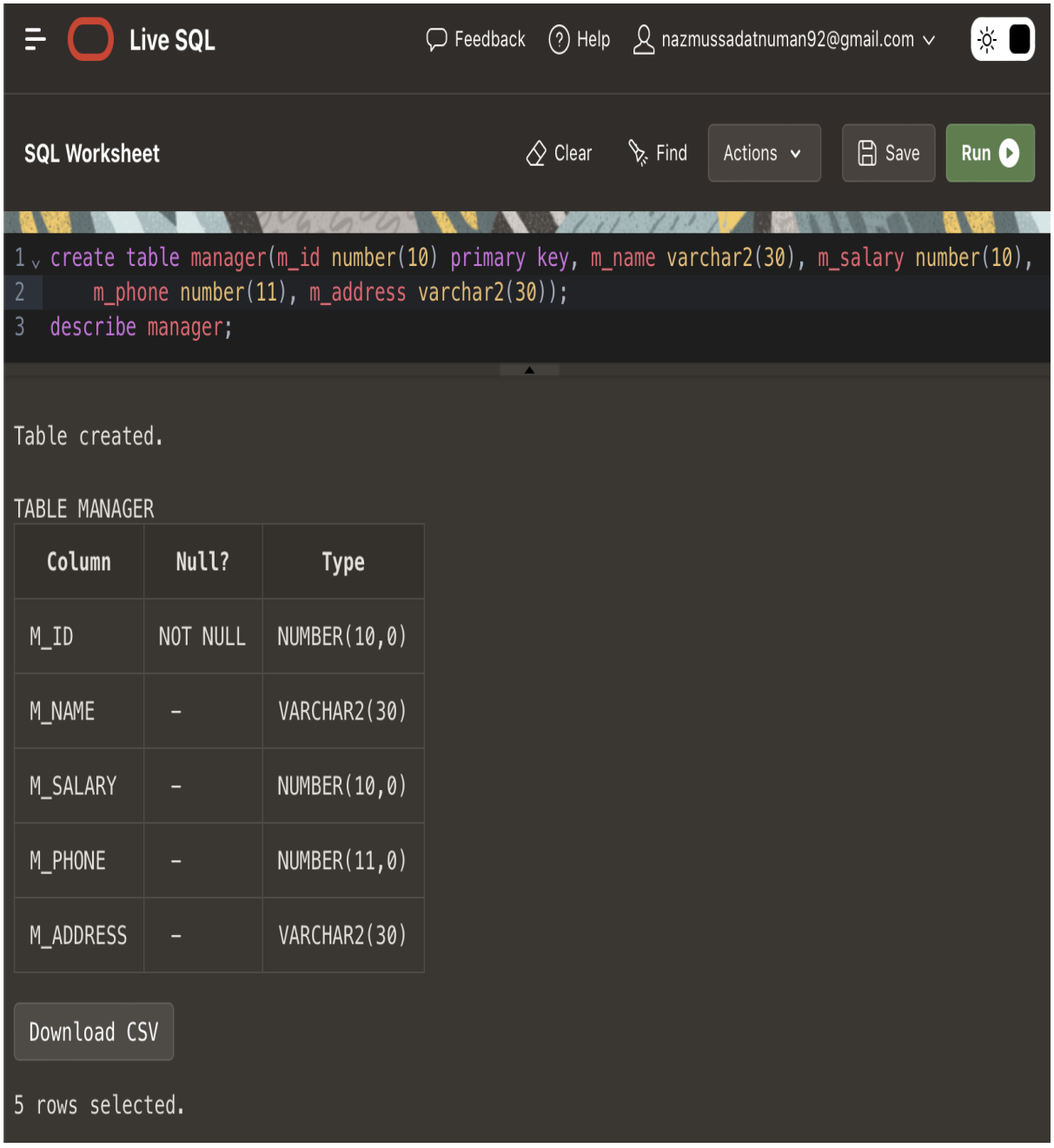


Fig6.1: creation and description of the manager table.

2. Employee Table:

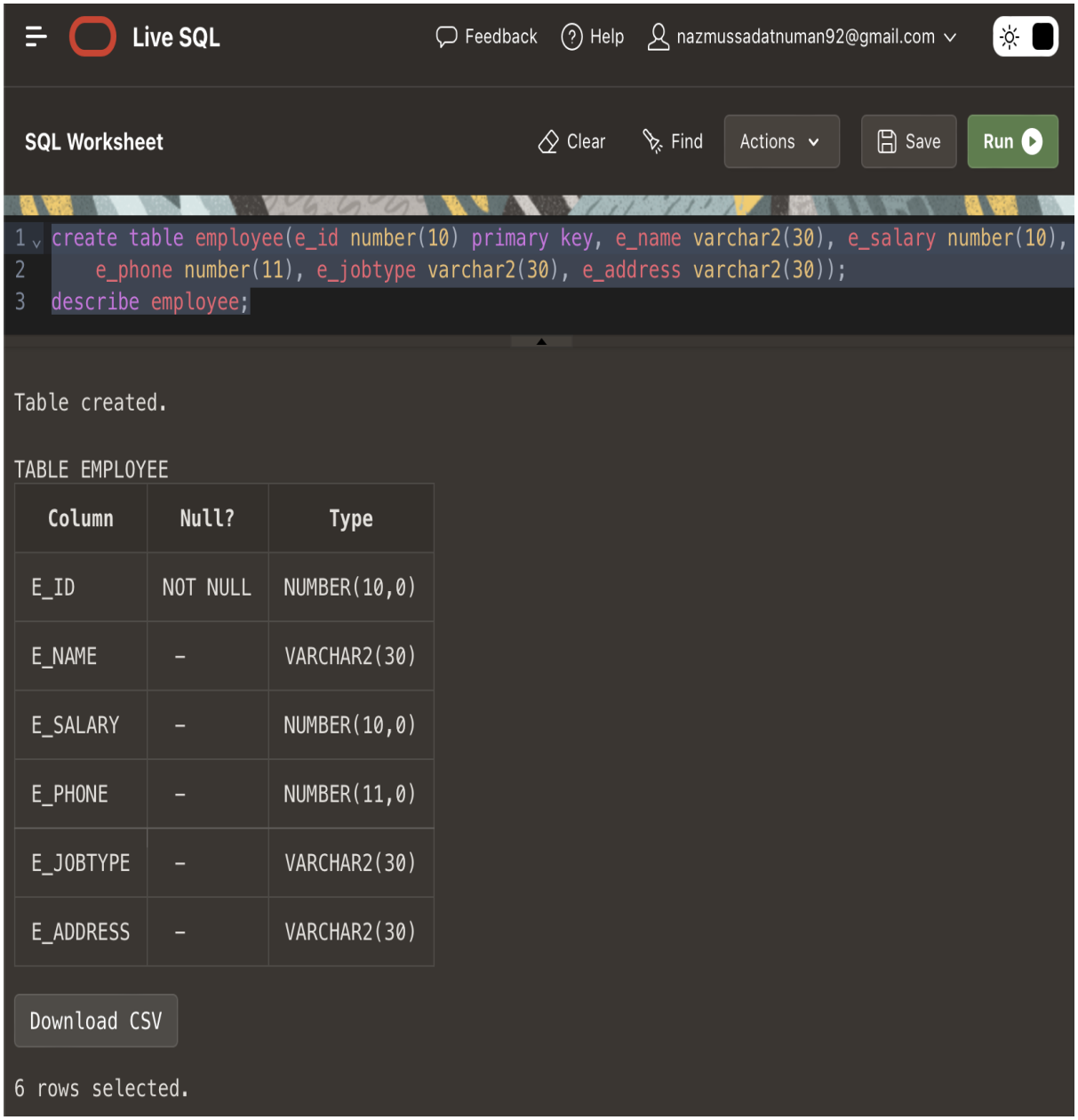


Fig6.2: creation and description of the employee table.

3. **Customer Table:**

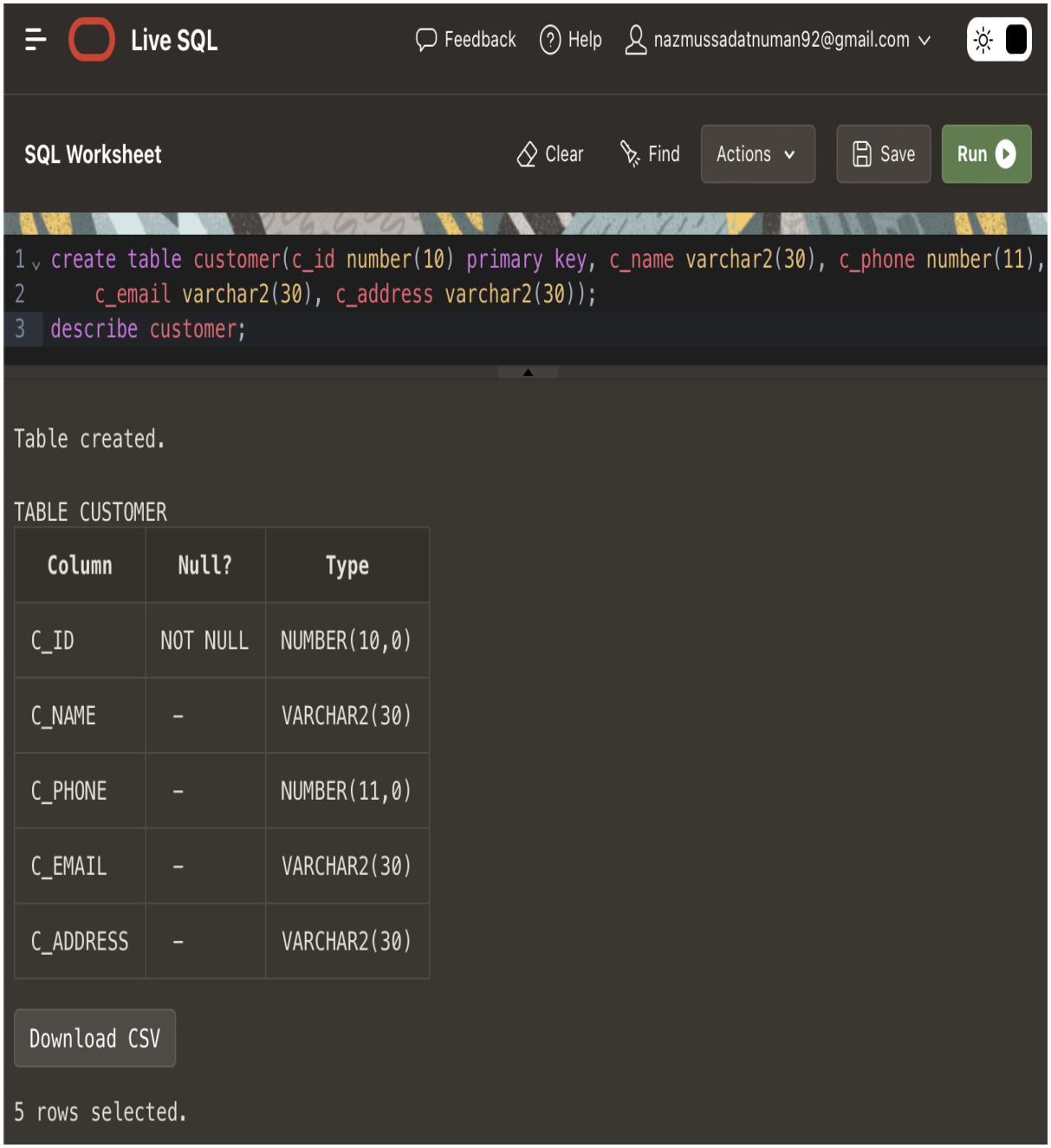


Fig6.3: creation and description of the customer table.

4. Car Table:

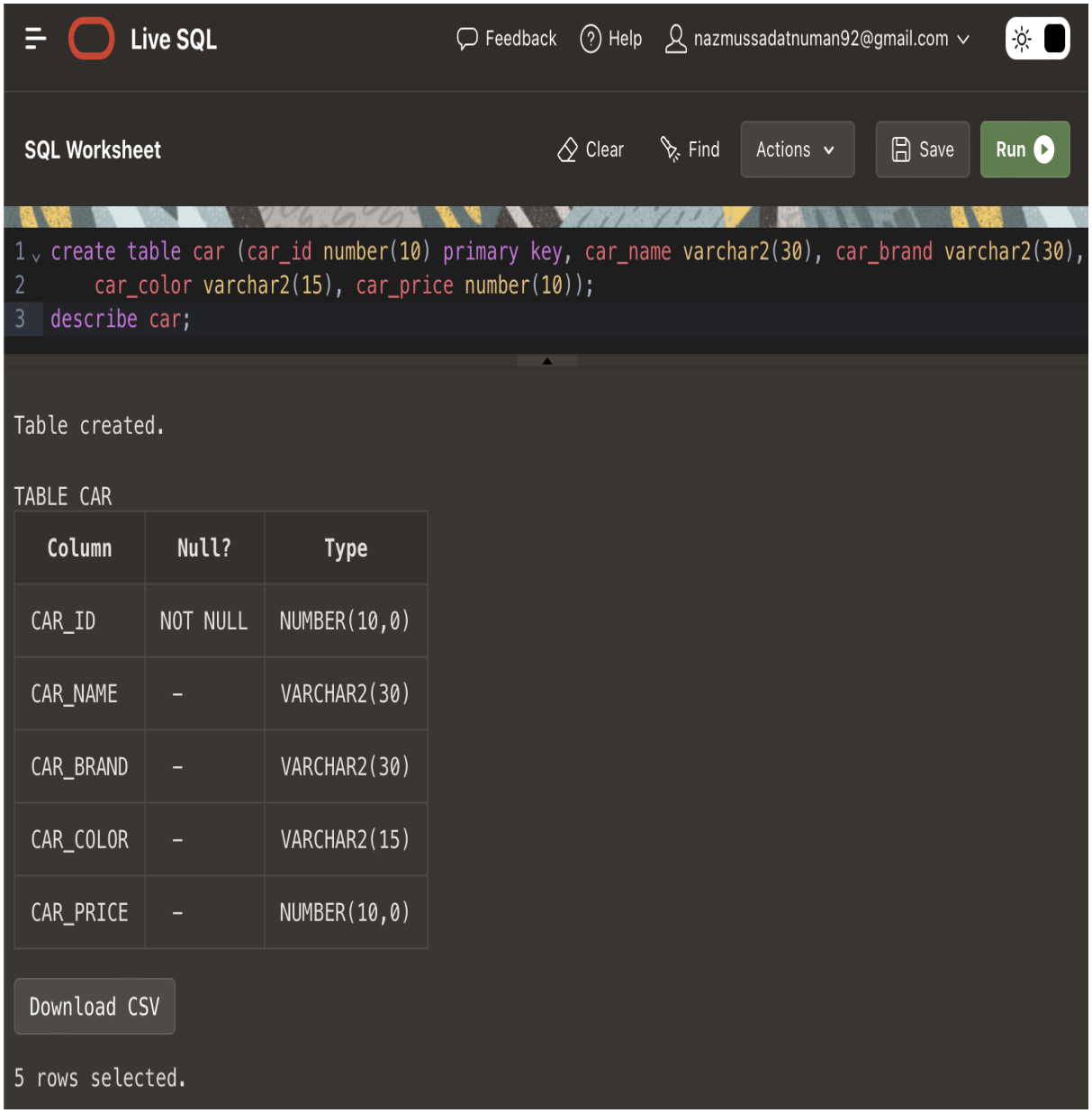


Fig6.4: creation and description of the table.

5. Pay Table:

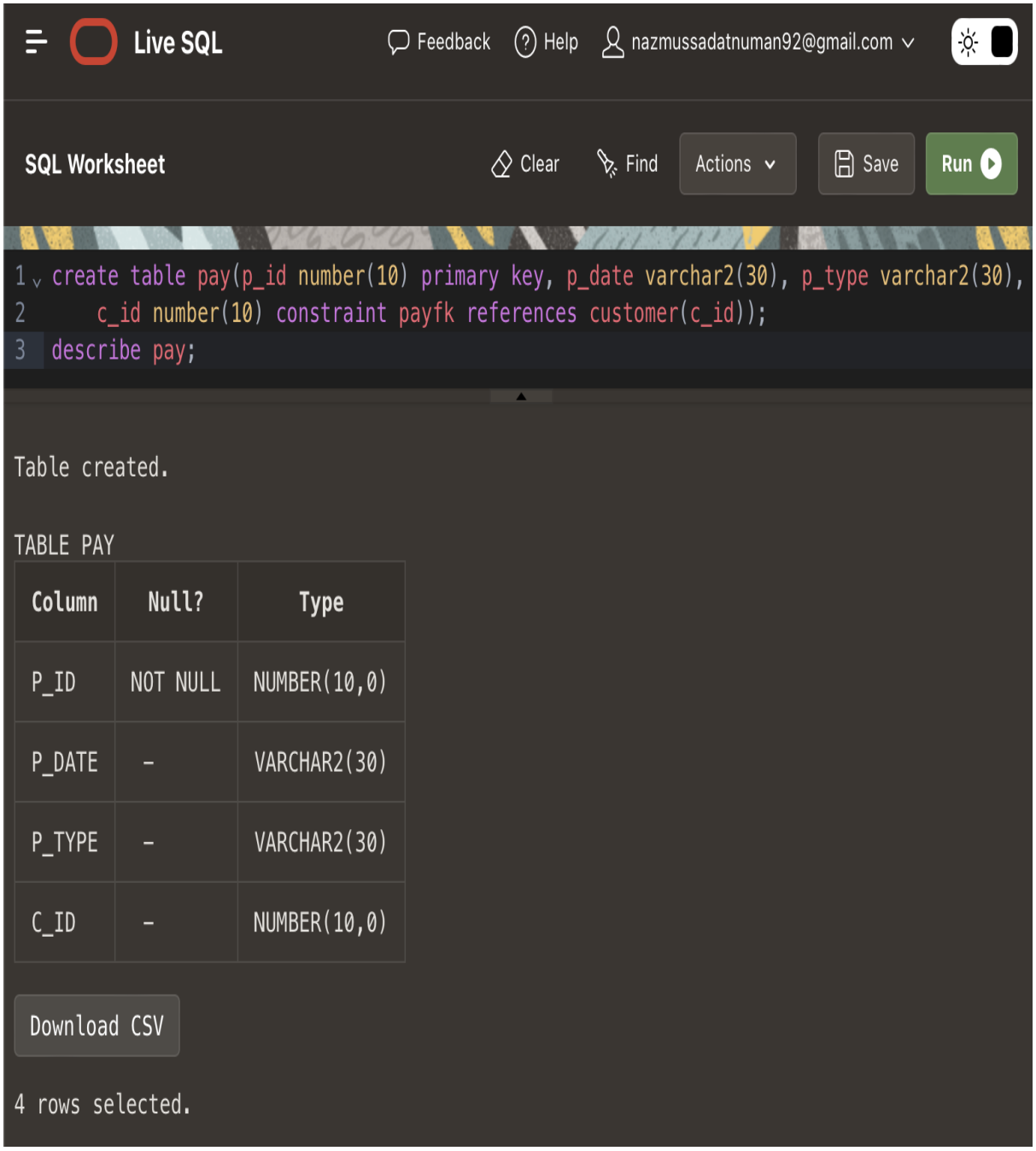


Fig6.5: creation and description of the pay table.

6. Mange Table:

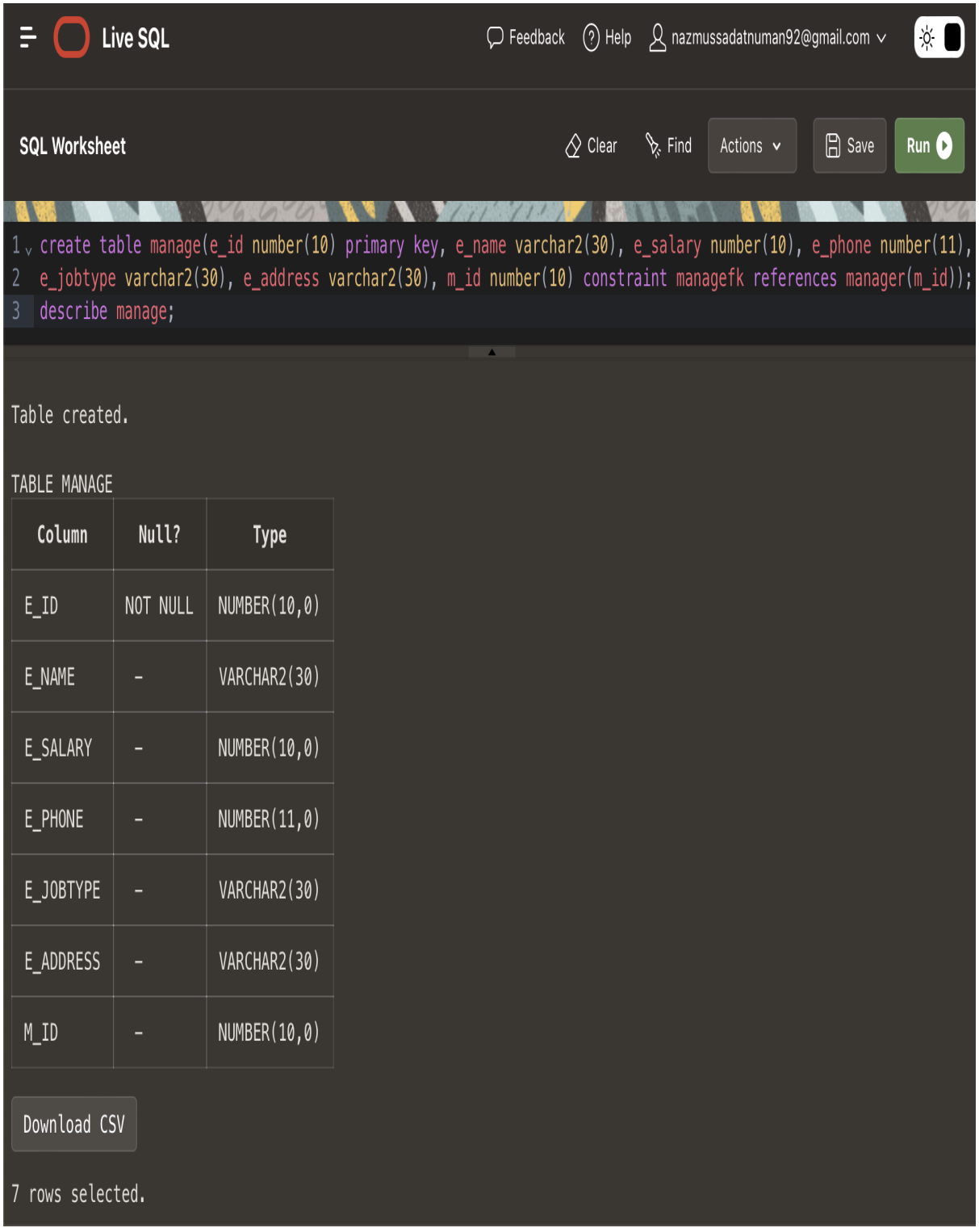


Fig6.6: creation and description of the manage table.

7. Assist Table:

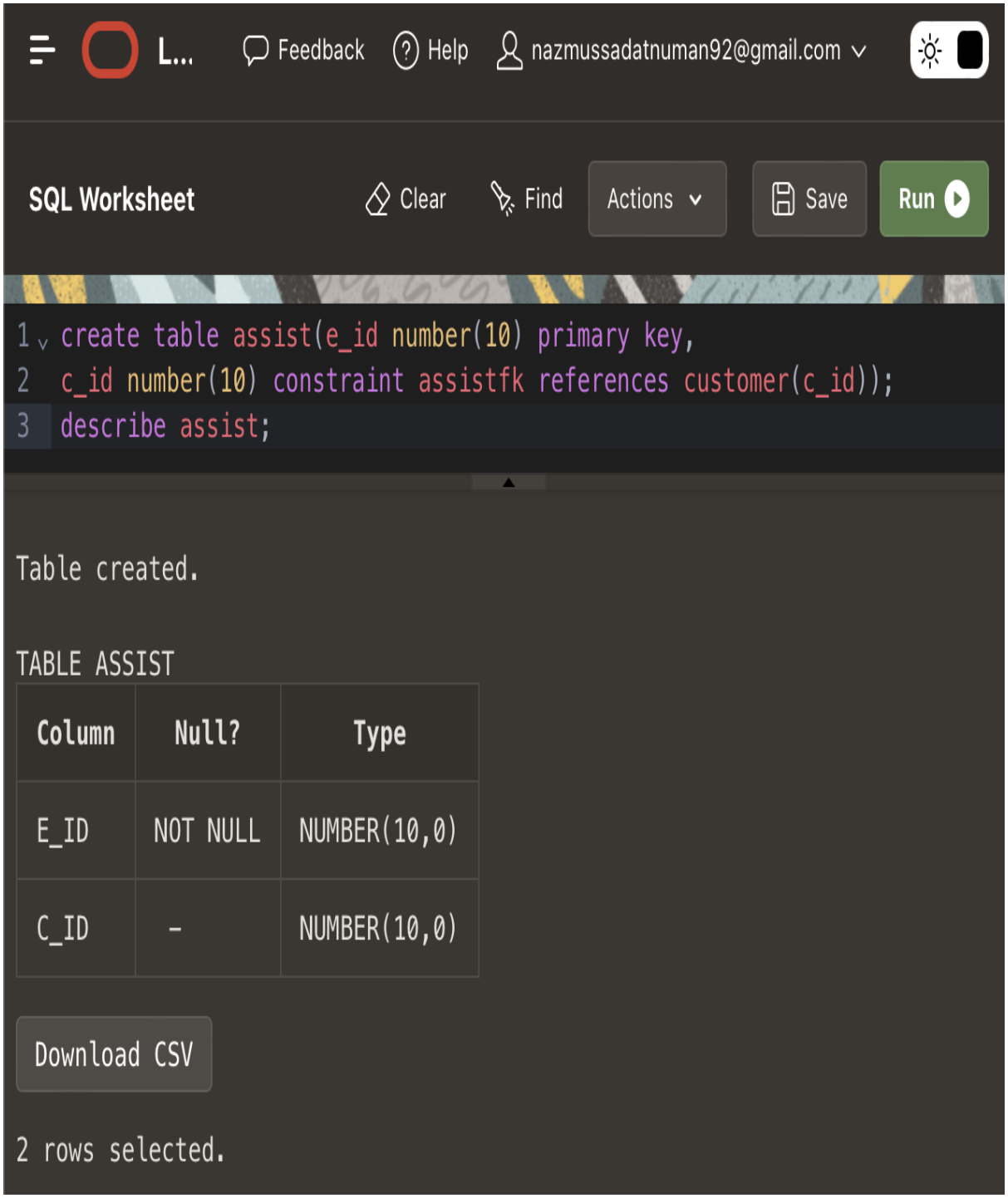


Fig6.7: creation and description of the assist table.

8. Buy Table:

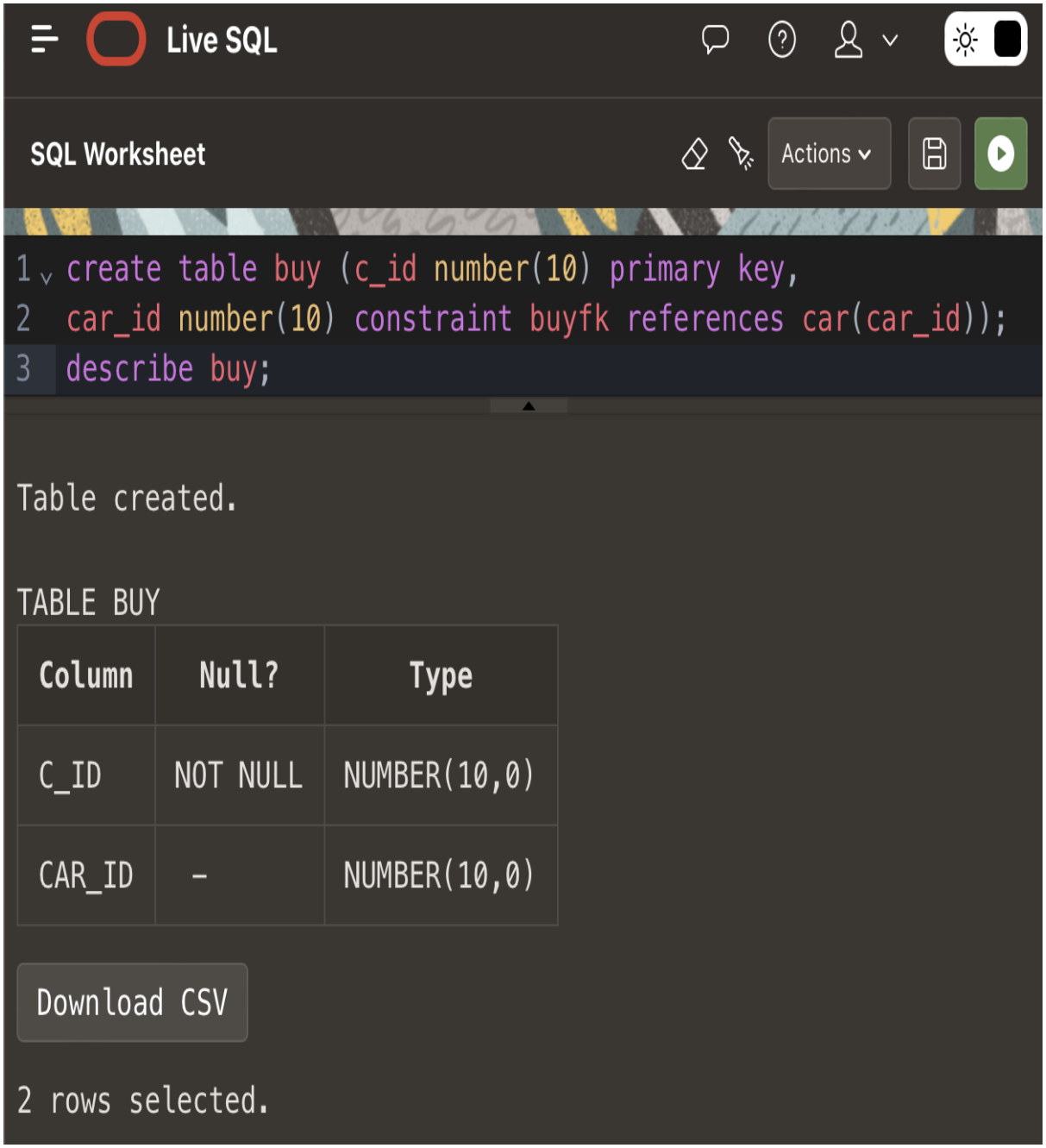


Fig6.8: creation and description of the buy table.

Value Insertion

1.**Manager:**

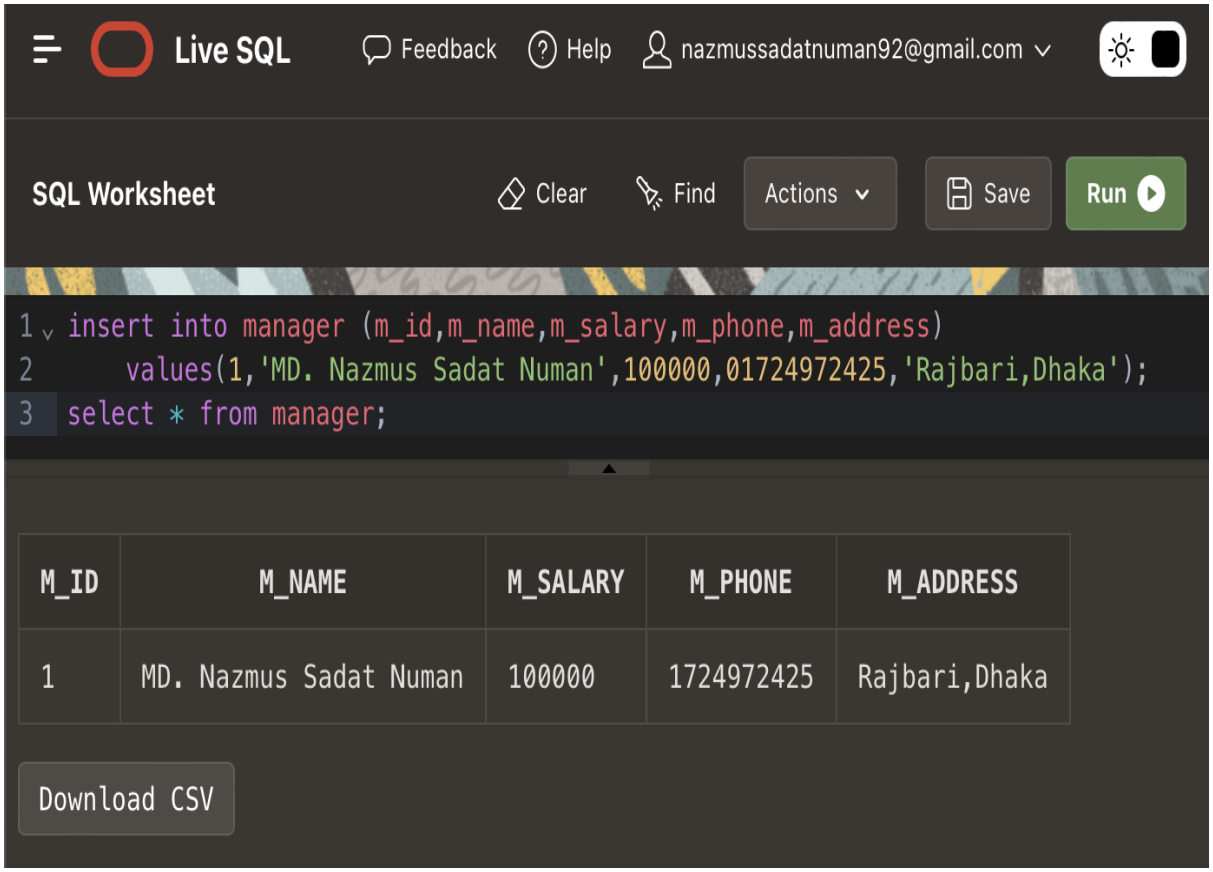


Fig7.1: Input values for the manager table and all its data.

2.**Employee:**

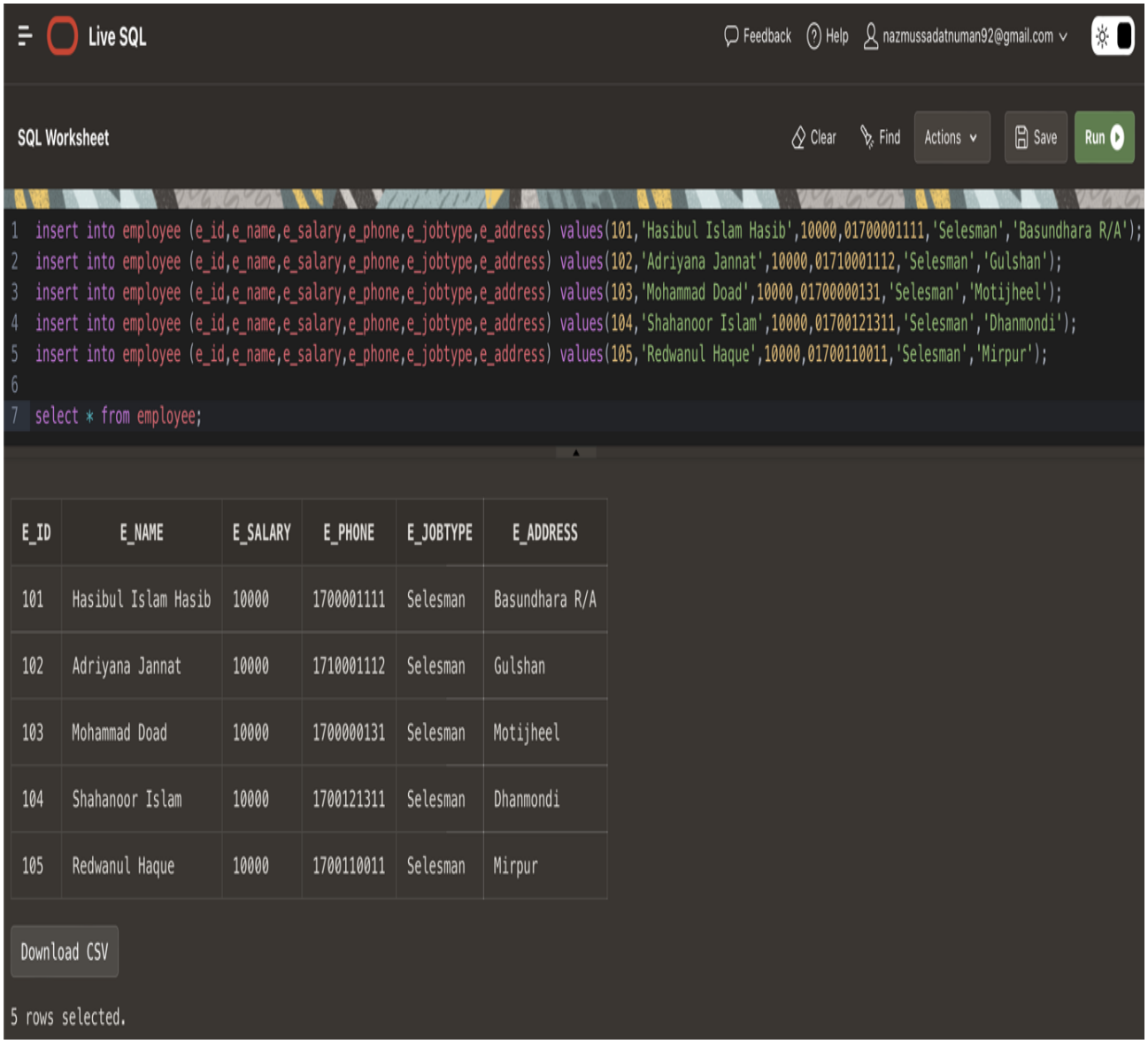


Fig7.2: Input values for the employee table and all its data.

3.**Customer:**

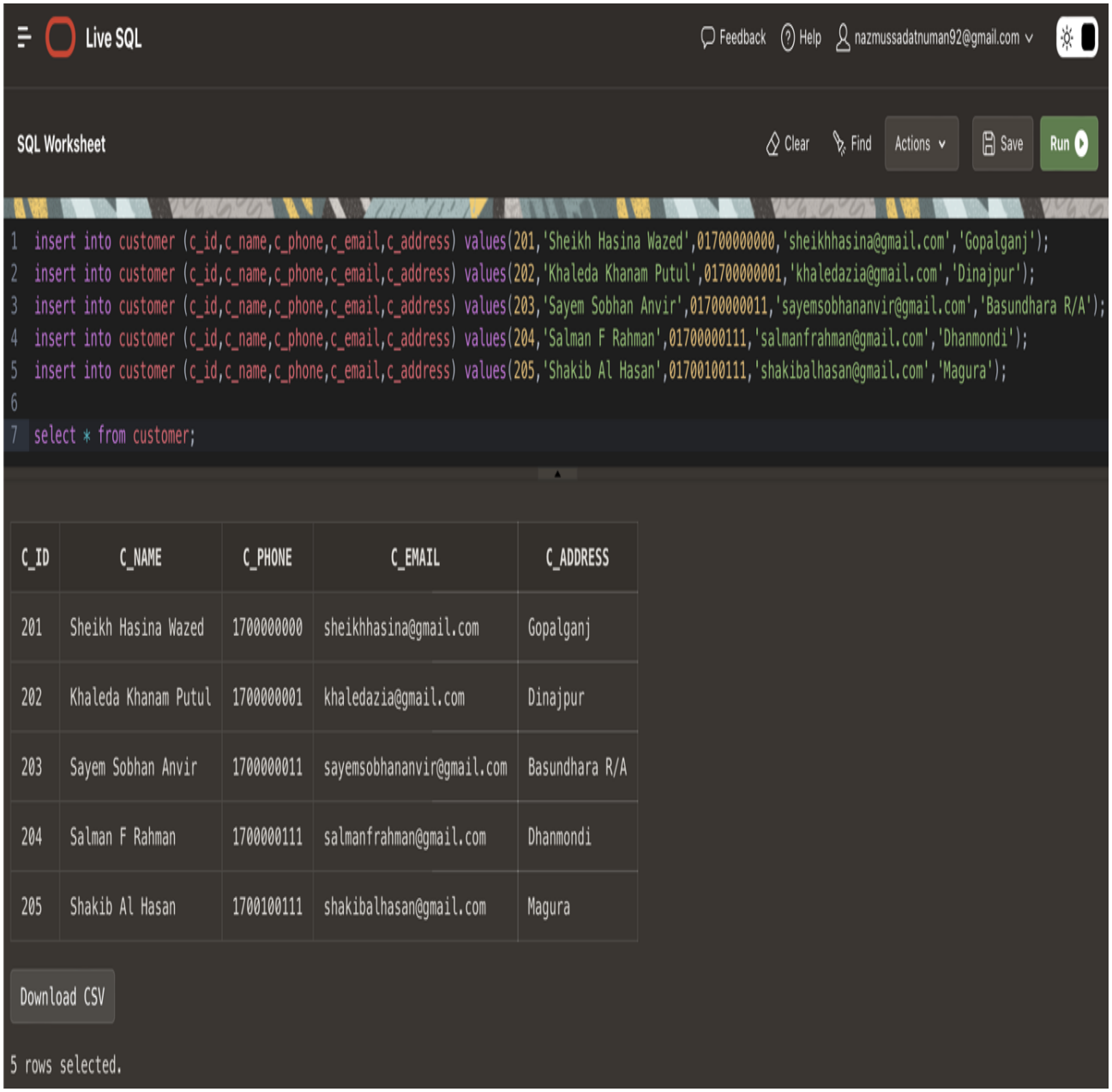


Fig7.3: Input values for the customer table and all its data.

4.**Car:**

Fig7.4: Input values for the car table.

CAR_ID	CAR_NAME	CAR_BRAND	CAR_COLOR	CAR_PRICE
301	Lamborghini Huracán Evo	Lamborghini	Blu Asteria	17519880
302	Rolls-Royce Ghost	Rolls-Royce	Black Sapphire	15894760
303	Bentley Flying Spur	Bentley	Beluga	14582340
304	Mercedes-Maybach S-Class	Mercedes-Maybach	Obsidian Black	13877000
305	Range Rover SVAutobiography	Land Rover	Corris Grey	13757500
306	Ferrari Roma	Ferrari	Rosso Corsa	12873200
307	Porsche Panamera Turbo S	Porsche	Black	12342200
308	Aston Martin DB11 V8	Aston Martin	Quantum Silver	11811160
309	Lexus LS 500h	Lexus	Deep Blue Mica	11280000
310	Quattroporte GranTurismo	Maserati	Nero Ribelle	10848840

Download CSV

10 rows selected.

Fig7.5: All data of car table.

5.**Pay:**

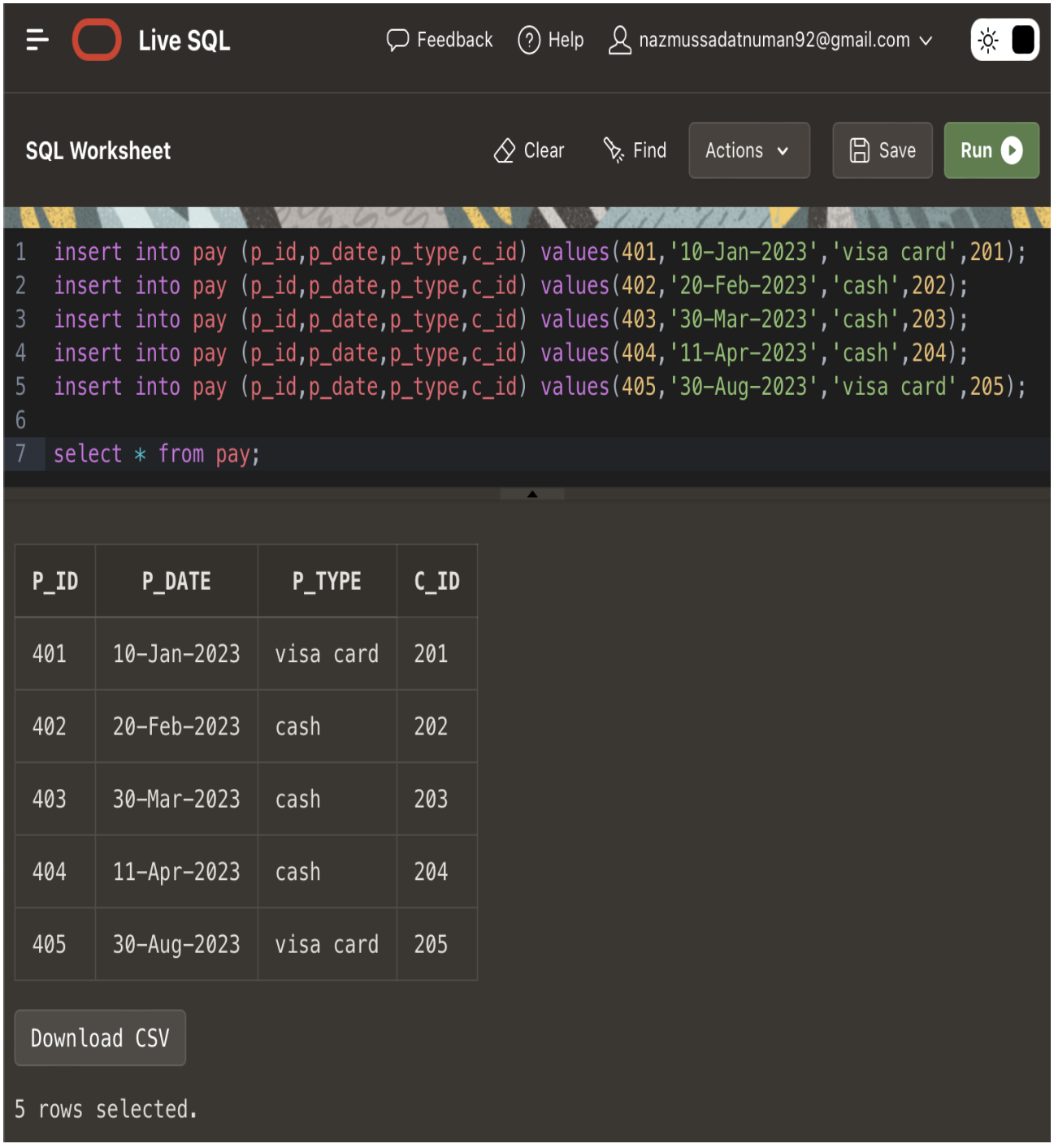


Fig7.6: Input values for the pay table and all its data.

6.**Manage:**

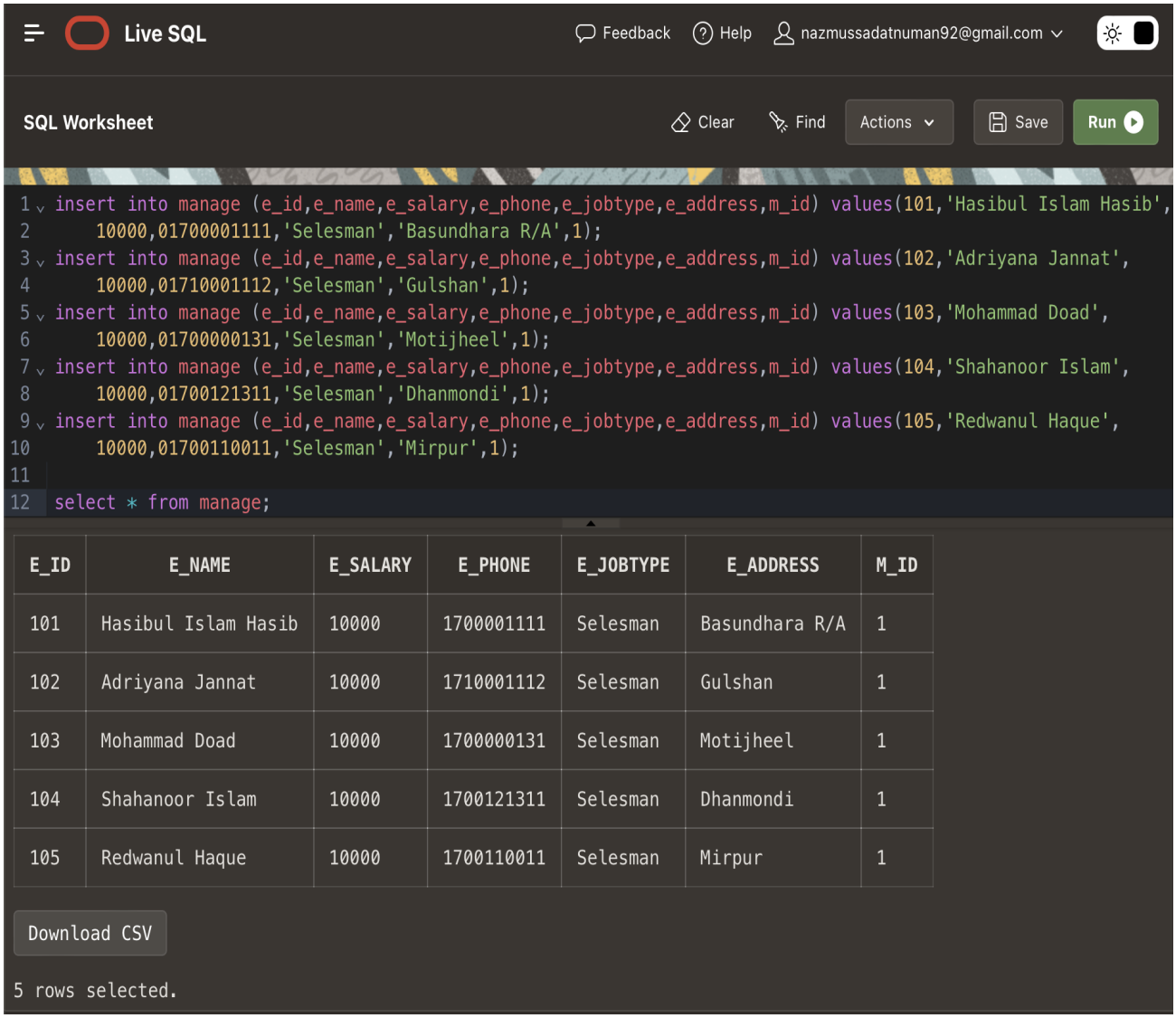


Fig7.7: Input values for the manage table and all its data.

7.**Assist:**

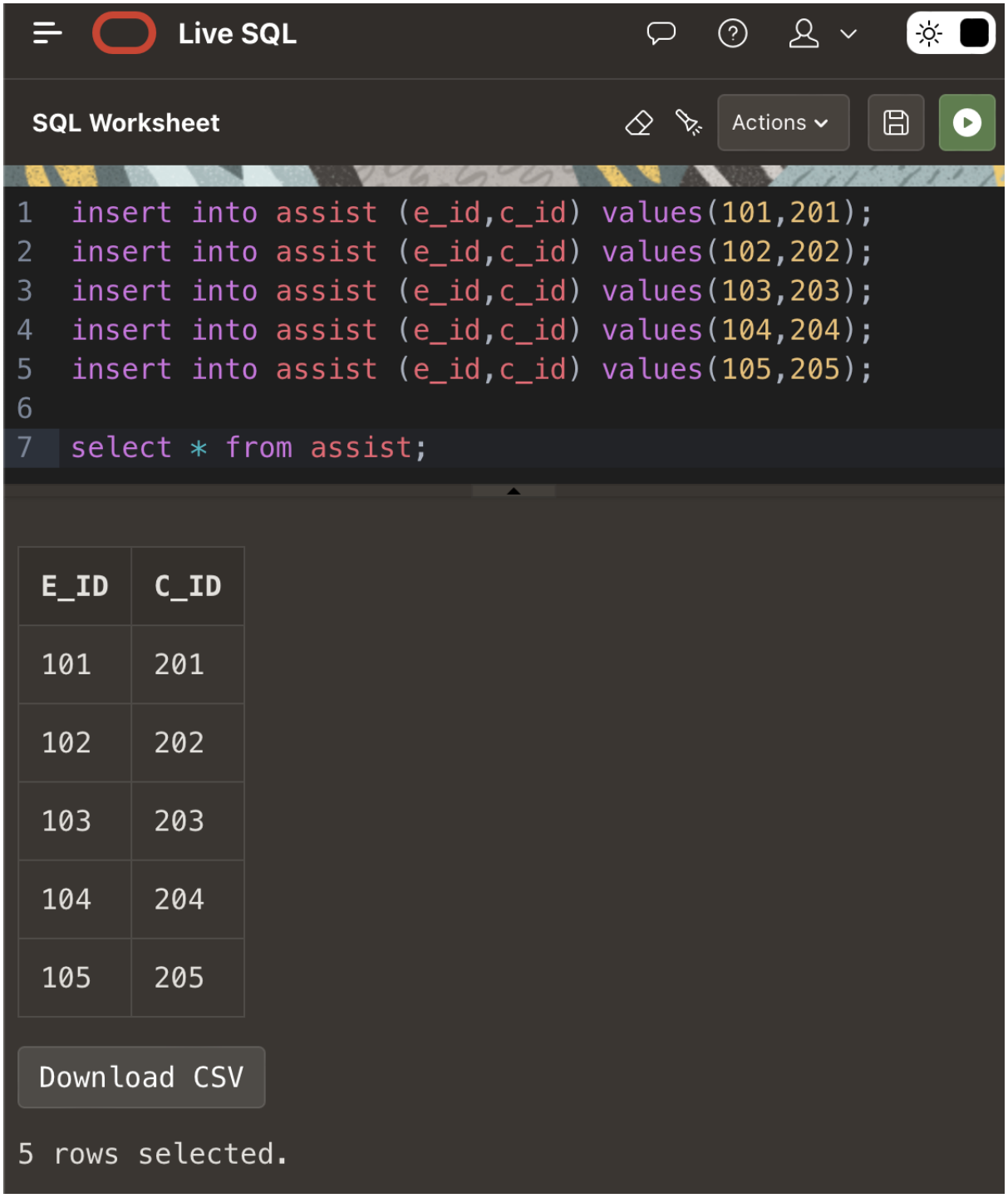


Fig7.8: Input values for the assist table and all its data.

8.**Buy:**

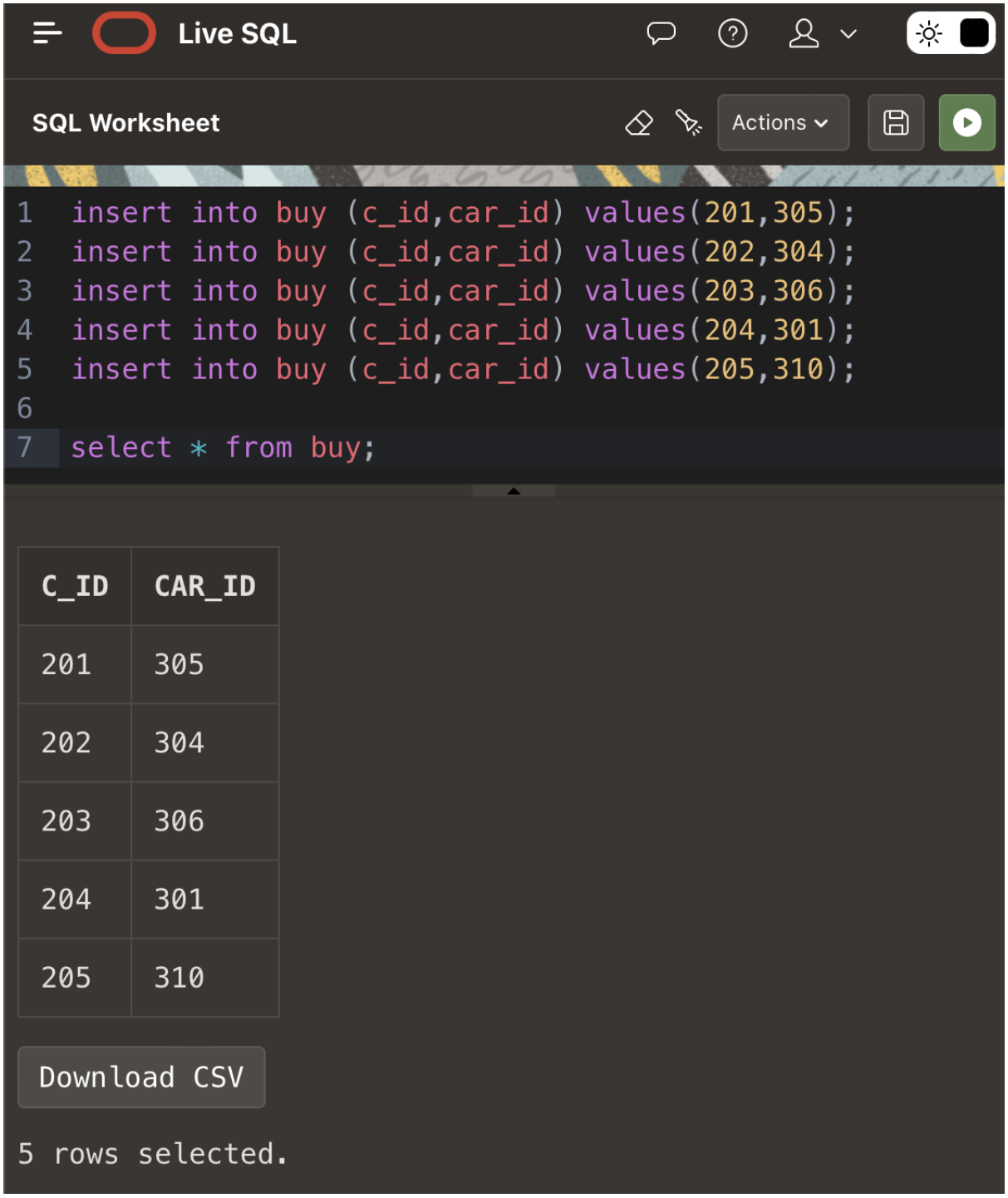


Fig7.9: Input values for the buy table and all its data.

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;

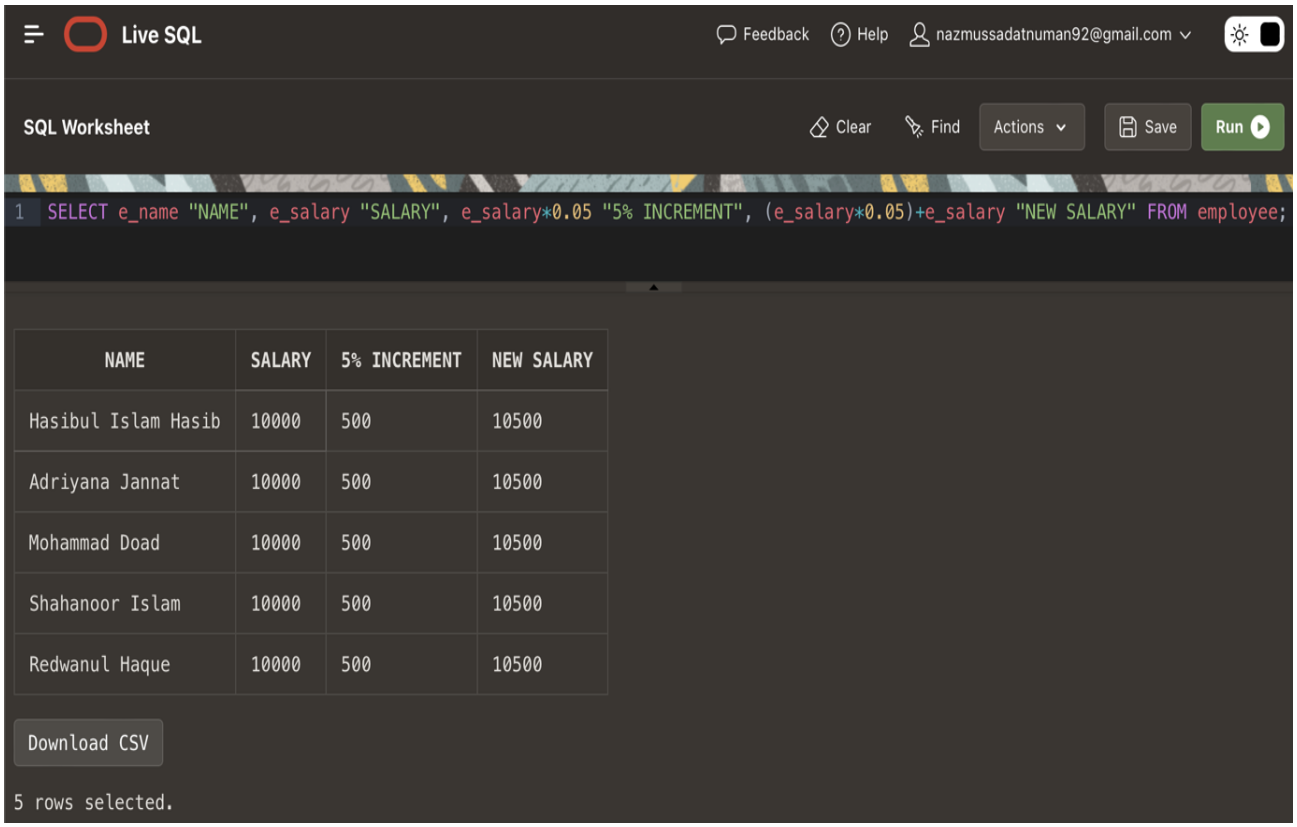


Fig8.1: Simple Query command and full table as a result

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;

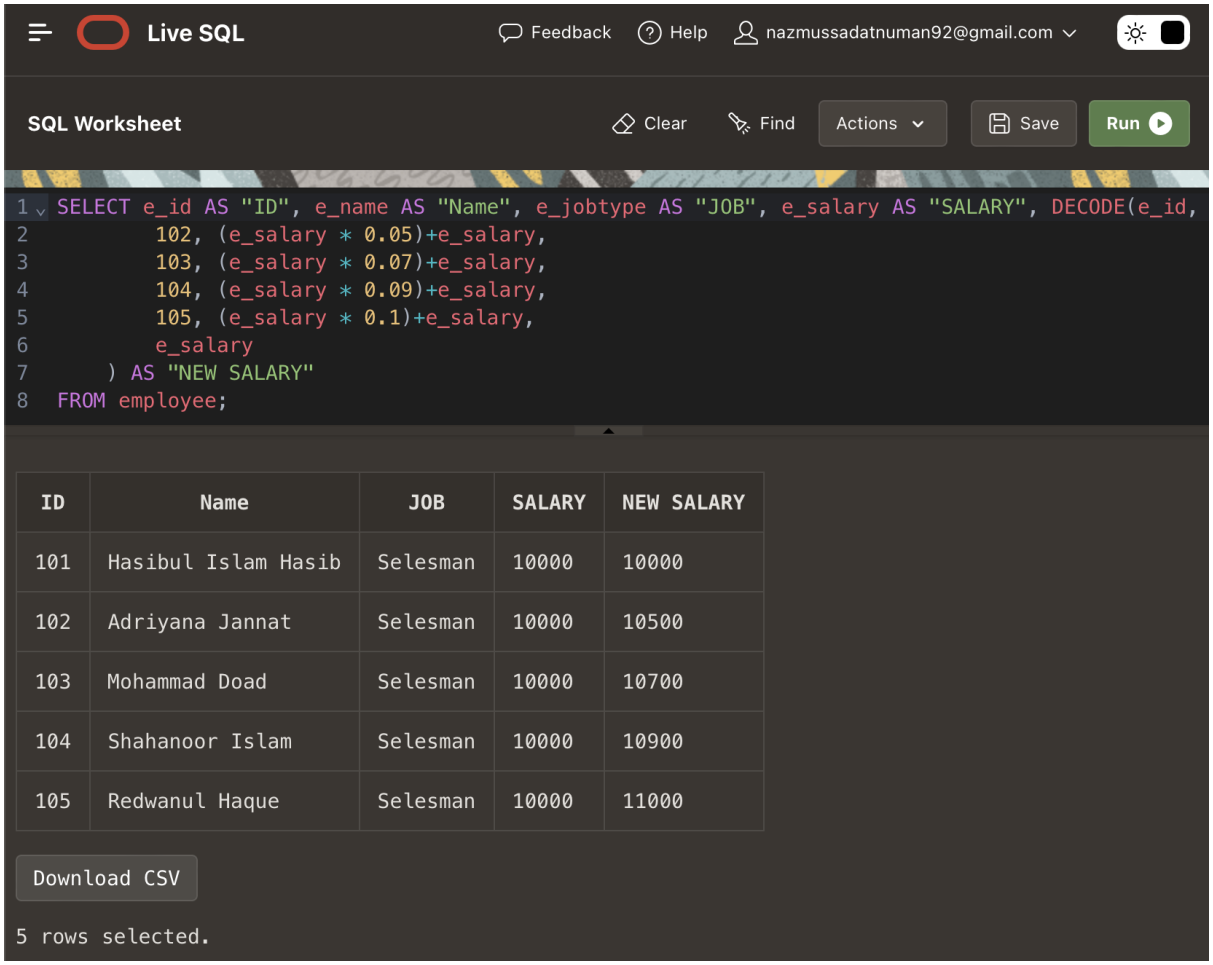


Fig8.2: Single row function command and full table as a result.

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;

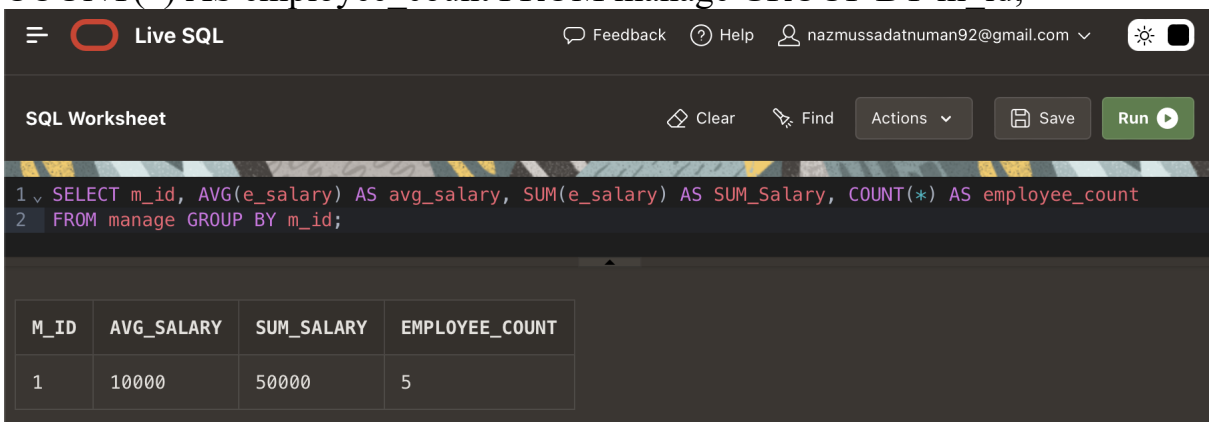


Fig8.3: Multiple row function command and full table as a result.

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);

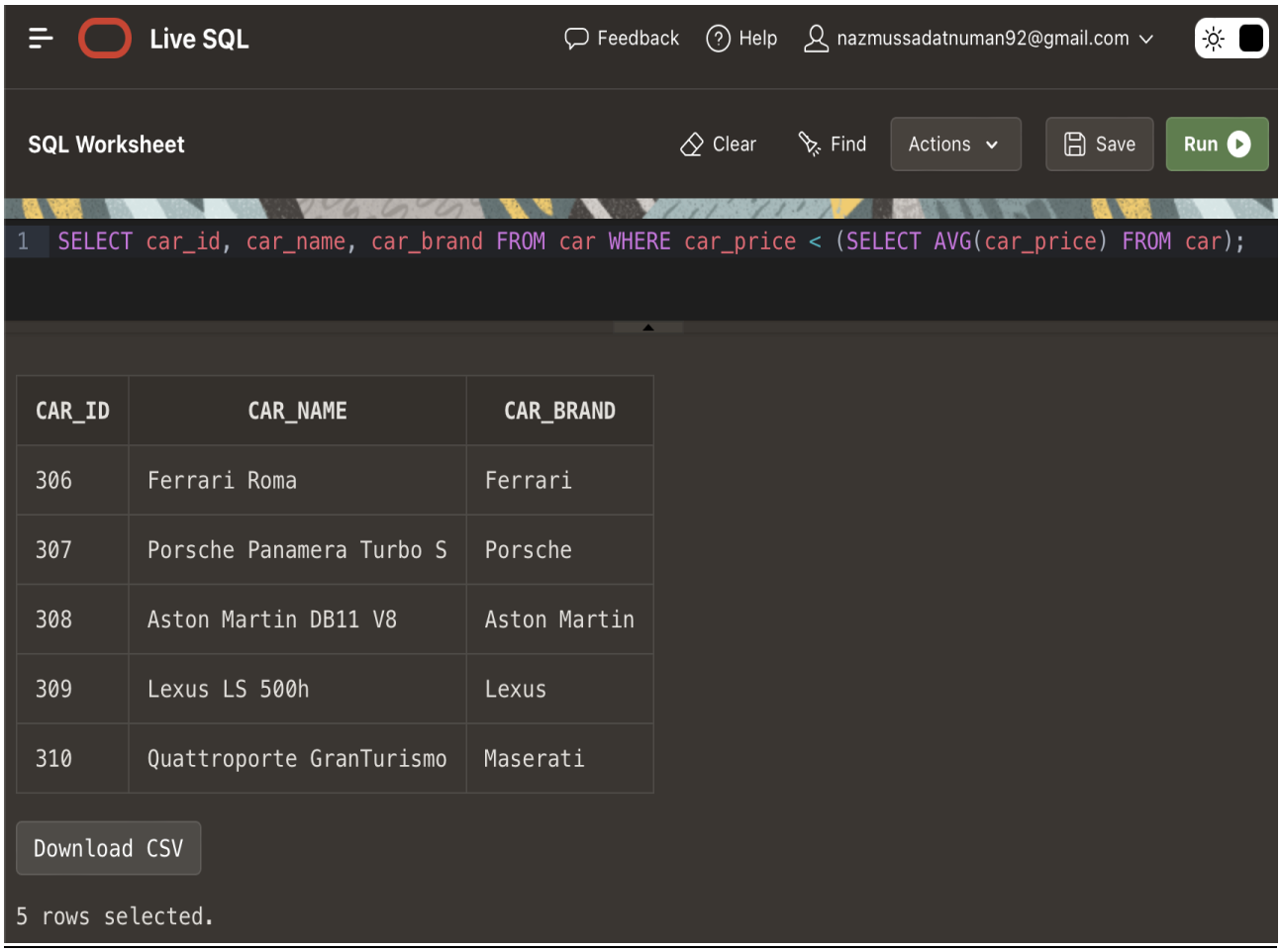


Fig8.4: Single row subquery command and full table as a result.

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'));

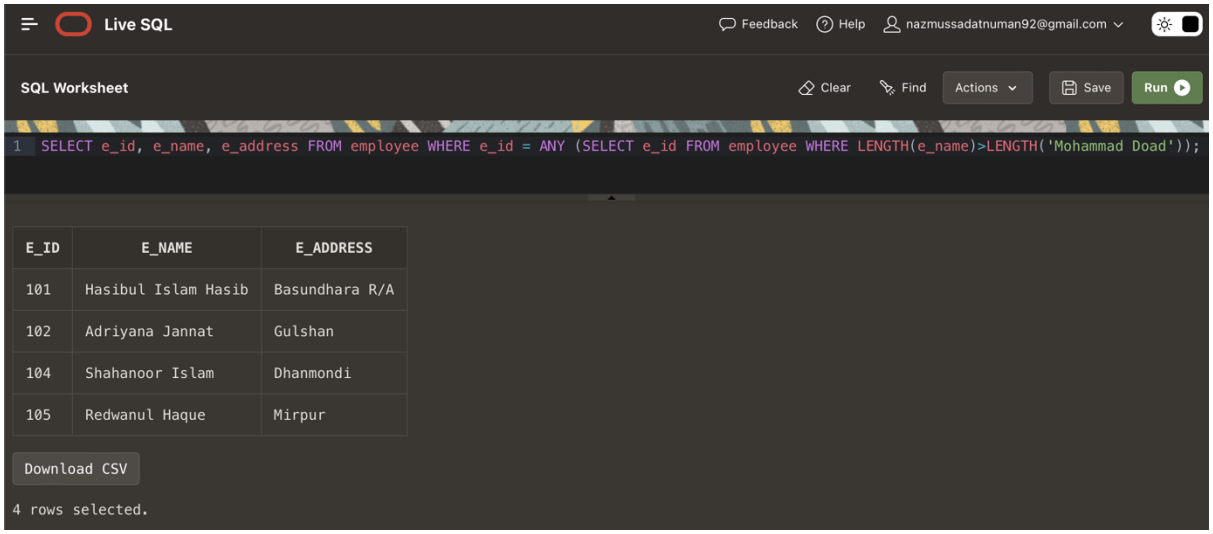


Fig8.5: Multiple row subquery command and full table as a result.

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(+);

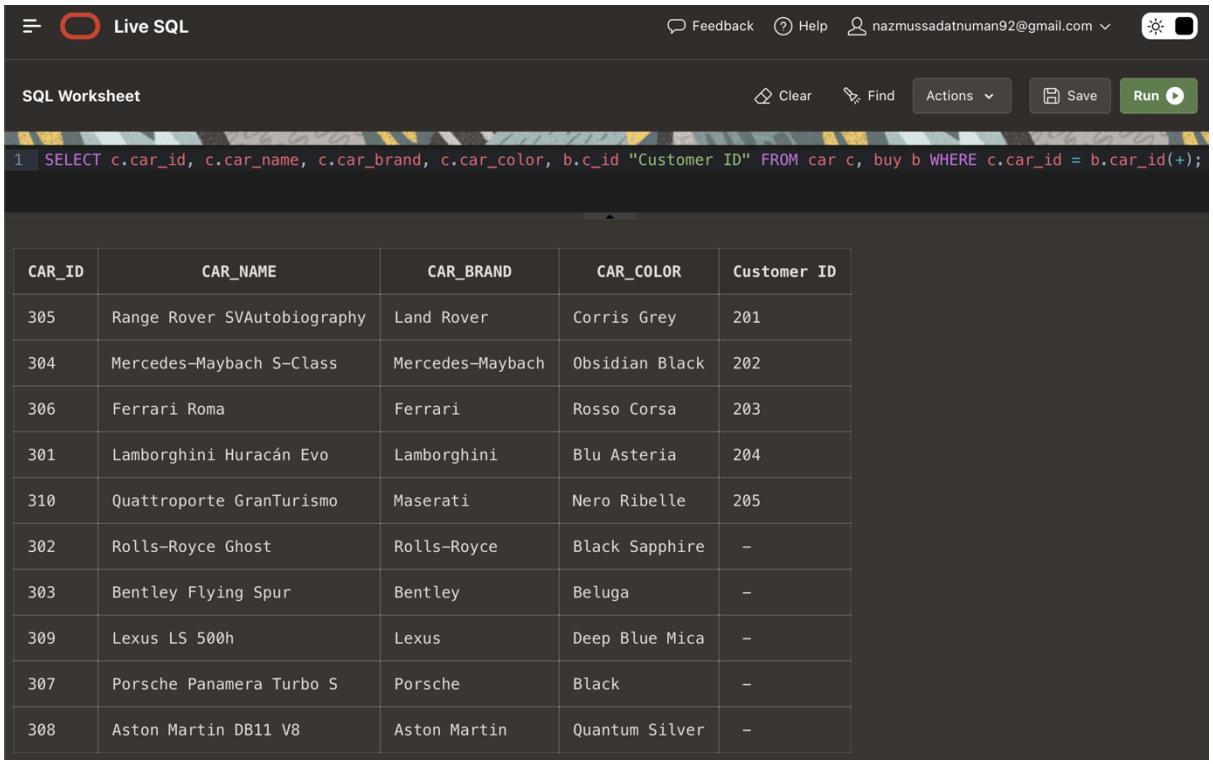
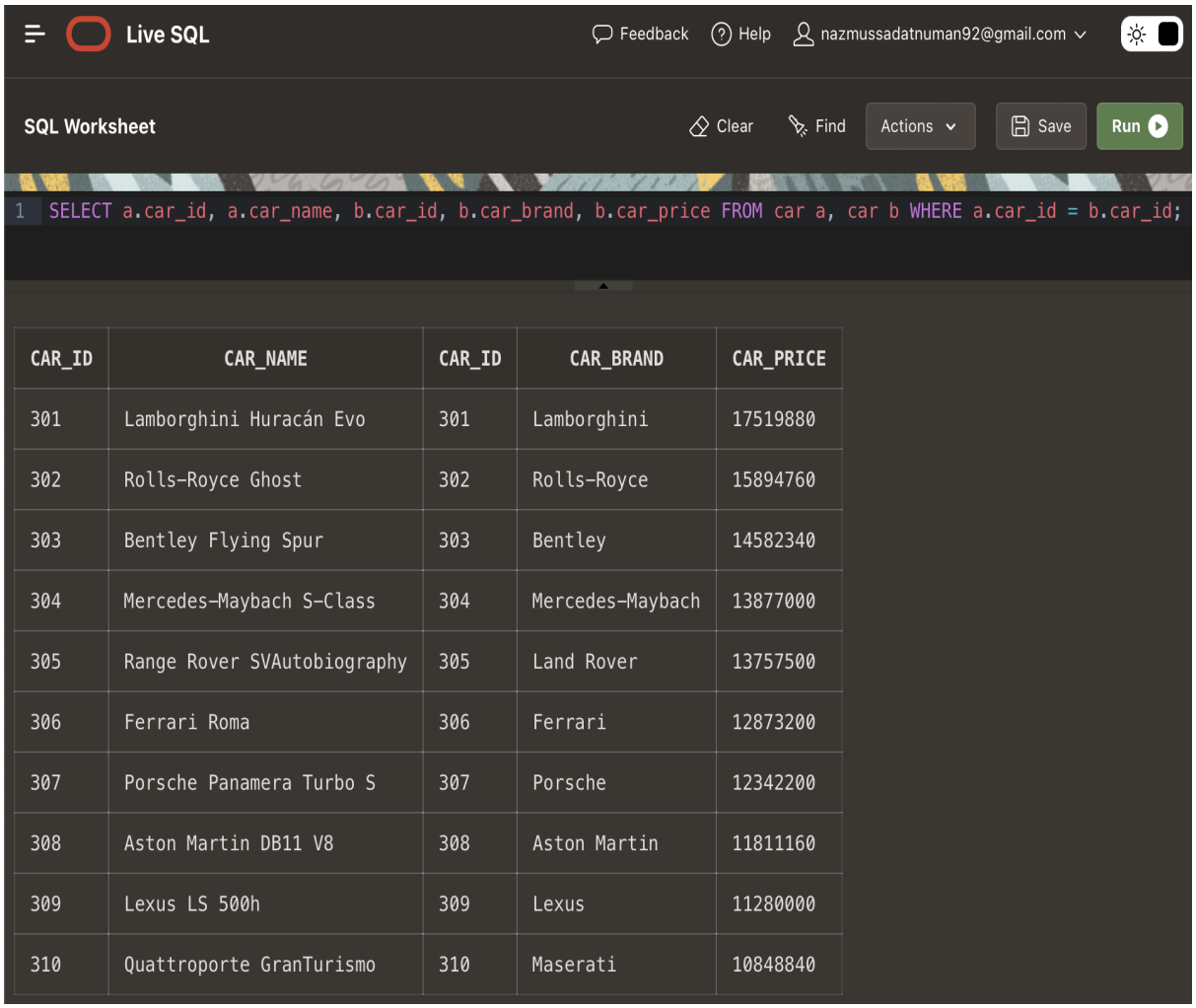


Fig8.6: Right outer join creation command and full table as a result.

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;



CAR_ID	CAR_NAME	CAR_ID	CAR_BRAND	CAR_PRICE
301	Lamborghini Huracán Evo	301	Lamborghini	17519880
302	Rolls-Royce Ghost	302	Rolls-Royce	15894760
303	Bentley Flying Spur	303	Bentley	14582340
304	Mercedes-Maybach S-Class	304	Mercedes-Maybach	13877000
305	Range Rover SVAutobiography	305	Land Rover	13757500
306	Ferrari Roma	306	Ferrari	12873200
307	Porsche Panamera Turbo S	307	Porsche	12342200
308	Aston Martin DB11 V8	308	Aston Martin	11811160
309	Lexus LS 500h	309	Lexus	11280000
310	Quattroporte GranTurismo	310	Maserati	10848840

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;

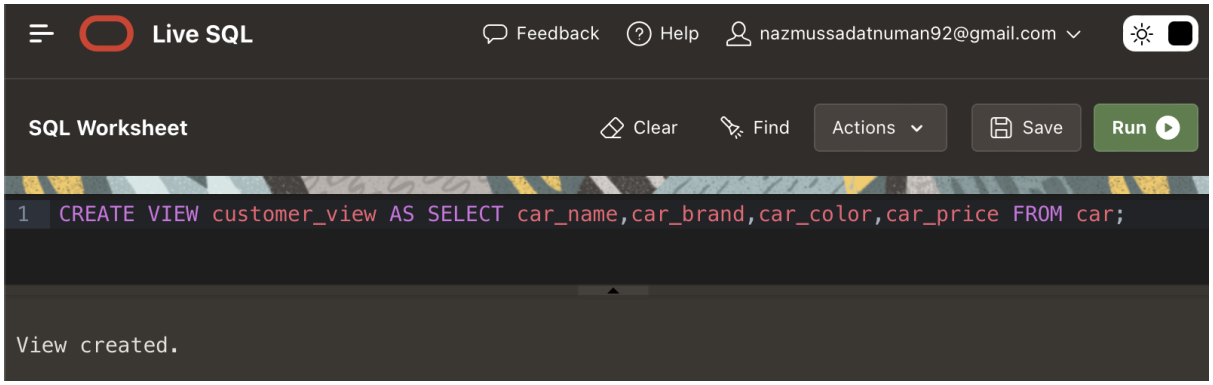


Fig8.8: Simple view creation command.

DESCRIBE customer_view;

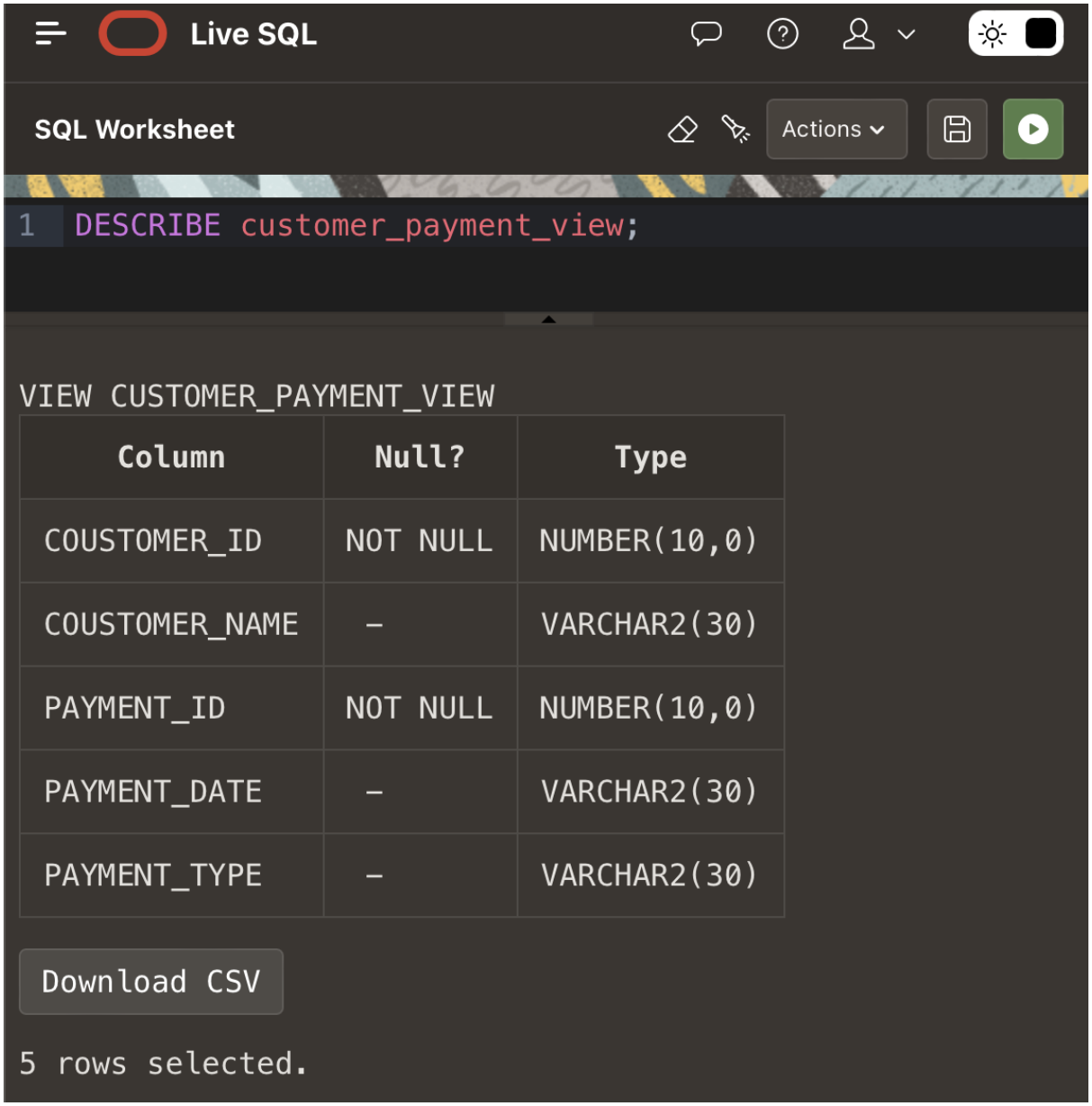
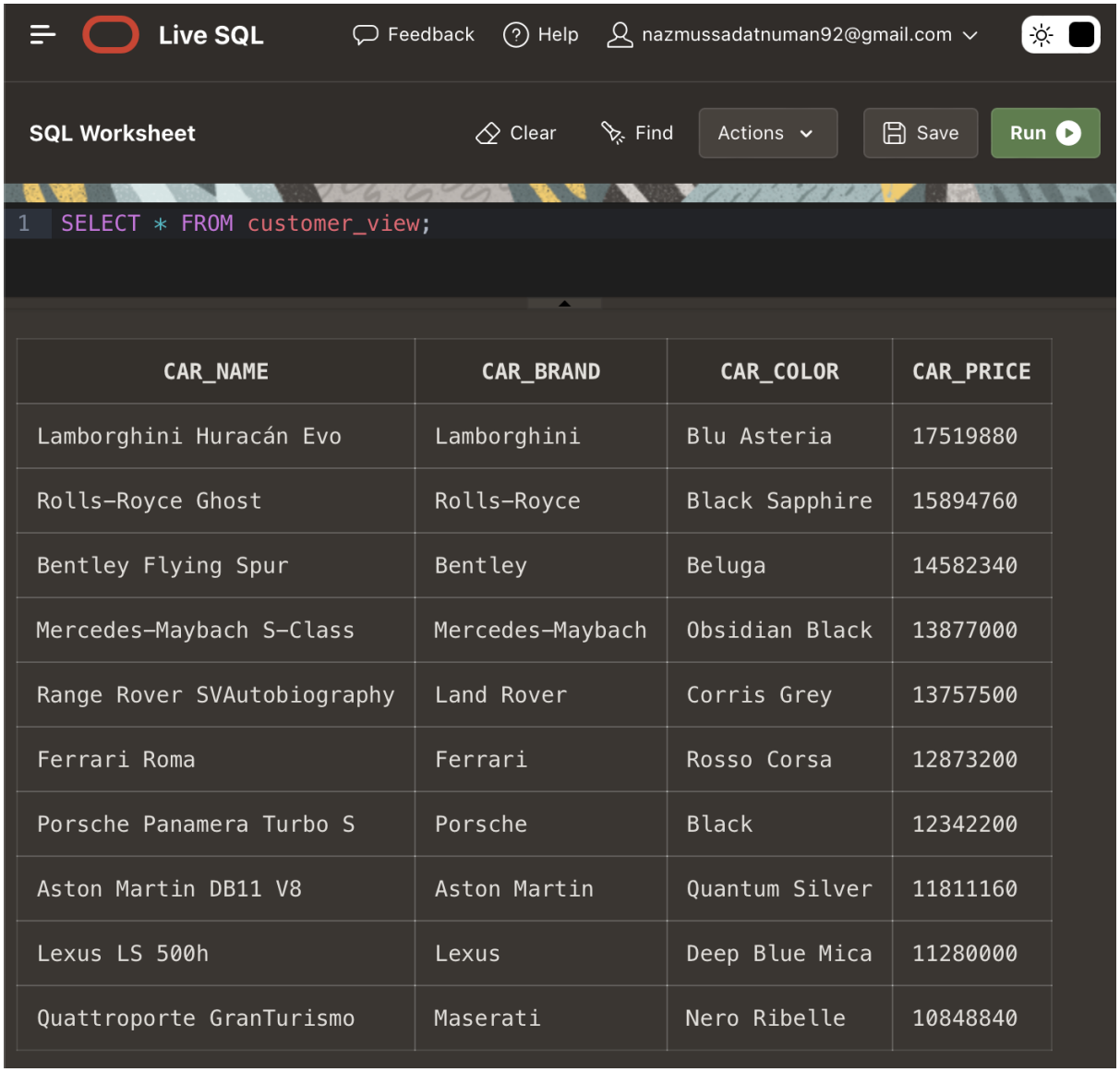


Fig8.9: Description of the simple view.

SELECT * FROM customer_view;



CAR_NAME	CAR_BRAND	CAR_COLOR	CAR_PRICE
Lamborghini Huracán Evo	Lamborghini	Blu Asteria	17519880
Rolls-Royce Ghost	Rolls-Royce	Black Sapphire	15894760
Bentley Flying Spur	Bentley	Beluga	14582340
Mercedes-Maybach S-Class	Mercedes-Maybach	Obsidian Black	13877000
Range Rover SVAutobiography	Land Rover	Corris Grey	13757500
Ferrari Roma	Ferrari	Rosso Corsa	12873200
Porsche Panamera Turbo S	Porsche	Black	12342200
Aston Martin DB11 V8	Aston Martin	Quantum Silver	11811160
Lexus LS 500h	Lexus	Deep Blue Mica	11280000
Quattroporte GranTurismo	Maserati	Nero Ribelle	10848840

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;

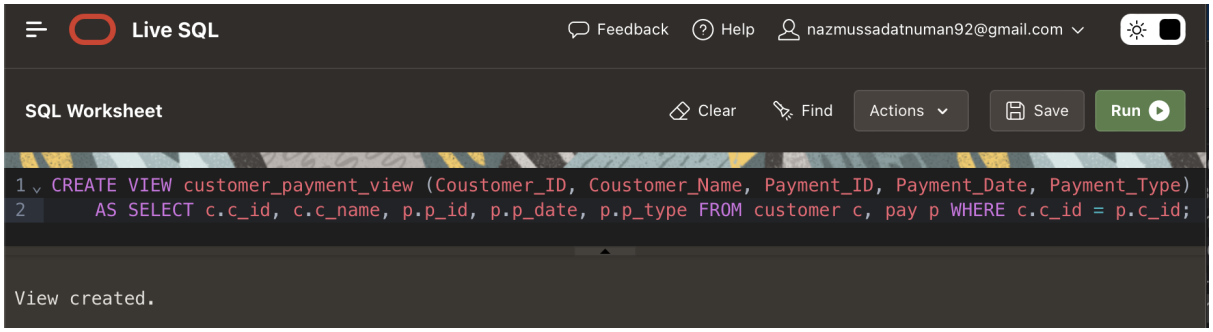


Fig8.11: Complex view creation command.

DESCRIBE customer_payment_view;

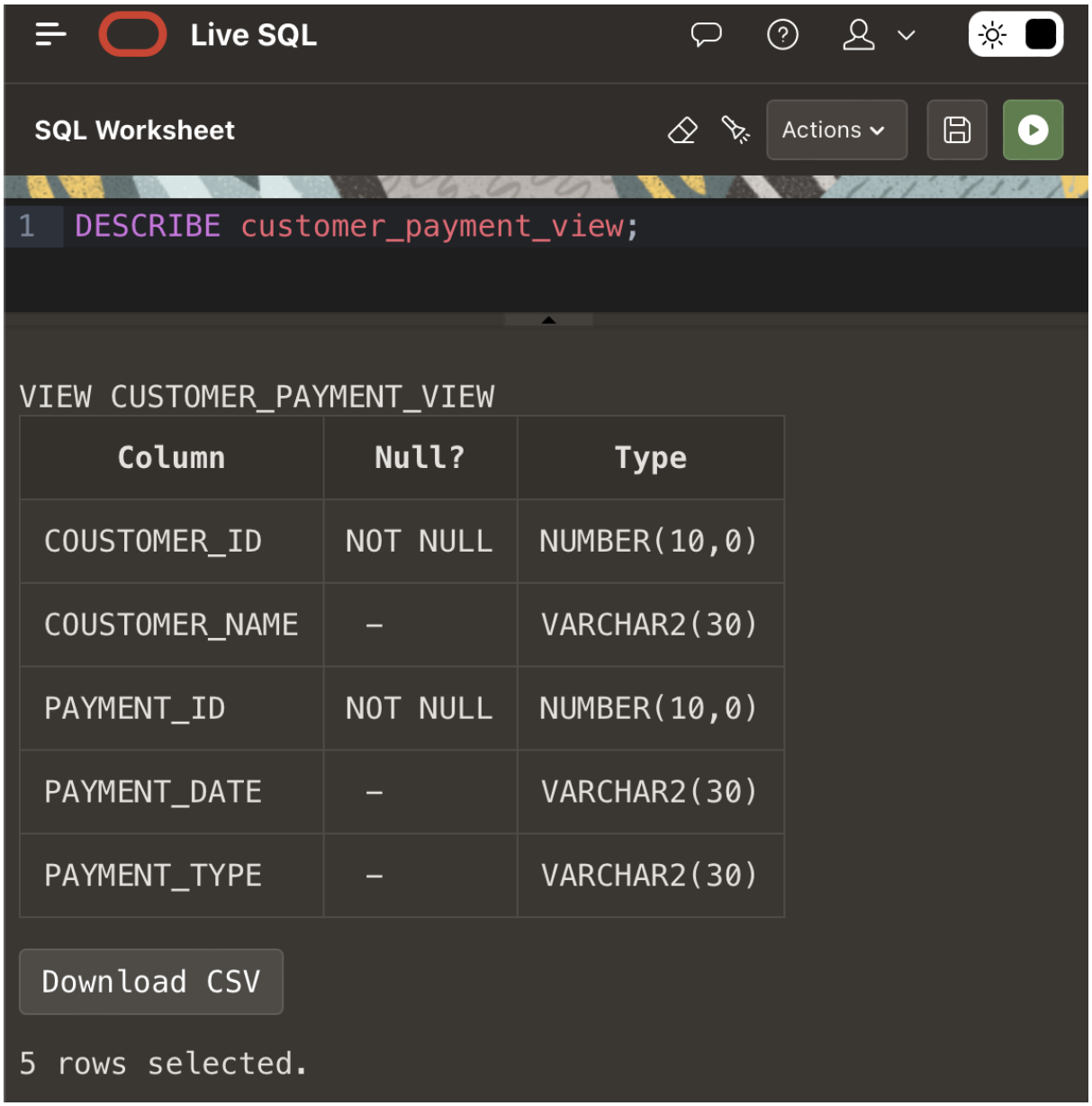


Fig8.12: Description of the complex view.

SELECT * FROM customer_payment_view;

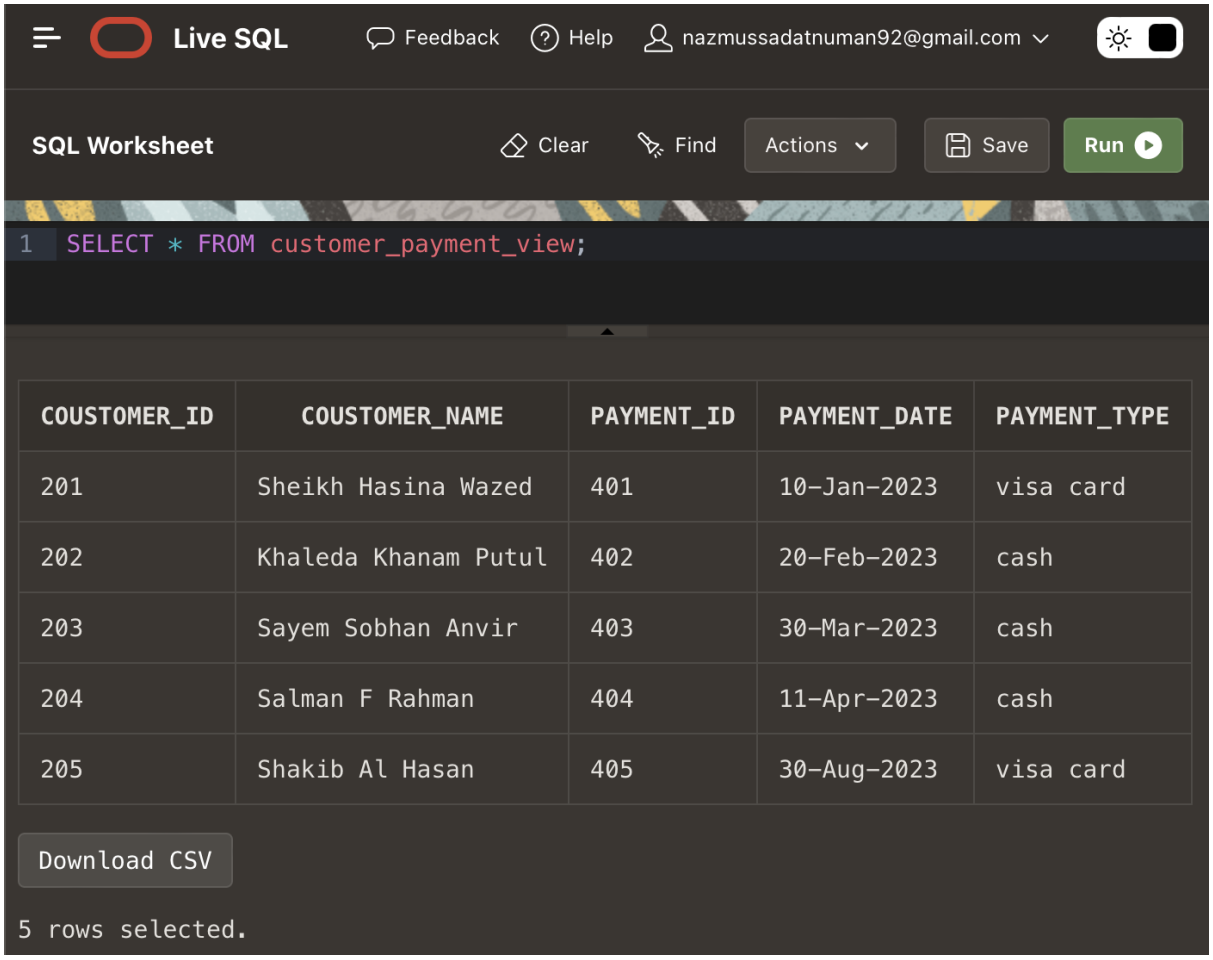


Fig8.13: Result of the complex view as a whole table.

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.