```
#################################################################################
# Toronto Housing Market Analysis & Rental Profit Calculator (ShinyApps)
# By: Sam Vuong, Raymond (shanhua) Huang, Carmon Ho, Kyle Murphy
#################################################################################
# This code package has 5 modules:
# 1. House Data Cleanup
# 2. Rental Data Cleanup
# 3. Data Analysis & Visualization
# 4. UI code for Rental Profit Calculator
# 5. Server code for Rental Profile Calculator
#
# The First 2 modules read
# ontario_property_listings_ORIG.csv and ontario_rental_listings_ORIG.csv data files and
# create ontario_property_listings.csv and ontario_rental_listings.csv data files for the others.
#################################################################################

#--------------------------------------------------------------------------------
#################### House Data Cleanup ########################################
#--------------------------------------------------------------------------------

#--------------------------------------------------------------------------------
## Install and load the required packages
#--------------------------------------------------------------------------------
if(!require(dplyr))
  install.packages("dplyr")
if(!require(tidyr))
  install.packages("tidyr")

#--------------------------------------------------------------------------------
## Data Loading
#--------------------------------------------------------------------------------
#house_data <- read.csv("housedata.csv")
house_data  <- read.csv("ontario_property_listings_ORIG.csv")

dim(house_data) #--41368    97

#colnames(house_data)
#str(house_data)

#--------------------------------------------------------------------------------
## Data Cleanup
#--------------------------------------------------------------------------------
```

```
clean_house_data <- house_data

# Remove URL column because it contains the address
clean_house_data = select(clean_house_data, -URL)
dim(clean_house_data) #--41368    96

#-------------------------------------------------------------------------------
## Data Cleanup for Status
#-------------------------------------------------------------------------------

# Remove white spaces
clean_house_data$Status <- trimws(clean_house_data$Status)
dim(house_data) #--41368

# Summary by Status
clean_house_data %>%
  group_by(Status) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

clean_house_data <- clean_house_data %>% filter(Status != "Lease")
dim(house_data) #--41368

# Change Status "Sale" to "Sold"
clean_house_data$Status[ clean_house_data$Status == "Sale"] <- "Sold"

# Summary by Status
clean_house_data %>%
  group_by(Status) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

#-------------------------------------------------------------------------------
# Re-group Type values
#-------------------------------------------------------------------------------

# Remove white spaces
clean_house_data$Type   <- trimws(clean_house_data$Type)

# Check NULL and non-NULL counts
sum( is.na(clean_house_data$Type) ) #--2
sum( !is.na(clean_house_data$Type) ) #--39206

# Save OLD Type values in a new column
clean_house_data$TypeOld <- clean_house_data$Type
```

```
dim(clean_house_data)

# Set Type = Other
clean_house_data$Type <- "Other"

# Set New Type values based on OLD Type values
clean_house_data$Type[ grepl("Detached",      clean_house_data$TypeOld) == TRUE ] <-
"Detached"
clean_house_data$Type[ grepl("Single Family", clean_house_data$TypeOld) == TRUE ] <-
"Detached"

clean_house_data$Type[ grepl("Semi-Detached", clean_house_data$TypeOld) == TRUE ] <-
"Semi-Detached"
clean_house_data$Type[ grepl("Semi Detached", clean_house_data$TypeOld) == TRUE ] <-
"Semi-Detached"
clean_house_data$Type[ grepl("SEMI-DETACHED", clean_house_data$TypeOld) == TRUE ]
<- "Semi-Detached"
clean_house_data$Type[ grepl("Link",          clean_house_data$TypeOld) == TRUE ] <-
"Semi-Detached"

clean_house_data$Type[ grepl("Condo",         clean_house_data$TypeOld) == TRUE ] <-
"Condo"
clean_house_data$Type[ grepl("Apartment",     clean_house_data$TypeOld) == TRUE ] <-
"Condo"
clean_house_data$Type[ grepl("Apt",           clean_house_data$TypeOld) == TRUE ] <-
"Condo"

clean_house_data$Type[ grepl("Townhouse",     clean_house_data$TypeOld) == TRUE ] <-
"Townhouse"
clean_house_data$Type[ grepl("Twnhouse",      clean_house_data$TypeOld) == TRUE ] <-
"Townhouse"

clean_house_data$Type[ grepl("Multiplex",     clean_house_data$TypeOld) == TRUE ] <-
"Multiplex"
clean_house_data$Type[ grepl("Duplex",        clean_house_data$TypeOld) == TRUE ] <-
"Multiplex"
clean_house_data$Type[ grepl("Triplex",       clean_house_data$TypeOld) == TRUE ] <-
"Multiplex"
clean_house_data$Type[ grepl("Fourplex",      clean_house_data$TypeOld) == TRUE ] <-
"Multiplex"

clean_house_data$Type[ grepl("Comm",          clean_house_data$TypeOld) == TRUE ] <-
"Commercial"
```

```
clean_house_data$Type[ grepl("Business",      clean_house_data$TypeOld) == TRUE ] <-
"Commercial"
clean_house_data$Type[ grepl("Industrial",    clean_house_data$TypeOld) == TRUE ] <-
"Commercial"
clean_house_data$Type[ grepl("Investment",    clean_house_data$TypeOld) == TRUE ] <-
"Commercial"
clean_house_data$Type[ grepl("Office",        clean_house_data$TypeOld) == TRUE ] <-
"Commercial"
clean_house_data$Type[ grepl("Retail",        clean_house_data$TypeOld) == TRUE ] <-
"Commercial"

clean_house_data$Type[ grepl("Land",          clean_house_data$TypeOld) == TRUE ] <-
"Vacant-Land"
clean_house_data$Type[ grepl("Lots",          clean_house_data$TypeOld) == TRUE ] <-
"Vacant-Land"
clean_house_data$Type[ grepl("No Building",   clean_house_data$TypeOld) == TRUE ] <-
"Vacant-Land"

# Count Summary by Type
#house_types <-
clean_house_data %>%
  group_by(Type) %>% summarise(Count = n())

# Remove OLD Type column
clean_house_data = select(clean_house_data, -TypeOld)
dim(clean_house_data)

#--------------------------------------------------------------------------------
## Data Cleanup for Area
#--------------------------------------------------------------------------------

# Remove white spaces
clean_house_data$Area <- trimws(clean_house_data$Area)

# Summary by Area
clean_house_data %>%
  group_by(Area) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

#--------------------------------------------------------------------------------
## Re-group Community / Neighbourhood values
#--------------------------------------------------------------------------------
```

```
# Remove white spaces
clean_house_data$Community <- trimws(clean_house_data$Community)

# Save OLD Community values in a new column
clean_house_data$CommunityOld <- clean_house_data$Community
dim(clean_house_data)

# Summary by Community
clean_house_data %>%
  group_by(Community) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

# Check NULL and non-NULL counts
sum( is.na(clean_house_data$CommunityOld) ) #--1179
sum( !is.na(clean_house_data$CommunityOld) ) #--40189

# Set New Community values based on OLD Community values
clean_house_data$Community[ grepl("Waterfront",            clean_house_data$CommunityOld)
== TRUE ] <- "Waterfront"
clean_house_data$Community[ grepl("Willowdale",            clean_house_data$CommunityOld)
== TRUE ] <- "Willowdale"
clean_house_data$Community[ grepl("Mount Pleasant",
clean_house_data$CommunityOld) == TRUE ] <- "Mount Pleasant"
clean_house_data$Community[ grepl("Islington-City Centre",
clean_house_data$CommunityOld) == TRUE ] <- "Islington-City Centre"
clean_house_data$Community[ grepl("Newtonbrook",           clean_house_data$CommunityOld)
== TRUE ] <- "Newtonbrook"
clean_house_data$Community[ grepl("Brampton",              clean_house_data$CommunityOld)
== TRUE ] <- "Brampton"

# Summary by Community
clean_house_data %>%
  group_by(Community) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% head(20)

# Remove OLD Community column
clean_house_data = select(clean_house_data, -CommunityOld)
dim(clean_house_data) #--41110   100

# Rename column "Community" to "Neighbourhood"
colnames(clean_house_data)[colnames(clean_house_data) == "Community"] <-
"Neighbourhood"
```

```
clean_house_data %>%
  group_by(Neighbourhood) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% head(20)

#-------------------------------------------------------------------------------------
# Re-group Age values
#-------------------------------------------------------------------------------------

# Remove white spaces
clean_house_data$Age <- trimws(clean_house_data$Age)

clean_house_data %>% filter(Age == "New")    %>% nrow() #--970
clean_house_data %>% filter(Age == "0to5")   %>% nrow() #--4338
clean_house_data %>% filter(Age == "6to10")  %>% nrow() #--1021
clean_house_data %>% filter(Age == "6to15")  %>% nrow() #--2791
clean_house_data %>% filter(Age == "11to15") %>% nrow() #--535
clean_house_data %>% filter(Age == "16to30") %>% nrow() #--3266
clean_house_data %>% filter(Age == "31to50") %>% nrow() #--2607
clean_house_data %>% filter(Age == "51to99") %>% nrow() #--1878
clean_house_data %>% filter(Age == "100+")   %>% nrow() #--605
clean_house_data %>% filter(Age == "999")    %>% nrow() #--1

clean_house_data$Age[ clean_house_data$Age == "New"]    <- "0"
clean_house_data$Age[ clean_house_data$Age == "0to5"]   <- "3"
clean_house_data$Age[ clean_house_data$Age == "6to10"]  <- "8"
clean_house_data$Age[ clean_house_data$Age == "6to15"]  <- "10"
clean_house_data$Age[ clean_house_data$Age == "11to15"] <- "13"
clean_house_data$Age[ clean_house_data$Age == "16to30"] <- "23"
clean_house_data$Age[ clean_house_data$Age == "31to50"] <- "40"
clean_house_data$Age[ clean_house_data$Age == "51to99"] <- "75"
clean_house_data$Age[ clean_house_data$Age == "100+"]   <- "101"
clean_house_data$Age[ clean_house_data$Age == "999"]    <- "101"

unique(clean_house_data$Age) %>% sort()

#-------------------------------------------------------------------------------------
# Re-group Basement values
#-------------------------------------------------------------------------------------

# Remove white spaces
clean_house_data$Basement <- trimws(clean_house_data$Basement)

# Check NULL and non-NULL counts
```

```
sum( is.na(clean_house_data$Basement) ) #--695
sum( !is.na(clean_house_data$Basement) ) #--40673

# Save OLD Basement values in a new column
clean_house_data$BasementOld <- clean_house_data$Basement
dim(clean_house_data)

# Set Basement = Unfinished
clean_house_data$Basement[!is.na(clean_house_data$BasementOld)]   <- "Unfinished"

# Set New Basement values based on OLD Basement values
clean_house_data$Basement[ grepl("Apartment",   clean_house_data$BasementOld) == TRUE
] <- "Finished"
clean_house_data$Basement[ grepl("Fin W/O",     clean_house_data$BasementOld) == TRUE ]
<- "Finished"
clean_house_data$Basement[ grepl("Finished",    clean_house_data$BasementOld) == TRUE ]
<- "Finished"
clean_house_data$Basement[ grepl("Full, Suite", clean_house_data$BasementOld) == TRUE ]
<- "Finished"
clean_house_data$Basement[ grepl("Part Bsmt",   clean_house_data$BasementOld) == TRUE
] <- "Part Finished"
clean_house_data$Basement[ grepl("Part Fin",    clean_house_data$BasementOld) == TRUE ]
<- "Part Finished"
clean_house_data$Basement[ grepl("Partial",     clean_house_data$BasementOld) == TRUE ]
<- "Part Finished"

clean_house_data$Basement[clean_house_data$BasementOld == "None"]  <- "None"
clean_house_data$Basement[clean_house_data$BasementOld == "No"]    <- "None"

clean_house_data$Basement[clean_house_data$Type        == "Condo"] <- "None"

clean_house_data %>%
  group_by(Basement) %>%
  summarise(Count = n()) #--%>% arrange(desc(Count))

# Remove OLD Basement column
clean_house_data = select(clean_house_data, -BasementOld)
dim(clean_house_data)

#----------------------------------------------------------------------------
# Add a new column FinishedBasement
#----------------------------------------------------------------------------
```

```
clean_house_data %>%
  group_by(Basement) %>%
  summarise(Count = n()) #--%>% arrange(desc(Count))

# Add a new column FinishedBasement and default to O (false)
clean_house_data$FinishedBasement <- 0
dim(clean_house_data)

# Set FinishedBasement = 1 if the column Basement is "Finished"
clean_house_data$FinishedBasement[ clean_house_data$Basement == "Finished" ] <- 1

clean_house_data %>%
  group_by(FinishedBasement) %>%
  summarise(Count = n()) #--%>% arrange(desc(Count))

#-----------------------------------------------------------------------------
# Add a new column NearSchool
#-----------------------------------------------------------------------------

# Add a new column NearSchool and default to O (false)
clean_house_data$NearSchool <- 0
dim(clean_house_data)

# Set NearSchool = 1 if the column Feature contain key work "School"
clean_house_data$NearSchool[ grepl("School", clean_house_data$Feature,
ignore.case=TRUE) == TRUE ] <- 1

clean_house_data %>%
  group_by(NearSchool) %>% summarise(Count = n())

#-----------------------------------------------------------------------------
# Add a new column NearPark
#-----------------------------------------------------------------------------

# Add a new column NearPark and default to O (false)
clean_house_data$NearPark <- 0
dim(clean_house_data)

# Set NearSchool = 1 if the column Feature contain key work "Park"
clean_house_data$NearPark[ grepl("Park", clean_house_data$Feature, ignore.case=TRUE) ==
TRUE ] <- 1

clean_house_data %>%
  group_by(NearPark) %>% summarise(Count = n())

#-----------------------------------------------------------------------------------
## Data Cleanup for Price
#-----------------------------------------------------------------------------------

dim(clean_house_data) #--41368

# Check NULL and non-NULL counts
sum( is.na(clean_house_data$ListPrice) ) #--1183
sum( !is.na(clean_house_data$ListPrice) ) #--40185

clean_house_data <- clean_house_data %>% filter( !is.na(clean_house_data$ListPrice) )
dim(clean_house_data) #--40185

# Price Summary
clean_house_data %>%
  group_by(Type) %>%
  summarise(Count  = n(),
            min    = min(ListPrice),    max  = max(ListPrice),
            median = median(ListPrice), mean = mean(ListPrice) ) %>% arrange(desc(Count))

# Check and remove outliers
boxplot(clean_house_data$ListPrice)

clean_house_data %>% filter(ListPrice < 1000)     %>% nrow() #--3
clean_house_data %>% filter(ListPrice < 2000)     %>% nrow() #--233
clean_house_data %>% filter(ListPrice < 5000)     %>% nrow() #--1187
clean_house_data %>% filter(ListPrice < 10000)    %>% nrow() #--1217
clean_house_data %>% filter(ListPrice < 100000)   %>% nrow() #--1244

clean_house_data %>% filter(ListPrice > 2000000)  %>% nrow() #--1104
clean_house_data %>% filter(ListPrice > 3000000)  %>% nrow() #--328
clean_house_data %>% filter(ListPrice > 4000000)  %>% nrow() #--113
clean_house_data %>% filter(ListPrice > 5000000)  %>% nrow() #--57
clean_house_data %>% filter(ListPrice > 7500000)  %>% nrow() #--17
clean_house_data %>% filter(ListPrice > 10000000) %>% nrow() #--6
clean_house_data %>% filter(ListPrice > 12000000) %>% nrow() #--4
clean_house_data %>% filter(ListPrice > 14000000) %>% nrow() #--1

# Remove Price outliers
clean_house_data <- clean_house_data %>% filter(ListPrice >= 100000 & ListPrice <=
5000000)
```

```
dim(clean_house_data) #--38884

# Double Check Price outliers
boxplot(clean_house_data$ListPrice)

# Price Summary
clean_house_data %>%
  group_by(Type) %>%
  summarise(Count  = n(),
            min    = min(ListPrice),    max  = max(ListPrice),
            median = median(ListPrice), mean = mean(ListPrice) )  %>% arrange(desc(Count))

#-----------------------------------------------------------------------------------
## Data Cleanup for Taxes
#-----------------------------------------------------------------------------------

dim(clean_house_data) #--38884

# Check NULL and non-NULL counts
sum( is.na(clean_house_data$Taxes) ) #--1280
sum( !is.na(clean_house_data$Taxes) ) #--37604

clean_house_data <- clean_house_data %>% filter( !is.na(clean_house_data$Taxes) )
dim(clean_house_data) #--37604

# Taxes Summary
clean_house_data %>%
  group_by(Type) %>%
  summarise(Count  = n(),
            min    = min(Taxes),    max  = max(Taxes),
            median = median(Taxes), mean = mean(Taxes) ) %>% arrange(desc(Count))

# Check and remove outliers
boxplot(clean_house_data$Taxes)

clean_house_data %>% filter(Taxes < 100)    %>% nrow() #--72
clean_house_data %>% filter(Taxes < 500)    %>% nrow() #--112
clean_house_data %>% filter(Taxes < 1000)   %>% nrow() #--389

clean_house_data %>% filter(Taxes > 10000)  %>% nrow() #--812
clean_house_data %>% filter(Taxes > 15000)  %>% nrow() #--189
clean_house_data %>% filter(Taxes > 20000)  %>% nrow() #--59
clean_house_data %>% filter(Taxes > 50000)  %>% nrow() #--7
clean_house_data %>% filter(Taxes > 100000) %>% nrow() #--6

# Remove Taxes outliers
clean_house_data <- clean_house_data %>% filter(Taxes >= 1000 & Taxes <= 20000)
dim(clean_house_data) #--37156

# Check and remove Price outliers
boxplot(clean_house_data$Taxes)

# Taxes Summary
clean_house_data %>%
  group_by(Type) %>%
  summarise(Count  = n(),
            min    = min(Taxes),    max  = max(Taxes),
            median = median(Taxes), mean = mean(Taxes) )  %>% arrange(desc(Count))

#---------------------------------------------------------------------------------
# Save the new dataset
#---------------------------------------------------------------------------------
dim(clean_house_data) #--37156   99
write.csv(clean_house_data,"ontario_property_listings.csv",  row.names = FALSE)

#---------------------------------------------------------------------------------
#################### Rental Data CLeanup ########################################
#---------------------------------------------------------------------------------

#---------------------------------------------------------------------------------
## Install and load the required packages
#---------------------------------------------------------------------------------
if(!require(dplyr))
  install.packages("dplyr")
if(!require(tidyr))
  install.packages("tidyr")

#---------------------------------------------------------------------------------
## Data Loading
#---------------------------------------------------------------------------------
#rental_data <- read.csv("finalrental.csv", header=TRUE)
rental_data  <- read.csv("ontario_rental_listings_ORIG.csv", header=TRUE)

dim(rental_data) #--11647   56

#colnames(rental_data)
```

```
#str(rental_data)

#---------------------------------------------------------------------------------
## Data Cleanup
#---------------------------------------------------------------------------------

clean_rental_data <- rental_data

# Remove Link column because it contains the address
clean_rental_data = select(clean_rental_data, -Link)
dim(clean_rental_data) #--11647   55

#---------------------------------------------------------------------------------
## Data Cleanup for Status
#---------------------------------------------------------------------------------

# Remove white spaces
clean_rental_data$Status <- trimws(clean_rental_data$Status)

dim(clean_rental_data) #--11647

# Summary by Status
clean_rental_data %>%
  group_by(Status) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

clean_rental_data <- clean_rental_data %>% filter(Status == "Lease")
dim(clean_rental_data) #--11063

# Summary by Status
clean_rental_data %>%
  group_by(Status) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

#---------------------------------------------------------------------------------
## Re-group Type values
#---------------------------------------------------------------------------------

# Remove white spaces
clean_rental_data$Type  <- trimws(clean_rental_data$Type)

# Summary by Type
clean_rental_data %>%
  group_by(Type) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% head(20)

# Save OLD Type values in a new column
clean_rental_data$TypeOld <- clean_rental_data$Type
dim(clean_rental_data)

# Check NULL and non-NULL counts
sum( is.na(clean_rental_data$TypeOld) ) #--0
sum( !is.na(clean_rental_data$TypeOld) ) #--11063

# Set Type = Other
clean_rental_data$Type <- "Other"

# Set New Type values based on OLD Type values
clean_rental_data$Type[ grepl("Detached",      clean_rental_data$TypeOld) == TRUE ] <-
"Detached"
clean_rental_data$Type[ grepl("Single Family", clean_rental_data$TypeOld) == TRUE ] <-
"Detached"

clean_rental_data$Type[ grepl("Semi-Detached", clean_rental_data$TypeOld) == TRUE ] <-
"Semi-Detached"
clean_rental_data$Type[ grepl("Semi Detached", clean_rental_data$TypeOld) == TRUE ] <-
"Semi-Detached"
clean_rental_data$Type[ grepl("SEMI-DETACHED", clean_rental_data$TypeOld) == TRUE ] <-
"Semi-Detached"
clean_rental_data$Type[ grepl("Link",          clean_rental_data$TypeOld) == TRUE ] <-
"Semi-Detached"

clean_rental_data$Type[ grepl("Condo",         clean_rental_data$TypeOld) == TRUE ] <-
"Condo"
clean_rental_data$Type[ grepl("Apartment",     clean_rental_data$TypeOld) == TRUE ] <-
"Condo"
clean_rental_data$Type[ grepl("Apt",           clean_rental_data$TypeOld) == TRUE ] <- "Condo"

clean_rental_data$Type[ grepl("Townhouse",     clean_rental_data$TypeOld) == TRUE ] <-
"Townhouse"
clean_rental_data$Type[ grepl("Twnhouse",      clean_rental_data$TypeOld) == TRUE ] <-
"Townhouse"

clean_rental_data$Type[ grepl("Multiplex",     clean_rental_data$TypeOld) == TRUE ] <-
"Multiplex"
clean_rental_data$Type[ grepl("Duplex",        clean_rental_data$TypeOld) == TRUE ] <-
```

```
"Multiplex"
clean_rental_data$Type[ grepl("Triplex",       clean_rental_data$TypeOld) == TRUE ] <-
"Multiplex"
clean_rental_data$Type[ grepl("Fourplex",      clean_rental_data$TypeOld) == TRUE ] <-
"Multiplex"

clean_rental_data$Type[ grepl("Comm",          clean_rental_data$TypeOld) == TRUE ] <-
"Commercial"
clean_rental_data$Type[ grepl("Business",      clean_rental_data$TypeOld) == TRUE ] <-
"Commercial"
clean_rental_data$Type[ grepl("Industrial",    clean_rental_data$TypeOld) == TRUE ] <-
"Commercial"
clean_rental_data$Type[ grepl("Investment",    clean_rental_data$TypeOld) == TRUE ] <-
"Commercial"
clean_rental_data$Type[ grepl("Office",        clean_rental_data$TypeOld) == TRUE ] <-
"Commercial"
clean_rental_data$Type[ grepl("Retail",        clean_rental_data$TypeOld) == TRUE ] <-
"Commercial"

clean_rental_data$Type[ grepl("Land",          clean_rental_data$TypeOld) == TRUE ] <-
"Vacant-Land"
clean_rental_data$Type[ grepl("Lots",          clean_rental_data$TypeOld) == TRUE ] <-
"Vacant-Land"
clean_rental_data$Type[ grepl("No Building",   clean_rental_data$TypeOld) == TRUE ] <-
"Vacant-Land"

# Summary by Type
clean_rental_data %>%
  group_by(Type) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

# Remove OLD Type column
clean_rental_data = select(clean_rental_data, -TypeOld)
dim(clean_rental_data)

#----------------------------------------------------------------------------------
## Data Cleanup for Area
#----------------------------------------------------------------------------------

# Remove white spaces
clean_rental_data$Area  <- trimws(clean_rental_data$Area)

dim(clean_rental_data)
```

```
# Summary by Area
clean_rental_data %>%
  group_by(Area) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

clean_rental_data <- clean_rental_data %>% filter(Area == "Toronto")
dim(clean_rental_data) #--11621

#-------------------------------------------------------------------------------
## Re-group Community values
#-------------------------------------------------------------------------------

# Remove white spaces
clean_rental_data$Community  <- trimws(clean_rental_data$Community)

# Summary by Community
clean_rental_data %>%
  group_by(Community) %>%
  summarise(Count = n()) %>% arrange(desc(Count))

# Save OLD Community values in a new column
clean_rental_data$CommunityOld <- clean_rental_data$Community
dim(clean_rental_data)

# Check NULL and non-NULL counts
sum( is.na(clean_rental_data$CommunityOld) ) #--0
sum( !is.na(clean_rental_data$CommunityOld) ) #--11063

# Set New Community values based on OLD Community values
clean_rental_data$Community[ grepl("Waterfront",           clean_rental_data$CommunityOld) ==
TRUE ] <- "Waterfront"
clean_rental_data$Community[ grepl("Willowdale",           clean_rental_data$CommunityOld)
== TRUE ] <- "Willowdale"
clean_rental_data$Community[ grepl("Mount Pleasant",       clean_rental_data$CommunityOld)
== TRUE ] <- "Mount Pleasant"
clean_rental_data$Community[ grepl("Islington-City Centre", clean_rental_data$CommunityOld)
== TRUE ] <- "Islington-City Centre"
clean_rental_data$Community[ grepl("Newtonbrook",          clean_rental_data$CommunityOld)
== TRUE ] <- "Newtonbrook"
clean_rental_data$Community[ grepl("Brampton",             clean_rental_data$CommunityOld)
== TRUE ] <- "Brampton"
```

```
# Summary by Community
clean_rental_data %>%
  group_by(Community) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% head(20)

# Remove OLD Community column
clean_rental_data = select(clean_rental_data, -CommunityOld)
dim(clean_rental_data)

# Rename column "Community" to "Neighbourhood"
colnames(clean_rental_data)[colnames(clean_rental_data) == "Community"] <-
"Neighbourhood"

clean_rental_data %>%
  group_by(Neighbourhood) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% head(20)

#----------------------------------------------------------------------------------
# Re-group Basement values
#----------------------------------------------------------------------------------

# Remove white spaces
clean_rental_data$Basement <- trimws(clean_rental_data$Basement)

clean_rental_data %>%
  group_by(Basement) %>%
  summarise(Count = n()) #--%>% arrange(desc(Count))

# Save OLD Basement values in a new column
clean_rental_data$BasementOld <- clean_rental_data$Basement
dim(clean_rental_data)

# Check NULL and non-NULL counts
sum( is.na(clean_rental_data$Basement) ) #--52
sum( !is.na(clean_rental_data$Basement) ) #--11011

# Set Basement = Unfinished
clean_rental_data$Basement[!is.na(clean_rental_data$BasementOld)]   <- "Unfinished"

# Set New Basement values based on OLD Basement values
clean_rental_data$Basement[ grepl("Apartment",   clean_rental_data$BasementOld) == TRUE ]
<- "Finished"
clean_rental_data$Basement[ grepl("Fin W/O",     clean_rental_data$BasementOld) == TRUE ]
```

```
<- "Finished"
clean_rental_data$Basement[ grepl("Finished",    clean_rental_data$BasementOld) == TRUE ]
<- "Finished"
clean_rental_data$Basement[ grepl("Full, Suite", clean_rental_data$BasementOld) == TRUE ]
<- "Finished"
clean_rental_data$Basement[ grepl("Part Bsmt",   clean_rental_data$BasementOld) == TRUE ]
<- "Part Finished"
clean_rental_data$Basement[ grepl("Part Fin",    clean_rental_data$BasementOld) == TRUE ]
<- "Part Finished"
clean_rental_data$Basement[ grepl("Partial",     clean_rental_data$BasementOld) == TRUE ] <-
"Part Finished"

clean_rental_data$Basement[clean_rental_data$BasementOld == "None"]  <- "None"
clean_rental_data$Basement[clean_rental_data$BasementOld == "No"]    <- "None"

clean_rental_data$Basement[clean_rental_data$Type        == "Condo"] <- "None"

clean_rental_data %>%
  group_by(Basement) %>%
  summarise(Count = n()) #--%>% arrange(desc(Count))

# Remove OLD Basement column
clean_rental_data = select(clean_rental_data, -BasementOld)
dim(clean_rental_data)

#---------------------------------------------------------------------------
# Add a new column FinishedBasement
#---------------------------------------------------------------------------

clean_rental_data %>%
  group_by(Basement) %>%
  summarise(Count = n()) #--%>% arrange(desc(Count))

# Add a new column FinishedBasement and default to O (false)
clean_rental_data$FinishedBasement <- 0
dim(clean_rental_data)

# Set FinishedBasement = 1 if the column Basement is "Finished"
clean_rental_data$FinishedBasement[ clean_rental_data$Basement == "Finished" ] <- 1

clean_rental_data %>%
  group_by(FinishedBasement) %>%
  summarise(Count = n()) #--%>% arrange(desc(Count))
```

```
#----------------------------------------------------------------------------------
## Data Cleanup for Price
#----------------------------------------------------------------------------------

# Remove white spaces
clean_rental_data$Price <- trimws(clean_rental_data$Price)

dim(clean_rental_data) #--11063

clean_rental_data %>% filter(Price == "Sign up to See") %>% nrow() #--1078

clean_rental_data <- clean_rental_data %>% filter(Price != "Sign up to See")
dim(clean_rental_data) #--9984

clean_rental_data$Price <- gsub("\\$", "", gsub(",", "", clean_rental_data$Price) )
clean_rental_data$Price <- as.numeric(clean_rental_data$Price)

# Check and remove Price outliers
boxplot(clean_rental_data$Price)

clean_rental_data %>% filter(Price < 100)     %>% nrow() #--10
clean_rental_data %>% filter(Price < 500)     %>% nrow() #--22
clean_rental_data %>% filter(Price < 1000)    %>% nrow() #--42
clean_rental_data %>% filter(Price < 1500)    %>% nrow() #--206
clean_rental_data %>% filter(Price < 2000)    %>% nrow() #--2824

clean_rental_data %>% filter(Price > 3000)    %>% nrow() #--1399
clean_rental_data %>% filter(Price > 4000)    %>% nrow() #--471
clean_rental_data %>% filter(Price > 5000)    %>% nrow() #--228
clean_rental_data %>% filter(Price > 6000)    %>% nrow() #--140
clean_rental_data %>% filter(Price > 7000)    %>% nrow() #--75
clean_rental_data %>% filter(Price > 8000)    %>% nrow() #--25
clean_rental_data %>% filter(Price > 10000)   %>% nrow() #--4

clean_rental_data <- clean_rental_data %>% filter(Price >= 1000 & Price <= 6000)
dim(clean_rental_data) #--9789

# Double Check Price outliers
boxplot(clean_rental_data$Price)

# Price Summary
clean_rental_data %>%
  group_by(Area) %>%
  summarise(Count = n(),
            min   = min(Price), max = max(Price),
            median = median(Price), mean = mean(Price) )

#-----------------------------------------------------------------------------------
# Save the new dataset
#-----------------------------------------------------------------------------------
dim(clean_rental_data) #--9789   56
write.csv(clean_rental_data,"ontario_rental_listings.csv",  row.names = FALSE)

#-----------------------------------------------------------------------------------
#################### Data Analysis & Visualization ##################################
#-----------------------------------------------------------------------------------

#-----------------------------------------------------------------------------------
# Install and load the required packages
#-----------------------------------------------------------------------------------
if(!require(dplyr))
  install.packages("dplyr")
if(!require(tidyr))
  install.packages("tidyr")
if(!require(corrplot))
  install.packages("corrplot")
if(!require(ggplot2))
  install.packages("ggplot2")
if(!require(ggthemes))
  install.packages("ggthemes")

#-----------------------------------------------------------------------------------
#Get clean house data
#-----------------------------------------------------------------------------------
fulldata  <- read.csv("ontario_property_listings.csv")

#Ensure correct date format
#-----------------------------------------------------------------------------------
fulldata$ListDate   <-as.Date(fulldata$ListDate)
fulldata$SoldDate   <-as.Date(fulldata$SoldDate)
fulldata$ClosedDate <-as.Date(fulldata$ClosedDate)

#Selecting data for visualization
#-----------------------------------------------------------------------------------
house <- subset(fulldata,
              fulldata$Status=="Sold" & !is.na(fulldata$ListPrice) & !is.na(fulldata$SoldPrice),
              c(Status,Area,Type,Bathrooms,Bedrooms,ParkingTotal,ListPrice,SoldPrice,
                Taxes,ListDate,SoldDate,ClosedDate,Basement,Garage,Feature,
                Age,NearSchool,NearPark,FinishedBasement))
#View(house)
#check how many records were kept
NROW(house)
NROW(fulldata)
NROW(house)/NROW(fulldata)*100
#View(house)
#Grouping by
houseg<- house %>% select(Status,Area,Type,SoldPrice) %>%
  group_by(Status,Area,Type) %>%
dplyr::summarise(AveragePrice=mean(SoldPrice),count=n())
#View(houseg)

#------------------------------------------------------------------------------------
#1. Bubble plot region price and count + type
#------------------------------------------------------------------------------------
ggplot(houseg, aes(x=Area, y=AveragePrice, size = count,color=Type)) +
   geom_point(alpha=2)+
 scale_size(range = c(.1, 24), name="Sales Count")+
ylim(200000,1500000)

#------------------------------------------------------------------------------------
#2. Bubble plot - Price Difference vs region and house type
#------------------------------------------------------------------------------------
houseg1<- house %>% mutate(PriceDifference=SoldPrice-ListPrice) %>%
  select(Status,Area,Type,PriceDifference) %>%
  group_by(Status,Area,Type) %>%
  dplyr::summarise(AveragePriceDifference=mean(PriceDifference),count=n())
ggplot(houseg1, aes(x=Area, y=AveragePriceDifference, size = count,color=Type)) +
   geom_point(alpha=2)+
 scale_size(range = c(.1, 24), name="Sales Count")+
ylim(-25000,100000)

#------------------------------------------------------------------------------------
#3. Boxplot - Price Difference vs region and house type
#------------------------------------------------------------------------------------
houseg11<- house %>% mutate(PriceDifference=SoldPrice-ListPrice) %>%
  select(Status,Area,Type,PriceDifference) %>%
  filter(Type %in% c("Semi-Detached","Detached","Condo"))%>%
  filter(Area %in% c("Toronto","York","Peel","Halton","Durham"))
```

```
ggplot(data=houseg11,aes(x=Area,y=PriceDifference/1000,fill=Type))+geom_boxplot()+ylim(-10
0,100)

#-------------------------------------------------------------------------------------
#4. Bar chart - Date Difference vs region and house type
#-------------------------------------------------------------------------------------
houseg2<- house %>% mutate(DateDiff=difftime(SoldDate, ListDate, units = "days")) %>%
  filter(Type %in% c("Detached","Semi-Detached","Condo")) %>%
  filter(Area %in% c("Toronto","York","Peel","Hamilton","Halton","Durham"))%>%
  select(Status,Area,Type,DateDiff) %>% group_by(Status,Area,Type) %>%
  dplyr::summarise(AverageDateDiff=mean(DateDiff),count=n())
#houseg2[order(houseg2$AverageDateDiff),]
#View(houseg2)
#ggplot(data=houseg2,aes(x=Area,y=AverageDateDiff))+stat_density2d(aes(color=Type))+geo
m_point()
#ggplot(data=houseg2,aes(x=Area,y=AverageDateDiff,shape=Type,size=count))+geom_point()
#https://www.r-graph-gallery.com/48-grouped-barplot-with-ggplot2.html
ggplot(houseg2,aes(fill=Type,x=Area,y=AverageDateDiff)) +geom_bar(position="dodge",
stat="identity") +
    ggtitle("Average Date Diff") +
    xlab("Area") +
    ylab("Average Date Diff") +
    theme_bw()

#-------------------------------------------------------------------------------------
# Basement Finished / Unfinished Data Analysis
#-------------------------------------------------------------------------------------
sum(is.na(fulldata$Basement))/nrow(fulldata)
sum(is.na(fulldata$Feature))/nrow(fulldata)
sum(is.na(fulldata$Garage))/nrow(fulldata)

housea <- house %>% filter(Type %in% c("Detached","Semi-Detached","Townhouse"))
unique(housea$Basement)

houseb<- housea %>% mutate(PriceDifference=SoldPrice-ListPrice) %>%
  select(Basement,SoldPrice) %>%
  group_by(Basement) %>%
  dplyr::summarise(AverageSoldPrice=mean(SoldPrice),count=n())

housec<- houseb %>%
  filter(Basement %in% c("Apartment","Finished","Part Finished","Unfinished"))
#check how many records were kept
housec
```

```
#-----------------------------------------------------------------------------------
#5. Basement Finished / Unfinished Pie Chart
#-----------------------------------------------------------------------------------
ggplot(housec, aes(x="",y=count, fill=Basement))+
  geom_bar(stat="identity",width = 1,color="white") +
  coord_polar("y",start=0)+theme_void()

#-----------------------------------------------------------------------------------
#6. Basement Finished / Unfinished Price Difference Chart
#-----------------------------------------------------------------------------------
ggplot(housec,aes(x=Basement,y=AverageSoldPrice,fill=Basement)) +
  geom_bar(position="dodge", stat="identity") +
  scale_y_continuous(breaks=seq(0,1000000,250000)) +
  xlab("Basement") +
  ylab("Average Sold Price") +
  ggtitle("Average Sold Price") +
  theme_bw()

#-----------------------------------------------------------------------------------
#7. Age effect on Sales Volume and House Price
#-----------------------------------------------------------------------------------
houseage<- house %>% select(Age,Type,SoldPrice) %>%
  group_by(Age,Type) %>% dplyr::summarise(AveragePrice=mean(SoldPrice),count=n())
#View(houseage)

houseage1<- house %>% select(Age,SoldPrice) %>%
  group_by(Age) %>% dplyr::summarise(AveragePrice=mean(SoldPrice),count=n())
#View(houseage1)
houseage2<-na.omit(houseage1)
#View(houseage2)

coeff<-1000
ggplot(houseage2, aes(Age)) +
  geom_line(aes(y=count), colour="blue") +
  geom_line(aes(y=AveragePrice/coeff), colour="red") +
  scale_y_continuous(name = "Sales Count",
    sec.axis = sec_axis(~.*coeff, breaks =seq(0,4000000,1250000), name = "Average Price"))

#-----------------------------------------------------------------------------------
#8. School & Park effect on Sales Volume and House Price
#-----------------------------------------------------------------------------------
houseSchoolPark <- house %>%
  select(Type, NearSchool, NearPark, SoldPrice) %>%
  filter(Type %in% c("Detached", "Semi-Detached", "Condo")) %>%
  group_by(Type, NearSchool, NearPark) %>%
  summarise(AveragePrice=mean(SoldPrice), count=n())
#View(houseSchoolPark)
head(houseSchoolPark)

#-------------------------------------------------------------------------------
################### UI code for Rental Profit Calculator ###########################
#-------------------------------------------------------------------------------
#ui.R
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
## Install and load the required packages
#-------------------------------------------------------------------------------
if(!require(shiny))
  install.packages("shiny")
if(!require(shinythemes))
  install.packages("shinythemes")

if(!require(dplyr))
  install.packages("dplyr")
if(!require(tidyr))
  install.packages("tidyr")

#-------------------------------------------------------------------------------
## Data Preparation for UI for ShinyApps
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Read Data
#-------------------------------------------------------------------------------
fullhousedata  <- read.csv("ontario_property_listings.csv")
fullrentaldata <- read.csv("ontario_rental_listings.csv")
rates_data     <- read.csv("ontario_mortgage_rates.csv")

house_data  <- fullhousedata
rental_data <- fullrentaldata

#-------------------------------------------------------------------------------
# Rental Data Preparation
#-------------------------------------------------------------------------------
```

```
# Top Type
top_type <- rental_data %>%
  group_by(Type) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% head(1)
top_type <- as.character(top_type$Type)
top_type

# Top Areas
top_areas <- rental_data %>%
  group_by(Area) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% select(Area) %>% head(10)
top_area <- as.character(top_areas$Area %>% head(1))
top_area

# Top Neighbourhoods
top_neighbourhoods <- rental_data %>%
  group_by(Neighbourhood) %>%
  summarise(Count = n()) %>% arrange(desc(Count)) %>% select(Neighbourhood) %>%
head(20)
top_neighbourhood <- as.character(top_neighbourhoods$Neighbourhood %>% head(1))
top_neighbourhood

top_bedrooms <- 3
top_bathrooms <- 2

top_rent_data <- rental_data %>% filter(Area == top_area)
dim(top_rent_data) #--9844   57 #--42

rent_min  <- as.integer( min(top_rent_data$Price)  / 100 ) * 100
rent_max  <- as.integer( max(top_rent_data$Price)  / 100 + 1 ) * 100
rent_avg  <- as.integer( mean(top_rent_data$Price) / 100 + 1 ) * 100
rent_avg  <- as.integer( (rent_min + rent_max) / 2 / 100 + 1 ) * 100
rent_step <- 100
rent_min
rent_avg
rent_max

#------------------------------------------------------------------------------------
# House Data Preparation
#------------------------------------------------------------------------------------
top_house_data <- house_data %>% filter(Area == top_area)
dim(top_house_data) #--11191   100 #--12
```

```
hprice_min  <- as.integer( min(top_house_data$ListPrice)  / 1000   + 1 ) * 1000
hprice_avg  <- as.integer( mean(top_house_data$ListPrice) / 10000  + 1 ) * 10000
hprice_max  <- as.integer( max(top_house_data$ListPrice)  / 10000  + 1 ) * 10000
hprice_step <- 5000
hprice_min
hprice_avg
hprice_max

taxes_min  <- as.integer( min(top_house_data$Taxes)  / 100 ) * 100
taxes_max  <- as.integer( max(top_house_data$Taxes)  / 100 + 1 ) * 100
taxes_avg  <- as.integer( mean(top_house_data$Taxes) / 100 + 1 ) * 100
taxes_step <- 100
taxes_min
taxes_avg
taxes_max

#-----------------------------------------------------------------------------------
# Rates Data Preparation
#-----------------------------------------------------------------------------------
imin  <- min(rates_data$Rate) * 100
imax  <- max(rates_data$Rate) * 100
istep <- 0.01

#-----------------------------------------------------------------------------------
#==================== Define UI for ShinyApps ========================
#-----------------------------------------------------------------------------------

ui <- fluidPage(theme = shinytheme("flatly"),
  navbarPage(
    "Toronto Housing Market",id = "inTabset",
    #==================== Rental Profit Calculator ====================
    tabPanel("Rental Profit Calculator", icon = icon("chart-line"),
      sidebarPanel(
        selectInput("Area", "Area:", top_area, ""),
        selectInput("Neighbourhood", "Neighbourhood:", top_neighbourhoods, ""),
        radioButtons("Type", "Property Type:", c("Condo", "Detached") ),
        sliderInput("Bedrooms",    "No. of Bedrooms:",  min=1,max=10,value=2,step=1),
        sliderInput("Bathrooms",   "No. of Bathrooms:", min=1,max=10,value=2,step=1)
      ),
      mainPanel(
        h4("Home Listing Data:"),
        DT::dataTableOutput("HouseDataTable")
      ),
      mainPanel(
        HTML('<hr style="height:50px; color:purple;">'),
        h4("Home Prices:"),
        tableOutput("HousePriceTable"),
        h4("Property Taxes:"),
        tableOutput("PropertyTaxTable")
      ),
      sidebarPanel(
        sliderInput("HomePrice",   "Home Price:",
min=hprice_min,max=hprice_max,value=hprice_avg,step=hprice_step),
        sliderInput("PercentDown", "Down Payment (%):", min=5,max=100,value=20,step=5),
        sliderInput("Interest",    "Interest Rate:", min=imin,max=imax,value=imin,step=istep),
        selectInput("Term",        "Length of loan (years):", c(10,15,20,25,30),25),
      ),
      mainPanel(
        HTML('<hr style="height:50px; color:purple;">'),
        h4("Land Transfer Tax:"),
        tableOutput("LandTransferTaxTable"),
        h4("Monthly Mortgage Payment:"),
        tableOutput("MortgagePaymentTable")
      ),
      mainPanel(
        HTML('<hr style="height:50px; color:purple;">'),
        h4("Rental Data:"),
        DT::dataTableOutput("RentalDataTable")
      ),
      sidebarPanel(sliderInput("MonthlyRent", "Monthly Rent:",
                           min=rent_min,max=rent_max,value=rent_avg,step=rent_step)
      ),
      mainPanel(
        HTML('<hr style="height:50px; color:purple;">'),
        h4("Monthly Rental Prices:"),
        tableOutput("RentalPriceTable"),
        h4("Monthly Rental Profit:"),
        tableOutput("RentalProfitTable")
      )
    ),
    #==================== About Calculator ===========================
    tabPanel("About Rental Profit Calculator", icon = icon("cloud"),
      includeHTML("About_Rental_Profit_Calculator.html")
    ),
    #==================== About Interest Rate =========================
    tabPanel("About Interest Rate", icon = icon("table"),
      mainPanel(
        DT::dataTableOutput("InterestRateTable")
      ),
    ),
    #==================== Market Analysis =============================
    tabPanel("Market Analysis", icon = icon("chart-bar"),
      sidebarPanel(
        selectInput(inputId = "visualizeOption",
          label = "Housing Market - Visualization:",
          choices = c("1.Region and House Type effect on Price and Sales Volume",
                      "2.Region and House Type effect on Sold Price Diff-Bubble",
                      "3.Region and House Type effect on Sold Price Diff-Boxplot",
                      "4.Region and House Type effect on Time on Market",
                      "5.Finished / Unfinished Basement effect on Market Share",
                      "6.Finished / Unfinished Basement effect on Sold Price",
                      "7.Age effect on Sales Volume and House Price"
                      ),
          selected =  ""),
      ),
      mainPanel(
        plotOutput(outputId = "plot1", height = "600px")
      )
    )
))

#---------------------------------------------------------------------------------
############### Server code for Rental Profit Calculator ##############################
#---------------------------------------------------------------------------------
#server.R
#---------------------------------------------------------------------------------

#---------------------------------------------------------------------------------
## Install and load the required packages
#---------------------------------------------------------------------------------
if(!require(dplyr))
  install.packages("dplyr")
if(!require(tidyr))
  install.packages("tidyr")
if(!require(corrplot))
  install.packages("corrplot")
if(!require(ggplot2))
  install.packages("ggplot2")
if(!require(ggthemes))
```

```
  install.packages("ggthemes")
if(!require(scales))
  install.packages("scales")

#----------------------------------------------------------------------------------
## Data Preparation for Server for ShinyApps
#----------------------------------------------------------------------------------

#----------------------------------------------------------------------------------
# Read Data
#----------------------------------------------------------------------------------
fullhousedata  <- read.csv("ontario_property_listings.csv")
fullrentaldata <- read.csv("ontario_rental_listings.csv")
rates_data     <- read.csv("ontario_mortgage_rates.csv")

#----------------------------------------------------------------------------------
# Selecting data for visualization
#----------------------------------------------------------------------------------
house <- subset(fullhousedata, fullhousedata$Status=="Sold" &
                    !is.na(fullhousedata$ListPrice) &
                    !is.na(fullhousedata$SoldPrice),

c(Status,Area,Neighbourhood,Type,Rooms,Bathrooms,Bedrooms,ParkingTotal,ListPrice,SoldPri
ce,

Taxes,ListDate,SoldDate,ClosedDate,Basement,FinishedBasement,Garage,Feature,Age))
#View(house)
#check how many records were kept
NROW(house)
NROW(fullhousedata)
NROW(house)/NROW(fullhousedata)*100
#change date format
house$ClosedDate<-as.Date(house$ClosedDate)
house$SoldDate<-as.Date(house$SoldDate)
house$ListDate<-as.Date(house$ListDate)
#View(house)
#Grouping by
houseg<- house %>% select(Status,Area,Type,SoldPrice) %>%
  group_by(Status,Area,Type) %>%
dplyr::summarise(AveragePrice=mean(SoldPrice),count=n())
#View(houseg)

#----------------------------------------------------------------------------------
```

```
# Basement data for visualization
#-------------------------------------------------------------------------------------
sum(is.na(fullhousedata$Basement))/nrow(fullhousedata)
sum(is.na(fullhousedata$Feature))/nrow(fullhousedata)
sum(is.na(fullhousedata$Garage))/nrow(fullhousedata)

housea <- house %>% filter(Type %in% c("Detached","Semi-Detached","Townhouse"))
unique(housea$Basement)

houseb<- housea %>% mutate(PriceDifference=SoldPrice-ListPrice) %>%
  select(Basement,SoldPrice) %>%
  group_by(Basement) %>%
  dplyr::summarise(AverageSoldPrice=mean(SoldPrice),count=n())

housec<- houseb %>%
  filter(Basement %in% c("Apartment","Finished","Part Finished","Unfinished"))
#check how many records were kept
#housec

#-------------------------------------------------------------------------------------
# House Data Preparation
#-------------------------------------------------------------------------------------
house_data <- house %>%
  subset(Type %in% c("Condo", "Detached") &
         !is.na(house$Taxes) &
         !is.na(house$SoldPrice) ) %>%
  mutate(ListPrice  <- as.integer(ListPrice) ) %>%
  mutate(SoldPrice  <- as.integer(SoldPrice) ) %>%
  select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Basement, Taxes,
ListPrice, SoldPrice)

#-------------------------------------------------------------------------------------
# Rental Data Preparation
#-------------------------------------------------------------------------------------
rental_data <- fullrentaldata %>%
  subset(Type %in% c("Condo", "Detached") &
         !is.na(fullrentaldata$Price) ) %>%
  select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Basement, Price)

#-------------------------------------------------------------------------------------
# Helpercode code for Rental Profit Calculator
#-------------------------------------------------------------------------------------
# Function: calculateMortgagepayment
```

```
#---------------------------------------------------------------------------------
calculateMortgagepayment <- function(HomePrice, PercentDown, InterestRate, TermInYears)
{
  if (HomePrice >= 1000000 & PercentDown < 20) {
    PercentDown <- as.integer(20)
  }
  else if (HomePrice > 500000 & PercentDown < 10) {
    MinimumDownPayment <- (500000 * 0.05) + (HomePrice - 500000) * 0.10
    PercentDown     <- MinimumDownPayment / HomePrice * 100
  }

  DownPayment    <- HomePrice * PercentDown / 100
  MortgageAmount <- HomePrice - DownPayment

  if (PercentDown < 10) {
    MortgageInsurance <-   MortgageAmount * 4.00 / 100
  }
  else if (PercentDown < 15) {
    MortgageInsurance <-   MortgageAmount * 3.10 / 100
  }
  else if (PercentDown < 20) {
    MortgageInsurance <-   MortgageAmount * 2.80 / 100
  }
  else {
    MortgageInsurance <- 0.0
  }

  TermInYears <- as.integer(TermInYears)

  if (MortgageInsurance > 0.0 & TermInYears > 25) {
    TermInYears <- as.integer(25)
  }

  P <- MortgageAmount + MortgageInsurance
  i <- InterestRate / 100 / 12
  n <- TermInYears * 12

  M <- P * ( i * (1+i)^n ) / ( (1+i)^n - 1 )

  TotalLoanAmount <- P
  MortgagePayment  <- M

  MortgagePaymentTable <- data.frame(MortgagePayment, DownPayment, MortgageAmount,
                                 MortgageInsurance, TotalLoanAmount,
                                 PercentDown, InterestRate, TermInYears)
  return(MortgagePaymentTable)
}

#----------------------------------------------------------------------------------
#==================== Define Server logic for ShinyApps ===================
#----------------------------------------------------------------------------------

server <- function(input, output) {

#==================== HouseDataTable ====================
output$HouseDataTable <- DT::renderDataTable({

  houseDataTable <- house_data %>%
    filter(Neighbourhood == input$Neighbourhood) %>%
    filter(Area      == input$Area) %>%
    filter(Type      == input$Type) %>%
    filter(Bedrooms  == input$Bedrooms) %>%
    filter(Bathrooms == input$Bathrooms) %>%
    select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Basement, Taxes,
ListPrice, SoldPrice)

  attach(houseDataTable)
  houseDataTable <- houseDataTable[order(ListPrice),]
  detach(houseDataTable)

  houseDataTable <- houseDataTable %>%
    mutate(Taxes       = formatC(Taxes,     format="d", big.mark=",") ) %>%
    mutate(ListPrice   = formatC(ListPrice, format="d", big.mark=",") ) %>%
    mutate(SoldPrice   = formatC(SoldPrice, format="d", big.mark=",") )

  DT::datatable(houseDataTable, options = list(pageLength = 5) )
})

#==================== HousePriceTable ====================
output$HousePriceTable <- renderTable({

  HousePriceSummary <- house_data %>%
    filter(Neighbourhood == input$Neighbourhood) %>%
    filter(Area      == input$Area) %>%
    filter(Type      == input$Type) %>%
    filter(Bedrooms  == input$Bedrooms) %>%
    filter(Bathrooms == input$Bathrooms) %>%
    select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Taxes, ListPrice,
SoldPrice) %>%
    group_by(Area,Neighbourhood,Type) %>%
    dplyr::summarise( NumberOfHomes = n(),
                      MinimumHomePrice = as.integer( min(ListPrice) ),
                      AverageHomePrice = as.integer( mean(ListPrice) ),
                      MaximumHomePrice = as.integer( max(ListPrice) ) )

  attach(HousePriceSummary)
  HousePriceTable <- data.frame(AverageHomePrice, MinimumHomePrice,
MaximumHomePrice, NumberOfHomes )
  detach(HousePriceSummary)

  HousePriceTable <- HousePriceTable %>%
    mutate(AverageHomePrice = formatC(AverageHomePrice, format="d", big.mark=",") ) %>%
    mutate(MinimumHomePrice = formatC(MinimumHomePrice, format="d", big.mark=",") ) %>%
    mutate(MaximumHomePrice = formatC(MaximumHomePrice, format="d", big.mark=",") )
})

#=================== PropertyTaxTable ===================
output$PropertyTaxTable <- renderTable({

  PropertyTaxSummary <- house_data %>%
    filter(Neighbourhood == input$Neighbourhood) %>%
    filter(Area      == input$Area) %>%
    filter(Type      == input$Type) %>%
    filter(Bedrooms  == input$Bedrooms) %>%
    filter(Bathrooms == input$Bathrooms) %>%
    select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Taxes, ListPrice,
SoldPrice) %>%
    group_by(Area,Neighbourhood,Type) %>%
    dplyr::summarise( NumberOfHomes = n(),
                      MinimumPropertyTax = as.integer( min(Taxes) ),
                      AveragePropertyTax = as.integer( mean(Taxes) ),
                      MaximumPropertyTax = as.integer( max(Taxes) ) )

  attach(PropertyTaxSummary)
  PropertyTaxTable <- data.frame(AveragePropertyTax, MinimumPropertyTax,
MaximumPropertyTax, NumberOfHomes )
  detach(PropertyTaxSummary)

  PropertyTaxTable <- PropertyTaxTable %>%
    mutate(AveragePropertyTax = formatC(AveragePropertyTax, format="d", big.mark=",") )
%>%
    mutate(MinimumPropertyTax = formatC(MinimumPropertyTax, format="d", big.mark=",") )
%>%
    mutate(MaximumPropertyTax = formatC(MaximumPropertyTax, format="d", big.mark=",") )
})

#==================== LandTransferTaxTable ====================
output$LandTransferTaxTable <- renderTable({

  HomePrice <- input$HomePrice

  if (HomePrice <= 55000) {
    MLTT <- HomePrice * 0.50 / 100
    PLTT <- MLTT
  }
  else if (HomePrice <= 250000) {
      MLTT <- HomePrice * 1.00 / 100
      PLTT <- MLTT - 275
  }
  else if (HomePrice <= 400000) {
    MLTT <- HomePrice * 1.50 / 100
    PLTT <- MLTT - 1525
  }
  else if (HomePrice <= 2000000) {
    MLTT <- HomePrice * 2.00 / 100
    PLTT <- MLTT - 3525
  }
  else {
    MLTT <- HomePrice * 2.50 / 100
    PLTT <- MLTT - 13525
  }
  LTT <- MLTT + PLTT

  HomePrice = formatC(HomePrice, format="d", big.mark=",")
  TorontoLandTransferTax = formatC(MLTT, format="d", big.mark=",")
  OntarioLandTransferTax = formatC(PLTT, format="d", big.mark=",")
  TotalLandTransferTax   = formatC(LTT,  format="d", big.mark=",")

  LandTransferTaxTable <- data.frame(TotalLandTransferTax, TorontoLandTransferTax,
                                    OntarioLandTransferTax, HomePrice)
})
```

```
#==================== MortgagePaymentTable ====================
output$MortgagePaymentTable <- renderTable({

  HomePrice    <- input$HomePrice
  PercentDown  <- input$PercentDown
  InterestRate <- input$Interest
  TermInYears <- as.integer(input$Term)

  # Calculate mortgage payment
  MortgagePaymentTable <- calculateMortgagepayment(HomePrice, PercentDown,
InterestRate, TermInYears)

  MortgagePaymentTable <- MortgagePaymentTable %>%
    mutate(DownPayment       = formatC(DownPayment,       format="d", big.mark=",") ) %>%
    mutate(MortgageAmount    = formatC(MortgageAmount,    format="d", big.mark=",") ) %>%
    mutate(MortgageInsurance = formatC(MortgageInsurance, format="d", big.mark=",") ) %>%
    mutate(TotalLoanAmount  = formatC(TotalLoanAmount,  format="d", big.mark=",") ) %>%
    mutate(MortgagePayment   = formatC(MortgagePayment,   format="d", big.mark=",") )
})

#==================== InterestRateTable ====================
output$InterestRateTable <- DT::renderDataTable({

  InterestRateTable <- rates_data
  InterestRateTable <- select(InterestRateTable, -ID)

  attach(InterestRateTable)
  InterestRateTable <- InterestRateTable[order(Rate),]
  detach(InterestRateTable)

  InterestRateTable$Rate <- InterestRateTable$Rate * 100
  colnames(InterestRateTable)[colnames(InterestRateTable) == "Rate"] <- "RateInPercent"
  colnames(InterestRateTable)[colnames(InterestRateTable) == "Term"] <- "TermInYears"

  DT::datatable(InterestRateTable, options = list(pageLength = 10) )
})

#==================== RentalDataTable ====================
output$RentalDataTable <- DT::renderDataTable({

  rentalDataTable <- rental_data %>%
    filter(Neighbourhood == input$Neighbourhood) %>%
    filter(Area      == input$Area) %>%
    filter(Type      == input$Type) %>%
    filter(Bedrooms  == input$Bedrooms) %>%
    filter(Bathrooms == input$Bathrooms) %>%
    mutate(Price = formatC(Price, format="d", big.mark=",") ) %>%
    select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Basement, Price)

  attach(rentalDataTable)
  rentalDataTable <- rentalDataTable[order(Price),]
  detach(rentalDataTable)

  DT::datatable(rentalDataTable, options = list(pageLength = 5) )
})

#==================== RentalPriceTable ====================
output$RentalPriceTable <- renderTable({

  RentalPriceSummary <- rental_data %>%
    filter(Neighbourhood == input$Neighbourhood) %>%
    filter(Area      == input$Area) %>%
    filter(Type      == input$Type) %>%
    filter(Bedrooms  == input$Bedrooms) %>%
    filter(Bathrooms == input$Bathrooms) %>%
    select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Basement, Price) %>%
    group_by(Area,Neighbourhood,Type) %>%
    dplyr::summarise( NumberOfRentals = n(),
                      MinimumRentPrice = as.integer( min(Price) ),
                      AverageRentPrice = as.integer( mean(Price) ),
                      MaximumRentPrice = as.integer( max(Price) ) )

  attach(RentalPriceSummary)
  RentalPriceTable <- data.frame(AverageRentPrice, MinimumRentPrice, MaximumRentPrice,
NumberOfRentals)
  detach(RentalPriceSummary)

  RentalPriceTable <- RentalPriceTable %>%
    mutate(AverageRentPrice = formatC(AverageRentPrice, format="d", big.mark=",") ) %>%
    mutate(MinimumRentPrice = formatC(MinimumRentPrice, format="d", big.mark=",") ) %>%
    mutate(MaximumRentPrice = formatC(MaximumRentPrice, format="d", big.mark=",") )
})

#==================== RentalProfitTable ====================
output$RentalProfitTable <- renderTable({
  RentalPriceSummary <- rental_data %>%
    filter(Neighbourhood == input$Neighbourhood) %>%
    filter(Area      == input$Area) %>%
    filter(Type      == input$Type) %>%
    filter(Bedrooms  == input$Bedrooms) %>%
    filter(Bathrooms == input$Bathrooms) %>%
    select(Area, Neighbourhood, Type, Rooms, Bedrooms, Bathrooms, Basement, Price) %>%
    group_by(Area,Neighbourhood,Type) %>%
    dplyr::summarise( NumberOfRentals = n(),
                      MinimumRentPrice = as.integer( min(Price) ),
                      AverageRentPrice = as.integer( mean(Price) ),
                      MaximumRentPrice = as.integer( max(Price) ) )

  HomePrice    <- input$HomePrice
  PercentDown  <- input$PercentDown
  InterestRate <- input$Interest
  TermInYears <- as.integer(input$Term)

  # Calculate mortgage payment
  MortgagePaymentTable <- calculateMortgagepayment(HomePrice, PercentDown,
InterestRate, TermInYears)

  MonthlyRent    <- input$MonthlyRent
  MonthlyProfit  <- MonthlyRent - MortgagePaymentTable$MortgagePayment

  MonthlyRent    = formatC(MonthlyRent,   format="d", big.mark=",")
  MonthlyProfit  = formatC(MonthlyProfit, format="d", big.mark=",")

  RentalProfitTable <- data.frame(MonthlyRent, MonthlyProfit)
})

#=================== Market Analysis Tab ===================
output$plot1 <- reactivePlot(function() {

#-----------------------------------------------------------------------------------
# Plotting chart based on input option
#-----------------------------------------------------------------------------------

#-----------------------------------------------------------------------------------
#1. Bubble plot - Region and House Type effect on House Price and Sales Volume
#-----------------------------------------------------------------------------------
if (input$visualizeOption == "1.Region and House Type effect on Price and Sales Volume") {
plot1 <- ggplot(houseg, aes(x=Area, y=AveragePrice, size = count,color=Type)) +
   geom_point(alpha=2)+
   scale_size(range = c(.1, 24), name="Sales Count")+
   ylim(200000,1500000)
}

#--------------------------------------------------------------------------------
#2. Bubble plot - Region and House Type effect on Sold Price Diff
#--------------------------------------------------------------------------------
if (input$visualizeOption == "2.Region and House Type effect on Sold Price Diff-Bubble") {
houseg1 <- house %>% mutate(PriceDifference=SoldPrice-ListPrice) %>%
  select(Status,Area,Type,PriceDifference) %>%
  group_by(Status,Area,Type) %>%
  dplyr::summarise(AveragePriceDifference=mean(PriceDifference),count=n())
plot1 <- ggplot(houseg1, aes(x=Area, y=AveragePriceDifference, size = count,color=Type)) +
   geom_point(alpha=2)+
   scale_size(range = c(.1, 24), name="Sales Count")+
   ylim(-25000,100000)
}

#--------------------------------------------------------------------------------
#3. Box plot - Region and House Type effect on Sold Price Diff
#--------------------------------------------------------------------------------
if (input$visualizeOption == "3.Region and House Type effect on Sold Price Diff-Boxplot") {
houseg11 <- house %>% mutate(PriceDifference=SoldPrice-ListPrice) %>%
  select(Status,Area,Type,PriceDifference) %>%
  filter(Type %in% c("Semi-Detached","Detached","Condo"))%>%
  filter(Area %in% c("Toronto","York","Peel","Halton","Durham"))
plot1 <- ggplot(data=houseg11,aes(x=Area,y=PriceDifference/1000,fill=Type))+
  geom_boxplot()+ylim(-100,100)
}

#--------------------------------------------------------------------------------
#4. Bar chart - Region and House Type effect on Time on Market
#--------------------------------------------------------------------------------
if (input$visualizeOption == "4.Region and House Type effect on Time on Market") {
houseg2 <- house %>% mutate(DateDiff=difftime(SoldDate, ListDate, units = "days")) %>%
  filter(Type %in% c("Detached","Semi-Detached","Condo")) %>%
  filter(Area %in% c("Toronto","York","Peel","Hamilton","Halton","Durham"))%>%
  select(Status,Area,Type,DateDiff) %>% group_by(Status,Area,Type) %>%
  dplyr::summarise(AverageDateDiff=mean(DateDiff),count=n())
#houseg2[order(houseg2$AverageDateDiff),]
#View(houseg2)
plot1 <- ggplot(houseg2,aes(fill=Type,x=Area,y=AverageDateDiff)) +
    geom_bar(position="dodge", stat="identity") +
    ggtitle("Average Date Diff") +
    xlab("Area") +
    ylab("Average Date Diff") +
    theme_bw()
}

#--------------------------------------------------------------------------------
#5. Pie chart - Finished / Unfinished Basement effect on Market Share
#--------------------------------------------------------------------------------
if (input$visualizeOption == "5.Finished / Unfinished Basement effect on Market Share") {
plot1 <- ggplot(housec, aes(x="",y=count, fill=Basement))+
  geom_bar(stat="identity",width = 1,color="white") +
  coord_polar("y",start=0)+
  theme_void()
}

#--------------------------------------------------------------------------------
#6. Bar chart - Finished / Unfinished Basement effect on Sold Price
#--------------------------------------------------------------------------------
if (input$visualizeOption == "6.Finished / Unfinished Basement effect on Sold Price") {
plot1 <- ggplot(housec,aes(x=Basement,y=AverageSoldPrice,fill=Basement)) +
  geom_bar(position="dodge", stat="identity") +
  scale_y_continuous(breaks=seq(0,1000000,250000)) +
  xlab("Basement") +
  ylab("Average Sold Price") +
  ggtitle("Average Sold Price") +
  theme_bw()
}

#--------------------------------------------------------------------------------
#7. Age effect on Sales Volume and House Price
#--------------------------------------------------------------------------------
if (input$visualizeOption == "7.Age effect on Sales Volume and House Price") {
houseage<- house %>% select(Age,Type,SoldPrice) %>%
  group_by(Age,Type) %>% dplyr::summarise(AveragePrice=mean(SoldPrice),count=n())

houseage1<- house %>% select(Age,SoldPrice) %>%
  group_by(Age) %>% dplyr::summarise(AveragePrice=mean(SoldPrice),count=n())
houseage2<-na.omit(houseage1)

coeff <- 1000
plot1 <- ggplot(houseage2, aes(Age)) +
  geom_line(aes(y=count), colour="blue") +
  geom_line(aes(y=AveragePrice/coeff), colour="red") +
  scale_y_continuous(name = "Sales Count",
           sec.axis = sec_axis(~.*coeff, breaks =seq(0,4000000,1250000), name = "Average
Price"))
}

#Return the Plot chart
#-------------------------------------------------------------------------------------
print(plot1)
}

)}

#-------------------------------------------------------------------------------------
##################################### END #####################################
#-------------------------------------------------------------------------------------
```