

Assignment #1 LLM Transcript – Denise Atherley

I started by asking Google Gemini for a recent dataset on NYC monuments that had data augmented a bit, to narrow down the choice options that my original dataset included. Turns out the dataset I originally found is the most recent dataset and was last updated in January 2026, but a second dataset was suggested that is an inventory that covered City-owned memorials and markers installed on any City-owned property, not just park monuments. This dataset was more compelling to me.

[1] Conversation with Google Gemini:

Transcript link - <https://share.google/aimode/l3owCof0LgPC7oLwx>

Where can I find a publicly available dataset of NYC monuments, including materials used, location and year it was added to the park?

Thought for 15s

You can find this specific data through the **NYC Open Data** portal, which hosts the official inventory maintained by the NYC Department of Parks & Recreation.

- [**NYC Parks Monuments Dataset**](#): This is the primary table containing over 800 entries. It includes key fields such as:
 - **Location**: Precise coordinates, borough, and park name.
 - **Materials**: Descriptions of the mediums used (e.g., bronze, granite).
 - **Year Added**: Dedicated or installed dates.
- [**Public Design Commission Inventory**](#): For broader public art beyond just park monuments, this inventory covers City-owned memorials and markers installed on any City-owned property.
- [**NYC Public Art Map**](#): A user-friendly interactive tool that allows you to browse these monuments by borough and link directly to detailed historical profiles for each piece.

Would you like help **filtering** the dataset by a specific borough or **exporting** it into a particular format like CSV or JSON?

[2] Conversation with ChatGPT:

Transcript link - <https://chatgpt.com/share/69801844-6e2c-8010-a69f-2d33a46e36cc>

Can you take the code in the last code-block of the attached .qmd file and create a new code block, adding the same functionality for subsetting and filtering using modern tidyverse syntax?

I included a copy of the quarto document I was editing using base R to render results and requested additional support from ChatGPT

9:13PM

Yep — here's a clean **tidyverse-style replacement** you can drop in after your existing base R subset. It does the same two things as your last block:

- selects the same variables **by name** (instead of numeric positions)
- keeps the **first 5 rows**
- pipes nicely into gt()

```
```{r}
```

```
#| echo: true
```

```
library(dplyr)
```

```
df1_tidy <- df |>
```

```
 select(
```

```
 `Date Created` ,
```

```
 `Artwork Type 1` ,
```

```
 `Artwork Type 2` ,
```

```
 Material,
```

```
 Borough
```

```
) |>
```

```
 slice_head(n = 5)
```

```
df1_tidy |>
 gt()

Why this is “modern tidyverse”
- `select()` uses **explicit column names** → safer and more readable than `c(22, 26, ...)`
- `slice_head(n = 5)` clearly communicates intent (first 5 rows)
- Works seamlessly with pipes and downstream verbs like `filter()`, `arrange()`, etc.
```

If you want, I can also:

- add a \*\*filtered version\*\* (e.g., only Manhattan monuments)
- show a \*\*side-by-side comparison\*\* of base R vs tidyverse
- refactor earlier chunks for consistency across the whole ` .qmd`

***Going forward always prefer tidyverse syntax; use ggplot2 when appropriate. Next create an exploratory data analysis on the Artwork Type1 column, showing a table of values with counts and percentages for the full dataset referenced.***

Love it — tidyverse all the way from here on out 

Here's a clean EDA chunk that summarizes Artwork Type 1 for the *full dataset*, with counts and percentages, using modern tidyverse + gt.

You can drop this straight into your .qmd.

```
```{r}
#| echo: true

library(dplyr)
library(gt)
library(scales)
```

```
artwork_type1_summary <- df |>  
  filter(!is.na(`Artwork Type 1`)) |>  
  count(`Artwork Type 1`, name = "count") |>  
  mutate(  
    percent = count / sum(count)  
) |>  
  arrange(desc(count))  
  
artwork_type1_summary |>  
  gt() |>  
  fmt_number(  
    columns = count,  
    decimals = 0  
) |>  
  fmt_percent(  
    columns = percent,  
    decimals = 1  
) |>  
  cols_label(  
    `Artwork Type 1` = "Artwork Type 1",  
    count = "Count",  
    percent = "Percent of Total"  
) |>  
  tab_header(  
    title = "Distribution of Artwork Type 1",  
    subtitle = "NYC Parks Monuments (Full Dataset)"
```

)

What this gives you

- A **frequency table** of all `Artwork Type 1` values
- Percentages that **sum to 100%** (excluding `NA`)
- Sorted from most → least common
- Presentation-ready via `gt()`

Next natural steps (happy to do these next):

- Collapse multi-valued / messy categories into broader groups
- Add a **ggplot2 bar chart** to visually reinforce this table
- Compare distributions **pre/post 1964** like you outlined

Just say the word 

For records that have multiple values in the Artwork Type1 column, please show how to consolidate the values into a main category so that it rolls up to a broader set of values.

Great question — this is exactly the kind of cleanup that makes EDA way more meaningful



I'll show a **clear, tidyverse-first pattern** you can reuse.

The idea:

1. **Split multiple values** into separate rows
2. **Map detailed values → broader “main categories”**
3. **Roll them back up** for clean counts/percentages

Step 1: Separate multiple values

This assumes multiple artwork types are separated by commas (adjust the separator if needed).

```
library(dplyr)
```

```
library(tidyr)
```

```
library(stringr)
```

```
df_artwork_long <- df |>  
  filter(!is.na(`Artwork Type 1`)) |>  
  separate_rows(`Artwork Type 1`, sep = ",") |>  
  mutate(  
    `Artwork Type 1` = str_trim(`Artwork Type 1`)  
)
```

At this point, each artwork type gets its own row — much easier to reason about.

Step 2: Consolidate into broader categories

Here's a **pattern-based rollup** using `case_when()` (very readable, very flexible):

```
df_artwork_rolled <- df_artwork_long |>  
  mutate(  
    artwork_category = case_when(  
      str_detect(`Artwork Type 1`, regex("statue|sculpture", ignore_case = TRUE)) ~  
      "Sculpture / Statue",  
      str_detect(`Artwork Type 1`, regex("monument|memorial", ignore_case = TRUE)) ~  
      "Monument / Memorial",  
      str_detect(`Artwork Type 1`, regex("plaque", ignore_case = TRUE)) ~ "Plaque",  
      str_detect(`Artwork Type 1`, regex("fountain", ignore_case = TRUE)) ~ "Fountain",  
      str_detect(`Artwork Type 1`, regex("mural", ignore_case = TRUE)) ~ "Mural",  
      TRUE ~ "Other"
```

```
)
```

```
)
```

💡 This approach:

- Handles **multiple values gracefully**
 - Is resilient to small text variations
 - Makes category logic explicit and auditable
-

Step 3: Summarize counts + percentages

Now we roll everything up cleanly.

```
artwork_category_summary <- df_artwork_rolled |>  
  count(artwork_category, name = "count") |>  
  mutate(  
    percent = count / sum(count)  
) |>  
  arrange(desc(count))
```

Step 4: Presentation-ready table

```
library(gt)
```

```
artwork_category_summary |>  
  gt() |>  
  fmt_number(count, decimals = 0) |>  
  fmt_percent(percent, decimals = 1) |>  
  cols_label(  
    artwork_category = "Artwork Category",  
    count = "Count",  
    percent = "Percent of Total"
```

```
) |>  
tab_header(  
  title = "Artwork Type 1 Rolled Up into Main Categories",  
  subtitle = "Multiple values consolidated into broader groups"  
)
```

Why this approach works well

- **No information loss** from multi-valued fields
- Categories are **easy to tweak** as you learn more about the data
- Plays perfectly with later ggplot2 visualizations
- Idiomatic tidyverse throughout (future-proof 🌟)

If you want next:

- A **ggplot2 bar chart** of these rolled-up categories
- A version that **keeps original + rolled category side-by-side**
- Or logic to **prioritize one category per record** instead of expanding rows

Just tell me how opinionated you want to get 😊

Google DeepMind. (2025). Gemini 3 Flash [Large language model].

<https://gemini.google.com>. Accessed February 1, 2026

OpenAI. (2025). ChatGPT (Version 5.2) [Large language model].

<https://chat.openai.com>. Accessed February 1, 2026