# SQL Assignment

Data Source Information:

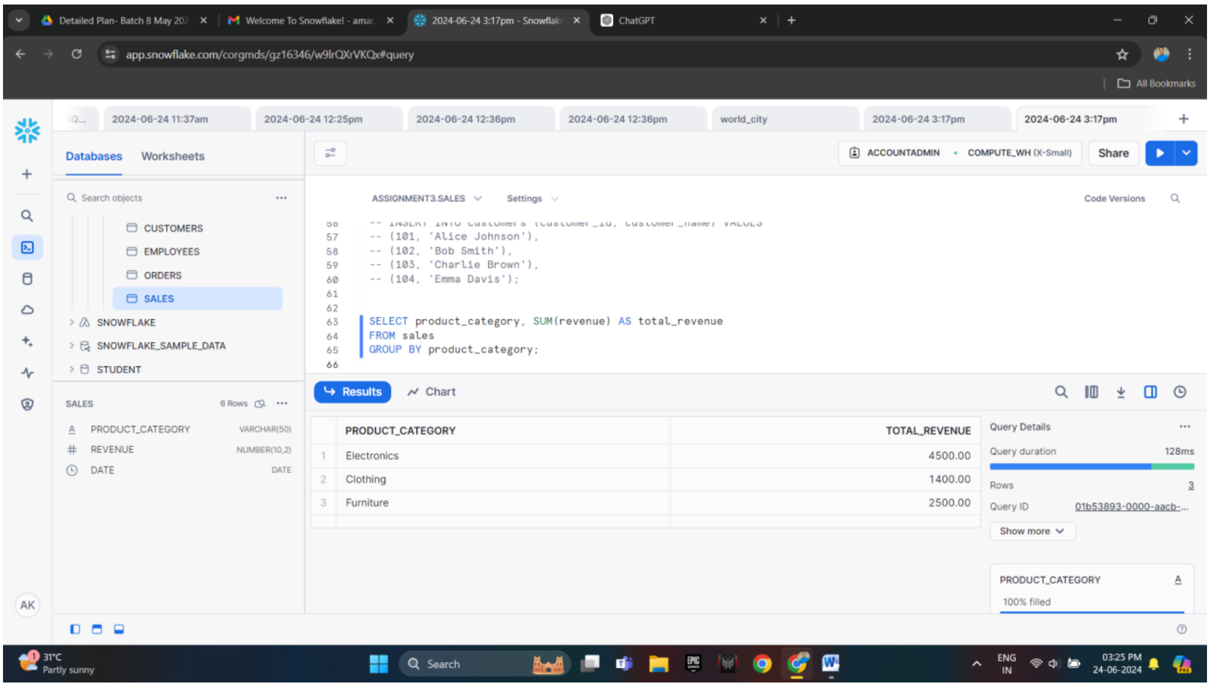
Sales Table:

Table Name: sales
Columns:
product_category (VARCHAR): The category of the product being sold.
revenue (DECIMAL): The revenue generated from each sale.
date (DATE): The date when the sale occurred.
Employees Table:

Table Name: employees
Columns:
employee_id (INT): Unique identifier for each employee.
employee_name (VARCHAR): The name of the employee.
salary (DECIMAL): The salary of the employee.
Orders Table:

Table Name: orders
Columns:
order_id (INT): Unique identifier for each order.
customer_id (INT): Unique identifier for each customer.
purchase_amount (DECIMAL): The total purchase amount for each order.

1. **Aggregate Functions:**
   - Write a SQL query to calculate the total revenue generated by each product category. Use the SUM() function for this calculation.



SALES 6 Rows

| | |
|---|---|
| PRODUCT_CATEGORY | VARCHAR(50) |
| REVENUE | NUMBER(10,2) |
| DATE | DATE |

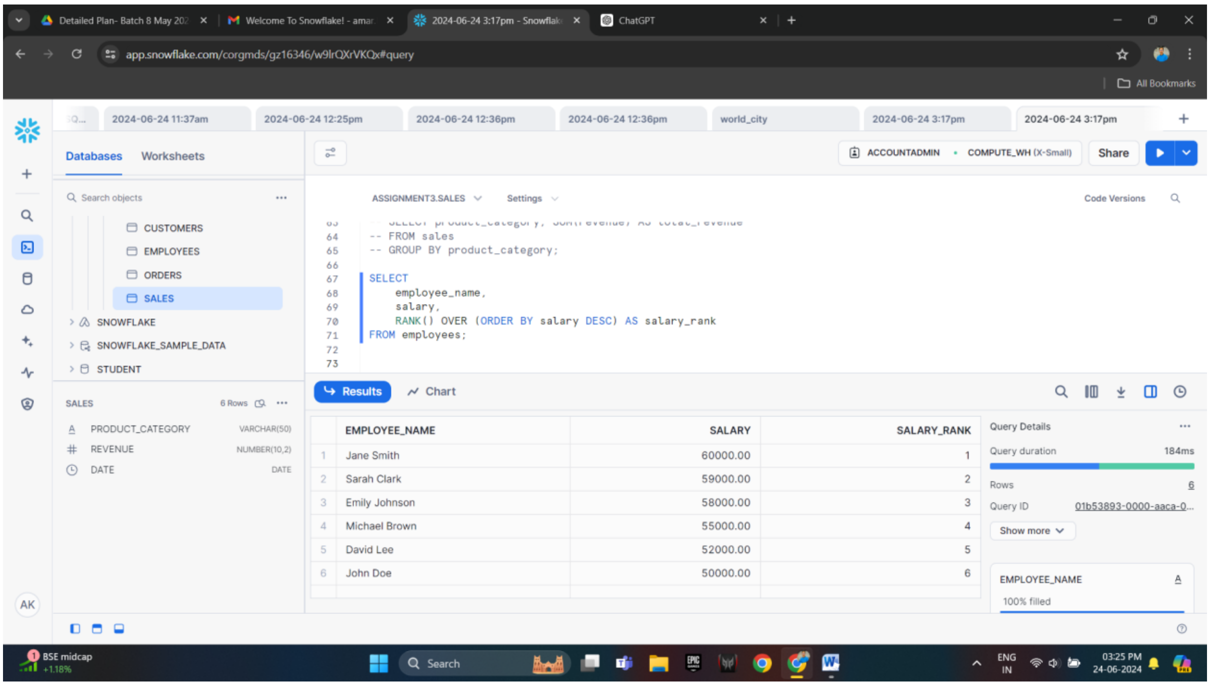
```
-- INSERT INTO customers (customer_id, customer_name) VALUES
-- (101, 'Alice Johnson'),
-- (102, 'Bob Smith'),
-- (103, 'Charlie Brown'),
-- (104, 'Emma Davis');


SELECT product_category, SUM(revenue) AS total_revenue
FROM sales
GROUP BY product_category;
```

| | PRODUCT_CATEGORY | TOTAL_REVENUE |
|---|---|---|
| 1 | Electronics | 4500.00 |
| 2 | Clothing | 1400.00 |
| 3 | Furniture | 2500.00 |

Query Details
Query duration 128ms
Rows 3
Query ID 01b53893-0000-aacb-...
PRODUCT_CATEGORY 100% filled

2. **Ranking Functions:**
   - Use the RANK() function to assign ranks to employees based on their salary. Display the employee name, salary, and rank in the result set.



SALES — 6 Rows

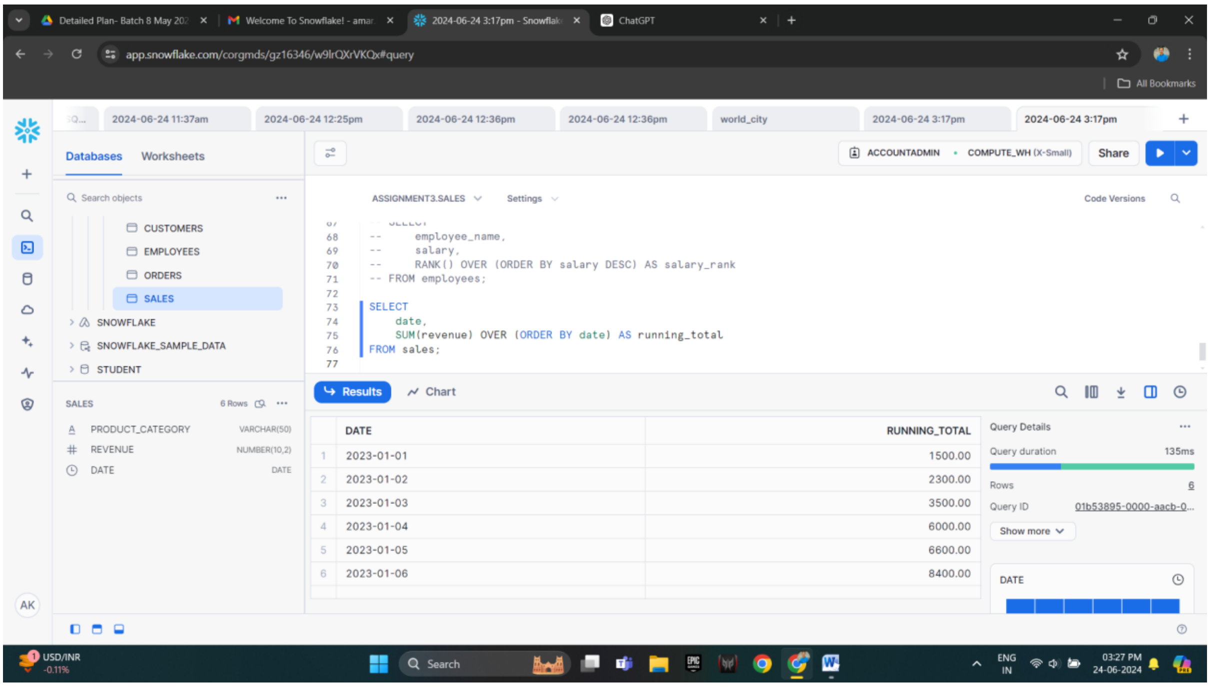
| | |
|---|---|
| PRODUCT_CATEGORY | VARCHAR(50) |
| REVENUE | NUMBER(10,2) |
| DATE | DATE |

```sql
-- SELECT product_category, SUM(revenue) AS total_revenue
-- FROM sales
-- GROUP BY product_category;

SELECT
    employee_name,
    salary,
    RANK() OVER (ORDER BY salary DESC) AS salary_rank
FROM employees;
```

| | EMPLOYEE_NAME | SALARY | SALARY_RANK |
|---|---|---|---|
| 1 | Jane Smith | 60000.00 | 1 |
| 2 | Sarah Clark | 59000.00 | 2 |
| 3 | Emily Johnson | 58000.00 | 3 |
| 4 | Michael Brown | 55000.00 | 4 |
| 5 | David Lee | 52000.00 | 5 |
| 6 | John Doe | 50000.00 | 6 |

3. **Window Functions:**
   - Write a SQL query to calculate the running total of sales revenue over time. Use the ROW_NUMBER() function to achieve this.



SALES — 6 Rows

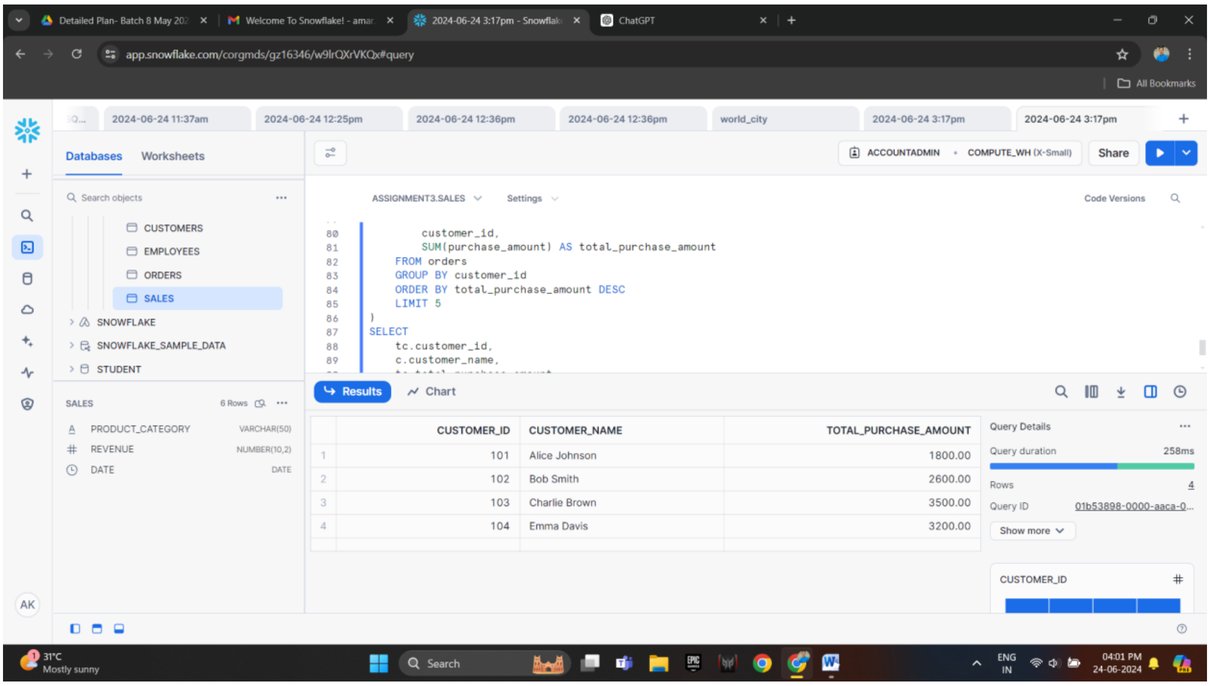
| | |
|---|---|
| PRODUCT_CATEGORY | VARCHAR(50) |
| REVENUE | NUMBER(10,2) |
| DATE | DATE |

```sql
-- SELECT
--     employee_name,
--     salary,
--     RANK() OVER (ORDER BY salary DESC) AS salary_rank
-- FROM employees;

SELECT
    date,
    SUM(revenue) OVER (ORDER BY date) AS running_total
FROM sales;
```

| | DATE | RUNNING_TOTAL |
|---|---|---|
| 1 | 2023-01-01 | 1500.00 |
| 2 | 2023-01-02 | 2300.00 |
| 3 | 2023-01-03 | 3500.00 |
| 4 | 2023-01-04 | 6000.00 |
| 5 | 2023-01-05 | 6600.00 |
| 6 | 2023-01-06 | 8400.00 |

4. **Common Table Expressions (CTEs):**
   - Create a CTE that lists the top 5 customers based on their total purchase amount. Include customer ID, name, and total purchase amount in the result.



```sql
        customer_id,
        SUM(purchase_amount) AS total_purchase_amount
    FROM orders
    GROUP BY customer_id
    ORDER BY total_purchase_amount DESC
    LIMIT 5
)
SELECT
    tc.customer_id,
    c.customer_name,
```

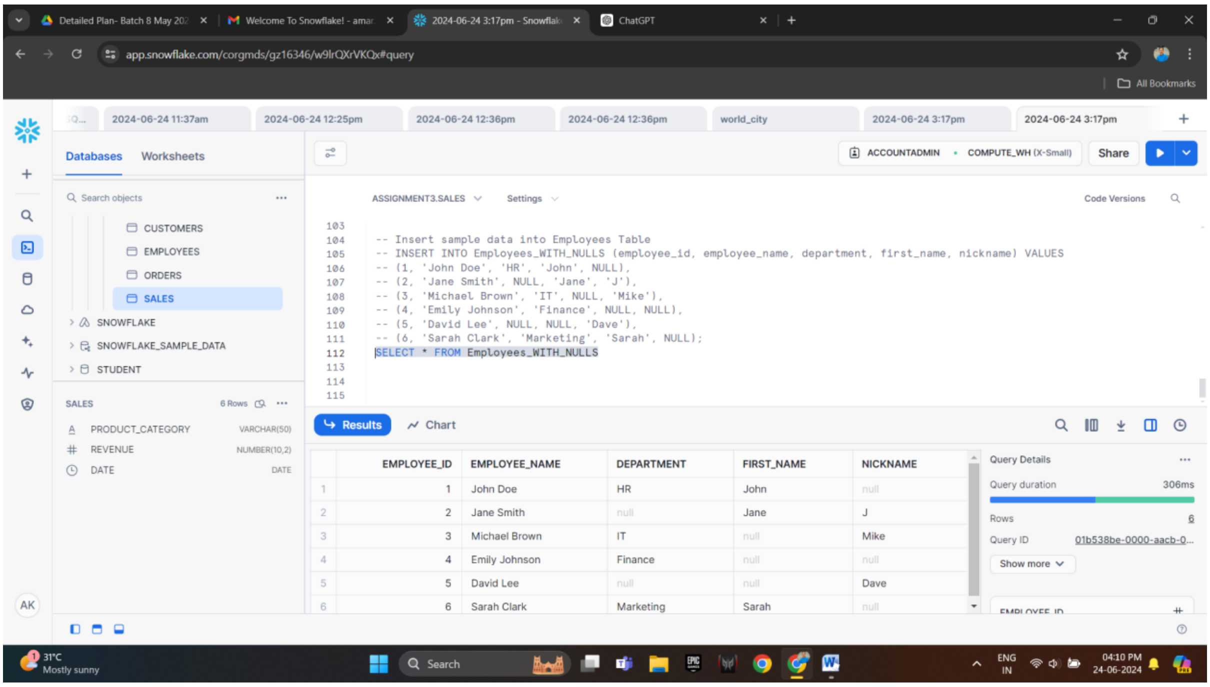
| | CUSTOMER_ID | CUSTOMER_NAME | TOTAL_PURCHASE_AMOUNT |
|---|---|---|---|
| 1 | 101 | Alice Johnson | 1800.00 |
| 2 | 102 | Bob Smith | 2600.00 |
| 3 | 103 | Charlie Brown | 3500.00 |
| 4 | 104 | Emma Davis | 3200.00 |

Query duration 258ms

Rows 4

5. **NVL vs COALESCE:**

  - Explain the difference between the NVL() and COALESCE() functions in SQL. Provide an example for each function demonstrating its usage.

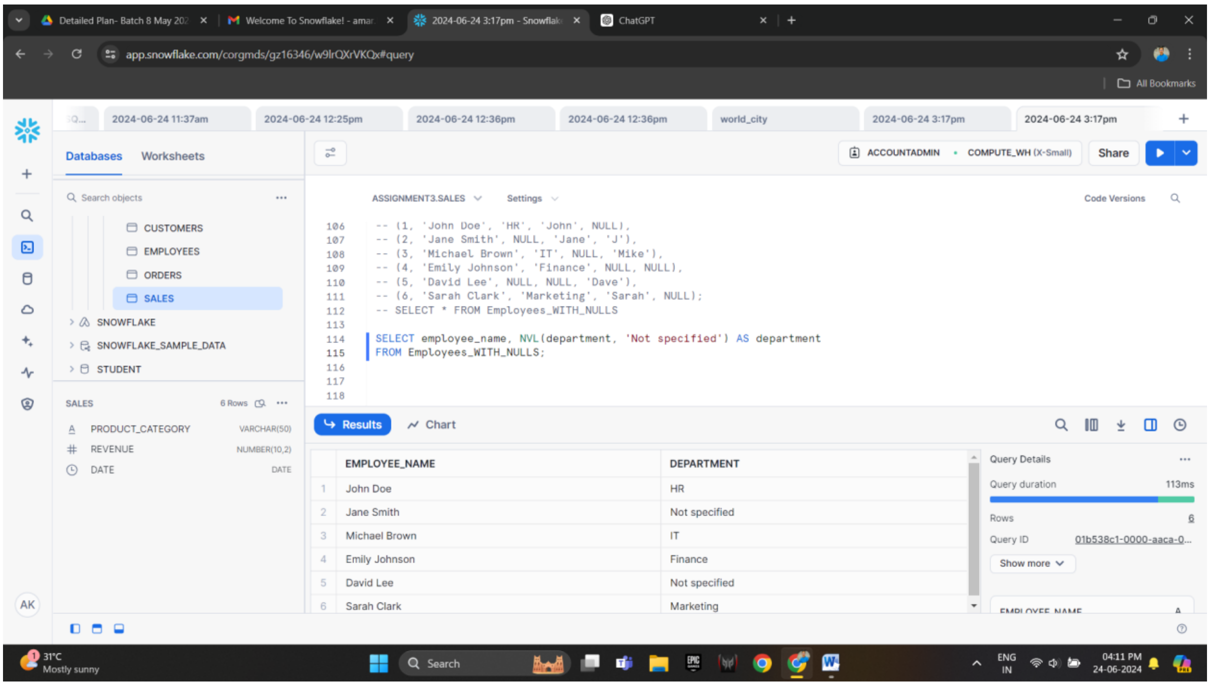


```sql
-- Insert sample data into Employees Table
-- INSERT INTO Employees_WITH_NULLS (employee_id, employee_name, department, first_name, nickname) VALUES
-- (1, 'John Doe', 'HR', 'John', NULL),
-- (2, 'Jane Smith', NULL, 'Jane', 'J'),
-- (3, 'Michael Brown', 'IT', NULL, 'Mike'),
-- (4, 'Emily Johnson', 'Finance', NULL, NULL),
-- (5, 'David Lee', NULL, NULL, 'Dave'),
-- (6, 'Sarah Clark', 'Marketing', 'Sarah', NULL);
SELECT * FROM Employees_WITH_NULLS
```

| | EMPLOYEE_ID | EMPLOYEE_NAME | DEPARTMENT | FIRST_NAME | NICKNAME |
|---|---|---|---|---|---|
| 1 | 1 | John Doe | HR | John | null |
| 2 | 2 | Jane Smith | null | Jane | J |
| 3 | 3 | Michael Brown | IT | null | Mike |
| 4 | 4 | Emily Johnson | Finance | null | null |
| 5 | 5 | David Lee | null | null | Dave |
| 6 | 6 | Sarah Clark | Marketing | Sarah | null |

Query duration 306ms

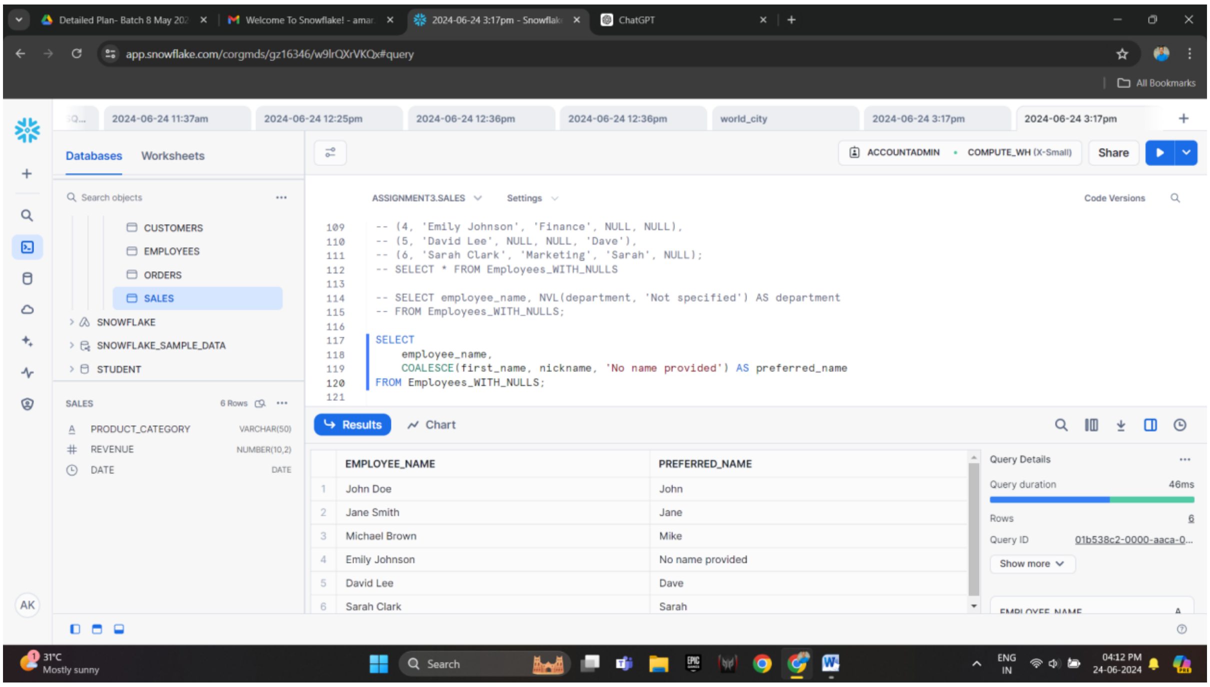
Rows 6



```sql
-- (1, 'John Doe', 'HR', 'John', NULL),
-- (2, 'Jane Smith', NULL, 'Jane', 'J'),
-- (3, 'Michael Brown', 'IT', NULL, 'Mike'),
-- (4, 'Emily Johnson', 'Finance', NULL, NULL),
-- (5, 'David Lee', NULL, NULL, 'Dave'),
-- (6, 'Sarah Clark', 'Marketing', 'Sarah', NULL);
-- SELECT * FROM Employees_WITH_NULLS

SELECT employee_name, NVL(department, 'Not specified') AS department
FROM Employees_WITH_NULLS;
```

| | EMPLOYEE_NAME | DEPARTMENT |
|---|---|---|
| 1 | John Doe | HR |
| 2 | Jane Smith | Not specified |
| 3 | Michael Brown | IT |
| 4 | Emily Johnson | Finance |
| 5 | David Lee | Not specified |
| 6 | Sarah Clark | Marketing |



```sql
-- (4, 'Emily Johnson', 'Finance', NULL, NULL),
-- (5, 'David Lee', NULL, NULL, 'Dave'),
-- (6, 'Sarah Clark', 'Marketing', 'Sarah', NULL);
-- SELECT * FROM Employees_WITH_NULLS

-- SELECT employee_name, NVL(department, 'Not specified') AS department
-- FROM Employees_WITH_NULLS;

SELECT
    employee_name,
    COALESCE(first_name, nickname, 'No name provided') AS preferred_name
FROM Employees_WITH_NULLS;
```

| | EMPLOYEE_NAME | PREFERRED_NAME |
|---|---|---|
| 1 | John Doe | John |
| 2 | Jane Smith | Jane |
| 3 | Michael Brown | Mike |
| 4 | Emily Johnson | No name provided |
| 5 | David Lee | Dave |
| 6 | Sarah Clark | Sarah |