MoP — Simulating Role-Based Access (PII masking in Python)

GOAL:

- Create a tiny customer dataset with PII (name, phone, address).
- Write a function that returns a *masked view* of the data depending on the role.
- Save/preview each role's output and keep an *access audit log*.

1) One-time setup

Create new folder: "training_PII"
Create new notebook in jupyter browse in above folder: "Privacy"

1.1 Install Required Packages

Run in Anaconda Prompt or Terminal:

```
pip install pandas
```

2) Create a sample CSV (customer_data.csv)

```
import pandas as pd

data = [
  {"customer_id":1001, "name":"Asha Mehta",  "phone":"+91-9876543210", "address":"23 Lodi Rd, Delhi",   "plan_type":"Prepaid",  "arpu_inr":180},
  {"customer_id":1002, "name":"Ravi Kumar",  "phone":"+91-9988776655", "address":"12 Marine Dr, Mumbai","plan_type":"Postpaid", "arpu_inr":280},
  {"customer_id":1003, "name":"Sneha Rao",   "phone":"+91-9090909090", "address":"7 Anna Salai, Chennai","plan_type":"Prepaid", "arpu_inr":210},
  {"customer_id":1004, "name":"Manoj Singh", "phone":"+91-9123456789", "address":"44 CR Park, Delhi",   "plan_type":"Postpaid", "arpu_inr":320},
  {"customer_id":1005, "name":"Divya Jain",  "phone":"+91-9000011111", "address":"18 Park St, Kolkata", "plan_type":"Prepaid",  "arpu_inr":120}
]
pd.DataFrame(data).to_csv("customer_data.csv", index=False)
print("✅ customer_data.csv created")
```

3) Define masking policies and the access function

> We *mask, we don't drop*—so row counts stay identical across roles.

```
import pandas as pd
from datetime import datetime
import os

AUDIT_LOG = "access_audit.log"

def log_access(role: str, fields_returned: list):
    ts = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(AUDIT_LOG, "a", encoding="utf-8") as f:
        f.write(f"[{ts}] role={role} fields={fields_returned}\n")

# ---- Role policy (easy to read/modify) ----
#   - "mask" => partial masking
#   - "hide" => fully redacted
#   - "show" => no change
POLICY = {
    "junior_analyst": {
        "name":   "hide",
        "phone":  "hide",
        "address":"hide",
        "arpu_inr":"show",
        "plan_type":"show"
    },
    "senior_analyst": {
        "name":   "hide",
        "phone":  "mask",     # keep only first 3 digits, mask rest
        "address":"hide",
        "arpu_inr":"show",
        "plan_type":"show"
    },
    "manager": {
        "name":   "show",
        "phone":  "show",
        "address":"show",
        "arpu_inr":"show",
        "plan_type":"show"
    }
}

def apply_masking(df: pd.DataFrame, role: str) -> pd.DataFrame:
    if role not in POLICY:
        raise ValueError(f"Unknown role: {role}")

    pol = POLICY[role]
    out = df.copy()

    # NAME
    if pol.get("name") == "hide" and "name" in out:
        out["name"] = "ANONYMIZED"

    # PHONE
    if "phone" in out:
        if pol.get("phone") == "hide":
            out["phone"] = "XXXXX"
        elif pol.get("phone") == "mask":
            # keep first 3 digits/characters; mask the rest uniformly
            out["phone"] = out["phone"].astype(str).str[:3] + "-XXX-XXXX"
    # ADDRESS
    if pol.get("address") == "hide" and "address" in out:
        out["address"] = "HIDDEN"

    # Everything else is passed through (“show”)
    return out

def get_customer_data(role: str, fields: list | None = None) -> pd.DataFrame:
    """Return a role-appropriate view. Optional `fields` lets you choose columns."""
    df = pd.read_csv("customer_data.csv")

    # Always keep stable technical columns (id)
    mandatory = ["customer_id"]
    if fields is None:
        fields = list(df.columns)  # start with all
    fields = list(dict.fromkeys(mandatory + fields))  # ensure unique, keep order
    df = df[fields]

    masked = apply_masking(df, role)
    log_access(role, fields)
    return masked
```

4) See what each role gets (and save to files)

```
cols = ["customer_id", "name", "phone", "address", "plan_type", "arpu_inr"]

print(" Junior Analyst view:")
jun = get_customer_data("junior_analyst", cols)
display(jun.head())
jun.to_csv("view_junior.csv", index=False)

print("\n Senior Analyst view:")
sen = get_customer_data("senior_analyst", cols)
display(sen.head())
sen.to_csv("view_senior.csv", index=False)

print("\n Manager view:")
mgr = get_customer_data("manager", cols)
display(mgr.head())
mgr.to_csv("view_manager.csv", index=False)

print("\n Audit Log preview:")
print(open(AUDIT_LOG, "r", encoding="utf-8").read().splitlines()[-3:])
```

What you should see

* *Junior Analyst*: name=ANONYMIZED, phone=XXXXX, address=HIDDEN
* *Senior Analyst*: name=ANONYMIZED, phone=+91-XXX-XXXX style, address=HIDDEN
* *Manager*: full unmasked data
* Files created: view_junior.csv, view_senior.csv, view_manager.csv
* access_audit.log records who accessed what