Database Security

1) Install the Oracle DBMS on the machine on the Red Hat Enterprise Linux (RHEL).

2)

1. Database Firewall and Network Segmentation:

Configuration: Implement a database firewall or access controls to restrict network access to the Oracle DBMS. Utilize VLANs or network segmentation to isolate the database server from other systems.

Why: Restricting network access helps prevent unauthorized access and lateral movement within the network, protecting against remote exploitation and unauthorized data retrieval.

2. User Access and Authentication Controls:

Configuration: Enforce strong password policies, use authentication methods like PKI, and restrict user privileges within the database using Oracle's built-in features and role-based access controls.

Why: Strong authentication and access controls limit the potential for unauthorized database access, data manipulation, and privilege escalation.

3. Oracle Database Auditing:

Configuration: Enable auditing features within Oracle Database to track and log database activities, including login attempts, data changes, and system operations.

Why: Auditing helps in monitoring and detecting unusual or malicious activities within the database, enhancing security, and assisting in regulatory compliance.

4. To enable basic database auditing in Oracle:

   ALTER SYSTEM SET AUDIT_TRAIL=DB;

   AUDIT CREATE SESSION;

   Operating System Hardening:

Configuration: Apply standard Linux hardening practices (as mentioned in the previous answer) to the host machine running the Oracle DBMS. This includes firewall settings, regular system updates, and secure user authentication.

Why: A secure underlying operating system is critical because vulnerabilities at this level can impact the security of the Oracle DBMS.

5. Database Patching and Updates:

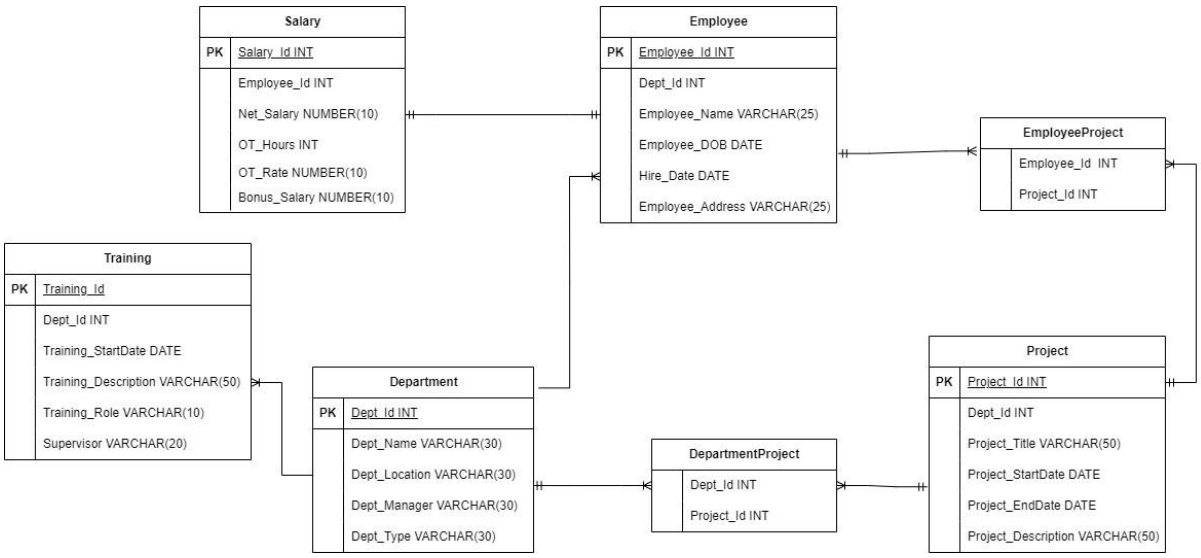
Configuration: Regularly apply Oracle patches, updates, and security fixes to the DBMS software. Oracle releases Critical Patch Updates (CPUs) quarterly, which should be applied promptly.

Why: Oracle issues patches to address known security vulnerabilities. Failing to apply patches leaves the system vulnerable to attacks targeting these vulnerabilities.

To apply an Oracle Database patch, you can use the OPatch utility:

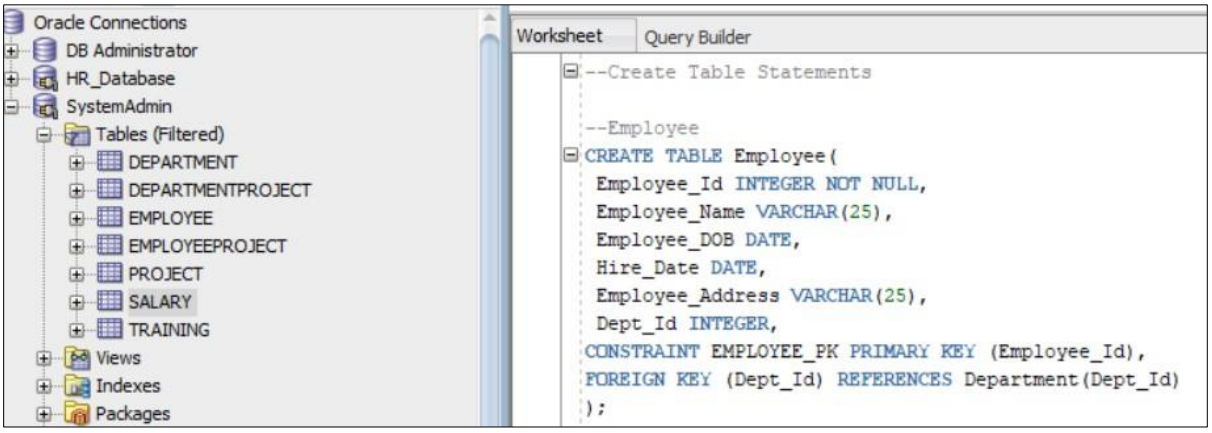
```
# Navigate to the OPatch directory
cd $ORACLE_HOME/OPatch

# Apply the patch
./opatch apply -silent
```

3)



Create Table Statements & Record Insert Statements

Employee Table



Department Table

```
CREATE TABLE Department(
 Dept_Id INTEGER NOT NULL,
 Dept_Name VARCHAR(30),
Dept_Location VARCHAR(30),
Dept_Manager VARCHAR(30),
Dept_Type VARCHAR2(30),
 CONSTRAINT DEPARTMENT_PK PRIMARY KEY (Dept_Id)
 );
```

```
--Department
INSERT INTO Department VALUES (200,'HR','Kandy','Amal','none');
INSERT INTO Department VALUES (201,'Account','Kandy','Anuradha','none');
INSERT INTO Department VALUES (202,'Production','Kandy','Janith','none');
INSERT INTO Department VALUES (203,'Sales','Kandy','kamal','none');
INSERT INTO Department VALUES (204,'Quality Assurance','Kandy','Amila','none');
INSERT INTO Department VALUES (205, 'IT' ,'Kandy','Ruwan','none');
INSERT INTO Department VALUES (206,'Development','Kandy','Shan','none');
INSERT INTO Department VALUES (207,' Purchasing','Kandy','Yasas','none');
INSERT INTO Department VALUES (208,'Division','Kandy','Kumudu','none');
INSERT INTO Department VALUES (209,'Project management','Kandy','Nisal','none');
```

Project Table

```
--EmployeeProject
CREATE TABLE EmployeeProject(
Employee_Id INTEGER,
 Project_Id INTEGER,
 CONSTRAINT EMPPROJECT_PK PRIMARY KEY (Employee_Id,Project_Id),
 FOREIGN KEY (Employee_Id) REFERENCES Employee(Employee_Id),
 FOREIGN KEY (Project_Id) REFERENCES Project(Project_Id)
 );
```

```
--EmployeeProject
INSERT INTO EmployeeProject VALUES (105,403);
INSERT INTO EmployeeProject VALUES (101,400);
INSERT INTO EmployeeProject VALUES (106,404);
INSERT INTO EmployeeProject VALUES (102,406);
INSERT INTO EmployeeProject VALUES (108,408);
INSERT INTO EmployeeProject VALUES (104,402);
INSERT INTO EmployeeProject VALUES (109,407);
INSERT INTO EmployeeProject VALUES (103,409);
INSERT INTO EmployeeProject VALUES (107,401);
INSERT INTO EmployeeProject VALUES (110,405);
```

Training Table

```
--Training
CREATE TABLE Training(
 Training_Id INTEGER NOT NULL,
 Training_StartDate DATE,
Training_Description VARCHAR(50),
Training_Role VARCHAR(10),
Supervisor VARCHAR(20),
Dept_Id INTEGER,
CONSTRAINT TRAINING_PK PRIMARY KEY (Training_Id),
FOREIGN KEY (Dept_Id) REFERENCES Department (Dept_Id)
 );
```

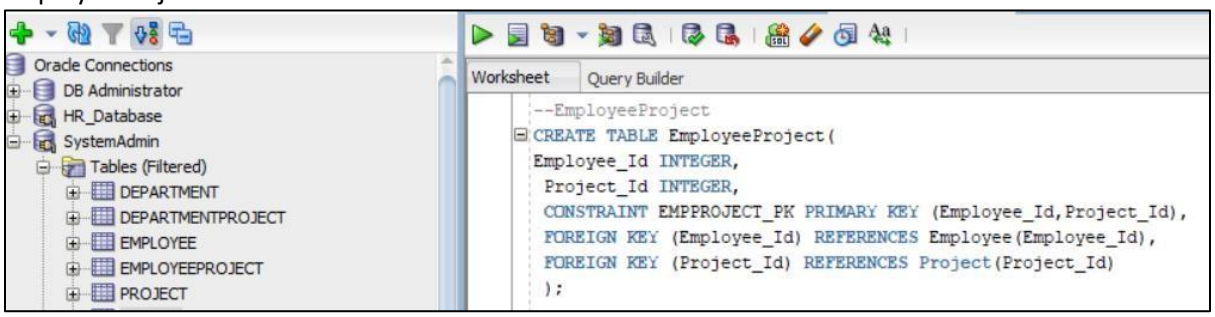
```
--Training
INSERT INTO Training VALUES (500,to_date('22-MAY-2022', 'dd-MON-yyyy'),'abc','QA','Amal','208');
INSERT INTO Training VALUES (501,to_date('28-JAN-2023', 'dd-MON-yyyy'),'def','SE','Kaml','203');
INSERT INTO Training VALUES (502,to_date('11-DEC-2020', 'dd-MON-yyyy'),'ghi','CYBER','Kasun','201');
INSERT INTO Training VALUES (503,to_date('21-JULY-2022', 'dd-MON-yyyy'),'jkl','ACCOUNTANT','Shan','202');
INSERT INTO Training VALUES (504,to_date('30-AUG-2020', 'dd-MON-yyyy'),'mno','MANAGER','Shan','204');
INSERT INTO Training VALUES (505,to_date('2-FEB-2021', 'dd-MON-yyyy'),'pqr','Consultant','Janith','207');
INSERT INTO Training VALUES (506,to_date('1-JAN-2022', 'dd-MON-yyyy'),'stu','SE','Visal','200');
INSERT INTO Training VALUES (507,to_date('10-MAY-2021', 'dd-MON-yyyy'),'vwz','OPARATOR','Supun','206');
INSERT INTO Training VALUES (508,to_date('20-OCT-2021', 'dd-MON-yyyy'),'yza','ANALYTIC','Amal','201');
INSERT INTO Training VALUES (509,to_date('23-NOV-2023', 'dd-MON-yyyy'),'bcd','ADMIN','Amal','205');
```

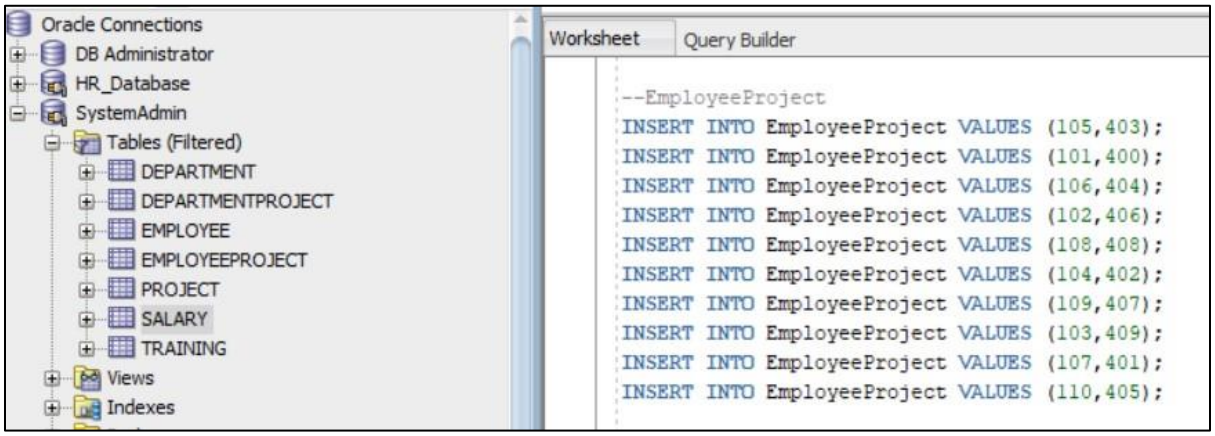
Salary Table

```
--Salary
CREATE TABLE Salary(
 Salary_Id INTEGER NOT NULL,
 Net_Salary NUMBER(10),
 OT_Hours INTEGER,
OT_Rate NUMBER(10),
 Bonus_Salary NUMBER(10),
Employee_Id INTEGER,
 CONSTRAINT SALARY_PK PRIMARY KEY (Salary_Id),
 FOREIGN KEY (Employee_Id) REFERENCES Employee (Employee_Id)
 );
```

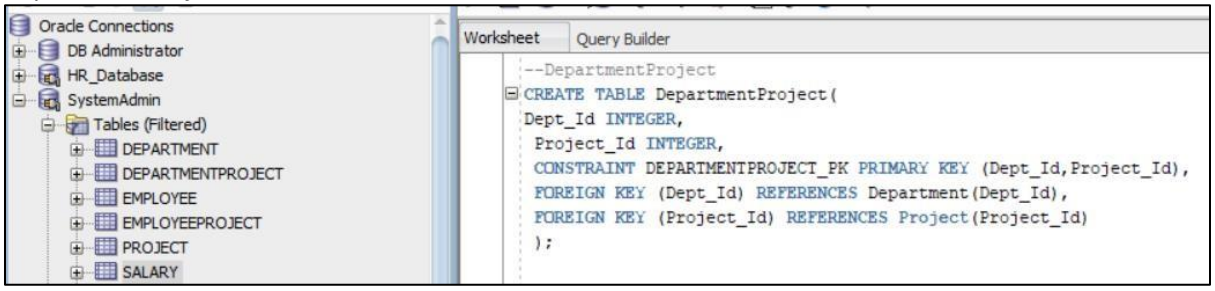
```
--Salary
INSERT INTO Salary VALUES (300,57600,30,200,10000,105);
INSERT INTO Salary VALUES (301,23500,25,200,2500,103);
INSERT INTO Salary VALUES (302,50000,32,200,1500,107);
INSERT INTO Salary VALUES (303,45000,28,200,10500,101);
INSERT INTO Salary VALUES (304,55000,10,200,2550,106);
INSERT INTO Salary VALUES (305,65000,26,200,7500,110);
INSERT INTO Salary VALUES (306,100000,20,200,9000,102);
INSERT INTO Salary VALUES (307,35000,35,200,550,104);
INSERT INTO Salary VALUES (308,85000,5,200,1000,108);
INSERT INTO Salary VALUES (309,25500,22,200,700,109);
```

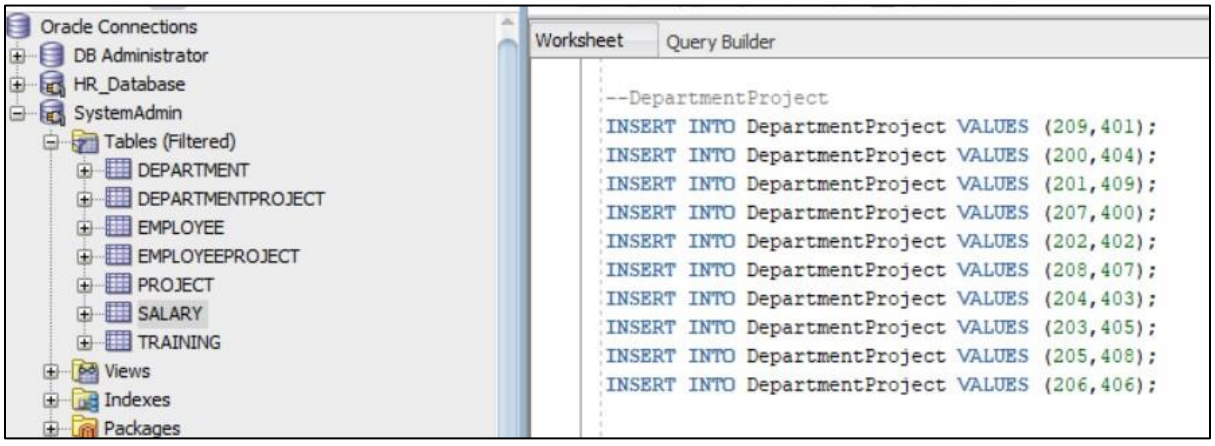
EmployeeProject Table





DepartmentProject Table





4) A

Create the 3 users

```
--Create the 3 users
CREATE USER C##SystemAdmin IDENTIFIED BY SystemAdmin;
CREATE USER C##Manager IDENTIFIED BY Manager;
CREATE USER C##Executive IDENTIFIED BY Executive;
```

Create the Admin, Manager and Executive roles.

```
--Create the 3 roles
CREATE ROLE C##system_admin_role;
CREATE ROLE C##manager_role;
CREATE ROLE C##executive_role;
```

```
ALTER USER system_admin DEFAULT ROLE ALL;
ALTER USER manager DEFAULT ROLE manager_role;
ALTER USER executive DEFAULT ROLE executive_role;
```

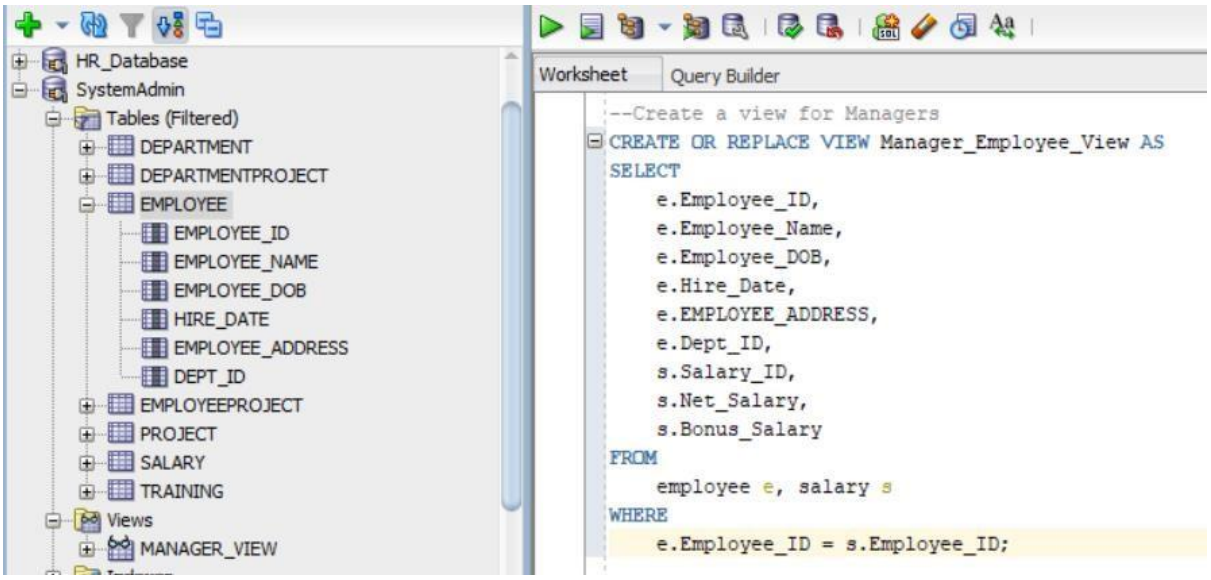
Allocate the roles/user permissions.

```
--System Admin (Full Permissions): Grant full database permissions, including system permissions.
GRANT CONNECT TO C##SystemAdmin;
GRANT CREATE SESSION, CREATE VIEW, CREATE TABLE, ALTER SESSION, CREATE SEQUENCE TO C##SystemAdmin;
GRANT CREATE SYNONYM, CREATE DATABASE LINK, RESOURCE, UNLIMITED TABLESPACE TO C##SystemAdmin;
```

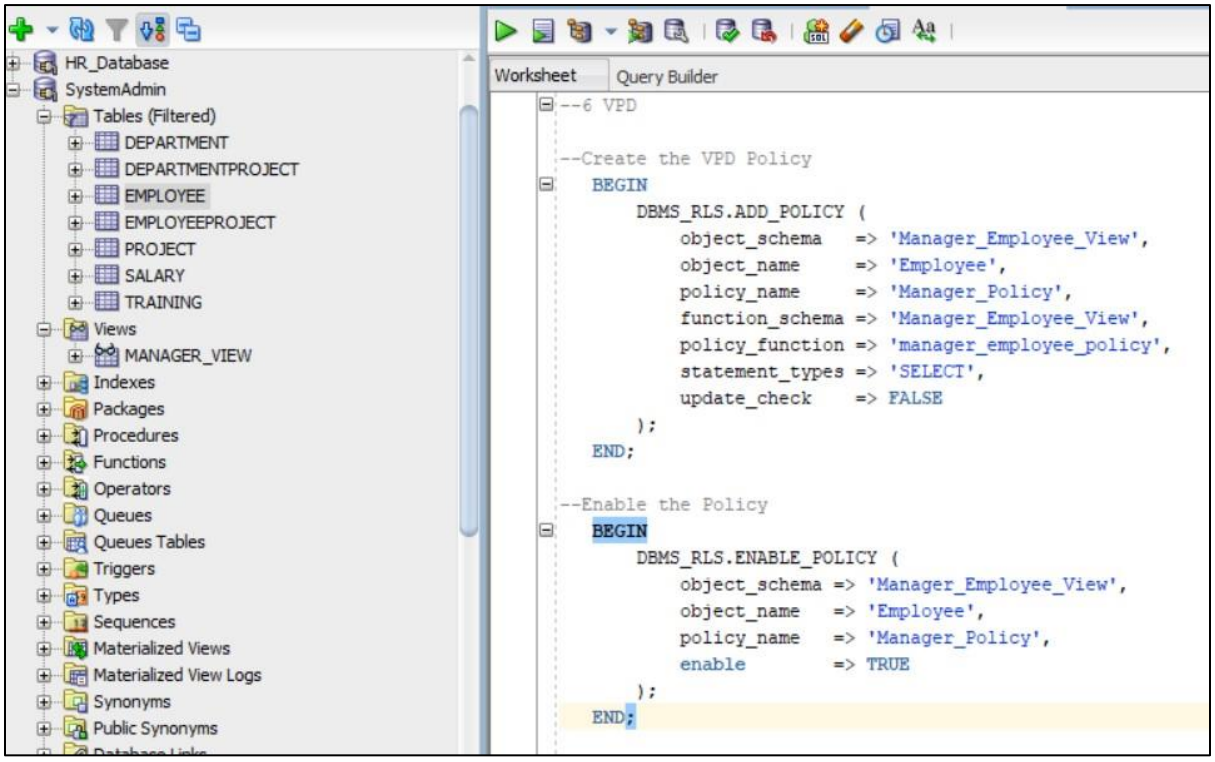
```
--Manager (Read/Write Permissions): Grant read and write permissions to the database.
GRANT SELECT, INSERT, UPDATE, DELETE ON C##SystemAdmin.employee TO C##manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON C##SystemAdmin.department TO C##manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON C##SystemAdmin.project TO C##manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON C##SystemAdmin.salary TO C##manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON C##SystemAdmin.training TO C##manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON C##SystemAdmin.EmployeeProject TO C##manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON C##SystemAdmin.DepartmentProject TO C##manager;
```

```
--Executive (Read-Only Permissions): Grant read-only permissions to the database.
GRANT SELECT ON C##SystemAdmin.employee TO C##executive;
GRANT SELECT ON C##SystemAdmin.department TO C##executive;
GRANT SELECT ON C##SystemAdmin.project TO C##executive;
GRANT SELECT ON C##SystemAdmin.salary TO C##executive;
GRANT SELECT ON C##SystemAdmin.training TO C##executive;
GRANT SELECT ON C##SystemAdmin.DepartmentProject TO C##executive;
GRANT SELECT ON C##SystemAdmin.EmployeeProject TO C##executive;
```

5)



6) Create a VPD



7)

a) Identifying the specific columns in HR database that contain sensitive data requiring encryption. These might include fields net_salary, salary_id, and personal identification information.

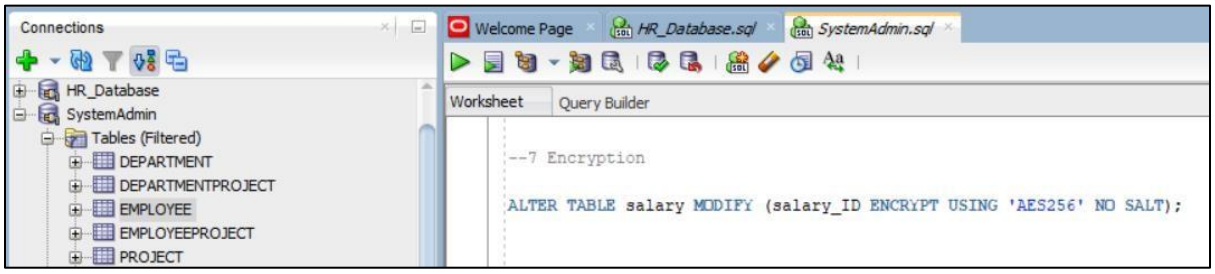
b) Oracle TDE requires a wallet to store encryption keys securely. Create a wallet using the `mkstore` utility, which is typically located in the Oracle home directory.

```
mkstore -wrl /path_to_wallet_directory -create
```

c) To use the wallet, open it using the `mkstore` utility:

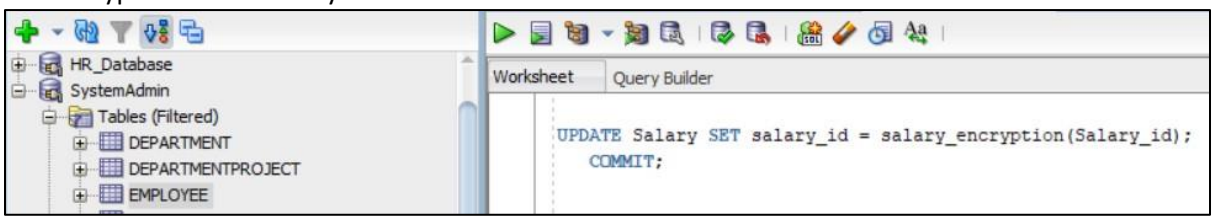
```
mkstore -wrl /path_to_wallet_directory -createCredential mydb_alias username
```

d) Need to configure TDE for the specific database where the sensitive data resides. Do this by setting the `ENCRYPTION` attribute for the sensitive columns using the `ALTER TABLE` statement.



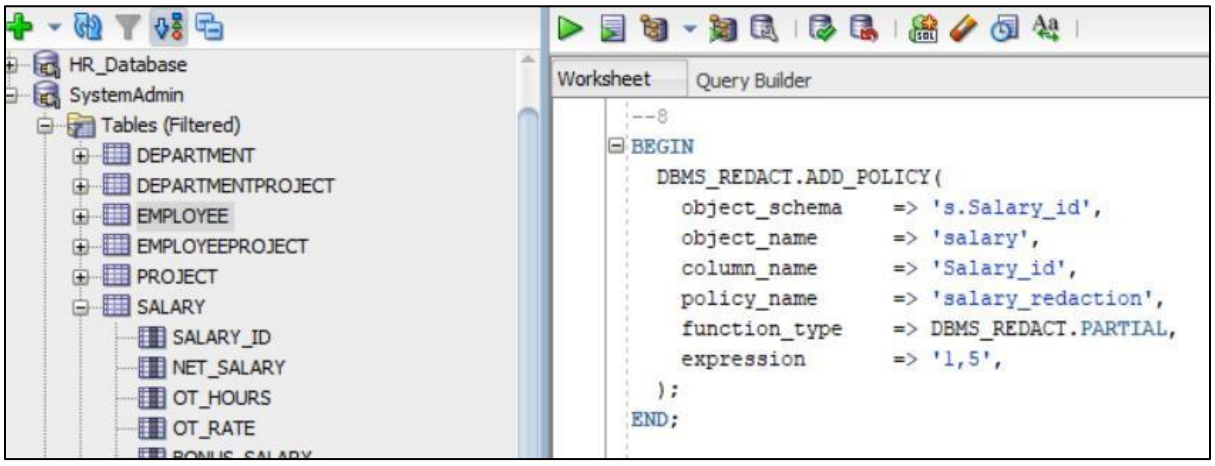
we're encrypting "Salary_ID" in "Salary" using AES256 encryption with no salt.

e) Update the existing data in the sensitive columns to encrypt them, or new data will be encrypted automatically when inserted.

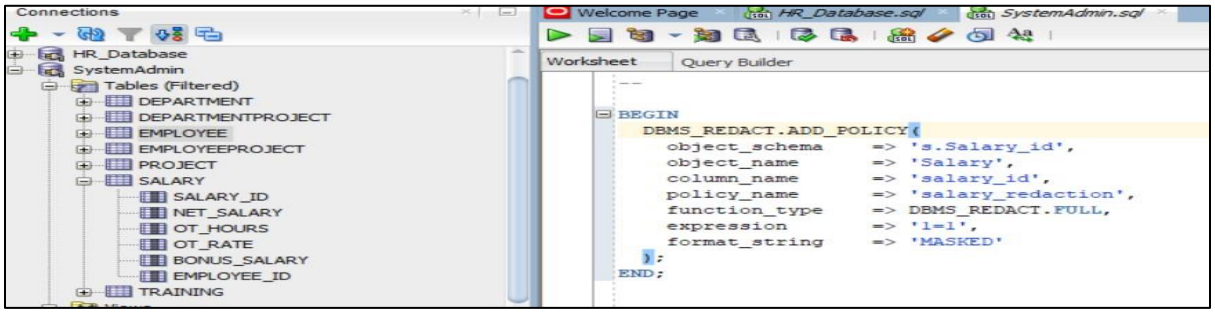


8)

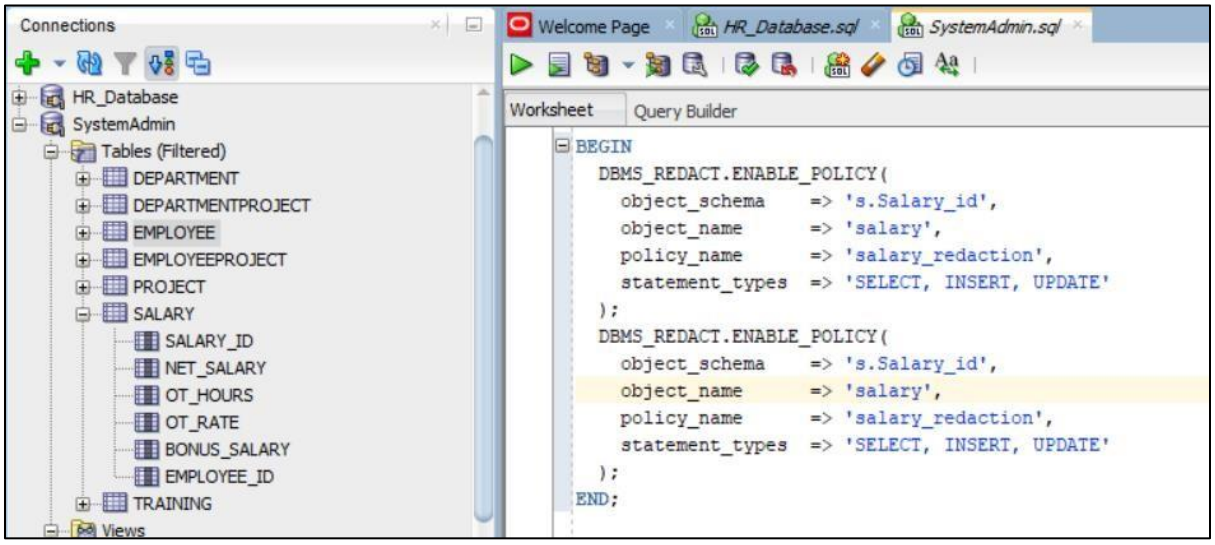
a) Create a Redaction Policy for SSN Masking.



b) Create a Redaction Policy for Salary Masking:



c) Enable the Redaction Policies:



9)

a) PL/SQL function that defines the audit condition

```
--9 FGA
CREATE OR REPLACE FUNCTION is_salary_access (
        Salary_id  IN INTEGER,
        Net_salary  IN NUMBER,
    ) RETURN BOOLEAN AS
    BEGIN
        IF INSTR(SELECT net_salary FROM Salary, 'Salary_id') > 0 THEN
            RETURN TRUE;
        ELSE
            RETURN FALSE;
        END IF;
    END;
```

b) Use the `DBMS_FGA.ADD_POLICY` procedure to create the FGA policy.

```
---
BEGIN
    DBMS_FGA.ADD_POLICY (
        object_schema     => 's.Salary_id',
        object_name       => 'salary',
        policy_name       => 'Salary_Audit_Policy',
        audit_condition   => 'is_salary_access(salary, salary_id, SELECT * FROM Salary)',
        audit_column      => 'Salary_id',
        handler_schema    => 'salary',
        statement_types   => 'SELECT',
        audit_trail       => DBMS_FGA.DB,
        enable            => TRUE
    );
END;
```

c) After creating the FGA policy, enable it using the `DBMS_FGA.ENABLE_POLICY` procedure:

```
---
BEGIN
    DBMS_FGA.ENABLE_POLICY (
        object_schema => 's.salary_id',
        object_name   => 'salary',
        policy_name   => 'Salary_Audit_Policy',
        enable        => TRUE
    );
END;
```