Assignment 1 LLM Transciprt

Muhammad Suffyan Khan

LLM Transcript and Citation

Tool Used: ChatGPT (OpenAI, GPT-5)
Purpose: Debugging and improving reproducibility + transformation workflow in Quarto/R
Course: DATA 607 – Assignment 1
Date Accessed: February 1, 2026

Citation:
OpenAI. (2026). *ChatGPT (GPT-5)* [Large language model]. https://chat.openai.com. Accessed February 1, 2026.

I used ChatGPT to help me debug and improve a reproducible workflow for loading and transforming the NYC 311 dataset in R using Quarto. I worked iteratively by running the code locally, checking outputs, and refining the transformations. The most useful assistance was around (1) standardizing inconsistent column names using janitor::make_clean_names(), (2) parsing datetime fields robustly using lubridate::parse_date_time(), (3) detecting and correcting invalid negative response times caused by inconsistent timestamps in the source dataset, and (4) implementing creative extensions through exploratory summaries and feature engineering (Top 10 complaint types, a rule-based issue_priority variable, and borough-level comparisons).

[1] Conversation with ChatGPT (Column name standardization using janitor)

I'm getting an error in `select()` that the column complaint_type doesn't exist. What should I do?

Assistant (ChatGPT): This error indicates the dataset's column names don't match what you're referencing (case differences, spaces, punctuation). A robust solution is to standardize column names once using the `janitor` package so that all names become clean `snake_case`. This avoids fragile backticks and prevents future selection/rename issues.

Suggested code:

```r
library(janitor)

colnames(nyc_311) <- janitor::make_clean_names(colnames(nyc_311))
colnames(nyc_311)
```

After cleaning names, column selection becomes reliable:

```r
nyc_311_subset <- nyc_311 %>%
  select(
    complaint_type,
    descriptor,
    borough,
    agency,
    status,
    created_date,
    closed_date
  )
```

---

[2] Conversation with ChatGPT (Datetime parsing using mutate + lubridate)

How should I convert created/closed date columns into datetime format?

Assistant (ChatGPT): Because NYC 311 datasets often contain timestamps in multiple formats, using `as.POSIXct()` with a single format can fail or create missing values unexpectedly. A more robust approach is using `lubridate::parse_date_time()` which can parse multiple possible formats.

Suggested code:

```r
library(lubridate)

nyc_311_clean <- nyc_311_clean %>%
  mutate(
    created_date = parse_date_time(created_date,
                                   orders = c("mdy HMS p", "mdy HMS", "ymd HMS", "ymd HMSz")),
    closed_date  = parse_date_time(closed_date,
                                   orders = c("mdy HMS p", "mdy HMS", "ymd HMS", "ymd HMSz"))
  )
```

This ensures the datetime conversion works even when timestamp formats vary in the source file.

---

[3] Conversation with ChatGPT (Negative response time correction)

Why is response time coming out negative? What should I do — just add a note or correct it?

Assistant (ChatGPT): Negative response time happens when the dataset contains records where `closed_date` is earlier than `created_date`. This is a common real-world data quality issue in large administrative datasets. You should **correct it in code** and also **document it briefly**, because otherwise summary statistics become misleading.

Recommended approach:

- Calculate response time only when `closed_date` exists
- Treat negative values as invalid → set to `NA`
- Optionally cap extreme outliers (example > 1 year)

Corrected code:

```
nyc_311_clean <- nyc_311_clean %>%
  mutate(
    response_time_hours = case_when(
      is.na(closed_date) ~ NA_real_,
      TRUE ~ as.numeric(difftime(closed_date, created_date, units = "hours"))
    ),
    response_time_hours = ifelse(response_time_hours < 0, NA_real_, response_time_hours),
    response_time_hours = ifelse(response_time_hours > 24*365, NA_real_, response_time_hours)
  )
```

This keeps response time analysis valid and avoids incorrect conclusions from corrupted timestamps.

---

[4] Conversation with ChatGPT (Creative extension implementation)

I want to do some creative work with the data. Like exploratory analysis, some charts, tables. Give three ideas and respective code as well

Assistant (ChatGPT): Recommended adding simple creative extensions to strengthen the report beyond the minimum requirements while remaining fully reproducible. Suggested focusing on exploratory summaries and feature engineering using the cleaned dataset, including: identifying the most common complaint types, creating a rule-based priority feature, and comparing service demand across boroughs.

Top complaint categories (frequency table + bar plot)

Simple feature engineering using complaint type keywords

Borough comparison using grouped summaries and visualization

Suggested code (Top 10 complaint types):

```
complaint_summary <- nyc_311_clean %>%
  count(complaint_type, sort = TRUE) %>%
  mutate(
    pct_label = scales::percent(n / sum(n), accuracy = 0.1)
  ) %>%
  slice_head(n = 10)

complaint_summary

ggplot(complaint_summary, aes(x = reorder(complaint_type, n), y = n)) +
  geom_col() +
  coord_flip() +
  geom_text(aes(label = pct_label), hjust = -0.1) +
  labs(
    title = "Top 10 NYC 311 Complaint Types",
    x = "Complaint Type",
    y = "Count"
  ) +
  theme_minimal()
```

Suggested code (Feature engineering: create issue_priority):

```
nyc_311_clean <- nyc_311_clean %>%
  mutate(
    issue_priority = case_when(
```

```r
      str_detect(tolower(complaint_type), "heat|hot water|electric|gas") ~ "High",
      str_detect(tolower(complaint_type), "noise") ~ "Medium",
      TRUE ~ "Low/Other"
    )
  )

nyc_311_clean %>%
  count(issue_priority, sort = TRUE)

nyc_311_clean %>%
  count(issue_priority) %>%
  ggplot(aes(x = issue_priority, y = n)) +
  geom_col() +
  labs(
    title = "Engineered Feature: Complaint Priority (Simple Rule-Based)",
    x = "Priority",
    y = "Count"
  ) +
  theme_minimal()
```

Suggested code (Borough comparison: counts and response time summary):

```r
borough_summary <- nyc_311_clean %>%
  group_by(borough) %>%
  summarise(
    total_requests = n(),
    closed_requests = sum(!is.na(response_time_hours)),
    median_response_hours = median(response_time_hours, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(desc(total_requests))

borough_summary

ggplot(borough_summary, aes(x = reorder(borough, total_requests), y = total_requests)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Total NYC 311 Requests by Borough",
    x = "Borough",
    y = "Total Requests"
```

```
  ) +
  theme_minimal()
```

ChatGPT is AI and can make mistakes.