CONSOLE CODE

GROUP 5D

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>


EXEC SQL include sqlda.h;
//EXEC SQL WHENEVER NOT FOUND DO BREAK;
EXEC SQL WHENEVER SQLERROR CALL print_sqlca();


sqlda_t *sqlda1; /* an output descriptor */
sqlda_t *sqlda2; /* an input descriptor  */


//EXEC SQL WHENEVER SQLERROR STOP;

void print_sqlca();
int main()
{


    EXEC SQL BEGIN DECLARE SECTION;
    const char *target = "201401103@10.100.71.21";
    const char *user = "201401103";
    const char *password = "alskdj123";
    int intval;
    float floatval;
    int c;
    unsigned long long int longlongval;

//    const char *stmt = "SELECT batch FROM student WHERE sid='1'"; //side query
  //  const char *stmtinsert = "INSERT INTO student VALUES(?,?,?,?,?);";
    char query[1024]= "";
    EXEC SQL END DECLARE SECTION;

    EXEC SQL CONNECT TO :target USER :user USING :password;
    EXEC SQL set search_path to researchportal;

    int mode;

    char *scanmode = (char *) malloc(sizeof(char)); // 1 for insert


    while(true)
    {
    printf("Select action\n0: Select\n1: Insert\n2: Delete\n3: Update\n Ctrl-c to quit\n");
    gets(scanmode);
//    printf("%s\n", scanmode);
```

```
    // scan for action to perform

//   printf("Select action\n0: Select\n1: Insert\n2: Delete\n3: Update\n");
//   gets(scanmode);




   mode = atoi(scanmode);

    if (mode==1)
        {

           printf("enter query for insert\n");
           gets(query);

          // sprintf(query, "INSERT INTO student (sid, sname, prog_name, batch, cpi) values ('105',
'Tingu', 'BTech', 1998, 9.0)");
         // strcpy(query, temp);


          // EXEC SQL EXECUTE IMMEDIATE "INSERT INTO student (sid, sname, prog_name,
batch, cpi) values ('101', 'Pingu', 'BTech', 1996, 10.0)";
         // EXEC SQL PREPARE stmt1 FROM :query;
        //  EXEC SQL EXECUTE stmt1;

           //EXEC SQL EXECUTE IMMEDIATE :query;

           EXEC SQL PREPARE mystmt FROM :query;
           EXEC SQL EXECUTE mystmt;
         //  EXEC SQL EXECUTE mystmt USING 102, 'Tingu', 'BTech', 1996, 9.0;


           EXEC SQL COMMIT;

          // return;
        }
    else if (mode==2)
    {
           printf("enter query for delete\n");
           gets(query);

           EXEC SQL PREPARE mystmt FROM :query;
           EXEC SQL EXECUTE mystmt;


           EXEC SQL COMMIT;


      //     return;
    }
```

```
    else if (mode==3)
    {
//              EXEC SQL EXECUTE IMMEDIATE "UPDATE student SET sname='Pingu' WHERE
sid='100'";

            printf("enter query for update\n");
            gets(query);

            EXEC SQL PREPARE mystmt FROM :query;
            EXEC SQL EXECUTE mystmt;


            EXEC SQL COMMIT;

        //    return;
    }


  else
    {
    printf("enter query for select\n");

       gets(query);


    //fflush(stdout);
   // printf("%s\n",query); //check
     //fflush(stdout);
//end test test ok

    EXEC SQL PREPARE stmt1 FROM :query;
    EXEC SQL DECLARE cur1 CURSOR FOR stmt1;

    /* Create a SQLDA structure for an input parameter */
    /*sqlda2 = (sqlda_t *)malloc(sizeof(sqlda_t) + sizeof(sqlvar_t));
    memset(sqlda2, 0, sizeof(sqlda_t) + sizeof(sqlvar_t));
    sqlda2->sqln = 2; // a number of input variables

    sqlda2->sqlvar[0].sqltype = ECPGt_char;
    sqlda2->sqlvar[0].sqldata = "postgres";
    sqlda2->sqlvar[0].sqllen  = 8;

    intval = 1;
    sqlda2->sqlvar[1].sqltype = ECPGt_int;
    sqlda2->sqlvar[1].sqldata = (char *) &intval;
    sqlda2->sqlvar[1].sqllen  = sizeof(intval);*/



    // open the cursor
    EXEC SQL OPEN cur1; // USING DESCRIPTOR sqlda2; // cursor for result of query
```

```
while (true) // uttam sqlca.sqlcode==0
  {
     sqlda_t *cur_sqlda;

     /* Assign descriptor to the cursor  */
     EXEC SQL FETCH NEXT FROM cur1 INTO DESCRIPTOR sqlda1;

     for (cur_sqlda = sqlda1 ; cur_sqlda != NULL ; cur_sqlda = cur_sqlda->desc_next)
     {
        int i;
        char name_buf[1024];
        char var_buf[1024];

        /* Print every column in a row. */
        for (i=0 ; i<(cur_sqlda->sqld); i++) // 1 less
        {
           sqlvar_t v = cur_sqlda->sqlvar[i];
           char *sqldata = v.sqldata;
           short sqllen  = v.sqllen;

           strncpy(name_buf, v.sqlname.data, v.sqlname.length);
           name_buf[v.sqlname.length] = '\0';

           //printf("dtype:%d\n",v.sqltype);
           switch (v.sqltype)
           {
              case ECPGt_char:
                 memset(&var_buf, 0, sizeof(var_buf));
                 memcpy(&var_buf, sqldata, (sizeof(var_buf)<=sqllen ? sizeof(var_buf)-1 : sqllen) );
                 printf("%s = %s \n", name_buf, var_buf);
                 break;

              case ECPGt_int: // integer
                 memcpy(&intval, sqldata, sqllen);
                 snprintf(var_buf, sizeof(var_buf), "%d", intval);
                 printf("%s = %s \n", name_buf, var_buf);
                 break;

              /*case ECPGt_decimal: // bigint
                 memcpy(&floatval, sqldata, sqllen);
                 snprintf(var_buf, sizeof(var_buf), "%f", floatval);
                 break;*/



              case ECPGt_numeric:

                // char *PGTYPESnumeric_to_asc(numeric *num,0);
                 printf( "%s = %s\n", name_buf, (PGTYPESnumeric_to_asc(sqldata,2)) );
                //  printf("is this repearing?\n");
                 break;
```

```
                case 18:
                   printf("%s = %s\n", name_buf, PGTYPESdate_to_asc(*v.sqldata));
                  // printf("%s = %s \n", name_buf, var_buf);
                   break;

                default:
                {
                   int i;


                   memset(var_buf, 0, sizeof(var_buf));
                   for (i = 0; i < sqllen; i++)
                   {
                      char tmpbuf[16];



                      snprintf(tmpbuf, sizeof(tmpbuf), "%c", (unsigned char) sqldata[i]); // "%02x" ->
"%c"
                      strncat(var_buf, tmpbuf, sizeof(var_buf));

                      printf("%s = %s \n", name_buf, var_buf);
                   }
                }
                  // break; new changed
             }

             //printf("%s = %s (type: %d)\n", name_buf, var_buf, v.sqltype);
            if (sqlca.sqlcode==100)
              break;

             //printf("%s = %s \n", name_buf, var_buf); // removed

          }

          printf("\n");

           printf("--------------------------------------------------------------------------------\n");
            if (sqlca.sqlcode==100)
              break;
       }

          if (sqlca.sqlcode==100)
              break;
    }

     EXEC SQL CLOSE cur1;
    EXEC SQL COMMIT;

    printf("\n");
    printf("\n");
    printf("\n");
```

```
printf("**********************************************************************************
**********************\n");
   }
}
   EXEC SQL DISCONNECT ALL;
return 0;
}

void
print_sqlca()
{
   fprintf(stderr, "==== sqlca ====\n");
   fprintf(stderr, "sqlcode: %ld\n", sqlca.sqlcode);
   fprintf(stderr, "sqlerrm.sqlerrml: %d\n", sqlca.sqlerrm.sqlerrml);
   fprintf(stderr, "sqlerrm.sqlerrmc: %s\n", sqlca.sqlerrm.sqlerrmc);
   fprintf(stderr, "sqlerrd: %ld %ld %ld %ld %ld %ld\n",
sqlca.sqlerrd[0],sqlca.sqlerrd[1],sqlca.sqlerrd[2],
                                        sqlca.sqlerrd[3],sqlca.sqlerrd[4],sqlca.sqlerrd[5]);
   fprintf(stderr, "sqlwarn: %d %d %d %d %d %d %d %d\n", sqlca.sqlwarn[0], sqlca.sqlwarn[1],
sqlca.sqlwarn[2],
                                        sqlca.sqlwarn[3], sqlca.sqlwarn[4], sqlca.sqlwarn[5],
                                        sqlca.sqlwarn[6], sqlca.sqlwarn[7]);
   fprintf(stderr, "sqlstate: %5s\n", sqlca.sqlstate);
   fprintf(stderr, "===============\n");
}
```