Complex types, Table Inheritance, Arrays

1. Complex Types (Structured Types):

Complex types, also known as structured types or user-defined types, allow you to define custom data types that encapsulate multiple attributes, potentially including other complex types.

Creation: Use CREATE TYPE to define a structured type.

```
CREATE TYPE Address_Type AS OBJECT (
        street VARCHAR(100),
        city VARCHAR(50),
        zip_code VARCHAR(10)
    );
```

Usage: You can then use this type as a column in a table.

```
CREATE TABLE Customers (
        customer_id INT PRIMARY KEY,
        name VARCHAR(100),
        address Address_Type
    );
```

Accessing attributes: Access individual components using dot notation.

```
SELECT c.name, c.address.city FROM Customers c;
```

2. Table Inheritance:

Table inheritance allows a table (subtable) to inherit the structure and behavior (columns, constraints, triggers) from another table (supertable). This promotes modularity and schema reuse.

Creation: Use INHERITS during table creation.

```
CREATE TABLE Persons (
        person_id INT PRIMARY KEY,
        name VARCHAR(100)
    );

    CREATE TABLE Employees (
        employee_id INT PRIMARY KEY,
        salary DECIMAL(10, 2)
    ) INHERITS (Persons);
```

Behavior: Queries on the supertable (e.g., Persons) can optionally include data from its subtables (e.g., Employees).

SELECT * FROM Persons; -- Might show data from both Persons and Employees

SELECT * FROM ONLY Persons; -- Shows only data explicitly in Persons

3. Arrays Implementation:

Arrays allow storing ordered collections of elements of the same data type within a single column, which can be useful for representing multi-valued attributes or lists.

Declaration: Declare an array type by appending [] to the data type.

```
CREATE TABLE Products (
        product_id INT PRIMARY KEY,
        product_name VARCHAR(100),
        tags VARCHAR(50)[] -- Array of strings
    );
```

INSERT INTO Products (product_id, product_name, tags) VALUES (1, 'Laptop', ARRAY['electronics', 'computer', 'portable']);

Querying: Use array functions and operators for querying.

SELECT product_name FROM Products WHERE 'electronics' = ANY(tags);

<u>Examples:</u>

1. Complex Types (Object Types)

Creating Object Types

```
sql
-- Create person object type
CREATE TYPE person_type AS OBJECT (
    person_id NUMBER,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    date_of_birth DATE,
    MEMBER FUNCTION get_age RETURN NUMBER,
    MEMBER FUNCTION get_full_name RETURN VARCHAR2
);
/

-- Create type body with member functions
CREATE TYPE BODY person_type AS
        MEMBER FUNCTION get_age RETURN NUMBER IS
             BEGIN
             RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, date_of_birth)/12);
END;
```

```
MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
        BEGIN
                RETURN first_name || ' ' || last_name;
        END;
END;
/
```

Using Object Types in Tables

```
CREATE TABLE employees (
    employee_id NUMBER PRIMARY KEY,
    employee_info person_type,
    work_address address_type,
    hire_date DATE
);

INSERT INTO employees VALUES (
    1,
    person_type(101, 'John', 'Doe', TO_DATE('1990-05-15', 'YYYY-MM-DD')),
    address_type('123 Main St', 'New York', 'NY', '10001', 'USA'),
    SYSDATE
);

INSERT INTO employees VALUES (
    2,
    person_type(102, 'Jane', 'Smith', TO_DATE('1985-08-20', 'YYYY-MM-DD')),
    address_type('456 Oak Ave', 'Los Angeles', 'CA', '90001', 'USA'),
    SYSDATE
);
SELECT
    employee_id,
    e.employee_info.get_full_name() as full_name,
    e.employee_info.get_age() as age,
    e.work_address.city as city
FROM employees e;
```

2. Table Inheritance (Using Object Types)

While Oracle 11g doesn't have true table inheritance like PostgreSQL, we can simulate it using object types and REFs.

```sql
CREATE TYPE person_obj AS OBJECT (
    person_id NUMBER,
    name VARCHAR2(100),
    email VARCHAR2(100)
) NOT FINAL;
/

CREATE TYPE student_obj UNDER person_obj (
    student_id NUMBER,
    major VARCHAR2(50),
    gpa NUMBER
);
/
-- Teacher type inheriting from person
CREATE TYPE teacher_obj UNDER person_obj (
    teacher_id NUMBER,
    department VARCHAR2(50),
    salary NUMBER
);
/
-- Create tables for each type
CREATE TABLE people OF person_obj;
CREATE TABLE students OF student_obj;
CREATE TABLE teachers OF teacher_obj;
-- Insert data
INSERT INTO people VALUES (person_obj(1, 'General Person', 'person@email.com')
);

INSERT INTO students VALUES (student_obj(2, 'Student Name', 'student@email.com', 1001
, 'Computer Science', 3.8));

INSERT INTO teachers VALUES (teacher_obj(3, 'Teacher Name', 'teacher@email.com', 2001
, 'Mathematics', 75000));

Polymorphic query:
SELECT VALUE(p) FROM people p
UNION ALL
SELECT VALUE(s) FROM students s
```

```
UNION ALL
SELECT VALUE(t) FROM teachers t;
```

3. Arrays (VARRAY and Nested Tables)

VARRAY Implementation

```
CREATE TYPE phone_list AS VARRAY(5) OF VARCHAR2(15);
/
CREATE TYPE skill_list AS VARRAY(10) OF VARCHAR2(50);
/

CREATE TABLE consultants (
    consultant_id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    phones phone_list,
    skills skill_list);

INSERT INTO consultants VALUES (
    1,
    'Alice Johnson',
    phone_list('555-0101', '555-0102'),
    skill_list('Java', 'Oracle', 'SQL', 'Python'));

INSERT INTO consultants VALUES (
    2,
    'Bob Wilson',
    phone_list('555-0201'),
    skill_list('C++', 'Linux', 'Networking'));

SELECT
    consultant_id,
    name,
    p.column_value as phone,
    s.column_value as skill
FROM consultants c,
TABLE(c.phones) p,
TABLE(c.skills) s;
```

Nested Tables Implementation

```sql
CREATE TYPE project_type AS OBJECT (
    project_id NUMBER,
    project_name VARCHAR2(100),
    start_date DATE,
    end_date DATE);
/

CREATE TYPE project_table AS TABLE OF project_type;
/

CREATE TABLE departments (
    dept_id NUMBER PRIMARY KEY,
    dept_name VARCHAR2(100),
    projects project_table) NESTED TABLE projects STORE AS projects_nt;

INSERT INTO departments VALUES (
    1,
    'IT Department',
    project_table(
        project_type(1, 'Database Migration', DATE '2023-01-01', DATE '2023-06-30'),
        project_type(2, 'System Upgrade', DATE '2023-03-01', DATE '2023-12-31')));

INSERT INTO departments VALUES (
    2,
    'HR Department',
    project_table(
       project_type(3, 'Recruitment System', DATE '2023-02-01', DATE '2023-08-31')));

SELECT
    d.dept_id,
    d.dept_name,
    p.project_id,
    p.project_name,
    p.start_date,
    p.end_date
FROM departments d,
TABLE(d.projects) p;
```

4. Advanced Example: Complex Employee Management System

```
CREATE TYPE detailed_address AS OBJECT (
    street VARCHAR2(100),
    city VARCHAR2(50),
    state VARCHAR2(2),
    zip_code VARCHAR2(10),
    MEMBER FUNCTION is_valid RETURN BOOLEAN,
    MEMBER FUNCTION format_address RETURN VARCHAR2
);
/

CREATE TYPE BODY detailed_address AS
    MEMBER FUNCTION is_valid RETURN BOOLEAN IS
    BEGIN
        RETURN street IS NOT NULL AND city IS NOT NULL
               AND state IS NOT NULL AND zip_code IS NOT NULL;
    END;

    MEMBER FUNCTION format_address RETURN VARCHAR2 IS
    BEGIN
        RETURN street || ', ' || city || ', ' || state || ' ' || zip_code;
    END;
END;
/

CREATE TYPE dependent_type AS OBJECT (
    dependent_id NUMBER,
    name VARCHAR2(100),
    relationship VARCHAR2(50),
    date_of_birth DATE
);
/

CREATE TYPE dependents_table AS TABLE OF dependent_type;
/
```

Main Table employee table with complex types:

```
CREATE TABLE company_employees (
    emp_id NUMBER PRIMARY KEY,
    basic_info person_type,
    address detailed_address,
    emergency_contacts phone_list,
    dependents dependents_table,
    hire_date DATE,
    salary NUMBER
) NESTED TABLE dependents STORE AS dependents_nt;

INSERT INTO company_employees VALUES (
    1001,
    person_type(1001, 'Michael', 'Brown', TO_DATE('1980-03-15', 'YYYY-MM-DD')),
    detailed_address('789 Corporate Blvd', 'Chicago', 'IL', '60601'),
    phone_list('555-1001', '555-1002'),
    dependents_table(
        dependent_type(1, 'Sarah Brown', 'Spouse', TO_DATE('1982-07-20', 'YYYY-MM-DD'
)),
        dependent_type(2, 'Emily Brown', 'Daughter', TO_DATE('2010-11-05', 'YYYY-MM-D
D'))
    ),
    DATE '2020-01-15',
    85000
);
```

Complex queries

```
SELECT
    emp_id,
    e.basic_info.get_full_name() as employee_name,
    e.address.format_address() as full_address,
    d.name as dependent_name,
    d.relationship
FROM company_employees e,
TABLE(e.dependents) d
WHERE e.basic_info.get_age() > 30;
```

5. Utility Functions for Complex Types:

Function to calculate total dependents

```
CREATE OR REPLACE FUNCTION get_dependent_count(p_emp_id NUMBER)
```

```
RETURN NUMBER IS
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM TABLE(SELECT dependents FROM company_employees WHERE emp_id = p_emp_id);

    RETURN v_count;
END;
/
```

Procedure to add new dependent

```
CREATE OR REPLACE PROCEDURE add_dependent(
    p_emp_id IN NUMBER,
    p_dep_id IN NUMBER,
    p_name IN VARCHAR2,
    p_relationship IN VARCHAR2,
    p_dob IN DATE
) IS
BEGIN
    UPDATE company_employees e
    SET e.dependents = e.dependents MULTISET UNION ALL dependents_table(
        dependent_type(p_dep_id, p_name, p_relationship, p_dob)
    )
    WHERE e.emp_id = p_emp_id;

    COMMIT;
END;
/
```

Validate the functions

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Dependents for employee 1001: ' || get_dependent_count(1001
));
```

Adding a new dependent

```
    add_dependent(1001, 3, 'James Brown', 'Son', DATE '2015-05-10');
    DBMS_OUTPUT.PUT_LINE('Dependents after addition: ' || get_dependent_count(1001));
END;/
```

Lab Exercises

1. **Create a university database** with:

- Student, Professor, and Staff types inheriting from Person
- Use VARRAY for student courses
- Use nested tables for professor research projects

1. Create Base Object Types

```
-- Create Address type
CREATE TYPE address_type AS OBJECT (
    street VARCHAR2(100),
    city VARCHAR2(50),
    state VARCHAR2(50),
    zip_code VARCHAR2(10),
    country VARCHAR2(50)
);
/

-- Create base Person type (NOT FINAL for inheritance)
CREATE TYPE person_type AS OBJECT (
    person_id NUMBER,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    date_of_birth DATE,
    email VARCHAR2(100),
    phone VARCHAR2(15),
    address address_type,

    -- Member functions
    MEMBER FUNCTION get_age RETURN NUMBER,
    MEMBER FUNCTION get_full_name RETURN VARCHAR2
) NOT FINAL;
/

-- Implement Person type body
CREATE TYPE BODY person_type AS
    MEMBER FUNCTION get_age RETURN NUMBER IS
    BEGIN
        RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, date_of_birth)/12);
    END get_age;

    MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN first_name || ' ' || last_name;
    END get_full_name;
END;
/
```

Create Course and Research Project Types

```
CREATE TYPE course_type AS OBJECT (
    course_code VARCHAR2(10),
    course_name VARCHAR2(100),
    credits NUMBER,
    semester VARCHAR2(20),
    grade VARCHAR2(2)
);
/
```

Create VARRAY for student courses (max 6 courses per student)

```
CREATE TYPE course_list AS VARRAY(6) OF course_type;
/
```

Create Research Project type for nested table

```
CREATE TYPE research_project_type AS OBJECT (
    project_id NUMBER,
    project_name VARCHAR2(100),
    funding_amount NUMBER,
    start_date DATE,
    end_date DATE,
    status VARCHAR2(20)
);
/
```

Create nested table type for research projects

```
CREATE TYPE project_list AS TABLE OF research_project_type;
```

```
/
```

Create Inherited Types

Student type inheriting from Person

```
CREATE TYPE student_type UNDER person_type (
    student_id VARCHAR2(20),
    enrollment_date DATE,
    major VARCHAR2(50),
    current_semester VARCHAR2(20),
    gpa NUMBER,
    courses course_list,  -- VARRAY for courses

    -- Override member function
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2
);
/

CREATE TYPE BODY student_type AS
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN 'Student: ' || first_name || ' ' || last_name || ' (ID: ' || student_i
d || ')';
    END get_full_name;
END;
/
```

Professor type inheriting from Person

```
CREATE TYPE professor_type UNDER person_type (
    professor_id VARCHAR2(20),
    department VARCHAR2(50),
    salary NUMBER,
    hire_date DATE,
    office_number VARCHAR2(20),
    research_projects project_list,  -- Nested table for research projects
```

Override member function

```
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2,
```

Additional member functions

```
    MEMBER FUNCTION get_years_of_service RETURN NUMBER,
    MEMBER FUNCTION get_total_research_funding RETURN NUMBER
);/
```

Implement Professor type body

```
CREATE TYPE BODY professor_type AS
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN 'Prof. ' || first_name || ' ' || last_name || ' (ID: ' || professor_id || ')';
    END get_full_name;

    MEMBER FUNCTION get_years_of_service RETURN NUMBER IS
    BEGIN
        RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, hire_date)/12);
    END get_years_of_service;

    MEMBER FUNCTION get_total_research_funding RETURN NUMBER IS
        v_total NUMBER := 0;
    BEGIN
        IF research_projects IS NOT NULL THEN
            FOR i IN 1..research_projects.COUNT LOOP
                v_total := v_total + research_projects(i).funding_amount;
            END LOOP;
        END IF;
        RETURN v_total;
    END get_total_research_funding;
END;
/
```

Staff type inheriting from Person

```
CREATE TYPE staff_type UNDER person_type (
    staff_id VARCHAR2(20),
    position VARCHAR2(50),
    department VARCHAR2(50),
    salary NUMBER,
    hire_date DATE,
    -- Override member function
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2,

    -- Additional member function
    MEMBER FUNCTION get_years_of_service RETURN NUMBER
);
/
-- Implement Staff type body
CREATE TYPE BODY staff_type AS
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN 'Staff: ' || first_name || ' ' || last_name || ' (ID: ' || staff_id ||
')';
    END get_full_name;

    MEMBER FUNCTION get_years_of_service RETURN NUMBER IS
    BEGIN
        RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, hire_date)/12);
    END get_years_of_service;
END;
/
```

Create Tables

Create table for Students

```
CREATE TABLE students OF student_type;
```

Create table for Professors with nested table storage

```
CREATE TABLE professors OF professor_type
NESTED TABLE research_projects STORE AS professor_projects_nt;
```

Create table for Staff

```
CREATE TABLE staff OF staff_type;
```

Inserting Sample Data in to the table

Insert Students with VARRAY courses

```
INSERT INTO students VALUES (
    student_type(
```

```sql
        1001,                                    -- person_id
        'John',                                  -- first_name
        'Smith',                                 -- last_name
        TO_DATE('2000-05-15', 'YYYY-MM-DD'),    -- date_of_birth
        'john.smith@university.edu',             -- email
        '555-0101',                              -- phone
        address_type('123 College Ave', 'Boston', 'MA', '02115', 'USA'), -- address
        'S1001',                                 -- student_id
        DATE '2022-09-01',                       -- enrollment_date
        'Computer Science',                      -- major
        'Fall 2024',                             -- current_semester
        3.75,                                    -- gpa
        course_list(                             -- VARRAY courses
            course_type('CS101', 'Introduction to Programming', 3, 'Fall 2024', 'A'),
            course_type('MATH201', 'Calculus II', 4, 'Fall 2024', 'B+'),
            course_type('PHY101', 'Physics I', 4, 'Fall 2024', 'A-'),
            course_type('ENG101', 'English Composition', 3, 'Fall 2024', 'A')
        )
    )
);

INSERT INTO students VALUES (
    student_type(
        1002,
        'Emily',
        'Johnson',
        TO_DATE('2001-08-22', 'YYYY-MM-DD'),
        'emily.johnson@university.edu',
        '555-0102',
        address_type('456 University St', 'Boston', 'MA', '02116', 'USA'),
        'S1002',
        DATE '2023-01-15',
        'Electrical Engineering',
        'Fall 2024',
        3.60,
        course_list(
            course_type('EE201', 'Circuit Analysis', 4, 'Fall 2024', 'B'),
            course_type('MATH202', 'Differential Equations', 4, 'Fall 2024', 'A-'),
            course_type('CS102', 'Data Structures', 3, 'Fall 2024', 'B+'),
            course_type('CHEM101', 'General Chemistry', 4, 'Fall 2024', 'B')
        )
    )
);

INSERT INTO students VALUES (
    student_type(
        1003,
        'Michael',
        'Brown',
        TO_DATE('1999-12-10', 'YYYY-MM-DD'),
        'michael.brown@university.edu',
        '555-0103',
        address_type('789 Campus Road', 'Cambridge', 'MA', '02138', 'USA'),
        'S1003',
        DATE '2021-09-01',
        'Mathematics',
        'Fall 2024',
        3.90,
        course_list(
            course_type('MATH301', 'Advanced Calculus', 4, 'Fall 2024', 'A'),
            course_type('MATH305', 'Linear Algebra', 3, 'Fall 2024', 'A'),
            course_type('CS201', 'Algorithms', 3, 'Fall 2024', 'A-')
        )
    )
);

-- Insert Professors with nested table research projects
INSERT INTO professors VALUES (
    professor_type(
        2001,
        'Sarah',
        'Wilson',
        TO_DATE('1975-03-18', 'YYYY-MM-DD'),
        'sarah.wilson@university.edu',
        '555-0201',
        address_type('321 Faculty Lane', 'Cambridge', 'MA', '02139', 'USA'),
        'P2001',
        'Computer Science',
```

```sql
        95000,
        DATE '2010-08-15',
        'CS-301',
        project_list(  -- Nested table research projects
            research_project_type(1, 'Machine Learning Optimization', 150000,
                DATE '2023-01-01', DATE '2024-12-31', 'Active'),
            research_project_type(2, 'Natural Language Processing', 200000,
                DATE '2022-06-01', DATE '2025-05-31', 'Active'),
            research_project_type(3, 'Computer Vision Research', 120000,
                DATE '2021-03-01', DATE '2023-12-31', 'Completed')
        )
    )
);

INSERT INTO professors VALUES (
    professor_type(
        2002,
        'Robert',
        'Chen',
        TO_DATE('1968-11-25', 'YYYY-MM-DD'),
        'robert.chen@university.edu',
        '555-0202',
        address_type('654 Academic Way', 'Boston', 'MA', '02117', 'USA'),
        'P2002',
        'Electrical Engineering',
        110000,
        DATE '2005-01-10',
        'EE-205',
        project_list(
            research_project_type(4, 'Renewable Energy Systems', 300000,
                DATE '2024-01-01', NULL, 'Active'),
            research_project_type(5, 'Smart Grid Technology', 250000,
                DATE '2022-09-01', DATE '2024-08-31', 'Active')
        )
    )
);

INSERT INTO professors VALUES (
    professor_type(
        2003,
        'Lisa',
        'Garcia',
        TO_DATE('1972-07-08', 'YYYY-MM-DD'),
        'lisa.garcia@university.edu',
        '555-0203',
        address_type('987 Research Blvd', 'Cambridge', 'MA', '02140', 'USA'),
        'P2003',
        'Mathematics',
        85000,
        DATE '2015-03-20',
        'MATH-102',
        project_list(
            research_project_type(6, 'Number Theory Applications', 80000,
                DATE '2023-09-01', DATE '2025-08-31', 'Active'),
            research_project_type(7, 'Mathematical Modeling', 95000,
                DATE '2022-01-15', DATE '2023-12-15', 'Completed')
        )
    )
);

-- Insert Staff
INSERT INTO staff VALUES (
    staff_type(
        3001,
        'Jennifer',
        'Davis',
        TO_DATE('1985-09-14', 'YYYY-MM-DD'),
        'jennifer.davis@university.edu',
        '555-0301',
        address_type('111 Admin Street', 'Boston', 'MA', '02118', 'USA'),
        'ST3001',
        'Administrative Assistant',
        'Computer Science',
        55000,
        DATE '2018-03-15'
    )
);
```

```sql
INSERT INTO staff VALUES (
    staff_type(
        3002,
        'David',
        'Martinez',
        TO_DATE('1978-04-30', 'YYYY-MM-DD'),
        'david.martinez@university.edu',
        '555-0302',
        address_type('222 Service Road', 'Cambridge', 'MA', '02141', 'USA'),
        'ST3002',
        'IT Support Specialist',
        'Information Technology',
        65000,
        DATE '2015-06-01'
    )
);

INSERT INTO staff VALUES (
    staff_type(
        3003,
        'Maria',
        'Thompson',
        TO_DATE('1990-12-05', 'YYYY-MM-DD'),
        'maria.thompson@university.edu',
        '555-0303',
        address_type('333 Office Park', 'Boston', 'MA', '02119', 'USA'),
        'ST3003',
        'Library Assistant',
        'Library Services',
        48000,
        DATE '2020-02-10'
    ));
```

Example Query Execution

```sql
-- Query 1: Display all students with their courses (using VARRAY)
SELECT
    s.student_id,
    s.get_full_name() as student_name,
    s.major,
    s.gpa,
    c.course_code,
    c.course_name,
    c.credits,
    c.grade
FROM students s,
TABLE(s.courses) c
ORDER BY s.student_id, c.course_code;

-- Query 2: Display professors with their research projects (using nested table)
SELECT
    p.professor_id,
    p.get_full_name() as professor_name,
    p.department,
    r.project_name,
    r.funding_amount,
    r.start_date,
    r.end_date,
    r.status
FROM professors p,
TABLE(p.research_projects) r
ORDER BY p.professor_id, r.project_id;

-- Query 3: Show all university members using inheritance
SELECT 'Student' as role, student_id as id, get_full_name() as name, major as departm
ent FROM students
UNION ALL
SELECT 'Professor' as role, professor_id as id, get_full_name() as name, department
FROM professors
UNION ALL
SELECT 'Staff' as role, staff_id as id, get_full_name() as name, department
FROM staff;

-- Query 4: Professor Research funding summary
SELECT
    professor_id,
    get_full_name() as professor_name,
    department,
    get_total_research_funding() as total_funding,
    get_years_of_service() as years_of_service
FROM professors;

-- Query 5: Student course load by major
SELECT
    major,
    COUNT(*) as student_count,
    AVG(gpa) as average_gpa,
    SUM((SELECT COUNT(*) FROM TABLE(s.courses))) as total_courses
FROM students s
GROUP BY major;

-- Query 6: Active research projects with details
SELECT
    p.get_full_name() as professor,
    p.department,
    r.project_name,
    r.funding_amount,
    CASE
        WHEN r.end_date IS NULL THEN 'Ongoing'
        WHEN r.end_date > SYSDATE THEN 'Active'
        ELSE 'Completed'
    END as project_status
FROM professors p,
TABLE(p.research_projects) r
WHERE r.status = 'Active' OR (r.end_date IS NULL OR r.end_date > SYSDATE)
ORDER BY r.funding_amount DESC;
```

****