

Wealth Insights – FULL 2-Table PySpark ETL (Complete Code)

This PDF contains the complete end-to-end code (no omissions).

```
#!/usr/bin/env python3
# =====
# Wealth Insights ETL (last 6 complete months) - Build 2 tables
#
# OUTPUT TABLES:
# 1) dm_ib_dev.wealth_insights_customer_dim
# 2) dm_ib_dev.wealth_insights_accounts_fact
#
# SOURCES (as provided):
# - Digital      : dm_ib.digital_banking_master
# - InvestPath  : eil.d_involved_party_h + eil.d_arrangement_to_involved_party_relationship_h + eil.d_arrangement_h
# - RCIF        : eil.d_involved_party_h + eil.d_involved_party_address_h
# - RCIF_Number : eil.m_involved_party_h (6mo) UNION dm_ib.digital_banking_master (6mo)
# - Wealth      : eil.m_involved_party_h + eil.m_arrangement_to_involved_party_relationship_h + eil.m_arrangement_h
#
# NOTES:
# - Uses last 6 COMPLETE months (month boundaries).
# - Digital fix:
#   * digital_base (no aggregation)
#   * digital_agg (aggregate login dates)
#   * digital_monthly (flags computed using month_dt)
# =====

from pyspark.sql import SparkSession, functions as F

APP_NAME = "wealth_insights_etl_6mo_dim_fact"

DB = "dm_ib_dev"
DIM = f"{DB}.wealth_insights_customer_dim"
FACT = f"{DB}.wealth_insights_accounts_fact"

def build_spark() -> SparkSession:
    spark = (
        SparkSession.builder
        .appName(APP_NAME)
        .enableHiveSupport()
        .config("spark.sql.catalogImplementation", "hive")
        .config("spark.sql.adaptive.enabled", "true")
        .config("spark.sql.adaptive.skewJoin.enabled", "true")
        .config("spark.sql.shuffle.partitions", "64")
        .getOrCreate()
    )
    spark.sparkContext.setLogLevel("WARN")
    spark.sql(f"USE {DB}")
    return spark

def main():
    spark = build_spark()

    # -----
    # 0) Date Window: last 6 complete months
    # -----
    spark.sql("""
CREATE OR REPLACE TEMP VIEW date_window AS
SELECT
    add_months(trunc(current_date(), 'MM'), -6) AS start_month,
    trunc(current_date(), 'MM')                AS end_month
""")
    spark.sql("SELECT * FROM date_window").show(truncate=False)

    # -----
    # 1) RCIF universe (RCIF_Number idea)
    # -----
    spark.sql("""
CREATE OR REPLACE TEMP VIEW rcif_universe AS
```

```

SELECT DISTINCT CAST(rcif_number AS STRING) AS rcif_number
FROM (
    SELECT CAST(rcif_cust_nbr AS STRING) AS rcif_number
    FROM eil.m_involved_party_h
    WHERE CAST(business_date AS DATE) >= add_months(trunc(current_date(),'MM'), -6)

    UNION

    SELECT CAST(rcif_customer_nbr AS STRING) AS rcif_number
    FROM dm_ib.digital_banking_master
    WHERE CAST(ods_business_dt AS DATE) >= add_months(trunc(current_date(),'MM'), -6)
) u
WHERE rcif_number IS NOT NULL AND TRIM(rcif_number) <> ''
""")

# -----
# 2) CUSTOMER DIM (RCIF latest snapshot)
# -----
spark.sql("""
CREATE OR REPLACE TEMP VIEW rcif_latest_date AS
SELECT MAX(CAST(business_date AS DATE)) AS last_dt
FROM eil.d_involved_party_h
""")

spark.sql("""
CREATE OR REPLACE TEMP VIEW customer_dim_stg AS
SELECT
    CAST(ip.rcif_cust_nbr AS STRING)                AS rcif_number,
    CAST(ip.involved_party_id AS STRING)             AS involved_party_id,
    CAST(ip.cust_internet_banking_nbr AS STRING)     AS cust_internet_banking_nbr,
    CAST(ip.involved_party_name AS STRING)           AS involved_party_name,
    CAST(ip.birth_date AS DATE)                     AS birth_date,
    CAST(addr.city_name AS STRING)                   AS city_name,
    CAST(addr.state_name AS STRING)                  AS state_name,
    CAST(addr.country_name AS STRING)                AS country_name,

CASE
    WHEN ip.birth_date BETWEEN DATE '1900-01-01' AND DATE '1924-12-31' THEN 'GI Generation (1900-1924)'
    WHEN ip.birth_date BETWEEN DATE '1925-01-01' AND DATE '1945-12-31' THEN 'Traditionalist (1925-1945)'
    WHEN ip.birth_date BETWEEN DATE '1946-01-01' AND DATE '1964-12-31' THEN 'Baby Boomer (1946-1964)'
    WHEN ip.birth_date BETWEEN DATE '1965-01-01' AND DATE '1980-12-31' THEN 'Gen X (1965-1980)'
    WHEN ip.birth_date BETWEEN DATE '1981-01-01' AND DATE '1996-12-31' THEN 'Millennial (1981-1996)'
    WHEN ip.birth_date >= DATE '1997-01-01' THEN 'Centennial (1997-???)'
    ELSE 'Unknown'
END AS customer_generation,

(SELECT last_dt FROM rcif_latest_date) AS as_of_date

FROM eil.d_involved_party_h ip
INNER JOIN rcif_latest_date ld
    ON CAST(ip.business_date AS DATE) = ld.last_dt
LEFT JOIN eil.d_involved_party_address_h addr
    ON ip.involved_party_id = addr.involved_party_id
    AND ip.business_date = addr.business_date
WHERE ip.source_system_code = 'CF'
    AND NVL(ip.deceased_ind,'N') = 'N'
""")

spark.sql("""
CREATE OR REPLACE TEMP VIEW customer_dim AS
SELECT d.*
FROM customer_dim_stg d
INNER JOIN rcif_universe u
    ON d.rcif_number = u.rcif_number
""")

spark.sql(f"DROP TABLE IF EXISTS {DIM}")
spark.sql(f"""
CREATE TABLE {DIM}
STORED AS PARQUET
AS
SELECT
    rcif_number,
    involved_party_id,
    cust_internet_banking_nbr,
    involved_party_name,
    birth_date,
    customer_generation,

```

```

    city_name,
    state_name,
    country_name,
    as_of_date
FROM customer_dim
""")

# -----
# 3) DIGITAL monthly (FIXED)
# -----

# Step 3.1: base rows (no aggregation)
spark.sql("""
CREATE OR REPLACE TEMP VIEW digital_base AS
SELECT
    TRUNC(CAST(dbm.ods_business_dt AS DATE), 'MM') AS month_dt,
    CAST(dbm.rcif_customer_nbr AS STRING)          AS rcif_number,
    CAST(dbm.ibn AS STRING)                        AS relt_ibn,
    CAST(dbm.olb_last_login_date AS DATE)          AS olb_login_dt,
    CAST(dbm.mob_last_login_date AS DATE)          AS mob_login_dt
FROM dm_ib.digital_banking_master dbm
WHERE CAST(dbm.ods_business_dt AS DATE) >= add_months(trunc(current_date(),'MM'), -6)
    AND CAST(dbm.ods_business_dt AS DATE) < trunc(current_date(),'MM')
    AND dbm.rcif_customer_nbr IS NOT NULL
""")

# Step 3.2: aggregate to (month_dt, rcif_number, relt_ibn)
spark.sql("""
CREATE OR REPLACE TEMP VIEW digital_agg AS
SELECT
    month_dt,
    rcif_number,
    relt_ibn,
    MAX(olb_login_dt) AS lst_login_olb,
    MAX(mob_login_dt) AS lst_login_mob
FROM digital_base
GROUP BY month_dt, rcif_number, relt_ibn
""")

# Step 3.3: compute flags using month_dt
spark.sql("""
CREATE OR REPLACE TEMP VIEW digital_monthly AS
SELECT
    month_dt,
    rcif_number,
    relt_ibn,
    lst_login_olb,
    lst_login_mob,

    CASE
        WHEN datediff(month_dt, lst_login_mob) <= 90 THEN 'Mobile Active'
        ELSE 'Non Mobile Active'
    END AS mobile_active_flag,

    CASE
        WHEN lst_login_mob IS NULL THEN 'Non Mobile User'
        ELSE 'Mobile User'
    END AS mobile_flag,

    CASE
        WHEN datediff(month_dt, lst_login_olb) <= 90 THEN 'OLB Active'
        ELSE 'Non OLB Active'
    END AS olb_active_flag,

    CASE
        WHEN lst_login_olb IS NULL THEN 'Non OLB User'
        ELSE 'OLB User'
    END AS olb_flag,

    CASE
        WHEN (datediff(month_dt, lst_login_mob) <= 90)
            OR (datediff(month_dt, lst_login_olb) <= 90)
        THEN 'Digital Active'
        ELSE 'Non Digital Active'
    END AS digitally_active_flag
FROM digital_agg
""")

# Step 3.4: roll up to customer-month

```

```

spark.sql("""
CREATE OR REPLACE TEMP VIEW digital_customer_month AS
SELECT
    month_dt,
    rcif_number,
    MAX(CASE WHEN digitally_active_flag = 'Digital Active' THEN 1 ELSE 0 END) AS digital_active_ind,
    MAX(CASE WHEN mobile_active_flag = 'Mobile Active' THEN 1 ELSE 0 END) AS mobile_active_ind,
    MAX(CASE WHEN olb_active_flag = 'OLB Active' THEN 1 ELSE 0 END) AS olb_active_ind,
    MAX(CASE WHEN mobile_flag = 'Mobile User' THEN 1 ELSE 0 END) AS mobile_user_ind,
    MAX(CASE WHEN olb_flag = 'OLB User' THEN 1 ELSE 0 END) AS olb_user_ind,
    COUNT(DISTINCT relt_ibn) AS digital_enrollments_cnt
FROM digital_monthly
GROUP BY month_dt, rcif_number
""")

# -----
# 4) WEALTH monthly
# -----
spark.sql("""
CREATE OR REPLACE TEMP VIEW wealth_last_dates AS
SELECT DISTINCT CAST(business_date AS DATE) AS last_dt
FROM eil.m_involved_party_h
WHERE CAST(business_date AS DATE) >= add_months(trunc(current_date(),'MM'), -6)
AND CAST(business_date AS DATE) < trunc(current_date(),'MM')
""")

spark.sql("""
CREATE OR REPLACE TEMP VIEW wealth_pwl AS
SELECT
    TRUNC(CAST(ind.business_date AS DATE), 'MM') AS month_dt,
    CAST(ind.rcif_cust_nbr AS STRING) AS rcif_number,

CASE
    WHEN ind.private_client_code IN ('039','539','339') THEN 'Private Wealth'
    WHEN ind.private_client_trust_code IN ('239','739') THEN 'Private Wealth'
    ELSE CASE
        WHEN ar.business_service_segment_type_code IN ('IS_CT','IS_IT') THEN 'Institutional Services'
        WHEN ar.business_service_segment_type_code IN ('REGIS_FC','REGIS') THEN 'Investment Services'
        WHEN ar.business_service_segment_type_code IN ('PWM') THEN 'Private Wealth'
        ELSE CONCAT(ar.business_service_segment_type_code,' Category???)
    END
END AS business_group,

COUNT(DISTINCT CASE WHEN ar.business_service_segment_type_code = 'IS_CT' THEN ar.arrangement_id END) AS corporate_trust_count,
COUNT(DISTINCT CASE WHEN ar.business_service_segment_type_code = 'IS_IT' THEN ar.arrangement_id END) AS institutional_trust_count,
COUNT(DISTINCT CASE WHEN ar.business_service_segment_type_code = 'REGIS_FC' THEN ar.arrangement_id END) AS investment_count,
COUNT(DISTINCT CASE WHEN ar.business_service_segment_type_code = 'REGIS' THEN ar.arrangement_id END) AS insurance_count,
COUNT(DISTINCT CASE WHEN ar.business_service_segment_type_code = 'PWM' THEN ar.arrangement_id END) AS pwm_count,
COUNT(DISTINCT CASE WHEN ar.source_system_code = 'TR' THEN ar.arrangement_id END) AS trust_count,

COUNT(DISTINCT CASE
    WHEN ar.source_system_code IN ('DA','SV','CC','MG','LS','TM','PC','LO','BW','CM','CS','EL','IC','MA','PF','PR','SD','TR','')
    THEN ar.arrangement_id END
) AS banking_count,

COUNT(ar.arrangement_id) AS accts_cnt

FROM eil.m_involved_party_h ind
INNER JOIN wealth_last_dates d
    ON CAST(ind.business_date AS DATE) = d.last_dt

INNER JOIN eil.m_arrangement_to_involved_party_relationship_h a2i
    ON ind.involved_party_id = a2i.involved_party_id
    AND ind.business_date = a2i.business_date
    AND ind.source_system_code = a2i.source_system_code

INNER JOIN eil.m_arrangement_h ar
    ON a2i.arrangement_id = ar.arrangement_id
    AND a2i.arrangement_source_system_code = ar.source_system_code
    AND a2i.business_date = ar.business_date

WHERE ind.source_system_code = 'CF'
AND NVL(ind.deceased_ind,'N') = 'N'
AND ar.closed_ind = 'N'
AND ar.source_system_code IN ('BI','RN','TR','DA','SV','CC','LS','MG','TM','PC','LO','BW','CS','IC','MA','PF','PR','SD','CM')
AND (
    CASE
        WHEN ind.private_client_code IN ('039','539','339') THEN 1

```

```

        WHEN ind.private_client_trust_code IN ('239','739') THEN 1
        ELSE CASE
            WHEN ar.business_service_segment_type_code IN ('IS_CT','IS_IT','REGIS_FC','REGIS','PWM') THEN 1
            ELSE 0
        END
    END
END
) = 1

GROUP BY
    TRUNC(CAST(ind.business_date AS DATE), 'MM'),
    CAST(ind.rcif_cust_nbr AS STRING),
    ar.business_service_segment_type_code,
    ind.private_client_code,
    ind.private_client_trust_code
""")

spark.sql("""
CREATE OR REPLACE TEMP VIEW wealth_customer_month AS
SELECT
    month_dt,
    rcif_number,
    MAX(business_group) AS business_group,

    MAX(
        CASE
            WHEN business_group = 'Private Wealth' THEN
                CASE
                    WHEN trust_count > 0 AND banking_count > 0 THEN 'Banking & IM&T'
                    ELSE CASE
                        WHEN (investment_count + trust_count) > 0 AND banking_count = 0 THEN 'Investments Only'
                        ELSE 'Banking only'
                    END
                END
            WHEN business_group = 'Investment Services' THEN
                CASE
                    WHEN investment_count > 0 AND insurance_count = 0 THEN 'Investment'
                    WHEN investment_count = 0 AND insurance_count > 0 THEN 'Insurance'
                    ELSE 'Insurance & Investment'
                END
            ELSE
                CASE
                    WHEN corporate_trust_count > 0 AND institutional_trust_count = 0 THEN 'Corporate Trust'
                    WHEN corporate_trust_count = 0 AND institutional_trust_count > 0 THEN 'Institutional Trust'
                    WHEN pwm_count > 0 THEN 'Banking only'
                    ELSE 'Corporate & Institutional Trust'
                END
            END
        ) AS division,

    SUM(accts_cnt) AS wealth_accounts_cnt,

    SUM(corporate_trust_count) AS corporate_trust_count,
    SUM(institutional_trust_count) AS institutional_trust_count,
    SUM(investment_count) AS investment_count,
    SUM(insurance_count) AS insurance_count,
    SUM(pwm_count) AS pwm_count,
    SUM(trust_count) AS trust_count,
    SUM(banking_count) AS banking_count
FROM wealth_pwl
GROUP BY month_dt, rcif_number
""")

# -----
# 5) InvestPath (latest snapshot, mapped to month)
# -----
spark.sql("""
CREATE OR REPLACE TEMP VIEW investpath_last_date AS
SELECT MAX(CAST(business_date AS DATE)) AS last_dt
FROM eil.d_involved_party_h
""")

spark.sql("""
CREATE OR REPLACE TEMP VIEW investpath_accounts AS
SELECT
    TRUNC(ld.last_dt, 'MM') AS month_dt,
    CAST(ind.rcif_cust_nbr AS STRING) AS rcif_number,

```

```

        CAST(ind.involved_party_id AS STRING)          AS ip_id,
        CAST(ar.arrangement_id AS STRING)             AS account_id,
        CAST(ar.current_balance_amt AS DOUBLE)        AS balance_amt
FROM eil.d_involved_party_h ind
INNER JOIN investpath_last_date ld
    ON CAST(ind.business_date AS DATE) = ld.last_dt

INNER JOIN eil.d_arrangement_to_involved_party_relationship_h a2i
    ON ind.involved_party_id = a2i.involved_party_id
    AND ind.business_date = a2i.business_date
    AND ind.source_system_code = a2i.source_system_code

INNER JOIN eil.d_arrangement_h ar
    ON a2i.arrangement_id = ar.arrangement_id
    AND a2i.arrangement_source_system_code = ar.source_system_code
    AND a2i.business_date = ar.business_date

WHERE ind.source_system_code = 'CF'
    AND NVL(ind.deceased_ind, 'N') = 'N'
    AND ar.closed_ind = 'N'
    AND ar.account_type_code = 'IP'
    AND ar.source_system_code = 'RN'
""")

spark.sql("""
CREATE OR REPLACE TEMP VIEW investpath_customer_month AS
SELECT
    month_dt,
    rcif_number,
    COUNT(DISTINCT ip_id) AS investpath_customers_cnt,
    COUNT(DISTINCT account_id) AS investpath_accounts_cnt,
    SUM(balance_amt) AS investpath_balance_amt,
    COUNT(DISTINCT CASE WHEN balance_amt > 0 THEN account_id END) AS investpath_accounts_funded_cnt
FROM investpath_accounts
GROUP BY month_dt, rcif_number
""")

# -----
# 6) Combine into FACT (customer-month grain)
# -----
spark.sql("""
CREATE OR REPLACE TEMP VIEW fact_stg AS
SELECT
    COALESCE(d.month_dt, w.month_dt, i.month_dt) AS month_dt,
    COALESCE(d.rcif_number, w.rcif_number, i.rcif_number) AS rcif_number,

    -- Digital
    COALESCE(d.digital_enrollments_cnt, 0) AS digital_enrollments_cnt,
    COALESCE(d.digital_active_ind, 0) AS digital_active_ind,
    COALESCE(d.mobile_active_ind, 0) AS mobile_active_ind,
    COALESCE(d.olb_active_ind, 0) AS olb_active_ind,
    COALESCE(d.mobile_user_ind, 0) AS mobile_user_ind,
    COALESCE(d.olb_user_ind, 0) AS olb_user_ind,

    -- Wealth
    w.business_group,
    w.division,
    COALESCE(w.wealth_accounts_cnt, 0) AS wealth_accounts_cnt,
    COALESCE(w.corporate_trust_count, 0) AS corporate_trust_count,
    COALESCE(w.institutional_trust_count, 0) AS institutional_trust_count,
    COALESCE(w.investment_count, 0) AS investment_count,
    COALESCE(w.insurance_count, 0) AS insurance_count,
    COALESCE(w.pwm_count, 0) AS pwm_count,
    COALESCE(w.trust_count, 0) AS trust_count,
    COALESCE(w.banking_count, 0) AS banking_count,

    -- InvestPath
    COALESCE(i.investpath_customers_cnt, 0) AS investpath_customers_cnt,
    COALESCE(i.investpath_accounts_cnt, 0) AS investpath_accounts_cnt,
    COALESCE(i.investpath_balance_amt, 0.0) AS investpath_balance_amt,
    COALESCE(i.investpath_accounts_funded_cnt, 0) AS investpath_accounts_funded_cnt

FROM digital_customer_month d
FULL OUTER JOIN wealth_customer_month w
    ON d.month_dt = w.month_dt AND d.rcif_number = w.rcif_number
FULL OUTER JOIN investpath_customer_month i
    ON COALESCE(d.month_dt, w.month_dt) = i.month_dt
    AND COALESCE(d.rcif_number, w.rcif_number) = i.rcif_number

```

```

"""

spark.sql("""
CREATE OR REPLACE TEMP VIEW fact_filtered AS
SELECT f.*
FROM fact_stg f
INNER JOIN rcif_universe u
    ON f.rcif_number = u.rcif_number
WHERE f.month_dt >= add_months(trunc(current_date(),'MM'), -6)
    AND f.month_dt < trunc(current_date(),'MM')
""")

# Write FACT (partitioned by month_dt)
spark.sql(f"DROP TABLE IF EXISTS {FACT}")
spark.sql(f"""
CREATE TABLE {FACT} (
    rcif_number STRING,
    digital_enrollments_cnt BIGINT,
    digital_active_ind INT,
    mobile_active_ind INT,
    olb_active_ind INT,
    mobile_user_ind INT,
    olb_user_ind INT,
    business_group STRING,
    division STRING,
    wealth_accounts_cnt BIGINT,
    corporate_trust_count BIGINT,
    institutional_trust_count BIGINT,
    investment_count BIGINT,
    insurance_count BIGINT,
    pwm_count BIGINT,
    trust_count BIGINT,
    banking_count BIGINT,
    investpath_customers_cnt BIGINT,
    investpath_accounts_cnt BIGINT,
    investpath_balance_amt DOUBLE,
    investpath_accounts_funded_cnt BIGINT
)
PARTITIONED BY (month_dt DATE)
STORED AS PARQUET
""")

spark.sql(f"""
INSERT OVERWRITE TABLE {FACT} PARTITION (month_dt)
SELECT
    rcif_number,
    digital_enrollments_cnt,
    digital_active_ind,
    mobile_active_ind,
    olb_active_ind,
    mobile_user_ind,
    olb_user_ind,
    business_group,
    division,
    wealth_accounts_cnt,
    corporate_trust_count,
    institutional_trust_count,
    investment_count,
    insurance_count,
    pwm_count,
    trust_count,
    banking_count,
    investpath_customers_cnt,
    investpath_accounts_cnt,
    investpath_balance_amt,
    investpath_accounts_funded_cnt,
    month_dt
FROM fact_filtered
""")

# -----
# 7) Quick checks similar to Power BI cards
# -----
print("\n===== QUICK CHECKS (6-month window) =====\n")

spark.sql(f"""
SELECT COUNT(DISTINCT rcif_number) AS wealth_users
FROM {FACT}

```

```

WHERE wealth_accounts_cnt > 0
""").show()

spark.sql(f"""
SELECT COUNT(DISTINCT rcif_number) AS digital_active_customers
FROM {FACT}
WHERE digital_active_ind = 1
""").show()

spark.sql(f"""
SELECT month_dt, SUM(digital_enrollments_cnt) AS digital_enrollments
FROM {FACT}
GROUP BY month_dt
ORDER BY month_dt
""").show(50, False)

spark.sql(f"""
SELECT
    SUM(wealth_accounts_cnt) AS total_wealth_accounts,
    CASE WHEN COUNT(DISTINCT CASE WHEN wealth_accounts_cnt > 0 THEN rcif_number END) = 0 THEN NULL
         ELSE SUM(wealth_accounts_cnt) / COUNT(DISTINCT CASE WHEN wealth_accounts_cnt > 0 THEN rcif_number END)
    END AS accounts_per_user
FROM {FACT}
""").show()

spark.sql(f"""
WITH w AS (
    SELECT
        COUNT(DISTINCT CASE WHEN wealth_accounts_cnt > 0 THEN rcif_number END) AS wealth_users,
        COUNT(DISTINCT CASE WHEN wealth_accounts_cnt > 0 AND digital_active_ind = 1 THEN rcif_number END) AS wealth_digital_active
    FROM {FACT}
)
SELECT
    wealth_users,
    wealth_digital_active,
    CASE WHEN wealth_users = 0 THEN NULL ELSE wealth_digital_active / wealth_users END AS adoption_pct
FROM w
""").show(truncate=False)

spark.sql(f"""
SELECT
    month_dt,
    SUM(investpath_customers_cnt) AS investpath_customers,
    SUM(investpath_accounts_cnt) AS investpath_accounts,
    SUM(investpath_balance_amt) AS investpath_aum,
    CASE WHEN SUM(investpath_accounts_cnt) = 0 THEN NULL
         ELSE SUM(investpath_balance_amt) / SUM(investpath_accounts_cnt)
    END AS avg_balance_per_ip_account,
    SUM(investpath_accounts_funded_cnt) AS investpath_accounts_funded
FROM {FACT}
WHERE investpath_accounts_cnt > 0
GROUP BY month_dt
ORDER BY month_dt
""").show(50, False)

print("\n===== DONE. Tables created =====\n")
print(f"DIM : {DIM}")
print(f"FACT: {FACT}")

spark.stop()

if __name__ == "__main__":
    main()

```