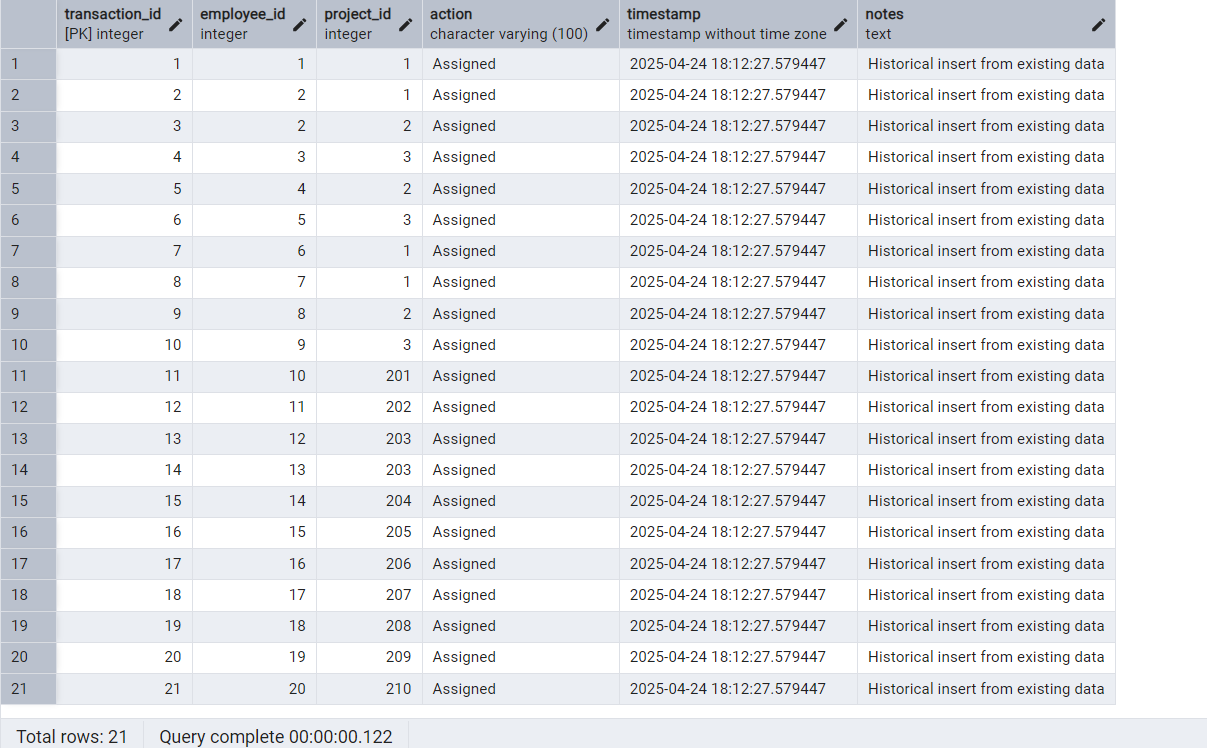
**This assignment focuses on applying advanced SQL techniques like views and triggers within a Company Management System database that I previously created. The goal was to reduce repetitive tasks, improve data tracking, and simplify complex queries.**

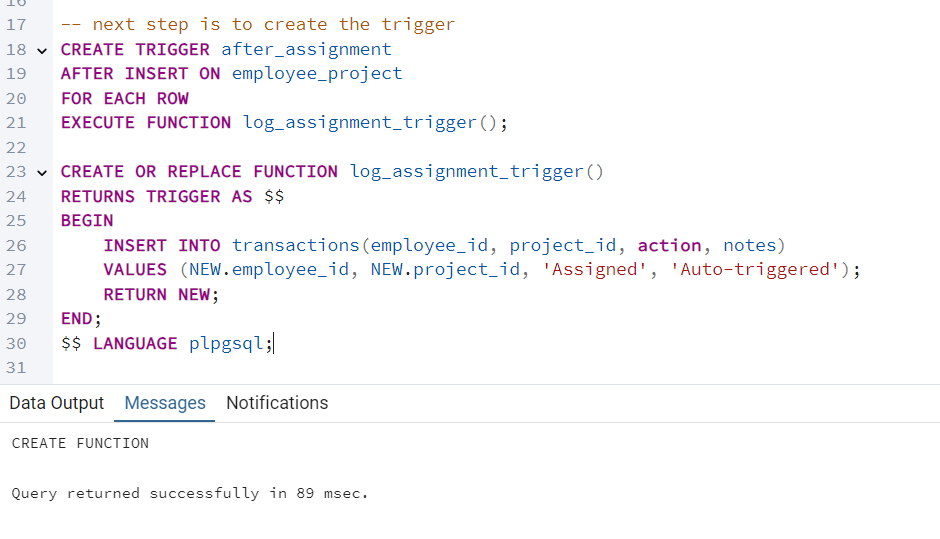
**I began by revisiting my existing schema, which includes employees, projects, departments, and salary records. To enhance the system’s functionality, I created a transactions table to log key events such as project assignments. I then implemented a trigger that automatically records these events in real-time, ensuring no manual logging is needed.**

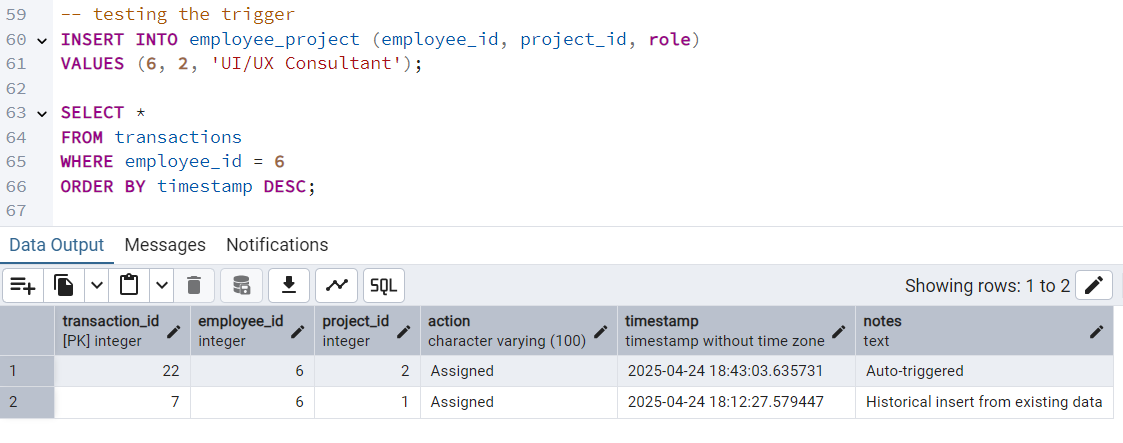
**In addition to automation, I created several views to make data exploration easier and faster. These views help summarize employee roles, department salaries, transaction history, and ongoing project involvement, making the overall system more efficient and easier to manage.**

**Step 1:** Existing employee-project assignments were inserted into the transactions table using a single INSERT ... SELECT statement to ensure historical consistency before enabling trigger-based automation.

To enhance the Company Management System, a transactions table was introduced to capture real-time actions such as project assignments. A view named employee\_project\_history allows simplified access to this log. Additionally, a trigger on the employee\_project table automatically records assignment events in the transactions table, thus reducing manual logging and improving data consistency. These features improve query efficiency, enable audit trails, and demonstrate transactional thinking using RDBMS tools.\







Trigger worked successfully.

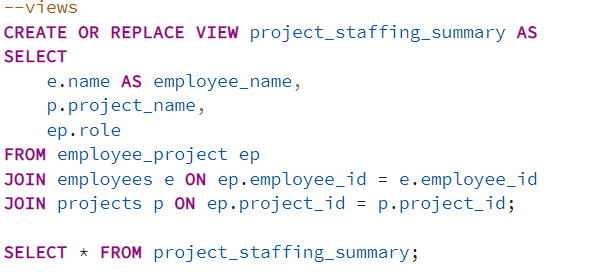
Riya Mehra (employee\_id 6), already assigned to Project Apollo, was assigned to Project Orion as a UI/UX Consultant. The trigger successfully logged this event into the transactions table with an automatic timestamp and action note, demonstrating seamless integration between assignment and logging.

VIEWS.

A **View** in SQL is a **virtual table** created by a query. It does not store data itself but provides a way to **simplify complex queries** or **abstract away raw table structures**, making it easier to interact with the database.

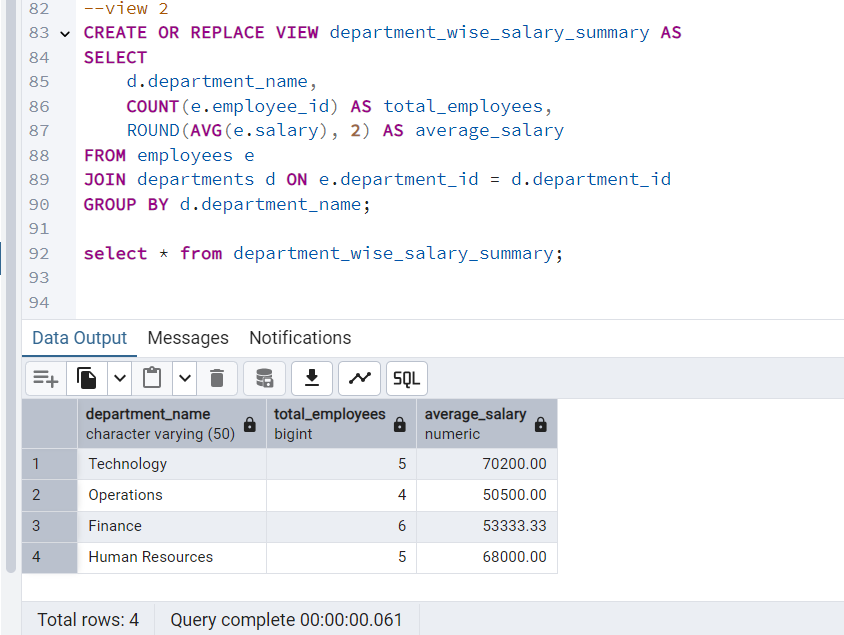
**🧰 Why Use Views?**

* ✅ Save time on **repetitive joins**
* ✅ Improve **data readability**
* ✅ Enhance **security** by showing only selected columns or rows
* ✅ Maintain **consistency** in business logic
* ✅ Simplify data analysis & reporting



I have created a view table to see who is working on what project and their specific role



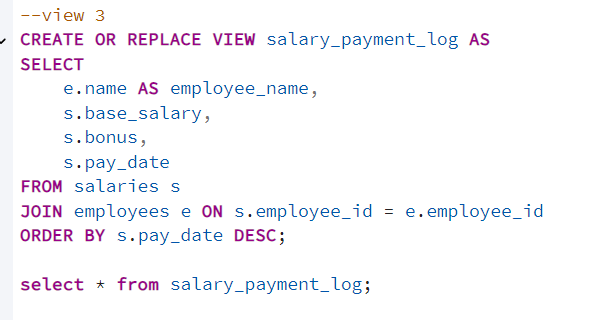
Let us then say that HR department wants what is the avg salary for each department

**📌 Use Case:**

HR can instantly see how departments compare in terms of headcount and average salary, without writing multiple JOIN and GROUP BY clauses each time.

**View 3: salary\_payment\_log**

**🔍 Purpose:**

Simple, readable breakdown of who got paid how much and when.

**🏁 Component 5 Summary: Exploration with Views and Triggers**

In this component, advanced features of SQL were implemented within a Company Management System to enable efficient data management and streamline common operations. A transactions table was introduced to capture real-time employee actions such as project assignments.

A **trigger function** was created to automatically insert records into this table whenever a new assignment was made in the employee\_project table, thereby reducing manual entry and ensuring consistent activity logging. This was successfully tested with both existing and newly added employees.

To further optimize exploration and save time, several **SQL views** were developed:

* project\_staffing\_summary: Summarized employee-project-role mappings
* department\_wise\_salary\_summary: Provided salary insights per department
* active\_project\_roster: Listed employees involved in ongoing projects
* salary\_payment\_log: Offered a payroll activity timeline for audit and review

These views and triggers together showcased the power of relational database features in creating a robust, audit-friendly, and time-efficient management system.