

In [3]:

```
from prettytable import PrettyTable
import random

print("\n")
print("Welcome To The Best Practices Movie Schedule Generator!")
print("\n")

# input receiving class
number_of_screens = int(
    input("Enter number of screens in your multiplex mall: "))
print("\n")

class MovieClass:
    def __init__(self, name, demand_score, cast_weight, success_rate,
                 time_running, feature, language, local_language):

        # defining characteristics of movie
        self.name = name
        self.demand_score = demand_score
        self.cast_weight = cast_weight
        self.success_rate = success_rate
        self.priority = (demand_score + cast_weight + success_rate) / 2
        self.feature = feature
        self.language = language
        self.local_lang = local_lang

        # Determining priority based on success rate
        self.rate = (self.success_rate / 5) * 100
        if 20 < self.rate <= 50:
            self.priority /= 2
        if self.rate <= 20:
            self.priority = 0

        # determining priority based on time running
        self.time_running = time_running
        if 0 <= self.time_running <= 1:
            self.priority = self.priority
        elif 1 < self.time_running <= 2:
            self.priority *= 0.75
        elif 2 < self.time_running <= 3:
            self.priority *= 0.5
        elif 3 < self.time_running <= 4:
            self.priority *= 0.25
        else:
            self.priority = 0

        # determining feature based on features
        self.feature = feature
        if f == 1:
            self.priority += 0.5
        elif f == 2:
            self.priority += 0.5
        elif f == 3:
            self.priority += 0.75
        else:
            self.priority = self.priority

        if language.lower() != local_lang.lower():
```

```

        self.priority *= 0.25
    else:
        self.priority = self.priority

# dictionary of the movie name with priority

N = int(input("Enter number of movies you would like to schedule: "))
local_lang = input("Enter local language of your region: ")
Movie = {}
Details = {}
for i in range(N):
    deets = {}
    print(
        "
    )
    n = input("Enter movie name : ")
    # demand score scale of 1-10
    ds = int(input("Enter demand score (Out of 10): "))
    deets['DemSco'] = ds
    # cw scale of 1-5
    cw = int(input("Enter cast weightage (Out of 5): "))
    deets['CastW'] = cw
    # sr scale of 1-5
    sr = int(input("Enter success rate (Out of 5): "))
    deets['SRate'] = sr
    t = int(
        input(
            "Enter the number of weeks the movie has been running / Enter 0 if the movie
        ))
    deets['RunTime'] = t
    f = int(
        input(
            "Enter the number accordingly:\n1.3D\n2.Dolby\n3.Both 3d & Dolby\n4.None: "
        ))
    deets['Feature'] = f
    lang = input("Enter language of the movie: ")
    deets['Language'] = lang
    deets['Local_language'] = local_lang
    movie = MovieClass(n, ds, cw, sr, t, f, lang, local_lang)
    Movie[movie.name] = movie.priority
    Details[movie.name] = deets
print(Details)
print("\n")
print("\n")
# input for friday through sunday
N = int(input("Enter number of movies to be released on Friday : "))
new_movies = {}
new_details = {}
for i in range(N):
    deets = {}
    print(
        "
    )
    n = input("Enter movie name: ")
    # demand score scale of 1-10
    ds = int(input("Enter demand score (Out of 10): "))
    deets['DemSco'] = ds
    # cw scale of 1-5
    cw = int(input("Enter cast weightage (Out of 10): "))
    deets['CastW'] = cw

```

```

# sr scale of 1-5
sr = int(input("Enter success rate (Out of 5): "))
deets['SRate'] = sr
deets['RunTime'] = 0
f = int(
    input(
        "Enter the number accordingly:\n1.3D\n2.Dolby\n3.Both 3d & dolby\n4.None: "
    )
)
deets['Feature'] = f
lang = input("Enter language of the movie: ")
deets['Language'] = lang
deets['Local_language'] = local_lang
movie = MovieClass(n, ds, cw, sr, t, f, lang, local_lang)
new_movies[movie.name] = movie.priority
new_details[movie.name] = deets

#####
#Functions

Slots = ['Slot1', 'Slot2', 'Slot3', 'Slot4', 'Slot5', 'Slot6']
print("\n")
day = 0
days = list()
while day != -1:
    day = int(
        input(
            "Enter days of the week on which special occasions are celebrated: \n1--Mond
        )
    )
    if 1 <= day <= 7 and day not in days:
        days.append(day)
    else:
        continue

Screens = []
for i in range(number_of_screens):
    screen = 'Screen' + str(i + 1)
    Screens.append(screen)
# Screens=list()
schedule = dict()

# nscreens=int(input("Enter number of screens: "))

def initialization():
    global schedule
    global Screens
    global number_of_screens

    for i in range(number_of_screens):
        schedule[Screens[i]] = {
            'Slot1': '-',
            'Slot2': '-',
            'Slot3': '-',
            'Slot4': '-',
            'Slot5': '-',
            'Slot6': '-'
        }

# display table
def display_table():

```

```

myTable = PrettyTable(
    ['Screens', 'Slot1', 'Slot2', 'Slot3', 'Slot4', 'Slot5', 'Slot6'])
for i in schedule:
    myTable.add_row([
        i, schedule[i]['Slot1'], schedule[i]['Slot2'],
        schedule[i]['Slot3'], schedule[i]['Slot4'], schedule[i]['Slot5'],
        schedule[i]['Slot6']
    ])
print(myTable)

# sorting the dictionary of movies based of priority
def sorting():
    x = (dict(sorted(Movie.items(), key=lambda item: item[1])))
    # print("\n",x)
    return x

# inverting the dictionary x

def invert_dict(d):
    inverse = dict()
    for key in d:
        val = d[key]
        if val not in inverse:
            inverse[val] = [key]
        else:
            inverse[val].append(key)
    # print("\n", inverse)
    return inverse

# determining increasing order in which movies are to be sorted

def increasing_order(inv_x):
    movies_f = list()
    for i in inv_x:
        if len(inv_x[i]) == 1:
            movies_f.append(inv_x[i][0])
        else:
            compare = dict()
            for j in inv_x[i]:
                compare[j] = (Details[j]['DemSco'], Details[j]['CastW'],
                             Details[j]['SRate'], Details[j]['RunTime'])
            y = (dict(sorted(compare.items(), key=lambda item: item[1])))
            for i in y:
                movies_f.append(i)
    movies_f = movies_f[::-1]
    # print("\n", movies_f)
    return movies_f

# determining percentage of screens/shows or slots for each movie

def percents(n, total_percent=100):
    assert isinstance(n, int) and n > 0
    return [x / sum(range(n + 1)) * total_percent for x in range(1, n + 1)]

```

```

def determine_no_of_slots(movies_f, nslots):
    global number_of_screens
    temp = list()
    temp = percents(len(movies_f))
    temp = temp[::-1]
    # print(temp)

    p_movie = dict()
    j = 0
    for i in movies_f:
        p_movie[i] = round(temp[j] / 100 * (nslots * number_of_screens))
        j += 1

    p_movieList = list()
    for key in p_movie:
        p_movieList.append(key)

    total_no_showtimes = number_of_screens * nslots
    no_shows_alloted = sum(p_movie.values())
    excess_shows = no_shows_alloted - total_no_showtimes
    for i in range(len(p_movieList) - 1, -1, -1):
        if excess_shows > 0:
            if excess_shows > p_movie[p_movieList[i]]:
                t = p_movie[p_movieList[i]]
                p_movie[p_movieList[i]] = 0
                excess_shows -= t
            else:
                p_movie[p_movieList[i]] -= excess_shows
                excess_shows = 0

    # print(p_movie)
    return p_movie

# main scheduling problem
def scheduling(p_movie, l, u):
    global number_of_screens
    p_movieList = list()
    for key in p_movie:
        p_movieList.append(key)

    prime_slots = list()

    if l == 2 and u == 5:
        prime_slots = ['Slot4', 'Slot5']
    if l == 2 and u == 6:
        prime_slots = ['Slot5', 'Slot6']
    if l == 1 and u == 6:
        prime_slots = ['Slot1', 'Slot5', 'Slot6']

    number_of_prime_showtimes = number_of_screens * len(prime_slots)

    i = 0
    j = 0
    for key in p_movieList:
        while i < p_movie[key] and j != number_of_prime_showtimes:
            SlotName = random.choice(prime_slots)
            randomscreen = random.randint(0, number_of_screens - 1)
            ScreenName = Screens[randomscreen]
            if schedule[ScreenName][SlotName] == '-':

```

```

        schedule[ScreenName][SlotName] = key
        i += 1
        j += 1
    else:
        continue
    if i == p_movie[key]:
        del p_movie[key]
    else:
        p_movie[key] -= i
    i = 0

i = 0
for key in p_movie:
    while i < p_movie[key]:
        randomslot = random.randint(1 - 1, u - 1)
        SlotName = Slots[randomslot]
        randomscreen = random.randint(0, number_of_screens - 1)
        ScreenName = Screens[randomscreen]
        if schedule[ScreenName][SlotName] == '-':
            schedule[ScreenName][SlotName] = key
            i += 1
        else:
            continue
    i = 0

#####

# function _main_
Days = [
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
    'Sunday'
]
initialization()
for i in range(1, 5):
    print(Days[i - 1] + ":\n")
    x = sorting()
    inv_x = invert_dict(x)
    movies_f = increasing_order(inv_x)
    # weekdays
    if i not in days:
        p_movie = determine_no_of_slots(movies_f, 4)
        scheduling(p_movie, 2, 5)
        # print("\n", schedule)
        display_table()
    else:
        p_movie = determine_no_of_slots(movies_f, 6)
        scheduling(p_movie, 1, 6)
        # print("\n", schedule)
        display_table()
    initialization()

#####

initialization()
for i in range(5, 8):

    print(Days[i - 1] + ":\n")
    Details.update(new_details)
    Movie.update(new_movies)

```



```

x = sorting()
inv_x = invert_dict(x)
movies_f = increasing_order(inv_x)

# weekends
if i not in days:
    if i == 5:
        p_movie = determine_no_of_slots(movies_f, 4)
        scheduling(p_movie, 2, 5)
        # print("\n", schedule)
        display_table()
    else:
        p_movie = determine_no_of_slots(movies_f, 5)
        scheduling(p_movie, 2, 6)
        # print("\n", schedule)
        display_table()
else:
    p_movie = determine_no_of_slots(movies_f, 6)
    scheduling(p_movie, 1, 6)
    # print("\n", schedule)
    display_table()
initialization()

```

Welcome to the Best Practices Movie Schedule Generator.

Enter number of screens in your multiplex mall: 5

Enter number of movies you would like to schedule: 2

Enter local language of your region: tamil

Enter movie name : vikram

Enter demand score (Out of 10): 9

Enter cast weightage (Out of 5): 5

Enter success rate (Out of 5): 5

Enter the number of weeks the movie has been running / Enter 0 if the movie is newly released: 0

Enter the number accordingly:

1 2 3

In []: