```
select * from dba_objects
where object_name like 'PO_HEADERS%'
AND (OBJECT_TYPE  LIKE 'T%' OR OBJECT_TYPE LIKE 'SY%')
AND OWNER IN ('APPS', 'PO')
AND OBJECT_ID IN (41472,1945578);

 apps.fnd_program.add_to_group (program_short_name  => l_program_short_name,
                                                  program_application =>
                                                  l_program_application,

                                                  request_group       =>
                                                  l_request_group,

                                                  group_application   =>
                                                  l_group_application

                                 );

  TRUNC((rq.actual_completion_date – rq.actual_start_date) * 24)
       || ‘ Hour ‘
       || MOD(
             TRUNC(
                (rq.actual_completion_date – rq.actual_start_date) * 1440),
             60)
       || ‘ Min ‘
       || MOD(
             TRUNC(
                (rq.actual_completion_date – rq.actual_start_date) * 86400),
             60)
       || ‘ Sec’
          “Time Taken”
SELECT SYS_CONTEXT ('USERENV', 'DB_NAME')


SELECT SYS_CONTEXT ('USERENV', 'DB_NAME')
            db_name INTO gc_database FROM DUAL


/*
fnd_user                     ------> user information
PER_ALL_PEOPLE_F            ------> connected user infromation
FND_USER_RESP_GROUPS_direct ------> contains responsiblity details
FND_RESPONSIBILITY            ------> contains responsiblity id, key (table)
FND_RESPONSIBILITY_VL        ------> contains aplications id, responsiblity id, key and
name (view)
FND_APPLICATION_VL            ------> contains aplications id, short name, name and
product code (view)
fnd_user_pkg.createuser

*/

;

select *
from per_all_people_f;

select user_name, user_id
from fnd_user;

select user_name, user_id, start_date, end_date, employee_id
    from fnd_user
    where lower(user_name) = 'operations';

select employee_number, npw_number, current_employee_flag, current_npw_flag, person_id,
effective_start_date, effective_end_date, start_date, BUSINESS_GROUP_ID, first_name,
last_name, full_name
from per_all_people_f;

select  fu.user_name,
          fu.user_id,
          fu.start_date,
          fu.end_date,
          fu.employee_id,
          papf.employee_number,
          papf.npw_number,
          papf.current_employee_flag,
          papf.current_npw_flag,
          papf.person_id,
          papf.effective_start_date,
          papf.effective_end_date,
          papf.start_date,
          papf.BUSINESS_GROUP_ID,
          papf.first_name,
          papf.last_name,
          papf.full_name
from per_all_people_f papf,
        fnd_user fu
where 1=1
    and fu.employee_id = &id
    and fu.employee_id = PAPF.person_id;



select *
from fnd_users;fnd_program.delete_program
fnd_program.delete_executable
fnd_program.program_exists
fnd_program.executable_exists
apps.fnd_program.remove_from_group
;

-- delete program
begin

    apps.fnd_program.delete_program( 'XXQGEN_TEST_DK', 'PO');
   DBMS_OUTPUT.PUT_LINE('Program successfully deleted.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error deleting program: ' || SQLERRM);

end;
/

--done

begin

    fnd_program.delete_executable(
        executable_short_name           => 'XXQGEN_TEST_DK',
        application                             => 'PO'
    );

end;
/

-- done

begin

    apps.fnd_program.remove_from_group(
        program_short_name             => 'XXQGEN_TEST_DK',
        program_application              => 'PO',
        request_group                      => 'All Reports',
        group_application                 => 'Purchasing'
    );
    dbms_output.put_line('remove from group successfull');
exception
    when others then
```

```
        dbms_output.put_line('error in remove the group : ' || sqlcode || ' - ' || sqlerrm);

end;
/

-- done

BEGIN
   -- Output the result of the program_exists function
   DBMS_OUTPUT.PUT_LINE(
      'Program Exists: ' ||
      CASE
         WHEN apps.fnd_program.program_exists(
                  program     => 'XXQGEN_TEST_DK',
                  application => 'PO' -- 'Purchasing' maps to 'PO' in short name
              ) THEN 'Yes'
         ELSE 'No'
      END
   );
END;
/

-- done


BEGIN
   -- Output the result of the program_exists function
   DBMS_OUTPUT.PUT_LINE(
      'Executable Exists: ' ||
      CASE
         WHEN apps.fnd_program.executable_exists(
                  executable_short_name     => 'XXQGEN_TEST_DK',
                  application => 'PO' -- 'Purchasing' maps to 'PO' in short name
              ) THEN 'Yes'
         ELSE 'No'
      END
   );
END;
/



SELECT DISTINCT UPCVL.CONCURRENT_PROGRAM_ID,
            UPCVL.CONCURRENT_PROGRAM_NAME,
            UPCVL.EXECUTABLE_APPLICATION_ID,
            UPCVL.EXECUTABLE_ID,
            FEFV.EXECUTION_FILE_NAME,
            FEFV.DESCRIPTION,
            UPCVL.APPLICATION_ID,
            FAVL.APPLICATION_NAME,
            FAVL.APPLICATION_SHORT_NAME ,
            FCR.REQUEST_ID,
            SC.MEANING STATUS_CODE,
            PC.MEANING PHASE_CODE,
            FCR.ACTUAL_COMPLETION_DATE - FCR.ACTUAL_START_DATE
FROM    FND_CONCURRENT_PROGRAMS_VL UPCVL,
             FND_APPLICATION_VL FAVL,
             FND_EXECUTABLES_FORM_V FEFV,
             FND_CONCURRENT_REQUESTS FCR,
            (SELECT LOOKUP_CODE, MEANING
             FROM FND_LOOKUP_VALUES
             WHERE LOOKUP_TYPE = 'CP_STATUS_CODE' AND LANGUAGE = 'US'
             AND ENABLED_FLAG = 'Y') SC,
             (SELECT LOOKUP_CODE, MEANING
              FROM FND_LOOKUP_VALUES
              WHERE LOOKUP_TYPE = 'CP_PHASE_CODE' AND LANGUAGE = 'US'
              AND ENABLED_FLAG = 'Y') PC
WHERE 1=1
```

```
AND     UPCVL.CONCURRENT_PROGRAM_NAME = 'XXQGEN_USER_DATA_PGK_DK'
AND     UPCVL.CONCURRENT_PROGRAM_ID  = FCR.CONCURRENT_PROGRAM_ID
AND     UPCVL.APPLICATION_ID = FAVL.APPLICATION_ID
AND     UPCVL.EXECUTABLE_ID = FEFV.EXECUTABLE_ID
AND     SC.LOOKUP_CODE = FCR.STATUS_CODE
AND     PC.LOOKUP_CODE = FCR.PHASE_CODE;select prha.requisition_header_id,
PRLA.LINE_NUM,
PRLA.PURCHASE_BASIS,
         prha.preparer_id,
         PAPFPRE.FULL_NAME,
         lub.user_name LAST_UPDATED_BY,
         cb.user_name CREATED_BY,
         PRHA.ORG_ID,
         HOU.NAME ORGANIZATION_NAME,
         PRHA.TYPE_LOOKUP_CODE,
         PRHA.AUTHORIZATION_STATUS,
         PLC1.DESCRIPTION,
         PLC1.DISPLAYED_FIELD,
         PRHA.DESCRIPTION,
         PRHA.ENABLED_FLAG,
         PRHA.SUMMARY_FLAG,
         PRHA.SEGMENT1,
         PRLA.REQUISITION_LINE_ID,
         PRLA.CATEGORY_ID,
         MC.SEGMENT1 || ' ' || MC.SEGMENT2 CATEGORY_NAME,
         PRLA.QUANTITY,
         PRLA.UNIT_PRICE,
         PRLA.QUANTITY * PRLA.UNIT_PRICE AMOUNT,
         PRLA.TO_PERSON_ID,
         PAPF.FULL_NAME,
         MSIB.SEGMENT1 || ' ' || MSIB.DESCRIPTION ITEM_ID,
         PRLA.ITEM_DESCRIPTION,
         PRLA.NEED_BY_DATE,
         PRLA.DELIVER_TO_LOCATION_ID,
         HLA.ADDRESS_LINE_1,
         HLA.DESCRIPTION,
         HLA.POSTAL_CODE,
         PRLA.LINE_NUM,
         PRDA.DISTRIBUTION_ID,
         PRDA.REQUISITION_LINE_ID,
         PRDA.SET_OF_BOOKS_ID,
         PRDA.DISTRIBUTION_NUM,
         PRLA.REQUEST_ID,
         gcc.ACCOUNT_TYPE,
         gcc.ENABLED_FLAG,
         gcc.SUMMARY_FLAG,
         gcc.COMPANY_COST_CENTER_ORG_ID
from
        po_requisition_headers_all prha,
        po_requisition_lines_all prla,
        po_req_distributions_all prda,
        PER_ALL_PEOPLE_F PAPFPRE,
        fnd_user lub,
        fnd_user cb,
        HR_LOCATIONS_ALL HLA,
        PER_ALL_PEOPLE_F PAPF,
        HR_ORGANIZATION_UNITS HOU,
        MTL_CATEGORIES MC,
        MTL_SYSTEM_ITEMS_B MSIB,
        ORG_ORGANIZATION_DEFINITIONS OOD,
        GL_CODE_COMBINATIONS gcc,
        PO_LOOKUP_CODES PLC1
 --       PER_ALL_PEOPLE_F PAPFREQ
where
        1=1
and   prha.requisition_header_id = prla.requisition_header_id
and   prha.segment1 = '14438'
AND  PAPFPRE.PERSON_ID = PRHA.PREPARER_ID
and   prla.last_updated_by = lub.USER_id
and   prla.created_by = cb.USER_id
and   prda.requisition_line_id = prla.requisition_line_id
AND  PRLA.TO_PERSON_ID = PAPF.PERSON_ID
AND  HLA.LOCATION_ID = PRLA.DELIVER_TO_LOCATION_ID
AND  HOU.ORGANIZATION_ID = PRHA.ORG_ID
AND  PRHA.AUTHORIZATION_STATUS = PLC1.LOOKUP_CODE
AND  PLC1.LOOKUP_TYPE ='AUTHORIZATION STATUS'
AND  PRLA.CATEGORY_ID = MC.CATEGORY_ID
--AND  HLA.PERSON_ID = PRLA.DELIVER_TO_LOCATION_ID
AND  MSIB.INVENTORY_ITEM_ID = PRLA.ITEM_ID
AND  MSIB.ORGANIZATION_ID = OOD.ORGANIZATION_ID
AND  OOD.ORGANIZATION_ID = PRLA.ORG_ID
and   prda.code_combination_id = gcc.code_combination_id;
;



SELECT *
FROM PO_LOOKUP_CODES
WHERE LOOKUP_CODE = 'INCOMPLETE'
AND LOOKUP_TYPE ='AUTHORIZATION STATUS'
;

select *
from dba_objects
where object_name like 'HR%LOC%';

SELECT *
FROM HR_LOCATIONS_ALL
WHERE LOCATION_ID = 204;

select *
from  po_req_distributions_all;

select *
from gl_code_combinations;

select *
from  po_req_distributions_all prda,
        gl_code_combinations gcc
where prda.CODE_COMBINATION_ID = gcc.code_combination_id;

SELECT

    --prda.deliver_to_location_id,
    --prda.quantity,
    --prda.amount,
    --prda.gl_encumbrance_id,
    --prda.charge_account_id,
    prda.budget_account_id,
    --prda.status_code,
    gcc.code_combination_id,
    gcc.segment1 AS company_code,
    gcc.segment2 AS cost_center,
    gcc.segment3 AS account,
    gcc.segment4 AS sub_account,
    gcc.segment5 AS product_code,
    gcc.description AS gl_account_description
FROM
    po_req_distributions_all prda,
    gl_code_combinations gcc
WHERE
    prda.code_combination_id = gcc.code_combination_id;

SELECT *
FROM ORG_ORGANIZATION_DEFINITIONS;
ORG_ORGANISATION_DEFINATIONS
```

```
ORG_ORGANIZATION_DEFINATIONS
ORG_ORGANIZATION_DEFINITIONS

SELECT *
FROM TABS
WHERE TABLE_NAME LIKE '%ORG_%_OR%_D';

SELECT *
FROM MTL_SYSTEM_ITEMS_B;
select *
from
        po_requisition_headers_all prha,
        po_requisition_lines_all prla,
        po_req_distributions_all prda
where
        1=1
and   prha.requisition_header_id = prla.requisition_header_id
and   prha.segment1 = '14438'
and   prda.requisition_line_id = prla.requisition_line_id;

select *
from po_requisition_headers_all;

SELECT *
FROM MTL_CATEGORIES
WHERE CATEGORY_ID = 1;

select *
from po_requisition_lines_all;

select *
from po_req_distributions_all;

select *
from fnd_user
where user_id = 1318;

select *
from ap_suppliers
where vendor_id = 30163;

select org_id, hou.name
from ap_supplier_sites_all assa,
        hr_operating_units hou
where vendor_id = 30163
and    HOU.ORGANIZATION_ID = assa.org_id;

SELECT *
FROM MTL_SYSTEM_ITEMS_B;

GL_CODE_COMBINATIONS

hr_operating_units

select *
from ap_supplier_sites_all;



select assa.vendor_id, assa.vendor_site_code, ass.vendor_name ,count(assa.vendor_id)
from ap_supplier_sites_all assa, ap_suppliers ass
where
            assa.vendor_id = ass.vendor_id
group by assa.vendor_id, assa.vendor_site_code, ass.vendor_name
having count(assa.vendor_id) >1;

select * from po_requisition_lines_all;

--question : vendor all data from ap_supplier_sites_all, ap_supplier without using group
```

```
by
-- HOW MANY PARAMETER CAN BE ENTER IN CONCURRENT PROGRAM
-- FETCH PARAMETER WHICH IS CONNECTED WITH CONCURRENT PROGRAM FROM THE SQL QUERRY
-- PRINT PATAMETER WHICH IS USED IN CONCUREENT PROGRAM

SELECT REQUISITION_HEADER_IDLINE_NUMLINE_TYPE_IDCATEGORY_IDITEM_DESCRIPTIONUNIT_MEAS_LOOKUP_CODEUNIT_PRICEQUANTITYDELIVER_TO_LOCATION_IDTO_PERSON_IDLAST_UPDATED_BYCREATED_BYORG_IDREQUISITION_HEADER_IDREQUISITION_LINE_ID
FROM po_requisition_lines_all;


select prha.requisition_header_id
,prla.line_num
,prla.requisition_line_id
,mc.category_id
,papf.full_name
,mc.segment2
,prla.unit_meas_lookup_code
,prla.unit_price
,prla.quantity
,prla.unit_price*prla.quantity Amount
,prla.need_by_date
,msib.segment1
,msib.description
,ood.organization_code
,ood.organization_name
,prla.to_person_id
,hl.location_code
,plt.line_type
from po_requisition_lines_all prla
,po_requisition_headers_all prha
,mtl_categories mc
,per_all_people_f papf
,mtl_system_items_b msib
,org_organization_definitions ood
,hr_locations hl
,po_line_types plt
where prla.requisition_header_id=prha.requisition_header_id
and mc.category_id=prla.category_id
and msib.inventory_item_id=prla.item_id
and prla.to_person_id=papf.person_id
and ood.organization_id=msib.organization_id
and ood.organization_id = prla.org_id
and prla.deliver_to_location_id=hl.location_id
and prla.line_type_id=plt.line_type_id
and prha.segment1 = '14438'
and PRLA. -- ADD EXIST CODE
;

DESC FND_USER;
PER_ALL_PEOPLE_F
FND_FLEX_VALUE_SET

SELECT *
FROM MTL_SYSTEM_ITEMS_B
WHERE segment1 = 'P2P Cycle';

SELECT *
FROM PO_REQUISITION_LINES_ALL
WHERE ITEM_ID = 210959;



SELECT
    PRD.REQ_LINE_QUANTITY
    ,GCC.CHART_OF_ACCOUNTS_ID
    ,GCC.SEGMENT1||'-'||GCC.SEGMENT2||'-'||GCC.SEGMENT3||'-'||GCC.SEGMENT4||'-'||GCC.SEGM
    ENT5 ACCOUNT_DETAIL
FROM
    PO_REQ_DISTRIBUTIONS_ALL PRD
    ,GL_CODE_COMBINATIONS GCC
    ,PO_REQUISITION_LINES_ALL PRL
    ,PO_REQUISITION_HEADERS_ALL PRH
WHERE
    PRD.CODE_COMBINATION_ID=GCC.CODE_COMBINATION_ID
    AND PRL.REQUISITION_LINE_ID=PRD.REQUISITION_LINE_ID
    AND PRL.REQUISITION_HEADER_ID=PRH.REQUISITION_HEADER_ID
    AND PRH.SEGMENT1='14438';

    select prha.requisition_header_id,
         prha.preparer_id,
         PAPFPRE.FULL_NAME PREPARER_FULL_NAME,
         lub.user_name LAST_UPDATED_BY,
         cb.user_name CREATED_BY,
         PRHA.ORG_ID,
         HOU.NAME ORGANIZATION_NAME,
         PRHA.TYPE_LOOKUP_CODE,
         PRHA.AUTHORIZATION_STATUS AUTH_STATUS,
         PLC1.DESCRIPTION,
         PLC1.DISPLAYED_FIELD,
         PRHA.DESCRIPTION HEADER_DESC,
         PRHA.ENABLED_FLAG,
         PRHA.SUMMARY_FLAG,
         PRHA.SEGMENT1,
         PRLA.REQUISITION_LINE_ID,
         PRLA.CATEGORY_ID,
         PRLA.PURCHASE_BASIS,
         MC.SEGMENT1 || ' ' || MC.SEGMENT2 CATEGORY_NAME,
         PRLA.QUANTITY,
         PRLA.UNIT_PRICE,
         PRLA.QUANTITY * PRLA.UNIT_PRICE AMOUNT,
         PRLA.TO_PERSON_ID,
         PAPF.FULL_NAME REQ_PERSON,
         MSIB.SEGMENT1 || ' ' || MSIB.DESCRIPTION ITEM_ID,
         PRLA.ITEM_DESCRIPTION,
         PRLA.DESTINATION_TYPE_CODE,
         PRLA.NEED_BY_DATE,
         PRLA.DELIVER_TO_LOCATION_ID,
         HLA.ADDRESS_LINE_1,
         HLA.DESCRIPTION LOC_DESC,
         HLA.POSTAL_CODE,
         PRLA.LINE_NUM,
         PRDA.DISTRIBUTION_ID,
         PRDA.REQUISITION_LINE_ID,
         PRDA.SET_OF_BOOKS_ID,
         PRDA.DISTRIBUTION_NUM,
         PRLA.REQUEST_ID,
         gcc.ACCOUNT_TYPE,
         gcc.ENABLED_FLAG,
         gcc.SUMMARY_FLAG,
         gcc.COMPANY_COST_CENTER_ORG_ID,
         GCC.SEGMENT1||'-'||GCC.SEGMENT2||'-'||GCC.SEGMENT3||'-'||GCC.SEGMENT4||'-'||GCC.SEGMENT5 ACCOUNT_DETAIL
from
        po_requisition_headers_all prha,
        po_requisition_lines_all prla,
        po_req_distributions_all prda,
        PER_ALL_PEOPLE_F PAPFPRE,
        fnd_user lub,
        fnd_user cb,
        HR_LOCATIONS_ALL HLA,
        PER_ALL_PEOPLE_F PAPF,
        HR_ORGANIZATION_UNITS HOU,
        MTL_CATEGORIES MC,
        MTL_SYSTEM_ITEMS_B MSIB,
        ORG_ORGANIZATION_DEFINITIONS OOD,
        GL_CODE_COMBINATIONS gcc,
        PO_LOOKUP_CODES PLC1
 --       PER_ALL_PEOPLE_F PAPFREQ
where
        1=1
and   prha.requisition_header_id = prla.requisition_header_id
and   prha.segment1 = '14438'
AND  PAPFPRE.PERSON_ID = PRHA.PREPARER_ID
and   prla.last_updated_by = lub.USER_id
and   prla.created_by = cb.USER_id
and   prda.requisition_line_id = prla.requisition_line_id
AND  PRLA.TO_PERSON_ID = PAPF.PERSON_ID
AND  HLA.LOCATION_ID = PRLA.DELIVER_TO_LOCATION_ID
AND  HOU.ORGANIZATION_ID = PRHA.ORG_ID
AND  PRHA.AUTHORIZATION_STATUS = PLC1.LOOKUP_CODE
AND  PLC1.LOOKUP_TYPE ='AUTHORIZATION STATUS'
AND  PRLA.CATEGORY_ID = MC.CATEGORY_ID
--AND  HLA.PERSON_ID = PRLA.DELIVER_TO_LOCATION_ID
AND  MSIB.INVENTORY_ITEM_ID = PRLA.ITEM_ID
AND  MSIB.ORGANIZATION_ID = OOD.ORGANIZATION_ID
AND  OOD.ORGANIZATION_ID = PRLA.ORG_ID
and   prda.code_combination_id = gcc.code_combination_id;CREATE OR REPLACE PACKAGE BODY xxqgen_api_pkg_dk AS

    /***************************************************************************************************
     * Procedure: CREATE_USER
     * Description: Creates a new user using FND_USER_PKG API.
     ***************************************************************************************************/
    PROCEDURE create_user (
        p_user_name           IN VARCHAR2,
        p_unencrypted_password IN VARCHAR2
    ) IS
    BEGIN
        fnd_user_pkg.createuser(
            x_user_name                => p_user_name,
            x_owner                    => NULL,
            x_unencrypted_password     => p_unencrypted_password,
            x_session_number           => NULL,
            x_start_date               => NULL,
            x_end_date                 => NULL,
            x_last_logon_date          => NULL,
            x_description              => NULL,
            x_password_date            => NULL,
            x_password_accesses_left   => NULL,
            x_password_lifespan_accesses => NULL,
            x_password_lifespan_days   => NULL,
            x_employee_id              => NULL,
            x_email_address            => NULL,
            x_fax                      => NULL,
            x_customer_id              => NULL,
            x_supplier_id              => NULL
        );

        -- Debug message
        DBMS_OUTPUT.PUT_LINE('User created successfully.');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error creating user: ' || SQLCODE || ' - ' || SQLERRM);
    END create_user;

    /***************************************************************************************************
     * Procedure: ADD_TO_GROUP
     * Description: Adds a program to a request group.
     ****************************************************************************************************/
    PROCEDURE add_to_group (
        p_program_short_name  IN VARCHAR2,
        p_program_application IN VARCHAR2,
        p_request_group       IN VARCHAR2,
        p_group_application   IN VARCHAR2
    ) IS
    BEGIN
        apps.fnd_program.add_to_group(
            program_short_name  => p_program_short_name,
            program_application => p_program_application,
            request_group       => p_request_group,
            group_application   => p_group_application
        );
        DBMS_OUTPUT.PUT_LINE('Program added to group successfully.');
        COMMIT;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error adding program to group: ' || SQLCODE || ' - ' || SQLERRM);
    END add_to_group;


    /****************************************************************************************************
     * Procedure: SUBMIT_REQUEST
     * Description: Submits a concurrent request for a specific program.
     ****************************************************************************************************/
    PROCEDURE submit_request (
        p_application IN VARCHAR2,
        p_program     IN VARCHAR2,
        p_user_id     IN NUMBER,
        p_resp_id     IN NUMBER,
        p_resp_appl_id IN NUMBER
    ) IS
        l_request_id NUMBER;
    BEGIN
        apps.fnd_global.apps_initialize(p_user_id, p_resp_id, p_resp_appl_id);

        l_request_id := fnd_request.submit_request(
            application => p_application,
            program     => p_program
        );

        IF l_request_id = 0 THEN
            DBMS_OUTPUT.PUT_LINE('Request submission failed.');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Request submitted successfully. Request ID: ' || l_request_id);
        END IF;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error submitting request: ' || SQLCODE || ' - ' || SQLERRM);
    END submit_request;


    /****************************************************************************************************
     * Procedure: EXECUTABLE_EXIST
     * Description: Checks if an executable exists.
     ****************************************************************************************************/
    PROCEDURE executable_exist (
        p_executable_short_name IN VARCHAR2,
        p_application           IN VARCHAR2
    ) IS
    BEGIN
        IF apps.fnd_program.executable_exists(
            executable_short_name => p_executable_short_name,
            application           => p_application
        ) THEN
            DBMS_OUTPUT.PUT_LINE('Executable exists.');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Executable does not exist.');
        END IF;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error checking executable existence: ' || SQLCODE || ' - ' || SQLERRM);
    END executable_exist;


    /****************************************************************************************************
     * Procedure: PROGRAM_EXIST
     * Description: Checks if a program exists.
     ****************************************************************************************************/
    PROCEDURE program_exist (
        p_program IN VARCHAR2,
        p_application IN VARCHAR2
    ) IS
    BEGIN
        IF apps.fnd_program.program_exists(
            program     => p_program,
            application => p_application
        ) THEN
            DBMS_OUTPUT.PUT_LINE('Program exists.');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Program does not exist.');
        END IF;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error checking program existence: ' || SQLCODE || ' - ' || SQLERRM);
    END program_exist;


    /****************************************************************************************************
     * Procedure: REMOVE_FROM_GROUP
     * Description: Removes a program from a request group.
     ****************************************************************************************************/
    PROCEDURE remove_from_group (
        p_program_short_name  IN VARCHAR2,
        p_program_application IN VARCHAR2,
        p_request_group       IN VARCHAR2,
        p_group_application   IN VARCHAR2
    ) IS
    BEGIN
        apps.fnd_program.remove_from_group(
            program_short_name  => p_program_short_name,
            program_application => p_program_application,
            request_group       => p_request_group,
            group_application   => p_group_application
        );
        DBMS_OUTPUT.PUT_LINE('Program removed from group successfully.');
    EXCEPTION
```

```
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error removing program from group: ' || SQLCODE || ' - ' || SQLERRM);
    END remove_from_group;


    /***************************************************************************************************
     * Procedure: DELETE_EXECUTABLE
     * Description: Deletes an executable.
     **************************************************************************************************/
    PROCEDURE delete_executable (
        p_executable_short_name IN VARCHAR2,
        p_application           IN VARCHAR2
    ) IS
    BEGIN
        apps.fnd_program.delete_executable(
            executable_short_name => p_executable_short_name,
            application           => p_application
        );
        DBMS_OUTPUT.PUT_LINE('Executable successfully deleted.');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error deleting executable: ' || SQLCODE || ' - ' || SQLERRM);
    END delete_executable;


    /***************************************************************************************************
     * Procedure: DELETE_PROGRAM
     * Description: Deletes a program from the system.
     **************************************************************************************************/
    PROCEDURE delete_program (
        p_program_short_name IN VARCHAR2,
        p_application_code   IN VARCHAR2
    ) IS
    BEGIN
        apps.fnd_program.delete_program(
            program_short_name  => p_program_short_name,
            application => p_application_code
        );
        DBMS_OUTPUT.PUT_LINE('Program successfully deleted.');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error deleting program: ' || SQLCODE || ' - ' || SQLERRM);
    END delete_program;


    /***************************************************************************************************
     * Procedure: MAIN
     * Description: Main procedure.
     **************************************************************************************************/
    PROCEDURE main IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Executing main process...');
    END main;

END xxqgen_api_pkg_dk;
/
CREATE OR REPLACE PACKAGE BODY XXQGEN_INVOICE_AP_STATUS
```

```
IS
    FUNCTION beforereport (P_END_CREATION_DATE IN DATE, P_START_CREATION_DATE IN DATE,
    P_TRADING_PARTNER IN VARCHAR2) RETURN BOOLEAN
    AS
        CURSOR CUR_INVOICE_DATA IS
            SELECT
                inv1.vendor_id AS c_vendor_id,
                hp.party_name AS trading_partner,
                TO_CHAR(inv1.invoice_date, 'DD-MON-RRRR HH24:MI') AS invoice_date,
                b.batch_name AS batch_name,
                inv1.invoice_id AS c_invoice_id,
                inv1.invoice_num AS invoice_num,
                (SELECT DISTINCT poh.segment1 AS c_po_number
                   FROM po_headers poh,
                        po_distributions_all pda,
                        ap_invoice_distributions_all aida
                  WHERE poh.po_header_id = pda.po_header_id
                    AND pda.po_distribution_id = aida.po_distribution_id
                    AND aida.invoice_id = inv1.invoice_id) AS po_number,
                DECODE(inv1.invoice_currency_code,
                       'USD', inv1.invoice_amount,
                       inv1.base_amount) AS original_amount,
                DECODE(inv1.invoice_currency_code,
                       'USD', inv1.invoice_amount,
                       inv1.base_amount)
                - DECODE(inv1.payment_currency_code,
                         'USD', NVL(inv1.amount_paid, 0) + NVL(discount_amount_taken, 0),
                         ROUND((DECODE(inv1.payment_cross_rate_type,
                                       'EMU FIXED', 1 / inv1.payment_cross_rate,
                                       inv1.exchange_rate) * NVL(inv1.amount_paid, 0)),
                                       f.precision)
                         + ROUND((DECODE(inv1.payment_cross_rate_type,
                                         'EMU FIXED', 1 / inv1.payment_cross_rate,
                                         inv1.exchange_rate) *
                                         NVL(inv1.discount_amount_taken, 0)),
                                         f.precision)
                        ) AS amount_remaining,
                inv1.description AS c_description,
                apps.ap_invoices_pkg.get_approval_status(
                    inv1.invoice_id, inv1.invoice_amount,
                    inv1.payment_status_flag, inv1.invoice_type_lookup_code
                ) AS approval_status,
                aiaha.last_update_date AS due_date,
                apt.name AS payment_term,
                aiaha.response AS notes,
                aiaha.approver_name
            FROM
                hz_parties hp,
                ap_invoices_all inv1,
                ap_batches_all b,
                ap_payment_schedules_all s,
                ap_terms apt,
                ap_inv_aprvl_hist_all aiaha,
                fnd_currencies_vl f
            WHERE hp.party_id = inv1.party_id
              AND (hp.party_id = P_TRADING_PARTNER OR P_TRADING_PARTNER IS NULL)
              AND b.batch_id(+) = inv1.batch_id
              AND s.invoice_id(+) = inv1.invoice_id
              AND apt.term_id = inv1.terms_id
              AND aiaha.invoice_id(+) = inv1.invoice_id
              AND f.currency_code = inv1.invoice_currency_code
              AND f.currency_code = 'USD'
              AND TRUNC(inv1.creation_date) >= NVL(TRUNC(P_START_CREATION_DATE),
              TRUNC(inv1.creation_date))
              AND TRUNC(inv1.creation_date) <= NVL(TRUNC(P_END_CREATION_DATE),
              TRUNC(inv1.creation_date));

        TYPE CUR_INVOICE_DATA_TYPE IS TABLE OF CUR_INVOICE_DATA%ROWTYPE INDEX BY
        PLS_INTEGER;
        INV_TBL CUR_INVOICE_DATA_TYPE;

    BEGIN
        -- Free unused memory
        DBMS_SESSION.free_unused_user_memory;

        -- Open the cursor and fetch data in bulk
        OPEN CUR_INVOICE_DATA;
        LOOP
            FETCH CUR_INVOICE_DATA BULK COLLECT INTO INV_TBL LIMIT 100;

            -- Process the fetched data
            FORALL IDX IN 1 .. INV_TBL.COUNT
                INSERT INTO XXQGEN_PO_APPROV_STATUS_DK -- Replace with actual table name
                (VENDOR_ID,
                TRADING_PARTNER,
                INVOICE_DATE,
                BATCH_NAME,
                INVOICE_ID,
                INVOICE_NUM,
                PO_NUMBER,
                ORIGINAL_AMOUNT,
                 AMOUNT_REMAINING,
                 DESCRIPTION,
                 APPROVAL_STATUS,
                 DUE_DATE,
                 payment_term,
                 NOTES,
                 APPROVER_NAME)
                VALUES
                (INV_TBL(IDX).c_vendor_id,
                INV_TBL(IDX).trading_partner,
                INV_TBL(IDX).invoice_date,
                INV_TBL(IDX).batch_name,
                INV_TBL(IDX).c_invoice_id,
                INV_TBL(IDX).invoice_num,
                INV_TBL(IDX).po_number,
                INV_TBL(IDX).original_amount,
                INV_TBL(IDX).amount_remaining,
                INV_TBL(IDX).c_description,
                INV_TBL(IDX).approval_status,
                INV_TBL(IDX).due_date,
                 INV_TBL(IDX).payment_term,
                 INV_TBL(IDX).notes,
                 INV_TBL(IDX).approver_name);

            EXIT WHEN INV_TBL.COUNT = 0;
        END LOOP;

        -- Close the cursor
        CLOSE CUR_INVOICE_DATA;

        RETURN TRUE;

    EXCEPTION
        WHEN OTHERS THEN
            FND_FILE.PUT_LINE(FND_FILE.LOG, 'ERROR: ' || SQLCODE || ' : ' || SQLERRM);
            RETURN FALSE;
    END beforereport;

    FUNCTION afterreport (p_orgid IN NUMBER) RETURN BOOLEAN
    AS
    BEGIN
        -- Uncomment and specify the table name if truncation is needed
        -- EXECUTE IMMEDIATE 'TRUNCATE TABLE XXQGEN_PO_APPROV_STATUS_DK';
        RETURN TRUE;
    EXCEPTION
        WHEN OTHERS THEN
            FND_FILE.PUT_LINE(FND_FILE.LOG, 'ERROR in afterReport: ' || SQLERRM);
            RETURN FALSE;
    END afterreport;

END XXQGEN_INVOICE_AP_STATUS;
CREATE OR REPLACE PACKAGE BODY XXQGEN_PO_REQ_BULK_COLL_DK IS
    FUNCTION beforereport (
        P_REQ_TYPE          IN VARCHAR2,
        P_REQUESTER_NAME    IN VARCHAR2,
        P_LINE_NUM          IN VARCHAR2,
        P_REQ_HDR_ID        IN VARCHAR2,
        P_ORG_ID            IN VARCHAR2
    ) RETURN BOOLEAN
    AS
        -- Cursor for Q_MAIN
        CURSOR CUR_HEAD_DATA IS
            SELECT
                PRHA.REQUISITION_HEADER_ID HEADER_ID,
                PRHA.PREPARER_ID,
                PAPF.EMAIL_ADDRESS,
                PAPF.FULL_NAME,
                PRHA.SEGMENT1,
                PRHA.DESCRIPTION,
                PRHA.AUTHORIZATION_STATUS AUTH_STATUS,
                PRHA.TYPE_LOOKUP_CODE LOOKUP_CODE,
                PRHA.ORG_ID,
                HOU.NAME ORG_NAME
            FROM
                PO_REQUISITION_HEADERS_ALL PRHA,
                PER_ALL_PEOPLE_F PAPF,
                HR_ORGANIZATION_UNITS HOU,
                PO_DOCUMENT_TYPES_ALL_VL PDTAV
            WHERE
                PRHA.PREPARER_ID = PAPF.PERSON_ID
                AND PRHA.ORG_ID = HOU.ORGANIZATION_ID
                AND PDTAV.DOCUMENT_SUBTYPE = PRHA.TYPE_LOOKUP_CODE
                AND PRHA.ORG_ID = PDTAV.ORG_ID
                AND PDTAV.DOCUMENT_TYPE_CODE = 'REQUISITION'
                AND (PRHA.REQUISITION_HEADER_ID = P_REQ_HDR_ID OR P_REQ_HDR_ID IS NULL)
                AND (PRHA.ORG_ID = P_ORG_ID OR P_ORG_ID IS NULL)
                AND (PRHA.TYPE_LOOKUP_CODE = P_REQ_TYPE OR P_REQ_TYPE IS NULL)
                AND (PAPF.FULL_NAME = P_REQUESTER_NAME OR P_REQUESTER_NAME IS NULL);

        -- Cursor for Q_MAIN2
        CURSOR CUR_LINE_DATA (HEADER_ID IN NUMBER) IS
            SELECT
                PRLA.LINE_NUM,
                MSIB.SEGMENT1 NAME,
                PRLA.ITEM_DESCRIPTION DESCRIPTION,
                PRLA.ITEM_ID,
                PRLA.QUANTITY,
                PRLA.UNIT_PRICE,
                PRLA.QUANTITY * PRLA.UNIT_PRICE AMOUNT,
                GCC.SEGMENT1 || '-' || GCC.SEGMENT2 || '-' || GCC.SEGMENT3 || '-' ||
                GCC.SEGMENT4 AS CHARGE_ACCOUNT
            FROM
                PO_REQUISITION_LINES_ALL PRLA,
                PO_REQ_DISTRIBUTIONS_ALL PRDA,
                MTL_SYSTEM_ITEMS_B MSIB,
                GL_CODE_COMBINATIONS GCC
            WHERE
                PRDA.REQUISITION_LINE_ID = PRLA.REQUISITION_LINE_ID
                AND PRDA.CODE_COMBINATION_ID = GCC.CODE_COMBINATION_ID
                AND MSIB.INVENTORY_ITEM_ID = PRLA.ITEM_ID
                AND (PRLA.REQUISITION_HEADER_ID = HEADER_ID OR HEADER_ID IS NULL);

        -- Variable to store fetched data
        TYPE CUR_HEAD_DATA_TYPE IS TABLE OF CUR_HEAD_DATA%ROWTYPE INDEX BY PLS_INTEGER;
        HEAD_TBL        CUR_HEAD_DATA_TYPE;

        TYPE CUR_LINE_DATA_TYPE IS TABLE OF CUR_LINE_DATA%ROWTYPE INDEX BY PLS_INTEGER;
        LINE_TBL         CUR_LINE_DATA_TYPE;

    BEGIN
        -- Fetch data from Q_MAIN
        OPEN CUR_HEAD_DATA;
        LOOP
            FETCH CUR_HEAD_DATA BULK COLLECT INTO HEAD_TBL LIMIT 100;

            -- Exit when no more rows are found
            EXIT WHEN HEAD_TBL.COUNT = 0;

            FOR I IN 1 .. HEAD_TBL.COUNT LOOP
                -- Fetch line data for the current header
                OPEN CUR_LINE_DATA(HEAD_TBL(I).HEADER_ID);
                LOOP
                    FETCH CUR_LINE_DATA BULK COLLECT INTO LINE_TBL LIMIT 100;

                    -- Exit when no more rows are found
                    EXIT WHEN LINE_TBL.COUNT = 0;

                    FOR J IN 1 .. LINE_TBL.COUNT LOOP
                        -- Insert consolidated data into staging table
                        INSERT INTO XXQGEN_PO_REQ_DK (
                            HEADER_ID,
                            PREPARER_ID,
                            EMAIL_ADDRESS,
                            FULL_NAME,
                            SEGMENT1,
                            DESCRIPTION,
                            AUTH_STATUS,
                            LOOKUP_CODE,
                            ORG_ID,
                            ORG_NAME,
                            LINE_NUM,
                            ITEM_NAME,
                            ITEM_DESCRIPTION,
                            ITEM_ID,
                            QUANTITY,
                            UNIT_PRICE,
                            AMOUNT,
                            CHARGE_ACCOUNT
                        )
                        VALUES (
                            HEAD_TBL(I).HEADER_ID,
                            HEAD_TBL(I).PREPARER_ID,
                            HEAD_TBL(I).EMAIL_ADDRESS,
                            HEAD_TBL(I).FULL_NAME,
                            HEAD_TBL(I).SEGMENT1,
                            HEAD_TBL(I).DESCRIPTION,
                            HEAD_TBL(I).AUTH_STATUS,
                            HEAD_TBL(I).LOOKUP_CODE,
                            HEAD_TBL(I).ORG_ID,
                            HEAD_TBL(I).ORG_NAME,
                            LINE_TBL(J).LINE_NUM,
                            LINE_TBL(J).NAME,
                            LINE_TBL(J).DESCRIPTION,
                            LINE_TBL(J).ITEM_ID,
                            LINE_TBL(J).QUANTITY,
                            LINE_TBL(J).UNIT_PRICE,
                            LINE_TBL(J).AMOUNT,
                            LINE_TBL(J).CHARGE_ACCOUNT
                        );
                    END LOOP;
                END LOOP;
                CLOSE CUR_LINE_DATA;
            END LOOP;
        END LOOP;
        CLOSE CUR_HEAD_DATA;

        RETURN TRUE;
    EXCEPTION
        WHEN OTHERS THEN
            FND_FILE.PUT_LINE(FND_FILE.LOG, 'Error in beforeTrigger: ' || SQLCODE || ' - ' || SQLERRM);
            RETURN FALSE;
    END beforereport;

    FUNCTION afterreport
    RETURN BOOLEAN
    IS
    BEGIN
        RETURN TRUE;
    EXCEPTION
        WHEN OTHERS THEN
            RETURN FALSE;
    END afterreport;
END XXQGEN_PO_REQ_BULK_COLL_DK;
-- 1st Query
SELECT
    RQHA.SEGMENT1,
    PAPF.PERSON_ID PREPARER_ID,
    FU.USER_ID LAST_UPDATED_BY,
    FU.USER_NAME CREATED_BY,
    RQHA.DESCRIPTION,
    RQHA.TYPE_LOOKUP_CODE TYPEE,
    PLC.LOOKUP_CODE,
    HOU.ORGANIZATION_ID ORG_ID
FROM
    PO_REQUISITION_HEADERS_ALL RQHA,
    PER_ALL_PEOPLE_F PAPF,
    FND_USER FU,
    PO_DOCUMENT_TYPES_ALL_VL PDT,
    PO_LOOKUP_CODES PLC,
    HR_OPERATING_UNITS HOU
WHERE
    RQHA.PREPARER_ID=PAPF.PERSON_ID
   AND RQHA.LAST_UPDATED_BY = FU.USER_ID
   AND RQHA.AUTHORIZATION_STATUS =PLC. LOOKUP_CODE
   AND PLC. LOOKUP_TYPE ='AUTHORIZATION STATUS'
   AND RQHA.TYPE_LOOKUP_CODE=PDT.DOCUMENT_SUBTYPE
   AND RQHA.ORG_ID=PDT.ORG_ID
   AND PDT.DOCUMENT_TYPE_CODE = 'REQUISITION'
   AND RQHA.ORG_ID=HOU.ORGANIZATION_ID
   AND RQHA.SEGMENT1='14438'
;

-- 2nd query
select prha.requisition_header_id
,prla.line_num
,prla.requisition_line_id
,mc.category_id
,papf.full_name
,mc.segment2
,prla.unit_meas_lookup_code
,prla.unit_price
,prla.quantity
,prla.unit_price*prla.quantity Amount
,prla.need_by_date
,msib.segment1
,msib.description
,ood.organization_code
,ood.organization_name
,prla.to_person_id
,hl.location_code
,plt.line_type
from po_requisition_lines_all prla
,po_requisition_headers_all prha
```

```
,mtl_categories mc
,per_all_people_f papf
,mtl_system_items_b msib
,org_organization_definitions ood
,hr_locations hl
,po_line_types plt
where prla.requisition_header_id=prha.requisition_header_id
and mc.category_id=prla.category_id
and msib.inventory_item_id=prla.item_id
and prla.to_person_id=papf.person_id
and ood.organization_id=msib.organization_id
and ood.organization_id = prla.org_id
and prla.deliver_to_location_id=hl.location_id
and prla.line_type_id=plt.line_type_id
and prha.segment1 = '14438';

-- 3rd query
SELECT
    PRD.REQ_LINE_QUANTITY
    ,GCC.CHART_OF_ACCOUNTS_ID
    ,GCC.SEGMENT1||'-'||GCC.SEGMENT2||'-'||GCC.SEGMENT3||'-'||GCC.SEGMENT4||'-'||GCC.SEGMENT5 ACCOUNT_DETAIL
FROM
    PO_REQ_DISTRIBUTIONS_ALL PRD
    ,GL_CODE_COMBINATIONS GCC
    ,PO_REQUISITION_LINES_ALL PRL
    ,PO_REQUISITION_HEADERS_ALL PRH
WHERE
    PRD.CODE_COMBINATION_ID=GCC.CODE_COMBINATION_ID
    AND PRL.REQUISITION_LINE_ID=PRD.REQUISITION_LINE_ID
    AND PRL.REQUISITION_HEADER_ID=PRH.REQUISITION_HEADER_ID
    AND PRH.SEGMENT1='14438'
    ;

-- 4th Query
 select prha.requisition_header_id,
         prha.preparer_id,
         PAPFPRE.FULL_NAME PREPARER_FULL_NAME,
         lub.user_name LAST_UPDATED_BY,
         cb.user_name CREATED_BY,
         PRHA.ORG_ID,
         HOU.NAME ORGANIZATION_NAME,
         PRHA.TYPE_LOOKUP_CODE,
         PRHA.AUTHORIZATION_STATUS AUTH_STATUS,
         PLC1.DESCRIPTION,
         PLC1.DISPLAYED_FIELD,
         PRHA.DESCRIPTION HEADER_DESC,
         PRHA.ENABLED_FLAG,
         PRHA.SUMMARY_FLAG,
         PRHA.SEGMENT1,
         PRLA.REQUISITION_LINE_ID,
         PRLA.CATEGORY_ID,
         PRLA.PURCHASE_BASIS,
         MC.SEGMENT1 || ' ' || MC.SEGMENT2 CATEGORY_NAME,
         PRLA.QUANTITY,
         PRLA.UNIT_PRICE,
         PRLA.QUANTITY * PRLA.UNIT_PRICE AMOUNT,
         PRLA.TO_PERSON_ID,
         PAPF.FULL_NAME REQ_PERSON,
         MSIB.SEGMENT1 || ' ' || MSIB.DESCRIPTION ITEM_ID,
         PRLA.ITEM_DESCRIPTION,
         PRLA.DESTINATION_TYPE_CODE,
         PRLA.NEED_BY_DATE,
         PRLA.DELIVER_TO_LOCATION_ID,
         HLA.ADDRESS_LINE_1,
         HLA.DESCRIPTION LOC_DESC,
         HLA.POSTAL_CODE,
         PRLA.LINE_NUM,
         PRDA.DISTRIBUTION_ID,
         PRDA.REQUISITION_LINE_ID,
         PRDA.SET_OF_BOOKS_ID,
         PRDA.DISTRIBUTION_NUM,
         PRLA.REQUEST_ID,
         gcc.ACCOUNT_TYPE,
         gcc.ENABLED_FLAG,
         gcc.SUMMARY_FLAG,
         gcc.COMPANY_COST_CENTER_ORG_ID,
         GCC.SEGMENT1||'-'||GCC.SEGMENT2||'-'||GCC.SEGMENT3||'-'||GCC.SEGMENT4||'-'||GCC.SEGMENT5 ACCOUNT_DETAIL
from
        po_requisition_headers_all prha,
        po_requisition_lines_all prla,
        po_req_distributions_all prda,
        PER_ALL_PEOPLE_F PAPFPRE,
        fnd_user lub,
        fnd_user cb,
        HR_LOCATIONS_ALL HLA,
        PER_ALL_PEOPLE_F PAPF,
        HR_ORGANIZATION_UNITS HOU,
        MTL_CATEGORIES MC,
        MTL_SYSTEM_ITEMS_B MSIB,
        ORG_ORGANIZATION_DEFINITIONS OOD,
        GL_CODE_COMBINATIONS gcc,
        PO_LOOKUP_CODES PLC1
 --       PER_ALL_PEOPLE_F PAPFREQ
where
        1=1
and   prha.requisition_header_id = prla.requisition_header_id
and   prha.segment1 = '14438'
AND  PAPFPRE.PERSON_ID = PRHA.PREPARER_ID
and   prla.last_updated_by = lub.USER_id
and   prla.created_by = cb.USER_id
and   prda.requisition_line_id = prla.requisition_line_id
AND  PRLA.TO_PERSON_ID = PAPF.PERSON_ID
AND  HLA.LOCATION_ID = PRLA.DELIVER_TO_LOCATION_ID
AND  HOU.ORGANIZATION_ID = PRHA.ORG_ID
AND  PRHA.AUTHORIZATION_STATUS = PLC1.LOOKUP_CODE
AND  PLC1.LOOKUP_TYPE ='AUTHORIZATION STATUS'
AND  PRLA.CATEGORY_ID = MC.CATEGORY_ID
--AND  HLA.PERSON_ID = PRLA.DELIVER_TO_LOCATION_ID
AND  MSIB.INVENTORY_ITEM_ID = PRLA.ITEM_ID
AND  MSIB.ORGANIZATION_ID = OOD.ORGANIZATION_ID
AND  OOD.ORGANIZATION_ID = PRLA.ORG_ID
and   prda.code_combination_id = gcc.code_combination_id;
--1. Basic Query: Fetching Requisition Headers
--Retrieve requisition details like requisition number, description, and status.

SELECT
    prh.requisition_header_id,
    prh.segment1 AS requisition_number,
    prh.description,
    prh.authorization_status,
    prh.preparer_id,
    prh.creation_date
FROM
    po_requisition_headers_all prh
WHERE
    prh.authorization_status = 'APPROVED'; -- Filter for approved requisitions


--2. Intermediate Query: Requisition Lines with Items
--Fetch requisition lines along with item details.

SELECT
    prh.segment1 AS requisition_number,
    prh.description AS requisition_description,
```

```sql
    prl.line_num AS requisition_line_number,
    prl.item_description,
    prl.quantity,
    prl.unit_price,
    prl.need_by_date
FROM
    po_requisition_headers_all prh
JOIN
    po_requisition_lines_all prl
ON
    prh.requisition_header_id = prl.requisition_header_id
WHERE
    prh.authorization_status = 'APPROVED'
AND
    prl.item_id IS NOT NULL; -- Ensure the line is tied to an item

--3. Advanced Query: Requisition to Purchase Order Link
--Track requisitions converted to purchase orders.

SELECT
    prh.segment1 AS requisition_number,
    prh.description AS requisition_description,
    prl.line_num AS requisition_line_number,
    prl.quantity AS req_quantity,
    pol.po_header_id,
    poh.segment1 AS po_number,
    pol.quantity AS po_quantity,
    pol.unit_price AS po_unit_price,
    poh.authorization_status AS po_status
FROM
    po_requisition_headers_all prh
JOIN
    po_requisition_lines_all prl
ON
    prh.requisition_header_id = prl.requisition_header_id
JOIN
    po_req_distributions_all prd
ON
    prl.requisition_line_id = prd.requisition_line_id
JOIN
    po_distributions_all pd
ON
    prd.distribution_id = pd.req_distribution_id
JOIN
    po_lines_all pol
ON
    pd.po_line_id = pol.po_line_id
JOIN
    po_headers_all poh
ON
    pol.po_header_id = poh.po_header_id
WHERE
    prh.authorization_status = 'APPROVED'
AND
    poh.authorization_status IN ('APPROVED', 'IN PROCESS'); -- Filter for PO status

--4. Advanced Query: Requisition to Receiving and Invoices
--Track the full P2P flow from requisitions to receiving and invoices.

SELECT
    prh.segment1 AS requisition_number,
    prl.line_num AS requisition_line_number,
    prl.item_description,
    prl.quantity AS req_quantity,
    poh.segment1 AS po_number,
    pol.quantity AS po_quantity,
    rcv.transaction_quantity AS received_quantity,
    ap.invoice_num,
    ap.invoice_amount,
    ap.payment_status_flag
FROM
    po_requisition_headers_all prh
JOIN
    po_requisition_lines_all prl
ON
    prh.requisition_header_id = prl.requisition_header_id
JOIN
    po_req_distributions_all prd
ON
    prl.requisition_line_id = prd.requisition_line_id
JOIN
    po_distributions_all pd
ON
    prd.distribution_id = pd.req_distribution_id
JOIN
    po_lines_all pol
ON
    pd.po_line_id = pol.po_line_id
JOIN
    po_headers_all poh
ON
    pol.po_header_id = poh.po_header_id
LEFT JOIN
    rcv_transactions rcv
ON
    pol.po_line_id = rcv.po_line_id
LEFT JOIN
    ap_invoices_all ap
ON
    pd.po_distribution_id = ap.po_distribution_id
WHERE
    prh.authorization_status = 'APPROVED'
AND
    poh.authorization_status = 'APPROVED'
AND
    rcv.transaction_type IN ('RECEIVE', 'DELIVER')
AND
    ap.invoice_status = 'VALIDATED';
```

Explanation of Key Tables:

PO_REQUISITION_HEADERS_ALL: Stores requisition headers.
PO_REQUISITION_LINES_ALL: Stores requisition line details.
PO_HEADERS_ALL: Stores purchase order headers.
PO_LINES_ALL: Stores purchase order line details.
RCV_TRANSACTIONS: Stores receiving transactions.
AP_INVOICES_ALL: Stores invoice information.

Notes:

Adjust filters (authorization_status, transaction_type, etc.) based on your requirements.
Use appropriate date filters to limit the data set.
Always test these queries in a non-production environment before use.

You said:

more questions

ChatGPT said:

ChatGPT

Here are additional P2P requisition-related questions, ranging from basic to advanced, that you might encounter during an interview or while working on Oracle EBS:

Basic Questions

What is the P2P process?
What is a requisition, and why is it created in the P2P cycle?
What are the key tables involved in storing requisition details in Oracle EBS?
What is the difference between requisition types: Internal and Purchase Requisition?
What is the significance of the authorization_status column in the PO_REQUISITION_HEADERS_ALL table?

How is the preparer of a requisition identified in the database?
What is the purpose of the item_id in PO_REQUISITION_LINES_ALL?
Intermediate Questions
How do you identify approved requisitions in Oracle EBS?
How can you fetch the list of requisitions created within a specific date range?
How do you link requisitions to purchase orders in the database?
What are the different statuses a requisition can have, and what do they mean?
How do you differentiate between standard purchase orders and blanket purchase agreements in a query?
Explain the relationship between PO_REQUISITION_LINES_ALL and PO_REQ_DISTRIBUTIONS_ALL.
What is the significance of the org_id column in requisition-related tables?
Advanced Questions
How can you track a requisition’s progress from creation to invoice payment?
How would you write a query to find requisitions pending approval?
How do you identify requisitions that are fully received but not invoiced?
How can you optimize a query linking requisitions, purchase orders, receiving, and invoices?
What is the difference between RECEIVE and DELIVER transaction types in the receiving process?
How would you find out if a requisition line has multiple distributions?
Explain how financials (GL codes, cost centers, etc.) are tied to requisitions in the database.
How do you handle requisitions created with invalid items or suppliers?
Scenario-Based Questions
A requisition has been approved but isn’t showing up in the buyer’s worklist. What steps would you take to debug this?
A requisition is created, but the approval workflow fails. How would you identify the issue?
Some requisitions are stuck in a pre-approved status. How can you find and resolve these requisitions?
How do you identify requisitions that have not been converted to purchase orders?
A purchase order linked to a requisition is closed, but the requisition line is still open. How would you debug this?
Technical Questions
How do you enable debugging for a P2P workflow in Oracle EBS?
What are the APIs used to create and update requisitions programmatically?
How do you handle changes to a requisition after it’s been approved?
Explain the role of PO_REQAPPROVAL_INIT1 in requisition approval workflows.
What is the difference between po_requisition_headers_all and po_requisition_headers views?
How do you ensure data consistency between requisitions and purchase orders in a multi-org setup?
Analytical Questions
How do you create a report that lists all requisitions and their associated purchase orders, including their statuses?
How would you find the total spend by supplier based on requisitions?
How can you identify trends in requisition creation by department or preparer?
How would you identify high-value requisitions that require additional approvals?
How can you create a dashboard to monitor requisitions pending in various statuses?
Let me know if you'd like detailed queries or solutions to any of these questions!'

You said:
query questions on fnd user , people , concurrent program, request program etc all ebs operation
ChatGPT said:
ChatGPT
Here’s a comprehensive list of query questions related to Oracle EBS operations involving FND Users, People, Concurrent Programs, Request Programs, and more. These queries cover various functional areas and help understand EBS processes:

FND_USER Queries
Retrieve a list of all active users:

sql

Copy code

```
SELECT
    user_id,
    user_name,
    start_date,
    end_date,
    last_logon_date
FROM
    fnd_user
WHERE
    account_status = 'ACTIVE';
```

Find user details based on responsibilities assigned:

sql
Copy code

```
SELECT
    fu.user_name,
    fu.description AS user_description,
    fr.responsibility_name,
    frv.start_date,
    frv.end_date
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups_direct frv
ON
    fu.user_id = frv.user_id
JOIN
    fnd_responsibility_tl fr
ON
    frv.responsibility_id = fr.responsibility_id
WHERE
    frv.end_date IS NULL;
```

Track users who haven’t logged in for a long time:

sql
Copy code

```
SELECT
    user_name,
    last_logon_date,
    account_status
FROM
    fnd_user
WHERE
    last_logon_date < SYSDATE - 90; -- No login in the last 90 days
```

Identify locked user accounts:

sql
Copy code

```
SELECT
    user_name,
    start_date,
    end_date
FROM
    fnd_user
WHERE
    account_status = 'LOCKED';
```

HR People (PER_ALL_PEOPLE_F) Queries

Retrieve active employees and their user accounts:

sql
Copy code

```
SELECT
    paf.person_id,
    papf.full_name,
    fnd_user.user_name,
    papf.date_of_birth
FROM
    per_all_people_f papf
```

```
JOIN
    per_all_assignments_f paf
ON
    papf.person_id = paf.person_id
JOIN
    fnd_user
ON
    paf.person_id = fnd_user.employee_id
WHERE
    SYSDATE BETWEEN papf.effective_start_date AND papf.effective_end_date
AND
    paf.assignment_status_type IN ('ACTIVE');
```

Find employees who don’t have user accounts:

sql
Copy code

```
SELECT
    papf.person_id,
    papf.full_name
FROM
    per_all_people_f papf
WHERE
    papf.person_id NOT IN (
        SELECT
            employee_id
        FROM
            fnd_user
        WHERE
            employee_id IS NOT NULL
    );
```

Concurrent Programs Queries
Fetch details of all concurrent programs:

sql
Copy code

```
SELECT
    concurrent_program_id,
    concurrent_program_name,
    description,
    enabled_flag
FROM
    fnd_concurrent_programs;
```

Retrieve details of requests submitted for a specific program:

sql
Copy code;

```
SELECT
    fcr.request_id,
    fcr.concurrent_program_id,
    fcr.status_code,
    fcr.phase_code,
    fcr.request_date
FROM
    fnd_concurrent_requests fcr
JOIN
    fnd_concurrent_programs fcp
ON
    fcr.concurrent_program_id = fcp.concurrent_program_id
WHERE
    fcp.concurrent_program_name = 'YOUR_PROGRAM_NAME';
```

Identify programs with the most failures:

sql
Copy code

```
SELECT
    fcp.concurrent_program_name,
    COUNT(*) AS failure_count
FROM
    fnd_concurrent_requests fcr
```

```
JOIN
    fnd_concurrent_programs fcp
ON
    fcr.concurrent_program_id = fcp.concurrent_program_id
WHERE
    fcr.phase_code = 'COMPLETED'
AND
    fcr.status_code = 'ERROR'
GROUP BY
    fcp.concurrent_program_name
ORDER BY
    failure_count DESC;
```

Retrieve running requests:

sql

Copy code

```
SELECT
    request_id,
    concurrent_program_id,
    phase_code,
    status_code,
    logfile_name
FROM
    fnd_concurrent_requests
WHERE
    phase_code = 'RUNNING';
```

Request Sets and Programs Queries

Fetch details of request sets:

sql

Copy code

```
SELECT
    request_set_id,
    request_set_name,
    application_id
FROM
    fnd_request_sets;
```

List all requests in a request set:

sql

Copy code

```
SELECT
    frs.request_set_id,
    frs.request_set_name,
    frs.stage_id,
    fcp.concurrent_program_name
FROM
    fnd_request_set_stages frss
JOIN
    fnd_request_sets frs
ON
    frss.request_set_id = frs.request_set_id
JOIN
    fnd_concurrent_programs fcp
ON
    frss.concurrent_program_id = fcp.concurrent_program_id;
```

Security and Responsibility Queries

Retrieve responsibilities assigned to a user:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups frg
ON
```

```
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility_tl fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    fu.user_name = 'YOUR_USER_NAME';
```

Check if a responsibility is end-dated:

sql
Copy code

```
SELECT
    responsibility_name,
    start_date,
    end_date
FROM
    fnd_responsibility
WHERE
    responsibility_id = 'RESPONSIBILITY_ID';
```

Diagnostic and Utility Queries
Check invalid logins:

sql
Copy code

```
SELECT
    user_name,
    last_logon_date,
    failed_login_attempts
FROM
    fnd_user
WHERE
    failed_login_attempts > 0;
```

Audit changes to user accounts:

sql
Copy code

```
SELECT
    user_id,
    user_name,
    last_update_date,
    last_updated_by
FROM
    fnd_user
WHERE
    last_update_date > SYSDATE - 30; -- Changes in the last 30 days
```

Monitor long-running concurrent requests:

sql
Copy code

```
SELECT
    request_id,
    user_name,
    concurrent_program_name,
    requested_start_date,
    elapsed_time
FROM
    fnd_concurrent_requests
WHERE
    elapsed_time > 30; -- Requests running for more than 30 minutes
```

Fetch error messages for failed requests:

sql
Copy code

```
SELECT
    request_id,
    phase_code,
    status_code,
    log_file_name,
    oracle_process_id
```

```
FROM
    fnd_concurrent_requests
WHERE
    phase_code = 'COMPLETED'
AND
    status_code = 'ERROR';
```

Let me know which area you'd like to explore further or if you need explanations or more queries!'

You said:
more querry questions
ChatGPT said:
ChatGPT
Here’s an extended set of query questions for Oracle EBS that you can explore across different domains such as FND Users, Concurrent Programs, Responsibilities, and Security, along with some more advanced operational queries:

FND_USER and User-Related Queries
List users with expired accounts:

sql
Copy code

```
SELECT
    user_id,
    user_name,
    end_date
FROM
    fnd_user
WHERE
    end_date < SYSDATE;
```

Find the last logon details for a specific user:

sql
Copy code

```
SELECT
    user_name,
    last_logon_date
FROM
    fnd_user
WHERE
    user_name = 'YOUR_USER_NAME';
```

Identify all system administrators in the application:

sql
Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups_direct furg
ON
    fu.user_id = furg.user_id
JOIN
    fnd_responsibility fr
ON
    furg.responsibility_id = fr.responsibility_id
WHERE
    fr.responsibility_name LIKE '%System Administrator%';
```

Find users linked to terminated employees:

sql
Copy code

```
SELECT
    fu.user_name,
    fu.employee_id
FROM
    fnd_user fu
JOIN
    per_all_people_f papf
ON
    fu.employee_id = papf.person_id
WHERE
    papf.effective_end_date < SYSDATE;
```

Responsibilities and Roles Queries

List responsibilities not assigned to any user:

sql

Copy code

```
SELECT
    responsibility_name
FROM
    fnd_responsibility
WHERE
    responsibility_id NOT IN (
        SELECT
            DISTINCT responsibility_id
        FROM
            fnd_user_resp_groups_direct
    );
```

Fetch a user’s responsibilities with effective dates:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    frg.start_date,
    frg.end_date
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups frg
ON
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    fu.user_name = 'YOUR_USER_NAME';
```

Check if a specific responsibility is available in a given application:

sql

Copy code

```
SELECT
    responsibility_name
FROM
    fnd_responsibility
WHERE
    application_id = 'APPLICATION_ID'
AND
    responsibility_name = 'RESPONSIBILITY_NAME';
```

Concurrent Programs Queries

Fetch all disabled concurrent programs:

sql

Copy code

```
SELECT
    concurrent_program_name,
    description
FROM
```

```
    fnd_concurrent_programs
WHERE
    enabled_flag = 'N';
Find completed concurrent requests with warnings:

sql
Copy code
SELECT
    request_id,
    phase_code,
    status_code
FROM
    fnd_concurrent_requests
WHERE
    phase_code = 'COMPLETED'
AND
    status_code = 'WARNING';
Get the execution time of a specific request:

sql
Copy code
SELECT
    request_id,
    requested_start_date,
    actual_start_date,
    actual_completion_date,
    (actual_completion_date - actual_start_date) * 24 * 60 AS execution_time_minutes
FROM
    fnd_concurrent_requests
WHERE
    request_id = 'REQUEST_ID';
Retrieve logs of failed concurrent requests:

sql
Copy code
SELECT
    request_id,
    logfile_name,
    logfile_node
FROM
    fnd_concurrent_requests
WHERE
    status_code = 'ERROR';
People (PER_ALL_PEOPLE_F) Queries
Find all employees under a specific manager:

sql
Copy code
SELECT
    employee_id,
    full_name,
    assignment_id
FROM
    per_all_assignments_f
WHERE
    manager_id = 'MANAGER_PERSON_ID';
List employees in a specific department:

sql
Copy code
SELECT
    papf.person_id,
    papf.full_name,
    per.organization_id,
    hrorg.name AS department_name
FROM
    per_all_people_f papf
JOIN
    per_all_assignments_f per
```

```
ON
    papf.person_id = per.person_id
JOIN
    hr_all_organization_units hrorg
ON
    per.organization_id = hrorg.organization_id
WHERE
    hrorg.name = 'DEPARTMENT_NAME';
```

Get details of all terminated employees:

sql
Copy code

```
SELECT
    papf.person_id,
    papf.full_name,
    paf.effective_end_date
FROM
    per_all_people_f papf
JOIN
    per_all_assignments_f paf
ON
    papf.person_id = paf.person_id
WHERE
    paf.assignment_status_type = 'TERMINATED';
```

System Administration Queries
Find top N requests consuming the most runtime:

sql
Copy code

```
SELECT
    request_id,
    concurrent_program_id,
    elapsed_time,
    requested_start_date
FROM
    fnd_concurrent_requests
ORDER BY
    elapsed_time DESC
FETCH FIRST 10 ROWS ONLY;
```

List all custom concurrent programs in the system:

sql
Copy code

```
SELECT
    concurrent_program_name,
    description
FROM
    fnd_concurrent_programs
WHERE
    concurrent_program_name LIKE 'XX%';
```

Find all requests submitted by a specific user:

sql
Copy code

```
SELECT
    fcr.request_id,
    fcr.concurrent_program_id,
    fcr.phase_code,
    fcr.status_code,
    fcr.request_date
FROM
    fnd_concurrent_requests fcr
JOIN
    fnd_user fu
ON
    fcr.requested_by = fu.user_id
WHERE
    fu.user_name = 'YOUR_USER_NAME';
```

Audit which users have added or updated concurrent programs:

sql
Copy code

```
SELECT
    concurrent_program_id,
    concurrent_program_name,
    last_updated_by,
    last_update_date
FROM
    fnd_concurrent_programs
WHERE
    last_update_date > SYSDATE - 30; -- Programs updated in the last 30 days
```

Diagnostic and Validation Queries

Check users with invalid configurations:

sql
Copy code

```
SELECT
    fu.user_name,
    fu.account_status
FROM
    fnd_user fu
WHERE
    NVL(fu.end_date, SYSDATE) < SYSDATE
OR
    NVL(fu.start_date, SYSDATE) > SYSDATE;
```

Validate if a concurrent program is associated with any request group:

sql
Copy code

```
SELECT
    fcp.concurrent_program_name,
    frg.request_group_name
FROM
    fnd_request_group_units frgu
JOIN
    fnd_request_groups frg
ON
    frgu.request_group_id = frg.request_group_id
JOIN
    fnd_concurrent_programs fcp
ON
    frgu.request_unit_id = fcp.concurrent_program_id
WHERE
    fcp.concurrent_program_name = 'YOUR_PROGRAM_NAME';
```

Check for unused responsibilities:

sql
Copy code

```
SELECT
    responsibility_id,
    responsibility_name
FROM
    fnd_responsibility
WHERE
    responsibility_id NOT IN (
        SELECT
            DISTINCT responsibility_id
        FROM
            fnd_user_resp_groups_direct
    );
```

Would you like additional questions on any specific area? Or do you need explanations or step-by-step solutions? Let me know!

You said:
more querry questions
ChatGPT said:
ChatGPT
Here are more query questions for Oracle EBS focusing on FND User, Concurrent Programs, Responsibilities, Requests, Data Security, and other operational aspects:

FND User and Security Queries
Find all inactive users:

sql
Copy code
```
SELECT
    user_name,
    start_date,
    end_date
FROM
    fnd_user
WHERE
    NVL(end_date, SYSDATE) < SYSDATE;
```
Check if a specific user has logged in within the past 30 days:

sql
Copy code
```
SELECT
    user_name,
    last_logon_date
FROM
    fnd_user
WHERE
    user_name = 'USERNAME'
AND
    last_logon_date >= SYSDATE - 30;
```
List all users with SYSADMIN privileges:

sql
Copy code
```
SELECT
    fu.user_name,
    fr.responsibility_name
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups_direct furg
ON
    fu.user_id = furg.user_id
JOIN
    fnd_responsibility fr
ON
    furg.responsibility_id = fr.responsibility_id
WHERE
    fr.responsibility_name LIKE '%SYSADMIN%';
```
Audit changes to user details in the last 7 days:

sql
Copy code
```
SELECT
    user_name,
    last_update_date,
    last_updated_by
FROM
    fnd_user
WHERE
    last_update_date >= SYSDATE - 7;
```
List users linked to multiple employees:

sql
Copy code
```
SELECT
```

```
    user_name,
    employee_id
FROM
    fnd_user
GROUP BY
    user_name,
    employee_id
HAVING
    COUNT(employee_id) > 1;
```

Responsibilities and Role Management

Get responsibilities assigned to users for a specific application:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    frg.start_date,
    frg.end_date
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups frg
ON
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    fr.application_id = 'APPLICATION_ID';
```

Check users with responsibilities that have expired:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    frg.end_date
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups frg
ON
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    frg.end_date < SYSDATE;
```

Find responsibilities assigned to multiple users:

sql

Copy code

```
SELECT
    responsibility_name,
    COUNT(user_id) AS user_count
FROM
    fnd_user_resp_groups
GROUP BY
    responsibility_name
HAVING
    COUNT(user_id) > 1;
```

Concurrent Programs and Requests

List all concurrent programs created or modified in the last 6 months:

sql

Copy code

```sql
SELECT
    concurrent_program_name,
    description,
    last_update_date
FROM
    fnd_concurrent_programs
WHERE
    last_update_date >= SYSDATE - 180;
```

Retrieve all scheduled concurrent requests for a specific user:

sql
Copy code

```sql
SELECT
    request_id,
    phase_code,
    status_code,
    requested_start_date
FROM
    fnd_concurrent_requests
WHERE
    requested_by = (SELECT user_id FROM fnd_user WHERE user_name = 'USERNAME');
```

Identify requests that failed due to errors:

sql
Copy code

```sql
SELECT
    request_id,
    phase_code,
    status_code,
    log_file_name
FROM
    fnd_concurrent_requests
WHERE
    phase_code = 'COMPLETED'
AND
    status_code = 'ERROR';
```

Audit request groups containing custom concurrent programs:

sql
Copy code

```sql
SELECT
    frg.request_group_name,
    fcp.concurrent_program_name
FROM
    fnd_request_group_units frgu
JOIN
    fnd_request_groups frg
ON
    frgu.request_group_id = frg.request_group_id
JOIN
    fnd_concurrent_programs fcp
ON
    frgu.request_unit_id = fcp.concurrent_program_id
WHERE
    fcp.concurrent_program_name LIKE 'XX%';
```

Retrieve historical runtime for a specific program:

sql
Copy code

```sql
SELECT
    request_id,
    actual_start_date,
    actual_completion_date,
    (actual_completion_date - actual_start_date) * 24 * 60 AS runtime_minutes
FROM
    fnd_concurrent_requests
WHERE
    concurrent_program_id = (SELECT concurrent_program_id FROM fnd_concurrent_programs
    WHERE concurrent_program_name = 'PROGRAM_NAME');
```

Data Security and Auditing

List security profiles associated with a specific responsibility:

sql

Copy code

```
SELECT
    fr.responsibility_name,
    fp.security_profile_name
FROM
    fnd_responsibility fr
JOIN
    fnd_profile_options fpo
ON
    fr.responsibility_id = fpo.level_id
JOIN
    fnd_security_profiles fp
ON
    fpo.profile_option_value = fp.security_profile_id
WHERE
    fr.responsibility_name = 'RESPONSIBILITY_NAME';
```

Find all profile options and their values for a specific user:

sql

Copy code

```
SELECT
    fpo.profile_option_name,
    fu.user_name,
    fpor.profile_option_value
FROM
    fnd_profile_options fpo
JOIN
    fnd_profile_option_responsibilities fpor
ON
    fpo.profile_option_id = fpor.profile_option_id
JOIN
    fnd_user fu
ON
    fpor.level_value = fu.user_id
WHERE
    fu.user_name = 'USERNAME';
```

Track changes to responsibilities for a specific user:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    furg.creation_date,
    furg.last_update_date
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups_direct furg
ON
    fu.user_id = furg.user_id
JOIN
    fnd_responsibility fr
ON
    furg.responsibility_id = fr.responsibility_id
WHERE
    fu.user_name = 'USERNAME';
```

People and Assignment Queries

Fetch active employees with assigned responsibilities:

sql

Copy code

```
SELECT
    papf.person_id,
    papf.full_name,
    fr.responsibility_name
FROM
    per_all_people_f papf
JOIN
    fnd_user fu
ON
    papf.person_id = fu.employee_id
JOIN
    fnd_user_resp_groups frg
ON
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    papf.effective_end_date IS NULL;
```

List terminated employees still linked to active user accounts:

sql
Copy code

```
SELECT
    fu.user_name,
    papf.full_name,
    papf.effective_end_date
FROM
    fnd_user fu
JOIN
    per_all_people_f papf
ON
    fu.employee_id = papf.person_id
WHERE
    papf.effective_end_date < SYSDATE
AND
    NVL(fu.end_date, SYSDATE) > SYSDATE;
```

Would you like more advanced queries, or help with any of these?

You said:
yes more advance querry questions
ChatGPT said:
ChatGPT
Here are advanced query questions related to Oracle EBS for various operations, diving deeper into real-world use cases:

Advanced FND User Queries
Identify users with overlapping responsibilities:

sql
Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    COUNT(*)
FROM
    fnd_user_resp_groups frg
JOIN
    fnd_user fu
ON
    frg.user_id = fu.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
```

```
    frg.start_date <= NVL(frg.end_date, SYSDATE)
GROUP BY
    fu.user_name,
    fr.responsibility_name
HAVING
    COUNT(*) > 1;
```

Track inactive users assigned to critical responsibilities:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    frg.start_date,
    frg.end_date
FROM
    fnd_user fu
JOIN
    fnd_user_resp_groups frg
ON
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    fu.end_date IS NOT NULL
AND
    fr.responsibility_name IN ('SYSADMIN', 'SECURITY ADMINISTRATOR');
```

Fetch user account lock/unlock activity logs:

sql

Copy code

```
SELECT
    fu.user_name,
    fl.message_text,
    fl.creation_date
FROM
    fnd_logins fl
JOIN
    fnd_user fu
ON
    fl.user_id = fu.user_id
WHERE
    fl.message_text LIKE '%Account%Locked%'
OR
    fl.message_text LIKE '%Account%Unlocked%';
```

Concurrent Program and Request Queries

Fetch the top 10 longest-running concurrent requests:

sql

Copy code

```
SELECT
    request_id,
    concurrent_program_id,
    actual_start_date,
    actual_completion_date,
    ROUND((actual_completion_date - actual_start_date) * 1440, 2) AS runtime_minutes
FROM
    fnd_concurrent_requests
WHERE
    phase_code = 'C'
AND
    status_code = 'C'
ORDER BY
    runtime_minutes DESC
FETCH FIRST 10 ROWS ONLY;
```

List concurrent programs without request groups:

sql
Copy code

```
SELECT
    concurrent_program_name
FROM
    fnd_concurrent_programs fcp
WHERE
    NOT EXISTS (
        SELECT
            1
        FROM
            fnd_request_group_units frgu
        WHERE
            frgu.request_unit_id = fcp.concurrent_program_id
    );
```

Identify failed requests for a specific program in the last 30 days:

sql
Copy code

```
SELECT
    request_id,
    log_file_name,
    phase_code,
    status_code
FROM
    fnd_concurrent_requests
WHERE
    concurrent_program_id = (SELECT concurrent_program_id FROM fnd_concurrent_programs WHERE concurrent_program_name = 'PROGRAM_NAME')
AND
    phase_code = 'C'
AND
    status_code = 'E'
AND
    requested_start_date >= SYSDATE - 30;
```

Retrieve concurrent programs with a specific parameter value:

sql
Copy code

```
SELECT
    fcp.concurrent_program_name,
    fcp.description,
    fcr.argument_text
FROM
    fnd_concurrent_requests fcr
JOIN
    fnd_concurrent_programs fcp
ON
    fcr.concurrent_program_id = fcp.concurrent_program_id
WHERE
    fcr.argument_text LIKE '%VALUE%';
```

Responsibility and Security

Find responsibilities without any users assigned:

sql
Copy code

```
SELECT
    responsibility_name
FROM
    fnd_responsibility
WHERE
    NOT EXISTS (
        SELECT
            1
        FROM
            fnd_user_resp_groups_direct
        WHERE
            fnd_user_resp_groups_direct.responsibility_id = fnd_responsibility.responsibility_id
    );
```

Audit changes to responsibility assignments in the last 90 days:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    frg.creation_date,
    frg.last_update_date
FROM
    fnd_user_resp_groups_direct frg
JOIN
    fnd_user fu
ON
    frg.user_id = fu.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    frg.last_update_date >= SYSDATE - 90;
```

Find users with expired responsibilities still active:

sql

Copy code

```
SELECT
    fu.user_name,
    fr.responsibility_name,
    frg.end_date
FROM
    fnd_user_resp_groups_direct frg
JOIN
    fnd_user fu
ON
    frg.user_id = fu.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    NVL(frg.end_date, SYSDATE) < SYSDATE
AND
    NVL(fu.end_date, SYSDATE) > SYSDATE;
```

Employee and Person Queries

Fetch users without employees linked:

sql

Copy code

```
SELECT
    user_name
FROM
    fnd_user
WHERE
    employee_id IS NULL;
```

Identify terminated employees still associated with active responsibilities:

sql

Copy code

```
SELECT
    papf.full_name,
    papf.person_id,
    fu.user_name,
    fr.responsibility_name
FROM
    per_all_people_f papf
JOIN
    fnd_user fu
ON
```

```
    papf.person_id = fu.employee_id
JOIN
    fnd_user_resp_groups frg
ON
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
WHERE
    papf.effective_end_date < SYSDATE
AND
    NVL(fu.end_date, SYSDATE) > SYSDATE;
```

Fetch employees with roles across multiple responsibilities:

sql
Copy code

```
SELECT
    papf.person_id,
    papf.full_name,
    fr.responsibility_name,
    COUNT(*) AS responsibility_count
FROM
    per_all_people_f papf
JOIN
    fnd_user fu
ON
    papf.person_id = fu.employee_id
JOIN
    fnd_user_resp_groups frg
ON
    fu.user_id = frg.user_id
JOIN
    fnd_responsibility fr
ON
    frg.responsibility_id = fr.responsibility_id
GROUP BY
    papf.person_id,
    papf.full_name,
    fr.responsibility_name
HAVING
    COUNT(*) > 1;
```

Advanced Logging and Debugging

Retrieve failed concurrent requests with log file details:

sql
Copy code

```
SELECT
    fcr.request_id,
    fcp.concurrent_program_name,
    fcr.log_file_name,
    fcr.phase_code,
    fcr.status_code
FROM
    fnd_concurrent_requests fcr
JOIN
    fnd_concurrent_programs fcp
ON
    fcr.concurrent_program_id = fcp.concurrent_program_id
WHERE
    fcr.phase_code = 'C'
AND
    fcr.status_code = 'E';
```

Track changes to system profile values in the past 30 days:

sql
Copy code

```
SELECT
    fpo.profile_option_name,
    fps.level_value,
    fps.profile_option_value,
    fps.last_update_date,
    fps.last_updated_by
FROM
    fnd_profile_option_values fps
JOIN
    fnd_profile_options fpo
ON
    fps.profile_option_id = fpo.profile_option_id
WHERE
    fps.last_update_date >= SYSDATE - 30;/* Formatted on 12/18/2024 12:07:52 PM (QP5 v5.163.1008.3004) */
SELECT *
FROM fnd_user
here;

SELECT *
  FROM fnd_user
 WHERE user_name = 'OPERATIONS';

SELECT *
  FROM fnd_user
 WHERE user_id = 1318;

--1006
SELECT * FROM PER_ALL_PEOPLE_F
WHERE PERSON_ID = 10061;

SELECT SYSDATE FROM DUAL;


SELECT * FROM operations;

SELECT *
  FROM fnd_user
 WHERE user_name = 'ADITYA'
       AND TRIM (SYSDATE) BETWEEN start_date AND end_date;

SELECT *
  FROM fnd_user
 WHERE user_name = 'OPERATIONS'
       AND TRIM (SYSDATE) BETWEEN start_date AND NVL (end_date, SYSDATE + 1);

SELECT *
  FROM PER_ALL_PEOPLE_F;

 --WHERE PERSON_ID = 25;

SELECT * FROM per_all_people_f;

SELECT *
  FROM PER_ALL_PEOPLE_F
 WHERE PERSON_ID = 25
       AND TRUNC (SYSDATE) BETWEEN EFFECTIVE_START_DATE
                               AND EFFECTIVE_END_DATE
       AND (CURRENT_EMPLOYEE_FLAG = 'Y' OR CURRENT_NPW_FLAG = 'Y');


/****************************************
       *
       *****************************************/

SELECT
       fu.user_id,
       fu.user_name user_name,
       fu.start_date,
       fu.end_date,
       fu.description,
       papf.person_id,
       NVL (papf.EMPLOYEE_NUMBER, papf.NPW_NUMBER) employee_id,
       papf.full_name,
       papf.EFFECTIVE_START_DATE,
       papf.EFFECTIVE_END_DATE,
       papf.CURRENT_EMPLOYEE_FLAG,
       papf.CURRENT_NPW_FLAG,
       papf.EMAIL_ADDRESS
  FROM fnd_user fu, per_all_people_f papf
 WHERE
        fu.employee_id = papf.person_id
       AND user_name = 'ADITYA'
       AND TRIM (SYSDATE) BETWEEN fu.start_date
                              AND NVL (fu.end_date, SYSDATE + 1)
       AND TRIM (SYSDATE) BETWEEN papf.EFFECTIVE_START_DATE
                              AND papf.EFFECTIVE_END_DATE
       AND (papf.CURRENT_EMPLOYEE_FLAG = 'Y' OR papf.CURRENT_NPW_FLAG = 'Y');



  SELECT START_DATE,
         END_DATE,
         RESPONSIBILITY_APPLICATION_ID,
         RESPONSIBILITY_ID,
         SECURITY_GROUP_ID,
         LAST_UPDATE_DATE,
         LAST_UPDATED_BY,
         CREATED_BY,
         CREATION_DATE,
         LAST_UPDATE_LOGIN,
         USER_ID
    FROM FND_USER_RESP_GROUPS_direct
  WHERE (responsibility_id, responsibility_application_id) IN
            (SELECT responsibility_id, application_id
               FROM fnd_responsibility
              WHERE USER_ID = 1014243)
ORDER BY responsibility_application_id, responsibility_id, security_group_id;


--*************************



  SELECT START_DATE,
         END_DATE,
         RESPONSIBILITY_APPLICATION_ID,
         RESPONSIBILITY_ID,
         SECURITY_GROUP_ID,
         LAST_UPDATE_DATE,
         LAST_UPDATED_BY,
         CREATED_BY,
         CREATION_DATE,
         LAST_UPDATE_LOGIN,
         USER_ID
    FROM FND_USER_RESP_GROUPS_direct
   WHERE (responsibility_id, responsibility_application_id) IN
            (SELECT responsibility_id, application_id
               FROM fnd_responsibility
              WHERE (   version = '4'
                     OR version = 'W'
                     OR version = 'M'
                     OR version = 'H'))
       --  AND (USER_ID = 1318)
ORDER BY responsibility_application_id, responsibility_id, security_group_id;

select * from dba_objects where object_type LIKE '%TABLE%';


SELECT responsibility_id, application_id, version
```

```
               FROM fnd_responsibility
              WHERE (   version = '4'
                     OR version = 'W'
                     OR version = 'M'
                     OR version = 'H');


select * from dba_objects where object_type LIKE '%TABLE%';

select * from dba_objects where object_NAME LIKE '%FND%RES%VL%'; /*for  language*/

select * from dba_objects where object_NAME LIKE '%FND%RES%TL%';

select * from dba_objects where object_NAME LIKE '%FND%APP%VL%';

select * from dba_objects where object_NAME LIKE '%FND%USER%RESP%';

select *
from FND_USER_RESP_GROUPS;

SELECT * FROM FND_RESPONSIBILITY;
SELECT * FROM FND_USER_RESP_GROUPS_direct;
--where user_id = 1014254
--order by last_update_date desc;
SELECT * FROM FND_RESPONSIBILITY_VL;
SELECT * FROM FND_APPLICATION_VL;
select *
from fnd_user
where user_name = 'DEV';

fnd_user_resp_groups_api

xxqgen_user_data_pkg_ ;

--FND_APPLICATION_VL
select * from fnd_responsibility;

SELECT
          fu.user_id,
       fu.user_name user_name,
       fu.start_date,
       fu.end_date,
       fu.description,
       papf.person_id,
       NVL (papf.EMPLOYEE_NUMBER, papf.NPW_NUMBER) employee_id,
       papf.full_name,
       papf.EFFECTIVE_START_DATE,
       papf.EFFECTIVE_END_DATE,
       papf.CURRENT_EMPLOYEE_FLAG,
       papf.CURRENT_NPW_FLAG,
       papf.EMAIL_ADDRESS,
       frv.RESPONSIBILITY_NAME,
       fav.APPLICATION_NAME,
       frv.RESPONSIBILITY_KEY,
       frv.DESCRIPTION,
       frv.START_DATE
  FROM
       fnd_user fu,
       per_all_people_f papf,
       FND_USER_RESP_GROUPS_direct furg,
       FND_RESPONSIBILITY_VL frv,
       FND_APPLICATION_VL fav
       where
       furg.responsibility_id = frv.RESPONSIBILITY_ID
       and fu.user_id=furg.user_id
       and frv.APPLICATION_ID=fav.APPLICATION_ID
       and  fu.employee_id = papf.person_id
   --    and user_name = 'DEV'
       AND TRIM (SYSDATE) BETWEEN fu.start_date
```

```sql
                              AND NVL (fu.end_date, SYSDATE + 1)
       AND TRIM (SYSDATE) BETWEEN papf.EFFECTIVE_START_DATE
                              AND papf.EFFECTIVE_END_DATE
       AND (papf.CURRENT_EMPLOYEE_FLAG = 'Y' OR papf.CURRENT_NPW_FLAG = 'Y');

       ;

select * from fnd_user;

select * from dba_objects where object_name like '%fnd%';

select * from all_objects where object_name like '%fnd%';

select *
from fnd_user_resp_groups;

select *
from  FND_RESPONSIBILITY_VL frv;

select *
from fnd_concurrent_program_vl;

SELECT *
FROM FND_APPLICATION_VL
WHERE APPLICATION_ID = 201;

SELECT * FROM FND_CONCURRENT_PROGRAMS_VL
WHERE USER_CONCURRENT_PROGRAM_NAME like 'XXQGEN%DK';

SELECT * FROM FND_EXECUTABLES_FORM_V
WHERE EXECUTABLE_ID = 16642;


SELECT * FROM FND_CONCURRENT_REQUESTS
WHERE 1=1;
-- AND REQUEST_ID =5912035 -- 5912104
--AND CONCURRENT_PROGRAM_ID = 218704;


-- PHASE
SELECT LOOKUP_CODE, MEANING
  FROM FND_LOOKUP_VALUES
WHERE LOOKUP_TYPE = 'CP_PHASE_CODE' AND LANGUAGE = 'US'
       AND ENABLED_FLAG = 'Y';

--STATUS
SELECT LOOKUP_CODE, MEANING
  FROM FND_LOOKUP_VALUES
WHERE LOOKUP_TYPE = 'CP_STATUS_CODE' AND LANGUAGE = 'US'
       AND ENABLED_FLAG = 'Y';

       SELECT DISTINCT
       UPCVL.CONCURRENT_PROGRAM_ID,
       UPCVL.CONCURRENT_PROGRAM_NAME,
       UPCVL.EXECUTABLE_APPLICATION_ID,
       UPCVL.EXECUTABLE_ID,
       FEFV.EXECUTION_FILE_NAME,
       FEFV.DESCRIPTION,
       UPCVL.APPLICATION_ID,
       FAVL.APPLICATION_NAME,
       FAVL.APPLICATION_SHORT_NAME,
       FCR.REQUEST_ID,
       SC.MEANING AS STATUS_CODE,
       PC.MEANING AS PHASE_CODE,
       FCR.ACTUAL_COMPLETION_DATE - FCR.ACTUAL_START_DATE AS EXECUTION_DURATION
FROM   FND_CONCURRENT_PROGRAMS_VL UPCVL
JOIN   FND_APPLICATION_VL FAVL
       ON UPCVL.APPLICATION_ID = FAVL.APPLICATION_ID
JOIN   FND_EXECUTABLES_FORM_V FEFV
```

```
       ON UPCVL.EXECUTABLE_ID = FEFV.EXECUTABLE_ID
JOIN   FND_CONCURRENT_REQUESTS FCR
       ON UPCVL.CONCURRENT_PROGRAM_ID = FCR.CONCURRENT_PROGRAM_ID
JOIN   (SELECT LOOKUP_CODE, MEANING
        FROM FND_LOOKUP_VALUES
        WHERE LOOKUP_TYPE = 'CP_STATUS_CODE'
          AND LANGUAGE = 'US'
          AND ENABLED_FLAG = 'Y') SC
       ON SC.LOOKUP_CODE = FCR.STATUS_CODE
JOIN   (SELECT LOOKUP_CODE, MEANING
        FROM FND_LOOKUP_VALUES
        WHERE LOOKUP_TYPE = 'CP_PHASE_CODE'
          AND LANGUAGE = 'US'
          AND ENABLED_FLAG = 'Y') PC
       ON PC.LOOKUP_CODE = FCR.PHASE_CODE
WHERE  UPCVL.CONCURRENT_PROGRAM_NAME = 'XXQGEN_USER_DATA_PGK_DK';


SELECT DISTINCT UPCVL.CONCURRENT_PROGRAM_ID,
            UPCVL.CONCURRENT_PROGRAM_NAME,
            UPCVL.EXECUTABLE_APPLICATION_ID,
            UPCVL.EXECUTABLE_ID,
            FEFV.EXECUTION_FILE_NAME,
            FEFV.DESCRIPTION,
            UPCVL.APPLICATION_ID,
            FAVL.APPLICATION_NAME,
            FAVL.APPLICATION_SHORT_NAME ,
            FCR.REQUEST_ID,
            SC.MEANING STATUS_CODE,
            PC.MEANING PHASE_CODE,
            FCR.ACTUAL_COMPLETION_DATE - FCR.ACTUAL_START_DATE
FROM    FND_CONCURRENT_PROGRAMS_VL UPCVL,
             FND_APPLICATION_VL FAVL,
             FND_EXECUTABLES_FORM_V FEFV,
             FND_CONCURRENT_REQUESTS FCR,
            (SELECT LOOKUP_CODE, MEANING
             FROM FND_LOOKUP_VALUES
             WHERE LOOKUP_TYPE = 'CP_STATUS_CODE' AND LANGUAGE = 'US'
             AND ENABLED_FLAG = 'Y') SC,
             (SELECT LOOKUP_CODE, MEANING
              FROM FND_LOOKUP_VALUES
              WHERE LOOKUP_TYPE = 'CP_PHASE_CODE' AND LANGUAGE = 'US'
              AND ENABLED_FLAG = 'Y') PC
WHERE 1=1
AND     UPCVL.CONCURRENT_PROGRAM_NAME = 'XXQGEN%'
AND     UPCVL.CONCURRENT_PROGRAM_ID  = FCR.CONCURRENT_PROGRAM_ID
AND     UPCVL.APPLICATION_ID = FAVL.APPLICATION_ID
AND     UPCVL.EXECUTABLE_ID = FEFV.EXECUTABLE_ID
AND     SC.LOOKUP_CODE = FCR.STATUS_CODE
AND     PC.LOOKUP_CODE = FCR.PHASE_CODE;



SELECT DISTINCT UPCVL.CONCURRENT_PROGRAM_ID,
            UPCVL.CONCURRENT_PROGRAM_NAME,
            UPCVL.EXECUTABLE_APPLICATION_ID,
            UPCVL.EXECUTABLE_ID,
            FEFV.EXECUTION_FILE_NAME,
            FEFV.DESCRIPTION,
            UPCVL.APPLICATION_ID,
            FAVL.APPLICATION_NAME,
            FAVL.APPLICATION_SHORT_NAME ,
            FCR.REQUEST_ID
FROM    FND_CONCURRENT_PROGRAMS_VL UPCVL,
             FND_APPLICATION_VL FAVL,
             FND_EXECUTABLES_FORM_V FEFV,
             FND_CONCURRENT_REQUESTS FCR
WHERE 1=1
AND     UPCVL.CONCURRENT_PROGRAM_NAME = 'XXQGEN_USER_DATA_PGK_DK'
AND     UPCVL.CONCURRENT_PROGRAM_ID  = FCR.CONCURRENT_PROGRAM_ID
AND     UPCVL.APPLICATION_ID = FAVL.APPLICATION_ID
AND     UPCVL.EXECUTABLE_ID = FEFV.EXECUTABLE_ID;


SELECT *
FROM FND_CONCURRENT_REQUESTS
WHERE CONCURRENT_PROGRAM_ID = 218711;

--  FND_CONCURRENT_PROGRAMS_TL T, FND_CONCURRENT_PROGRAMS B
  SELECT *
  FROM FND_CONCURRENT_PROGRAMS;

  SELECT *
  FROM FND_CONCURRENT_PROGRAMS_TL;

  SELECT *
  FROM V$INSTANCE;

  SELECT *
  FROM ICX_PARAMETERS;




SELECT fcpv.application_id,
            fcpv.concurrent_program_name,
            fcpv.last_update_date,
            fcpv.executable_application_id,
            fcpv.executable_id,
            FCPV.USER_CONCURRENT_PROGRAM_NAME,
            fefv.EXECUTION_FILE_NAME,
            fefv.description,
            fefv.application_name,
            FCPV.EXECUTABLE_APPLICATION_ID,
            fcr.request_id,
            fcr.last_updated_by,
            fcr.STATUS_CODE,
            fcr.PHASE_CODE,
            fcr.PP_END_DATE,
            fcr.PP_START_DATE,
            fcr.EXP_DATE,
            fcr.CRM_TSTMP,
            fcr.ACTUAL_COMPLETION_DATE,
            fcr.ACTUAL_START_DATE,
            fcr.NUMBER_OF_ARGUMENTS,
            fcr.RESPONSIBILITY_ID,
            fcr.RESPONSIBILITY_APPLICATION_ID,
            fcr.HOLD_FLAG
FROM    FND_CONCURRENT_PROGRAMS_VL fcpv,
            FND_EXECUTABLES_FORM_V FEFV,
            FND_CONCURRENT_REQUESTS FCR
WHERE USER_CONCURRENT_PROGRAM_NAME like 'XXQGEN%DK'
and       fcpv.executable_id = fefv.executable_id
and       FCpv.CONCURRENT_PROGRAM_ID  = FCR.CONCURRENT_PROGRAM_ID;

select *
from fnd_executables_form_v;

select *
from FND_CONCURRENT_REQUESTS;

select *
from fnd_concurrent_programs_vl
where user_concurrent_program_name like'XXQGEN%DK';


select *
from XDO_DS_DEFINITIONS_B
```

```
where data_source_code like '%dk';

select *
from XDO_DS_DEFINITIONS_TL
where data_source_code like '%dk'
order by creation_date desc;

SELECT
FROM FND_USER FU , PER_ALL_PEOPLE_F PAPF
WHERE FU.EMPLOYEE_ID = PAPF.PERSON_ID
AND FU.USER_NAME ='OPERATIONS';



select fu.user_name, fu.creation_date, papf.full_name, papf.sex, papf.nationality, papf.start_date, fu.end_date
from fnd_user fu, per_all_people_f papf
where fu.employee_id = papf.person_id
and    fu.user_name = 'OPERATIONS' ;
```