STA240 Final Project

Anthony Zhao, Abby Li, William Yan

Scenario 1

Customer Arrival

Poisson process (rate = λ)

- T_k: Arrival time of the kth customer
- W_k: Time between the $k-1$th arrival and the kth arrival

$$W_k = T_k - T_{k-1}.$$

$W_k \sim Pois(\lambda)$

where $\lambda = 5$ customers per hour

Service Time

$S_k \sim Exp(\lambda)$

where $\lambda = 6$ customers per hour, so the average customer needs to wait 1/6 hours = 10 minutes.

Arrival Times

```r
library(tidyverse)
library(lubridate)
library(knitr)

set.seed(121)

# simulating the arrival times of customers throughout the day

# Poisson process (lambda = 5)
# Tk= arrival time of the kth customer
# Wk= time between the k-1th customer arrival and the kth customer arrival where Wk ~ Pois(la

# set parameter
lambdaA <- 5 # in units: customers per hour
opening_time <- hm("10:00")
closing_time <- hm("22:00")
hours <- hour(closing_time) - hour(opening_time) # operating hours: 10am to 10pm
total_time <- hours*60 # operating hours in minutes
lambdaA <- lambdaA/60 # customers per minute
# converting to minutes because our lambda is low, and we can can get greater precision in ar

n <- ceiling(lambdaA*total_time) # max number of customers the store can have throughout the

# generate W1,..,Wn (calculating the time between the arrival times of 2 customers)
W_sample <- rexp(n, rate= lambdaA)

# calculate T or the arrival times by summing together the Wi arrival times

T_sample <- numeric(n)

for(i in 1:n) {
  T_sample[i] <- sum(W_sample[1:i])
}

# all possible arrival times of customers throughout the day (X minutes after opening)

# however, the store is only open for 12 hours or 720 minutes so we must get rid of the valu

arrival_times_s1 <- T_sample[T_sample <= total_time]

arrival_times_s1
```

```
 [1]  15.48044  19.99824  21.21837  37.74527  48.04027  56.58220  61.71317
 [8]  87.79562  90.61075  98.27931 104.41170 104.58245 106.38726 119.22966
[15] 129.63617 139.66808 156.91180 161.87999 195.55182 199.32194 199.90017
[22] 203.53702 207.00779 207.92471 210.76699 246.48358 255.32026 261.08139
[29] 267.57539 292.35047 334.71035 351.57806 355.43151 378.50418 389.27844
[36] 394.13197 394.80570 423.04531 426.98859 443.30627 445.14078 462.77508
[43] 469.07521 470.41177 482.74142 505.02956 511.81598 548.50360 548.97865
[50] 549.67378 550.72973 565.97825 588.13573 589.15853 591.93323 599.78691
[57] 603.96967 631.88036 653.39341 653.56135
```

```
opening_time + minutes(floor(max(arrival_times_s1)))
```

```
[1] "10H 653M 0S"
```

Arrival Times Analysis

In this simulation, the number of customers that will be arriving within the operating hours is 60, with the first customer arriving 15 minutes after opening and the last customer arriving 67 minutes before closing

Serving Times

```
# given the output from above, simulate the serving times of customers before they leave

# notice that service time is modeled by exp(6)
lambdaS <- 6 # customers per hour
lambdaS <- lambdaS/60 # customers per minute

# simulate customer's service time
# n= only simulating the service time for those where T_sample <= total_time
service_times_s1 <- rexp(length(arrival_times_s1), rate= lambdaS)

#these are the serving times for each arriving customer before they leave
service_times_s1
```

```
 [1] 18.3226535  3.6118119 23.3500006  6.1564912 72.6485441 11.8925547
 [7]  2.4963011 25.2319737  4.5310184 15.6495337  3.7841440  1.1254289
[13] 18.2791955 14.6349111 37.8510680  5.4804337  4.8154802  3.7401264
[19]  4.4097259  5.6360781  8.8267210  7.1504251  3.9439739 13.3167754
```

```
[25]  3.9331742  5.2054586  0.6097514  8.0874264 16.0986024  4.6551948
[31]  5.0889073  6.6017693 10.2738857 16.6394394  5.4733361 15.7197986
[37]  6.6819781  1.6205570 25.0606132  3.8022200  3.6576534 30.6646839
[43]  4.9999949  1.2557795  4.9764633  1.9951685 16.4762087  1.7785269
[49] 11.3563572  4.3010519  1.1843851 17.6175588  8.2691186  7.0229399
[55] 14.1578059 23.6575172  8.8494207 14.7084648  9.8187471  0.2995134
```

Time of the day with arrival time

```
# Start time as POSIXct
start_time <- as.POSIXct("10:00", format = "%H:%M", tz = "UTC")

# Add minutes to the start time

time_of_day <- sapply(arrival_times_s1, function(m) {
  m <- round(m)  # Round to nearest whole number
  new_time <- start_time + (m * 60)  # Add minutes converted to seconds
  format(new_time, "%H:%M")  # Format as "HH:MM"
})

print(time_of_day)
```

```
 [1] "10:15" "10:20" "10:21" "10:38" "10:48" "10:57" "11:02" "11:28" "11:31"
[10] "11:38" "11:44" "11:45" "11:46" "11:59" "12:10" "12:20" "12:37" "12:42"
[19] "13:16" "13:19" "13:20" "13:24" "13:27" "13:28" "13:31" "14:06" "14:15"
[28] "14:21" "14:28" "14:52" "15:35" "15:52" "15:55" "16:19" "16:29" "16:34"
[37] "16:35" "17:03" "17:07" "17:23" "17:25" "17:43" "17:49" "17:50" "18:03"
[46] "18:25" "18:32" "19:09" "19:09" "19:10" "19:11" "19:26" "19:48" "19:49"
[55] "19:52" "20:00" "20:04" "20:32" "20:53" "20:54"
```

Waiting Times

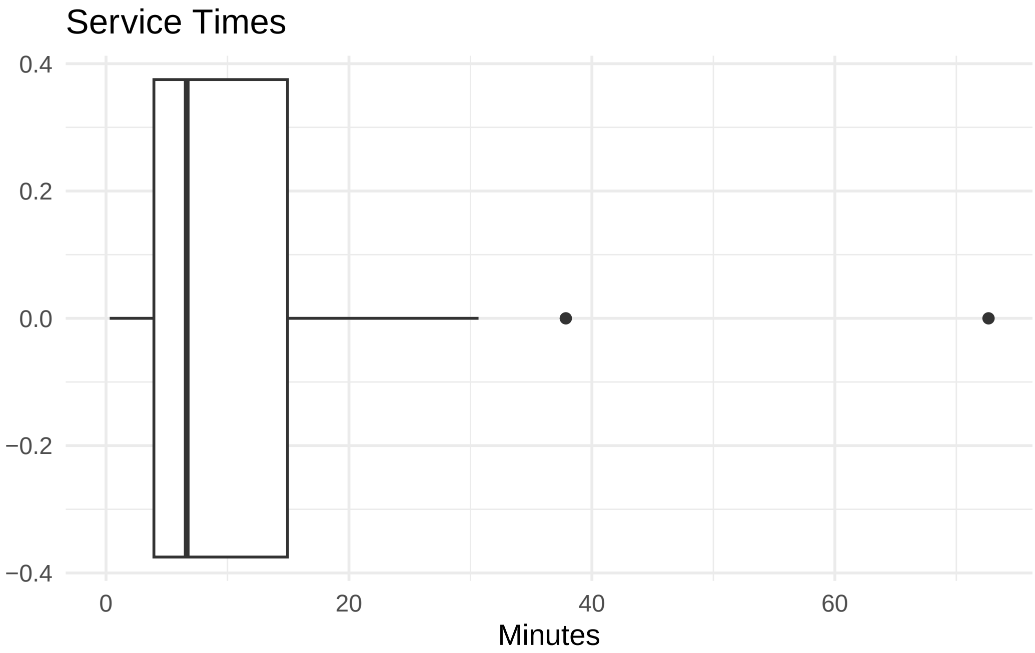
```
# determining waiting times

# for each observation (customer), calculate when the service begins and when it ends
# serving ends = service begins + service time
# service begins: either when the customer walks in, or when the previous customer leaves (a

# compare this to the arrival time
# if arrival time > time service ends then wait time = 0
# but if arrival time < service time ends then wait time = time service ends- arrival time

# variable initialization
waiting_times_s1 <- numeric(length(arrival_times_s1))  # generating times for each customer
service_start <- numeric(length(arrival_times_s1))
service_end <- numeric(length(arrival_times_s1))
current_end <- numeric(0) # service end time for current customer (i)

# iterate over each customer
for (i in 1:length(arrival_times_s1)) {

  # only includes observations where service time > arrival time => which means there is a wa
  # gets rid of observations where service < arrival time => 0 wait time
  if (length(current_end) > 0) {
    current_end <- current_end[current_end > arrival_times_s1[i]]
  }

 if (length(current_end) == 0) {
   # scenario 1: if there is no waiting time, service starts at the customer arrival
    service_start[i] <- arrival_times_s1[i]
  } else {
    # scenario 2: if there is a waiting time, service starts at the end of the previous custo
    previous_end <- service_end[i - 1]
    service_start[i] <- max(arrival_times_s1[i], previous_end)
  }

  # update the service end time for current customer by adding when service starts and how lo
  service_end[i] <- service_start[i] + service_times_s1[i]

  # add this service end time to current end services
  current_end <- c(current_end , service_end[i])

  # update waiting time
  waiting_times_s1[i] <- service_start[i] - arrival_times_s1[i]
}


scen1_sim_results <- data.frame(
  customer = 1:length(arrival_times_s1),
  arrival_time = arrival_times_s1,
  service_length = service_times_s1,
```

```r
  service_start = service_start,
  service_end = service_end,
  waiting_time = waiting_times_s1,
  time_of_day = time_of_day
)

print(head(scen1_sim_results, 5)) # printing first 5 customers
```

```
  customer arrival_time service_length service_start service_end waiting_time
1        1     15.48044      18.322653      15.48044    33.80309      0.00000
2        2     19.99824       3.611812      33.80309    37.41490     13.80485
3        3     21.21837      23.350001      37.41490    60.76490     16.19654
4        4     37.74527       6.156491      60.76490    66.92140     23.01963
5        5     48.04027      72.648544      66.92140   139.56994     18.88113
  time_of_day
1       10:15
2       10:20
3       10:21
4       10:38
5       10:48
```

Serving and Waiting Times Analysis

```r
# boxplot(service_times, horizontal= TRUE, main= "Service Times", xlab= "Minutes")

scen1_sim_results %>%
  ggplot(aes(x= service_length)) +
  geom_boxplot() +
  labs(
    x= "Minutes",
    title = "Service Times"
  ) +
  theme_minimal()
```



```
mean(service_times_s1)
```

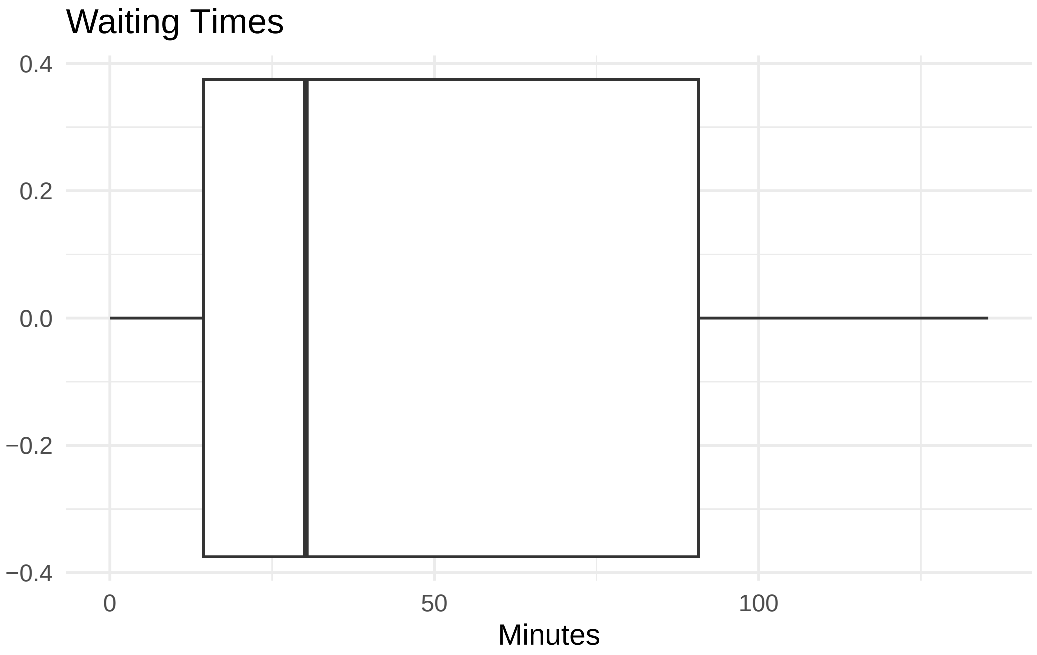
```
[1] 10.65808
```

The average service time is 11 minutes, with the data skewed right, consistent with an exponential distribution. This indicates that service times tend to lower.

```
# boxplot(waiting_times_s1, horizontal= TRUE, main= "Waiting Times", xlab= "Minutes")

mean(waiting_times_s1)
```

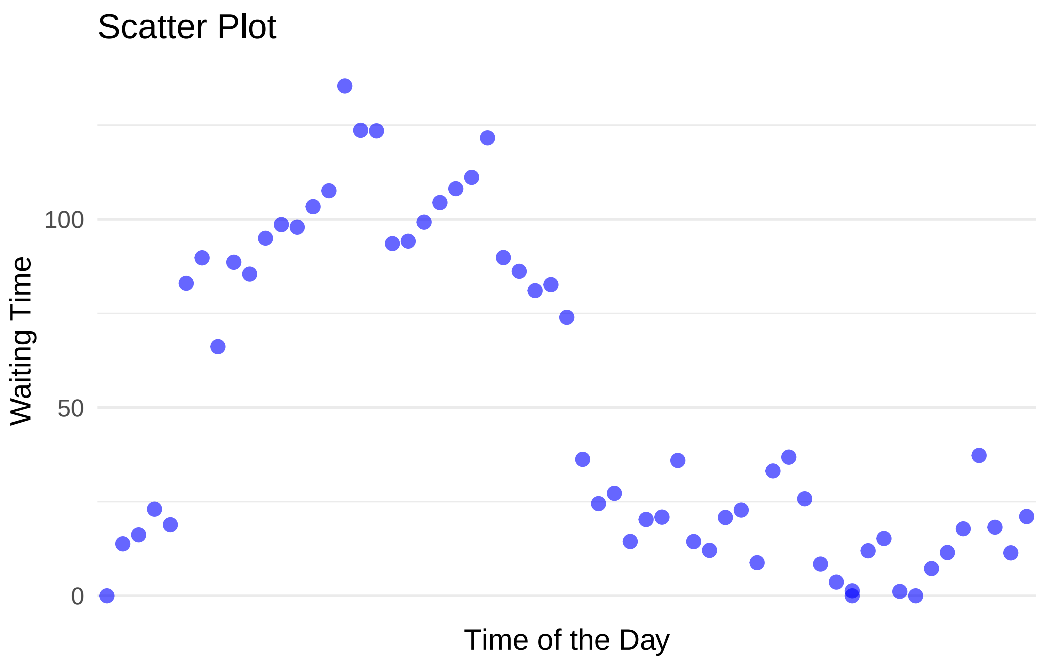
```
[1] 50.594
```

```
scen1_sim_results %>%
  ggplot(aes(x= waiting_time)) +
  geom_boxplot() +
  labs(
    x= "Minutes",
    title = "Waiting Times"
  ) +
  theme_minimal()
```

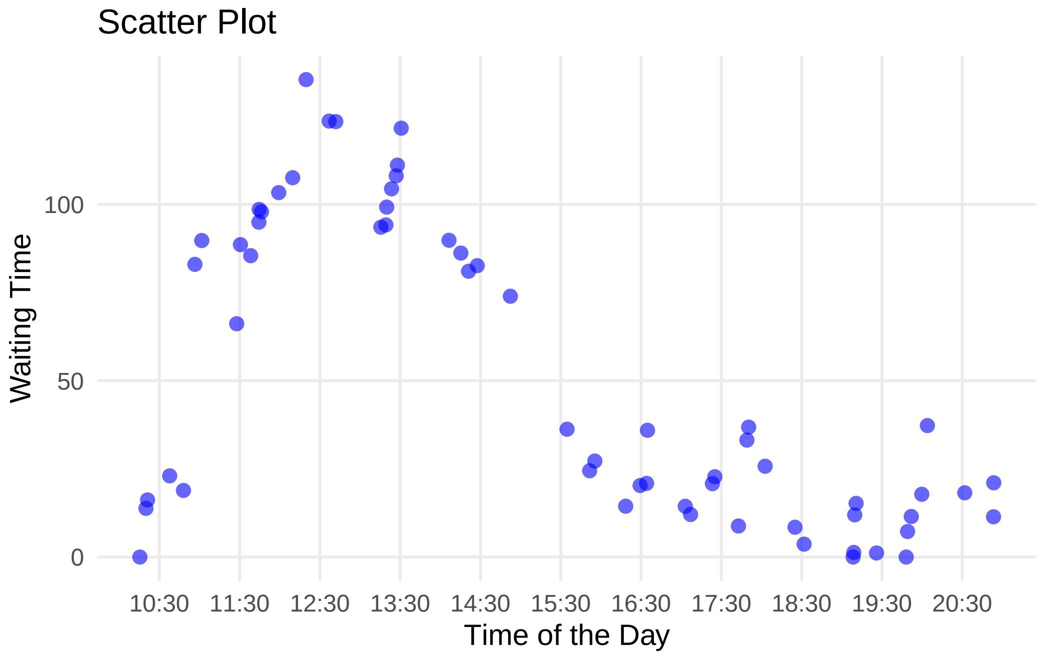


Waiting times tends to be slightly right-skewed and on average, the waiting time is 51 minutes.

```
#Label for 30 min interval
breaks <- seq(30, 720, by = 30)
labels <- sprintf("%02d:%02d", 10 + breaks %/% 60, breaks %% 60)

ggplot(scen1_sim_results, aes(x = time_of_day, y = waiting_times_s1)) +
  geom_point(color = "blue", size = 2, alpha = 0.6) +
    scale_x_discrete(
    breaks = breaks,
    labels = labels
  ) +
  labs(
    title = "Scatter Plot",
    x = "Time of the Day",
    y = "Waiting Time"
  ) +
  theme_minimal()
```



```
library(ggplot2)


# Custom breaks and labels for 30-minute intervals
breaks <- seq(30, 720, by = 60)
labels <- sprintf("%02d:%02d", 10 + breaks %/% 60, breaks %% 60)

# Scatter plot with x-axis as numeric time in minutes
ggplot(scen1_sim_results, aes(x = arrival_times_s1, y = waiting_times_s1)) +
  geom_point(color = "blue", size = 2, alpha = 0.6) +
  scale_x_continuous(
    breaks = breaks,
    labels = labels
  ) +
  labs(
    title = "Scatter Plot",
    x = "Time of the Day",
    y = "Waiting Time"
  ) +
  theme_minimal()+
  theme(panel.grid.minor = element_blank())
```



Scenario 2

Arrival and Service

Assumptions:

1. 5 dining tables and L chefs with operating hours 10am - 10pm
2. each table only seats one customer
3. service time modeled by an exponential distribution with rate $S = 3L$, so that the more chefs there are, the faster the service times become **(this is not very realistic)**

```r
# first, we generate the arrival times similar in scenario 1
lambdaA <- 24 # per hour
opening_time <- hm("10:00")
closing_time <- hm("22:00")
hours <- hour(closing_time) - hour(opening_time)
total_time <- hours*60 # operating hours in minutes
lambdaA <- lambdaA/60 # per minute
num_chefs = 2

n <- ceiling(lambdaA*total_time) # max number of customers
```

```
W_sample <- rexp(n, rate= lambdaA)
T_sample <- numeric(n)

for(i in 1:n) {
  T_sample[i] <- sum(W_sample[1:i])
}

arrival_times <- T_sample[T_sample <= total_time]

# next, we generate the service times similar to scenario 1
# make a function to do this
calc_service_times <- function(arrivals, chefs) {
  # Ensure rate is per unit time
  minute_rate = (3*chefs) / 60
  services = rexp(length(arrivals), rate = minute_rate)
  return(services) # in minutes
}
# if we only have one chef
service_times <- calc_service_times(arrivals = arrival_times, chefs = num_chefs)
```

Waiting Times

To model waiting times, we iterate through the day minute by minute.

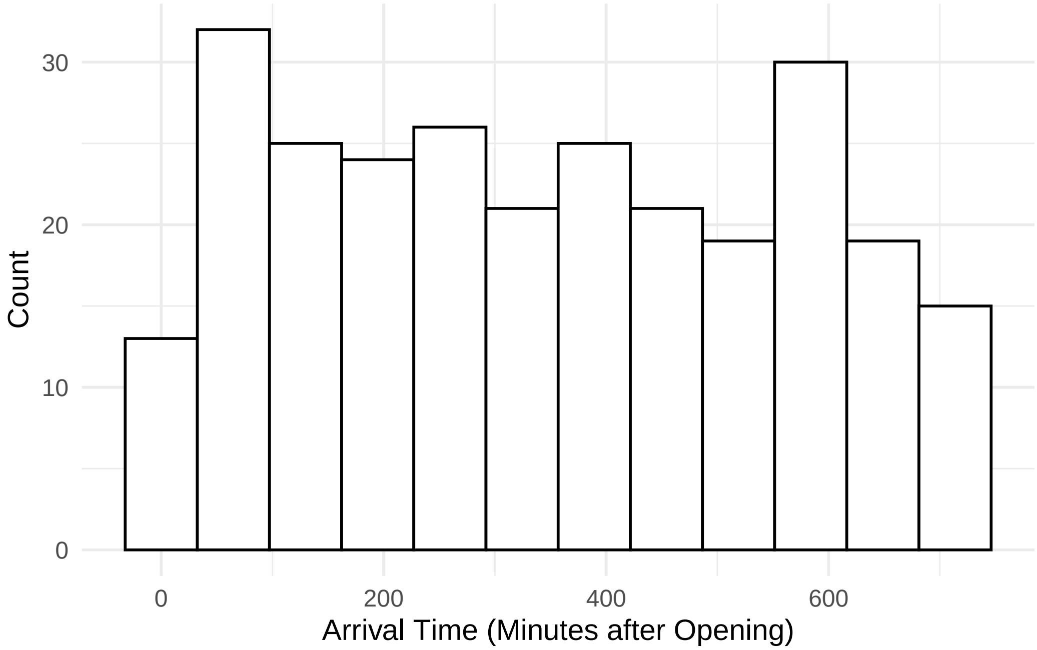
```
tables <- 5
arrival_times_temp <- arrival_times

# number of people in line each minute
queue_size_history <- numeric(total_time)

# number of tables occupied each minute
occupied_tables_history <- rep(0, total_time)

# timer to track remaining waiting time for each table in the restaurant
# each element is one table in the restaurant
# -1 means empty
# otherwise, number of remaining service minutes
tables_timer <- rep(-1, tables)

# the amount of minutes each customer of that day waited
waiting_times <- numeric(0)
```

```
# the arrival_times indices of the people currently in line
# in order to know how long their eventual service time will be
queue <- numeric(0)

# an internal counter separate from the time
customers_entered <- 0
for (i in 1:total_time) {
  occupied_tables_history[i+1] = occupied_tables_history[i]

  # update the waiting timer for all occupied tables
  tables_timer[tables_timer > 0] <- tables_timer[tables_timer > 0] - 1
  # update the number of available tables in the next minute
  # based on the number of tables who have finished timers
  occupied_tables_history[i+1] = occupied_tables_history[i+1] - sum(tables_timer == 0)
  # mark the finished tables as available tables for the next minute
  tables_timer[tables_timer == 0] <- tables_timer[tables_timer == 0] - 1

  # has the next customer arrived?
  if(length(arrival_times_temp) > 0){
    if(arrival_times_temp[1] < i) {
      # if so, add them to the back of the queue
      queue = c(queue, as.integer(customers_entered+1)) # add 1 for 1-indexing
      # remove the 1st element of arrival_times
      arrival_times_temp = arrival_times_temp[-1]
      # start the waiting timer for this customer by appending 0
      waiting_times = c(waiting_times, 0)

      customers_entered = customers_entered + 1
    }
  }
  # are any tables currently open and there is a person in line?
  if(occupied_tables_history[i+1] < tables & length(queue) > 0) {
    # if so, then seat the first person in line
    # at the first available table
    for (j in 1:tables) {
      if(tables_timer[j] == -1) {
        # queue[1] has the customer index of the first person in line
        tables_timer[j] = round(service_times[queue[1]])
        break
      }
    }
    # the next minute there will be one more occupied table
    occupied_tables_history[i+1] = occupied_tables_history[i+1] + 1
    # remove the first person in the queue
    queue = queue[-1]
  }
  # update the waiting time for each person in the queue
  for (customer_index in queue) {
    waiting_times[customer_index] = waiting_times[customer_index] + 1
  }
  # keep track of how long the line is at each minute
  queue_size_history[i] = length(queue)
}

occupied_tables_history <- occupied_tables_history[-1]
```

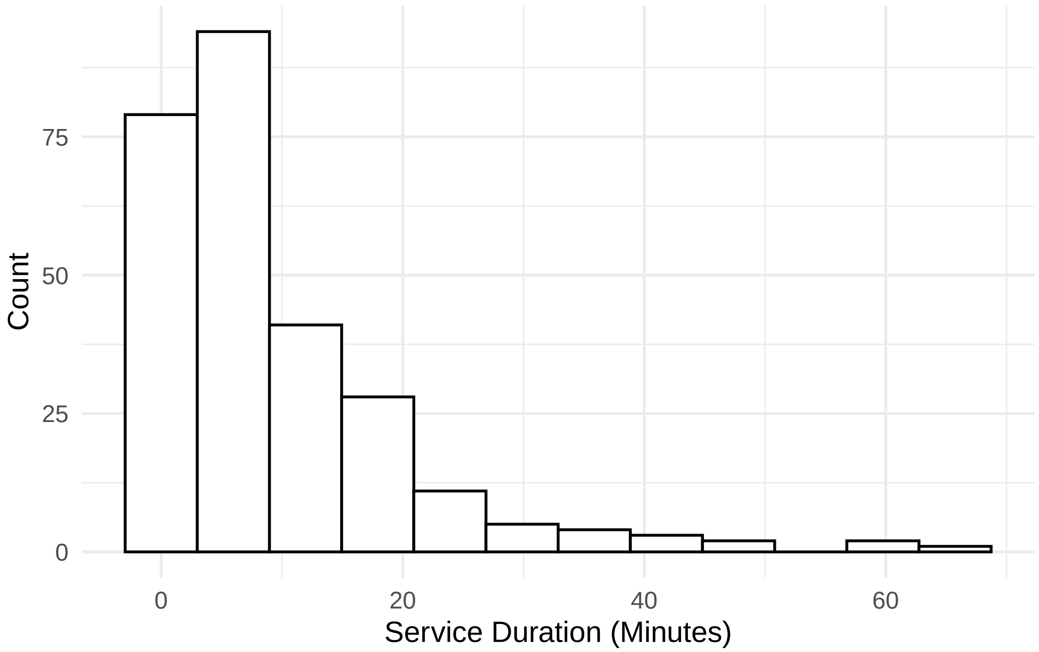
```
scen2_sim_results_by_customer <- data.frame(
  customer = 1:length(arrival_times),
  arrival_time = arrival_times,
  service_length = service_times,
  waiting_time = waiting_times
)
```

```
scen2_sim_results_by_minute <- data.frame(
  minutes_since_opening = 1:total_time,
  time_of_day = I(lapply(1:total_time, function(i) opening_time + minutes(i))),
  queue_size = queue_size_history,
  occupied_tables = occupied_tables_history
)
```

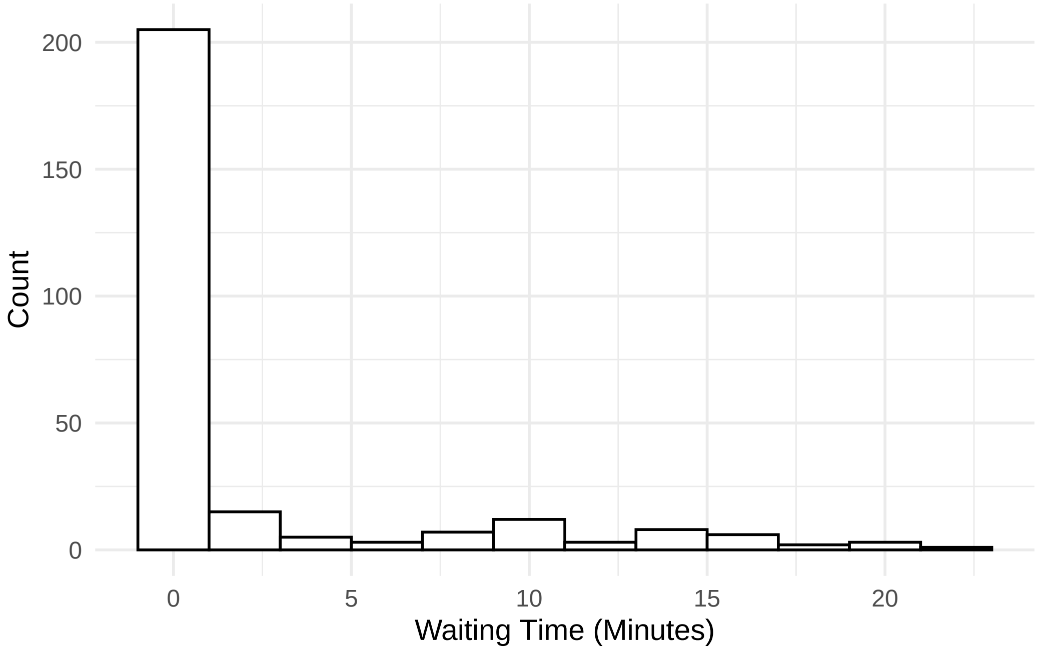
```
scen2_sim_results_by_customer |>
  ggplot(aes(x = arrival_time)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Arrival Time (Minutes after Opening)",
    y = "Count"
  ) +
  theme_minimal()
```



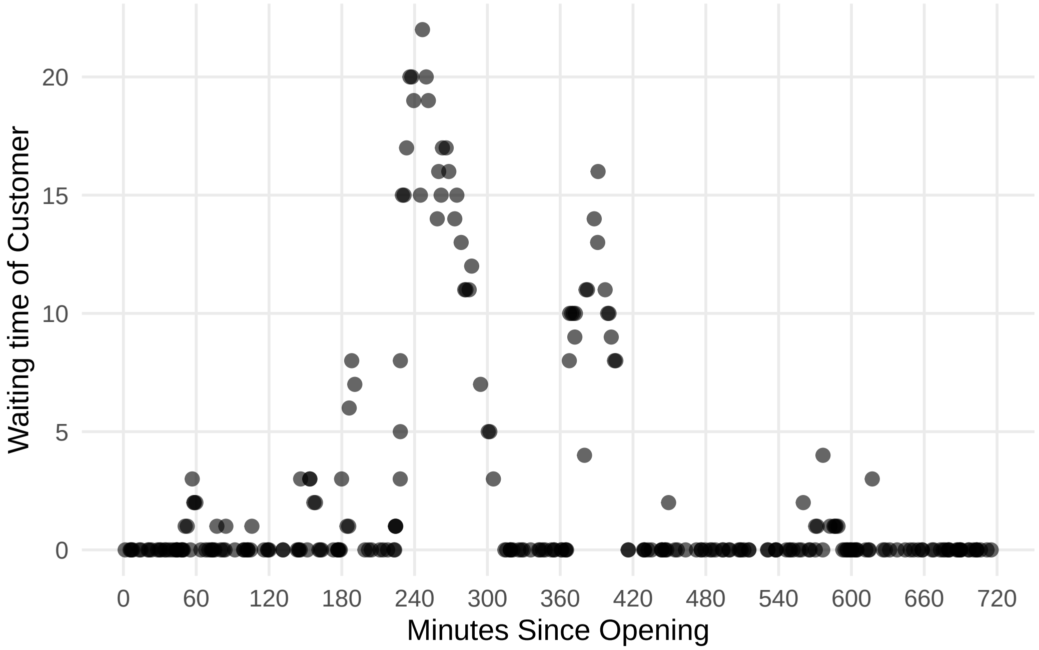
```r
scen2_sim_results_by_customer |>
  ggplot(aes(x = service_length)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Service Duration (Minutes)",
    y = "Count"
  ) +
  theme_minimal()
```



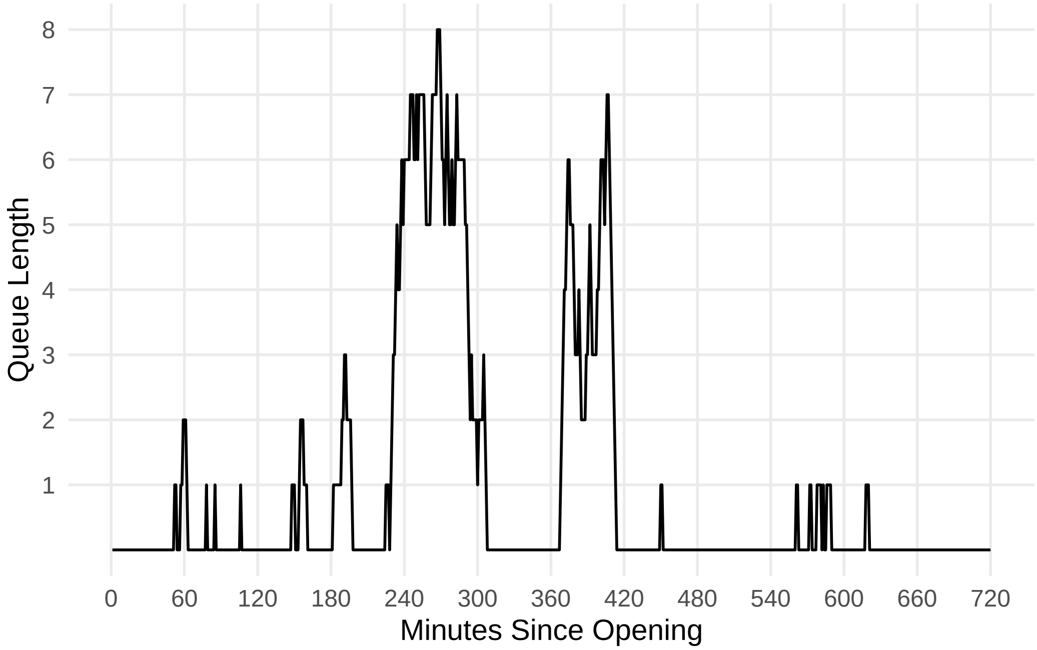
```r
scen2_sim_results_by_customer |>
  ggplot(aes(x = waiting_time)) +
  geom_histogram(bins = 12, color = "black", fill = "white") +
  labs(
    x = "Waiting Time (Minutes)",
    y = "Count"
  ) +
  theme_minimal()
```



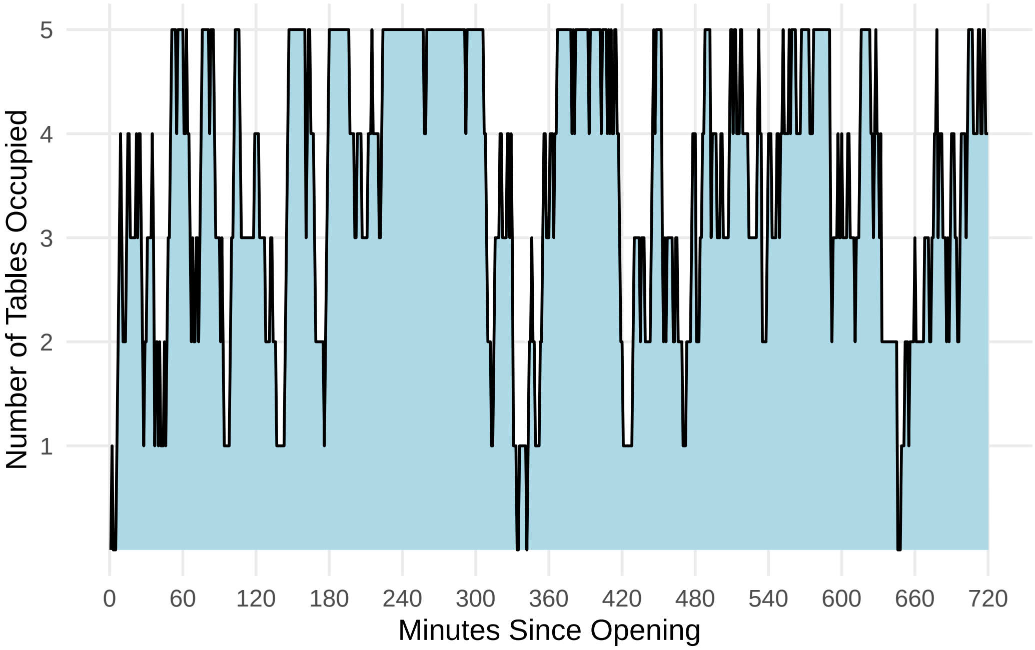
```
scen2_sim_results_by_customer |>
  ggplot(aes(x = arrival_time, y = waiting_time)) +
  geom_point(size = 2, alpha = 0.6) +
  scale_x_continuous(breaks = seq(0, total_time, by = 60)) +
  labs(
    x = "Minutes Since Opening",
    y = "Waiting time of Customer"
  ) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank())
```



```r
scen2_sim_results_by_minute |>
  ggplot(aes(x = minutes_since_opening, y = queue_size)) +
  geom_line() +
  scale_y_continuous(breaks = seq(1, max(queue_size_history), by = 1)) +
  scale_x_continuous(breaks = seq(0, total_time, by = 60)) +
  labs(
    x = "Minutes Since Opening",
    y = "Queue Length"
  ) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank())
```



```r
scen2_sim_results_by_minute |>
  ggplot(aes(x = minutes_since_opening, y = occupied_tables)) +
  geom_area(fill = "lightblue") +
  geom_line() +
  scale_y_continuous(breaks = seq(1, tables, by = 1)) +
  scale_x_continuous(breaks = seq(0, total_time, by = 60)) +
  labs(
    x = "Minutes Since Opening",
    y = "Number of Tables Occupied"
  ) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank())
```



Restaurant Profits

Assumptions:

1. each customer spends $50 per meal (customers who are still in the queue when the restaurant closes won't pay)
2. each chef earns a wage of $40 per hour (paid for the entire duration of the restaurant's operating hours)
3. Customer will not wait longer than 30 minutes

```
# assumption of five tables and 3 chefs

# setting number of chefs to a certain number
# chefs <- 3
# meal <- 50 #50 dollars per meal
# wage <- 12*40 #40 dollars per hour for 12 hrs of work
# revenue <- total_customers * meal
# costs <- wage*chefs
# profit <- revenue - costs
# profit
# only including customers who are waiting for less than 30 mins because will leave the line

filtered_customers <- which(waiting_times < 30)
total_customers <- length(filtered_customers)
total_customers
```

```
[1] 270
```

```
meal <- 50 #50 dollars per meal
wage <- 12*40 #40 dollars per hour for 12 hrs of work

profit_table <- data.frame(
  Chefs = integer(),
  Revenue = numeric(),
  Costs = numeric(),
  Profit = numeric()
)

# if people get mad, the meal will only be 25 because they are mad

for (chefs in 1:10) {

  revenue <- total_customers * meal

  # Calculate costs (wage per chef * number of chefs)
  costs <- wage * chefs

  # Calculate profit
  profit <- revenue - costs

  profit_table <- rbind(profit_table, data.frame(Chefs = chefs, Revenue = revenue, Costs = co

}

print(profit_table)
```

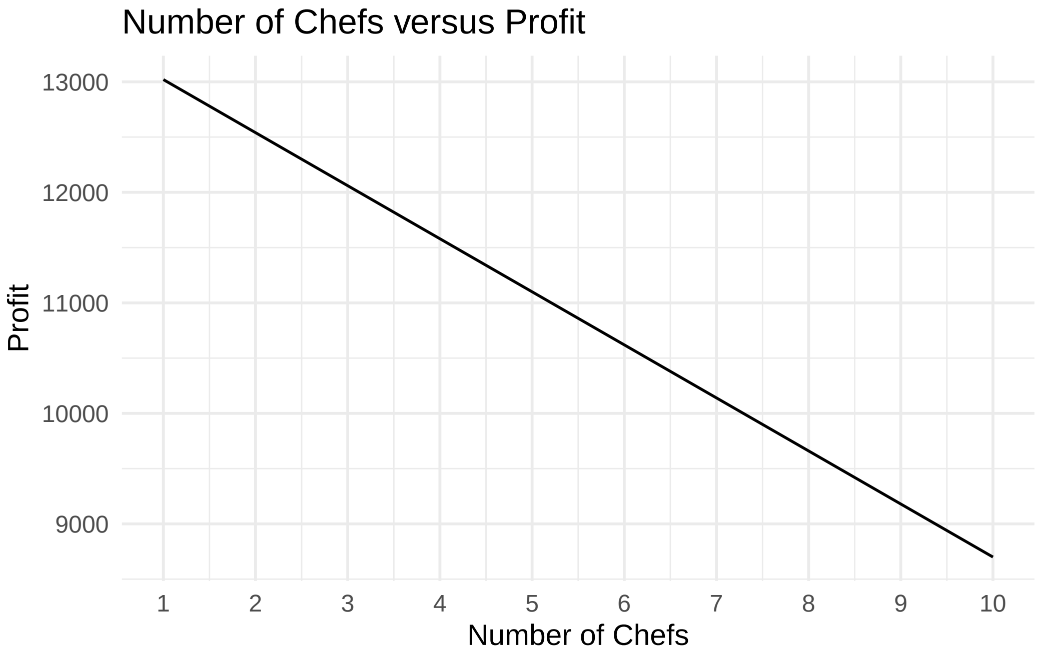
```
   Chefs Revenue Costs Profit
1      1   13500   480  13020
2      2   13500   960  12540
3      3   13500  1440  12060
4      4   13500  1920  11580
```

```
5       5    13500   2400   11100
6       6    13500   2880   10620
7       7    13500   3360   10140
8       8    13500   3840    9660
9       9    13500   4320    9180
10     10    13500   4800    8700
```

```r
profit_table
```

```
   Chefs  Revenue  Costs  Profit
1       1    13500    480   13020
2       2    13500    960   12540
3       3    13500   1440   12060
4       4    13500   1920   11580
5       5    13500   2400   11100
6       6    13500   2880   10620
7       7    13500   3360   10140
8       8    13500   3840    9660
9       9    13500   4320    9180
10     10    13500   4800    8700
```

```r
profit_table %>%
  ggplot(aes(x = Chefs, y = Profit)) +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 10, 1))+
  labs(
    x= "Number of Chefs",
    y= "Profit",
    title= "Number of Chefs versus Profit"
  ) +
  theme_minimal()
```



Maximizing Profits

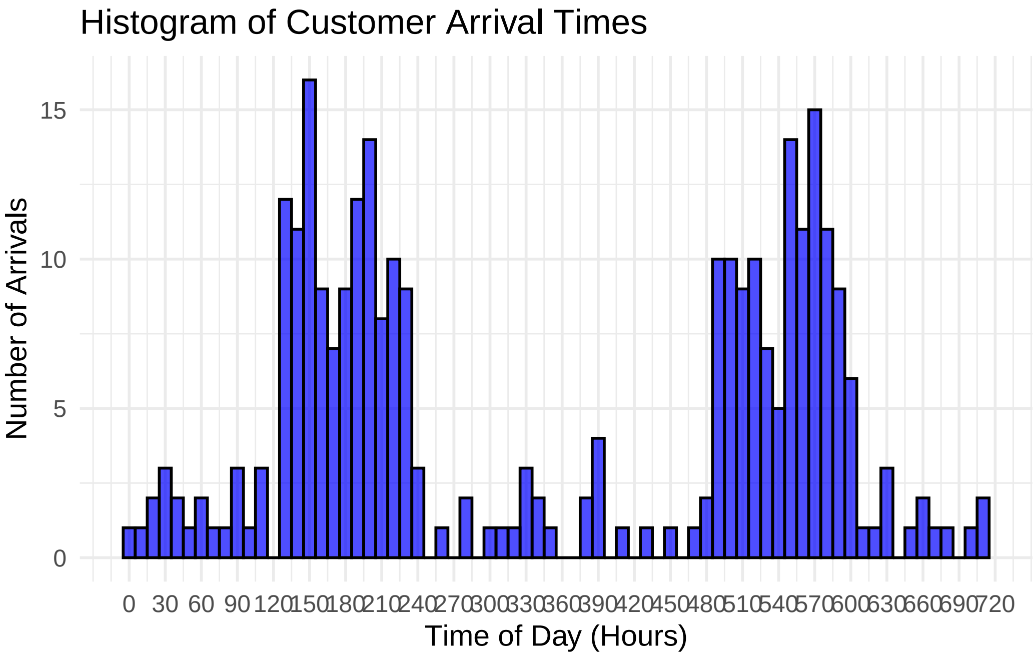
Should we run this simulation multiple times to create a PDF of the total daily profits? How many chefs should we hire?

Down-time of Restaurant

How does the occupancy of the restaurant vary throughout the day? Does that inform any of our recommendations?

Scenario 3

```
simulate_differential_arrival_times <- function(
    lunch_peak_start,lunch_peak_end,
    dinner_peak_start, dinner_peak_end,
    lambda_down, lambda_peak, total_time
) {
  # Convert arrival rates to per minute
  rate_down <- lambda_down / 60
  rate_peak <- lambda_peak / 60

  # Convert peak times to minutes from opening
  lunch_peak_start_min <- as.numeric(as.duration(lunch_peak_start - opening_time), units = "m
  lunch_peak_end_min <- as.numeric(as.duration(lunch_peak_end - opening_time), units = "minut

  dinner_peak_start_min <- as.numeric(as.duration(dinner_peak_start - opening_time), units =
  dinner_peak_end_min <- as.numeric(as.duration(dinner_peak_end - opening_time), units = "min

  # Initialize list to store arrival times
  arrival_times <- numeric()
  current_time <- 0  # Start at 0 minutes (opening time)

  # Generate arrival times
  while (current_time < total_time) {
    # Determine the arrival rate based on current time
    if ((current_time >= lunch_peak_start_min && current_time < lunch_peak_end_min) ||
        (current_time >= dinner_peak_start_min && current_time < dinner_peak_end_min)) {
      arrival_rate <- rate_peak  # Peak time rate
    } else {
      arrival_rate <- rate_down  # Downtime rate
    }

    # Generate the next interarrival time from the exponential distribution
    next_arrival <- rexp(1, arrival_rate)

    # Update the current time
    current_time <- current_time + next_arrival

    # If within the operating hours, add the arrival time to the list
    if (current_time < total_time) {
      arrival_times <- c(arrival_times, current_time)
    }
  }

  # Return arrival_times
  return(arrival_times)
}

arrival_times <- simulate_differential_arrival_times(
  lunch_peak_start = hm("12:00"),
  lunch_peak_end = hm("14:00"),
  dinner_peak_start = hm("18:00"),
  dinner_peak_end = hm("20:00"),
  lambda_down = 6, lambda_peak = 60, total_time = total_time)

ggplot(data = data.frame(arrival_times), aes(x = arrival_times)) +
  geom_histogram(binwidth = 10, fill = "blue", color = "black", alpha = 0.7) +
  labs(
    title = "Histogram of Customer Arrival Times",
    x = "Time of Day (Hours)",
    y = "Number of Arrivals"
  ) +
  scale_x_continuous(
    breaks = seq(0, 720, by = 30)
  ) +
  theme_minimal()
```



```
restaurant_sim <- function(arrivals, chefs, tables, minutes) {
  # calculate service times
  service_times = rexp(length(arrivals), rate = (3*chefs) / 60)

  # set up tracking
  arrival_times_temp <- arrival_times
```

```
queue_size_history <- numeric(total_time)

# number of tables occupied each minute
occupied_tables_history <- rep(0, total_time)

# timer to track remaining waiting time for each table in the restaurant
# each element is one table in the restaurant
# -1 means empty
# otherwise, number of remaining service minutes
tables_timer <- rep(-1, tables)

# the amount of minutes each customer of that day waited
waiting_times <- numeric(0)

# the arrival_times indices of the people currently in line
# in order to know how long their eventual service time will be
queue <- numeric(0)

# an internal counter separate from the time
customers_entered <- 0

# iterate through the day
for (i in 1:total_time) {
  occupied_tables_history[i+1] = occupied_tables_history[i]

  # update the waiting timer for all occupied tables
  tables_timer[tables_timer > 0] <- tables_timer[tables_timer > 0] - 1
  # update the number of available tables in the next minute
  # based on the number of tables who have finished timers
  occupied_tables_history[i+1] = occupied_tables_history[i+1] - sum(tables_timer == 0)
  # mark the finished tables as available tables for the next minute
  tables_timer[tables_timer == 0] <- tables_timer[tables_timer == 0] - 1

  # has the next customer arrived?
  if(length(arrival_times_temp) > 0){
    if(arrival_times_temp[1] < i) {
      # if so, add them to the back of the queue
      queue = c(queue, as.integer(customers_entered+1)) # add 1 for 1-indexing
      # remove the 1st element of arrival_times
      arrival_times_temp = arrival_times_temp[-1]
      # start the waiting timer for this customer by appending 0
      waiting_times = c(waiting_times, 0)
      customers_entered = customers_entered + 1
    }
  }
  # are any tables currently open and there is a person in line?
  if(occupied_tables_history[i+1] < tables & length(queue) > 0) {
    # if so, then seat the first person in line
    # at the first available table
    for (j in 1:tables) {
      if(tables_timer[j] == -1) {
        # queue[1] has the customer index of the first person in line
        tables_timer[j] = round(service_times[queue[1]])
        break
      }
    }
    # the next minute there will be one more occupied table
    occupied_tables_history[i+1] = occupied_tables_history[i+1] + 1
    # remove the first person in the queue
    queue = queue[-1]
  }
  # update the waiting time for each person in the queue
  for (customer_index in queue) {
    waiting_times[customer_index] = waiting_times[customer_index] + 1
  }
  # keep track of how long the line is at each minute
  queue_size_history[i] = length(queue)
}

occupied_tables_history <- occupied_tables_history[-1]

# calculate outputs (the things we actually care about) from the simulation

# average waiting time across all customers
avg_waiting_time <- mean(waiting_times)
# number of customers who waited >30 minutes (and made us less money)
long_waits <- length(waiting_times[waiting_times > 30])
# average queue length throughout the day
avg_queue_length <- mean(queue_size_history)
# maximum queue length that day
max_queue_length <- max(queue_size_history)
# average table occupancy in the restaurant
avg_tables_occupied <- mean(occupied_tables_history)
```

```
  # return all of it, as a data frame with one row
  sim_output <- data.frame(
    num_chefs = chefs,
    num_tables = tables,
    avg_waiting_time = avg_waiting_time,
    long_waits = long_waits,
    avg_queue_length = avg_queue_length,
    max_queue_length = max_queue_length,
    avg_tables_occupied = avg_tables_occupied
  )
  return(sim_output)
}

total_time <- 720

df <- numeric(7)
for(a in 1:10) {
  num_chefs = a
  for(b in 1:10) {
    num_tables = b
    for(i in 1:5) {
      arrival_times <- simulate_differential_arrival_times(
      lunch_peak_start = hm("12:00"),
      lunch_peak_end = hm("14:00"),
      dinner_peak_start = hm("18:00"),
      dinner_peak_end = hm("20:00"),
      lambda_down = 6, lambda_peak = 60, total_time = total_time)
      df <- rbind(df, restaurant_sim(arrival_times, num_chefs, num_tables, total_time))
    }
  }
}

df <- df[-1, ]

df
```

```
    num_chefs num_tables avg_waiting_time long_waits avg_queue_length
2           1          1     2.957054e+02        250     1.059611e+02
3           1          1     2.830039e+02        250     1.014097e+02
4           1          1     2.960451e+02        278     1.184181e+02
5           1          1     3.241238e+02        297     1.382028e+02
6           1          1     3.133322e+02        282     1.257681e+02
```

7	1	2	2.924529e+02	254	1.121069e+02
8	1	2	2.610685e+02	233	8.992361e+01
9	1	2	2.772426e+02	264	1.047361e+02
10	1	2	2.210797e+02	234	7.707083e+01
11	1	2	2.649170e+02	265	1.019194e+02
12	1	3	2.305072e+02	256	8.900139e+01
13	1	3	2.342045e+02	248	8.587500e+01
14	1	3	2.316866e+02	269	9.138750e+01
15	1	3	2.520753e+02	275	1.022306e+02
16	1	3	2.171111e+02	204	7.056111e+01
17	1	4	1.867224e+02	256	7.287361e+01
18	1	4	1.894715e+02	239	6.920972e+01
19	1	4	2.024646e+02	276	8.351667e+01
20	1	4	1.542308e+02	266	6.404861e+01
21	1	4	1.875360e+02	244	7.240972e+01
22	1	5	1.012016e+02	203	3.415556e+01
23	1	5	1.725077e+02	241	6.229444e+01
24	1	5	1.486292e+02	235	5.511667e+01
25	1	5	1.755072e+02	249	6.727778e+01
26	1	5	1.658073e+02	274	6.931667e+01
27	1	6	1.214176e+02	229	4.603750e+01
28	1	6	9.409363e+01	231	3.489306e+01
29	1	6	1.089130e+02	210	3.827083e+01
30	1	6	1.005388e+02	217	3.602639e+01
31	1	6	1.155840e+02	211	3.820694e+01
32	1	7	1.150854e+02	255	4.491528e+01
33	1	7	9.142568e+01	244	3.758611e+01
34	1	7	5.716154e+01	169	2.064167e+01
35	1	7	1.077040e+02	240	4.143611e+01
36	1	7	1.030353e+02	207	3.649167e+01
37	1	8	6.339847e+01	188	2.298194e+01
38	1	8	6.611475e+01	177	2.240556e+01
39	1	8	7.603679e+01	232	3.157639e+01
40	1	8	9.164583e+01	242	3.665833e+01
41	1	8	8.051163e+01	199	2.885000e+01
42	1	9	5.637818e+01	173	2.153333e+01
43	1	9	4.838403e+01	143	1.767361e+01
44	1	9	7.762921e+01	210	2.878750e+01
45	1	9	5.311290e+01	161	1.829444e+01
46	1	9	6.608230e+01	170	2.230278e+01
47	1	10	5.707483e+01	214	2.330556e+01
48	1	10	2.850204e+01	96	9.698611e+00
49	1	10	5.145183e+01	179	2.150972e+01

50	1	10	2.944151e+01	125	1.083611e+01
51	1	10	4.781852e+01	172	1.793194e+01
52	2	1	2.742148e+02	256	1.028306e+02
53	2	1	2.780361e+02	267	1.069667e+02
54	2	1	2.470619e+02	194	7.205972e+01
55	2	1	2.566511e+02	262	9.909583e+01
56	2	1	2.798659e+02	241	9.562083e+01
57	2	2	2.217306e+02	251	8.345694e+01
58	2	2	1.903123e+02	247	7.110278e+01
59	2	2	2.332923e+02	268	9.202083e+01
60	2	2	1.931419e+02	262	7.940278e+01
61	2	2	1.890547e+02	222	6.721944e+01
62	2	3	1.401099e+02	247	5.487639e+01
63	2	3	1.052792e+02	234	3.874861e+01
64	2	3	1.173700e+02	249	4.450278e+01
65	2	3	1.296882e+02	254	5.025417e+01
66	2	3	1.205674e+02	242	4.722222e+01
67	2	4	8.893431e+01	221	3.384444e+01
68	2	4	7.717667e+01	227	3.215694e+01
69	2	4	8.422794e+01	220	3.181944e+01
70	2	4	7.457447e+01	207	2.920833e+01
71	2	4	7.006618e+01	204	2.646944e+01
72	2	5	4.857285e+01	192	2.037361e+01
73	2	5	4.895122e+01	169	1.951250e+01
74	2	5	6.311314e+01	192	2.401806e+01
75	2	5	4.130579e+01	144	1.388333e+01
76	2	5	4.666008e+01	166	1.639583e+01
77	2	6	2.429464e+01	67	7.558333e+00
78	2	6	3.323311e+01	132	1.366250e+01
79	2	6	2.152692e+01	88	7.773611e+00
80	2	6	4.250974e+01	185	1.818472e+01
81	2	6	1.664314e+01	57	5.894444e+00
82	2	7	3.256877e+01	141	1.216806e+01
83	2	7	1.126199e+01	21	4.238889e+00
84	2	7	1.585283e+01	55	5.834722e+00
85	2	7	1.806897e+01	72	7.277778e+00
86	2	7	1.049597e+01	37	3.615278e+00
87	2	8	1.105285e+01	11	3.776389e+00
88	2	8	1.269366e+01	1	5.006944e+00
89	2	8	1.700000e+01	67	5.950000e+00
90	2	8	1.683398e+01	40	6.055556e+00
91	2	8	1.425869e+01	56	5.129167e+00
92	2	9	9.725564e+00	4	3.593056e+00

93	2	9	6.803448e+00	0	2.740278e+00
94	2	9	7.135036e+00	18	2.715278e+00
95	2	9	9.650000e+00	37	3.484722e+00
96	2	9	7.924188e+00	6	3.048611e+00
97	2	10	3.289963e+00	0	1.229167e+00
98	2	10	3.347826e+00	0	1.283333e+00
99	2	10	2.610294e+00	0	9.861111e-01
100	2	10	4.746269e+00	0	1.766667e+00
101	2	10	2.659004e+00	0	9.638889e-01
102	3	1	2.094315e+02	265	8.493611e+01
103	3	1	2.304173e+02	243	8.512639e+01
104	3	1	2.410627e+02	252	9.073333e+01
105	3	1	1.681250e+02	209	5.417361e+01
106	3	1	2.343036e+02	227	8.037917e+01
107	3	2	1.474321e+02	235	5.733472e+01
108	3	2	1.106444e+02	225	4.149167e+01
109	3	2	1.328957e+02	249	5.131250e+01
110	3	2	1.608276e+02	262	6.477778e+01
111	3	2	1.336227e+02	255	5.066528e+01
112	3	3	7.218147e+01	200	2.596528e+01
113	3	3	6.214074e+01	212	2.330278e+01
114	3	3	5.597153e+01	186	2.184444e+01
115	3	3	7.991349e+01	219	3.207639e+01
116	3	3	4.843515e+01	130	1.607778e+01
117	3	4	3.052245e+01	111	1.038611e+01
118	3	4	3.318450e+01	149	1.249028e+01
119	3	4	3.822407e+01	131	1.279444e+01
120	3	4	4.530797e+01	173	1.736806e+01
121	3	4	3.669349e+01	154	1.330139e+01
122	3	5	1.846617e+01	87	6.822222e+00
123	3	5	1.771761e+01	72	7.406944e+00
124	3	5	1.550485e+01	58	6.654167e+00
125	3	5	1.779848e+01	66	6.501389e+00
126	3	5	2.423256e+01	142	1.013056e+01
127	3	6	9.098039e+00	3	3.222222e+00
128	3	6	1.239941e+01	12	5.820833e+00
129	3	6	7.503759e+00	0	2.772222e+00
130	3	6	1.065603e+01	36	4.173611e+00
131	3	6	1.191575e+01	13	4.518056e+00
132	3	7	1.761246e+00	0	7.069444e-01
133	3	7	2.333333e+00	0	9.527778e-01
134	3	7	1.603774e+00	0	5.902778e-01
135	3	7	3.688889e+00	0	1.383333e+00

136	3	7	4.528053e+00	0	1.905556e+00
137	3	8	1.035857e-01	0	3.611111e-02
138	3	8	9.031142e-01	0	3.625000e-01
139	3	8	3.357143e-01	0	1.305556e-01
140	3	8	1.188192e+00	0	4.472222e-01
141	3	8	1.840580e+00	0	7.055556e-01
142	3	9	1.401961e+00	0	5.958333e-01
143	3	9	3.802281e-03	0	1.388889e-03
144	3	9	9.003559e-01	0	3.513889e-01
145	3	9	2.677419e-01	0	1.152778e-01
146	3	9	1.199301e+00	0	4.763889e-01
147	3	10	0.000000e+00	0	0.000000e+00
148	3	10	1.382900e+00	0	5.166667e-01
149	3	10	1.850575e+00	0	6.708333e-01
150	3	10	0.000000e+00	0	0.000000e+00
151	3	10	1.390977e-01	0	5.138889e-02
152	4	1	1.922458e+02	215	6.408194e+01
153	4	1	2.384101e+02	256	9.205278e+01
154	4	1	1.979403e+02	250	7.367778e+01
155	4	1	1.760034e+02	265	7.235694e+01
156	4	1	2.125788e+02	294	9.182222e+01
157	4	2	6.659684e+01	193	2.340139e+01
158	4	2	1.155068e+02	259	4.748611e+01
159	4	2	7.169778e+01	153	2.240556e+01
160	4	2	7.186742e+01	189	2.635139e+01
161	4	2	8.357679e+01	239	3.401111e+01
162	4	3	3.157692e+01	137	1.140278e+01
163	4	3	3.882609e+01	151	1.488333e+01
164	4	3	4.103957e+01	180	1.584583e+01
165	4	3	3.231439e+01	140	1.184861e+01
166	4	3	2.993657e+01	113	1.114306e+01
167	4	4	1.817308e+01	60	7.875000e+00
168	4	4	1.911184e+01	70	8.069444e+00
169	4	4	2.536393e+01	126	1.074444e+01
170	4	4	1.329562e+01	10	5.059722e+00
171	4	4	2.069424e+01	85	7.990278e+00
172	4	5	5.289683e+00	0	1.851389e+00
173	4	5	4.659864e+00	0	1.902778e+00
174	4	5	6.078652e+00	0	2.254167e+00
175	4	5	9.881890e+00	2	3.486111e+00
176	4	5	5.287823e+00	0	1.990278e+00
177	4	6	3.016835e+00	0	1.244444e+00
178	4	6	2.003731e+00	0	7.458333e-01

179	4	6	2.669065e+00	0	1.030556e+00
180	4	6	1.594828e+00	0	5.138889e-01
181	4	6	2.169014e+00	0	8.555556e-01
182	4	7	6.853933e-01	0	2.541667e-01
183	4	7	4.900000e-01	0	2.041667e-01
184	4	7	1.970037e+00	0	7.305556e-01
185	4	7	1.969112e-01	0	7.083333e-02
186	4	7	4.098361e-03	0	1.388889e-03
187	4	8	1.985560e-01	0	7.638889e-02
188	4	8	0.000000e+00	0	0.000000e+00
189	4	8	1.602606e+00	0	6.833333e-01
190	4	8	0.000000e+00	0	0.000000e+00
191	4	8	0.000000e+00	0	0.000000e+00
192	4	9	0.000000e+00	0	0.000000e+00
193	4	9	0.000000e+00	0	0.000000e+00
194	4	9	7.701613e-01	0	2.652778e-01
195	4	9	0.000000e+00	0	0.000000e+00
196	4	9	1.716418e-01	0	6.388889e-02
197	4	10	0.000000e+00	0	0.000000e+00
198	4	10	0.000000e+00	0	0.000000e+00
199	4	10	0.000000e+00	0	0.000000e+00
200	4	10	0.000000e+00	0	0.000000e+00
201	4	10	0.000000e+00	0	0.000000e+00
202	5	1	1.139123e+02	196	3.607222e+01
203	5	1	1.774100e+02	263	7.392083e+01
204	5	1	1.882226e+02	244	7.162917e+01
205	5	1	1.504419e+02	238	5.578889e+01
206	5	1	2.097664e+02	283	8.856806e+01
207	5	2	6.630935e+01	201	2.560278e+01
208	5	2	7.536949e+01	226	3.088056e+01
209	5	2	5.915693e+01	174	2.251250e+01
210	5	2	6.110345e+01	191	2.215000e+01
211	5	2	4.591150e+01	139	1.441111e+01
212	5	3	1.441870e+01	46	4.926389e+00
213	5	3	2.771970e+01	108	1.016389e+01
214	5	3	2.147619e+01	113	8.769444e+00
215	5	3	2.720588e+01	122	1.027778e+01
216	5	3	1.762414e+01	52	7.098611e+00
217	5	4	6.062963e+00	0	2.273611e+00
218	5	4	8.132404e+00	0	3.241667e+00
219	5	4	7.108303e+00	0	2.734722e+00
220	5	4	6.743682e+00	0	2.594444e+00
221	5	4	1.657895e+00	0	5.250000e-01

222	5	5	4.557823e+00	0	1.861111e+00
223	5	5	2.683849e+00	0	1.084722e+00
224	5	5	7.313433e-01	0	2.722222e-01
225	5	5	4.093985e+00	0	1.512500e+00
226	5	5	2.645390e+00	0	1.036111e+00
227	5	6	2.081481e+00	0	7.805556e-01
228	5	6	1.592308e+00	0	5.750000e-01
229	5	6	5.017794e-01	0	1.958333e-01
230	5	6	3.601695e-01	0	1.180556e-01
231	5	6	5.124555e-01	0	2.000000e-01
232	5	7	2.013652e-01	0	8.194444e-02
233	5	7	0.000000e+00	0	0.000000e+00
234	5	7	0.000000e+00	0	0.000000e+00
235	5	7	1.149425e-02	0	4.166667e-03
236	5	7	5.019763e-01	0	1.763889e-01
237	5	8	0.000000e+00	0	0.000000e+00
238	5	8	0.000000e+00	0	0.000000e+00
239	5	8	0.000000e+00	0	0.000000e+00
240	5	8	0.000000e+00	0	0.000000e+00
241	5	8	0.000000e+00	0	0.000000e+00
242	5	9	0.000000e+00	0	0.000000e+00
243	5	9	0.000000e+00	0	0.000000e+00
244	5	9	0.000000e+00	0	0.000000e+00
245	5	9	0.000000e+00	0	0.000000e+00
246	5	9	0.000000e+00	0	0.000000e+00
247	5	10	0.000000e+00	0	0.000000e+00
248	5	10	0.000000e+00	0	0.000000e+00
249	5	10	0.000000e+00	0	0.000000e+00
250	5	10	0.000000e+00	0	0.000000e+00
251	5	10	0.000000e+00	0	0.000000e+00
252	6	1	8.009705e+01	184	2.636528e+01
253	6	1	1.736678e+02	280	7.187917e+01
254	6	1	1.153682e+02	255	4.742917e+01
255	6	1	1.053843e+02	236	3.922639e+01
256	6	1	8.756574e+01	209	3.052639e+01
257	6	2	5.141219e+01	180	1.992222e+01
258	6	2	4.458219e+01	177	1.808056e+01
259	6	2	3.661484e+01	145	1.439167e+01
260	6	2	4.238603e+01	139	1.601250e+01
261	6	2	3.932982e+01	167	1.556806e+01
262	6	3	9.788927e+00	0	3.929167e+00
263	6	3	1.638745e+01	53	6.168056e+00
264	6	3	1.945608e+01	67	7.998611e+00

265	6	3	1.151931e+01	20	3.727778e+00
266	6	3	1.152734e+01	42	4.098611e+00
267	6	4	2.717172e+00	0	1.120833e+00
268	6	4	7.487273e+00	0	2.859722e+00
269	6	4	3.450758e+00	0	1.265278e+00
270	6	4	5.609023e+00	0	2.072222e+00
271	6	4	2.482490e+00	0	8.861111e-01
272	6	5	1.625000e+00	0	6.319444e-01
273	6	5	1.003425e+00	0	4.069444e-01
274	6	5	1.054264e+00	0	3.777778e-01
275	6	5	7.582418e-01	0	2.875000e-01
276	6	5	1.935065e+00	0	8.277778e-01
277	6	6	5.298013e-01	0	2.222222e-01
278	6	6	7.509158e-01	0	2.847222e-01
279	6	6	0.000000e+00	0	0.000000e+00
280	6	6	1.480144e-01	0	5.694444e-02
281	6	6	0.000000e+00	0	0.000000e+00
282	6	7	0.000000e+00	0	0.000000e+00
283	6	7	0.000000e+00	0	0.000000e+00
284	6	7	0.000000e+00	0	0.000000e+00
285	6	7	0.000000e+00	0	0.000000e+00
286	6	7	0.000000e+00	0	0.000000e+00
287	6	8	0.000000e+00	0	0.000000e+00
288	6	8	1.445783e-01	0	5.000000e-02
289	6	8	0.000000e+00	0	0.000000e+00
290	6	8	0.000000e+00	0	0.000000e+00
291	6	8	0.000000e+00	0	0.000000e+00
292	6	9	0.000000e+00	0	0.000000e+00
293	6	9	0.000000e+00	0	0.000000e+00
294	6	9	0.000000e+00	0	0.000000e+00
295	6	9	0.000000e+00	0	0.000000e+00
296	6	9	0.000000e+00	0	0.000000e+00
297	6	10	0.000000e+00	0	0.000000e+00
298	6	10	0.000000e+00	0	0.000000e+00
299	6	10	0.000000e+00	0	0.000000e+00
300	6	10	0.000000e+00	0	0.000000e+00
301	6	10	0.000000e+00	0	0.000000e+00
302	7	1	9.715734e+01	234	3.859306e+01
303	7	1	8.621603e+01	244	3.436667e+01
304	7	1	1.179818e+02	238	4.489861e+01
305	7	1	6.267273e+01	152	1.915000e+01
306	7	1	1.362672e+02	241	4.958611e+01
307	7	2	2.893233e+01	112	1.068889e+01

308	7	2	2.019679e+01	64	6.984722e+00
309	7	2	2.716236e+01	114	1.022361e+01
310	7	2	2.898120e+01	115	1.070694e+01
311	7	2	2.426923e+01	118	9.640278e+00
312	7	3	9.146953e+00	0	3.544444e+00
313	7	3	9.615385e+00	29	3.645833e+00
314	7	3	7.127413e+00	0	2.563889e+00
315	7	3	9.688889e+00	0	4.238889e+00
316	7	3	5.627451e+00	0	1.993056e+00
317	7	4	1.228571e+00	0	4.180556e-01
318	7	4	1.267559e+00	0	5.263889e-01
319	7	4	1.621429e+00	0	6.305556e-01
320	7	4	4.602524e+00	0	2.026389e+00
321	7	4	1.600707e+00	0	6.291667e-01
322	7	5	4.943396e-01	0	1.819444e-01
323	7	5	9.840637e-01	0	3.430556e-01
324	7	5	4.773519e-01	0	1.902778e-01
325	7	5	2.150943e-01	0	7.916667e-02
326	7	5	1.590747e+00	0	6.208333e-01
327	7	6	4.905660e-02	0	1.805556e-02
328	7	6	0.000000e+00	0	0.000000e+00
329	7	6	0.000000e+00	0	0.000000e+00
330	7	6	0.000000e+00	0	0.000000e+00
331	7	6	0.000000e+00	0	0.000000e+00
332	7	7	0.000000e+00	0	0.000000e+00
333	7	7	0.000000e+00	0	0.000000e+00
334	7	7	2.322581e-01	0	1.000000e-01
335	7	7	0.000000e+00	0	0.000000e+00
336	7	7	0.000000e+00	0	0.000000e+00
337	7	8	0.000000e+00	0	0.000000e+00
338	7	8	0.000000e+00	0	0.000000e+00
339	7	8	0.000000e+00	0	0.000000e+00
340	7	8	0.000000e+00	0	0.000000e+00
341	7	8	1.066667e-01	0	4.444444e-02
342	7	9	0.000000e+00	0	0.000000e+00
343	7	9	0.000000e+00	0	0.000000e+00
344	7	9	0.000000e+00	0	0.000000e+00
345	7	9	0.000000e+00	0	0.000000e+00
346	7	9	0.000000e+00	0	0.000000e+00
347	7	10	0.000000e+00	0	0.000000e+00
348	7	10	0.000000e+00	0	0.000000e+00
349	7	10	0.000000e+00	0	0.000000e+00
350	7	10	0.000000e+00	0	0.000000e+00

351	7	10	0.000000e+00	0	0.000000e+00
352	8	1	6.120307e+01	188	2.218611e+01
353	8	1	6.317917e+01	155	2.105972e+01
354	8	1	7.988889e+01	217	2.895972e+01
355	8	1	6.481140e+01	153	2.052361e+01
356	8	1	1.070090e+02	294	4.934306e+01
357	8	2	2.630682e+01	120	9.645833e+00
358	8	2	1.515574e+01	21	5.136111e+00
359	8	2	2.241825e+01	89	8.188889e+00
360	8	2	1.728512e+01	61	5.809722e+00
361	8	2	1.890637e+01	62	7.011111e+00
362	8	3	4.022814e+00	0	1.469444e+00
363	8	3	5.635379e+00	0	2.168056e+00
364	8	3	2.923875e+00	0	1.173611e+00
365	8	3	8.286275e+00	0	2.934722e+00
366	8	3	1.106873e+01	9	4.473611e+00
367	8	4	1.271429e+00	0	4.944444e-01
368	8	4	1.094340e-01	0	4.027778e-02
369	8	4	1.810811e+00	0	7.444444e-01
370	8	4	1.268657e+00	0	4.722222e-01
371	8	4	5.870307e-01	0	2.388889e-01
372	8	5	1.066901e+00	0	4.208333e-01
373	8	5	1.408451e-02	0	5.555556e-03
374	8	5	0.000000e+00	0	0.000000e+00
375	8	5	4.269663e-01	0	1.583333e-01
376	8	5	3.824561e-01	0	1.513889e-01
377	8	6	0.000000e+00	0	0.000000e+00
378	8	6	0.000000e+00	0	0.000000e+00
379	8	6	0.000000e+00	0	0.000000e+00
380	8	6	0.000000e+00	0	0.000000e+00
381	8	6	0.000000e+00	0	0.000000e+00
382	8	7	0.000000e+00	0	0.000000e+00
383	8	7	0.000000e+00	0	0.000000e+00
384	8	7	0.000000e+00	0	0.000000e+00
385	8	7	0.000000e+00	0	0.000000e+00
386	8	7	0.000000e+00	0	0.000000e+00
387	8	8	0.000000e+00	0	0.000000e+00
388	8	8	0.000000e+00	0	0.000000e+00
389	8	8	0.000000e+00	0	0.000000e+00
390	8	8	0.000000e+00	0	0.000000e+00
391	8	8	0.000000e+00	0	0.000000e+00
392	8	9	0.000000e+00	0	0.000000e+00
393	8	9	0.000000e+00	0	0.000000e+00

394	8	9	0.000000e+00	0	0.000000e+00
395	8	9	0.000000e+00	0	0.000000e+00
396	8	9	0.000000e+00	0	0.000000e+00
397	8	10	0.000000e+00	0	0.000000e+00
398	8	10	0.000000e+00	0	0.000000e+00
399	8	10	0.000000e+00	0	0.000000e+00
400	8	10	0.000000e+00	0	0.000000e+00
401	8	10	0.000000e+00	0	0.000000e+00
402	9	1	5.715102e+01	160	1.944722e+01
403	9	1	9.349635e+01	216	3.558056e+01
404	9	1	7.131250e+01	205	2.694028e+01
405	9	1	6.037903e+01	164	2.079722e+01
406	9	1	8.268421e+01	232	3.272917e+01
407	9	2	9.716216e+00	0	2.995833e+00
408	9	2	1.872378e+01	62	7.437500e+00
409	9	2	1.517910e+01	49	5.650000e+00
410	9	2	1.347350e+01	33	5.295833e+00
411	9	2	1.716245e+01	50	6.602778e+00
412	9	3	2.753623e+00	0	1.055556e+00
413	9	3	2.710623e+00	0	1.027778e+00
414	9	3	4.526531e+00	0	1.540278e+00
415	9	3	1.476190e+00	0	5.597222e-01
416	9	3	2.600000e+00	0	9.930556e-01
417	9	4	1.059322e-01	0	3.472222e-02
418	9	4	1.971326e-01	0	7.638889e-02
419	9	4	8.520900e-01	0	3.680556e-01
420	9	4	3.802817e-01	0	1.500000e-01
421	9	4	5.686901e-01	0	2.472222e-01
422	9	5	0.000000e+00	0	0.000000e+00
423	9	5	0.000000e+00	0	0.000000e+00
424	9	5	0.000000e+00	0	0.000000e+00
425	9	5	2.872727e-01	0	1.097222e-01
426	9	5	0.000000e+00	0	0.000000e+00
427	9	6	0.000000e+00	0	0.000000e+00
428	9	6	0.000000e+00	0	0.000000e+00
429	9	6	0.000000e+00	0	0.000000e+00
430	9	6	0.000000e+00	0	0.000000e+00
431	9	6	0.000000e+00	0	0.000000e+00
432	9	7	0.000000e+00	0	0.000000e+00
433	9	7	0.000000e+00	0	0.000000e+00
434	9	7	0.000000e+00	0	0.000000e+00
435	9	7	0.000000e+00	0	0.000000e+00
436	9	7	0.000000e+00	0	0.000000e+00

437	9	8	0.000000e+00	0	0.000000e+00
438	9	8	0.000000e+00	0	0.000000e+00
439	9	8	0.000000e+00	0	0.000000e+00
440	9	8	0.000000e+00	0	0.000000e+00
441	9	8	0.000000e+00	0	0.000000e+00
442	9	9	0.000000e+00	0	0.000000e+00
443	9	9	0.000000e+00	0	0.000000e+00
444	9	9	0.000000e+00	0	0.000000e+00
445	9	9	0.000000e+00	0	0.000000e+00
446	9	9	0.000000e+00	0	0.000000e+00
447	9	10	0.000000e+00	0	0.000000e+00
448	9	10	0.000000e+00	0	0.000000e+00
449	9	10	0.000000e+00	0	0.000000e+00
450	9	10	0.000000e+00	0	0.000000e+00
451	9	10	0.000000e+00	0	0.000000e+00
452	10	1	5.401172e+01	175	1.920417e+01
453	10	1	5.596774e+01	198	2.168750e+01
454	10	1	4.959615e+01	162	1.790972e+01
455	10	1	4.811348e+01	179	1.884444e+01
456	10	1	5.757971e+01	189	2.207222e+01
457	10	2	1.088475e+01	0	4.459722e+00
458	10	2	1.117489e+01	10	3.461111e+00
459	10	2	1.140840e+01	0	4.151389e+00
460	10	2	1.063158e+01	35	4.208333e+00
461	10	2	9.200743e+00	4	3.437500e+00
462	10	3	4.544484e+00	0	1.773611e+00
463	10	3	1.108209e+00	0	4.125000e-01
464	10	3	1.939597e+00	0	8.027778e-01
465	10	3	2.760618e+00	0	9.930556e-01
466	10	3	2.334615e+00	0	8.430556e-01
467	10	4	0.000000e+00	0	0.000000e+00
468	10	4	1.808118e-01	0	6.805556e-02
469	10	4	0.000000e+00	0	0.000000e+00
470	10	4	8.923077e-02	0	4.027778e-02
471	10	4	1.646091e-01	0	5.555556e-02
472	10	5	0.000000e+00	0	0.000000e+00
473	10	5	1.634981e-01	0	5.972222e-02
474	10	5	0.000000e+00	0	0.000000e+00
475	10	5	1.098039e-01	0	3.888889e-02
476	10	5	0.000000e+00	0	0.000000e+00
477	10	6	0.000000e+00	0	0.000000e+00
478	10	6	0.000000e+00	0	0.000000e+00
479	10	6	0.000000e+00	0	0.000000e+00

```
480        10          6     0.000000e+00          0     0.000000e+00
481        10          6     0.000000e+00          0     0.000000e+00
482        10          7     0.000000e+00          0     0.000000e+00
483        10          7     0.000000e+00          0     0.000000e+00
484        10          7     0.000000e+00          0     0.000000e+00
485        10          7     0.000000e+00          0     0.000000e+00
486        10          7     0.000000e+00          0     0.000000e+00
487        10          8     0.000000e+00          0     0.000000e+00
488        10          8     0.000000e+00          0     0.000000e+00
489        10          8     0.000000e+00          0     0.000000e+00
490        10          8     0.000000e+00          0     0.000000e+00
491        10          8     0.000000e+00          0     0.000000e+00
492        10          9     0.000000e+00          0     0.000000e+00
493        10          9     0.000000e+00          0     0.000000e+00
494        10          9     0.000000e+00          0     0.000000e+00
495        10          9     0.000000e+00          0     0.000000e+00
496        10          9     0.000000e+00          0     0.000000e+00
497        10         10     0.000000e+00          0     0.000000e+00
498        10         10     0.000000e+00          0     0.000000e+00
499        10         10     0.000000e+00          0     0.000000e+00
500        10         10     0.000000e+00          0     0.000000e+00
501        10         10     0.000000e+00          0     0.000000e+00
    max_queue_length avg_tables_occupied
2                229           0.9861111
3                225           0.9805556
4                258           0.9152778
5                264           0.9833333
6                249           0.9972222
7                225           1.9305556
8                200           1.8111111
9                201           1.9069444
10               171           1.7472222
11               221           1.7597222
12               184           2.8527778
13               170           2.7138889
14               193           2.6666667
15               207           2.6111111
16               131           2.8166667
17               155           3.4513889
18               143           3.4736111
19               174           3.6347222
20               152           3.6416667
21               154           3.5277778
```

```
22                   84           4.1555556
23                  130           4.3430556
24                  116           4.3500000
25                  144           4.3555556
26                  161           4.4152778
27                  100           5.4319444
28                   80           5.1222222
29                   75           5.0763889
30                   78           5.2069444
31                   72           4.9930556
32                  101           5.9805556
33                   97           5.8888889
34                   65           5.6916667
35                   89           5.8833333
36                   83           6.3138889
37                   62           6.3833333
38                   59           6.5666667
39                   76           6.7555556
40                   87           6.8000000
41                   67           6.6972222
42                   61           6.7333333
43                   67           6.4611111
44                   66           7.5097222
45                   63           7.1611111
46                   61           7.4277778
47                   67           6.9916667
48                   37           6.6319444
49                   68           7.4166667
50                   40           7.1250000
51                   52           7.0000000
52                  206           0.9944444
53                  215           0.9444444
54                  153           0.8652778
55                  209           0.9652778
56                  185           0.9902778
57                  169           1.7750000
58                  144           1.7791667
59                  175           1.7111111
60                  179           1.8250000
61                  134           1.8222222
62                  121           2.5708333
63                   91           2.6361111
64                  119           2.4569444
```

65	95	2.5791667
66	114	2.6222222
67	74	3.2750000
68	78	3.3611111
69	70	3.4069444
70	64	3.3625000
71	69	3.3138889
72	60	3.6555556
73	68	3.6680556
74	67	3.6819444
75	52	3.4333333
76	58	3.2347222
77	47	3.2430556
78	53	4.2291667
79	35	3.6402778
80	60	4.2625000
81	32	3.3055556
82	45	4.0875000
83	24	3.1805556
84	32	3.9138889
85	42	3.4958333
86	32	3.2069444
87	23	3.3708333
88	27	3.9375000
89	41	3.4638889
90	34	3.7736111
91	38	3.6763889
92	24	3.8055556
93	21	3.8583333
94	26	3.7000000
95	32	3.6027778
96	23	3.7458333
97	13	3.6388889
98	10	3.7430556
99	9	3.7430556
100	19	3.7458333
101	12	4.0152778
102	191	0.9125000
103	181	0.9333333
104	187	0.9361111
105	126	0.8513889
106	148	0.9097222
107	119	1.7444444

108	100	1.7652778
109	105	1.6958333
110	127	1.7958333
111	116	1.7152778
112	83	2.2555556
113	62	2.1694444
114	72	2.0541667
115	80	2.5625000
116	59	1.9472222
117	44	2.4208333
118	46	2.3458333
119	51	2.4847222
120	62	2.6125000
121	49	2.4416667
122	38	2.5958333
123	34	2.5555556
124	29	2.5736111
125	33	2.6583333
126	42	2.6625000
127	22	2.5444444
128	25	3.0402778
129	18	2.7819444
130	34	2.4777778
131	29	2.4388889
132	8	2.4555556
133	9	2.5361111
134	6	2.3263889
135	13	2.3152778
136	11	2.6777778
137	2	2.2388889
138	5	2.4944444
139	3	2.4430556
140	5	2.4486111
141	3	2.3791667
142	7	2.6805556
143	1	2.4708333
144	7	2.6333333
145	2	2.7000000
146	4	2.6861111
147	0	2.1986111
148	5	2.5416667
149	6	2.7361111
150	0	1.9944444

151	1	2.4305556
152	130	0.9041667
153	171	0.9055556
154	161	0.9236111
155	151	0.9500000
156	200	0.9097222
157	57	1.5847222
158	95	1.6861111
159	77	1.5236111
160	75	1.5027778
161	76	1.6861111
162	38	1.8361111
163	49	2.0319444
164	50	2.0000000
165	44	1.9638889
166	40	1.8930556
167	33	2.1861111
168	41	2.1138889
169	42	2.3555556
170	29	1.6930556
171	37	2.0583333
172	13	1.7319444
173	11	1.9652778
174	16	1.9472222
175	29	1.9527778
176	15	1.7611111
177	8	2.0875000
178	10	1.8583333
179	11	2.1152778
180	8	1.7597222
181	9	1.9430556
182	3	1.9083333
183	3	2.0861111
184	6	2.0736111
185	1	1.8194444
186	1	1.6875000
187	3	2.0527778
188	0	1.6513889
189	6	2.2402778
190	0	1.9194444
191	0	1.7638889
192	0	1.8638889
193	0	2.0166667

194	3	1.9472222
195	0	1.8333333
196	1	1.9041667
197	0	1.8375000
198	0	1.8236111
199	0	1.9611111
200	0	1.9236111
201	0	1.8152778
202	75	0.9736111
203	153	0.8875000
204	140	0.8750000
205	123	0.9041667
206	177	0.9416667
207	66	1.5611111
208	80	1.4319444
209	62	1.5902778
210	71	1.5347222
211	47	1.2444444
212	34	1.3472222
213	46	1.6263889
214	34	1.6111111
215	46	1.6736111
216	30	1.6013889
217	17	1.6402778
218	19	1.4819444
219	17	1.5777778
220	16	1.5055556
221	8	1.2972222
222	10	1.7055556
223	6	1.6708333
224	8	1.4652778
225	13	1.7194444
226	9	1.5861111
227	4	1.7069444
228	6	1.5138889
229	3	1.5527778
230	3	1.4291667
231	3	1.7611111
232	1	1.6805556
233	0	1.4013889
234	0	1.4208333
235	1	1.4180556
236	2	1.4763889

237	0	1.6097222
238	0	1.2513889
239	0	1.5611111
240	0	1.5083333
241	0	1.6416667
242	0	1.9458333
243	0	1.5666667
244	0	1.8083333
245	0	1.5763889
246	0	1.4500000
247	0	1.4513889
248	0	1.6152778
249	0	1.6722222
250	0	1.3500000
251	0	1.5916667
252	75	0.8361111
253	143	0.8819444
254	104	0.9388889
255	95	0.8763889
256	87	0.8125000
257	59	1.4055556
258	62	1.3736111
259	60	1.2986111
260	68	1.2833333
261	48	1.2944444
262	23	1.2638889
263	30	1.3555556
264	34	1.4291667
265	29	0.9972222
266	31	1.1736111
267	12	1.3138889
268	14	1.3750000
269	12	1.3416667
270	20	1.4027778
271	10	1.2916667
272	7	1.5055556
273	4	1.3736111
274	4	1.4208333
275	6	1.3361111
276	5	1.4888889
277	2	1.4750000
278	4	1.5236111
279	0	1.2916667

280	1	1.3250000
281	0	1.1652778
282	0	1.1916667
283	0	1.2361111
284	0	1.3472222
285	0	1.1569444
286	0	1.1666667
287	0	1.4861111
288	1	1.2444444
289	0	1.1097222
290	0	1.3930556
291	0	1.1055556
292	0	1.3416667
293	0	1.2958333
294	0	1.2611111
295	0	1.1611111
296	0	1.3916667
297	0	1.1777778
298	0	1.3486111
299	0	1.1444444
300	0	1.1263889
301	0	1.3097222
302	110	0.8861111
303	97	0.8083333
304	95	0.8763889
305	51	0.8375000
306	86	0.8750000
307	47	1.0736111
308	33	0.9805556
309	45	1.1097222
310	45	1.1319444
311	42	1.2402778
312	25	1.1402778
313	29	1.1236111
314	16	1.0125000
315	23	1.2680556
316	13	1.0236111
317	6	1.1152778
318	4	1.1166667
319	6	1.1708333
320	12	1.3527778
321	4	1.1486111
322	3	1.1041667

323	4	1.0597222
324	2	1.1375000
325	2	1.0763889
326	6	1.2763889
327	1	1.1666667
328	0	1.1041667
329	0	1.0583333
330	0	1.1097222
331	0	1.2277778
332	0	1.1250000
333	0	1.1597222
334	1	1.3166667
335	0	1.0472222
336	0	1.1902778
337	0	0.9972222
338	0	0.9361111
339	0	1.1638889
340	0	1.0388889
341	1	1.2847222
342	0	1.1166667
343	0	1.0180556
344	0	1.2388889
345	0	1.0722222
346	0	1.0416667
347	0	1.3930556
348	0	1.2416667
349	0	1.1916667
350	0	1.3472222
351	0	1.0763889
352	64	0.7694444
353	58	0.7902778
354	80	0.8527778
355	55	0.7486111
356	112	0.8694444
357	45	1.0430556
358	26	0.8833333
359	38	1.1458333
360	35	0.8722222
361	42	0.9916667
362	12	1.0333333
363	15	1.0277778
364	12	1.0819444
365	19	1.0027778

366	24	1.2166667
367	5	1.0458333
368	1	0.9597222
369	7	1.1027778
370	6	0.9527778
371	3	1.1250000
372	3	1.1208333
373	1	1.0166667
374	0	0.8763889
375	2	1.0305556
376	1	1.0458333
377	0	0.8166667
378	0	1.0263889
379	0	0.9833333
380	0	0.9791667
381	0	1.2375000
382	0	0.8111111
383	0	1.1777778
384	0	1.1638889
385	0	1.0347222
386	0	0.9083333
387	0	1.1430556
388	0	0.8125000
389	0	0.9194444
390	0	1.0763889
391	0	1.0250000
392	0	1.0472222
393	0	1.0666667
394	0	0.9625000
395	0	0.8597222
396	0	1.0555556
397	0	1.0819444
398	0	0.9347222
399	0	1.0458333
400	0	1.1527778
401	0	1.0000000
402	62	0.7500000
403	79	0.8680556
404	92	0.7583333
405	74	0.7027778
406	71	0.8652778
407	20	0.6930556
408	31	0.9625000

409	34	0.9555556
410	32	0.8861111
411	36	0.9763889
412	7	0.9500000
413	9	0.8486111
414	11	0.8541667
415	8	0.8458333
416	12	0.8611111
417	1	0.7555556
418	2	0.9402778
419	2	1.0750000
420	1	0.9000000
421	2	1.1152778
422	0	0.8166667
423	0	0.9916667
424	0	1.0111111
425	1	1.0041667
426	0	0.8847222
427	0	1.0333333
428	0	0.9986111
429	0	0.8819444
430	0	1.0444444
431	0	0.7805556
432	0	0.8000000
433	0	0.7791667
434	0	0.8861111
435	0	1.0666667
436	0	0.9069444
437	0	0.9208333
438	0	0.9500000
439	0	0.8833333
440	0	1.0097222
441	0	0.9541667
442	0	0.7944444
443	0	0.8291667
444	0	0.9708333
445	0	1.0013889
446	0	1.0347222
447	0	1.0972222
448	0	0.6694444
449	0	0.9194444
450	0	0.8527778
451	0	0.9194444

452	66	0.7402778
453	60	0.7888889
454	59	0.7055556
455	63	0.7375000
456	59	0.7986111
457	25	0.8125000
458	26	0.7402778
459	24	0.7791667
460	32	0.8430556
461	24	0.7777778
462	11	0.8527778
463	7	0.7708333
464	7	0.9277778
465	12	0.8055556
466	9	0.7986111
467	0	0.8611111
468	2	0.7722222
469	0	0.8708333
470	2	1.0041667
471	1	0.7444444
472	0	0.9777778
473	1	0.8194444
474	0	0.7986111
475	1	0.8041667
476	0	0.8597222
477	0	0.8652778
478	0	0.7375000
479	0	0.6472222
480	0	0.6375000
481	0	0.9541667
482	0	0.8680556
483	0	0.7708333
484	0	0.8486111
485	0	0.9458333
486	0	0.8555556
487	0	0.8250000
488	0	0.7569444
489	0	0.8833333
490	0	0.8805556
491	0	0.6777778
492	0	0.7305556
493	0	0.7722222
494	0	0.8597222

```
495                     0            0.8180556
496                     0            0.9291667
497                     0            0.7166667
498                     0            0.7361111
499                     0            0.9208333
500                     0            0.7791667
501                     0            0.6819444
```

```
# MEAN WAITING TIMES
df |>
  group_by(num_chefs, num_tables) |>
  summarise(mean = mean(avg_waiting_time), variance = var(avg_waiting_time)) |>
  kable()
```

```
`summarise()` has grouped output by 'num_chefs'. You can override using the
`.groups` argument.
```

num_chefs	num_tables	mean	variance
1	1	302.4420799	263.2093651
1	2	263.3521486	708.6061443
1	3	233.1169626	156.4117643
1	4	184.0850579	319.2414708
1	5	152.7306214	938.2409450
1	6	108.1094104	121.8450076
1	7	94.8823777	518.4590724
1	8	75.5414944	130.1534533
1	9	60.3173267	135.6901334
1	10	42.8577452	171.6686617
2	1	267.1659507	210.9673442
2	2	205.5063456	422.4201925
2	3	122.6029371	172.0832860
2	4	78.9959118	57.1862031
2	5	49.7206140	65.3613180
2	6	27.6415103	105.4710341
2	7	17.6497059	79.4904904
2	8	14.3678343	6.7035180
2	9	8.2476473	1.8947676
2	10	3.3306711	0.7439870
3	1	216.6680349	875.9878896
3	2	137.0845135	349.8536407

num_chefs	num_tables	mean	variance
3	3	63.7284759	157.4962333
3	4	36.7864950	31.7258607
3	5	18.7439330	10.6602543
3	6	10.3145972	4.1018820
3	7	2.7830589	1.6256502
3	8	0.8742371	0.4792825
3	9	0.7546323	0.3597019
3	10	0.6745144	0.7703845
4	1	203.4356721	553.6868484
4	2	81.8491177	392.7843672
4	3	34.7387079	23.8355617
4	4	19.3277437	19.0173258
4	5	6.2395822	4.3989920
4	6	2.2906946	0.3129572
4	7	0.6692881	0.5974925
4	8	0.3602324	0.4897333
4	9	0.1883606	0.1113027
4	10	0.0000000	0.0000000
5	1	167.9506607	1368.9233277
5	2	61.5701462	116.0291172
5	3	21.6889214	34.0577406
5	4	5.9410495	6.2922534
5	5	2.9424780	2.2448938
5	6	1.0096387	0.6038181
5	7	0.1429671	0.0476156
5	8	0.0000000	0.0000000
5	9	0.0000000	0.0000000
5	10	0.0000000	0.0000000
6	1	112.4166281	1367.9979171
6	2	42.8650146	31.9486271
6	3	13.7358239	16.2801950
6	4	4.3493430	4.5974934
6	5	1.2751990	0.2369256
6	6	0.2857463	0.1145544
6	7	0.0000000	0.0000000
6	8	0.0289157	0.0041806
6	9	0.0000000	0.0000000
6	10	0.0000000	0.0000000
7	1	100.0590050	807.5094731
7	2	25.9083827	13.8670232
7	3	8.2412182	3.2191952

num_chefs	num_tables	mean	variance
7	4	2.0641578	2.0467181
7	5	0.7523194	0.2968596
7	6	0.0098113	0.0004813
7	7	0.0464516	0.0107888
7	8	0.0213333	0.0022756
7	9	0.0000000	0.0000000
7	10	0.0000000	0.0000000
8	1	75.2183121	370.5340849
8	2	20.0144596	19.4140775
8	3	6.3874142	10.9183419
8	4	1.0094722	0.4416167
8	5	0.3780817	0.1880803
8	6	0.0000000	0.0000000
8	7	0.0000000	0.0000000
8	8	0.0000000	0.0000000
8	9	0.0000000	0.0000000
8	10	0.0000000	0.0000000
9	1	73.0046227	231.8027105
9	2	14.8510100	12.1780969
9	3	2.8133934	1.1956555
9	4	0.4208253	0.0896733
9	5	0.0574545	0.0165051
9	6	0.0000000	0.0000000
9	7	0.0000000	0.0000000
9	8	0.0000000	0.0000000
9	9	0.0000000	0.0000000
9	10	0.0000000	0.0000000
10	1	53.0537600	16.5636637
10	2	10.6600706	0.7514889
10	3	2.5375047	1.6298222
10	4	0.0869303	0.0074917
10	5	0.0546604	0.0059624
10	6	0.0000000	0.0000000
10	7	0.0000000	0.0000000
10	8	0.0000000	0.0000000
10	9	0.0000000	0.0000000
10	10	0.0000000	0.0000000

```
# MEAN LONGEST WAITING TIME OF THE DAY
df |>
```

```r
  group_by(num_chefs, num_tables) |>
  summarise(mean = mean(long_waits), variance = var(long_waits)) |>
  kable()
```

```
`summarise()` has grouped output by 'num_chefs'. You can override using the
`.groups` argument.
```

num_chefs	num_tables	mean	variance
1	1	271.4	431.8
1	2	250.0	245.5
1	3	250.4	785.3
1	4	256.2	233.2
1	5	240.4	657.8
1	6	219.6	97.8
1	7	223.0	1231.5
1	8	207.6	793.3
1	9	171.4	602.3
1	10	157.2	2175.7
2	1	244.0	876.5
2	2	250.0	315.5
2	3	245.2	57.7
2	4	215.8	96.7
2	5	172.6	406.8
2	6	105.8	2790.7
2	7	65.2	2161.2
2	8	35.0	805.5
2	9	13.0	225.0
2	10	0.0	0.0
3	1	239.2	476.2
3	2	245.2	226.2
3	3	189.4	1259.8
3	4	143.6	555.8
3	5	85.0	1128.0
3	6	12.8	199.7
3	7	0.0	0.0
3	8	0.0	0.0
3	9	0.0	0.0
3	10	0.0	0.0
4	1	256.0	810.5
4	2	206.6	1790.8

num_chefs	num_tables	mean	variance
4	3	144.2	592.7
4	4	70.2	1765.2
4	5	0.4	0.8
4	6	0.0	0.0
4	7	0.0	0.0
4	8	0.0	0.0
4	9	0.0	0.0
4	10	0.0	0.0
5	1	244.8	1054.7
5	2	186.2	1050.7
5	3	88.2	1310.2
5	4	0.0	0.0
5	5	0.0	0.0
5	6	0.0	0.0
5	7	0.0	0.0
5	8	0.0	0.0
5	9	0.0	0.0
5	10	0.0	0.0
6	1	232.8	1419.7
6	2	161.6	347.8
6	3	36.4	709.3
6	4	0.0	0.0
6	5	0.0	0.0
6	6	0.0	0.0
6	7	0.0	0.0
6	8	0.0	0.0
6	9	0.0	0.0
6	10	0.0	0.0
7	1	221.8	1536.2
7	2	104.6	519.8
7	3	5.8	168.2
7	4	0.0	0.0
7	5	0.0	0.0
7	6	0.0	0.0
7	7	0.0	0.0
7	8	0.0	0.0
7	9	0.0	0.0
7	10	0.0	0.0
8	1	201.4	3373.3
8	2	70.6	1351.3
8	3	1.8	16.2

num_chefs	num_tables	mean	variance
8	4	0.0	0.0
8	5	0.0	0.0
8	6	0.0	0.0
8	7	0.0	0.0
8	8	0.0	0.0
8	9	0.0	0.0
8	10	0.0	0.0
9	1	195.4	1023.8
9	2	38.8	576.7
9	3	0.0	0.0
9	4	0.0	0.0
9	5	0.0	0.0
9	6	0.0	0.0
9	7	0.0	0.0
9	8	0.0	0.0
9	9	0.0	0.0
9	10	0.0	0.0
10	1	180.6	188.3
10	2	9.8	215.2
10	3	0.0	0.0
10	4	0.0	0.0
10	5	0.0	0.0
10	6	0.0	0.0
10	7	0.0	0.0
10	8	0.0	0.0
10	9	0.0	0.0
10	10	0.0	0.0

```r
# MEAN QUEUE LENGTH
df |>
  group_by(num_chefs, num_tables) |>
  summarise(mean = mean(avg_queue_length), variance = var(avg_queue_length)) |>
  kable()
```

```
`summarise()` has grouped output by 'num_chefs'. You can override using the
`.groups` argument.
```

num_chefs	num_tables	mean	variance
1	1	117.9519444	222.2075758
1	2	97.1513889	189.8501225
1	3	87.8111111	130.8596807
1	4	72.4116667	50.9318920
1	5	57.6322222	202.1952994
1	6	38.6869444	18.9765910
1	7	36.2141667	86.8600081
1	8	28.4944444	35.9340268
1	9	21.7183333	19.6079277
1	10	16.6563889	37.9199022
2	1	95.3147222	186.8602726
2	2	78.6405556	97.5175727
2	3	47.1208333	36.7312548
2	4	30.6997222	8.3465108
2	5	18.8366667	15.0397296
2	6	10.6147222	26.5720743
2	7	6.6269444	11.6320145
2	8	5.1836111	0.8405222
2	9	3.1163889	0.1674560
2	10	1.2458333	0.1049749
3	1	79.0697222	207.1671368
3	2	53.1163889	74.5446136
3	3	23.8533333	34.2192405
3	4	13.2680556	6.4864902
3	5	7.5030556	2.2751190
3	6	4.1013889	1.4187336
3	7	1.1077778	0.2912197
3	8	0.3363889	0.0704460
3	9	0.3080556	0.0610594
3	10	0.2477778	0.1031584
4	1	78.7983333	157.3964153
4	2	30.7311111	108.4275237
4	3	13.0247222	4.7419284
4	4	7.9477778	4.0460287
4	5	2.2969444	0.4659697
4	6	0.8780556	0.0770258
4	7	0.2522222	0.0817338
4	8	0.1519444	0.0893360
4	9	0.0658333	0.0131960
4	10	0.0000000	0.0000000
5	1	65.1958333	400.1123756

num_chefs	num_tables	mean	variance
5	2	23.1113889	35.8861848
5	3	8.2472222	5.1041811
5	4	2.2738889	1.0775824
5	5	1.1533333	0.3561889
5	6	0.3738889	0.0833038
5	7	0.0525000	0.0060160
5	8	0.0000000	0.0000000
5	9	0.0000000	0.0000000
5	10	0.0000000	0.0000000
6	1	43.0852778	325.0331304
6	2	16.7950000	4.8314705
6	3	5.1844444	3.4409147
6	4	1.6408333	0.6632085
6	5	0.5063889	0.0483493
6	6	0.1127778	0.0175245
6	7	0.0000000	0.0000000
6	8	0.0100000	0.0005000
6	9	0.0000000	0.0000000
6	10	0.0000000	0.0000000
7	1	37.3188889	137.0961426
7	2	9.6488889	2.4073113
7	3	3.1972222	0.8145033
7	4	0.8461111	0.4430096
7	5	0.2830556	0.0445239
7	6	0.0036111	0.0000652
7	7	0.0200000	0.0020000
7	8	0.0088889	0.0003951
7	9	0.0000000	0.0000000
7	10	0.0000000	0.0000000
8	1	28.4144444	148.3633542
8	2	7.1583333	3.2948775
8	3	2.4438889	1.7499803
8	4	0.3980556	0.0720289
8	5	0.1472222	0.0291869
8	6	0.0000000	0.0000000
8	7	0.0000000	0.0000000
8	8	0.0000000	0.0000000
8	9	0.0000000	0.0000000
8	10	0.0000000	0.0000000
9	1	27.0988889	50.4807151
9	2	5.5963889	2.8146514

num_chefs	num_tables	mean	variance
9	3	1.0352778	0.1208571
9	4	0.1752778	0.0181283
9	5	0.0219444	0.0024078
9	6	0.0000000	0.0000000
9	7	0.0000000	0.0000000
9	8	0.0000000	0.0000000
9	9	0.0000000	0.0000000
9	10	0.0000000	0.0000000
10	1	19.9436111	3.3659458
10	2	3.9436111	0.2171437
10	3	0.9650000	0.2502704
10	4	0.0327778	0.0009921
10	5	0.0197222	0.0007836
10	6	0.0000000	0.0000000
10	7	0.0000000	0.0000000
10	8	0.0000000	0.0000000
10	9	0.0000000	0.0000000
10	10	0.0000000	0.0000000

```r
# MEAN MAX QUEUE LENGTH FOR EACH DAY
df |>
  group_by(num_chefs, num_tables) |>
  summarise(mean = mean(max_queue_length), variance = var(max_queue_length)) |>
  kable()
```

```
`summarise()` has grouped output by 'num_chefs'. You can override using the
`.groups` argument.
```

num_chefs	num_tables	mean	variance
1	1	245.0	300.5
1	2	203.6	460.8
1	3	177.0	842.5
1	4	155.6	128.3
1	5	127.0	856.0
1	6	81.0	122.0
1	7	87.0	200.0
1	8	70.2	129.7
1	9	63.6	7.8

num_chefs	num_tables	mean	variance
1	10	52.8	211.7
2	1	193.6	642.8
2	2	160.2	399.7
2	3	108.0	196.0
2	4	71.0	28.0
2	5	61.0	44.0
2	6	45.4	140.3
2	7	35.0	72.0
2	8	32.6	56.3
2	9	25.2	17.7
2	10	12.6	15.3
3	1	166.6	803.3
3	2	113.4	118.3
3	3	71.2	112.7
3	4	50.4	49.3
3	5	35.2	24.7
3	6	25.6	38.3
3	7	9.4	7.3
3	8	3.6	1.8
3	9	4.2	7.7
3	10	2.4	8.3
4	1	162.6	667.3
4	2	76.0	181.0
4	3	44.2	28.2
4	4	36.4	29.8
4	5	16.8	50.2
4	6	9.2	1.7
4	7	2.8	4.2
4	8	1.8	7.2
4	9	0.8	1.7
4	10	0.0	0.0
5	1	133.6	1461.8
5	2	65.2	148.7
5	3	38.0	56.0
5	4	15.4	18.3
5	5	9.2	6.7
5	6	3.8	1.7
5	7	0.8	0.7
5	8	0.0	0.0
5	9	0.0	0.0
5	10	0.0	0.0

num_chefs	num_tables	mean	variance
6	1	100.8	670.2
6	2	59.4	52.8
6	3	29.4	16.3
6	4	13.6	14.8
6	5	5.2	1.7
6	6	1.4	2.8
6	7	0.0	0.0
6	8	0.2	0.2
6	9	0.0	0.0
6	10	0.0	0.0
7	1	87.8	496.7
7	2	42.4	30.8
7	3	21.2	43.2
7	4	6.4	10.8
7	5	3.4	2.8
7	6	0.2	0.2
7	7	0.2	0.2
7	8	0.2	0.2
7	9	0.0	0.0
7	10	0.0	0.0
8	1	73.8	549.2
8	2	37.2	53.7
8	3	16.4	26.3
8	4	4.4	5.8
8	5	1.4	1.3
8	6	0.0	0.0
8	7	0.0	0.0
8	8	0.0	0.0
8	9	0.0	0.0
8	10	0.0	0.0
9	1	75.6	122.3
9	2	30.6	38.8
9	3	9.4	4.3
9	4	1.6	0.3
9	5	0.2	0.2
9	6	0.0	0.0
9	7	0.0	0.0
9	8	0.0	0.0
9	9	0.0	0.0
9	10	0.0	0.0
10	1	61.4	9.3

num_chefs	num_tables	mean	variance
10	2	26.2	11.2
10	3	9.2	5.2
10	4	1.0	1.0
10	5	0.4	0.3
10	6	0.0	0.0
10	7	0.0	0.0
10	8	0.0	0.0
10	9	0.0	0.0
10	10	0.0	0.0

```r
# MEAN TABLES OCCUPIED
df |>
  group_by(num_chefs, num_tables) |>
  summarise(mean = mean(avg_tables_occupied), variance = var(avg_tables_occupied)) |>
  kable()
```

```
`summarise()` has grouped output by 'num_chefs'. You can override using the
`.groups` argument.
```

num_chefs	num_tables	mean	variance
1	1	0.9725000	0.0010633
1	2	1.8311111	0.0070434
1	3	2.7322222	0.0102415
1	4	3.5458333	0.0078868
1	5	4.3238889	0.0096850
1	6	5.1661111	0.0280648
1	7	5.9516667	0.0520625
1	8	6.6405556	0.0283704
1	9	7.0586111	0.2032758
1	10	7.0330556	0.0798268
2	1	0.9519444	0.0027552
2	2	1.7825000	0.0021370
2	3	2.5730556	0.0049794
2	4	3.3438889	0.0025662
2	5	3.5347222	0.0385831
2	6	3.7361111	0.2394637
2	7	3.5769444	0.1687004
2	8	3.6444444	0.0527623

num_chefs	num_tables	mean	variance
2	9	3.7425000	0.0096833
2	10	3.7772222	0.0197816
3	1	0.9086111	0.0011645
3	2	1.7433333	0.0015706
3	3	2.1977778	0.0551414
3	4	2.4611111	0.0096914
3	5	2.6091667	0.0023945
3	6	2.6566667	0.0637186
3	7	2.4622222	0.0230029
3	8	2.4008333	0.0098810
3	9	2.6341667	0.0089657
3	10	2.3802778	0.0842654
4	1	0.9186111	0.0003671
4	2	1.5966667	0.0075739
4	3	1.9450000	0.0063740
4	4	2.0813889	0.0596312
4	5	1.8716667	0.0131989
4	6	1.9527778	0.0227103
4	7	1.9150000	0.0288420
4	8	1.9255556	0.0541441
4	9	1.9130556	0.0051887
4	10	1.8722222	0.0043383
5	1	0.9163889	0.0016526
5	2	1.4725000	0.0198123
5	3	1.5719444	0.0165503
5	4	1.5005556	0.0168003
5	5	1.6294444	0.0111086
5	6	1.5927778	0.0189905
5	7	1.4794444	0.0134379
5	8	1.5144444	0.0241692
5	9	1.6694444	0.0407649
5	10	1.5361111	0.0174238
6	1	0.8691667	0.0023451
6	2	1.3311111	0.0030079
6	3	1.2438889	0.0282452
6	4	1.3450000	0.0020154
6	5	1.4250000	0.0052826
6	6	1.3561111	0.0209323
6	7	1.2197222	0.0060170
6	8	1.2677778	0.0288015
6	9	1.2902778	0.0076215

num_chefs	num_tables	mean	variance
6	10	1.2213889	0.0102089
7	1	0.8566667	0.0010739
7	2	1.1072222	0.0088738
7	3	1.1136111	0.0107469
7	4	1.1808333	0.0097795
7	5	1.1308333	0.0074907
7	6	1.1333333	0.0042660
7	7	1.1677778	0.0097757
7	8	1.0841667	0.0195270
7	9	1.0975000	0.0076065
7	10	1.2500000	0.0158825
8	1	0.8061111	0.0027726
8	2	0.9872222	0.0130781
8	3	1.0725000	0.0073171
8	4	1.0372222	0.0063034
8	5	1.0180556	0.0078906
8	6	1.0086111	0.0227637
8	7	1.0191667	0.0254788
8	8	0.9952778	0.0171148
8	9	0.9983333	0.0077078
8	10	1.0430556	0.0067872
9	1	0.7888889	0.0054909
9	2	0.8947222	0.0139269
9	3	0.8719444	0.0019381
9	4	0.9572222	0.0207710
9	5	0.9416667	0.0075241
9	6	0.9477778	0.0128864
9	7	0.8877778	0.0129682
9	8	0.9436111	0.0021688
9	9	0.9261111	0.0115494
9	10	0.8916667	0.0236728
10	1	0.7541667	0.0015037
10	2	0.7905556	0.0015147
10	3	0.8311111	0.0037892
10	4	0.8505556	0.0103787
10	5	0.8519444	0.0055195
10	6	0.7683333	0.0191671
10	7	0.8577778	0.0038769
10	8	0.8047222	0.0076848
10	9	0.8219444	0.0059408

10	10	0.7669444	0.0086337