```
import pandas as pd
import json
from datetime import datetime

users_file = "/content/drive/MyDrive/Colab Notebooks/Fetch OA/users.json"
receipts_file = "/content/drive/MyDrive/Colab Notebooks/Fetch OA/receipts.json"
brands_file = "/content/drive/MyDrive/Colab Notebooks/Fetch OA/brands.json"

def load_json(file_path):
    if file_path.endswith(".gz"):
        return pd.read_json(file_path, compression="gzip", lines=True)
    else:
        with open(file_path, "r") as file:
            data = [json.loads(line) for line in file]
        return pd.json_normalize(data)

# Load datasets
users_df = load_json(users_file)
receipts_df = load_json(receipts_file)
brands_df = load_json(brands_file)


missing_values = {
    "Users": users_df.isnull().sum(),
    "Receipts": receipts_df.isnull().sum(),
    "Brands": brands_df.isnull().sum()
}
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
{'Users': active                0
role                 0
signUpSource        48
state               56
_id.$oid             0
createdDate.$date    0
lastLogin.$date     62
dtype: int64, 'Receipts': bonusPointsEarned          575
bonusPointsEarnedReason    575
pointsEarned               510
purchasedItemCount         484
rewardsReceiptItemList     440
rewardsReceiptStatus         0
totalSpent                 435
userId                       0
_id.$oid                     0
createDate.$date             0
dateScanned.$date            0
finishedDate.$date         551
modifyDate.$date             0
pointsAwardedDate.$date    582
purchaseDate.$date         448
dtype: int64, 'Brands': barcode           0
category        155
categoryCode    650
name              0
topBrand        612
_id.$oid          0
cpg.$id.$oid      0
cpg.$ref          0
brandCode       234
dtype: int64}
```

```
list_columns = [col for col in receipts_df.columns if receipts_df[col].apply(lambda x: isinstance(x, list)).any()]
receipts_no_lists = receipts_df.drop(columns=list_columns, errors='ignore')
duplicate_counts = {
    "Users": users_df.duplicated().sum(),
    "Receipts": receipts_no_lists.duplicated().sum(),
    "Brands": brands_df.duplicated().sum()
}
print("\nDuplicate Records:")
print(duplicate_counts)
```

```

Duplicate Records:
{'Users': 283, 'Receipts': 0, 'Brands': 0}
```

```
receipts_df["totalSpent"] = pd.to_numeric(receipts_df["totalSpent"], errors='coerce')
invalid_total_spent = receipts_df[(receipts_df["totalSpent"] < 0) | receipts_df["totalSpent"].isnull()]
print("\nInvalid Total Spent:")
print(invalid_total_spent.shape[0])
```

```

Invalid Total Spent:
435
```

```
current_date = datetime.now().timestamp() * 1000
receipts_df["purchaseDate.$date"] = pd.to_numeric(receipts_df["purchaseDate.$date"], errors='coerce')
invalid_dates = receipts_df[receipts_df["purchaseDate.$date"] > current_date]
print("\nFuture Purchase Dates:")
print(invalid_dates.shape[0])
```

```

Future Purchase Dates:
0
```

```
orphaned_receipts = receipts_df[~receipts_df["userId"].isin(users_df["_id.$oid"])]
print("\nOrphaned Receipts (User Mismatch):")
print(orphaned_receipts.shape[0])
```

```

Orphaned Receipts (User Mismatch):
148
```

```
receipts_barcodes = receipts_df.explode("rewardsReceiptItemList")["rewardsReceiptItemList"].apply(
    lambda x: x.get("barcode") if isinstance(x, dict) else None
)
orphaned_brands = brands_df[~brands_df["barcode"].isin(receipts_barcodes.dropna())]
print("\nOrphaned Brands (Unused Barcodes):")
print(orphaned_brands.shape[0])
```

```

Orphaned Brands (Unused Barcodes):
1150
```