1.1.1. Ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 读取数据集
data = pd.read_csv('patient_data.csv')
```

```
# 1. 统计住院天数超过7天的患者数量及其占比
# 创建新列'RiskLevel'，根据住院天数判断风险等级
data['RiskLevel'] = np.where(data['DaysInHospital']>7, '高风险患者', '低风险患者')
# 统计不同风险等级的患者数量
risk_counts = data['RiskLevel'].value_counts()
# 计算高风险患者占比
high_risk_ratio = risk_counts['高风险患者'] /  len(data)
# 计算低风险患者占比
low_risk_ratio = risk_counts['低风险患者'] / len(data)

# 输出结果
print("高风险患者数量:", risk_counts['高风险患者'])
print("低风险患者数量:", risk_counts['低风险患者'])
print("高风险患者占比:", high_risk_ratio)
print("低风险患者占比:", low_risk_ratio)
```

```
# 2. 统计不同BMI区间中高风险患者的比例和患者数
# 定义BMI区间和标签
bmi_bins = [0, 18.5, 24.9, 29.9, np.inf]
bmi_labels = ['低于18.5', '18.5～24.9', '25.0～29.9', '高于30.0']
# 根据BMI值分配BMI区间标签
data['BMIRange'] = pd.cut(data['BMI'], bins=bmi_bins, labels=bmi_labels)
# 计算每个BMI区间中高风险患者的比例
bmi_risk_rate = data.groupby('BMIRange')['RiskLevel'].apply(lambda x: (x == '高风险患者').mean())
# 统计每个BMI区间的患者数量
bmi_patient_count = data['BMIRange'].value_counts()

# 输出结果
print("BMI区间中高风险患者的比例和患者数:")
print(bmi_risk_rate)    #高风险患者的比例
print(bmi_patient_count)    #高风险患者的患者数
```

```
# 3. 统计不同年龄区间中高风险患者的比例和患者数
# 定义年龄区间和标签
age_bins = [0, 25, 35, 45, 55, 65, np.inf]
age_labels = ['低于25岁', '26岁-35岁', '36岁-45岁', '46岁-55岁', '56岁-65岁', '高于65岁']
# 根据年龄值分配年龄区间标签
data['AgeRange'] = pd.cut(data['Age'], bins=age_bins, labels=age_labels)
# 计算每个年龄区间中高风险患者的比例
age_risk_rate = data.groupby('AgeRange')['RiskLevel'].apply(lambda x: (x == '高风险患者').mean(
# 统计每个年龄区间的患者数量
age_patient_count = data['AgeRange'].value_counts()

# 输出结果
print("年龄区间中高风险患者的比例和患者数:")
print(age_risk_rate)   #高风险患者的比例
print(age_patient_count)   #高风险患者的患者数
```

1.1.2. Ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 读取数据集
data = pd.read_csv('sensor_data.csv')
```

```
# 1. 传感器数据统计
# 对传感器类型进行分组，并计算每个组的数据数量和平均值
sensor_stats = data.groupby('SensorType')['Value'].agg(['count', 'mean'])
# 输出结果
print("传感器数据数量和平均值:")
print(sensor_stats)
```

```
# 2. 按位置统计温度和湿度数据
# 筛选出温度和湿度数据，然后按位置和传感器类型分组，计算每个组的平均值
location_stats = data[data['SensorType'].isin(['Temperature', 'Humidity'])].groupby(['Location', 'SensorType'])['Value'].mean().unstack()
# 输出结果
print("每个位置的温度和湿度数据平均值:")
print(location_stats)
```

```
# 3. 数据清洗和异常值处理
# 标记异常值
data['is_abnormal'] = np.where(
    ((data['SensorType'] == 'Temperature') & ((data['Value'] < -10) | (data['Value'] > 50))) |
    ((data['SensorType'] == 'Humidity') & ((data['Value'] < 0) | (data['Value'] > 100))),
    True, False
)
# 输出异常值数量
print("异常值数量:", data['is_abnormal'].sum())
# 填补缺失值
# 使用前向填充和后向填充的方法填补缺失值
data['Value'].fillna(method='ffill', inplace=True)
data['Value'].fillna(method='bfill', inplace=True)
# 保存清洗后的数据
# 删除用于标记异常值的列，并将清洗后的数据保存到新的CSV文件中
cleaned_data = data.drop(columns=['is_abnormal'])
cleaned_data.to_csv('cleaned_sensor_data.csv', index=False)
print("数据清洗完成，已保存为 'cleaned_sensor_data.csv'")
```

1.1.3. Ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 读取数据集
data = pd.read_csv('credit_data.csv')
```

```
# 1. 数据完整性审核
missing_values = data.isnull().sum()        #数据缺失值统计 2分
duplicate_values = data.duplicated().sum()      #数据重复值统计 2分
# 输出结果
print("缺失值统计:")
print(missing_values)
print("重复值统计:")
print(duplicate_values)
```

```
# 2. 数据合理性审核
data['is_age_valid'] =data['Age'].between(18, 70)            #Age数据的合理性审核 2分
data['is_income_valid'] =data['Income'] > 2000          #Income数据的合理性审核 2分
data['is_loan_amount_valid'] =data['LoanAmount'] < (data['Income'] * 5)    #LoanAmount数据的合理性审核 2分
data['is_credit_score_valid'] =data['CreditScore'].between(300, 850)   #CreditScore数据的合理性审核 2分
# 合理性检查结果
validity_checks = data[['is_age_valid', 'is_income_valid', 'is_loan_amount_valid', 'is_credit_score_valid']].all(axis=1)
data['is_valid'] = validity_checks
# 输出结果
print("数据合理性检查:")
print(data[['is_age_valid', 'is_income_valid', 'is_loan_amount_valid', 'is_credit_score_valid', 'is_valid']].describe())
```

```
# 3. 数据清洗和异常值处理
# 标记不合理数据
invalid_rows = data[~data['is_valid']]
# 删除不合理数据行
cleaned_data = data[data['is_valid']]
# 删除标记列
cleaned_data = cleaned_data.drop(columns=['is_age_valid', 'is_income_valid', 'is_loan_amount_valid', 'is_credit_score_valid', 'is_valid'])
# 保存清洗后的数据
cleaned_data.to_csv('cleaned_credit_data.csv', index=False)
print("数据清洗完成，已保存为 'cleaned_credit_data.csv'")
```

1.1.4. Ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. 数据采集
# 从本地文件中读取数据  2分
data =  pd.read_csv('user_behavior_data.csv')
print("数据采集完成，已加载到DataFrame中")

# 打印数据的前5条记录  2分
print(data.head())
```

```
# 2. 数据清洗与预处理
# 处理缺失值  2分
data = data.dropna()

# 数据类型转换
data['Age'] =data['Age'].astype(int)   # Age数据类型转换 2分
data['PurchaseAmount'] = data['PurchaseAmount'].astype(float) # PurchaseAmount数据类型转换  2分
data['ReviewScore'] = data['ReviewScore'].astype(int)  # ReviewScore数据类型转换 2分

# 处理异常值  2分
data = data[(data['Age'].between(18, 70)) &
            (data['PurchaseAmount'] > 0) &
            (data['ReviewScore'].between(1, 5))]

# 数据标准化
data['PurchaseAmount'] = (data['PurchaseAmount'] - data['PurchaseAmount'].mean()) / data['PurchaseAmount'].std()  # PurchaseAmount数据标准化
data['ReviewScore'] = (data['ReviewScore'] - data['ReviewScore'].mean()) / data['ReviewScore'].std()   # ReviewScore数据标准化 2分

# 保存清洗后的数据  1分
data.to_csv('cleaned_user_behavior_data.csv', index=False)
print("数据清洗完成，已保存为 'cleaned_user_behavior_data.csv'")
```

```
# 3. 数据统计
# 统计每个购买类别的用户数
purchase_category_counts = data['PurchaseCategory'].value_counts()
print("每个购买类别的用户数:\n", purchase_category_counts)

# 统计不同性别的平均购买金额
gender_purchase_amount_mean = data.groupby('Gender')['PurchaseAmount'].mean()
print("不同性别的平均购买金额:\n", gender_purchase_amount_mean)

# 统计不同年龄段的用户数
bins = [18, 25, 35, 45, 55, 65, 70]
labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '65+']
data['AgeGroup'] = pd.cut(data['Age'], bins=bins, labels=labels, right=False)
age_group_counts = data['AgeGroup'].value_counts().sort_index()
print("不同年龄段的用户数:\n", age_group_counts)
```

1.1.5. Ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. 数据采集
# 从本地文件中读取数据  2分
data = pd.read_csv('vehicle_traffic_data.csv')
print("数据采集完成，已加载到DataFrame中")

# 打印数据的前5条记录
print(data.head())
```

```
# 2. 数据清洗与预处理
# 处理缺失值  2分
data = data.dropna()

# 数据类型转换
data['Age'] = data['Age'].astype(int)                #Age数据类型转换 1分
data['Speed'] = data['Speed'].astype(float)            #Speed数据类型转换 1分
data['TravelDistance'] = data['TravelDistance'].astype(float)  #TravelDistance数据类型转换 1分
data['TravelTime'] = data['TravelTime'].astype(float)     #TravelTime数据类型转换 1分

# 处理异常值  2分
data = data[(data['Age'].between(18, 70))  &
            (data['Speed'].between(0, 200)) &
            (data['TravelDistance'].between(1, 1000)) &
            (data['TravelTime'].between(1, 1440))]

# 保存清洗后的数据  1分
data.to_csv('cleaned_vehicle_traffic_data.csv', index=False)
print("数据清洗完成，已保存为 'cleaned_vehicle_traffic_data.csv'")
```

```
# 3. 数据合理性审核
# 审核字段合理性 1分
unreasonable_data = data[~((data['Age'].between(18, 70)) &
                           (data['Speed'].between(0, 200)) &
                           (data['TravelDistance'].between(1, 1000)) &
                           (data['TravelTime'].between(1, 1440)))]
print("不合理的数据:\n", unreasonable_data)

# 4. 数据统计
# 统计每种交通事件的发生次数  2分
traffic_event_counts = data['TrafficEvent'].value_counts()
print("每种交通事件的发生次数:\n", traffic_event_counts)

# 统计不同性别的平均车速、行驶距离和行驶时间  2分
gender_stats = data.groupby('Gender').agg({'Speed': 'mean', 'TravelDistance': 'mean', 'TravelTime': 'mean'})
print("不同性别的平均车速、行驶距离和行驶时间:\n", gender_stats)

# 统计不同年龄段的驾驶员数  2分
age_bins = [18, 25, 35, 45, 55, 65, 70]
age_labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '66-70']
data['AgeGroup'] = pd.cut(data['Age'], bins=age_bins, labels=age_labels, right=False)
age_group_counts = data['AgeGroup'].value_counts()
print("不同年龄段的驾驶员数:\n", age_group_counts)
```