📄Technical Design Document

Project: Mutual Fund Portfolio360 Dashboard

Objective:

Build a data pipeline and interactive dashboard to track mutual fund investments using Snowflake, DBT, and Streamlit.

---

1. 📦Data Sources and Raw Layer (Snowflake)

1.1 Tables and Structures

These tables will be created in the **RAW schema** of your Snowflake database.

a. `investor_master`

```sql
CREATE OR REPLACE TABLE raw.investor_master (
    investor_id       STRING PRIMARY KEY,
    name              STRING,
    email             STRING,
    phone             STRING,
    pan_number        STRING,
    created_date      DATE
);
```

Sample Insert:

```sql
INSERT INTO raw.investor_master VALUES
('INV001', 'Raj Mehta', 'raj@example.com', '9999988888', 'ABCDE1234F',
'2023-01-01'),
('INV002', 'Priya Kapoor', 'priya@example.com', '8888877777', 'PQRSX6789L',
'2023-01-05');
```

---

b. `mutual_fund_transactions`

```sql
CREATE OR REPLACE TABLE raw.mutual_fund_transactions (
    transaction_id     STRING PRIMARY KEY,
    investor_id        STRING,
    fund_name          STRING,
    scheme_code        STRING,
    transaction_type   STRING,  -- BUY / SELL / SIP
    amount             FLOAT,
    nav_at_time        FLOAT,
    units              FLOAT,
    transaction_date   DATE
);
```

Sample Insert:

```
INSERT INTO raw.mutual_fund_transactions VALUES
('TXN001', 'INV001', 'Axis Bluechip Fund', 'AXIS123', 'BUY', 5000, 50.0, 100,
'2023-01-15'),
('TXN002', 'INV001', 'Axis Bluechip Fund', 'AXIS123', 'BUY', 3000, 60.0, 50,
'2023-03-01'),
('TXN003', 'INV002', 'HDFC Midcap Opp', 'HDFC456', 'BUY', 4000, 40.0, 100,
'2023-02-10');
```

---

c. `nav_history`

```
CREATE OR REPLACE TABLE raw.nav_history (
    scheme_code       STRING,
    fund_name         STRING,
    nav_date          DATE,
    nav_value         FLOAT
);
```

Sample Insert:

```
INSERT INTO raw.nav_history VALUES
('AXIS123', 'Axis Bluechip Fund', '2023-01-15', 50.0),
('AXIS123', 'Axis Bluechip Fund', '2023-03-01', 60.0),
('AXIS123', 'Axis Bluechip Fund', '2024-06-30', 75.0),
('HDFC456', 'HDFC Midcap Opp', '2023-02-10', 40.0),
('HDFC456', 'HDFC Midcap Opp', '2024-06-30', 68.0);
```

---

d. `fund_master`

```
CREATE OR REPLACE TABLE raw.fund_master (
    scheme_code        STRING PRIMARY KEY,
    fund_name           STRING,
    category            STRING,  -- Large Cap, Mid Cap, etc.
    amc_name            STRING,  -- Fund house
    benchmark_index     STRING
);
```

Sample Insert:

```
INSERT INTO raw.fund_master VALUES
('AXIS123', 'Axis Bluechip Fund', 'Large Cap', 'Axis AMC', 'Nifty 100'),
('HDFC456', 'HDFC Midcap Opp', 'Mid Cap', 'HDFC AMC', 'Nifty Midcap 150');
```

---

2. 💣 DBT Architecture and Components

2.1 Folder Structure

```
dbt_mutual_funds/
├── models/
│   ├── staging/
│   │   ├── stg_investor_master.sql
│   │   ├── stg_transactions.sql
│   │   ├── stg_nav_history.sql
│   │   └── stg_fund_master.sql
│   ├── marts/
│   │   └── fct_current_holdings.sql
├── tests/
├── snapshots/
├── docs/
└── dbt_project.yml
```

2.2 Models

a. `stg_transactions.sql`

```
SELECT
  transaction_id,
  investor_id,
  scheme_code,
  transaction_type,
  amount,
  nav_at_time,
  units,
  transaction_date
FROM {{ source('raw', 'mutual_fund_transactions') }}
```

b. `fct_current_holdings.sql`

```
WITH buys AS (
  SELECT investor_id, scheme_code,
         SUM(CASE WHEN transaction_type = 'BUY' THEN units ELSE 0 END) AS
total_units,
         SUM(CASE WHEN transaction_type = 'BUY' THEN amount ELSE 0 END) AS
total_invested
  FROM {{ ref('stg_transactions') }}
  GROUP BY 1, 2
),
latest_nav AS (
  SELECT scheme_code, MAX(nav_date) AS latest_date
  FROM {{ ref('stg_nav_history') }}
  GROUP BY 1
),
nav AS (
  SELECT nav1.scheme_code, nav_value
  FROM {{ ref('stg_nav_history') }} nav1
  JOIN latest_nav nav2 ON nav1.scheme_code = nav2.scheme_code AND nav1.nav_date
= nav2.latest_date
)

SELECT
  b.investor_id,
  b.scheme_code,
  f.fund_name,
  b.total_units,
  ROUND(b.total_invested / NULLIF(b.total_units, 0), 2) AS avg_nav,
  b.total_invested,
  n.nav_value AS current_nav,
  ROUND(b.total_units * n.nav_value, 2) AS current_value,
  ROUND((b.total_units * n.nav_value - b.total_invested) /
NULLIF(b.total_invested, 0) * 100, 2) AS return_percentage
FROM buys b
JOIN nav n ON b.scheme_code = n.scheme_code
JOIN {{ ref('stg_fund_master') }} f ON b.scheme_code = f.scheme_code
```

2.3 DBT Snapshots (optional)

Track changes in NAV or transaction corrections.

```
-- snapshots/snap_nav_history.sql
{% snapshot snap_nav_history %}
{{
    config(
      target_schema='snapshots',
      unique_key='scheme_code, nav_date',
      strategy='check',
      check_cols=['nav_value']
    )
}}
SELECT * FROM {{ source('raw', 'nav_history') }}
{% endsnapshot %}
```

2.4 DBT Tests

```
version: 2

models:
  - name: stg_transactions
    columns:
      - name: transaction_id
        tests:
          - not_null
          - unique
  - name: fct_current_holdings
    columns:
      - name: investor_id
        tests:
          - not_null
```

2.5 DBT Docs

Use `description` blocks in your `.yml` files:

```
models:
  - name: fct_current_holdings
    description: "Final fact table showing investor-wise holdings and returns."
```

Then run:

```
dbt docs generate
dbt docs serve
```

---

3. 📈Streamlit App (Optional Frontend)

Use `fct_current_holdings` as the data source. Example:

```
import streamlit as st
import pandas as pd
from snowflake.snowpark.context import get_active_session

session = get_active_session()
df = session.table("fct_current_holdings").to_pandas()

st.title("💼 Mutual Fund Dashboard")
st.dataframe(df)
```

🔗End-to-End Flow:

1. Load raw data into Snowflake
2. Build DBT staging + transformation models
3. Apply tests, docs, snapshots
4. Deploy with `dbt run`
5. Visualize with Streamlit or BI tool

Let me know if you want this exported as PDF or continued with automation/test pipeline setup.