```
library(tidyverse)

library(tidyr)

library(dplyr)


# EJERCICIO N° 1


#PREGUNTA 1: Cuáles de las siguientes expresiones valen 99 para x = 10 en R? Analicen la sintaxis como si estuvieran programando

x=10


10x - 1


(x)(x) - 1


abs(x*x) - abs(9-x)


11 * x - x + 1


#el error en la expresion numero 1 es la falta del simbolo de multiplicación * entre 10 y x (10*x)

#el error de la expresion #2 es el mismo de la numero 1, falta el simbolo * enre (x) y (x)

# La expresión 3 no presenta errores y es la unica que funciona

#La solución de la expresión 4 es incorrecta porque no se obtiene el valor deseado



#PREGUNTA 2: Un vector contiene una serie de ganancias ordenadas de manera creciente.
Escriban el código que genera:



g <- c(10, 50, 180, 350, 490)
```

#La suma de todas las ganancias.

sum(g)

#La segunda ganancia más grande

x <- length(g)

g[x-1]

#La diferencia más grande entre las ganancias.

d <- diff(g)

max(d)

#Un booleano que responda a la pregunta: ¿La más grande diferencia ente dos ganancias es mayor a 10?

t <- max(d)

nchar(t) < 10

#la respuesta es verdadera porque en el booleano se cuentan los dígitos de la más grande diferencia que en mi caso es 170 y serían 3 dígitos así que 3 < 10 es verdadero.

#La menor diferencia positiva entre dos ganancias.

d <- diff(g)

min(d)

#El máximo número de ganancias que pueden sumar sin pasar de 10000.

cumsum(g)<10000

#---

```
# EJERCICIO 2: Dplyr en los Aeropuertos


library(nycflights13)

View(planes)

View(flights)

View(weather)


#Pregunta 3

#Instalen la librería nycflights13. Escriban el código para encontrar todos los vuelos que:


# el código para encontrar todos los vuelos que:


# Fueron de SFO(San Francisco) hasta OAK(Oakland).


vuelos <- flights %>%
  select(origin, dest)


vuelosV2 <- mutate(vuelos,
                   dest = factor(origin, labels = c('OAK')),
                   origin = factor(dest, labels = c('SFO')))
View(vuelosV2)


#viajes en enero


vuelos_enero <- flights %>%
  select(month)


vuelos_eneroV2 <- mutate(vuelos_enero,
                         month = factor(1))
```

```
View(vuelos_eneroV2)



#Tienen demoras de mas de una hora (las demoras están en minutos).



demora <- select(flights, flight, dep_delay, arr_delay) %>%
  mutate(demora = dep_delay + arr_delay) %>%
  filter(demora > 60)

View(demora)



#Salieron entre medianoche y las 5 a.m.

vuelos_madrugadaV2 <- filter(vuelos_madrugada, hour <= 5)
View(vuelos_madrugadaV2)
#en el rango de vuelos entre medianoche y las 5 a.m no existen vuelos, únicamente un vuelo a la 1 a.m y vuelos desde las 5 a.m en adelante, se debe ordenar la columna hour de la tabla para poder visualizar el viaje de la 1 a.m.



#Tuvieron una demora de llegada 2 veces mas grande que la de salida.



flights %>% select(retraso_salida = dep_delay, retraso_llegada = arr_delay) %>%
  mutate(diferencia_retrasos = (retraso_llegada - retraso_salida)) %>%
  filter(retraso_salida == diferencia_retrasos) %>% View()
```

#para observar en qué casos la demora de llegada es 2 veces mayor a la de salida realicé una diferencia entre los dos retrasos e igualé la diferencia de retrasos y el retraso de salida para obtener unicamente los valores de retraso de salida que restados sean la mitad de los retrasos de llegada.

#---

#Pregunta 4 Lean la ayuda de select().

#Escriban 2 formas de selecionar las dos variables de retraso.

```
help("select")
```

```
flights %>% select(dep_time, arr_time) %>% View()
```

```
select(flights, dep_time, arr_time) %>% View()
```

#Pregunta 5

#Ordenen la tabla por fecha de salida y tiempo. ¿Cuáles fueron los vuelos que sufrieron las mayores demoras? ¿Cuáles recuperaron la mayor cantidad de tiempo durante el vuelo?

```
fecha_vuelos <- flights %>% select(time_hour)
```

```
flights[order(time_hour)]
```

```
(flights = gsub(" .*$", "", time_hour))
```

Pregunta 6

#Calculen la velocidad en mph usando el tiempo (que está en minutos) y la distancia (que está en millas). ¿Cuál fue el avión que voló más rápido?

```
velocidad <- flights %>% select(air_time, distance) %>%
        mutate(tiempo.horas = (air_time/60),
            distancia.metros = distance*1609.34)
velocidad %>% filter(tiempo.horas, distancia.metros) %>%
    select(tiempo.horas, distancia.metros) %>%
    mutate(velocidad.mph = (distancia.metros / tiempo.horas)) %>% View()
```

#Pregunta 7 En dplyr el comando pipeline %>% se lee entonces. Significa:

x% > % f(y) −→ f(x,y).

#Es decir pasa x como primer argumento de f. Qué significan las siguientes líneas de código:

```
flights %>%
filter(! is.na(dep_delay)) %>%
group_by(date, hour) %>%
summarise(delay = mean(dep_delay), n = n()) %>%
filter(n > 10)
```

#en la primera linea lo que hace es seleccionar la tabla de flights que es la que se va a utiizar

#entonces es argumento de filter que selecciona filas a excepcion de dep_delay

#entonces es argumento de group_by que agrupa filas que contienen date y hour, pero hay una observacion y es que la columna date no existe en la tabla lo que genera un error.

#entonces es argumento de summarise que ordena los datos de la columna dep_delay

#entonces es argumento de filter que selecciona los valores que son > 10