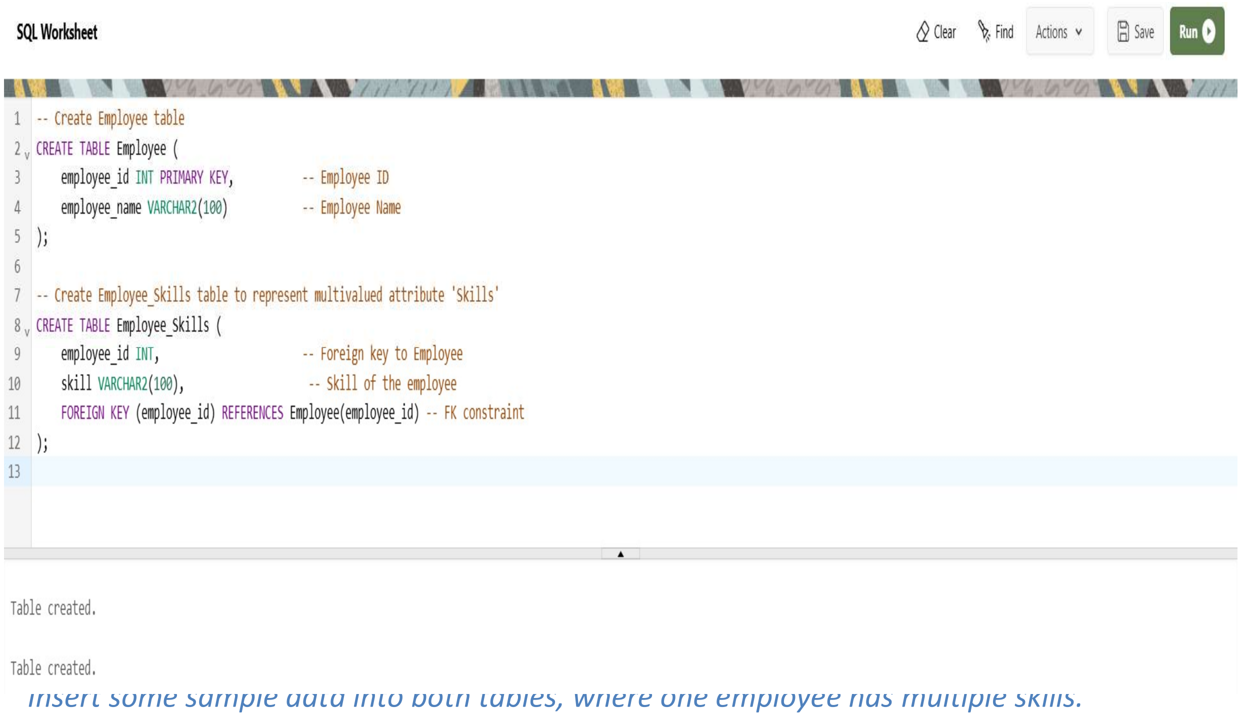
Use SQL queries to extract and analyze data from a database.

Step 1: Create Tables for Employee and Skills

Here, we represent the Employee entity with a multivalued attribute Skills by creating two tables: one for Employee and one for Employee_Skills (which will store the skills as multiple rows).

```
-- Create Employee table
CREATE TABLE Employee (
   employee_id INT PRIMARY KEY,          -- Employee ID
   employee_name VARCHAR2(100)           -- Employee Name
);
-- Create Employee_Skills table to represent multivalued attribute 'Skills'
CREATE TABLE Employee_Skills (
   employee_id INT,                      -- Foreign key to Employee
   skill VARCHAR2(100),                  -- Skill of the employee
   FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) -- FK constraint
);
```



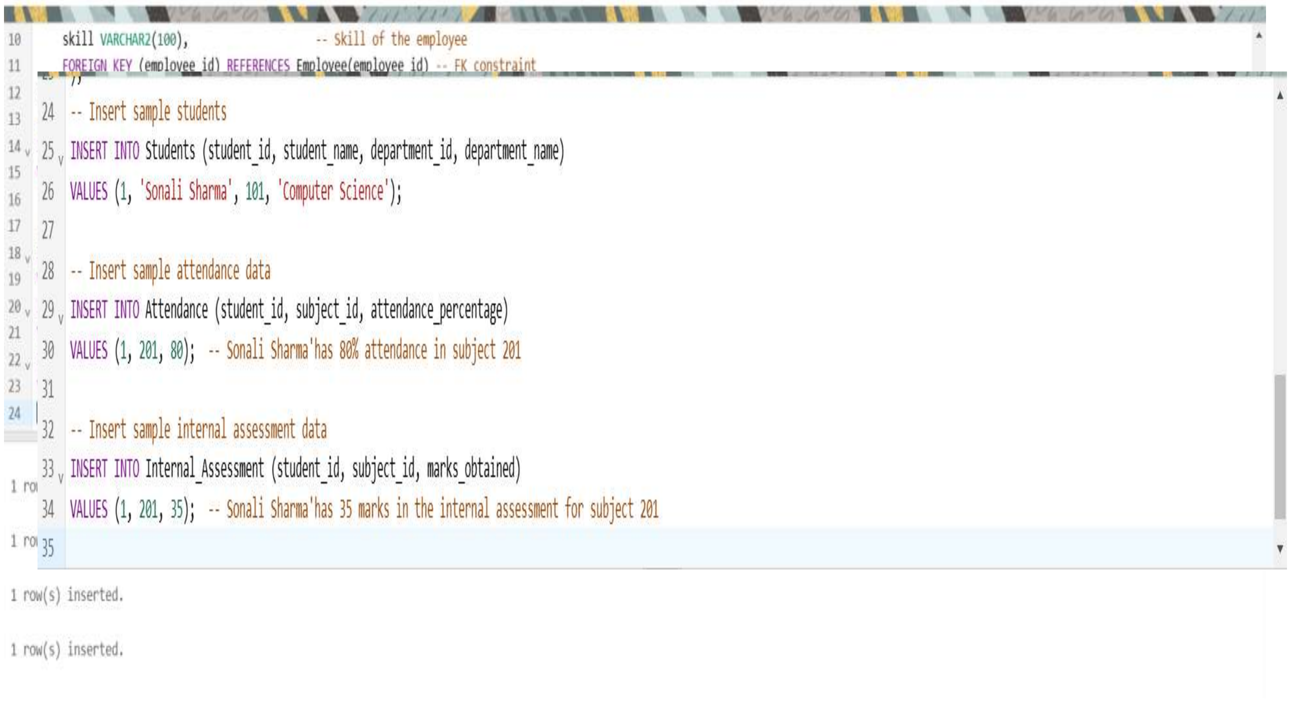
```
-- Create Employee table
CREATE TABLE Employee (
    employee_id INT PRIMARY KEY,          -- Employee ID
    employee_name VARCHAR2(100)           -- Employee Name
);

-- Create Employee_Skills table to represent multivalued attribute 'Skills'
CREATE TABLE Employee_Skills (
    employee_id INT,                      -- Foreign key to Employee
    skill VARCHAR2(100),                  -- Skill of the employee
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) -- FK constraint
);
```

```
Table created.

Table created.
```

Insert some sample data into both tables, where one employee has multiple skills.

-- Insert data into Employee table

```
INSERT INTO Employee (employee_id, employee_name)
VALUES (1, 'Sonali Sharma’);

-- Insert multiple skills for employee with ID = 1
INSERT INTO Employee_Skills (employee_id, skill)
VALUES (1, 'Java');
INSERT INTO Employee_Skills (employee_id, skill)
VALUES (1, 'Python');
INSERT INTO Employee_Skills (employee_id, skill)
VALUES (1, 'SQL');
```



```
    skill VARCHAR2(100),                -- Skill of the employee
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) -- FK constraint
```

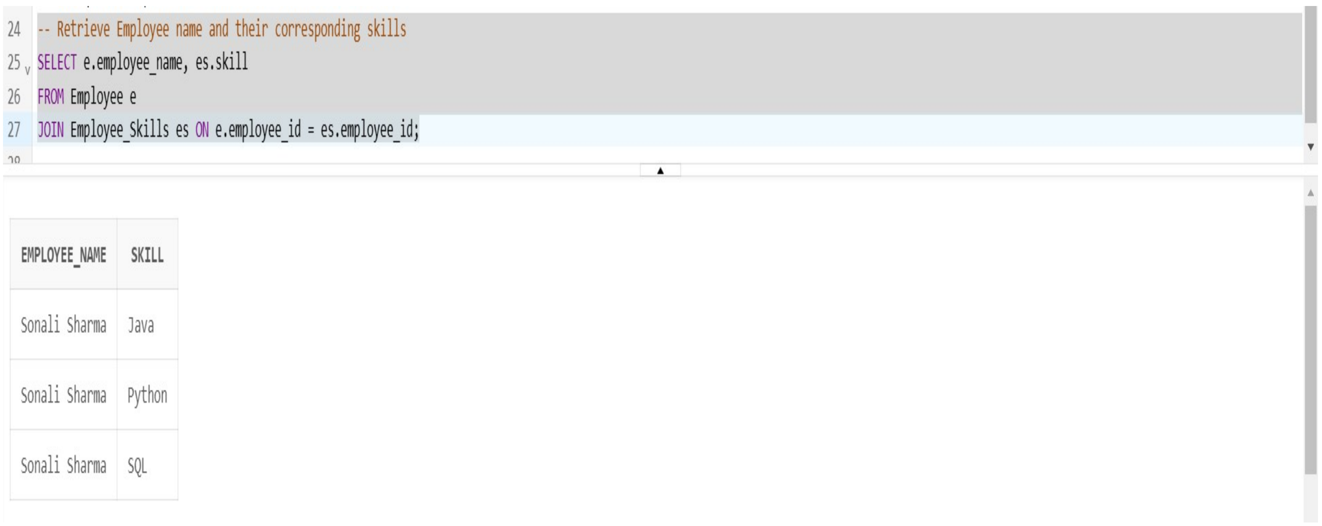
```
-- Insert sample students
INSERT INTO Students (student_id, student_name, department_id, department_name)
VALUES (1, 'Sonali Sharma', 101, 'Computer Science');

-- Insert sample attendance data
INSERT INTO Attendance (student_id, subject_id, attendance_percentage)
VALUES (1, 201, 80);  -- Sonali Sharma'has 80% attendance in subject 201

-- Insert sample internal assessment data
INSERT INTO Internal_Assessment (student_id, subject_id, marks_obtained)
VALUES (1, 201, 35);  -- Sonali Sharma'has 35 marks in the internal assessment for subject 201
```

```
1 row(s) inserted.

1 row(s) inserted.
```

Step 3: Querying the Data

To retrieve the employee and their associated skills, we perform a JOIN between the Employee and Employee_Skills tables.

```
-- Retrieve Employee name and their corresponding skills
SELECT e.employee_name, es.skill
FROM Employee e
JOIN Employee_Skills es ON e.employee_id = es.employee_id;
```


```
-- Retrieve Employee name and their corresponding skills
SELECT e.employee_name, es.skill
FROM Employee e
JOIN Employee_Skills es ON e.employee_id = es.employee_id;
```

EMPLOYEE_NAME	SKILL
Sonali Sharma	Java
Sonali Sharma	Python
Sonali Sharma	SQL

Step 1: Create Nested Table Type for Order Items

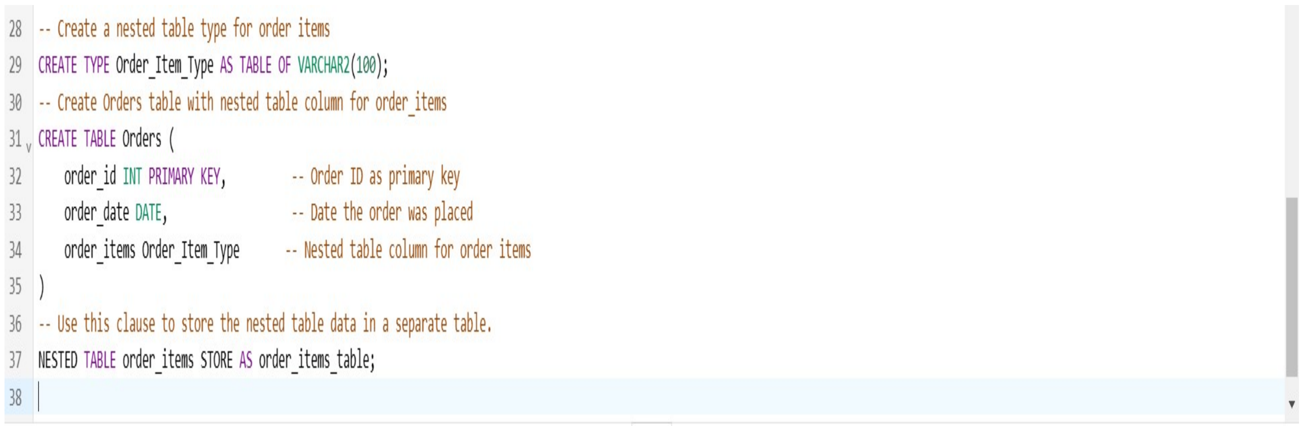
We define a nested table type called Order_Item_Type, which will hold a collection of Order Items.

```
-- Create a nested table type for order items
CREATE TYPE Order_Item_Type AS TABLE OF VARCHAR2(100);
```

Step 2: Create the Orders Table with Nested Table Column

Next, we create the Orders table. This table will include a column of type Order_Item_Type, which is the nested table.

```
CREATE TABLE Orders (
    order_id INT PRIMARY KEY,          -- Order ID as primary key
    order_date DATE,                  -- Date the order was placed
    order_items Order_Item_Type       -- Nested table column for order items
)
-- Use this clause to store the nested table data in a separate table.
NESTED TABLE order_items STORE AS order_items_table;
```


```
-- Create a nested table type for order items
CREATE TYPE Order_Item_Type AS TABLE OF VARCHAR2(100);
-- Create Orders table with nested table column for order_items
CREATE TABLE Orders (
    order_id INT PRIMARY KEY,         -- Order ID as primary key
    order_date DATE,                  -- Date the order was placed
    order_items Order_Item_Type      -- Nested table column for order items
)
-- Use this clause to store the nested table data in a separate table.
NESTED TABLE order_items STORE AS order_items_table;
```

Step 3: Insert Data into the Orders Table

We insert an order with multiple items into the Orders table. The items are provided as a list in the Order_Item_Type.

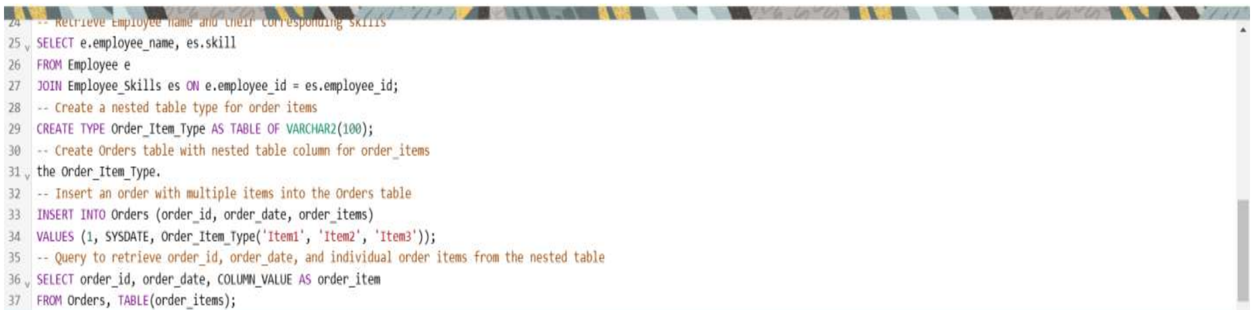
```
-- Insert an order with multiple items into the Orders table
INSERT INTO Orders (order_id, order_date, order_items)
VALUES (1, SYSDATE, Order_Item_Type('Item1', 'Item2', 'Item3'));
```

Step 4: Querying the Nested Table

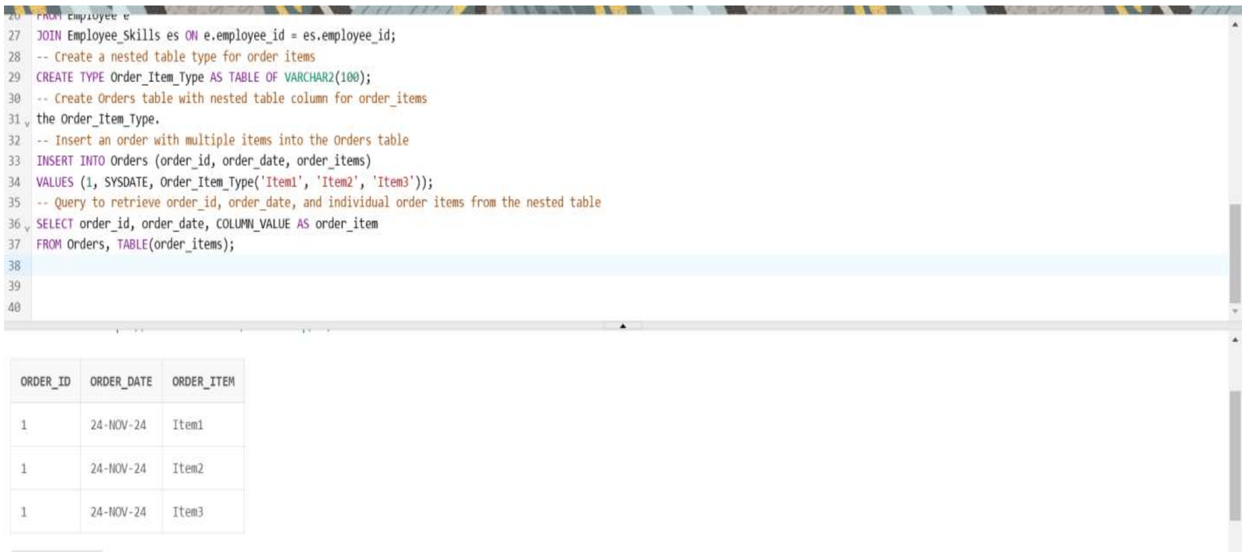
To retrieve the order and its items, we use the TABLE function to unnest the nested table and return each individual item.

```
-- Query to retrieve order_id, order_date, and individual order items from the nested table
SELECT order_id, order_date, COLUMN_VALUE AS order_item
FROM Orders, TABLE(order_items);
```

This query will give you each order_id and its corresponding items from the nested table.



```
-- Retrieve Employee name and their corresponding skills
SELECT e.employee_name, es.skill
FROM Employee e
JOIN Employee_Skills es ON e.employee_id = es.employee_id;
-- Create a nested table type for order items
CREATE TYPE Order_Item_Type AS TABLE OF VARCHAR2(100);
-- Create Orders table with nested table column for order_items
the Order_Item_Type.
-- Insert an order with multiple items into the Orders table
INSERT INTO Orders (order_id, order_date, order_items)
VALUES (1, SYSDATE, Order_Item_Type('Item1', 'Item2', 'Item3'));
-- Query to retrieve order_id, order_date, and individual order items from the nested table
SELECT order_id, order_date, COLUMN_VALUE AS order_item
FROM Orders, TABLE(order_items);
```



```sql
FROM Employee e
JOIN Employee_Skills es ON e.employee_id = es.employee_id;
-- Create a nested table type for order items
CREATE TYPE Order_Item_Type AS TABLE OF VARCHAR2(100);
-- Create Orders table with nested table column for order_items
the Order_Item_Type.
-- Insert an order with multiple items into the Orders table
INSERT INTO Orders (order_id, order_date, order_items)
VALUES (1, SYSDATE, Order_Item_Type('Item1', 'Item2', 'Item3'));
-- Query to retrieve order_id, order_date, and individual order items from the nested table
SELECT order_id, order_date, COLUMN_VALUE AS order_item
FROM Orders, TABLE(order_items);
```

ORDER_ID	ORDER_DATE	ORDER_ITEM
1	24-NOV-24	Item1
1	24-NOV-24	Item2
1	24-NOV-24	Item3