Laboratory Enterprise Database Systems	
Laboratory 03	**Oracle Net Services and the Listener** ***Network connections, listener configuration, TNS aliases and Docker port publishing***

Laboratory preparation

Requirements
Docker Desktop or Docker Engine, with an existing Oracle XE container
Oracle Database 21c XE using `gvenzl/oracle-xe:21` and service `XEPDB1`
A *pre-published* range of TCP ports and three ports assigned by the instructor
SQL*Plus inside Docker and Visual Studio Code with Oracle SQL Developer Extension
`SYSTEM` password, and individual remote-server credentials when assigned

Introduction and theoretical background

Purpose. This laboratory explains how an Oracle database accepts connections over a network. You will check the default Oracle listener, configure additional listening endpoints, register the `XEPDB1` service, create connection aliases, test connections from inside and outside Docker, and diagnose failures. You will also use an individually assigned account on the instructor's server. The environment is Oracle Database XE in a **Docker container**, not an Oracle installation on Windows or a VMware virtual machine.

1. Client, database server and session

A **database client** sends connection requests and SQL statements to a server. In this laboratory, the clients are **SQL*Plus** and **Oracle SQL Developer Extension for Visual Studio Code**. An **Oracle instance** manages memory and background processes. A **database session** is created after a successful login and permits operations allowed for the authenticated account.

Oracle XE uses the multitenant architecture. The container database (**CDB**) is normally `XE` and the default pluggable database (**PDB**) is `XEPDB1`. The service `XEPDB1` directs a network connection to that PDB. A **Docker container** is a Linux execution environment and is completely different from an Oracle CDB or PDB.

Term	Meaning in this laboratory
Docker container	Linux environment running Oracle XE, named `oracle-xe` in the examples
Oracle instance / SID	The database instance, normally `XE`
Oracle PDB	Pluggable database, normally `XEPDB1`
Oracle service	The logical connection target, `XEPDB1`
Database account	`SYSTEM` locally, or an individual user on the instructor's server

2. What does Oracle Net Listener do?

Oracle Net Services handles database network connectivity. The **Oracle Net Listener** is a server-side process that accepts a new connection request and locates the required database service. It does not store tables and does not execute the SQL query on behalf of the client. After the connection is established, an Oracle server process handles the session.

A successful connection involves distinct stages. First, the client reaches a network address and TCP port. Second, the listener recognises the requested service. Third, Oracle accepts the account credentials and creates the session. A positive TCP test proves only the first stage. A positive `tnsping` response also does *not* prove that the user can log in.

3. Host address, container address and published TCP ports

Docker uses its own network namespace. Inside the container, `localhost` refers to the *container*. In Windows PowerShell or Visual Studio Code running on the host, `localhost` refers to the *Windows host*. Docker's `-p` option forwards a published host port to a port inside the container.

Example	Meaning
`-p 1521:1521`	Host port `1521` forwards to the listener port `1521` inside Docker
`-p 25000-25019:25000-25019`	Example only: the same 20 TCP port numbers are forwarded individually
`localhost:25001` on Windows	A host-side connection to a published Docker port
`localhost:25001` inside Docker	A direct connection to a listener endpoint in the container

Critical distinction: Publishing a port in Docker does **not** start an Oracle listener on that port. Creating a listener endpoint does **not** publish a Docker port. Both layers are necessary for a connection from Windows to work. The instructor has already prepared the Docker port range. **Students do not create, remove or reconfigure the Docker container.** The range `25000-25019` above is illustrative, not your actual allocation.

4. One listener can use several ports

Oracle's default TCP listener typically uses port `1521`. The `LSNRCTL` utility reports listener status, starts or stops a listener and lists services. Oracle allows one listener to have several TCP listening endpoints, and it also allows several separately named listeners. Each listening process must have its own compatible network address and port combinations.

In this laboratory you will leave the original `LISTENER` on port `1521` unchanged. You will add `LISTENER_LAB` with two TCP ports, `PORT_A` and `PORT_B`, and `LISTENER_2` with one TCP port, `PORT_C`. These port numbers must all belong to the range *already published* by Docker.

The notation `HOST = 0.0.0.0` in a server's listener configuration indicates that the listener accepts connections on available IPv4 interfaces. It does not mean that clients should use `0.0.0.0` as their connection address. In this laboratory, clients use `localhost` or a specific server address.

5. The difference between listener.ora and tnsnames.ora

The `listener.ora` file is a *server-side* configuration file. It defines listeners and their network addresses. The `tnsnames.ora` file defines *client-side* connection aliases, such as `XE_2026`, which refer to a host, a port and a service. It can also contain an address-only alias that Oracle uses for dynamic service registration.

The Oracle client may connect without any TNS alias using **Easy Connect** syntax:

```
sqlplus -L system@//localhost:1521/XEPDB1
```

Here `system` is the database user, `localhost` is the address, `1521` is the TCP port and `XEPDB1` is the service. The password is entered interactively. Do not put the password in a screenshot, a connection command or a report.

6. Dynamic service registration and LOCAL_LISTENER

Starting a listener does not guarantee that it knows the `XEPDB1` service. Oracle's `LREG` background process registers database services with listeners. The instance parameter `LOCAL_LISTENER` identifies the listener addresses that should receive registrations. `ALTER SYSTEM REGISTER` requests registration immediately.

The lab will define a `L03_REG` address-only alias containing port `1521` and the three additional ports. You will then set `LOCAL_LISTENER` to this alias. The default listener remains included so that the normal connection continues to work.

Service is not SID. The instance SID is `XE`, while the PDB service is `XEPDB1`. Static `SID_LIST` declarations are not a substitute for normal dynamic PDB service registration. For this laboratory, use the service name `XEPDB1` and verify that the listener reports it with a `READY` handler.

7. Configuration locations in the gvenzl Oracle XE image

Oracle XE is installed on **Linux inside Docker**. Therefore paths beginning with `C:\` and Windows administrative services from older instructions do not apply. In the `gvenzl/oracle-xe:21` image, the active network files may be symbolic links from Oracle Base Home to persistent files in `/opt/oracle/oradata/dbconfig/XE/`. Their real location is confirmed by `lsnrctl status`, `readlink` and the container's environment. Never assume the Windows host has the same `tnsnames.ora` file as Linux inside Docker.

8. Connecting to the instructor's server

The original laboratory includes a connection to the instructor's independent Oracle server. These parameters are *not* your local Docker container. The remote target in the source laboratory is `172.16.3.170` with service `FREEPDB1` and an individually allocated TCP port and account. The instructor must confirm that the remote service is active for the current class. Your local Docker port range does not control the port on that server.

9. Diagnostic tools

Tool	What it verifies
`docker ps`, `docker port`	Container state and the host-to-container port mappings
`Test-NetConnection`	TCP accessibility of an address and port from Windows
`lsnrctl status`	Listener process, endpoints and summary of known services
`lsnrctl services`	Registered services and their connection handlers
`tnsping`	Oracle Net name resolution and listener response, not authentication
`sqlplus -L`	Complete login to an Oracle database service
Visual Studio Code	External client connection through Docker's published port

Connecting to Oracle XE in Docker

1. The already prepared environment

The instructor provides an existing `oracle-xe` container and a published TCP port range. No student needs to repeat `docker run`. On Windows, check the container using PowerShell:

```
docker ps -a
docker port oracle-xe
docker logs --tail 30 oracle-xe
```

If the container exists but is stopped, use `docker start oracle-xe`. Do *not* create a second container with the same name. Check that the normal Oracle port `1521` and your assigned extra ports are published. Your instructor provides the actual range and the three distinct ports.

*2. SQL*Plus inside the container*

Use this command in Windows PowerShell to start a shell as the `oracle` operating-system account:

```
docker exec -it --user oracle oracle-xe bash -l
```

Inside the container, connect to the local PDB using its default service:

```
sqlplus -L system@//localhost:1521/XEPDB1
```

Enter the password supplied by the instructor. The source L02 example used `Oracle123` when a new database was initialised with this password. An existing Docker volume retains its actual account password. You must not assume that restarting the container resets it.

Verify the identity of the session:

```
SELECT USER AS connected_user,
       SYS_CONTEXT('USERENV','CON_NAME') AS pdb_name,
       SYS_CONTEXT('USERENV','SERVICE_NAME') AS service_name
FROM dual;
```

The expected user and PDB are `SYSTEM` and `XEPDB1`. Type `EXIT` to leave SQL*Plus and `exit` to leave the Linux shell.

*3. Administrative SQL*Plus session*

Only when a task explicitly requires instance configuration, use a local administrator session inside Docker:

```
sqlplus / as sysdba
SHOW CON_NAME
SHOW PARAMETER local_listener
```

This connects as `SYS` with the `SYSDBA` role using operating-system authentication. Confirm that the container is `CDB$ROOT` before using `ALTER SYSTEM SET LOCAL_LISTENER`. Do not connect as `SYSTEM AS SYSDBA`. Return to the normal `SYSTEM` account for routine network tests.

4. Visual Studio Code with the Oracle extension

Install **Oracle SQL Developer Extension for VS Code**, published by Oracle. Open its Database Connections panel and create a connection with the following values. Wording of the controls may vary with extension version.

Connection property	Value
Connection name	`XE-XEPDB1-L03`
Username	`SYSTEM`
Password	Provided password for the existing XE database
Role	**Default**, not `SYSDBA`
Host	`localhost` on the Windows computer
Port	`1521` at the start of the laboratory
Connection type	Basic or equivalent host/port/service connection
Service name	`XEPDB1`, not the SID `XE`

Connect, open a SQL worksheet and run the identity query given above. When you test a different listener, change *only* the port and connection name. Do not confuse the Linux container's `tnsnames.ora` with any separate client-side TNS file on Windows.

> **Evidence during class:** Every required **FOTO** must be captured from your own running environment *during the laboratory*. Show a visible current date and your name or student identifier, for example using a PowerShell window title and date. The report must not expose passwords or the remote control phrase to other students.

Laboratory tasks

Working rule. Complete tasks in order. The published Docker range already exists. Replace every placeholder `PORT_A`, `PORT_B`, `PORT_C` and `REMOTE_PORT` with the assigned *numeric* value before executing the example. Replace `YOUR_LOGIN` with your assigned account. Do not run commands containing these placeholders literally. The first three ports are local Docker ports, while `REMOTE_PORT` is the instructor-server port.

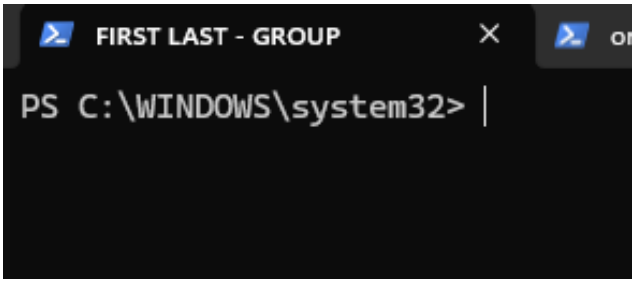


Figure 1: Windows Powershell window

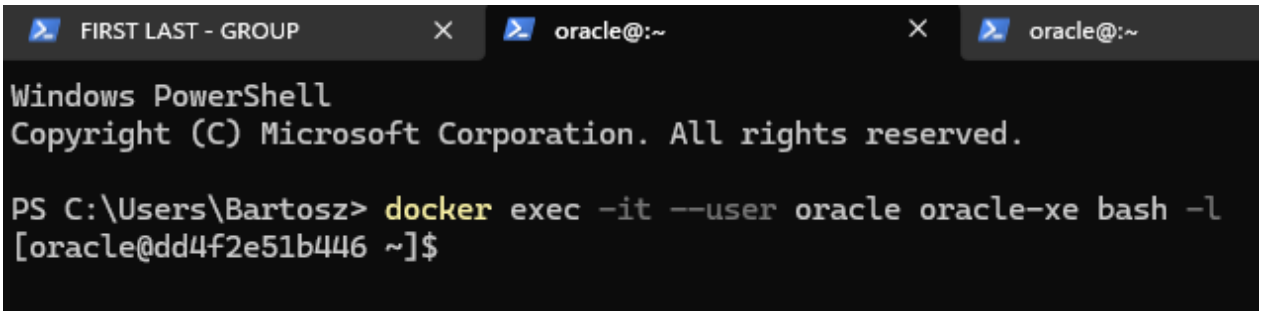


Figure 2: Docker (linux) window

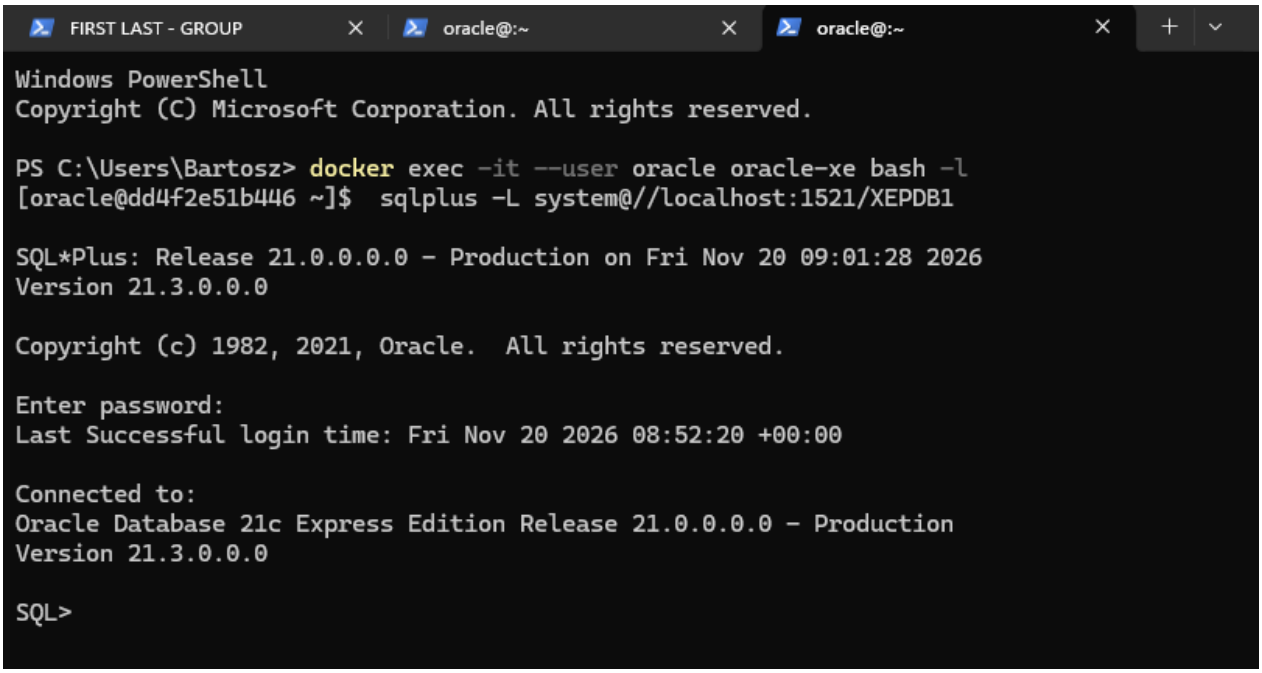


Figure 3: Docker Oracle DB window

Part I. Baseline network verification

1. Identify the host, container and published port range.

FOTO In Windows PowerShell, identify the current user and date, Docker container and published TCP ports. Record the instructor's assigned range separately.

```
$Host.UI.RawUI.WindowTitle = 'FIRST LAST - GROUP'
whoami
Get-Date
docker ps -a
docker port oracle-xe
```

Record and explain: The container name, image, state, default port and already published range. Explain why a published port does not prove that a listener is active.

2. Verify the Oracle installation inside Linux.

Enter the container as shown above. Identify the Linux user, SID, Oracle home and container host name.

```
whoami
hostname
echo "$ORACLE_HOME"
echo "$ORACLE_SID"
```

Record and explain: Which environment belongs to Windows, and which to the Linux container?

3. Perform a full SQL*Plus login to XEPDB1.

FOTO Connect to `SYSTEM` through `localhost:1521/XEPDB1` inside Docker. Run:

```
SELECT USER, SYS_CONTEXT('USERENV','CON_NAME') AS pdb,
       SYS_CONTEXT('USERENV','SERVICE_NAME') AS service
FROM dual;
```

Record and explain: The actual login, PDB and service. Explain why a Docker container is not an Oracle PDB.

4. Connect from Visual Studio Code on Windows.

FOTO Create the `XE-XEPDB1-L03` connection using the properties given in the preparation section. Open an SQL worksheet, execute the identity query and compare the result with SQL*Plus.

Record and explain: The host port, database user and PDB. State whether both clients reach the same Oracle database.

5. Inspect the default listener and registered services.

Inside Docker, run the following Linux commands, not Windows commands:

```
lsnrctl status LISTENER
lsnrctl services LISTENER
```

Record and explain: The listening port, listener parameter file and whether `XEPDB1` is listed. Interpret `READY` and `UNKNOWN` if both appear.

Required step: Installing the Nano text editor
The Oracle XE Docker image does not include the Nano text editor by default. Open a Windows terminal and enter the container as the root user:

```
docker exec -u 0 -it oracle-xe bash
```

Install Nano using the Oracle Linux package manager:

```
microdnf install -y nano
```

After installation, the Nano editor can be used to modify Oracle configuration files, including `listener.ora`.

6. Locate and back up the active configuration.

FOTO **Docker (linux)**. Check where the `listener.ora` and `tnsnames.ora` files point. In the standard image the persistent copies are normally found under `/opt/oracle/oradata/dbconfig/XE/`. Verify the path before modifying anything.

```
ls -l /opt/oracle/homes/OraDBHome21cXE/network/admin/
readlink -f /opt/oracle/homes/OraDBHome21cXE/network/admin/listener.ora
readlink -f /opt/oracle/homes/OraDBHome21cXE/network/admin/tnsnames.ora
```

After confirming the directory, make separate backups:

```
NETCFG=/opt/oracle/oradata/dbconfig/XE
cp "$NETCFG/listener.ora" "$NETCFG/listener.ora.L03.bak"
cp "$NETCFG/tnsnames.ora" "$NETCFG/tnsnames.ora.L03.bak"
```

Record and explain: The real file paths, one important difference between `listener.ora` and `tnsnames.ora`, and the backup file names.

7. Compare TNSPING with a complete login.

Inside Docker, run:

```
tnsping XEPDB1
sqlplus -L system@//localhost:1521/XEPDB1
```

Record and explain: What `tnsping` checks, what SQL*Plus additionally checks, and which command actually creates a database session.

Part II. Additional listeners in an existing Docker container

8. Select three distinct ports from the prepared range.

The instructor provides the range and the assigned values. **Do not calculate ports from a student number and do not choose random ports outside the published range.** Record:

Parameter	Value to enter
Prepared range	_____ through _____
`PORT_A` for `LISTENER_LAB`	_____
`PORT_B` for `LISTENER_LAB`	_____
`PORT_C` for `LISTENER_2`	_____

Record and explain: Why two listeners cannot bind to the same address and TCP port, and why the default `1521` must remain available.

9. Add two named listeners in listener.ora.

Keep the existing default `LISTENER` definition and any other original settings. Edit the verified persistent `listener.ora` file. Append the following complete definitions after replacing the three port placeholders. The syntax uses spaces and balanced parentheses.

```
LISTENER_LAB =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 0.0.0.0)(PORT = PORT_A))
    )
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 0.0.0.0)(PORT = PORT_B))
    )
  )

LISTENER_2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 0.0.0.0)(PORT = PORT_C))
  )
```

Editing note. Use a text editor available in the prepared Docker environment to update the existing file. Do not append duplicate listener definitions. Preserve the original settings and do not create a static `SID_NAME` entry for the PDB.

Record and explain: How many TCP endpoints each named listener has. Explain why `0.0.0.0` is used in a listening configuration, but not as the host in a client connection.

10. Start and inspect both extra listeners.

FOTO In the Docker Linux shell, execute:

```
lsnrctl start LISTENER_LAB
lsnrctl start LISTENER_2
lsnrctl status LISTENER_LAB
lsnrctl status LISTENER_2
-- status must show services like: XE, XEPDB1
```

Record and explain: The actual three endpoint port numbers. Distinguish the two processes from the three TCP ports. Service registration will be checked separately.

Required step: Registering Oracle services with additional listeners
By default, Oracle registers its services with the listener on port `1521`. Additional listeners require explicit configuration through the `LOCAL_LISTENER` parameter.
Connect to the database as `SYSDBA` and execute:

```
ALTER SYSTEM SET LOCAL_LISTENER =
'(ADDRESS_LIST=
  (ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=PORT_A))
  (ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=PORT_B))
  (ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=PORT_C))
)' SCOPE=BOTH;
```

Force immediate service registration:

```
ALTER SYSTEM REGISTER;
```

Verify that the services `XE` and `XEPDB1` are registered with the additional listeners:

```
lsnrctl services LISTENER_LAB
lsnrctl services LISTENER_2
```

Important: Execute the SQL statements in SQL*Plus. Execute the `lsnrctl` commands in the container's Linux shell. The address `127.0.0.1` refers to the Oracle Docker container.

11. Create an address-only registration alias.

Append this *registration alias* to the active `tnsnames.ora` inside Docker. Replace all port placeholders. Do not delete existing `XE` and `XEPDB1` entries.

```
L03_REG =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = PORT_A))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = PORT_B))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = PORT_C))
    )
  )
```

Record and explain: Why this alias does not contain `SERVICE_NAME` and why the address `127.0.0.1` refers to the Docker container in this file.

12. Register XEPDB1 with the new listeners.

FOTO Open a local SQL*Plus administrator connection `sqlplus / as sysdba` *inside Docker*. Ensure that the current container is `CDB$ROOT`, then run:

```
SHOW CON_NAME;
SHOW PARAMETER local_listener;
ALTER SYSTEM SET LOCAL_LISTENER='L03_REG' SCOPE=BOTH;
ALTER SYSTEM REGISTER;
SHOW PARAMETER local_listener;
```

If your original `LOCAL_LISTENER` was non-default, record the original value before making the change. The parameter is persistent and affects where services are registered.

Record and explain: The value before and after the change, the purpose of `LREG` and why `ALTER SYSTEM REGISTER` is useful.

13. Confirm that every listener recognises the PDB service.

FOTO Back in the Linux shell, execute:

```
lsnrctl services LISTENER
lsnrctl services LISTENER_LAB
lsnrctl services LISTENER_2
```

Record and explain: Whether service `XEPDB1` is known to all three listeners and has a `READY` handler. A running listener without the service is not a successful result.

14. Test three local ports without any TNS alias.

FOTO Inside Docker, use three separate SQL*Plus sessions. Replace each port before running the command. After each successful login, run the SQL query and type `EXIT`.

```
sqlplus -L system@//localhost:PORT_A/XEPDB1
sqlplus -L system@//localhost:PORT_B/XEPDB1
sqlplus -L system@//localhost:PORT_C/XEPDB1
```

```
SELECT USER AS login,
       SYS_CONTEXT('USERENV','CON_NAME') AS pdb,
       SYS_CONTEXT('USERENV','SERVICE_NAME') AS service
FROM dual;
```

Record and explain: Three connection outcomes and why one PDB can be accessible through several listener endpoints.

15. Test the three published ports from Windows.

FOTO In **Windows PowerShell**, and not in the container, check the published ports. Each command must use a real numeric port:

```
Test-NetConnection -ComputerName localhost -Port PORT_A
Test-NetConnection -ComputerName localhost -Port PORT_B
Test-NetConnection -ComputerName localhost -Port PORT_C
```

Record and explain: The `TcpTestSucceeded` result for each. Explain why this result alone does not verify `XEPDB1` or the database password.

16. Connect to the new ports from Visual Studio Code.

FOTO On the **Windows host**, create or duplicate three Basic connections in Oracle SQL Developer Extension for VS Code. Set Host to `localhost`, Service name to `XEPDB1`, Username to `SYSTEM` and Port to `PORT_A`, `PORT_B` or `PORT_C`. Use distinct connection labels. Run the session identity query for each.

Record and explain: Which ports work, what database each connection reaches and why these tests use Docker's published ports.

17. Stop a laboratory listener and diagnose the failure.

FOTO Stop *only* `LISTENER_2`, leaving the default listener running:

```
lsnrctl stop LISTENER_2
lsnrctl status LISTENER_2
```

From a *new* SQL*Plus session, try `localhost:PORT_C/XEPDB1` and compare it with `localhost:PORT_A/XEPDB1`. A previously established session is not a valid substitute for a new connection test. Restore the listener:

```
lsnrctl start LISTENER_2
```

As `SYSDBA` inside Docker, rerun the registration command from Task 12 and then repeat the failed login.

Record and explain: The failure message, whether `PORT_A` remained available and the result after restoring `LISTENER_2`.

18. Create and test three local TNS aliases.

FOTO Append the following *client connection aliases* to `tnsnames.ora` **inside Docker**. Unlike `L03_REG`, these entries specify the `XEPDB1` service.

```
XE_2026 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = PORT_A))
    (CONNECT_DATA = (SERVICE_NAME = XEPDB1))
  )

XE_2026A =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = PORT_B))
    (CONNECT_DATA = (SERVICE_NAME = XEPDB1))
  )
```

```
XE_2026B =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = PORT_C))
    (CONNECT_DATA = (SERVICE_NAME = XEPDB1))
  )
```

Test each alias in the Linux shell with `tnsping` and then with SQL*Plus. No listener restart is necessary merely because a client-side alias changed.

```
tnsping XE_2026
sqlplus -L system@XE_2026
tnsping XE_2026A
sqlplus -L system@XE_2026A
tnsping XE_2026B
sqlplus -L system@XE_2026B
```

Record and explain: Three alias names, their associated ports and the distinction between address resolution and successful authentication.

Part III. Remote connection to the instructor's Oracle server

This part is performed only if the instructor confirms that the source laboratory's remote Oracle endpoint is currently available. The source parameters are `172.16.3.170`, service `FREEPDB1`, an **individually assigned remote port**, and an **individual login and password**. Never substitute your local `XEPDB1` port or your local `SYSTEM` password for these values.
Port: 42786
Login: LAB_179454
Password: rZTovtirR31JobZFMjbjA1Ng
Table: LABORATORIUM4

19. Verify the assigned remote TCP port from Windows.

FOTO Record your assigned `REMOTE_PORT` and run in Windows PowerShell:

```
Test-NetConnection -ComputerName 172.16.3.170 -Port REMOTE_PORT
```

Record and explain: The port and result. State whether the response proves anything about your individual Oracle account.

20. Configure a remote TNS alias in Docker.

FOTO Inside the Oracle XE container, append the following alias to the active `tnsnames.ora`. Use the actual remote port.

```
LAB_REMOTE =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 172.16.3.170)(PORT = REMOTE_PORT))
    (CONNECT_DATA = (SERVICE_NAME = FREEPDB1))
  )
```

Check the listener response through the alias and compare with a direct Easy Connect target:

```
tnsping LAB_REMOTE
sqlplus -L YOUR_LOGIN@//172.16.3.170:REMOTE_PORT/FREEPDB1
```

Record and explain: Whether the container can reach the remote network and why neither `LAB_REMOTE` nor the remote host belongs to your local Docker listener configuration.

21. Discover and read your remote verification table.

FOTO Log in using your individual account through `LAB_REMOTE`, not `SYSTEM`. Work through the following steps **in order**. Do not execute all queries as one unexplained block.

Step A. Identify your session. Find the current database username and service using `SYS_CONTEXT` with the `USERENV` namespace. Use the theoretical example from Part I as a starting point. Record the result.

Step B. Find the table. Use the Oracle data dictionary view `USER_TABLES` to find the table named `LABORATORIUM4`. Determine what this view contains and why it lists only tables owned by your account.

Step C. Discover the columns. Find the column names and data types for `LABORATORIUM4` using `USER_TAB_COLUMNS`. Determine which column stores the personal verification value.

Step D. Read the value. Write your own `SELECT` statement using the table and column identified above. Show the individual verification value privately to the instructor together with your student number. Do not publish it in the report.

Record and explain: The identity of the remote session, the discovered table, its relevant column and confirmation of successful access.

Part IV. Independent exercises

For these tasks, work independently. The instructor must approve available port numbers from the *same pre-published Docker range*. Do not create new Docker port mappings. Additional listener names must not conflict with existing names.

22. Create two additional listeners with your own names.

FOTO Choose two unused ports and two listener names based on your initials, such as `LISTENER_BK1` and `LISTENER_BK2`. Add both listener definitions to the existing configuration without breaking the default listener. Start them, inspect their status, extend registration so that `XEPDB1` becomes available on the new ports, and provide the results.

Record and explain: The names, both ports, the edited registration configuration, `READY` service handlers and the successful or unsuccessful result of a full SQL*Plus login.

23. Create independent aliases and use an external graphical client.

FOTO Create one unique client alias for each new listener port, log in from SQL*Plus inside Docker through both aliases, then establish a Basic connection from Oracle SQL Developer Extension for VS Code on Windows to both published ports. Document every successful connection.

Record and explain: Two different connection paths and the exact service, host and port used by the graphical client. Explain why a TNS alias is not a listener.

24. Independently verify and explain listener behavior.

FOTO Without writing Bash, PowerShell or another program, select one additional listener and demonstrate four distinct checks: Docker port publication, TCP accessibility from Windows, registered `XEPDB1` service and a successful SQL*Plus login. Then stop only this laboratory listener, repeat the relevant checks and restore it. Use the existing commands presented in this instruction.

Record and explain: Which checks remain positive after stopping the listener, which fail, and why a TCP connection test is not equivalent to Oracle authentication.

Troubleshooting and recovery

Diagnose the layer that failed. Follow the order Docker → TCP port → listener → service → database credentials. Avoid deleting the container or database volume as a first reaction.

1. Quick checks

```
# Windows PowerShell
docker ps -a
docker port oracle-xe
docker logs --tail 60 oracle-xe
Test-NetConnection localhost -Port PORT_A

# Linux shell inside Docker
lsnrctl status LISTENER
lsnrctl status LISTENER_LAB
lsnrctl services LISTENER_LAB
tnsping XE_2026

# SQL*Plus in Docker, connected as SYSDBA
SHOW PARAMETER local_listener
SHOW PDBS
ALTER SYSTEM REGISTER;
```

The `SHOW PDBS` output should include `XEPDB1` in an open state for normal network login. If the PDB is not open, inform the instructor before making administrative changes.

2. Common problems and safe fixes

Symptom	Probable cause	What to do
`docker exec` fails	Container is stopped or has a different name	Check `docker ps -a`, then `docker start oracle-xe` if appropriate
Windows port test fails, but local Docker SQL*Plus works	Docker port was not published, the wrong host port was used or access is blocked	Check `docker port oracle-xe`. Use only a published port. Ask the instructor to repair port publishing if missing
`TNS-12541` or `ORA-12541`	No listener responds at the address and port	Check `lsnrctl status`, host address and port, then start the named listener if authorised
`ORA-12514`	Listener is reachable but does not know the requested service	Check `lsnrctl services`, `XEPDB1`, `LOCAL_LISTENER` and `ALTER SYSTEM REGISTER`
`ORA-12154`	Alias cannot be resolved	Check the `tnsnames.ora` used by *that client*, spelling, parentheses and `TNS_ADMIN` if configured
`ORA-01017`	Incorrect database username or password	Check credentials and case, distinguish local `SYSTEM` from the remote personal account
`ORA-12505`	A SID-based connection was used for an unknown SID	Use **Service name** `XEPDB1` in the graphical client
`ORA-12170` or timeout	Network path, firewall or listener response delay	Test the TCP port from the same client computer, then inspect Docker and listener status
`TNS-12542` or address already in use	Another process is using the selected TCP address and port	Check the assigned range and listener endpoint definitions, choose an unused instructor-approved port
`TNS-01190` on stop/start	OS user is not authorised to control the listener	Enter the container as `oracle`, or ask the instructor. Do not escalate permissions without approval
`XEPDB1` is not listed with `READY`	Service registration or PDB state is not correct	Check `SHOW PDBS`, `LOCAL_LISTENER`, the registration alias and the current listener ports
SQL*Plus alias works, VS Code alias does not	Linux and Windows have different TNS files or lookup settings	Use a Basic connection from VS Code, or configure a verified Windows-side alias file
Cannot edit network file	Wrong path, permissions or symlink target	Verify `lsnrctl status` and `readlink -f`, use the persistent configuration path as the `oracle` user

3. Recovering a broken listener configuration

The normal database connection on port `1521` must remain functional. If a student configuration prevents listeners from starting, recover the original file *only from your own verified backup*:

```
# Linux shell inside Docker, with instructor approval
NETCFG=/opt/oracle/oradata/dbconfig/XE
cp "$NETCFG/listener.ora.L03.bak" "$NETCFG/listener.ora"
cp "$NETCFG/tnsnames.ora.L03.bak" "$NETCFG/tnsnames.ora"
```

Then return the instance's `LOCAL_LISTENER` setting to its *recorded original value*. In a standard default installation where it was empty, use the following local `SYSDBA` commands:

```
ALTER SYSTEM SET LOCAL_LISTENER='' SCOPE=BOTH;
ALTER SYSTEM REGISTER;
```

Restart or start `LISTENER` only if needed, and retest `localhost:1521/XEPDB1`. If the original value was different, **do not** blindly set it to empty. Restoring a listener file alone does not necessarily restore the database initialization parameter. An existing Docker container cannot acquire additional `-p` port mappings merely by running `docker start` again.

4. Windows paths and SQL*Plus paths

A Windows-host SQL client (including Oracle SQL Developer Extension for VS Code) cannot automatically read `/opt/oracle/oradata/dbconfig/XE/tnsnames.ora`. This is a Linux file *inside Docker*. A Windows copy of the alias file has to be provided and configured separately if aliases are to be resolved on Windows. Direct Basic connections require no TNS alias file.

Assessment and report requirements

A task counts as completed only when the student demonstrates the requested action and explains the result. A `tnsping` result cannot replace a successful SQL*Plus login. Screenshots must show the output of the student's actual Docker environment during class. Use the exact **FOTO** markers beside tasks to identify evidence requests.

Requirements by grade

ECTS	Completed tasks	Minimum screenshots	Required live check
E	1–17	4	Docker ports, default and additional listeners, `XEPDB1` registration, SQL*Plus and VS Code connections
D	1–21	5	Above plus remote TNS alias, individual remote login and guided verification
C	1–23	6	Above plus privileges investigation and controlled negative test
B	1–26	7	Above plus client-side TNS interpretation, troubleshooting and additional listeners
A	1–28	8	Above plus independent aliases, graphical connections and systematic listener diagnosis without programming
F / FX	Requirements for E not met	–	Work not accepted or correction required

If the instructor's remote server is unavailable, the instructor must specify an equivalent check for the remote tasks. The screenshot minima are totals, not additional counts. Every screenshot must show the relevant result from the student's own environment during class. Include task numbers and short interpretations, but never disclose passwords or personal verification values.

ECTS and Polish grading scale

ECTS	ECTS descriptor	Polish grade	Description
A	Excellent	5.0 (very good)	Independent, correct and complete work
B	Very Good	4.5 (good plus)	Above-average performance with minor shortcomings
C	Good	4.0 (good)	Reliable performance with some shortcomings
D	Satisfactory	3.5 (satisfactory plus)	Acceptable work with notable gaps
E	Pass	3.0 (satisfactory)	Minimum requirements met
F / FX	Fail	2.0 (unsatisfactory)	Not passed or correction required

Official reference documentation

- Oracle, *Configuring and Administering Oracle Net Listener*:
  `https://docs.oracle.com/en/database/oracle/oracle-database/21/netag/configuring-and-administering-oracle-net-listener.html`
- Oracle, *Connecting to Oracle Database XE*:
  `https://docs.oracle.com/en/database/oracle/oracle-database/21/xeinl/connecting-oracle-database-free.html`
- Oracle, *Oracle SQL Developer Extension for VS Code*:
  `https://docs.oracle.com/en/database/oracle/sql-developer-vscode/`
- Docker, *docker container run and port publishing*:
  `https://docs.docker.com/reference/cli/docker/container/run/`
- gvenzl, *Oracle XE Docker image and image details*:
  `https://github.com/gvenzl/oci-oracle-xe`