Part A:

1.

code:

awk 'NR<=6 || NR>(total-5)' total=$(wc -l < consumer_complaints.csv) consumer_complaints.csv

answer:

```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ awk 'NR<=6 || NR>(total-5)' total=$(wc -l < consumer_complaints.csv) consumer_complaints.csv
"Date_received","Product","Sub_product","Issue","Sub_issue","Consumer_complaint_narrative","Company_public_response","Company","State",
"ZIP_code","Tags","Consumer_consent_provided","Submitted_via","Complaint_ID"
2015-08-09,"Credit reporting",NA,"Incorrect information on credit report","Information is not mine",NA,"Company chooses not to provide
a public response","Experian Information Solutions Inc.","NJ","08872",NA,"Consent not provided","Web",1509954
2019-12-23,"Student loan","Federal student loan servicing","Dealing with your lender or servicer","Trouble with how payments are being
handled",NA,NA,"AES/PHEAA","MA","019XX",NA,NA,"Web",3475943
2019-01-29,"Credit reporting  credit repair services  or other personal consumer reports","Credit reporting","Problem with a credit rep
orting company's investigation into an existing problem","Was not notified of investigation status or results",NA,NA,"EQUIFAX  INC.","N
Y","10801",NA,"Consent not provided","Web",3136759
2015-08-19,"Mortgage","Conventional adjustable mortgage (ARM)","Loan servicing  payments  escrow account",NA,NA,"Company chooses not to
 provide a public response","WELLS FARGO & COMPANY","CA","94526",NA,"Consent not provided","Web",1527601
2016-03-04,"Credit card",NA,"Billing disputes",NA,"I am dissatisfied with the current outcome of a dispute that was initiated with Disc
over Card regarding a single transaction that occurred on XXXX/XXXX/2015 in the amount of {$280.00}. I have corresponded with Discover
Card at least four times since XXXX/XXXX/2015 ( which I have enclosed as an attachment to this complaint ). I believe that the credit c
ard issuer has violated consumer protection laws by failing to implement the Special Rule for Credit Card Purchase protection despite o
verwhelming paperwork evidence submitted by me that shows the merchant has conducted business in bad faith less favorable to the consum
er. I have sustained a monetary loss as a result of merchants bad faith and intent. I have patiently utilized the internal Discover Car
d dispute process over the past three months with the credit card issuer always favoring the merchant ; I have repeatedly submitted irr
efutable paperwork evidence that has shown that the merchant has conducted business in bad faith. I have tried in good faith to address
 my complaint with the merchant and Discover Card but believe that I will not receive a favorable outcome.",NA,"DISCOVER BANK","NV","89
1XX",NA,"Consent provided","Web",1816726
2016-01-29,"Mortgage","Conventional fixed mortgage","Application  originator  mortgage broker",NA,NA,"Company believes it acted appropr
iately as authorized by contract or law","American Financial Network Inc.","CA","91321",NA,"Consent not provided","Web",1765368
2019-11-27,"Debt collection","Other debt","Attempts to collect debt not owed","Debt was paid",NA,NA,"Diversified Consultants  Inc.","WI
","531XX",NA,NA,"Web",3452575
2019-05-07,"Mortgage","Conventional home mortgage","Struggling to pay mortgage",NA,"Hello  We were approved for a loan modification  we
 did trial payments and what ever was requested this all occurred in XXXX. Loan modification documents to finalize were sent to our hom
e in XX/XX/2019. We never received so I contacted Ocwen early XX/XX/2019 and advised us would have a packet remailed. We did receive th
em and signed and returned. I did fail not to register agreement  sent through regular mail. I attempted to make my payment on XXXX and
 payment could not go through. Contacted Ocwen and was advised my loan modification was denied because signed documents were never rece
ived  I was also advised to provide a tracking # in which I do not have because I did not register documents. Was advised had to submit
 all paperwork over again and had to wait to see what outcome is! Meanwhile I have a default notice. I am asking for your assistance  I
 think its unfair in how our situation is being handled. It also makes no sense! We have struggled so much to keep our house and just n
eed help so that this gets resolved. Can someone please please get back to me? I am afraid my home can go into foreclosure and we dont
want to loose our home. I appreciate the time you are taking in reading my email and hope to hear from someone soon!",NA,"Ocwen Financi
al Corporation","CA","951XX",NA,"Consent provided","Web",3235219
2019-07-18,"Mortgage","Conventional home mortgage","Closing on a mortgage",NA,"I started the process to refinance my current mortgage.
The closing lawyer attempted to obtain the payoff statement  but they would not provide that statement for 4 business as they claimed t
hey are behind. The lawyer informed me I could obtain the information sooner. I called and spoke to a representative on XX/XX/XXXX who
said she could request I receive the payoff statement in 24 hours if she requested an expedite through a supervisor. I called back on X
X/XX/XXXX  my closing date on the refinance  to determine the payoff amount as I had not yet received the payoff statement. I was told
that it had not yet been generated  but still could be done today. I asked to speak to a supervisor and was told that one was not avail
able and could call me within 24 hours. They also told me that there was no one within the company that could help me with this inquiry
.","Company has responded to the consumer and the CFPB and chooses not to provide a public response","Freedom Mortgage Company","NC","2
75XX",NA,"Consent provided","Web",3311105
2019-11-13,"Student loan","Federal student loan servicing","Dealing with your lender or servicer","Trouble with how payments are being
handled",NA,NA,"Nelnet  Inc.","IN","476XX",NA,NA,"Web",3437084
```

Explanation:

awk 'NR<=6 || NR>(total-5) :

If the current action is in the first 6 or last 5 lines, the action is executed

total=$(wc -l < consumer_complaints.csv) :

Gets the total number of lines in the file

The console outputs the contents of rows that meet these criteria

2.

code:

stat -c '%s' consumer_complaints.csv | numfmt --to=iec --suffix=B --format='%.0f' && wc -l < consumer_complaints.csv

answer:

```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ stat -c '%s' consumer_complaints.csv | numfmt --to=iec --suffix=B --format='%.0f' && wc -l < consumer_complaints.csv
832MB
1471767
```

Explanation:

stat -c '%s' consumer_complaints.csv:

Used to get the size of the consumer_complaints.csv file and output it in bytes

| :The output of the previous command is passed to the next command as input

numfmt --to=iec --suffix=B --format='%.0f': Is formatted file size

&& : Execute the current command after it succeeds

wc -l < consumer_complaints.csv: Calculate the number of lines in the consumer_complaints.csv file

The final output is the file size and the number of lines

3.

code:

head -n 1 consumer_complaints.csv | tr ',' '\n' | wc -l && head -n 1 consumer_complaints.csv | tr ',' '\n'

answer:

```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ head -n 1 consumer_complaints.csv | tr ',' '\n' | wc -l &&  head -n 1 consumer_complaints.csv | tr ',' '\n'
14
"Date_received"
"Product"
"Sub_product"
"Issue"
"Sub_issue"
"Consumer_complaint_narrative"
"Company_public_response"
"Company"
"State"
"ZIP_code"
"Tags"
"Consumer_consent_provided"
"Submitted_via"
"Complaint_ID"
```

Explanation:

Run the Head -n 1 consumer_complaints.csv command to obtain the first line of the consumer_complaints.csv file

tr ',' \n' This part of the command is used to replace commas (,) with newlines (\n)

wc -l command is used to count the number of rows entered

First get the first line of the consumer_complaints.csv file, then replace the comma with a newline, and finally count the number of replaced lines

4.

code:

awk -F "," 'NR > 1 && $1 != "NA" && $1 != "" { dates[$1] = 1 } END { for (date in dates) print date }' consumer_complaints.csv | sort | { head -n 1; tail -n 1; } | paste -s -d ' '

answer:

```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ awk -F "," 'NR > 1 && $1 != "NA" && $1 != "" { dates[$1] = 1 } END { for (date in dates) print date }' consumer_complaints.csv | sort | { head -n 1; tail -n 1; } | paste -s -d ' '
2011-12-01 2020-01-07
```

Explanation:

The -F "," option specifies that the field separator is comma.

NR > 1 && $1! = "NA" && $1 ! = "" The filter line number is greater than 1, and the value of the first field is not equal to "NA" or null

{dates[$1] = 1} takes the first field of the row that meets the condition as the index, sets the value to 1, and builds an array of dates for recording unduplicated dates

END {for (date in dates) print date} After processing the file, iterate through the set of dates, printing all the dates

{ head -n 1; tail -n 1; } This section of the command is used to get the first and last lines in the sorted result.

paste -s -d "" The paste -s -d" command is used to combine multiple lines of text into one line and separate them with Spaces

First extract the non-duplicate dates from the consumer_complaints.csv file and sort them in lexicographical order. Then get the earliest date and the latest date from the sorting result; Finally, combine the two dates into one line, separated by a space

5.

code:

count=$(cut -d ',' -f 6 consumer_complaints.csv | grep -i -n -m 1 "Student loan" | cut -d ':' -f 1) && total=$(cut -d ',' -f 6 consumer_complaints.csv | grep -i -o "Student loan" | wc -l) && echo "First occurrence position: $count" && echo "Total occurrences: $total"

answer:

```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ count=$(cut -d ',' -f 6 consumer_complaints.csv | grep -i -n -m 1 "Student loan" | cut -d ':' -f 1) && total=$(cut -d ',' -f 6 consum
er_complaints.csv | grep -i -o "Student loan" | wc -l) && echo "First occurrence position: $count" && echo "Total occurrences: $total"
First occurrence position: 251
Total occurrences: 31651
```

Explanation:

count=$(cut -d ',' -f 6 consumer_complaints.csv | grep -i -n -m 1 "Student loan" | cut -d ':' -f 1) This section of the command is used to get the location where "Student loan" first appears

total=$(cut -d ',' -f 6 consumer_complaints.csv | grep -i -o "Student loan" | wc -l) This command is used to collect Student statistics The total number of times loan appears in the sixth field

echo "First occurrence position: $count" && echo "Total occurrences: $total" The echo "first occurrence position: $count" && echo" total occurrences: $total" command outputs the result of the

first occurrence and total occurrences
Extract the information containing "Student loan" from the sixth field of the consumer_complaints.csv file, count the location and total number of occurrences for the first time, and output the result

6.

code:

{ tail -n +2 consumer_complaints.csv | grep -v '^Date received' | cut -d ',' -f 2 | awk '!/^NA$/ && $0 != ""' | sort | uniq -c | sort -t ' ' -k 1 -n -r | head -n 5 ; tail -n +2 consumer_complaints.csv | grep -v '^Date received' | cut -d ',' -f 2 | awk '!/^NA$/ && $0 != ""' | sort | uniq | wc -l ; }

answer:

```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ { tail -n +2 consumer_complaints.csv | grep -v '^Date received' | cut -d ',' -f 2 | awk '!/^NA$/ && $0 != ""' | sort | uniq -c | sort
 -t ' ' -k 1 -n -r | head -n 5 ; tail -n +2 consumer_complaints.csv | grep -v '^Date received' | cut -d ',' -f 2 | awk '!/^NA$/ && $0 !
= ""' | sort | uniq | wc -l ; }
 322817 "Credit reporting  credit repair services  or other personal consumer reports"
 293191 "Mortgage"
 275584 "Debt collection"
 140432 "Credit reporting"
  89190 "Credit card"
18
```

Explanation:

The tail -n +2 consumer_complaints.csv command is used to extract the content starting from line 2 of the consumer_complaints.csv file

The grep -v '^Date received' command is used to filter out lines containing "Date received", that is, skip the header line in the file

The cut -d ',' -f2 command is used to extract the second field, using a comma as the field separator

awk '! /^NA$/ && $0 ! The = "" command is used to filter out lines with a value of "NA" or an empty string

The sort command is used to sort the input.

The uniq -c command is used to count and remove repeated rows and display the number of repeated rows in front of each row

sort -t ''-k 1 -n -r Used to sort the first field (number of repetitions) in reverse order

The head -n 5 command is used to extract the first five rows of the sorting result, that is, the top five values that occur most frequently

The uniq | wc -l command It is used to collect statistics on the number of non-duplicate values

Get the value of the second field from the consumer_complaint.csv file, filter, sort, count, and extract it to get the top 5 most frequent non-duplicate values, and calculate the total number of non-duplicate values

7.

code:

awk -F ',' 'NR > 1 && $6 != "NA" && $6 != "" {count_fraud += gsub(/fraud/, "&", $6); count_account += gsub(/account/, "&", $6); if ($6 ~ /fraud/ && $6 ~ /account/) count_both++} END {print "Count of 'fraud':", count_fraud; print "Count of 'account':", count_account; print "Count of Rows with 'fraud' and 'account':", count_both}' consumer_complaints.csv

answer:

```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ awk -F ',' 'NR > 1 && $6 != "NA" && $6 != "" {count_fraud += gsub(/fraud/, "&", $6); count_account += gsub(/account/, "&", $6); if ($
6 ~ /fraud/ && $6 ~ /account/) count_both++} END {print "Count of 'fraud':", count_fraud; print "Count of 'account':", count_account; p
rint "Count of Rows with 'fraud' and 'account':", count_both}' consumer_complaints.csv
Count of fraud: 119303
Count of account: 710697
Count of Rows with fraud and account: 40490
```

Explanation:

awk -F ',' Sets the field separator to comma to specify the format of the csv file.

NR > 1 & $6! = "NA" && $6 ! = "" This condition is used to filter out the first row (header) as well as rows where the sixth field is "NA" or an empty string.

{count_fraud += gsub(/fraud/, "&", $6); count_account += gsub(/account/, "&", $6); if ($6 ~ /fraud/ && $6 ~ /account/) count_both++} This section operates on rows that satisfy the condition

The END block means to do the following after all rows have been processed.

print "Count of 'fraud':", count_fraud The number of times the keyword "fraud" is printed.

print "Count of 'account':", count_account The number of times the output contains the keyword "account".

print "Count of Rows with 'fraud' and 'account':", count_both Print "count of rows with 'fraud' and

'account':", count_both
Count the number of keyword "fraud" and "account" in the sixth field of the consumer_complaints.csv file, and output the corresponding statistical result. Finally, the output shows the number of lines containing "fraud", the number of lines containing "account", and the number of lines containing both "fraud" and "account"
8.
code:
```
awk -F ',' 'NR > 1 && $4 != "NA" && $4 != "" && tolower($4) ~ /account/ {gsub(/"/, "", $9);
count[$9]++} END {for (state in count) {print "State:", state, "Count:", count[state]}}'
consumer_complaints.csv | sort -k4,4nr | head -n 1
```
answer:
```
15335@LAPTOP-S7VLOUDT /cygdrive/c/Users/15335/Downloads/consumer_complaints
$ awk -F ',' 'NR > 1 && $4 != "NA" && $4 != "" && tolower($4) ~ /account/ {gsub(/"/, "", $9); count[$9]++} END {for (state in count) {p
rint "State:", state, "Count:", count[state]}}' consumer_complaints.csv | sort -k4,4nr | head -n 1
State: CA Count: 27225
```
Explanation:
awk -F ',' Sets the field separator to comma to specify the format of the csv file.
NR > 1 && $4! = "NA" && $4 ! = "" &&tolower ($4) ~ /account/ This condition is used to filter out the first row (header) and rows with the fourth field being "NA" or an empty string, and the fourth field containing the "account" keyword after case is ignored.
{gsub(/"/, "", $9); count[$9]++} This section operates on rows that satisfy the condition.
gsub(/"/, "", $9) is used to remove double quotes from the 9th field.
count[$9]++ Counts the number of occurrences of each state using the value of the 9th field as the key.
The END block means to do the following after all rows have been processed.
for (state in count) {print "State:", state, "Count:", count[state]} Loops through each state with a for loop, printing the state and the corresponding number of occurrences.
sort-k4,4nr Sorts the output by the fourth field (number of occurrences) in reverse order.
-k4,4nr Indicates that the sorting key is the fourth field. nr indicates that the sorting is in reverse order.
head-n 1 Retrieves the first row of the sorted result, that is, the state that occurs the most times.
Count the number of occurrences of the fourth field (status field) that contains the keyword "account" in the consumer_complaints.csv file, and output the status with the highest number of occurrences and the corresponding occurrence times
9.
code:
```
awk -F',' 'NR == 1 || (substr($1,1,4) > 2015 && $9 ~ /FL/ && $13 ~ /Web/ && $2 ~ /Mortgage/) {print
$1 "," $2 "," $6 "," $9 "," $13}' consumer_complaints.csv > filtered_narratives.csv ; echo "remainder
is: $(wc -l < filtered_narratives.csv)" ; head -n 6 filtered_narratives.csv
```
answer:
```
remainder is: 7204
"Date_received","Product","Consumer_complaint_narrative","State","Submitted_via"
2019-01-23,"Mortgage",NA,"FL","Web"
2017-04-11,"Mortgage",NA,"FL","Web"
2018-07-02,"Mortgage",NA,"FL","Web"
2016-01-07,"Mortgage","In a letter dated XXXX/XXXX/XXXX  the bank ( falsely ) claims that the entire forgiven amount {$150000.00} was p
rincipal. That amount as principal is not possible -- based on the declared principal at the time of my original purchase and the new p
rincipal due per the work out plan.  The bank  under the direction of the Federal Bankruptcy court for mediation  finally offered me a
workout plan based on the bank 's determination of the current value of my home. To my knowledge  there was NO independent appraisal. T
hat mediated agreement was signed XXXX XXXX XXXX.   The details for that FUTURE workout plan included the bank 's claim that I owed {$2
90000.00}. I do not know how they made up this number. There was never a time when I owed such an amount as principal.   The original n
ote I signed in XXXX was for a principal amount of {$180000.00}. The new principal amount from the XXXX agreement is {$120000.00} ( int
erest bearing ) + {$13000.00} ( non-interest bearing ).   That is a difference of {$49000.00} in principal. Where the {$150000.00} clai
med as PRINCIPAL by the bank came from is a complete mystery. It seems the bank can make up whatever numbers they like and make whateve
r claims they wish and too bad for the consumer.   First  I 'm told by the bank contact provided that I will have an answer by XXXX XXX
X. -- to the simple question I asked in XXXX.   Then  I receive a notice that it will be XXXX XXXX.   Today  I received a notice today
that the bank will contact me sometime within 60 days.   It took the bank TWO months to send a notice that they will respond sometime i
n the next TWO Months? What then? Another notice that the bank will take another 6 months to send another notice?   This is not reasona
ble. The bank has the information -- they had it when the provided the work-out plan -- in XXXX XXXX and they had it when they sent the
 confirming paperwork in XXXX of XXXX and they had it when they sent the final paperwork in XXXX of XXXX. The bank is refusing to discl
ose essential financial information to me - the consumer.   What purpose does it serve to not provide me with the proper  lawful disclo
sures of financial information? I did NOT ask to renegotiate or change anything -- I simply requested the specific numbers for principa
l  interest  insurance  and taxes.","FL","Web"
2018-06-06,"Mortgage",NA,"FL","Web"
```
Explanation:
awk -F',' Sets the field separator to comma to specify the format of the csv file.
The condition NR == 1 indicates that the first row (header) is processed and the header information is preserved.
(substr($1,1,4) > 2015 && $9 ~ /FL/ && $13 ~ /Web/ && $2 ~ /Mortgage/) This condition is used

to filter the eligible rows.
substr($1, 4) > 2015 means that the first 4 characters of the first field (the year part) are extracted and compared to 2015, keeping only the lines larger than 2015.
$9 ~ /FL/ Indicates the line with "FL" in the 9th field (state code field).
$13 ~ /Web/ Indicates the line with "Web" in the 13th field (Complaint Source field).
$2 ~ /Mortgage/ Indicates the row where the second field (product field) contains "Mortgage".
{print $1 "," $2 "," $6 "," $9 "," $13} operates on the rows that meet the condition, extracts the fields 1, 2, 6, 9, and 13, and prints them in the specified format.
consumer_complaints.csv > filtered_narratives.csv Saves the extracted results to the filtered_narratives.csv file.
echo "remainder is: $(wc -l < filtered_narratives.csv)" Prints the number of lines in the filtered_narratives.csv file, that is, the number of remaining lines.
head -n 6 filtered_narratives.csv Print the first six lines of the filtered_narratives.csv file to display the extraction result.

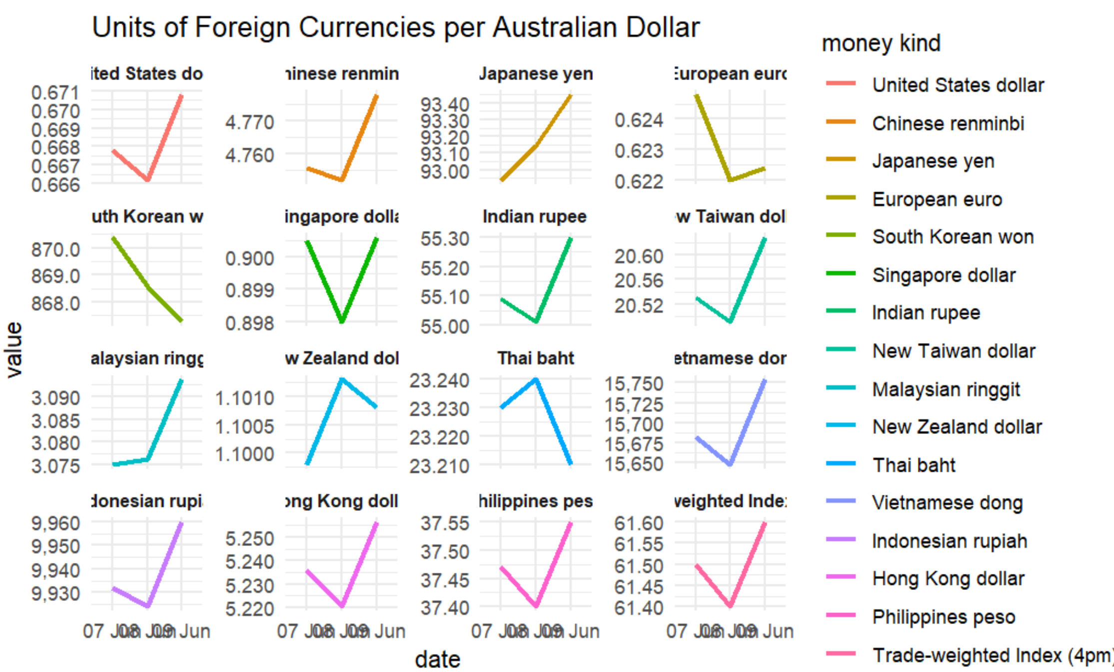
Part B:

code:

```
# Load required packages
library(ggplot2)
library(dplyr)
library(lubridate)
library(rvest)
library(stringr)
library(reshape2)
# Load HTML content
html <- read_html("https://www.rba.gov.au/statistics/frequency/exchange-rates.html")
# Locate table element using CSS selector or XPath expression
table <- html %>% html_node("table")  # CSS selector
# Alternatively,
# table <- html %>% html_nodes(xpath = "//table")  # XPath expression
# Extract table data
data <- table %>% html_table(fill = TRUE)
# Convert data to DataFrame
df <- data.frame(data)
df <- t(df)
df[1, 1] <- 'date'
# Extract the first row of data
col_names <- df[1, ]
# Set the first row as column names
colnames(df) <- col_names
# Convert matrix or array to data frame
df <- as.data.frame(df)
# Remove the first row
df <- df[-1, ]
# Get the current number of rows
num_rows <- nrow(df)
# Set new row names as integer sequence
new_row_names <- 1:num_rows
# Assign the new row names to the data frame
rownames(df) <- new_row_names
# Convert date format
df1 <- melt(df, id = "date")
colnames(df1) <- c('date', 'money_kind', 'value')
# Convert the 'value' column to numeric
df1$value <- as.numeric(df1$value)
# Select the top 16 currency types
top16_currencies <- df1 %>%
```

```
  group_by(money_kind) %>%
  summarise(total_value = sum(value)) %>%
  top_n(16, total_value) %>%
  arrange(desc(total_value)) %>%
  pull(money_kind)
# Filter out the data for the top 16 currencies
df_top16 <- df1 %>%
  filter(money_kind %in% top16_currencies)
df_top16$date <- str_sub(df_top16$date, start = 1, end = 6)
# Create a line plot, displayed in a 4x4 panel, and apply logarithmic scale to the y-axis
ggplot(df_top16, aes(x = date, y = value, colour = money_kind, group = money_kind)) +
  geom_line(size = 1) +
  labs(x = "date", y = "value", colour = "money kind", title = "Units of Foreign Currencies per Australian Dollar") +
  theme_minimal() +
  facet_wrap(~ money_kind, nrow = 4, ncol = 4, scales = "free_y") +
  theme(strip.text = element_text(size = 8, face = "bold")) +
  scale_y_continuous(trans = "log10", labels = scales::comma)  # Use English labels with comma as the thousands separator
```

answer:

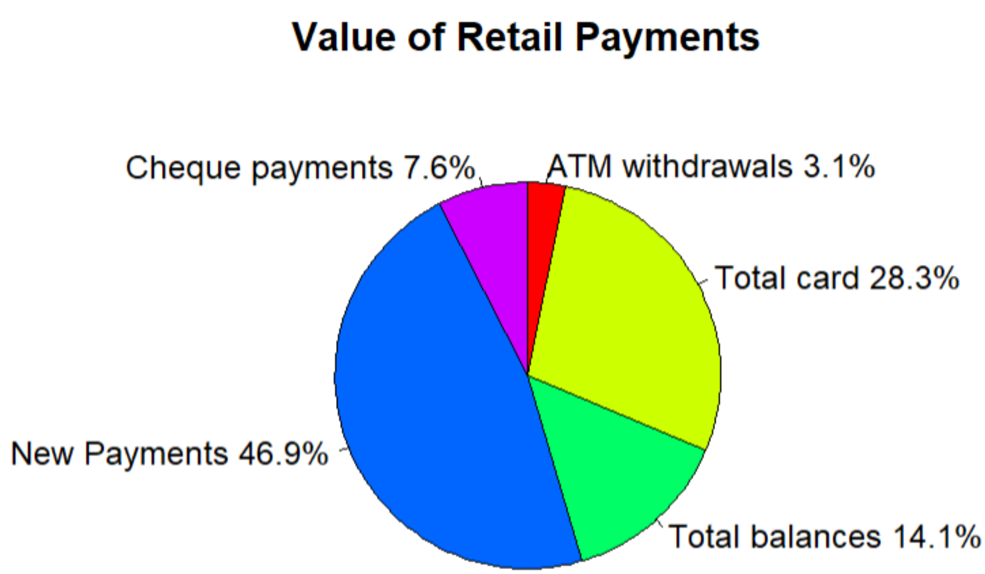


Code:

```
# Load HTML content
html <- read_html("https://www.rba.gov.au/statistics/frequency/retail-payments/2023/retail-payments-0423.html")
# Use XPath expressions to locate table elements
table <- html %>% html_nodes(xpath = "//table")
# Extract table data
data <- table %>% html_table(fill = TRUE)
# Convert the data to a DataFrame
data <- data[[1]]
data <- data[-nrow(data), ]
data <- data[, 1:2]
data <- data[-c(1:3), ]
```

```
colnames(data) <- c("Value of Retail Payments", "Value ($ billion)")
data$`Value ($ billion)` <- as.numeric(gsub("[^0-9.]", "", data$`Value ($ billion)`))
data <- data[!grepl("of which", data$`Value of Retail Payments`), ]
data$Value <- gsub("^([[:alpha:]]+\\s+[[:alpha:]]+).*", "\\1", data$`Value of Retail Payments`)
# Generated pie chart
library(scales)  # The scales package is loaded to format percentages
percent_values <- percent(data$`Value ($ billion)` / sum(data$`Value ($ billion)`))
pie(data$`Value ($ billion)`,
    labels = paste(data$Value, percent_values),
    col = rainbow(length(data$`Value ($ billion)`)),
    main = "Value of Retail Payments",
    clockwise = TRUE)
```

answer:



Explanation:
First, I loaded some necessary packages, including ggplot2, dplyr, lubrication, vest, and stringr. These packages provide the ability to manipulate data, parse HTML, manipulate strings, and draw graphs.

Use the read_html function to load HTML content as a processable HTML object.
Use CSS selectors or XPath expressions to locate table elements. The table element is obtained through the html_node function.
Use the html_table function to extract table data.
Converts the extracted data to a data frame DataFrame.
Perform a series of processing operations on the data frame, including setting column names, deleting rows, transposing data, processing date formats, and so on.
The top 16 currency categories were selected based on criteria.
Create a linear plot, draw curves by date and currency class, and apply a logarithmic scale on the Y-axis.
Use XPath expressions to locate table elements.
Extract table data.
Converts data into data frames.

Perform a series of processing operations on the data frame, including deleting rows, processing column names, converting data types, and so on.
Generate a pie chart showing retail payments
Part C:
Task 1
Code:

```
library(tidyverse)
library(ggplot2)
library(scales)
# Read the CSV file named "ptv_data".
ptv_data <- read.csv('property_transaction_victoria.csv')

# Check dimensions (number of rows and columns)
dim(ptv_data)

# Display the structure of the dataframe
str(ptv_data)

# Check the unique values in the "state" column
unique(ptv_data$state)

# Only the data in the Vic state is retained
ptv_data_vic <- filter(ptv_data, state == "Vic")

# Check the unique values in the "state" column after filtering
unique(ptv_data_vic$state)
```

Answer:

```
[1] 836781      28
'data.frame':   836781 obs. of  28 variables:
 $ id                  : int  141077012 141774568 141373204 141606240 141603340 141602852 141632592 141276588
140369299 141416144 ...
 $ badge               : chr  "Sold" "Sold" "Sold" "Sold" ...
 $ url                 : chr  "https://www.realestate.com.au/sold/property-apartment-vic-abbotsford-141077012"
"https://www.realestate.com.au/sold/property-townhouse-vic-abbotsford-141774568"
"https://www.realestate.com.au/sold/property-apartment-vic-abbotsford-141373204"
"https://www.realestate.com.au/sold/property-house-vic-abbotsford-141606240" ...
 $ suburb              : chr  "Abbotsford" "Abbotsford" "Abbotsford" "Abbotsford" ...
 $ state               : chr  "Vic" "Vic" "Vic" "Vic" ...
 $ postcode            : int  3067 3067 3067 3067 3067 3067 3067 3067 3067 3067 ...
 $ short_address       : chr  "925/627 Victoria Street" "38 Federation Lane" "703/251 Johnston Street" "11A Clarke
Street" ...
 $ full_address        : chr  "925/627 Victoria Street, Abbotsford, Vic 3067" "38 Federation Lane, Abbotsford, Vic
3067" "703/251 Johnston Street, Abbotsford, Vic 3067" "11A Clarke Street, Abbotsford, Vic 3067" ...
 $ property_type       : chr  "apartment" "townhouse" "apartment" "house" ...
 $ price               : chr  "$590,000" "$1,170,000" "$700,000" "$1,750,000" ...
 $ bedrooms            : int  2 3 2 3 3 3 2 3 1 2 ...
 $ bathrooms           : int  1 2 2 1 2 2 2 1 1 2 ...
 $ parking_spaces      : int  1 1 1 2 2 2 1 0 1 1 ...
 $ building_size       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ building_size_unit  : chr  NA NA NA NA ...
 $ land_size           : num  -1 -1 -1 382 -1 -1 -1 -1 -1 -1 ...
 $ land_size_unit      : chr  NA NA NA "m²" ...
 $ listing_company_id  : chr  "DVAURA" "XPPEDN" "IPSSCG" "XBSRIC" ...
 $ listing_company_name : chr  "Ray White Southbank & Port Phillip -" "Marshall White Manningham - DONCASTER EAST"
"JMRE - NORTH MELBOURNE" "BigginScott - Richmond" ...
```

```
 $ listing_company_phone: chr  "0381020200" "0398229999" "0393282002" "0394264000" ...
 $ auction_date         : logi  NA NA NA NA NA NA ...
 $ available_date       : logi  NA NA NA NA NA NA ...
 $ sold_date            : chr  "2023-04-11" "2023-04-06" "2023-04-04" "2023-03-25" ...
 $ description          : chr  "-This 2 BR apartment in The Parkhouse is a genuine standout.<br/>-From award-winning
SJB Architect with interio"| __truncated__ "Entry to property at 1 Abbott Street, Abbottsford. <br/><br/>Peacefully
and privately tucked away between the v"| __truncated__ "This Spacious contemporary apartment is located in the Heart
of Abbotsford, with easy access to the Eastern Fre"| __truncated__ "Freestanding blonde brick veneer home in the
prized River precinct. Moments from the Yarra River and amazing pa"| __truncated__ ...
 $ images               : chr  "[{'link': 'https://i2.au.reastatic.net/1144x888-
format=webp/66b49391dab7dba51d5f30c8e7d0e4a4ff892729ab0d8294af6"| __truncated__ "[{'link':
'https://i2.au.reastatic.net/1144x888-format=webp/4c66be2f50b8ffea587a432ea869335fb2fb5da7709cb88307e"| __truncated__
"[{'link': 'https://i2.au.reastatic.net/1144x888-format=webp/6f8ba89511a93fd410c8d3a06861e70652a19eca4007ddf20bb"|
__truncated__ "[{'link': 'https://i2.au.reastatic.net/1144x888-
format=webp/b921e54585ca1695400dbf5d85f3eda16fcea5d687a84c7e0b7"| __truncated__ ...
 $ images_floorplans    : chr  "[{'link': 'https://i2.au.reastatic.net/1144x888-
format=webp/19cd773a4a32b9c282eca06d435052771d72e0e59af7af92232"| __truncated__ "[{'link':
'https://i2.au.reastatic.net/1144x888-format=webp/174ab973b51a98525c2d08ec74e85eb0791a3faadee7f48d563"| __truncated__
"[]" "[{'link': 'https://i2.au.reastatic.net/1144x888-
format=webp/0a31b7b3d73baa0e2d88ff723806b5d49c29ca5c79c45371069"| __truncated__ ...
 $ listers              : chr  "[{'id': '1881998', 'name': 'Max Hui', 'agent_id': None, 'job_title': 'Sales
Consultant', 'url': 'https://www.re"| __truncated__ "[{'id': '2984135', 'name': 'Fabio Forlano', 'agent_id': None,
'job_title': 'Sales Executive', 'url': 'https://w"| __truncated__ "[{'id': '858455', 'name': 'John Sdregas',
'agent_id': 'b33aa855-9c6a-40c4-913b-232e2fb651b3', 'job_title': 'Man"| __truncated__ "[{'id': '167626', 'name':
'Andrew Crotty', 'agent_id': 'a8109f08-a34e-471b-8624-0d4c329172f7', 'job_title': 'Di"| __truncated__ ...
 $ inspections          : chr  "[]" "[]" "[]" "[]" ...
[1] "Vic" "Qld" "NT"
[1] "Vic"
```

Explanation:

The code reads a CSV file named "property_transaction_victoria.csv" into a data box named "ptv_data".

It checks the dimensions (number of rows and columns) of the data frame.

It shows the structure of the data frame, displaying column names and their data types.

It checks for unique values in the "state" column.

It filters data frames to keep only records with a status of "Vic".

It checks for unique values in the Status column after filtering.

Task 2

Code:

```
# Task 2:
# Delete unnecessary columns
ptv_data_vic <- ptv_data_vic %>%
  select(-badge, -url, -building_size_unit, -land_size_unit, -listing_company_id, -listing_company_phone, -auction_date, -available_date, -images, -images_floorplans, -listers, -inspections)

# Check the dimensions of the updated dataframe
dim(ptv_data_vic)

# Display the first 5 lines
head(ptv_data_vic, 5)
```

Answer:

Description: df [5 × 16]

	id <int>	suburb <chr>	state <chr>	postcode <int>	short_address <chr>
1	141077012	Abbotsford	Vic	3067	925/627 Victoria Street
2	141774568	Abbotsford	Vic	3067	38 Federation Lane
3	141373204	Abbotsford	Vic	3067	703/251 Johnston Street
4	141606240	Abbotsford	Vic	3067	11A Clarke Street
5	141603340	Abbotsford	Vic	3067	158 Gipps Street

5 rows | 1-6 of 16 columns

full_address <chr>	property_type <chr>	price <chr>	bedrooms <int>	bathrooms <int>
925/627 Victoria Street, Abbotsford, Vic 3067	apartment	$590,000	2	1
38 Federation Lane, Abbotsford, Vic 3067	townhouse	$1,170,000	3	2
703/251 Johnston Street, Abbotsford, Vic 3067	apartment	$700,000	2	2
11A Clarke Street, Abbotsford, Vic 3067	house	$1,750,000	3	1
158 Gipps Street, Abbotsford, Vic 3067	townhouse	$1,412,000	3	2

5 rows | 7-11 of 16 columns

parking_spaces <int>	building_size <dbl>	land_size <dbl>	listing_company_name <chr>	sold_date <chr>
1	NA	-1	Ray White Southbank & Port Phillip -	2023-04-11
1	NA	-1	Marshall White Manningham - DONCASTER EAST	2023-04-06
1	NA	-1	JMRE - NORTH MELBOURNE	2023-04-04
2	NA	382	BigginScott - Richmond	2023-03-25
2	NA	-1	BigginScott - Richmond	2023-03-25

5 rows | 12-16 of 16 columns

description <chr>
-This 2 BR apartment in The Parkhouse is a genuine standout. -From award-winning SJB Architect with interior design by Sue Carr.<b...
Entry to property at 1 Abbott Street, Abbottsford.   Peacefully and privately tucked away between the vibrancy of inner-Melbour...
This Spacious contemporary apartment is located in the Heart of Abbotsford, with easy access to the Eastern Freeway, Melbourne CBD, w...
Freestanding blonde brick veneer home in the prized River precinct. Moments from the Yarra River and amazing parkland, this large hom...
Behind the timeless period façade of this classic Abbotsford terrace, discover light-filled interiors that have been updated to meet the de...

5 rows | 17-17 of 16 columns

Explanation:
The code uses the dplyr package to remove unnecessary columns from the "ptv_data_vic" data box using the select() function.
It checks the dimensions of the updated data framework.
It displays the first 5 rows of the updated data box.
Task 3
Code:

```
# Filter data
filtered_data_suburb <- ptv_data_vic %>%
  filter(suburb %in% c('Clayton', 'Mount Waverley', 'Glen Waverley', 'Abbotsford')) %>%
  filter(property_type %in% c('apartment', 'house', 'townhouse', 'unit'))

# Remove rows with missing values
filtered_data_suburb_naomit <- na.omit(filtered_data_suburb)

# Select the required columns
filtered_data_named <- filtered_data_suburb_naomit %>%
  select(suburb, property_type, price)

# Convert dollar signs and commas to numeric types
filtered_data_named$price <- gsub('\\$', "", filtered_data_named$price)
filtered_data_named$price <- gsub(",", "", filtered_data_named$price)
filtered_data_named$price <- as.numeric(filtered_data_named$price)

# Remove rows with missing values
filtered_data_named <- na.omit(filtered_data_named)

# Output the filtered and cleaned data
str(filtered_data_named)

# Summarize the data by suburb and property type
summary_filtered_data <- filtered_data_named %>%
  group_by(suburb, property_type) %>%
  summarise(
    Max_Price = max(price),
    Min_Price = min(price),
    Mean_Price = mean(price),
    Median_Price = median(price)
  )

# Output summary_filtered_data
summary_filtered_data
```

Answer:

suburb <chr>	property_type <chr>	Max_Price <dbl>	Min_Price <dbl>	Mean_Price <dbl>	Median_Price <dbl>
Abbotsford	apartment	1075000	301000	596210.4	569000.0
Abbotsford	house	1425000	830000	1115375.0	1112500.0
Abbotsford	townhouse	907000	725000	798250.0	780500.0
Abbotsford	unit	727500	545000	636250.0	636250.0
Clayton	apartment	685000	295000	447876.9	425000.0
Clayton	house	1635000	525000	845611.7	733099.5
Clayton	townhouse	1550000	380000	703026.7	671500.0
Clayton	unit	600000	318700	412563.6	390000.0
Glen Waverley	apartment	850000	360000	545139.2	540000.0
Glen Waverley	house	4975000	580000	1670182.1	1398000.0

1-10 of 16 rows Previous 1 2 Next

suburb <chr>	property_type <chr>	Max_Price <dbl>	Min_Price <dbl>	Mean_Price <dbl>	Median_Price <dbl>
Glen Waverley	townhouse	2220000	696000	1364133.3	1295000.0
Glen Waverley	unit	1420000	660000	1043200.0	1156000.0
Mount Waverley	apartment	465000	465000	465000.0	465000.0
Mount Waverley	house	3300000	735000	1583125.9	1390000.0
Mount Waverley	townhouse	1856500	465000	1147750.0	1048500.0
Mount Waverley	unit	1230000	600000	829500.0	775000.0

11-16 of 16 rows Previous 1 2 Next

Explanation:
Use the filter() function in dplyr to filter the data by specific suburb and attribute type.
Use the na.omit() function to remove missing rows.
Use the select() function to select the required columns.
Clean up the "price" column by removing special characters and converting them to numbers.
Rows with missing values are deleted again.
Print a structure for filtering and cleaning data.
The data was summarized by suburb and property type, and the highest, lowest, average and median prices were calculated.
Task 4
Code:

```
# See the number of missing values in each column
missing_value_counts <- colSums(is.na(ptv_data_vic))

# Calculate the percentage of missing values
missing_value_percent <- missing_value_counts / nrow(ptv_data_vic)

# Conversion percentage value
missing_value_percent <- percent(missing_value_percent, accuracy = 0.001)

# output missing_value_percent value
missing_value_percent
```

Answer:

```
                 id              suburb               state            postcode       short_address 
           "0.000%"            "0.000%"            "0.000%"            "0.000%"            "1.473%" 
       full_address       property_type               price            bedrooms           bathrooms 
           "1.473%"            "0.000%"            "0.000%"            "4.763%"            "4.763%" 
     parking_spaces       building_size           land_size listing_company_name           sold_date 
           "4.763%"           "94.567%"            "0.001%"            "4.846%"            "1.240%" 
        description 
           "0.002%" 
```

Explanation:
Use the colsum() and is.na() functions to view and count the number of missing values in each

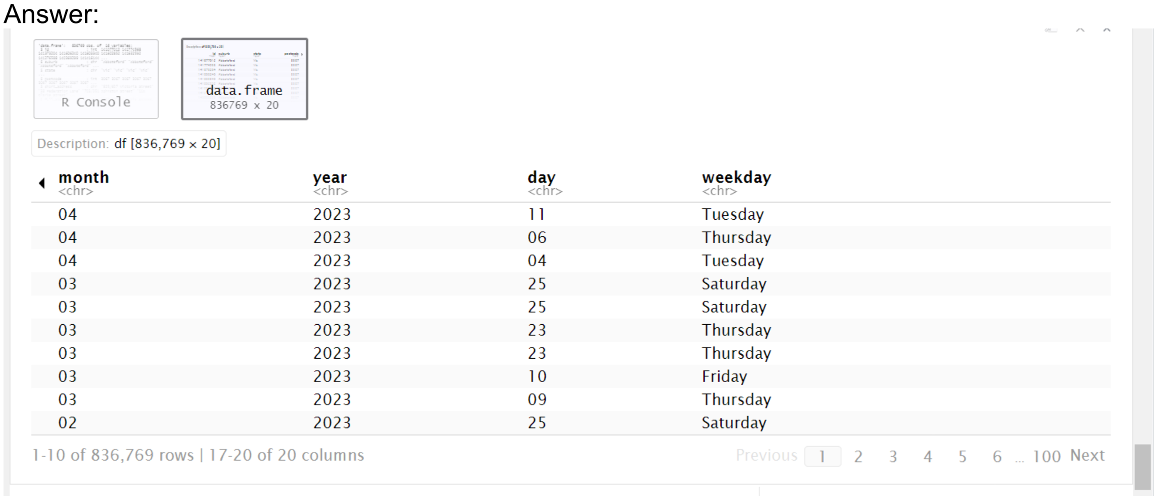
column. The percentage of missing values in each column is then calculated. The percent() function in the scales package is also used to format the percentage value. The percentage of the last output missing value.

Task 5

Code:

```
# View the structure of the data
str(ptv_data_vic)

# Change the "sold_date" column to Date style
ptv_data_vic$sold_date <- as.Date(ptv_data_vic$sold_date)

# Check the structure of the updated data
str(ptv_data_vic)

# Processing date formatting
Sys.setlocale("LC_TIME", "English")

# Processing year, month, week, day
ptv_data_vic$month <- format(ptv_data_vic$sold_date, "%m")
ptv_data_vic$day <- format(ptv_data_vic$sold_date, "%d")
ptv_data_vic$weekday <- format(ptv_data_vic$sold_date, "%A")
ptv_data_vic$year <- format(ptv_data_vic$sold_date, "%Y")

ptv_data_vic
```

Answer:



month <chr>	year <chr>	day <chr>	weekday <chr>
04	2023	11	Tuesday
04	2023	06	Thursday
04	2023	04	Tuesday
03	2023	25	Saturday
03	2023	25	Saturday
03	2023	23	Thursday
03	2023	23	Thursday
03	2023	10	Friday
03	2023	09	Thursday
02	2023	25	Saturday

Explanation:

Examine the structure of the raw data.

Convert the "sold_date" column to Date format using the as.Date() function.

Check the structure of the updated data.

Use Sys.setlocale() to set the locale for the date format to English.

Use the format() function to extract the year, month, day, and day from the "sold_date" column.

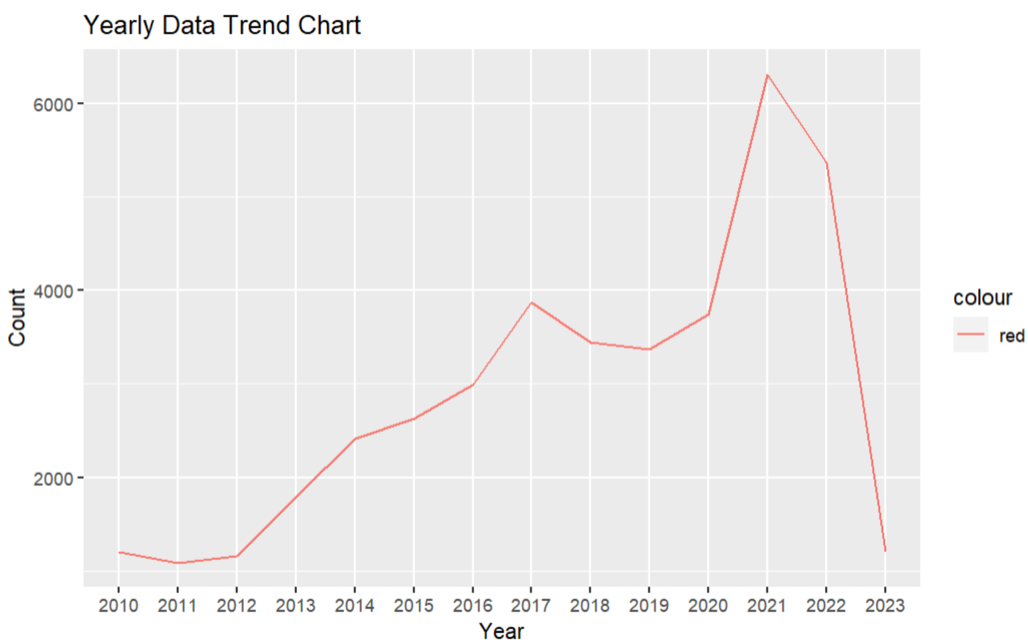
Task 6

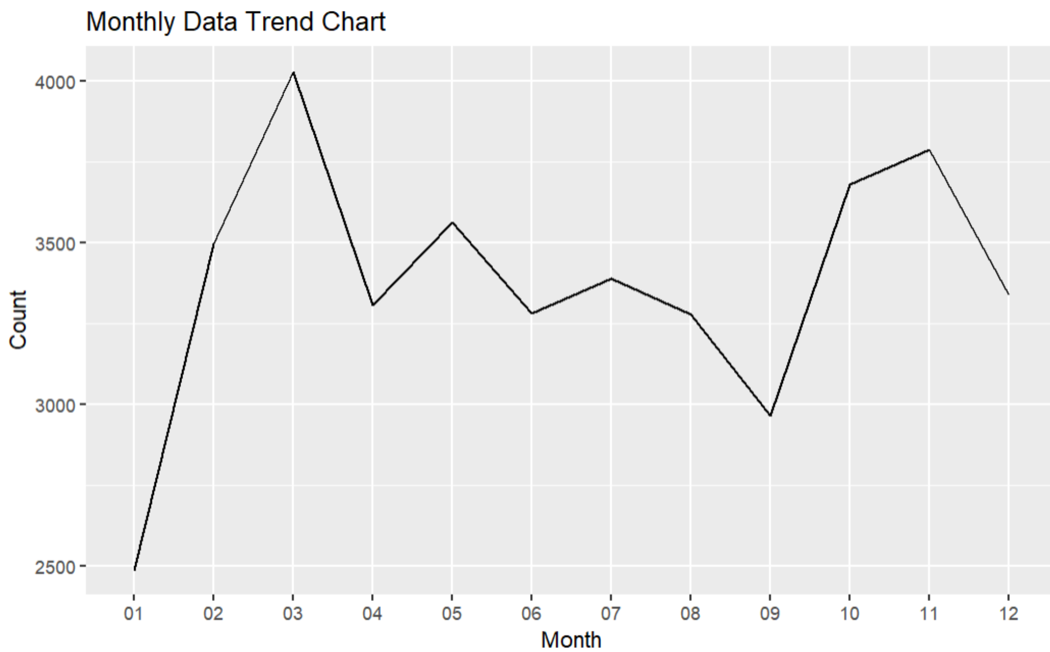
Code:

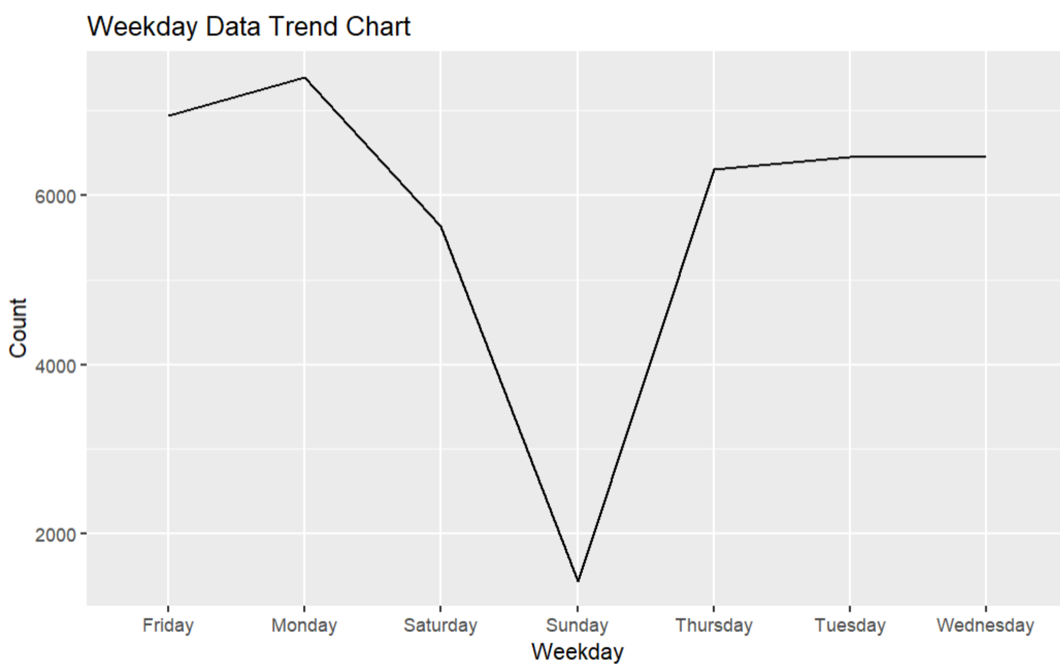
```
# Sort the data by sold date
ptv_sorted <- ptv_data_vic %>% arrange(sold_date)

# Delete the null sold_date value
unique_dates <- ptv_sorted$sold_date %>% na.omit()

# Print the unique dates
```

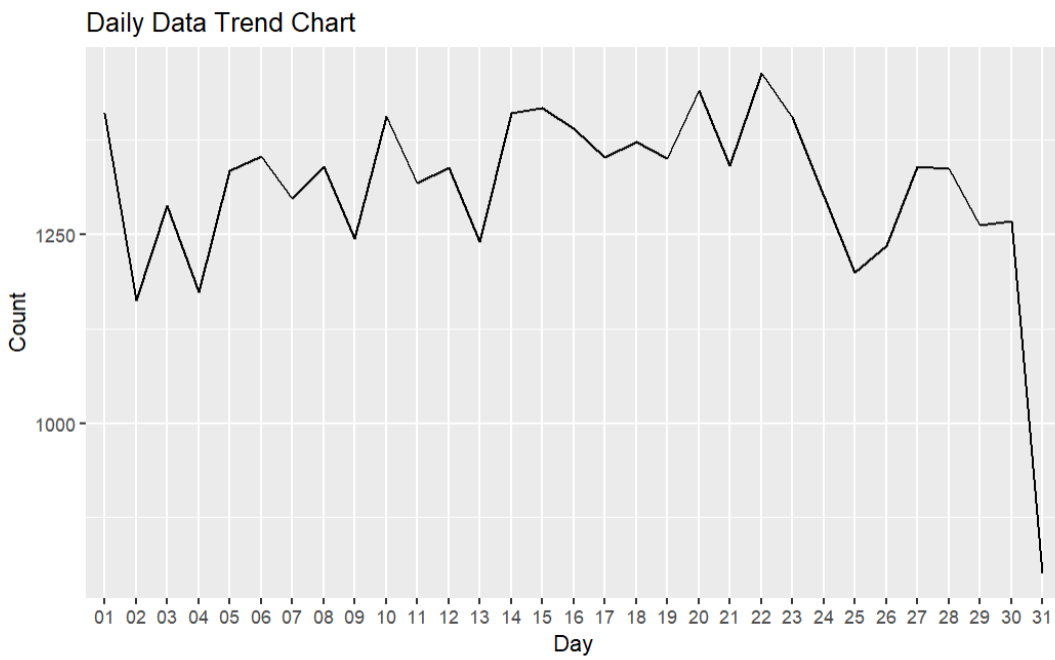
```
unique(unique_dates)

# Print the earliest and latest dates
earliest_date <- head(unique_dates, 1)
latest_date <- tail(unique_dates, 1)
earliest_date
latest_date

# Calculate the yearly trend
yearly_trend_data <- ptv_data_vic %>%
  na.omit() %>%
  group_by(year) %>%
  summarise(count = n())

# Chart yearly trend
ggplot(yearly_trend_data, aes(x = year, y = count, group = 1, color = 'red')) +
  geom_line() +
  labs(title = 'Yearly Data Trend Chart', x = 'Year', y = 'Count')

# Calculate the monthly trend
monthly_trend_data <- ptv_data_vic %>%
  na.omit() %>%
  group_by(month) %>%
  summarise(count = n())

# Chart monthly trend
ggplot(monthly_trend_data, aes(x = month, y = count, group = 1)) +
  geom_line() +
  labs(title = 'Monthly Data Trend Chart', x = 'Month', y = 'Count')

# Calculate the weekday trend
weekday_trend_data <- ptv_data_vic %>%
  na.omit() %>%
  group_by(weekday) %>%
  summarise(count = n())

# Chart weekday trend
ggplot(weekday_trend_data, aes(x = weekday, y = count, group = 1)) +
  geom_line() +
  labs(title = 'Weekday Data Trend Chart', x = 'Weekday', y = 'Count')

# Calculate the daily trend
daily_trend_data <- ptv_data_vic %>%
  na.omit() %>%
  group_by(day) %>%
  summarise(count = n())

# Chart daily trends
ggplot(daily_trend_data, aes(x = day, y = count, group = 1)) +
  geom_line() +
  labs(title = 'Daily Data Trend Chart', x = 'Day', y = 'Count')
```
Answer:

```
 [905] "2012-06-23" "2012-06-24" "2012-06-25" "2012-06-26" "2012-06-27" "2012-06-28"
"2012-06-29" "2012-06-30"
 [913] "2012-07-01" "2012-07-02" "2012-07-03" "2012-07-04" "2012-07-05" "2012-07-06"
"2012-07-07" "2012-07-08"
 [921] "2012-07-09" "2012-07-10" "2012-07-11" "2012-07-12" "2012-07-13" "2012-07-14"
"2012-07-15" "2012-07-16"
 [929] "2012-07-17" "2012-07-18" "2012-07-19" "2012-07-20" "2012-07-21" "2012-07-22"
"2012-07-23" "2012-07-24"
 [937] "2012-07-25" "2012-07-26" "2012-07-27" "2012-07-28" "2012-07-29" "2012-07-30"
"2012-07-31" "2012-08-01"
 [945] "2012-08-02" "2012-08-03" "2012-08-04" "2012-08-05" "2012-08-06" "2012-08-07"
"2012-08-08" "2012-08-09"
 [953] "2012-08-10" "2012-08-11" "2012-08-12" "2012-08-13" "2012-08-14" "2012-08-15"
"2012-08-16" "2012-08-17"
 [961] "2012-08-18" "2012-08-19" "2012-08-20" "2012-08-21" "2012-08-22" "2012-08-23"
"2012-08-24" "2012-08-25"
 [969] "2012-08-26" "2012-08-27" "2012-08-28" "2012-08-29" "2012-08-30" "2012-08-31"
"2012-09-01" "2012-09-02"
 [977] "2012-09-03" "2012-09-04" "2012-09-05" "2012-09-06" "2012-09-07" "2012-09-08"
"2012-09-09" "2012-09-10"
 [985] "2012-09-11" "2012-09-12" "2012-09-13" "2012-09-14" "2012-09-15" "2012-09-16"
"2012-09-17" "2012-09-18"
 [993] "2012-09-19" "2012-09-20" "2012-09-21" "2012-09-22" "2012-09-23" "2012-09-24"
"2012-09-25" "2012-09-26"
 [ reached 'max' / getOption("max.print") -- omitted 3852 entries ]
[1] "2010-01-01"
[1] "2023-04-14"
```




Monthly Data Trend Chart
4000
3500
3000
2500
Count
01
02
03
04
05
06
07
08
09
10
11
12
Month

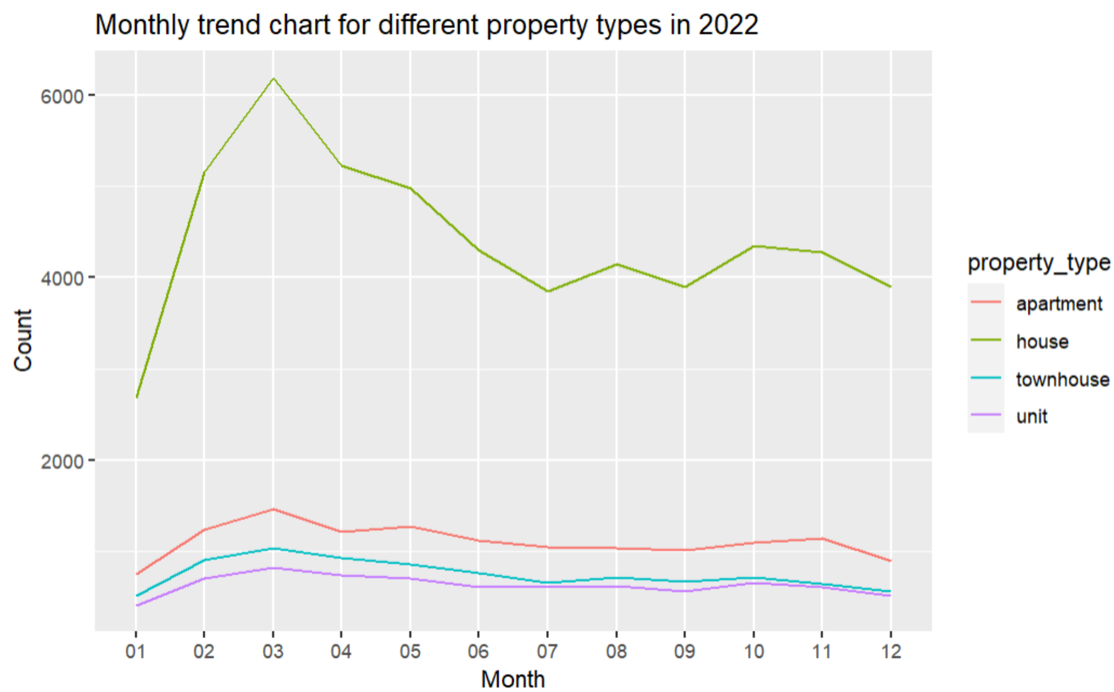

Weekday Data Trend Chart
6000
4000
2000
Count
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Weekday



Explanation:
Sort the data by date of sale.
Extract unique dates without losing values.
Print unique dates, as well as the earliest and latest dates.
Annual trends are calculated by grouping data by year and counting the number of transactions.
Plot the annual trend using ggplot and geom_line().
Calculate monthly, weekday, and daily trends in a similar way and plot them using ggplot
Task 7
Code:

```
# Filtered data
ptv_2022_vic_data <- ptv_data_vic %>%
  filter(year == "2022" & property_type %in% c('apartment', 'house', 'townhouse', 'unit'))

# Group data by month and property and calculate the number of transactions
ptv_2022_vic_data_count <- ptv_2022_vic_data %>%
  group_by(month, property_type) %>%
  summarise(count = n(), .groups = 'drop')

# Trend mapping
ggplot(ptv_2022_vic_data_count, aes(x = month, y = count, group = property_type, color = property_type)) +
  geom_line() +
  labs(title = 'Monthly trend chart for different property types in 2022', x = 'Month', y = 'Count')
```

Answer:



Explanation:
Filter data for 2022 and specific attribute types.
Group the data by month and attribute type, and calculate the transaction count.
Using ggplot and geom_line(), monthly trends for different property types in 2022 are plotted.
Task 8
Code:

```
# Remove the dollar sign and comma and convert them to a number type
ptv_2022_data_vic_price <- ptv_data_vic
ptv_2022_data_vic_price$price <- gsub('\\$', "", ptv_2022_data_vic_price$price)
ptv_2022_data_vic_price$price <- gsub(",", "", ptv_2022_data_vic_price$price)
ptv_2022_data_vic_price$price <- as.numeric(ptv_2022_data_vic_price$price)
ptv_2022_data_vic_price <- na.omit(ptv_2022_data_vic_price, cols = "price")

# Filter data
ptv_2022_data_vic_price <- ptv_2022_data_vic_price %>%
  filter(year == "2022" & property_type %in% c('apartment', 'house', 'townhouse', 'unit'))
# Group by property type and month and calculate total and average prices
ptv_2022_data_vic_price <- ptv_2022_data_vic_price %>%
  group_by(property_type, month) %>%
  summarise(
    total_price = sum(price),
    average_price = mean(price)
  )

ptv_2022_data_vic_priceAnswer:
```

A tibble: 48 x 4 Groups: property_type [4]

property_type <chr>	month <chr>	total_price <dbl>	average_price <dbl>
apartment	01	69591312	650386.1
apartment	02	108642339	696425.2
apartment	03	113270530	792101.6
apartment	04	100172399	720664.7
apartment	05	96853460	768678.3
apartment	06	89812587	718500.7
apartment	07	76589888	750881.3
apartment	08	80695287	720493.6
apartment	09	71166276	635413.2
apartment	10	85886119	645760.3

1-10 of 48 rows Previous 1 2 3 4 5 Next

Explanation:
Removes special characters from the price column and converts them to numbers.
Filter 2022 and specific property type data and remove NA values.
The data is grouped by property type and month, and total and average prices are calculated.
Task 9.1
Code:

```
# Filter data for the year 2022
ptv_2022_data_vic_suburb <- ptv_data_vic %>%
  filter(year == '2022') %>%
  group_by(suburb) %>%
  summarise(count = n(), .groups = 'drop') %>%
  arrange(desc(count))

# Top 10 suburbs with the most output
head(ptv_2022_data_vic_suburb, 10)
```

Answer:

suburb <chr>	count <int>
Pakenham	1012
Tarneit	984
Melbourne	953
Clyde North	940
Craigieburn	904
South Yarra	761
Werribee	752
Point Cook	736
Richmond	717
Berwick	703

1-10 of 10 rows

Explanation:
I first sifted through the 2022 data and grouped it by suburb, calculating the number of transactions in each suburb. The top 10 suburbs with the highest number of transactions are then displayed in descending order by number of transactions.
Task 9.2
Code:

```
# Get the top 10 suburbs
vic_data_top_10 <- head(ptv_2022_data_vic_suburb, 10)
vic_data_top_10_suburb <- vic_data_top_10$suburb

# Filter data for the top 10 suburbs
ptv_2022_vic_data_top_10 <- ptv_2022_vic_data %>%
  filter(suburb %in% vic_data_top_10_suburb) %>%
  group_by(property_type) %>%
  summarise(count = n(), .groups = 'drop') %>%
  arrange(desc(count))

# Output the most data
```

head(ptv_2022_vic_data_top_10, 1)Answer:

property_type <chr>	count <int>
house	5411

1 row

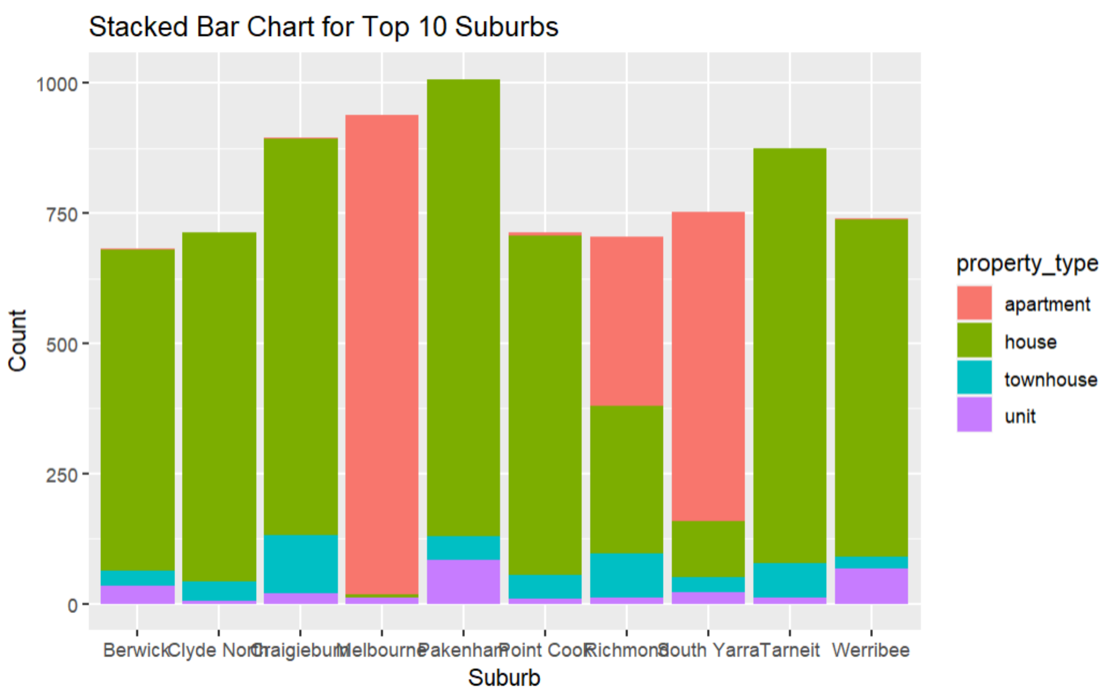
Explanation:
I extracted data from the top 10 suburbs with the highest number of transactions and filtered it based on those suburbs. It was then grouped by property type to calculate the number of transactions for each property type in these suburbs. Finally, the number of transactions is ranked in descending order, and the most traded property types in these suburbs are output

Task 9.3

Code:

```
# Group and calculate transaction counts by suburb and property type
ptv_2022_top_10_data <- ptv_2022_vic_data %>%
  filter(suburb %in% vic_data_top_10_suburb) %>%
  group_by(suburb, property_type) %>%
  summarise(count = n(), .groups = 'drop')

# Draw a stacked bar chart
ggplot(ptv_2022_top_10_data, aes(x = suburb, y = count, fill = property_type)) +
  geom_bar(stat = "identity") +
  labs(title = "Stacked Bar Chart for Top 10 Suburbs", x = "Suburb", y = "Count")
```

Answer:



Explanation:
Again, I screened the data based on the top 10 suburbs with the highest number of transactions, grouped them by suburb and property type, and calculated the number of transactions for different property types in each suburb. A stacked bar chart was then used to visualize the number of transactions across property types in these suburbs

Task 10.1

Code:

```
# Select the suburbs of interest
```

```
suburbs <- c('Kew', 'South Yarra', 'Caulfield', 'Clayton', 'Glen Waverley', 'Burwood', 'Abbotsford')

# Filter data for property type 'house' and the selected suburbs
ptv_house <- ptv_data %>%
  filter(property_type == 'house' & suburb %in% suburbs)

# Remove the dollar and comma characters and convert them to numbers
ptv_house$price <- gsub('\\$', "", ptv_house$price)
ptv_house$price <- gsub(",", "", ptv_house$price)
ptv_house$price <- as.numeric(ptv_house$price)

# Delete rows that are missing values
ptv_house_new <- ptv_house[, c("suburb", "price", "parking_spaces", "bathrooms", "land_size", "bedrooms")]
ptv_house_new <- na.omit(ptv_house_new)

# Calculate the mean and median of different variables
summary_stats <- ptv_house_new %>%
  group_by(suburb) %>%
  summarise(
    mean_bedrooms = mean(bedrooms),
    mean_bathrooms = mean(bathrooms),
    mean_parking_spaces = mean(parking_spaces),
    mean_land_size = mean(land_size),
    mean_price = mean(price),
    median_bedrooms = median(bedrooms),
    median_bathrooms = median(bathrooms),
    median_parking_spaces = median(parking_spaces),
    median_land_size = median(land_size),
    median_price = median(price)
  )

summary_stats
```

Answer:

R Console

tbl_df
7 x 11

A tibble: 7 × 11

suburb <chr>	mean_bedrooms <dbl>	mean_bathrooms <dbl>	mean_parking_spaces <dbl>	mean_land_size <dbl>	mean_price <dbl>
Abbotsford	2.621283	1.352113	0.7308294	36.71283	1129526
Burwood	3.620112	1.749845	1.8566108	458.29804	1188183
Caulfield	3.570957	1.980198	2.1023102	137.30363	1701995
Clayton	3.688431	1.577320	1.9072165	375.87663	1059806
Glen Waverley	3.904606	2.120783	2.0116068	554.79629	1447572
Kew	3.766350	2.260021	2.1323840	390.37877	2542391
South Yarra	2.909820	1.765531	1.2064128	55.81170	2167750

7 rows | 1-6 of 11 columns

R Console

tbl_df
7 x 11

A tibble: 7 x 11

mean_price <dbl>	median_bedrooms <dbl>	median_bathrooms <dbl>	median_parking_spaces <dbl>	median_land_size <dbl>	median_price <dbl>
1129526	3	1	0	-1	1050000
1188183	3	2	2	586	1150000
1701995	4	2	2	-1	1625000
1059806	3	1	2	322	1004000
1447572	4	2	2	664	1320000
2542391	4	2	2	352	2194000
2167750	3	2	1	-1	1755000

7 rows | 6-11 of 11 columns

Explanation:
I selected some suburbs and filtered out the data with the attribute type "house". Then, we remove the dollar and comma characters from the price column and convert them to numeric types. Next, select the relevant columns and delete the rows with missing values. Finally, they were grouped by suburb, and the mean and median of the different variables were calculated.
Task 10.2
Code:

```
# Calculate correlations between different variables
ptv_house_cor <- ptv_house %>%
  group_by(suburb) %>%
  summarise(
    cor_bedrooms_bathrooms = cor(bedrooms, bathrooms),
    cor_bathrooms_price = cor(bathrooms, price),
    cor_bedrooms_parking_spaces = cor(bedrooms, parking_spaces),
    cor_parking_spaces_land_size = cor(parking_spaces, land_size),
    cor_bedrooms_land_size = cor(bedrooms, land_size),
    cor_bedrooms_price = cor(bedrooms, price),
    cor_bathrooms_parking_spaces = cor(bathrooms, parking_spaces),
    cor_bathrooms_land_size = cor(bathrooms, land_size),
    cor_parking_spaces_price = cor(parking_spaces, price),
    cor_land_size_price = cor(land_size, price)
  )

ptv_house_cor
```

suburb <chr>	cor_bedrooms_bathrooms <dbl>	cor_bedrooms_parking_spaces <dbl>	cor_bedrooms_land_size <dbl>
Abbotsford	0.4766582	0.3014821	0.20123077
Burwood	0.7075537	0.1732773	0.16268302
Caulfield	0.6017068	0.3097690	0.13496621
Clayton	0.8250284	0.5578885	0.18171487
Glen Waverley	0.6495664	0.2456680	0.06374553
Kew	NA	NA	NA
South Yarra	0.5951536	0.5420237	0.45268586

7 rows | 1-4 of 11 columns

cor_bedrooms_price <dbl>	cor_bathrooms_parking_spaces <dbl>	cor_bathrooms_land_size <dbl>	cor_bathrooms_price <dbl>
NA	0.2976634	0.01083550	NA
NA	0.1816507	0.05919068	NA
NA	0.3096220	0.09516561	NA
NA	0.5475348	0.13839578	NA
NA	0.2166541	0.01544867	NA
NA	NA	NA	NA
NA	0.5269101	0.22330575	NA

7 rows | 5-8 of 11 columns

cor_bathrooms_price <dbl>	cor_parking_spaces_land_size <dbl>	cor_parking_spaces_price <dbl>	cor_land_size_price <dbl>
NA	0.13218909	NA	NA
NA	0.06960827	NA	NA
NA	0.17895408	NA	NA
NA	0.24853842	NA	NA
NA	0.09699488	NA	NA
NA	NA	NA	NA
NA	0.33745236	NA	NA

7 rows | 8-11 of 11 columns

Explanation:
Grouped by suburb, I calculated the correlation between the different variables
Task 11.1
Code:

```
# Removes missing values and calculates the length
ptv_description <- ptv_data_vic %>%
  na.omit() %>%
  mutate(description_length = nchar(gsub('<br/>', '', description)),
         description_length_group = cut(description_length, breaks = c(0, 500, 1000, 1500, 2000,
2500, Inf),
                                        labels = c('[1,500]', '[501,1000]', '[1001,1500]', '[1501,2000]', '[2001,2500]',
'>=2500')))
ptv_description
```

Answer:

Description: df [40,595 × 22]

month <chr>	year <chr>	day <chr>	weekday <chr>	description_length <int>	description_length_group <fctr>
02	2023	24	Friday	1433	[1001,1500]
02	2023	10	Friday	1046	[1001,1500]
12	2022	09	Friday	1176	[1001,1500]
09	2022	14	Wednesday	2141	[2001,2500]
03	2022	09	Wednesday	1350	[1001,1500]
01	2022	17	Monday	1865	[1501,2000]
01	2022	12	Wednesday	1283	[1001,1500]
12	2021	03	Friday	1415	[1001,1500]
12	2021	02	Thursday	1569	[1501,2000]
12	2021	02	Thursday	3206	>=2500

1-10 of 40,595 rows | 18-23 of 22 columns   Previous 1 2 3 4 5 6 ... 100 Next

Explanation:
I first removed the line containing the missing value and calculated the length of the description text. Use the na.omit() function to remove missing values, and use the nchar() function to calculate the character length of the data. Before calculating the length, I use the gsub() function to replace the "
" tag in the description text with an empty string. Next, I use the cut() function to divide the descriptive text length into different groups, using predefined split points (0, 500, 1000, 1500, 2000, 2500, and infinity). And set a label for each group (" [1500] ", "[501100]", "[1001150]", "[1501200]", "[2001250]" and "& gt; = 2500 "). Finally, create a new data ptv_description that contains information about the description text length and the description text length group.
Task 11.2
Code:

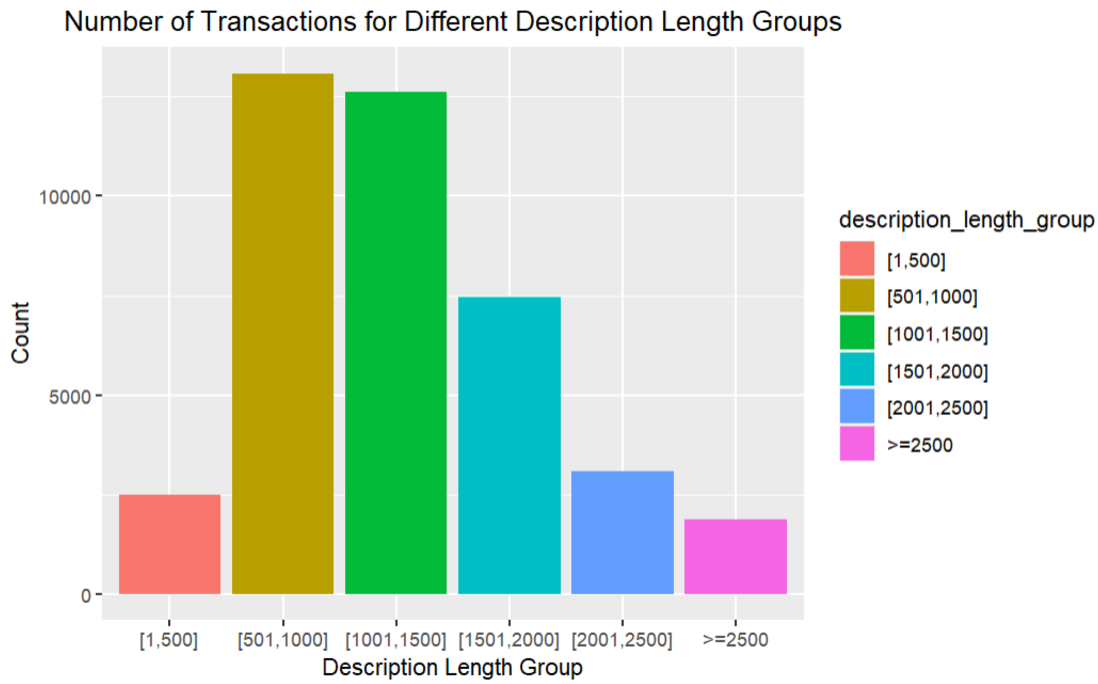
```
# Group the data and count the number of lists
ptv_description_length_group <- ptv_description %>%
  group_by(description_length_group) %>%
  summarise(count = n(), .groups = 'drop')

# Draw bar charts
ggplot(ptv_description_length_group, aes(x = description_length_group, y = count, fill =
description_length_group)) +
  geom_bar(stat = 'identity') +
  labs(x = 'Description Length Group', y = 'Count') +
```

```
  ggtitle('Number of Transactions for Different Description Length Groups') +
  theme(plot.title = element_text(hjust = 0.5))
ptv_description_length_group
```

Answer:

description_length_group <fctr>	count <int>
[1,500]	2483
[501,1000]	13080
[1001,1500]	12608
[1501,2000]	7464
[2001,2500]	3078
>=2500	1882

6 rows



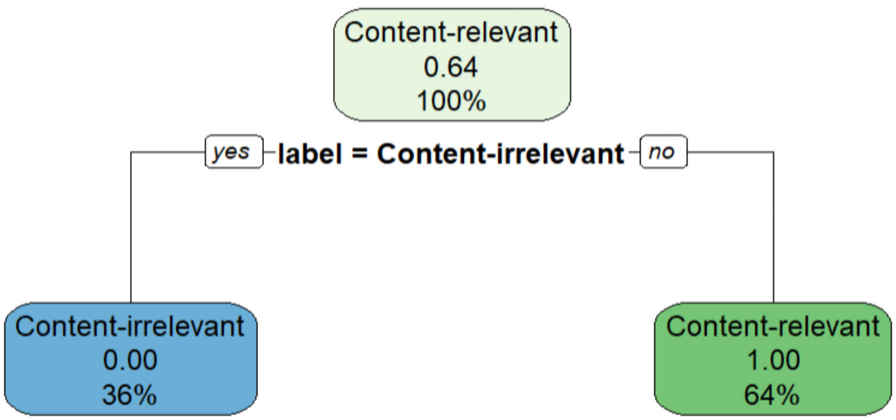
Explanation:
I group the data and calculate the quantity. I use the group_by() function to group the description text length groups, then the summarise() function to calculate the number of each group. Finally, I create a data named ptv_description_length_group.
Finally, I use stacked bar charts to visualize the number of property transactions in groups of different descriptive text lengths.
Task D:
Code:

```
library(rpart)
library(dplyr)
library(rpart.plot)
library(tidyverse)

# Read data
```

```
train_data <- read.csv('forum_liwc_train.csv')
test_data <- read.csv('forum_liwc_test.csv')
# Remove rows with missing values
train_data <- na.omit(train_data)
test_data <- na.omit(test_data)

# Select independent variables
independent_vars <- train_data[, 2:90]  # Select features from column 2 to 90 as independent variables

# Extract labels
labels <- train_data$label

# Build Model 1
model1 <- rpart(labels ~ ., data = train_data, method = 'class')

# The training data evaluates the performance of the model
train_predictions <- predict(model1, train_data, type = 'class')
train_accuracy <- sum(train_predictions == labels) / length(labels)
train_accuracy

# Turn unit_faculty and demographic_sex into digital encodings
train_data$unit_faculty <- as.factor(train_data$unit_faculty)
train_data$demographic_sex <- as.factor(train_data$demographic_sex)

# Computational correlation
cor_unit_faculty <- cor(as.numeric(train_data$unit_faculty), as.numeric(train_data$label))
cor_unit_faculty

# Computational correlation
cor_demographic_sex <- cor(as.numeric(train_data$demographic_sex), as.numeric(train_data$label))
cor_demographic_sex

# Gets test_independent_vars and test_labels
test_independent_vars <- test_data[, 2:90]
test_labels <- test_data$label

# Prediction of test data
test_predictions <- predict(model1, test_data, type = 'class')

# Calculate the accuracy of the model
test_accuracy <- sum(test_predictions == test_labels) / length(test_labels)
test_accuracy

# Evaluate model performance
confusion_matrix <- table(test_labels, test_predictions)

precision <- confusion_matrix[2, 2] / sum(confusion_matrix[, 2])

recall <- confusion_matrix[2, 2] / sum(confusion_matrix[2, ])

f1_score <- 2 * precision * recall / (precision + recall)
f1_score

# Plot the decision tree model
```

rpart.plot(model1)
Answer:

```
[1] NA
Warning: NAs introduced by coercion[1] NA
Warning: NAs introduced by coercion[1] NA
[1] NA
                      test_predictions
test_labels            Content-irrelevant Content-relevant
  Content-irrelevant                  424                0
  Content-relevant                      0              686
[1] 1
[1] 1
[1] 1
```



Explanation:
Import the required R packages (rpart, dplyr, rpart.plot, and tidyverse).
Read CSV files for training and test data.
Select the independent variables in the training data and store them in the independent_vars variable.
Extract the labels of the training data and store them in the labels variable.
Use the rpart function to build a decision tree model (Model 1) with labels and all independent variables (.) A link was established between the two.
The model was evaluated using training data. First, a prediction is made on the training data (the predict function), and then the train_accuracy is calculated.
The representation of unit_faculty and demographic_sex are described. unit_faculty may be a categorical variable, which can be represented by either unique heat coding or factorization;
Check how well the model fits the test data. First, the independent variables and labels of the test data are extracted, and then a prediction is made on the test data (the predict function).
Calculate the model's test_accuracy on test data.
Evaluate model performance. First, create a confusion_matrix to compare the consistency between actual and predicted labels. precision, recall, and F1 score (f1_score) are then calculated to evaluate the performance of the model.
Finally, visualize the decision tree model using the rpart.plot function.
a. The code first reads the training and test data sets. It then selects all the independent variables as input features and uses these features to build a classification tree Model (Model 1). The performance of the model was evaluated on the training data set, and the accuracy rate was calculated as the evaluation index.
b. Analyze the correlation between features and target variables: By calculating the correlation

coefficient between features and labels, you can understand the linear correlation between them. In the code, I calculated the correlation coefficient between unit_faculty, demographic_sex and label, because the calculation result is 1, indicating that there is a strong linear relationship between the feature and the target variable.
c. To improve the performance of Model 1
Explanation:
I used the na.omit() function to remove lines in the data that contain missing values. This ensures that the model is not disturbed by missing values during training and testing

References:
[1]https://www.r-bloggers.com/2021/04/decision-trees-in-r/

[2]https://datatricks.co.uk/confusion-matrix-in-r-two-simple-methods
[3]https://www.digitalocean.com/community/tutorials/confusion-matrix-in-r
[4]https://rpubs.com/monuchacko/585317
[5]https://www.geeksforgeeks.org/confusion-matrix-machine-learning/