Requirements:

Logical database design and implementation

1. Modify class diagram if it is required. Check additional requirements for your group.
2. Create logical schema in your group's Oracle account according to specifications:
    i. Logical schema must contain at least 10 object tables resulted from the conversion of your class diagram to SQL statements. Create types, type bodies (where it is appropriate) and object tables. Some of your types are to have REF properties.
    ii. Introduce at least 10 member methods:
        1. a) At least one of the methods should be MAP method.
        2. b) At least one of the method should be ORDER method.
        3. c) Introduce one method (other than MAP or ORDER) in a supertype. Override this method in one of the subtypes of the supertype.
    iii. You may need to introduce additional attributes for primary keys in association classes describing many-to-many associations.
3. Create sample data and populate your object tables (three-four objects for each object table).
4. Demonstrate that your object tables are populated.

Working with your database

Formulate SQL queries based on object-relational approach. Prepare a file with SQL queries described below and execute queries against your database from the file:

1. Formulate the specified query (I will formulate this query based on your modified class diagram).
2. Formulate a query demonstrating that MAP method declared in the corresponding data type works. You are to use ORDER BY VALUE(...) clause.
3. Formulate a query demonstrating that ORDER method declared in the corresponding data type works. You may use ORDER BY VALUE(...) clause or call the method directly.
4. Formulate a query or two demonstrating that the method from the supertype which was overridden in a subtype works. If you formulate two queries, they will be counted as one.
5. Formulate additional 9 queries. All together you need to formulate 13 queries including mine.
6. All queries are to be based on object-relational features and retrieve data from more than one object table. Query diversity is important.

Implementation:

Objects:

```
CREATE OR REPLACE TYPE RecreationalVehicle AS OBJECT (
  rv_id integer,
  make varchar2(50),
  model varchar2(50),
  year number,
  price number,
  availability varchar2(15),
  rvCondition varchar(15),

  CONSTRUCTOR FUNCTION RecreationalVehicle(
    rv_id integer,
    make varchar2,
    model varchar2,
    year number,
    price number,
    availability varchar2,
    rvCondition varchar2
  ) RETURN SELF AS RESULT,

  MEMBER FUNCTION calculateDeprication RETURN integer,
  MEMBER FUNCTION displayDetails RETURN varchar2,
  MEMBER FUNCTION displayAvailability RETURN varchar2,
  map member function comps return integer
) NOT FINAL;
/

CREATE OR REPLACE TYPE UsedRV UNDER RecreationalVehicle (
  mileage integer,
  previous_owners integer,
  MEMBER FUNCTION calculateExpectedLife RETURN integer,
  OVERRIDING member function calculateDeprication RETURN integer
);
/

CREATE OR REPLACE TYPE NewRV UNDER RecreationalVehicle (
  warranty_period integer,
  MEMBER FUNCTION calculateWarrantyEndYear RETURN integer
);
/
```

```
CREATE OR REPLACE TYPE DemoRV UNDER RecreationalVehicle (
  mileage integer,
  price_discount integer,
  CONSTRUCTOR FUNCTION DemoRV(
    rv_id integer,
    make varchar2,
    model varchar2,
    year number,
    price number,
    availability varchar2,
    rvCondition varchar2,
    mileage integer,
    price_discount integer
  ) RETURN SELF AS RESULT
);
/


CREATE OR REPLACE TYPE Employee AS OBJECT (
  employee_id INTEGER,
  employee_name VARCHAR2(50),
  position VARCHAR2(50),
  contact INTEGER,
  rv_id ref RecreationalVehicle,
  MEMBER FUNCTION listAvailableEmployee RETURN VARCHAR2,
  MEMBER FUNCTION searchRv(rv_id integer) RETURN BOOLEAN
);
/

CREATE OR REPLACE TYPE RVCustomer AS OBJECT (
  customer_id INTEGER,
  first_name VARCHAR2(50),
  last_name VARCHAR2(50),
  dob DATE,
  email VARCHAR2(100),
  phone_number INTEGER,
  employee_id ref Employee,
  MEMBER FUNCTION displayCustomerDetails RETURN VARCHAR2
);
/

CREATE OR REPLACE TYPE Payment AS OBJECT (
    payment_id INTEGER,
    amount INTEGER,
   payment_date DATE,
   MEMBER FUNCTION validatePayment RETURN BOOLEAN
);
/

CREATE OR REPLACE TYPE RVOrder AS OBJECT (
   order_id INTEGER,
   order_date DATE,
   customer_id ref RVCustomer,
   payment_id ref Payment,
   rv_id ref RecreationalVehicle,
   MEMBER FUNCTION calculateOrderTotal RETURN INTEGER
);
/

CREATE OR REPLACE TYPE Supplier AS OBJECT (
   supplier_id INTEGER,
   name VARCHAR2(50),
   address VARCHAR2(100),
   phone_number VARCHAR2(20),
   rv_id ref RecreationalVehicle,
   MEMBER FUNCTION ListSuppliedPart RETURN VARCHAR2,
   MEMBER FUNCTION displaySupplierAddress RETURN VARCHAR2
);
/

CREATE OR REPLACE TYPE RVServiceAppointment AS OBJECT (
   appointment_id INTEGER,
   service_date DATE,
   service_type VARCHAR2(50),
   rv_id ref RecreationalVehicle,
   MEMBER FUNCTION isServiceOverdue RETURN BOOLEAN,
   MEMBER FUNCTION getRvDetails RETURN VARCHAR2,
   ORDER MEMBER FUNCTION sort(appt in RVServiceAppointment) RETURN INTEGER
);
/

CREATE OR REPLACE TYPE RVServiceCentre AS OBJECT (
   location_id INTEGER,
   address VARCHAR2(100),
   contact VARCHAR2(20),
   appointment_id ref RVServiceAppointment,
   MEMBER FUNCTION listAvailableServices RETURN VARCHAR2
);
```

```
/

CREATE OR REPLACE TYPE RVPart AS OBJECT (
   part_id INTEGER,
   part_name VARCHAR2(50),
   price INTEGER,
   appointment_id REF RVServiceAppointment,
   location_id REF RVServiceCentre,
   MEMBER FUNCTION checkPartAvailability RETURN BOOLEAN
);
/

save group3_milestone#2_version6.sql
Created file group3_milestone#2_version6.sql
```

Tables:

```
CREATE TABLE RV_Table OF RecreationalVehicle (
  PRIMARY KEY (rv_id)
);

CREATE TABLE UsedRV_Table OF UsedRV (
  PRIMARY KEY (rv_id)
);


CREATE TABLE NewRV_Table OF NewRV (
  PRIMARY KEY (rv_id)
);

CREATE TABLE DemoRV_Table OF DemoRV (
  PRIMARY KEY (rv_id)
);

CREATE TABLE Employee_Table OF Employee (
  PRIMARY KEY (employee_id)
);

CREATE TABLE RVCustomer_Table OF RVCustomer (
  PRIMARY KEY (customer_id)
);

CREATE TABLE Payment_Table OF Payment (
  PRIMARY KEY (payment_id)
);

CREATE TABLE RVOrder_Table OF RVOrder (
  PRIMARY KEY (order_id)
);

CREATE TABLE Supplier_Table OF Supplier (
  PRIMARY KEY (supplier_id)
);

CREATE TABLE RVServiceAppointment_Table OF RVServiceAppointment (
  PRIMARY KEY (appointment_id)
);

CREATE TABLE RVServiceCentre_Table OF RVServiceCentre (
  PRIMARY KEY (location_id)
);

CREATE TABLE RVPart_Table OF RVPart (
  PRIMARY KEY (part_id)
);
```

Type bodies:

```
CREATE OR REPLACE TYPE BODY RecreationalVehicle AS
  CONSTRUCTOR FUNCTION RecreationalVehicle(
    rv_id INTEGER,
    make VARCHAR2,
    model VARCHAR2,
    year NUMBER,
    price NUMBER,
    availability VARCHAR2,
    rvCondition VARCHAR2
  ) RETURN SELF AS RESULT IS
  BEGIN
    SELF.rv_id := rv_id;
    SELF.make := make;
    SELF.model := model;
    SELF.year := year;
    SELF.price := price;
    SELF.availability := availability;
    SELF.rvCondition := rvCondition;
    RETURN;
  END;
  MEMBER FUNCTION calculateDeprication RETURN INTEGER IS
    depreciation INTEGER;
  BEGIN
    -- For simplicity, let's assume 10% depreciation per year
    depreciation := ROUND(price * (SELF.year / 10));
    RETURN depreciation;
  END;

  MEMBER FUNCTION displayDetails RETURN VARCHAR2 IS
    details VARCHAR2(500);
  BEGIN
    details := 'RV ID: ' || TO_CHAR(rv_id) || ', Make: ' || make || ', Model: ' || model || ', Year: '
|| TO_CHAR(year) || ', Price: $' || TO_CHAR(price) || ', Availability: ' || availability || ',
Condition: ' || rvCondition;
    RETURN details;
  END;

  MEMBER FUNCTION displayAvailability RETURN VARCHAR2 IS
  BEGIN
    RETURN availability;
  END;

  MAP MEMBER FUNCTION comps RETURN INTEGER IS
  BEGIN
    -- Calculate a value based on the year
    RETURN 5 + (EXTRACT(YEAR FROM SYSDATE) - year);
  END;
END;
/

CREATE OR REPLACE TYPE BODY UsedRV AS
  OVERRIDING MEMBER FUNCTION calculateDeprication RETURN INTEGER IS
  BEGIN
    -- Assuming 15% depreciation per year for used RVs
    RETURN ROUND((SYSDATE - TO_DATE('01-JAN-' || TO_CHAR(SELF.year), 'DD-MON-YYYY')) /
365) * (SELF.price * 0.15);
  END;

  MEMBER FUNCTION calculateExpectedLife RETURN INTEGER IS
  BEGIN
    -- Assuming average lifespan of 15 years for used RVs
    RETURN 15;
  END;
```

```
END;
/

CREATE OR REPLACE TYPE BODY NewRV AS
  MEMBER FUNCTION calculateWarrantyEndYear RETURN integer IS
  BEGIN
    -- Assuming warranty period is 3 years
    RETURN year + 3; -- Adding 3 years to the current year to calculate the end year of the
warranty
  END;
END;
/

CREATE OR REPLACE TYPE BODY DemoRV AS
  CONSTRUCTOR FUNCTION DemoRV(
    rv_id integer,
    make varchar2,
    model varchar2,
    year number,
    price number,
    availability varchar2,
    rvCondition varchar2,
    mileage integer,
    price_discount integer
  ) RETURN SELF AS RESULT IS
  BEGIN
    SELF.rv_id := rv_id;
    SELF.make := make;
    SELF.model := model;
    SELF.year := year;
    SELF.price := price;
    SELF.availability := availability;
    SELF.rvCondition := rvCondition;
    SELF.mileage := mileage;
    SELF.price_discount := price_discount;
    RETURN;
  END;
END;
/

CREATE OR REPLACE TYPE BODY Employee AS
  MEMBER FUNCTION listAvailableEmployee RETURN VARCHAR2 IS
  BEGIN
    -- Return list of available employees
    RETURN 'Employee ID: ' || TO_CHAR(employee_id) || ', Name: ' || employee_name || ',
Position: ' || position || ', Contact: ' || TO_CHAR(contact);
  END;

  -- Define searchRv function in the type specification
  MEMBER FUNCTION searchRv(rv_id integer) RETURN BOOLEAN IS
    rv_count INTEGER;
  BEGIN
    -- Implement search logic for RV by ID
    SELECT COUNT(*)
    INTO rv_count
    FROM RV_Table
    WHERE rv_id = rv_id;

    RETURN rv_count > 0; -- Return TRUE if RV ID exists, FALSE otherwise
  END;
END;
/

CREATE OR REPLACE TYPE BODY RVCustomer AS
  MEMBER FUNCTION displayCustomerDetails RETURN VARCHAR2 IS
    customer_info VARCHAR2(500); -- Adjusted length constraint
  BEGIN
    -- Implement logic to display customer details
    customer_info := 'Customer ID: ' || TO_CHAR(customer_id) || ', Name: ' || first_name || ' '
|| last_name || ', Date of Birth: ' || TO_CHAR(dob, 'YYYY-MM-DD') || ', Email: ' || email || ',
Phone Number: ' || TO_CHAR(phone_number);
    RETURN customer_info;
  END;
END;
/

CREATE OR REPLACE TYPE BODY Payment AS
  MEMBER FUNCTION validatePayment RETURN BOOLEAN IS
    payment_valid BOOLEAN;
  BEGIN
    -- Implement logic to validate payment
    -- For simplicity, let's assume all payments are valid
    payment_valid := TRUE;
    RETURN payment_valid;
  END;
END;
/
```

```
CREATE OR REPLACE TYPE BODY RVOrder AS
  MEMBER FUNCTION calculateOrderTotal RETURN INTEGER IS
    order_total INTEGER := 0;
  BEGIN
    -- Implement logic to calculate order total
    -- For simplicity, let's assume the order total is the sum of all payments associated with the
order
    SELECT COALESCE(SUM(amount), 0)
    INTO order_total
    FROM Payment_Table
    WHERE order_id = SELF.order_id;

    RETURN order_total;
  END;
END;
/


CREATE OR REPLACE TYPE BODY Supplier AS
  MEMBER FUNCTION ListSuppliedPart RETURN VARCHAR2 IS
    supplied_parts_info VARCHAR2(500); -- Adjusted length constraint
  BEGIN
    -- Implement logic to list supplied parts
    -- For simplicity, let's assume all suppliers provide parts
    supplied_parts_info := 'Supplier ID: ' || TO_CHAR(supplier_id) || ', Name: ' || name || ',
Address: ' || address || ', Phone Number: ' || phone_number || CHR(10);
    RETURN supplied_parts_info;
  END;

  MEMBER FUNCTION displaySupplierAddress RETURN VARCHAR2 IS
  BEGIN
    -- Implement logic to display supplier address
    RETURN 'Supplier Address: ' || address;
  END;
END;
/

CREATE OR REPLACE TYPE BODY RVServiceCentre AS
  MEMBER FUNCTION listAvailableServices RETURN VARCHAR2 IS
  BEGIN
    -- Implement logic to list available services at the service center
    -- For simplicity, let's assume all services are available
    RETURN 'Service Center Address: ' || address || ', Contact: ' || contact;
  END;
```

```
END;
/

CREATE OR REPLACE TYPE BODY RVPart AS
  MEMBER FUNCTION checkPartAvailability RETURN BOOLEAN IS
    is_available BOOLEAN;
  BEGIN
    -- For simplicity, let's assume all parts are available
    is_available := TRUE;
    RETURN is_available;
  END;
END;
/


CREATE OR REPLACE TYPE BODY RVServiceAppointment AS
  MEMBER FUNCTION isServiceOverdue RETURN BOOLEAN IS
    service_overdue BOOLEAN;
    last_service_date DATE;
  BEGIN
    -- Retrieve the last service date of the RV
    SELECT MAX(service_date)
    INTO last_service_date
    FROM RVServiceAppointment_Table
    WHERE rv_id = SELF.rv_id;

    -- Check if the current date is past the recommended service date
    IF last_service_date IS NULL THEN
      service_overdue := TRUE; -- No previous service, so service is overdue
    ELSE
      service_overdue := last_service_date < (SYSDATE - INTERVAL '1' YEAR); -- Assuming annual
service interval
    END IF;

    RETURN service_overdue;
  END;

  MEMBER FUNCTION getRvDetails RETURN VARCHAR2 IS
    rv_details VARCHAR2(500);
    rv_ref RecreationalVehicle;
  BEGIN
    -- Retrieve details of the RV
    SELECT DEREF(rv_id)
    INTO rv_ref
    FROM RVServiceAppointment_Table
    WHERE appointment_id = SELF.appointment_id;

    -- Format RV details
    rv_details := 'RV ID: ' || rv_ref.rv_id || ', Make: ' || rv_ref.make || ', Model: ' || rv_ref.model
|| ', Year: ' || rv_ref.year || ', Price: $' || TO_CHAR(rv_ref.price) || ', Availability: ' ||
rv_ref.availability || ', Condition: ' || rv_ref.rvCondition;

    RETURN rv_details;
  END;

  ORDER MEMBER FUNCTION sort(appt IN RVServiceAppointment) RETURN INTEGER IS
  BEGIN
    -- Compare appointment dates for sorting
    IF SELF.service_date < appt.service_date THEN
      RETURN -1;
    ELSIF SELF.service_date > appt.service_date THEN
      RETURN 1;
    ELSE
      RETURN 0;
    END IF;
  END;
END;
/
```

Inserting rows:

```
INSERT INTO RV_Table VALUES (
  NewRV(1, 'Brand1', 'Model1', 2024, 50000, 'Available', 'New', 3)
);

INSERT INTO RV_Table VALUES (
  NewRV(2, 'Brand2', 'Model2', 2024, 45000, 'Available', 'New', 5)
);

INSERT INTO RV_Table VALUES (
  NewRV(3, 'Brand3', 'Model1', 2024, 56102, 'Unavailable ', 'New', 7)
);

INSERT INTO RV_Table VALUES (
  NewRV(4, 'Brand3', 'ModelZ', 2024, 36372, 'Available ', 'New', 9)
);

INSERT INTO RV_Table VALUES (
  NewRV(13, 'BrandP', 'ModelZ', 2025, 36372, 'Unavailable ', 'New', 9)
);


INSERT INTO NewRV_Table VALUES (
  NewRV(1, 'Brand1', 'Model1', 2024, 50000, 'Available', 'New', 3)
);

INSERT INTO NewRV_Table VALUES (
  NewRV(2, 'Brand2', 'Model2', 2024, 45000, 'Available', 'New', 5)
);

INSERT INTO NewRV_Table VALUES (
  NewRV(3, 'Brand3', 'Model1', 2024, 56102, 'Unavailable ', 'New', 7)
);

INSERT INTO NewRV_Table VALUES (
  NewRV(4, 'Brand3', 'ModelZ', 2024, 36372, 'Available ', 'New', 9)
);

INSERT INTO NewRV_Table VALUES (
  NewRV(13, 'BrandP', 'ModelZ', 2025, 36372, 'Unavailable ', 'New', 9)
);


INSERT INTO RV_Table VALUES (
  UsedRV(5, 'Brand1', 'Model2', 2023, 26382, 'Available', 'Used', 30000, 1)
);

INSERT INTO RV_Table VALUES (
  UsedRV(6, 'Brand2', 'ModelX', 2022, 26832, 'Available', 'Used', 25000, 2)
);

INSERT INTO RV_Table VALUES (
  UsedRV(7, 'Brand3', 'ModelZ', 2021, 32527, 'Available ', 'Used', 35000, 3)
);

INSERT INTO RV_Table VALUES (
  UsedRV(8, 'Brand4', 'ModelQ', 2023, 36338, 'Available ', 'Used', 32000, 1)
);

INSERT INTO UsedRV_Table VALUES (
  UsedRV(5, 'Brand1', 'Model2', 2023, 26382, 'Available', 'Used', 30000, 1)
);
```

```
INSERT INTO UsedRV_Table VALUES (
  UsedRV(6, 'Brand2', 'ModelX', 2022, 26832, 'Available', 'Used', 25000, 2)
);

INSERT INTO UsedRV_Table VALUES (
  UsedRV(7, 'Brand3', 'ModelZ', 2021, 32527, 'Available ', 'Used', 35000, 3)
);

INSERT INTO UsedRV_Table VALUES (
  UsedRV(8, 'Brand4', 'ModelQ', 2023, 36338, 'Available ', 'Used', 32000, 1)
);


-- Inserting initial DemoRV data into RV_Table for DemoRV

INSERT INTO RV_Table VALUES (
  DemoRV(9, 'DemoZ', 'DemoModelZ', 2023, 31000, 'Available', 'Demo', 21000, 1100)
);

INSERT INTO RV_Table VALUES (
  DemoRV(10, 'DemoX', 'DemoModel3', 2023, 36000, 'Unavailable', 'Demo', 16000, 900)
);

INSERT INTO RV_Table VALUES (
  DemoRV(11, 'DemoV', 'DemoModel9', 2022, 42000, 'Unavailable', 'Demo', 26000, 1300)
);

INSERT INTO RV_Table VALUES (
  DemoRV(12, 'Demo4', 'DemoModel4', 2022, 43000, 'Available', 'Demo', 19000, 1000)
);

-- Inserting overridden data into DemoRV_Table
INSERT INTO DemoRV_Table VALUES (
  DemoRV(9, 'Demo3', 'DemoModel5', 2023, 39000, 'Available', 'Demo', 20000, 1000)
);

INSERT INTO DemoRV_Table VALUES (
  DemoRV(10, 'Demo2', 'DemoModel3', 2023, 37000, 'Unavailable', 'Demo', 15000, 800)
);

INSERT INTO DemoRV_Table VALUES (
  DemoRV(11, 'Demo3', 'DemoModel9', 2022, 42282, 'Unavailable', 'Demo', 25000, 1200)
);
```

```
INSERT INTO DemoRV_Table VALUES (
  DemoRV(12, 'Demo4', 'DemoModel4', 2022, 44000, 'Available', 'Demo', 18000, 900)
);

-- Insert statements for Employee_Table
INSERT INTO Employee_Table VALUES (
  Employee(1, 'John Doe', 'Manager', 1234567890, NULL)
);

INSERT INTO Employee_Table VALUES (
  Employee(2, 'Jane Smith', 'Salesperson', 9876543210, NULL)
);

INSERT INTO Employee_Table VALUES (
  Employee(3, 'Michael Johnson', 'Technician', 5554443333, NULL)
);

INSERT INTO Employee_Table VALUES (
  Employee(4, 'Emily Davis', 'Receptionist', 1112223333, NULL)
);

-- Insert statements for RVCustomer_Table
INSERT INTO RVCustomer_Table VALUES (
  RVCustomer(1, 'Alice', 'Johnson', TO_DATE('1990-05-15', 'YYYY-MM-DD'),
'alice@example.com', 5551112222, (SELECT REF(e) FROM Employee_Table e WHERE
e.employee_id = 1))
);

INSERT INTO RVCustomer_Table VALUES (
  RVCustomer(2, 'Bob', 'Smith', TO_DATE('1985-08-25', 'YYYY-MM-DD'), 'bob@example.com',
5553334444, (SELECT REF(e) FROM Employee_Table e WHERE e.employee_id = 2))
);

INSERT INTO RVCustomer_Table VALUES (
  RVCustomer(3, 'Carol', 'Williams', TO_DATE('1978-12-10', 'YYYY-MM-DD'),
'carol@example.com', 5555556666, (SELECT REF(e) FROM Employee_Table e WHERE
e.employee_id = 3))
);

INSERT INTO RVCustomer_Table VALUES (
  RVCustomer(4, 'David', 'Brown', TO_DATE('1982-03-20', 'YYYY-MM-DD'),
'david@example.com', 5557778888, (SELECT REF(e) FROM Employee_Table e WHERE
e.employee_id = 4))
);

-- Insert statements for Payment_Table
INSERT INTO Payment_Table VALUES (
  Payment(1, 5000, SYSDATE)
);

INSERT INTO Payment_Table VALUES (
  Payment(2, 7000, SYSDATE)
);

INSERT INTO Payment_Table VALUES (
  Payment(3, 4500, SYSDATE)
);

INSERT INTO Payment_Table VALUES (
  Payment(4, 6000, SYSDATE)
);

-- Insert statements for RVOrder_Table
INSERT INTO RVOrder_Table VALUES (
  RVOrder(2, SYSDATE, (SELECT REF(rc) FROM RVCustomer_Table rc WHERE rc.customer_id = 2),
(SELECT REF(p) FROM Payment_Table p WHERE p.payment_id = 2), (SELECT REF(rv) FROM
RV_Table rv WHERE rv.rv_id = 2))
);

INSERT INTO RVOrder_Table VALUES (
  RVOrder(3, SYSDATE, (SELECT REF(rc) FROM RVCustomer_Table rc WHERE rc.customer_id = 3),
(SELECT REF(p) FROM Payment_Table p WHERE p.payment_id = 3), (SELECT REF(rv) FROM
RV_Table rv WHERE rv.rv_id = 3))
);

INSERT INTO RVOrder_Table VALUES (
  RVOrder(4, SYSDATE, (SELECT REF(rc) FROM RVCustomer_Table rc WHERE rc.customer_id = 4),
(SELECT REF(p) FROM Payment_Table p WHERE p.payment_id = 4), (SELECT REF(rv) FROM
RV_Table rv WHERE rv.rv_id = 4))
);

INSERT INTO RVOrder_Table VALUES (
  RVOrder(1, SYSDATE, (SELECT REF(rc) FROM RVCustomer_Table rc WHERE rc.customer_id = 1),
(SELECT REF(p) FROM Payment_Table p WHERE p.payment_id = 1), (SELECT REF(rv) FROM
RV_Table rv WHERE rv.rv_id = 1))
);
```

```
-- Insert statements for Supplier_Table
INSERT INTO Supplier_Table VALUES (
  Supplier(1, 'RV Parts Supplier', '123 Main St, Anytown, USA', '555-123-4567', (SELECT REF(rv)
FROM RV_Table rv WHERE rv.rv_id = 1))
);

INSERT INTO Supplier_Table VALUES (
  Supplier(2, 'RV Accessories Supplier', '456 Elm St, Othertown, USA', '555-987-6543', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 2))
);

INSERT INTO Supplier_Table VALUES (
  Supplier(3, 'RV Service Supplier', '789 Oak St, Anycity, USA', '555-222-3333', (SELECT REF(rv)
FROM RV_Table rv WHERE rv.rv_id = 3))
);

INSERT INTO Supplier_Table VALUES (
  Supplier(4, 'RV Rental Supplier', '101 Pine St, Somewhere, USA', '555-444-5555', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 4))
);

-- Insert statements for RVServiceAppointment_Table
INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(1, TO_DATE('2024-02-20', 'YYYY-MM-DD'), 'Annual Maintenance',
(SELECT REF(rv) FROM RV_Table rv WHERE rv.rv_id = 1))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(2, TO_DATE('2024-02-21', 'YYYY-MM-DD'), 'Repair', (SELECT REF(rv)
FROM RV_Table rv WHERE rv.rv_id = 2))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(3, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Inspection', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 3))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(4, TO_DATE('2024-02-23', 'YYYY-MM-DD'), 'Tire Replacement', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 4))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(5, TO_DATE('2024-02-24', 'YYYY-MM-DD'), 'Oil Change', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 5))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(6, TO_DATE('2024-02-25', 'YYYY-MM-DD'), 'Winterization', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 6))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(7, TO_DATE('2024-02-26', 'YYYY-MM-DD'), 'Interior Cleaning', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 7))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(8, TO_DATE('2024-02-27', 'YYYY-MM-DD'), 'Battery Check', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 8))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(9, TO_DATE('2024-02-28', 'YYYY-MM-DD'), 'Tire Rotation', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 9))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(10, TO_DATE('2024-02-29', 'YYYY-MM-DD'), 'Generator Inspection',
(SELECT REF(rv) FROM RV_Table rv WHERE rv.rv_id = 10))
);

INSERT INTO RVServiceAppointment_Table VALUES (
  RVServiceAppointment(11, TO_DATE('2024-03-01', 'YYYY-MM-DD'), 'Awning Repair', (SELECT
REF(rv) FROM RV_Table rv WHERE rv.rv_id = 11))
);


-- Insert statements for RVServiceCentre_Table
INSERT INTO RVServiceCentre_Table VALUES (
  RVServiceCentre(1, 'RV Service Center 1 Address', '555-111-2222', NULL)
);

INSERT INTO RVServiceCentre_Table VALUES (
  RVServiceCentre(2, 'RV Service Center 2 Address', '555-333-4444', NULL)
);
```

```
INSERT INTO RVServiceCentre_Table VALUES (
  RVServiceCentre(3, 'RV Service Center 3 Address', '555-555-6666', NULL)
);

INSERT INTO RVServiceCentre_Table VALUES (
  RVServiceCentre(4, 'Richmond hill Service Centre', '555-777-8888', NULL)
);

INSERT INTO RVServiceCentre_Table VALUES (
  RVServiceCentre(5, 'Thornhill Service Centre', '555-777-8888', NULL)
);


-- Insert statements for RVPart_Table
INSERT INTO RVPart_Table VALUES (
  RVPart(1, 'Engine Oil', 50, (SELECT REF(app) FROM RVServiceAppointment_Table app WHERE
app.appointment_id = 1), (SELECT REF(sc) FROM RVServiceCentre_Table sc WHERE
sc.location_id = 5))
);

INSERT INTO RVPart_Table VALUES (
  RVPart(2, 'Air Filter', 20, (SELECT REF(app) FROM RVServiceAppointment_Table app WHERE
app.appointment_id = 2), (SELECT REF(sc) FROM RVServiceCentre_Table sc WHERE
sc.location_id = 5))
);

INSERT INTO RVPart_Table VALUES (
  RVPart(3, 'Brake Pads', 100, (SELECT REF(app) FROM RVServiceAppointment_Table app
WHERE app.appointment_id = 3), (SELECT REF(sc) FROM RVServiceCentre_Table sc WHERE
sc.location_id = 5))
);

INSERT INTO RVPart_Table VALUES (
  RVPart(4, 'Tire', 200, (SELECT REF(app) FROM RVServiceAppointment_Table app WHERE
app.appointment_id = 4), (SELECT REF(sc) FROM RVServiceCentre_Table sc WHERE
sc.location_id = 5))
);

INSERT INTO RVPart_Table VALUES (
  RVPart(5, 'Router', 200, (SELECT REF(app) FROM RVServiceAppointment_Table app WHERE
app.appointment_id = 4), (SELECT REF(sc) FROM RVServiceCentre_Table sc WHERE
sc.location_id = 1))
);
```

```
INSERT INTO RVPart_Table VALUES (
  RVPart(6, 'Rim', 200, (SELECT REF(app) FROM RVServiceAppointment_Table app WHERE
app.appointment_id = 4), (SELECT REF(sc) FROM RVServiceCentre_Table sc WHERE
sc.location_id = 2))
);
```

Queries:

1. Find models of recreation vehicles which were serviced with parts available at Thornhill Service centre.

```
SELECT DISTINCT rv.model
FROM RV_Table rv
JOIN RVServiceAppointment_Table sa ON sa.rv_id = REF(rv)
JOIN RVPart_Table rp ON rp.appointment_id = REF(sa)
JOIN RVServiceCentre_Table sc ON rp.location_id = REF(sc)
WHERE sc.address = 'Thornhill Service Centre';
```

2. Formulate a query demonstrating that MAP method declared in the corresponding data type works. You are to use ORDER BY VALUE(...) clause.

```
SELECT rv.rv_id, rv.make, rv.model, rv.year, rv.price, rv.availability, rv.rvCondition, rv.comps()
AS num_components
FROM RV_Table rv
ORDER BY rv.comps() DESC;
```

Or for different view:

```
SELECT
   'RV_ID: ' || LPAD(rv.rv_id, 5) AS "RV_ID",
   'MAKE: ' || LPAD(rv.make, 15) AS "MAKE",
   'MODEL: ' || LPAD(rv.model, 30) AS "MODEL",
   'YEAR: ' || LPAD(rv.year, 5) || '   PRICE: $' || TO_CHAR(rv.price, '99999.99') AS
"YEAR   PRICE",
   'AVAILABILITY: ' || LPAD(rv.availability, 15) || '   RVCONDITION: ' || LPAD(rv.rvCondition, 10)
AS "AVAILABILITY   RVCONDITION",
   'NUM_COMPONENTS: ' || LPAD(rv.comps(), 3) AS "NUM_COMPONENTS"
FROM
   RV_Table rv
ORDER BY
```

```
    "NUM_COMPONENTS" DESC,
    "RV_ID";
```

3. Formulate a query demonstrating that ORDER method declared in the corresponding data type works. You may use ORDER BY VALUE(...) clause or call the method directly.

```
SELECT sa_sorted.appointment_id, sa_sorted.service_date, sa_sorted.service_type, rv.model
FROM (
    SELECT sa.*, VALUE(sa).sort(RVServiceAppointment(0, SYSDATE, 'Dummy', NULL)) as order_value
    FROM RVServiceAppointment_Table sa
) sa_sorted
JOIN RV_Table rv ON sa_sorted.rv_id = REF(rv);
```

4. Formulate a query or two demonstrating that the method from the supertype which was overridden in a subtype works. If you formulate two queries, they will be counted as one.

```
COLUMN RV_ID FORMAT 999
COLUMN MAKE FORMAT A15
COLUMN MODEL FORMAT A30
COLUMN YEAR FORMAT 9999
COLUMN PRICE FORMAT 99999
COLUMN DEPRECIATION FORMAT 999999999.99 HEADING 'DEPRECIATION'

SELECT
    rv.rv_id,
    rv.make,
    rv.model,
    rv.year,
    TO_CHAR(rv.price, '99999') AS price,
    TO_CHAR(rv.calculateDeprication(), '999999999.99') AS DEPRECIATION
FROM
    UsedRV_Table rv;
```

Or different view:

```
SELECT rv.rv_id, rv.make, rv.model, rv.year, rv.price, rv.calculateDeprication() AS depreciation
```

```
FROM UsedRV_Table rv;
```

5. Formulate additional 9 queries. All together you need to formulate 13 queries including mine.

   1. List all recreational vehicles (RVs) and their service appointments:

```
SELECT
   RPAD(NVL(TO_CHAR(rv.rv_id), 'NULL'), 5) AS "RV_ID",
   RPAD(NVL(rv.make, 'NULL'), 15) AS "MAKE",
   RPAD(NVL(rv.model, 'NULL'), 30) AS "MODEL",
   RPAD(NVL(TO_CHAR(sa.appointment_id), 'NULL'), 15) AS "APPOINTMENT_ID",
   NVL(TO_CHAR(sa.service_date, 'DD-MON-YY'), 'NULL') AS "SERVICE_DATE"
FROM
   RV_Table rv
LEFT JOIN
   RVServiceAppointment_Table sa ON sa.rv_id = REF(rv);
```

2. Display RV with the calculated warranty end year for new RVs.

```
SELECT
   rv.rv_id,
   rv.make,
   rv.model,
   rv.year,
   rv.price,
   rv.displayDetails() AS rv_details,
   n.calculateWarrantyEndYear() AS warranty_end_year
FROM
   RV_Table rv
JOIN
   NewRV_Table n ON rv.rv_id = n.rv_id;
```

3. Retrieve supplier information along with the RV details they are associated with:

```
SELECT
   s.supplier_id,
   s.name,
   s.displaySupplierAddress() AS supplier_address,
   DEREF(s.rv_id).displayDetails() AS rv_details
```

```
FROM
   Supplier_Table s;
```

4. List the details of used RVs along with their expected life:

```
SELECT
   rv.displayDetails() AS rv_details,
   u.calculateExpectedLife() AS expected_life
FROM
   UsedRV_Table u
JOIN
   RV_Table rv ON u.rv_id = rv.rv_id;
```

5. Retrieve RVs and their corresponding payment details:

```
SELECT
   rv.rv_id,
   rv.make,
   rv.model,
   p.amount,
   p.payment_date
FROM
   RV_Table rv
JOIN
   RVOrder_Table o ON rv.rv_id = o.rv_id.rv_id
JOIN
   Payment_Table p ON o.payment_id.payment_id = p.payment_id;
```

6. List all RV orders with customer details and total payment amount:

```
SELECT
   o.order_id,
   o.order_date,
   DEREF(o.customer_id).displayCustomerDetails() AS customer_details,
   o.calculateOrderTotal() AS total_payment_amount
FROM
   RVOrder_Table o;
```

7. list all RV parts manufactured in Thornhill that are under $100:

```
SELECT

  pt.part_name,

  pt.price

FROM

  RVPart_Table pt

JOIN

  RVServiceCentre_Table sc ON DEREF(pt.location_id).location_id = sc.location_id

WHERE

  sc.address = 'Thornhill Service Centre' AND pt.price < 100;
```

8. List the average RV purchase price for customers under 40:

```
SELECT
  AVG(rv.price) AS avgU40yo_purchase_price
FROM
  RVOrder_Table ord
JOIN
  RVCustomer_Table cust ON ord.customer_id = REF(cust)
JOIN
  RV_Table rv ON ord.rv_id = REF(rv)
WHERE
  EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM cust.dob) < 40;
```

9. List all RVs with their orders, along with customer details and payment information:

```
SELECT
    rv.rv_id,
    rv.make,
    rv.model,
    o.order_id,
    c.displayCustomerDetails() AS customer_details,
    p.amount AS payment_amount,
    p.payment_date
FROM
    RV_Table rv
```

```
JOIN
   RVOrder_Table o ON rv.rv_id = o.rv_id.rv_id
JOIN
   RVCustomer_Table c ON o.customer_id.customer_id = c.customer_id
JOIN
   Payment_Table p ON o.payment_id.payment_id = p.payment_id;
```

10. Retrieve the total number of service appointments scheduled for each RV model:

```
SELECT
   rv.model,
   COUNT(sa.appointment_id) AS total_service_appointments
FROM
   RV_Table rv
LEFT JOIN
   RVServiceAppointment_Table sa ON rv.rv_id = sa.rv_id.rv_id
GROUP BY
   rv.model;
```

11. Retrieve all service appointment Details for RV with ModelalQ

```
SELECT
   sa.appointment_id,
   sa.service_date,
   DEREF(sa.rv_id).make as make ,
   DEREF(sa.rv_id).model as model
FROM
   RVServiceAppointment_Table sa
WHERE
   DEREF(sa.rv_id).make = 'Brand4' AND DEREF(sa.rv_id).model = 'ModelQ';
```

12. list showing the employee details along with the corresponding customer details for each employee

```
SELECT
   LPAD(e.employee_name, 50) AS EMPLOYEE_NAME,
   LPAD(e.position, 50) AS POSITION,
   LPAD(c.first_name || ' ' || c.last_name, 50) AS CUSTOMER_NAME
FROM
   Employee_Table e
JOIN
   RVCustomer_Table c ON e.employee_id = c.employee_id.employee_id;
```

Extra Queries:

13. Return RVinfo for all used vehicle built after 2020 in order of least mileage

```
SELECT
        rv_id, make, model, mileage, year
FROM
        UsedRV_Table
WHERE
        year > 2020
ORDER BY
        mileage ASC;
```

14. Finding the availability and condition of RV with model Z

```
        SELECT
                availability, rvCondition
        FROM
                RV_Table
        WHERE
                model = 'ModelZ';
```