Importing necessary libraries:

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Dataset:

In [2]:
```
proj1=pd.read_csv("Los_Angeles_International_Airport_-_Passenger_Traffic_By_
proj1
```

Out[2]:

	DataExtractDate	ReportPeriod	Terminal	Arrival_Departure	Domestic_Internati
0	05/10/2021 06:01:09 AM	04/01/2021 12:00:00 AM	T1	Departure	Dome
1	05/03/2021 03:08:02 PM	03/01/2021 12:00:00 AM	T5	Departure	Dome
2	05/27/2021 03:16:34 PM	04/01/2021 12:00:00 AM	T5	Departure	Dome
3	07/10/2021 06:01:27 AM	06/01/2021 12:00:00 AM	T6	Arrival	Internati
4	05/10/2021 06:01:09 AM	04/01/2021 12:00:00 AM	T8	Arrival	Dome
...	...	...	...	...	
7878	11/18/2023 05:28:20 AM	10/01/2023 12:00:00 AM	TBIT	Arrival	Internati
7879	11/14/2023 05:28:19 AM	10/01/2023 12:00:00 AM	TBIT West Gates	Arrival	Internati
7880	11/15/2023 05:28:26 AM	10/01/2023 12:00:00 AM	TBIT	Departure	Dome
7881	11/15/2023 05:28:26 AM	10/01/2023 12:00:00 AM	T2	Arrival	Dome
7882	11/15/2023 05:28:26 AM	10/01/2023 12:00:00 AM	T4	Arrival	Internati

7883 rows × 6 columns

Exploring the data:

In [3]:
```
proj1.head()
```

Out[3]:

	DataExtractDate	ReportPeriod	Terminal	Arrival_Departure	Domestic_International
0	05/10/2021 06:01:09 AM	04/01/2021 12:00:00 AM	T1	Departure	Domestic
1	05/03/2021 03:08:02 PM	03/01/2021 12:00:00 AM	T5	Departure	Domestic
2	05/27/2021 03:16:34 PM	04/01/2021 12:00:00 AM	T5	Departure	Domestic
3	07/10/2021 06:01:27 AM	06/01/2021 12:00:00 AM	T6	Arrival	International
4	05/10/2021 06:01:09 AM	04/01/2021 12:00:00 AM	T8	Arrival	Domestic

In [4]:
```
proj1.dtypes
```

Out[4]:
```
DataExtractDate           object
ReportPeriod              object
Terminal                  object
Arrival_Departure         object
Domestic_International    object
Passenger_Count            int64
dtype: object
```

In [5]:
```
proj1.describe()
```

Out[5]:

	Passenger_Count
count	7883.000000
mean	150115.125079
std	152305.478195
min	0.000000
25%	19849.500000
50%	97231.000000
75%	259746.500000
max	908951.000000

Data Cleaning:

Checking for missing values:

In [6]:
```
proj1.isnull().sum()
```

Out[6]:
```
DataExtractDate           0
ReportPeriod              0
Terminal                  0
Arrival_Departure         0
Domestic_International    0
Passenger_Count           0
dtype: int64
```

Converting to date and time:

In [7]:
```
proj1['DataExtractDate'] = pd.to_datetime(proj1['DataExtractDate'])

# Create new columns for date and time
proj1['EXDate'] = proj1['DataExtractDate'].dt.date
proj1['EXTime'] = proj1['DataExtractDate'].dt.time
```

In [8]:
```
proj1['ReportPeriod'] = pd.to_datetime(proj1['ReportPeriod'], format='%m/%d/

# Create a new column for date
proj1['RPTDate'] = proj1['ReportPeriod'].dt.date
```

In [9]:
```
proj1.head()
```

Out[9]:

	DataExtractDate	ReportPeriod	Terminal	Arrival_Departure	Domestic_International
0	2021-05-10 06:01:09	2021-04-01	T1	Departure	Domestic
1	2021-05-03 15:08:02	2021-03-01	T5	Departure	Domestic
2	2021-05-27 15:16:34	2021-04-01	T5	Departure	Domestic
3	2021-07-10 06:01:27	2021-06-01	T6	Arrival	International
4	2021-05-10 06:01:09	2021-04-01	T8	Arrival	Domestic

Converting columns to appropriate data types:

In [10]:
```
proj1['Domestic_International'] = proj1['Domestic_International'].astype('ca
proj1['Arrival_Departure'] = proj1['Arrival_Departure'].astype('category')
proj1['Terminal'] = proj1['Terminal'].astype('str')
```

In [11]:
```
proj1['Passenger_Count'] = pd.to_numeric(proj1['Passenger_Count'], errors='c
proj1 = proj1.loc[~proj1['Terminal'].isin(['Imperial Terminal', 'Miscellaneo
proj1.loc[proj1['Terminal'] == 'TBIT West Gates', 'Terminal'] = 'TBIT_WG'
```

In [12]:
```
proj1
```

Out[12]:

	DataExtractDate	ReportPeriod	Terminal	Arrival_Departure	Domestic_Internati
0	2021-05-10 06:01:09	2021-04-01	T1	Departure	Dome
1	2021-05-03 15:08:02	2021-03-01	T5	Departure	Dome
2	2021-05-27 15:16:34	2021-04-01	T5	Departure	Dome
3	2021-07-10 06:01:27	2021-06-01	T6	Arrival	Internati
4	2021-05-10 06:01:09	2021-04-01	T8	Arrival	Dome
...	...	...	...	...	
7878	2023-11-18 05:28:20	2023-10-01	TBIT	Arrival	Internati
7879	2023-11-14 05:28:19	2023-10-01	TBIT_WG	Arrival	Internati
7880	2023-11-15 05:28:26	2023-10-01	TBIT	Departure	Dome
7881	2023-11-15 05:28:26	2023-10-01	T2	Arrival	Dome
7882	2023-11-15 05:28:26	2023-10-01	T4	Arrival	Internati

7097 rows × 9 columns

Dropping the missing values and duplicates:

In [13]:
```
proj1 = proj1.dropna()
proj1 = proj1.drop_duplicates()
```

In [14]:
```
proj1.head()
```

Out[14]:

	DataExtractDate	ReportPeriod	Terminal	Arrival_Departure	Domestic_International
0	2021-05-10 06:01:09	2021-04-01	T1	Departure	Domestic
1	2021-05-03 15:08:02	2021-03-01	T5	Departure	Domestic
2	2021-05-27 15:16:34	2021-04-01	T5	Departure	Domestic
3	2021-07-10 06:01:27	2021-06-01	T6	Arrival	International
4	2021-05-10 06:01:09	2021-04-01	T8	Arrival	Domestic

In [15]:
```
proj1.columns
```

Out[15]:
```
Index(['DataExtractDate', 'ReportPeriod', 'Terminal', 'Arrival_Departure',
       'Domestic_International', 'Passenger_Count', 'EXDate', 'EXTime',
       'RPTDate'],
      dtype='object')
```

In [16]:
```
average_passenger_count = proj1.groupby(['Arrival_Departure', 'Domestic_Inte

average_counts_df = pd.pivot_table(average_passenger_count, values='Passenge
                                   index=['Terminal'],
                                   columns=['Arrival_Departure', 'Domestic_I
                                   aggfunc='mean', fill_value=0)

average_counts_df.columns = ['_'.join(col).strip() for col in average_counts

average_counts_df
```

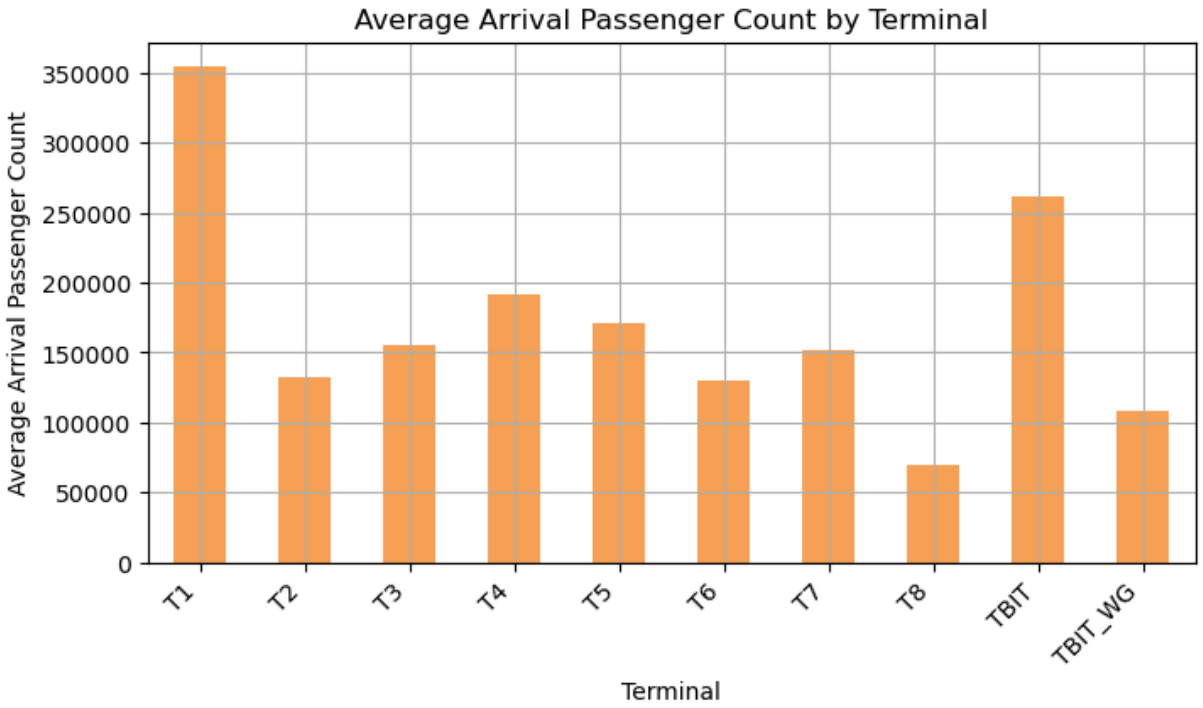
Out[16]:

Terminal	Arrival_Domestic	Arrival_International	Departure_Domestic	Departure_Inter
T1	354044.378505	0.000000	352428.593458	329
T2	132597.121495	131873.090909	131786.630841	13735
T3	231852.354167	17026.396226	226932.729167	3503
T4	344668.018692	29768.300493	336670.509346	4288
T5	290258.443925	28740.522472	296076.771028	26274
T6	232399.327103	24997.601896	228366.733645	5140
T7	254699.471963	48606.201878	260751.584112	36930
T8	125758.380282	3570.039326	124034.727700	7050
TBIT	10769.109827	463822.528037	27219.680851	400900
TBIT_WG	65220.900000	152050.766667	63863.766667	144005

Exploratory Data analysis (EDA):

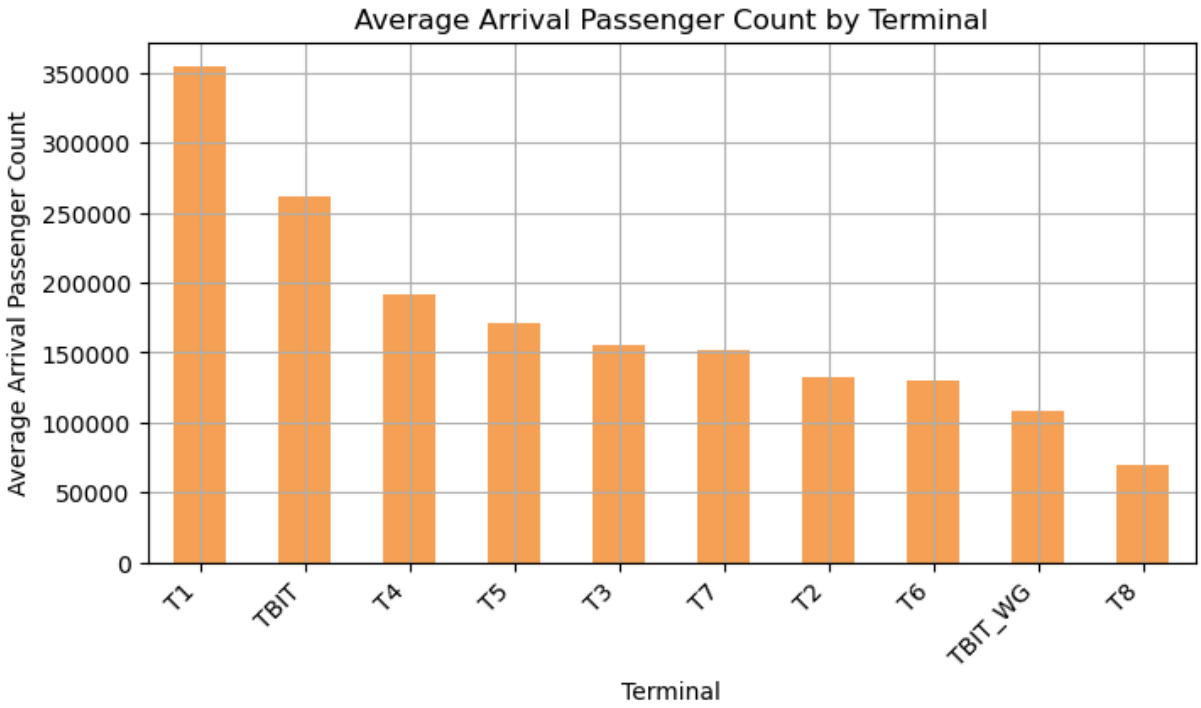
1. Average Arrival Passenger Count by Terminal:

In [17]:
```
terminal_order = ['T1', 'T2', 'T3', 'T4', 'T5', 'T6', 'T7', 'T8', 'TBIT', 'T
average_arrival_counts = proj1[proj1['Arrival_Departure'] == 'Arrival'].grou

fig, ax = plt.subplots(figsize=(8, 4))

average_arrival_counts.plot(kind='bar', ax=ax, color='#f5a455')

ax.set_xlabel('Terminal')
ax.set_ylabel('Average Arrival Passenger Count')
ax.set_title('Average Arrival Passenger Count by Terminal')
plt.grid(True)
plt.xticks(rotation=45, ha='right')
plt.show()
```



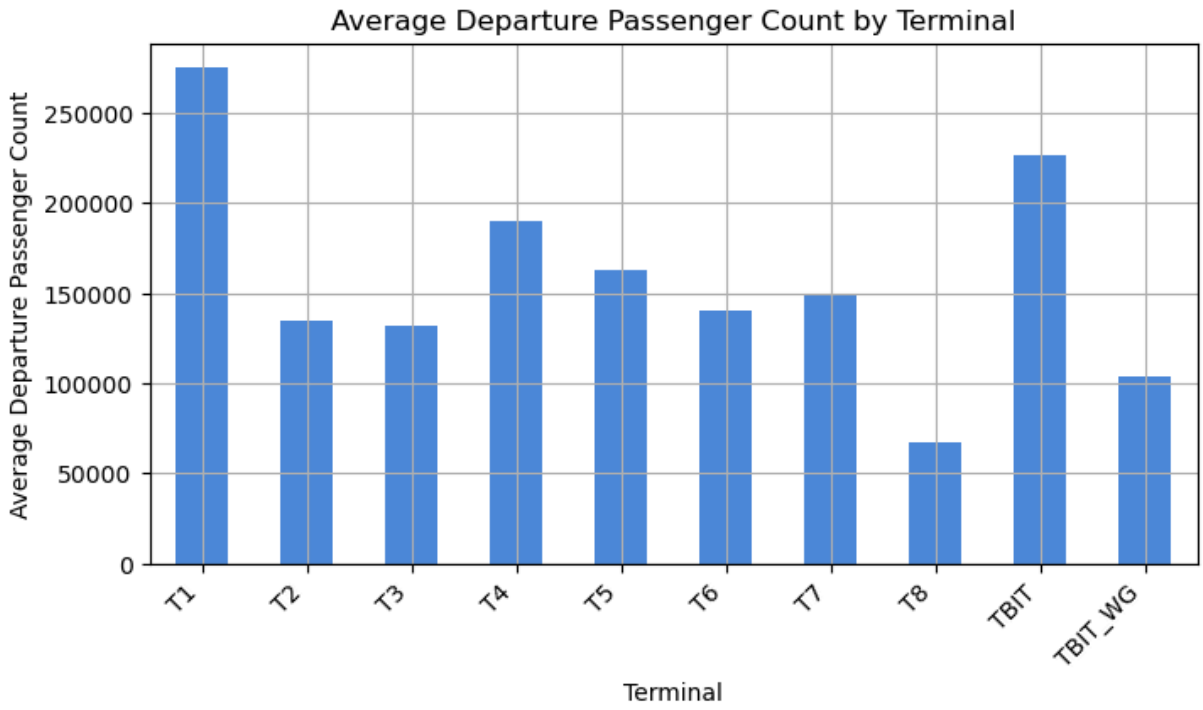
In [18]:
```
average_arrival_counts = proj1[proj1['Arrival_Departure'] == 'Arrival'].grou

average_arrival_counts_sorted = average_arrival_counts.sort_values(ascending

fig, ax = plt.subplots(figsize=(8, 4))

average_arrival_counts_sorted.plot(kind='bar', ax=ax, color='#f5a455')

ax.set_xlabel('Terminal')
ax.set_ylabel('Average Arrival Passenger Count')
ax.set_title('Average Arrival Passenger Count by Terminal')
plt.grid(True)
```

```
plt.xticks(rotation=45, ha='right')
plt.show()
```



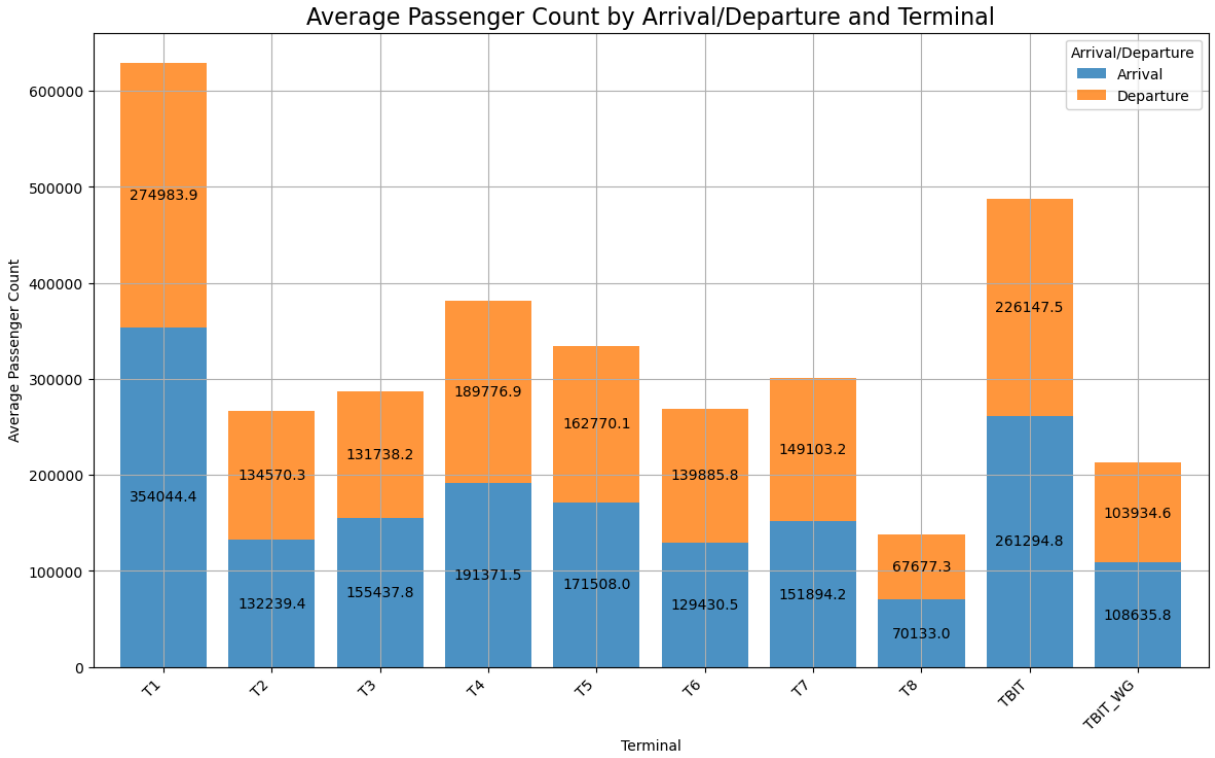
2. Average Departure Passenger Count by Terminal:

In [19]:
```
average_departure_counts = proj1[proj1['Arrival_Departure'] == 'Departure'].

fig, ax = plt.subplots(figsize=(8, 4))

average_departure_counts.plot(kind='bar', ax=ax, color='#4b89d9')

ax.set_xlabel('Terminal')
ax.set_ylabel('Average Departure Passenger Count')
ax.set_title('Average Departure Passenger Count by Terminal')

plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.show()
```



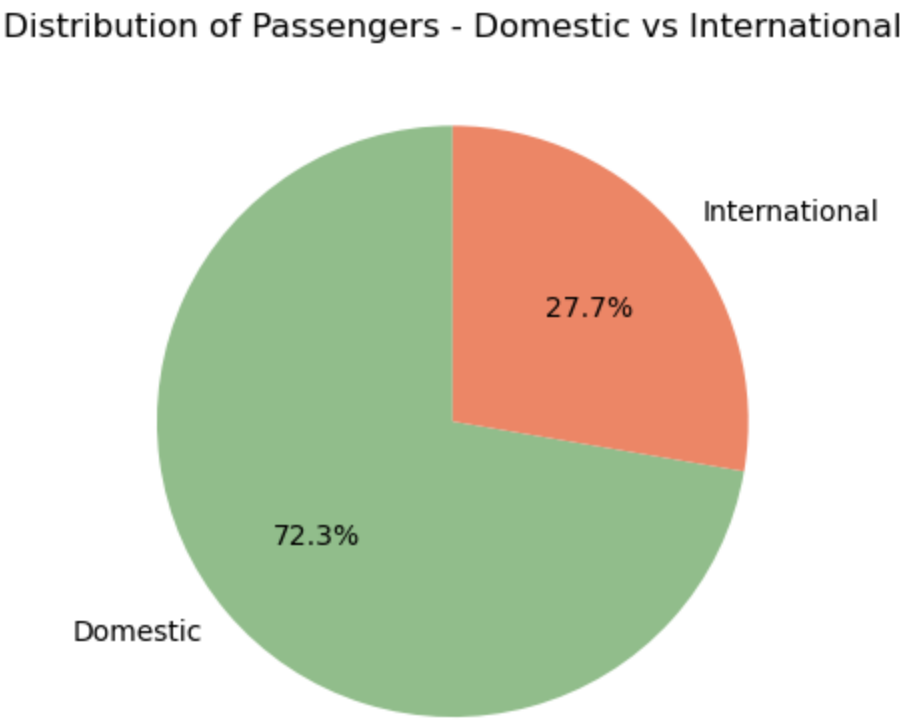
3. Average Counts Based on Arrival/Departure and Terminal:

In [20]:
```
average_counts = proj1.groupby(['Terminal', 'Arrival_Departure'])['Passenger

fig, ax = plt.subplots(figsize=(14, 8))
bars = average_counts.plot(kind='bar', stacked=True, width=0.8, ax=ax, alpha

ax.set_xlabel('Terminal')
ax.set_ylabel('Average Passenger Count')
ax.set_title('Average Passenger Count by Arrival/Departure and Terminal',for
ax.legend(title='Arrival/Departure')

for bar in bars.patches:
    width, height = bar.get_width(), bar.get_height()
    x, y = bar.get_xy()
    ax.annotate(f'{height:.1f}', (x + width/2, y + height/2), ha='center', v

plt.xticks(rotation=45, ha='right')
plt.grid(True)

plt.show()
```



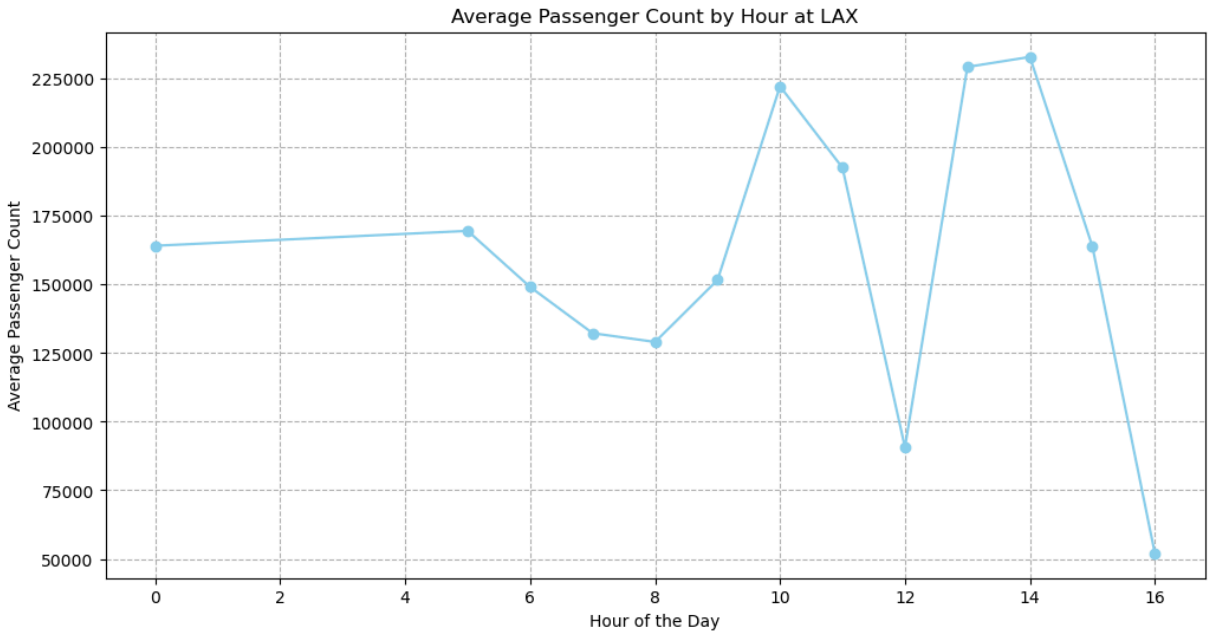
4. Distribution of Passengers between Domestic and International Flights:

In [21]:
```
passenger_distribution = proj1.groupby('Domestic_International')['Passenger_

fig, ax = plt.subplots()

ax.pie(passenger_distribution, labels=passenger_distribution.index, autopct=

ax.set_title('Distribution of Passengers - Domestic vs International')

plt.show()
```



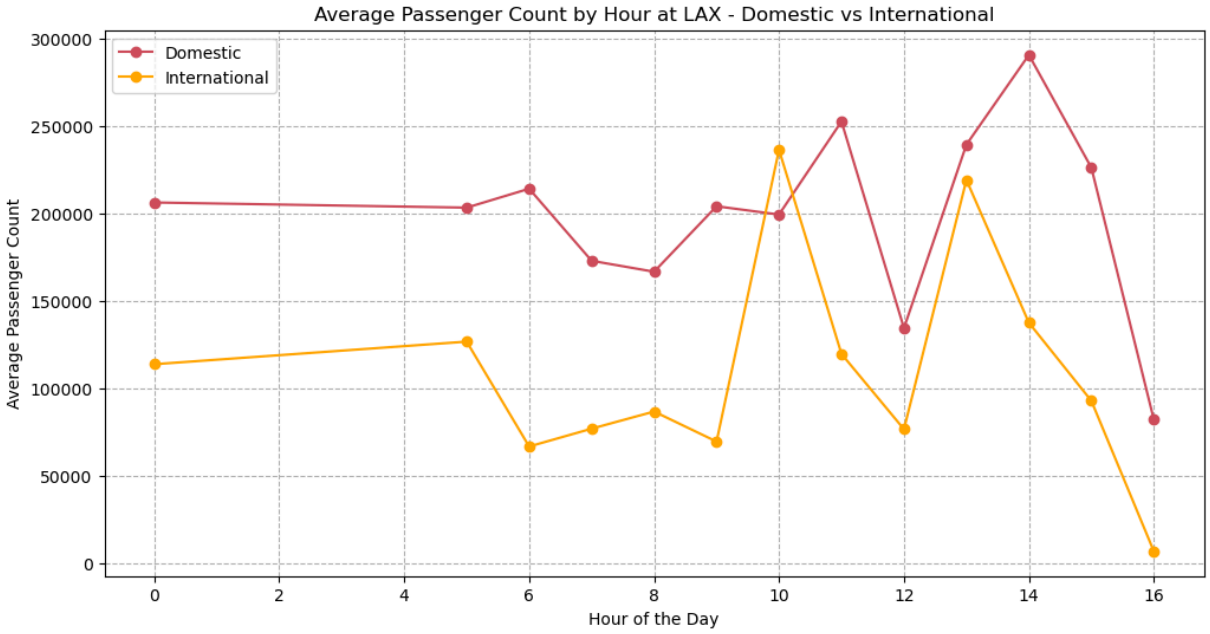
5. Overall Peak times at LAX:

In [22]:
```
proj1['Hour'] = proj1['DataExtractDate'].dt.hour

average_hourly_counts = proj1.groupby('Hour')['Passenger_Count'].mean()

plt.figure(figsize=(12, 6))

average_hourly_counts.plot(marker='o', linestyle='-', color='skyblue')

plt.title('Average Passenger Count by Hour at LAX')
plt.xlabel('Hour of the Day')
plt.ylabel('Average Passenger Count')
plt.grid(True, linestyle = '--')

plt.show()
```



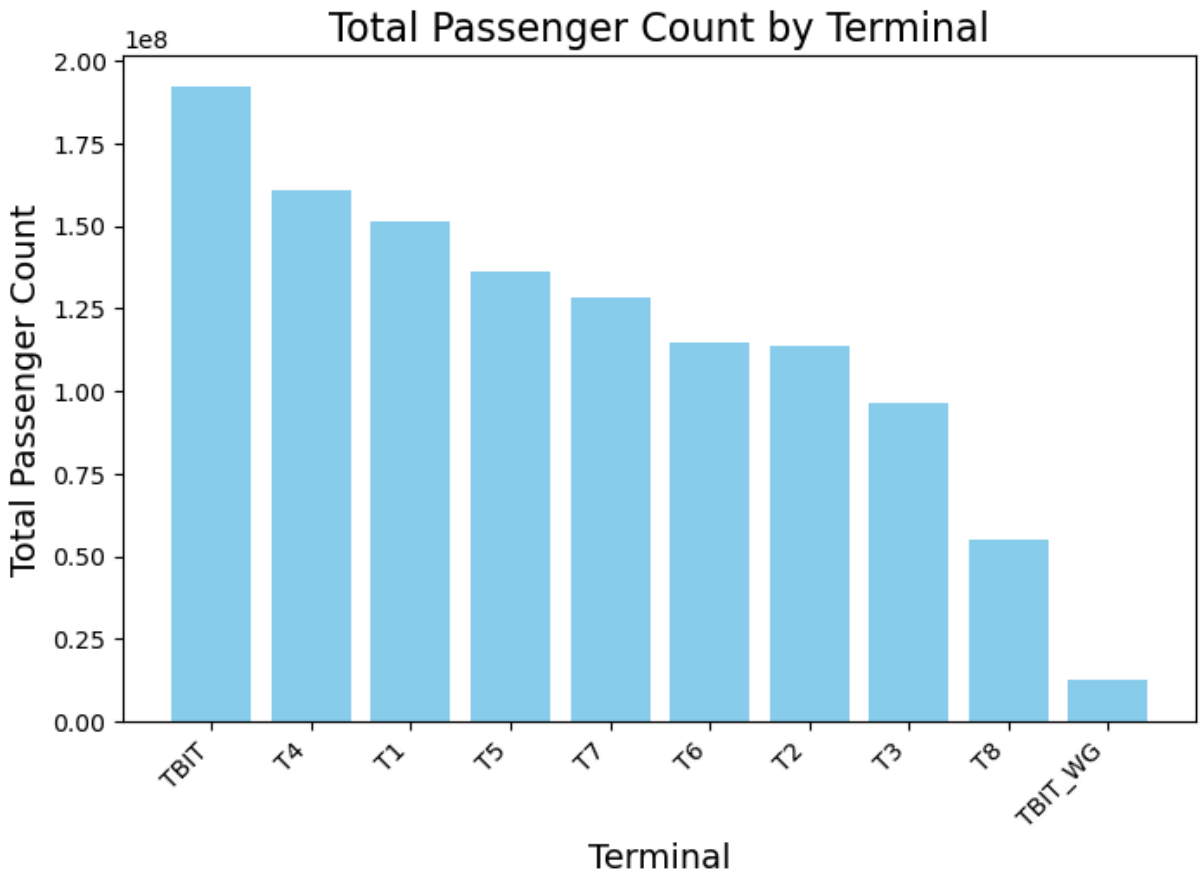
6. Peak times for domestic and international flights:

In [23]:
```
plt.figure(figsize=(12, 6))
domestic_data = proj1[proj1['Domestic_International'] == 'Domestic']
domestic_hourly_counts = domestic_data.groupby('Hour')['Passenger_Count'].me
domestic_hourly_counts.plot(marker='o', linestyle='-', color='#ce4b5a', labe

international_data = proj1[proj1['Domestic_International'] == 'International
international_hourly_counts = international_data.groupby('Hour')['Passenger_
international_hourly_counts.plot(marker='o', linestyle='-', color='orange',

plt.title('Average Passenger Count by Hour at LAX - Domestic vs Internationa
plt.xlabel('Hour of the Day')
plt.ylabel('Average Passenger Count')

plt.legend()
plt.grid(True, linestyle = '--')

plt.show()
```



7. Total Passenger Count by Terminal with Busiest Terminal Highlighted:

In [24]:
```
terminal_total_passengers = proj1.groupby('Terminal')['Passenger_Count'].sum

terminal_total_passengers_sorted = terminal_total_passengers.sort_values(by=

busiest_terminal = terminal_total_passengers_sorted.iloc[0]['Terminal']

fig, ax = plt.subplots(figsize=(8, 5))
ax.bar(terminal_total_passengers_sorted['Terminal'], terminal_total_passenge

ax.set_title('Total Passenger Count by Terminal',fontsize = 16)
ax.set_xlabel('Terminal',fontsize = 14)
ax.set_ylabel('Total Passenger Count',fontsize = 14)

ax.annotate(f'Busiest Terminal: {busiest_terminal}', xy=(2.5, 0), xytext=(10
            textcoords='offset points', ha='center', va='top', fontsize=12,

plt.xticks(rotation=45, ha='right')

plt.show()
```



Busiest Terminal: TBIT

In [25]:
```
proj1.columns
```

Out[25]:
```
Index(['DataExtractDate', 'ReportPeriod', 'Terminal', 'Arrival_Departure',
       'Domestic_International', 'Passenger_Count', 'EXDate', 'EXTime',
       'RPTDate', 'Hour'],
      dtype='object')
```

Statistical Analysis:

Chi-square test to assess the independence of the Terminal and Arrival/Departure:

In [26]:
```
from scipy.stats import chi2_contingency

terminal_arrival_contingency = pd.crosstab(proj1['Terminal'], proj1['Arrival
chi2_stat, p_value, _, _ = chi2_contingency(terminal_arrival_contingency)

if p_value < 0.05:
    print("The Terminal and Arrival/Departure are not independent.")
else:
    print("There is no significant relationship between Terminal and Arrival
```

```
There is no significant relationship between Terminal and Arrival/Departure
categories.
```

ANOVA for Terminal Comparison:

In [27]:
```
from scipy.stats import f_oneway

terminals = proj1['Terminal'].unique()
terminal_groups = [proj1[proj1['Terminal'] == terminal]['Passenger_Count'] f

f_stat, p_value = f_oneway(*terminal_groups)

if p_value < 0.05:
    print("There are significant differences in passenger counts among the t
else:
    print("There are no significant differences in passenger counts among th
```

```
There are significant differences in passenger counts among the terminals.
```

In [ ]:

In [ ]: