# Confluence-Ready Coding & Naming Standards

## Navigation

This document is structured with headings. When imported into Confluence, headings will automatically generate a navigation tree or can be linked using the Table of Contents macro.

Quick Links (Confluence will auto-generate navigation):

• 4.1 Coding Standards

• 4.2 Naming Conventions

## 4.1 Detailed Coding Standards

### Quick Glance

* Project & Code Org: Thin main() → reusable functions; packaged as wheel; notebooks orchestrate only.
* Config: JSON/YAML overlays; no literals/secrets; validate config at load.
* Spark Session: Created in common lib and injected; AQE & DPP enabled; UTC enforced.
* DataFrames: Schema-first; use Spark built-ins; avoid RDDs.
* I/O: Default to Delta; follow zone structure (landing → historical → harvest → publish).
* Performance: Join clean keys; handle skew; tune shuffle partitions; avoid unnecessary cache.
* Idempotency: Safe re-runs; use high-water marks; Delta MERGE.
* Observability: Structured JSON logs; row counts; DQ metrics per zone.

### Detailed Glance

#### 4.1.1 Project & Code Organization

Use thin entry points (main) with orchestration only. Reusable functions live in /src/common, packaged as wheel. Notebooks are orchestration only.

Example:

# /src/data\_products/fdp\_sales/sales\_pipeline.py  
from common.audit\_utils import add\_audit\_columns  
from common.session import get\_spark  
  
def run\_sales\_pipeline(config, run\_id):  
 spark = get\_spark("fdp\_sales\_pipeline", config["spark"])  
 df\_landing\_sales = spark.read.schema(config["schemas"]["sales"]).parquet(config["landing\_path"])  
 df\_harvest\_sales = transform\_sales(df\_landing\_sales)  
 df\_harvest\_sales = add\_audit\_columns(df\_harvest\_sales, run\_id, "CRM")  
 df\_harvest\_sales.write.format("delta").mode("overwrite").save(config["harvest\_path"])

#### 4.1.2 Configuration Management

Environment overlays are stored in JSON/YAML per environment. Secrets in Key Vault, not config.

Example (dev.json):

{  
 "landing\_path": "abfss://landing@devstorage/retail/fdp\_sales/",  
 "harvest\_path": "abfss://harvest@devstorage/retail/fdp\_sales/",  
 "spark": {  
 "shuffle\_partitions": 200,  
 "timezone": "UTC"  
 },  
 "schemas": {  
 "sales": "customer\_id STRING, order\_id STRING, amount DOUBLE"  
 }  
}

#### 4.1.3 Spark Session Standards

Spark session setup in common lib with injected configs.

Example (/src/common/session.py):

from pyspark.sql import SparkSession  
  
def get\_spark(app\_name: str, spark\_conf: dict):  
 builder = SparkSession.builder.appName(app\_name)  
 for k, v in spark\_conf.items():  
 builder = builder.config(k, v)  
 builder = (builder  
 .config("spark.sql.adaptive.enabled", "true")  
 .config("spark.sql.optimizer.dynamicPartitionPruning.enabled", "true")  
 .config("spark.sql.session.timeZone", "UTC"))  
 return builder.getOrCreate()

#### 4.1.4 DataFrame Standards

Always explicit schemas, prefer built-ins, avoid RDD.

Example:

schema = "customer\_id STRING, order\_id STRING, amount DOUBLE"  
df\_landing\_sales = spark.read.schema(schema).json(config["landing\_path"])

#### 4.1.5 I/O Standards

Delta format is default. Zones follow landing → historical → harvest → publish. Partition by business date.

Example:

df\_harvest\_sales.write.format("delta").mode("overwrite") \  
 .partitionBy("year", "month", "day") \  
 .save(config["harvest\_path"])

#### 4.1.6 Performance Practices

Tune shuffle partitions. Clean join keys. Handle skew with salting/broadcast.

Example:

from pyspark.sql.functions import broadcast  
df\_result = df\_large.join(broadcast(df\_small), "id")

#### 4.1.7 Idempotency & Re-Runs

Jobs safe to rerun. Use Delta MERGE and high-water marks.

Example:

target.alias("t").merge(  
 source.alias("s"),  
 "t.customer\_id = s.customer\_id"  
).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()

#### 4.1.8 Observability & Audit

Structured JSON logging and audit columns (ingest\_ts, run\_id, src\_system).

Example:

df\_harvest\_sales = df\_harvest\_sales \  
 .withColumn("ingest\_ts", current\_timestamp()) \  
 .withColumn("run\_id", lit(run\_id)) \  
 .withColumn("src\_system", lit("CRM"))

## 4.2 Detailed Naming Conventions

### Quick Glance

* Functions: verb\_noun()
* Variables: snake\_case
* DataFrames: prefixed with df\_ (zone included if relevant)
* Columns: snake\_case; audit cols ingest\_ts, run\_id, src\_system
* Tables: <domain>\_<product>\_<layer>\_<entity>
* Paths: <zone>/<product>/<entity>/year=YYYY/month=MM/day=DD/
* Files: <entity>\_<yyyymmdd>\_<run\_id>.parquet

### Detailed Glance with Examples

#### Functions

def load\_landing\_sales():  
 ...  
def transform\_sales\_orders(df\_sales):  
 ...

#### Variables

is\_incremental\_load = True  
batch\_run\_id = "20250821\_1230"

#### DataFrames

df\_landing\_sales = spark.read.parquet(config["landing\_path"])  
df\_harvest\_sales = transform\_sales(df\_landing\_sales)  
df\_publish\_customers = df\_harvest\_sales.filter("status = 'ACTIVE'")

#### Columns

df\_harvest\_sales = df\_harvest\_sales \  
 .withColumn("ingest\_ts", current\_timestamp()) \  
 .withColumn("run\_id", lit(run\_id)) \  
 .withColumn("src\_system", lit("CRM"))

#### Tables

retail\_fdp\_harvest\_sales  
banking\_cdp\_publish\_transactions

#### Paths

/harvest/fdp\_sales/orders/year=2025/month=08/day=21/

#### Files

orders\_20250821\_12345.parquet