**TITLE – BRAZILIAN MEDICAL APPOINTMENT REPORT DATA ANALYSIS REPORT**

DATASET DESCRPITION:

The dataset is about medical appointments in BRAZIL, a sample over 100k the information of this dataset is collected from KAGGLE.

Tools Used: Python (Pandas), Numpy.  
  
This report details the data cleaning and preparation process for the Brazilian medical No-Show Appointments Dataset, which contains medical appointment records with patient demographics, appointment scheduling details, and whether the patient attended ("No") or missed ("Yes") their appointment.   
  
OBJECTIVES:

* Identify and handle missing/incorrect data
* Remove duplicates
* Standardize text and date formats
* Rename columns for clarity
* Convert data types appropriately
* Engineer new features for better analysis

INITIAL DATA ASSESSMENT:  
  
Dataset Overview 

* Rows: [110](tel:110),[527](tel:527) (before cleaning)
* Columns: 14

Key Variables:

* Patient Id, Appointment ID (unique identifiers)
* Gender (M/F)
* Scheduled Day, Appointment Day (timestamps)
* Age (numeric)
* Neighbourhood (location)
* Medical conditions (`Scholarship, Hipertension, Diabetes, Alcoholism, Handcap)
* SMS\_received (if a reminder was sent)
* No-show (target variable: "Yes" if missed, "No" if attended)

INITIAL DATA QUALITY ISSUES

Issue Example Impact

Inconsistent column naming (Hipertension, Handcap) Hipertension (misspelled) Harder to interpret    
Text case inconsistencies (`No-show` values) "No" vs. "no" Affects grouping   
Date columns as strings [2016-04-29](tel:2016-04-29)T18:38:08Z Needs datetime conversion   
Possible age outliers Negative ages, extreme values Skews analysis   
Potential duplicates Same `PatientId` & `AppointmentID` Overcounting

DATA CLEANING VALUES   
  
Handling Missing Values  
✅ Check:  
python -  
print([df.isnull](http://df.isnull)().sum())  
  
 Result: No missing values found.   
  
Removing Duplicates  
✅ Action:  
python -   
df.drop\_duplicates(inplace=True)  
  
 Result: 4 duplicate rows removed  
 Final count: [110](tel:110),[523](tel:523) rows

STANDARDIZING TEXT VALUES  
  
✅ Actions:  
1. Gender: Convert to uppercase (`M`/`F`)   
python  
   df['Gender'] = df['Gender'].[str.upper](http://str.upper)()  
     
2. No-show: Standardize to `Yes`/`No`

python  
   df['No-show'] = df['No-show'].[str.title](http://str.title)()

COLUMN RENAMING & FORMATTING   
  
✅ Actions:  
Original Column Cleaned Column Reason   
PatientId patient\_id Snake case, lowercase   
AppointmentID appointment\_id Snake case, lowercase   
ScheduledDay scheduled\_day Snake case, lowercase    
AppointmentDay appointment\_day Snake case, lowercase   
Hipertension hypertension Correct spelling    
Handcap handicap More accurate term    
No-show no\_show Snake case    
  
python  
[df.columns](http://df.columns) = [df.columns.str.lower](http://df.columns.str.lower)()  
[df.rename](http://df.rename)(columns={  
    'patientid': 'patient\_id',  
    'appointmentid': 'appointment\_id',  
    'scheduledday': 'scheduled\_day',  
    'appointmentday': 'appointment\_day',  
    'hipertension': 'hypertension',  
    'handcap': 'handicap',  
    'no-show': 'no\_show'  
}, inplace=True)   
  
DATE FORMATTING

✅ Actions:  
- Convert `scheduled\_day` and `appointment\_day` to `datetime`   
- Extract useful temporal features:   
  - Day of the week (for no-show trends)   
  - Time between scheduling & appointment

python  
df['scheduled\_day'] = pd.to\_datetime(df['scheduled\_day'])  
df['appointment\_day'] = pd.to\_datetime(df['appointment\_day'])  
  
Days between scheduling & appointment  
df['days\_between'] = (df['appointment\_day'] - df['scheduled\_day']).[dt.days](http://dt.days)  
df['days\_between'] = df['days\_between'].apply(lambda x: 0 if x < 0 else x)  # Fix negative values  
  
Day of week  
df['appointment\_dow'] = df['appointment\_day'].dt.day\_name()  
df['scheduled\_dow'] = df['scheduled\_day'].dt.day\_name()

HANDLING OUTLIERS (AGE)

✅ Actions:   
- Negative ages → Set to `0`  
- Unrealistic ages (e.g., >[110](tel:110)) → Capped at `[110](tel:110)`  
- Age groups created for better analysis  
  
python  
df['age'] = df['age'].apply(lambda x: 0 if x < 0 else x)  
df['age'] = df['age'].apply(lambda x: [110](tel:110) if x > [110](tel:110) else x)  
  
Age groups  
bins = [0, 12, 19, 30, 50, 65, [110](tel:110)]  
labels = ['Child', 'Teen', 'Young Adult', 'Adult', 'Middle Aged', 'Senior']  
df['age\_group'] = [pd.cut](http://pd.cut)(df['age'], bins=bins, labels=labels, right=False)

DATA TYPE CONVERSION  
  
✅ Actions: 

Column Original Type New Type Reason    
  
scholarship int64 int8 Optimize memory   
hypertension int64 int8 Optimize memory   
diabetes int64 int8 Optimize memory   
alcoholism int64 int8 Optimize memory   
handicap int64 int8 Optimize memory   
sms\_received int64 int8 Optimize memory   
age int64 int8 Optimize memory   
  
python  
numeric\_cols = ['scholarship', 'hypertension', 'diabetes', 'alcoholism', 'handicap', 'sms\_received']  
df[numeric\_cols] = df[numeric\_cols].astype('int8')  
df['age'] = df['age'].astype('int8')  
   
  
FUTURE DATA STRUCTURE

Cleaned Columns 

Column Description Type   
  
patient\_id Unique patient ID int64    
appointment\_id Unique appointment ID int64    
gender Patient gender (M/F) object    
scheduled\_day When appointment was scheduled datetime    
appointment\_day When appointment occurred datetime   
age Patient age ([0-110](tel:0-110)) int8 neighbourhood Location of appointment object    
scholarship Welfare program (0/1) int8    
hypertension Hypertension status (0/1) int8    
diabetes Diabetes status (0/1) int8    
alcoholism Alcoholism status (0/1) int8    
handicap Handicap status (0/1) int8 sms\_received SMS reminder sent (0/1) int8    
no\_show Missed appointment (Yes/No) object   
days\_between Days between scheduling & appointment int64   
age\_group Age category (Child, Teen, etc.) category   
appointment\_dow Day of week (Monday-Sunday) object    
scheduled\_dow Day of week when scheduled object

Key Improvement

* No missing data
* No duplicates
* Consistent naming & formatting
* Optimized data types (memory-efficient)
* New features (`days\_between`, `age\_group`)

Next Steps (Exploratory Analysis Ideas)  
No-show rate by:  
  - Gender   
  - Age group   
  - Neighbourhood   
  - Days between scheduling & appointment   
  - SMS reminders 

Trends over time:  
  - Are no-shows higher on certain days?   
  - Does lead time affect attendance?   
  
  
CONCLUSION  
The cleaned dataset is now ready for analysis, with:   
✅ Consistent formatting  
✅ Correct data types   
✅ Removed duplicates   
✅ New useful features

Cleaned Dataset: 109529 rows, 14 columns, new dataset 19 columns.

Ready for further analysis and data modeling.

Future Additions:

* EDA
* Predictive modeling
* Dashboard visualization of datasets

Prepared by: [HARINIPRIYASELVAM]   
Date: [21/04/2025]