# Unit Test Report

**Unit tests**

- Module: python_to_postgres.py
    - Test: Insert to wardrobe database:
        - Equivalent classes: list of integers U list of strings U null value
        - Test cases:{ (1,2,3,4,4) U ('Brih')}
    - Test: Delete from wardrobe database
        - Equivalent classes: list of integers U list of strings U NULL
        - Test cases: { (1,2,3,4,4) U ('Brih')}
    - Test: Get from wardrobe database
        - Equivalent classes: * U
- Module: Recommender_tests.py
    - Test: Ensure that the recommender can be loaded :
        - Equivalent classes: {non existent file U non path string U existing file path equal to number of rows in csv file }
        - Test cases: { (./csv_files/outfits.csv) -> 108 U ((./nonexistant.csv) -> Error}
    - Test: Checks that Recommender handles a failure to load
        - Equivalent classes :{valid csv input file U invalid csv file}
        - Test cases : {(No dataframe) -> Error}
    - Test: Assure clothing data is normalized:
        - Equivalent classes: {dataframe with pieceIDs U empty dataframe}
        - Test cases: { (dataframe) -> ['hat', 'shirt', 'sweater', 'jacket', 'bottom_layer', 'shoes', 'misc']}
    - Test: Ensure that a model can be saved and then loaded again:
        - Equivalent classes: {non existent machine learning model U existing machine learning model}
        - Test cases: { ("wavestyled") -> wavestyled model is saved}
    - Test: Ensure the model can train with the data
        - Equivalent classes : {Valid encoded dataframe, invalid dataframe}
        - Test cases: { (./csv_files/outfits.csv) -> 108}
    - Test: Evaluate whether the model can generate recommendations:
        - Equivalent classes: {valid data for the model U invalid data U non existent data }
        - Test cases: { (./csv_files/outfits.csv) -> 0/1 labels, valid fit/weather/occasion }
- Module: wardrobe_test.py
    - Test: Ensure a wardrobe can be loaded from csv file:
        - Equivalent classes: {}
        - Test cases: { (./csv_files/outfits.csv).length = 156 }

# Unit Test Report

  - Test: Test the initialization of a wardrobe object:
    - Equivalent classes: {Valid wardrobe initialization code}
    - Test cases: { Obj(Wardrobe).length = 0 }
  - Test: Test adding an item to the wardrobe
    - Equivalent Classes: {Valid wardrobe object U Invalid Wardrobe object}
    - Test Cases: {Obj(Wardrobe).length = 157 }
  - Test: Test deleting an item from the wardrobe
    - Equivalent Classes: {Valid pieceid U Invalid pieceid}
    - Test Cases: {Obj(Wardrobe).length = 156 & Obj(Wardrobe).get(deletedID) ==-1}
  - Test: Try deleting an non existent item from the wardrobe
    - Equivalent Classes: {Invalid pieceid U Invalid pieceid}
    - Test Cases: {Obj(Wardrobe).length = Obj(Wardrobe).length & Obj(Wardrobe).get(pieceid) = None}
  - Test: Test updating an item in the wardrobe
    - Equivalent Classes: {Valid Wardrobe Object U Invalid Wardrobe Object}
    - Test Cases: {Obj(Wardrobe).get(item ID) == updated item}
  - Test: Test wardrobe outfit generation
    - Equivalent Classes: {Valid Occasion & Valid Weather U Valid Occasion & Invalid Weather U Valid Weather & Invalid Occasion U Invalid Weather & Invalid Occasion}
    - Test Cases: {Obj(Wardrobe).gen(oc, we).length = 7 (size of fit) & Obj(Wardrobe).gen(oc,we) are all integers and pieceids in the dataframe}
  - Test: Test wardrobe get random fits
    - Equivalent Classes: {Valid Occasion & Valid Weather U Either Weather or Occasion Invalid U Valid Fits U Invalid Fits}
    - Test Cases: {Obj(Wardrobe).genRandom()[1] has valid occasions and weathers, Obj(Wardrobe).genRandom()[0] has valid fits that are all of size 7 and have valid pieceids}
- Module: user_tests.py
  - Test: Test users can be generated
    - Equivalent Classes: {Valid User Wardrobe U Invalid User Wardrobe}
    - Test Cases: {Obj(Wardrobe).length == 156}
  - Test: wardrobe Operations
    - Test: Test add operation
      - Equivalent Classes: {Valid Wardrobe Object U Invalid Wardrobe Object}
      - Test Cases: {Obj(Wardrobe).length == 157}
    - Test: Test update operation

# Unit Test Report

    - Equivalent Classes: {Valid Wardrobe Object U Invalid Wardrobe Object }
    - Test Cases: {Obj(Wardrobe).get(item ID) == updated item}
  - Test: Test delete operation
    - Equivalent Classes: {Valid Wardrobe Object U Invalid Wardrobe Object}
    - Test Cases: {Obj(Wardrobe).get(PieceID) == None and Obj(Wardrobe).length == 156}
- Test: User Calibration
  - Equivalent Classes: {Valid Ratings U Invalid Ratings U Valid Attributes U Invalid Attributes U Valid Fits U Invalid Fits U Valid Recommender U Invalid Recommender}
  - Test Cases: {Obj(User).RecommenderNEW.length == Obj(User).RecommenderOLD.length + number of items added & All Ratings are 1s or 0s & All Attributes are Valid mappings & All Fits are integers and valid pieceids }
- Test: Load the UserBase Object
  - Equivalent Classes: {Valid User Base Object U Invalid User Base Object}
  - Test Cases: {add User to UserBase and check that UserBase.length == 1 and UserBase.get(UserID) works}