SNIPPET 1 — Incremental Load (CRM Customers)

```
crm["updated_at"] = pd.to_datetime(crm["updated_at"])
last_watermark = pd.to_datetime("2024-01-01")
crm_incremental = crm[crm["updated_at"] > last_watermark]
```

SNIPPET 2 — Customer Anonymization (PII Handling)

```
import hashlib
def generate_customer_key(val):
return hashlib.sha256(val.encode()).hexdigest()
crm_incremental["CustomerKey"] =
crm_incremental["customer_id"].apply(generate_customer_key)
```

SNIPPET 3 — SCD Type 1 Upsert (Pseudo-code)

```
MERGE INTO dim_customer_anonymized AS tgt
USING crm_incremental AS src
ON tgt.CustomerKey = src.CustomerKey
WHEN MATCHED THEN
  UPDATE SET *
WHEN NOT MATCHED THEN
  INSERT *
```

SNIPPET 4 — Data Quality Checks

```
invalid_txns = transactions[transactions["txn_amount"] < 0]
```

SQL 1 — Incremental Load (CRM)

```
SELECT *
FROM crm_customers
WHERE updated_at > (
  SELECT MAX(last_processed_at)
  FROM control_table
  WHERE source_name = 'crm_customers'
);
```

SQL 2 — CustomerKey Generation (Hashing)

```
SELECT
  SHA2(customer_id, 256) AS CustomerKey,
  city,
  country
FROM crm_customers;
```

SQL 3 — SCD Type 1 Merge

```
MERGE INTO dim_customer_anonymized tgt
USING staging_customers src
ON tgt.CustomerKey = src.CustomerKey
WHEN MATCHED THEN
  UPDATE SET
    tgt.city = src.city,
    tgt.country = src.country
WHEN NOT MATCHED THEN
  INSERT (CustomerKey, city, country)
  VALUES (src.CustomerKey, src.city, src.country);
```

SQL 4 — Data Quality Check (Negative Transactions)

```
SELECT *
FROM core_transactions
WHERE txn_amount < 0;
```