

Collinson Test

Bunmi Osula



October 18, 2020

UDS

Contents

[INTRODUCTION 2](#_Toc53946358)

[SQL TO CREATE TABLES 2](#_Toc53946359)

[VENDING MACHINE CODE WALKTHROUGH 8](#_Toc53946360)

[API Automation 18](#_Toc53946361)

# INTRODUCTION

This document is created as a walkthrough for the Vending Machine program written in PL/SQL and API Automation written in Java. Comments will be written to reflect what each section of code does. Screenshots of each section of code will be taken and explained in the table below:

Beverage.txt script attached can be run from SQL Plus prompt if converted to .sql extension.

The tables created are the tables that the procedures will be reading from and populating.

# SQL TO CREATE TABLES

|  |  |  |
| --- | --- | --- |
| Steps | SQL QUERY | EXPLANATION |
| 1. | 1. conn system/Passwordv7@OIDDB2 2. grant connect, resource to Bunmi identified by Osula; 3. conn Bunmi/Osula@OIDDB2 | 1. Connects as System with privilege users.  2. Create or/and add Privilege to user Bunmi.  3. Connects as the new user Bunmi to run this script |
| 2. | DROP SEQUENCE TRANSACTION\_SEQ;  CREATE SEQUENCE TRANSACTION\_SEQ  START WITH 1  INCREMENT BY 1  MINVALUE 1  MAXVALUE 999999  ORDER  CYCLE  ; | This SQL code drops sequence TRANSACTION\_SEQ if it exist and creates a new sequence TRANSACTION\_SEQ starting at 1 and being incremented every time it is called. |
| 3. | DROP TABLE BEVERAGE\_INVENTORY CASCADE CONSTRAINTS;  CREATE TABLE BEVERAGE\_INVENTORY  (BEVERAGE\_NO number(4) Primary key  ,BEVERAGE\_NAME varchar2(12)  ,BEVERAGE\_TYPE varchar2(12)  ,BEVERAGE\_TRANS\_ID number(4)  ,BEVERAGE\_PRICE number(7,2)  ,TOTAL\_DAY\_SOLD number(6)  ,BEVERAGE\_AVAIL\_NO number(6)  )  ; | This SQL code drops table BEVERAGE\_INVENTORY with all its constraints and referential integrity and creates a new table BEVERAGE\_INVENTORY. |
| 4. | --Level 1 Tables  DROP TABLE TRANSACTION\_TABLE CASCADE CONSTRAINTS;  create TABLE TRANSACTION\_TABLE  (TRANSACTION\_ID number(4) Primary key  ,AMOUNT\_INCOMING number(7,2)  ,TRANS\_STATUS varchar2(12)  ,REFUND\_AMT number(7,2)  ,TOTAL\_AMOUNT number(7,2)  ,PRIOR\_DAY\_SOLD number(6)  ,TRANS\_DATE date  ,START\_TRANSTIME varchar2(12)  ,END\_TRANSTIME varchar2(12)  ,TRANS\_BEVERAGE\_NO number(4) References BEVERAGE\_INVENTORY(BEVERAGE\_NO)  ,CHANGE\_AMOUNT number(7,2))  ; | This SQL code drops table TRANSACTION\_TABLE with all its constraints and referential integrity and creates a new table TRANSACTION\_TABLE. |
| 5. | DROP TABLE INCOMING\_FUNDS CASCADE CONSTRAINTS;  CREATE TABLE INCOMING\_FUNDS  (INCOMING\_TRANS\_ID number(4) references TRANSACTION\_TABLE(TRANSACTION\_ID)  ,TRANS\_SINGLE\_FUND number(7,2)  ,TRANS\_SUM\_FUNDS number(7,2)  ,INC\_TRANSFUNDS\_DATE Date  ,CHANGE\_AMOUNT number(7,2)  ,INC\_TRANS\_STATUS varchar2(12)  )  ; | This SQL code drops table INCOMING\_FUNDS with all its constraints and references and creates a new table INCOMING\_FUNDS. |
| 6. | DROP TABLE INCOMING\_FUNDS\_TEMP CASCADE CONSTRAINTS;  CREATE TABLE INCOMING\_FUNDS\_TEMP  (INCOMING\_TRANS\_ID number(4) references TRANSACTION\_TABLE(TRANSACTION\_ID)  ,TRANS\_SINGLE\_FUND number(7,2)  )  ; | This SQL code drops table INCOMING\_FUNDS\_TEMP with all its constraints and references and creates a new table INCOMING\_FUNDS\_TEMP. |
| 7. | create or replace Trigger TRIGG\_SLOT\_DROP\_COINS  AFTER INSERT on INCOMING\_FUNDS\_TEMP  for each row  DECLARE  V\_INCOMING\_TRANS\_ID INCOMING\_FUNDS.INCOMING\_TRANS\_ID%TYPE;  V\_TRANS\_SINGLE\_FUND number(7,2) :=0;  V\_TRANS\_SUM\_FUNDS number(7,2) := 0;  V\_INC\_TRANSFUNDS\_DATE Date;  V\_CHANGE\_AMOUNT number(7,2) :=0;  V\_INC\_TRANS\_STATUS varchar2(12) ;  V\_AMOUNT\_INCOMING number(7,2) ;  V\_TRANS\_STATUS varchar2(12);  V\_REFUND\_AMT number(7,2);  V\_TOTAL\_AMOUNT number(7,2);  V\_PRIOR\_DAY\_SOLD number(6);  V\_END\_TRANSTIME varchar2(12);  V\_BEVERAGE\_NO number(4) ;  V\_BEVERAGE\_PRICE number(7,2);  V\_TOTAL\_DAY\_SOLD number(6);  V\_BEVERAGE\_AVAIL\_NO number(6);    V\_penny number(2) := 1;  V\_nickel number(2) := 5;  V\_dime number(2) := 10;  V\_quarter number(2) := 25; | This PL/SQL code creates Trigger TRIGG\_SLOT\_DROP\_COINS if does not already exist and replaces the trigger if it exist.  The variables are declared in the section with their data type. |
| 8. | BEGIN  IF :NEW.TRANS\_SINGLE\_FUND not IN(V\_penny,V\_nickel,V\_dime,V\_quarter) THEN  -- To check coins dropped in Slot.  RAISE\_APPLICATION\_ERROR(-20534,'Accepting only coins of 1,5,10,25 Cents');  RAISE\_APPLICATION\_ERROR(-20535,'i.e. penny, nickel, dime, and quarter.');  ELSE  --Need To Trap the latest Transaction ID from The Transaction Table  --The Transaction Table is triggered by the Customer Selection  -- and is the first table to be populated  --The Sequence Transaction\_seq generates the Transaction ID from the Machine  -- on the Transaction  V\_TRANS\_SINGLE\_FUND := :NEW.TRANS\_SINGLE\_FUND;  SELECT max(TRANSACTION\_ID)  into V\_INCOMING\_TRANS\_ID  from  TRANSACTION\_TABLE  where TRANS\_DATE > SYSDATE-1;  -- Collecting information to declared Variables from Transaction\_Table  select  TRANSACTION\_ID,TOTAL\_AMOUNT  ,TRANS\_DATE, CHANGE\_AMOUNT, TRANS\_STATUS, TRANS\_BEVERAGE\_NO  into  V\_INCOMING\_TRANS\_ID,V\_TRANS\_SUM\_FUNDS  ,V\_INC\_TRANSFUNDS\_DATE, V\_CHANGE\_AMOUNT, V\_INC\_TRANS\_STATUS, V\_BEVERAGE\_NO  From TRANSACTION\_TABLE  where TRANSACTION\_ID = V\_INCOMING\_TRANS\_ID ;  -- Inserting every single Coin drop into the table Incoming Funds  INSERT into INCOMING\_FUNDS  (INCOMING\_TRANS\_ID,TRANS\_SINGLE\_FUND,TRANS\_SUM\_FUNDS  ,INC\_TRANSFUNDS\_DATE, CHANGE\_AMOUNT, INC\_TRANS\_STATUS)  values  (V\_INCOMING\_TRANS\_ID,V\_TRANS\_SINGLE\_FUND,V\_TRANS\_SUM\_FUNDS  ,V\_INC\_TRANSFUNDS\_DATE, V\_CHANGE\_AMOUNT, V\_INC\_TRANS\_STATUS  ) ;  select sum(TRANS\_SINGLE\_FUND) into V\_TRANS\_SUM\_FUNDS  from INCOMING\_FUNDS  where INCOMING\_TRANS\_ID = V\_INCOMING\_TRANS\_ID;  update INCOMING\_FUNDS  set TRANS\_SUM\_FUNDS = V\_TRANS\_SUM\_FUNDS  where INCOMING\_TRANS\_ID = V\_INCOMING\_TRANS\_ID;  ------  select  BEVERAGE\_PRICE ,TOTAL\_DAY\_SOLD ,BEVERAGE\_AVAIL\_NO  into  V\_BEVERAGE\_PRICE, V\_TOTAL\_DAY\_SOLD ,V\_BEVERAGE\_AVAIL\_NO  from BEVERAGE\_INVENTORY  where BEVERAGE\_NO=V\_BEVERAGE\_NO;  /\*  V\_TOTAL\_DAY\_SOLD:=V\_TOTAL\_DAY\_SOLD+1;  V\_BEVERAGE\_AVAIL\_NO:=V\_BEVERAGE\_AVAIL\_NO-1;  update BEVERAGE\_INVENTORY  set TOTAL\_DAY\_SOLD = V\_TOTAL\_DAY\_SOLD  , BEVERAGE\_AVAIL\_NO=V\_BEVERAGE\_AVAIL\_NO  where BEVERAGE\_NO=V\_BEVERAGE\_NO;  \*/  V\_TRANS\_STATUS:='Paid-Fully';  V\_TOTAL\_AMOUNT := V\_TRANS\_SUM\_FUNDS;  V\_REFUND\_AMT := 0;  update transaction\_table  set AMOUNT\_INCOMING = V\_BEVERAGE\_PRICE  ,TRANS\_STATUS=V\_TRANS\_STATUS, TOTAL\_AMOUNT=V\_TRANS\_SUM\_FUNDS  ,PRIOR\_DAY\_SOLD=V\_TOTAL\_DAY\_SOLD+1  , CHANGE\_AMOUNT=V\_TRANS\_SUM\_FUNDS-V\_BEVERAGE\_PRICE  ,END\_TRANSTIME=to\_char(SYSDATE,'HH24:MI:SS')  ,REFUND\_AMT = 0  where TRANSACTION\_ID = V\_INCOMING\_TRANS\_ID ;  update INCOMING\_FUNDS  set TRANS\_SUM\_FUNDS = V\_TRANS\_SUM\_FUNDS  ,CHANGE\_AMOUNT=V\_TRANS\_SUM\_FUNDS-V\_BEVERAGE\_PRICE  where INCOMING\_TRANS\_ID = V\_INCOMING\_TRANS\_ID;  End IF;  End;  / | Trigger TRIGG\_SLOT\_DROP\_COINS starts its action to check coins dropped in the vending machine and populates and updates all required tables. |
| 9. | INSERT into BEVERAGE\_INVENTORY  (BEVERAGE\_NO ,BEVERAGE\_NAME ,BEVERAGE\_TYPE ,BEVERAGE\_PRICE,TOTAL\_DAY\_SOLD,BEVERAGE\_AVAIL\_NO)  values  (25, 'Coca-cola', 'Soft Drink',55,17,2076);  INSERT into BEVERAGE\_INVENTORY  (BEVERAGE\_NO ,BEVERAGE\_NAME ,BEVERAGE\_TYPE ,BEVERAGE\_PRICE,TOTAL\_DAY\_SOLD,BEVERAGE\_AVAIL\_NO)  values  (35, 'Pepsi', 'Soft Drink',50,31,1098);  INSERT into BEVERAGE\_INVENTORY  (BEVERAGE\_NO ,BEVERAGE\_NAME ,BEVERAGE\_TYPE ,BEVERAGE\_PRICE,TOTAL\_DAY\_SOLD,BEVERAGE\_AVAIL\_NO)  values  (45, 'Soda', 'Soft Drink',45,19,2100);  INSERT into BEVERAGE\_INVENTORY  (BEVERAGE\_NO ,BEVERAGE\_NAME ,BEVERAGE\_TYPE ,BEVERAGE\_PRICE,TOTAL\_DAY\_SOLD,BEVERAGE\_AVAIL\_NO)  values  (55, 'Red Bull', 'Energy Drink',70,22,1056); | This SQL code populates BEVERAGE\_INVENTORY table, which is the lookup reference data table. |
| 10. | INSERT into TRANSACTION\_TABLE  (TRANSACTION\_ID ,TRANS\_STATUS ,REFUND\_AMT ,TOTAL\_AMOUNT ,PRIOR\_DAY\_SOLD  ,TRANS\_DATE ,START\_TRANSTIME ,END\_TRANSTIME,TRANS\_BEVERAGE\_NO ,CHANGE\_AMOUNT)  values  (Transaction\_seq.nextval,'Completed', 0, 75, 17, to\_char(SYSDATE-1,'DD-MON-YYYY'),  to\_char(SYSDATE-1,'HH24-MI-SS'),to\_char(SYSDATE-1,'HH24-MI-SS'), 25,75-55);  INSERT into TRANSACTION\_TABLE  (TRANSACTION\_ID ,TRANS\_STATUS ,REFUND\_AMT ,TOTAL\_AMOUNT ,PRIOR\_DAY\_SOLD  ,TRANS\_DATE ,START\_TRANSTIME ,END\_TRANSTIME,TRANS\_BEVERAGE\_NO ,CHANGE\_AMOUNT)  values  (Transaction\_seq.nextval,'Completed', 0, 50, 31, to\_char(SYSDATE-1,'DD-MON-YYYY'),  to\_char(SYSDATE-1,'HH24-MI-SS'),to\_char(SYSDATE-1,'HH24-MI-SS'), 35,50-50);  INSERT into TRANSACTION\_TABLE  (TRANSACTION\_ID ,TRANS\_STATUS ,REFUND\_AMT ,TOTAL\_AMOUNT ,PRIOR\_DAY\_SOLD  ,TRANS\_DATE ,START\_TRANSTIME ,END\_TRANSTIME,TRANS\_BEVERAGE\_NO ,CHANGE\_AMOUNT)  values  (Transaction\_seq.nextval,'Completed', 0, 75, 22, to\_char(SYSDATE,'DD-MON-YYYY'),  to\_char(SYSDATE,'HH24-MI-SS'),to\_char(SYSDATE,'HH24-MI-SS'), 35,75-70);    INSERT into TRANSACTION\_TABLE  (TRANSACTION\_ID ,TRANS\_STATUS ,REFUND\_AMT ,PRIOR\_DAY\_SOLD  ,TRANS\_DATE ,START\_TRANSTIME ,TRANS\_BEVERAGE\_NO )  values  (Transaction\_seq.nextval,'Pending', 0, 22, to\_char(SYSDATE,'DD-MON-YYYY'),  to\_char(SYSDATE,'HH24-MI-SS'), 25); | This SQL code populates TRANSACTION\_TABLE table. |
| 11. | set lines 200 pages 40  select \* from BEVERAGE\_INVENTORY;  select \* from TRANSACTION\_TABLE;  select \* from INCOMING\_FUNDS;  select \* from INCOMING\_FUNDS\_TEMP; | This SQL queries sets the lines size, page size and lists everything in the tables populated above to verify that they were populated as expected. |
| 12. | INSERT INTO INCOMING\_FUNDS\_TEMP  (INCOMING\_TRANS\_ID,TRANS\_SINGLE\_FUND)  values  (Transaction\_seq.currval, 1);  INSERT INTO INCOMING\_FUNDS\_TEMP  (INCOMING\_TRANS\_ID,TRANS\_SINGLE\_FUND)  values  (Transaction\_seq.currval, 10);  INSERT INTO INCOMING\_FUNDS\_TEMP  (INCOMING\_TRANS\_ID,TRANS\_SINGLE\_FUND)  values  (Transaction\_seq.currval, 25);  INSERT INTO INCOMING\_FUNDS\_TEMP  (INCOMING\_TRANS\_ID,TRANS\_SINGLE\_FUND)  values  (Transaction\_seq.currval, 5);  INSERT INTO INCOMING\_FUNDS\_TEMP  (INCOMING\_TRANS\_ID,TRANS\_SINGLE\_FUND)  values  (Transaction\_seq.currval, 25); | This SQL code populates INCOMING\_FUNDS\_TEMP table. |
| 13. | set lines 200 pages 40  select \* from BEVERAGE\_INVENTORY;  select \* from TRANSACTION\_TABLE;  select \* from INCOMING\_FUNDS;  select \* from INCOMING\_FUNDS\_TEMP; | This SQL queries sets the lines size, page size and lists everything in the tables populated above to verify that they were populated as expected. |

# VENDING MACHINE CODE WALKTHROUGH

|  |  |  |
| --- | --- | --- |
| Steps | CODE | EXPLANATION |
| 1. | 1. conn system/Passwordv7@OIDDB2 2. grant connect, resource to Bunmi identified by Osula; 3. conn Bunmi/Osula@OIDDB2 | 1. Connects as System with privilege users.  2. Create or/and add Privilege to user Bunmi.  3. Connects as the new user Bunmi to run this script |
| 2. | DROP SEQUENCE P\_TRANSACTION\_SEQ;  CREATE SEQUENCE P\_TRANSACTION\_SEQ  START WITH 1  INCREMENT BY 1  MINVALUE 1  MAXVALUE 999999  ORDER  CYCLE  ; | This SQL code drops Sequence P\_TRANSACTION\_SEQ if it already exist and creates a new Sequence P\_TRANSACTION\_SEQ if it does not exist. |
| 3. | --Level 0 Tables  DROP TABLE P\_BEVERAGE\_INVENTORY CASCADE CONSTRAINTS;  CREATE TABLE P\_BEVERAGE\_INVENTORY  (P\_BEVERAGE\_NO number(4) Primary key  ,P\_BEVERAGE\_NAME varchar2(12)  ,P\_BEVERAGE\_TYPE varchar2(12)  ,P\_BEVERAGE\_TRANS\_ID number(4)  ,P\_BEVERAGE\_PRICE number(7,2)  ,P\_TOTAL\_DAY\_SOLD number(6)  ,P\_BEVERAGE\_AVAIL\_NO number(6)  )  ; | Level 0 tables are tables that have to be created first because their Primary Keys will be referenced by a Foreign key in another table (level 0 tables don’t reference any table although they are referenced by other tables).  This SQL code drops table P\_BEVERAGE\_INVENTORY with all its constraints and references and creates a new table P\_BEVERAGE\_INVENTORY. |
| 4. | --Level 1 Tables  DROP TABLE P\_TRANSACTION\_TABLE CASCADE CONSTRAINTS;  create TABLE P\_TRANSACTION\_TABLE  (P\_TRANSACTION\_ID number(4) Primary key  ,P\_AMOUNT\_INCOMING number(7,2)  ,P\_TRANS\_STATUS varchar2(12)  ,P\_REFUND\_AMT number(7,2)  ,P\_TOTAL\_AMOUNT number(7,2)  ,P\_PRIOR\_DAY\_SOLD number(6)  ,P\_TRANS\_DATE date  ,P\_START\_TRANSTIME varchar2(12)  ,P\_END\_TRANSTIME varchar2(12)  ,P\_TRANS\_BEVERAGE\_NO number(4) References P\_BEVERAGE\_INVENTORY(P\_BEVERAGE\_NO)  ,P\_CHANGE\_AMOUNT number(7,2))  ; | Level 1 tables are tables that have to be created after Level 0 tables because their Foreign keys references Primary Keys in level 0 tables.  This SQL code drops table P\_TRANSACTION\_TABLE with all its constraints and references and creates a new table P\_TRANSACTION\_TABLE. |
| 5. | --Level 3 Tables  CREATE TABLE P\_INCOMING\_FUNDS  (P\_INCOMING\_TRANS\_ID number(4) references P\_TRANSACTION\_TABLE(P\_TRANSACTION\_ID)  ,P\_TRANS\_SINGLE\_FUND number(7,2)  ,P\_TRANS\_SUM\_FUNDS number(7,2)  ,P\_INC\_TRANSFUNDS\_DATE Date  ,P\_CHANGE\_AMOUNT number(7,2)  ,P\_INC\_TRANS\_STATUS varchar2(12)  )  ; | Level 2 tables are tables that have to be created after Levels 1 tables because their Foreign keys reference Primary Keys in level 1 tables and may reference Primary Keys in Level 0.  This SQL code drops table P\_INCOMING\_FUNDS with all its constraints and references, then creates a new table P\_INCOMING\_FUNDS. |
| 6. | DROP TABLE P\_INCOMING\_FUNDS\_TEMP CASCADE CONSTRAINTS;    CREATE TABLE P\_INCOMING\_FUNDS\_TEMP  (P\_INCOMING\_TRANS\_ID number(4) references P\_TRANSACTION\_TABLE(P\_TRANSACTION\_ID)  ,P\_TRANS\_SINGLE\_FUND number(7,2)  )  ; | This SQL code drops table P\_INCOMING\_FUNDS\_TEMP with all its constraints and references, then creates a new table P\_INCOMING\_FUNDS\_TEMP. (DML Trigger: TRIGG\_SLOT\_DROP\_COINS sits on this table) |
| 7 | -- Inserting into Placeholder Table  INSERT into P\_BEVERAGE\_INVENTORY  (P\_BEVERAGE\_NO ,P\_BEVERAGE\_NAME ,P\_BEVERAGE\_TYPE ,P\_BEVERAGE\_PRICE,P\_TOTAL\_DAY\_SOLD,P\_BEVERAGE\_AVAIL\_NO)  values  (25, 'Coca-cola', 'Soft Drink',55,17,2076);  INSERT into P\_BEVERAGE\_INVENTORY  (P\_BEVERAGE\_NO ,P\_BEVERAGE\_NAME ,P\_BEVERAGE\_TYPE ,P\_BEVERAGE\_PRICE,P\_TOTAL\_DAY\_SOLD,P\_BEVERAGE\_AVAIL\_NO)  values  (35, 'Pepsi', 'Soft Drink',50,31,1098);  INSERT into P\_BEVERAGE\_INVENTORY  (P\_BEVERAGE\_NO ,P\_BEVERAGE\_NAME ,P\_BEVERAGE\_TYPE ,P\_BEVERAGE\_PRICE,P\_TOTAL\_DAY\_SOLD,P\_BEVERAGE\_AVAIL\_NO)  values  (45, 'Soda', 'Soft Drink',45,19,2100);  INSERT into P\_BEVERAGE\_INVENTORY  (P\_BEVERAGE\_NO ,P\_BEVERAGE\_NAME ,P\_BEVERAGE\_TYPE ,P\_BEVERAGE\_PRICE,P\_TOTAL\_DAY\_SOLD,P\_BEVERAGE\_AVAIL\_NO)  values  (55, 'Red Bull', 'Energy Drink',70,22,1056);  INSERT into P\_BEVERAGE\_INVENTORY  (P\_BEVERAGE\_NO ,P\_BEVERAGE\_NAME ,P\_BEVERAGE\_TYPE ,P\_BEVERAGE\_PRICE,P\_TOTAL\_DAY\_SOLD,P\_BEVERAGE\_AVAIL\_NO)  values  (75, 'Fanta', 'Soft Drink',45,476,0);  commit; | This SQL code populates P\_BEVERAGE\_INVENTORY lookup reference data table. |
| 8 | SELECT \* FROM P\_BEVERAGE\_INVENTORY; | This SQL queries verify that the inserts code above has populated the table P\_BEVERAGE\_INVENTORY as expected. |
| 9 | CREATE OR REPLACE PROCEDURE Vending\_Machine\_Proc(  F\_BEVERAGE\_NO NUMBER  ,F\_AMOUNT\_TOTAL NUMBER  ,F\_INPUT1 NUMBER  ,F\_INPUT2 NUMBER  ,F\_INPUT3 NUMBER  ,F\_INPUT4 NUMBER  ) AS | This command creates a procedure if one does not exist or replaces an existing procedure. It accepts 6 formal parameters. |
| 10 | V\_P\_BEVERAGE\_NO P\_BEVERAGE\_INVENTORY.P\_BEVERAGE\_NO%TYPE;  V\_P\_BEVERAGE\_NAME P\_BEVERAGE\_INVENTORY.P\_BEVERAGE\_NAME%TYPE;  V\_P\_BEVERAGE\_TYPE P\_BEVERAGE\_INVENTORY.P\_BEVERAGE\_TYPE%TYPE;  V\_P\_BEVERAGE\_TRANS\_ID P\_BEVERAGE\_INVENTORY.P\_BEVERAGE\_TRANS\_ID%TYPE;  V\_P\_BEVERAGE\_PRICE P\_BEVERAGE\_INVENTORY.P\_BEVERAGE\_PRICE%TYPE;  V\_P\_TOTAL\_DAY\_SOLD P\_BEVERAGE\_INVENTORY.P\_TOTAL\_DAY\_SOLD%TYPE;  V\_P\_BEVERAGE\_AVAIL\_NO P\_BEVERAGE\_INVENTORY.P\_BEVERAGE\_AVAIL\_NO%TYPE;    V\_P\_AMOUNT\_INCOMING P\_TRANSACTION\_TABLE.P\_AMOUNT\_INCOMING%TYPE;  V\_P\_TRANS\_DATE P\_TRANSACTION\_TABLE.P\_TRANS\_DATE%TYPE;  V\_P\_TRANSACTION\_ID P\_TRANSACTION\_TABLE.P\_TRANSACTION\_ID%TYPE;  V\_P\_TRANS\_STATUS P\_TRANSACTION\_TABLE.P\_TRANS\_STATUS%TYPE;  V\_P\_REFUND\_AMT P\_TRANSACTION\_TABLE.P\_REFUND\_AMT%TYPE;  V\_P\_TOTAL\_AMOUNT P\_TRANSACTION\_TABLE.P\_TOTAL\_AMOUNT%TYPE;  V\_P\_PRIOR\_DAY\_SOLD P\_TRANSACTION\_TABLE.P\_PRIOR\_DAY\_SOLD%TYPE;  V\_P\_START\_TRANSTIME P\_TRANSACTION\_TABLE.P\_START\_TRANSTIME%TYPE;  V\_P\_END\_TRANSTIME P\_TRANSACTION\_TABLE.P\_END\_TRANSTIME%TYPE;  V\_P\_TRANS\_BEVERAGE\_NO P\_TRANSACTION\_TABLE.P\_TRANS\_BEVERAGE\_NO%TYPE;  --V\_P\_CHANGE\_AMOUNT P\_TRANSACTION\_TABLE.P\_CHANGE\_AMOUNT%TYPE;  V\_P\_WRONG\_COINS P\_INCOMING\_FUNDS.P\_TRANS\_SUM\_FUNDS%TYPE;  V\_P\_TRANS\_SINGLE\_FUND P\_INCOMING\_FUNDS.P\_TRANS\_SINGLE\_FUND%TYPE;  V\_P\_TRANS\_SUM\_FUNDS P\_INCOMING\_FUNDS.P\_TRANS\_SUM\_FUNDS%TYPE;  V\_P\_INCOMING\_TRANS\_ID P\_INCOMING\_FUNDS.P\_INCOMING\_TRANS\_ID%TYPE;  V\_P\_INC\_TRANSFUNDS\_DATE P\_INCOMING\_FUNDS.P\_INC\_TRANSFUNDS\_DATE%TYPE;  V\_P\_CHANGE\_AMOUNT P\_INCOMING\_FUNDS.P\_CHANGE\_AMOUNT%TYPE;  V\_P\_INC\_TRANS\_STATUS P\_INCOMING\_FUNDS.P\_INC\_TRANS\_STATUS%TYPE;  F\_P\_TRANS\_SINGLE\_FUND P\_INCOMING\_FUNDS.P\_TRANS\_SINGLE\_FUND%TYPE;  V\_P\_DIFFERENCE\_AMOUNT P\_INCOMING\_FUNDS.P\_CHANGE\_AMOUNT%TYPE;  V\_P\_BEV\_P\_COST\_BALANCE P\_INCOMING\_FUNDS.P\_TRANS\_SINGLE\_FUND%TYPE;  V\_INPUT1 NUMBER;  V\_INPUT2 NUMBER;  V\_INPUT3 NUMBER;  V\_INPUT4 NUMBER;  V\_INPUT5 NUMBER;  V\_P\_MOD\_VAL NUMBER; | These Section of the PL/SQL code declares all the variables that will be used by the application. |
| 11 | BEGIN  SELECT  P\_BEVERAGE\_NO , P\_BEVERAGE\_NAME , P\_BEVERAGE\_TYPE , P\_BEVERAGE\_AVAIL\_NO ,P\_BEVERAGE\_PRICE , P\_TOTAL\_DAY\_SOLD  INTO  V\_P\_BEVERAGE\_NO , V\_P\_BEVERAGE\_NAME ,V\_P\_BEVERAGE\_TYPE , V\_P\_BEVERAGE\_AVAIL\_NO, V\_P\_BEVERAGE\_PRICE,V\_P\_TOTAL\_DAY\_SOLD  FROM P\_BEVERAGE\_INVENTORY  WHERE P\_BEVERAGE\_NO = F\_BEVERAGE\_NO; | The procedure starts at begin.  The Select statement gets all the probable data about the beverage into some of the variables declared above. |
| 12 | V\_P\_START\_TRANSTIME:=TO\_CHAR(SYSDATE,'HH24:MI:SS');  V\_P\_MOD\_VAL := MOD(F\_AMOUNT\_TOTAL,1);    IF V\_P\_MOD\_VAL != '0' THEN  V\_P\_WRONG\_COINS:=F\_AMOUNT\_TOTAL;    NULL;    DBMS\_OUTPUT.PUT\_LINE('This Machine only Accepts coins of 1,5,10,25 Cents i.e. penny, nickel, dime, and quarter and nothing more ');  DBMS\_OUTPUT.PUT\_LINE('Kindly ensure that those are the coins fed into the machine');  DBMS\_OUTPUT.PUT\_LINE('Do collect the rejected coins '|| V\_P\_WRONG\_COINS ||' below '); | This section of code checks the coins entered in multiples of 1, 5, 10 and 25 cents.  A message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package.function. |
| 13 | elsif V\_P\_MOD\_VAL =0 THEN  IF V\_P\_BEVERAGE\_AVAIL\_NO = 0 THEN  null;  V\_P\_REFUND\_AMT:=F\_AMOUNT\_TOTAL;  DBMS\_OUTPUT.PUT\_LINE('The ' || V\_P\_BEVERAGE\_NAME || ' '|| V\_P\_BEVERAGE\_TYPE || ' No. ' ||V\_P\_BEVERAGE\_NO  || ' You are trying to purchase is Presenlty out of stock ');  DBMS\_OUTPUT.PUT\_LINE('Kindly select another drink to continue or try at another time or from another machine');  DBMS\_OUTPUT.PUT\_LINE('Please take your full refund amount of '||V\_P\_REFUND\_AMT ||' Cent');  DBMS\_OUTPUT.PUT\_LINE('Many thanks for your understanding'); | This section of PL/SQL handles Beverages that are out of stock. If the stock required is 0, then procedure refunds the Customers money and sends message to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package.function. |
| 14 | elsif V\_P\_BEVERAGE\_AVAIL\_NO >= 1 THEN  if F\_AMOUNT\_TOTAL < V\_P\_BEVERAGE\_PRICE then  V\_P\_DIFFERENCE\_AMOUNT := V\_P\_BEVERAGE\_PRICE - F\_AMOUNT\_TOTAL;  null;  DBMS\_OUTPUT.PUT\_LINE('The ' || V\_P\_BEVERAGE\_NAME || ' '|| V\_P\_BEVERAGE\_TYPE || ' No. ' ||V\_P\_BEVERAGE\_NO  || ' You are trying to purchase is '|| V\_P\_BEVERAGE\_PRICE || ' Cents' );  DBMS\_OUTPUT.PUT\_LINE('This Machine Accepts coins of 1,5,10,25 Cents i.e. penny, nickel, dime, and quarter ');  DBMS\_OUTPUT.PUT\_LINE('Please Make payments accordingly');  DBMS\_OUTPUT.PUT\_LINE('You Just Paid '||F\_AMOUNT\_TOTAL||' Cents');  DBMS\_OUTPUT.PUT\_LINE('You need to add '||V\_P\_DIFFERENCE\_AMOUNT ||' Cent to purchase this item' ); | This section of PL/SQL code handles a situation where the product is available and amount entered is less than the beverage price.  Procedure calculates how much more is required to complete the transaction and sends message to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package.function. |
| 15 | elsif F\_AMOUNT\_TOTAL >= V\_P\_BEVERAGE\_PRICE then    V\_P\_CHANGE\_AMOUNT := F\_AMOUNT\_TOTAL-V\_P\_BEVERAGE\_PRICE;  DBMS\_OUTPUT.PUT\_LINE('The ' || V\_P\_BEVERAGE\_NAME || ' '|| V\_P\_BEVERAGE\_TYPE || ' No. ' ||V\_P\_BEVERAGE\_NO  || ' You are trying to purchase is '|| V\_P\_BEVERAGE\_PRICE || ' Cents' );  DBMS\_OUTPUT.PUT\_LINE('This Machine Accepts coins of 1,5,10,25 Cents i.e. penny, nickel, dime, and quarter ');  DBMS\_OUTPUT.PUT\_LINE('Please Make payments accordingly');  DBMS\_OUTPUT.PUT\_LINE('You Just Paid '||F\_AMOUNT\_TOTAL||' Cents');  DBMS\_OUTPUT.PUT\_LINE('Your Change will be '||V\_P\_CHANGE\_AMOUNT ||' Cent'); | This section of PL/SQL code handles a situation where amount entered is greater than the beverage price.  Procedure calculates the difference in amount entered and Price of the beverage and Passes it into the Change variable. Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package. function. |
| 16 | DBMS\_OUTPUT.PUT\_LINE('Kindly Confirm to Proceed with your Order by Selecting the Button 1 for Yes and Button 2 for No ');  DBMS\_OUTPUT.PUT\_LINE('or you may abort this Transaction by selecting the CANCEL button 3' ); | Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package.function to confirm they are happy or not happy to go ahead with the purchase. |
| 17 | IF F\_INPUT1=1 or F\_INPUT3=2 then  Update P\_BEVERAGE\_INVENTORY set  P\_BEVERAGE\_AVAIL\_NO = P\_BEVERAGE\_AVAIL\_NO-1  where  P\_BEVERAGE\_NO = F\_BEVERAGE\_NO;  SELECT P\_TRANSACTION\_SEQ.nextval into V\_P\_TRANSACTION\_ID  from dual;  V\_P\_AMOUNT\_INCOMING:=V\_P\_BEVERAGE\_PRICE;  V\_P\_TRANS\_STATUS:='SUCCESSFUL';  V\_P\_REFUND\_AMT:=0;  V\_P\_TOTAL\_AMOUNT:=F\_AMOUNT\_TOTAL;  V\_P\_PRIOR\_DAY\_SOLD:= V\_P\_TOTAL\_DAY\_SOLD;  V\_P\_TRANS\_DATE:=TO\_CHAR(SYSDATE,'DD-MON-YYYY');  V\_P\_END\_TRANSTIME:=TO\_CHAR(SYSDATE,'HH24:MI:SS');  V\_P\_TRANS\_BEVERAGE\_NO:= F\_BEVERAGE\_NO;  V\_P\_CHANGE\_AMOUNT:=F\_AMOUNT\_TOTAL-V\_P\_BEVERAGE\_PRICE;  Insert into P\_TRANSACTION\_TABLE  (P\_TRANSACTION\_ID  ,P\_AMOUNT\_INCOMING  ,P\_TRANS\_STATUS  ,P\_REFUND\_AMT  ,P\_TOTAL\_AMOUNT  ,P\_PRIOR\_DAY\_SOLD  ,P\_TRANS\_DATE  ,P\_START\_TRANSTIME  ,P\_END\_TRANSTIME  ,P\_TRANS\_BEVERAGE\_NO  ,P\_CHANGE\_AMOUNT)  values  (V\_P\_TRANSACTION\_ID  ,V\_P\_AMOUNT\_INCOMING  ,V\_P\_TRANS\_STATUS  ,V\_P\_REFUND\_AMT  ,V\_P\_TOTAL\_AMOUNT  ,V\_P\_PRIOR\_DAY\_SOLD  ,V\_P\_TRANS\_DATE  ,V\_P\_START\_TRANSTIME  ,V\_P\_END\_TRANSTIME  ,V\_P\_TRANS\_BEVERAGE\_NO  ,V\_P\_CHANGE\_AMOUNT)  ;  Insert INTO P\_INCOMING\_FUNDS  (P\_INCOMING\_TRANS\_ID  ,P\_TRANS\_SINGLE\_FUND  ,P\_TRANS\_SUM\_FUNDS  ,P\_INC\_TRANSFUNDS\_DATE  ,P\_CHANGE\_AMOUNT  ,P\_INC\_TRANS\_STATUS  )  values  (V\_P\_TRANSACTION\_ID  ,V\_P\_TOTAL\_AMOUNT  ,V\_P\_TOTAL\_AMOUNT  ,SYSDATE  ,V\_P\_CHANGE\_AMOUNT  ,'Completed')  ;    INSERT INTO P\_INCOMING\_FUNDS\_TEMP  (P\_INCOMING\_TRANS\_ID  ,P\_TRANS\_SINGLE\_FUND  )  values  (V\_P\_TRANSACTION\_ID  ,V\_P\_TOTAL\_AMOUNT)  ; | If the input entered is 1 or Input 3 Cancellation is negated by selectin No, then the tables will be updated with the sales and amount entered. |
| 18 | Else  null;  DBMS\_OUTPUT.PUT\_LINE('We are Sorry to see you are not happy with your order TransactionID:-'||V\_P\_TRANSACTION\_ID );  DBMS\_OUTPUT.PUT\_LINE('Kindly collect your full refund of $'||V\_P\_TRANS\_SUM\_FUNDS ||' from the Return tray below here') ;  DBMS\_OUTPUT.PUT\_LINE('We hope to see you again soon') ; | If the customer cancels the transaction then the tables will not be updated.  Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package.function |
| 19 | exception  when NO\_DATA\_FOUND then  DBMS\_OUTPUT.PUT\_LINE('YOU MAY HAVE PUNCHED IN A BEVERAGE NO. NOT KNOWN TO THE MACHINE');  DBMS\_OUTPUT.PUT\_LINE('Confirm that the Beverage No. ' || F\_BEVERAGE\_NO||' is a valid number ');  DBMS\_OUTPUT.PUT\_LINE('IF THE ABOVE BEVERAGE NO IS CORRECT, Kindly call Supply support on 08451234567'); | This handles the exceptions when beverage number entered is not recognized.  Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package. function |
| 20 | when ZERO\_DIVIDE then  DBMS\_OUTPUT.PUT\_LINE('The coins inserted seem to be unacceptable' );  DBMS\_OUTPUT.PUT\_LINE(F\_P\_TRANS\_SINGLE\_FUND || ' seem to be causing a problem');  DBMS\_OUTPUT.PUT\_LINE('If the above is not the case, Kindly call Supply support on 08451234567'); | This handles the exceptions when the coin entered is not recognized or the person has entered no coin.  Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package. |
| 21 | when TIMEOUT\_ON\_RESOURCE then  DBMS\_OUTPUT.PUT\_LINE('We can see you are trying to purchase Item No. ' ||F\_BEVERAGE\_NO);  DBMS\_OUTPUT.PUT\_LINE('THE OPERATION HAS TAKEN TOO LONG TO BE EXECUTED AND HENCE TIMED OUT BY THE SERVER ');  DBMS\_OUTPUT.PUT\_LINE('PLEASE RE-TRY THE OPERATION AND IF TIMED OUT AGAIN, ACCEPT OUR APOLIGIES AND RE-TRY');  DBMS\_OUTPUT.PUT\_LINE('IF THE ABOVE OPERATION IS VALID, Kindly call Supply support on 08451234567'); | This handles the timeout when the machine is performing update or network is busy .  Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package.function |
| 22 | when PROGRAM\_ERROR then  DBMS\_OUTPUT.PUT\_LINE('Please check the ' ||F\_BEVERAGE\_NO || ' and re-start' );  DBMS\_OUTPUT.PUT\_LINE('EITHER '|| F\_P\_TRANS\_SINGLE\_FUND || ' IS NOT A Currency OR '|| F\_BEVERAGE\_NO || ' IS NOT A valid Product number ');  DBMS\_OUTPUT.PUT\_LINE('If the above is not the case, Kindly call Supply support on 08451234567');  DBMS\_OUTPUT.PUT\_LINE('Press the RESET Button'); | This handles where there is an error in the code, but code compiled okay.  Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package. function |
| 22 | DBMS\_OUTPUT.PUT\_LINE('Please pick your full refund from the Coin tray below');  DBMS\_OUTPUT.PUT\_LINE('If the above is not the case, Kindly call Supply support on 08451234567'); | This handles the refund.  Message is sent to the customer using the Oracle built in DBMS\_OUTPUT.PUT\_LINE package. function |
| 23 | DBMS\_OUTPUT.PUT\_LINE('You would have received any funds previously inserted in the tray by now');  DBMS\_OUTPUT.PUT\_LINE('Kindly Start again and the process shall trigger start as soon as you are inserting coins'); | This is the final error message to the customer to restart again. |
| 24 | when OTHERS then  DBMS\_OUTPUT.PUT\_LINE('THERE SEEM TO BE A PROBLEM WITH EITHER THE '||F\_BEVERAGE\_NO|| ' OR THE '||F\_P\_TRANS\_SINGLE\_FUND ||' WHICH WE CANNOT TELL AT THIS STAGE');  DBMS\_OUTPUT.PUT\_LINE('If the above is not the case, Kindly call Supply support on 08451234567'); | This handles any other error in the program and facilitates an exit for application instead of hanging. |
| 25 | END Vending\_Machine\_Proc; | This is the end of the procedure. |
| 26. | exec Vending\_Machine\_Proc(35,80,1,1,2,1);  exec Vending\_Machine\_Proc(75,66,1,1,1,1);  exec Vending\_Machine\_Proc(85,57,1,1,2,1);  exec Vending\_Machine\_Proc(45,76,1,1,1,1);  exec Vending\_Machine\_Proc(25,85,1,1,1,1);  exec Vending\_Machine\_Proc(55,70,1,1,2,1);  exec Vending\_Machine\_Proc(55,81.5,1,1,1,1);  exec Vending\_Machine\_Proc(45,30,1,1,1,1);  exec Vending\_Machine\_Proc(25,33,1,1,2,1);  exec Vending\_Machine\_Proc(55,52,1,1,1,1);  exec Vending\_Machine\_Proc(55,65.5,1,2,1,1);  exec Vending\_Machine\_Proc(35,80,1,1,2,1);  exec Vending\_Machine\_Proc(75,66,1,1,1,1);  exec Vending\_Machine\_Proc(85,57,1,1,2,1);  exec Vending\_Machine\_Proc(45,76,1,1,1,1);  exec Vending\_Machine\_Proc(25,85,1,1,1,1);  exec Vending\_Machine\_Proc(55,70,1,1,2,1);  exec Vending\_Machine\_Proc(55,81.5,1,2,1,1);  exec Vending\_Machine\_Proc(45,30,1,1,1,1);  exec Vending\_Machine\_Proc(25,33,1,1,2,1);  exec Vending\_Machine\_Proc(55,52,1,1,1,1);  exec Vending\_Machine\_Proc(45,65.5,1,2,2,1); | This executes the procedure passing it through several tests, varying prices, beverage, drinks and response. |
| 27. | select \* from P\_BEVERAGE\_INVENTORY;  select \* from P\_TRANSACTION\_TABLE;  Select \* from P\_INCOMING\_FUNDS;    Select \* from P\_INCOMING\_FUNDS\_TEMP; | This SQL queries verify that in executing procedure Vending\_Machine\_Proc: P\_BEVERAGE\_INVENTORY, P\_TRANSACTION\_TABLE, P\_INCOMING\_FUNDS and P\_INCOMING\_FUNDS\_TEMP tables are populated or updated as expected. |

# API Automation

Test Scenario 1

Feature: City bike location worldwide

Description: Ensure Frankfurt is displayed as a city in Germany and corresponding longitude and latitude are displayed

Scenario: Successful Login with valid credentials

Given I have the API is up and running

When I enter the API Endpoint with the City and Location

And Germany is Country(DE)

Then Frankfurt should be listed as a City

And the corresponding longitude and latitude should be displayed

Test Scenario 2

Feature: City bike location worldwide

Description: Ensure Moscow is displayed as a city in Russia and corresponding longitude and latitude are displayed

Scenario: Successful Login with valid credentials

Given I have the API is up and running

When I enter the API Endpoint with the City and Location

And Russia is Country(RU)

Then Moscow should be listed as a City

And the corresponding longitude and latitude should be displayed

Test Scenario 2

Feature: Wrong Location ID entered

Description: Ensure Moscow is displayed as a city in Russia and corresponding longitude and latitude are displayed

Scenario: When an invalid location id is entered, error message should be raised ‘Location ID does not exist’

Given I have the API is up and running

When I enter the API Endpoint with the wrong City and Location

Then Russia is Country(RU)

|  |  |  |
| --- | --- | --- |
| Steps | Screenshots | EXPLANATION |
| 1. |  | Copied the API (<http://api.citybik.es/v2/networks>) endpoint into Postman to get a full listing |
| 2. |  | Applied Location id to get a listing of Frankfurt. |
| 3. | Feature: City bike location worldwide  Description: Ensure Frankfurt is displayed as a city in Germany and corresponding longitude and latitude are displayed  Scenario: Successful Login with valid credentials    Given I have the API is up and running  When I enter the API Endpoint with the City and Location  And Germany is Country(DE)  Then Frankfurt should be listed as a City  And the corresponding longitude and latitude should be displayed | Created the Feature file(Featurecombikecity.feature) |
| 4. | package StepDefinitions;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  import cucumber.api.java.en.Given;  import cucumber.api.java.en.Then;  import cucumber.api.java.en.When;  public class stepDef\_BikeCity {  WebDriver driver = null;    @Given("^I have the API is up and running$")  public void i\_have\_the\_API\_is\_up\_and\_running() throws Throwable {  driver = new FirefoxDriver();  }  @When("^I enter the API Endpoint with the City and Location$")  public void i\_enter\_the\_API\_Endpoint\_with\_the\_City\_and\_Location() throws Throwable {  driver.navigate().to("http://api.citybik.es/v2/networks/visa-frankfurt?fields=location");  Thread.sleep(3000);    }  @When("^Germany is Country\\(DE\\)$")  public void germany\_is\_Country\_DE() throws Throwable {  String countryText = driver.findElement(By.id("/network/location/country")).getText();  System.out.println(countryText);  countryText.contains("DE");    }  @Then("^Frankfurt should be listed as a City$")  public void frankfurt\_should\_be\_listed\_as\_a\_City() throws Throwable {  String cityText = driver.findElement(By.id("/network/location/city")).getText();  System.out.println(cityText);  cityText.contains("Frankfurt");    }  @Then("^the corresponding longitude and latitude should be displayed$")  public void the\_corresponding\_longitude\_and\_latitude\_should\_be\_displayed() throws Throwable {  System.out.println(driver.findElement(By.id("/network/location/longitude")).getText());  System.out.println(driver.findElement(By.id("/network/location/latitude")).getText());    }  } | Generated the glue file from the Feature file and copied it into the Step definition file(StepDef\_BikeCity.Java) |
| 5. | package StepDefinitions;  import org.junit.runner.RunWith;  import cucumber.api.CucumberOptions;  import cucumber.api.junit.Cucumber;  @RunWith(Cucumber.class)  @CucumberOptions(features="src/test/java", glue = "StepDefinitions")  public class RunBikeCityTest {    } | Created a runner file to run the Step definition file(StepDef\_BikeCity.Java) |
| 6. |  | Run the test successfully on command prompt using the ‘mvn test’. |
| 7. |  | Run the test successfully on the Eclipse Console. Country Code, City, longitude and latitude were all displayed as expected. |