JAY ARRE TALOSIG
COM-231
Mrs. Jensen Santillan

Objectives

By the end of this activity, students will be able to:

• Create and use Clustered and Non-clustered Indexes.

• Optimize queries using Composite and Covering

Indexes.

• Work with Views, including Simple, Updatable, and

Indexed Views.

Part 1: Setting Up the Database and Table

Step 1: Create a New Database

1. Open SQL Server Management Studio (SSMS).

2. Click New Query and enter:

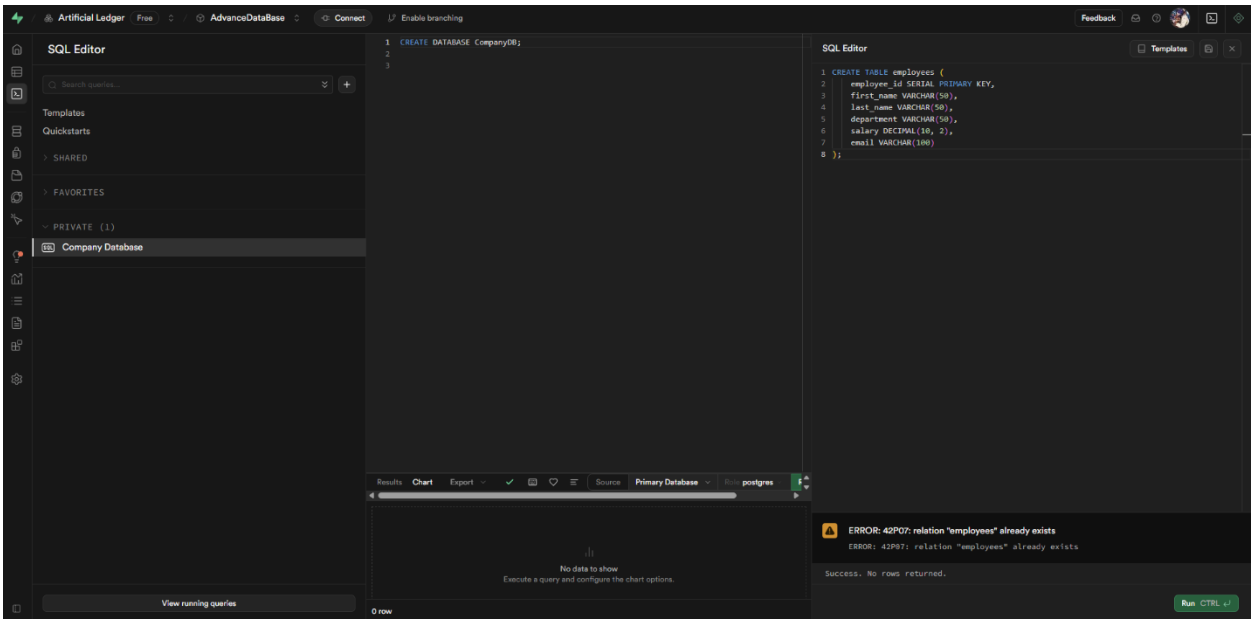
CREATE DATABASE Company;

3. Execute the query by clicking F5 or the Execute

button.

4. Select the new database to work with:

USE CompanyDB;



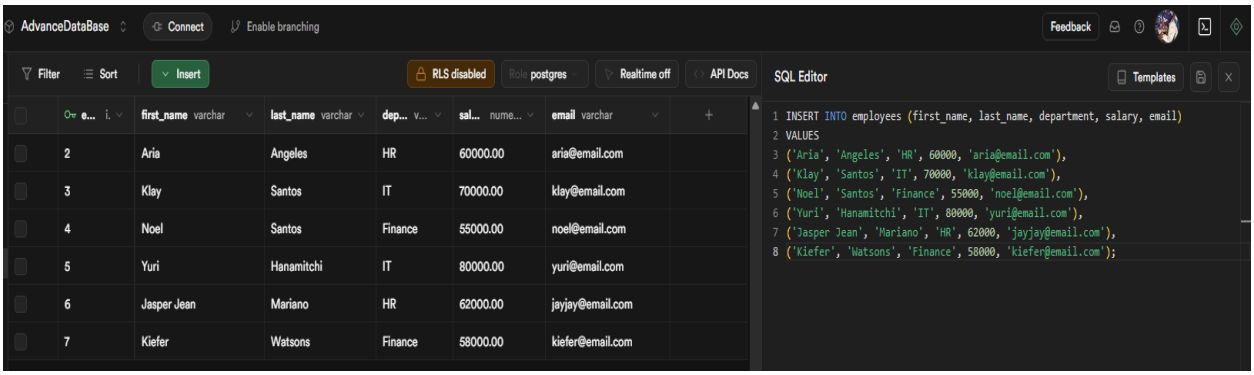
Part 2: Creating the Employees Table

Step 2: Create an Employee Table

Part 3: Inserting Sample Data

Step 3: Insert Sample Employees

Run the following query to add employee records:



Part 4: Indexing

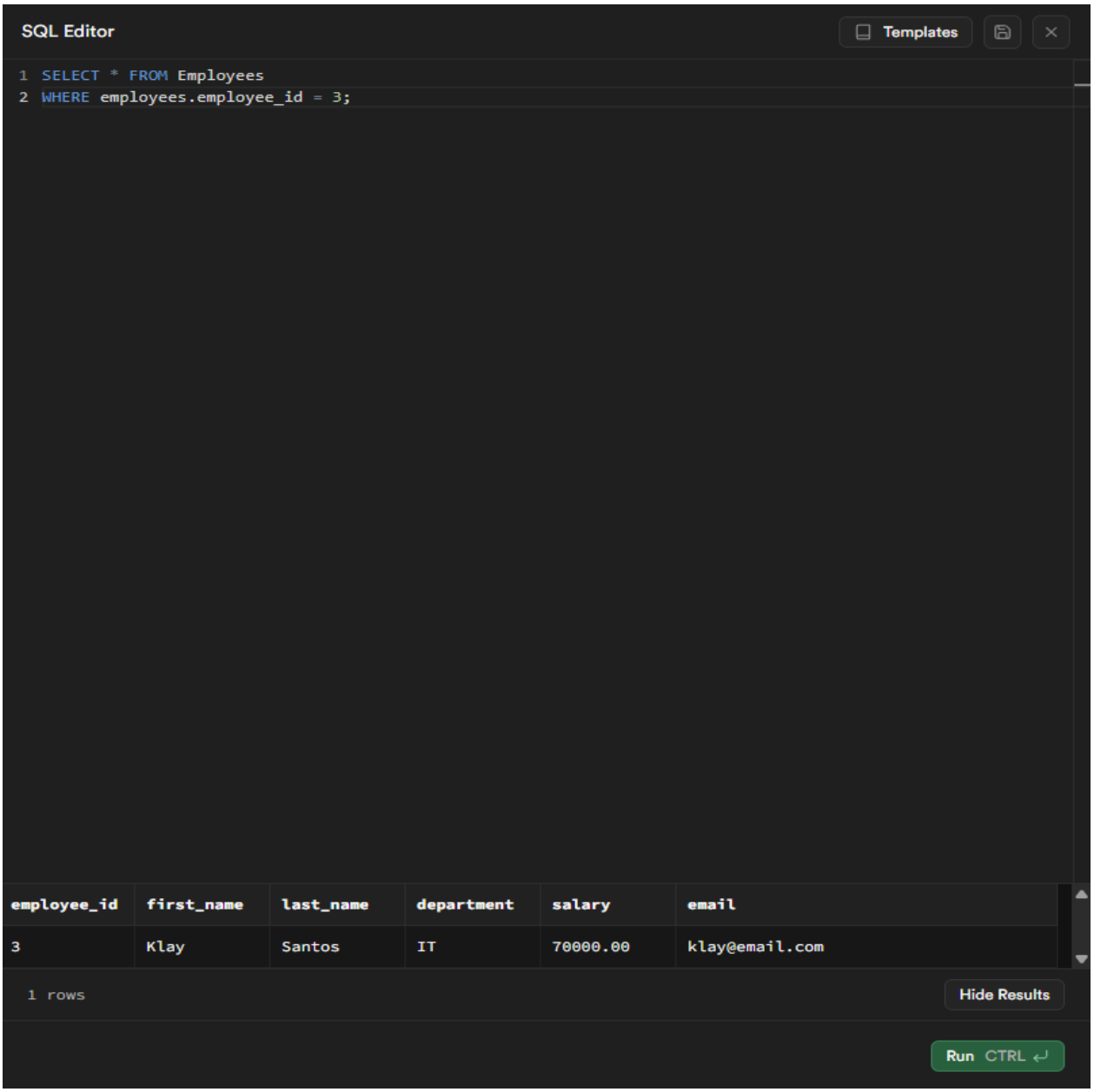
Step 4: Create a Clustered Index

Run the following SQL:

```
CREATE CLUSTERED INDEX IDX_EmployeeID
ON Employees(EmployeeID);
```

TASK: Run the following **before and after** adding the index to compare execution time:

```
SELECT * FROM Employees
WHERE EmployeeID = 3;
```



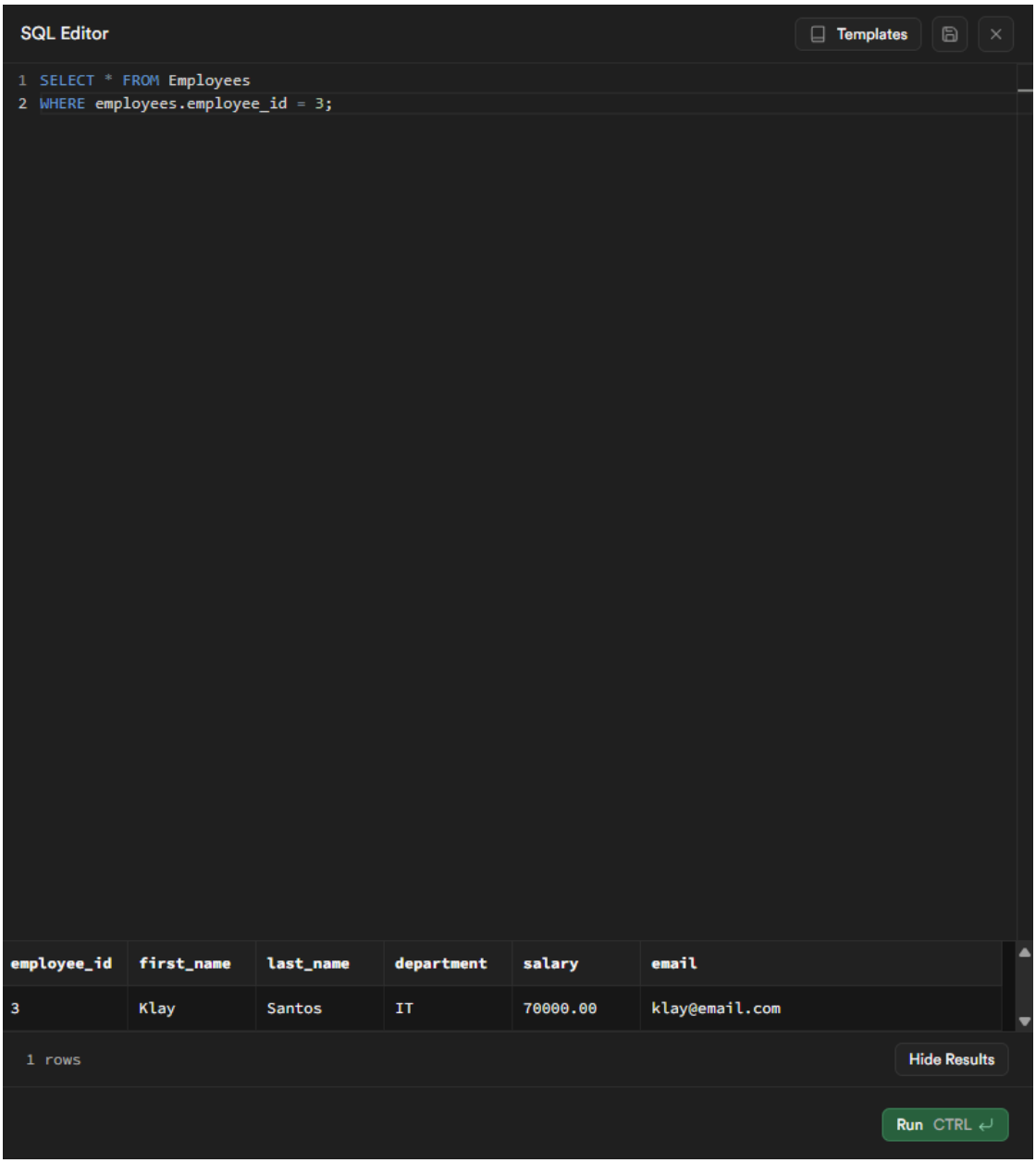
Step 5: Create a Non-Clustered Index

Run this SQL to speed up searches on **LastName**

CREATE NONCLUSTERED INDEX IDX_LastName
ON Employees(LastName);

TASK: Run the following query before and after indexing

SELECT * FROM Employees
WHERE LastName = ‘Watsons’;

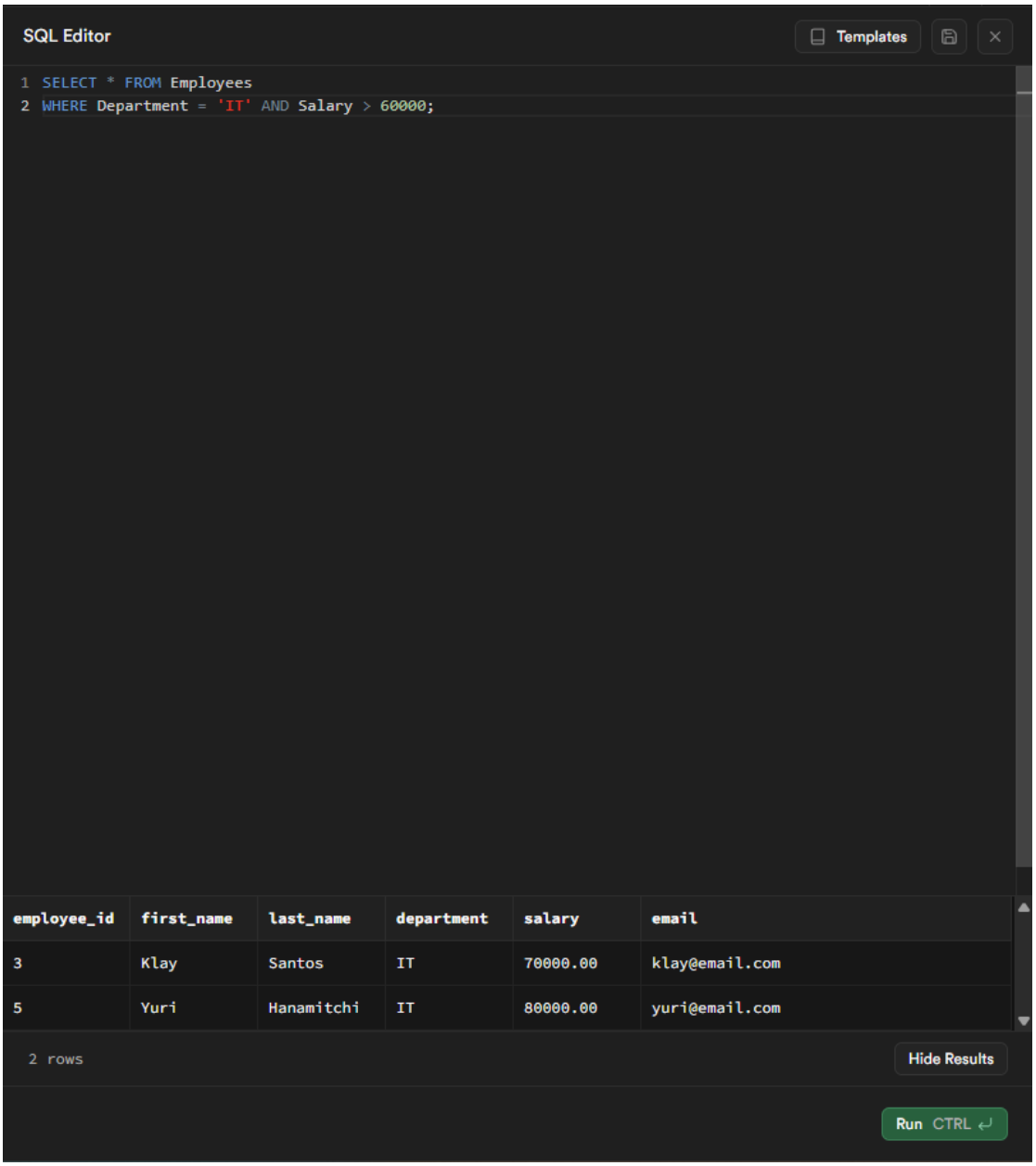


Step 6: Create a Composite Index

CREATE NONCLUSTERED INDEX IDX_DeptSalary
ON Employees(Department, Salary);

Task: Run and Compare

SELECT * FROM Employees
WHERE Department = 'IT' AND Salary > 60000;



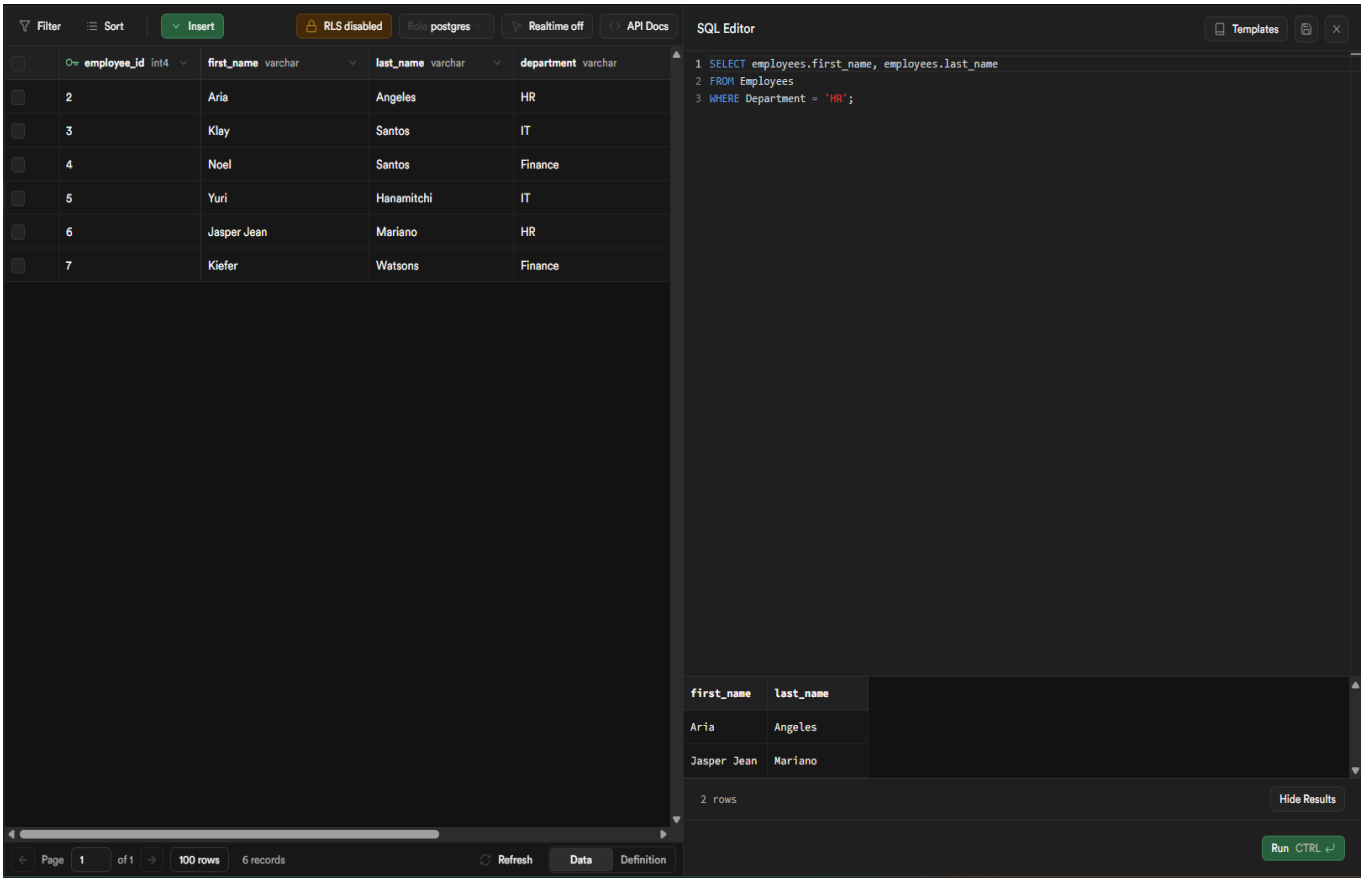
Step 7: Create a Covering Index

Create an index that covers **FirstName and LastName** when searching by Department:

CREATE NONCLUSTERED INDEX IDX_Covering
ON Employees(Department) INCLUDE (FirstName, LastName);

TASK: Run the query before and after theindexing:

SELECT FirstName, LastName
FROM Employees
WHERE Department = ‘HR’



Step 8: Create a Unique Index

Ensure that **Email** is unique

ALTER TABLE Employees
ADD CONSTRAINT UQ_Email UNIQUE (Email);

TASK: Try inserting a duplicate email to see if it prevents it.

Part 5: Creating Views

Step 9: Create a Simple View

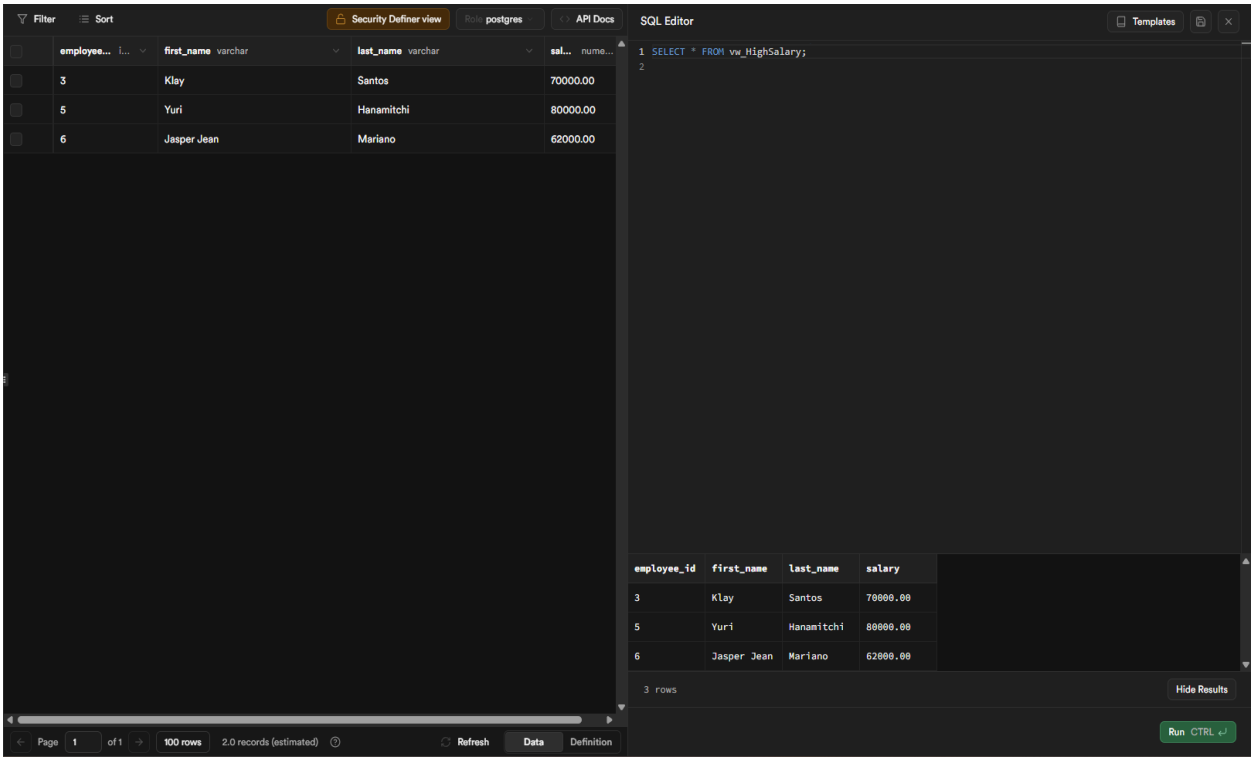
Create a view to show **high salary employees:**

CREATE VIEW vw_HighSalary **AS**
SELECT EmployeeID, FirstName, LastName, Salary

FROM Employees
WHERE Salary > 60000;

TASK: Run:

SELECT * FROM vw_HighSalary;



Insert a new employee with **Salary > 60,000** and check if they appear in the view.

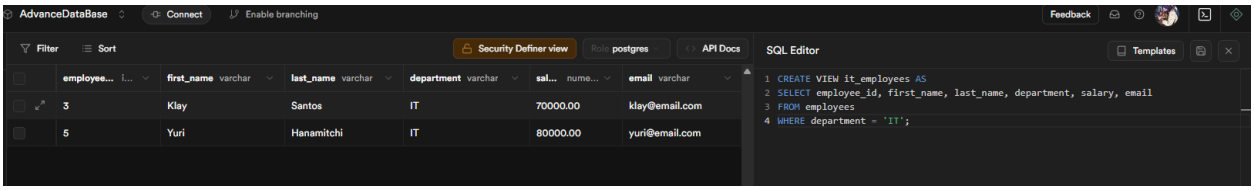
Step 10: Create a View with Aggregation
Create a view for **average salary per department:**

CREATE VIEW vw_AvgSalary AS
SELECT Department, AVG(Salary) AS AvgSalary
FROM Employees
GROUP BY Department;

Step 11: Create an Updatable View

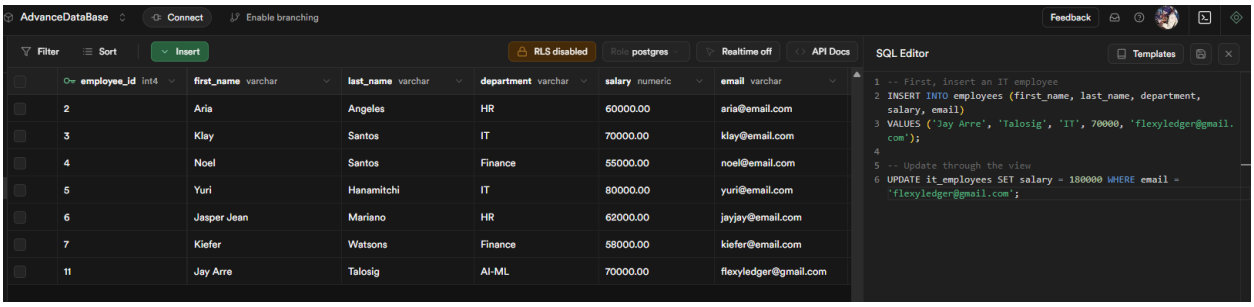
Create a **filtered view** for IT employees

CREATE VIEW vw_ITEmployees AS
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employees
WHERE Department = 'IT'
WITH CHECK OPTION;



TASK: Try updating an employee's salary in this view:

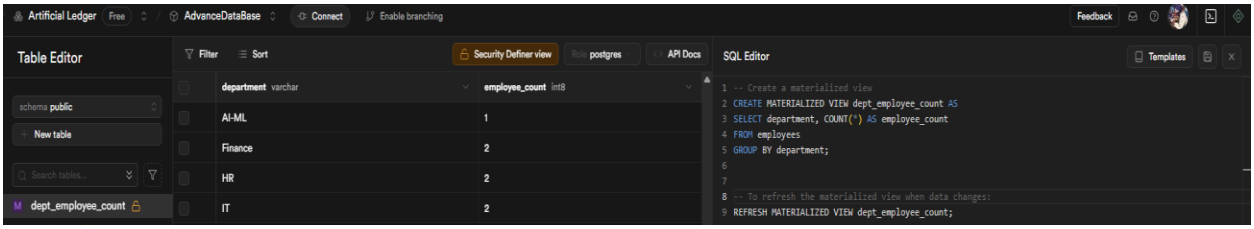
UPDATE vw_ITEmployees
SET Salary = 90000
WHERE EmployeeID = 2;



Try inserting an employee from HR in the view. Does it work? Why not?

Step 12: Create an Indexed View
Create a view with indexing:

CREATE VIEW vw_IndexedSalary WITH SCHEMABINDING
AS
SELECT Department, COUNT_BIG(*)
AS EmpCount, AVG(Salary) AS AvgSalary
FROM dbo.Employees
GROUP BY Department;



Add a Clustered Index:

CREATE UNIQUE CLUSTERED INDEX IDX_vw_IndexedSalary
ON vw_IndexedSalary(Department);

TASK: Run and Compare query speed before and after.

SELECT * **FROM** vw_IndexedSalary
WHERE Department = ‘HR’;

Part 6: Joins and Indexes

Step 13: Optimized Joins

1. Create a Departments Table:

CREATE TABLE Departments (

DeptID **INT** PRIMARY KEY,
DeptName NVARCHAR(50)
);

2. Insert sample data.

INSERT INTO Departments
VALUES (1, ‘HR’), (2, ‘IT’), (3, ‘Finance’);

3. Add DeptID to Employees.

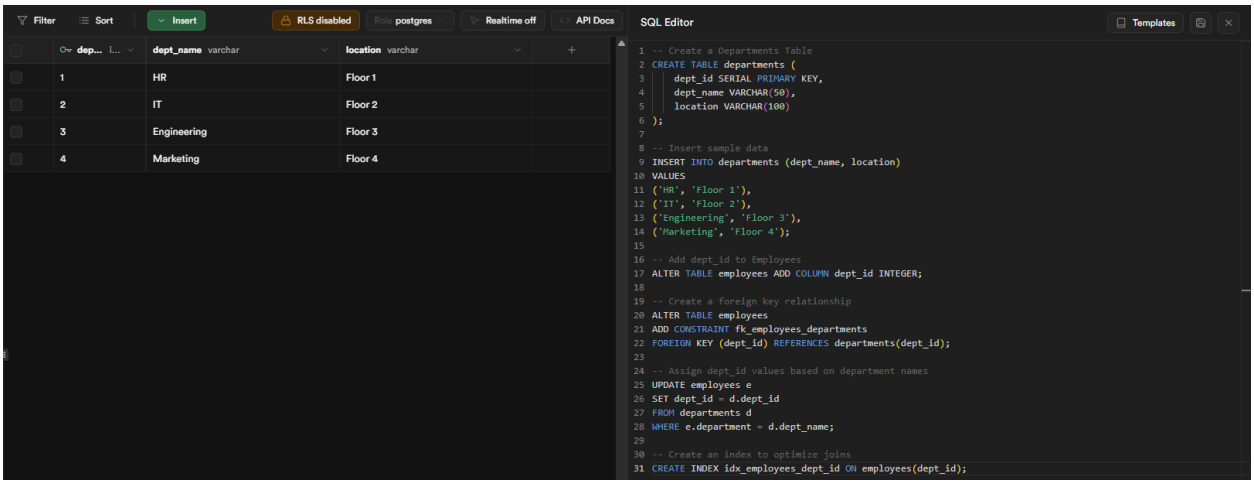
ALTER TABLE Employees **ADD** DeptID **INT**;

4. Assign DeptID values.

UPDATE Employees **SET** DeptID = 1 **WHERE** Department = ‘HR’;
UPDATE Employees **SET** DeptID = 2 **WHERE** Department = ‘IT’;
UPDATE Employees **SET** DeptID = 3 **WHERE** Department = ‘Finance’;

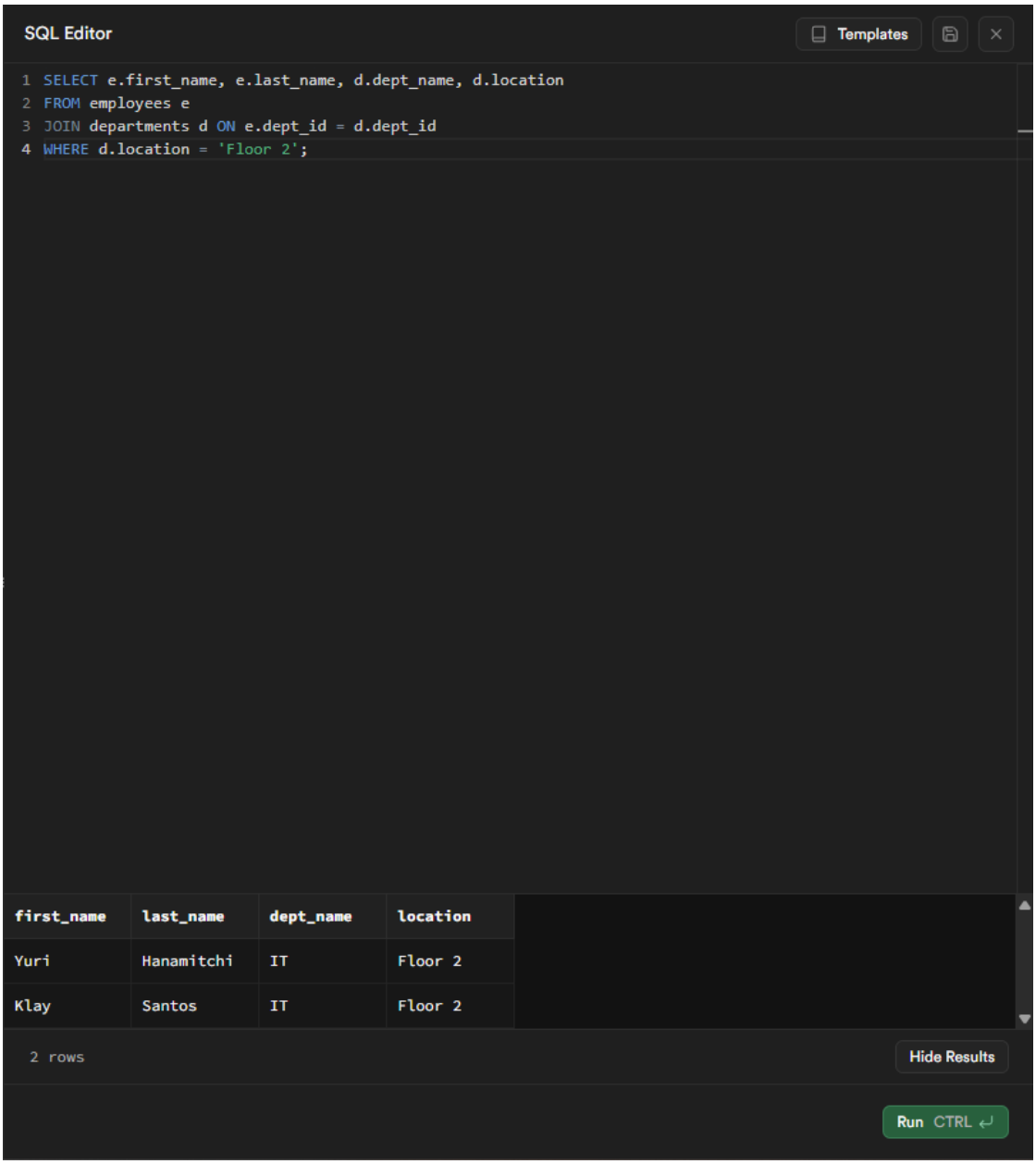
5. Optimize the join:

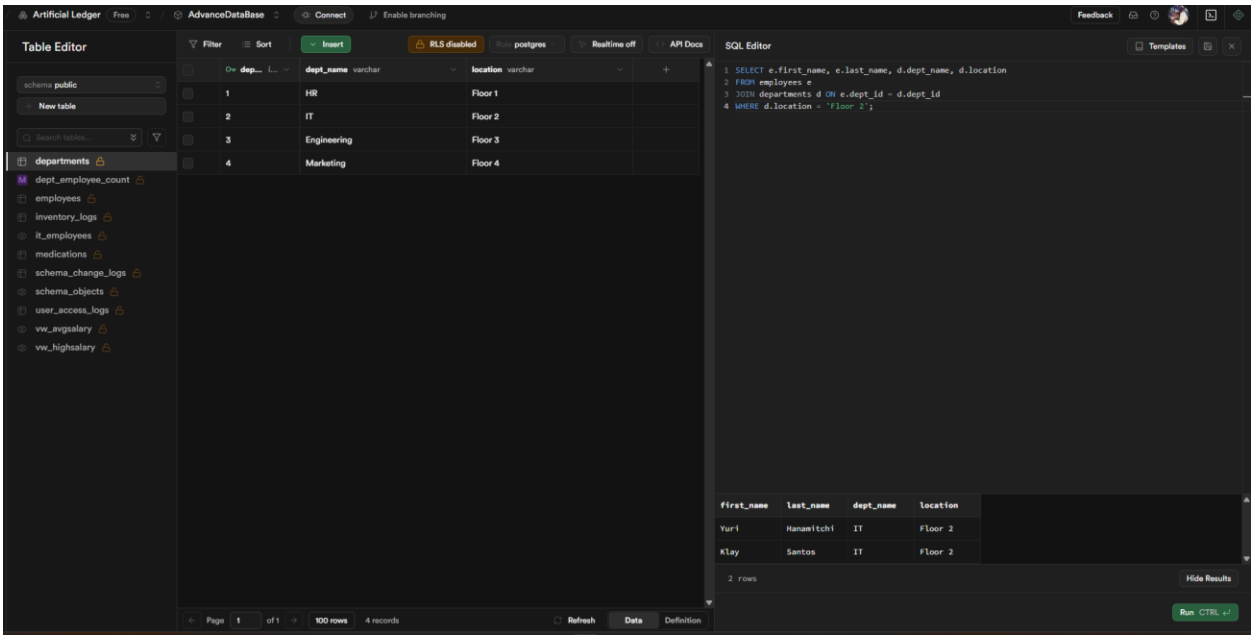
CREATE NONCLUSTERED INDEX IDX_DeptID
ON Employees(DeptID);



TASK: Run and Compare:

SELECT e.EmplyeeID, e.FirstName, e.LastName, d.DeptName
FROM Employees e
JOIN Departments d **ON** e.DeptID = d.DeptID;




Artificial Ledger
Free
AdvanceDataBase
Connect
Enable branching
Feedback
Table Editor
schema public
New table
Search tables...
departments
dept_employee_count
employees
inventory_logs
it_employees
medications
schema_change_logs
schema_objects
user_access_logs
vw_avgsalary
vw_highsalary
Filter
Sort
Insert
RLS disabled
Role postgres
Realtime off
API Docs
dep... i...
dept_name varchar
location varchar
1 HR Floor 1
2 IT Floor 2
3 Engineering Floor 3
4 Marketing Floor 4
Page 1 of 1
100 rows
4 records
Refresh
Data
Definition
SQL Editor
Templates
1 SELECT e.first_name, e.last_name, d.dept_name, d.location
2 FROM employees e
3 JOIN departments d ON e.dept_id = d.dept_id
4 WHERE d.location = 'Floor 2';
first_name last_name dept_name location
Yuri Hanamitchi IT Floor 2
Klay Santos IT Floor 2
2 rows
Hide Results
Run CTRL