Python Code Explaination for Monthly Billing Generation

1. Import Required Modules

```
from datetime import datetime
from collections import defaultdict
from calendar import monthrange
```

Explanation:

- datetime: Used for parsing and handling dates.
- defaultdict: A subclass of dictionary that simplifies grouped data aggregation.
- monthrange: Provides the start weekday and number of days in a month .

2. Function: parse_date

```
def parse_date(date_str):
    return datetime.strptime(date_str, "%Y-%m-%d")
```

Explanation:

This function converts a date string (like "2024-11-01") into a datetime object. Required for date comparisons.

3. Function: get_month_range

```
def get_month_range(target_month):
    year, month = map(int, target_month.split("-"))
    start_date = datetime(year, month, 1)
    end_date = datetime(year, month, monthrange(year, month)[1])
    return start_date, end_date
```

Explanation:

- Extracts the start and end dates of the billing month (e.g., November 2024).
- Useful to determine how many days to pro-rate the item cost.

4. Function: generate_monthly_bill

```
def generate_monthly_bill(item_list: list, target_month: str) -> dict:
```

This function calculates monthly billing for all items based on their active periods and quantities.

4.1 Determine the Billing Month Range

```
month_start, month_end = get_month_range(target_month)
total_days_in_month = (month_end - month_start).days + 1
```

- Computes total number of days in the billing month.

4.2 Group Items for Consolidated Billing

```
    grouped_items = defaultdict(lambda: {"qty": 0, "amount": 0.0})
```

- Prepares to group line items by (item_code, rate, billing_period) to consolidate similar charges.

4.3 Process Each Item

```
for item in item_list:
    start_date = parse_date(item["start_date"])
    stop_date = parse_date(item["stop_date"])
    active_start = max(start_date, month_start)
    active_end = min(stop_date, month_end)
    if active_start > active_end:
        continue
```

- Calculates the active period of each item within the target billing month.
- Skips the item if it's not active during the month.

4.4 Calculate Pro-rated Amount

```
    qty = int(item["qty"])
    rate = float(item["rate"])
    active_days = (active_end - active_start).days + 1
    amount = rate * qty * (active_days / total_days_in_month)
```

- Pro-rates the amount based on how many days the item was active.
- Handles both integer and string formats for qty and rate

4.5 Group by Item + Rate + Billing Period

```
    billing_period = f"{month_start.date()} to {month_end.date()}"
    key = (item["item_code"], rate, billing_period)
    grouped_items[key]["qty"] += qty
    grouped_items[key]["amount"] += amount
```

- Aggregates all similar entries into a single line item.

4.6 Format Final Output

```
line_items = []
total_revenue = 0.0
for (item_code, rate, billing_period), values in grouped_items.items():
```

```
amt = round(values["amount"], 2)
line_items.append({
    "item_code": item_code,
    "rate": rate,
    "qty": values["qty"],
    "amount": amt,
    "billing_period": billing_period
})
total_revenue += amt
```

- Rounds and formats each grouped line item.
- Computes the total revenue.

4.7 Return Output Dictionary

```
return {
    "line_items": line_items,
    "total_revenue": round(total_revenue, 2)
}
```

- Final dictionary output structured as required.

5. Input Dataset (item_list)

```
item_list = [
    {
        "idx": 1,
        "item_code": "Executive Desk (4*2)",
        "sales_description": "Dedicated Executive Desk",
        "qty": 10,
        "rate": "1000",
        "amount": "10000",
        "start_date": "2023-11-01",
        "stop_date": "2024-10-17",
    },
    {
        "idx": 2,
        "item_code": "Executive Desk (4*2)",
        "qty": "10",
        "rate": "1080",
```

```
    "amount": "10800",
    "start_date": "2024-10-18",
    "stop_date": "2025-10-31",
  },
  {
    "idx": 3,
    "item_code": "Executive Desk (4*2)",
    "qty": 15,
    "rate": "1080",
    "amount": "16200",
    "start_date": "2024-11-01",
    "stop_date": "2025-10-31",
  },
  {
    "idx": 4,
    "item_code": "Executive Desk (4*2)",
    "qty": 5,
    "rate": "1000",
    "amount": "5000",
    "start_date": "2024-11-01",
    "stop_date": "2025-10-31",
  },
  {
    "idx": 5,
    "item_code": "Manager Cabin",
    "qty": 5,
    "rate": 5000,
    "amount": 25000,
    "start_date": "2024-11-01",
    "stop_date": "2025-10-31",
  },
  {
    "idx": 6,
    "item_code": "Manager Cabin",
    "qty": 7,
```

```
    "rate": "5000",
    "amount": 35000,
    "start_date": "2024-12-15",
    "stop_date": "2025-10-31",
  },
  {
    "idx": 7,
    "item_code": "Manager Cabin",
    "qty": 10,
    "rate": 4600,
    "amount": 46000,
    "start_date": "2023-11-01",
    "stop_date": "2024-10-17",
  },
  {
    "idx": 8,
    "item_code": "Parking (2S)",
    "qty": 10,
    "rate": 1000,
    "amount": 10000,
    "start_date": "2024-11-01",
    "stop_date": "2025-10-31",
  },
  {
    "idx": 9,
    "item_code": "Parking (2S)",
    "qty": 10,
    "rate": 0,
    "amount": 0,
    "start_date": "2024-11-01",
    "stop_date": "2025-10-31",
  },
  {
    "idx": 10,
    "item_code": "Executive Desk (4*2)",
```

```
        "qty": "8",
        "rate": "1100",
        "amount": "8800",
        "start_date": "2024-11-15",
        "stop_date": "2025-01-31",
    },
    {
        "idx": 11,
        "item_code": "Manager Cabin",
        "qty": "3",
        "rate": "5200",
        "amount": "15600",
        "start_date": "2024-10-10",
        "stop_date": "2024-11-10",
    },
    {
        "idx": 12,
        "item_code": "Conference Table",
        "qty": 1,
        "rate": "20000",
        "amount": "20000",
        "start_date": "2024-11-05",
        "stop_date": "2024-11-20",
    },
    {
        "idx": 13,
        "item_code": "Parking (2S)",
        "qty": 5,
        "rate": "1000",
        "amount": "5000",
        "start_date": "2024-11-15",
        "stop_date": "2025-02-28",
    },
    {
        "idx": 14,
```

```
        "item_code": "Reception Desk",
        "qty": 2,
        "rate": "7000",
        "amount": "14000",
        "start_date": "2024-11-01",
        "stop_date": "2025-03-31",
    },
    {
        "idx": 15,
        "item_code": "Reception Desk",
        "qty": 1,
        "rate": "7000",
        "amount": "7000",
        "start_date": "2024-11-10",
        "stop_date": "2024-11-25",
    },
    {
        "idx": 16,
        "item_code": "Breakout Area",
        "qty": 3,
        "rate": "3000",
        "amount": "9000",
        "start_date": "2024-01-01",
        "stop_date": "2024-01-31",
    }

]
```


- Contains all input entries with item details such as item_code, qty, rate, and activity dates.
- Used as a base for monthly billing computation.

6. Generate and Print Output

```
result = generate_monthly_bill(item_list, "2024-11")
print(result)
```

```
import json
print(json.dumps(result, indent=4))
```

- Calls the billing function for November 2024.
- Pretty-prints the result in JSON format with proper indentation for readability.

Sample Output Format


```
{
   "line_items": [
      {
         "item_code": "Executive Desk (4*2)",
         "rate": 1080.0,
         "qty": 25,
         "amount": 27000.0,
         "billing_period": "2024-11-01 to 2024-11-30"
      },
      ...
   ],
   "total_revenue": 107960.0
}
```