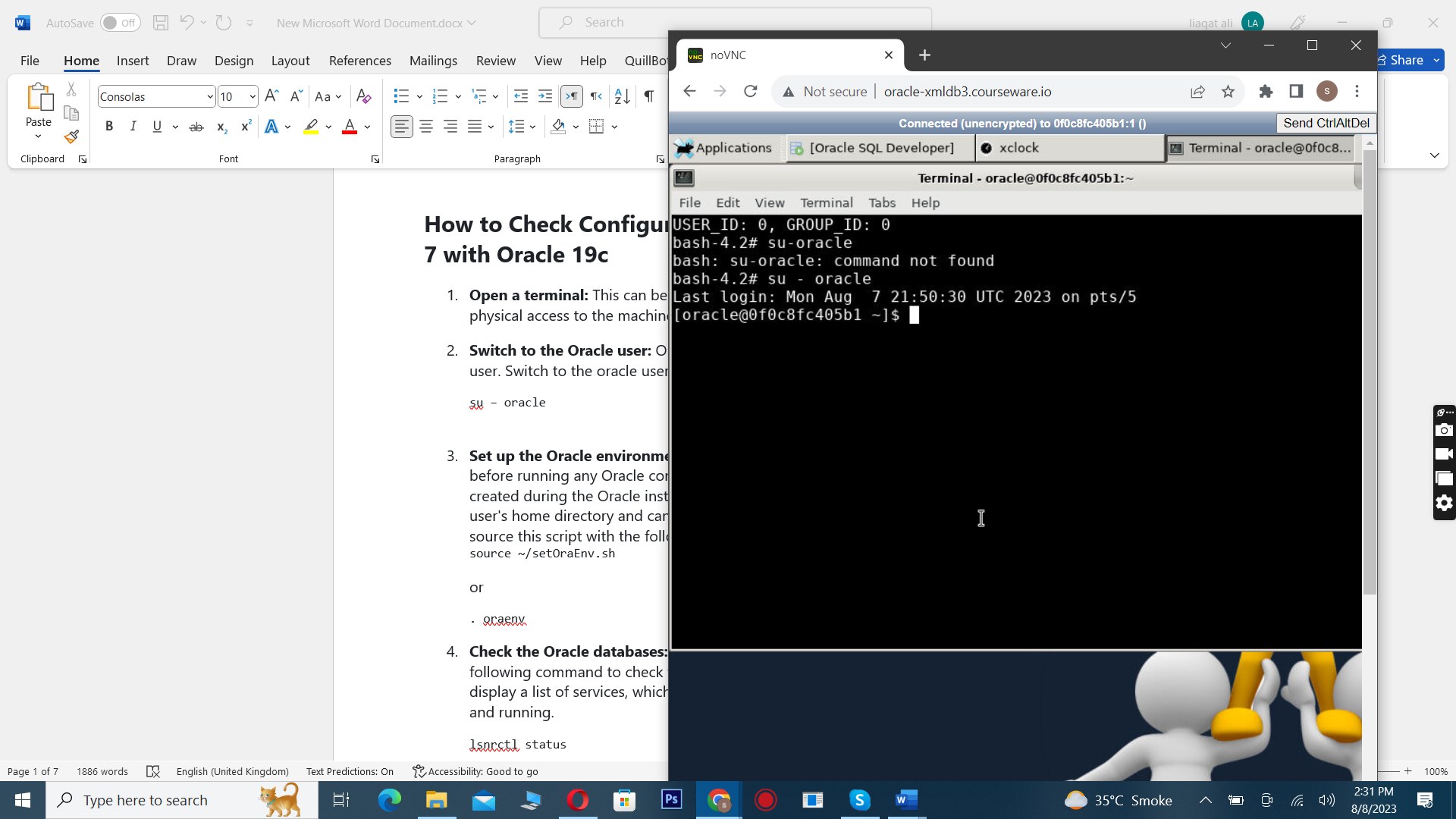
**How to Check Configured Oracle Databases on CentOS 7 with Oracle 19c**

1. **Open a terminal:** This can be done via SSH into the server or directly if you have physical access to the machine.
2. **Switch to the Oracle user:** Oracle installations are usually owned by the "oracle" user. Switch to the oracle user with the following command:

su – oracle



1. **Set up the Oracle environment:** You need to set up the Oracle environment before running any Oracle commands. This is usually done by running a script created during the Oracle installation. This script is usually located in the oracle user's home directory and can have names like setOraEnv.sh or oraenv. You can source this script with the following command:

source ~/setOraEnv.sh

or

. oraenv (the dot in front is important!)

A screenshot of a computer

Description automatically generated

1. **Check the Oracle databases:** After the environment setup, you can run the following command to check the status of the Oracle listener. This command will display a list of services, which includes the Oracle databases currently configured and running.

lsnrctl status

A computer screen shot of a computer screen

Description automatically generated

If you're looking for specific databases, you can also query the v$database view in SQL\*Plus to get information about the database currently connected to. This can be done with the following command:

sqlplus / as sysdba

Once you're in the SQL\*Plus prompt, run the following command:

SELECT name FROM v$database;

This will give you the name of the database that SQL\*Plus is currently connected to.

**Note:** You need to have the necessary permissions to perform these actions. If you're not the Oracle system administrator, you may need to ask them for help.

# Lab: Starting Oracle EM Express and Logging In on CentOS 7 with Oracle 19c

## Objective:

In this lab, we will start Oracle Enterprise Manager (EM) Express on a CentOS 7 server with Oracle 19c, and then log in to EM Express using specific credentials.

## Steps:

1. **Open a terminal:** You can SSH into the server or open a terminal if you have physical access to the machine.
2. **Switch to the Oracle user:** Oracle installations are usually owned by the "oracle" user. Switch to this user with the following command:

su - oracle

1. **Set up the Oracle environment:** Before you can run any Oracle commands, you need to set up the Oracle environment. This is usually done by running a script that was created during the Oracle installation. This script is usually located in the Oracle user's home directory and is named something like setOraEnv.sh or oraenv. You can source this script with the following command:

source ~/setOraEnv.sh

or

. oraenv

1. **Set the EM Express HTTPS port:** Connect to your database as SYSDBA and set the HTTPS port for EM Express with the following commands:

sqlplus / as sysdba

Once you're in the SQL\*Plus prompt, run the following command:

EXEC DBMS\_XDB\_CONFIG.SETHTTPSPORT(5500);

This command sets the HTTPS port for EM Express to 5500. You can choose a different port if you like, but remember that port numbers below 1024 are considered "privileged" on Unix-like systems, and only root can bind to them.

1. **Verify the configuration:** You can verify that EM Express is now configured to listen on the port you set by running the following command in SQL\*Plus:

SELECT DBMS\_XDB\_CONFIG.GETHTTPSPORT() FROM dual;

The output of this command should now be 5500, the port number you set in the previous step.

1. **Exit SQL\*Plus:** Once you've finished with SQL\*Plus, you can exit with the following command:

exit

1. **Access EM Express:** Once EM Express is configured, you can access it by opening a web browser and navigating to https://<hostname>:5500/em, replacing <hostname> with the hostname or IP address of your Oracle server.
2. **Log in to EM Express:** You can log in to EM Express using the following credentials:
   * **Username:** sys
   * **Password:** fenago
   * **Container:** LEAVE\_BLANK or use 'CDB$ROOT' as you are connecting to the root container of a CDB.

# Lab: Getting Started with Oracle XML DB

## Objective:

In this lab, we will check if Oracle XML DB is installed on our system. Oracle XML DB is a high-performance, native XML storage and retrieval technology that is delivered as a part of all versions of Oracle Database. If it's installed, we will have access to the database schema (user account) XDB and the view RESOURCE\_VIEW.

## Steps:

1. **Open a terminal:** You can SSH into the server or open a terminal if you have physical access to the machine.
2. **Switch to the Oracle user:** Oracle installations are usually owned by the "oracle" user. Switch to this user with the following command:

su - oracle

1. **Set up the Oracle environment:** Before you can run any Oracle commands, you need to set up the Oracle environment. This is usually done by running a script that was created during the Oracle installation. This script is usually located in the Oracle user's home directory and is named something like setOraEnv.sh or oraenv. You can source this script with the following command:

source ~/setOraEnv.sh

or

. oraenv

1. **Check if the XDB user exists:** Connect to your database as SYSDBA and check the existence of the XDB user with the following commands:

sqlplus / as sysdba

Once you're in the SQL\*Plus prompt, run the following command:

SELECT username FROM all\_users WHERE username = 'XDB';

If Oracle XML DB is installed, this will return 'XDB'. If the query doesn't return anything, it means that Oracle XML DB is not installed.

1. **Check if the RESOURCE\_VIEW view exists:** Still in the SQL\*Plus prompt, check the existence of the RESOURCE\_VIEW view with the following command:

SELECT \* FROM all\_objects WHERE object\_type = 'VIEW' AND object\_name = 'RESOURCE\_VIEW';

If Oracle XML DB is installed, this will return information about the RESOURCE\_VIEW view. If the query doesn't return anything, it means that the RESOURCE\_VIEW view does not exist, indicating that Oracle XML DB is not installed.

1. **Exit SQL\*Plus:** Once you've finished with SQL\*Plus, you can exit with the following command:

# Lab: Creating XMLType Tables and Columns in Oracle Database

## Objective:

In this lab, we will create tables with XMLType columns in an Oracle Database. XMLType is a system-defined opaque type for handling XML data. It has predefined member functions to extract XML nodes and fragments.

## Steps:

1. **Open a terminal:** You can SSH into the server or open a terminal if you have physical access to the machine.
2. **Switch to the Oracle user:** Oracle installations are usually owned by the "oracle" user. Switch to this user with the following command:

su - oracle

1. **Set up the Oracle environment:** Before you can run any Oracle commands, you need to set up the Oracle environment. This is usually done by running a script that was created during the Oracle installation. This script is usually located in the Oracle user's home directory and is named something like setOraEnv.sh or oraenv. You can source this script with the following command:

source ~/setOraEnv.sh

or

. oraenv

1. **Create a table with an XMLType column:** Connect to your database as SYSDBA and create a table named mytable1 with an XMLType column named xml\_column and a primary key named key\_column with the following commands:

sqlplus / as sysdba

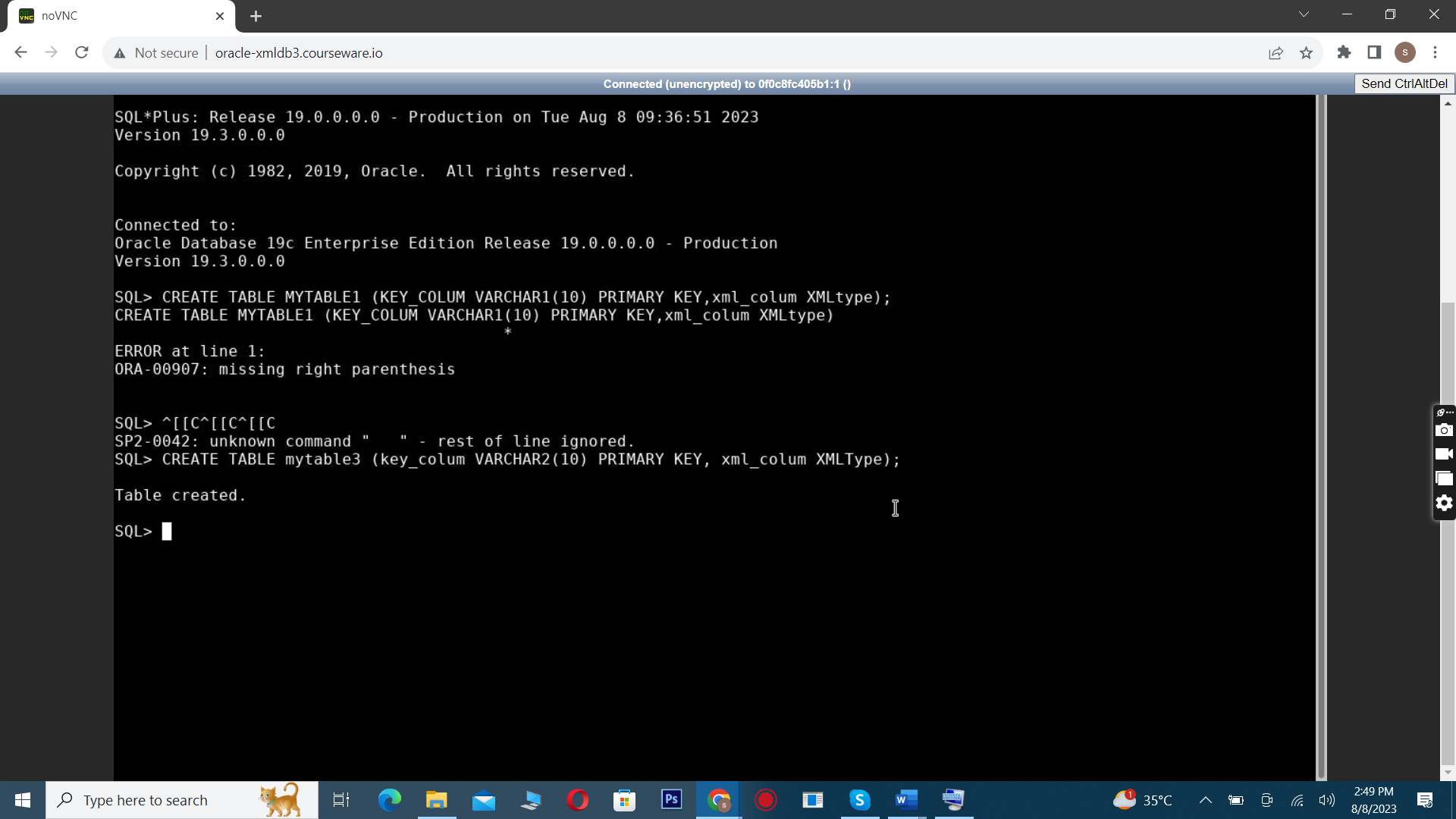
A screenshot of a computer

Description automatically generated

C

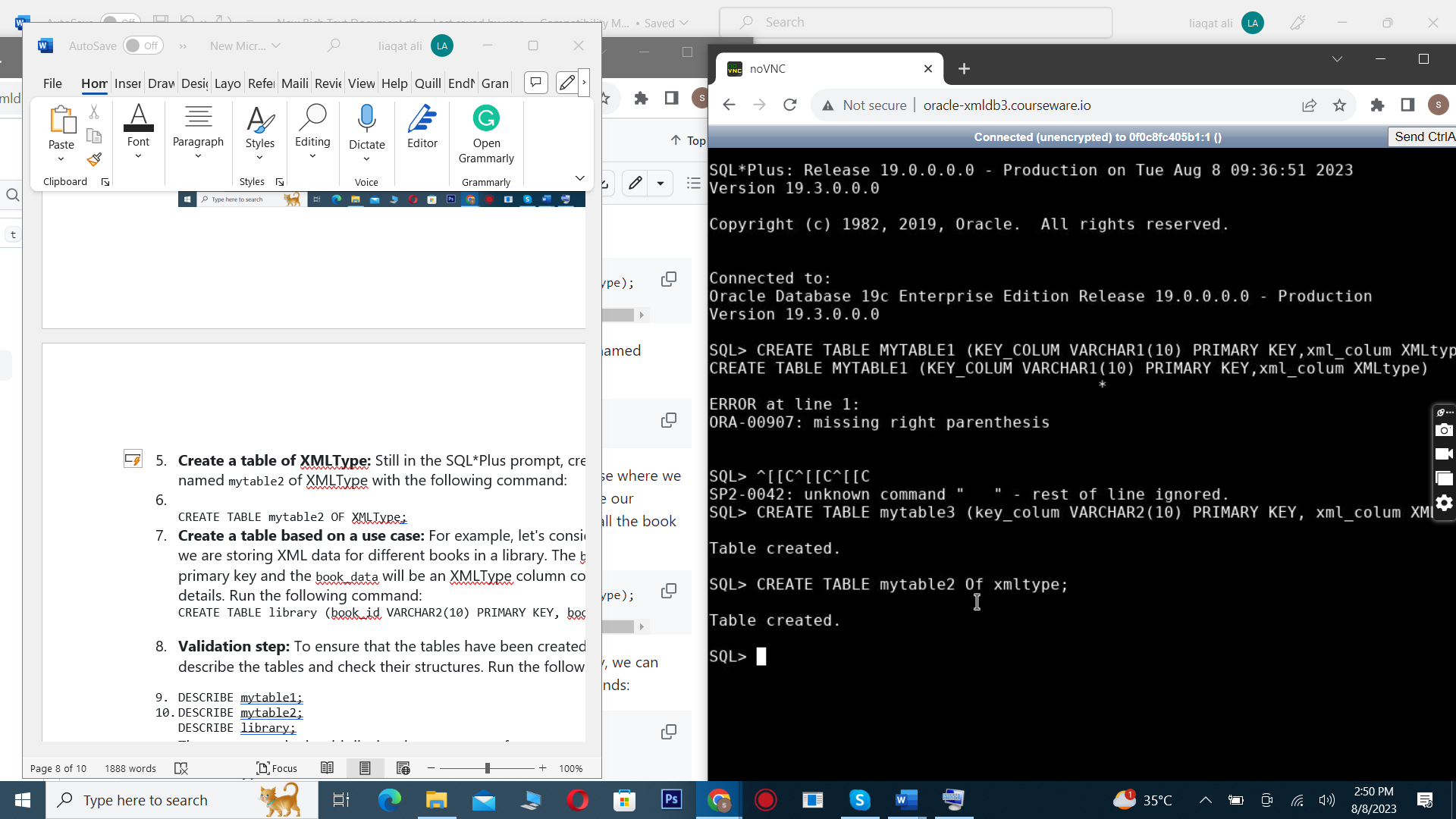
Once you're in the SQL\*Plus prompt, run the following command:

CREATE TABLE mytable1 (key\_column VARCHAR2(10) PRXMLIMARY KEY, xml\_column XMLType);



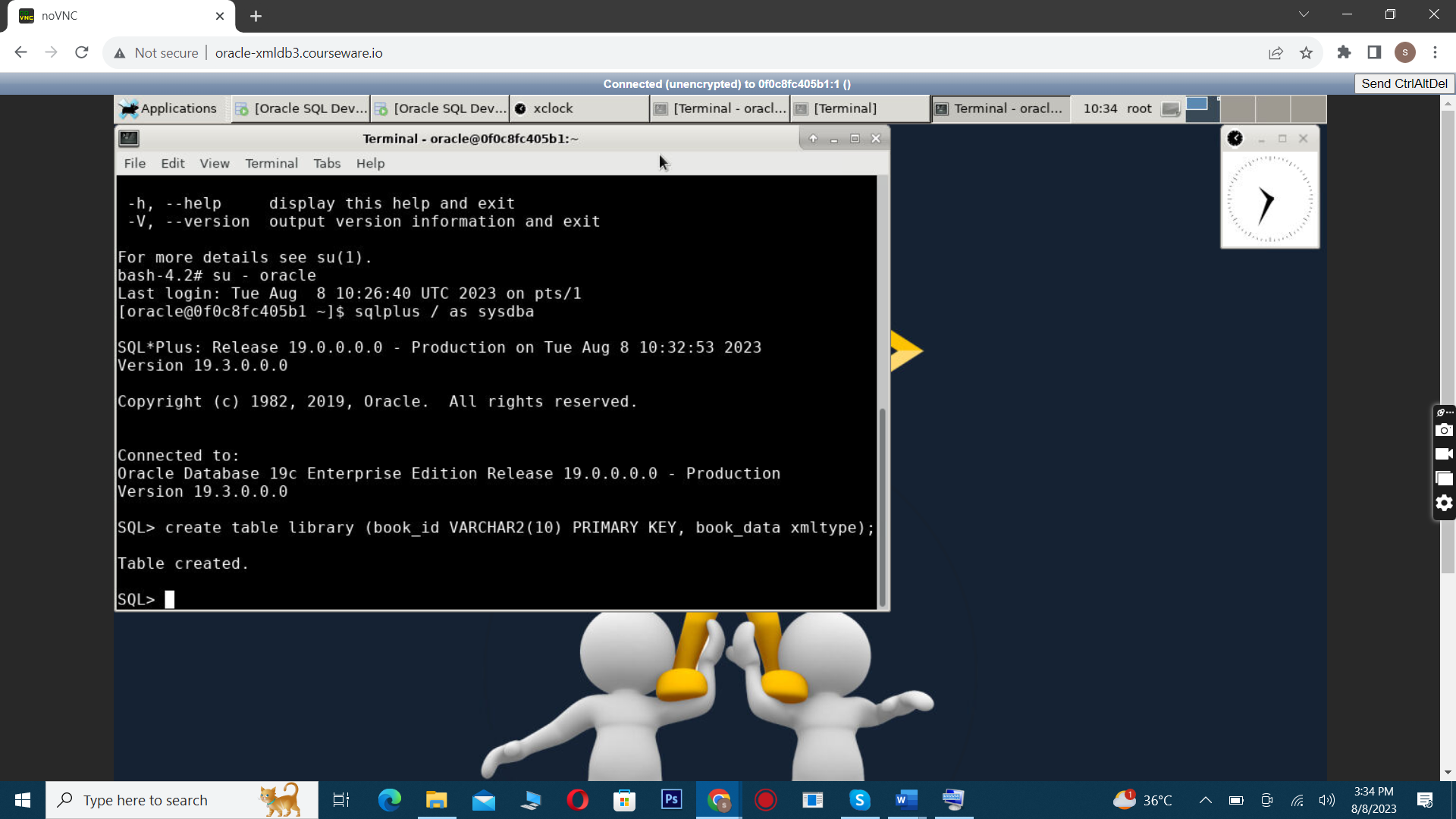
1. **Create a table of XMLType:** Still in the SQL\*Plus prompt, create a table named mytable2 of XMLType with the following command:

CREATE TABLE mytable2 OF XMLType;

sqcCrea\_sq

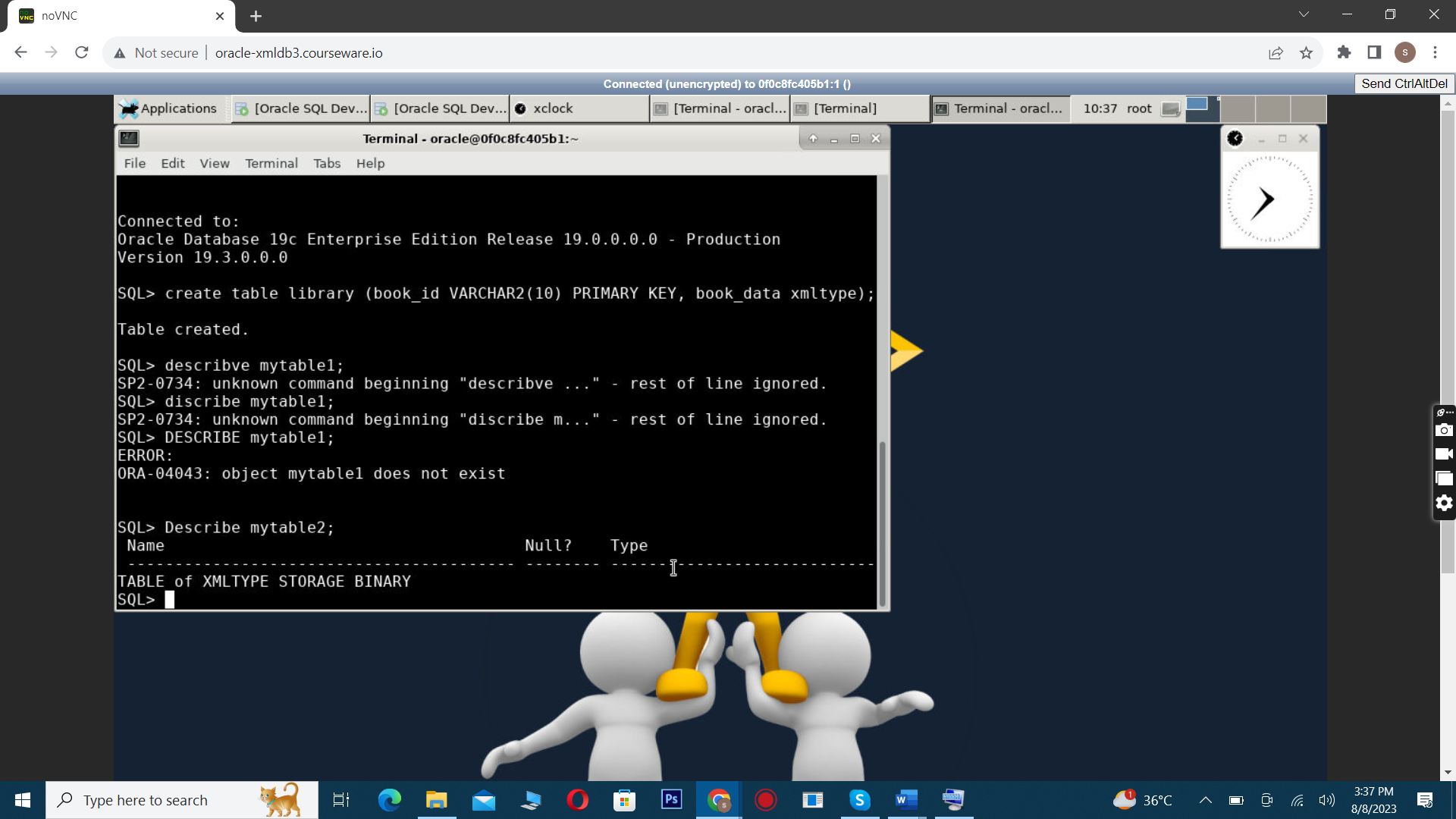
1. **Create a table based on a use case:** For example, let's consider a use case where we are storing XML data for different books in a library. The book\_id will be our primary key and the book\_data will be an XMLType column containing all the book details. Run the following command:

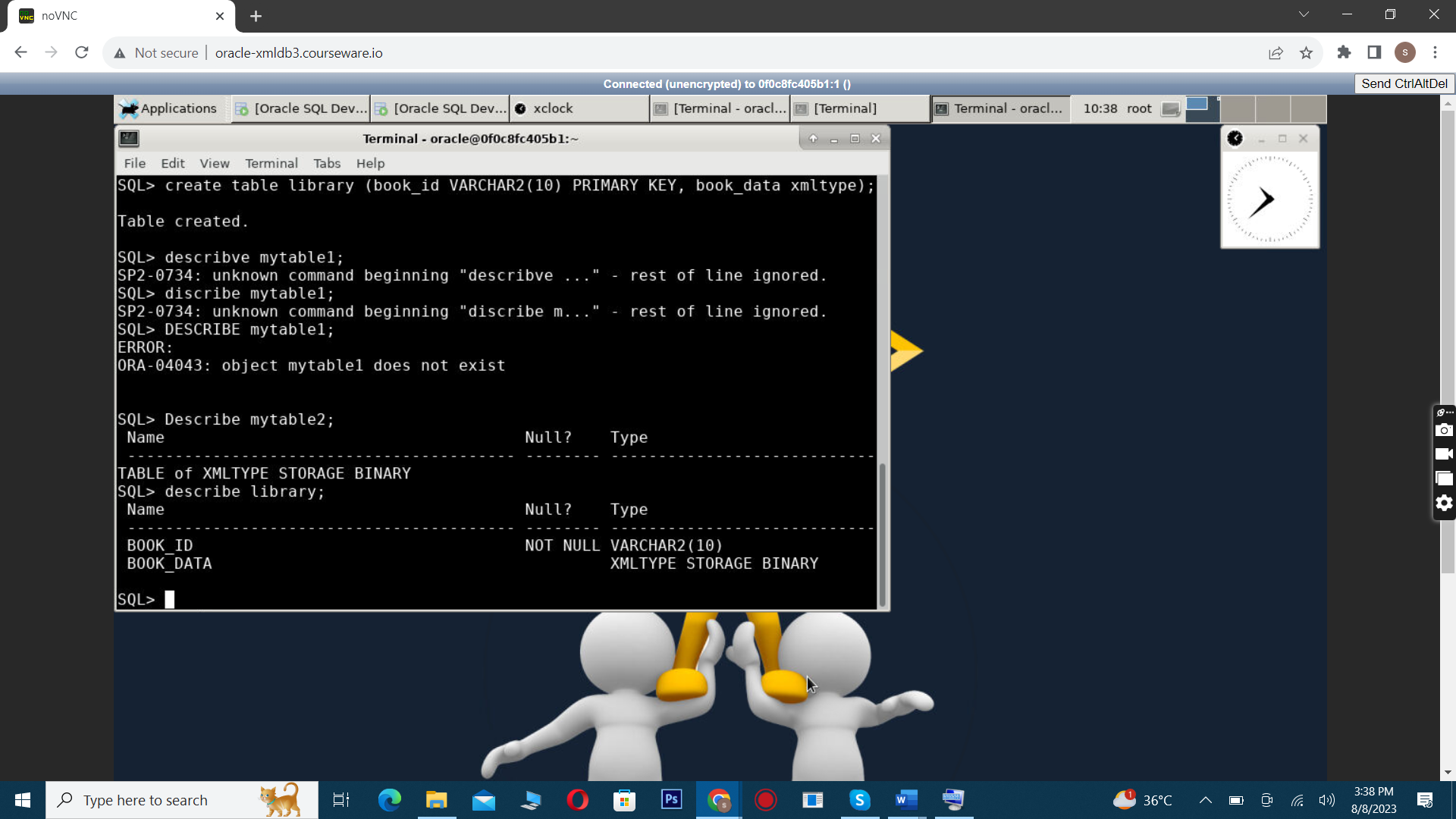
CREATE TABLE library (book\_id VARCHAR2(10) PRIMARY KEY, book\_data XMLType);



1. **Validation step:** To ensure that the tables have been created successfully, we can describe the tables and check their structures. Run the following commands:
2. DESCRIBE mytable1;
3. DESCRIBE mytable2;

DESCRIBE library;





These commands should display the structures of mytable1, mytable2, and library, confirming that they were created successfully.

1. **Exit SQL\*Plus:** Once you've finished with SQL\*Plus, you can exit with the following command:

exit

**Note:** You need to have the necessary permissions to perform these actions. If you're not the Oracle system administrator, you may need to ask them for help.

# **Labe 5**

# **Lab: Connecting to Oracle Database and Validating Prior Labs using SQL Developer**

## Objective

In this lab, we will connect to the Oracle Database using SQL Developer and validate the work done in the previous labs. SQL Developer is a graphical tool that enhances productivity and simplifies database development tasks.

## Prerequisites

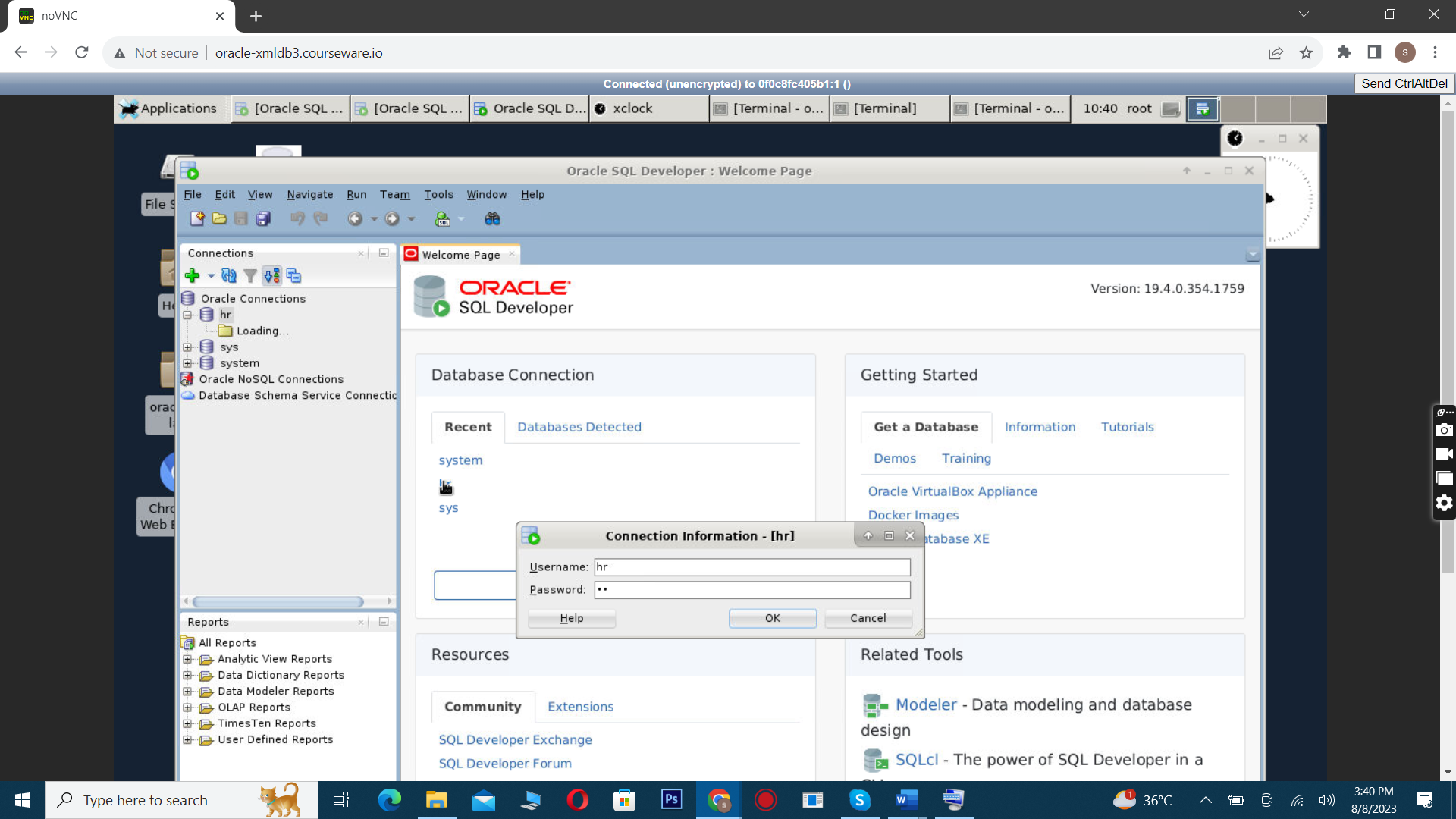
* Oracle SQL Developer installed on your local machine. If you don't have it installed, you can download it from the [official Oracle website](https://www.oracle.com/tools/downloads/sqldev-downloads.html).
* Access to the Oracle Database server, including the server name (or IP address), port, and necessary credentials.

## Steps

1. **Open SQL Developer:** You can do this by finding SQL Developer in your applications menu or list of installed programs, depending on your operating system.
2. **Add a new connection:** Click on the green plus (+) button or go to File -> New -> Database Connection.
3. **Enter the connection details:** In the New / Select Database Connection window, enter the following details (this may already exist and if it does - use the SYS connection):
   * **Connection Name:** Enter a name for this connection. You can choose any name that makes sense to you.
   * **Username:** Enter the username for your Oracle Database. In this case, we will use the SYS user.
   * **Password:** Enter the password for your Oracle Database. In this case, the password is fenago. Remember to check the Save password box if you don't want to enter the password every time you connect.
   * **Hostname:** Enter the hostname or IP address of your Oracle Database server.
   * **Port:** Enter the port number on which your Oracle Database is listening. The default port for Oracle Database is 1521.
   * **SID:** Enter the SID of your Oracle Database. This is usually the name of the database instance.

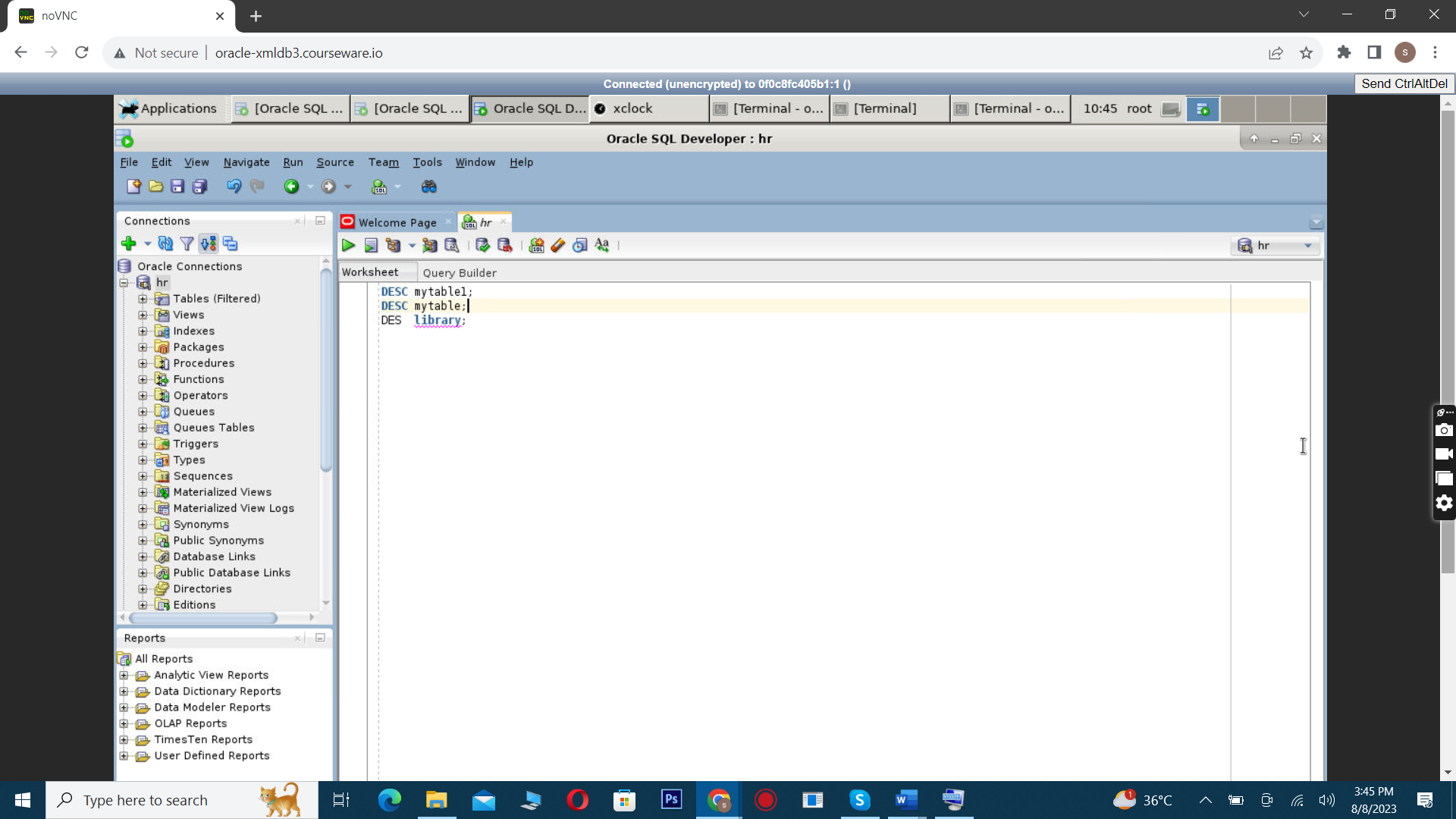
Click on the Test button to check the connection. If everything is set up correctly, you should see a Success message.

1. **Save and connect:** Click on the Connect button to connect to the Oracle Database.



1. **Validate the previous labs:** Once connected, you can run SQL commands to validate the work done in the previous labs. For example, to validate the creation of XMLType tables and columns, you could run the following commands:
2. DESC mytable1;

DESC mytable2;



A computer screen shot of a white screen

Description automatically generated

Each of these commands should display the structure of the corresponding table, confirming that the tables were created successfully.

Additionally, you can navigate to the "Tables" view to confirm the existence of the tables. This can be found in the "Connections" tab, under your connection -> "Schemas" -> "SYS" -> "Tables". The tables mytable1 and library should be listed there.

1. **Disconnect and close SQL Developer:** Once you've finished, remember to disconnect from the Oracle Database by right-clicking on the connection and selecting Disconnect. You can close SQL Developer by going to File -> Exit.

**Note:** You need to have the necessary permissions to connect to the Oracle Database and run these commands. If you're not the Oracle system administrator, you may need to ask them for help.

# Lab: Partitioning Tables That Contain XMLType Data Stored as Binary XML

## Objective

In this lab, we will learn how to partition tables that contain XMLType data stored as binary XML. Partitioning can greatly improve the performance of queries on large tables by dividing the data into smaller, more manageable pieces, called partitions.

## Prerequisites

* Oracle SQL Developer installed on your local machine.
* Access to the Oracle Database server, including the server name (or IP address), port, and necessary credentials.
* Basic knowledge of SQL and database partitioning concepts.

## Steps

1. **Open SQLPlus or SQL Developer:** You can open SQLPlus in a terminal or SSH session, or open SQL Developer on your local machine, depending on your preference.
2. **Connect to the Oracle Database:** If you're using SQLPlus, you can connect directly from the terminal with the following command:

sqlplus / as sysdba

If you're using SQL Developer, create a new connection or use an existing one to connect to the database.

1. **Create a relational table with an XMLType column:** The following SQL command creates a relational table named reltab with a primary key column key\_col and an XMLType column xml\_col. The table is partitioned by range on the key\_col:
2. CREATE TABLE reltab (
3. key\_col VARCHAR2(10) PRIMARY KEY,
4. xml\_col XMLType
5. )
6. XMLTYPE xml\_col STORE AS BINARY XML
7. PARTITION BY RANGE (key\_col)
8. (
9. PARTITION P1 VALUES LESS THAN ('abc'),
10. PARTITION P2 VALUES LESS THAN (MAXVALUE)

);

Run this command in the SQLPlus prompt or in a SQL Developer query window.

1. **Create an XMLType table with a virtual column:** The following SQL command creates an XMLType table named po\_binaryxml with a virtual column date\_col. The virtual column targets the orderDate attribute of the PurchaseOrder element in an XML document stored in the table. The table is partitioned by range on the date\_col:
2. CREATE TABLE po\_binaryxml OF XMLType
3. XMLTYPE STORE AS BINARY XML
4. VIRTUAL COLUMNS
5. (
6. date\_col AS (XMLCast(XMLQuery('/PurchaseOrder/@orderDate' PASSING OBJECT\_VALUE RETURNING CONTENT) AS DATE))
7. )
8. PARTITION BY RANGE (date\_col)
9. (
10. PARTITION orders2001 VALUES LESS THAN (to\_date('01-JAN-2002')),
11. PARTITION orders2002 VALUES LESS THAN (MAXVALUE)

);

Run this command in the SQLPlus prompt or in a SQL Developer query window.

1. **Validate the table creation:** You can validate the creation of the tables and their partitions by describing the tables:
2. DESC reltab;

DESC po\_binaryxml;

These commands should display the structures of the reltab and po\_binaryxml tables, including their partitions, confirming that they were created successfully.

1. **Disconnect from the Oracle Database:** Once you've finished, remember to disconnect from the Oracle Database. In SQLPlus, you can do this with the exit command. In SQL Developer, you can right-click on the connection and select Disconnect.

# 7 Lab: Enforcing XML Data Integrity Using the Database

## Objective

In this lab, we will learn how to use SQL constraints and database triggers to enforce data integrity on XML data stored in Oracle Database. This includes ensuring that the XML documents are well-formed and conform to the XML schema, and enforcing rules such as uniqueness and foreign-key relations using SQL constraints.

## Prerequisites

* Oracle SQL Developer installed on your local machine.
* Access to the Oracle Database server, including the server name (or IP address), port, and necessary credentials.
* Basic knowledge of SQL, XML, and database integrity constraints.

## Steps

1. **Open SQLPlus or SQL Developer:** You can open SQLPlus in a terminal or SSH session, or open SQL Developer on your local machine, depending on your preference.
2. **Connect to the Oracle Database:** If you're using SQLPlus, you can connect directly from the terminal with the following command:

sqlplus / as sysdba

If you're using SQL Developer, create a new connection or use an existing one (SYS) to connect to the database.

1. **Create an XMLType table with a virtual column and a unique constraint:** The following SQL command creates an XMLType table named po\_binaryxml with a virtual column c\_xtabref, and a unique constraint reference\_is\_unique on that column. This ensures that the value of node /PurchaseOrder/Reference/text() is unique across all documents stored in the table:
2. CREATE TABLE po\_binaryxml OF XMLType
3. (CONSTRAINT reference\_is\_unique UNIQUE (c\_xtabref))
4. XMLTYPE STORE AS BINARY XML
5. VIRTUAL COLUMNS
6. (c\_xtabref AS (XMLCast(XMLQuery('/PurchaseOrder/Reference'
7. PASSING OBJECT\_VALUE RETURNING CONTENT)

AS VARCHAR2(32))));

Run this command in the SQLPlus prompt or in a SQL Developer query window.

1. **Insert XML data into the table:** You can insert XML data into the table using the INSERT INTO command. For example, to insert XML data from a file named Invoice.xml located in directory 'XMLDIR', you can use the following command:
2. INSERT INTO po\_binaryxml
3. VALUES (XMLType(bfilename('XMLDIR', 'Invoice.xml'),

nls\_charset\_id('AL32UTF8')));

1. **Validate the data insertion:** After inserting the data, you can run a SELECT command to check that the data was inserted correctly. For example, to select all records from the po\_binaryxml table, you can use the following command:

SELECT \* FROM po\_binaryxml;

This command should return the XML data that you inserted, confirming that the data was inserted successfully and the unique constraint was enforced.

1. **Disconnect from the Oracle Database:** Once you've finished, remember to disconnect from the Oracle Database. In SQLPlus, you can do this with the exit command. In SQL Developer, you can right-click on the connection and select Disconnect.

# **8 Lab: Loading XML Content Using SQL or PL/SQL**

## Objective

In this lab, we will learn how to load XML content into an Oracle Database using SQL or PL/SQL. We will use a simple INSERT operation to load an XML document into the database.

## Prerequisites

* Oracle SQL Developer installed on your local machine.
* Access to the Oracle Database server, including the server name (or IP address), port, and necessary credentials.
* Basic knowledge of SQL, XML, and Oracle PL/SQL.

## Steps

1. **Open SQLPlus or SQL Developer:** You can open SQLPlus in a terminal or SSH session, or open SQL Developer on your local machine, depending on your preference.
2. **Connect to the Oracle Database:** If you're using SQLPlus, you can connect directly from the terminal with the following command:

sqlplus / as sysdba

If you're using SQL Developer, follow the steps below to connect to the SYS user:

* + In the **Connections** pane, click on the **New Connection** button.
  + Fill in the **Connection Name** (you can use any name you like), **Username** (SYS), and **Password**.
  + Check the **Save Password** box if you don't want to enter the password every time you connect.
  + From the **Role** dropdown menu, select SYSDBA.
  + Fill in the **Hostname**, **Port**, and **SID** as per your Oracle Database server configuration.
  + Click **Test** to ensure the connection details are correct. If the **Status** is Success, click **Connect**.

1. **Create a Database Directory:** Before you can insert XML content into an XMLType table, you must create a database directory object that points to the directory containing the file to be processed. To do this, you must have the CREATE ANY DIRECTORY privilege. Run the following command:

CREATE DIRECTORY xmldir AS path\_to\_folder\_containing\_XML\_file;

1. **Insert XML Content into an XMLType Table:** The following SQL command inserts XML content from a file named purchaseOrder.xml located in the directory 'XMLDIR' into an XMLType table named mytable2:

INSERT INTO mytable2 VALUES (XMLType(bfilename('XMLDIR', 'purchaseOrder.xml'), nls\_charset\_id('AL32UTF8')));

1. **Validate the data insertion:** After inserting the data, you can run a SELECT command to check that the data was inserted correctly. For example, to select all records from the mytable2 table, you can use the following command:

SELECT \* FROM mytable2;

This command should return the XML data that you inserted, confirming that the data was inserted successfully.

1. **Disconnect from the Oracle Database:** Once you've finished, remember to disconnect from the Oracle Database. In SQLPlus, you can do this with the exit command. In SQL Developer, you can right-click on the connection and select Disconnect.