

Part 1 - To Remain with the Assignment after Marking

Student ID: E11261	Student Name: Belpa Gamage Ranuga Disansa
Module Code: CI6320	Module Name: Advanced Data Modelling
Assignment number: 1	ESoft Module Leader: Mr. W A D B C Goonatillaka
Date set: 16th of March 2024	Date due: 7th of April 2024

Guidelines for the Submission of Coursework

1. Print this cover sheet and securely attach both pages to your assignment. You can help us ensure work is marked more quickly by submitting at the specified location for your module. You are advised to keep a copy of every assignment.
2. Coursework deadlines are strictly enforced by the University.
3. You should not leave the handing in of work until the last minute. Once an assignment has been submitted it cannot be submitted again.

Academic Misconduct: Plagiarism and/or **collusion** constitute **academic misconduct** under the University's Academic Regulations. Examples of academic misconduct in coursework: making available your work to other students; presenting work produced in collaboration with other students as your own (unless an explicit assessment requirement); submitting work, taken from sources that are not properly referenced, as your own. By printing and submitting this coversheet with your coursework you are confirming that the work is your own.

ESoft Office Use Only:

Date stamp: work received.

Part 2 – Student Feedback

Student ID:	Student Name:
Module Code:	Module Name:
Assignment number:	ESoft Module Leader:
Date set:	Date due:

Strengths (areas with well-developed answers)

Weaknesses (areas with room for improvement)

Additional Comments

ESoft Module Lecturer:

Provisional mark as %:

ESoft Module Marker:

Date marked:

Acknowledgement

I want to express my appreciation to those who have helped complete this report.

I have been able to complete this report because of the Academic advisor Mr. W A D B C Goonatillaka for their invaluable support, mentorship, and feedback and the faculty members of CI6320(Advanced Data Modelling).

Additionally, I would like to thank my family members for their unwavering encouragement and help throughout the report.

Thank you.

Table of Contents

Part A	8
1. Introduction	8
1.1 What is data modelling?	8
1.2 Importance of Data Models	8
2. Relational Data Model (RDM)	8
2.1 History	8
2.2 Core Principles	8
2.3 Characteristics	8
3. Object-Oriented Data Model (OODM)	9
3.1 History	9
3.2 Core Principles	9
3.3 Characteristics	9
4. Object-Relational Data Model (ORDM)	9
4.1 History	9
4.2 Core Principles	9
4.3 Characteristics	9
5. Summary	10
5.1 Comparison Table	10
5.2 Critical Discussion on Which Model to use in real world application scenarios.	10
Part B.....	11
1. Setting Up the Development Environment	11
2. Publication	12
2.1 Type Creation	12
2.2 Body Creation	13
2.3 Table Creation	14
2.4 Data Insertion	15
2.5 Data Retrieval.....	16
3. Book.....	17
3.1 Type Creation.....	17
3.2 Body Creation	18
3.3 Table Creation	19
3.4 Data Insertion	20

4. Journal	21
4.1 Type Creation	21
4.2 Body Creation	22
4.3 Table Creation	22
4.4 Data Insertion	23
5. Member.....	24
5.1 Type Creation	24
5.2 Body Creation	25
5.3 Table Creation	26
5.4 Data Insertion	27
6. Loans.....	28
6.1 Table Creation	28
6.2 Data Insertion	29
7. Additional Features	30
7.1 Additional attributes for Publications.....	30
7.2 Loan Details Viewer.....	31
References.....	33

Table of Figures

Figure 1Oracle Database Version	11
Figure 2Creating / Replacing Type Publication Execution	12
Figure 3Creating / Replacing Type Publication SQL.....	12
Figure 4Creating / Replacing Publication Type Body Execution.....	13
Figure 5Creating / Replacing Publication Body SQL	13
Figure 6Creating Publications Table Execution	14
Figure 7Creating Publications Table SQL.....	14
Figure 8Inserting Data into the Publications Table Execution.....	15
Figure 9Inserting into the Publications Table SQL.....	15
Figure 10Retriving Data from Publications Table Execution	16
Figure 11Retriving Data from Publications Table SQL	16
Figure 12Alternating the Type Publications to be extended.....	17
Figure 13Creating Type Book Execution.....	17
Figure 14Creating Type Book SQL	17
Figure 15Creating / Replacing Body Type Execution.....	18
Figure 16Creating / Replacing Body Type SQL	18
Figure 17Create Table Books Execution.....	19
Figure 18Create Table Books SQL	19
Figure 19Inserting into the Books Table Execution.....	20
Figure 20Inserting into the Books Table SQL	20
Figure 21 Alternating the Type Publications to be extended.....	21
Figure 22Creating Type Journal Execution	21
Figure 23Creating Type Journal SQL.....	21
Figure 24Creating / Replacing Journal Type Body Execution.....	22
Figure 25Creating / Replacing Journal Type Bodyy SQL	22
Figure 26Creating Journals Table Execution	22
Figure 27Creating / Replacing Journal Table SQL.....	23
Figure 28Inserting Data into the Journals Table Execution.....	23
Figure 29Creating Type Memeber Execution	24
Figure 30Creating Type Member SQL	24
Figure 31Creating / Replacing Member Type Body Execution	25
Figure 32Creating / Replacing Member Type Body SQL.....	25
Figure 33Creating Members Table Execution	26
Figure 34Creating Memebers Table SQL	26
Figure 35Inserting Data into the Members Table Execution	27
Figure 36Inserting Data into the Members Table SQL.....	27
Figure 37Creating Table Loans Execution	28
Figure 38Creating Table Loans SQL.....	28
Figure 39Inserting Data into the Loans Table Execution	29
Figure 40Inserting Data into the Loans Table SQL.....	29
Figure 41Altering Publications Table Execution	30
Figure 42Altering Publications Table SQL.....	30
Figure 43Loan Details Viewer Execution.....	31

Figure 44Loan Details Viewer SQL32

Part A

1. Introduction

1.1 What is data modelling?

A data model is a visual blueprint used to design databases or software systems depicting data entities and their relationships. It helps businesses order and organize their data effectively, making it easier to design or re-engineer databases that align with business and application requirements. A Data model helps by creating a bridge between business and technical teams by incorporating real-world objects into a structure for a database. A data model is a critical first step after defining business requirements. (IBM, no date; Staff, 2023; G, 2018; DASCAs, no date)

1.2 Importance of Data Models

- Documentation
- Ensure data integrity.
- Higher Quality
- Decision Making

2. Relational Data Model (RDM)

2.1 History

The relational data model was introduced by E.F. Codd at IBM in San Jose, where a new data representation framework called the relational data model was established. It suggested that all the data could be stored in a tabular structure, in turn leading to higher productivity in the early 1980s (Newcomb and Couch, 2010). It is taken into consideration as the landmark or the start of database systems. (Khan, no date)

2.2 Core Principles

The core principle of real data models is that data is organized in tables with columns and rows. Each table has rows, and each row is considered an instance of a relation. The columns are the attributes that are characteristics of the data. The model is built by implementing a computer presentation of mathematical theories of set theory and predicate logic. Relationships are used to store information about objects in a database. (Khan, no date.)

2.3 Characteristics

- Table based structure.
- Atomic values
- Normalization
- SQL

3. Object-Oriented Data Model (OODM)

3.1 History

The object-oriented Data model was created to define operations for designing schemas, creating databases, retrieving objects and navigating, while also having additional object-oriented principles such as aggregation, generalization and particularization relationships (Zhao, 1988).

3.2 Core Principles

The object-oriented data model represents the real world as objects with problems, attributes, and relationships. OODM was created by combining relational data model concepts with object-oriented programming principles (GFG, 2021). This approach allows classes to be grouped into items with comparable qualities, vacillating the organization and management of data structures while allowing for a smooth transition from the design concept to implementation in object-oriented databases (Janecatalla, 2012; Alzahrani, 2016).

3.3 Characteristics

- Classes which are similar to blueprints can create objects which is instances of a class (GFG, 2021)
- Allows for features such as inheritance which allows subclasses to inherit attributes and methods from classes (**GFG 2021**), which is useful for code reuse.
- Operations are performed on the data that is encapsulated by the objects.

4. Object-Relational Data Model (ORDM)

4.1 History

With the limitations of both the relational and object-oriented data models, research in the 1990s led to the development of the object-oriented data model, which takes fundamental concepts from both the relational and object-oriented data model while addressing areas where improvement was sought. (Castro, 2020)

4.2 Core Principles

The Object-oriented model combines the important features of both the relational and object-oriented data models, and it has extracted important core principles such as Supporting objects, classes, and inheritance from the Object-oriented data model and data types and tables from the Relational Data Model (Castro, 2020; Auziņš, 2018).

4.3 Characteristics

- supports complex Data Types such as Arrays, Nested Tables, and user-defined types.
- Has object-oriented principles in combination with the features of the relational data model, in turn allowing for the creation of much more advanced objects with relational principles.
- Creates a data model which has the most important features to be able to model the real world while having the flexibility to represent complex relationships and structures.

5. Summary

5.1 Comparison Table

Feature	Object Oriented Data Model (OODM)	Relational Data Model (RDM)	Object Relational Data Model (ORDM)
Data Representation	Objects with attributes and methods	Tables with rows and columns	Tables with rows, columns, and some OO concepts (inheritance, complex data types)
Relationships	Inheritance, Aggregation, and Association	Foreign Keys	Foreign Keys and some OO concepts (inheritance)
Performance	Potentially faster due to no joins	Can be efficient for specific queries	Can be efficient for specific queries but may be slower for complex relationships
Advantages	Code reuse (inheritance) and Semantic modelling easier to model complex relationships	flexible and efficient for certain queries Secure and Scalable	Combines benefits of OO and Relational models. Supports complex data types: Inheritance
Disadvantages	no strong mathematical foundation Difficulty with persistence for complex structures	can be complex to design for large data sets not ideal for complex querying	can be complex to manage; may not be as performant as pure OO for complex relationships

5.2 Critical Discussion on Which Model to use in real world application scenarios.

Selecting a data model for real-world applications relies heavily on a comprehensive understanding of the specific requirements, constraints, and characteristics of the application. Relational Data Models (RDM) are extremely effective in scenarios where data consistency and integrity are important, data for example a banking systems. Object-oriented models (OODM) have extremely rare use cases because they are useful in complex data modelling scenarios such as the natural representation of entities with a lot of behaviours and relationships in applications such as multimedia or gaming. Object Relational Data Models (ORDM) have a balance between flexibility and data integrity, making them extremely useful for hybrid scenarios such as social media platforms and e-commerce systems. The choice between the different data models requires careful evaluations of the advantages and disadvantages of each model, considering factors such as complexity, development, and requirements.

Part B

1. Setting Up the Development Environment

To set up the development environment to execute the following given programs: First, make sure that Oracle Database and Developer are installed on the development machine. Then configure the connection with the Oracle database, and then it would be possible to replicate the following programs. The following attached figure shows the Oracle database version that was used in development.

```
SQL> SELECT * FROM v$version;

BANNER
-----
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Product
PL/SQL Release 10.2.0.1.0 - Production
CORE      10.2.0.1.0      Production
TNS for 32-bit Windows: Version 10.2.0.1.0 - Production
NLSRTL Version 10.2.0.1.0 - Production
```

Figure 1 Oracle Database Version

2. Publication

2.1 Type Creation

```
SQL> CREATE OR REPLACE TYPE Publication AS OBJECT (  
  2     title VARCHAR2(255),  
  3     publication_date DATE,  
  4     publication_type VARCHAR2(20),  
  5     -- Define member function to display basic information  
  6     MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2  
  7 );  
  8 /  
  
Type created.
```

Figure 2 Creating / Replacing Type Publication Execution

In this given SQL script creates or replaces a User Defined Type (UDT) called 'Publication'. This type has 3 attributes which are: 'title' with VARCHAR (255) datatype, 'publication_date' with DATE datatype and 'publication_type' with VARCHAR2(20) datatype, and finally this defines a function called 'displayBasicInfo()' which is expected to of type 'VARCHAR2'.

```
1 -- Define Publication as an object type  
2 CREATE OR REPLACE TYPE Publication AS OBJECT (  
3     title VARCHAR2(255),  
4     publication_date DATE,  
5     publication_type VARCHAR2(20),  
6     -- Define member function to display basic information  
7     MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2  
8 );
```

Figure 3 Creating / Replacing Type Publication SQL

2.2 Body Creation

```
SQL> -- Define body for Publication type
SQL> CREATE OR REPLACE TYPE BODY Publication AS
2   -- Implementation of the displayBasicInfo function
3   MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2 IS
4   BEGIN
5       RETURN 'Title: ' || title || ', Publication Date: ' || TO_CHAR(publication_date, 'DD-MON-YYYY') || ', Type:
' || publication_type;
6   END displayBasicInfo;
7 END;
8 /

Type body created.
```

Figure 4 Creating / Replacing Publication Type Body Execution

This block of SQL code defines the body type for the type `Publication`. It has the following components:

1. `CREATE OR REPLACE TYPE BODY Publication AS`: This statement either creates or replaces the Body for the Type Publication
2. `MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2 IS`: This statement creates a member function called `displayBasicInfo()` which returns the data type of `VARCHAR2`.
3. `BEGIN`: This marks the start of the function implementation
4. `RETURN 'Title: ' || title || ', Publication Date: ' || TO_CHAR(publication_date, 'DD-MON-YYYY') || ', Type: ' || publication_type;`: This SQL statement returns a concatenated string with details such as title, publication_date and publication_type
5. `END displayBasicInfo;`: Marks the end of the displayBasicInfo function definition.

```
1 -- Define body for Publication type
2 CREATE OR REPLACE TYPE BODY Publication AS
3   -- Implementation of the displayBasicInfo function
4   MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2 IS
5   BEGIN
6       RETURN 'Title: ' || title || ', Publication Date: ' || TO_CHAR(publication_date, 'DD-MON-YYYY') || ', Type: ' || publication_type;
7   END displayBasicInfo;
8 END;
9
```

Figure 5 Creating / Replacing Publication Body SQL

2.3 Table Creation

```
SQL> CREATE TABLE Publications (  
2     publication_id NUMBER PRIMARY KEY,  
3     title VARCHAR2(255),  
4     publication_date DATE,  
5     publication_type VARCHAR2(20),  
6     -- Ensure publication_type is either 'Book' or 'Journal'  
7     CONSTRAINT chk_pub_type CHECK (publication_type IN ('Book', 'Journal'))  
8 );
```

Table created.

Figure 6 Creating Publications Table Execution

The given SQL script creates a table named `Publications` with four columns: `publication_id`, `title`, `publication_date` and `publication_type`. This table contains a constraint named `chk_pub_type` that ensures that the column will only contain the values `Book` or `Journal`.

```
1 -- Create table for Publications  
2 CREATE TABLE Publications (  
3     publication_id NUMBER PRIMARY KEY,  
4     title VARCHAR2(255),  
5     publication_date DATE,  
6     publication_type VARCHAR2(20),  
7     -- Ensure publication_type is either 'Book' or 'Journal'  
8     CONSTRAINT chk_pub_type CHECK (publication_type IN ('Book', 'Journal'))  
9 );  
10
```

Figure 7 Creating Publications Table SQL

2.4 Data Insertion

```
SQL> INSERT INTO Publications VALUES (1, 'The Great Gatsby', TO_DATE('2023-05-10', 'YYYY-MM-DD'), 'Book');

1 row created.

SQL> INSERT INTO Publications VALUES (2, 'National Geographic', TO_DATE('2023-04-15', 'YYYY-MM-DD'), 'Journal');

1 row created.

SQL> INSERT INTO Publications VALUES (3, 'Harry Potter and the Philosopher's Stone', TO_DATE('2023-06-20', 'YYYY-MM-DD'), 'Book');

1 row created.
```

Figure 8 Inserting Data into the Publications Table Execution

These 3 statements enter 3 records into the Publications Table, where each INSERT INTO inserts one row of data. Each value corresponds to a column in the table, and there are 4 columns, so each insertion has 4 values corresponding to each column. The TO_DATE function is used to convert strings into the DATE data type/structure.

```
1 -- Inserting data for the publication titled 'The Great Gatsby' into the Publications table
2 INSERT INTO Publications VALUES (
3     1, -- publication_id
4     'The Great Gatsby', -- title
5     TO_DATE('2023-05-10', 'YYYY-MM-DD'), -- publication_date
6     'Book' -- publication_type
7 );
8
9 -- Inserting data for the publication titled 'National Geographic' into the Publications table
10 INSERT INTO Publications VALUES (
11     2, -- publication_id
12     'National Geographic', -- title
13     TO_DATE('2023-04-15', 'YYYY-MM-DD'), -- publication_date
14     'Journal' -- publication_type
15 );
16
17 -- Inserting data for the publication titled 'Harry Potter and the Philosopher's Stone' into the Publications table
18 INSERT INTO Publications VALUES (
19     3, -- publication_id
20     'Harry Potter and the Philosopher's Stone', -- title
21     TO_DATE('2023-06-20', 'YYYY-MM-DD'), -- publication_date
22     'Book' -- publication_type
23 );
24
```

Figure 9 Inserting into the Publications Table SQL

2.5 Data Retrieval

```
SQL> SELECT p.title, Publication(p.title, p.publication_date, p.publication_type).displayBasicInfo() AS basic_info
2 FROM Publications p;
```

TITLE	BASIC_INFO
The Great Gatsby	Title: The Great Gatsby, Publication Date: 10-MAY-2023, Type: Book
National Geographic	Title: National Geographic, Publication Date: 15-APR-2023, Type: Journal
Harry Potter and the Philosopher's Stone	Title: Harry Potter and the Philosopher's Stone, Publication Date: 20-JUN-2023, Type: Book

Figure 10Retriving Data from Publications Table Execution

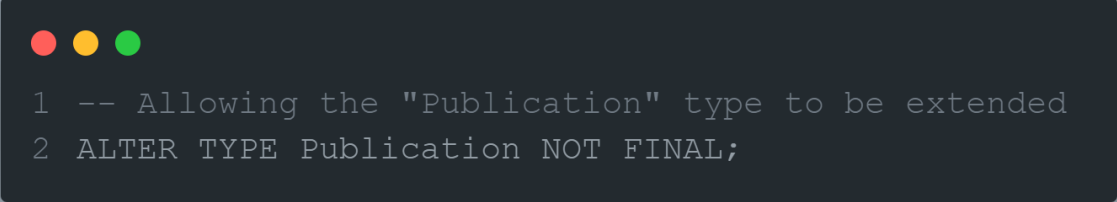
This query selects data from the Publications table, and from each row, it retrieves its title column and uses the other attributes to create an instance of the `Publication` type and then calls the `displayBasicInfo` method on the `Publication` instance and the returned information is aliased as `basic_info`, finally the title alongside the string from the `displayBasicInfo()`.

```
1 -- Query to retrieve basic information for each publication using the displayBasicInfo method
2 SELECT p.title, Publication(p.title, p.publication_date, p.publication_type).displayBasicInfo() AS basic_info
3 FROM Publications p;
```

Figure 11Retriving Data from Publications Table SQL

3. Book

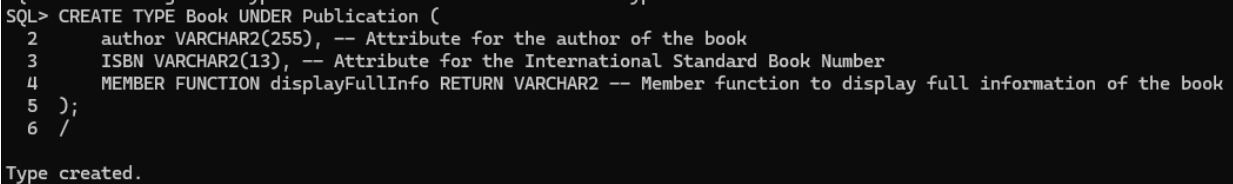
Before Executing any of the following SQL scripts the following SQL code must be executed to ensure that any errors won't be raised.



```
1 -- Allowing the "Publication" type to be extended
2 ALTER TYPE Publication NOT FINAL;
```

Figure 12 Alternating the Type Publications to be extended.

3.1 Type Creation

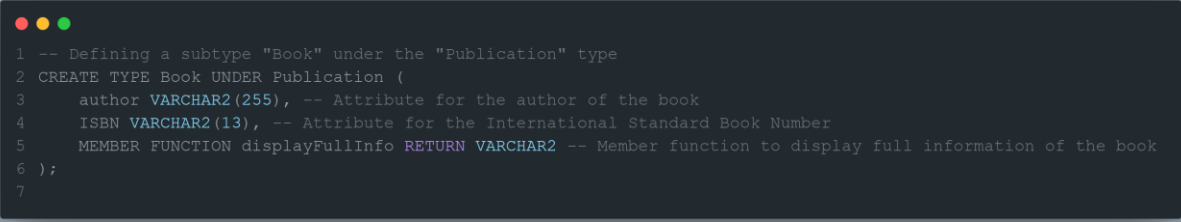


```
SQL> CREATE TYPE Book UNDER Publication (
2   author VARCHAR2(255), -- Attribute for the author of the book
3   ISBN VARCHAR2(13), -- Attribute for the International Standard Book Number
4   MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 -- Member function to display full information of the book
5 );
6 /

Type created.
```

Figure 13 Creating Type Book Execution

This code segment creates a subtype `Book` on top of the `Publication` supertype, it introduces more attributes such as `author` with VARCHAR2(255) datatype, `ISBN` with VARCHAR2(13) datatype; and the subtype introduces a new member function called `displayFullInfo()` which returns the data type `VARCHAR2`.



```
1 -- Defining a subtype "Book" under the "Publication" type
2 CREATE TYPE Book UNDER Publication (
3   author VARCHAR2(255), -- Attribute for the author of the book
4   ISBN VARCHAR2(13), -- Attribute for the International Standard Book Number
5   MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 -- Member function to display full information of the book
6 );
7
```

Figure 14 Creating Type Book SQL

3.2 Body Creation

```
SQL> CREATE OR REPLACE TYPE BODY Book AS
2     MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS -- Implementation of the member function
3     BEGIN
4         RETURN ', Author: ' || author || ', ISBN: ' || ISBN; -- Returning the full information of the book
5     END displayFullInfo; -- End of the member function
6 END; -- End of the type body
7
8 /

Type body created.
```

Figure 15 Creating / Replacing Body Type Execution

This code implements the `displayFullInfo` member function belonging to the `Book` subtype. The function is implemented so that a concatenated string with the details of all the attributes of the Book instance is returned.

```
1 -- Defining the body of the member function "displayFullInfo" for the "Book" subtype
2 CREATE OR REPLACE TYPE BODY Book AS
3     MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS -- Implementation of the member function
4     BEGIN
5         RETURN ', Author: ' || author || ', ISBN: ' || ISBN; -- Returning the full information of the book
6     END displayFullInfo; -- End of the member function
7 END; -- End of the type body
8
```

Figure 16 Creating / Replacing Body Type SQL

3.3 Table Creation

```
SQL> CREATE TABLE Books (  
2     ISBN VARCHAR2(13) PRIMARY KEY,  
3     publication_id NUMBER UNIQUE,  
4     author VARCHAR2(255),  
5     CONSTRAINT fk_books_publications FOREIGN KEY (publication_id) REFERENCES Publications(publication_id) ON DELETE CASCADE  
6 );  
  
Table created.
```

Figure 17 Create Table Books Execution

This SQL code creates the table `Books`, which is used to store information regarding books. The table has 3 attributes and 1 constraint, The attributes are: ISBN with VARCHAR2(13) datatype, publication_id with NUMBER datatype and author with VARCHAR2(255) datatype. The constraint in the Books table is that the `publication_id` but be referenced in Publications(publication_id) in turn creating a foreign key constraint.

```
1 CREATE TABLE Books (  
2     ISBN VARCHAR2(13) PRIMARY KEY,  
3     publication_id NUMBER UNIQUE,  
4     author VARCHAR2(255),  
5     CONSTRAINT fk_books_publications FOREIGN KEY (publication_id) REFERENCES Publications(publication_id) ON DELETE CASCADE  
6 );  
7
```

Figure 18 Create Table Books SQL

3.4 Data Insertion

```
SQL> -- Add test data to the Books table
SQL> INSERT INTO Books (ISBN, publication_id, author)
  2 VALUES ('1234567890123', 1, 'J.K. Rowling');

1 row created.

SQL>
SQL> INSERT INTO Books (ISBN, publication_id, author)
  2 VALUES ('9876543210987', 2, 'George R. R. Martin');

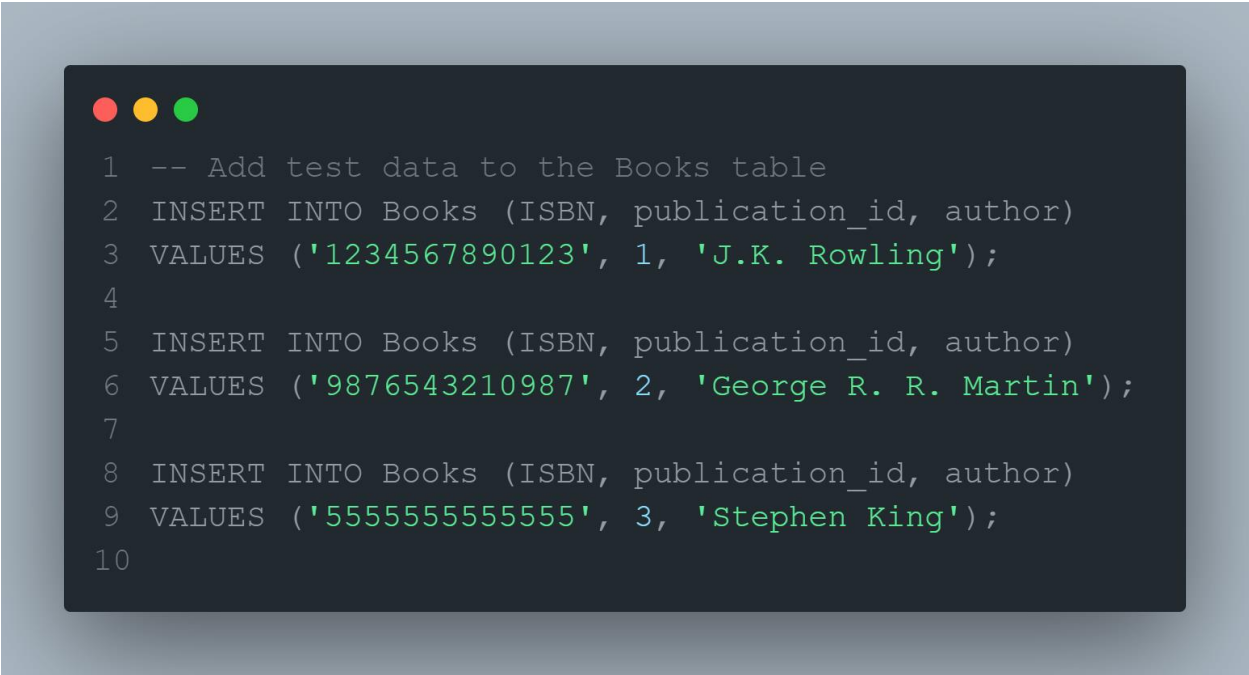
1 row created.

SQL>
SQL> INSERT INTO Books (ISBN, publication_id, author)
  2 VALUES ('5555555555555', 3, 'Stephen King');

1 row created.
```

Figure 19 Inserting into the Books Table Execution

These 3 statements enter 3 records into the Books Table, where each INSERT INTO inserts one row of data. Each value corresponds to a column in the table, and there are 3 columns, so each insertion has 3 values corresponding to each column. Due to the constraint, all the 'publication_id's are already in the 'Publications' table.

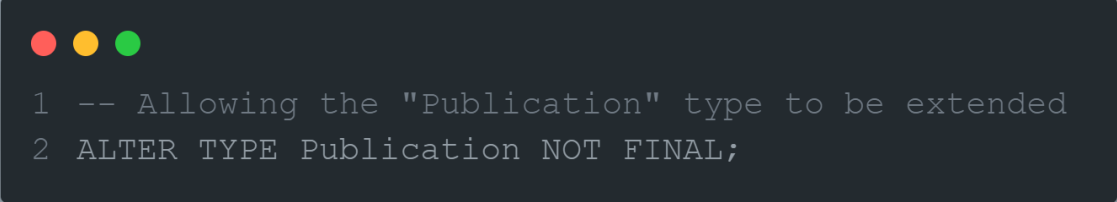


```
1 -- Add test data to the Books table
2 INSERT INTO Books (ISBN, publication_id, author)
3 VALUES ('1234567890123', 1, 'J.K. Rowling');
4
5 INSERT INTO Books (ISBN, publication_id, author)
6 VALUES ('9876543210987', 2, 'George R. R. Martin');
7
8 INSERT INTO Books (ISBN, publication_id, author)
9 VALUES ('5555555555555', 3, 'Stephen King');
10
```

Figure 20 Inserting into the Books Table SQL

4. Journal

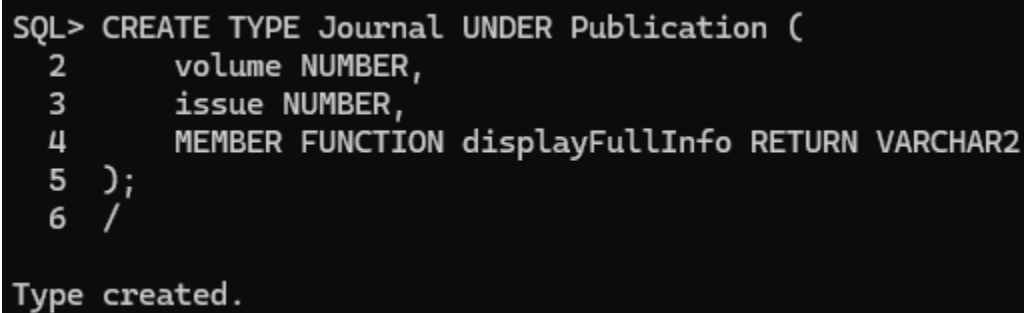
Before Executing any of the following SQL scripts the following SQL code must be executed to ensure that any errors won't be raised.



```
1 -- Allowing the "Publication" type to be extended
2 ALTER TYPE Publication NOT FINAL;
```

Figure 21 Alternating the Type Publications to be extended.

4.1 Type Creation

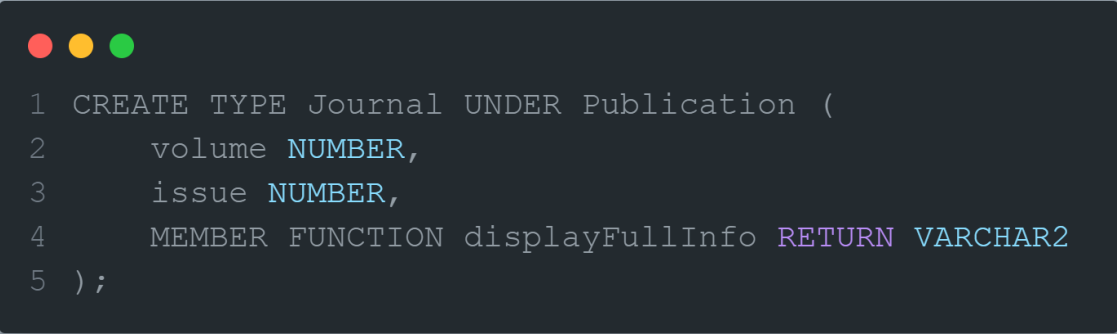


```
SQL> CREATE TYPE Journal UNDER Publication (
2     volume NUMBER,
3     issue NUMBER,
4     MEMBER FUNCTION displayFullInfo RETURN VARCHAR2
5 );
6 /

Type created.
```

Figure 22 Creating Type Journal Execution

This SQL statement creates a new object type, which is a subtype of the Publication supertype. It has 2 attributes and a member function called `displayFullInfo()` which returns the type of VARCHAR2.



```
1 CREATE TYPE Journal UNDER Publication (
2     volume NUMBER,
3     issue NUMBER,
4     MEMBER FUNCTION displayFullInfo RETURN VARCHAR2
5 );
```

Figure 23 Creating Type Journal SQL

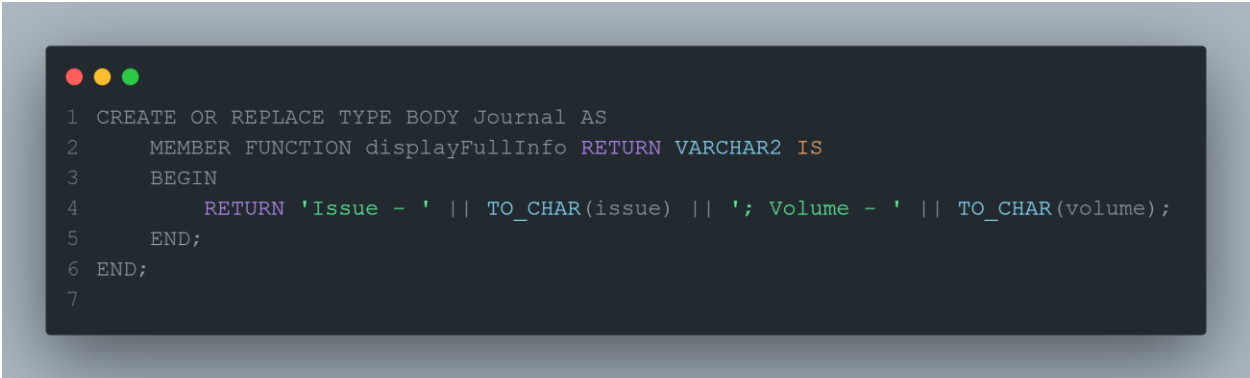
4.2 Body Creation

```
SQL> CREATE OR REPLACE TYPE BODY Journal AS
2     MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS
3     BEGIN
4         RETURN 'Issue - ' || TO_CHAR(issue) || '; Volume - ' || TO_CHAR(volume);
5     END;
6 END;
7 /

Type body created.
```

Figure 24 Creating / Replacing Journal Type Body Execution

This code creates or replaces the body of the type `Journal` by implementing the function `displayFullInfo()` and it returns a string that contains all the attributes.



```
1 CREATE OR REPLACE TYPE BODY Journal AS
2     MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS
3     BEGIN
4         RETURN 'Issue - ' || TO_CHAR(issue) || '; Volume - ' || TO_CHAR(volume);
5     END;
6 END;
7
```

Figure 25 Creating / Replacing Journal Type Bodyy SQL

4.3 Table Creation

```
SQL> CREATE TABLE Journals (
2     publication_id NUMBER PRIMARY KEY,
3     volume NUMBER,
4     issue NUMBER,
5     CONSTRAINT fk_journals_publications FOREIGN KEY (publication_id) REFERENCES Publications(publication_id) ON DEL
ETE CASCADE
6 );

Table created.
```

Figure 26 Creating Journals Table Execution

This SQL Code creates the table `Journals`, which is used to store information regarding Journals. The table has 3 attributes and 1 constraint, the attributes are volume with NUMBER datatype, publication_id with NUMBER datatype (which is the primary key) and issue with NUMBER datatype. The constraint in the Journals table is that the `publication_id` references in

Publications(publication_id) in turn create a foreign key constraint.

```
1 CREATE TABLE Journals (  
2     publication_id NUMBER PRIMARY KEY,  
3     volume NUMBER,  
4     issue NUMBER,  
5     CONSTRAINT fk_journals_publications FOREIGN KEY (publication_id) REFERENCES Publications(publication_id) ON DELETE CASCADE  
6 );  
7
```

Figure 27 Creating / Replacing Journal Table SQL

4.4 Data Insertion

```
SQL> -- Insert test data into the Journals table  
SQL> INSERT INTO Journals (publication_id, volume, issue)  
2 VALUES (1, 10, 1);  
  
1 row created.  
  
SQL>  
SQL> INSERT INTO Journals (publication_id, volume, issue)  
2 VALUES (2, 11, 2);  
  
1 row created.  
  
SQL>  
SQL> INSERT INTO Journals (publication_id, volume, issue)  
2 VALUES (3, 12, 3);  
  
1 row created.
```

Figure 28 Inserting Data into the Journals Table Execution

These 3 statements enter 3 records into the Journals Table, where each INSERT INTO inserts one row of data. Each value corresponds to a column in the table, and there are 3 columns, so each insertion has 3 values corresponding to each column. Due to the constraint, all the `publication_id`'s are already in the `Publications` table.

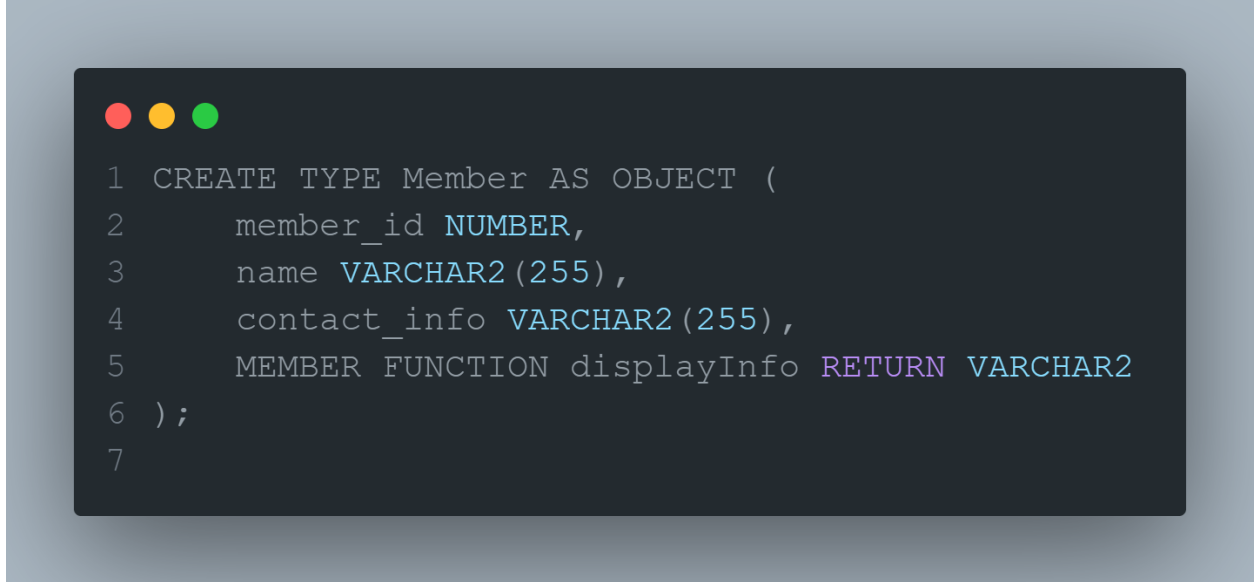
5. Member

5.1 Type Creation

```
SQL> CREATE TYPE Member AS OBJECT (  
  2     member_id NUMBER,  
  3     name VARCHAR2(255),  
  4     contact_info VARCHAR2(255),  
  5     MEMBER FUNCTION displayInfo RETURN VARCHAR2  
  6 );  
  7 /  
  
Type created.
```

Figure 29 Creating Type Member Execution

This SQL statement creates a new object type Member, it has 3 attributes (member_id, name, contact_info) and a member function called `displayFullInfo()` which returns the type of VARCHAR2.



```
1 CREATE TYPE Member AS OBJECT (  
2     member_id NUMBER,  
3     name VARCHAR2(255),  
4     contact_info VARCHAR2(255),  
5     MEMBER FUNCTION displayInfo RETURN VARCHAR2  
6 );  
7
```

Figure 30 Creating Type Member SQL

5.2 Body Creation

```
SQL> CREATE OR REPLACE TYPE BODY Member AS
2     MEMBER FUNCTION displayInfo RETURN VARCHAR2 IS
3     BEGIN
4         RETURN 'Member ID: ' || TO_CHAR(member_id) || ', Name: ' || name || ', Contact Info: ' || contact_info;
5     END;
6 END;
7 /

Type body created.
```

Figure 31 Creating / Replacing Member Type Body Execution

This code creates or replaces the body of the type `Member` by implementing the function `displayFullInfo()` and it returns a string that contains all the attributes.

```
1 CREATE OR REPLACE TYPE BODY Member AS
2     MEMBER FUNCTION displayInfo RETURN VARCHAR2 IS
3     BEGIN
4         RETURN 'Member ID: ' || TO_CHAR(member_id) || ', Name: ' || name || ', Contact Info: ' || contact_info;
5     END;
6 END;
7
```

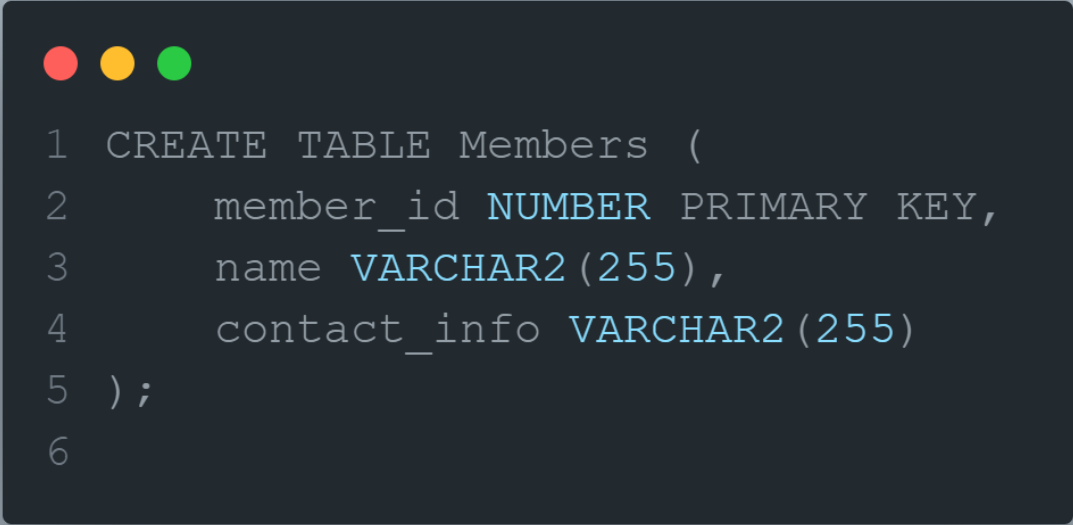
Figure 32 Creating / Replacing Member Type Body SQL

5.3 Table Creation

```
SQL> CREATE TABLE Members (  
2     member_id NUMBER PRIMARY KEY,  
3     name VARCHAR2(255),  
4     contact_info VARCHAR2(255)  
5 );  
  
Table created.
```

Figure 33 Creating Members Table Execution

This SQL Code creates the table `Members`, which is used to store information regarding Members. The table has 3 attributes, the attributes are: member_id with NUMBER datatype (which is the primary key), name with VARCHAR2(255) datatype and contact_info with VARCHAR(255) datatype.



```
1 CREATE TABLE Members (  
2     member_id NUMBER PRIMARY KEY,  
3     name VARCHAR2(255),  
4     contact_info VARCHAR2(255)  
5 );  
6
```

Figure 34 Creating Memembers Table SQL

5.4 Data Insertion

```
SQL> INSERT INTO Members (member_id, name, contact_info)
  2  VALUES (1, 'John Doe', 'john.doe@example.com');

1 row created.

SQL>
SQL> INSERT INTO Members (member_id, name, contact_info)
  2  VALUES (2, 'Jane Smith', 'jane.smith@example.com');


1 row created.

SQL>
SQL> INSERT INTO Members (member_id, name, contact_info)
  2  VALUES (3, 'Michael Johnson', 'michael.johnson@example.com');

1 row created.
```

Figure 35 Inserting Data into the Members Table Execution

These 3 statements enter 3 records into the Members Table, where each INSERT INTO inserts one row of data. Each value corresponds to a column in the Table, and there are 3 columns, so each insertion has 3 values corresponding to each column.



```
1 INSERT INTO Members (member_id, name, contact_info)
2 VALUES (1, 'John Doe', 'john.doe@example.com');
3
4 INSERT INTO Members (member_id, name, contact_info)
5 VALUES (2, 'Jane Smith', 'jane.smith@example.com');
6
7 INSERT INTO Members (member_id, name, contact_info)
8 VALUES (3, 'Michael Johnson', 'michael.johnson@example.com');
9
10
```

Figure 36 Inserting Data into the Members Table SQL

6. Loans

To execute the following SQL code it is required that the above SQL code has already been executed.

6.1 Table Creation

```
SQL> CREATE TABLE Loans (  
2     member_id NUMBER,  
3     publication_id NUMBER,  
4     loan_date DATE,  
5     return_date DATE,  
6     PRIMARY KEY (member_id, publication_id),  
7     CONSTRAINT fk_loans_members FOREIGN KEY (member_id) REFERENCES Members(member_id),  
8     CONSTRAINT fk_loans_publications FOREIGN KEY (publication_id) REFERENCES Publications(publication_id)  
9 );  
  
Table created.
```

Figure 37 Creating Table Loans Execution

This SQL statement creates a new object type Loans, it has 4 attributes: member_id, publication_id (which both are primary keys), loan_date, and return_date. There are 2 constraints in the table, 1 which make sure that the member_id exists in the Members (member_id) attribute, and another to make sure that the `publication_id` exists in the Publications(publication_id).

```
1 CREATE TABLE Loans (  
2     member_id NUMBER,  
3     publication_id NUMBER,  
4     loan_date DATE,  
5     return_date DATE,  
6     PRIMARY KEY (member_id, publication_id),  
7     CONSTRAINT fk_loans_members FOREIGN KEY (member_id) REFERENCES Members(member_id),  
8     CONSTRAINT fk_loans_publications FOREIGN KEY (publication_id) REFERENCES Publications(publication_id)  
9 );  
10
```

Figure 38 Creating Table Loans SQL

6.2 Data Insertion

```
SQL> -- Inserting data for loan of 'The Great Gatsby' to John Doe
SQL> INSERT INTO Loans (member_id, publication_id, loan_date, return_date)
  2 VALUES (1, 1, TO_DATE('2023-07-01', 'YYYY-MM-DD'), TO_DATE('2023-07-08', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> -- Inserting data for loan of 'National Geographic' to Jane Smith
SQL> INSERT INTO Loans (member_id, publication_id, loan_date, return_date)
  2 VALUES (2, 2, TO_DATE('2023-07-03', 'YYYY-MM-DD'), TO_DATE('2023-07-10', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> -- Inserting data for loan of 'Harry Potter and the Philosopher's Stone' to Michael Johnson
SQL> INSERT INTO Loans (member_id, publication_id, loan_date, return_date)
  2 VALUES (3, 3, TO_DATE('2023-07-05', 'YYYY-MM-DD'), TO_DATE('2023-07-12', 'YYYY-MM-DD'));

1 row created.
```

Figure 39 Inserting Data into the Loans Table Execution

These 3 statements enter 3 records into the Loans Table, where each INSERT INTO inserts one row of data. Each value corresponds to a column in the Table, and there are 4 columns, so each insertion has 4 values corresponding to each column.

```
1 -- Inserting data for loan of 'The Great Gatsby' to John Doe
2 INSERT INTO Loans (member_id, publication_id, loan_date, return_date)
3 VALUES (1, 1, TO_DATE('2023-07-01', 'YYYY-MM-DD'), TO_DATE('2023-07-08', 'YYYY-MM-DD'));
4
5 -- Inserting data for loan of 'National Geographic' to Jane Smith
6 INSERT INTO Loans (member_id, publication_id, loan_date, return_date)
7 VALUES (2, 2, TO_DATE('2023-07-03', 'YYYY-MM-DD'), TO_DATE('2023-07-10', 'YYYY-MM-DD'));
8
9 -- Inserting data for loan of 'Harry Potter and the Philosopher's Stone' to Michael Johnson
10 INSERT INTO Loans (member_id, publication_id, loan_date, return_date)
11 VALUES (3, 3, TO_DATE('2023-07-05', 'YYYY-MM-DD'), TO_DATE('2023-07-12', 'YYYY-MM-DD'));
12
```

Figure 40 Inserting Data into the Loans Table SQL

7. Additional Features

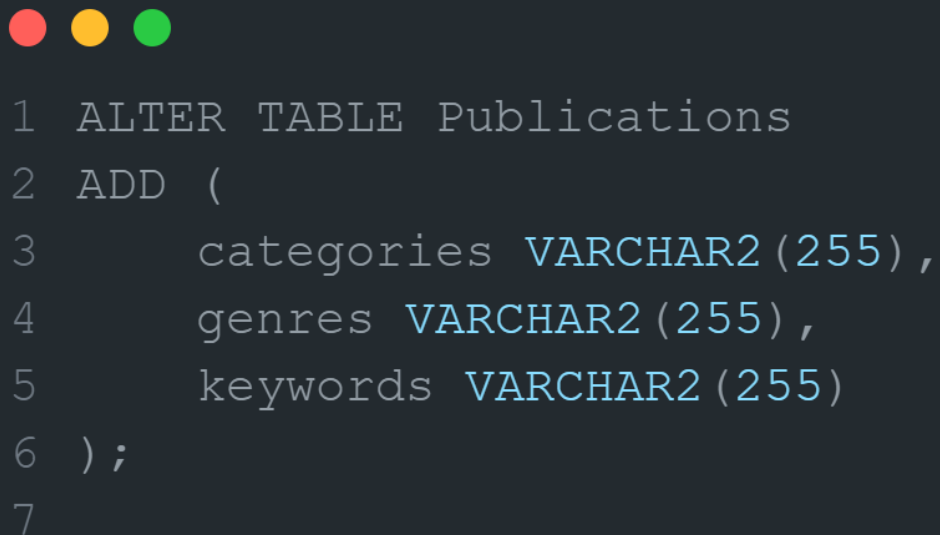
7.1 Additional attributes for Publications

```
SQL> ALTER TABLE Publications
2  ADD (
3      categories VARCHAR2(255),
4      genres VARCHAR2(255),
5      keywords VARCHAR2(255)
6  );

Table altered.
```

Figure 41 Altering Publications Table Execution

The given SQL segment alters the table Publications to include 3 new columns / attributes: categories, genres, and keywords and all of them are of type VARCHAR2 with maximum characters of 255.



```
1 ALTER TABLE Publications
2 ADD (
3     categories VARCHAR2(255),
4     genres VARCHAR2(255),
5     keywords VARCHAR2(255)
6 );
7
```

Figure 42 Altering Publications Table SQL

7.2 Loan Details Viewer

```
SQL> CREATE OR REPLACE VIEW LoanDetails AS
 2  SELECT l.member_id,
 3         m.name AS member_name,
 4         m.contact_info AS member_contact,
 5         l.publication_id,
 6         p.title AS publication_title,
 7         p.publication_date AS publication_date,
 8         p.publication_type AS publication_type,
 9         l.loan_date,
10        l.return_date
11  FROM Loans l
12 JOIN Members m ON l.member_id = m.member_id
13 JOIN Publications p ON l.publication_id = p.publication_id;
```

View created.

```
SQL> SELECT * FROM LoanDetails;
```

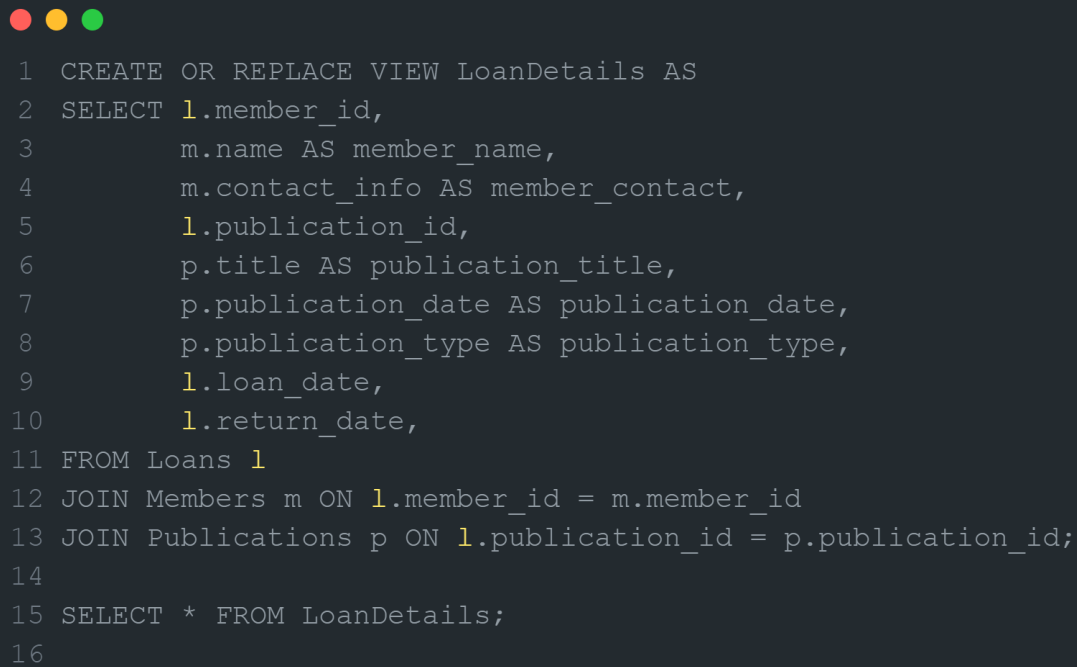
MEMBER_ID	MEMBER_NAME	MEMBER_CONTACT	PUBLICATION_ID	PUBLICATION_TITLE	PUBLICATION_TYPE	LOAN_DATE	RETURN_DA
1							

MEMBER_ID	MEMBER_NAME	MEMBER_CONTACT	PUBLICATION_ID	PUBLICATION_TITLE	PUBLICATION_TYPE	LOAN_DATE	RETURN_DA
John Doe							

MEMBER_ID	MEMBER_NAME
-----------	-------------

Figure 43 Loan Details Viewer Execution

This code creates a view named LoanDetails which collects data from Tables: Loans, Members and Publications and displays details such as member and publication details, loan dates in turn making it easier to analyze loan related information.



```
1 CREATE OR REPLACE VIEW LoanDetails AS
2 SELECT l.member_id,
3        m.name AS member_name,
4        m.contact_info AS member_contact,
5        l.publication_id,
6        p.title AS publication_title,
7        p.publication_date AS publication_date,
8        p.publication_type AS publication_type,
9        l.loan_date,
10       l.return_date,
11 FROM Loans l
12 JOIN Members m ON l.member_id = m.member_id
13 JOIN Publications p ON l.publication_id = p.publication_id;
14
15 SELECT * FROM LoanDetails;
16
```

Figure 44 Loan Details Viewer SQL

References

Ameya (2024) 'What is relational Database with Real-Life examples,' RedSwitches, 31 January.

<https://www.redswitches.com/blog/what-is-relational-database/>.

GfG (2021) Basic object oriented data model. <https://www.geeksforgeeks.org/basic-object-oriented-data-model/>.

Object-Oriented Data Model and its Application / Free essay example (2022).

<https://studycorgi.com/object-oriented-data-model-and-its-application/>.

Bhagwat, S. (2022) *What is Object Oriented Model in DBMS? - Scaler Topics*.

<https://www.scaler.com/topics/object-oriented-model-in-dbms/>.

Dancuk, M. (2023) What is an Object-Oriented Database. <https://phoenixnap.com/kb/object-oriented-database>.

Object-relational data model (no date). <https://www.tutorialspoint.com/Object-relational-Data-Model>.

Auziņš, A. *et al.* (2018) 'Object-Relational database structure model and structure optimisation,' *Applied Computer Systems (Online)*, 23(1), pp. 28–36. <https://doi.org/10.2478/acss-2018-0004>.

Jordan, M. (2023) *What is Database Scalability? Definition & FAQs / ScyllaDB*.

<https://www.scylladb.com/glossary/database-scalability/#:~:text=Are%20relational%20databases%20scalable%3F,a%20vertical%20approach%20to%20scaling>.

Murphy, E. (2022) 'Flexible Relational Data,' *induro*, 13 May.

https://induro.io/blog/2022/05/flexible_relational_data/#:~:text=When%20relationships%20are%20predictable%20and,flexible%20to%20accurately%20represent%20it.

Ot, A. (2023) *Relational Data Model 101: Key components & benefits*.

<https://www.datamation.com/big-data/relational-data-model/#:~:text=Relational%20data%20models%20also%20aid,data%20sets%20in%20various%20applications>.

Pahi, N. (2013) *The relational model and normalization*. <https://blog.oureducation.in/relational-model-normalization/>.

What is data modeling? | IBM (no date b). <https://www.ibm.com/topics/data-modeling>.

What is a Data Model? (no date b). <https://cedar.princeton.edu/understanding-data/what-data-model>.

Staff, C. (2023b) *What is a data model?* <https://www.coursera.org/articles/data-model>.

G, M. (2018b) *Why Data Modelling is Important*. <https://www.linkedin.com/pulse/why-data-modelling-important-munish-goswami/>.

What is Data Modeling and Why Do You Need It? (no date b). <https://www.dasca.org/world-of-big-data/article/what-is-data-modeling-and-why-do-you-need-it>.

Vivek, J. (2024b) *Data modeling: benefits, types, importance & steps involved*.

<https://www.zucisystems.com/blog/what-is-data-modeling-and-why-is-it-important/>.

Team, P.B. (no date b) *What is data modelling?* <https://powerbi.microsoft.com/en-my/what-is-data-modeling/>.

Watt, A. (2014b) *Chapter 7 The Relational Data model*.

<https://opentextbc.ca/dbdesign01/chapter/chapter-7-the-relational-data-model/#:~:text=The%20relational%20data%20model%20was%20modern%20commercial%20database%20management%20systems>.

- Newcomb, P.H. and Couch, R. (2010b) 'Veterans Health Administration's VISTA MUMPS modernization pilot,' in *Elsevier eBooks*, pp. 301–345. <https://doi.org/10.1016/b978-0-12-374913-0.00012-3>.
- Khan, S. (no date b) *Brief history of the relational model*.
<https://www.scribd.com/document/94827565/Brief-History-of-the-Relational-Model>.
- Peterson, R. (2024b) *Relational Data model in DBMS / Database Concepts & example*.
<https://www.guru99.com/relational-data-model-dbms.html>.
- GfG (2024b) *Relational model in DBMS*. <https://www.geeksforgeeks.org/relational-model-in-dbms/>.
- T, N. (2021b) *Relational Data model*. <https://binaryterms.com/relational-data-model.html>.