Complete Guide: ETL Pipeline for Junior Data Engineer

Introduction to ETL

ETL stands for Extract, Transform, Load. It is a data integration process that collects data from multiple sources, transforms it into a consistent format, and loads it into a target system (data warehouse, database, or files). Why is ETL important? Because raw data is always messy, unstructured, and inconsistent. Before you can use it for analytics or machine learning, you must make it clean, structured, and standardized.

ETL Pipeline Architecture

An ETL pipeline has three main phases: 1. Extraction: CSV, JSON, XML, APIs, GPS, Sensors, Images, Videos. 2. Transformation: cleaning, unit conversion, feature engineering. 3. Loading: store in database, data warehouse, or data lake. It also includes logging, monitoring, and maintenance.

Key Concepts for Junior Data Engineer

Essential knowledge: SQL basics, Python (Pandas, NumPy), file formats, APIs, data cleaning, statistics. Advanced: Data warehouses, orchestration tools, distributed systems, cloud ETL. Soft Skills: Communication, problem solving, curiosity.

Common Interview Questions

Basics: What is ETL? Differences ETL vs ELT? Explain ETL steps. File Formats: Difference between CSV, JSON, Parquet. SQL: How do you join tables? What is indexing? Transformations: Handle missing data? GPS distance? Image preprocessing? Real-world: What if schema changes? How do you monitor ETL jobs?

Best Practices for ETL Maintenance

Log each step, keep schema docs updated, validate input data, write unit tests, version control with Git, schedule jobs with Airflow, automate error alerts.

Conclusion

For a Junior Data Engineer, mastering ETL pipelines is the foundation of data engineering. Learn to extract, transform, load, and always monitor and maintain your pipelines. Then move towards scalable ETL with Spark, Airflow, and cloud platforms.

Appendix: Full Python ETL Code

```
import pandas as pd
import glob
import json
import xml.etree.ElementTree as ET
from datetime import datetime
from geopy.distance import geodesic
import cv2
import os
import numpy as np

log_file = "etl_log.txt"
target_file = "final_dataset.csv"

def log_progress(message):
    now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    with open(log_file, "a") as f:
        f.write(now + " | " + message + "\n")
    print("■", message)

def extract_from_csv(file):
    return pd.read_csv(file)

def extract_from_json(file):
    return pd.read_json(file, lines=True)

def extract_from_xml(file):
    data = []
    tree = ET.parse(file)
    root = tree.getroot()
    for person in root:
        name = person.find("name").text
        height = float(person.find("height").text)
        weight = float(person.find("weight").text)
        data.append({"name": name, "height": height, "weight": weight})
    return pd.DataFrame(data)

def extract_from_gps(file):
    df = pd.read_csv(file)
    return df

def extract_from_sensor(file):
    df = pd.read_csv(file)
    return df

def extract_images(folder):
    images = []
    for img_file in glob.glob(os.path.join(folder, "*.jpg")):
        images.append(img_file)
    return images

def extract_videos(folder):
    videos = []
    for vid_file in glob.glob(os.path.join(folder, "*.mp4")):
        videos.append(vid_file)
    return videos

def extract():
    all_data = pd.DataFrame()
    for f in glob.glob("*.csv"):
        all_data = pd.concat([all_data, extract_from_csv(f)], ignore_index=True)
    for f in glob.glob("*.json"):
        all_data = pd.concat([all_data, extract_from_json(f)], ignore_index=True)
    for f in glob.glob("*.xml"):
        all_data = pd.concat([all_data, extract_from_xml(f)], ignore_index=True)
    for f in glob.glob("gps_data/*.csv"):
        gps_df = extract_from_gps(f)
        gps_df["source"] = "gps"
        all_data = pd.concat([all_data, gps_df], ignore_index=True)
    for f in glob.glob("sensor_data/*.csv"):
        sensor_df = extract_from_sensor(f)
        sensor_df["source"] = "sensor"
        all_data = pd.concat([all_data, sensor_df], ignore_index=True)
    images = extract_images("images")
```

```
    videos = extract_videos("videos")
    return all_data, images, videos

def transform_structured(df):
    if "height" in df.columns:
        df["height"] = round(df["height"] * 0.0254, 2)
    if "weight" in df.columns:
        df["weight"] = round(df["weight"] * 0.45359237, 2)
    if "name" in df.columns:
        df["name"] = df["name"].astype(str).str.title()
    return df

def transform_gps(df):
    if "lat" in df.columns and "lon" in df.columns:
        df["distance_m"] = 0.0
        for i in range(1, len(df)):
            p1 = (df.loc[i-1,"lat"], df.loc[i-1,"lon"])
            p2 = (df.loc[i,"lat"], df.loc[i,"lon"])
            df.loc[i,"distance_m"] = geodesic(p1, p2).meters
        df["cumulative_distance"] = df["distance_m"].cumsum()
    return df

def transform_sensor(df):
    if "voltage" in df.columns:
        df["temperature_C"] = (df["voltage"] - 0.5) * 100
    return df

def transform_image(img_path):
    img = cv2.imread(img_path)
    img = cv2.resize(img, (224,224))
    img = img / 255.0
    return img

def transform_video(video_path):
    cap = cv2.VideoCapture(video_path)
    frames = []
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame = cv2.resize(frame, (224,224))
        frames.append(frame / 255.0)
    cap.release()
    return np.array(frames[:16])

def load_data(df, target_file):
    df.to_csv(target_file, index=False)
    log_progress(f"Data saved to {target_file}")

log_progress("ETL Job Started")
log_progress("Extraction Started")
structured_data, image_files, video_files = extract()
log_progress("Extraction Finished")

log_progress("Transformation Started")
structured_data = transform_structured(structured_data)
structured_data = transform_gps(structured_data)
structured_data = transform_sensor(structured_data)
if len(image_files) > 0:
    img_array = transform_image(image_files[0])
if len(video_files) > 0:
    vid_array = transform_video(video_files[0])
log_progress("Transformation Finished")

log_progress("Loading Started")
load_data(structured_data, target_file)
log_progress("Loading Finished")
log_progress("ETL Job Completed ■")
```