1. Set an environment variable for sensor type.

```
ghannam@Ghannam:~$ export SENSOR_TYPE=temperature
ghannam@Ghannam:~$ echo $SENSOR_TYPE
temperature
ghannam@Ghannam:~$
```

2. Write scripts/sensor_script.py to simulate data logging (timestamps + random values).

```
ghannam@Ghannam:~/iot_logger/scripts$ cat sensor_script.py
#!/usr/bin/env python3
import time
import random
from datetime import datetime

# Simulate logging
while True:
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    value = random.uniform(20.0, 30.0)  # random float between 20 and 30
    print(f"{timestamp} - Sensor Value: {value:.2f}")
    time.sleep(2)  # wait 2 seconds
ghannam@Ghannam:~/iot_logger/scripts$ ./sensor_script.py
2025-09-04 14:00:50 - Sensor Value: 20.72
2025-09-04 14:00:52 - Sensor Value: 20.47
2025-09-04 14:00:54 - Sensor Value: 24.11
2025-09-04 14:00:56 - Sensor Value: 27.50
2025-09-04 14:00:58 - Sensor Value: 22.28
2025-09-04 14:01:00 - Sensor Value: 25.60
```

3. Redirect script output to logs/temperature.log while running as a background process.

```
ghannam@Ghannam:~/iot_logger/logs$ nohup ~/iot_logger/scripts/sensor_script.py > ~/iot_logger/logs/temperature.log 2>&1 &
[4] 4216
ghannam@Ghannam:~/iot_logger/logs$ tail -f ~/iot_logger/logs/temperature.log
```

4. Find the PID of the process, inspect file descriptors in /proc/<pid>/fd.

```
ghannam@Ghannam:~/iot_logger$ ps aux | grep sensor_script.py
ghannam     3420  0.0  0.1  20724 10988 pts/0    S    14:06   0:01 python3 /home/ghannam/iot_logger/scripts/sensor_script.py
ghannam     4216  0.0  0.1  20724 10980 pts/0    S    17:39   0:00 python3 /home/ghannam/iot_logger/scripts/sensor_script.py
ghannam     4244  0.0  0.0   9288  2220 pts/0    S+   17:41   0:00 grep --color=auto sensor_script.py
ghannam@Ghannam:~/iot_logger$
```

5. Filter log data into another file.

```
ghannam@Ghannam:~/iot_logger$ grep "Sensor Value: 2[5-9]" ~/iot_logger/logs/temperature.log > ~/iot_logger/logs/filtered.log
grep: /home/ghannam/iot_logger/logs/temperature.log: binary file matches
```

6. Use wildcards to copy logs to data/.

```
ghannam@Ghannam:~/iot_logger$ cp ~/iot_logger/logs/*.log ~/iot_logger/data/
ghannam@Ghannam:~/iot_logger$
```

7. Clear variable when done.

```
ghannam@Ghannam:~/iot_logger$ unset SENSOR_TYPE
ghannam@Ghannam:~/iot_logger$
```

8. Challenge – Pipes & FD inspection

```
ghannam@Ghannam:~/iot_logger$ cat pipeline_fd.sh
#!/bin/bash

echo "Starting pipeline: (ls -l; sleep 20) | grep .py"
(ls -l; sleep 20) | grep .py &

# give processes a moment to start
sleep 1

# Show relevant processes
echo
echo "Pipeline processes running:"
ps -o pid,ppid,comm | egrep "ls|grep|sleep"

echo
for PID in $(ps -o pid= -C ls -C grep -C sleep); do
    CMD=$(ps -p $PID -o comm=)
    echo "Inspecting file descriptors of $CMD (PID $PID):"
    ls -l /proc/$PID/fd
    echo
done

echo "Done. Processes will exit after sleep finishes."

ghannam@Ghannam:~/iot_logger$ ./pipeline_fd.sh
Starting pipeline: (ls -l; sleep 20) | grep .py

Pipeline processes running:
   4589    4588 sleep
   4590    4588 grep
   4594    4588 grep

Inspecting file descriptors of sleep (PID 4589):
total 0
lrwx------ 1 ghannam ghannam 64 Sep  4 18:05 0 -> /dev/pts/0
l-wx------ 1 ghannam ghannam 64 Sep  4 18:05 1 -> 'pipe:[31016]'
lrwx------ 1 ghannam ghannam 64 Sep  4 18:05 2 -> /dev/pts/0
```

```
Inspecting file descriptors of grep (PID 4590):
total 0
lr-x------ 1 ghannam ghannam 64 Sep  4 18:05 0 -> 'pipe:[31016]'
lrwx------ 1 ghannam ghannam 64 Sep  4 18:05 1 -> /dev/pts/0
lrwx------ 1 ghannam ghannam 64 Sep  4 18:05 2 -> /dev/pts/0

Done. Processes will exit after sleep finishes.
ghannam@Ghannam:~/iot_logger$
```

What's the difference between ' ' and " " in shell?

' ' → single quotes
Everything inside is literal text.
Variables, command substitutions, and special characters are not expanded.

" " → double quotes
Most things inside are expanded:
Variables ($USER, $HOME, etc.)
Command substitution (`date` or $(date))
But still protects text from word splitting and globbing.

[-f filename] vs [-d dirname]

[-f filename]
True if the given path exists and is a regular file.
"Regular file" = not a directory, not a device, not a socket, etc.

[-d dirname]
True if the given path exists and is a directory.

Streams
A process has three main streams:
stdin (0): input
stdout (1): normal output
stderr (2): error output
Redirection
Overwrite (>): sends output to a file, replacing its contents.
Append (>>): sends output to a file, but adds at the end instead of replacing.
stderr redirection (2>): sends error messages to a different place.
stdout + stderr together: both normal output and errors go to the same destination.

```
#!/bin/bash

for i in 1 2 3 4 5
do
  echo "Number: $i"
done
```

```
#!/bin/bash

echo "Enter first number:"
read a
echo "Enter second number:"
read b
echo "Choose operation (+ or -):"
read op

if [ "$op" = "+" ]; then
    echo "Result: $((a + b))"
elif [ "$op" = "-" ]; then
    echo "Result: $((a - b))"
else
    echo "Invalid operation"
fi
```