**DB2 Notes**

**sites**

**Tutorials**

<http://www.mainframegurukul.com/tutorials/database/db2_tutorials/sample-db2-cobol-compile-jcl.html>

<http://www.mainframegurukul.com/tutorials/database/db2_tutorials/db2.html>

<http://www.sql-tutorial.net/SQL-JOIN.asp>

http://mainframealldtime.wordpress.com/category/db2/

**DB2 TOOLS**

<http://www.dbforums.com/db2/1622056-how-dclgen-can-used-create-host-variables-cobol-db2-program.html>

<https://sites.google.com/site/cobolmaterial/dclgen-tutorial>

<http://mainframestutor.in/dclgen-in-db2/>

**Buffers**

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2z10.doc.intro%2Fsrc%2Ftpc%2Fdb2z_bufferpoolsanddatacaching.htm>

**Example Programs**

<http://www.mainframegurukul.com/tutorials/database/db2_tutorials/cobol-db2-sample-program.html>

-----------------------------------------------------------------------------------------------------------------------------

**Imp Points**

* **Common DB2 abends are -206, +100, -551, -803, -805, -811, -818**
* <http://www.youtube.com/watch?v=lNkI710RxMc>

DB2 windows details db2admin/sandynami

* **Good Book - DBMS by CJ Date**
* **DB2 Certifications**

Different types of data applications, **data warehousing**, and **OLAP**.

* **NAB – National Australian Bank**

**The project uses a combination of IMSDB and DB2 for the business requirements.**

**The DB2 part is mainly used for rules and after checking the rules the required data is fetched from IMSDB.**

**IMSDB contains segments**

**Table structure**

**Srve\_mgmt\_parm\_unit**

**MPID PMPID EFFDATE STATUS PERC INDICATOR**

**CAMS – Customer savings**

**CASB - Business Accounts**

-----------------------------------------------------------------------------------------------------------------------------

**Questions and their Explanation**

**Q. Explain SQL?**

**ANS.**

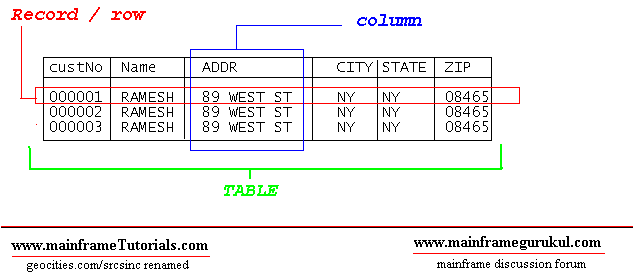
SQL (structured query language) is used to acccess, manipulate or control the

access of rantional database. Databases like **DB2, ORACLE, SQL SERVER 2000**

support the SQL (Structured query language).

In relational database data represented as tables. Tables contains the data

in rows and column. Rows are called as records, columns called as fields.



**PRIMARY KEY** : PRIMARY KEY is one column or combination columns. This primary key

is to identify the records uniquely in the table. Applications/programs normally

proides PRIMARY KEY value to get the record values from the table. PRIMARY KEY

columns have the unique index (unqiue indexes wont allow duplicate values) defined

on them, so that processing time of SQL will be very less compare than the non-index columns.

**FOREIGN KEY:**  FOREIGN KEY is one or more columns in a table that refers to the

primary key of another table. Through this key we can establish a relation between

two tables. If table B foreign key refers table A's primary key, DB2 / ORACLE wont

allow to enter the records into table B if the foreign key value in the input data

is not present in the table A's primary key list of values.

DB2 SQL (Structured query language) commands can be classified into 3 main categories

**DDL, DML and DCL.**

**DDL ( Data Definition Language ):** These statements usaully used by DBA's or Senior Programmers to create table spaces, tables , indexes etc...

**COMMANDS :**

CREATE

DECLARE

DROP

ALTER

**DML (Data Manipulatoin Language )**: using these statements we can select, insert, update

and delete the data in the database.

**COMMANDS :**

SELECT,

INSERT,

UPDATE,

DELETE

**DCL (Data Control Language )**: Using these statements DBA / user who has authority to

grant can grant accecss or revoke the authority of accessing DB2 database.

**COMMANDS :**

GRANT

REVOKE

**Q. Explain DDL Statements in DB2?**

**ANS.**

Following are the **DDL ( Data Definition Language )** statements

**CREATE**

**DECLARE**

**ALTER**

**DROP**

**CREATE**

CREATE statement is used to create following DB2 database objects. The system catalog is updated whenever you Create a DB2 database object.

1. Buffer pools
2. Event monitors
3. Functions
4. Indexes
5. Schemas
6. Stored procedures
7. Tables
8. Table spaces
9. Triggers
10. Views

**EG-**

CREATE TABLE employee(

Emp\_id smallint not null,

Emp\_first\_name varchar(30),

Emp\_m\_name varchar(1),

EMP-LAST\_NAME VARCHAR(30),

SALARY DECIMAL(100,2),

DEP smallint )

**DECLARE**

The DECLARE statement is similar to the CREATE statement, except that it is used to

create temporary tables that exist only for the duration of a database connection.

Temporary tables are useful when you are working with intermediate results. Declared

tables can be referenced like any other table, and they can be altered or dropped like

any other table. A table is the only object that can be declared. The system catalog

is not updated when you declare a temporary table. You can declare a temporary table

by using the **DECLARE GLOBAL TEMPORARY TABLE** statement.

**DECLARE GLOBAL TEMPORARY TABLE session.emp1**

**LIKE employee**

**ON COMMIT PRESERVE ROWS**

**NOT LOGGED**

**IN tempspace**

In this example, the DECLARE GLOBAL TEMPORARY TABLE statement is used to

declare a temporary table named emp1, located in an existing user temporary table

space named TEMPSPACE. The columns in this table will have the same names and

definitions as the columns in the EMPLOYEE table. The rows of the temporary table

will be preserved (not deleted) whenever a COMMIT statement is processed.

Finally, changes to the temporary table are not logged (this is the only option).

session is a schema name.

**ALTER**

The ALTER statement is used to change some of characterstics of following DB2 objects.

1. Buffer pools
2. Tables
3. Table spaces
4. Views

Example - In the following example, we can alter the table defined.

ALTER TABLE EMPLOYEE

ADD DATE\_OF\_JOIN DATE

**Note:** you cannot alter an index. You must drop it and then create a new one with a different definition

**DROP**

In DB2, You can drop any db2 objects created using CREATE or DECLARE statement

You can drop any of the following db2 objects

1. Buffer pools
2. Event monitors
3. Functions
4. Indexes
5. Schemas
6. Stored procedures
7. Tables
8. Table spaces
9. Triggers
10. Views

The DROP statement will delete the object definition of catalog. Following is an

**Ex:**

**DROP TABLE EMPLOYEE**

**Q. Explain DML Statements in DB2?**

**ANS.**

You can use following statements to manipulate data on db2 database.

1. SELECT
2. INSERT
3. UPDATE
4. DELETE

**SELECT**

SELECT statement is used to get all data or required data from the table.

**Simple Syntax**

**SELECT < Column Names > / \* / function(< Column Name >) FROM < TABLE NAME >**

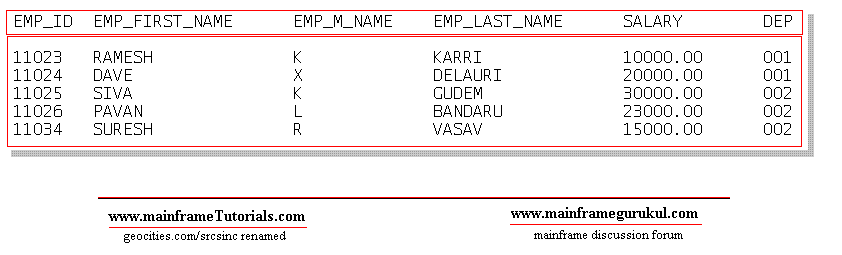
**[ WHERE < Condition > .... ]**

Example (s)

1. To get all data from EMPLOYEE table , issue

SELECT \* FROM EMPLOYEE

Following is a the result set returned by the query



**INSERT**

INSERT statement is used to insert record(s) into table / view. Inserting a row into

view also inserts the row into table.

**Simple Syntax**

**INSERT INTO < TABLE NAME / VIEW NAME > [ (<COLUMN NAME1>,<COLUMN NAME2>...) ]**

**VALUES ( value1, value2,.... )**

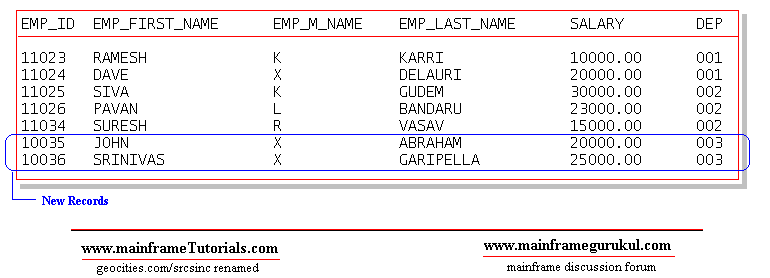
**Example 1:**

**INSERT INTO EMPLOYEE ( EMP\_ID, EMP\_FIRST\_NAME,EMP\_M\_NAME,EMP\_LAST\_NAME, SALARY, DEP)**

**VALUES ( 10035, 'JOHN', 'X', 'ABRAHAM', 20000.00, 003 ) ,**

**( 10036, 'SRINIVAS', 'X', 'GARIPELLA', 25000.00, 003 )**

Above insert statemet, insert two records into EMPLOYEE Table, Now table contains following records.



**Example2.**

Specify a fullselect to identify data that is to be copied from other tables or views. A

fullselect is a statement that generates a result table. For example:

**CREATE TABLE emp LIKE EMPLOYEE**

**INSERT INTO emp**

**SELECT EMP\_ID,EMP\_FIRST\_NAME,EMP\_M\_NAME,EMP\_LAST\_NAME,SALARY, DEP**

**FROM EMPLOYEE**

**WHERE DEP = 002**

**Q.DB2 compile program**

ANS.

//DB2COMP (XXX,XXX),'COMPILE JCL',

// CLASS=A,MSGCLASS=A,NOTIFY=&SYSUID

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* COMPILATION, LINK EDIT AND THE BIND STEP FOR A COBOL DB2 PROGRAM \*

//\* WILL BE DONE BY SUBMITTING THIS JOB. \*

//\* THE DB2 REGIONS AND CORRESPONDING PARAMETERS NEEDS TO BE CHANGED \*

//\* WITH RESPECT TO THE PROGRAM \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* PRECOMPILE DB2 PROGRAM \*

//\*-------------- LOCATION OF DBRM LIBRARY -------------------------\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//PC EXEC PGM=**DSNHPC**,

// PARM='HOST(COB2),APOST,SOURCE',

// REGION=4096K

//**DBRMLIB** DD DISP=SHR,

// DSN=**DEV.SURESH.DBRM(DB2PROG)**<------------------------ (1)

//STEPLIB DD DISP=SHR,

// DSN=SYSX.DB2.XXX.XXXXX

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*SYSIN -----------INPUT COBOL DB2 PROGRAM LOCATION-----------------\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//**SYSIN**DD DISP=SHR,

// DSN=**DEV.SURESH.SRC(DB2PROG)**<---------------------- (2)

//SYSCIN DD DISP=(MOD,PASS),

// DSN=&&TEMP,

// SPACE=(800,(500,500)),

// UNIT=SYSDA

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* DCLGEN MEMBER LOCATION \*

//\*SYSLIB-----------------INPUT SOURCE LIBRARY FOR SQL---------------\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//**SYSLIB**DD DISP=SHR,

// DSN=**DEV.SURESH.DCL**<---------------------- (3)

// DD DISP=SHR,

// DSN=**DEV.SURESH.CPY**

//SYSPRINT DD SYSOUT=T

//SYSTERM DD SYSOUT=T

//SYSUDUMP DD SYSOUT=\*

//SYSUT1 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//SYSUT2 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* COMPILATION \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//COB EXEC PGM=**IGYCRCTL**,

// COND=(4,LT,PC),

// PARM=('SIZE(4000K),BUFSIZE(32760),LIST,LIB,MAP,OBJECT',

// 'DATA(31),XREF,RENT'),

// REGION=4M

//STEPLIB DD DISP=SHR,

// DSN=XXXX.XXXXXX

//SYSIN DD DISP=(OLD,DELETE),

// DSN=&&TEMP

//SYSLIN DD DISP=(MOD,PASS),

// DSN=&&LOADTMP,

// SPACE=(800,(500,500)),

// UNIT=SYSDA

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*--------------SOURCE LIBRARIES FOR COBOL DB2 CODE (COPY LIBRARIES)\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//**SYSLIB**DD DISP=SHR,

// DSN=**DEV.SURESH.DCL**<----------------- (4)

// DD DSN=**DEV.SURESH.CPY**,DISP=SHR

//SYSPRINT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYSUT1 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//SYSUT2 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//SYSUT3 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//SYSUT4 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//SYSUT5 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//SYSUT6 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//SYSUT7 DD SPACE=(800,(500,500),,,ROUND),

// UNIT=SYSDA

//\*

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* LINK EDIT \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//LKED EXEC PGM=**IEWL**,

// COND=((4,LT,COB),(4,LT,PC)),

// PARM='XREF'

//SYSLIB DD DISP=SHR,

// DSN=SXXX.SXXXXXXX

// DD DISP=SHR,

// DSN=XXXX.DB2.XXX.XXXXLOAD

// DD DISP=SHR,

// DSN=SYS1.VSCLLIB

//SYSLIN DD DISP=(OLD,DELETE),

// DSN=&&LOADTMP

//\* DD DDNAME=SYSIN

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*----------------LOCATION OF LOAD LIBRARY--------------------------\*

//SYSLMOD DD DISP=SHR,

// DSN=**DEV.SURESH.LOADLIB(DB2PROG)**<------------ (5)

//SYSPRINT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYSUT1 DD SPACE=(1024,(50,50)),

// UNIT=SYSDA

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* BIND - BIND THE DB2 PACKAGE \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//BIND EXEC PGM=**IKJEFT01**,

// COND=(4,LT),

// REGION=4096K

//STEPLIB DD DISP=SHR,

// DSN=XXX4.DB2.XXXX.XXXXLOAD

//**DBRMLIB** DD DISP=SHR,

// DSN=**DEV.SURESH.DBRM(DB2PROG)**<--------------- (6)

//SYSPRINT DD SYSOUT=\*

//SYSTSPRT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYSTSIN DD \*

**DSN SYSTEM (DEVDB )**

**BIND MEMBER (DB2PROG) -**

**PACKAGE (PACKG11) -**

**LIBRARY ('DEV.SURESH.DBRM') -** <---------------- (7)

**ACTION (REP) -**

**ISOLATION (CS) -**

**VALIDATE (BIND)-**

**RELEASE (COMMIT) -**

**OWNER (SURESH) -**

**QUALIFIER (DEVQUALI)**

END

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bottom of Data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(1) - When we precompiled, precompiler will create the DBRM, it will

be placed in the pds specified here.

(2) - Location of COBOL-DB2 program

(3) - Needs to speficiy DCLGEN member locations

(4) - Needs to specify DCLGEN and COPYBOOK locations here

(5) - Load module location, load module will be created here. this

location needs to be given in run jcl.

(6) & (7) - specify the location of DBRM, ( same location used in step1 ).

**Q. Extract data from DB2 table directly using IKJEFT01**

**ANS.**

//STEP002 EXEC PGM=**IKJEFT01**,

// **DYNAMNBR**=20,COND=(0,NE)

//**STEPLIB** DD DISP=SHR,

// DSN=SYS.DB2.XXX.RUNXX.LOAD

// DD DISP=SHR,

// DSN=SYS.DB2.XXX.SDSNEXIT

// DD DISP=SHR,

// DSN=SYS.DB2.XXX.SDSNLOAD

//**SYSIN**DD \*

**SELECT \***

**FROM EMP\_TABLE**

**WITH UR;**

//\*

//**SYSTSIN**DD \*

DSN SYSTEM(XXX) RETRY(20)

RUN PROGRAM(**DSNTIAUL**) PLAN(**DSNTIAUL**) PARMS(**'SQL'**) -

LIB ('SYS.DB2.XXX.RUNXX.LOAD')

END

/\*

//\*\*\*UNLOAD FILE

//**SYSREC00**DD DISP=(NEW,CATLG,DELETE),

// DSN=DEV.UNLOAD.FILE,

// UNIT=SYSDA,SPACE=(CYL,(50,50),RLSE)

//\*

//\*\*\*TABLE STRUCTURE

//SYSPUNCH DD DUMMY

//\*

//SYSTSPRT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT

**Q.Binding DB2 with IKJEFT01**

**ANS.**

//\* DB2 BIND JCL

//\*

//**BIND** EXEC PGM=IKJEFT01,

// COND=(4,LT),

// REGION=4096K

//**STEPLIB** DD DISP=SHR,

// DSN=XXX4.DB2.XXXX.XXXXLOAD

//**DBRMLIB** DD DISP=SHR,

// DSN=DEV.SURESH.DBRM(DB2PROG) <--------------- (1)

//SYSPRINT DD SYSOUT=\*

//SYSTSPRT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//**SYSTSIN** DD \*

**DSN SYSTEM (DEVDB)**

BIND MEMBER (DB2PROG) -

PACKAGE (PACKG11) -

LIBRARY ('DEV.SURESH.DBRM') - <---------------- (2)

ACTION (REP) -

ISOLATION (CS) -

VALIDATE (BIND)-

RELEASE (COMMIT) -

OWNER (SURESH) -

QUALIFIER (DEVQUALI)

END

/\*

**Q. Running Cobol DB2 program using IKJEFT01**

**ANS.**

//\*\* COBOL DB2 RUN JCL

//STEP01 EXEC PGM=**IKJEFT01**,

// DYNAMNBR=20

//\*

//**STEPLIB** DD DISP=SHR,

// DSN=SYSL.DB2.DEV.XXXXLOAD

// DD DISP=SHR,

// DSN=SYSL.DB2.DEV.XXXXYYYY

//\*

//**SYSTSPRT**DD SYSOUT=\*

//SYSTSIN DD \*

DSN SYSTEM(DEVDB)

RUN PROGRAM(DB2PROG) -

PLAN(PLAN11111) -

LIBRARY('DEV.SURESH.LOADLIB')

END

/\*

//SYSOUT DD SYSOUT=\*

//SYSIN DD \*

//\*

//\* COBOL DB2 RUN JCL ENDS

**Q. Writing a COBOL DB2 Program.**

**ANS.**Let us assume we are writing a cobol program to read EMPLOYEE

table and get the details of employee with the name XXXXXXX.

Let us go in step wise..

create the following table in db2 or assume it is there in db2 database.

**EMPLYEE**

**EMPID EMPNAME DEPARTMENT SALARY DESIGNATION**

1000 XXXXXXX XX 10000 SE

1001 YYYYYYY YY 9000 SE

1002 ZZZZZZZ ZZ 20000 MA

**STEP 1.** We need to declare the table structure in the

**WORKING-STORAGE SECTION** or **LINKAGE SECTION**.

**EXEC SQL**

DECLARE DSNXXX.EMPLOYEE

( EMPID CHAR(10) NOT NULL,

EMPNAME CHAR(30) NOT NULL,

DEPARTMENT CHAR(2) NOT NULL,

SALARY DECIMAL(10,2) NOT NULL,

DESIGNATION CHAR(4) NOT NULL )

**END-EXEC.**

we can use **DB2 tool called DCLGEN** to generate this declaration

for us and can include that copy book here.

if you create a copybook using DCLGEN. Use following sntax to include

**EXEC SQL**

**INCLUDE**<copybookname>

**END-EXEC.**

**STEP 2.** Declare host variables in WORKING-STORAGE SECTION.

HOST VARIABLES - A host variable is a data item declared in cobol to use

it in embedded SQL.

For EMPLOYEE table, host variable declaration is look like as follows...

01 EMPOYEE-RECORD.

05 HV-EMPID PIC X(10).

05 HV-EMPNAME PIC X(30).

05 HV-DEPARTMENT PIC X(2).

05 HV-SALARY PIC S9(8)V99 COMP-3.

05 HV-DESIGNATION PIC CHAR(4).

If you use db2 tool DCLGEN, it will automatically creates this structure also

along with table declaration specified in step1.

**STEP 3.** Include SQLCA as follows in WORKING-STORAGE SECTION.

EXEC SQL

INCLUDE SQLCA

END-EXEC.

What is SQLCA?

SQLCA - SQL communication area.

When a SQL statement executes, DB2 places a value in SQLCODE AND

SQLSTATE host variables or any other fields of SQLCA. based on

the values in these variables we can know whether sql ran

sucessfully or not.

SQLCA contains a declartion of fields like SQLCODE,SQLSTATE and

SQLERRD etc....

**STEP 4.** Add a sql statement in procdure division to get the details of employee

with the name XXXXXXX.

DISPLAY ' PROGRAM STARTED .... '

.........

EXEC SQL

SELECT SALARY

**INTO :HV-SALARY**

FROM EMPLOYEE

WHERE EMPNAME = 'XXXXXXX'

END-EXEC.

IF SQLCODE = 0

DISPLAY ' SQL EXECUTED SUCESSFULLY '

DISPLAY ' EMPLOYEE SALARY IS ' HV-SALARY

ELSE

DISPLAY ' SQL FAILED '

DIAPLY ' SQL CODE ' SQLCODE

END-IF.

....

....

DISPLAY ' PROGRAM ENDED'.

Here SQLCODE = 0 means, sql ran sucessfully without any issues. Hence

we are displaying the HV-SALARY into the spool.

If SQLCODE NOT = 0 , there is issue in executing the sql statement.

Now we have compeleted coding a cobol-db2 program. our next step is to

compile the program.

**Q. Using RunJCl for DB2**

**ANS.**

**Q.Joins**

**ANS.** http://www.sql-tutorial.net/SQL-JOIN.asp

Customers:

CustomerIDFirstNameLastName Email DOB Phone

1 John Smith John.Smith@yahoo.com 2/4/1968 626 222-2222

2 Steven Goldfish goldfish@fishhere.net 4/4/1974 323 455-4545

3 Paula Brown pb@herowndomain.org 5/24/1978 416 323-3232

4 James Smith jim@supergig.co.uk 20/10/1980 416 323-8888

Sales:

**CustomerIDDateSaleAmount**

2 5/6/2004 $100.22

1 5/7/2004 $99.95

3 5/7/2004 $122.95

3 5/13/2004 $100.00

4 5/22/2004 $555.55

There are 2 types of SQL JOINS – INNER JOINS and OUTER JOINS. If you don't put INNER or OUTER keywords in front of the SQL JOIN keyword, then INNER JOIN is used. In short "INNER JOIN" = "JOIN" (note that different databases have different syntax for their JOIN clauses).

If the Sales table has the following rows:

Sales:

CustomerID Date SaleAmount

2 5/6/2004 $100.22

1 5/6/2004 $99.95

The INNER JOIN will select all rows from both tables as long as there is a match between the columns we are matching on. In case we have a customer in the Customers table, which still hasn't made any orders (there are no entries for this customer in the Sales table), this customer will not be listed in the result of our SQL query above.

Even though Paula and James are listed as customers in the Customers table they won't be displayed because they haven't purchased anything yet.

But what if you want to display all the customers and their sales, no matter if they have ordered something or not? We’ll do that with the help of SQL OUTER JOIN clause.

The second type of SQL JOIN is called SQL OUTER JOIN and it has 2 sub-types called LEFT OUTER JOIN and RIGHT OUTER JOIN.

As you can see we have selected everything from the Customers (first table). For all rows from Customers, which don’t have a match in the Sales (second table), the SalesPerCustomer column has amount NULL (NULL means a column contains nothing).

The RIGHT OUTER JOIN or just RIGHT JOIN behaves exactly as SQL LEFT JOIN, except that it returns all rows from the second table (the right table in our SQL JOIN statement).

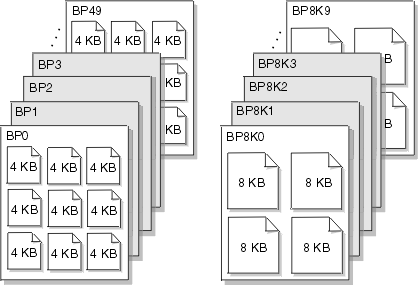
**Q. Explain DB2 buffers?**

ANS.Buffer pools are a key element of DB2® performance, and help you to avoid delays when retrieving data.

DB2 can retrieve a page from a buffer pool faster than it can from disk. When data is already in a buffer, an application program avoids the delay of waiting for DB2 to retrieve the data from disk.

DB2 lets you use up to 50 buffer pools that contain 4 KB pages and up to 10 buffer pools each that contain 8 KB, 16 KB, and 32 KB pages.

Figure 1. Buffer pools with 4 KB and 8 KB pages



At any time, pages in a virtual buffer pool can be in use, updated, or available.

* In-use pages are currently being read or updated. The data that they contain is available for use by other applications.
* Updated pages contain data that has changed but is not yet written to disk.
* Available pages are ready for use. An incoming page of new data can overwrite available pages.

To avoid disk I/O, you can use updated and available pages that contain data.

When data in the buffer changes, that data must eventually be written back to disk. Because DB2 does not need to write the data to disk right away, the data can remain in the buffer pool for other uses. The data remains in the buffer until DB2 decides to use the space for another page. Until that time, applications can read or change the data without a disk I/O operation.

The key factor that affects the performance of buffer pools is their size. The method that DB2 uses to access buffer pools also affects performance.

**Buffer pool size**

The size of buffer pools is critical to the performance characteristics of an application or a group of applications that access data in those buffer pools.

Tuning your buffer pools can improve the response time and throughput for your applications and provide optimum resource utilization. For example, applications that do online transaction processing are more likely to need large buffer pools because they often need to reaccess data. In that case, storing large amounts of data in a buffer pool enables applications to access data more efficiently.

By making buffer pools as large as possible, you can achieve the following benefits:

* Fewer I/O operations result, which means faster access to your data.
* I/O contention is reduced for the most frequently used tables and indexes.
* Sort speed is increased because of the reduction in I/O contention for work files.

You can use the ALTER BUFFERPOOL command to change the size and other characteristics of a buffer pool at any time while DB2 is running. Use the DISPLAY BUFFERPOOL and ALTER BUFFERPOOL commands to gather buffer pool information and change buffer pool sizes.

**Q. Expalain DCLGEN**

**ANS.**DCLGEN stands for **DeCLarationGENerator.**It is an IBM provided function which generates INCLUDE members for DB2 tables for use in COBOL and PL/1 programs. These INCLUDE members contain [SQLhttp://images.intellitxt.com/ast/adTypes/icon1.png](http://www.geekinterview.com/question_details/11191) table declarations and working storage structures.

In COBOL DB2 programs the host variables are needed for the column names of the tables. This can be done by a DB2 utility called DCLGEN Declaration Generator. This is primarily a copybook of host variables and also contains table structure and helps in programming reference. We can code this manually but using DCLGEN we can eliminate the mistakes by replacing all underscores in column names into (hyphens).

DCLGEN is a declaration generator used to generate DB2 equivalent COBOL variables called Host variables.

Which can be used to generate host variables with equivalent data types of DB2 columns.

DB2 Table  <>  DCLGEN <> COBOL

**Host variables:**

1)  Can be used to pass the data from cobol program to db2 table or DB2 table to COBOL

2)  When host variables are coded with sql statements it must be prefixed with : like :hv-cname.

3) Table name must be supplied as input to DCLGEN & partition dataset should be as output.

4) After creating DCLGEN variables which must be copied to application program in  WORKING-STORAGE SECTION by using include command i.e.

Exec sql

Inlcude DCLGEN

End-exec.

5) Include & copy have the same functionality

**Q. What is reorg in DB2**

**ANS:**REORG is used to help DB2 point to accurate data (ie, indexes should become aware of fresh data and no longer include deleted data), as well as "collapse" empty page space created by deletion of data and/or indexes. It can also help move data that is related closer to each other for more efficient access (especially true in the case of a cluster index).

RUNSTATS is used to help gather updated statistics on the volume and/or distribution of data within tables and indexes. This information is stored in the system tables and is used by many things including the optimizer to determine the optimal path to query the data. This is especially true as data grows. The data may distribute in a way that causes the optimizer to now include an index that it didn't before in its access path (or perhaps vice versa and choose not to use an index). RUNSTATS can also affect how a REORG runs - at least in older versions of DB2.

Based on what I have read from DB2 DBA's you generally want to run

1. RUNSTATS
2. REORG
3. RUNSTATS

The first RUNSTATS helps the REORG to work appropriately and efficiently. The RUNSTATS after is more to make sure that stats are now accurate given all the data movement around on pages. Depending on where you read you may see DBA's mention the last RUNSTATS is no longer needed with newer versions of DB2. Since we haven't seen definitive answers on that and since most DB2 DBA's I've heard of follow the above order, our company has chosen to use that order (for the record we are on DB2 LUW 9.7 FP4 and we plan on migrating to V 10.1 sometime next year).

Also, just for the sake of completeness, generally when you update the statistics and you affect the optimizer you want all applications that call into DB2 to make sure they are taking advantage of those new optimizations. So a REBIND of packages will make sure that static packages know of the updated statistics and optimized query paths, and a FLUSH PACKAGE CACHE DYNAMIC cleans out dynamic queries from things like Hibernate, so that way the queries will be rebound with the updated query paths. (**NOTE**: on z/OS I see you have BIND and REBIND options. Perhaps the FLUSH PACKAGE CACHE DYNAMIC is under those or has a different name on z.)

So ultimately I would do the following:

1. RUNSTATS
2. REORG
3. RUNSTATS
4. REBIND packages
5. FLUSH PACKAGE CACHE DYNAMIC (on LUW) or whatever the equivalent is on z/OS.

**Q.What is clustered index in DB2**

**ANS.**In other words, a clustered index stores the actual data, where a non-clustered index is a pointer to the data. In most DBMSs, you can only have one clustered index per table, though there are systems that support multiple clusters (DB2 being an example).   
  
Like a regular index that is stored unsorted in a database table, a clustered index can be a composite index, such as a concatenation of first name and last name in a table of personal information.

if Oracle likes heaps, but SQL Server likes clusters, how do you design for both? I’d say you can’t.

**Q.What is cluster ratio in db2**

**Q.What is clustering in db2**

**Q.What is member Cluster in db2**

**Q. What is error code for end of record/record not found in DB2?**

A. 100

**Q. Then How you will find that, Error is for end of record or record not found?**

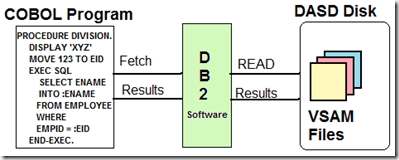
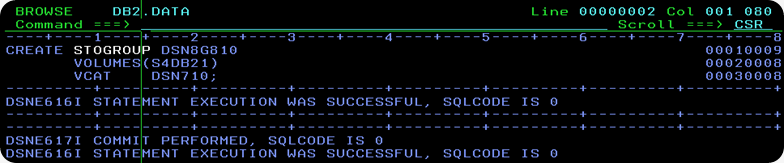
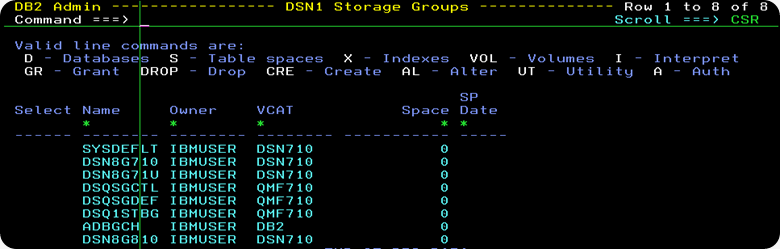
A. If we are using select statement, sql error code 100 represents record not found

If we are using cursor , in that situation, sql error code 100 represents end of records

**Q. What is DB2? How does it compare with VSAM Files?**

DB2 is the Data-Management System from IBM. DB2 is a newer technology, whilst VSAM is old. VSAM(which stands for Virtual Sequential Access Method) is just a Data Access-Method. You can browse a VSAM Dataset in Sequential Mode – Record-by-record. You can do random access – jump(hop), fly directly to a particular Record . Skip-Sequential is like the best of both worlds – Jump to 1st record, and then start browsing the subsequent records one-by-one. Sounds cool, ain't it, but hold on and have a look at what DB2 Software offers!   
  
DB2 is a complete Software Package, that allows you to create, store, update and retrieve Data in a Systematic Manner.   
  
1. Data Abstraction : It presents a Logical, Structured view of the Data to the User. The user doesn’t need to worry about the details of how the Data is Physically stored on the DASD-Disk.   
2. Data Independence : DB2 provides a layer of Independence, between the application or COBOL Program and the Data. Every-time the structure of the Data changes(say you add a new field to the EMPLOYEE File), you don’t need re-write the entire COBOL Source code again.   
3. Data Security : DB2 Software controls the Access to the Data. Not everybody can see all the Data. You can see it, only if you've got all the necessary privileges.   
4. Transaction Management and Concurrency : DB2 Manages all the Transactions, and ensures the Data remains consistent. It provides a Locking-Mechanism, for concurrent access to the Data.   
5. Data Recovery : When there are Crashes, DB2 allows you to recover swiftly, and restore the original Data.

**Q. What are the DB2 Data Structures?**

DB2 Software physically stores the Data in VSAM Files on the DASD Disk.   
  
  [](http://lh5.ggpht.com/_sQvdFWqMlMg/TLr7VQc5_2I/AAAAAAAADS4/I4SxVVzTG3A/s1600-h/Image321%5b2%5d.png)  
STOGROUP : The Set of DASD Volumes that will be used to house the VSAM Files containing the Data. In the example below, I've created a STOGROUP called DSN8G810 with the DASD Volume S4DB21.   
  
[](http://lh4.ggpht.com/_sQvdFWqMlMg/TLr7ZkSf2qI/AAAAAAAADTA/CQATyBVTz4o/s1600-h/Image322%5b4%5d.png)  
  
The information about all the STOGROUP's(Storage Groups) is stored in the DB2 Catalog Table SYSIBM.SYSSTOGROUP. The snapshot of all the STOGROUP's is shown in the picture below.   
  
[](http://lh3.ggpht.com/_sQvdFWqMlMg/TLr7fL7Yk0I/AAAAAAAADTI/LwFPJn6WWus/s1600-h/Image323%5b5%5d.png)

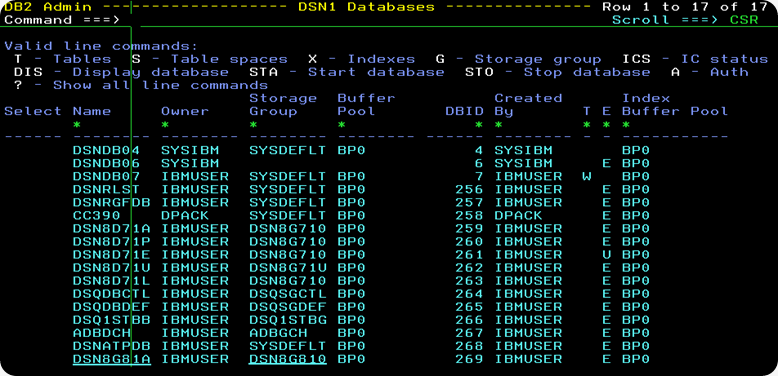
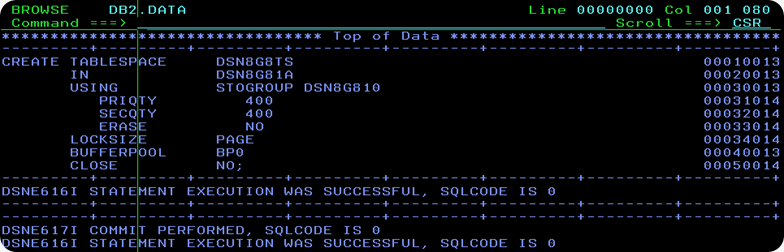
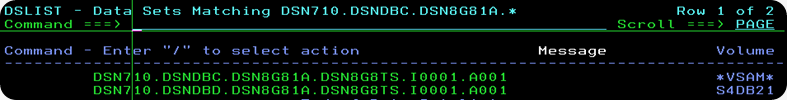
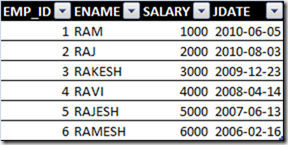
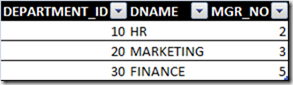
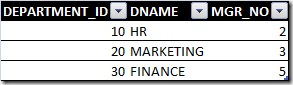
Database : A Database is a Logical-Grouping of Database Objects, related together by application, or subsystem. A Database contains no Data of its own, but it merely groups like DB2 Objects. To create a New Database, use the CREATE DATABASE Statement. In the example below, I have created a New Database called DSN8G81A on the STOGROUP DSN8G810.   
  
[](http://lh5.ggpht.com/_sQvdFWqMlMg/TLr7j42j3wI/AAAAAAAADTQ/OGazorw2fac/s1600-h/Image324%5b4%5d.png)  
  
The information on all the Databases in DB2 System is stored in the DB2 Catalog Table SYSIBM.SYSDATABASE. The below picture is a snapshot of it.   
  
[](http://lh6.ggpht.com/_sQvdFWqMlMg/TLr7oJCfWiI/AAAAAAAADTY/GyPbwoY6o0M/s1600-h/Image325%5b6%5d.png)

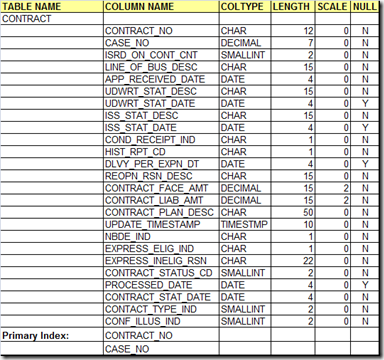
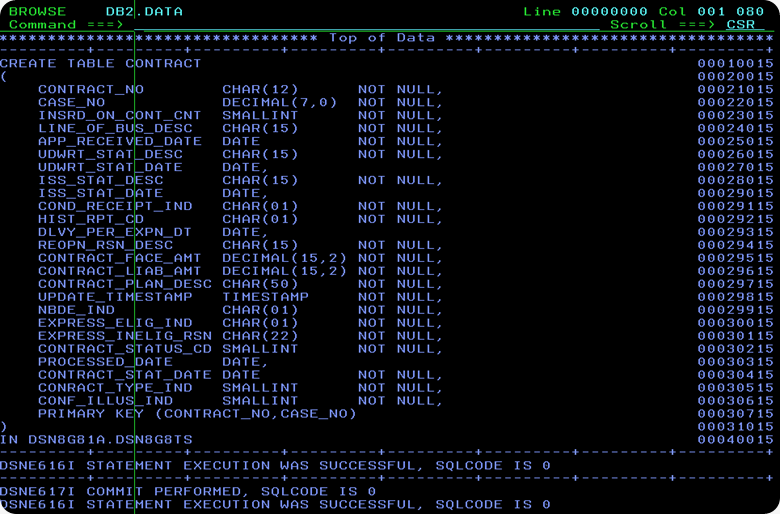
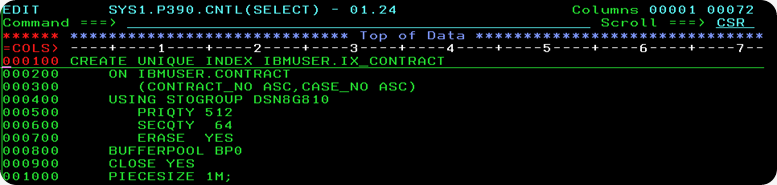
Table-space : A Table-Space is a VSAM File which stores or houses the DB2 Table-Data Physically. You can create a Brand-New Table-space File, by using the CREATE TABLESPACE Statement.   
  
[](http://lh6.ggpht.com/_sQvdFWqMlMg/TLr7su2xL9I/AAAAAAAADTg/niHSmHvOuSk/s1600-h/Image326%5b4%5d.png)  
  
In the picture above, I have created a new Table-space File called DSN8G8TS, which has a Primary-Space of 400KB and Secondary Space of 400KB. Just like the contents in a Book, the data in a Table-Space File is stored in Fixed-Size Pages. DSN8G8TS Table-Space is created under the DSN8G81A Database Umbrella. The new Table-Space File will be stored on DSN8G810 Storage-Group. A buffer-pool is the place in Main Storage, where DB2 will store Pages fetched from the Table-Space.   
  
[](http://lh5.ggpht.com/_sQvdFWqMlMg/TLr7xXNRzyI/AAAAAAAADTo/RW_gflI-nOA/s1600-h/Image327%5b4%5d.png)  
  
The information about all Table-Space Files is stored in the DB2 Catalog Table SYSIBM.SYSTABLESPACE. A snapshot of this is shown in the above picture.

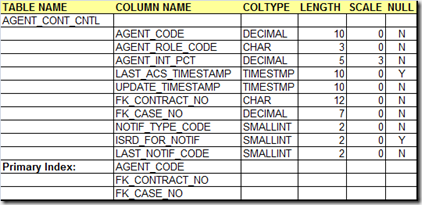
Once the Table-Space File has been newly created, you can actually see it using DSLIST Utility in ISPF Menu 3.4. The VSAM Dataset-name has the format vcat.DSNDBC.dbname.tsname.\*.   
  
[](http://lh6.ggpht.com/_sQvdFWqMlMg/TLr71nrVUdI/AAAAAAAADTw/KkYughDZy4w/s1600-h/Image328%5b4%5d.png)

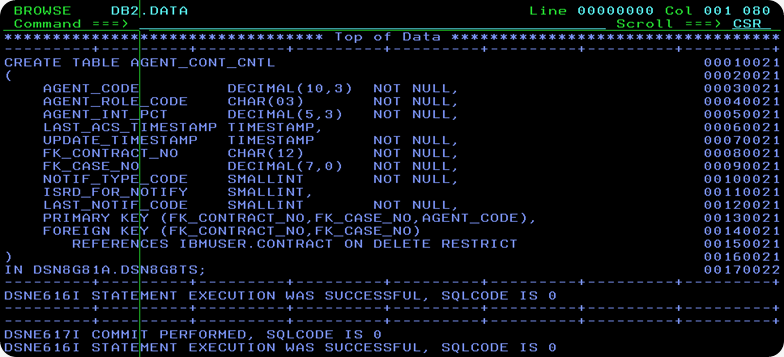
**Q. What are Tables, Primary Keys and Foreign Keys?**

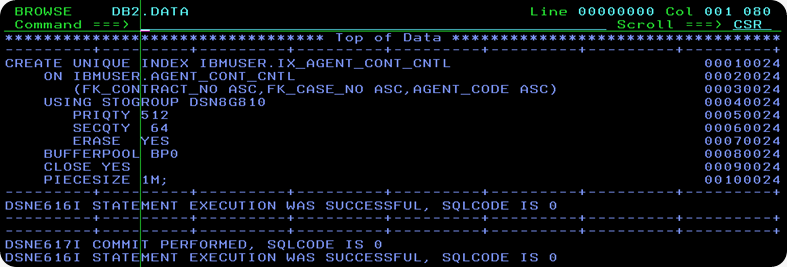
A Table stores Data. Tables look like a spreadsheet, they consist of Rows and Columns, as shown in the picture below.    
  
  [](http://lh5.ggpht.com/_sQvdFWqMlMg/TLr75l5oELI/AAAAAAAADT4/7CgUTrOJER8/s1600-h/Image330%5b5%5d.png)    [](http://lh3.ggpht.com/_sQvdFWqMlMg/TLr7-fQnpjI/AAAAAAAADUA/04I7kB9oDE4/s1600-h/Image331%5b2%5d.png)  
The Employee-ID Number is unique for each Employee. No two Employees have the same Employee-ID in the Company. Similarly, the Department-ID is unique for each Department. Employee-ID Column is said to be the Primary-Key or   
Primary-Index in the EMPLOYEE Table. Department-ID is said to be the Primary-Key or Primary-Index in the DEPARTMENT Table.   
  
Suppose you want to establish a Relationship, and store the information about, which Employee works for which Department? The EMPLOYEE Table should be expanded and an additional column is added to it, called FK\_DEPT\_ID. EMPLOYEE.FK\_DEPT\_ID Column refers to DEPARTMENT\_ID Primary-key column of the DEPARTMENT Table. FK\_DEPT\_ID is called Foreign-Key. Here, is how it works. RAJ has FK\_DEPT\_ID=10. Look-up Department-No 10 in the DEPARTMENT Table, and it is the HR Department. It means RAM works in HR Department.   
  
 [](http://lh5.ggpht.com/_sQvdFWqMlMg/TLr8DiKANoI/AAAAAAAADUI/Y9N-73XJc0Q/s1600-h/Image332%5b2%5d.png)     [](http://lh6.ggpht.com/_sQvdFWqMlMg/TLr8JPIj86I/AAAAAAAADUQ/KpzBqxnL65Y/s1600-h/Image331%5b3%5d%5b2%5d.png)

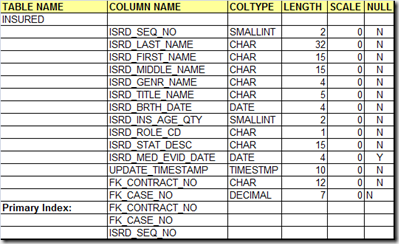
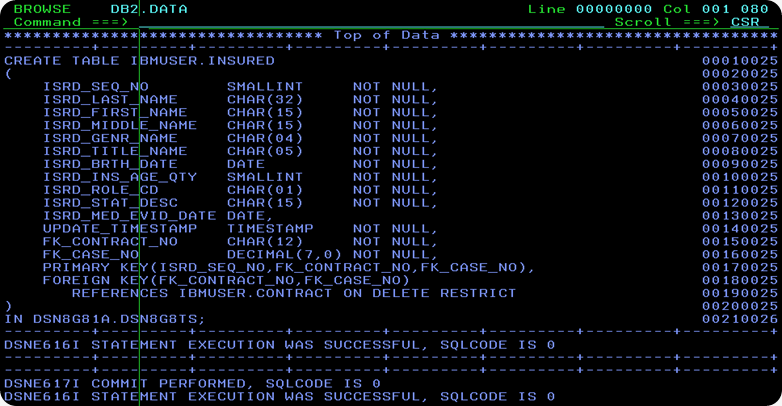
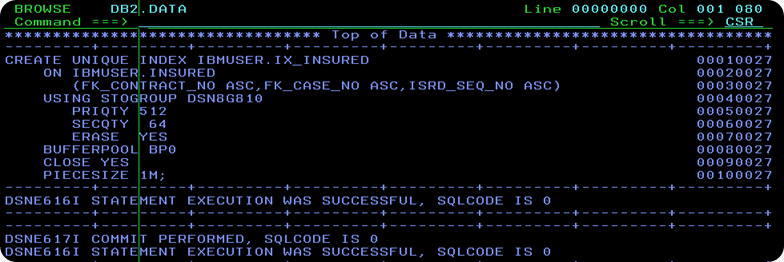
**Q. How do you write the DDL to create Tables and Index Key-Columns?**

As an example, let us build a toy Database-System, called New Business Tracking(NBT), which stores the information about new Insurance Policies(New Business) of an Agent for the first 90 days. This Database System will have three DB2 Tables – CONTRACT, AGENT\_CONT\_CNTL and INSURED.    
  
CONTRACT Table Schema   
The CONTRACT Table stores the basic Details of the Contract, what is the Contract-No,how many Insured's does this Contract cover(ISRD\_ON\_CONT\_CNT), is it a Life or Annuity Policy(LINE\_OF\_BUS\_DESC), when the Application was received, when was the Policy under-written, when the Policy got Issued, what is the Face Amount(Sum-Assured) and many other details.   
 [](http://lh4.ggpht.com/_sQvdFWqMlMg/TLr8PNq1R7I/AAAAAAAADUY/S3xagYlovPM/s1600-h/Image329%5b5%5d.png)   
  
DDL For CONTRACT Table   
You write the CREATE TABLE Statement listing all the Columns followed by their SQL-types, along with whether the Column accepts NULL Values or is it NOT NULL. The List of Columns is a Comma-Separated List. [](http://lh3.ggpht.com/_sQvdFWqMlMg/TLr8V6tZwrI/AAAAAAAADUg/uBj_mHU5vUA/s1600-h/Image333%5b5%5d.png)  
  
Primary Index For CONTRACT Table   
Here's, how I have created  the Primary-Index on the CONTRACT\_NO, CASE\_NO Columns of the CONTRACT Table. I have created a UNIQUE INDEX on the CONTRACT Table.   
    
[](http://lh4.ggpht.com/_sQvdFWqMlMg/TLsmf1ZB1EI/AAAAAAAADVA/iSkwig_FqFo/s1600-h/Image337%5b4%5d.png)

AGENT\_CONT\_CNTL Table Schema   
AGENT\_CONT\_CNTL Table stores the basic data about the Insurance Agent, and the Insurance Policies sold by him.   
 [](http://lh4.ggpht.com/_sQvdFWqMlMg/TLsLbnbWQQI/AAAAAAAADUw/kFiDAbwAuCQ/s1600-h/Image335%5b3%5d.png)

DDL For AGENT\_CONT\_CNTL Table   
The AGENT\_CONT\_CNTL Table stores Agent Details. How do you know, which Policy belongs to which Agent-ID? For each Agent, the Policy-No sold by him is stored in FK\_CONTRACT\_NO Column, and this refers to the CONTRACT\_NO Column in CONTRACT Table. Similarly, FK\_CASE\_NO refers to the CASE\_NO in the CONTRACT Table. Thus, the foreign-key(FK\_CONTRACT\_NO,FK\_CASE\_NO) refers to the   
Primary-Key(CONTRACT\_NO,CASE\_NO) in the CONTRACT Table. One needs to understand, that you cannot create a Foreign-Key without defining a UNIQUE INDEX.   
  
[](http://lh3.ggpht.com/_sQvdFWqMlMg/TLsLenwlg5I/AAAAAAAADU4/ETfq3VQdKdQ/s1600-h/Image336%5b5%5d.png)

Primary Index For AGENT\_CONT\_CNTL Table   
I have created a New Unique-Index IX\_AGENT\_CONT\_CNTL on the columns FK\_CONTRACT\_NO, FK\_CASE\_NO and AGENT\_CODE of the AGENT\_CONT\_CNTL Table.   
    
[](http://lh3.ggpht.com/_sQvdFWqMlMg/TLsmojDNdoI/AAAAAAAADVI/Wpu_ZfqiGoc/s1600-h/Image338%5b5%5d.png)

INSURED Table Schema   
INSURED Table stores  information  about the Insured, or Policy-Holder. In case of a Join-Policy, a Contract may have two or three Insured's. In that case, each Insured for that Policy has a different ISRD\_SEQ\_NO. The Insured's Name, Date-of-Birth etc. is stored in  this Table.   
  [](http://lh3.ggpht.com/_sQvdFWqMlMg/TLsnH3LAXFI/AAAAAAAADVQ/-HTqp_wV9Ls/s1600-h/Image339%5b2%5d.png)    
  
DDL For INSURED Table   
The CREATE TABLE Statement for the INSURED Table is shown in the picture below. How do you know, who is the Policy-Holder for a particular Contract-No? This information is stored  in the Columns FK\_CONTRACT\_NO and FK\_CASE\_NO, which is the  Foreign-Key, and refers to the corresponding columns in the CONTRACT Table.   
[](http://lh3.ggpht.com/_sQvdFWqMlMg/TLsvrSdyvpI/AAAAAAAADVY/pH4pX_63wkw/s1600-h/Image340%5b5%5d.png)  
  
Primary Index For INSURED Table   
I have created a Primary Unique-Index on the ISRD\_SEQ\_NO,FK\_CONTRACT\_NO and FK\_CASE\_NO Columns of the INSURED Table.   
  
[](http://lh4.ggpht.com/_sQvdFWqMlMg/TLsvwhvWGUI/AAAAAAAADVg/GVFKTrCjVh4/s1600-h/Image341%5b5%5d.png)

# DB2 SQL Error Codes

## Retrieving SQL Return Code Information & Messages

COBOL programs executing SQL statements communicate with DB2 via a Working Storage area called the SQL Communications Area (SQLCA).

When DB2 executes SQL statements, it returns the results of the operation into the SQLCODE and SQLSTATE fields in the SQLCA. SQLCODE provides key information about the success or failure of SQL statement execution.

If the SQLWARNO field in the SQLCA contains 'W', DB2 has set at least one of the SQL warning flags (SQLWARN1 through SQLWARNA). These flags provide additional info about execution of specific types of SQL.

Prior to DB2 V8, COBOL programs could call a subroutine called DSNTIAR that would convert a SQLCODE in the SQLCA into more a detailed text message with diagnostics about the return code.

New with DB2 V8, COBOL programs can now execute a GET DIAGNOSTICS statement that will return all previous SQLCA values and provide additional information about new DB2 V8 extended object names and new SQL functions.

The new GET DIAGNOSTICS function replaces existing SQLCA processing now found in most DB2 COBOL programs. GET DIAGNOSTICS also passes a text message about SQLCODE directly to programs.

## SQLCODE Overview

* If SQLCODE = 0, execution was successful.
* If SQLCODE > 0, execution was successful with a warning.
* If SQLCODE < 0, execution was not successful.
* If SQLCODE = 100, "no data" was found. For example, a FETCH statement returned no data because the cursor was positioned after the last row of the result table.
* New with DB2 V8, when DB2 processes a multiple row FETCH statement, the contents of SQLCODE is set to +100 if the last row in the table has been returned with the set of rows.

## SQLCODE - Successful SQL Execution

**SQL Return Code +100** ROW NOT FOUND FOR FETCH, UPDATE OR DELETE, OR THE RESULT OF A QUERY IS AN EMPTY TABLE.  
**Suggestion:** If expecting data, verify WHERE clause for accuracy and completeness.

**SQL Return Code +117** THE NUMBER OF INSERT VALUES IS NOT THE SAME AS THE NUMBER OF OBJECT COLUMNS.  
**Suggestion:** Correct SQL statement to provide only one value for each column in the table.

**SQL Return Code +231** CURRENT POSITION OF CURSOR cursor-name IS NOT VALID FOR FETCH OF THE CURRENT ROW.  
**Suggestion:** Be certain to FETCH to position on a row after opening a cursor. If cursor is declared SENSITIVE STATIC SCROLL, the row may be a hole, from which no values can be fetched.

**SQL Return Code +304** A VALUE WITH DATA TYPE data-type1 CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE OF THE HOST VARIABLE IN POSITION position-number WITH DATA TYPE data-type2.  
**Suggestion:** Verify DCLGEN host variable definitions are current with DB2 catalog table/view attributes.

**SQL Return Code +347** THE RECURSIVE COMMON TABLE EXPRESSION name MAY CONTAIN AN INFINITE LOOP.  
**Suggestion:** Verify predicate in the SQL WHERE clause of the form "counter\_col < constant" or "counter\_col < :hostvar".

**SQL Return Code +802** EXCEPTION ERROR exception-type HAS OCCURRED DURING operation-type OPERATION ON data-type DATA, POSITION position-number.  
**Suggestion:** Check arithmetic operation for divide by zero or result to exceed size of host variable.

## SQLCODE - Unsuccessful SQL Execution

**SQL Error Code -117** THE NUMBER OF VALUES ASSIGNED IS NOT THE SAME AS THE NUMBER OF SPECIFIED OR IMPLIED COLUMNS.  
**Suggestion:** Provide one value for each column in the table.

**SQL Error Code -150** THE OBJECT OF THE INSERT, DELETE, OR UPDATE STATEMENT IS A VIEW, SYSTEM-MAINTAINED MATERIALIZED QUERY TABLE, OR TRANSITION TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED.  
**Suggestion:** Be certain to specify base DB2 table/view names for INSERT statements.

**SQL Error Code -180** THE DATE, TIME OR TIMESTAMP VALUE value IS INVALID.  
**Suggestion:** Verify the data value is in the correct range and value type.

**SQL Error Code -181** THE STRING REPRESENTATION OF A DATETIME VALUE IS NOT A VALID DATETIME VALUE.  
**Suggestion:** Verify data format with the SQL Reference Guide.

**SQL Error Code -204** name IS AN UNDEFINED NAME.  
**Suggestion:** Correct DB2 CREATOR or OBJECT NAMEs located in SQL statements.

**SQL Error Code -227** FETCH fetch-orientation IS NOT ALLOWED, BECAUSE CURSOR cursor-name HAS AN UNKNOWN POSITION (sqlcode,sqlstate).  
**Suggestion:** CLOSE and re-OPEN the cursor; For scrollable use (FIRST, LAST, BEFORE, AFTER, or ABSOLUTE) to establish valid position.

**SQL Error Code -305** THE NULL VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER position-number BECAUSE NO INDICATOR VARIABLE IS SPECIFIED.  
**Suggestion:** Add null indicator variable to SELECT statement in the format of "column:hostvarind".

**SQL Error Code -501** THE CURSOR IDENTIFIED IN A FETCH OR CLOSE STATEMENT IS NOT OPEN.  
**Suggestion:** Correct logic in application program to OPEN the cursor before the FETCH or CLOSE statement.

**SQL Error Code -502** THE CURSOR IDENTIFIED IN AN OPEN STATEMENT IS ALREADY OPEN.  
**Suggestion:** Correct logic in application program to CLOSE the CURSOR before the OPEN statement.

**SQL Error Code -503** A COLUMN CANNOT BE UPDATED BECAUSE IT IS NOT IDENTIFIED IN THE UPDATE CLAUSE OF THE SELECT STATEMENT OF THE CURSOR.  
**Suggestion:** Use FOR UPDATE statement in your cursor.

**SQL Error Code -530** THE INSERT OR UPDATE VALUE OF FOREIGN KEY constraint-name IS INVALID.  
**Suggestion:** Ensure that INSERT row for DB2 PARENT table is completed before INSERT row in CHILD table

**SQL Error Code -532** THE RELATIONSHIP constraint-name RESTRICTS THE DELETION OF ROW WITH RID X'rid-number'.  
**Suggestion:** Change the program to DELETE CHILD table row before DELETE of row on PARENT table.

**SQL Error Code -551** auth-id DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION operation ON OBJECT object-name.  
**Suggestion:** Contact the support DBA to GRANT the needed privilege.

**SQL Error Code -803** AN INSERTED OR UPDATED VALUE IS INVALID BECAUSE THE INDEX IN INDEX SPACE indexspace-name CONSTRAINS COLUMNS OF THE TABLE SO NO TWO ROWS CAN CONTAIN DUPLICATE VALUES IN THOSE COLUMNS. RID OF EXISTING ROW IS Xrid.  
**Suggestion:** Verify DB2 INDEX and, if needed, change the statement to an UPDATE.

**SQL Error Code -805** DBRM OR PACKAGE NAME location-name.collection-id.dbrm-name.consistency-token NOT FOUND IN PLAN plan-name. REASON reason.  
**Suggestion:** Ensure COLLECTION name is in DB2 PLAN. Recompile and BIND the DB2 program. Verify correct LOAD library is being used.

**SQL Error Code -811** THE RESULT OF AN EMBEDDED SELECT STATEMENT OR A SUBSELECT IN THE SET CLAUSE OF AN UPDATE STATEMENT IS A TABLE OF MORE THAN ONE ROW, OR THE RESULT OF A SUBQUERY OF A BASIC PREDICATE IS MORE THAN ONE VALUE.  
**Suggestion:** -811 is often detected after program check for DB2 data existence. Consider using new DB2 V8 FETCH FIRST ROW ONLY feature instead.

**SQL Error Code -818** THE PRECOMPILER-GENERATED TIMESTAMP x IN THE LOAD MODULE IS DIFFERENT FROM THE BIND TIMESTAMP y BUILT FROM THE DBRM z.  
**Suggestion:** Recompile and BIND the DB2 program. Verify correct LOAD library is being used.

**SQL Error Code -904** UNSUCCESSFUL EXECUTION CAUSED BY AN UNAVAILABLE RESOURCE. REASON reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name.  
**Suggestion:** -904 is usually caused because a database utility job has started the desired DB2 object in utility mode. Check DB2 Master Log for more details on the resource name – contact DBA.

**SQL Error Code -911** THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR TIMEOUT. REASON reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name.  
**Suggestion:** Review DB2 Master Log to find process holding DB2 locks. Consider adding additional COMMITs to program holding the DB2 resource.

**SQL Error Code -913** UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT. REASON CODE reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name.  
**Suggestion:** Review DB2 Master Log to find process holding DB2 locks. Consider adding additional COMMITs to program holding the DB2 resource.

**SQL Error Code -922** AUTHORIZATION FAILURE: error-type ERROR. REASON reason-code.  
**Suggestion:** Connection to DB2 has failed due authority for USER or PLAN. Contact DBA to check DB2 authorizations.

**SQL Error Code -927** THE LANGUAGE INTERFACE (LI) WAS CALLED WHEN THE CONNECTING ENVIRONMENT WAS NOT ESTABLISHED. THE PROGRAM SHOULD BE INVOKED UNDER THE DSN COMMAND.

**Interview Questions**

**Q: What is JOIN and what are the different types of JOIN.  
A:** The ability to join rows and combine data from two or more tables is one of the most powerful features of relational system. There are three type of joins:1. Equi-join; 2.Non-equijoin; 3.self-join

**Q: Which is the most widely used batch performance monitor for DB2?  
A:** DB2PM

**Q: Can I alter a table (like adding a column) when other user is selecting some columns or updating some columns from the same table?**   
**A:** Yes. It is possible until the updating or selection is committed, DB2 table will not be restructured. New column definition will be there but it will not be included until all the tasks on the table are committed.

**Q: What are the different methods of accessing DB2 from TSO? How is the connection established between TSO & DB2? A:**  
There are three ways in establishing TSO/DB2 connection 1. **SPUFI** 2. QMF 3. CATALOG VISIBILITY; A thread between TSO & DB2 is established while attempting to make connection between TSO & DB2.

**Q: How many buffer bools are available in DB2?  
A:** Ten 32K size bufferpools and fifty 4K size buffer pools (BP0 to BP49). Default buffer pools are BP0, BP1, BP2 & BP32

**Q: What is B37 abend during SPUFI?  
A:** The B37 ABEND in the SPUFI is because of space requirements. The query has resulted in so many rows that the SPUFI.OUT file is not large enough to handle it; One possible solution is to increase the space allocation of SPUFI.OUT file.

**Q: What is the command used by TSO users to invoke DB2?  
A:** DSN RUN

**Q: What is the error code -803 ?  
A**: Unique Index violation

**Q: How do you filter out the rows retrieved from a DB2 table?  
A:** One way is to use the SQL WHERE clause.

**Q: What is a collection?  
A:** A collection is something that every programmer should assign/specify for every package. This is about 1-18 characters long.

**Q: What is Skeleton Cursor Table (SKCT)?  
A:** The Executable form of a Plan. This is stored in SYSIBM.SCT02 table.

**Q: What is the equivalent Cobol Data type for Decimal (x,y) in DB2? What does the current SQLID register contain?  
A:** PIC S9(X-Y)V9(Y) COMP-3; The current SQLID contains the current authorization ID.

**Q.Can u please tell me the corelation b/w Plan, Package and collection.**

The DB2 Precompiler splits the program into two parts: a COBOL and a DB2 part. The embedded SQL is stripped out of the program and put into a partitioned data set (PDS) member, called a **DBRM**. Just as the COBOL part has to be compiled, the **DBRM part has to go through BIND process to create the run-time executable code for the DB2 portion of the COBOL program**. To help the COBOL and DB2 part to find each other later at run time, the precompiler engraves each with identical timestamps called **consistency tokens**.   
  
**You can BIND the DBRM into a PLAN (the old way), or you can BIND the instructions into a PACKAGE**.   
  
A PLAN is an executable module containing the access path logic produced by the DB2 optimizer**. The DBRMs of more than one program or PACKAGES can be bound into a PLAN.**  
A PACKAGE is a single, bound DBRM with optimized access paths. The DBRM of a single program is bound into a PACKAGE. To execute a PACKAGE, it should be included in the package list of a PLAN. **PACKAGEs are not directly executed, they are only indirectly executed when the PLAN in which they are contained executes.**   
  
**The relationship between a DBRM and a PLAN is one-to-many, the relationship between a DBRM and a PACKAGE is always one-to-one.**   
  
As the number of DBRMs bound to a PLAN increases, binding the DBRM into a PLAN is not recommended. If we need to precompile and bind a new program or one of the programs changes and it is to be precompiled and bound again, all the programs (not just the modified/added program) will be rebound into the PLAN again. Then the BIND process could take hours to complete.   
  
**On the other hand, if a DBRM is bound to a PACKAGE and if the program is modified, only that PACKAGE would have to be rebound.**   
  
A collection is simply a way of grouping PACKAGEs into meaningful groups. You could use COLLECTIONs to separate programs for different application areas, such as payroll and inventory. Another use might be to have customized set of BIND parameters associated with different COLLECTIONs.   
  
**At run time, the load module starts up and eventually hits a paragraph containing a CALL to DB2. Then the COLLECTIONs named in the PLAN are searched for the PACKAGE with the same name and consistency token. If you don't find it anywhere in DB2, you get an -805 error. If you're using the older technique of binding DBRMs directly into PLANs, then an unsuccessful search will result in an -818 error code.**

**Q: Can we declare DB2 HOST variable in COBOL COPY book?  
A:** NO. If we declare DB2 host variable in COBOL COPY book, at the time of Pre-compilation we get the host variable not defined, because pre-compiler will not expand COBOL COPY book. So we declare it either in DCLGEN with EXEC SQL INCLUDE DCLGEN NAME END-EXEC or we directly hardcode it in the working storage section.

**Q: What should be specified along with a cursor in order to continue updating process after COMMIT?  
A:** With Hold option.

**Q: What is the name of the default DB2 catalog database?  
A:** DSNDB06

**Q:** [**CURSORS in DB2**](http://mainframegeek.me/2011/08/17/cursors-in-db2/)

When we are using the DB2 in our applications we can only have one row of data at a time. So what we will do if we don’t know which row exactly we need?, what if we have more than 1 row to work with? well the answer is “CURSORS”.

Cursor is used when more than one row are to be selected.  Cursors has mainly 4 control statements.

1. Declare.

A name will be assigned for particular SQL statement. The name should be unique in the scope of the program. there are no limits for the number of cursors which we can have in one application program. We can declare cursor in Working storage section or Procedure division.  
E.g.  
EXEC SQL                                    
    DECLARE CURREAD1 CURSOR FOR             
        SELECT NAME,SEQ FROM IBMGRP.MYNAM   
END-EXEC.

2. Open.

This statement builds the resultant table.   
E.g.   
EXEC SQL          
   OPEN CURREAD1  
END-EXEC.

3. Fetch.

Fetch statement will returns data from the resultant table (One row at a time) and assigns values to the specified host variables.  
E.g.  
EXEC SQL                                 
   FETCH CURREAD1 INTO :WS-NAME,:WS-SEQ  
END-EXEC

4. Close

Empty all the resources used by the cursor.  
E.g.  
EXEC SQL           
   CLOSE CURREAD1  
END-EXEC.

**All these control statements will throw specific SQLCODES.**

Few Snippets.

Read table.

EXEC SQL                                   
    DECLARE CURREAD1 CURSOR FOR            
        SELECT NAME,SEQ FROM IBMGRP.MYNAM  
END-EXEC.   
Open  
EXEC SQL          
   OPEN CURREAD1  
END-EXEC.

PERFORM UNTIL SQLCODE = 100                            
   EXEC SQL                                            
      FETCH CURREAD1 INTO :WS-NAME,:WS-SEQ             
   END-EXEC                                            
   MOVE SQLCODE             TO WS-SQLCODE              
   DISPLAY ‘SQLCODE FETCH ‘ WS-SQLCODE                 
   IF SQLCODE = 0 THEN                                 
    MOVE SPACES TO TEMP-MSG                            
    STRING                                             
    ‘NAME: ‘ DELIMITED BY SPACE ‘ ‘ DELIMITED BY SIZE  
    WS-NAME DELIMITED BY SPACE ‘,’ DELIMITED BY SIZE   
    ‘SEQ#’ DELIMITED BY SPACE ‘ ‘ DELIMITED BY SIZE    
    WS-SEQ DELIMITED BY SIZE INTO TEMP-MSG             
    DISPLAY TEMP-MSG                                   
   END-IF                                              
END-PERFORM.

Fetch name and SEQ till we hit SQLCODE 100 and display the data.

Update Table

We need to mention FOR UPDATE OF and the field name in declare statement

EXEC SQL                                   
    DECLARE CURUPDT1 CURSOR FOR            
        SELECT NAME,SEQ FROM IBMGRP.MYNAM  
        WHERE SEQ = :WS-SEQ                
        FOR UPDATE OF NAME                 
END-EXEC.        
here I will be updating NAME field of MYNAM table                             
EXEC SQL                                   
   OPEN CURUPDT1   
END-EXEC.  
MOVE ’002′               TO WS-SEQ.          
EXEC SQL                                     
   FETCH CURUPDT1 INTO :WS-NAME,:WS-SEQ-TMP  
END-EXEC                                          
EXEC SQL                                     
   UPDATE IBMGRP.MYNAM                       
   SET NAME = :WS-NAME                       
   WHERE CURRENT OF CURUPDT1                       
END-EXEC.      
**“CURRENT OF CURUPDT1″ statement will pick the current row to update.**      
EXEC SQL                                     
   CLOSE CURUPDT1                            
END-EXEC.

Delete Record

Like cursor for updating a record we need to mention FOR UPDATE OF in cursor declaration statement.

EXEC SQL                                   
    DECLARE CURDELT1 CURSOR FOR            
        SELECT NAME,SEQ FROM IBMGRP.MYNAM  
        WHERE SEQ = :WS-SEQ                
        FOR UPDATE OF NAME                 
END-EXEC.                                  
EXEC SQL                                   
   OPEN CURDELT1                           
END-EXEC.                                                                 
EXEC SQL                                     
   FETCH CURDELT1 INTO :WS-NAME,:WS-SEQ-TMP  
END-EXEC                                     
EXEC SQL                      
   DELETE FROM IBMGRP.MYNAM   
   WHERE SEQ = :WS-SEQ        
END-EXEC.                     
EXEC SQL                
   CLOSE CURDELT1       
END-EXEC.

**Q.What is difference between 911 and 913 error code in DB2 ?**

911 - THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO        
DEADLOCK OR TIMEOUT. REASON reason-code, TYPE OF RESOURCE   
resource-type, AND RESOURCE NAME resource-name

913-UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT.    
REASON reason-code, TYPE OF RESOURCE resource-type, AND  
RESOURCE NAME resource-name

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END of questions & and answers \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DB2 TOOLS**

 Go to **DB2I** PRIMARY OPTION MENU

4.  DB2I PRIMARY OPTION MENU

                             DB2I PRIMARY OPTION MENU          SSID: DSN          
 COMMAND ===>                                                                     
                                                                                  
 Select one of the following DB2 functions and press ENTER.                       
                                                                                  
  1  SPUFI                  (Process SQL statements)                              
  2  **DCLGEN**                 (Generate SQL and source language declarations)       
  3  PROGRAM PREPARATION    (Prepare a DB2 application program to run)            
  4  PRECOMPILE             (Invoke DB2 precompiler)                              
  5  BIND/REBIND/FREE       (BIND, REBIND, or FREE plans or packages)             
  6  RUN                    (RUN an SQL program)                                  
  7  DB2 COMMANDS           (Issue DB2 commands)                                  
  8  UTILITIES              (Invoke DB2 utilities)                                
  D  DB2I DEFAULTS          (Set global parameters)                               
  X  EXIT                   (Leave DB2I)                                          
                                                                                  
                    
                                                                                  
                                                                                  
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE     
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE

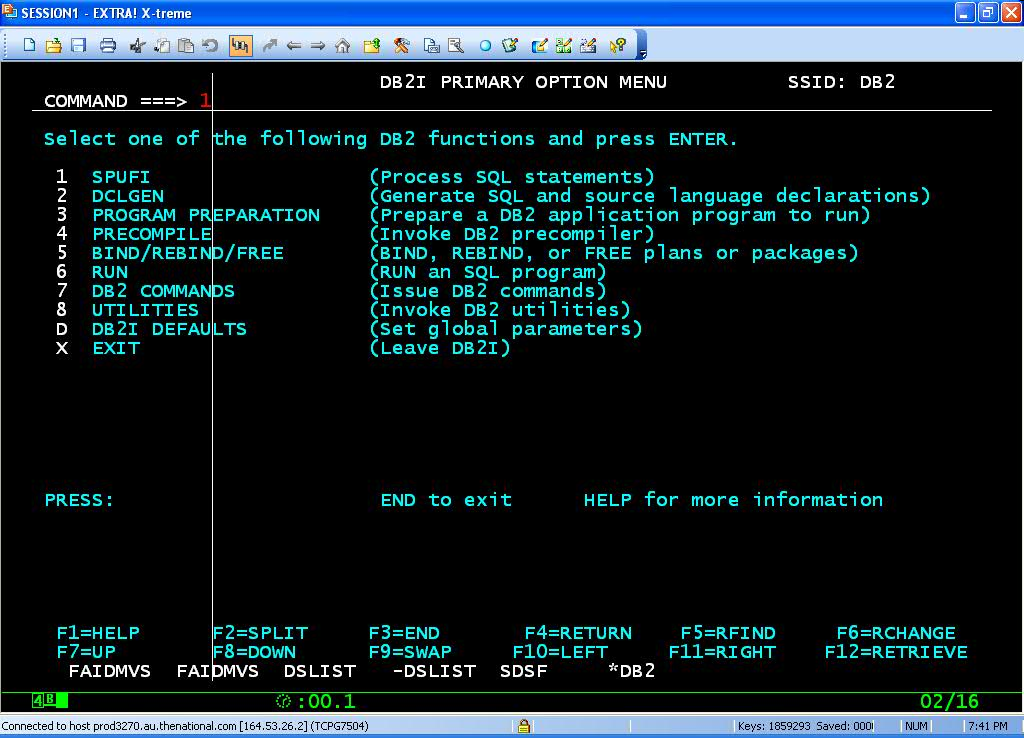
Fill the table name and the required options

                            DCLGEN                             SSID: DSN          
 ===>                                                                             
                                                                                  
 Enter table name for which declarations are required:                            
  1  **SOURCE TABLE NAME** ===> IND\_SUPP\_TAB                     (Unqualified)        
  2  TABLE OWNER ..... ===>                                     (Optional)        
  3  AT LOCATION ..... ===>                                     (Optional)        
                                                                                  
 Enter destination data set:          (Can be sequential or partitioned)          
  4  **DATA SET NAME** ... ===> 'INDU77.INDUS.DCLGENS(SUPPTAB)'                       
  5  DATA SET PASSWORD ===>           (If password protected)                     
                                                                                  
 Enter options as desired:                                                        
  6  ACTION .......... ===> ADD       (ADD new or REPLACE old declaration)        
  7  COLUMN LABEL .... ===> YES       (Enter YES for column label)                
  8  STRUCTURE NAME .. ===>                                     (Optional)        
  9  **FIELD NAME PREFIX** ===> HS-                                 (Optional)        
 10  DELIMIT DBCS .... ===> YES       (Enter YES to delimit DBCS identifiers)     
 11  COLUMN SUFFIX ... ===> YES       (Enter YES to append column name)           
 12  INDICATOR VARS .. ===> YES       (Enter YES for indicator variables)         
                                                                                  
                                                                                  
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE     
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE

|  |  |
| --- | --- |
| **Sample DCLGEN copy book :**      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* DCLGEN TABLE(IND\_STU)                                           \*        LIBRARY(INDU777.INDUS.COPYLIB(STUCPY))  \*        ACTION(REPLACE)                                          \*        LANGUAGE(COBOL)                                         \*        NAMES(HS-)                                              \*        STRUCTURE(STU-REC)  \*        QUOTE  \*        LABEL(YES)                                               \*        COLSUFFIX(YES)                                          \*        INDVAR(YES)                                              \* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*      EXEC SQL DECLARE IND\_STU TABLE                                     ( STU\_NO                         CHAR(5),                            STU\_NAME                       CHAR(20),                           STU\_ADDR                       CHAR(20)       ) END-EXEC.                                                   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* COBOL DECLARATION FOR TABLE IND\_STU                             \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  01  STU-REC.                                                      \*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*                       STU\_NO                                          10 HS-STU-NO            PIC X(5).                             \*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*                       STU\_NAME                                        10 HS-STU-NAME          PIC X(20).                            \*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*                       STU\_ADDR                                        10 HS-STU-ADDR          PIC X(20).                            \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* INDICATOR VARIABLE STRUCTURE                                    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  01  IIND-STU.       10 INDSTRUC           PIC S9(4) USAGE COMP OCCURS 3 TIMES.    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 3       \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |  |

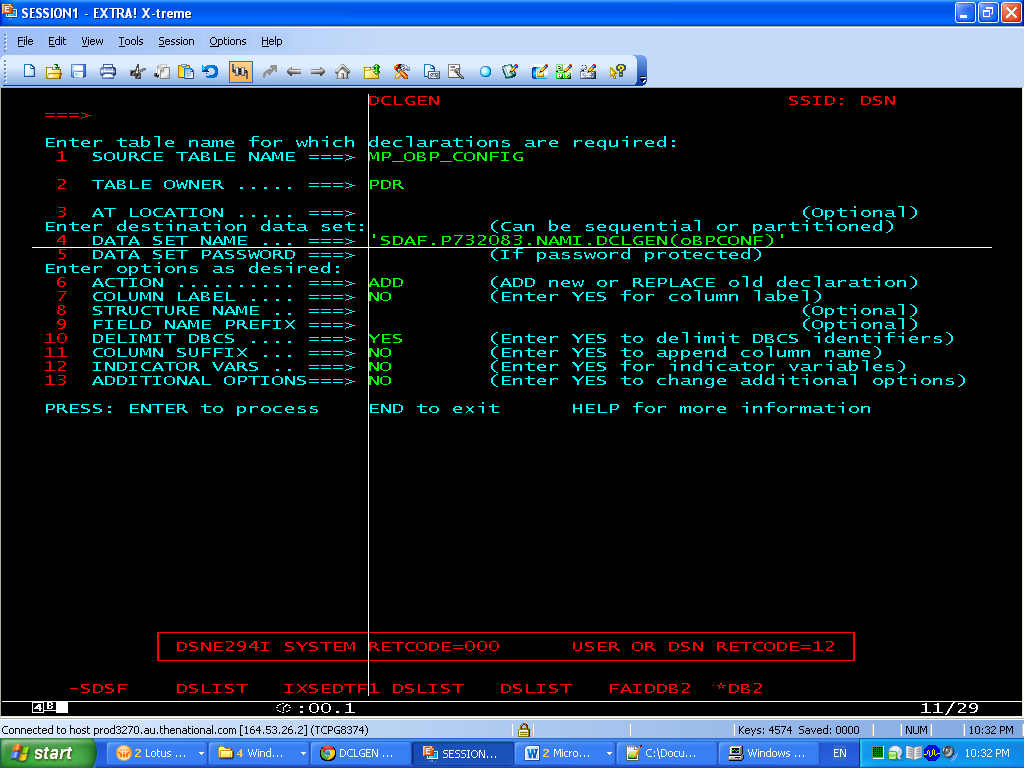
**DB2 Defaults settings**

D ( DB2I)



**DCLGEN**

**Go to D;;**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* DCLGEN TABLE(IND\_STU)                                          \*  
\*        LIBRARY(INDU777.INDUS.COPYLIB(STUCPY))                   \*  
\*        ACTION(REPLACE)                                         \*  
\*        LANGUAGE(COBOL)                                         \*  
\*        NAMES(HS-)                                              \*  
\*        STRUCTURE(STU-REC)                                      \*  
\*        QUOTE                                                   \*  
\*        LABEL(YES)                                              \*  
\*        COLSUFFIX(YES)                                          \*  
\*        INDVAR(YES)                                             \*  
\* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS   \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     EXEC SQL DECLARE IND\_STU TABLE                                
     ( STU\_NO                         CHAR(5),                     
       STU\_NAME                       CHAR(20),                    
       STU\_ADDR                       CHAR(20)

     ) END-EXEC.                                                   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* COBOL DECLARATION FOR TABLE IND\_STU                            \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 01  STU-REC.                                                      
\*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                       STU\_NO                                     
     10 HS-STU-NO            PIC X(5).                             
\*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                       STU\_NAME                                   
     10 HS-STU-NAME          PIC X(20).                            
\*    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*                       STU\_ADDR                                   
     10 HS-STU-ADDR          PIC X(20).                            
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* INDICATOR VARIABLE STRUCTURE                                   \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
**01  IIND-STU.**

**10 INDSTRUC           PIC S9(4) USAGE COMP OCCURS 3 TIMES.**    
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 3       \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ABENDS**

**+100**  -> End of cursor

**-180** -> String representation of DATE, TIME, TIMESTAMP is invalid

**-204** -> Object not defined to DB2

**-205** -> Column name not in specified DB22 table

**-206** -> Column does not exist in any table in the SELECT

**-551** -> Not authorized to access DB2

**-803** -> UnIque index violation. Try to insert duplicate record.

**-805** -> DBRM or PACKAGE not found in PLAN

**-811** -> Cursor should be used , when more than one row returned as an result

of an singleton SELECT query

**-818** -> Timestamp mismatch between plan and load module