In [1]:
```
import mysql.connector
from mysql.connector import Error
```

In [2]:
```
db_name = 'PARKEASY'
db_host = 'localhost'
db_username = 'root'
```

In [3]:
```
import pymysql
from getpass import getpass
def sql_connection():
    try:
        connection = pymysql.connect(
            host=db_host,
            port=int(3306),
            user=db_username,
            password=getpass('Enter password: '),
            db=db_name
        )
        if connection:
            print("Database connected successfully")
            return connection
        else:
            print("Not connected")
    except Exception as e:
        print(e)
```

In [4]:
```
conn = sql_connection()
conn
```

```
Enter password: ········
Database connected successfully
```
Out[4]:
```
<pymysql.connections.Connection at 0x10650ba90>
```

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [16]:
```
parking_lot_df = pd.read_sql_query("Select * from ParkingLot", conn)
parking_lot_df
```

Out[16]:

	LotID	Name	Location	Capacity
0	1111	Downtown Parking	Main Street	150
1	1112	Central Plaza Parking	Center Avenue	200
2	1113	Greenfield Park Parking	Park Street	100
3	1114	Metro Mall Parking	Shopping Mall	250
4	1115	Tech Hub Parking	Innovation Avenue	120
5	1116	Riverside Parking	Riverfront Drive	180

In [17]:
```
parking_slot_df = pd.read_sql_query("Select * from ParkingSlot", conn)
parking_slot_df
```

Out[17]:

	SlotID	Status	Price	Type
0	10001	Available	10.0	Standard
1	10002	Available	12.5	Premium
2	10003	Occupied	10.0	Standard
3	10004	Occupied	12.5	Premium
4	10005	Available	10.0	Standard
5	10006	Available	12.5	Premium
6	10007	Occupied	10.0	Standard
7	10008	Available	12.5	Premium
8	10009	Occupied	10.0	Standard
9	10010	Occupied	12.5	Premium
10	10201	Available	10.0	Standard
11	10202	Occupied	12.5	Premium
12	10203	Available	10.0	Standard
13	10204	Available	12.5	Premium
14	10205	Occupied	10.0	Standard
15	10206	Available	12.5	Premium
16	10207	Occupied	10.0	Standard
17	10208	Available	12.5	Premium
18	10209	Available	10.0	Standard
19	10210	Occupied	12.5	Premium
20	10401	Occupied	10.0	Standard
21	10402	Available	12.5	Premium
22	10403	Occupied	10.0	Standard
23	10404	Available	12.5	Premium
24	10405	Occupied	10.0	Standard
25	10406	Occupied	12.5	Premium
26	10407	Available	10.0	Standard
27	10408	Occupied	12.5	Premium
28	10409	Available	10.0	Standard
29	10410	Occupied	12.5	Premium
30	10601	Available	10.0	Standard
31	10602	Occupied	12.5	Premium
32	10603	Available	10.0	Standard
33	10604	Available	12.5	Premium
34	10605	Occupied	10.0	Standard
35	10606	Available	12.5	Premium
36	10607	Occupied	10.0	Standard
37	10608	Available	12.5	Premium
38	10609	Available	10.0	Standard
39	10610	Occupied	12.5	Premium
40	10801	Occupied	10.0	Standard
41	10802	Available	12.5	Premium
42	10803	Occupied	10.0	Standard
43	10804	Available	12.5	Premium
44	10805	Occupied	10.0	Standard
45	10806	Occupied	12.5	Premium
46	10807	Available	10.0	Standard
47	10808	Occupied	12.5	Premium
48	10809	Available	10.0	Standard
49	10810	Occupied	12.5	Premium
50	11001	Available	10.0	Standard
51	11002	Occupied	12.5	Premium
52	11003	Available	10.0	Standard
53	11004	Available	12.5	Premium
54	11005	Occupied	10.0	Standard
55	11006	Occupied	12.5	Premium
56	11007	Available	10.0	Standard
57	11008	Occupied	12.5	Premium
58	11009	Available	10.0	Standard
59	11010	Occupied	12.5	Premium

In [19]:

```
includes_df = pd.read_sql_query("Select * from Includes", conn)
includes_df
```

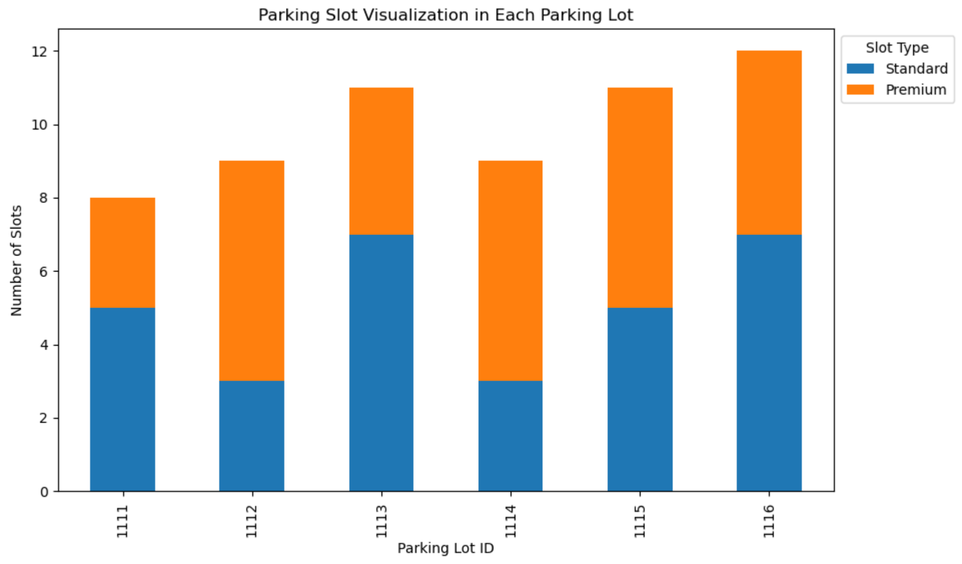
Out[19]:

	SlotID	LotID
0	10005	1111
1	10010	1111
2	10203	1111
3	10206	1111
4	10402	1111
5	10407	1111
6	10607	1111
7	10805	1111
8	10003	1112
9	10202	1112
10	10209	1112
11	10410	1112
12	10603	1112
13	10808	1112
14	10810	1112
15	11004	1112
16	11008	1112
17	10001	1113
18	10008	1113
19	10207	1113
20	10210	1113
21	10403	1113
22	10605	1113
23	10610	1113
24	10803	1113
25	10807	1113
26	11002	1113
27	11007	1113
28	10006	1114
29	10208	1114
30	10406	1114
31	10409	1114
32	10601	1114
33	10608	1114
34	10804	1114
35	11001	1114
36	11006	1114
37	10007	1115
38	10009	1115
39	10204	1115
40	10404	1115
41	10408	1115
42	10602	1115
43	10609	1115
44	10802	1115
45	10809	1115
46	11003	1115
47	11010	1115
48	10002	1116
49	10004	1116
50	10201	1116
51	10205	1116
52	10401	1116
53	10405	1116
54	10604	1116
55	10606	1116
56	10801	1116
57	10806	1116
58	11005	1116
59	11009	1116

1. Visualization to find the number of standard and premium parking slots in each parking lots.

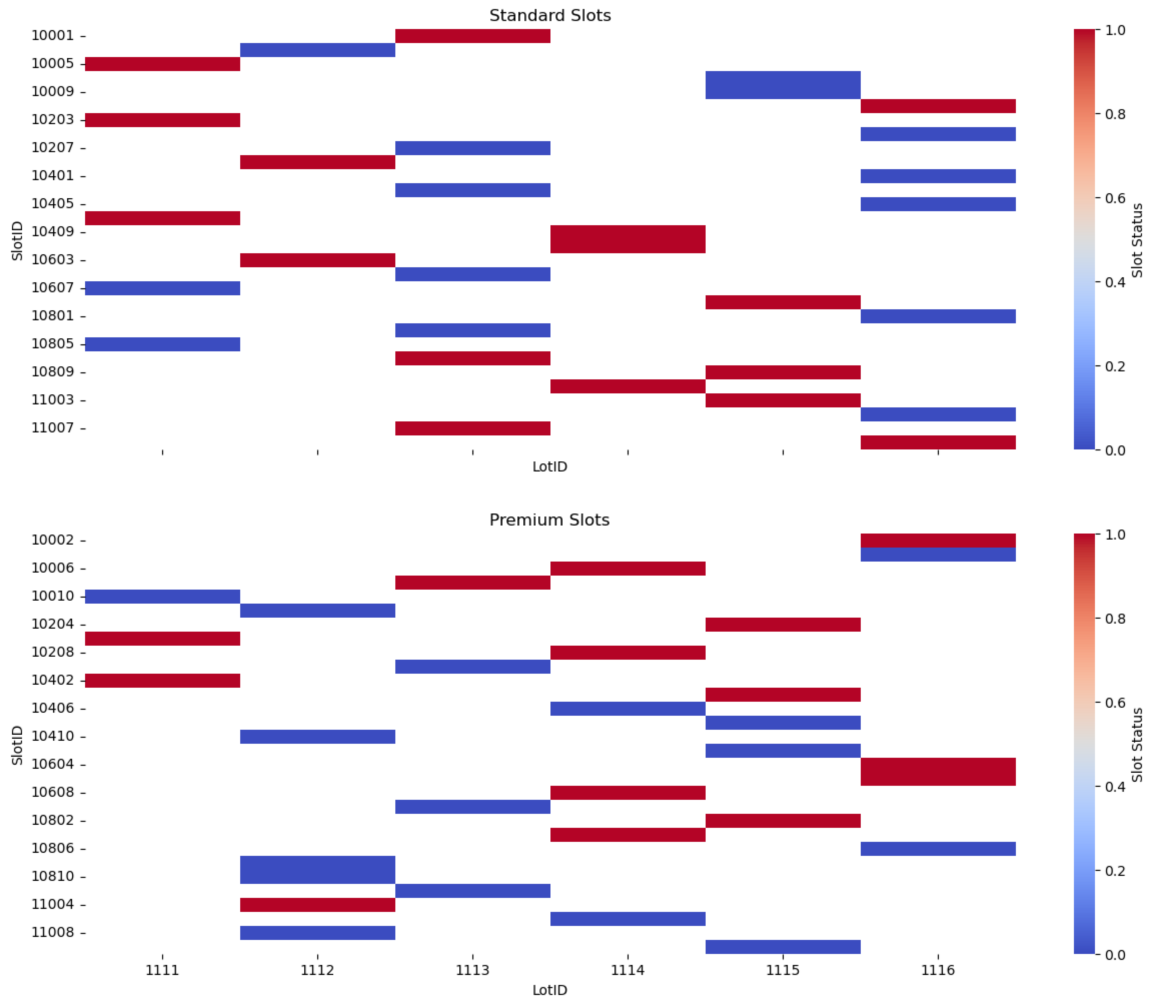
In [20]:

```python
merged_df = pd.merge(parking_slot_df, includes_df, on='SlotID', how='right')
merged_df['Standard'] = np.where(merged_df['Type'] == 'Standard', 1, 0)
merged_df['Premium'] = np.where(merged_df['Type'] == 'Premium', 1, 0)

grouped_df = merged_df.groupby('LotID')[['Standard', 'Premium']].sum()

fig, ax = plt.subplots(figsize=(10, 6))
grouped_df.plot(kind='bar', stacked=True, ax=ax)
ax.set_xlabel('Parking Lot ID')
ax.set_ylabel('Number of Slots')
ax.set_title('Parking Slot Visualization in Each Parking Lot')
ax.legend(title='Slot Type', bbox_to_anchor=(1, 1))

plt.show()
```



2. To check the availibilty of the parking slots.

In [48]:
```
heatmap_data = merged_df.copy()

heatmap_data['Status'] = heatmap_data['Status'].map({'Available': 1, 'Occupied': 0})

heatmap_data_standard = heatmap_data[heatmap_data['Type'] == 'Standard'].pivot(index='SlotID', columns='LotID', values='Status')
heatmap_data_premium = heatmap_data[heatmap_data['Type'] == 'Premium'].pivot(index='SlotID', columns='LotID', values='Status')

fig, axes = plt.subplots(2, 1, figsize=(15, 12), sharex=True)

sns.heatmap(heatmap_data_standard, cmap='coolwarm', cbar_kws={'label': 'Slot Status'}, ax=axes[0])
sns.heatmap(heatmap_data_premium, cmap='coolwarm', cbar_kws={'label': 'Slot Status'}, ax=axes[1])

axes[0].set_title('Standard Slots')
axes[1].set_title('Premium Slots')

plt.show()
```



In [21]:
```
booking_df = pd.read_sql_query("Select * from Booking", conn)
booking_df
```

```
Out[21]:
        BookingID        Date        StartTime          EndTime     Status  TransactionID  SlotID
   0           21  2023-12-18  0 days 15:30:00  0 days 19:30:00  Confirmed         100021   10401
   1           22  2023-12-19  0 days 12:15:00  0 days 16:15:00    Pending         100022   10402
   2           23  2023-12-20  0 days 10:00:00  0 days 14:00:00  Cancelled         100023   10403
   3           24  2023-12-21  0 days 14:45:00  0 days 18:45:00  Confirmed         100024   10404
   4           25  2023-12-22  0 days 11:30:00  0 days 15:30:00    Pending         100025   10405
 ...          ...         ...              ...              ...        ...            ...     ...
 125          166  2024-06-26  0 days 13:15:00  0 days 17:15:00  Confirmed         980103   10603
 126          167  2024-06-27  0 days 16:00:00  0 days 20:00:00    Pending         980104   10604
 127          168  2024-06-28  0 days 11:45:00  0 days 15:45:00  Cancelled         980105   10605
 128          169  2024-06-29  0 days 09:30:00  0 days 13:30:00  Confirmed         980106   10606
 129          170  2024-06-30  0 days 13:15:00  0 days 17:15:00    Pending         980107   10607

130 rows × 7 columns
```

```
In [30]: payment_df = pd.read_sql_query("Select * from Payment", conn)
         payment_df
```

```
Out[30]:
        TransactionID        Date             Time  PaymentStatus  Amount  MemberID
   0           100021  2023-02-20  0 days 14:30:00        Success    10.0      1001
   1           100022  2023-02-21  0 days 15:45:00        Success    12.5      1002
   2           100023  2023-02-22  0 days 16:15:00        Pending    10.0      1003
   3           100024  2023-02-23  0 days 17:00:00        Success    12.5      1004
   4           100025  2023-02-24  0 days 18:20:00        Pending    10.0      1005
 ...              ...         ...              ...            ...     ...       ...
 125           980103  2023-06-25  0 days 19:10:00        Success    12.5      1026
 126           980104  2023-06-26  0 days 20:30:00        Success    10.0      1027
 127           980105  2023-06-27  0 days 21:45:00        Pending    12.5      1028
 128           980106  2023-06-28  0 days 22:15:00        Success    10.0      1029
 129           980107  2023-06-29  0 days 23:00:00        Success    12.5      1030

130 rows × 6 columns
```

```
In [31]: discount_df = pd.read_sql_query("Select * from Discount", conn)
         discount_df
```

```
Out[31]:
    DiscountID  Status  Description  Percentage
 0         991  Active      10% off        10.0
 1         992  Active      15% off        15.0
 2         993  Active      20% off        20.0
 3         994  Active      25% off        25.0
```

```
In [32]: member_df = pd.read_sql_query("Select * from Member", conn)
         member_df
```

Out[32]:

	MemberID	PhoneNo	Email	Address	Name	MembershipID	Username
0	1001	123-456-7890	john.doe@email.com	123 Main St, Cityville	John Doe	10000001	john_doe_01
1	1002	234-567-8901	jane.smith@email.com	456 Oak St, Townsville	Jane Smith	10000002	jane_smith_02
2	1003	345-678-9012	bob.jones@email.com	789 Pine St, Villageton	Bob Jones	10000003	bob_jones_03
3	1004	456-789-0123	susan.white@email.com	987 Elm St, Hamletville	Susan White	10000004	susan_white_04
4	1005	567-890-1234	mike.brown@email.com	654 Birch St, Countryside	Mike Brown	10000005	mike_brown_05
5	1006	678-901-2345	emily.green@email.com	321 Cedar St, Hilltop	Emily Green	10000006	emily_green_06
6	1007	789-012-3456	david.gray@email.com	876 Maple St, Lakeside	David Gray	10000007	david_gray_07
7	1008	890-123-4567	laura.black@email.com	543 Redwood St, Riverside	Laura Black	10000008	laura_black_08
8	1009	901-234-5678	chris.baker@email.com	210 Fir St, Mountainside	Chris Baker	10000009	chris_baker_09
9	1010	012-345-6789	amy.taylor@email.com	135 Pinecone St, Meadowville	Amy Taylor	10000010	amy_taylor_10
10	1011	123-234-5678	robert.johnson@email.com	456 Birchwood St, Hillside	Robert Johnson	10000011	robert_johnson_11
11	1012	234-345-6789	olivia.martin@email.com	789 Maplewood St, Lakeshore	Olivia Martin	10000012	olivia_martin_12
12	1013	345-456-7890	william.moore@email.com	987 Oakwood St, Countrysidelake	William Moore	10000013	william_moore_13
13	1014	456-567-8901	grace.wilson@email.com	654 Redwoodwood St, Rivertown	Grace Wilson	10000014	grace_wilson_14
14	1015	567-678-9012	jackson.mitchell@email.com	321 Cedarwood St, Hilltopville	Jackson Mitchell	10000015	jackson_mitchell_15
15	1016	678-789-0123	ava.hill@email.com	876 Mapleside St, Lakesidehill	Ava Hill	10000016	ava_hill_16
16	1017	789-890-1234	nathan.adams@email.com	543 Pinehill St, Mountainsidetown	Nathan Adams	10000017	nathan_adams_17
17	1018	890-901-2345	emma.hayes@email.com	210 Firside St, Meadowtown	Emma Hayes	10000018	emma_hayes_18
18	1019	901-012-3456	samuel.cooper@email.com	135 Oakmeadow St, Rivertownship	Samuel Cooper	10000019	samuel_cooper_19
19	1020	012-123-4567	mia.wood@email.com	456 Redside St, Countrysidewood	Mia Wood	10000020	mia_wood_20
20	1021	123-234-5678	daniel.ross@email.com	789 Maplehill St, Hillsidevale	Daniel Ross	10000021	daniel_ross_21
21	1022	234-345-6789	hannah.perry@email.com	987 Pinetown St, Meadowhill	Hannah Perry	10000022	hannah_perry_22
22	1023	345-456-7890	jack.harrison@email.com	654 Oakvale St, Riversidehill	Jack Harrison	10000023	jack_harrison_23
23	1024	456-567-8901	lily.butler@email.com	321 Pinevale St, Hilltown	Lily Butler	10000024	lily_butler_24
24	1025	567-678-9012	ryan.long@email.com	876 Oakmeadow St, Lakesidevale	Ryan Long	10000025	ryan_long_25
25	1026	678-789-0123	zoey.fisher@email.com	543 Pinewood St, Countrysidevale	Zoey Fisher	10000026	zoey_fisher_26
26	1027	789-890-1234	ethan.price@email.com	210 Cedarhill St, Meadowville	Ethan Price	10000027	ethan_price_27
27	1028	890-901-2345	oliver.ramirez@email.com	135 Birchvale St, Hilltopvale	Oliver Ramirez	10000028	oliver_ramirez_28
28	1029	901-012-3456	amelia.ward@email.com	456 Pinetown St, Riversidevale	Amelia Ward	10000029	amelia_ward_29
29	1030	012-123-4567	luke.bell@email.com	789 Mapletown St, Lakesideville	Luke Bell	10000030	luke_bell_30
30	1031	123-234-5678	leah.miller@email.com	987 Redvale St, Hilltownville	Leah Miller	10000031	leah_miller_31
31	1033	345-456-7890	sophia.stewart@email.com	321 Cedarwood St, Hillsideville	Sophia Stewart	10000032	sophia_stewart_32
32	1034	456-567-8901	gabriel.kelly@email.com	876 Mapleside St, Lakesidehill	Gabriel Kelly	10000033	gabriel_kelly_33
33	1035	567-678-9012	madison.richards@email.com	543 Pinemeadow St, Mountainsidetown	Madison Richards	10000034	madison_richards_34
34	1036	678-789-0123	logan.brooks@email.com	210 Oakhill St, Meadowtown	Logan Brooks	10000035	logan_brooks_35
35	1037	789-890-1234	zoey.hill@email.com	135 Birchside St, Rivertownship	Zoey Hill	10000036	zoey_hill_36
36	1038	890-901-2345	connor.wheeler@email.com	456 Pinetown St, Countrysidewood	Connor Wheeler	10000037	connor_wheeler_37
37	1039	901-012-3456	chloe.martin@email.com	789 Maplemeadow St, Hillsidevale	Chloe Martin	10000038	chloe_martin_38
38	1040	012-123-4567	ethan.cooper@email.com	987 Oakvale St, Lakesidevale	Ethan Cooper	10000039	ethan_cooper_39
39	1041	123-234-5678	olivia.price@email.com	654 Pinewood St, Riversidehill	Olivia Price	10000040	olivia_price_40
40	1042	234-345-6789	jacob.ramirez@email.com	321 Mapletown St, Hilltopvale	Jacob Ramirez	10000041	jacob_ramirez_41
41	1043	345-456-7890	emma.ward@email.com	876 Birchvale St, Lakesideville	Emma Ward	10000042	emma_ward_42
42	1044	456-567-8901	aiden.bell@email.com	543 Pinetown St, Countrysidevale	Aiden Bell	10000043	aiden_bell_43
43	1045	567-678-9012	mia.brooks@email.com	210 Mapleside St, Meadowville	Mia Brooks	10000044	mia_brooks_44
44	1046	678-789-0123	lucas.hill@email.com	135 Cedarhill St, Hilltopville	Lucas Hill	10000045	lucas_hill_45
45	1047	789-890-1234	aubrey.cooper@email.com	456 Oakside St, Riversideville	Aubrey Cooper	10000046	aubrey_cooper_46
46	1048	890-901-2345	zoey.richards@email.com	789 Redvale St, Hilltownville	Zoey Richards	10000047	zoey_richards_47
47	1049	901-012-3456	noah.wheeler@email.com	987 Pinevale St, Countrysideville	Noah Wheeler	10000048	noah_wheeler_48
48	1050	012-123-4567	olivia.martin@email.com	654 Hillmeadow St, Lakesidevale	Olivia Martin	10000049	olivia_martin_49
49	1051	123-234-5678	liam.kelly@email.com	321 Mapleside St, Hillsidehill	Liam Kelly	10000050	liam_kelly_50

In [33]:

```
membership_df = pd.read_sql_query("Select * from Membership", conn)
membership_df
```

Out[33]:

	MembershipID	Name	StartDate	EndDate	Status	DiscountID
0	10000001	Silver Membership	2023-01-01	2023-12-31	Active	991.0
1	10000002	Gold Membership	2023-01-01	2023-12-31	Active	992.0
2	10000003	Bronze Membership	2023-01-01	2023-12-31	Cancelled	NaN
3	10000004	Silver Membership	2023-01-01	2023-12-31	Active	991.0
4	10000005	Gold Membership	2023-01-01	2023-12-31	Active	992.0
5	10000006	Bronze Membership	2023-01-01	2023-12-31	Cancelled	NaN
6	10000007	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
7	10000008	Silver Membership	2023-01-01	2023-12-31	Active	991.0
8	10000009	Gold Membership	2023-01-01	2023-12-31	Onhold	NaN
9	10000010	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
10	10000011	Platinum Membership	2023-01-01	2023-12-31	Cancelled	NaN
11	10000012	Silver Membership	2023-01-01	2023-12-31	Active	991.0
12	10000013	Gold Membership	2023-01-01	2023-12-31	Onhold	NaN
13	10000014	Bronze Membership	2023-01-01	2023-12-31	Cancelled	NaN
14	10000015	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
15	10000016	Silver Membership	2023-01-01	2023-12-31	Active	991.0
16	10000017	Gold Membership	2023-01-01	2023-12-31	Active	992.0
17	10000018	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
18	10000019	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
19	10000020	Silver Membership	2023-01-01	2023-12-31	Cancelled	NaN
20	10000021	Gold Membership	2023-01-01	2023-12-31	Active	992.0
21	10000022	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
22	10000023	Platinum Membership	2023-01-01	2023-12-31	Cancelled	NaN
23	10000024	Silver Membership	2023-01-01	2023-12-31	Active	991.0
24	10000025	Gold Membership	2023-01-01	2023-12-31	Active	992.0
25	10000026	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
26	10000027	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
27	10000028	Silver Membership	2023-01-01	2023-12-31	Cancelled	NaN
28	10000029	Gold Membership	2023-01-01	2023-12-31	Active	992.0
29	10000030	Bronze Membership	2023-01-01	2023-12-31	Cancelled	NaN
30	10000031	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
31	10000032	Silver Membership	2023-01-01	2023-12-31	Active	991.0
32	10000033	Gold Membership	2023-01-01	2023-12-31	Active	992.0
33	10000034	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
34	10000035	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
35	10000036	Silver Membership	2023-01-01	2023-12-31	Active	991.0
36	10000037	Gold Membership	2023-01-01	2023-12-31	Active	992.0
37	10000038	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
38	10000039	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
39	10000040	Silver Membership	2023-01-01	2023-12-31	Active	991.0
40	10000041	Gold Membership	2023-01-01	2023-12-31	Cancelled	NaN
41	10000042	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
42	10000043	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
43	10000044	Silver Membership	2023-01-01	2023-12-31	Cancelled	NaN
44	10000045	Gold Membership	2023-01-01	2023-12-31	Active	992.0
45	10000046	Bronze Membership	2023-01-01	2023-12-31	Active	993.0
46	10000047	Platinum Membership	2023-01-01	2023-12-31	Active	994.0
47	10000048	Silver Membership	2023-01-01	2023-12-31	Onhold	NaN
48	10000049	Gold Membership	2023-01-01	2023-12-31	Active	992.0
49	10000050	Bronze Membership	2023-01-01	2023-12-31	Onhold	NaN

In [34]:

```python
result_df = pd.read_sql_query("SELECT m.MemberID, m.Name, p.Amount AS TotalAmount, d.Percentage AS DiscountPercentage FROM Payment p JOIN Member m ON p.MemberID = m.MemberID LEFT JOIN Membersh
result_df
```

Out[34]:

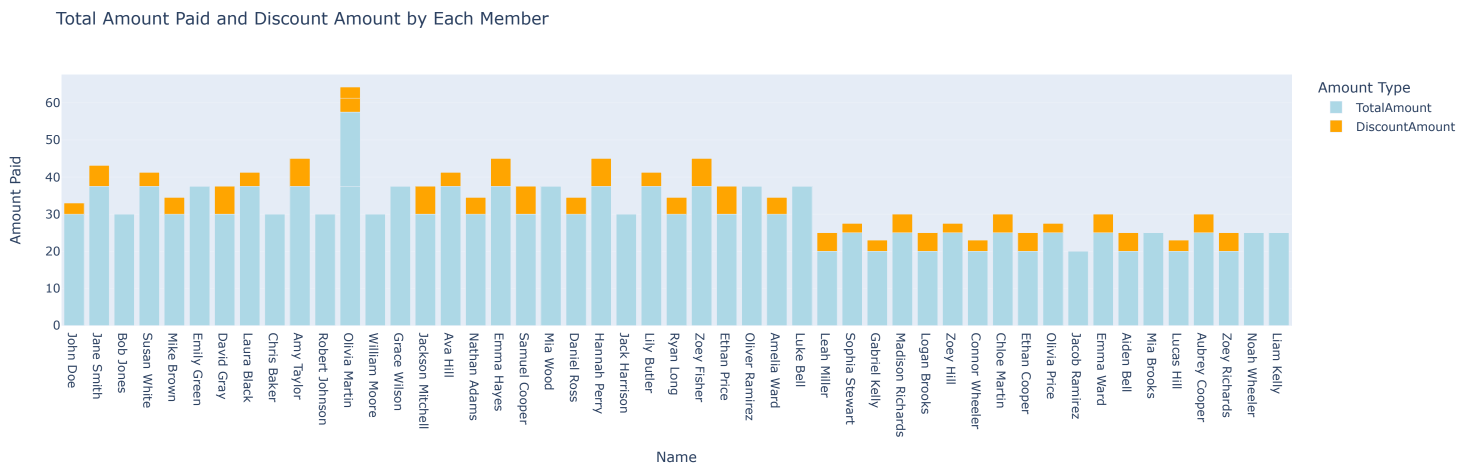
	MemberID	Name	TotalAmount	DiscountPercentage
0	1001	John Doe	10.0	10.0
1	1001	John Doe	10.0	10.0
2	1001	John Doe	10.0	10.0
3	1002	Jane Smith	12.5	15.0
4	1002	Jane Smith	12.5	15.0
...	...	...	...	...
125	1049	Noah Wheeler	12.5	NaN
126	1050	Olivia Martin	10.0	15.0
127	1050	Olivia Martin	10.0	15.0
128	1051	Liam Kelly	12.5	NaN
129	1051	Liam Kelly	12.5	NaN

130 rows × 4 columns

3. Visualization of total amount paid and discounted amount for each member

In [36]:

```python
import plotly.express as px

result_df['DiscountAmount'] = result_df['TotalAmount'] * (result_df['DiscountPercentage'] / 100)

result_df['NetAmount'] = result_df['TotalAmount'] - result_df['DiscountAmount']

grouped_df = result_df.groupby(['MemberID', 'Name']).agg({'TotalAmount': 'sum', 'DiscountAmount': 'sum'}).reset_index()

fig = px.bar(grouped_df, x='Name', y=['TotalAmount', 'DiscountAmount'],
             labels={'value': 'Amount Paid', 'variable': 'Amount Type'},
             title='Total Amount Paid and Discount Amount by Each Member',
             color_discrete_map={'TotalAmount': 'lightblue', 'DiscountAmount': 'orange'},
             height=500)

fig.show()
```



In [49]:
```
admin_df = pd.read_sql_query("Select * from Admin", conn)
admin_df
```

Out[49]:

	AdminID	Name	Address	Email	PhoneNo	Username
0	1	John Smith	123 Main St	john.smith@email.com	555-1234	john_smith
1	2	Alice Johnson	456 Oak St	alice.johnson@email.com	555-5678	alice_johnson
2	3	Bob Brown	789 Pine St	bob.brown@email.com	555-4321	bob_brown
3	4	Emily Davis	101 Elm St	emily.davis@email.com	555-8765	emily_davis
4	5	David White	202 Maple St	david.white@email.com	555-9876	david_white
5	6	Samantha Miller	303 Birch St	samantha.miller@email.com	555-3456	samantha_miller
6	7	Michael Wilson	404 Cedar St	michael.wilson@email.com	555-6543	michael_wilson
7	8	Olivia Moore	505 Pine St	olivia.moore@email.com	555-7890	olivia_moore
8	9	William Taylor	606 Oak St	william.taylor@email.com	555-2345	william_taylor
9	10	Emma Harris	707 Elm St	emma.harris@email.com	555-5432	emma_harris
10	41	Christopher Hall	111 Cedar St	christopher.hall@email.com	555-9876	christopher_hall
11	42	Mia Turner	222 Pine St	mia.turner@email.com	555-6543	mia_turner
12	43	Andrew Brooks	333 Elm St	andrew.brooks@email.com	555-7890	andrew_brooks
13	44	Sophia Ward	444 Birch St	sophia.ward@email.com	555-2345	sophia_ward
14	45	Matthew Butler	555 Oak St	matthew.butler@email.com	555-5432	matthew_butler

In [50]:
```
incident_df = pd.read_sql_query("Select * from Incident", conn)
incident_df
```

Out[50]:

	IncidentID	ResolutionStatus	Date_Time	Description	MemberID
0	2001	Resolved	2023-04-10 10:30:00	Network connectivity issue	1001
1	2002	Pending	2023-04-12 14:15:00	Software installation problem	1003
2	2003	Resolved	2023-04-15 11:45:00	Printer malfunction	1005
3	2004	Pending	2023-04-18 09:20:00	Password reset request	1007
4	2005	Resolved	2023-04-20 16:00:00	Hardware replacement needed	1009

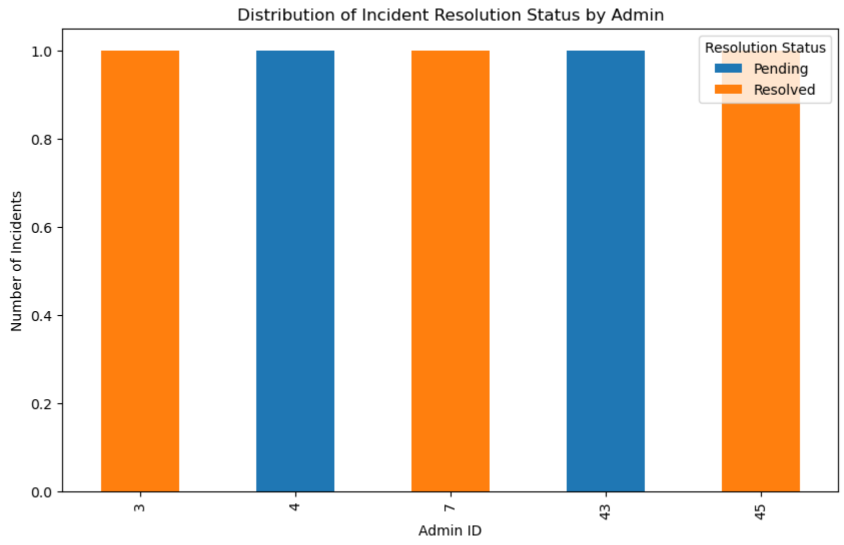
In [52]:
```
inspect_df = pd.read_sql_query("Select * from Inspect", conn)
inspect_df
```

Out[52]:

	IncidentID	AdminID
0	2001	3
1	2002	4
2	2003	7
3	2004	43
4	2005	45

4. Visualizing the distribution of incident resolution status performed by each admin.

In [53]:
```
m_df = pd.merge(incident_df, inspect_df, on='IncidentID')
m_df = pd.merge(m_df, admin_df, on='AdminID')

incident_counts = pd.crosstab(m_df['AdminID'], m_df['ResolutionStatus'])

# Plotting the bar chart
incident_counts.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Distribution of Incident Resolution Status by Admin')
plt.xlabel('Admin ID')
plt.ylabel('Number of Incidents')
plt.legend(title='Resolution Status')
plt.show()
```



In [54]:
```python
vehicle_df = pd.read_sql_query("Select * from Vehicle", conn)
vehicle_df
```

Out[54]:

	VehicleNumber	Make	Color	VehicleType	BookingID	MemberID
0	ABC123	Toyota	Blue	Sedan	21	1001
1	ABC789	Ford	Gray	Truck	48	1028
2	BCD234	Nissan	White	Sedan	31	1011
3	BCD890	Toyota	Red	SUV	57	1038
4	CDE123	Honda	Black	Coupe	40	1020
5	CDE789	Ford	White	Sedan	86	1047
6	DEF234	Nissan	Blue	Sedan	49	1029
7	DEF456	Ford	Green	Truck	23	1003
8	EFG123	Chevrolet	Green	Truck	58	1039
9	EFG567	Hyundai	Silver	SUV	32	1012
10	FGH234	Nissan	Silver	SUV	87	1048
11	FGH456	Chevrolet	Gray	Truck	41	1021
12	GHI567	Hyundai	Red	SUV	50	1030
13	GHI789	Chevrolet	White	Sedan	24	1004
14	HIJ456	Honda	White	Sedan	59	1040
15	HIJ890	Toyota	Black	Coupe	33	1013
16	IJK567	Hyundai	Black	Coupe	88	1049
17	IJK789	Ford	Blue	Sedan	42	1022
18	JKL234	Nissan	Silver	SUV	25	1005
19	JKL890	Toyota	Green	Truck	51	1031
20	KLM123	Honda	Gray	Truck	34	1014
21	KLM789	Ford	Silver	SUV	60	1041
22	LMN234	Nissan	Red	SUV	43	1023
23	LMN890	Toyota	Gray	Truck	89	1050
24	MNO123	Honda	White	Sedan	52	1033
25	MNO567	Hyundai	Black	Coupe	26	1006
26	NOP234	Nissan	Black	Coupe	81	1042
27	NOP456	Ford	Blue	Sedan	35	1015
28	OPQ123	Chevrolet	Blue	Sedan	90	1051
29	OPQ567	Hyundai	Green	Truck	44	1024
30	PQR456	Chevrolet	Silver	SUV	53	1034
31	PQR890	Toyota	Gray	Truck	27	1007
32	QRS567	Hyundai	Gray	Truck	82	1043
33	QRS789	Chevrolet	Red	SUV	36	1016
34	RST890	Toyota	White	Sedan	45	1025
35	STU123	Honda	Blue	Sedan	28	1008
36	STU789	Ford	Black	Coupe	54	1035
37	TUV234	Nissan	Green	Truck	37	1017
38	TUV890	Toyota	Blue	Sedan	83	1044
39	UVW123	Honda	Silver	SUV	46	1026
40	VWX234	Nissan	Gray	Truck	55	1036
41	VWX456	Ford	Red	SUV	29	1009
42	WXY123	Chevrolet	Red	SUV	84	1045
43	WXY567	Hyundai	White	Sedan	38	1018
44	XYZ456	Chevrolet	Black	Coupe	47	1027
45	XYZ789	Honda	Red	SUV	22	1002
46	YZA567	Hyundai	Blue	Sedan	56	1037
47	YZA789	Chevrolet	Green	Truck	30	1010
48	ZAB456	Honda	Green	Truck	85	1046
49	ZAB890	Toyota	Silver	SUV	39	1019

In [57]:
```python
merged_df = pd.merge(vehicle_df, booking_df, on='BookingID')
merged_df
```

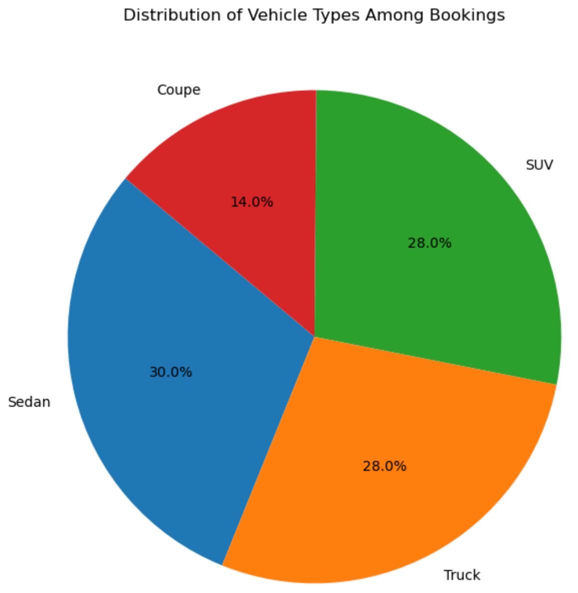
Out[57]:

	VehicleNumber	Make	Color	VehicleType	BookingID	MemberID	Date	StartTime	EndTime	Status	TransactionID	SlotID
0	ABC123	Toyota	Blue	Sedan	21	1001	2023-12-18	0 days 15:30:00	0 days 19:30:00	Confirmed	100021	10401
1	ABC789	Ford	Gray	Truck	48	1028	2024-01-14	0 days 13:45:00	0 days 17:45:00	Confirmed	100048	10808
2	BCD234	Nissan	White	Sedan	31	1011	2023-12-28	0 days 11:00:00	0 days 15:00:00	Pending	100031	10601
3	BCD890	Toyota	Red	SUV	57	1038	2024-01-23	0 days 14:30:00	0 days 18:30:00	Confirmed	100057	11007
4	CDE123	Honda	Black	Coupe	40	1020	2024-01-06	0 days 10:45:00	0 days 14:45:00	Pending	100040	10610
5	CDE789	Ford	White	Sedan	86	1047	2024-04-06	0 days 16:15:00	0 days 20:15:00	Confirmed	980023	11003
6	DEF234	Nissan	Blue	Sedan	49	1029	2024-01-15	0 days 11:30:00	0 days 15:30:00	Pending	100049	10809
7	DEF456	Ford	Green	Truck	23	1003	2023-12-20	0 days 10:00:00	0 days 14:00:00	Cancelled	100023	10403
8	EFG123	Chevrolet	Green	Truck	58	1039	2024-01-24	0 days 12:15:00	0 days 16:15:00	Pending	100058	11008
9	EFG567	Hyundai	Silver	SUV	32	1012	2023-12-29	0 days 09:45:00	0 days 13:45:00	Cancelled	100032	10602
10	FGH234	Nissan	Silver	SUV	87	1048	2024-04-07	0 days 11:00:00	0 days 15:00:00	Pending	980024	11004
11	FGH456	Chevrolet	Gray	Truck	41	1021	2024-01-07	0 days 12:30:00	0 days 16:30:00	Cancelled	100041	10801
12	GHI567	Hyundai	Red	SUV	50	1030	2024-01-16	0 days 09:15:00	0 days 13:15:00	Cancelled	100050	10810
13	GHI789	Chevrolet	White	Sedan	24	1004	2023-12-21	0 days 14:45:00	0 days 18:45:00	Confirmed	100024	10404
14	HIJ456	Honda	White	Sedan	59	1040	2024-01-25	0 days 10:00:00	0 days 14:00:00	Cancelled	100059	11009
15	HIJ890	Toyota	Black	Coupe	33	1013	2023-12-30	0 days 14:30:00	0 days 18:30:00	Confirmed	100033	10603
16	IJK567	Hyundai	Black	Coupe	88	1049	2024-04-08	0 days 09:45:00	0 days 13:45:00	Cancelled	980025	11005
17	IJK789	Ford	Blue	Sedan	42	1022	2024-01-08	0 days 16:15:00	0 days 20:15:00	Confirmed	100042	10802
18	JKL234	Nissan	Silver	SUV	25	1005	2023-12-22	0 days 11:30:00	0 days 15:30:00	Pending	100025	10405
19	JKL890	Toyota	Green	Truck	51	1031	2024-01-17	0 days 13:00:00	0 days 17:00:00	Confirmed	100051	11001
20	KLM123	Honda	Gray	Truck	34	1014	2023-12-31	0 days 12:15:00	0 days 16:15:00	Pending	100034	10604
21	KLM789	Ford	Silver	SUV	60	1041	2024-01-26	0 days 13:45:00	0 days 17:45:00	Confirmed	100060	11008
22	LMN234	Nissan	Red	SUV	43	1023	2024-01-09	0 days 11:00:00	0 days 15:00:00	Pending	100043	10803
23	LMN890	Toyota	Gray	Truck	89	1050	2024-04-09	0 days 14:30:00	0 days 18:30:00	Confirmed	980026	11006
24	MNO123	Honda	White	Sedan	52	1033	2024-01-18	0 days 10:45:00	0 days 14:45:00	Pending	100052	11002
25	MNO567	Hyundai	Black	Coupe	26	1006	2023-12-23	0 days 09:15:00	0 days 13:15:00	Cancelled	100026	10406
26	NOP234	Nissan	Black	Coupe	81	1042	2024-04-01	0 days 11:30:00	0 days 15:30:00	Pending	980018	10808
27	NOP456	Ford	Blue	Sedan	35	1015	2024-01-01	0 days 10:00:00	0 days 14:00:00	Cancelled	100035	10605
28	OPQ123	Chevrolet	Blue	Sedan	90	1051	2024-04-10	0 days 12:15:00	0 days 16:15:00	Pending	980027	11007
29	OPQ567	Hyundai	Green	Truck	44	1024	2024-01-10	0 days 09:45:00	0 days 13:45:00	Cancelled	100044	10804
30	PQR456	Chevrolet	Silver	SUV	53	1034	2024-01-19	0 days 12:30:00	0 days 16:30:00	Cancelled	100053	11003
31	PQR890	Toyota	Gray	Truck	27	1007	2023-12-24	0 days 13:00:00	0 days 17:00:00	Confirmed	100027	10407
32	QRS567	Hyundai	Gray	Truck	82	1043	2024-04-02	0 days 09:15:00	0 days 13:15:00	Cancelled	980019	10809
33	QRS789	Chevrolet	Red	SUV	36	1016	2024-01-02	0 days 13:45:00	0 days 17:45:00	Confirmed	100036	10606
34	RST890	Toyota	White	Sedan	45	1025	2024-01-11	0 days 14:30:00	0 days 18:30:00	Confirmed	100045	10805
35	STU123	Honda	Blue	Sedan	28	1008	2023-12-25	0 days 10:45:00	0 days 14:45:00	Pending	100028	10408
36	STU789	Ford	Black	Coupe	54	1035	2024-01-20	0 days 16:15:00	0 days 20:15:00	Confirmed	100054	11004
37	TUV234	Nissan	Green	Truck	37	1017	2024-01-03	0 days 11:30:00	0 days 15:30:00	Pending	100037	10607
38	TUV890	Toyota	Blue	Sedan	83	1044	2024-04-03	0 days 13:00:00	0 days 17:00:00	Confirmed	980020	10810
39	UVW123	Honda	Silver	SUV	46	1026	2024-01-12	0 days 12:15:00	0 days 16:15:00	Pending	100046	10806
40	VWX234	Nissan	Gray	Truck	55	1036	2024-01-21	0 days 11:00:00	0 days 15:00:00	Pending	100055	11005
41	VWX456	Ford	Red	SUV	29	1009	2023-12-26	0 days 12:30:00	0 days 16:30:00	Cancelled	100029	10409
42	WXY123	Chevrolet	Red	SUV	84	1045	2024-04-04	0 days 10:45:00	0 days 14:45:00	Pending	980021	11001
43	WXY567	Hyundai	White	Sedan	38	1018	2024-01-04	0 days 09:15:00	0 days 13:15:00	Cancelled	100038	10608
44	XYZ456	Chevrolet	Black	Coupe	47	1027	2024-01-13	0 days 10:00:00	0 days 14:00:00	Cancelled	100047	10807
45	XYZ789	Honda	Red	SUV	22	1002	2023-12-19	0 days 12:15:00	0 days 16:15:00	Pending	100022	10402
46	YZA567	Hyundai	Blue	Sedan	56	1037	2024-01-22	0 days 09:45:00	0 days 13:45:00	Cancelled	100056	11006
47	YZA789	Chevrolet	Green	Truck	30	1010	2023-12-27	0 days 16:15:00	0 days 20:15:00	Confirmed	100030	10410
48	ZAB456	Honda	Green	Truck	85	1046	2024-04-05	0 days 12:30:00	0 days 16:30:00	Cancelled	980022	11002
49	ZAB890	Toyota	Silver	SUV	39	1019	2024-01-05	0 days 13:00:00	0 days 17:00:00	Confirmed	100039	10609

5. Pie chart to represent the distribution of vehicle types among the bookings.

In [60]:

```python
vehicle_type_counts = merged_df['VehicleType'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(vehicle_type_counts, labels=vehicle_type_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Vehicle Types Among Bookings')
plt.show()
```


Distribution of Vehicle Types Among Bookings
Coupe
14.0%
SUV
28.0%
Sedan
30.0%
28.0%
Truck

```
In [ ]:
```