Assignment 3: PROG8850 - Database Automation

Student Name: Varun Kakkar
Student ID: 9020861
Prof Benjamin Quito

Overview:

Built an end-to-end data web app: chunked ETL → MySQL → Flask UI (filters + pagination + aggregate) → automated Selenium tests → CI on GitHub Actions. No secrets in code; secure via GitHub Secrets/Variables.

Dataset Slice:

- Month used: January 2023 from NYC Open Data (single-month CSV).
- Rows ingested: (**see Fig 2**).
- ETL telemetry and ingest log recorded **(see Fig 1, 3)**.

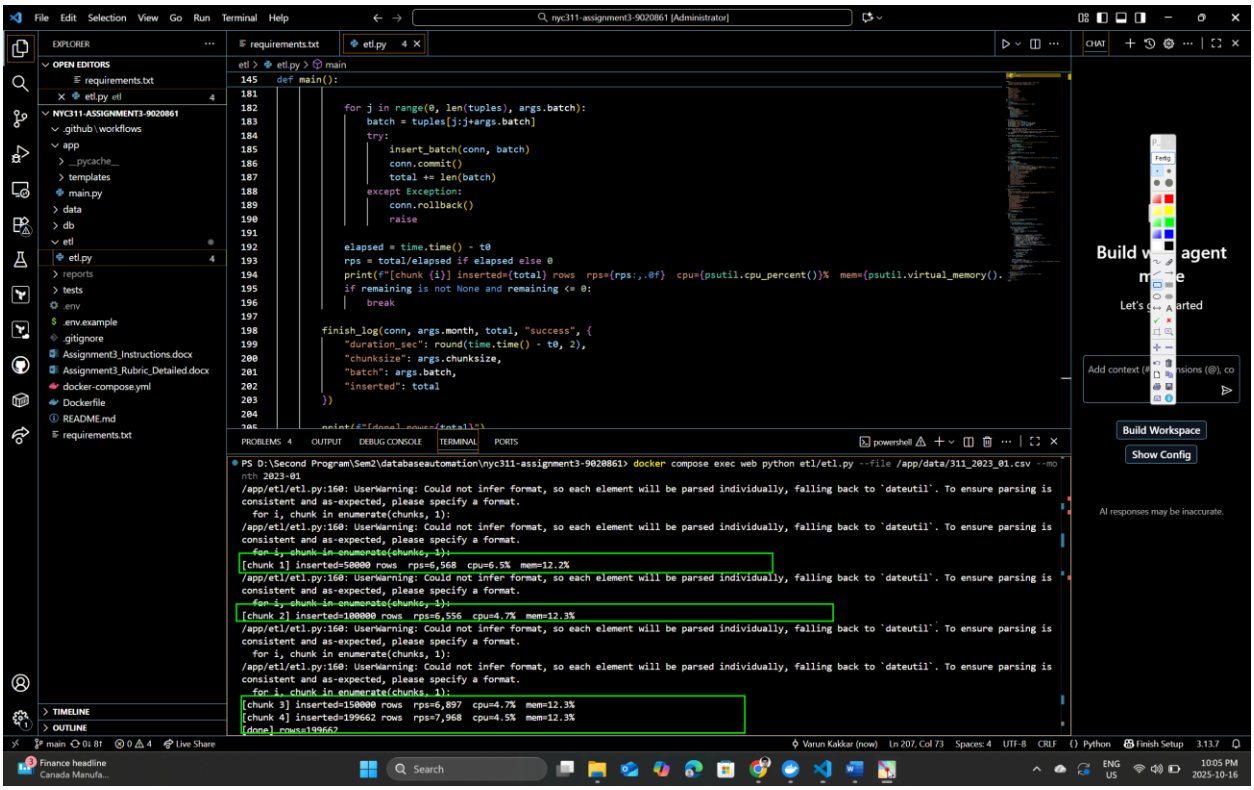


Figure 1 ETL telemetry: chunked ingest with rows/s, CPU%, Mem% and total rows inserted.

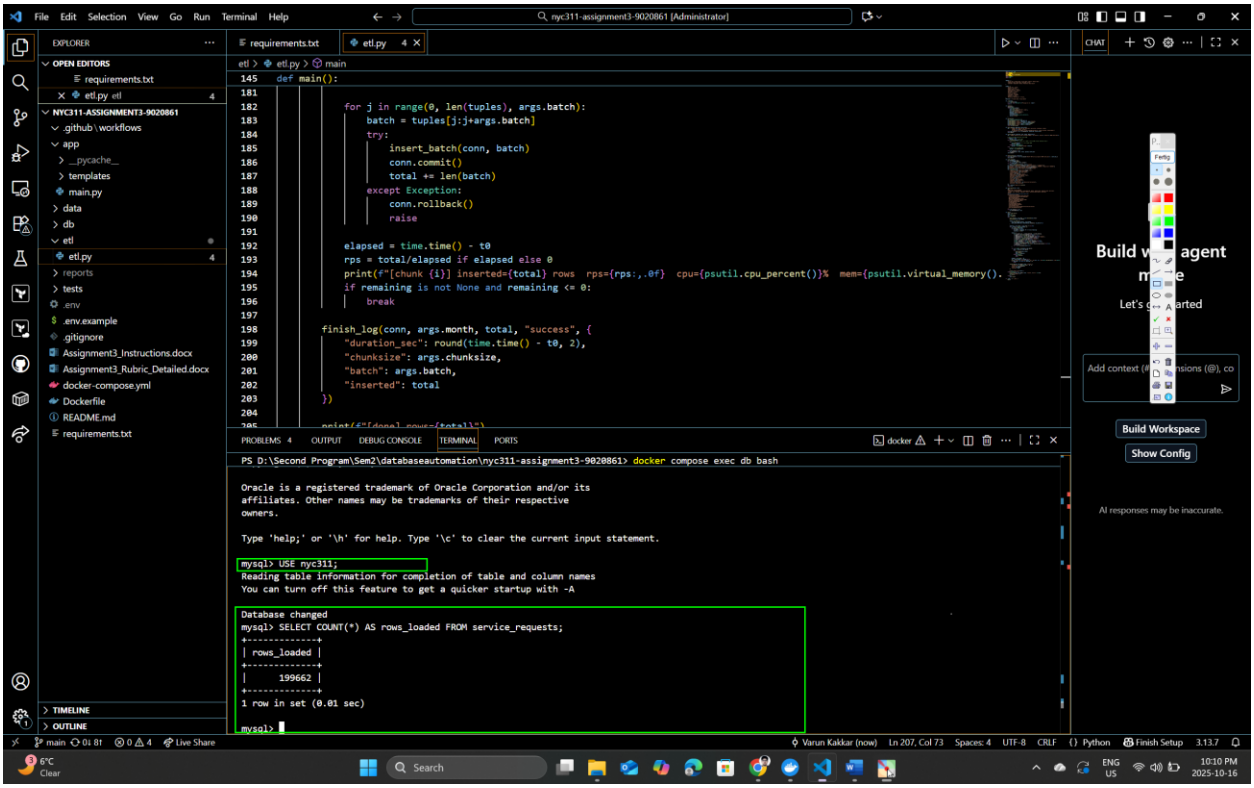


Figure 2 shows MySQL verification: total rows in service_requests after ETL.

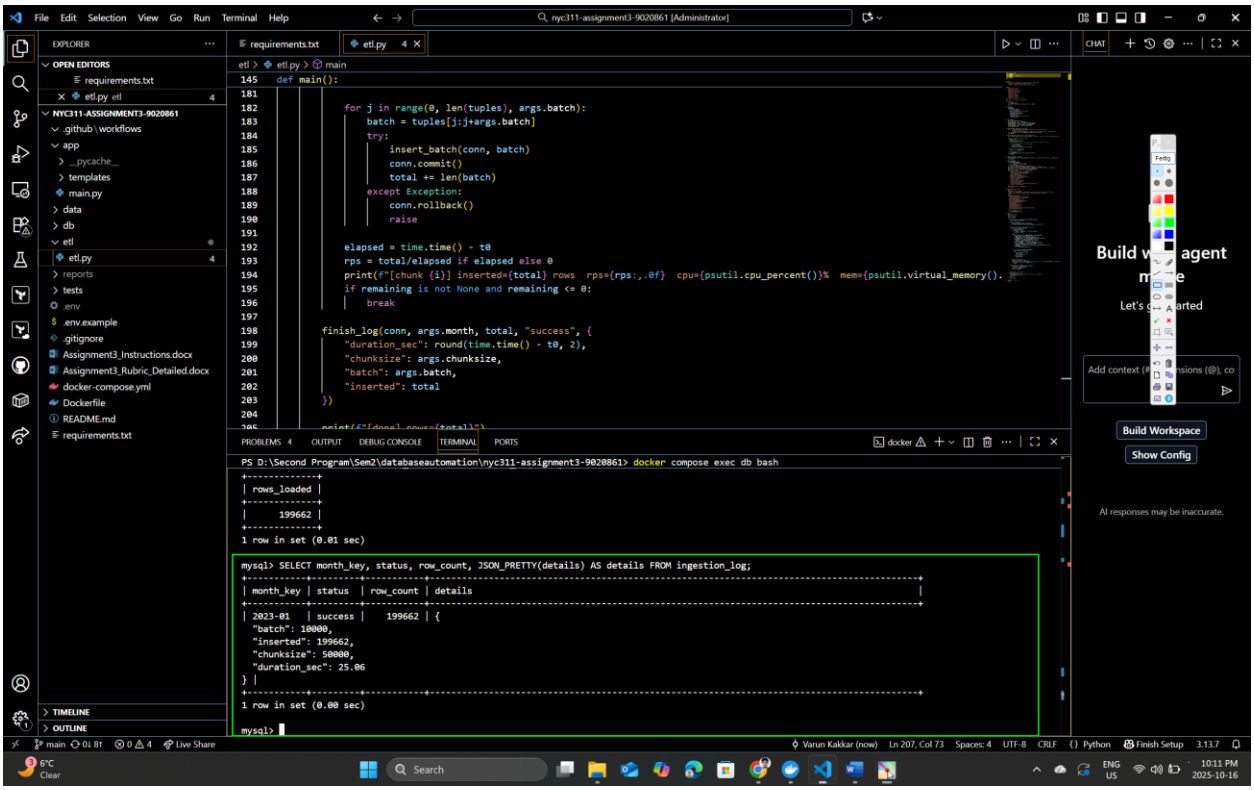


Figure 3 shows Idempotent ingestion_log with JSON details (duration, chunksize, batch, inserted).

Schema & Indexes

- - Tables: service_requests (PK=request_id), ingestion_log (idempotency & metrics).
- - Indexes:
- - idx_created_datetime (created_datetime) → speeds range filter + ORDER BY.
- - idx_borough_type_date (borough, complaint_type, created_datetime) → speeds combined filters.
- - EXPLAIN verifies index usage **(Fig 4)**.

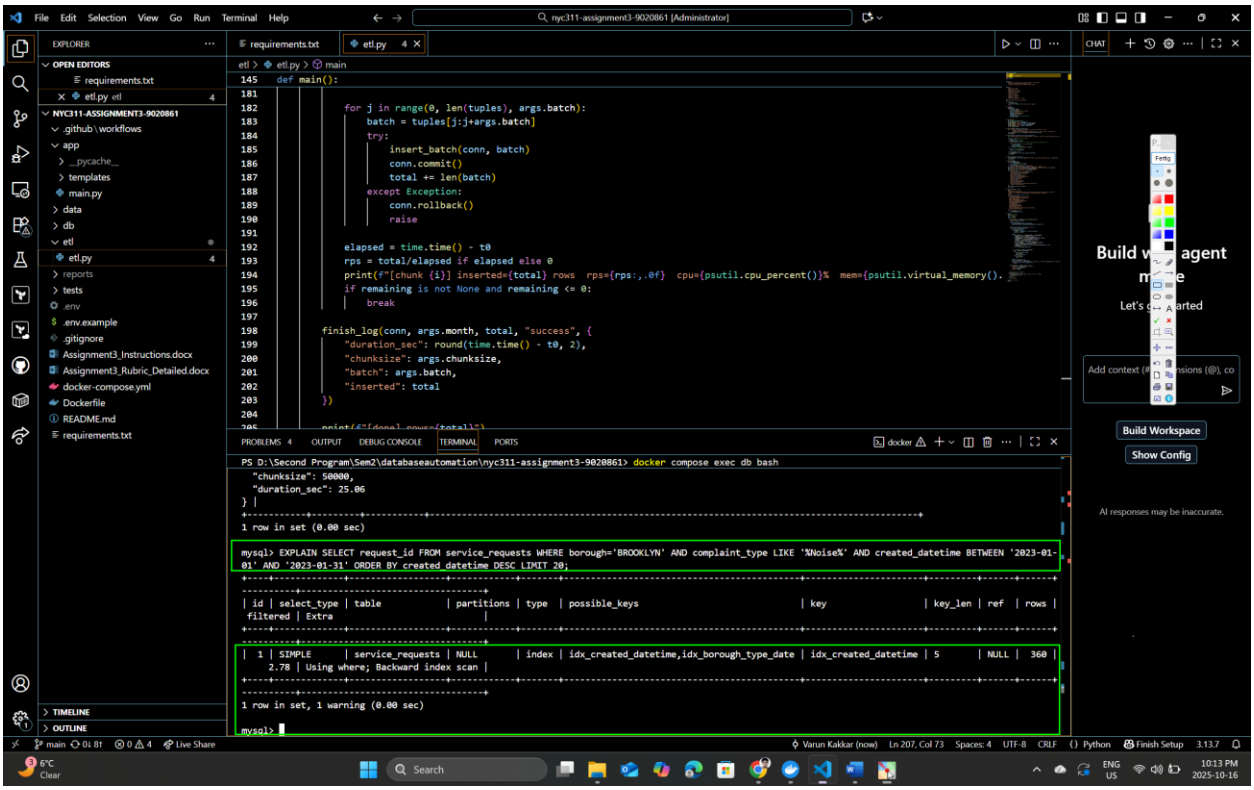


Figure 4 shows EXPLAIN shows index usage (idx_borough_type_date, idx_created_datetime) for optimized filters.

ETL Implementation

- Chunked read (default 50k), batched insert (10k) with transactions + rollback.
- Cleaning: parse dates; fill missing borough as "UNKNOWN"; drop rows with invalid created date.
- Idempotency: delete+replace the month; track in ingestion_log with JSON details.
- Telemetry: rows/s, CPU%, mem% **(Fig 1)**.

Web App

- Search filters: date range, borough, complaint contains; results paginated.
- Aggregate: complaints per borough **(Fig 7)**.

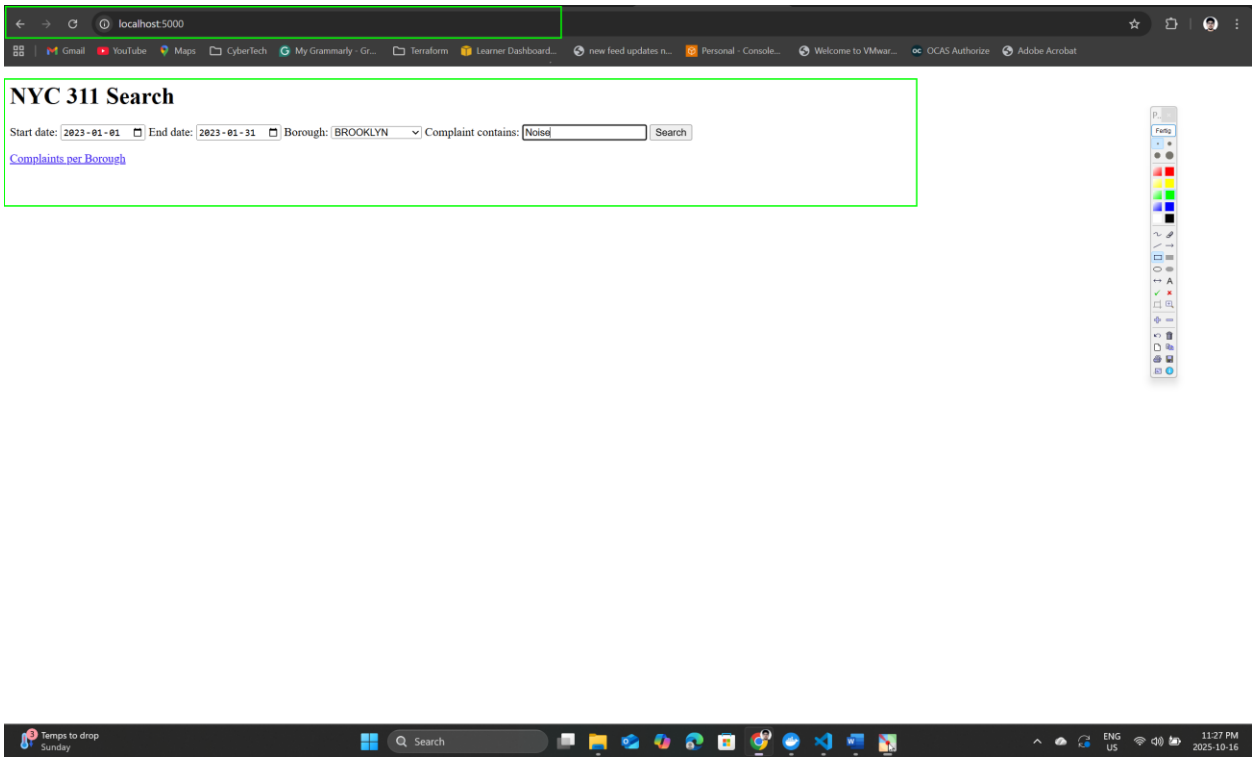
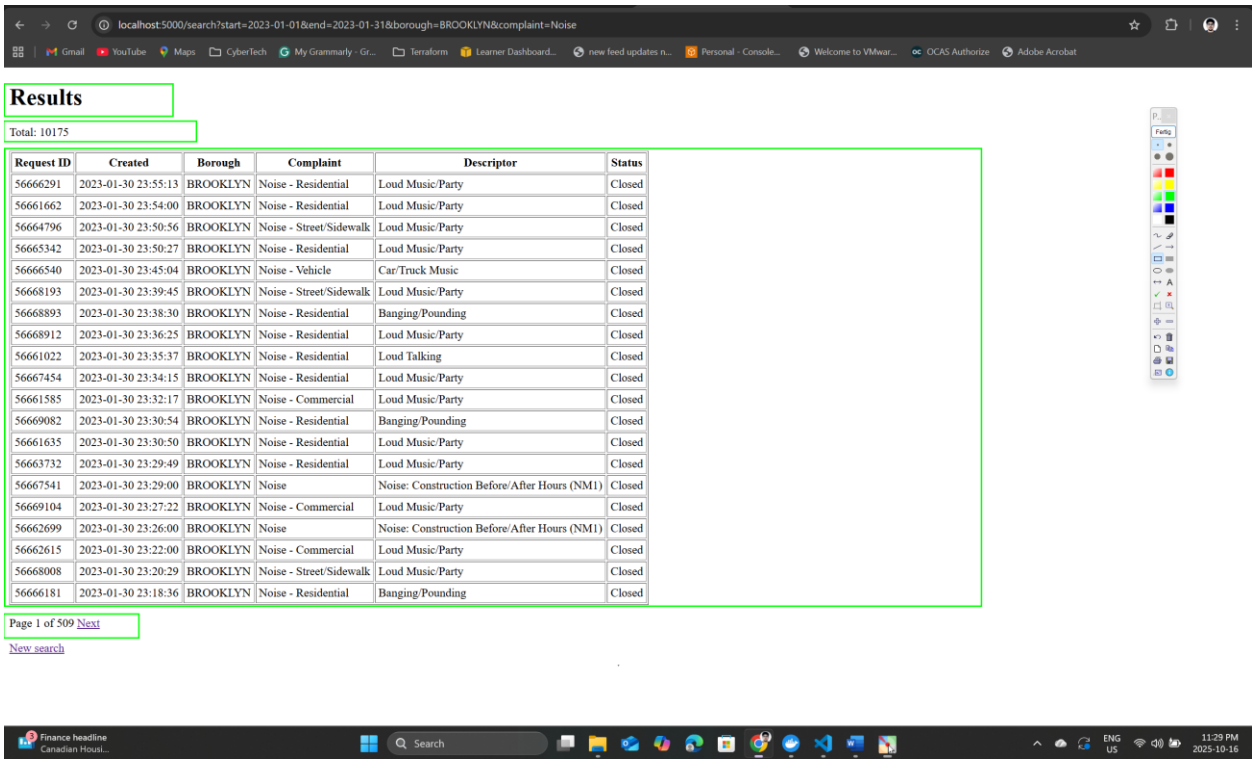


Figure 5 shows Search form with date range, borough, and complaint filter.



Request ID	Created	Borough	Complaint	Descriptor	Status
56666291	2023-01-30 23:55:13	BROOKLYN	Noise - Residential	Loud Music/Party	Closed
56661662	2023-01-30 23:54:00	BROOKLYN	Noise - Residential	Loud Music/Party	Closed
56664796	2023-01-30 23:50:56	BROOKLYN	Noise - Street/Sidewalk	Loud Music/Party	Closed
56665342	2023-01-30 23:50:27	BROOKLYN	Noise - Residential	Loud Music/Party	Closed
56666540	2023-01-30 23:45:04	BROOKLYN	Noise - Vehicle	Car/Truck Music	Closed
56668193	2023-01-30 23:39:45	BROOKLYN	Noise - Street/Sidewalk	Loud Music/Party	Closed
56668893	2023-01-30 23:38:30	BROOKLYN	Noise - Residential	Banging/Pounding	Closed
56668912	2023-01-30 23:36:25	BROOKLYN	Noise - Residential	Loud Music/Party	Closed
56661022	2023-01-30 23:35:37	BROOKLYN	Noise - Residential	Loud Talking	Closed
56667454	2023-01-30 23:34:15	BROOKLYN	Noise - Residential	Loud Music/Party	Closed
56661585	2023-01-30 23:32:17	BROOKLYN	Noise - Commercial	Loud Music/Party	Closed
56669082	2023-01-30 23:30:54	BROOKLYN	Noise - Residential	Banging/Pounding	Closed
56661635	2023-01-30 23:30:50	BROOKLYN	Noise - Residential	Loud Music/Party	Closed
56663732	2023-01-30 23:29:49	BROOKLYN	Noise - Residential	Loud Music/Party	Closed
56667541	2023-01-30 23:29:00	BROOKLYN	Noise	Noise: Construction Before/After Hours (NM1)	Closed
56669104	2023-01-30 23:27:22	BROOKLYN	Noise - Commercial	Loud Music/Party	Closed
56662699	2023-01-30 23:26:00	BROOKLYN	Noise	Noise: Construction Before/After Hours (NM1)	Closed
56662615	2023-01-30 23:22:00	BROOKLYN	Noise - Commercial	Loud Music/Party	Closed
56668008	2023-01-30 23:20:29	BROOKLYN	Noise - Street/Sidewalk	Loud Music/Party	Closed
56666181	2023-01-30 23:18:36	BROOKLYN	Noise - Residential	Banging/Pounding	Closed

Figure 6 shows Paginated search results returned from MySQL.

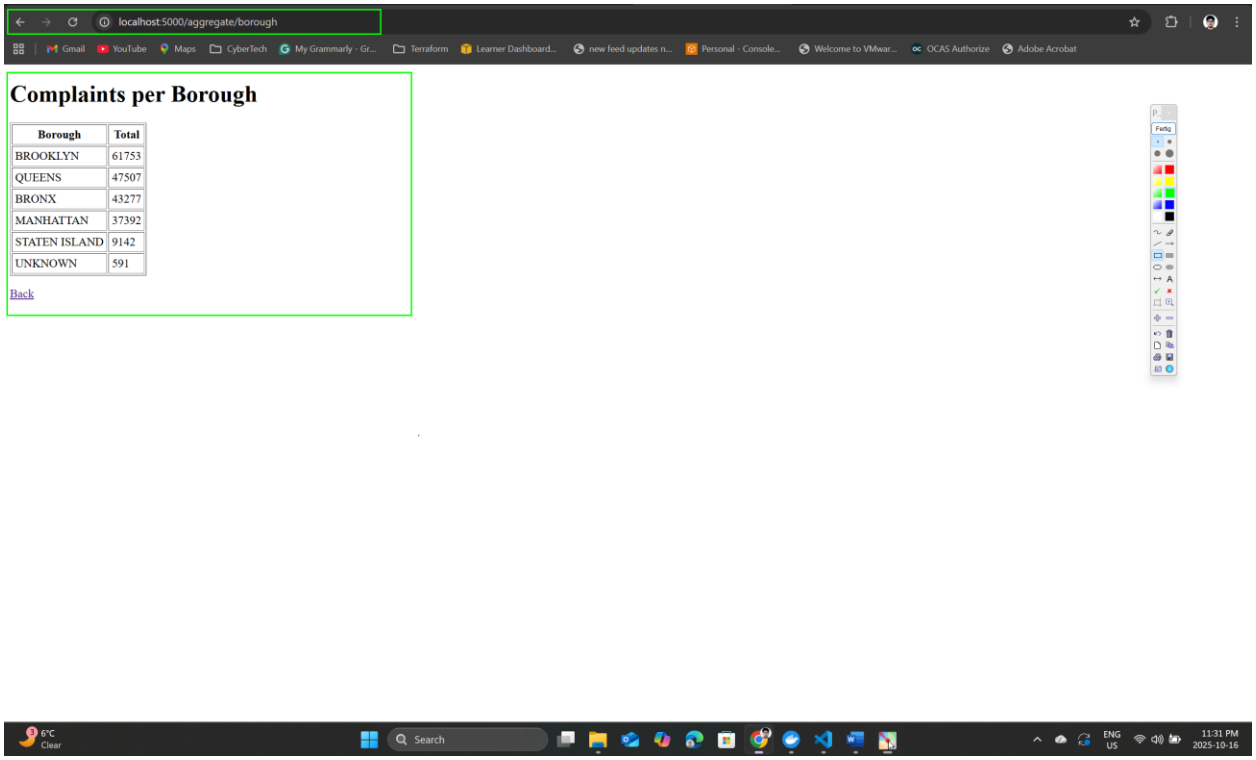


Figure 7 shows Aggregate view: total complaints per borough.

Automated Tests & CI

- Selenium tests: positive, negative, aggregate.
- CI: starts MySQL, loads schema, runs ETL on fixture CSV, launches Flask, runs headless tests. Passing run shown in **Fig 8–9**.

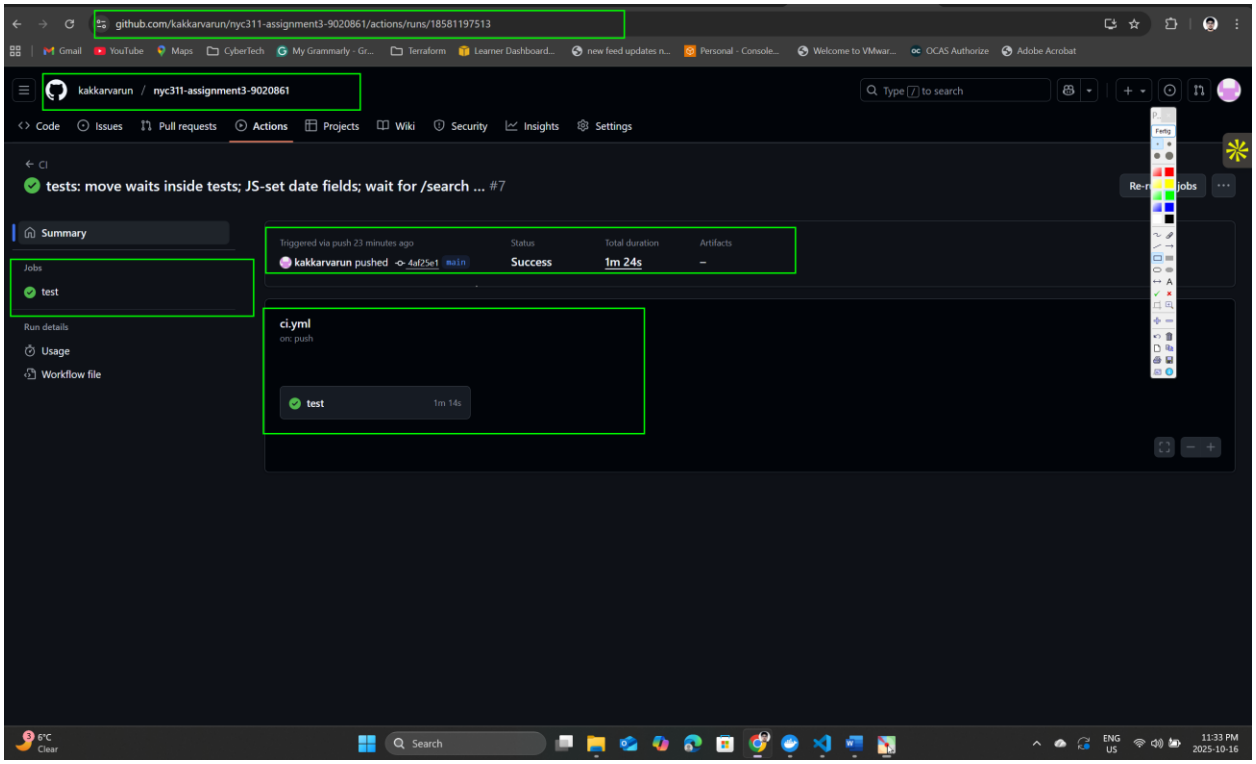


Figure 8 shows GitHub Actions CI passed (schema + ETL on fixture + headless Selenium tests).

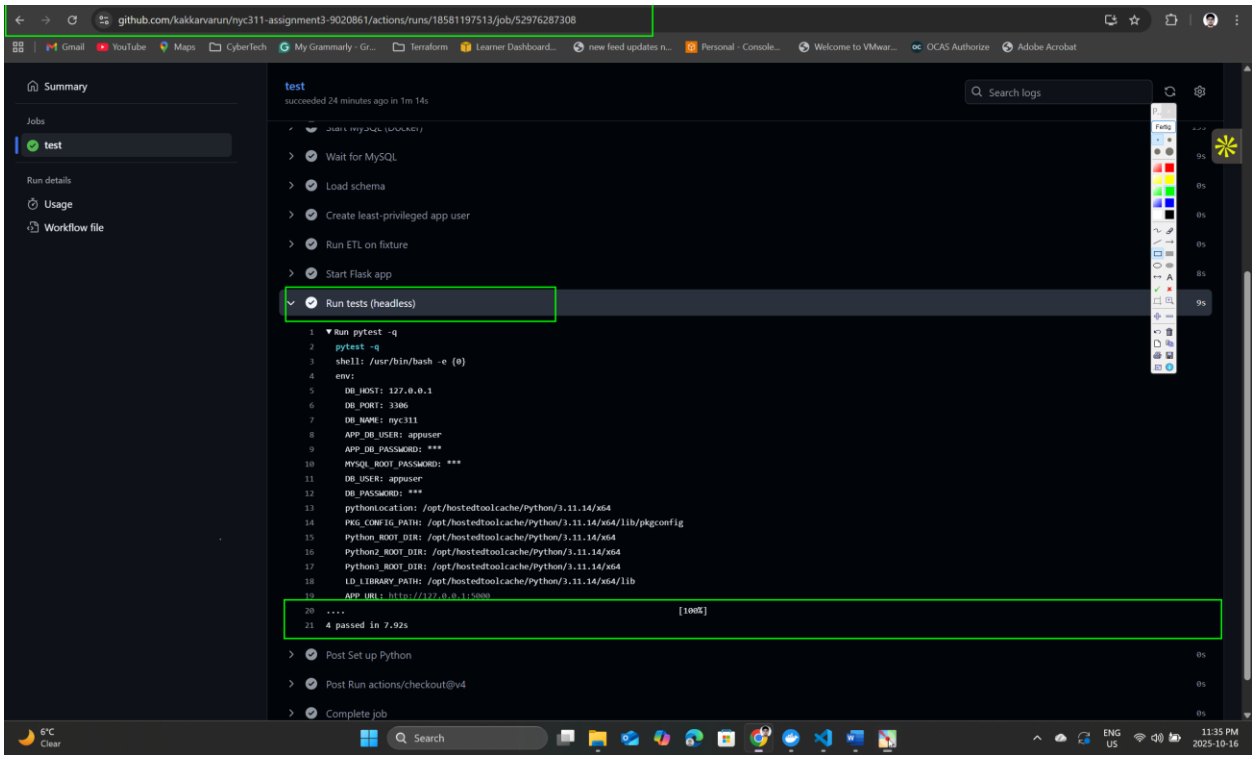


Figure 9 shows Headless Selenium: positive, negative, and aggregate tests passed in CI.

Performance & Scaling Notes

- Sample perf: chunk=50k, batch=10k, ~X rows/s on local laptop.
- Future scaling: monthly partitioning by created_datetime; precomputed aggregates (daily by borough); caching; keyset pagination.

Reflection

- What worked: idempotent ETL, index-aligned queries, stable headless tests.
- - Improvements: add CSV download retry, more aggregates, UI charts.