Oklahoma Tax Compliance Data Quality Dashboard

Prepared by: Will Thompson

Contact: Will.j.Thompson@outlook.com | (405) 802-5252

GitHub Repo: https://github.com/Willthompson99/oklahoma-tax-data-quality-dashboard

Executive Summary

This project simulates how the Oklahoma Tax Commission can detect filing anomalies, monitor data quality, and enhance audit readiness using open-source tools. It demonstrates a full-stack analytics pipeline from raw CSV data to interactive visualization in Power BI, powered by SQL logic and Python-based ETL.

It aligns directly with the OTC's mission of transparency, fairness, and compliance, and illustrates the capabilities expected in the Innovation Division.

Project Goals

- Identify and flag erroneous or suspicious tax return data

- Create a reproducible, Git-tracked solution with Python and SQL

- Visualize findings with a clean, interactive dashboard for decision-makers

- Propose process improvements rooted in data evidence

Technical Stack

- ETL: Python (`pandas`, `sqlite3`) used to clean and load data

- Query Engine: SQL used to apply rules for error detection

- Database: SQLite file generated from tax_returns.csv

- Dashboard: Built in Power BI with dynamic filters, KPIs, and visual breakdowns

- Version Control: GitHub-hosted project with documentation and dependencies listed

Python ETL File (create_database.py)

import pandas as pd

```
import sqlite3

# Load CSV file
df = pd.read_csv("tax_returns.csv")

# Connect to SQLite DB (it will create the file if it doesn't exist)
conn = sqlite3.connect("tax_returns.db")

# Insert data into a table named 'tax_returns'
df.to_sql("tax_returns", conn, if_exists="replace", index=False)

# Commit and close connection
conn.commit()
conn.close()
```

SQL Profiling Logic (data_quality_queries.sql)

```
-- 1. Returns with negative refund amounts
SELECT * FROM tax_returns WHERE Refund_Amount < 0;

-- 2. Returns where Payment_Date is before Filing_Date
SELECT * FROM tax_returns WHERE Payment_Date < Filing_Date;

-- 3. Returns with missing Paid_Amount or Return_Amount
SELECT * FROM tax_returns WHERE Paid_Amount IS NULL OR Return_Amount IS NULL;

-- 4. Summary statistics
SELECT MIN(Return_Amount), MAX(Return_Amount), AVG(Return_Amount), COUNT(*)
FROM tax_returns;

-- 5. Excessive deductions (over 50% of paid amount)
SELECT * FROM tax_returns WHERE Deductions_Claimed > (Paid_Amount * 0.5);
```

Data Quality Rules

Rule ID	Description	SQL Logic
1	Negative Refunds	Refund_Amount < 0
2	Premature Payments	Payment_Date < Filing_Date
3	Missing Amounts	Paid_Amount IS NULL OR Return_Amount IS NULL
4	Excessive Deductions	Deductions_Claimed > Paid_Amount * 0.5

Power BI Dashboard Features

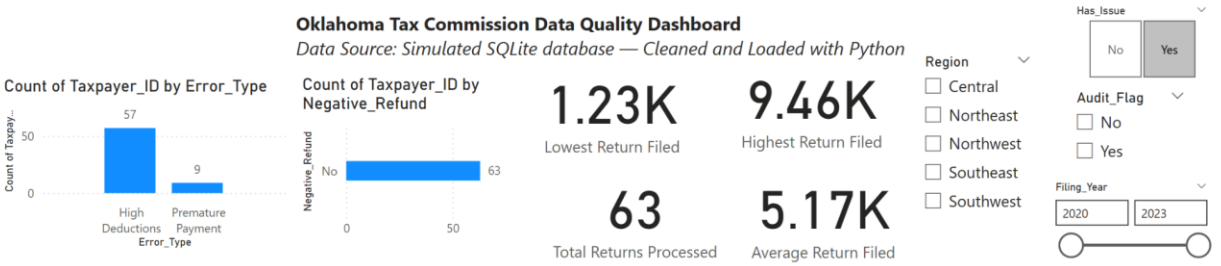
- KPI Cards: Show return amount range and averages

- Filters: Audit flags, filing year, error type, and region

- Error Count Visuals: Bar charts by error type

- Detailed Drillthrough Table: Filterable for audit investigation

Dashboard Snapshot



Count of Taxpayer_ID	Sum of Return_Amount	Sum of Paid_Amount	Sum of Refund_Amount	Region	Audit_Flag	Negative_Refund	Premature_Payment	Missing_Payment_Info	High_Deductions
14	76,890.77	54,299.63	23,523.16	Southwest	No	No	No	No	Yes
11	54,079.92	66,987.20	15,995.46	Central	No	No	No	No	Yes
11	52,634.99	49,647.48	15,686.24	Southeast	No	No	No	No	Yes
8	45,401.86	49,393.19	11,125.54	Northwest	No	No	No	No	Yes
3	14,442.75	14,954.74	7,750.61	Northeast	No	No	No	No	Yes
3	16,767.50	16,974.63	5,663.25	Northwest	No	No	Yes	No	No
3	15,913.28	6,152.07	5,631.75	Southeast	Yes	No	No	No	Yes
2	8,618.73	9,996.37	3,048.51	Southwest	No	No	Yes	No	No
1	6,051.49	6,697.91	1,080.57	Central	Yes	No	No	No	Yes
1	3,927.97	7,553.95	291.53	Northeast	No	No	Yes	No	No
1	2,078.79	6,192.13	43.18	Northeast	No	No	Yes	No	Yes
1	3,563.56	1,139.11	1,195.51	Northeast	Yes	No	No	No	Yes
1	9,455.49	2,304.05	2,553.41	Northwest	Yes	No	No	No	Yes
1	9,367.28	6,867.65	76.05	Southeast	No	No	Yes	No	Yes
1	4,226.19	9,430.57	2,105.90	Southwest	No	No	Yes	No	Yes
1	2,042.82	2,237.69	2,387.38	Southwest	Yes	No	No	No	Yes
63	**325,463.39**	**310,828.37**	**98,158.05**						

All data issues flagged using automated Python preprocessing and validated with SQL queries.

Reproducibility & OTC Role Alignment

✓ Full environment setup in README.md and requirements.txt

✓ Uses open-source stack (Python, SQL, Power BI)

✓ Audit-focused rules for operational improvement

✓ SQL, data modeling, and stakeholder presentation alignment