g-cancellation-project-final-draft

October 25, 2024

[431]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Hotel Booking Cancellation Prediction"))
```

1 Hotel Booking Cancellation Prediction

[432]:
```
from IPython.display import display, Markdown

# Display the heading and explanation
text = """
# 1. Data Pre-Processing

### Dropping Irrelevant Columns
- **Company and Agent Columns**: These columns have a high proportion of␣
 ↪missing values, with `company` missing 112,593 entries and `agent` missing␣
 ↪16,340 entries. Keeping them could introduce noise and negatively impact the␣
 ↪model's performance due to the lack of complete information.

### Handling Missing Values
- **Filling Missing Values**: Missing values in the `children`, `agent`, and␣
 ↪`company` columns were replaced with 0 because these represent scenarios␣
 ↪where there were no children, no agent involved, or no associated company.␣
 ↪For instance, when there are no children, it's logical to fill these missing␣
 ↪values with 0 rather than imputing a mean, as children counts are␣
 ↪non-continuous and should remain accurate to the real number of guests.

### Removing Inconsistent Values
- **Removing Rows with Zero Guests**: Rows where `adults`, `children`, and␣
 ↪`babies` were all zero were removed from the dataset since they represent␣
 ↪incomplete or invalid bookings (i.e., reservations made without any guests).␣
 ↪Such records do not contribute to meaningful insights regarding booking␣
 ↪patterns.
"""

display(Markdown(text))
```

2 1. Data Pre-Processing

2.0.1 Dropping Irrelevant Columns

- **Company and Agent Columns**: These columns have a high proportion of missing values, with `company` missing 112,593 entries and `agent` missing 16,340 entries. Keeping them could introduce noise and negatively impact the model's performance due to the lack of complete information.

2.0.2 Handling Missing Values

- **Filling Missing Values**: Missing values in the `children`, `agent`, and `company` columns were replaced with 0 because these represent scenarios where there were no children, no agent involved, or no associated company. For instance, when there are no children, it's logical to fill these missing values with 0 rather than imputing a mean, as children counts are non-continuous and should remain accurate to the real number of guests.

2.0.3 Removing Inconsistent Values

- **Removing Rows with Zero Guests**: Rows where `adults`, `children`, and `babies` were all zero were removed from the dataset since they represent incomplete or invalid bookings (i.e., reservations made without any guests). Such records do not contribute to meaningful insights regarding booking patterns.

[433]:
```
import pandas as pd

# Load the CSV file
file_path = r"C:\Users\Hassan Shoaib\Downloads\hotel_bookings.csv"
df = pd.read_csv(file_path)

# Check the first few rows of the dataset
df.head()
```

[433]:
```
          hotel  is_canceled  lead_time  arrival_date_year arrival_date_month  \
0  Resort Hotel            0        342               2015               July
1  Resort Hotel            0        737               2015               July
2  Resort Hotel            0          7               2015               July
3  Resort Hotel            0         13               2015               July
4  Resort Hotel            0         14               2015               July

   arrival_date_week_number  arrival_date_day_of_month  \
0                        27                          1
1                        27                          1
2                        27                          1
3                        27                          1
4                        27                          1

   stays_in_weekend_nights  stays_in_week_nights  adults  …  deposit_type  \
0                        0                     0       2  …    No Deposit
```

```
1                       0                   0       2  …    No Deposit
2                       0                   1       1  …    No Deposit
3                       0                   1       1  …    No Deposit
4                       0                   2       2  …    No Deposit

   agent company days_in_waiting_list customer_type   adr  \
0    NaN     NaN                    0     Transient   0.0
1    NaN     NaN                    0     Transient   0.0
2    NaN     NaN                    0     Transient  75.0
3  304.0     NaN                    0     Transient  75.0
4  240.0     NaN                    0     Transient  98.0

   required_car_parking_spaces  total_of_special_requests  reservation_status  \
0                            0                          0           Check-Out
1                            0                          0           Check-Out
2                            0                          0           Check-Out
3                            0                          0           Check-Out
4                            0                          1           Check-Out

  reservation_status_date
0              2015-07-01
1              2015-07-01
2              2015-07-02
3              2015-07-02
4              2015-07-03

[5 rows x 32 columns]
```

[434]:
```
df.isnull().values.any()
```

[434]:
```
True
```

[435]:
```
df.isnull().sum()
```

[435]:
```
hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
arrival_date_week_number        0
arrival_date_day_of_month       0
stays_in_weekend_nights         0
stays_in_week_nights            0
adults                          0
children                        4
babies                          0
meal                            0
```

```
country                            488
market_segment                       0
distribution_channel                 0
is_repeated_guest                    0
previous_cancellations               0
previous_bookings_not_canceled       0
reserved_room_type                   0
assigned_room_type                   0
booking_changes                      0
deposit_type                         0
agent                            16340
company                         112593
days_in_waiting_list                 0
customer_type                        0
adr                                  0
required_car_parking_spaces          0
total_of_special_requests            0
reservation_status                   0
reservation_status_date              0
dtype: int64
```

[436]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# 1.1 Handling Missing Values"))
```

3 1.1 Handling Missing Values

[437]:
```
# Get information about the dataset (data types, missing values, etc.)
df.info()

# Get a statistical summary of the numerical columns
df.describe()

# Check for missing values in the dataset
missing_values = df.isnull().sum()
print(missing_values)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
```

```
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
hotel                             0
is_canceled                       0
lead_time                         0
arrival_date_year                 0
arrival_date_month                0
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          4
babies                            0
meal                              0
country                         488
market_segment                    0
distribution_channel              0
is_repeated_guest                 0
```

```
previous_cancellations                0
previous_bookings_not_canceled        0
reserved_room_type                    0
assigned_room_type                    0
booking_changes                       0
deposit_type                          0
agent                             16340
company                          112593
days_in_waiting_list                  0
customer_type                         0
adr                                   0
required_car_parking_spaces           0
total_of_special_requests             0
reservation_status                    0
reservation_status_date               0
dtype: int64
```

[438]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Unique Values  "))
```

4 Unique Values

[439]:
```
df['adults'].unique()
```

[439]:
```
array([ 2,  1,  3,  4, 40, 26, 50, 27, 55,  0, 20,  6,  5, 10],
      dtype=int64)
```

[440]:
```
# Get unique values in each column
for column in df.columns:
    print(f"{column}: {df[column].unique()}")
```

```
hotel: ['Resort Hotel' 'City Hotel']
is_canceled: [0 1]
lead_time: [342 737   7  13  14   0   9  85  75  23  35  68  18  37  12  72 127
78
  48  60  77  99 118  95  96  69  45  40  15  36  43  70  16 107  47 113
  90  50  93  76   3   1  10   5  17  51  71  63  62 101   2  81 368 364
 324  79  21 109 102   4  98  92  26  73 115  86  52  29  30  33  32   8
 100  44  80  97  64  39  34  27  82  94 110 111  84  66 104  28 258 112
  65  67  55  88  54 292  83 105 280 394  24 103 366 249  22  91  11 108
 106  31  87  41 304 117  59  53  58 116  42 321  38  56  49 317   6  57
  19  25 315 123  46  89  61 312 299 130  74 298 119  20 286 136 129 124
 327 131 460 140 114 139 122 137 126 120 128 135 150 143 151 132 125 157
 147 138 156 164 346 159 160 161 333 381 149 154 297 163 314 155 323 340
 356 142 328 144 336 248 302 175 344 382 146 170 166 338 167 310 148 165
```

```
 172 171 145 121 178 305 173 152 354 347 158 185 349 183 352 177 200 192
 361 207 174 330 134 350 334 283 153 197 133 241 193 235 194 261 260 216
 169 209 238 215 141 189 187 223 284 214 202 211 168 230 203 188 232 709
 219 162 196 190 259 228 176 250 201 186 199 180 206 205 224 222 182 210
 275 212 229 218 208 191 181 179 246 255 226 288 253 252 262 236 256 234
 254 468 213 237 198 195 239 263 265 274 217 220 307 221 233 257 227 276
 225 264 311 277 204 290 266 270 294 319 282 251 322 291 269 240 271 184
 231 268 247 273 300 301 267 244 306 293 309 272 242 295 285 243 308 398
 303 245 424 279 331 281 339 434 357 325 329 278 332 343 345 360 348 367
 353 373 374 406 400 326 379 399 316 341 320 385 355 363 358 296 422 390
 335 370 376 375 397 289 542 403 383 384 359 393 337 362 365 435 386 378
 313 351 287 471 462 411 450 318 372 371 454 532 445 389 388 407 443 437
 451 391 405 412 419 420 426 433 440 429 418 447 461 605 457 475 464 482
 626 489 496 503 510 517 524 531 538 545 552 559 566 573 580 587 594 601
 608 615 622 629 396 410 395 423 408 409 448 465 387 414 476 479 467 490
 493 478 504 507 458 518 521 377 444 380 463]
arrival_date_year: [2015 2016 2017]
arrival_date_month: ['July' 'August' 'September' 'October' 'November' 'December'
'January'
 'February' 'March' 'April' 'May' 'June']
arrival_date_week_number: [27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50
 51 52 53  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
 22 23 24 25 26]
arrival_date_day_of_month: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24
 25 26 27 28 29 30 31]
stays_in_weekend_nights: [ 0  1  2  4  3  6 13  8  5  7 12  9 16 18 19 10 14]
stays_in_week_nights: [ 0  1  2  3  4  5 10 11  8  6  7 15  9 12 33 20 14 16 21
13 30 19 24 40
 22 42 50 25 17 32 26 18 34 35 41]
adults: [ 2  1  3  4 40 26 50 27 55  0 20  6  5 10]
children: [ 0.  1.  2. 10.  3. nan]
babies: [ 0  1  2 10  9]
meal: ['BB' 'FB' 'HB' 'SC' 'Undefined']
country: ['PRT' 'GBR' 'USA' 'ESP' 'IRL' 'FRA' nan 'ROU' 'NOR' 'OMN' 'ARG' 'POL'
 'DEU' 'BEL' 'CHE' 'CN' 'GRC' 'ITA' 'NLD' 'DNK' 'RUS' 'SWE' 'AUS' 'EST'
 'CZE' 'BRA' 'FIN' 'MOZ' 'BWA' 'LUX' 'SVN' 'ALB' 'IND' 'CHN' 'MEX' 'MAR'
 'UKR' 'SMR' 'LVA' 'PRI' 'SRB' 'CHL' 'AUT' 'BLR' 'LTU' 'TUR' 'ZAF' 'AGO'
 'ISR' 'CYM' 'ZMB' 'CPV' 'ZWE' 'DZA' 'KOR' 'CRI' 'HUN' 'ARE' 'TUN' 'JAM'
 'HRV' 'HKG' 'IRN' 'GEO' 'AND' 'GIB' 'URY' 'JEY' 'CAF' 'CYP' 'COL' 'GGY'
 'KWT' 'NGA' 'MDV' 'VEN' 'SVK' 'FJI' 'KAZ' 'PAK' 'IDN' 'LBN' 'PHL' 'SEN'
 'SYC' 'AZE' 'BHR' 'NZL' 'THA' 'DOM' 'MKD' 'MYS' 'ARM' 'JPN' 'LKA' 'CUB'
 'CMR' 'BIH' 'MUS' 'COM' 'SUR' 'UGA' 'BGR' 'CIV' 'JOR' 'SYR' 'SGP' 'BDI'
 'SAU' 'VNM' 'PLW' 'QAT' 'EGY' 'PER' 'MLT' 'MWI' 'ECU' 'MDG' 'ISL' 'UZB'
 'NPL' 'BHS' 'MAC' 'TGO' 'TWN' 'DJI' 'STP' 'KNA' 'ETH' 'IRQ' 'HND' 'RWA'
 'KHM' 'MCO' 'BGD' 'IMN' 'TJK' 'NIC' 'BEN' 'VGB' 'TZA' 'GAB' 'GHA' 'TMP'
 'GLP' 'KEN' 'LIE' 'GNB' 'MNE' 'UMI' 'MYT' 'FRO' 'MMR' 'PAN' 'BFA' 'LBY'
```

```
 'MLI' 'NAM' 'BOL' 'PRY' 'BRB' 'ABW' 'AIA' 'SLV' 'DMA' 'PYF' 'GUY' 'LCA'
 'ATA' 'GTM' 'ASM' 'MRT' 'NCL' 'KIR' 'SDN' 'ATF' 'SLE' 'LAO']
market_segment: ['Direct' 'Corporate' 'Online TA' 'Offline TA/TO'
'Complementary' 'Groups'
 'Undefined' 'Aviation']
distribution_channel: ['Direct' 'Corporate' 'TA/TO' 'Undefined' 'GDS']
is_repeated_guest: [0 1]
previous_cancellations: [ 0  1  2  3 26 25 14  4 24 19  5 21  6 13 11]
previous_bookings_not_canceled: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 20 21 22 23 24
 25 27 28 29 30 19 26 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72]
reserved_room_type: ['C' 'A' 'D' 'E' 'G' 'F' 'H' 'L' 'P' 'B']
assigned_room_type: ['C' 'A' 'D' 'E' 'G' 'F' 'I' 'B' 'H' 'P' 'L' 'K']
booking_changes: [ 3  4  0  1  2  5 17  6  8  7 10 16  9 13 12 20 14 15 11 21
18]
deposit_type: ['No Deposit' 'Refundable' 'Non Refund']
agent: [ nan 304. 240. 303.  15. 241.   8. 250. 115.   5. 175. 134. 156. 243.
 242.   3. 105.  40. 147. 306. 184.  96.   2. 127.  95. 146.   9. 177.
   6. 143. 244. 149. 167. 300. 171. 305.  67. 196. 152. 142. 261. 104.
  36.  26.  29. 258. 110.  71. 181.  88. 251. 275.  69. 248. 208. 256.
 314. 126. 281. 273. 253. 185. 330. 334. 328. 326. 321. 324. 313.  38.
 155.  68. 335. 308. 332.  94. 348. 310. 339. 375.  66. 327. 387. 298.
  91. 245. 385. 257. 393. 168. 405. 249. 315.  75. 128. 307.  11. 436.
   1. 201. 183. 223. 368. 336. 291. 464. 411. 481.  10. 154. 468. 410.
 390. 440. 495. 492. 493. 434.  57. 531. 420. 483. 526. 472. 429.  16.
 446.  34.  78. 139. 252. 270.  47. 114. 301. 193. 182. 135. 350. 195.
 352. 355. 159. 363. 384. 360. 331. 367.  64. 406. 163. 414. 333. 427.
 431. 430. 426. 438. 433. 418. 441. 282. 432.  72. 450. 180. 454. 455.
  59. 451. 254. 358. 469. 165. 467. 510. 337. 476. 502. 527. 479. 508.
 535. 302. 497. 187.  13.   7.  27.  14.  22.  17.  28.  42.  20.  19.
  45.  37.  61.  39.  21.  24.  41.  50.  30.  54.  52.  12.  44.  31.
  83.  32.  63.  60.  55.  56.  89.  87. 118.  86.  85. 210. 214. 129.
 179. 138. 174. 170. 153.  93. 151. 119.  35. 173.  58.  53. 133.  79.
 235. 192. 191. 236. 162. 215. 157. 287. 132. 234.  98.  77. 103. 107.
 262. 220. 121. 205. 378.  23. 296. 290. 229.  33. 286. 276. 425. 484.
 323. 403. 219. 394. 509. 111. 423.   4.  70.  82.  81.  74.  92.  99.
  90. 112. 117. 106. 148. 158. 144. 211. 213. 216. 232. 150. 267. 227.
 247. 278. 280. 285. 289. 269. 295. 265. 288. 122. 294. 325. 341. 344.
 346. 359. 283. 364. 370. 371.  25. 141. 391. 397. 416. 404. 299. 197.
  73. 354. 444. 408. 461. 388. 453. 459. 474. 475. 480. 449.]
company: [ nan 110. 113. 270. 178. 240. 154. 144. 307. 268.  59. 204. 312. 318.
  94. 174. 274. 195. 223. 317. 281. 118.  53. 286.  12.  47. 324. 342.
 373. 371. 383.  86.  82. 218.  88.  31. 397. 392. 405. 331. 367.  20.
  83. 416.  51. 395. 102.  34.  84. 360. 394. 457. 382. 461. 478. 386.
 112. 486. 421.   9. 308. 135. 224. 504. 269. 356. 498. 390. 513. 203.
 263. 477. 521. 169. 515. 445. 337. 251. 428. 292. 388. 130. 250. 355.
```

```
 254. 543. 531. 528.  62. 120.  42.  81. 116. 530. 103.  39.  16.  92.
  61. 501. 165. 291. 290.  43. 325. 192. 108. 200. 465. 287. 297. 490.
 482. 207. 282. 437. 225. 329. 272.  28.  77. 338.  72. 246. 319. 146.
 159. 380. 323. 511. 407. 278.  80. 403. 399.  14. 137. 343. 346. 347.
 349. 289. 351. 353.  54.  99. 358. 361. 362. 366. 372. 365. 277. 109.
 377. 379.  22. 378. 330. 364. 401. 232. 255. 384. 167. 212. 514. 391.
 400. 376. 402. 396. 302. 398.   6. 370. 369. 409. 168. 104. 408. 413.
 148.  10. 333. 419. 415. 424. 425. 423. 422. 435. 439. 442. 448. 443.
 454. 444.  52. 459. 458. 456. 460. 447. 470. 466. 484. 184. 485.  32.
 487. 491. 494. 193. 516. 496. 499.  29.  78. 520. 507. 506. 512. 126.
  64. 242. 518. 523. 539. 534. 436. 525. 541.  40. 455. 410.  45.  38.
  49.  48.  67.  68.  65.  91.  37.   8. 179. 209. 219. 221. 227. 153.
 186. 253. 202. 216. 275. 233. 280. 309. 321.  93. 316.  85. 107. 350.
 279. 334. 348. 150.  73. 385. 418. 197. 450. 452. 115.  46.  76.  96.
 100. 105. 101. 122.  11. 139. 142. 127. 143. 140. 149. 163. 160. 180.
 238. 183. 222. 185. 217. 215. 213. 237. 230. 234.  35. 245. 158. 258.
 259. 260. 411. 257. 271.  18. 106. 210. 273.  71. 284. 301. 305. 293.
 264. 311. 304. 313. 288. 320. 314. 332. 341. 352. 243. 368. 393. 132.
 220. 412. 420. 426. 417. 429. 433. 446. 357. 479. 483. 489. 229. 481.
 497. 451. 492.]
days_in_waiting_list: [  0  50  47  65 122  75 101 150 125  14  60  34 100  22
121  61  39   5
   1   8 107  43  52   2  11 142 116  13  44  97  83   4 113  18  20 185
  93 109   6  37 105 154  64  99  38  48  33  77  21  80  59  40  58  89
  53  49  69  87  91  57 111  79  98  85  63  15   3  41 224  31  56 187
 176  71  55  96 236 259 207 215 160 120  30  32  27  62  24 108 147 379
  70  35 178 330 223 174 162 391  68 193  10  76  16  28   9 165  17  25
  46   7  84 175 183  23 117  12  54  26  73  45  19  42  72  81  92  74
 167  36]
customer_type: ['Transient' 'Contract' 'Transient-Party' 'Group']
adr: [  0.    75.    98.   … 266.75 209.25 157.71]
required_car_parking_spaces: [0 1 2 8 3]
total_of_special_requests: [0 1 3 2 4 5]
reservation_status: ['Check-Out' 'Canceled' 'No-Show']
reservation_status_date: ['2015-07-01' '2015-07-02' '2015-07-03' '2015-05-06'
'2015-04-22'
 '2015-06-23' '2015-07-05' '2015-07-06' '2015-07-07' '2015-07-08'
 '2015-05-11' '2015-07-15' '2015-07-16' '2015-05-29' '2015-05-19'
 '2015-06-19' '2015-05-23' '2015-05-18' '2015-07-09' '2015-06-02'
 '2015-07-13' '2015-07-04' '2015-06-29' '2015-06-16' '2015-06-18'
 '2015-06-12' '2015-06-09' '2015-05-26' '2015-07-11' '2015-07-12'
 '2015-07-17' '2015-04-15' '2015-05-13' '2015-07-10' '2015-05-20'
 '2015-05-12' '2015-07-14' '2015-06-17' '2015-05-01' '2015-03-30'
 '2015-07-19' '2015-06-03' '2015-06-26' '2015-05-14' '2015-07-20'
 '2015-05-07' '2015-05-28' '2015-04-13' '2015-03-25' '2015-07-21'
 '2015-06-27' '2015-07-18' '2015-07-23' '2015-06-08' '2015-06-22'
 '2015-06-24' '2015-03-05' '2015-06-01' '2015-04-24' '2015-07-22'
 '2015-05-27' '2015-04-06' '2015-04-11' '2015-07-25' '2015-07-28'
```

'2015-07-29' '2015-06-25' '2015-07-24' '2015-06-05' '2015-06-30'
'2015-06-13' '2015-06-11' '2015-07-30' '2015-07-27' '2015-04-29'
'2015-06-04' '2015-07-26' '2015-08-01' '2015-08-02' '2015-06-15'
'2015-04-23' '2015-07-31' '2015-05-25' '2015-08-03' '2015-04-17'
'2015-08-04' '2015-08-06' '2015-05-15' '2015-05-09' '2015-03-17'
'2015-05-22' '2015-08-07' '2015-04-04' '2015-08-05' '2015-08-08'
'2015-08-10' '2015-05-04' '2015-06-06' '2015-08-09' '2015-08-15'
'2015-08-11' '2015-03-28' '2015-08-14' '2015-08-12' '2015-08-16'
'2015-05-16' '2015-08-21' '2015-08-13' '2015-08-17' '2015-04-20'
'2015-08-18' '2015-08-23' '2015-08-22' '2015-08-19' '2015-08-20'
'2015-08-29' '2015-03-31' '2015-05-30' '2015-08-25' '2015-04-14'
'2015-08-24' '2015-03-24' '2015-05-21' '2015-08-28' '2015-08-26'
'2015-08-27' '2015-08-30' '2015-08-31' '2015-09-06' '2015-09-03'
'2015-09-04' '2015-09-02' '2015-09-01' '2015-09-05' '2015-06-20'
'2015-09-07' '2015-09-10' '2015-09-11' '2015-09-08' '2015-09-09'
'2015-09-13' '2015-09-15' '2015-04-10' '2015-01-02' '2014-11-18'
'2015-09-12' '2015-09-17' '2015-09-14' '2015-04-07' '2015-09-19'
'2015-09-16' '2015-09-20' '2015-01-18' '2015-10-23' '2015-01-22'
'2015-01-01' '2015-09-22' '2015-09-24' '2015-09-18' '2015-09-21'
'2015-09-30' '2015-09-25' '2015-09-27' '2015-09-28' '2015-10-12'
'2015-09-29' '2015-09-23' '2015-10-01' '2015-09-26' '2015-04-18'
'2015-10-02' '2015-10-04' '2015-10-08' '2015-10-03' '2015-10-07'
'2015-10-09' '2015-10-11' '2015-10-05' '2015-10-06' '2015-10-10'
'2015-10-14' '2015-10-15' '2015-10-18' '2015-10-13' '2015-10-20'
'2015-10-19' '2015-10-31' '2015-10-16' '2015-10-21' '2015-10-22'
'2015-10-17' '2015-10-24' '2015-10-25' '2015-10-28' '2015-10-27'
'2015-10-26' '2015-10-30' '2015-11-05' '2015-10-29' '2015-11-03'
'2015-11-07' '2015-11-04' '2015-11-01' '2015-11-02' '2015-11-17'
'2015-11-06' '2015-11-10' '2015-11-08' '2015-11-09' '2015-11-15'
'2015-11-16' '2015-11-11' '2015-11-12' '2015-11-14' '2015-11-13'
'2015-11-18' '2015-11-22' '2015-11-19' '2015-11-21' '2015-11-20'
'2015-11-24' '2015-11-25' '2015-11-23' '2015-11-28' '2015-11-26'
'2015-11-27' '2015-11-29' '2015-12-04' '2015-12-01' '2015-12-06'
'2015-12-08' '2015-12-02' '2015-12-03' '2015-12-31' '2015-12-05'
'2015-12-10' '2015-12-17' '2015-11-30' '2015-12-12' '2015-12-07'
'2016-01-05' '2015-12-11' '2015-12-13' '2015-12-15' '2015-12-16'
'2015-12-19' '2015-12-18' '2015-12-26' '2015-12-27' '2015-12-22'
'2015-12-23' '2015-12-24' '2015-12-29' '2015-12-28' '2015-12-20'
'2015-12-30' '2016-01-02' '2016-01-01' '2015-12-25' '2016-01-03'
'2016-01-04' '2016-01-11' '2016-01-07' '2015-12-21' '2016-01-09'
'2016-01-10' '2016-01-08' '2016-01-06' '2016-01-12' '2016-01-13'
'2016-01-23' '2016-02-09' '2016-01-15' '2016-01-16' '2016-01-17'
'2016-01-19' '2016-01-18' '2016-01-21' '2016-01-24' '2016-01-22'
'2016-01-29' '2016-01-27' '2016-01-25' '2016-03-08' '2016-01-26'
'2016-01-20' '2016-01-30' '2016-02-01' '2016-02-02' '2016-02-08'
'2016-02-07' '2016-01-28' '2016-02-05' '2016-02-03' '2016-02-13'
'2016-02-10' '2016-02-04' '2016-02-12' '2016-02-11' '2016-02-16'
'2016-02-14' '2016-02-15' '2016-02-20' '2016-02-06' '2016-01-14'

```
'2016-02-17' '2016-02-21' '2016-02-24' '2016-02-25' '2016-02-19'
'2016-02-18' '2016-02-26' '2016-02-23' '2016-03-05' '2016-02-22'
'2016-02-27' '2016-03-03' '2016-03-24' '2016-03-04' '2016-02-29'
'2016-03-01' '2016-03-02' '2016-03-30' '2016-03-07' '2016-03-14'
'2016-03-21' '2016-03-09' '2016-03-12' '2016-03-22' '2016-03-10'
'2016-03-11' '2016-03-20' '2016-03-15' '2016-03-17' '2016-03-16'
'2016-03-19' '2016-03-27' '2016-03-18' '2016-03-26' '2016-03-31'
'2016-03-28' '2016-03-29' '2016-04-01' '2016-03-23' '2016-04-02'
'2016-03-25' '2016-03-13' '2016-04-04' '2016-04-03' '2016-04-05'
'2016-04-08' '2016-04-06' '2016-04-09' '2016-04-12' '2016-04-16'
'2016-04-17' '2016-04-27' '2016-04-14' '2016-04-18' '2016-04-21'
'2016-04-19' '2016-04-20' '2016-04-10' '2016-04-13' '2016-04-11'
'2016-04-07' '2016-04-15' '2016-04-22' '2016-04-23' '2016-04-26'
'2016-04-28' '2016-04-24' '2016-04-25' '2016-04-29' '2016-04-30'
'2016-05-01' '2016-05-10' '2016-05-02' '2016-05-07' '2016-05-08'
'2016-05-12' '2016-05-04' '2016-05-06' '2016-05-03' '2016-05-09'
'2016-05-05' '2016-05-13' '2016-05-14' '2016-05-18' '2016-05-19'
'2016-05-15' '2016-05-16' '2016-05-11' '2016-05-21' '2016-05-22'
'2016-05-20' '2016-05-24' '2016-05-25' '2016-05-26' '2016-05-23'
'2016-05-27' '2016-05-17' '2016-05-29' '2016-05-28' '2016-05-30'
'2016-05-31' '2016-06-01' '2016-06-03' '2016-06-08' '2016-06-02'
'2016-06-05' '2016-06-06' '2016-06-13' '2016-06-07' '2016-06-10'
'2016-06-11' '2016-06-16' '2016-06-12' '2016-06-14' '2016-06-17'
'2016-06-04' '2016-06-18' '2016-06-21' '2016-06-09' '2016-06-24'
'2016-06-20' '2016-06-25' '2016-06-22' '2016-06-26' '2016-06-23'
'2016-07-01' '2016-06-15' '2016-06-28' '2016-07-02' '2016-06-19'
'2016-06-27' '2016-07-04' '2016-06-30' '2016-07-05' '2016-07-08'
'2016-07-09' '2016-07-07' '2016-07-12' '2016-06-29' '2016-07-10'
'2016-07-15' '2016-07-03' '2016-07-16' '2016-07-14' '2016-07-18'
'2016-07-13' '2016-07-06' '2016-07-20' '2016-07-21' '2016-07-23'
'2016-07-19' '2016-07-11' '2016-07-28' '2016-07-17' '2016-07-25'
'2016-07-22' '2016-07-29' '2016-08-03' '2016-08-02' '2016-08-04'
'2016-08-08' '2016-08-10' '2016-08-01' '2016-08-06' '2016-03-06'
'2016-08-05' '2016-07-26' '2016-08-07' '2016-07-30' '2016-07-24'
'2016-08-12' '2016-07-27' '2016-08-13' '2016-08-18' '2016-08-16'
'2016-08-15' '2016-08-17' '2016-08-11' '2016-07-31' '2016-08-19'
'2016-09-01' '2016-08-23' '2016-08-26' '2016-08-20' '2016-08-21'
'2016-09-04' '2016-08-22' '2016-08-27' '2016-08-25' '2016-08-09'
'2016-09-05' '2016-08-24' '2016-09-10' '2016-08-29' '2016-09-09'
'2016-08-30' '2016-09-13' '2016-08-31' '2016-09-14' '2016-09-12'
'2016-09-15' '2016-08-14' '2016-09-02' '2016-09-08' '2016-09-19'
'2016-09-16' '2016-09-07' '2016-09-21' '2016-09-06' '2016-09-22'
'2016-09-17' '2016-09-20' '2016-09-03' '2016-09-26' '2016-09-23'
'2016-09-18' '2016-09-29' '2016-10-02' '2016-10-01' '2016-09-27'
'2016-09-25' '2016-10-05' '2016-09-11' '2016-09-30' '2016-10-09'
'2016-10-03' '2016-10-06' '2016-10-11' '2016-09-24' '2016-10-13'
'2016-09-28' '2016-10-08' '2016-10-07' '2016-10-16' '2016-08-28'
'2016-10-17' '2016-10-18' '2016-10-10' '2016-10-04' '2016-10-15'
```

'2016-10-19' '2016-10-21' '2016-10-12' '2016-10-24' '2016-10-26'
'2016-10-23' '2016-10-20' '2016-10-25' '2016-10-27' '2016-10-28'
'2016-10-30' '2016-10-29' '2016-11-01' '2016-11-04' '2016-10-14'
'2016-11-07' '2016-11-03' '2016-11-10' '2016-11-14' '2016-11-02'
'2016-10-31' '2016-11-11' '2016-11-08' '2016-11-05' '2016-11-25'
'2016-11-09' '2016-11-20' '2016-11-21' '2016-10-22' '2016-11-22'
'2016-11-16' '2016-11-23' '2016-11-17' '2016-11-06' '2016-11-15'
'2016-11-13' '2016-11-12' '2016-11-27' '2016-11-19' '2016-11-30'
'2016-11-18' '2016-12-02' '2016-12-04' '2016-11-29' '2016-12-07'
'2016-11-28' '2016-12-03' '2016-12-06' '2016-11-24' '2016-12-08'
'2016-12-05' '2016-12-10' '2016-12-13' '2016-12-14' '2016-12-16'
'2016-12-15' '2016-12-17' '2016-12-19' '2016-12-21' '2016-12-20'
'2016-12-22' '2016-12-23' '2016-12-24' '2016-12-01' '2016-12-27'
'2016-12-29' '2016-12-30' '2016-12-12' '2017-01-02' '2016-12-11'
'2017-01-03' '2017-01-04' '2017-01-01' '2016-12-26' '2017-01-06'
'2016-12-28' '2016-12-18' '2017-01-10' '2017-01-11' '2017-01-07'
'2017-01-12' '2017-01-16' '2017-01-14' '2017-01-13' '2017-01-05'
'2017-01-17' '2017-01-20' '2016-12-09' '2017-01-26' '2016-12-31'
'2017-01-23' '2017-01-27' '2017-01-28' '2017-01-19' '2017-01-25'
'2017-01-24' '2017-01-29' '2017-01-18' '2016-12-25' '2017-01-15'
'2017-01-21' '2017-02-01' '2017-02-02' '2017-01-31' '2017-02-03'
'2017-02-04' '2017-02-06' '2017-02-07' '2017-02-08' '2017-01-30'
'2017-02-09' '2017-01-09' '2017-02-11' '2017-02-10' '2017-02-12'
'2017-02-13' '2017-02-14' '2017-02-16' '2017-02-17' '2017-02-18'
'2017-02-19' '2017-02-20' '2017-02-15' '2017-02-21' '2017-02-22'
'2017-02-26' '2017-02-23' '2017-02-24' '2017-02-25' '2017-02-28'
'2017-03-05' '2017-02-27' '2017-03-03' '2017-03-06' '2017-03-02'
'2017-03-08' '2017-03-09' '2017-03-10' '2017-03-07' '2017-03-12'
'2017-03-13' '2017-03-14' '2017-03-01' '2017-03-18' '2017-03-17'
'2017-03-24' '2017-03-22' '2017-03-26' '2017-03-27' '2017-03-11'
'2017-03-28' '2017-03-29' '2017-03-30' '2017-03-31' '2017-03-19'
'2017-01-22' '2017-04-02' '2017-03-20' '2017-04-03' '2017-01-08'
'2017-03-23' '2017-04-05' '2017-02-05' '2017-04-04' '2017-03-15'
'2017-04-07' '2017-03-25' '2017-04-08' '2017-04-06' '2017-03-21'
'2017-04-10' '2017-04-01' '2017-04-11' '2017-04-13' '2017-04-15'
'2017-04-12' '2017-03-04' '2017-04-19' '2017-04-22' '2017-04-20'
'2017-05-02' '2017-04-09' '2017-04-23' '2017-04-24' '2017-04-16'
'2017-04-28' '2017-04-18' '2017-04-26' '2017-04-25' '2017-04-17'
'2017-04-21' '2017-05-03' '2017-05-04' '2017-03-16' '2017-05-05'
'2017-04-29' '2017-04-14' '2017-05-08' '2017-04-27' '2017-05-11'
'2017-05-01' '2017-05-10' '2017-05-13' '2017-05-06' '2017-05-14'
'2017-05-16' '2017-04-30' '2017-05-15' '2017-05-07' '2017-05-09'
'2017-05-17' '2017-05-21' '2017-05-12' '2017-05-22' '2017-05-24'
'2017-05-23' '2017-05-25' '2017-05-26' '2017-05-28' '2017-05-27'
'2017-05-29' '2017-05-19' '2017-05-31' '2017-05-20' '2017-06-01'
'2017-05-30' '2017-06-02' '2016-11-26' '2017-06-04' '2017-06-05'
'2017-06-06' '2017-06-07' '2017-05-18' '2017-06-09' '2017-06-10'
'2017-06-11' '2017-06-12' '2017-06-14' '2017-06-08' '2017-06-16'

```
'2017-06-13' '2017-06-03' '2017-06-24' '2017-06-20' '2017-06-19'
'2017-06-21' '2017-06-26' '2017-06-27' '2017-06-22' '2017-06-28'
'2017-06-15' '2017-06-29' '2017-06-30' '2017-06-18' '2017-07-04'
'2017-07-08' '2017-07-05' '2017-07-03' '2017-07-07' '2017-07-01'
'2017-07-06' '2017-07-11' '2017-07-12' '2017-06-23' '2017-07-13'
'2017-07-02' '2017-07-10' '2017-07-14' '2017-07-15' '2017-07-16'
'2017-07-18' '2017-07-17' '2017-07-19' '2017-07-20' '2017-07-21'
'2017-06-25' '2017-06-17' '2017-07-24' '2017-07-26' '2017-07-09'
'2017-07-27' '2017-07-28' '2017-07-31' '2017-07-29' '2017-07-22'
'2017-08-02' '2017-08-01' '2017-08-03' '2017-08-04' '2017-07-25'
'2017-07-23' '2017-08-09' '2017-08-10' '2017-07-30' '2017-08-07'
'2017-08-13' '2017-08-05' '2017-08-14' '2017-08-08' '2017-08-16'
'2017-08-17' '2017-08-15' '2017-08-18' '2017-08-20' '2017-08-22'
'2017-08-06' '2017-08-25' '2017-08-26' '2017-08-23' '2017-08-11'
'2017-08-27' '2017-08-21' '2017-08-29' '2017-08-31' '2017-08-12'
'2017-08-19' '2016-01-31' '2017-09-01' '2017-08-28' '2015-04-03'
'2015-01-21' '2015-01-28' '2015-01-29' '2015-01-30' '2015-02-02'
'2015-02-05' '2015-02-06' '2015-02-09' '2015-02-10' '2015-02-11'
'2015-02-12' '2015-02-19' '2015-02-20' '2015-02-23' '2015-02-24'
'2015-02-25' '2015-02-26' '2015-02-27' '2015-03-03' '2015-03-04'
'2015-03-06' '2015-03-09' '2015-03-11' '2015-03-12' '2015-03-18'
'2015-04-02' '2015-06-14' '2015-04-08' '2015-04-16' '2015-04-25'
'2015-04-28' '2015-05-08' '2017-09-06' '2016-02-28' '2015-12-09'
'2015-12-14' '2017-09-09' '2017-09-02' '2017-08-24' '2017-08-30'
'2017-09-03' '2017-09-04' '2017-09-05' '2017-09-07' '2017-09-08'
'2017-09-10' '2017-09-12' '2017-09-14' '2015-04-30' '2015-04-21'
'2015-04-05' '2015-03-13' '2015-05-05' '2015-03-29' '2015-06-10'
'2015-04-27' '2014-10-17' '2015-01-20' '2015-02-17' '2015-03-10'
'2015-03-23']
```

[441]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# 1.Removing Inconsistent values and Outliers "))
```

5 1.Removing Inconsistent values and Outliers

[442]:
```
df.fillna(0, inplace = True)
```

[443]:
```
df.isnull().sum()
```

[443]:
```
hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
```

```
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          0
babies                            0
meal                              0
country                           0
market_segment                    0
distribution_channel              0
is_repeated_guest                 0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
assigned_room_type                0
booking_changes                   0
deposit_type                      0
agent                             0
company                           0
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
reservation_status                0
reservation_status_date           0
dtype: int64
```

[444]:
```
df['meal'].value_counts()
```

[444]:
```
meal
BB           92310
HB           14463
SC           10650
Undefined     1169
FB             798
Name: count, dtype: int64
```

[445]:
```
df['children'].value_counts()
```

[445]:
```
children
0.0     110800
1.0       4861
2.0       3652
3.0         76
10.0         1
```

```
Name: count, dtype: int64
```

[446]:
```
df['adults'].value_counts()
```

[446]:
```
adults
2     89680
1     23027
3      6202
0       403
4        62
26        5
27        2
20        2
5         2
40        1
50        1
55        1
6         1
10        1
Name: count, dtype: int64
```

[447]:
```
len(df[df['adults']== 0])
```

[447]:
```
403
```

[448]:
```
len(df[df['babies']== 0])
```

[448]:
```
118473
```

[449]:
```
filter = (df['children'] == 0) & (df['adults'] == 0) & (df['babies'] == 0)
df[filter]
```

[449]:
```
                hotel  is_canceled  lead_time  arrival_date_year  \
2224    Resort Hotel            0          1               2015
2409    Resort Hotel            0          0               2015
3181    Resort Hotel            0         36               2015
3684    Resort Hotel            0        165               2015
3708    Resort Hotel            0        165               2015
…                …          …         …                  …
115029    City Hotel            0        107               2017
115091    City Hotel            0          1               2017
116251    City Hotel            0         44               2017
116534    City Hotel            0          2               2017
117087    City Hotel            0        170               2017

       arrival_date_month  arrival_date_week_number  \
2224              October                        41
```

```
2409              October                          42
3181             November                          47
3684             December                          53
3708             December                          53
…                   …                           …
115029              June                          26
115091              June                          26
116251              July                          28
116534              July                          28
117087              July                          30

        arrival_date_day_of_month  stays_in_weekend_nights  \
2224                            6                       0
2409                           12                       0
3181                           20                       1
3684                           30                       1
3708                           30                       2
…                             …                        …
115029                         27                       0
115091                         30                       0
116251                         15                       1
116534                         15                       2
117087                         27                       0

        stays_in_week_nights  adults  …  deposit_type  agent company  \
2224                       3       0  …    No Deposit    0.0   174.0
2409                       0       0  …    No Deposit    0.0   174.0
3181                       2       0  …    No Deposit   38.0     0.0
3684                       4       0  …    No Deposit  308.0     0.0
3708                       4       0  …    No Deposit  308.0     0.0
…                        …     …  …             …    …     …
115029                     3       0  …    No Deposit    7.0     0.0
115091                     1       0  …    No Deposit    0.0     0.0
116251                     1       0  …    No Deposit  425.0     0.0
116534                     5       0  …    No Deposit    9.0     0.0
117087                     2       0  …    No Deposit   52.0     0.0

       days_in_waiting_list    customer_type     adr  \
2224                      0  Transient-Party    0.00
2409                      0        Transient    0.00
3181                      0  Transient-Party    0.00
3684                    122  Transient-Party    0.00
3708                    122  Transient-Party    0.00
…                      …              …     …
115029                    0        Transient  100.80
115091                    0        Transient    0.00
116251                    0        Transient   73.80
```

```
116534                    0  Transient-Party   22.86
117087                    0        Transient    0.00

        required_car_parking_spaces  total_of_special_requests  \
2224                              0                          0
2409                              0                          0
3181                              0                          0
3684                              0                          0
3708                              0                          0
…                               …                          …
115029                            0                          0
115091                            1                          1
116251                            0                          0
116534                            0                          1
117087                            0                          0

        reservation_status reservation_status_date
2224             Check-Out              2015-10-06
2409             Check-Out              2015-10-12
3181             Check-Out              2015-11-23
3684             Check-Out              2016-01-04
3708             Check-Out              2016-01-05
…                      …                       …
115029           Check-Out              2017-06-30
115091           Check-Out              2017-07-01
116251           Check-Out              2017-07-17
116534           Check-Out              2017-07-22
117087           Check-Out              2017-07-29

[180 rows x 32 columns]
```

[450]:
```
data = df[~filter]
```

[451]:
```
data.shape
```

[451]:
```
(119210, 32)
```

[452]:
```
# Step 1: Checking the unique values in the 'adults' column
adults_value_counts = df['adults'].value_counts()

# Step 2: Get the unique values in 'adults' column
unique_adults = df['adults'].unique()

# Step 3: Filter rows where adults, children, and babies are all zero
filter_no_guests = (df['children'] == 0) & (df['adults'] == 0) & (df['babies']
 ↪== 0)
rows_with_no_guests = df[filter_no_guests]
```

```
# Step 4: Exclude the filtered rows (with no guests) from the dataset
cleaned_data = df[~filter_no_guests]

# Print the results
print(adults_value_counts)
print(unique_adults)
print(f"Number of rows with no guests: {len(rows_with_no_guests)}")
print(f"Shape of cleaned data: {cleaned_data.shape}")
adults_value_counts, unique_adults, len(rows_with_no_guests), cleaned_data.shape
```

```
adults
2     89680
1     23027
3      6202
0       403
4        62
26        5
27        2
20        2
5         2
40        1
50        1
55        1
6         1
10        1
Name: count, dtype: int64
[ 2  1  3  4 40 26 50 27 55  0 20  6  5 10]
Number of rows with no guests: 180
Shape of cleaned data: (119210, 32)
```

[452]:
```
(adults
 2     89680
 1     23027
 3      6202
 0       403
 4        62
 26        5
 27        2
 20        2
 5         2
 40        1
 50        1
 55        1
 6         1
 10        1
 Name: count, dtype: int64,
```

```
array([ 2,  1,  3,  4, 40, 26, 50, 27, 55,  0, 20,  6,  5, 10],
      dtype=int64),
180,
(119210, 32))
```

[453]:
```
print(adults_value_counts)
print(unique_adults)
print(f"Number of rows with no guests: {len(rows_with_no_guests)}")
print(f"Shape of cleaned data: {cleaned_data.shape}")
```

```
adults
2     89680
1     23027
3      6202
0       403
4        62
26        5
27        2
20        2
5         2
40        1
50        1
55        1
6         1
10        1
Name: count, dtype: int64
[ 2  1  3  4 40 26 50 27 55  0 20  6  5 10]
Number of rows with no guests: 180
Shape of cleaned data: (119210, 32)
```

[454]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Column Data Type Conversion"))
```

6 Column Data Type Conversion

[455]:
```
# Convert reservation_status_date to datetime format
df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date'])
```

[456]:
```
from IPython.display import display, Markdown

# Exploratory Data Analysis Section
eda_justification = """
# 2. Exploratory Data Analysis (EDA)
```

```
## Insights for Hotel Management
- **Cancellation Percentages**: Calculating cancellation percentages for City␣
 ↪and Resort hotels helps the management understand which type of hotel faces␣
 ↪more cancellations, enabling targeted actions to minimize cancellations.
- **Most Frequently Ordered Meal Types**: Identifying the most frequently␣
 ↪ordered meal types allows the hotel to ensure sufficient inventory and␣
 ↪improve customer satisfaction.
- **Returning Guests**: Determining the number of returning guests helps the␣
 ↪hotel understand customer loyalty and identify patterns in guest retention.
- **Most Booked Room Types**: Finding the most booked room types helps the␣
 ↪hotel optimize room availability and pricing strategies.
- **Correlation Between Room Types and Cancellations**: Exploring correlations␣
 ↪between room types and cancellations can provide insights into whether␣
 ↪certain rooms are more likely to be canceled, helping management improve␣
 ↪booking policies.
- **Country of our home guests**: By understanding where most of their guests␣
 ↪are coming from, the hotel can focus its marketing efforts more effectively␣
 ↪on countries that generate the highest number of bookings.
These insights were visualized using suitable chart types, such as bar charts,␣
 ↪pie charts, line charts, and heatmaps to effectively communicate findings to␣
 ↪stakeholders.
"""
display(Markdown(eda_justification))
```

7 2. Exploratory Data Analysis (EDA)

7.1 Insights for Hotel Management

- **Cancellation Percentages**: Calculating cancellation percentages for City and Resort hotels helps the management understand which type of hotel faces more cancellations, enabling targeted actions to minimize cancellations.
- **Most Frequently Ordered Meal Types**: Identifying the most frequently ordered meal types allows the hotel to ensure sufficient inventory and improve customer satisfaction.
- **Returning Guests**: Determining the number of returning guests helps the hotel understand customer loyalty and identify patterns in guest retention.
- **Most Booked Room Types**: Finding the most booked room types helps the hotel optimize room availability and pricing strategies.
- **Correlation Between Room Types and Cancellations**: Exploring correlations between room types and cancellations can provide insights into whether certain rooms are more likely to be canceled, helping management improve booking policies.
- **Country of our home guests**: By understanding where most of their guests are coming from, the hotel can focus its marketing efforts more effectively on countries that generate the highest number of bookings. These insights were visualized using suitable chart types, such as bar charts, pie charts, line charts, and heatmaps to effectively communicate findings to stakeholders.

[457]:
```
data.head(2)
```

[457]:
```
          hotel  is_canceled  lead_time  arrival_date_year arrival_date_month  \
0  Resort Hotel            0        342               2015               July
1  Resort Hotel            0        737               2015               July

   arrival_date_week_number  arrival_date_day_of_month  \
0                        27                          1
1                        27                          1

   stays_in_weekend_nights  stays_in_week_nights  adults  …  deposit_type  \
0                        0                    0       2  …    No Deposit
1                        0                    0       2  …    No Deposit

   agent company days_in_waiting_list customer_type  adr  \
0    0.0     0.0                    0     Transient  0.0
1    0.0     0.0                    0     Transient  0.0

   required_car_parking_spaces  total_of_special_requests  reservation_status  \
0                            0                          0           Check-Out
1                            0                          0           Check-Out

  reservation_status_date
0              2015-07-01
1              2015-07-01

[2 rows x 32 columns]
```

[458]:
```
data['hotel'].unique()
```

[458]:
```
array(['Resort Hotel', 'City Hotel'], dtype=object)
```

[459]:
```
data['is_canceled'].unique()
```

[459]:
```
array([0, 1], dtype=int64)
```

[460]:
```
resort = data[(data['hotel'] == 'Resort Hotel') & (data['is_canceled'] == 0)]
City = data[(data['hotel'] == 'City Hotel') & (data['is_canceled'] == 0)]
```

[461]:
```
resort.shape
```

[461]:
```
(28927, 32)
```

[462]:
```
City.shape
```

[462]:
```
(46084, 32)
```

[463]:
```
resort['country'].value_counts().index
```

[463]:
```
Index(['PRT', 'GBR', 'ESP', 'IRL', 'FRA', 'DEU',  'CN', 'NLD',     0, 'USA',
       …
       'MKD', 'SMR', 'BDI', 'SYR', 'CYM', 'UGA', 'COM', 'MUS', 'BIH', 'SAU'],
      dtype='object', name='country', length=119)
```

[464]:
```
labels = resort['country'].value_counts().index
values = resort['country'].value_counts()
```

[465]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Displaying Guest Country Wise  "))
```

8 Displaying Guest Country Wise

[466]:
```
import plotly.graph_objs as go
from plotly.offline import iplot
import plotly.express as px
```

[467]:
```
# Step 2:  unique values in 'hotel' and 'is_canceled' columns
print("Unique Hotels:", data['hotel'].unique())
print("Unique Cancellation Status:", data['is_canceled'].unique())

# Step 3:  DataFrames for Resort and City Hotels
resort = data[(data['hotel'] == 'Resort Hotel') & (data['is_canceled'] == 0)]
city = data[(data['hotel'] == 'City Hotel') & (data['is_canceled'] == 0)]

#  the shapes of the DataFrames
print("Resort DataFrame Shape:", resort.shape)
print("City DataFrame Shape:", city.shape)

# Step 4: labels and values for the pie chart
labels = resort['country'].value_counts().index  # Unique countries
values = resort['country'].value_counts().values  # Counts for each country

# Step 5: Creating the pie chart
trace = go.Pie(labels=labels, values=values, hoverinfo='label+percent',␣
 ↪textinfo='value')
```

```
Unique Hotels: ['Resort Hotel' 'City Hotel']
Unique Cancellation Status: [0 1]
Resort DataFrame Shape: (28927, 32)
City DataFrame Shape: (46084, 32)
```

[468]:
```
iplot([trace])
```

[469]:
```
data.columns
```

[469]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type', 'agent',
       'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date'],
      dtype='object')
```

[470]:
```
country_data = data[data['is_canceled'] == 0] ['country'].value_counts().
↪reset_index()
country_data.columns = ['country', 'no of guests']
country_data
```

[470]:

	country	no of guests
0	PRT	20977
1	GBR	9668
2	FRA	8468
3	ESP	6383
4	DEU	6067
..	…	…
161	BHR	1
162	DJI	1
163	MLI	1
164	NPL	1
165	FRO	1

[166 rows x 2 columns]

[471]:
```
!pip install folium
```

```
Requirement already satisfied: folium in c:\users\hassan
shoaib\anaconda3\lib\site-packages (0.17.0)
Requirement already satisfied: branca>=0.6.0 in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from folium) (0.8.0)
Requirement already satisfied: jinja2>=2.9 in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from folium) (3.1.2)
Requirement already satisfied: numpy in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from folium) (1.24.3)
Requirement already satisfied: requests in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from folium) (2.31.0)
```

```
Requirement already satisfied: xyzservices in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.1.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from requests->folium) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hassan
shoaib\anaconda3\lib\site-packages (from requests->folium) (2023.7.22)
```

[472]:
```
#map graph
import folium
from folium.plugins import HeatMap
```

[473]:
```
b = folium.Map()
```

[474]:
```
country_data.dtypes
```

[474]:
```
country         object
no of guests     int64
dtype: object
```

[475]:
```
guests = px.choropleth(country_data,
                       locations = country_data ['country'],
                       color = country_data['no of guests'],
                      hover_name = country_data['country'],
                      title = "Home Country Of Our Guests" )
```

[476]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Displaying Guest's Country Using Heat Map "))
```

9 Displaying Guest's Country Using Heat Map

[477]:
```
guests.show()
```

[478]:
```
data.head()
```

[478]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	\
0	Resort Hotel	0	342	2015	July	
1	Resort Hotel	0	737	2015	July	
2	Resort Hotel	0	7	2015	July	

```
3  Resort Hotel            0         13               2015               July
4  Resort Hotel            0         14               2015               July

   arrival_date_week_number  arrival_date_day_of_month  \
0                        27                          1
1                        27                          1
2                        27                          1
3                        27                          1
4                        27                          1

   stays_in_weekend_nights  stays_in_week_nights  adults  …  deposit_type  \
0                        0                     0       2  …    No Deposit
1                        0                     0       2  …    No Deposit
2                        0                     1       1  …    No Deposit
3                        0                     1       1  …    No Deposit
4                        0                     2       2  …    No Deposit

   agent company days_in_waiting_list customer_type   adr  \
0    0.0     0.0                    0     Transient   0.0
1    0.0     0.0                    0     Transient   0.0
2    0.0     0.0                    0     Transient  75.0
3  304.0     0.0                    0     Transient  75.0
4  240.0     0.0                    0     Transient  98.0

   required_car_parking_spaces  total_of_special_requests  reservation_status  \
0                            0                          0           Check-Out
1                            0                          0           Check-Out
2                            0                          0           Check-Out
3                            0                          0           Check-Out
4                            0                          1           Check-Out

  reservation_status_date
0              2015-07-01
1              2015-07-01
2              2015-07-02
3              2015-07-02
4              2015-07-03

[5 rows x 32 columns]
```

[479]:
```
#how much are guest paying per night
data2 = data[data['is_canceled'] == 0]
```

[480]:
```
data2
```

[480]:
```
               hotel  is_canceled  lead_time  arrival_date_year  \
0       Resort Hotel            0        342               2015
```

```
1       Resort Hotel            0        737               2015
2       Resort Hotel            0          7               2015
3       Resort Hotel            0         13               2015
4       Resort Hotel            0         14               2015
…              …          …         …                …
119385    City Hotel            0         23               2017
119386    City Hotel            0        102               2017
119387    City Hotel            0         34               2017
119388    City Hotel            0        109               2017
119389    City Hotel            0        205               2017

       arrival_date_month  arrival_date_week_number  \
0                    July                        27
1                    July                        27
2                    July                        27
3                    July                        27
4                    July                        27
…                     …                         …
119385             August                        35
119386             August                        35
119387             August                        35
119388             August                        35
119389             August                        35

        arrival_date_day_of_month  stays_in_weekend_nights  \
0                               1                       0
1                               1                       0
2                               1                       0
3                               1                       0
4                               1                       0
…                             …                       …
119385                         30                       2
119386                         31                       2
119387                         31                       2
119388                         31                       2
119389                         29                       2

        stays_in_week_nights  adults  …  deposit_type  agent company  \
0                          0       2  …    No Deposit    0.0     0.0
1                          0       2  …    No Deposit    0.0     0.0
2                          1       1  …    No Deposit    0.0     0.0
3                          1       1  …    No Deposit  304.0     0.0
4                          2       2  …    No Deposit  240.0     0.0
…                        …     …  …            …    …     …
119385                     5       2  …    No Deposit  394.0     0.0
119386                     5       3  …    No Deposit    9.0     0.0
119387                     5       2  …    No Deposit    9.0     0.0
```

```
119388                    5       2  …    No Deposit   89.0     0.0
119389                    7       2  …    No Deposit    9.0     0.0

       days_in_waiting_list customer_type     adr  \
0                         0     Transient    0.00
1                         0     Transient    0.00
2                         0     Transient   75.00
3                         0     Transient   75.00
4                         0     Transient   98.00
…                       …           …     …
119385                    0     Transient   96.14
119386                    0     Transient  225.43
119387                    0     Transient  157.71
119388                    0     Transient  104.40
119389                    0     Transient  151.20

        required_car_parking_spaces  total_of_special_requests  \
0                                 0                          0
1                                 0                          0
2                                 0                          0
3                                 0                          0
4                                 0                          1
…                               …                        …
119385                            0                          0
119386                            0                          2
119387                            0                          4
119388                            0                          0
119389                            0                          2

        reservation_status reservation_status_date
0                Check-Out              2015-07-01
1                Check-Out              2015-07-01
2                Check-Out              2015-07-02
3                Check-Out              2015-07-02
4                Check-Out              2015-07-03
…                      …                       …
119385           Check-Out              2017-09-06
119386           Check-Out              2017-09-07
119387           Check-Out              2017-09-07
119388           Check-Out              2017-09-07
119389           Check-Out              2017-09-07

[75011 rows x 32 columns]
```

[481]:
```
data2.columns
```

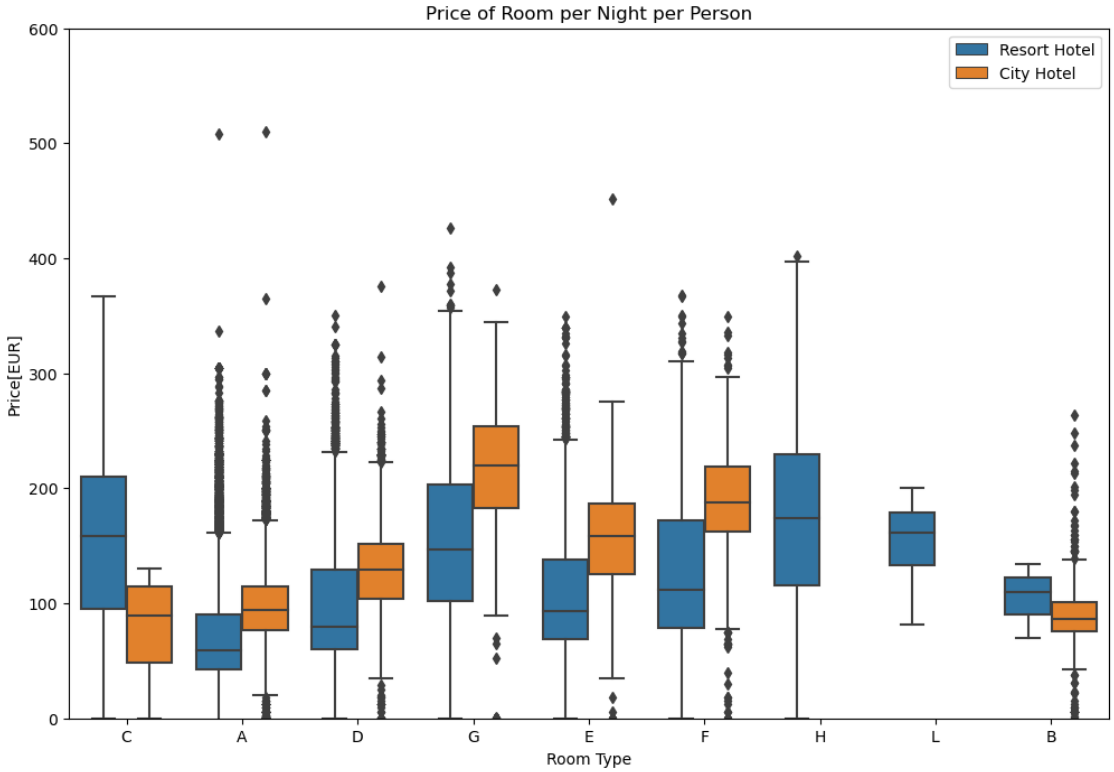
[481]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type', 'agent',
       'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date'],
      dtype='object')
```

[482]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Price Of Room Per NIght "))
```

10 Price Of Room Per NIght

[483]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[484]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x='reserved_room_type', y='adr', hue='hotel', data=data2)
plt.title("Price of Room per Night per Person")
plt.xlabel("Room Type")
plt.ylabel("Price[EUR]")
plt.legend(loc = 'upper right')
plt.ylim(0, 600)
plt.show()
```



[485]:
```
data_resort =[ resort['is_canceled'] == 0]
```

[486]:
```
data_city =[ city['is_canceled'] == 0]
```

[487]:
```
data_city
```

[487]:
```
[40060     True
 40066     True
 40070     True
 40071     True
 40072     True
           ...
 119385    True
 119386    True
 119387    True
 119388    True
 119389    True
 Name: is_canceled, Length: 46084, dtype: bool]
```

[488]:
```
data_resort
```

[488]:
```
[0        True
 1        True
 2        True
 3        True
 4        True
          …
 40055    True
 40056    True
 40057    True
 40058    True
 40059    True
 Name: is_canceled, Length: 28927, dtype: bool]
```

[489]:
```
data_resort = data[data['hotel'] == 'Resort Hotel']
```

[490]:
```
resort_hotel = data_resort.groupby(['arrival_date_month'])['adr'].mean().
 ↪reset_index()
resort_hotel
```

[490]:

	arrival_date_month	adr
0	April	77.849496
1	August	186.790574
2	December	69.051887
3	February	55.189716
4	January	49.507033
5	July	155.181299
6	June	110.481032
7	March	57.554652
8	May	78.758134
9	November	48.313643
10	October	62.132572
11	September	93.252030

[491]:
```
data_city = data[data['hotel'] == 'City Hotel']
```

[492]:
```
city_hotel = data_city.groupby(['arrival_date_month'])['adr'].mean().
 ↪reset_index()
city_hotel
```

[492]:

	arrival_date_month	adr
0	April	111.397415
1	August	114.857330
2	December	89.209560
3	February	85.327519
4	January	82.754477
5	July	110.945950
6	June	119.186056

```
7               March   92.973339
8                 May  121.764614
9            November   88.372486
10            October  100.119313
11          September  110.120296
```

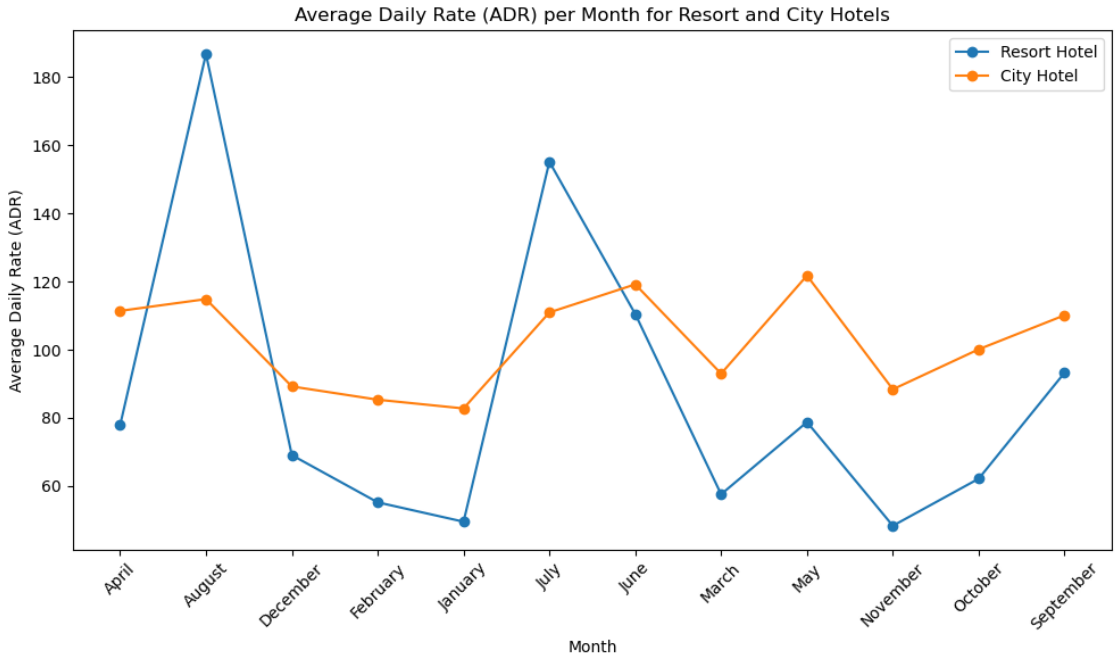
[493]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Grouping by hotel type, arrival month, and calculating the mean ADR
resort_adr = data_resort.groupby('arrival_date_month')['adr'].mean().
 ↪reset_index()
city_adr = data_city.groupby('arrival_date_month')['adr'].mean().reset_index()
```

[494]:
```
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
 ↪'August', 'September', 'October', 'November', 'December']

resort_adr['arrival_date_month'] = pd.
 ↪Categorical(resort_adr['arrival_date_month'], categories=months,␣
 ↪ordered=True)
city_adr['arrival_date_month'] = pd.Categorical(city_adr['arrival_date_month'],␣
 ↪categories=months, ordered=True)
```

[495]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("Displaying Booking Trends each Month Using Line Chart "))
```

Displaying Booking Trends each Month Using Line Chart

[496]:
```
# Plotting the line chart
plt.figure(figsize=(10, 6))
plt.plot(resort_adr['arrival_date_month'], resort_adr['adr'], label='Resort␣
 ↪Hotel', marker='o')
plt.plot(city_adr['arrival_date_month'], city_adr['adr'], label='City Hotel',␣
 ↪marker='o')
plt.title('Average Daily Rate (ADR) per Month for Resort and City Hotels')
plt.xlabel('Month')
plt.ylabel('Average Daily Rate (ADR)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



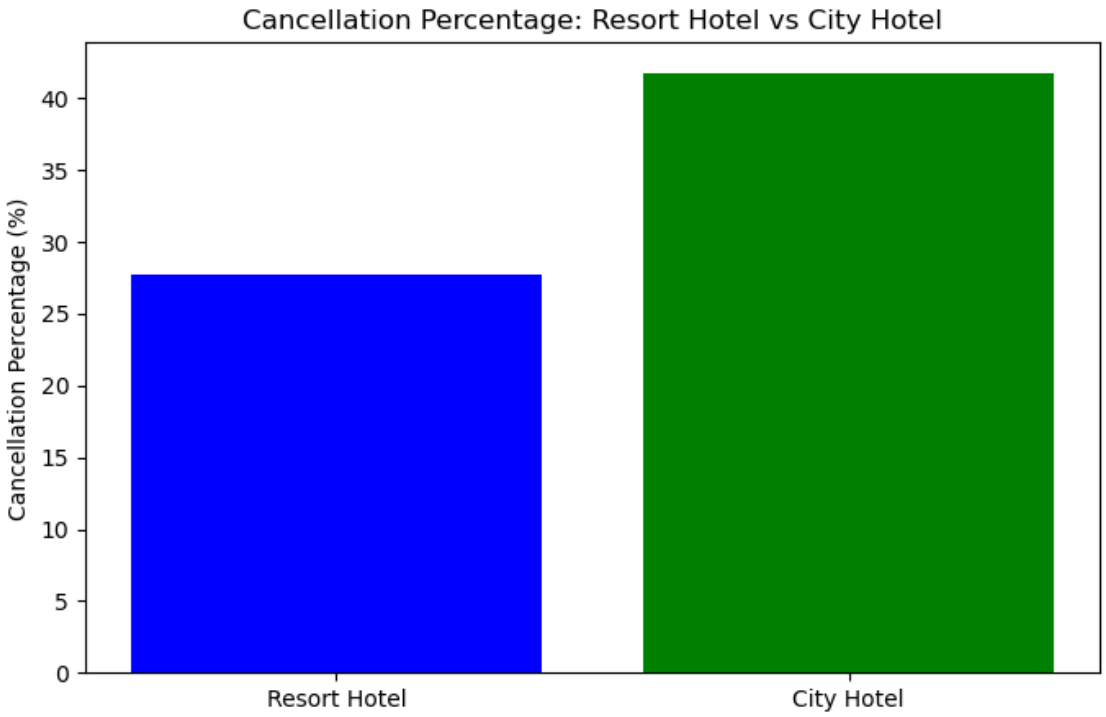
[497]:
```
#Goal: Compare cancellation percentages for City Hotel and Resort Hotel to see␣
 ↪which hotel faces more cancellations.
#Why it's important: High cancellation rates can cause financial losses.
#Understanding which hotel has a higher cancellation percentage helps the␣
 ↪management address the issue.
# Calculate cancellation percentage for each hotel
resort_cancel_rate = data_resort['is_canceled'].mean() * 100
city_cancel_rate = data_city['is_canceled'].mean() * 100
```

[498]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Calculating Cancellation Percentages for City and Resort␣
 ↪Hotels "))
```

11 Calculating Cancellation Percentages for City and Resort Hotels

[499]:
```
# Create a bar plot
plt.figure(figsize=(8, 5))
plt.bar(['Resort Hotel', 'City Hotel'], [resort_cancel_rate, city_cancel_rate],␣
 ↪color=['blue', 'green'])
plt.title('Cancellation Percentage: Resort Hotel vs City Hotel')
plt.ylabel('Cancellation Percentage (%)')
```

```
plt.show()
```



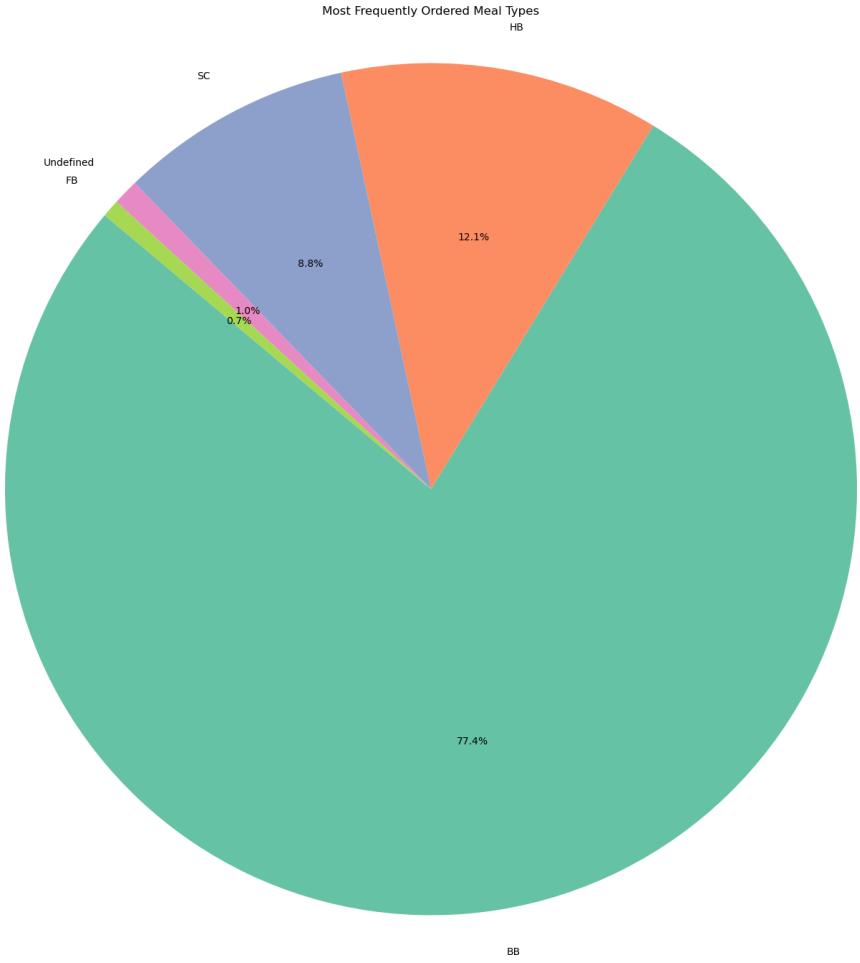
[500]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Identifying the Most Frequently Ordered Meal Types "))
```

12 Identifying the Most Frequently Ordered Meal Types

[501]:
```
#Goal: Identify which meal type is the most frequently ordered by guests.
#Why it's important: Understanding guest preferences can help hotels plan their␣
 ↪menu and services more efficiently,
# catering to popular demands.
# Counting occurrences of each meal type
meal_counts = data['meal'].value_counts()
```

[502]:
```
# Plotting a pie chart
plt.figure(figsize=(20, 17))
plt.pie(meal_counts, labels=meal_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140,  # Change startangle to rotate
colors=sns.color_palette('Set2'), labeldistance=1.1)
plt.title('Most Frequently Ordered Meal Types')
```

```
plt.axis('equal')
plt.show()
```

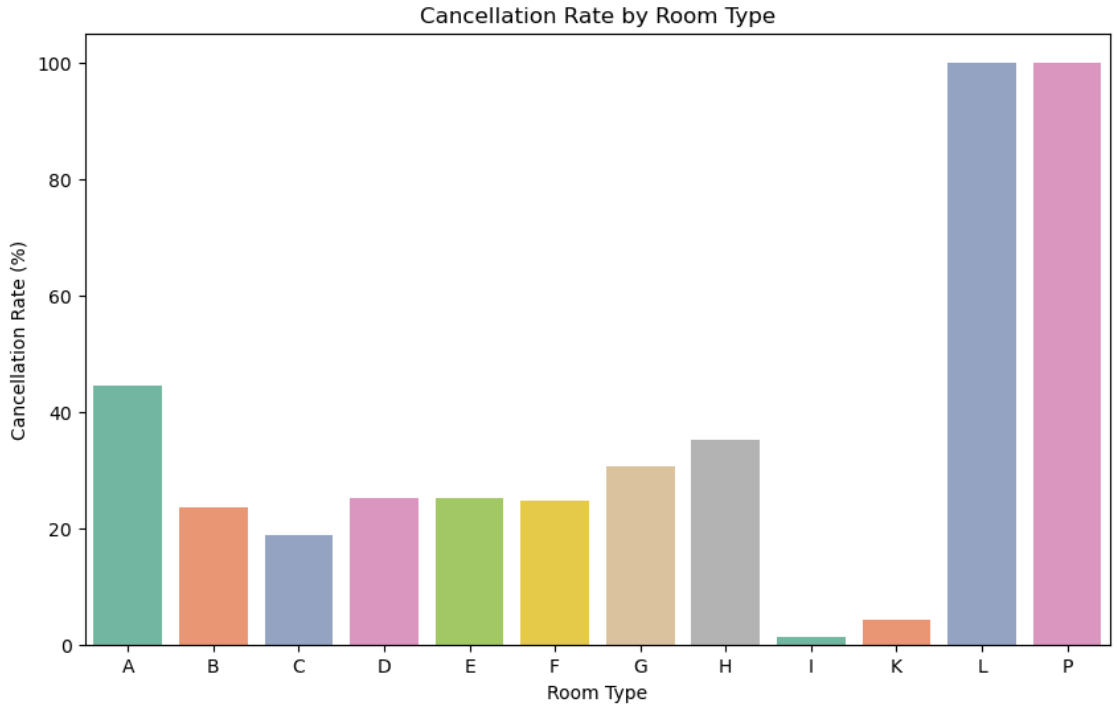


[503]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Exploring Correlations Between Room Types and Cancellations
↪"))
```

13 Exploring Correlations Between Room Types and Cancellations

[504]:
```
# Grouping by room type and cancellation status
room_cancellation = df.groupby('assigned_room_type')['is_canceled'].mean() *
↪100  # Assuming 'is_canceled' is 1 for canceled bookings
```

```
# Plotting the bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=room_cancellation.index, y=room_cancellation.values,␣
 ↪palette='Set2')
plt.title('Cancellation Rate by Room Type')
plt.xlabel('Room Type')
plt.ylabel('Cancellation Rate (%)')
plt.show()
```

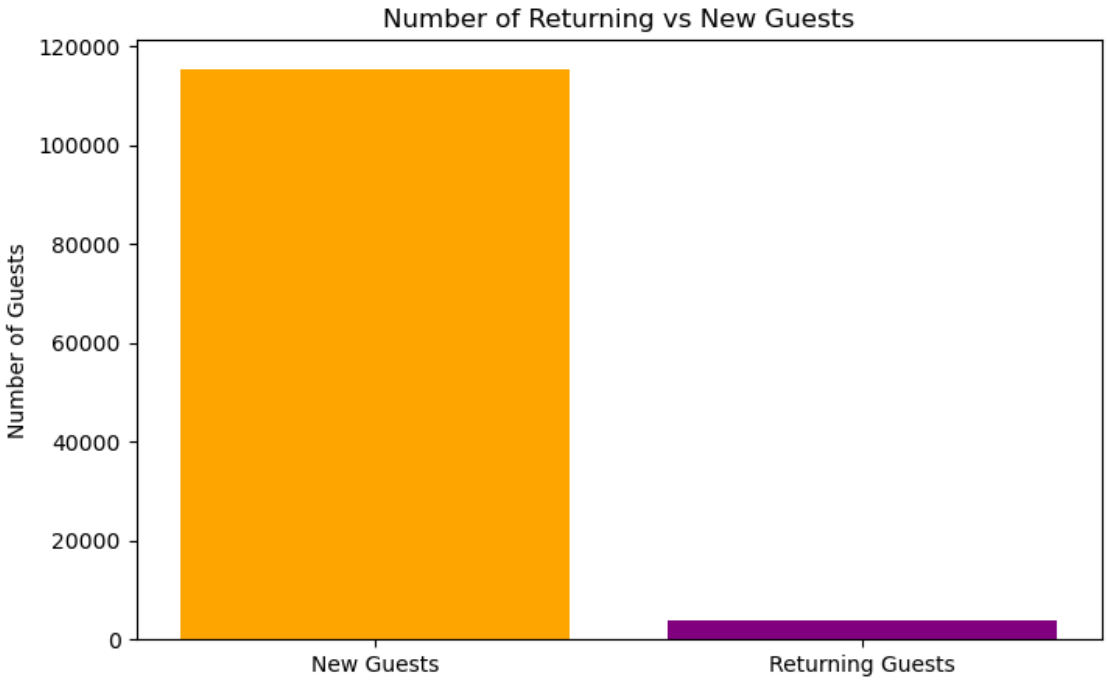


[505]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Discovering No. of returning Guest vs New Guest "))
```

14 Discovering No. of returning Guest vs New Guest

[506]:
```
# Counting returning vs new guests
returning_guests = data['is_repeated_guest'].value_counts()

# Plotting a bar chart
plt.figure(figsize=(8, 5))
plt.bar(['New Guests', 'Returning Guests'], returning_guests, color=['orange',␣
 ↪'purple'])
```

```
plt.title('Number of Returning vs New Guests')
plt.ylabel('Number of Guests')
plt.show()
```



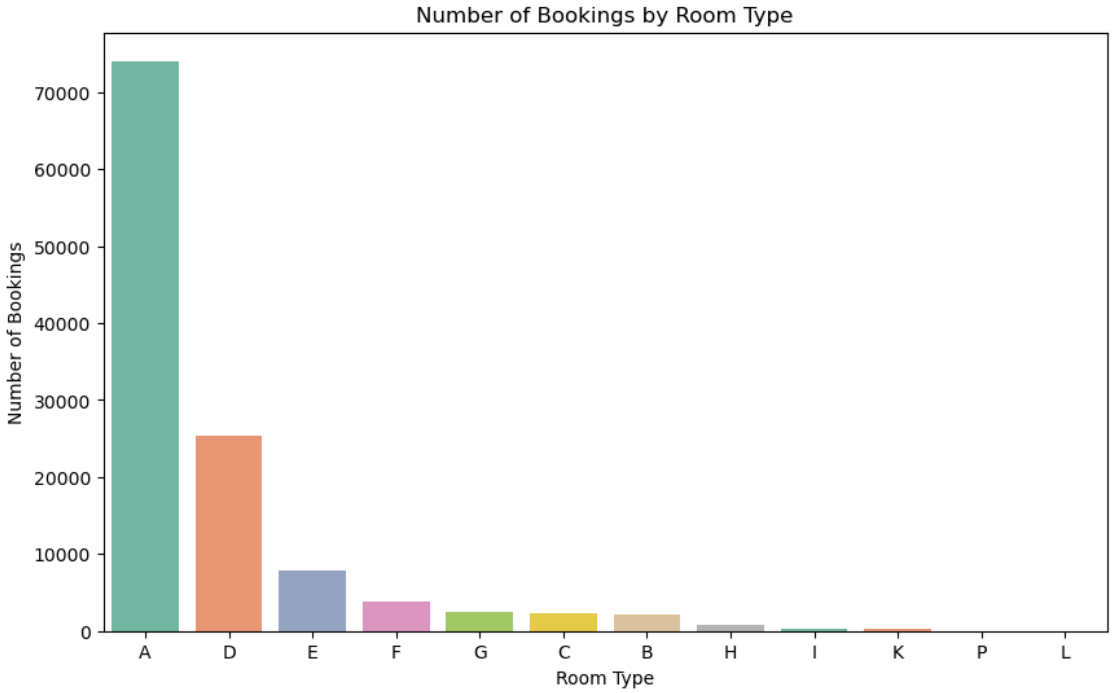
[507]:
```
# Grouping by 'assigned_room_type' and counting the number of occurrences
room_counts = df['assigned_room_type'].value_counts()
```

[508]:
```
# Displaying the most booked room type
most_booked_room_type = room_counts.idxmax()
most_booked_room_count = room_counts.max()
```

[509]:
```
print(f"The most booked room type is: {most_booked_room_type} with␣
↪{most_booked_room_count} bookings.")
```

```
The most booked room type is: A with 74053 bookings.
```

[510]:
```
# Optionally, you can visualize it using a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=room_counts.index, y=room_counts.values, palette='Set2')
plt.title('Number of Bookings by Room Type')
plt.xlabel('Room Type')
plt.ylabel('Number of Bookings')
plt.show()
```



[511]:
```
from IPython.display import display, Markdown

# Feature Engineering Section
feature_engineering_justification = """
# 3. Feature Engineering

## A. Encoding
- **One-Hot Encoding**:If these features were encoded using a numeric label (e.
 ↪g., assigning numbers 1, 2, 3 to different categories), the model might␣
 ↪mistakenly interpret these values as having a natural order, which could␣
 ↪negatively impact its predictions by introducing bias. One-hot encoding␣
 ↪addresses this issue by representing each category as a binary column,␣
 ↪ensuring that no false ordinal relationship is inferred and preserving the␣
 ↪correct nature of the categorical information. This allows the model to␣
 ↪effectively learn without assuming relationships that do not exist.

## B. Binning
- **Binning `lead_time` and `adr`**: `For lead_time, binning into 'short',␣
 ↪'medium', and 'long' helps capture different booking behaviors that might␣
 ↪correlate with cancellation rates (e.g., longer lead times might indicate␣
 ↪higher cancellation probabilities). Similarly, binning adr (average daily␣
 ↪rate) into 'Low', 'Medium', and 'High' can help to categorize guests into␣
 ↪different spending levels, which might reveal patterns in cancellations or␣
 ↪preferences.
```

```
## C. Scaling
- **Standard Scaling**: Scaling was applied to numerical features to␣
 ↪standardize them, which ensures that features with larger ranges do not␣
 ↪dominate the model. This is especially important for distance-based␣
 ↪algorithms or models sensitive to the scale of input features.

## D. Feature Selection
- **Selecting Relevant Features**: The selection of features like lead_time,␣
 ↪is_repeated_guest, previous_cancellations, booking_changes, adr, and encoded␣
 ↪categorical columns was driven by their importance in predicting whether a␣
 ↪booking would be canceled. This selection was made based on domain␣
 ↪knowledge, as these features have a logical relationship with customer␣
 ↪behavior and cancellations. For instance, lead_time can indicate the risk of␣
 ↪cancellation-longer times may increase the likelihood of change. By␣
 ↪selecting only the most informative features, the model's complexity is␣
 ↪reduced, which helps prevent overfitting and makes the model more efficient,␣
 ↪thereby enhancing its predictive accuracy.
"""
display(Markdown(feature_engineering_justification))
```

15 3. Feature Engineering

15.1 A. Encoding

- **One-Hot Encoding**:If these features were encoded using a numeric label (e.g., assigning numbers 1, 2, 3 to different categories), the model might mistakenly interpret these values as having a natural order, which could negatively impact its predictions by introducing bias. One-hot encoding addresses this issue by representing each category as a binary column, ensuring that no false ordinal relationship is inferred and preserving the correct nature of the categorical information. This allows the model to effectively learn without assuming relationships that do not exist.

15.2 B. Binning

- **Binning `lead_time` and `adr`**: 'For lead_time, binning into 'short', 'medium', and 'long' helps capture different booking behaviors that might correlate with cancellation rates (e.g., longer lead times might indicate higher cancellation probabilities). Similarly, binning adr (average daily rate) into 'Low', 'Medium', and 'High' can help to categorize guests into different spending levels, which might reveal patterns in cancellations or preferences.

15.3 C. Scaling

- **Standard Scaling**: Scaling was applied to numerical features to standardize them, which ensures that features with larger ranges do not dominate the model. This is especially important for distance-based algorithms or models sensitive to the scale of input features.

15.4 D. Feature Selection

- **Selecting Relevant Features**: The selection of features like lead_time, is_repeated_guest, previous_cancellations, booking_changes, adr, and encoded categorical columns was driven by their importance in predicting whether a booking would be canceled. This selection was made based on domain knowledge, as these features have a logical relationship with customer behavior and cancellations. For instance, lead_time can indicate the risk of cancellation—longer times may increase the likelihood of change. By selecting only the most informative features, the model's complexity is reduced, which helps prevent overfitting and makes the model more efficient, thereby enhancing its predictive accuracy.

[512]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Encoding "))
```

16 Encoding

[513]:
```
# Dropping irrelevant columns
df = df.drop(['company', 'agent'], axis=1)

# One-hot encoding categorical columns (e.g., meal type, hotel type)
df_encoded = pd.get_dummies(df, columns=['meal', 'hotel'])
```

[514]:
```
# Convert reservation status date to datetime format if not already
df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date'])

# Extract the month from the date
df['booking_month'] = df['reservation_status_date'].dt.month
```

[515]:
```
# Count bookings by month
most_booked_month = df['booking_month'].value_counts().idxmax()
print(f"The month with the most bookings is: {most_booked_month}")
```

```
The month with the most bookings is: 7
```

[516]:
```
room_type_columns = [col for col in df_encoded.columns if 'room_type' in col]
print(f"Encoded room type columns: {room_type_columns}")
```

```
Encoded room type columns: ['reserved_room_type', 'assigned_room_type']
```

[517]:
```
# Check if booking_month exists
if 'booking_month' not in df_encoded.columns:
    # If not, add it to df_encoded
    df_encoded['booking_month'] = df['reservation_status_date'].dt.month
```

[518]:
```
# Include the 'booking_month' as one of the relevant features
selected_features = ['lead_time', 'is_repeated_guest',␣
 ↪'previous_cancellations', 'booking_changes', 'adr', 'reserved_room_type',␣
 ↪'assigned_room_type', 'booking_month']

# Selecting the relevant features from the dataframe
X = df_encoded[selected_features]

#Including the month in which the most bookings occurred can help identify␣
 ↪seasonal trends in booking cancellations, as certain months may have higher␣
 ↪cancellation rates due to holidays or peak seasons. It adds context to the␣
 ↪time element of the booking behavior.
```

[519]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Binning "))
```

17 Binning

[520]:
```
# Binning lead_time into categories (short, medium, long)
bins = [0, 30, 90, 300]
labels = ['short', 'medium', 'long']
df_encoded['lead_time_binned'] = pd.cut(df_encoded['lead_time'], bins=bins,␣
 ↪labels=labels)
```

[521]:
```
# One-hot encoding categorical columns ('reserved_room_type',␣
 ↪'assigned_room_type', and any other categorical features)
df_encoded = pd.get_dummies(df_encoded, columns=['reserved_room_type',␣
 ↪'assigned_room_type', 'booking_month'], drop_first=True)
```

[522]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Feature Selection "))
```

18 Feature Selection

[523]:
```
# Now select the relevant features including the encoded ones
selected_features = ['lead_time', 'is_repeated_guest',␣
 ↪'previous_cancellations', 'booking_changes', 'adr'] + list(df_encoded.
 ↪columns[df_encoded.columns.str.startswith('reserved_room_type_')]) +␣
 ↪list(df_encoded.columns[df_encoded.columns.str.
 ↪startswith('assigned_room_type_')]) + list(df_encoded.columns[df_encoded.
 ↪columns.str.startswith('booking_month_')])
```

```
X = df_encoded[selected_features]
```

[524]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Scaling "))
```

19 Scaling

[525]:
```
from sklearn.preprocessing import StandardScaler

# Scaling the features after encoding
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[526]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Classifier Training Started from Here"))
```

20 Classifier Training Started from Here

[527]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Data Spliting"))
```

21 Data Spliting

[528]:
```
# Define the target variable (is_canceled column)
y = df_encoded['is_canceled']

# Now split the data
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)
```

[529]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Model Training"))
```

22 Model Training

[530]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)
```

[531]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Model Evaluation"))
```

23 Model Evaluation

[532]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Model evaluation
y_pred = rf_model.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy:', accuracy_score(y_test, y_pred))
```

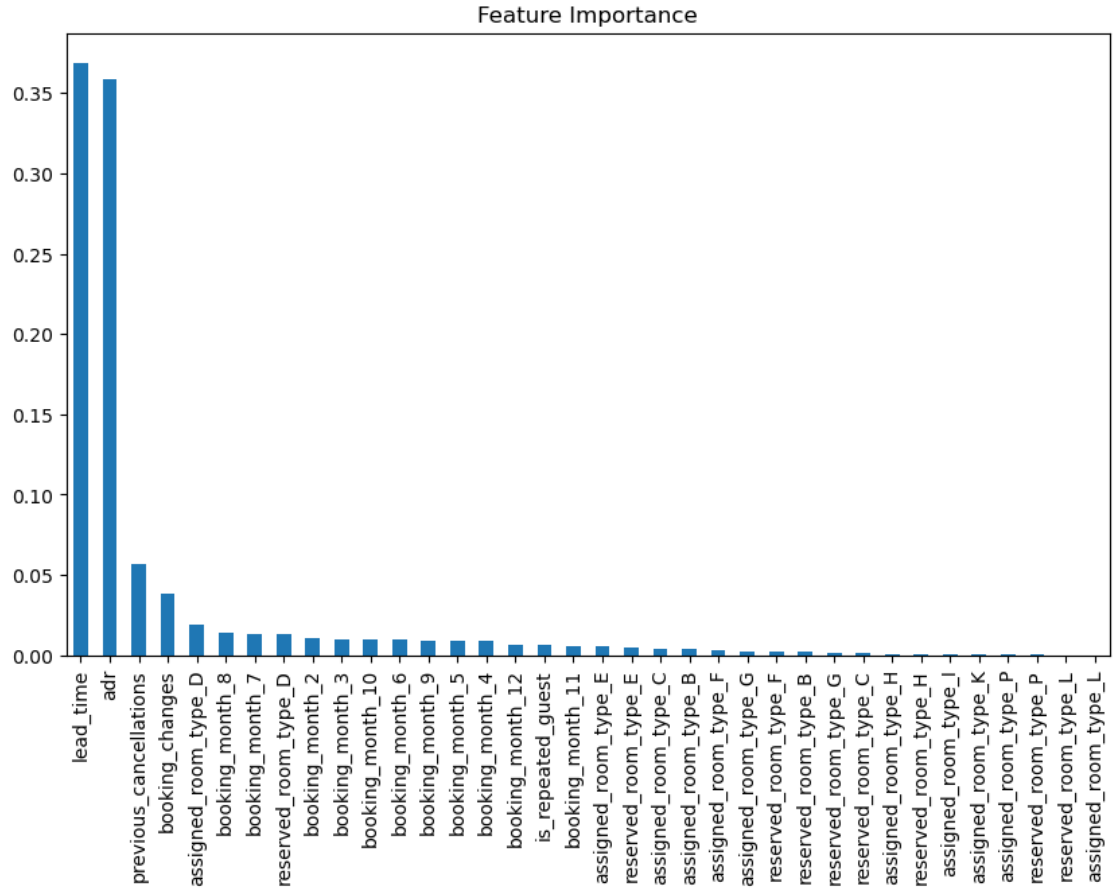
```
Accuracy: 0.8645615210654158
```

[533]:
```
from IPython.display import display, Markdown

# Display the heading
display(Markdown("# Feature Importance"))
```

24 Feature Importance

[534]:
```
# Feature importance from Random Forest
importances = rf_model.feature_importances_
features = selected_features

# Creating a bar plot for feature importance
import pandas as pd
import matplotlib.pyplot as plt

# Sort feature importance values
feature_importance = pd.Series(importances, index=features).
 ↪sort_values(ascending=False)

# Plot the feature importance
plt.figure(figsize=(10, 6))
```

```
feature_importance.plot(kind='bar')
plt.title('Feature Importance')
plt.show()
```



[ ]: