

LOGINRPGLE

```
FLOGINPAGE CF E      WORKSTN
DDSG      S      15 INZ
DIND      S      1 0 INZ
DCMD      S      200 INZ
DLEN      S      15 5
*
DCMDEXEC    PR      EXTPGM('QCMDEXC')
DCMD        200
DLEN        15 5
DUSDTAARA   UDS      DTAARA
DNAME       10
C          DOW    *IN03='0'
C          EXFMT  LOGINSCRN
C          EVAL    MSG = *BLANKS
C          IF     *IN03 = '1'
C          LEAVE
C          ENDIF
c          EXSR    LOADLOGIN
C          ENDDO
C          EVAL    *INLR = '1'
*-----
C          LOADLOGIN BEGSR
IND = 0;
DSG = '';
EXEC SQL
SELECT 1 INTO :IND FROM USERFILE WHERE
USERID = :USER AND PASSWORD = :PSWD ;
IF IND = 1;
*IN87 = '0';
C          *LOCK    IN    USDTAARA
C          EVAL    %SUBST(USDTAARA:1:10) = USER
C          OUT    USDTAARA
EXEC SQL
SELECT DESIG INTO :DSG FROM USERFILE WHERE
USERID = :USER AND PASSWORD = :PSWD ;
IF DSG = 'EMPLOYEE';
CMD = 'GO EMPMENU';
LEN = %LEN(%TRIM(CMD)) ;
CMDEXEC(CMD:LEN) ;
ELSEIF DSG = 'MANAGER';
CMD = 'GO MGRMENU';
LEN = %LEN(%TRIM(CMD)) ;
CMDEXEC(CMD:LEN) ;
ELSEIF DSG = 'ADMIN';
CMD = 'GO ADMENU';
LEN = %LEN(%TRIM(CMD)) ;
CMDEXEC(CMD:LEN) ;
ENDIF;
ELSE;
MSG = 'WRONG CREDENTIALS' ;
*IN87 = '1';
ENDIF;
C          ENDSR
```

This program is designed to handle the login process for a workstation. It continuously displays a login screen until the user either logs in successfully or exits the program. When the user enters their credentials, the

program checks the USERFILE database to validate them. If the credentials are correct, it determines the user's designation and navigates them to the appropriate menu (Employee, Manager, or Admin) based on their role. If the credentials are invalid, the program displays an error message. The program also updates a data area with the user's information upon successful login and ensures appropriate menu navigation through command execution. The main loop continues until the user decides to exit, at which point the program terminates.

STOCKENTRY

```
FSTOCKMST  IF A E      K DISK
FSTOCKSCN3 CF   E      WORKSTN
DITEMDTA    UDS       DTAARA
DSUB1        S         8
DTNUM       S         8 0
*
C          DOW      *IN12='0'
C          EXFMT    STCMSTENT
C          IF       *IN12='1'
C          LEAVE
C          ENDIF
C          IF       SITEENAME = *BLANKS OR SUNITMSR = *BLANKS
C                      OR SSTOCKINH = 0 OR SREORDLVL = 0 OR
C                      SPRICE = 0
C          EVAL     STCMSTMSG = 'ALL FIELDS REQUIRED'
C          ELSE
C          IF       SUNITMSR = 'QTY' OR SUNITMSR = 'KG' OR
C                      SUNITMSR = 'L'
C
EXEC SQL
  SET OPTION COMMIT = *NONE ;
EXEC SQL
  INSERT INTO STOCKMST  (ITEMCODE, ITEMNAME, UNITMSR,
  STOCKINH, REORDLVL, PRICE, VALUE)
  VALUES ('E_' || :SUB1, :SITEENAME, :SUNITMSR, :SSTOCKINH,
  :SREORDLVL, :SPRICE, (:SSTOCKINH * :SPRICE));
C          EVAL     TNUM = %DEC(SUB1:8:0)+1
C          EVAL     SUB1=%CHAR(TNUM)
C          EVAL     STCMSTMSG = 'RECORD ADDED SUCCESSFULLY'
C          ELSE
C          EVAL     STCMSTMSG = 'INCORRECT UNIT OF MEASURE'
C          ENDIF
C          ENDIF
C          ENDDO
C          EVAL     STCMSTMSG = *BLANKS
C          EVAL     *INLR='1'
```

This program handles the entry and validation of stock item details. It continuously displays an input screen for stock master entry until the user exits. The program checks that all fields are filled in and that the unit of measure is valid (either 'QTY', 'KG', or 'L'). If the inputs are valid, it inserts the new stock item into the `STOCKMST` database table, generates a new item code by incrementing a counter, and displays a success message. If the inputs are invalid or the unit of measure is incorrect, it displays an appropriate error message. The program terminates when the user decides to exit.

GRCVDENTRY

This program manages the entry and validation of goods received information. It displays an input screen for goods received entry until the user exits. The program checks that all necessary fields are filled and verifies the item code. If the item code is valid, it records the user's information, date, and time. It retrieves the item price from the `STOCKMST` database, calculates the total value, and inserts the received goods details into the `GOODSRCVD` database. It also updates the stock quantity and value in the `STOCKMST` database. If any fields are missing or the item code is invalid, it displays an appropriate error message. The program terminates when the user decides to exit.

```

HDFACTGRP (*NO)BNDDIR('SFLENDDIR')
FGOODSISSD IF A E          K DISK
FGOODRCVSCNCF   E          WORKSTN
DTNUM           S            8  0
DCHECKITEM     PR           1
D               D             10
*
DUSDTAARA     UDS          DTAARA
DNAME          *             10
*
DFLAG          S            1
DFLAG1         S            1
DWRUSERID     S            10
DWRDATE        S            D
DWRTIME        S            T
DWPRICE        S            10  2
DWRVALUE       S            12  2
*
*
C               DOW          *IN12='0'
C               EVAL         SCURRUSER = NAME
C               EXFMT        GRCVENTRY
C               IF           *IN12='1'
C               LEAVE
C               ENDIF
C               IF           SGRN = *BLANKS OR SRITEMCODE = *BLANKS
C                           OR SREQUESTNO = *BLANKS OR SQTYRCVD = 0
C               EVAL         SMSG = 'ALL FIELDS REQUIRED'
C               ELSE
C               EVAL         FLAG = CHECKITEM(SRITEMCODE)

EXEC SQL
  SET OPTION COMMIT = *NONE ;

FLAG1 = '0';
EXEC SQL
  SELECT '1' INTO :FLAG1 FROM GOODSRCVD
  WHERE RITEMCODE = :SRITEMCODE
  AND REQUESTNO = :SREQUESTNO;

C               IF           FLAG = '1' AND FLAG1 = '0'
C               EVAL         WRUSERID = NAME
C               EVAL         WRDATE=%DATE()
C               EVAL         WRTIME=%TIME()

EXEC SQL
  SELECT PRICE INTO :WPRICE FROM STOCKMST WHERE ITEMCODE = :SRITEMCODE ;
  WRVALUE = WPRICE*SQTYRCVD ;
EXEC SQL
  INSERT INTO GOODSRCVD (GRN, RITEMCODE, RUSERID,
    RDATE, RTIME, REQUESTNO, QTYRCVD,GOODSPRICE)
  VALUES (:SGRN, :SRITEMCODE, :WRUSERID, :WRDATE,
    :WRTIME, :SREQUESTNO, :SQTYRCVD,:WRVALUE);
EXEC SQL
  UPDATE STOCKMST

```

```

SET STOCKINH = STOCKINH + :SQTYRCVD,
    VALUE = (STOCKINH + :SQTYRCVD) * PRICE
WHERE ITEMCODE = :SRITEMCODE;

C             EVAL      SMSG = 'RECORD ADDED SUCCESSFULLY'
C             ELSE
IF FLAG1 = '1';
SMSG = 'RECORD ALREADY EXISTS';
ELSE;
C             EVAL      SMSG = 'ITEM DOES NOT EXIST'
C             ENDIF
C             ENDIF
C             ENDIF
C             ENDDO
C             EVAL      SMSG = '*BLANKS
C             EVAL      *INLR='1'

```

MOD_CHECK

```

H*NOMAIN
*
*INLR = '1' ;
PCHECKITEM     B          EXPORT
DCHECKITEM     PI         1
DWITEMCODE      10
DFLAG           S          1
FLAG='0' ;
EXEC SQL
  SET OPTION COMMIT = *NONE ;
EXEC SQL
  SELECT '1' INTO :FLAG FROM STOCKMST WHERE
  ITEMCODE = :WITEMCODE ;
  RETURN FLAG ;
PCHECKITEM     E

```

GISSENTRY

This program manages the issuance of goods. It displays an input screen for goods issuance until the user exits. It verifies that all necessary fields are filled and checks the item code's validity. If the item code is valid, it retrieves the available stock from the `STOCKMST` database. If the requested quantity exceeds the available stock, it displays an error message. If the stock is sufficient, it inserts the goods issuance record, updates the stock in the `STOCKMST` database, and displays a success message. If the item code is invalid or a duplicate record is detected, it displays an appropriate error message. The program increments a counter for each new issuance and terminates when the user decides to exit.

```

HDFACTGRF(*NO)BNDDIR('GISSDIR')
PGOODISSSD UF A E K DISK
PGOODISSSCNCF E WORKSTN
DCNT S 10 0
DSTOCK S 8 0
DISNDTA UDS DTAARA
DCNTC 8
DUSDTAARA UDS DTAARA
DUSER 10
DCHECKITEM PR 1
D 10
DFLAG S 1
*-----
EXEC SQL
SET OPTION COMMIT = *NONE;
*-----
CNT = %DEC(CNTC:8:0);
DOW *IN12 = '0';
  EXFMT GISSENTRY;
  SMSG = *BLANKS;
  *IN87 = '0';
  IF *IN12 = '1';
    LEAVE;
  ENDIF;
  IF SIITEMCODE = *BLANKS OR SQTYISSD = 0;
    *IN87 = '1';
    SMSG = 'BLANK FIELDS DETECTED';
  ELSE;
    FLAG = CHECKITEM(SIITEMCODE);
    IF FLAG = '1';
      *IN87 = '0';
      EXEC SQL
        SELECT STOCKINH INTO :STOCK FROM STOCKMST WHERE
ITEMCODE = :SIITEMCODE;
        IF SQTYISSD>STOCK;
          *IN87 = '1';
          SMSG = 'STOCK NOT AVAILABLE';
        ELSE;
          *IN87 = '0';
          ISN = 'S_'+CNTC;
          IITEMCODE = SIITEMCODE;
          IUSERID = USER;
          IDATE = %DATE();
          ITIME = %TIME();
          QTYISSD = SQTYISSD;
          WRITE(E) ISNREC;
          IF *ERROR();
            *IN87 = '1';
            SMSG = 'DUPLICATE RECORD';
          ELSE;
            EXEC SQL
              UPDATE STOCKMST SET STOCKINH = STOCKINH-:SQTYISSD,
              VALUE = (STOCKINH - :SQTYISSD) * PRICE
              WHERE ITEMCODE = :SIITEMCODE;
            *IN87 = '0';
            SMSG = 'RECORD ADDED';
          ENDIF;
          CNT = CNT + 1;
          CNTC = %CHAR(CNT);
        ENDIF;
      ELSE;
        *IN87 = '1';
        SMSG = 'ITEM NOT FOUND';
      ENDIF;
    ENDIF;
  ENDDO;
*
  *INLR = '1';

```

USERENTRY

```
FUSERFILE  IF A E          K DISK
FUSERSCN3  CF   E          WORKSTN
*
DCOUNTUSER      S          10  0

DOW      *IN12 = '0' ;
EXFMT    USERENTRY ;
IF      *IN12 = '1';
  LEAVE;
ENDIF ;

EXEC SQL
SET OPTION COMMIT = *NONE ;

IF      SUSERID = *BLANKS OR SDESIG = *BLANKS OR
      SPASSWORD = *BLANKS OR SREPASS = *BLANKS;■

  SMSG = 'ALL FIELDS REQUIRED';
ELSE ;
IF      SDESIG = 'ADMIN' OR SDESIG='OPERATOR' OR SDESIG='MANAGER' ;

  EXEC SQL
  SELECT COUNT(USERID) INTO :COUNTUSER FROM USERFILE
  WHERE USERID = :SUSERID ;

  IF COUNTUSER <> 0;
    SMSG = 'USER ALREADY EXISTS' ;
  ELSEIF    SPASSWORD <> SREPASS ;
    SMSG = 'PASSWORD AND RE-ENTERED DIFF' ;

  ELSE;
    EXEC SQL
    INSERT INTO USERFILE (USERID, PASSWORD, DESIG )
    VALUES (:SUSERID, :SPASSWORD, :SDESIG );

    SMSG = 'RECORD ADDED SUCCESSFULLY';
  ENDIF ;
ELSE ;
  SMSG = 'INVALID DESIGNATION' ;
ENDIF ;
ENDIF ;
ENDIF ;
ENDDO ;
SMSG = *BLANKS;
*INLR='1';
```

This program handles user entry and validation. It displays an input screen for user entry until the user exits. The program verifies that all fields are filled and checks if the user designation is either 'ADMIN', 'OPERATOR', or 'MANAGER'. If the fields are valid and the designation is correct, it checks if the user already exists in the `USERFILE` database. If the user exists, it displays an error message; if the password and re-entered password do not match, it displays a different error message. If the user does not exist and the passwords match, it inserts the new user details into the `USERFILE` database and displays a success message. If the designation is invalid, it displays an appropriate error message. The program terminates when the user decides to exit.

WANDTENTRY

```
HDFTACTGRP(*NO) BNDDIR('WANDTENTRY')
FWTSCN      CF   E          WORKSTN
FWEARTEAR   UF   A E       DISK
*
DCHECKITEM    PR     1
D             10
DFLAG        S     1
DUSDTAARA   UDS    DTAARA
DUSR         10
DWTDATAARA  UDS    DTAARA
DSUB1        8
DNUM         S     8  0
*****DECLARE DATA AREA AND PROCEDURE*****
*
EXEC SQL
    SET OPTION COMMIT = *NONE ;
C           DOW      *IN12 = '0'
C           EXFMT    WTENTRY
C           IF      *IN12 = '1'
C           LEAVE
C           ENDIF
C           IF      SWITEMCODE = *BLANKS OR SREASON = *BLANKS
C                   OR SQTYLOSS = 0
C
C           EVAL    SWMSG = 'ENTER ALL FIELDS'
C           ELSE
C           EVAL    FLAG=CHECKITEM(SWITEMCODE)
C           IF      FLAG = '0'
C           EVAL    SWMSG= 'ITEM NOT FOUND'
C           ELSE
C           EVAL    WTID='W_'+ SUB1
C           EVAL    NUM = %DEC(SUB1:8:0)+1
C           EVAL    SUB1=%CHAR(NUM)
C
C           EVAL    WITEMCODE=SWITEMCODE
C           EVAL    WUSERID=USR
C           EVAL    REASON=SREASON
C           EVAL    WDATE=%DATE()
C           EVAL    WTIME=%TIME()
C
C           EVAL    QTYLOSS=SQTYLOSS
C           WRITE   WTREC
EXEC SQL
    UPDATE STOCKMST SET STOCKINH = STOCKINH-:SQTYLOSS,
    VALUE = (STOCKINH - :SQTYLOSS) * PRICE
    WHERE ITEMCODE = :WITEMCODE;
C           EVAL    SWMSG = 'ENTERED SUCCESSFULLY'
C           ENDIF
C           ENDIF
C           ENDDO
C           EVAL    SWMSG = *BLANKS
C           EVAL    *INLR='1'
```

STOCKDSPLY

```

FSTOCKSCN3 CF E WORKSTN SFILE(STOCKSFL:RRN)
FSTOCKMST UF E K DISK
*
DSTOCKDS E DS EXTNAME(STOCKMST)
DRRN S 4 0
DSQLQRY S 200 INZ('')
*
C EVAL SCITEMCODE = 'ITEM CODE'
C EVAL SCITEMNAME = 'ITEM NAME'
C EVAL SCSTOCKINH = 'STOCKINH'
C EVAL SCREORDLVL = 'REORDLVL'
C EVAL SCVALUE = 'TOTAL VALUE'
C EVAL SCUNITMSR = 'UNIT MSR'
*
C DOW *IN12 = '0'
C IF *IN12 = '1'
C EVAL *IN12 = '0'
C LEAVE
C ENDIF
EXSR CLRSFL
EXSR LOADSFL
EXSR DSPSFL
EXSR OPTSFL
*
C IF *IN06 = '1'
C EVAL *IN06 = '0'
C CALL 'STOCKENTRY'
C ENDIF
*
C ENDDO
*
C EVAL *INLR = '1'
*-----*
*-----*
* CLEAR SUBFILE
*-----*
C CLRSFL BEGSR
C EVAL *IN93 = '1'
C WRITE STOCKCTL
C EVAL *IN93 = '0'
C ENDSR
*-----*
*-----*
* LOAD SUBFILE
*-----*
C LOADSFL BEGSR
C EVAL RRN = 0
C EXSR QRYSR
*
EXEC SQL
PREPARE S1 FROM :SQLQRY;
*
EXEC SQL
DECLARE CR CURSOR FOR S1;
*
EXEC SOL
OPEN CR;

```

```
*                                CLEAR                      STOCKDS
C
+
EXEC SQL
  FETCH CR INTO :STOCKDS;
*
C          DOW      SQLCODE <> 100 AND SQLCODE >= 0
C          EVAL     STITEMCODE = ITEMCODE
C          EVAL     STITEMNAME = ITEMNAME
C          EVAL     STUNITMSR = UNITMSR
C          EVAL     STSTOCKINH = %DEC(%EDITC(STOCKINH:'X')
C                           :8:0)
C          EVAL     STREORDLVL = %DEC(%EDITC(REORDLVL:'X')
C                           :8:0)
C          EVAL     STVALUE = %DEC(%EDITC(VALUE:'3'):12:2)
*
C          EVAL     RRN = RRN + 1
C          WRITE   STOCKSFL
*
EXEC SQL
```

```
      FETCH CR INTO :STOCKDS;
*
C          ENDDO
*
* EXEC SQL
* CLOSE CR;
*
C          ENDSR
*-----*
*-----*
* DISPLAY SUBFILE
*-----*
C      DSPPSFL      BEGSR
C                      WRITE      FOOTER
C                      EVAL       *IN92 = '1'
*
C                      IF        RRN > 0
C                      EVAL       *IN91 = '1'
C                      EVAL       *IN94 = '1'
```



```

HDFTACTGRP (*NO) BNDDIR('SFLBNNDIR')
FGOODRCVSCNCF   E           WORKSTN SFILE(GRCVSFL:RRN)
*
DRCVDDS         E DS          EXTNAME(GOODSRCVD)
DRRN            S             4  0
*
DQUOT           S             1    INZ('***')
DCOND1          S             100
DCOND2          S             100
DCOND3          S             100
DMAIN           S             500
*
DDATEVALIDID   PR            1  0
D                10
D                10
DVAL             S             1  0
*
DFLAG            S             1    INZ('0')
*

```

```

HDFTACTGRP (*NO) BNDDIR('SFLBNDDIR')
FGOODRCVSCNCF   E           WORKSTN SFILE(GRCVSFL:RRN)
*
DRCVDDS         E DS          EXTNAME(GOODSRCVD)
DRRN            S             4  0
*
DUSDTAARA      UDS          DTAARA
DUSER           10
*
DQUOT           S             1    INZ('***')
DCOND1          S             100
DCOND2          S             100
DCOND3          S             100
DMAIN           S             500
*
DDATEVALIDID   PR            1  0
D                10
D                10
DVAL             S             1  0
*
DFLAG            S             1    INZ('0')
*
EXEC SQL
  SET OPTION COMMIT = *NONE;
C           DOW      *IN12 = '0'
C           EVAL     SCURRUSER = USER
C           IF      *IN12 = '1'
C           LEAVE
C           ENDIF

*
*                                     -- DATA VALIDATION PROC
C           EVAL     VAL = DATEVALID(SFROMDATE:STODATE)

C           IF      VAL = 1
C           EVAL     SMSG1 = 'INVALID DATE FORMAT'
C           EVAL     SFROMDATE = *BLANKS
C           EVAL     STODATE = *BLANKS
C*
C           ITER
C           ELSEIF   VAL = 2

```

```

C           EVAL      SMSG1 = 'FROM DATE GREATER THAN TODATE'
C           EVAL      SFROMDATE = *BLANKS
C           EVAL      STODATE = *BLANKS
C           ELSE
C           EVAL      SMSG1 = *BLANKS
C           ENDIF
*
*                               ---VALIDATION PROC END
*                               ---VALIDATION ITEM

C           IF        SRITEMCODE <> *BLANKS
C           EVAL      FLAG = '0'

EXEC SQL
  SELECT '1' INTO :FLAG FROM GOODSRCVD WHERE RITEMCODE = :SRITEMCODE ;
*
C           IF        FLAG = '0'
C           EVAL      SMSG1 = ' ITEM DOES NOT EXIST '
C           ELSE
C           EVAL      SMSG1 = *BLANKS
C           ENDIF
*
C           ENDIF
*
*                               ---VALIDATION ITEM END

C           EXSR      CLRSR
C           EXSR      LOADSR
C           EXSR      DSPSR
*
C           ENDDO
*
C           EVAL      *INLR = '1'

*
*                               ---CLEAR SUBFILE SUBROUTINE
C     CLRSLR    BEGSR
C           EVAL      *IN93 = '1'
C           WRITE    GRCVCTL
C           EVAL      *IN93 = '0'
C           ENDSR
*
*                               --LOAD SUBFILE FROM DATABASE
C     LOADSR    BEGSR
C           EVAL      RRN=0

*
*                               --QUERIES
IF VAL = 0 ;

IF SRITEMCODE = *BLANKS ;
COND1 = '(SELECT RITEMCODE, REQUESTNO FROM GOODSRCVD)';
ELSE;
COND1 = 'SELECT RITEMCODE, REQUESTNO FROM GOODSRCVD ' +
'WHERE RITEMCODE = ' + QUOT + SRITEMCODE + QUOT ;
ENDIF ;

IF SFROMDATE = *BLANKS ;
COND2 = '(SELECT RITEMCODE, REQUESTNO FROM GOODSRCVD)';
ELSE;
COND2 = 'SELECT RITEMCODE, REQUESTNO FROM GOODSRCVD ' +
'WHERE RDATE >= ' + QUOT + SFROMDATE + QUOT ;

```

```

ENDIF ;

IF STODATE = *BLANKS ;
COND3 = '(SELECT RITEMCODE, REQUESTNO FROM GOODSRCVD)';
ELSE;
COND3 = 'SELECT RITEMCODE, REQUESTNO FROM GOODSRCVD ' +
'WHERE RDATE <= ' + QUOT + STODATE + QUOT ;
ENDIF ;

*
--DECLARE CURSOR
MAIN = 'SELECT * FROM GOODSRCVD ' +
      'WHERE (RITEMCODE, REQUESTNO) IN (' +
      COND1 + ' INTERSECT ' + COND2 + ' INTERSECT ' + COND3 + ')';

EXEC SQL
PREPARE S1 FROM :MAIN ;

EXEC SQL
DECLARE C1 CURSOR FOR S1 ;
*
--OPEN CURSOR
EXEC SQL
OPEN C1 ;

*
--FETCH INTO VARIABLES
EXEC SQL
FETCH C1 INTO :RCVDDS;
*
DOW SQLCODE <> 100;
  RRN += 1;
  SGRN = GRN ;
  SRITEMCODE = RITEMCODE ;
  SRUSERID = RUSERID ;
  SRDATE = RDATE ;
  SREQUESTNO = REQUESTNO ;
  SQTYRCVD = QTYRCVD ;
  WRITE GRCVSFL;
*
EXEC SQL
FETCH C1 INTO :RCVDDS;
*
C      ENDDO

*
--CLOSE CURSOR
EXEC SQL
CLOSE C1;
ENDIF;

C      ENDSR

*
--DISPLAY SUBFILE SUBROUTINE
C      DSPSR      BEGSR
C          WRITE    FOOTER
C          EVAL    *IN92='1'
*
C          IF       RRN>0
C          EVAL    *IN91='1'
C          EVAL    *IN94='1'
*
C          ENDIF
*
--DISPLAY CONTROL
*
C          EXFMT    GRCVCTL
*
C          EVAL    *IN91='0'
C          EVAL    *IN92='0'
C          EVAL    *IN94='0'
C          ENDSR
*****
 End of data ****

```

MOD_DATE

```
H*NOMAIN

*INLR = '1';

PDATEVALID      B          EXPORT
DDATEVALID      PI         1  0
DSFROMDATE       10
DSTODATE        10

DFLAG           S          1  0
DDATE1          S          D
DDATE2          S          D
DDI             S          10  0

C               if        SFROMDATE<>*blanks AND STODATE<>*blanks
C               EVAL      FLAG = 0

C               TEST(de)      SFROMDATE
C               IF        %error
C               EVAL      FLAG = 1
C               RETURN     FLAG
C               ENDIF

C               TEST(de)      STODATE
C               IF        %error
C               EVAL      FLAG = 1
C               RETURN     FLAG
C               ENDIF

C               EVAL      DATE1 = %DATE(SFROMDATE:*ISO)
C               EVAL      DATE2 = %DATE(STODATE:*ISO)
C               EVAL      DI = %DIFF(DATE2:DATE1:*DAYS)
C               if        DI < 0
C               EVAL      FLAG = 2
C               RETURN     FLAG
C               ENDIF

C               ELSE

C               if        SFROMDATE<>*blanks
C               EVAL      FLAG = 0

C               TEST(de)      SFROMDATE
C               IF        %error
C               EVAL      FLAG = 1
C               RETURN     FLAG
C               ENDIF
C               EVAL      FLAG = 0
C               ENDIF

C               if        STODATE<>*blanks
C               EVAL      FLAG = 0

C               TEST(de)      STODATE
C               IF        %error
C               EVAL      FLAG = 1
C               RETURN     FLAG
C               ENDIF
C               ENDIF

C               ENDIF
C               RETURN     FLAG
PDATEVALID      E
```

GISSSFL3

```

HDFTACTGRP(*NO) BNDDIR('SFLBNDDIR')
FGOODISSSCNCF   E           WORKSTN SFILE(GISSLFL:RRN)
*
DISSUDS          E DS          EXTNAME(GOODSISSD)
DRRN             S           4 0
*
DQUOT            S           1  inz(' ')
DCOND1           S           100
DCOND2           S           100
DCOND3           S           100
DMAIN            S           500
*
DDATEVALID      PR          1 0
D                 10
D                 10
DVAL              S           1 0
*
DFLAG             S           1
*
EXEC SQL
  SET OPTION COMMIT = *NONE;
*
C               DOW      *IN12 = '0'
C*               EVAL     SMSG1 = *blanks
C               IF       *IN12 = '1'
C               LEAVE
C               ENDIF
*               EVAL     VAL = DATEVALID(SFROMDATE:STODATE)
C
C               IF       VAL = 1
C               EVAL     SMSG1 = 'INVALID DATE FORMAT'
C               EVAL     SFROMDATE = *BLANKS
C               EVAL     STODATE = *BLANKS
C*               ITER
C               ELSEIF   VAL = 2
C               EVAL     SMSG1 = 'FROM DATE IS GREATER THAN TODATE'
C               EVAL     SFROMDATE = *BLANKS
C               EVAL     STODATE = *BLANKS
C               ELSE
C               EVAL     SMSG1 = *BLANKS
C               ENDIF
*               ---VALIDATION PROC END
*               ---VALIDATION ITEM
C               IF       S1ITEMCODE <> *BLANKS
C               EVAL     FLAG = '0'
EXEC SQL
  SELECT '1' INTO :FLAG FROM GOODSISSD WHERE IITEMCODE = :S1ITEMCODE;
*
C               IF       FLAG = '0'
C               EVAL     SMSG1 = ' ITEM DOES NOT EXIST '
C               ELSE
C               EVAL     SMSG1 = *BLANKS
C               ENDIF
*
C               ENDIF
*               ---VALIDATION ITEM END
C               EXSR     CLRCSR

```

```

C             EXSR      LOADSR
C             EXSR      DSPSR
*
C             ENDDO
*
C             EVAL      *INLR = '1'
*
*                         ---CLEAR SUBFILE SUBROUTINE
C     CLRSR      BEGSR
C             EVAL      *IN93 = '1'
C             WRITE    GISSCTL
C             EVAL      *IN93 = '0'
C             ENDSR
*
*                         --LOAD SUBFILE FROM DATABASE
C     LOADSR      BEGSR
C             EVAL      RRN=0
*
*
*                         --QUERIES
IF VAL = 0 ;
*
IF S1ITEMCODE = *BLANKS ;
COND1 = '(SELECT ISN FROM GOODSISSD)';
ELSE;
COND1 = 'SELECT ISN FROM GOODSISSD ' +
'WHERE IITEMCODE = ' + QUOT + S1ITEMCODE + QUOT ;
ENDIF ;
*
IF SFROMDATE = *BLANKS ;
COND2 = '(SELECT ISN FROM GOODSISSD)';
ELSE;
COND2 = 'SELECT ISN FROM GOODSISSD ' +
'WHERE IDATE >= ' + QUOT + SFROMDATE + QUOT ;
ENDIF ;
*
IF STODATE = *BLANKS ;
COND3 = '(SELECT ISN FROM GOODSISSD)';
ELSE;
COND3 = 'SELECT ISN FROM GOODSISSD ' +
'WHERE IDATE <= ' + QUOT + STODATE + QUOT ;
ENDIF ;
*
*
*                         --DECLARE CURSOR
MAIN = 'SELECT * FROM GOODSISSD ' +
'WHERE ISN IN (' +
COND1 + ' INTERSECT ' + COND2 + ' INTERSECT ' + COND3 + ')';
*
EXEC SQL
PREPARE S1 FROM :MAIN ;
*
EXEC SQL
DECLARE C1 CURSOR FOR S1 ;                                --OPEN CURSOR
*
EXEC SQL
OPEN C1 ;
*
*                         --FETCH INTO VARIABLES

```

```

EXEC SQL
  FETCH C1 INTO :ISSUDS;
*
  DOW SQLCODE <> 100;
    RRN += 1;
    WRITE GISSSFL;
*
  EXEC SQL
  FETCH C1 INTO :ISSUDS;
*
C          ENDDO

*
--CLOSE CURSOR

EXEC SQL
CLOSE C1;

ENDIF;
C          ENDSR

*          --DISPLAY SUBFILE SUBROUTINE
C      DSPSR      BEGSR
C          WRITE      FOOTER
C          EVAL      *IN92='1'
*
C          IF        RRN>0
C          EVAL      *IN91='1'
C          EVAL      *IN94='1'
C          ENDIF
*
*          --DISPLAY CONTROL
*
C          EXFMT      GISSCTL
*
C          EVAL      *IN91='0'
C          EVAL      *IN92='0'
C          EVAL      *IN94='0'
C          ENDSR

```

WTSFL

```

RDFTACTGRP (*NO) BNDDIR('SFLBNDDIR')
FWTSCN     CF   E           WORKSTN SFILE(WTSFL:RRN)
*
DWTD$      E DS           EXTNAME (WEARTEAR)
DRRN       S             4  0
*
DQUOT      S             1  INZ('****')
DCOND1     S             100
DCOND2     S             100
DCOND3     S             100
DMAIN      S             500
*
DDATEVALID PR            1  0
D           10
D           10
DVAL       S             1  0
*
DFLAG      S             1  INZ('0')
*
```

```

EXEC SQL
    SET OPTION COMMIT = *NONE;
C           DOW      *IN12 = '0'
C           EVAL     SMSG='HELLO'
C           IF      *IN12 = '1'
C           LEAVE
C           ENDIF
*
*                               -- DATA VALIDATION PROC
C           EVAL     VAL = DATEVALID(SFROMDATE:STODATE)
*
C           IF      VAL = 1
C           EVAL     SMSG = 'INVALID DATE FORMAT'
C           EVAL     SFROMDATE = *BLANKS
C           EVAL     STODATE = *BLANKS
C           ELSEIF   VAL = 2
C           EVAL     SMSG = 'FROM DATE IS GREATER THAN TODATE'
C           EVAL     SFROMDATE = *BLANKS
C           EVAL     STODATE = *BLANKS
C           ELSE
C           EVAL     SMSG = *BLANKS
C           ENDIF
*
*                               ---VALIDATION PROC END
*                               ---VALIDATION ITEM
C           EVAL     FLAG='0'
C           IF      SITEMCODE <> *BLANKS
EXEC SQL
    SELECT '1' INTO :FLAG FROM WEARTEAR WHERE WITEMCODE = :SITEMCODE ;
*
C           IF      FLAG = '0'
C           EVAL     SMSG = ' ITEM DOES NOT EXIST '
C           ELSE
C           EVAL     SMSG = *BLANKS
C           ENDIF
*
C           ENDIF
*                               ---VALIDATION ITEM END
*
*
C           EXSR     CLRSR
C           EXSR     LOADSR
C           EXSR     DSPSR
*
C           ENDDO
*
C           EVAL     *INLR = '1'
*
*                               ---CLEAR SUBFILE SUBROUTINE
C     CLRSPR    BEGSR
C     EVAL     *IN93 = '1'
C     WRITE    WTCTL
C     EVAL     *IN93 = '0'
C     ENDSR
*                               --LOAD SUBFILE FROM DATABASE
C     LOADSPR   BEGSR
C     EVAL     RRN=0

```

```

*
--QUERIES

IF VAL = 0;

IF SITEMCODE = *BLANKS ;
COND1 = '(SELECT WTID FROM WEARTEAR)';
ELSE;
COND1 = 'SELECT WTID FROM WEARTEAR ' +
'WHERE WITEMCODE = ' + QUOT + SITEMCODE + QUOT ;
ENDIF ;

IF SFROMDATE = *BLANKS ;
COND2 = '(SELECT WTID FROM WEARTEAR )';
ELSE;
COND2 = 'SELECT WTID FROM WEARTEAR ' +
'WHERE WDATE >= ' + QUOT + SFROMDATE + QUOT ;
ENDIF ;

IF STODATE = *BLANKS ;
COND3 = '(SELECT WTID FROM WEARTEAR )';
ELSE;
COND3 = 'SELECT WTID FROM WEARTEAR ' +
'WHERE WDATE <= ' + QUOT + STODATE + QUOT ;
ENDIF ;

*
--DECLARE CURSOR

MAIN = 'SELECT * FROM WEARTEAR ' +
      'WHERE WTID IN (' +
      COND1 + ' INTERSECT ' + COND2 + ' INTERSECT ' + COND3 + ')';

EXEC SQL
PREPARE S1 FROM :MAIN ;

EXEC SQL
DECLARE C1 CURSOR FOR S1 ;
*
--OPEN CURSOR
EXEC SQL
OPEN C1 ;

*
--FETCH INTO VARIABLES
EXEC SQL
FETCH C1 INTO :WTDS;
*
DOW SQLCODE <> 100;
  RRN += 1;
  WRITE WTSFL;
*
EXEC SQL
FETCH C1 INTO :WTDS;
*
C          ENDDO

*
--CLOSE CURSOR
EXEC SQL
CLOSE C1;

ENDIF;

```

```

C          ENDSR

*
*          --DISPLAY SUBFILE SUBROUTINE
C      DSPSR      BEGSR
C          WRITE     FOOTER
C          EVAL      *IN92='1'
*
C          IF       RRN>0
C          EVAL      *IN91='1'
C          EVAL      *IN94='1'
C          ENDIF
*
*          --DISPLAY CONTROL
*
C          EXFMT     WTCTL
*
C          EVAL      *IN91='0'
C          EVAL      *IN92='0'
C          EVAL      *IN94='0'
C          ENDSR

```

BLWLVLRPG

```

HOPTION(*NODEBUGIO:*SRCSTMT)
FBELLOWLVSCNCF   E           WORKSTN SFILE(BELOWSFL:RRN) INFDS(DS1)
DRRN             S           4 0
DDS1              DS
DSUB1            378    379I 0
DI                S           2 0
DCNT              S           4 0
DIND1             S           1 0
DSTOCK            S           8 0
DSTOCKDS          E DS        EXTNAME('STOCKMST') DIM(100)
D                   QUALIFIED
*-----
EXEC SQL
SET OPTION COMMIT = *NONE,DATFMT = *ISO,TIMFMT = *ISO;
*-----
C          EVAL      RCD_NBR = 1
C          DOW      *IN12 = '0'
C          IF       *IN12 = '1'
C          LEAVE

```

```

C          *
C          ENDDO
C          EVAL      *INLR = '1'
*****+
C      CLRSFL      BEGSR
C          EVAL      *IN93 = '1'
C          WRITE    BELOWCTL
C          EVAL      *IN93 = '0'
C          ENDSR
*****+
BEGSR LOADSFL;
RRN = 0;

```

```

EXEC SQL
SELECT COUNT(*) INTO :CNT FROM STOCKMST
WHERE STOCKINH < REORDLVL;

EXEC SQL
DECLARE C1 CURSOR FOR SELECT * FROM STOCKMST
WHERE STOCKINH < REORDLVL;

EXEC SQL
OPEN C1;

EXEC SQL
FETCH C1 FOR :CNT ROWS INTO :STOCKDS;
IF SQLCODE = 0;
FOR I = 1 TO CNT;
  IF STOCKDS(I).ITEMCODE = '';
    LEAVE ;
  ENDIF ;
  SITEMCODE = STOCKDS(I).ITEMCODE;
  SITEMNAME = STOCKDS(I).ITEMNAME;
  SUNITMSR = STOCKDS(I).UNITMSR;
  SSTOCKINH = STOCKDS(I).STOCKINH;
  SREORDLVL = STOCKDS(I).REORDLVL;
  SPRICE = STOCKDS(I).PRICE;
  SVALUE = STOCKDS(I).VALUE;
  RRN = RRN + 1;
  WRITE BELOWSFL;
  IF SPOS = STOCKDS(I).ITEMCODE;
    RCD_NBR = RRN;
  ENDIF;
ENDFOR;
ENDIF;
EXEC SQL
CLOSE C1;

ENDSR;
*****
C      DSPLYSFL      BEGSR
C          WRITE      FOOTER
C          EVAL      *IN92 = '1'
C          IF        RRN > 0
C          EVAL      *IN91 = '1'
C          EVAL      *IN94 = '1'
C          ENDIF
C          EXFMT      BELOWCTL
C          IF        SUB1 > 0
C          EVAL      RCD_NBR = SUB1
C          ELSE
C          EVAL      RCD_NBR = 1
C          ENDIF
C          EVAL      *IN92 = '0'
C          EVAL      *IN91 = '0'
C          EVAL      *IN94 = '0'
C          ENDSR

```

CLFMCGRPT

```
CALL PRTFMCGRPT
      CPYSPLF      FILE(FMCGRPT) TOFILE(FMCGFLAT) SPLNBR(*LAST)
      RUNQRY *N FMCGFLAT
```

PRTFMCGRPT

```
FFMCGRPT O E           PRINTER OFLIND(*IN99)
*
DSTOCKMSTDSDS
DDSITEMCODE          10
DDSITEMNAME          15
DDSUNITMSR           3
DDSSTOCKINH          8 0
DDSREORDLVL          8 0
DDSPRICE              10 2
DDSVALUE              12 2
DDSTOTALISS          8 0
*
C           EVAL     DATE = %DATE()
C           WRITE    HEADER
C           WRITE    HEADINGS
*
C/EXEC SQL
C+ DECLARE CR CURSOR FOR
C+ SELECT S.ITEMCODE,S.ITEMNAME,S.UNITMSR,S.STOCKINH,S.REORDLVL,
C+
C+ S.PRICE,S.VALUE,COALESCE(SUM(G.QTYISSD),0) AS TOTALISSD
C+ FROM STOCKMST S LEFT JOIN GOODSISSD G ON
C+ S.ITEMCODE = G.IITEMCODE
C+ GROUP BY S.ITEMCODE,S.ITEMNAME,S.UNITMSR,S.STOCKINH,S.REORDLVL,
C+ S.PRICE,S.VALUE
C+ ORDER BY TOTALISSD DESC
C/END-EXEC
*
EXEC SQL
OPEN CR;
*
C           CLEAR             STOCKMSTDSDS
*
EXEC SQL
FETCH CR INTO :STOCKMSTDSDS;
*
C           DOW     SQLCODE<>100 AND SQLCODE>=0
C           IF      *IN99 = '1'
C           EVAL   *IN99 = '0'
C           WRITE  HEADER
```

```

C           WRITE      HEADINGS
C           ENDIF
*
C           EVAL      RPITEMCODE = DSITEMCODE
C           EVAL      RPITEMNAME = DSITEMNAME
C           EVAL      RPUNITMSR = DSUNITMSR
C           EVAL      RPSTOCKINH = DSSTOCKINH
C           EVAL      RPREORDLVL = DSTOTALISS
C           EVAL      RPPRICE = DSPRICE
C           EVAL      RPVALUE = DSPRICE*DSTOTALISS
*
C           WRITE      DATA
*
EXEC SQL
  FETCH CR INTO :STOCKMSTD$;
*
C           ENDDO
*
EXEC SQL
  CLOSE CR;

```

```

*
C           EVAL      *INLR = '1'

```

CLGISSDRPT

```

PGM
  DCL      VAR(&STATUS) TYPE(*CHAR)          LEN(1) +
            VALUE('1')
  CALL    PGM(GISSRPTRPG)
  RTVDTAARA DTAARA(RPTDTAARA *ALL) RTNVAR(&STATUS)
  IF      COND(&STATUS = '0') THEN(DO)
  CPYSPLF   FILE(GISSRPT) TOFILE(GISSFLAT) SPLNBR(*LAST)
  RUNQRY *N GISSFLAT
  CHGDTAARA DTAARA(RPTDTAARA *ALL) VALUE('1')
  ENDDO
ENDPGM

```

GISSRPTRPG

```

FGISSRPTSCNCF   E          WORKSTN
FGISSRPT   O   E          PRINTER OFFLIND(*IN99)
DRRN           S          4   0
DI             S          2   0
DDI            S          10  0
DDATE1         S          D
DDATE2         S          D
DFROMD         S          D
DCNT           S          4   0
DIND1          S          1   0
DIND2          S          1   0
DGOODDS        E DS      EXTNAME('GOODSISSD') DIM(100)
D              QUALEFIED
DRPTDTAARA    UDS      DTAARA
DSTATUS         1
EXEC SQL
SET OPTION COMMIT = *NONE, DATFMT = *ISO, TIMFMT = *ISO;
*
C              DOW      *IN12 = '0'

```

```

C              EXFMT      RPTENTRY
C              EVAL       SMSG = ''
C              IF        *IN12 = '1'
C              LEAVE
C              ENDIF
C              if        SFROMDATE<>*blanks AND STODATE<>*blanks
C              EVAL       SMSG = ' '
C              TEST(de)    SFROMDATE
C              IF        %error
C              EVAL       SMSG='Invalid date format'
C              EVAL       SFROMDATE = *BLANKS
C              EVAL       STODATE = *BLANKS
C              iter
C              ENDIF
C              TEST(de)    STODATE
C              IF        %error
C              EVAL       SMSG='Invalid date format'
C              EVAL       SFROMDATE = *BLANKS
C              EVAL       STODATE = *BLANKS
C              iter

```

```

C           ENDIF

C           EVAL      DATE1 = %DATE(SFROMDATE)
C           EVAL      DATE2 = %DATE(STODATE)
C           EVAL      DI = %DIFF(DATE2:DATE1:*DAYS)
C           if       DI < 0
C           EVAL      SMSG='FROM DATE > TODATE'
C           EVAL      SFROMDATE = *BLANKS
C           EVAL      STODATE = *BLANKS
C           ITER
C           ENDIF

C           ELSE

C           if       SFROMDATE<>*blanks
C           EVAL      SMSG = ''
C           TEST(de)          SFROMDATE
C           IF       %error
C           EVAL      SMSG='INVALID DATE FORMAT'
C           EVAL      SFROMDATE = *BLANKS

```

```

C           iter
C           ENDIF
C           EVAL      SMSG = ''
C           ENDIF

C           if       STODATE<>*blanks
C           EVAL      SMSG = ''
C           TEST(de)          STODATE
C           IF       %error
C           EVAL      SMSG='INVALID DATE FORMAT'
C           EVAL      STODATE = *BLANKS
C           iter
C           ENDIF
C           ENDIF
C           ENDIF
C           EXSR      REPORTSR
C           EVAL      STATUS = '0'
*
C           ENDDO
C           EVAL      *INLR = '1'

```

```

*****
C      REPORTSR      BEGSR
IF SFROMDATE = *BLANKS;
EXEC SQL
SELECT MIN(IDATE) INTO :FROMMD FROM GOODSISSD;
ELSE;
FROMMD = %DATE(SFROMDATE);
ENDIF;

IF STODATE = *BLANKS;
STODATE = %CHAR(%DATE());
ENDIF;

EXEC SQL
SELECT COUNT(*) INTO :CNT FROM GOODSISSD
WHERE IDATE BETWEEN :FROMMD AND :STODATE;

EXEC SQL
DECLARE C1 CURSOR FOR SELECT * FROM GOODSISSD
WHERE IDATE BETWEEN :FROMMD AND :STODATE;

```

```
C           EVAL      DATE = %DATE()
C           EVAL      FROMDATE = %CHAR(FROMD)
C           EVAL      TODATE = STODATE
C           WRITE     HEADER
C           WRITE     BLANKS
C           WRITE     HEADING
C           WRITE     BLANKS1

EXEC SQL
OPEN C1;

EXEC SQL
    FETCH C1 FOR :CNT ROWS INTO :GOODDS;

IF *IN99 = '1';
C           EVAL      FROMDATE = %CHAR(FROMD)
C           EVAL      TODATE = STODATE
C           WRITE     HEADER
C           WRITE     BLANKS
C           WRITE     HEADING
```

```
C           WRITE     BLANKS1
*IN99 = '0';
ENDIF;

IF SQLCODE = 0;
FOR I = 1 TO CNT;
    IF GOODDS(I).ISN = '';
        LEAVE ;
    ENDIF ;
    ISN = GOODDS(I).ISN;
    IITEMCODE = GOODDS(I).IITEMCODE;
    IUSERID = GOODDS(I).IUSERID;
    IDATE = GOODDS(I).IDATE ;
    ITIME = GOODDS(I).ITIME;
    QTYISSD = GOODDS(I).QTYISSD;
C           WRITE     RECORD
ENDFOR;
ENDIF;

SMSG = 'REPORT GENERATED';
```

```
EXEC SQL
CLOSE C1;
C           EVAL      STATUS = '1'
C           EVAL      *IN12 = '1'
C           ENDSR
***** End of data *****
```

CLGRCVDRPT

```
PGM
      DCL      VAR(&STATUS) TYPE(*CHAR)           LEN(1) +
                  VALUE('1')
      CALL      PGM(GRCVRPTRPG)
      RTVDTAARA DTAARA(RPTDTAARA *ALL) RTNVAR(&STATUS)
      IF       COND(&STATUS = '0') THEN(DO)
      CPYSPLF   FILE(GRCVRPT) TOFILE(GRCVFLAT) SPLNBR(*LAST)
      RUNQRY *N GRCVFLAT
      CHGDTAARA DTAARA(RPTDTAARA *ALL) VALUE('1')
      ENDDO
ENDPGM
```

GRCVRPTRPG

```
FGRCVRPTSCNCF E WORKSTN
FGRCVRPT O E PRINTER OFLIND(*IN99)
DRRN S 4 0
DI S 2 0
DDI S 10 0
DDATE1 S D
DDATE2 S D
DFROMD S D
DCNT S 4 0
DIND1 S 1 0
DIND2 S 1 0
DGOODDS E DS EXTNAMES('GOODSRCVD') DIM(100)
D QUALIFIED
DRPTDTAARA UDS DTAARA
DSTATUS 1
EXEC SQL
SET OPTION COMMIT = *NONE, DATFMT = *ISO, TIMFMT = *ISO;
*
C          DOW      *IN12 = '0'
```

```
C          EXFMT RPTENTRY
C          EVAL  SMSG = ''
C          IF    *IN12 = '1'
C          LEAVE
C          ENDIF
C          if    SFROMDATE<>*blanks AND STODATE<>*blanks
C          EVAL  SMSG = ' '
C          TEST(de)      SFROMDATE
C          IF    %error
C          EVAL  SMSG='Invalid date format'
C          EVAL  SFROMDATE = *BLANKS
C          EVAL  STODATE = *BLANKS
C          iter
C          ENDIF
C          TEST(de)      STODATE
C          IF    %error
C          EVAL  SMSG='Invalid date format'
C          EVAL  SFROMDATE = *BLANKS
C          EVAL  STODATE = *BLANKS
C          iter
```

```

C           ENDIF

C           EVAL      DATE1 = %DATE(SFROMDATE)
C           EVAL      DATE2 = %DATE(STODATE)
C           EVAL      DI = %DIFF(DATE2:DATE1:*DAYS)
C           if       DI < 0
C           EVAL      SMSG='FROM DATE > TODATE'
C           EVAL      SFROMDATE = *BLANKS
C           EVAL      STODATE = *BLANKS
C           ITER
C           ENDIF

C           ELSE

C           if       SFROMDATE<>*blanks
C           EVAL      SMSG = ''
C           TEST(de)          SFROMDATE
C           IF       %error
C           EVAL      SMSG='INVALID DATE FORMAT'
C           EVAL      SFROMDATE = *BLANKS

```

```

C           iter
C           ENDIF
C           EVAL      SMSG = ''
C           ENDIF

C           if       STODATE<>*blanks
C           EVAL      SMSG = ''
C           TEST(de)          STODATE
C           IF       %error
C           EVAL      SMSG='INVALID DATE FORMAT'
C           EVAL      STODATE = *BLANKS
C           iter
C           ENDIF
C           ENDIF
C           ENDIF
C           EXSR      REPORTSR
C           EVAL      STATUS = '0'
*
C           ENDDO
C           EVAL      *INLR = '1'
*****
```

```

C     REPORTSR      BEGSR
IF SFROMDATE = *BLANKS;
EXEC SQL
SELECT MIN(RDATE) INTO :FROMD FROM GOODSRCVD;
ELSE;
FROMD = %DATE(SFROMDATE);
ENDIF;

IF STODATE = *BLANKS;
STODATE = %CHAR(%DATE());
ENDIF;

EXEC SQL
SELECT COUNT(*) INTO :CNT FROM GOODSRCVD
WHERE RDATE BETWEEN :FROMD AND :STODATE;

EXEC SQL
DECLARE C1 CURSOR FOR SELECT * FROM GOODSRCVD
WHERE RDATE BETWEEN :FROMD AND :STODATE;
```

```

C           EVAL      DATE = %DATE()
C           EVAL      FROMDATE = %CHAR(FROMD)
C           EVAL      TODATE = STODATE
C           WRITE     HEADER
C           WRITE     BLANKS
C           WRITE     HEADING
C           WRITE     BLANKS1

EXEC SQL
OPEN C1;

EXEC SQL
FETCH C1 FOR :CNT ROWS INTO :GOODDS;

IF *IN99 = '1';
C           EVAL      FROMDATE = %CHAR(FROMD)
C           EVAL      TODATE = STODATE
C           WRITE     HEADER
C           WRITE     BLANKS
C           WRITE     HEADING

C           WRITE     BLANKS1
*IN99 = '0';
ENDIF;

IF SQLCODE = 0;
FOR I = 1 TO CNT;
  IF GOODDS(I).GRN = '';
    LEAVE ;
  ENDIF ;
  GRN = GOODDS(I).GRN;
  RITEMCODE = GOODDS(I).RITEMCODE;
  RUSERID = GOODDS(I).RUSERID;
  RDATE = GOODDS(I).RDATE ;
  RTIME = GOODDS(I).RTIME;
  REQUESTNO = GOODDS(I).REQUESTNO;
  QTYRCVD = GOODDS(I).QTYRCVD;
  GOODSPRICE = GOODDS(I).GOODSPRICE;
C           WRITE     RECORD
ENDFOR;
ENDIF;

SMSG = 'REPORT GENERATED';

EXEC SQL
CLOSE C1;
C           EVAL      STATUS = '1'
C           EVAL      *IN12 = '1'
C           ENDSR

```

SHOWSPOOLF

```
PGM
    CALL      PGM(PURREQRPT)
    CPYSPLF  FILE(REORDERRPT) TOFILE(PURREQFLAT) SPLNBR(*LAST) +
              MBROPT(*REPLACE)
END:      RUNQRY   QRYFILE(PURREQFLAT)
ENDPGM
```

PURREQRPT

```
FSTOCKMST IF E          K DISK
FREORDERRPTO E          PRINTER OFLIND(*IN99)

*
*

DRNDTAARA     UDS          DTAARA
DSUB1           S            8
DNUM            S            8 0
C               EVAL         DATE=%DATE()
C               EVAL         rrequestno='R_'+ SUB1
C               EVAL         NUM = %DEC(SUB1:8:0)+1
C               EVAL         SUB1=%CHAR(NUM)
C               WRITE        HEADER
C               WRITE        HEADING
C *LOVAL       SETLL        STOCKMST
C               READ         STOCKMST
C               DOW          NOT %EOF()
*
C               IF           *IN99='1'
C               WRITE        HEADER

-
C               WRITE        HEADING
C               EVAL         *IN99='0'
C               ENDIF

*
C               IF           STOCKINH<=REORDLVL
C               WRITE        FIELDS
C               ENDIF
C               READ         STOCKMST
C               ENDDO

*SUMMARY
C               WRITE        FOOTER
C               EVAL         *INLR='1'
```

LOGIN

- 1) The login screen is exformatted recursively until the user successfully enters right credentials or press F3.
- 2) Check if all the fields are entered or not.
- 3) If entered, the username and password taken from screen is validated with the ones in pf.
- 4) If right credentials, designation of that user is taken from pf and stored into a variable.
- 5) The user name is stored in a data structure data area for later use in the session.

*LOCK IN USDTAARA

EVAL %SUBST(USDTAARA:1:10) = USER

OUT USDTAARA

- 5) Menu is shown based on the designation - employee, manager or admin using QCMDEXEC program which helps to run CL commands in RPGLE
- 6) If all fields are not entered or user failed validation -- msg -- 'WRONG CREDENTIALS'

STOCK MASTER ENTRY

- 1) The stock master entry screen is exformatted recursively until the user successfully enters correct details or press F12.
- 2) Check if all the fields are entered or not else -- 'ALL FIELDS REQUIRED'.
- 3) If entered, unit of measurement is checked -- kg, l, or qty -- if not correct 'INCORRECT UNIT OF MEASURE'.
- 4) If correct, the details are inserted into the pf (value = stock-in-hand*price) -- 'RECORD ADDED SUCCESSFULLY'.
- 5) The itemcode is taken from corresponding data area for this insert.
- 6) The itemcode is then incremented and overwritten in the data area.

GOODS RECEIVED ENTRY

- 1) The goods received entry screen is exformatted recursively until the user successfully enters correct details or press F12.
- 2) Check if all the fields are entered or not else -- 'ALL FIELDS REQUIRED'.

- 3) If entered, item code is validated using external procedure(in external module -- ITEMCHECK) -- if not correct 'ITEM NOT FOUND'.
- 4) If correct, the details are inserted into the goods received pf and stock in hand and value is updated in stock master pf -- 'RECORD ADDED SUCCESSFULLY'.
- 5) The userid is taken from corresponding data area for this insert.
- 6) The userid is then incremented and overwritten in the data area.

GOODS ISSUED ENTRY

- 1) The goods issued entry screen is exformatted recursively until the user successfully enters correct details or press F12.
- 2) Check if all the fields are entered or not else -- 'BLANK FIELDS DETECTED'.
- 3) If entered, item code is validated using external procedure(in external module -- ITEMCHECK) -- if not correct 'ITEM NOT FOUND'.
- 4) Then stock in hand is checked, whether stock is available for the issue that is if issue qty < stock in hand.
- 5) If already existing issue is entered -- 'DUPLICATE RECORD' -- using extender and %ERROR.
- 6) If correct, the details are inserted into the goods received pf (ISN taken from data area) and stock in hand and value is updated in stock master pf -- 'RECORD ADDED'.
- 7) The userid is taken from corresponding data area for this insert.
- 8) The userid is then incremented and overwritten in the data area.

WEAR AND TEAR ENTRY

- 1) The wear and tear entry screen is exformatted recursively until the user successfully enters correct details or press F12.
- 2) Check if all the fields are entered or not else -- 'ALL FIELDS REQUIRED'.
- 3) If entered, item code is validated using external procedure(in external module -- ITEMCHECK) -- if not correct 'ITEM NOT FOUND'.
- 4) If correct, the details are inserted into the wear or tear pf (WearTearID taken from data area) and stock in hand and value is updated in stock master pf -- 'RECORD ADDED'.
- 5) The userid is taken from corresponding data area for this insert.
- 6) The userid is then incremented and overwritten in the data area.

USER ENTRY

- 1) The user entry screen is exformatted recursively until the user successfully enters correct details or press F12.
- 2) Check if all the fields are entered or not else -- 'ALL FIELDS REQUIRED'.
- 3) If entered, designation is verified -- EMPLOYEE, ADMIN, MANAGER -- if not correct 'INVALID DESIGNATION'.
- 4) If duplicate record (takes count of userid and checks if = 0) -- 'USER ALREADY EXISTS'.
- 5) Password and re entered password checked -- if not same -- 'PASSWORD AND REENTERED DIFF'.
- 6) If correct, the details are inserted into the user file -- 'RECORD ADDED SUCCESSFULLY'.

STOCK MASTER SUBFILE

- 1) The field names are evaluated since order by field names are to be done.
- 2) The subfile screen is exformatted recursively until the user successfully enters correct details or press F12.
- 3) If F6 is pressed entry screen is shown
- 4) Subfile cleared
- 5) Load
rrn = 0
Select query prepared based on orderby of fields, declare cursor for that statement, open cursor, fetch records, write into subfile, increment rrn and again fetch, close cursor.
- 6) Display subfile
- 7) Check options using readc and dow -- if 2- update stock master pf(same checks as entry) , if 4 - delete record from pf.

ITEMS < REORDER LEVEL SUBFILE

- 1) The subfile screen is exformatted recursively until the user successfully enters correct details or press F12.
- 2) Take count
- 3) Bulk fetch into DS
- 4) Display
- 5) If position = itemcode update rcdnbr to rrn

GOODS ISSUED SUBFILE

- 1) Uses MOD_DATE external module with procedure DATEVALID for date validations
- 2) If both dates are not blank, it checks if they are valid date formats--if invalid flag is made 1--else checks the difference bw the 2 dates---if diff < 0--flag is made 2
- 3) If only one is blanks -- it checks if they are valid date formats--if invalid flag is made 1
- 4) Finally the flag is returned
- 5) If flag = 1, msg --'INVALID DATE FORMAT' -- clear the dates and then iter
- 6) If flag = 2, msg -- 'FROM DATE IS GREATER THAN TO DATE' -- clear dates
- 7) Then item code is checked -- if not found -- 'ITEM NOT FOUND'.
- 8) Now four conditions are made
 - cond 1 --> select isn from pf (if itemcode is blanks) else select isn where itemcode is present
 - cond 2 --> select isn from pf (if fromdate is blanks) else select isn where date >= fromdate
 - cond 3 --> select isn from pf (if todate is blanks) else select isn where date <= todate
 - main --> select * from pf where isn in (cond1 intersect 2 intersect 3)
- 9) prepare statement, declare cursor, fetch the records and display

GOODS RECEIVED SUBFILE

- 1) Uses MOD_DATE external module with procedure DATEVALID for date validations
- 2) If both dates are not blank, it checks if they are valid date formats--if invalid flag is made 1--else checks the difference bw the 2 dates---if diff < 0--flag is made 2
- 3) If only one is blanks -- it checks if they are valid date formats--if invalid flag is made 1
- 4) Finally the flag is returned
- 5) If flag = 1, msg --'INVALID DATE FORMAT' -- clear the dates and then iter
- 6) If flag = 2, msg -- 'FROM DATE IS GREATER THAN TO DATE' -- clear dates
- 7) Then item code is checked -- if not found -- 'ITEM NOT FOUND'.
- 8) Now four conditions are made
 - cond 1 --> select isn from pf (if itemcode is blanks) else select isn where itemcode is present
 - cond 2 --> select isn from pf (if fromdate is blanks) else select isn where date >= fromdate
 - cond 3 --> select isn from pf (if todate is blanks) else select isn where date <= todate

main --> select * from pf where isn in (cond1 intersect 2 intersect 3)

9) prepare statement, declare cursor, fetch the records and display

WEAR/TEAR SUBFILE

- 1) Uses MOD_DATE external module with procedure DATEVALID for date validations
- 2) If both dates are not blank, it checks if they are valid date formats--if invalid flag is made 1--else checks the difference bw the 2 dates---if diff < 0--flag is made 2
- 3) If only one is blanks -- it checks if they are valid date formats--if invalid flag is made 1
- 4) Finally the flag is returned
- 5) If flag = 1, msg --'INVALID DATE FORMAT' -- clear the dates and then iter
- 6) If flag = 2, msg -- 'FROM DATE IS GREATER THAN TO DATE' -- clear dates
- 7) Then item code is checked -- if not found -- 'ITEM NOT FOUND'.
- 8) Now four conditions are made

cond 1 --> select isn from pf (if itemcode is blanks) else select isn where itemcode is present

cond 2 --> select isn from pf (if fromdate is blanks) else select isn where date >= fromdate

cond 3 --> select isn from pf (if todate is blanks) else select isn where date <= todate

main --> select * from pf where isn in (cond1 intersect 2 intersect 3)

9) prepare statement, declare cursor, fetch the records and display

GOODS ISSUED REPORT

- 1) Declare a variable called 'status' with initial value as 1 in CLP
- 2) Call the report program
- 3) Retrieve the data from the data area containing status.
- 4) If status is 0--> copy the last spool file into a flatfile and runqry that flat file
- 5) Change the data areas value into 1

2) elaboration

- 1) Uses MOD_DATE external module with procedure DATEVALID for date validations
- 2) If both dates are not blank, it checks if they are valid date formats--if invalid flag is made 1--else checks the difference bw the 2 dates---if diff < 0--flag is made 2
- 3) If only one is blanks -- it checks if they are valid date formats--if invalid flag is made 1

- 4) Finally the flag is returned
- 5) If flag = 1, msg --'INVALID DATE FORMAT' -- clear the dates and then iter - set status
- 6) If flag = 2, msg -- 'FROM DATE IS GREATER THAN TO DATE' -- clear dates
- 7) If from date is blank equate it to min(date) found in pf.
- 8) If to date is blank equate it to current date
- 9) Fetch the records into cursor from pf and then write into report
- 10) Set status as 1
- 11) set status back to 0

GOODS RECEIVED REPORT

- 1) Declare a variable called 'status' with initial value as 1 in CLP
 - 2) Call the report program
 - 3) Retrieve the data from the data area containing status.
 - 4) If status is 0--> copy the last spool file into a flatfile and runqry that flat file
 - 5) Change the data areas value into 1
-
- 2) elaboration
 - 1) Uses MOD_DATE external module with procedure DATEVALID for date validations
 - 2) If both dates are not blank, it checks if they are valid date formats--if invalid flag is made 1--else checks the difference bw the 2 dates---if diff < 0--flag is made 2
 - 3) If only one is blanks -- it checks if they are valid date formats--if invalid flag is made 1
 - 4) Finally the flag is returned
 - 5) If flag = 1, msg --'INVALID DATE FORMAT' -- clear the dates and then iter - set status
 - 6) If flag = 2, msg -- 'FROM DATE IS GREATER THAN TO DATE' -- clear dates
 - 7) If from date is blank equate it to min(date) found in pf.
 - 8) If to date is blank equate it to current date
 - 9) Fetch the records into cursor from pf and then write into report
 - 10) Set status as 1
 - 11) set status back to 0

PURCHASE REQUEST REPORT

1) Call rpgle report prog from CLP

1) Take request number from data area

2) Fetch the records of items < reorder levele from stockmaster and then write into report.

2) Copy last spool file into flat file and runqry.

FMCG REPORT

1) Call rpgle report prog from CLP

```
1)SELECT S.ITEMCODE,S.ITEMNAME,S.UNITMSR,S.STOCKINH,S.REORDLVL,  
S.PRICE,S.VALUE,COALESCE(SUM(G.QTYISSD),0) AS TOTALISSD FROM STOCKMST S LEFT JOIN  
GOODSISSD G ON S.ITEMCODE = G.IITEMCODE GROUP BY  
S.ITEMCODE,S.ITEMNAME,S.UNITMSR,S.STOCKINH,S.REORDLVL, S.PRICE,S.VALUE ORDER BY  
TOTALISSD DESC
```

`COALESCE(SUM(G.QTYISSD),0) AS TOTALISSD`: This is calculating the total quantity issued (QTYISSD) from the GOODSISSD table (aliased as G). The COALESCE function ensures that if there are no matching rows, the value will be 0.

2) Copy last spool file into flat file and runqry.

```
Columns . . . : 1 80                                Browse                               GOPZLIB/PTESTSRC  
SEU==> FMT ** . .+.... 1 . .+.... 2 . .+.... 3 . .+.... 4 . .+.... 5 . .+.... 6 . .+.... 7 . .+.... 8  
***** Beginning of data *****  
0001.00 PGM                                         250208  
0002.00     CLRPFM    FILE(STOCKMST)                250208  
0003.00     CLRPFM    FILE(GOODSRCVD)               250208  
0004.00     CLRPFM    FILE(GOODSISSD)               250208  
0005.00     CLRPFM    FILE(WEARTEAR)                250208  
0006.00 ENDPGM                                     250208
```

Columns . . . :	6 80	Browse	GOPZLIB/PTESTSRC INSERTMST
SEU==>			
FMT ** ... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8			***** Beginning of data *****
0000.01 EXEC SQL SET OPTION COMMIT='NONE';		250208	
0001.00 exec sql		250208	
0002.00 INSERT INTO STOCKMST		250208	
0003.00 (ITEMCODE, ITEMNAME, UNITMSR, STOCKINH, REORDLVL, PRICE, VALUE)		250208	
0004.00 VALUES		250208	
0005.00 ('I_10001', 'Rice', 'KG', 50, 100, 45.00, 22500.00),		250210	
0006.00 ('I_10002', 'Wheat Flour', 'KG', 400, 80, 50.00, 20000.00),		250208	
0007.00 ('I_10003', 'Cooking Oil', 'L', 300, 50, 120.00, 36000.00),		250208	
0008.00 ('I_10004', 'Sugar', 'KG', 250, 60, 40.00, 10000.00),		250208	
0009.00 ('I_10005', 'Salt', 'KG', 150, 30, 20.00, 3000.00),		250208	
0010.00 ('I_10006', 'Milk Powder', 'KG', 18, 40, 150.00, 27000.00),		250210	
0011.00 ('I_10007', 'Tea Leaves', 'KG', 100, 25, 200.00, 20000.00),		250208	
0012.00 ('I_10008', 'coffee', 'KG', 80, 20, 300.00, 24000.00),		250208	
0013.00 ('I_10009', 'Soap', 'QTY', 50, 100, 25.00, 12500.00),		250210	
0014.00 ('I_10010', 'Shampoo', 'L', 200, 50, 150.00, 30000.00),		250208	
0015.00 ('I_10011', 'Toothpaste', 'QTY', 60, 90, 50.00, 20000.00),		250210	
0016.00 ('I_10012', 'Biscuits', 'QTY', 50, 80, 30.00, 10500.00),		250210	
0017.00 ('I_10013', 'Cold Drink', 'L', 600, 150, 60.00, 36000.00),		250208	
0018.00 ('I_10014', 'Chocolates', 'QTY', 80, 120, 15.00, 6750.00),		250210	
Columns . . . :	6 80	Browse	GOPZLIB/PTESTSRC INSERTMST
SEU==>			
FMT ** ... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8			***** End of data *****
0019.00 ('I_10015', 'Washing Powder', 'KG', 250, 50, 80.00, 20000.00),		250208	
0020.00 ('I_10016', 'Vegetable Oil', 'L', 350, 70, 130.00, 45500.00),		250208	
0021.00 ('I_10017', 'Butter', 'KG', 150, 30, 500.00, 75000.00),		250208	
0022.00 ('I_10018', 'Cheese', 'KG', 180, 45, 600.00, 108000.00),		250208	
0023.00 ('I_10019', 'Noodles', 'QTY', 100, 200, 25.00, 17500.00),		250210	
0024.00 ('I_10020', 'Pasta', 'QTY', 600, 150, 40.00, 24000.00);		250208	
0025.00		250208	
0026.00 *INLR = '1' ;		250208	
Columns . . . :	6 80	Browse	GOPZLIB/PTESTSRC INSERTRCVD
SEU==>			
FMT * *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8			***** Beginning of data *****
0000.01 *		250209	
0001.00 EXEC SQL SET OPTION COMMIT = *NONE;		250209	
0002.00 exec sql		250209	
0003.00 INSERT INTO GOODSRCVD		250209	
0004.00 (GRN, RITEMCODE, RUSERID, RDATE, RTIME,		250209	
0005.00 REQUESTNO, QTYRCVD, GOODSPRICE)		250209	
0006.00 VALUES		250209	
0047.00 ('Good', 'I_10001', 'U_10001', '2024-02-01',		250209	
0047.01 '08:30:00', 'REQ10001', 100, 4500.00),		250209	
0047.02 ('Bad', 'I_10001', 'U_10002', '2024-02-02',		250209	
0047.03 '09:00:00', 'REQ10002', 80, 3600.00),		250209	
0047.04 ('OK', 'I_10002', 'U_10003', '2024-02-03',		250209	
0047.05 '10:15:00', 'REQ10003', 50, 2000.00),		250209	
0047.06 ('Excelnt', 'I_10002', 'U_10004', '2024-02-04',		250209	
0047.07 '11:00:00', 'REQ10004', 60, 2400.00),		250209	
0047.08 ('Damaged', 'I_10003', 'U_10005', '2024-02-05',		250209	
0047.09 '12:30:00', 'REQ10005', 70, 2800.00),		250209	
0047.10 ('Rejectd', 'I_10003', 'U_10006', '2024-02-06',		250209	
0047.11 '13:45:00', 'REQ10006', 40, 1600.00),		250209	
0047.12 ('Approve', 'I_10004', 'U_10007', '2024-02-07',		250209	
0047.13 '14:30:00', 'REQ10007', 55, 2200.00),		250209	
0047.14 ('Pending', 'I_10004', 'U_10008', '2024-02-08',		250209	
0047.15 '15:00:00', 'REQ10008', 65, 2600.00),		250209	
0047.16 ('Faulty', 'I_10005', 'U_10009', '2024-02-09',		250209	
0047.17 '16:10:00', 'REQ10009', 90, 3600.00),		250209	
0047.18 ('Perfect', 'I_10005', 'U_10010', '2024-02-10',		250209	
0047.19 '17:20:00', 'REQ10010', 80, 3200.00),		250209	
0047.20 ('Good', 'I_10006', 'U_10001', '2024-02-11',		250209	
0047.21 '08:50:00', 'REQ10011', 100, 5000.00),		250209	
0047.22 ('Bad', 'I_10006', 'U_10002', '2024-02-12',		250209	
0047.23 '09:40:00', 'REQ10012', 95, 4750.00),		250209	
0047.24 ('OK', 'I_10007', 'U_10003', '2024-02-13',		250209	
0047.25 '10:30:00', 'REQ10013', 60, 2400.00),		250209	
0047.26 ('Excelnt', 'I_10007', 'U_10004', '2024-02-14',		250209	
0047.27 '11:20:00', 'REQ10014', 70, 2800.00),		250209	
0047.28 ('Damaged', 'I_10008', 'U_10005', '2024-02-15',		250209	
0047.29 '12:10:00', 'REQ10015', 85, 3400.00),		250209	
0047.30 ('Rejectd', 'I_10008', 'U_10006', '2024-02-16',		250209	
0047.31 '13:00:00', 'REQ10016', 90, 3600.00),		250209	

```

0047.32      ('Approve', 'I_10009', 'U_10007', '2024-02-17',          250209
0047.33      '14:50:00', 'REQ10017', 110, 4400.00),                  250209
0047.34      ('Pending', 'I_10009', 'U_10008', '2024-02-18',          250209
0047.35      '15:30:00', 'REQ10018', 120, 4800.00),                  250209
0047.36      ('Faulty', 'I_10010', 'U_10009', '2024-02-19',          250209
0047.37      '16:40:00', 'REQ10019', 130, 5200.00),                  250209
0047.38      ('Perfect', 'I_10010', 'U_10010', '2024-02-20',          250209
0047.39      '17:50:00', 'REQ10020', 140, 5600.00);                  250209
0048.00      *INLR = '1' ;
***** End of data *****

```

Columns . . . :	6 80	Browse	GOPZLIB/PTESTSRC INSERTISSD
SEU==>			
FMT **	... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8		
	***** Beginning of data *****		
0001.00	EXEC SQL SET OPTION COMMIT = *NONE ;	250208	
0002.00		250208	
0003.00	exec sql	250208	
0004.00	INSERT INTO GOODSISSD	250208	
0005.00	(ISN, IITEMCODE, IUSERID, IDATE, ITIME, QTYISSD)	250208	
0006.00	VALUES	250208	
0007.00	('S_10001', 'I_10001', 'U_10001', '2024-02-01', '09:30:00', 20),	250208	
0008.00	('S_10002', 'I_10002', 'U_10002', '2024-02-02', '10:45:00', 15),	250208	
0009.00	('S_10003', 'I_10002', 'U_10003', '2024-02-03', '11:15:00', 30),	250208	
0010.00	('S_10004', 'I_10002', 'U_10004', '2024-02-04', '12:30:00', 25),	250208	
0011.00	('S_10005', 'I_10003', 'U_10005', '2024-02-05', '13:50:00', 20),	250208	
0012.00	('S_10006', 'I_10003', 'U_10006', '2024-02-06', '15:10:00', 15),	250208	
0013.00	('S_10007', 'I_10004', 'U_10007', '2024-02-07', '16:25:00', 10),	250208	
0014.00	('S_10008', 'I_10004', 'U_10008', '2024-02-08', '17:45:00', 5),	250208	
0015.00	('S_10009', 'I_10005', 'U_10009', '2024-02-09', '18:30:00', 30),	250208	
0016.00	('S_10010', 'I_10005', 'U_10010', '2024-02-10', '19:15:00', 25),	250208	
0017.00	('S_10011', 'I_10006', 'U_10001', '2024-02-11', '08:50:00', 40),	250208	
0018.00	('S_10012', 'I_10006', 'U_10002', '2024-02-12', '09:35:00', 35),	250208	
0019.00	('S_10013', 'I_10007', 'U_10003', '2024-02-13', '10:55:00', 30),	250208	
0020.00	('S_10014', 'I_10007', 'U_10004', '2024-02-14', '11:25:00', 25),	250208	
0021.00	('S_10015', 'I_10008', 'U_10005', '2024-02-15', '12:50:00', 20),	250208	
0022.00	('S_10016', 'I_10008', 'U_10006', '2024-02-16', '13:55:00', 30),	250208	
0023.00	('S_10017', 'I_10009', 'U_10007', '2024-02-17', '14:35:00', 40),	250208	
0024.00	('S_10018', 'I_10009', 'U_10008', '2024-02-18', '15:45:00', 35),	250208	
0025.00	('S_10019', 'I_10010', 'U_10009', '2024-02-19', '16:58:00', 50),	250208	
0026.00	('S_10020', 'I_10010', 'U_10010', '2024-02-20', '18:20:00', 45);	250208	
0027.00		250208	
0028.00	*inlr = *on;	250208	
	***** End of data *****		

Columns . . . :	6 80	Browse	GOPZLIB/PTESTSRC INSERTWTI
SEU==>			
FMT **	... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8		
	***** Beginning of data *****		
0001.00	EXEC SQL	250208	
0002.00	SET OPTION COMMIT = *NONE , DATFMT = *ISO, TIMEFMT = *ISO ;	250208	
0003.00	exec sql	250208	
0004.00	INSERT INTO WEARTEAR	250208	
0005.00	(WTID, WITEMCODE, WUSERID, REASON, WDATE, WTIME, QTYLOSS)	250208	
0006.00	VALUES	250208	
0007.00	('W_10001', 'I_10001', 'U_001', 'Damaged during transport',	250208	
0008.00	'2024-02-01', '09:15:00', 10),	250208	
0009.00	('W_10002', 'I_10002', 'U_002', 'Expired shelf life',	250208	
0010.00	'2024-02-01', '10:45:00', 5),	250208	
0011.00	('W_10003', 'I_10003', 'U_003', 'Theft reported',	250208	
0012.00	'2024-02-02', '11:30:00', 12),	250208	
0013.00	('W_10004', 'I_10004', 'U_004', 'Faulty packaging',	250208	
0014.00	'2024-02-02', '12:20:00', 8),	250208	
0015.00	('W_10005', 'I_10005', 'U_005', 'Damaged during transport',	250208	
0016.00	'2024-02-03', '14:10:00', 15),	250208	
0017.00	('W_10006', 'I_10006', 'U_006', 'Expired shelf life',	250208	
0018.00	'2024-02-03', '15:05:00', 7),	250208	
0019.00	('W_10007', 'I_10007', 'U_007', 'Theft reported',	250208	

Columns . . . :	1 80	Browse	GOPZLIB/PTESTSRC INZDTAARA
SEU==>			
FMT **	...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8		
	***** Beginning of data *****		
0001.00	PGM	250209	
0002.00	CHGDTAARA DTAARA(ITEMDTA *ALL) VALUE('10021')	250209	
0002.01	CHGDTAARA DTAARA(ISNDTA *ALL) VALUE('10021')	250209	
0002.02	CHGDTAARA DTAARA(RNDTAARA *ALL) VALUE('1')	250209	
0002.03	CHGDTAARA DTAARA(WTDTAARA *ALL) VALUE('10011')	250209	
0003.00	ENDPGM	250209	
	***** End of data *****		

Columns . . . :	1 80	Browse	GOPZLIB/PCLSRC USER1RPTCL
SEU==>			
FMT **	...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8		
	***** Beginning of data *****		
0001.00	PGM	250209	
0001.01	DCLF FILE(USERFILE)	250209	
0002.00	OVRDBF FILE(USERFILE) SHARE(*YES)	250209	
0003.00	OPNQRYF FILE((USERFILE)) OPTION(*ALL) QRYSLT('DESIG +	250209	
0004.00	*EQ "ADMIN") KEYFLD(*FILE)	250209	
0004.01	CALL PGM(USERDETRPT)	250209	
0006.00	CLOF OPNID(USERFILE)	250209	
0007.00	DLTOVR FILE(USERFILE)	250209	
0008.00	ENDPGM	250209	
	***** End of data *****		

Columns . . . :	6 80	Browse	GOPZLIB/PRPGLESRC USERDETRPT
SEU==>			
FMT F	FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords+++++		
	***** Beginning of data *****		
0001.00	FUSERRET O E PRINTER OFLIND(*IN99)	250209	
0001.01	FUSERFILE IF E K DISK	250209	
0012.00	*	250206	
0012.02	*	250209	
0014.00	WRITE HEADER;	250209	
0015.00	WRITE HEADING;	250209	
0016.00		250209	
0042.01	SETLL *LOVAL USERFILE;	250209	
0042.02	READ USERFILE;	250209	
0042.03		250209	
0042.04	IF DESIG = 'ADMIN';	250209	
0042.05	WRITE ADMIN;	250209	
0042.06	ELSEIF DESIG = 'MANAGER';	250209	
0042.07	WRITE MANAGER;	250209	
0042.08	ELSEIF DESIG = 'OPERATOR';	250209	
0042.09	WRITE OPERATO;	250209	
0042.10	ENDIF;	250209	
0042.11		250209	
0042.12	DOW NOT %EOF();	250209	
0042.13	WRITE DETAILS;	250209	
0042.14	READ USERFILE;	250209	
0055.00	ENDDO;	250209	
0057.00		250209	
0059.93	WRITE FOOTER;	250209	
0060.00		250209	
0061.00	*INLR = '1';	250209	
	***** End of data *****		

Columns . . . :	1 80	Browse	GOPZLIB/PCLSRC CL1RPT
SEU==>			
FMT **	...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8		
	***** Beginning of data *****		
0001.00	CALL USER1RPTCL	250209	
0002.00	CPYSPLF FILE(USERRPT) TOFILE(USERPTFLAT) SPLNBR(*LAST)	250209	
0003.00	RUNQRY *N USERPTFLAT	250209	
	***** End of data *****		

Parameters or command

====> LAUNCHINV

F3=Exit

F9=Retrieve

F4=Prompt

F10=Command entry

2/10/25

LOGIN SCREEN

07:53:46

ENTER USERNAME : █

ENTER PASSWORD :

F1=HELP

F3=EXIT

2/10/25

LOGIN SCREEN

07:54:17

ENTER USERNAME : █

ENTER PASSWORD :

WRONG CREDENTIALS

F1=HELP

F3=EXIT

EMPMENU

MAIN MENU

Select one of the following:

1. ENTER STOCK RECORDS
2. VIEW STOCKS
3. GENERATE REPORTS

PRESS F3 TO SIGN OFF

Selection or command

====>

F3=Exit F4=Prompt F9=Retrieve F12=Cancel
F13=Information Assistant F16=System main menu

ENTRMENU

Entry Menu

Select one of the following:

1. ENTER NEW ITEMS
2. ENTER GOODS RECIEVED
3. ENTER GOODS ISSUED
4. ENTER LOST OR DAMAGED ITEMS

Selection or command

====>

F3=Exit F4=Prompt F9=Retrieve F12=Cancel
F13=Information Assistant F16=System main menu

2/10/25

STOCK MASTER ENTRY

RAM

ITEM NAME :
UNIT OF MEASURE :
STOCK IN HAND :
REORDER LEVEL :
PRICE :

F1=HELP

F12=CANCEL

2/10/25

STOCK MASTER ENTRY

RAM

ITEM NAME : MATTA RICE
UNIT OF MEASURE: KGS
STOCK IN HAND : 10000
REORDER LEVEL : 100
PRICE : 114.00

F1=HELP

F12=CANCEL

2/10/25

STOCK MASTER ENTRY

RAM

ITEM NAME :
UNIT OF MEASURE:
STOCK IN HAND :
REORDER LEVEL :
PRICE :

INCORRECT UNIT OF MEASURE

F1=HELP

F12=CANCEL

2/10/25

STOCK MASTER ENTRY

RAM

ITEM NAME : MATTA RICE
UNIT OF MEASURE: KG
STOCK IN HAND : 1000
REORDER LEVEL : 100
PRICE : 114.00

2/10/25

STOCK MASTER ENTRY

RAM

ITEM NAME :
UNIT OF MEASURE :
STOCK IN HAND :
REORDER LEVEL :
PRICE :

RECORD ADDED SUCCESSFULLY

F1=HELP

F12=CANCEL

2/10/25

GOODS RECEIVED ENTRY SCREEN

RAM

ENTER THE ITEM CODE :
ENTER THE REQUEST NUMBER :
ENTER THE QUANTITY RECEIVED:
ENTER THE GOODS RECEVED NOTE:

F1=HELP

F12=CANCEL

2/10/25

GOODS RECEIVED ENTRY SCREEN

RAM

ENTER THE ITEM CODE : I 10001
ENTER THE REQUEST NUMBER : REQ10001
ENTER THE QUANTITY RECEIVED: 100
ENTER THE GOODS RECEVED NOTE: GOOD

F1=HELP

F12=CANCEL

2/10/25

GOODS RECEIVED ENTRY SCREEN

RAM

ENTER THE ITEM CODE :

ENTER THE REQUEST NUMBER :

ENTER THE QUANTITY RECEIVED:

ENTER THE GOODS RECEVED NOTE:

RECORD ALREADY EXISTS

F1=HELP

F12=CANCEL

2/10/25

GOODS RECEIVED ENTRY SCREEN

RAM

ENTER THE ITEM CODE : I_00000004

ENTER THE REQUEST NUMBER : REQ_01

ENTER THE QUANTITY RECEIVED: 10

ENTER THE GOODS RECEVED NOTE: GOOD

RECORD ALREADY EXISTS

F1=HELP

F12=CANCEL

2/10/25

GOODS RECEIVED ENTRY SCREEN

RAM

ENTER THE ITEM CODE :

ENTER THE REQUEST NUMBER :

ENTER THE QUANTITY RECEIVED:

ENTER THE GOODS RECEVED NOTE:

ITEM DOES NOT EXIST

F1=HELP

F12=CANCEL

2/10/25

GOODS RECEIVED ENTRY SCREEN

RAM

ENTER THE ITEM CODE : I_10001
ENTER THE REQUEST NUMBER : REQ10021
ENTER THE QUANTITY RECEIVED: 10
ENTER THE GOODS RECEVED NOTE: GOOD█

ITEM DOES NOT EXIST

F1=HELP F12=CANCEL

Online M 14,52

2/10/25

GOODS RECEIVED ENTRY SCREEN

RAM

ENTER THE ITEM CODE : █
ENTER THE REQUEST NUMBER :
ENTER THE QUANTITY RECEIVED:
ENTER THE GOODS RECEVED NOTE:

RECORD ADDED SUCCESSFULLY

F1=HELP F12=CANCEL

2/10/25

GOODS ISSUE ENTRY SCREEN

RAM

ENTER THE ITEM CODE : I_10001
ENTER THE QUANTITY ISSUED : 60█

F1=HELP F12=CANCEL

2/10/25

GOODS ISSUE ENTRY SCREEN

RAM

ENTER THE ITEM CODE : [REDACTED]

ENTER THE QUANTITY ISSUED : [REDACTED]

STOCK NOT AVAILABLE

F1=HELP

F12=CANCEL

2/10/25

GOODS ISSUE ENTRY SCREEN

RAM

ENTER THE ITEM CODE : [REDACTED]

ENTER THE QUANTITY ISSUED : [REDACTED]

BLANK FIELDS DETECTED

F1=HELP

F12=CANCEL

2/10/25

GOODS ISSUE ENTRY SCREEN

RAM

ENTER THE ITEM CODE : [REDACTED]

ENTER THE QUANTITY ISSUED : [REDACTED]

ITEM NOT FOUND

F1=HELP

F12=CANCEL

2 / 10 / 25

GOODS ISSUE ENTRY SCREEN

RAM

ENTER THE ITEM CODE :

ENTER THE QUANTITY ISSUED :

RECORD ADDED

F1=HELP

F12=CANCEL