

SEF - PREWORK SOLUTION DOCUMENT

(Performed by Akshay Ghodke)

- GitHub Repository url:
<https://github.com/akshayghodke099/SEF-assignment-solution.git>

Technical Section (Head and Hand)

❖ Steps as per Problem Statement:

- Took **data** (CSV files: users.csv, orders.csv, payments.csv).
- Loaded them into a **local database** (DuckDB).
- Used **dbt (Data Build Tool)** to transform them into **clean, analytics-ready tables**.
- Added **tests** (to check data quality).
- Added **documentation** (to explain models).
- Pushed everything in a **GitHub repo** with a README and reflection.

Detailed Steps Performed

Step 1: Environment Setup

- **Commands:**

```
python -m venv sef-dbt-env  
sef-dbt-env\Scripts\activate
```

```
pip install dbt-core dbt-duckdb duckdb  
dbt --version
```

- **Result:** Virtual environment created, dbt + DuckDB installed.

Step 2: Initialized dbt Project

- **Command:**

```
dbt init sef_prework
```

Step 3: Downloaded CSVs and Kept in data folder:

- Placed inside sef_prework/data/

Step 4: Created DuckDB Database & Loaded CSVs:

- **Commands:**

```
import duckdb
conn = duckdb.connect("sef_prework/mydb.duckdb")

conn.execute("CREATE TABLE users AS SELECT * FROM
read_csv_auto('sef_prework /data/users.csv');")
conn.execute("CREATE TABLE orders AS SELECT * FROM
read_csv_auto('sef_prework/data/orders.csv');")

conn.execute("CREATE TABLE payments AS SELECT * FROM
read_csv_auto('sef_prework/data/payments.csv');")

conn.close()
```

- **Result:**
 - File mydb.duckdb created inside sef_prework/

Step 5: Configured dbt Profiles:

- **File:** profiles.yml in home directory (~/.dbt/ or C:\Users\admin\.dbt\).
- **Content:**

```
sef_prework:
  target: dev
  outputs:
    dev:
      type: duckdb
      path: c:/users/admin/desktop/ag/assignment
solution/sef_prework/mydb.duckdb
```

- **Result:** dbt connected to DuckDB correctly.

Step 6: Created Models:

- **Staging Models**
- **Intermediate Model**
- **Final Model**

Folder structure:

```
sef_prework/models/  
  staging/  
    ─ stg_users.sql  
    ─ stg_orders.sql  
    └─ stg_payments.sql  
  intermediate/  
    └─ int_orders.sql  
  final/  
    └─ fct_daily_revenue.sql
```

Step 7: Added Tests:

- **File:** schema.yml
- **Content:**

```
models:  
  - name: stg_users  
    columns:  
      - name: user_id  
      tests:  
        - not_null  
        - unique  
  - name: stg_orders  
    columns:  
      - name: order_id  
      tests:  
        - not_null  
        - unique  
  - name: stg_payments  
    columns:  
      - name: payment_id  
      tests:  
        - not_null  
        - unique
```

Step 8: Final Model & Documentation:

- **Added descriptions in schema.yml:**

```
models:  
  - name: fct_daily_revenue  
    description: "Aggregated daily revenue from  
orders with completed payments"
```

- **Commands:**

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

- **Result:**

Serving docs at <http://localhost:8080>

Documentation site opened in browser.

Step 9: Run & test Validated:

- **Command:**

```
dbt run
dbt test
```

- **Result:** Completed successfully
- **Tests output:**

PASS=3 WARN=0 ERROR=0 SKIP=0 TOTAL=3

Step 10: README File

Included:

- Setup instructions (Python, dbt, DuckDB, CSVs)
- Transformations applied (raw → staging → intermediate → final)
- Assumptions (Only completed payments count as revenue)
- Challenges (profiles.yml setup, column mismatches)
- Improvements (more tests, richer models)
- Time spent

Reflection section (Heart & Soul)

Step 11: Reflection:

1. **What was the first concept or tool you had to understand before this assignment became “doable”? How confident do you feel about it now, and what would you still want to explore?**

The first concept I needed to grasp was how dbt connects to DuckDB through the `profiles.yml` file. At first, this felt confusing because dbt would not run until the connection was set up correctly. Once I understood that `profiles.yml` is the bridge between dbt and the database, the assignment became doable. I now feel confident about configuring profiles and running models, but I would still like to explore advanced dbt features such as incremental models and snapshots.

2. How did you build the understanding or skills needed to begin the assignment, and what supported you in that process?

I built my understanding by reading dbt documentation, carefully examining error messages, and experimenting with small changes until things worked. Each time I hit an error—like mismatched column names or semicolons in SQL - I slowed down, checked the docs, and retried. What supported me most was a combination of official documentation, online examples, and persistence in debugging. Seeing the models finally run successfully gave me confidence that I was learning the right way.

3. What did you learn about yourself as a learner during this whole process?

This process taught me that I am patient and persistent when faced with technical challenges. Instead of getting discouraged by repeated errors, I treated them as clues to what I needed to fix. I also realized that I learn best by doing hands-on trial and error helped me more than just reading instructions. Spending over seven hours on this assignment showed me that I can stay focused, solve problem step by step, and grow more confident as I build new skills.

Step 12: Submission

- **Pushed to GitHub:**
 - Included sef_prework/ folder with models/, schema.yml, README.md, reflection
 - Excluded sef-dbt-env/ (environment folder)
- GitHub Repository url:
<https://github.com/akshayghodke099/SEF-assignment-solution.git>