NAME: SHUMAIM QAMAR
ROLL NO: BSDSF24A017

Python Lab — Smart HealthCare Analytics

Case Study

The National HealthCare Innovation Center (NHIC) wants to improve patient care through data analytics. As a Junior Data Analyst, your responsibility is to analyze patient vitals, identify risk cases, and generate insights.

Dataset Format

patients_data = ["id:1,name:Ali Khan,age:45,doctor:Dr. Ahmed,hr:88,bp:120/80,glucose:110,feedback:Good", "id:2,name:Ayesha Noor,age:63,doctor:Dr. Sara,hr:102,bp:145/95,glucose:160,feedback:Critical", ...]

Q1 – Patient Record Conversion (Dictionary Format)

Description and explanation:

We have to change the given data into structured data that is a dictionary. In which each patient data will be the dictionary so there will be a total 50 dictionaries in a list. Now the line by line explanation is as follows:

1. First we will split the data whenever a comma occurred.
2. A dictionary named entry is created in which each patient data will be stored.
3. Loop through every patient data.
4. Now here split every patient data by ':'. First will be stored in the key and the data after colon will be the value. Now if the key is 'id', 'key' or 'glucose' typecast their values to int. And then store that key:value in the entry dictionary.
5. At the end return the structured data.

– Patient Record Conversion (Dictionary Format)

```
f structured(data):
  conv = data.split(',')
  entry = {}

  for d in conv:
      key, value = d.split(':')
      if key == "id" or key == "age" or key == "hr" or key == "glucose":
          value = int(value)
      entry[key] = value
  return entry

ructured_data = [structured(data) for data in patients_data]

r d in structured_data:
  print(d)
```

```
id': 1, 'name': 'Ali Khan', 'age': 45, 'doctor': 'Dr. Ahmed', 'hr': 88, 'bp': '120/80', 'glucose': 110, 'feedback': 'Good'}
id': 2, 'name': 'Ayesha Noor', 'age': 63, 'doctor': 'Dr. Sara', 'hr': 102, 'bp': '145/95', 'glucose': 160, 'feedback': 'Critical
id': 3, 'name': 'Usman Tariq', 'age': 52, 'doctor': 'Dr. Ahmed', 'hr': 76, 'bp': '118/78', 'glucose': 95, 'feedback': 'Excellent
id': 4, 'name': 'Sana Fatima', 'age': 29, 'doctor': 'Dr. Bilal', 'hr': 91, 'bp': '130/85', 'glucose': 125, 'feedback': 'Good'}
id': 5, 'name': 'Hamza Ali', 'age': 38, 'doctor': 'Dr. Sara', 'hr': 110, 'bp': '150/92', 'glucose': 180, 'feedback': 'Critical'}
id': 6, 'name': 'Maryam Iqbal', 'age': 71, 'doctor': 'Dr. Ahmed', 'hr': 82, 'bp': '135/88', 'glucose': 140, 'feedback': 'Fair'}
id': 7, 'name': 'Imran Hussain', 'age': 66, 'doctor': 'Dr. Noor', 'hr': 97, 'bp': '142/90', 'glucose': 165, 'feedback': 'Fair'}
id': 8, 'name': 'Rida Aslam', 'age': 54, 'doctor': 'Dr. Sara', 'hr': 105, 'bp': '149/94', 'glucose': 172, 'feedback': 'Critical'
id': 9, 'name': 'Hassan Raza', 'age': 27, 'doctor': 'Dr. Ahmed', 'hr': 72, 'bp': '117/79', 'glucose': 88, 'feedback': 'Excellent
id': 10, 'name': 'Fatima Sheikh', 'age': 44, 'doctor': 'Dr. Bilal', 'hr': 95, 'bp': '134/89', 'glucose': 130, 'feedback': 'Good'
id': 11, 'name': 'Bilal Khan', 'age': 61, 'doctor': 'Dr. Sara', 'hr': 99, 'bp': '140/92', 'glucose': 155, 'feedback': 'Fair'}
id': 12, 'name': 'Khadija Amin', 'age': 36, 'doctor': 'Dr. Ahmed', 'hr': 85, 'bp': '128/84', 'glucose': 115, 'feedback': 'Good'}
```

Q2 – Count Total Patients

Description and explanation:

In this task we have to return the count of the total patients. Just use the built-in formula len() and return the len of the data passed in the parameter. As the data is of type list and list contains the dictionary data of each patient so we can find the count by usung the len() formula.

Q2 – Count Total Patients

```
[42]: def count_pateints(data):
          return len(data)

      print(count_pateints(structured_data))
```

```
50
```

Q3 – Average Heart Rate Per Doctor

Q3 – Average Heart Rate Per Doctor

Description and explanation:

In this task we have to find the average heart rate per Doctor. Now one doctor is treating multiple patients.

1. First we will make a dictionary for the doctors in which the key will be doctor and value will be the average of heart rate. Loop through every patient data. Store the doctor name and the heart rate.
2. I have used the function " dr_hr.get(doctor, 0)+ hr ". This will check if a doctor already exists or not. If the doctor is already in the dictionary named dr_hr. The heart rate will be added in the existing value. If doctor does not exist the function will simply return 0 and the value will be added with 0 and stored in the respective doctor value.
3. Same is with the dr_count data.
4. After executing the loop, every doctor's heart rate(that is summed up) will be divided by their respective count.
5. At the end return the dr_hr, that will contain the doctor and their average heart rate.

Q3 – Average Heart Rate Per Doctor

```
[171]: def avg_heartrate(data):
           dr_hr = {}
           average = {}
           dr_count = {}
           for d in data:
               doctor = d['doctor']
               hr = p['hr']
               dr_hr[doctor] = dr_hr.get(doctor, 0) + hr
               dr_count[doctor] = dr_count.get(doctor, 0) + 1

           avg_hr = {doc: dr_hr[doc] / dr_count[doc] for doc in dr_hr}
           return avg_hr

       print(avg_heartrate(structured_data))
```

```
{'Dr. Ahmed': 89.0, 'Dr. Sara': 89.0, 'Dr. Bilal': 89.0, 'Dr. Noor': 89.0}
```

Q4 – High-Risk Patients (HR > 100)

Description and explanation:

In this last, we will list all the patient who have heart rate above 100.

1. Create a list named abv.
2. Start loop.
3. If hr is greater than 100, append that patient name in the list.
4. At the end return the abv list.

Q4 – High-Risk Patients (HR > 100)

```
[14]: def above_100(data):
          abv = []
          for d in data:
              if d['hr'] > 100:
                abv.append(d['name'])
          return abv

      print(above_100(structured_data))
```

```
['Ayesha Noor', 'Hamza Ali', 'Rida Aslam', 'Zain Ali', 'Irfan Malik', 'Rizwan Ali', 'Ali Raza', 'Mubashir Ali', 'Saba Malik', 'Waleed Khan', 'Talha Sharif', 'Uzma Raza']
```

Q5 – Lowest Glucose Level

Description and explanation:

In this task, we have to find the lowest glucose level of the patient.

1. Store the first patients' glucose value in the min variable.
2. Create a min_patient variable to store that patient name having lowest glucose level.

3. Loop through every patient, if the glucose is less than min, that value will be stored in min variable and also the patient name.
4. At the end return the min value.

Q5 – Lowest Glucose Level

```
[12]: def lowest_level(data):
          min = data[0]['glucose']
          min_patient = ""
          for d in data:
              if d['glucose'] < min:
                  min = d['glucose']
                  min_patient = d['name']
          return min ,min_patient

      print(lowest_level(structured_data))
```

```
(85, 'Zara Malik')
```

Q6 – Highest Glucose Level

Description and explanation:

This task is the same as the previous one but this time we have to find the max glucose level. Just replace the smaller than sign to the greater sign in the condition.
And at the end return the max value.

Q6 – Highest Glucose Level

```
[13]: def lowest_level(data):
          min = 0
          min_patient = ""
          for d in data:
              if d['glucose'] > min:
                  min = d['glucose']
                  min_patient = d['name']
          return min ,min_patient

      print(lowest_level(structured_data))
```

```
(180, 'Hamza Ali')
```

Q7 – Blood Pressure Alert (BP > 140/90)

Description and explanation:

In this task we have to return the patient's data whose BP is above 140/90.

1. Make a list to store the patients data.
2. Loop through every patient. Now we are required to split the BP As it contains '/' this.
3. Use split function and store the values in two variables i.e systolic and diastolic.
4. Then check whether systolic is above 140 or diastolic is above 90.
5. If the condition meets, append the patient's data in the list.
6. At the end return the list.

Q7 – Blood Pressure Alert (BP > 140/90)

[107]:

```python
def bp_alert(data):
    alerts = []

    for p in data:
        systolic, diastolic = p["bp"].split("/")
        systolic = int(systolic)
        diastolic = int(diastolic)

        if systolic > 140 or diastolic > 90:
            alerts.append({
                "id": p["id"],
                "name": p["name"],
                "bp": p["bp"],
                "doctor": p["doctor"]
            })
    return alerts


high_bp_patients = bp_alert(structured_data)

for a in high_bp_patients:   # show first 5
    print(a)
```

```
{'id': 2, 'name': 'Ayesha Noor', 'bp': '145/95', 'doctor': 'Dr. Sara'}
{'id': 5, 'name': 'Hamza Ali', 'bp': '150/92', 'doctor': 'Dr. Sara'}
{'id': 7, 'name': 'Imran Hussain', 'bp': '142/90', 'doctor': 'Dr. Noor'}
{'id': 8, 'name': 'Rida Aslam', 'bp': '149/94', 'doctor': 'Dr. Sara'}
{'id': 11, 'name': 'Bilal Khan', 'bp': '140/92', 'doctor': 'Dr. Sara'}
{'id': 13, 'name': 'Zain Ali', 'bp': '147/93', 'doctor': 'Dr. Noor'}
{'id': 17, 'name': 'Irfan Malik', 'bp': '150/95', 'doctor': 'Dr. Noor'}
{'id': 21, 'name': 'Hira Asghar', 'bp': '140/91', 'doctor': 'Dr. Ahmed'}
{'id': 23, 'name': 'Rizwan Ali', 'bp': '150/96', 'doctor': 'Dr. Bilal'}
{'id': 27, 'name': 'Ali Raza', 'bp': '152/97', 'doctor': 'Dr. Bilal'}
{'id': 29, 'name': 'Mahnoor Shah', 'bp': '141/93', 'doctor': 'Dr. Sara'}
{'id': 32, 'name': 'Mubashir Ali', 'bp': '148/94', 'doctor': 'Dr. Ahmed'}
{'id': 37, 'name': 'Saba Malik', 'bp': '151/97', 'doctor': 'Dr. Sara'}
{'id': 39, 'name': 'Naima Iqbal', 'bp': '139/91', 'doctor': 'Dr. Ahmed'}
{'id': 40, 'name': 'Waleed Khan', 'bp': '155/98', 'doctor': 'Dr. Bilal'}
{'id': 44, 'name': 'Talha Sharif', 'bp': '149/96', 'doctor': 'Dr. Bilal'}
{'id': 47, 'name': 'Rehan Ahmed', 'bp': '141/92', 'doctor': 'Dr. Noor'}
{'id': 50, 'name': 'Uzma Raza', 'bp': '146/95', 'doctor': 'Dr. Sara'}
```

Q8 – Distinct Doctors Count

Description and explanation:

In this task, we have to find the count of all the doctors that are treating patients in the given data.

1. Create a list to store the doctor's names.
2. Loop through every patient data.
3. Store doctor’s name in any variable and then check the condition whether the stored doctor is already in the list or not if not then append doctor name in the list.
4. At the end use the len() function to return the length of the doctors list. This will indicate the total number of doctors treating patients.

Q8 – Distinct Doctors Count

```
10]: def distinct_doctors_count(data):
         doctors = []

         for patient in data:
             doctor = patient["doctor"]
             if doctor not in doctors:
                 doctors.append(doctor)

         return len(doctors)


     print("Distinct Doctors:", distinct_doctors_count(structured_data))
```

```
Distinct Doctors: 4
```

Q9 – Feedback Distribution:

Description and explanation:

Count how many patients fall under each feedback type (Excellent, Good, Fair, Critical).

1. Make a dictionary to store the required data.
2. Loop through every patient. Store the patient feedback in any variable.
3. Check if the feedback already exists in the dictionary or not. If yes then add in the count by 1 and if not than initialize it's count to 1.
4. At the end return the dictionary, whose key will be the feedback name and value will be its count.

Q9 – Feedback Distribution:

[111]:
```
def feedback_distribution(data):
    feedback_count = {}

    for patient in data:
        fb = patient["feedback"]

        if fb not in feedback_count:
            feedback_count[fb] = 1
        else:
            feedback_count[fb] += 1

    return feedback_count


print(feedback_distribution(structured_data))
```

```
{'Good': 15, 'Critical': 12, 'Excellent': 10, 'Fair': 13}
```

Q10 – Overall Average Heart Rate

Description and explanation:

This task is simple.

1. Take the sum of every patient's hr, also calculate the count.
2. Divide sum by count and return it. That will be the overall average heart rate.

[113]:
```
def overall_average_hr(data):
    total_hr = 0
    count = 0

    for patient in data:
        total_hr += patient["hr"]
        count += 1

    return total_hr / count


print("Overall Average Heart Rate:", overall_average_hr(structured_data))
```

```
Overall Average Heart Rate: 90.08
```

Q11 – Health Grade Assignment

Description and explanation:

- HR < 70 = A (Excellent)
- HR 70–89 = B (Stable)
- HR 90–110 = C (Concern)
- HR > 110 = D (High Risk)

1. Loop through every patient data. Store patient's hr in any variable.
2. Check the above mentioned conditions. If meet with any condition, Store the respective grade.
3. And add it to the hr_grade key.
4. At the end return the data.

Q11 – Health Grade Assignment

[118]:
```
def assign_hr_grade(data):
    for patient in data:
        hr = patient["hr"]

        if hr < 70:
            grade = "A (Excellent)"
        elif hr <= 89:
            grade = "B (Stable)"
        elif hr <= 110:
            grade = "C (Concern)"
        else:
            grade = "D (High Risk)"

        patient["hr_grade"] = grade

    return data

graded_data = assign_hr_grade(structured_data)

for p in graded_data:
    print(p["name"], "HR:", p["hr"], "→", p["hr_grade"])
```

```
Ali Khan HR: 88 → B (Stable)
Ayesha Noor HR: 102 → C (Concern)
Usman Tariq HR: 76 → B (Stable)
Sana Fatima HR: 91 → C (Concern)
Hamza Ali HR: 110 → C (Concern)
Maryam Iqbal HR: 82 → B (Stable)
Imran Hussain HR: 97 → C (Concern)
Rida Aslam HR: 105 → C (Concern)
Hassan Raza HR: 72 → B (Stable)
Fatima Sheikh HR: 95 → C (Concern)
Bilal Khan HR: 99 → C (Concern)
Khadija Amin HR: 85 → B (Stable)
Zain Ali HR: 101 → C (Concern)
Sara Usman HR: 78 → B (Stable)
Adeel Ahmad HR: 92 → C (Concern)
Lubna Tariq HR: 87 → B (Stable)
Irfan Malik HR: 108 → C (Concern)
Raheel Khan HR: 74 → B (Stable)
Saima Noor HR: 90 → C (Concern)
Taha Salman HR: 83 → B (Stable)
Hira Asghar HR: 96 → C (Concern)
Zara Malik HR: 70 → B (Stable)
Rizwan Ali HR: 104 → C (Concern)
Shazia Ahmed HR: 88 → B (Stable)
Muhammad Iqbal HR: 76 → B (Stable)
Neha Khan HR: 91 → C (Concern)
Ali Raza HR: 107 → C (Concern)
Fahad Saleem HR: 80 → B (Stable)
Mahnoor Shah HR: 98 → C (Concern)
Ammar Tariq HR: 75 → B (Stable)
Sana Javed HR: 90 → C (Concern)
```

Q12 – Glucose Improvement (− 5%)

Description and explanation:

Assume glucose is reduced by 5% for every patient. We have to calculate updated glucose values.

1. Store the glucose value of each patient in the old_glucose in the loop.
2. Calculate its new glucose by multiplying the old_glucose by 0.95
3. And store it in the glucose key of the data.
4. At the end return the updates data.

Q12 – Glucose Improvement (−5%)

[124]:
```python
def glucose_improvement(data):
    for patient in data:
        old_glucose = patient["glucose"]
        new_glucose = old_glucose * 0.95

        patient["updated_glucose"] = round(new_glucose, 2)

    return data

updated_data = glucose_improvement(structured_data)

for p in updated_data:
    print(p["name"],
          "Old:", p["glucose"],
          "Updated:", p["updated_glucose"])
```

```
Ali Khan Old: 110 Updated: 104.5
Ayesha Noor Old: 160 Updated: 152.0
Usman Tariq Old: 95 Updated: 90.25
Sana Fatima Old: 125 Updated: 118.75
Hamza Ali Old: 180 Updated: 171.0
Maryam Iqbal Old: 140 Updated: 133.0
Imran Hussain Old: 165 Updated: 156.75
Rida Aslam Old: 172 Updated: 163.4
Hassan Raza Old: 88 Updated: 83.6
Fatima Sheikh Old: 130 Updated: 123.5
Bilal Khan Old: 155 Updated: 147.25
Khadija Amin Old: 115 Updated: 109.25
Zain Ali Old: 162 Updated: 153.9
Sara Usman Old: 100 Updated: 95.0
Adeel Ahmad Old: 138 Updated: 131.1
Lubna Tariq Old: 120 Updated: 114.0
Irfan Malik Old: 175 Updated: 166.25
Raheel Khan Old: 93 Updated: 88.35
Saima Noor Old: 143 Updated: 135.85
Taha Salman Old: 108 Updated: 102.6
Hira Asghar Old: 150 Updated: 142.5
Zara Malik Old: 85 Updated: 80.75
Rizwan Ali Old: 168 Updated: 159.6
Shazia Ahmed Old: 118 Updated: 112.1
Muhammad Iqbal Old: 97 Updated: 92.15
Neha Khan Old: 135 Updated: 128.25
Ali Raza Old: 178 Updated: 169.1
Fahad Saleem Old: 112 Updated: 106.4
Mahnoor Shah Old: 158 Updated: 150.1
Ammar Tariq Old: 92 Updated: 87.4
Sana Javed Old: 140 Updated: 133.0
Mubashir Ali Old: 170 Updated: 161.5
```

Q13 – Doctor Summary

Description and explanation:

For each doctor show:

- number of assigned patients
- average HR
- average glucose

1. Make a dictionary in which the count, total hr and total glucose will be stored.
2. Loop through every patient, If the doctor is not in the dictionary already, it will initialize its keys values to zero.
3. Then count is incremented by 1, hr is added to total_hr and glucose is added to glucose.
4. After the loop terminates, calculate average hr and glucose by dividing it to the count calculated.
5. At the end, return the dictionary.

Q13 – Doctor Summary

[127]:
```python
def doctor_summary(data):
    doctors = {}

    for patient in data:
        doc = patient["doctor"]

        if doc not in doctors:
            doctors[doc] = {
                "count": 0,
                "total_hr": 0,
                "total_glucose": 0
            }

        doctors[doc]["count"] += 1
        doctors[doc]["total_hr"] += patient["hr"]
        doctors[doc]["total_glucose"] += patient["glucose"]

    summary = {}
    for doc, info in doctors.items():
        summary[doc] = {
            "patients": info["count"],
            "avg_hr": round(info["total_hr"] / info["count"], 2),
            "avg_glucose": round(info["total_glucose"] / info["count"], 2)
        }

    return summary

doctor_report = doctor_summary(structured_data)

for doc, info in doctor_report.items():
    print(doc, "->", info)
```

```
Dr. Ahmed -> {'patients': 16, 'avg_hr': 84.75, 'avg_glucose': 122.06}
Dr. Sara -> {'patients': 13, 'avg_hr': 93.69, 'avg_glucose': 140.69}
Dr. Bilal -> {'patients': 11, 'avg_hr': 96.18, 'avg_glucose': 147.73}
Dr. Noor -> {'patients': 10, 'avg_hr': 87.2, 'avg_glucose': 127.9}
```

Q15 – Critical Condition Tag

Q14 – Senior Patients (Age ≥ 60)

Description and explanation:

List all senior patients.

1. Create a list of senior patients.
2. Loop through every patient data.
3. Check the condition age>=60.
4. If it meets the condition, append that patient name to the list.
5. At the end, return the list of senior patients.

[17]:
```python
def get_senior_patients(patients):
    seniors = []

    for p in patients:
        if p["age"] >= 60:
            seniors.append(p)

    return seniors

senior_patients = get_senior_patients(structured_data)

for p in senior_patients:
    print(f'{p["id"]}: {p["name"]} ({p["age"]} yrs) - {p["doctor"]}')

print("Total senior patients:", len(senior_patients))
```

```
2: Ayesha Noor (63 yrs) - Dr. Sara
6: Maryam Iqbal (71 yrs) - Dr. Ahmed
7: Imran Hussain (66 yrs) - Dr. Noor
11: Bilal Khan (61 yrs) - Dr. Sara
13: Zain Ali (70 yrs) - Dr. Noor
17: Irfan Malik (68 yrs) - Dr. Noor
23: Rizwan Ali (62 yrs) - Dr. Bilal
27: Ali Raza (69 yrs) - Dr. Bilal
29: Mahnoor Shah (60 yrs) - Dr. Sara
37: Saba Malik (64 yrs) - Dr. Sara
40: Waleed Khan (62 yrs) - Dr. Bilal
44: Talha Sharif (66 yrs) - Dr. Bilal
45: Madiha Khan (61 yrs) - Dr. Ahmed
Total senior patients: 13
```

Q15 – Critical Condition Tag

Description and explanation:
Mark patients with either:

- BP > 140/90
- OR feedback = "Critical"

1. BP will be first split as splitted in the earlier task and then condition will be checked like if systolic is above 140 or diastolic is above 90 or feedback is equal to critical then patient tag will be severe otherwise Normal.
2. Add the new key named ‘condition_tag’ in the existing structured data.
3. And store the tags there.
4. At the end, return the updated data.

Q15 – Critical Condition Tag

[129]:
```python
def critical_condition_tag(data):
    for patient in data:
        systolic, diastolic = patient["bp"].split("/")
        systolic = int(systolic)
        diastolic = int(diastolic)

        if systolic > 140 or diastolic > 90 or patient["feedback"] == "Critical":
            tag = "Severe"
        else:
            tag = "Normal"

        patient["condition_tag"] = tag

    return data

tagged_data = critical_condition_tag(structured_data)

for p in tagged_data:
    print(p["name"],
          "BP:", p["bp"],
          "Feedback:", p["feedback"],
          "→", p["condition_tag"])
```

```
Ali Khan BP: 120/80 Feedback: Good → Normal
Ayesha Noor BP: 145/95 Feedback: Critical → Severe
Usman Tariq BP: 118/78 Feedback: Excellent → Normal
Sana Fatima BP: 130/85 Feedback: Good → Normal
Hamza Ali BP: 150/92 Feedback: Critical → Severe
Maryam Iqbal BP: 135/88 Feedback: Fair → Normal
Imran Hussain BP: 142/90 Feedback: Fair → Severe
Rida Aslam BP: 149/94 Feedback: Critical → Severe
Hassan Raza BP: 117/79 Feedback: Excellent → Normal
Fatima Sheikh BP: 134/89 Feedback: Good → Normal
Bilal Khan BP: 140/92 Feedback: Fair → Severe
Khadija Amin BP: 128/84 Feedback: Good → Normal
Zain Ali BP: 147/93 Feedback: Critical → Severe
Sara Usman BP: 122/80 Feedback: Excellent → Normal
Adeel Ahmad BP: 130/85 Feedback: Fair → Normal
```

Q16 – Sort By Age

Description and explanation:

Sort patients from oldest to youngest.

1. Use the built-in sorted function.
2. First parameter will be data, second the key will be every patient's age, and third we want age from oldest to youngest so the sorting will be reversed.
3. Lambda function is used in the key portion which will return each patient's age.
4. At the end return the sorted patients.

Q16 – Sort By Age

[132]:
```python
def sort_by_age(data):
    return sorted(data, key=lambda p: p["age"], reverse=True)

sorted_patients = sort_by_age(structured_data)

for p in sorted_patients:
    print(p["name"], "-", p["age"])
```

```
Maryam Iqbal - 71
Zain Ali - 70
Ali Raza - 69
Irfan Malik - 68
Imran Hussain - 66
Talha Sharif - 66
Saba Malik - 64
Ayesha Noor - 63
Rizwan Ali - 62
Waleed Khan - 62
Bilal Khan - 61
Madiha Khan - 61
Mahnoor Shah - 60
Saima Noor - 59
Shazia Ahmed - 58
Uzma Raza - 58
Mubashir Ali - 57
Naima Iqbal - 56
Adeel Ahmad - 55
Rida Aslam - 54
Usman Tariq - 52
Rehan Ahmed - 52
Fahad Saleem - 51
Hammad Sheikh - 49
Sara Usman - 48
Nida Malik - 48
Hira Asghar - 47
Ayesha Khan - 46
Ali Khan - 45
Fatima Sheikh - 44
Anam Fatima - 43
Muhammad Iqbal - 42
```

Q17 – Underrated Cases

Description and explanation:

Patients having normal HR (<90) but feedback “Fair” or “Critical”.

1. Create a list of patients having hr<90 but feedback fair or critical.
2. Loop through every patient data.
3. Check the condition.
4. If True, append that patient data into the list of results.
5. At the end, return the list of underrated patient cases.

Q17 – Underrated Cases

[140]:
```
def get_underrated_cases(patients):
    result = []
    for p in patients:
        if p['hr'] < 90 and (p['feedback'] == 'Fair' or p['feedback'] == 'Critical'):
            result.append(p)
    return result

underrated_cases = get_underrated_cases(structured_data)
for p in underrated_cases:
    print(f"{p['name']} (HR: {p['hr']}, Feedback: {p['feedback']})")
```

```
Maryam Iqbal (HR: 82, Feedback: Fair)
Madiha Khan (HR: 89, Feedback: Fair)
```

Q18 – Lowercase Feedback + Count 'critical'

Description and explanation:

Convert all feedback to lowercase and count how many are “critical”

1. Make the count variable to store the count of critical patients.
2. Loop through every patient data.
3. Use the lower() function to convert the feedback into the lower case.
4. Check if the feedback is equal to ‘convert’.
5. If True, increment the count variable by 1.
6. At the end, return the count.

Q18 – Lowercase Feedback + Count 'critical'

[146]:
```
def lowercase_feedback_and_count_critical(patients):
    count_critical = 0
    for p in patients:
        p['feedback'] = p['feedback'].lower()
        if p['feedback'] == 'critical':
            count_critical += 1
    return count_critical

critical_count = lowercase_feedback_and_count_critical(structured_data)
print("Number of patients with feedback 'critical': ", critical_count)
```

```
Number of patients with feedback 'critical':  12
```

Q19 – Doctor Load Check

Description and explanation:

Verify each doctor has at least 2 patients. Return list of overworked or understaffed doctors.

1. Create a dictionary of the doctor counts.
2. Loop through every patient data.
3. Store doctor and its count in the dictionary.
4. After the loop ends, store that doctor data in the list whose patient's count is less than 2.
5. At the end return that list.

Q19 – Doctor Load Check

[149]:
```
def doctor_load_check(patients):
    doctor_counts = {}
    for p in patients:
        doctor = p['doctor']
        doctor_counts[doctor] = doctor_counts.get(doctor, 0) + 1

    understaffed_doctors = [doc for doc, count in doctor_counts.items() if count < 2]

    return understaffed_doctors

understaffed = doctor_load_check(structured_data)
print("Doctors with less than 2 patients:", understaffed)
```

```
Doctors with less than 2 patients: []
```

Q20 – Patient Summary Card

Description and explanation:
Print a one-line summary:

Ali Khan | 45 yrs | Dr. Ahmed | HR:88 | BP:120/80 | Glucose:110 | Feedback:Good

This task is very simple. Just have to print the patient's data in the given format.

1. I have used the f"----------" function.
2. By using this, inside the double quotation marks we can write the output exactly like we want. Like we don't have to use commas every time. Because if we print it the other way, many comma separated values will be used and it will get messy.
3. If we want to output the data just use curly brackets first and write the variable name or whatever.

Q20 – Patient Summary Card

[151]:
```
def patient_summary_card(patients):
    for p in patients:
        print(f"{p['name']} | {p['age']} yrs | {p['doctor']} | HR:{p['hr']} | BP:{p['bp']} | Glucose:{p['glucose']} | Feedback:{p['feedback']}")


patient_summary_card(structured_data)
```

```
Ali Khan | 45 yrs | Dr. Ahmed | HR:88 | BP:120/80 | Glucose:110 | Feedback:good
Ayesha Noor | 63 yrs | Dr. Sara | HR:102 | BP:145/95 | Glucose:160 | Feedback:critical
Usman Tariq | 52 yrs | Dr. Ahmed | HR:76 | BP:118/78 | Glucose:95 | Feedback:excellent
Sana Fatima | 29 yrs | Dr. Bilal | HR:91 | BP:130/85 | Glucose:125 | Feedback:good
Hamza Ali | 38 yrs | Dr. Sara | HR:110 | BP:150/92 | Glucose:180 | Feedback:critical
Maryam Iqbal | 71 yrs | Dr. Ahmed | HR:82 | BP:135/88 | Glucose:140 | Feedback:fair
Imran Hussain | 66 yrs | Dr. Noor | HR:97 | BP:142/90 | Glucose:165 | Feedback:fair
Rida Aslam | 54 yrs | Dr. Sara | HR:105 | BP:149/94 | Glucose:172 | Feedback:critical
Hassan Raza | 27 yrs | Dr. Ahmed | HR:72 | BP:117/79 | Glucose:88 | Feedback:excellent
Fatima Sheikh | 44 yrs | Dr. Bilal | HR:95 | BP:134/89 | Glucose:130 | Feedback:good
Bilal Khan | 61 yrs | Dr. Sara | HR:99 | BP:140/92 | Glucose:155 | Feedback:fair
Khadija Amin | 36 yrs | Dr. Ahmed | HR:85 | BP:128/84 | Glucose:115 | Feedback:good
Zain Ali | 70 yrs | Dr. Noor | HR:101 | BP:147/93 | Glucose:162 | Feedback:critical
Sara Usman | 48 yrs | Dr. Bilal | HR:78 | BP:122/80 | Glucose:100 | Feedback:excellent
Adeel Ahmad | 55 yrs | Dr. Ahmed | HR:92 | BP:130/85 | Glucose:138 | Feedback:fair
Lubna Tariq | 32 yrs | Dr. Sara | HR:87 | BP:125/83 | Glucose:120 | Feedback:good
Irfan Malik | 68 yrs | Dr. Noor | HR:108 | BP:150/95 | Glucose:175 | Feedback:critical
Raheel Khan | 40 yrs | Dr. Ahmed | HR:74 | BP:119/77 | Glucose:93 | Feedback:excellent
Saima Noor | 59 yrs | Dr. Bilal | HR:90 | BP:136/90 | Glucose:143 | Feedback:fair
Taha Salman | 25 yrs | Dr. Sara | HR:83 | BP:121/81 | Glucose:108 | Feedback:good
Hira Asghar | 47 yrs | Dr. Ahmed | HR:96 | BP:140/91 | Glucose:150 | Feedback:fair
Zara Malik | 33 yrs | Dr. Noor | HR:70 | BP:115/75 | Glucose:85 | Feedback:excellent
Rizwan Ali | 62 yrs | Dr. Bilal | HR:104 | BP:150/96 | Glucose:168 | Feedback:critical
Shazia Ahmed | 58 yrs | Dr. Sara | HR:88 | BP:129/84 | Glucose:118 | Feedback:good
Muhammad Iqbal | 42 yrs | Dr. Ahmed | HR:76 | BP:116/78 | Glucose:97 | Feedback:excellent
Neha Khan | 37 yrs | Dr. Noor | HR:91 | BP:132/86 | Glucose:135 | Feedback:good
Ali Raza | 69 yrs | Dr. Bilal | HR:107 | BP:152/97 | Glucose:178 | Feedback:critical
Fahad Saleem | 51 yrs | Dr. Ahmed | HR:80 | BP:124/82 | Glucose:112 | Feedback:good
Mahnoor Shah | 60 yrs | Dr. Sara | HR:98 | BP:141/93 | Glucose:158 | Feedback:fair
Ammar Tariq | 30 yrs | Dr. Noor | HR:75 | BP:118/78 | Glucose:92 | Feedback:excellent
Sana Javed | 34 yrs | Dr. Bilal | HR:90 | BP:130/88 | Glucose:140 | Feedback:fair
```