HW2

Karthik Bobba

2025-02-11

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.2
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```
## Warning: package 'lubridate' was built under R version 4.4.2
```

```
## ── Attaching core tidyverse packages ───────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4      ✓ readr     2.1.5
## ✓ forcats   1.0.0      ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1      ✓ tibble    3.2.1
## ✓ lubridate 1.9.4      ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ─────────────────────────────────────── tidyverse_conflicts() ──
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```r
library(lubridate)
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.4.2
```

```
## 
## Attaching package: 'kableExtra'
## 
## The following object is masked from 'package:dplyr':
## 
##     group_rows
```

```r
#Loading neccessary libraries

inspections <- read_csv("restaurant_inspections.csv") #Loading data set into a df called inspect
ions
```

```
## Rows: 3875 Columns: 12
## ── Column specification ──────────────────────────────────────────────
## Delimiter: ","
## chr  (8): HSISID, DESCRIPTION, TYPE, INSPECTOR, NAME, RESTAURANTOPENDATE, CI...
## dbl  (3): OBJECTID, SCORE, PERMITID
## dttm (1): DATE_
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
inspections <- inspections %>%
  mutate(
    # Convert date columns to proper datetime format
    DATE_ = ymd_hms(DATE_),
    RESTAURANTOPENDATE = ymd_hms(RESTAURANTOPENDATE),

    # Standardize city names for question 3
    CITY = str_to_upper(CITY) %>%
      recode("RTP" = "RESEARCH TRIANGLE PARK",
             "CARY" = "CARY",
             "Apex" = "APEX",
             "RALEIGH" = "RALEIGH",
             "FUQUAY-VARINA" = "FUQUAY VARINA",
             "HOLLY SPRING" = "HOLLY SPRINGS",
             "MORRISVILE" = "MORRISVILLE",
             )
  )
```

1) Histogram

```
ggplot(inspections, aes(SCORE)) +
  geom_histogram(binwidth = 2, fill = "blue", color = "white") +
  labs(title = "Distribution of Inspection Scores",
       x = "Score", y = "Count")
```

Distribution of Inspection Scores
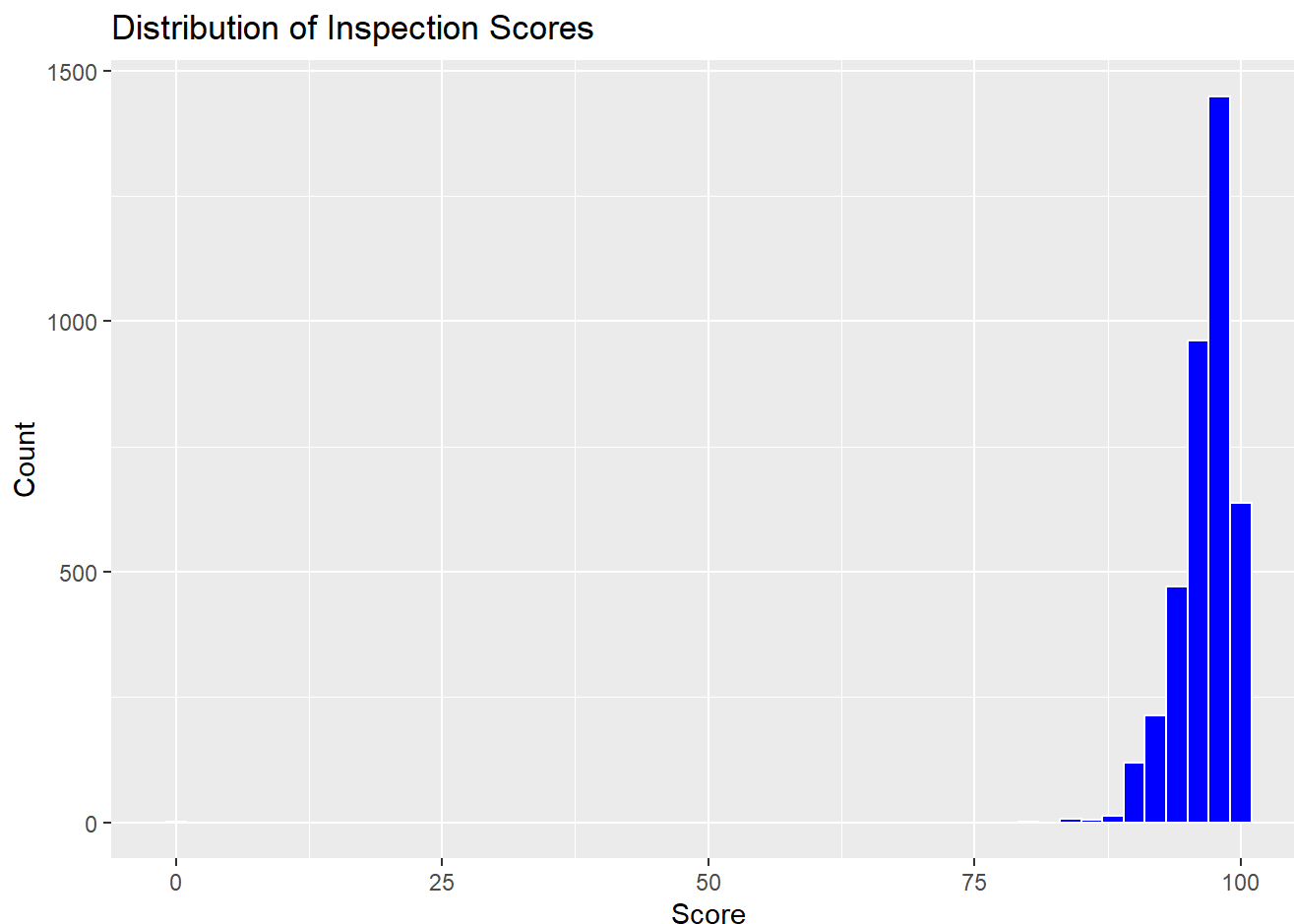

```
#essentially creating a histogram with score as the x axis and count as the y axis, showing the
distribution of scores
```

2) Trend for New vs Old Facility

We don't really see any trend visually

```
inspections %>%
  mutate(restaurant_age = time_length(DATE_ - RESTAURANTOPENDATE, "years")) %>% #Essentially wha
t is happening here is calculating how long it's been since they've opened, using the time_lengt
h function
  ggplot(aes(restaurant_age, SCORE)) +
  geom_point(alpha = 0.5) +        # Add transparency for overlapping points
  labs(title = "Inspection Scores by Restaurant Age",
       x = "Years Since Opening", y = "Score")
```

```
## Warning: Removed 296 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

Inspection Scores by Restaurant Age
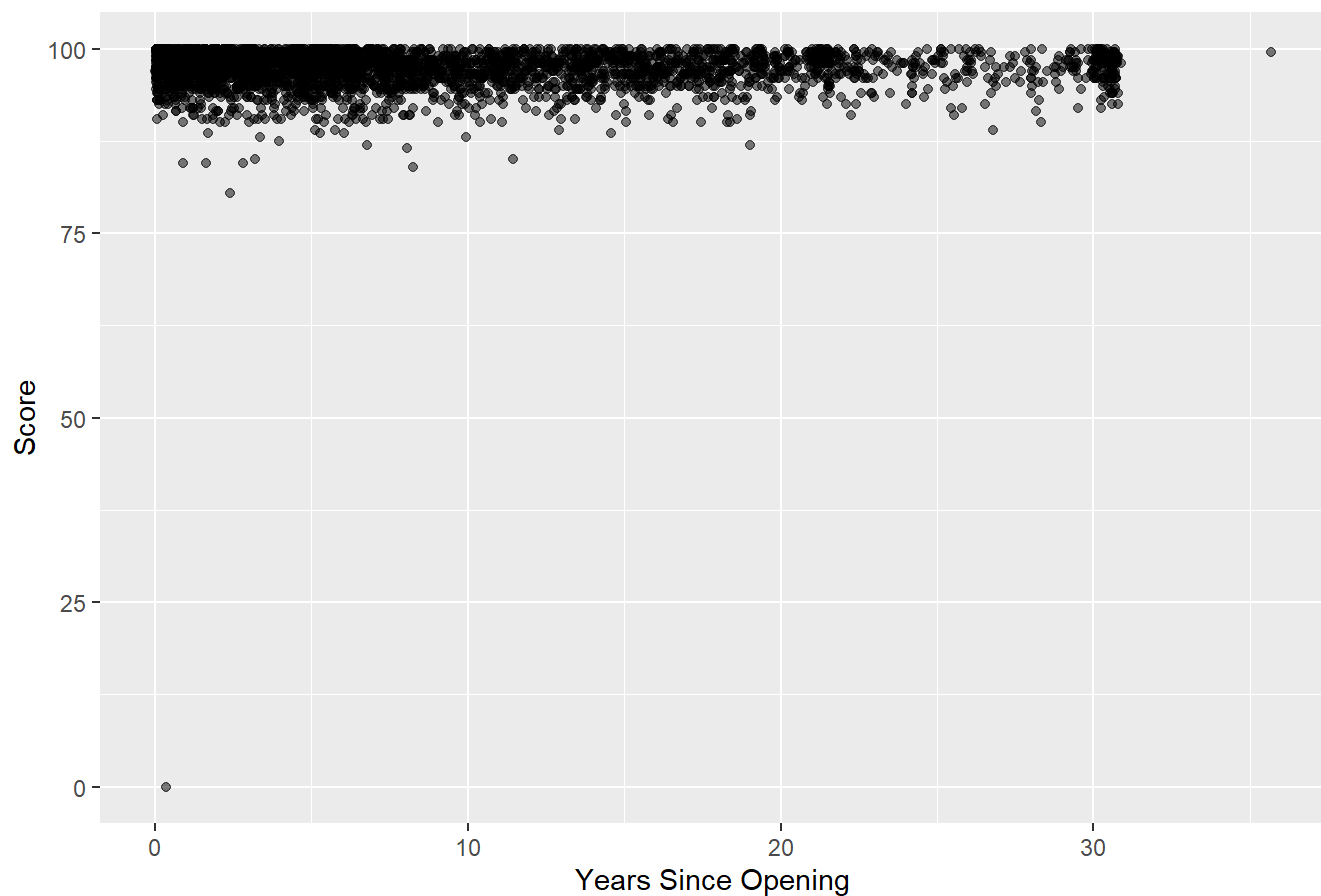

```
#We don't really see any trend visually
```

3) Vary by City?

Average score seems to have very minimal variation between cities

```
city_scores <- inspections %>%
  group_by(CITY) %>%
  summarise(
    avg_score = mean(SCORE, na.rm = TRUE),
    n_inspections = n(),  # Count number of inspections per city
    .groups = "drop"
  )

# Create formatted table with kableExtra
kable(city_scores, caption = "Average Scores by City") %>%
  kable_styling()
```

Average Scores by City

CITY	avg_score	n_inspections
ANGIER	94.50000	1

CITY	avg_score	n_inspections
APEX	97.57568	185
CARY	97.55323	573
CLAYTON	96.12500	4
FUQUAY VARINA	97.34649	114
GARNER	96.33835	133
HOLLY SPRINGS	98.35514	107
KNIGHTDALE	96.15432	81
MORRISVILLE	96.75862	174
NEW HILL	99.75000	2
NORTH CAROLINA	97.00000	1
RALEIGH	97.12375	1895
RESEARCH TRIANGLE PARK	98.75000	2
ROLESVILLE	97.08333	24
WAKE FOREST	97.30867	196
WENDELL	95.48571	35
WILLOW SPRING	96.50000	2
ZEBULON	94.88000	50
NA	94.15034	296

```
#Average score seems to have very minimal variation between cities
```

4) Inspector Ratings

Average score does seem to have some variation depending on the inpsector

```
# Calculate inspector performance metrics
inspector_scores <- inspections %>%
  group_by(INSPECTOR) %>%
  summarise(
    avg_score = mean(SCORE, na.rm = TRUE),
    n_inspections = n(),
    .groups = "drop"
  ) %>%
  arrange(desc(avg_score))  # arranging by highest avg score
#Essentially finding highest avg scores by inspector
kable(head(inspector_scores, 200), caption = "Inspectors by Average Score") %>%
  kable_styling()
```

Inspectors by Average Score

INSPECTOR	avg_score	n_inspections
James Smith	99.00000	1
Greta Welch	98.50000	2
Kaitlyn Yow	98.50000	1
John Wulffert	98.39286	154
Jamie Phelps	98.15333	150
Nicole Millard	98.08562	146
Zachary Carter	98.08000	150
Brittny Thomas	98.00000	3
Dipatrimarki Farkas	97.79355	155
Sarah Thompson	97.77899	138
David Adcock	97.69014	71
Ursula Gadomski	97.68976	166
Cristofer LeClair	97.67578	128
Melodee Johnson	97.65549	164
Loc Nguyen	97.63253	83
Ginger Johnson	97.57778	45
Laura McNeill	97.48582	141

INSPECTOR	avg_score	n_inspections
Patricia Sabby	97.14780	159
Joshua Volkan	97.08333	24
Shannon Flynn	97.02215	158
Nikia Lawrence	96.99029	103
Maria Powell	96.97191	89
Samatha Sparano	96.93066	137
Angela Myers	96.87681	138
Naterra McQueen	96.87413	143
Angela Stocks	96.67308	52
Elizabeth Jackson	96.56569	137
Joanne Rutkofske	96.28611	180
Christy Klaus	96.27143	140
Jackson Hooton	96.09375	16
Kendra Wiggins	96.06000	75
Lisa McCoy	96.01923	26
Daryl Beasley	95.81250	16
Karla Crowder	95.45652	23
Jason Dunn	94.89857	350
Lauren Harden	94.74444	45
Lucy Schrum	94.65476	126
Meghan Scott	94.32432	37
Thomas Jumalon	89.00000	3

5) Small Samples Sizes?

Yes, in both the inspectors and the city tables we see instances where the sample size is less than 5, as you can see in the n_inspections column, which could lead to possible extreme values, skewing the data.

6) Restaurants vs Other

As we see in the table, the scores for restaurant (where is_restaurant is TRUE) is slightly less than for non-restaurants

```
inspections %>%
  mutate(is_restaurant = FACILITYTYPE == "Restaurant") %>%
  group_by(is_restaurant) %>%
  summarise(
    avg_score = mean(SCORE, na.rm = TRUE),
    n = n(),  # Total inspections in each group
    .groups = "drop"
  ) %>%
  kable(caption = "Scores by Facility Type") %>%
  kable_styling()
```

Scores by Facility Type

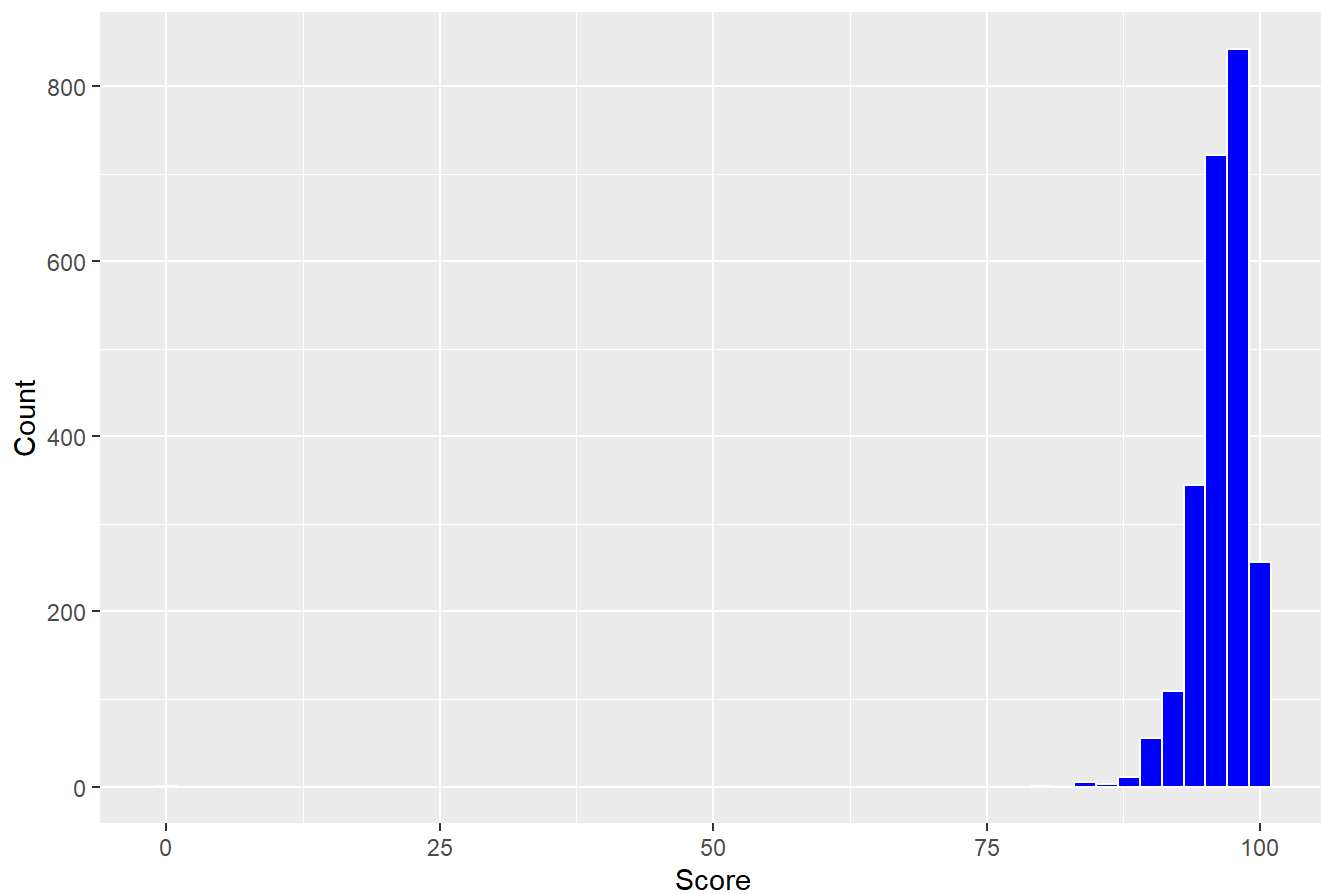| is_restaurant	avg_score	n
FALSE	98.05990	1227
TRUE	96.68070	2352
NA	94.15034	296

7) Restaurant Specific Analysis

Here we will be redoing all the prior analysis' but with a df that contains only restaurants

```
rest_only <- inspections %>%
  filter(FACILITYTYPE == "Restaurant") #Creates a df with only restaurants

#histogram
ggplot(rest_only, aes(SCORE)) +
  geom_histogram(binwidth = 2, fill = "blue", color = "white") +
  labs(title = "Distribution of Inspection Scores",
       x = "Score", y = "Count")
```

Distribution of Inspection Scores


```
#new vs old
rest_only %>%
  mutate(restaurant_age = time_length(DATE_ - RESTAURANTOPENDATE, "years")) %>% #Essentially wha
t is happening here is calculating how long it's been since they've opened, using the time_lengt
h function
  ggplot(aes(restaurant_age, SCORE)) +
  geom_point(alpha = 0.5) +        # Add transparency for overlapping points
  labs(title = "Inspection Scores by Restaurant Age",
       x = "Years Since Opening", y = "Score")
```

Inspection Scores by Restaurant Age
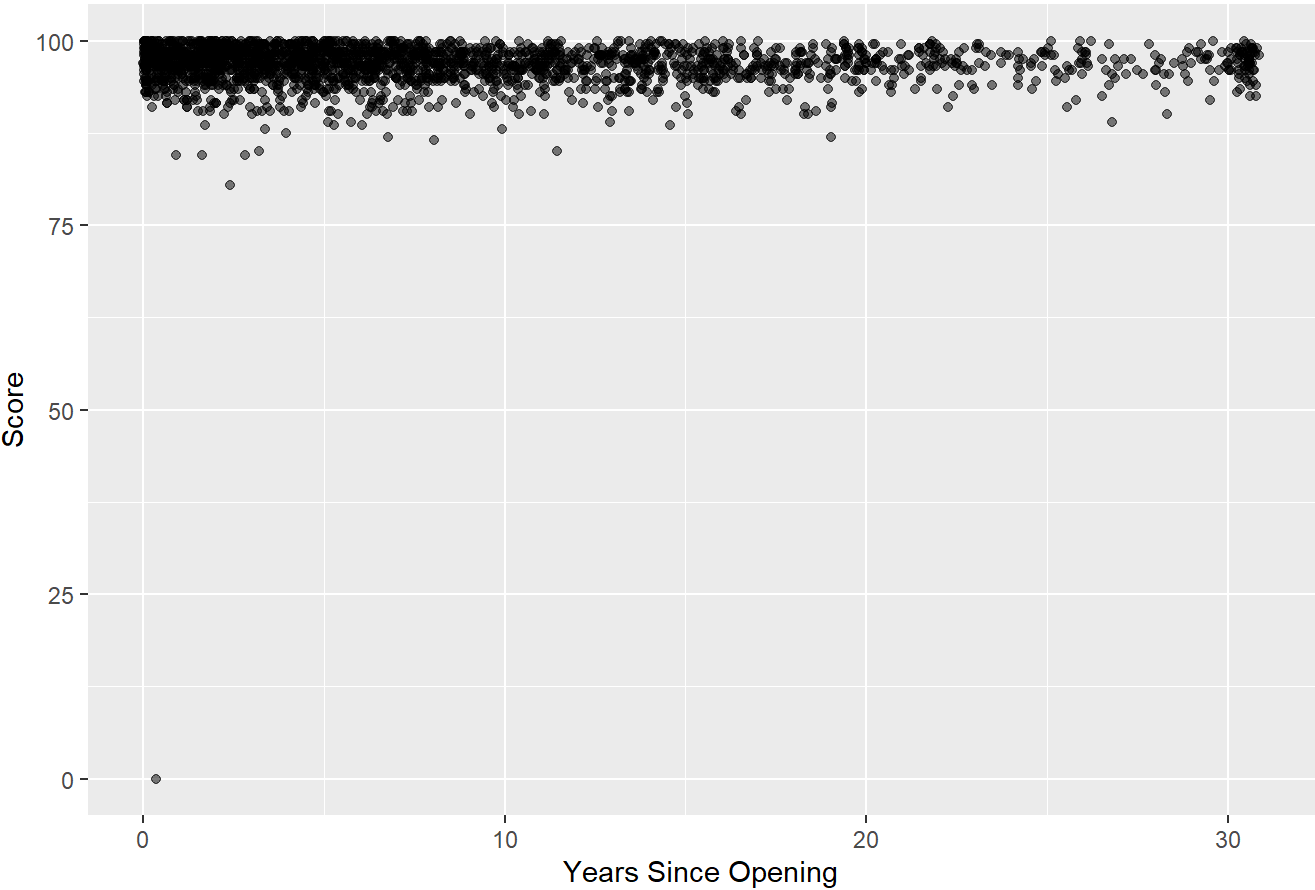

```
#by city
rest_city_scores <- rest_only %>%
  group_by(CITY) %>%
  summarise(
    avg_score = mean(SCORE, na.rm = TRUE),
    n_inspections = n(),  # Count number of inspections per city
    .groups = "drop"
  )

# Create formatted table with kableExtra
kable(rest_city_scores, caption = "Average Scores by City") %>%
  kable_styling()
```

Average Scores by City

CITY	avg_score	n_inspections
ANGIER	94.50000	1
APEX	97.08333	108
CARY	97.26355	406
CLAYTON	93.00000	1

CITY	avg_score	n_inspections
FUQUAY VARINA	96.90132	76
GARNER	95.79032	93
HOLLY SPRINGS	98.03125	80
KNIGHTDALE	95.14286	49
MORRISVILLE	96.65625	144
NEW HILL	100.00000	1
RALEIGH	96.59095	1193
RESEARCH TRIANGLE PARK	98.75000	2
ROLESVILLE	96.03846	13
WAKE FOREST	96.75564	133
WENDELL	94.57500	20
WILLOW SPRING	94.50000	1
ZEBULON	93.61290	31

```
#inspectors
# Calculate inspector performance metrics
rest_inspector_scores <- rest_only %>%
  group_by(INSPECTOR) %>%
  summarise(
    avg_score = mean(SCORE, na.rm = TRUE),
    n_inspections = n(),
    .groups = "drop"
  ) %>%
  arrange(desc(avg_score))  # arranging by highest avg score
#Essentially finding highest avg scores by inspector
kable(head(rest_inspector_scores, 200), caption = "Inspectors by Average Score") %>%
  kable_styling()
```

Inspectors by Average Score

INSPECTOR	avg_score	n_inspections
James Smith	99.00000	1
Greta Welch	98.50000	2

INSPECTOR	avg_score	n_inspections
John Wulffert	98.05856	111
Brittny Thomas	98.00000	3
Jamie Phelps	97.91667	108
Zachary Carter	97.81000	100
Dipatrimarki Farkas	97.69492	118
Loc Nguyen	97.64189	74
Ginger Johnson	97.62857	35
Nicole Millard	97.61458	48
Sarah Thompson	97.52551	98
Melodee Johnson	97.26639	122
Ursula Gadomski	97.20673	104
Jason Dunn	97.12500	4
Cristofer LeClair	97.05556	72
Joshua Volkan	96.97727	22
Laura McNeill	96.89674	92
Patricia Sabby	96.83621	116
Samatha Sparano	96.76068	117
Angela Myers	96.74038	104
Maria Powell	96.73333	60
Shannon Flynn	96.60714	112
Nikia Lawrence	96.45312	64
Jackson Hooton	96.29167	12
Naterra McQueen	96.18519	81
Angela Stocks	96.15278	36
Lisa McCoy	96.05556	18

INSPECTOR	avg_score	n_inspections
David Adcock	95.93750	8
Kendra Wiggins	95.89344	61
Christy Klaus	95.86000	100
Elizabeth Jackson	95.74375	80
Daryl Beasley	95.41667	12
Joanne Rutkofske	94.77174	92
Karla Crowder	94.39286	14
Lauren Harden	94.07353	34
Lucy Schrum	93.84884	86
Meghan Scott	93.31034	29
Thomas Jumalon	88.00000	2

#5) Again, sample size where it's less than 5 could yield extreme results where it can skew the overall data

git repo link: https://github.com/kabob-original/plan372hw2 (https://github.com/kabob-original/plan372hw2)