# Application Name: Online Job Bank System.

| | |
|---|---|
| **Course Title** | Database Systems 1 |
| **Course Number** | CPS 510 |
| **Semester/Year** | F2021 |
| **Instructor** | Dr. Abdolreza Abhari |
| **Assignment Number** | 10 |
| **Group** | 11 |
| **Assignment Title** | Final Project - Database Management System |

| Full Name | Student Number | Group # | Section Number | Initial |
|---|---|---|---|---|
| **Tusaif Azmat** | **500660278** | **11** | **04** | **T.A** |
| **Ankit Dheedsa** | **500975118** | | | **A.D** |
| **Mahdi Alam** | **500969935** | | | **M.A** |

# Table of Contents

**Application Description:**

This Online Job Bank application will work the same way as any other online job bank where job seekers and employers will use. We want to create an application similar to **Linkedin** and **Indeed** etc. These online job banks have been soaring in popularity and are crucial as they connect people all around the world on one platform.

We have selected this project because it will allow us to have hands-on experience with SQL database systems. We believe that we could use the experience and knowledge gained in this course to fully utilize it during the development process for this group project.

The main objective is to create a platform where users (Both Job Seekers and Employers) gain access to an online job bank database. Where they use as a job seeker will look through the database for jobs and be able to apply online using the provided services. Also an employer will gain access to the database and post new job postings and delete old postings. It will also have system administrator access along with all the company employees.

Our online Job Bank application will provide users with various job opportunities that can be accessed with just a click of a button. We will be taking inspiration for our application from other established services such as Linkedin and Indeed, we intend to incorporate networking components from Linkedin and job filtering components from Indeed. Our application will make possible the interactions between potential job candidates and hiring managers and provide an easy-to-use interface where users can easily identify and apply for jobs that are tailored to their needs.

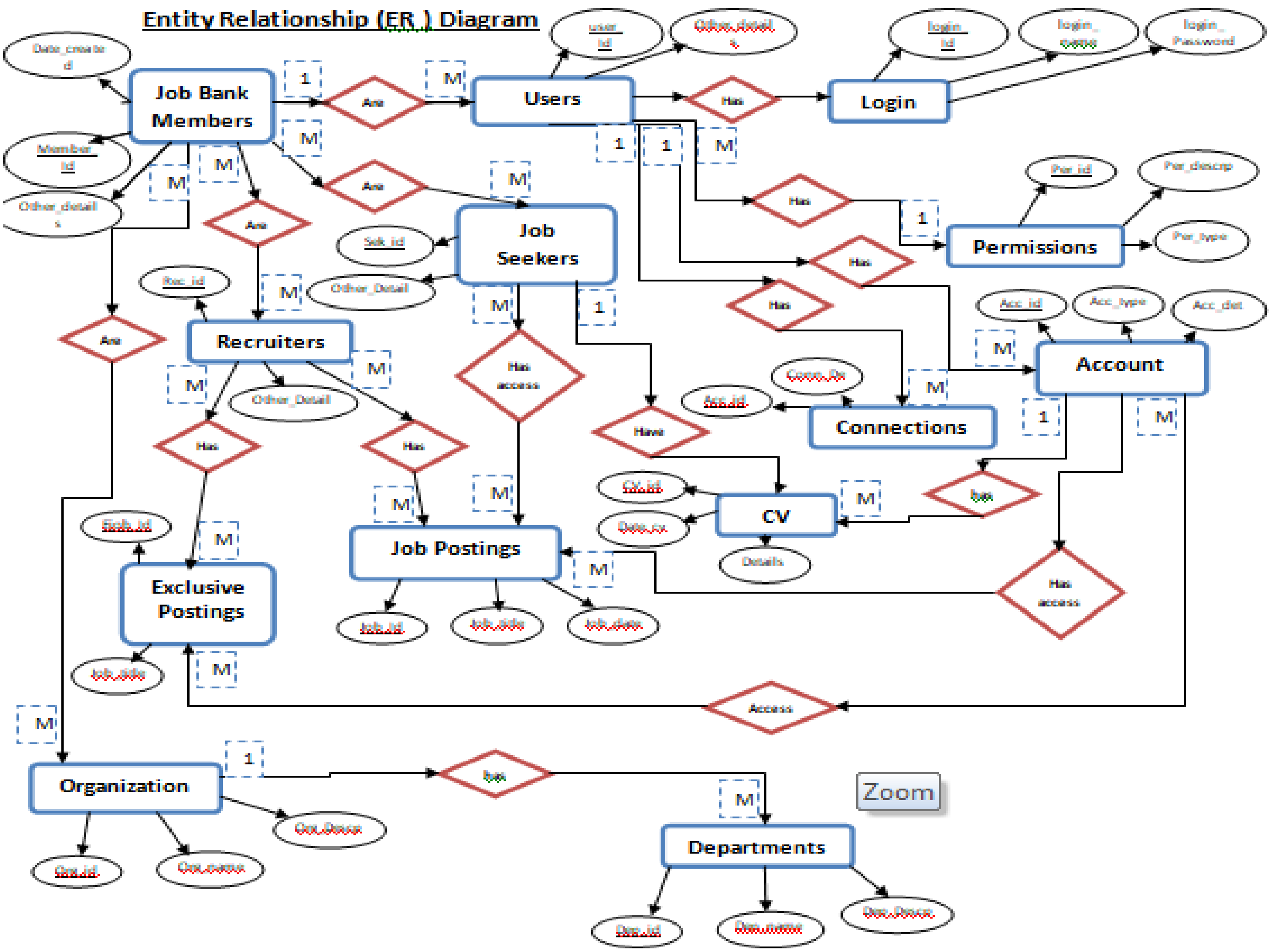
In our DBMS, users can also opt for a subscription based premium service in which they can get further detailed information about jobs and a direct connection to interviewers. Large and small companies also reap the benefits of these subscription services as they could pool these applicants in a different bracket and hire them based on company needs. This application will have the ability to hold specific types of information about jobs that will make it easier for the user to follow.

Some of the basic functions that our system will be capable of doing are outlined below:

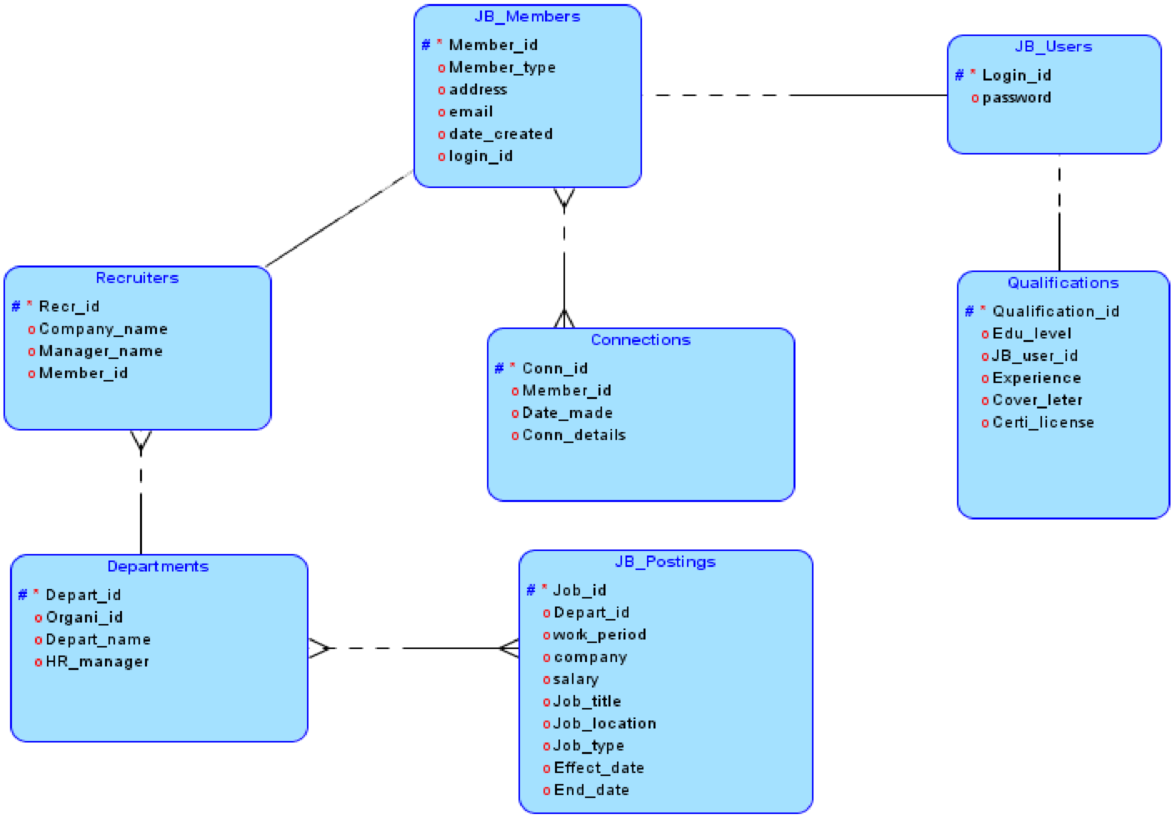
| Function | Description |
|---|---|
| User Login | This function will allow users to log in after authentication. After log in they could access the services provided. |
| Search Jobs | This is how users will search for jobs based on the search criteria. Users will search jobs based on employers and current jobs etc. |
| Job posting | This is where users will post jobs and remove existing postings. This option will be provided to the companies/employers. |
| Job Application Status | This is where users can see what jobs they applied to, the status of their application, and where they can accept job offers. |
| Dashboard | This is essentially a home page geared for the user. They can see news related to their job offers, see surveys and reports, see their Job Application Status, Job postings geared to them, etc. |

| | |
|---|---|
| Notifications Log | View important updates in a list. One can see the contents of the update, the timestamp of the update, and occasionally a button to do certain actions related to the notification. |
| | |

**Initial ER_Diagram:**



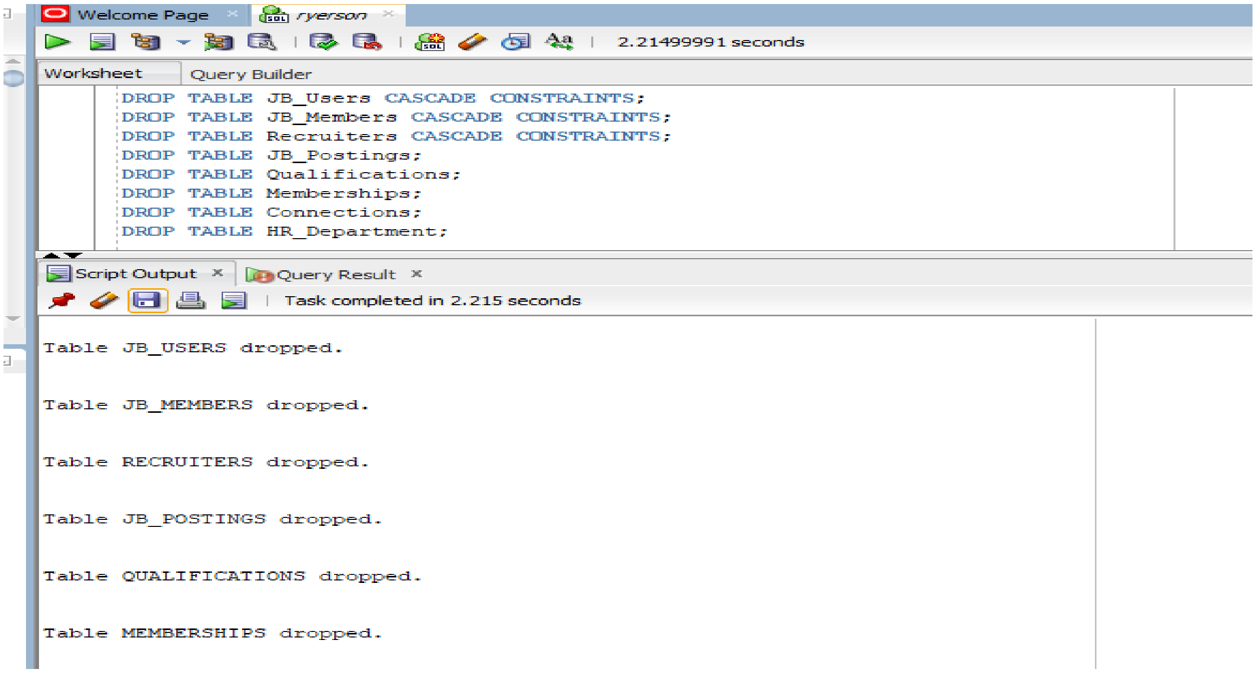
### Finalized ER-diagram after BCNF Normalization



## Schema Design:

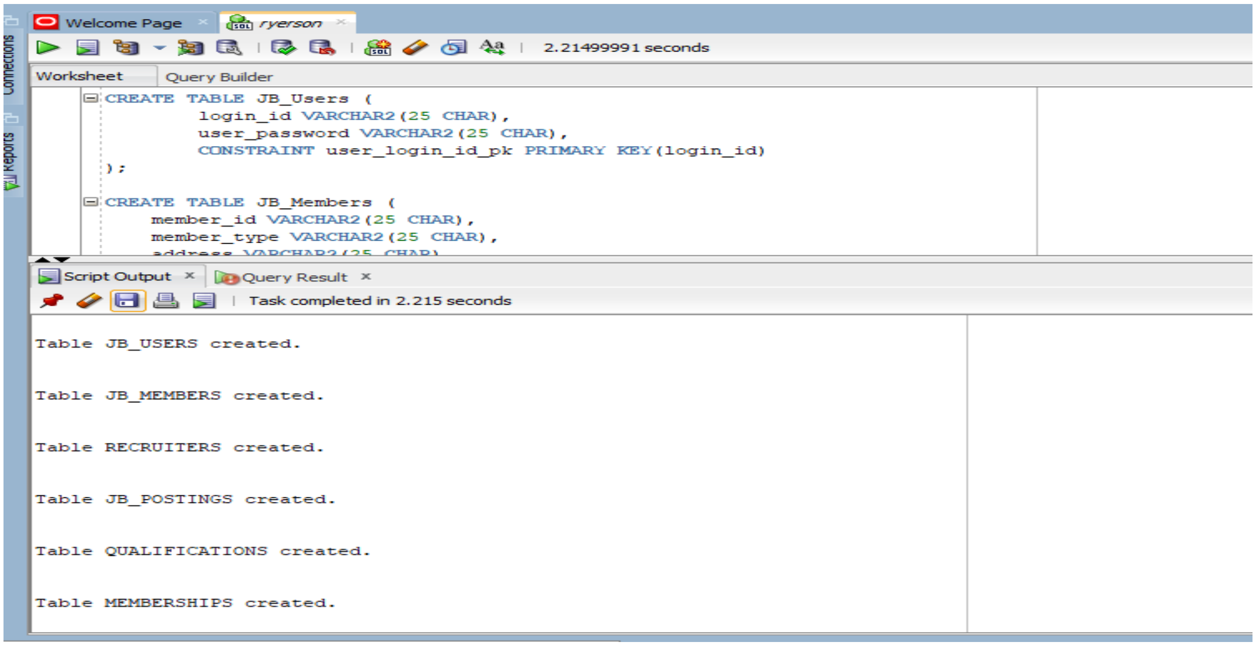
**Source Code:**

```
DROP TABLE JB_Users CASCADE CONSTRAINTS;
DROP TABLE JB_Members CASCADE CONSTRAINTS;
DROP TABLE Recruiters CASCADE CONSTRAINTS;
DROP TABLE JB_Postings;
DROP TABLE Qualifications;
DROP TABLE Memberships;
DROP TABLE Connections;
DROP TABLE HR_Department;

CREATE TABLE JB_Users (
        login_id VARCHAR2(25 CHAR),
        user_password VARCHAR2(25 CHAR),
        CONSTRAINT user_login_id_pk PRIMARY KEY(login_id)
);

CREATE TABLE JB_Members (
   member_id VARCHAR2(25 CHAR),
   member_type VARCHAR2(25 CHAR),
   address VARCHAR2(25 CHAR),
   org_id VARCHAR2(25 CHAR),
    email VARCHAR2(25 CHAR),
    date_Created VARCHAR2(25 CHAR),
    member_name VARCHAR2 (25 CHAR),
    CONSTRAINT member_pk PRIMARY KEY(member_id),
    CONSTRAINT JBUser_fk FOREIGN KEY (member_name) REFERENCES
    JB_Users(login_id) -- Foreign Key
);

CREATE TABLE Recruiters (
     recr_id VARCHAR2(25 CHAR),
          member_id VARCHAR2(25 CHAR),
          job_id VARCHAR2(25 CHAR),
          date_created VARCHAR2(25 CHAR),
          email VARCHAR2(25 CHAR),
          phone VARCHAR2(12 CHAR) DEFAULT '(000)-000-0000',
          address VARCHAR2(25 CHAR),
          CONSTRAINT Recruiters_pk PRIMARY KEY(recr_id),
     FOREIGN KEY (member_id) REFERENCES JB_Members (member_id)
);

CREATE TABLE JB_Postings (
          job_id VARCHAR2 (20 CHAR),
     work_period NUMBER,
     company VARCHAR2(25 CHAR),
     salary NUMBER,
     job_title VARCHAR2(25 CHAR),
     job_location VARCHAR2(25 CHAR),
     job_type VARCHAR2(25 CHAR),
          effective_date VARCHAR2(25 CHAR),
          end_date VARCHAR2(25 CHAR),
     CONSTRAINT jobs_pk PRIMARY KEY (job_id),--primary key
          FOREIGN KEY (company) REFERENCES recruiters (recr_id) --Foreign Key
);

CREATE TABLE Qualifications(
        qualification_id VARCHAR2(10 CHAR),
        user_degree VARCHAR2(50 CHAR) DEFAULT 'Undergraduate',
        JB_User_id VARCHAR2(25 CHAR) NOT NULL,
        years_experience NUMBER CHECK (years_experience BETWEEN 0 AND 30),
        coverLetter VARCHAR2(1000 CHAR),
        license VARCHAR2(25 CHAR),
        CONSTRAINT qualification_pk PRIMARY KEY (qualification_id),--Primary ID
        FOREIGN KEY (JB_user_id) REFERENCES JB_Users (login_id) --Foreign Key
);

CREATE TABLE Memberships (
          permission_id VARCHAR2(10 CHAR),
		user_id VARCHAR2(25 CHAR),
		Type VARCHAR2(25 CHAR),
		Description VARCHAR2(25 CHAR),
		Start_date VARCHAR2(25 CHAR),
		End_date VARCHAR2(25 CHAR),
		CONSTRAINT permission_pk PRIMARY KEY(permission_id),
		FOREIGN KEY (user_id) REFERENCES JB_Users (login_id)
	);

	CREATE TABLE Connections (
		conn_id VARCHAR2(12 CHAR),
		user_id VARCHAR2(25 CHAR),
		date_made VARCHAR2(25 CHAR),
		details VARCHAR2(150 CHAR),
		CONSTRAINT conn_pk PRIMARY KEY(conn_id),
		FOREIGN KEY (user_id) REFERENCES JB_Users (login_id)
	);

	CREATE TABLE HR_Department (
		depart_id VARCHAR2(25 CHAR),
		organization_id VARCHAR2(25 CHAR),
		depart_name VARCHAR2(50 CHAR),
		hr_manager VARCHAR2(50 CHAR),
		no_employees NUMBER,
		CONSTRAINT depart_pk PRIMARY KEY(depart_id),
		FOREIGN KEY (organization_id) REFERENCES Recruiters (recr_id)
);
```



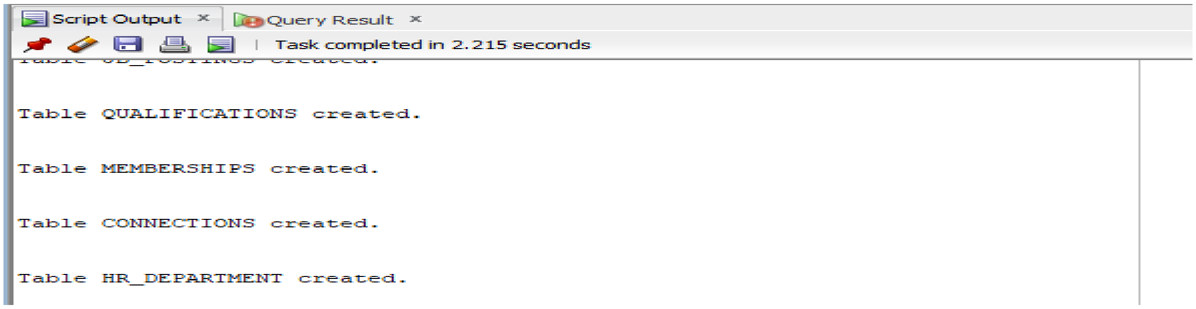
```
DROP TABLE JB_Users CASCADE CONSTRAINTS;
DROP TABLE JB_Members CASCADE CONSTRAINTS;
DROP TABLE Recruiters CASCADE CONSTRAINTS;
DROP TABLE JB_Postings;
DROP TABLE Qualifications;
DROP TABLE Memberships;
DROP TABLE Connections;
DROP TABLE HR_Department;
```

```
Table JB_USERS dropped.


Table JB_MEMBERS dropped.


Table RECRUITERS dropped.


Table JB_POSTINGS dropped.


Table QUALIFICATIONS dropped.


Table MEMBERSHIPS dropped.
```



```
CREATE TABLE JB_Users (
        login_id VARCHAR2(25 CHAR),
        user_password VARCHAR2(25 CHAR),
        CONSTRAINT user_login_id_pk PRIMARY KEY(login_id)
);

CREATE TABLE JB_Members (
    member_id VARCHAR2(25 CHAR),
    member_type VARCHAR2(25 CHAR),
    address VARCHAR2(25 CHAR)
```

```
Table JB_USERS created.

Table JB_MEMBERS created.

Table RECRUITERS created.

Table JB_POSTINGS created.

Table QUALIFICATIONS created.

Table MEMBERSHIPS created.
```

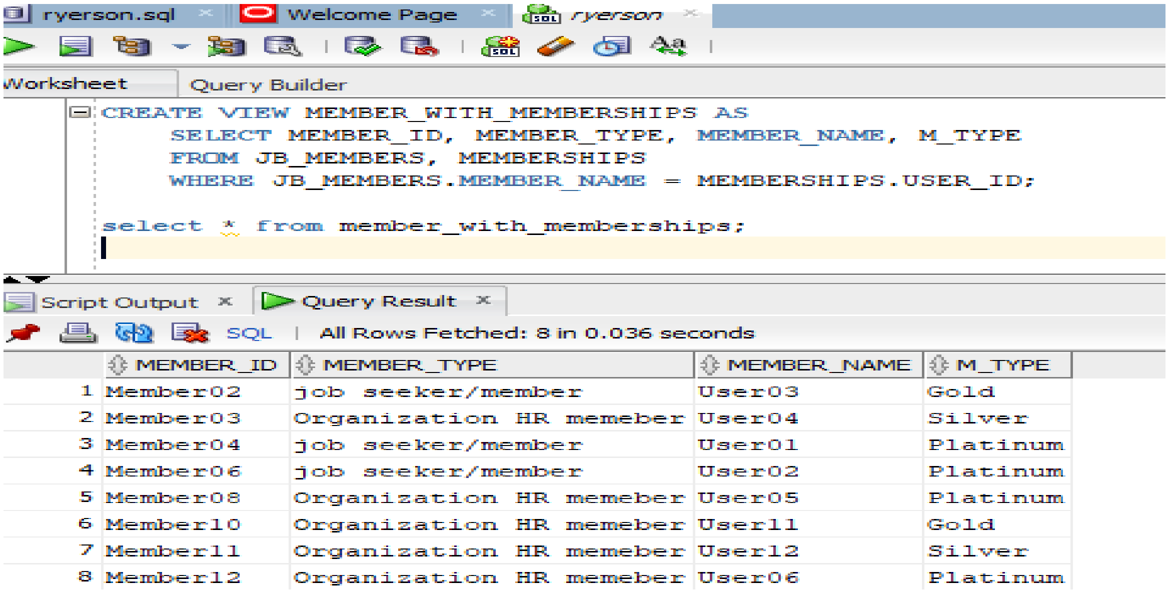


```
Table JB_POSTINGS created.

Table QUALIFICATIONS created.

Table MEMBERSHIPS created.

Table CONNECTIONS created.

Table HR_DEPARTMENT created.
```

## Views and Queries

-- Table Views

```
CREATE VIEW MEMBER_WITH_MEMBERSHIPS AS
   SELECT MEMBER_ID, MEMBER_TYPE, MEMBER_NAME, M_TYPE
   FROM JB_MEMBERS, MEMBERSHIPS
   WHERE JB_MEMBERS.MEMBER_NAME = MEMBERSHIPS.USER_ID;

select * from member_with_memberships;
```

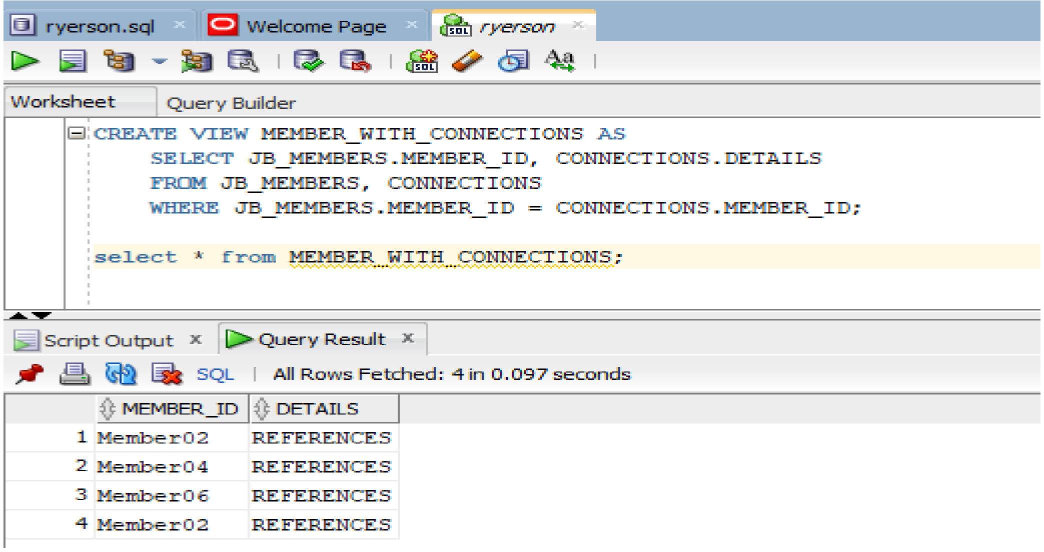


```
CREATE VIEW MEMBER_WITH_MEMBERSHIPS AS
     SELECT MEMBER_ID, MEMBER_TYPE, MEMBER_NAME, M_TYPE
     FROM JB_MEMBERS, MEMBERSHIPS
     WHERE JB_MEMBERS.MEMBER_NAME = MEMBERSHIPS.USER_ID;

select * from member_with_memberships;
```

All Rows Fetched: 8 in 0.036 seconds

|  | MEMBER_ID | MEMBER_TYPE | MEMBER_NAME | M_TYPE |
|---|---|---|---|---|
| 1 | Member02 | job seeker/member | User03 | Gold |
| 2 | Member03 | Organization HR memeber | User04 | Silver |
| 3 | Member04 | job seeker/member | User01 | Platinum |
| 4 | Member06 | job seeker/member | User02 | Platinum |
| 5 | Member08 | Organization HR memeber | User05 | Platinum |
| 6 | Member10 | Organization HR memeber | User11 | Gold |
| 7 | Member11 | Organization HR memeber | User12 | Silver |
| 8 | Member12 | Organization HR memeber | User06 | Platinum |

```
CREATE VIEW MEMBER_WITH_CONNECTIONS AS
   SELECT JB_MEMBERS.MEMBER_ID, CONNECTIONS.DETAILS
   FROM JB_MEMBERS, CONNECTIONS
   WHERE JB_MEMBERS.MEMBER_ID = CONNECTIONS.MEMBER_ID;

select * from MEMBER_WITH_CONNECTIONS;
```

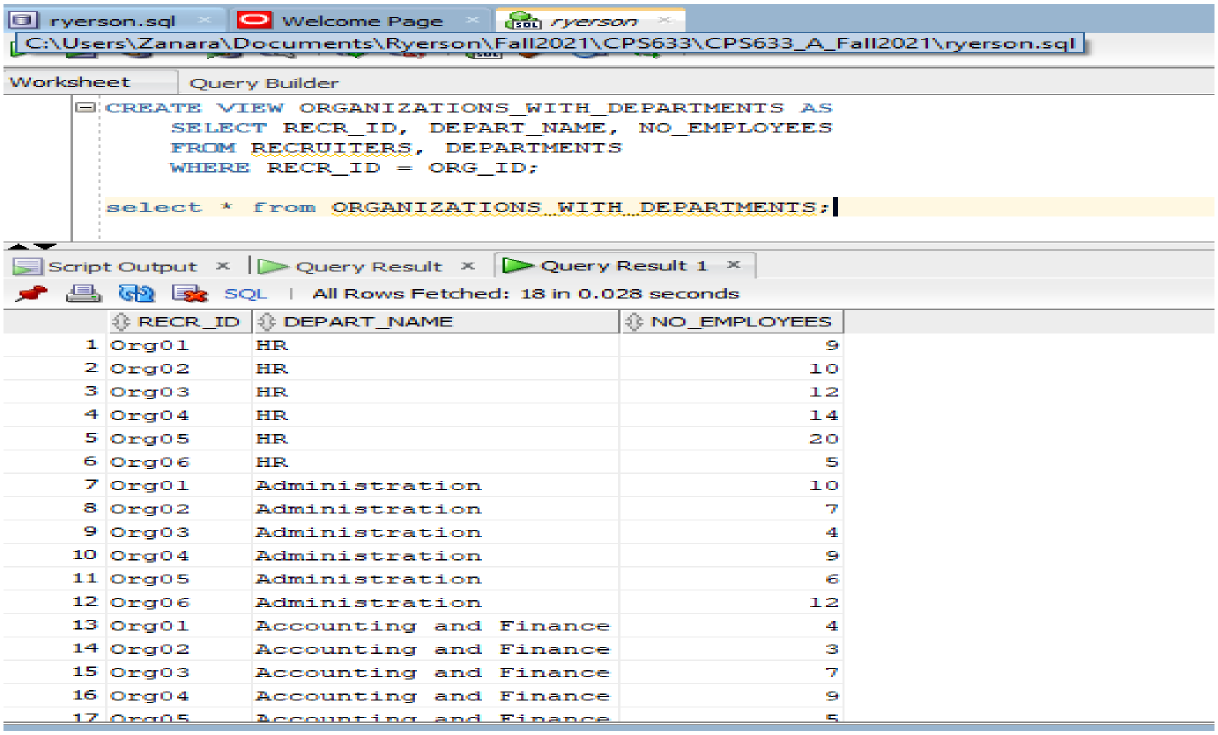


```
CREATE VIEW MEMBER_WITH_CONNECTIONS AS
     SELECT JB_MEMBERS.MEMBER_ID, CONNECTIONS.DETAILS
     FROM JB_MEMBERS, CONNECTIONS
     WHERE JB_MEMBERS.MEMBER_ID = CONNECTIONS.MEMBER_ID;

select * from MEMBER_WITH_CONNECTIONS;
```

All Rows Fetched: 4 in 0.097 seconds

|  | MEMBER_ID | DETAILS |
|---|---|---|
| 1 | Member02 | REFERENCES |
| 2 | Member04 | REFERENCES |
| 3 | Member06 | REFERENCES |
| 4 | Member02 | REFERENCES |

```
CREATE VIEW ORGANIZATIONS_WITH_DEPARTMENTS AS
   SELECT RECR_ID, DEPART_NAME, NO_EMPLOYEES
   FROM RECRUITERS, DEPARTMENTS
   WHERE RECR_ID = ORG_ID;

select * from ORGANIZATIONS_WITH_DEPARTMENTS;
```



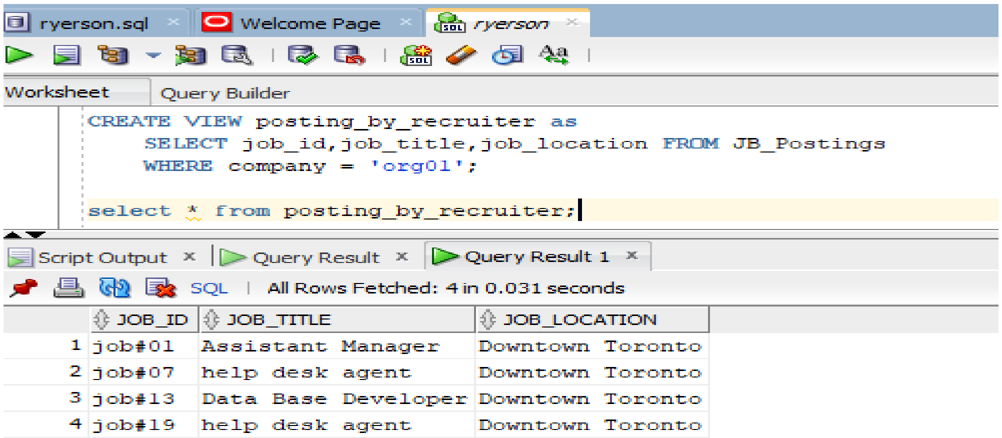
```
CREATE VIEW ORGANIZATIONS_WITH_DEPARTMENTS AS
     SELECT RECR_ID, DEPART_NAME, NO_EMPLOYEES
     FROM RECRUITERS, DEPARTMENTS
     WHERE RECR_ID = ORG_ID;

select * from ORGANIZATIONS_WITH_DEPARTMENTS;
```

All Rows Fetched: 18 in 0.028 seconds

| | RECR_ID | DEPART_NAME | NO_EMPLOYEES |
|---|---|---|---|
| 1 | Org01 | HR | 9 |
| 2 | Org02 | HR | 10 |
| 3 | Org03 | HR | 12 |
| 4 | Org04 | HR | 14 |
| 5 | Org05 | HR | 20 |
| 6 | Org06 | HR | 5 |
| 7 | Org01 | Administration | 10 |
| 8 | Org02 | Administration | 7 |
| 9 | Org03 | Administration | 4 |
| 10 | Org04 | Administration | 9 |
| 11 | Org05 | Administration | 6 |
| 12 | Org06 | Administration | 12 |
| 13 | Org01 | Accounting and Finance | 4 |
| 14 | Org02 | Accounting and Finance | 3 |
| 15 | Org03 | Accounting and Finance | 7 |
| 16 | Org04 | Accounting and Finance | 9 |
| 17 | Org05 | Accounting and Finance | 5 |

CREATE VIEW posting_by_recruiter as
SELECT job_id,job_title,job_location FROM JB_Postings
WHERE company = 'org01';

select * from posting_by_recruiter;



```
CREATE VIEW posting_by_recruiter as
    SELECT job_id,job_title,job_location FROM JB_Postings
    WHERE company = 'org01';

select * from posting_by_recruiter;
```

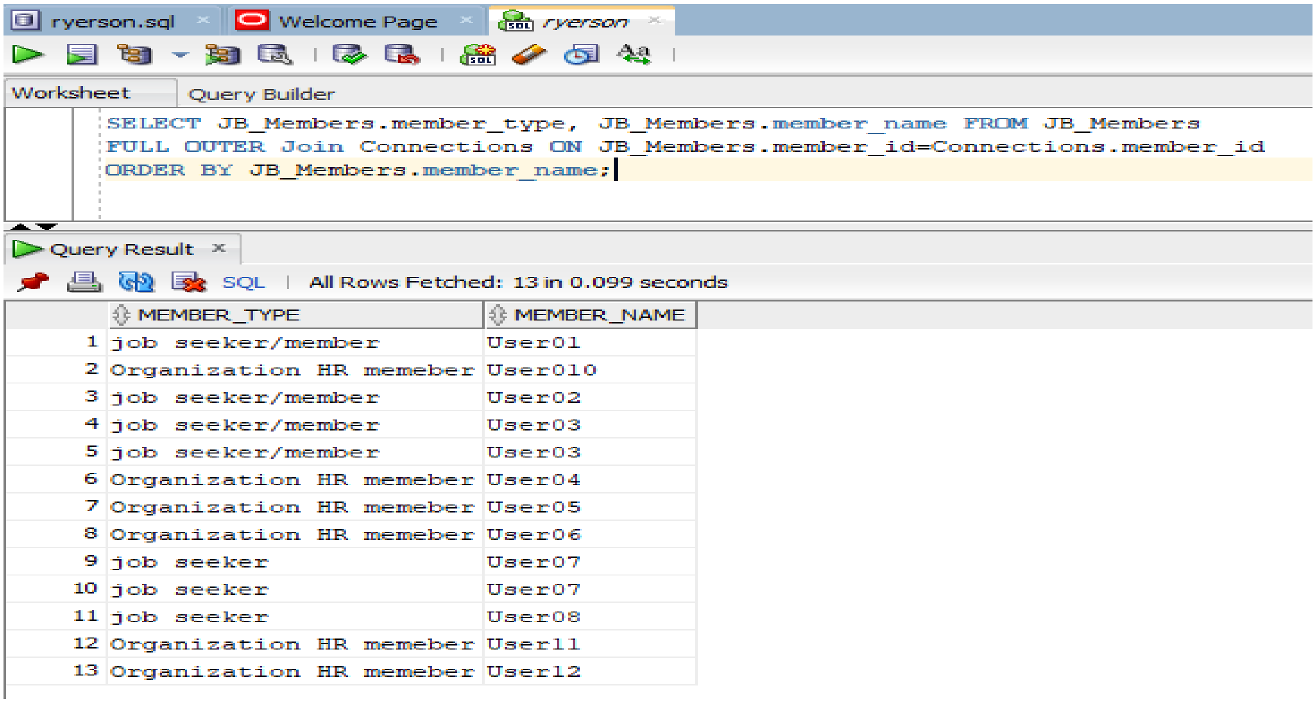
All Rows Fetched: 4 in 0.031 seconds

| | JOB_ID | JOB_TITLE | JOB_LOCATION |
|---|---|---|---|
| 1 | job#01 | Assistant Manager | Downtown Toronto |
| 2 | job#07 | help desk agent | Downtown Toronto |
| 3 | job#13 | Data Base Developer | Downtown Toronto |
| 4 | job#19 | help desk agent | Downtown Toronto |

**-- advanced Queries**

-- Join Queries

SELECT JB_Members.member_type, JB_Members.member_name FROM JB_Members
FULL OUTER Join Connections ON JB_Members.member_id=Connections.member_id
ORDER BY JB_Members.member_name;

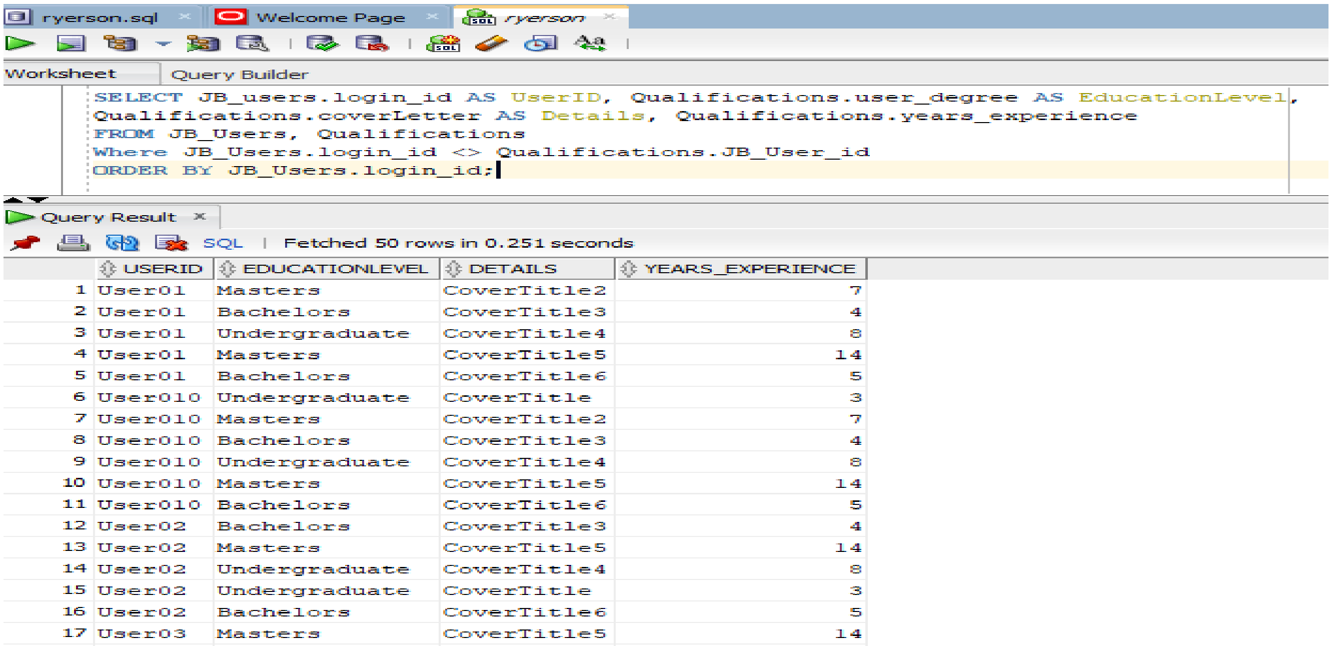


```
SELECT JB_Members.member_type, JB_Members.member_name FROM JB_Members
FULL OUTER Join Connections ON JB_Members.member_id=Connections.member_id
ORDER BY JB_Members.member_name;
```

All Rows Fetched: 13 in 0.099 seconds

|  | MEMBER_TYPE | MEMBER_NAME |
|---|---|---|
| 1 | job seeker/member | User01 |
| 2 | Organization HR memeber | User010 |
| 3 | job seeker/member | User02 |
| 4 | job seeker/member | User03 |
| 5 | job seeker/member | User03 |
| 6 | Organization HR memeber | User04 |
| 7 | Organization HR memeber | User05 |
| 8 | Organization HR memeber | User06 |
| 9 | job seeker | User07 |
| 10 | job seeker | User07 |
| 11 | job seeker | User08 |
| 12 | Organization HR memeber | User11 |
| 13 | Organization HR memeber | User12 |

SELECT JB_users.login_id AS UserID, Qualifications.user_degree AS EducationLevel,
Qualifications.coverLetter AS Details, Qualifications.years_experience
FROM JB_Users, Qualifications
Where JB_Users.login_id <> Qualifications.JB_User_id
ORDER BY JB_Users.login_id;



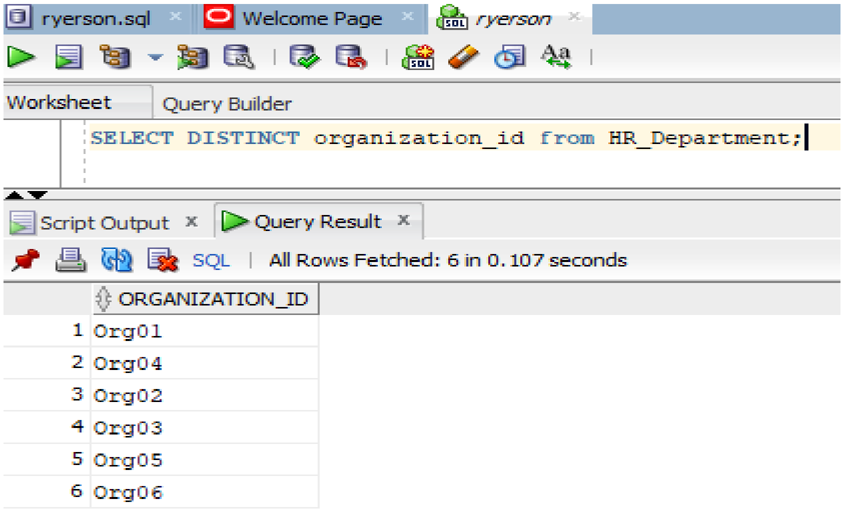
```
SELECT JB_users.login_id AS UserID, Qualifications.user_degree AS EducationLevel,
Qualifications.coverLetter AS Details, Qualifications.years_experience
FROM JB_Users, Qualifications
Where JB_Users.login_id <> Qualifications.JB_User_id
ORDER BY JB_Users.login_id;
```

Fetched 50 rows in 0.251 seconds

|  | USERID | EDUCATIONLEVEL | DETAILS | YEARS_EXPERIENCE |
|---|---|---|---|---|
| 1 | User01 | Masters | CoverTitle2 | 7 |
| 2 | User01 | Bachelors | CoverTitle3 | 4 |
| 3 | User01 | Undergraduate | CoverTitle4 | 8 |
| 4 | User01 | Masters | CoverTitle5 | 14 |
| 5 | User01 | Bachelors | CoverTitle6 | 5 |
| 6 | User010 | Undergraduate | CoverTitle | 3 |
| 7 | User010 | Masters | CoverTitle2 | 7 |
| 8 | User010 | Bachelors | CoverTitle3 | 4 |
| 9 | User010 | Undergraduate | CoverTitle4 | 8 |
| 10 | User010 | Masters | CoverTitle5 | 14 |
| 11 | User010 | Bachelors | CoverTitle6 | 5 |
| 12 | User02 | Bachelors | CoverTitle3 | 4 |
| 13 | User02 | Masters | CoverTitle5 | 14 |
| 14 | User02 | Undergraduate | CoverTitle4 | 8 |
| 15 | User02 | Undergraduate | CoverTitle | 3 |
| 16 | User02 | Bachelors | CoverTitle6 | 5 |
| 17 | User03 | Masters | CoverTitle5 | 14 |

--DISTINCT Queries

SELECT DISTINCT organization_id from HR_Department;

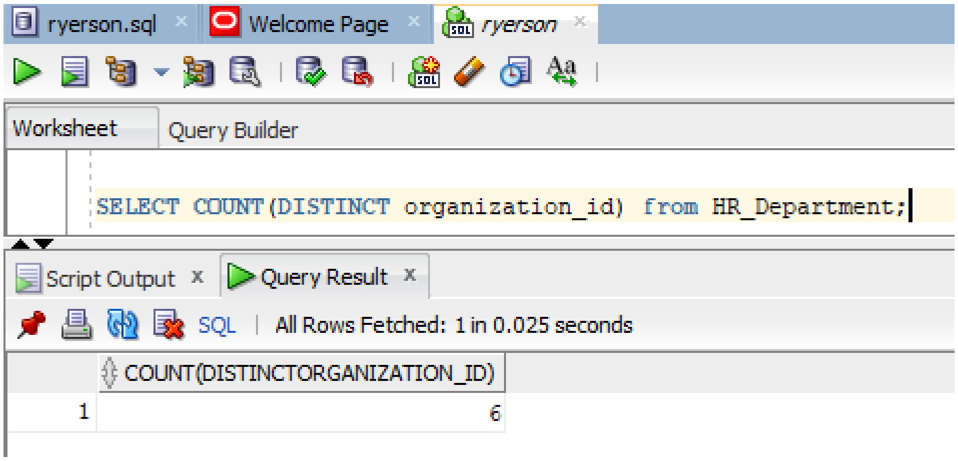


```
SELECT DISTINCT organization_id from HR_Department;
```

All Rows Fetched: 6 in 0.107 seconds

```
  ORGANIZATION_ID
1 Org01
2 Org04
3 Org02
4 Org03
5 Org05
6 Org06
```

SELECT COUNT(DISTINCT organization_id) from HR_Department;

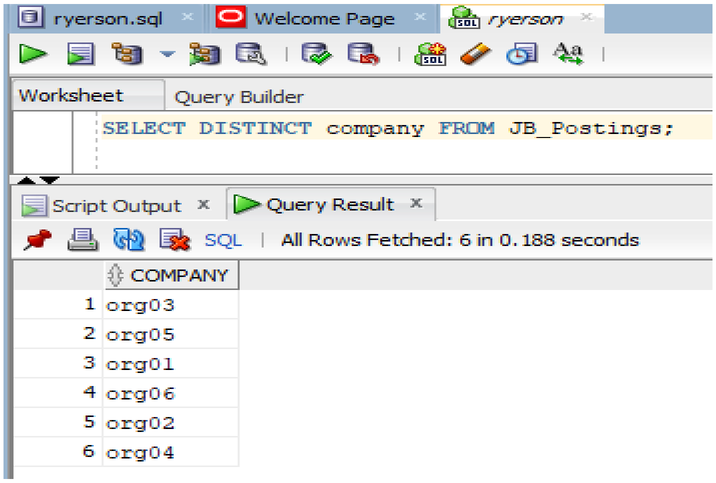


```
SELECT COUNT(DISTINCT organization_id) from HR_Department;
```

All Rows Fetched: 1 in 0.025 seconds

```
  COUNT(DISTINCTORGANIZATION_ID)
1                              6
```

SELECT DISTINCT company FROM JB_Postings;



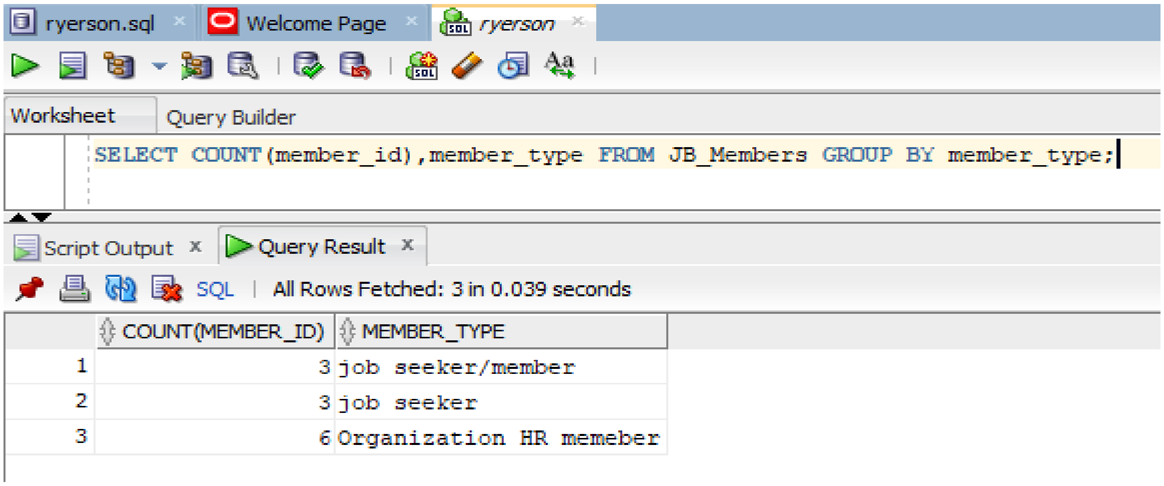
```
SELECT DISTINCT company FROM JB_Postings;
```

All Rows Fetched: 6 in 0.188 seconds

```
  COMPANY
1 org03
2 org05
3 org01
4 org06
5 org02
6 org04
```

--Grouping/Sorting Commands

SELECT COUNT(member_id),member_type FROM JB_Members GROUP BY member_type;

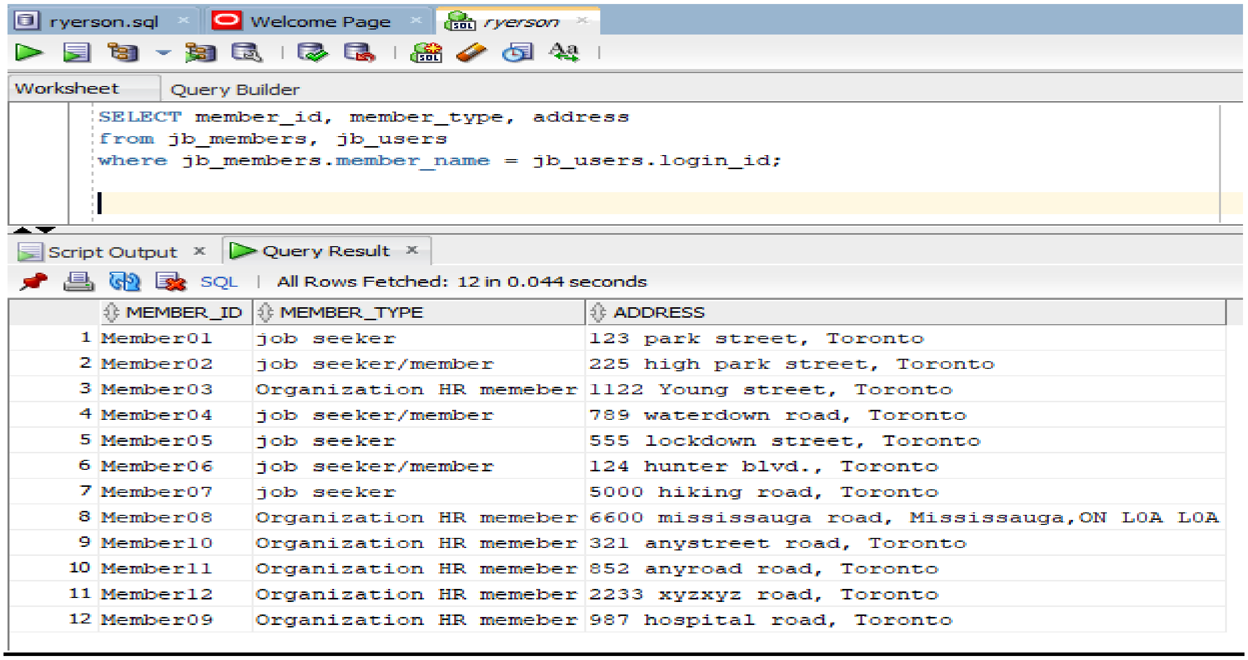


```
SELECT COUNT(member_id),member_type FROM JB_Members GROUP BY member_type;
```

All Rows Fetched: 3 in 0.039 seconds

| | COUNT(MEMBER_ID) | MEMBER_TYPE |
|---|---|---|
| 1 | 3 | job seeker/member |
| 2 | 3 | job seeker |
| 3 | 6 | Organization HR memeber |

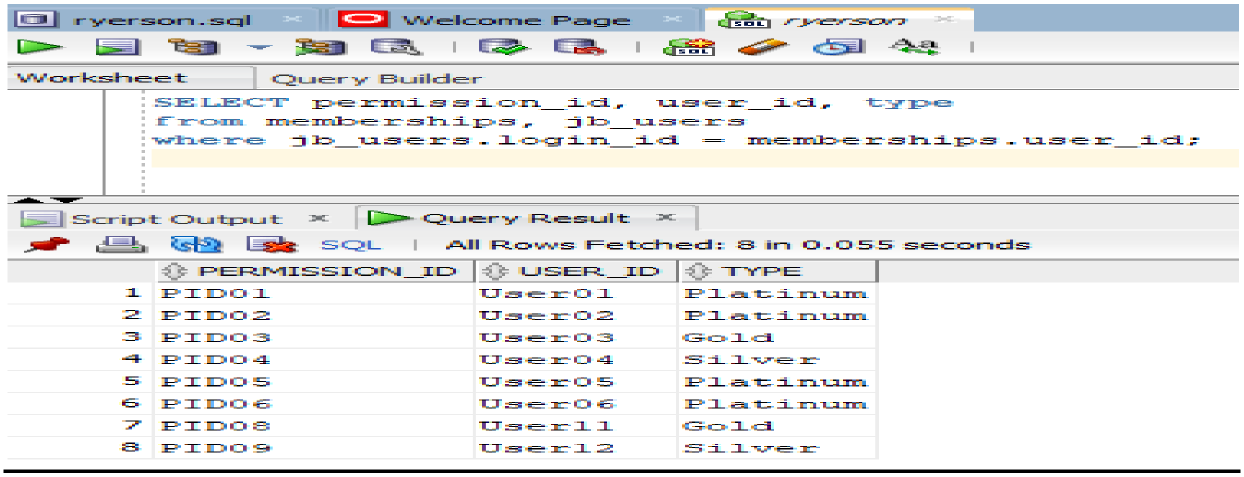
- Use of **Where** clause



```
SELECT member_id, member_type, address
from jb_members, jb_users
where jb_members.member_name = jb_users.login_id;
```

All Rows Fetched: 12 in 0.044 seconds

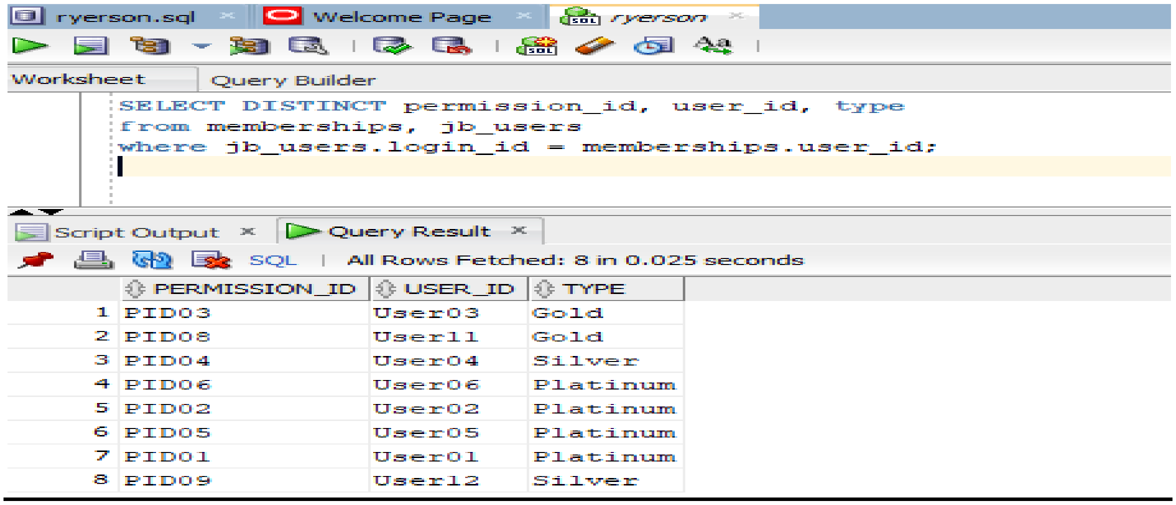
| | MEMBER_ID | MEMBER_TYPE | ADDRESS |
|---|---|---|---|
| 1 | Member01 | job seeker | 123 park street, Toronto |
| 2 | Member02 | job seeker/member | 225 high park street, Toronto |
| 3 | Member03 | Organization HR memeber | 1122 Young street, Toronto |
| 4 | Member04 | job seeker/member | 789 waterdown road, Toronto |
| 5 | Member05 | job seeker | 555 lockdown street, Toronto |
| 6 | Member06 | job seeker/member | 124 hunter blvd., Toronto |
| 7 | Member07 | job seeker | 5000 hiking road, Toronto |
| 8 | Member08 | Organization HR memeber | 6600 mississauga road, Mississauga,ON L0A L0A |
| 9 | Member10 | Organization HR memeber | 321 anystreet road, Toronto |
| 10 | Member11 | Organization HR memeber | 852 anyroad road, Toronto |
| 11 | Member12 | Organization HR memeber | 2233 xyzxyz road, Toronto |
| 12 | Member09 | Organization HR memeber | 987 hospital road, Toronto |



```
SELECT permission_id, user_id, type
from memberships, jb_users
where jb_users.login_id = memberships.user_id;
```

All Rows Fetched: 8 in 0.055 seconds

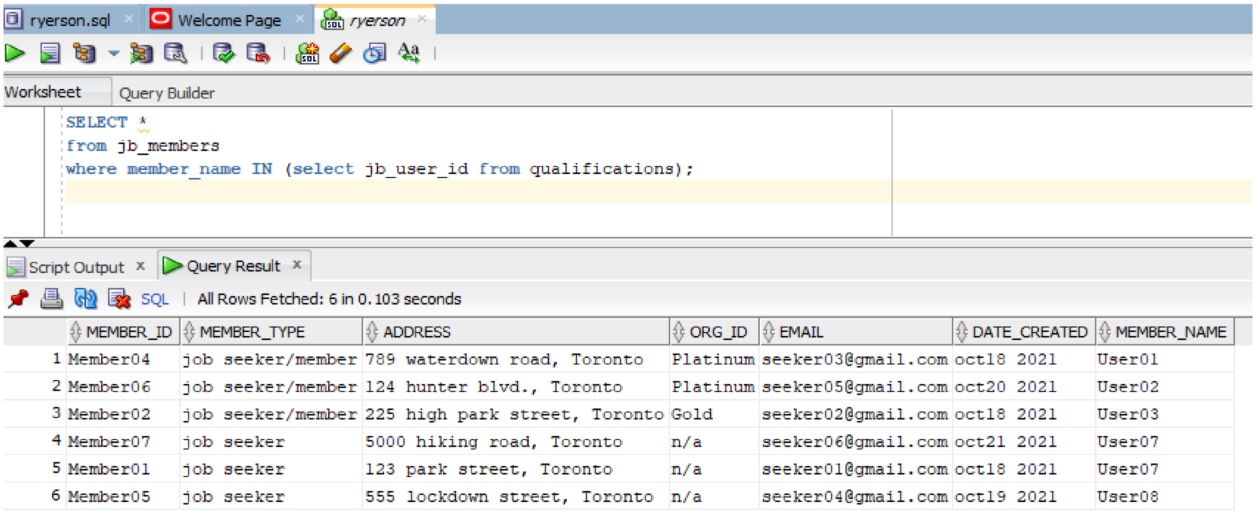
| | PERMISSION_ID | USER_ID | TYPE |
|---|---|---|---|
| 1 | PID01 | User01 | Platinum |
| 2 | PID02 | User02 | Platinum |
| 3 | PID03 | User03 | Gold |
| 4 | PID04 | User04 | Silver |
| 5 | PID05 | User05 | Platinum |
| 6 | PID06 | User06 | Platinum |
| 7 | PID08 | User11 | Gold |
| 8 | PID09 | User12 | Silver |



```sql
SELECT DISTINCT permission_id, user_id, type
from memberships, jb_users
where jb_users.login_id = memberships.user_id;
```

All Rows Fetched: 8 in 0.025 seconds

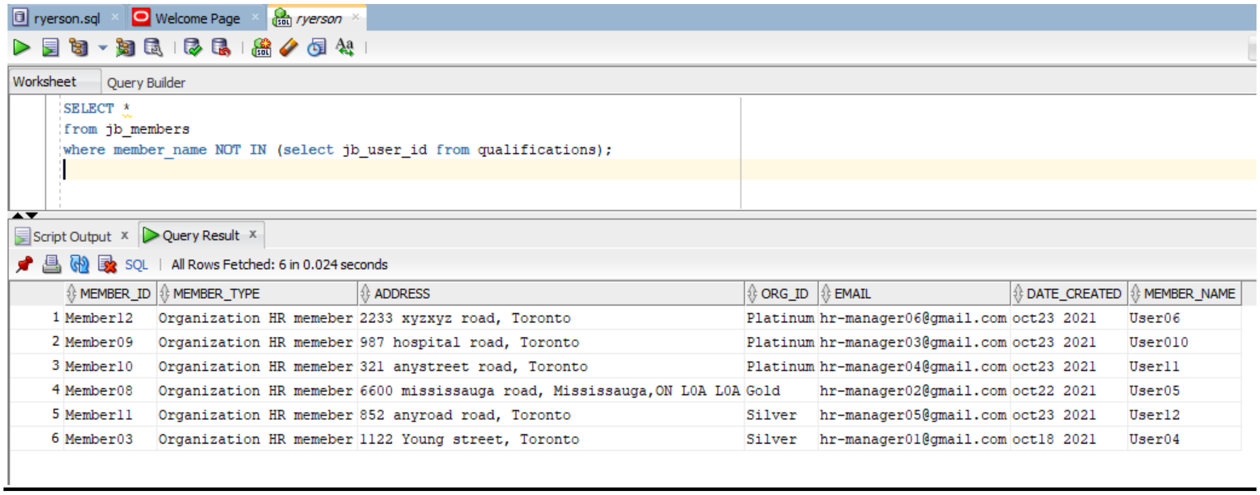
| | PERMISSION_ID | USER_ID | TYPE |
|---|---|---|---|
| 1 | PID03 | User03 | Gold |
| 2 | PID08 | User11 | Gold |
| 3 | PID04 | User04 | Silver |
| 4 | PID06 | User06 | Platinum |
| 5 | PID02 | User02 | Platinum |
| 6 | PID05 | User05 | Platinum |
| 7 | PID01 | User01 | Platinum |
| 8 | PID09 | User12 | Silver |



```sql
SELECT *
from jb_members
where member_name IN (select jb_user_id from qualifications);
```

All Rows Fetched: 6 in 0.103 seconds

| | MEMBER_ID | MEMBER_TYPE | ADDRESS | ORG_ID | EMAIL | DATE_CREATED | MEMBER_NAME |
|---|---|---|---|---|---|---|---|
| 1 | Member04 | job seeker/member | 789 waterdown road, Toronto | Platinum | seeker03@gmail.com | oct18 2021 | User01 |
| 2 | Member06 | job seeker/member | 124 hunter blvd., Toronto | Platinum | seeker05@gmail.com | oct20 2021 | User02 |
| 3 | Member02 | job seeker/member | 225 high park street, Toronto | Gold | seeker02@gmail.com | oct18 2021 | User03 |
| 4 | Member07 | job seeker | 5000 hiking road, Toronto | n/a | seeker06@gmail.com | oct21 2021 | User07 |
| 5 | Member01 | job seeker | 123 park street, Toronto | n/a | seeker01@gmail.com | oct18 2021 | User07 |
| 6 | Member05 | job seeker | 555 lockdown street, Toronto | n/a | seeker04@gmail.com | oct19 2021 | User08 |



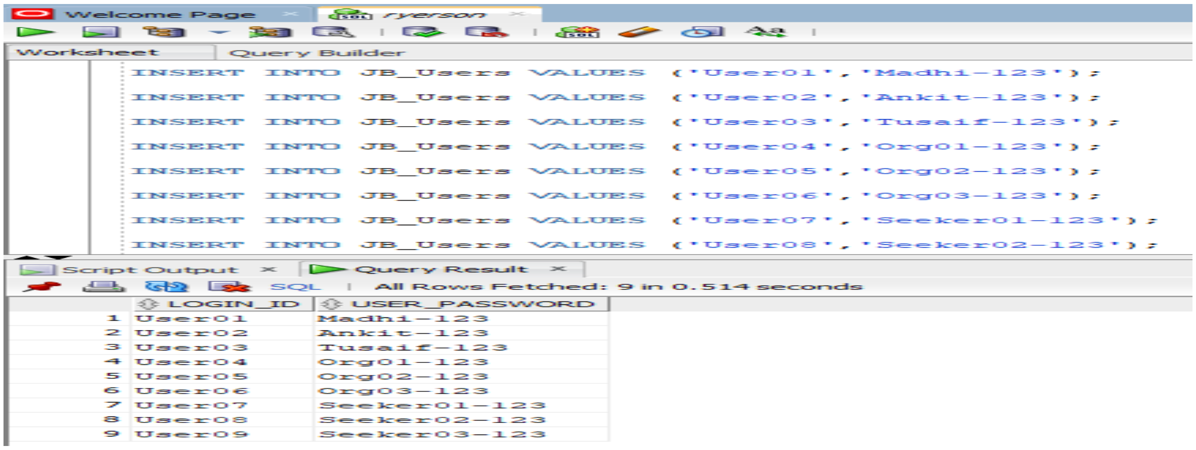
```sql
SELECT *
from jb_members
where member_name NOT IN (select jb_user_id from qualifications);
```

All Rows Fetched: 6 in 0.024 seconds

| | MEMBER_ID | MEMBER_TYPE | ADDRESS | ORG_ID | EMAIL | DATE_CREATED | MEMBER_NAME |
|---|---|---|---|---|---|---|---|
| 1 | Member12 | Organization HR memeber | 2233 xyzxyz road, Toronto | Platinum | hr-manager06@gmail.com | oct23 2021 | User06 |
| 2 | Member09 | Organization HR memeber | 987 hospital road, Toronto | Platinum | hr-manager03@gmail.com | oct23 2021 | User010 |
| 3 | Member10 | Organization HR memeber | 321 anystreet road, Toronto | Platinum | hr-manager04@gmail.com | oct23 2021 | User11 |
| 4 | Member08 | Organization HR memeber | 6600 mississauga road, Mississauga,ON L0A L0A | Gold | hr-manager02@gmail.com | oct22 2021 | User05 |
| 5 | Member11 | Organization HR memeber | 852 anyroad road, Toronto | Silver | hr-manager05@gmail.com | oct23 2021 | User12 |
| 6 | Member03 | Organization HR memeber | 1122 Young street, Toronto | Silver | hr-manager01@gmail.com | oct18 2021 | User04 |

## Screen Shots:

### 1. JB_Users Table Views and Queries.

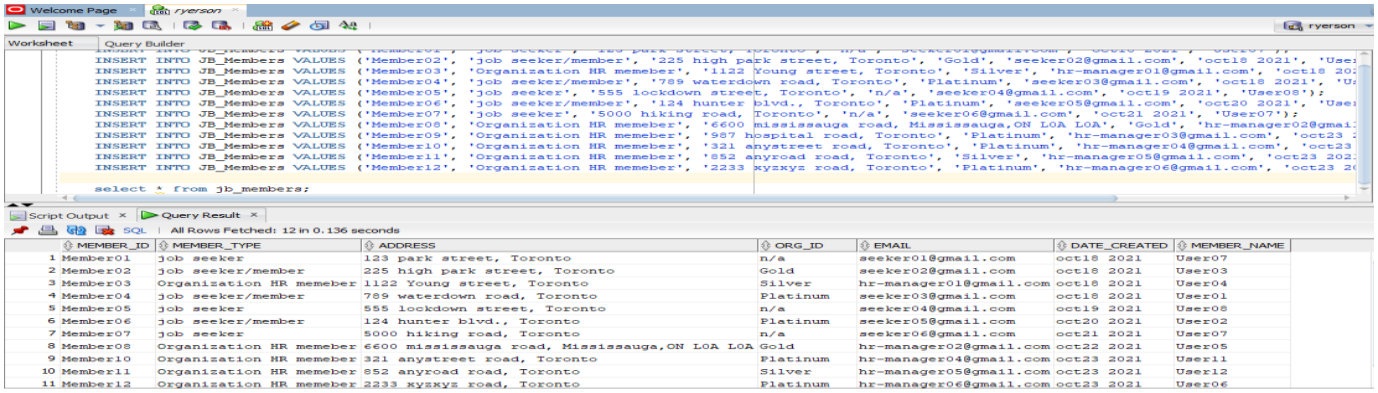


```
INSERT INTO JB_Users VALUES ('User01','Madhi-123');
INSERT INTO JB_Users VALUES ('User02','Ankit-123');
INSERT INTO JB_Users VALUES ('User03','Tusaif-123');
INSERT INTO JB_Users VALUES ('User04','Org01-123');
INSERT INTO JB_Users VALUES ('User05','Org02-123');
INSERT INTO JB_Users VALUES ('User06','Org03-123');
INSERT INTO JB_Users VALUES ('User07','Seeker01-123');
INSERT INTO JB_Users VALUES ('User08','Seeker02-123');
```

All Rows Fetched: 9 in 0.514 seconds

| | LOGIN_ID | USER_PASSWORD |
|---|---|---|
| 1 | User01 | Madhi-123 |
| 2 | User02 | Ankit-123 |
| 3 | User03 | Tusaif-123 |
| 4 | User04 | Org01-123 |
| 5 | User05 | Org02-123 |
| 6 | User06 | Org03-123 |
| 7 | User07 | Seeker01-123 |
| 8 | User08 | Seeker02-123 |
| 9 | User09 | Seeker03-123 |

## 2. JB_Members Table Views and Queries.

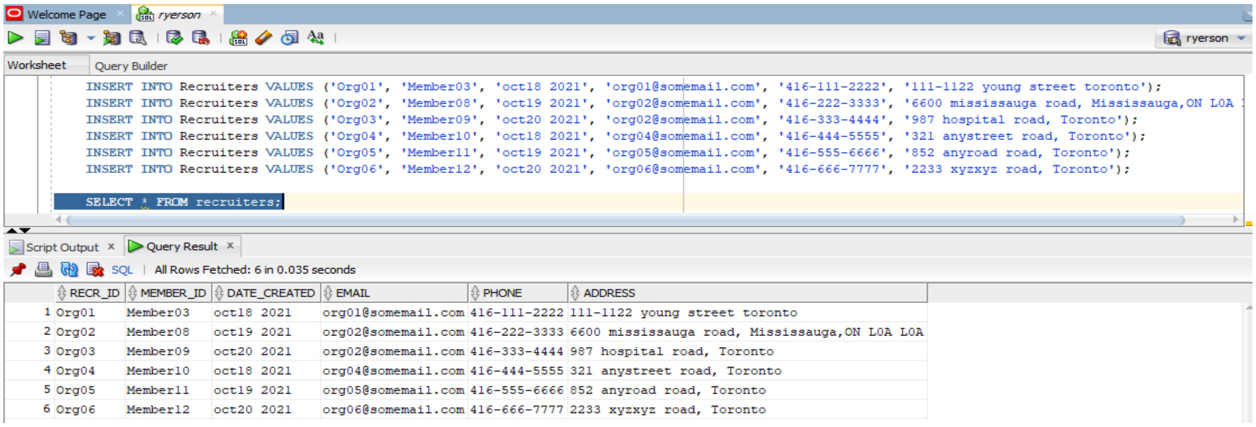


```
INSERT INTO JB_Members VALUES ('Member02', 'job seeker/member', '225 high park street, Toronto', 'Gold', 'seeker02@gmail.com', 'oct18 2021', 'User
INSERT INTO JB_Members VALUES ('Member03', 'Organization HR memeber', '1122 Young street, Toronto', 'Silver', 'hr-manager01@gmail.com', 'oct18 202
INSERT INTO JB_Members VALUES ('Member04', 'job seeker/member', '789 waterdown road, Toronto', 'Platinum', 'seeker03@gmail.com', 'oct18 2021', 'U
INSERT INTO JB_Members VALUES ('Member05', 'job seeker', '555 lockdown street, Toronto', 'n/a', 'seeker04@gmail.com', 'oct19 2021', 'User08');
INSERT INTO JB_Members VALUES ('Member06', 'job seeker/member', '124 hunter blvd., Toronto', 'Platinum', 'seeker05@gmail.com', 'oct20 2021', 'User
INSERT INTO JB_Members VALUES ('Member07', 'job seeker', '5000 hiking road, Toronto', 'n/a', 'seeker06@gmail.com', 'oct21 2021', 'User07');
INSERT INTO JB_Members VALUES ('Member08', 'Organization HR memeber', '6600 mississauga road, Mississauga,ON L0A L0A', 'Gold', 'hr-manager02@gmai
INSERT INTO JB_Members VALUES ('Member09', 'Organization HR memeber', '987 hospital road, Toronto', 'Platinum', 'hr-manager03@gmail.com', 'oct23 2
INSERT INTO JB_Members VALUES ('Member10', 'Organization HR memeber', '321 anystreet road, Toronto', 'Platinum', 'hr-manager04@gmail.com', 'oct23
INSERT INTO JB_Members VALUES ('Member11', 'Organization HR memeber', '852 anyroad road, Toronto', 'Silver', 'hr-manager05@gmail.com', 'oct23 202
INSERT INTO JB_Members VALUES ('Member12', 'Organization HR memeber', '2233 xyzxyz road, Toronto', 'Platinum', 'hr-manager06@gmail.com', 'oct23 20

select * from jb_members;
```

All Rows Fetched: 12 in 0.136 seconds

| | MEMBER_ID | MEMBER_TYPE | ADDRESS | ORG_ID | EMAIL | DATE_CREATED | MEMBER_NAME |
|---|---|---|---|---|---|---|---|
| 1 | Member01 | job seeker | 123 park street, Toronto | n/a | seeker01@gmail.com | oct18 2021 | User07 |
| 2 | Member02 | job seeker/member | 225 high park street, Toronto | Gold | seeker02@gmail.com | oct18 2021 | User03 |
| 3 | Member03 | Organization HR memeber | 1122 Young street, Toronto | Silver | hr-manager01@gmail.com | oct18 2021 | User04 |
| 4 | Member04 | job seeker/member | 789 waterdown road, Toronto | Platinum | seeker03@gmail.com | oct18 2021 | User01 |
| 5 | Member05 | job seeker | 555 lockdown street, Toronto | n/a | seeker04@gmail.com | oct19 2021 | User08 |
| 6 | Member06 | job seeker/member | 124 hunter blvd., Toronto | Platinum | seeker05@gmail.com | oct20 2021 | User02 |
| 7 | Member07 | job seeker | 5000 hiking road, Toronto | n/a | seeker06@gmail.com | oct21 2021 | User07 |
| 8 | Member08 | Organization HR memeber | 6600 mississauga road, Mississauga,ON L0A L0A | Gold | hr-manager02@gmail.com | oct22 2021 | User05 |
| 9 | Member10 | Organization HR memeber | 321 anystreet road, Toronto | Platinum | hr-manager04@gmail.com | oct23 2021 | User11 |
| 10 | Member11 | Organization HR memeber | 852 anyroad road, Toronto | Silver | hr-manager05@gmail.com | oct23 2021 | User12 |
| 11 | Member12 | Organization HR memeber | 2233 xyzxyz road, Toronto | Platinum | hr-manager06@gmail.com | oct23 2021 | User06 |

## 3. JB_Members Table Views and Queries.

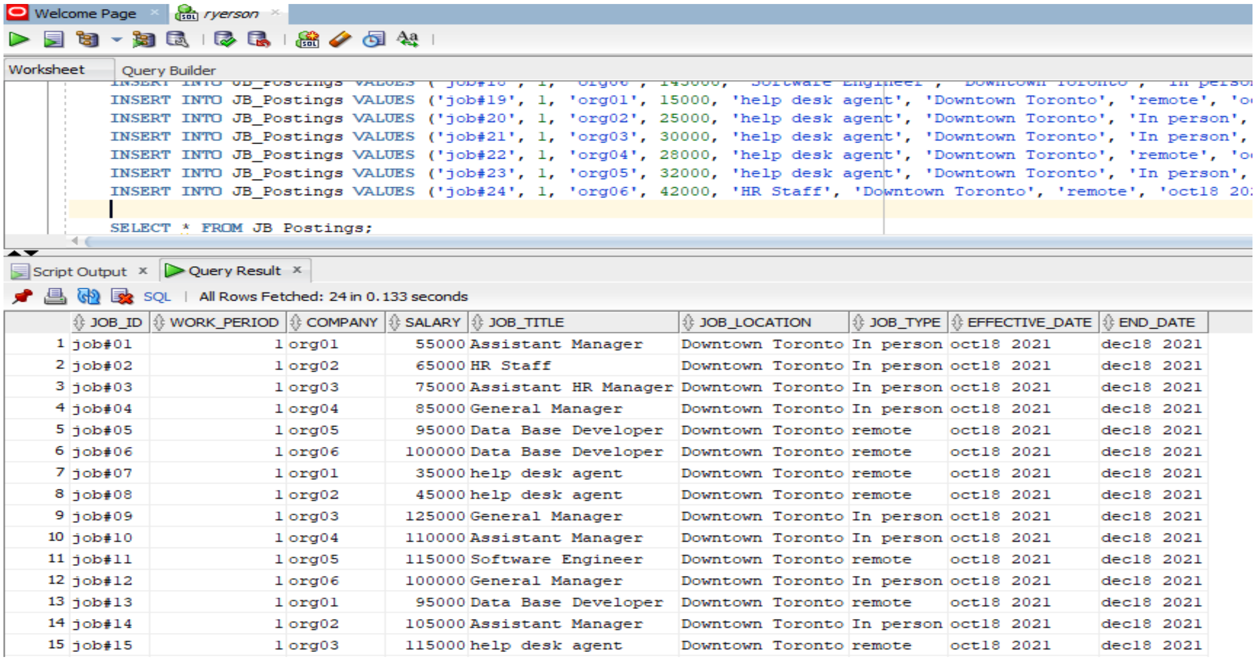


```
INSERT INTO Recruiters VALUES ('Org01', 'Member03', 'oct18 2021', 'org01@somemail.com', '416-111-2222', '111-1122 young street toronto');
INSERT INTO Recruiters VALUES ('Org02', 'Member08', 'oct19 2021', 'org02@somemail.com', '416-222-3333', '6600 mississauga road, Mississauga,ON L0A
INSERT INTO Recruiters VALUES ('Org03', 'Member09', 'oct20 2021', 'org02@somemail.com', '416-333-4444', '987 hospital road, Toronto');
INSERT INTO Recruiters VALUES ('Org04', 'Member10', 'oct18 2021', 'org04@somemail.com', '416-444-5555', '321 anystreet road, Toronto');
INSERT INTO Recruiters VALUES ('Org05', 'Member11', 'oct19 2021', 'org05@somemail.com', '416-555-6666', '852 anyroad road, Toronto');
INSERT INTO Recruiters VALUES ('Org06', 'Member12', 'oct20 2021', 'org06@somemail.com', '416-666-7777', '2233 xyzxyz road, Toronto');

SELECT * FROM recruiters;
```

All Rows Fetched: 6 in 0.035 seconds

| | RECR_ID | MEMBER_ID | DATE_CREATED | EMAIL | PHONE | ADDRESS |
|---|---|---|---|---|---|---|
| 1 | Org01 | Member03 | oct18 2021 | org01@somemail.com | 416-111-2222 | 111-1122 young street toronto |
| 2 | Org02 | Member08 | oct19 2021 | org02@somemail.com | 416-222-3333 | 6600 mississauga road, Mississauga,ON L0A L0A |
| 3 | Org03 | Member09 | oct20 2021 | org02@somemail.com | 416-333-4444 | 987 hospital road, Toronto |
| 4 | Org04 | Member10 | oct18 2021 | org04@somemail.com | 416-444-5555 | 321 anystreet road, Toronto |
| 5 | Org05 | Member11 | oct19 2021 | org05@somemail.com | 416-555-6666 | 852 anyroad road, Toronto |
| 6 | Org06 | Member12 | oct20 2021 | org06@somemail.com | 416-666-7777 | 2233 xyzxyz road, Toronto |

## 4. JB_Postings Table Views and Queries.

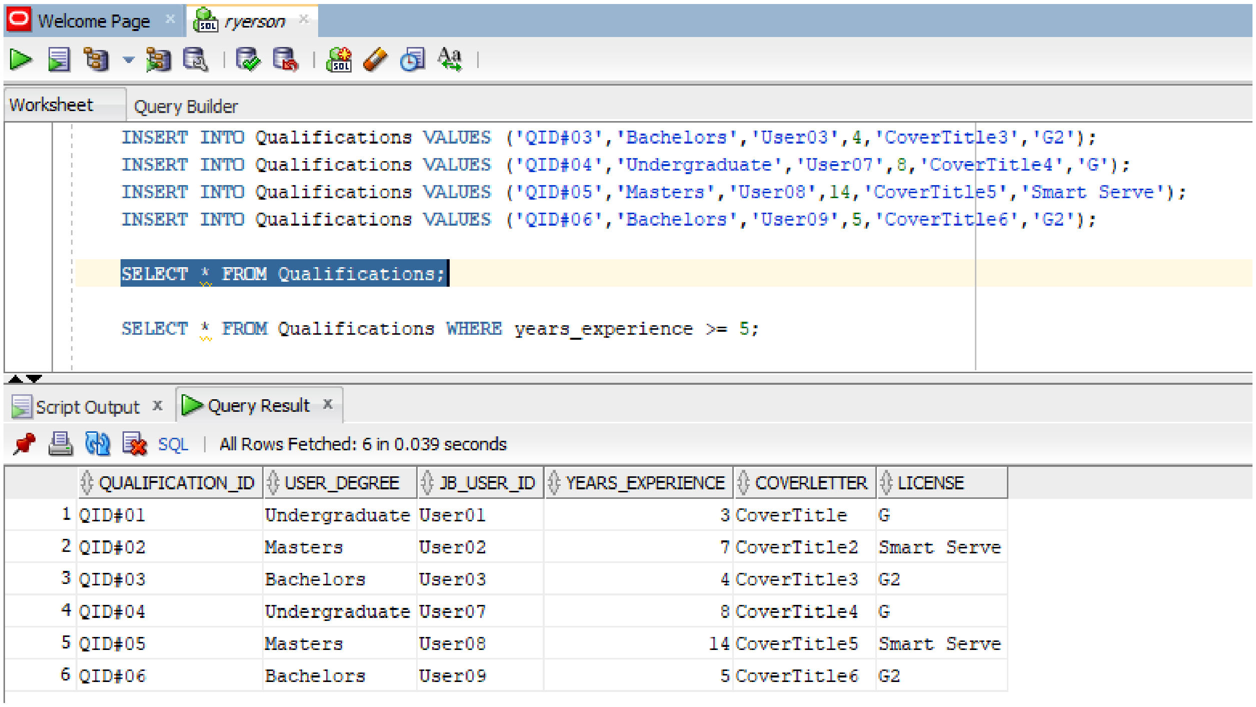


```
INSERT INTO JB_Postings VALUES ('job#18', 1, 'org06', 145000, 'Software Engineer', 'Downtown Toronto', 'In person
INSERT INTO JB_Postings VALUES ('job#19', 1, 'org01', 15000, 'help desk agent', 'Downtown Toronto', 'remote', 'o
INSERT INTO JB_Postings VALUES ('job#20', 1, 'org02', 25000, 'help desk agent', 'Downtown Toronto', 'In person',
INSERT INTO JB_Postings VALUES ('job#21', 1, 'org03', 30000, 'help desk agent', 'Downtown Toronto', 'In person',
INSERT INTO JB_Postings VALUES ('job#22', 1, 'org04', 28000, 'help desk agent', 'Downtown Toronto', 'remote', 'o
INSERT INTO JB_Postings VALUES ('job#23', 1, 'org05', 32000, 'help desk agent', 'Downtown Toronto', 'In person',
INSERT INTO JB_Postings VALUES ('job#24', 1, 'org06', 42000, 'HR Staff', 'Downtown Toronto', 'remote', 'oct18 20

SELECT * FROM JB Postings;
```

All Rows Fetched: 24 in 0.133 seconds

| | JOB_ID | WORK_PERIOD | COMPANY | SALARY | JOB_TITLE | JOB_LOCATION | JOB_TYPE | EFFECTIVE_DATE | END_DATE |
|---|---|---|---|---|---|---|---|---|---|
| 1 | job#01 | 1 | org01 | 55000 | Assistant Manager | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 2 | job#02 | 1 | org02 | 65000 | HR Staff | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 3 | job#03 | 1 | org03 | 75000 | Assistant HR Manager | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 4 | job#04 | 1 | org04 | 85000 | General Manager | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 5 | job#05 | 1 | org05 | 95000 | Data Base Developer | Downtown Toronto | remote | oct18 2021 | dec18 2021 |
| 6 | job#06 | 1 | org06 | 100000 | Data Base Developer | Downtown Toronto | remote | oct18 2021 | dec18 2021 |
| 7 | job#07 | 1 | org01 | 35000 | help desk agent | Downtown Toronto | remote | oct18 2021 | dec18 2021 |
| 8 | job#08 | 1 | org02 | 45000 | help desk agent | Downtown Toronto | remote | oct18 2021 | dec18 2021 |
| 9 | job#09 | 1 | org03 | 125000 | General Manager | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 10 | job#10 | 1 | org04 | 110000 | Assistant Manager | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 11 | job#11 | 1 | org05 | 115000 | Software Engineer | Downtown Toronto | remote | oct18 2021 | dec18 2021 |
| 12 | job#12 | 1 | org06 | 100000 | General Manager | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 13 | job#13 | 1 | org01 | 95000 | Data Base Developer | Downtown Toronto | remote | oct18 2021 | dec18 2021 |
| 14 | job#14 | 1 | org02 | 105000 | Assistant Manager | Downtown Toronto | In person | oct18 2021 | dec18 2021 |
| 15 | job#15 | 1 | org03 | 115000 | help desk agent | Downtown Toronto | remote | oct18 2021 | dec18 2021 |

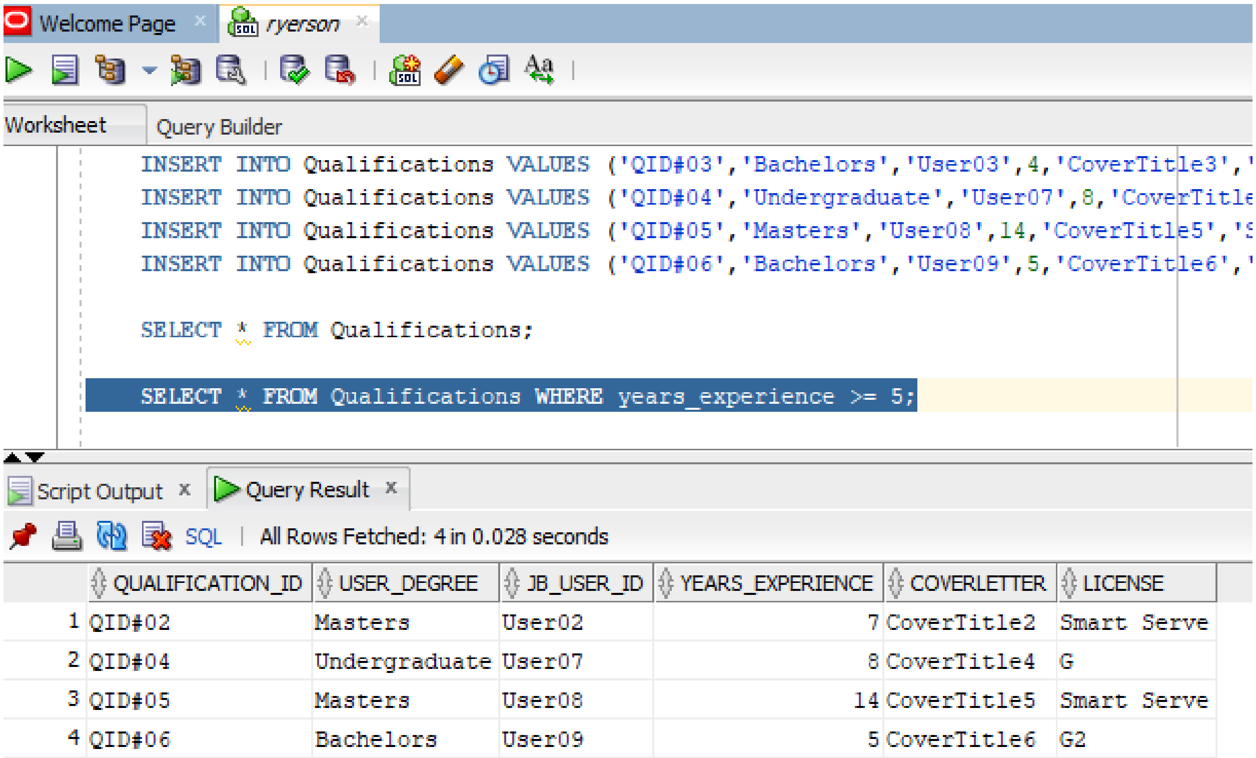
## 5. Qualifications Table Views and Queries.



```
INSERT INTO Qualifications VALUES ('QID#03','Bachelors','User03',4,'CoverTitle3','G2');
INSERT INTO Qualifications VALUES ('QID#04','Undergraduate','User07',8,'CoverTitle4','G');
INSERT INTO Qualifications VALUES ('QID#05','Masters','User08',14,'CoverTitle5','Smart Serve');
INSERT INTO Qualifications VALUES ('QID#06','Bachelors','User09',5,'CoverTitle6','G2');

SELECT * FROM Qualifications;

SELECT * FROM Qualifications WHERE years_experience >= 5;
```

All Rows Fetched: 6 in 0.039 seconds

| | QUALIFICATION_ID | USER_DEGREE | JB_USER_ID | YEARS_EXPERIENCE | COVERLETTER | LICENSE |
|---|---|---|---|---|---|---|
| 1 | QID#01 | Undergraduate | User01 | 3 | CoverTitle | G |
| 2 | QID#02 | Masters | User02 | 7 | CoverTitle2 | Smart Serve |
| 3 | QID#03 | Bachelors | User03 | 4 | CoverTitle3 | G2 |
| 4 | QID#04 | Undergraduate | User07 | 8 | CoverTitle4 | G |
| 5 | QID#05 | Masters | User08 | 14 | CoverTitle5 | Smart Serve |
| 6 | QID#06 | Bachelors | User09 | 5 | CoverTitle6 | G2 |

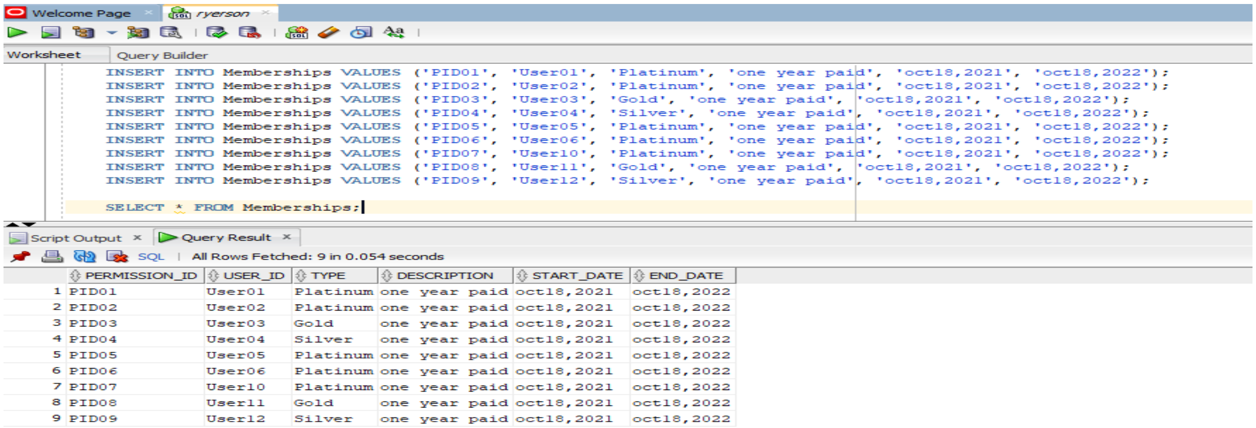


```
INSERT INTO Qualifications VALUES ('QID#03','Bachelors','User03',4,'CoverTitle3','
INSERT INTO Qualifications VALUES ('QID#04','Undergraduate','User07',8,'CoverTitle
INSERT INTO Qualifications VALUES ('QID#05','Masters','User08',14,'CoverTitle5','S
INSERT INTO Qualifications VALUES ('QID#06','Bachelors','User09',5,'CoverTitle6','

SELECT * FROM Qualifications;

SELECT * FROM Qualifications WHERE years_experience >= 5;
```

All Rows Fetched: 4 in 0.028 seconds

|   | QUALIFICATION_ID | USER_DEGREE | JB_USER_ID | YEARS_EXPERIENCE | COVERLETTER | LICENSE |
|---|---|---|---|---|---|---|
| 1 | QID#02 | Masters | User02 | 7 | CoverTitle2 | Smart Serve |
| 2 | QID#04 | Undergraduate | User07 | 8 | CoverTitle4 | G |
| 3 | QID#05 | Masters | User08 | 14 | CoverTitle5 | Smart Serve |
| 4 | QID#06 | Bachelors | User09 | 5 | CoverTitle6 | G2 |

## 6. Memberships Table Views and Queries.

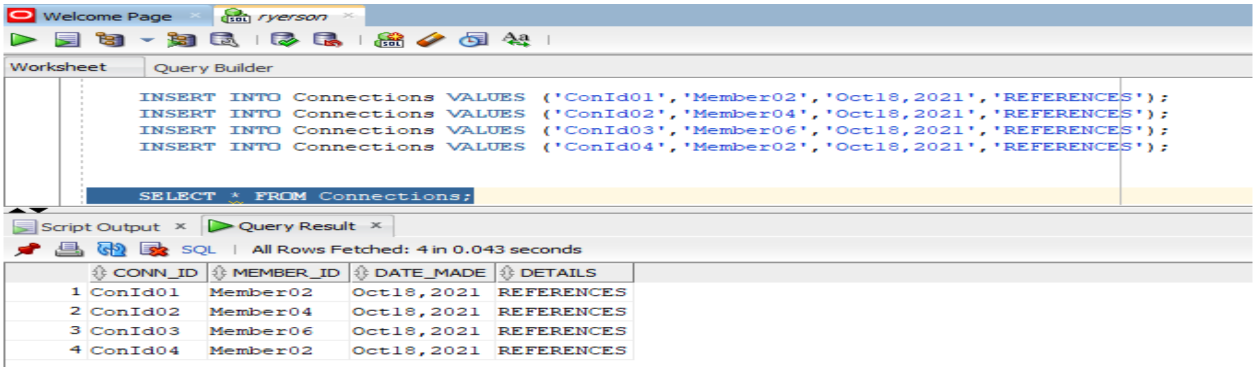


```
INSERT INTO Memberships VALUES ('PID01', 'User01', 'Platinum', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID02', 'User02', 'Platinum', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID03', 'User03', 'Gold', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID04', 'User04', 'Silver', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID05', 'User05', 'Platinum', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID06', 'User06', 'Platinum', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID07', 'User10', 'Platinum', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID08', 'User11', 'Gold', 'one year paid', 'oct18,2021', 'oct18,2022');
INSERT INTO Memberships VALUES ('PID09', 'User12', 'Silver', 'one year paid', 'oct18,2021', 'oct18,2022');

SELECT * FROM Memberships;
```

All Rows Fetched: 9 in 0.054 seconds

|   | PERMISSION_ID | USER_ID | TYPE | DESCRIPTION | START_DATE | END_DATE |
|---|---|---|---|---|---|---|
| 1 | PID01 | User01 | Platinum | one year paid | oct18,2021 | oct18,2022 |
| 2 | PID02 | User02 | Platinum | one year paid | oct18,2021 | oct18,2022 |
| 3 | PID03 | User03 | Gold | one year paid | oct18,2021 | oct18,2022 |
| 4 | PID04 | User04 | Silver | one year paid | oct18,2021 | oct18,2022 |
| 5 | PID05 | User05 | Platinum | one year paid | oct18,2021 | oct18,2022 |
| 6 | PID06 | User06 | Platinum | one year paid | oct18,2021 | oct18,2022 |
| 7 | PID07 | User10 | Platinum | one year paid | oct18,2021 | oct18,2022 |
| 8 | PID08 | User11 | Gold | one year paid | oct18,2021 | oct18,2022 |
| 9 | PID09 | User12 | Silver | one year paid | oct18,2021 | oct18,2022 |

## 7. Connections Table Views and Queries.

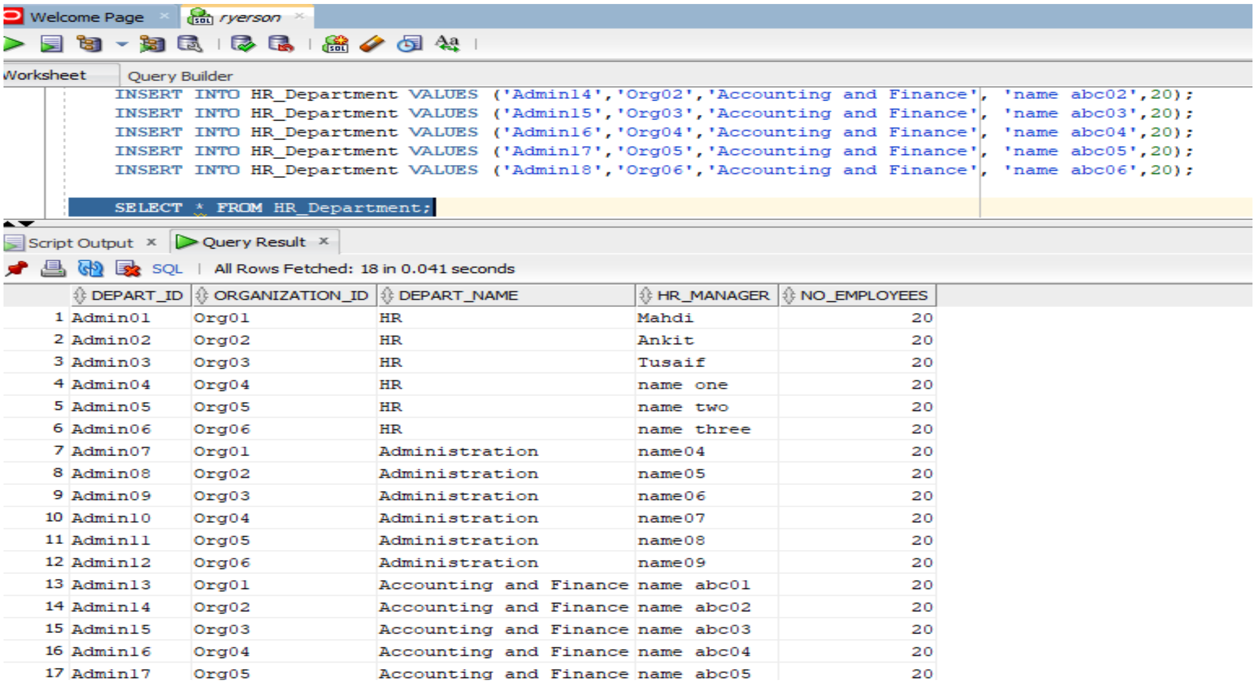


```sql
INSERT INTO Connections VALUES ('ConId01','Member02','Oct18,2021','REFERENCES');
INSERT INTO Connections VALUES ('ConId02','Member04','Oct18,2021','REFERENCES');
INSERT INTO Connections VALUES ('ConId03','Member06','Oct18,2021','REFERENCES');
INSERT INTO Connections VALUES ('ConId04','Member02','Oct18,2021','REFERENCES');


SELECT * FROM Connections;
```

All Rows Fetched: 4 in 0.043 seconds

| | CONN_ID | MEMBER_ID | DATE_MADE | DETAILS |
|---|---|---|---|---|
| 1 | ConId01 | Member02 | Oct18,2021 | REFERENCES |
| 2 | ConId02 | Member04 | Oct18,2021 | REFERENCES |
| 3 | ConId03 | Member06 | Oct18,2021 | REFERENCES |
| 4 | ConId04 | Member02 | Oct18,2021 | REFERENCES |

## HR_Department Table Views and Queries.



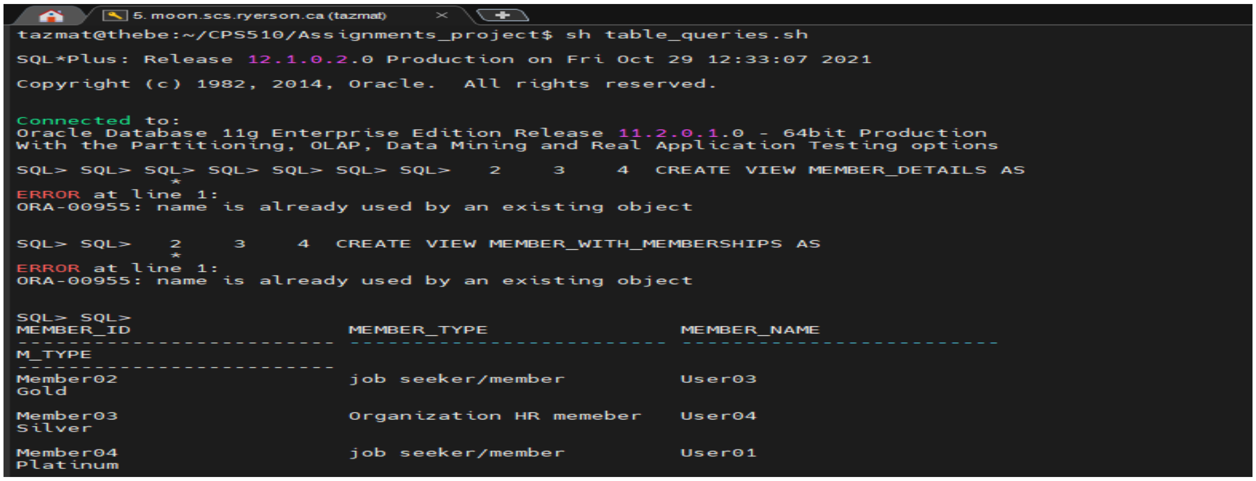
```sql
INSERT INTO HR_Department VALUES ('Admin14','Org02','Accounting and Finance', 'name abc02',20);
INSERT INTO HR_Department VALUES ('Admin15','Org03','Accounting and Finance', 'name abc03',20);
INSERT INTO HR_Department VALUES ('Admin16','Org04','Accounting and Finance', 'name abc04',20);
INSERT INTO HR_Department VALUES ('Admin17','Org05','Accounting and Finance', 'name abc05',20);
INSERT INTO HR_Department VALUES ('Admin18','Org06','Accounting and Finance', 'name abc06',20);

SELECT * FROM HR_Department;
```

All Rows Fetched: 18 in 0.041 seconds

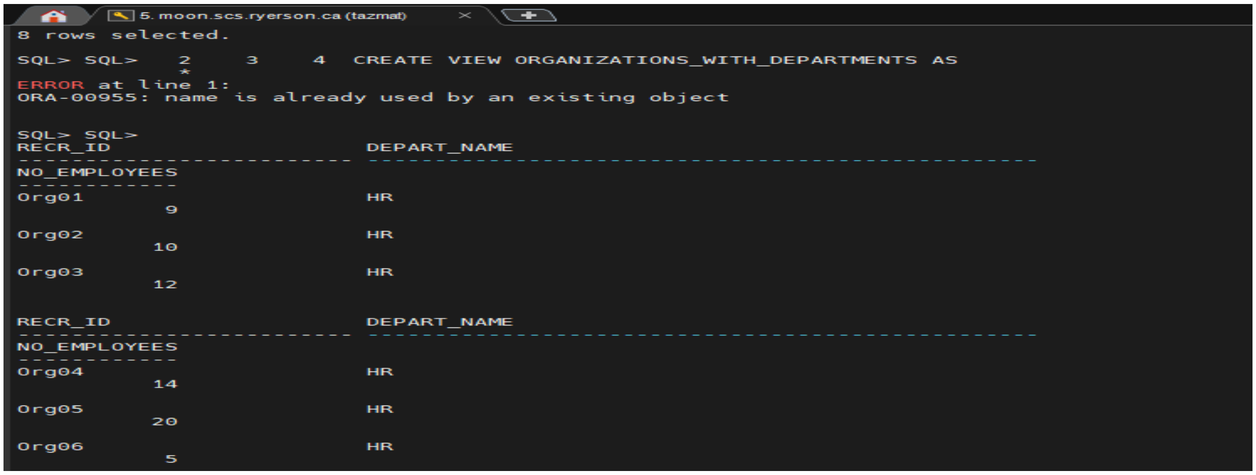
| | DEPART_ID | ORGANIZATION_ID | DEPART_NAME | HR_MANAGER | NO_EMPLOYEES |
|---|---|---|---|---|---|
| 1 | Admin01 | Org01 | HR | Mahdi | 20 |
| 2 | Admin02 | Org02 | HR | Ankit | 20 |
| 3 | Admin03 | Org03 | HR | Tusaif | 20 |
| 4 | Admin04 | Org04 | HR | name one | 20 |
| 5 | Admin05 | Org05 | HR | name two | 20 |
| 6 | Admin06 | Org06 | HR | name three | 20 |
| 7 | Admin07 | Org01 | Administration | name04 | 20 |
| 8 | Admin08 | Org02 | Administration | name05 | 20 |
| 9 | Admin09 | Org03 | Administration | name06 | 20 |
| 10 | Admin10 | Org04 | Administration | name07 | 20 |
| 11 | Admin11 | Org05 | Administration | name08 | 20 |
| 12 | Admin12 | Org06 | Administration | name09 | 20 |
| 13 | Admin13 | Org01 | Accounting and Finance | name abc01 | 20 |
| 14 | Admin14 | Org02 | Accounting and Finance | name abc02 | 20 |
| 15 | Admin15 | Org03 | Accounting and Finance | name abc03 | 20 |
| 16 | Admin16 | Org04 | Accounting and Finance | name abc04 | 20 |
| 17 | Admin17 | Org05 | Accounting and Finance | name abc05 | 20 |

# Shell Script

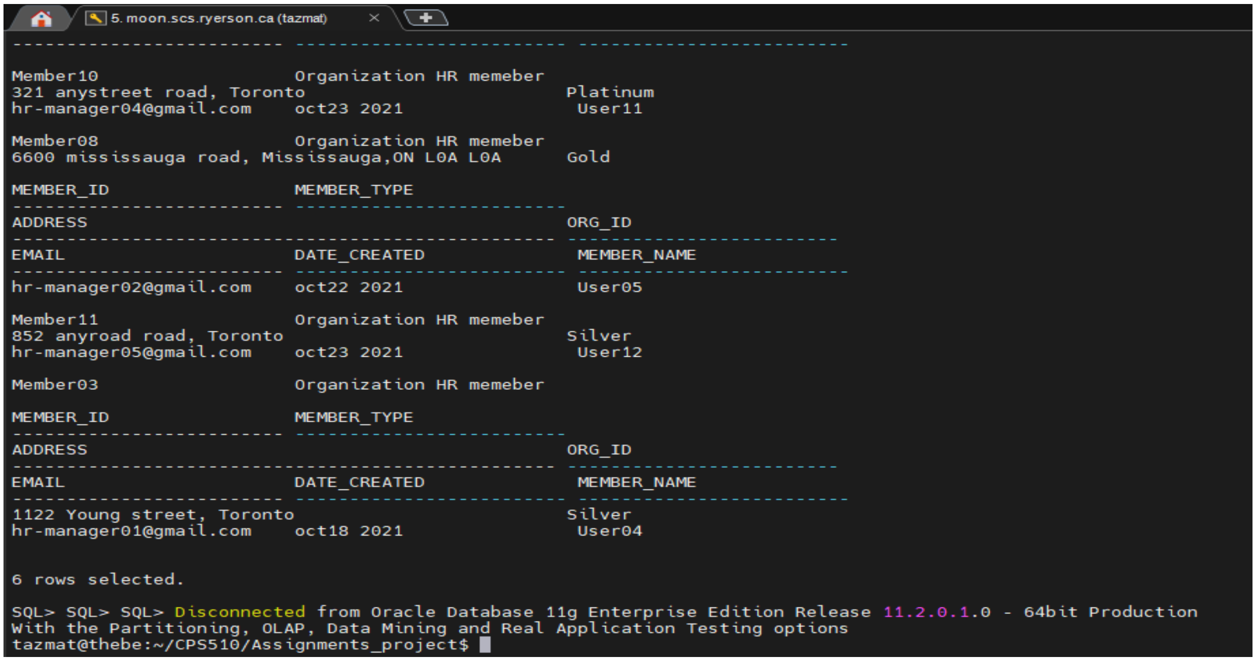
-- table_queries.sh



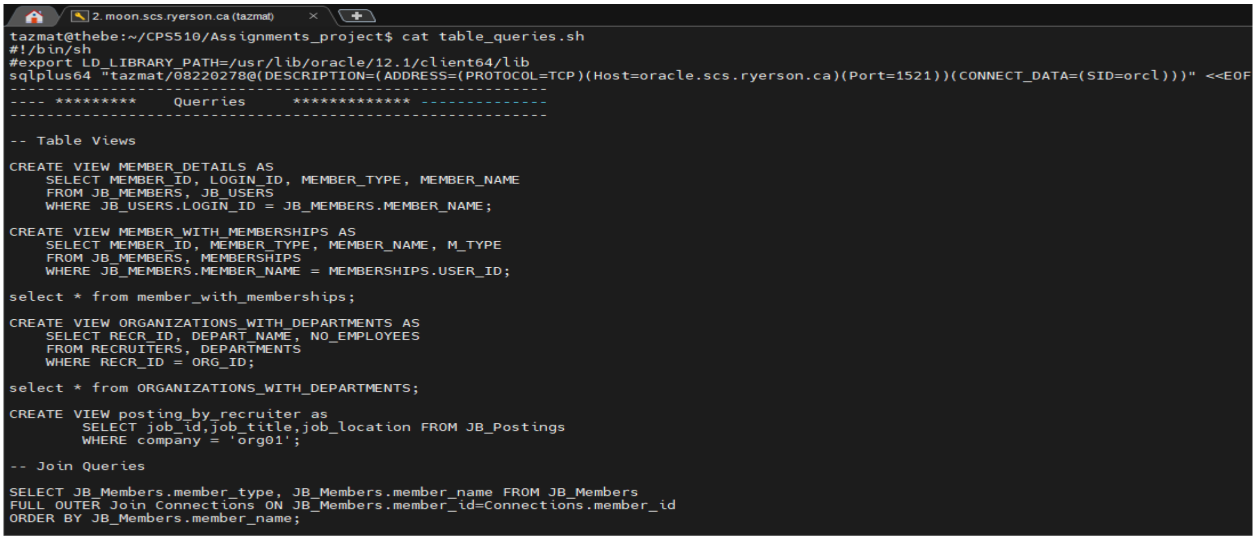
```
tazmat@thebe:~/CPS510/Assignments_project$ sh table_queries.sh
SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 29 12:33:07 2021
Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> SQL> SQL> SQL> SQL> SQL>   2    3    4  CREATE VIEW MEMBER_DETAILS AS
                                 *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> SQL>   2    3    4  CREATE VIEW MEMBER_WITH_MEMBERSHIPS AS
                                 *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> SQL>
MEMBER_ID                 MEMBER_TYPE               MEMBER_NAME
------------------------- ------------------------- -------------------------
M_TYPE
-------------------------
Member02                  job seeker/member         User03
Gold

Member03                  Organization HR memeber   User04
Silver

Member04                  job seeker/member         User01
Platinum
```



```
8 rows selected.

SQL> SQL>   2    3    4  CREATE VIEW ORGANIZATIONS_WITH_DEPARTMENTS AS
                *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> SQL>
RECR_ID                   DEPART_NAME
------------------------- --------------------------------------------------
NO_EMPLOYEES
------------
Org01                     HR
           9

Org02                     HR
          10

Org03                     HR
          12


RECR_ID                   DEPART_NAME
------------------------- --------------------------------------------------
NO_EMPLOYEES
------------
Org04                     HR
          14

Org05                     HR
          20

Org06                     HR
           5
```

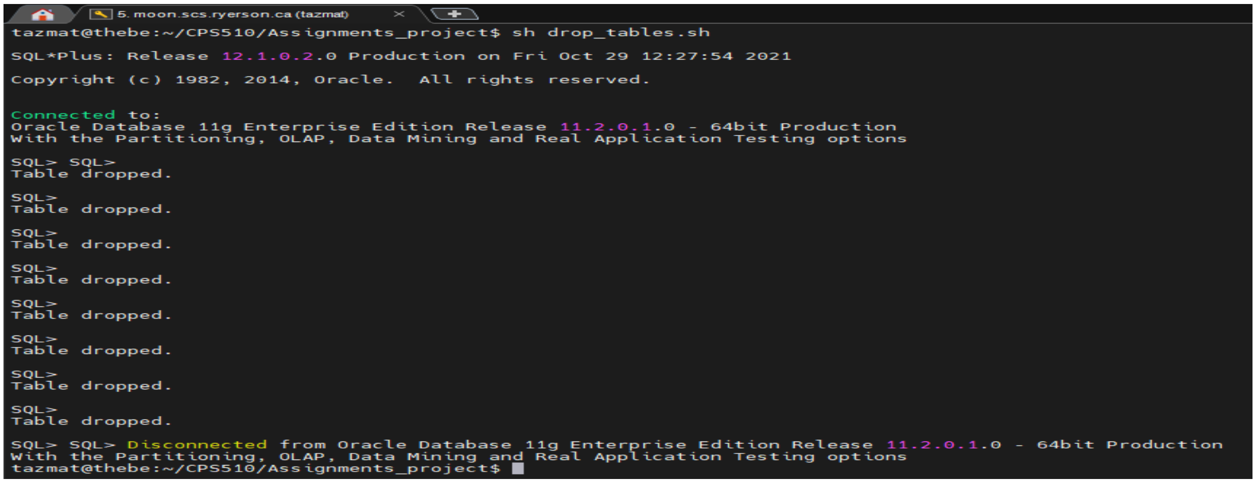


```
------------------------- ------------------------- -------------------------

Member10                  Organization HR memeber
321 anystreet road, Toronto                          Platinum
hr-manager04@gmail.com    oct23 2021                  User11

Member08                  Organization HR memeber
6600 mississauga road, Mississauga,ON L0A L0A       Gold

MEMBER_ID                 MEMBER_TYPE
------------------------- -------------------------
ADDRESS                                             ORG_ID
--------------------------------------------------  -------------------------
EMAIL                     DATE_CREATED              MEMBER_NAME
------------------------- ------------------------- -------------------------
hr-manager02@gmail.com    oct22 2021                User05

Member11                  Organization HR memeber
852 anyroad road, Toronto                          Silver
hr-manager05@gmail.com    oct23 2021                User12

Member03                  Organization HR memeber

MEMBER_ID                 MEMBER_TYPE
------------------------- -------------------------
ADDRESS                                             ORG_ID
--------------------------------------------------  -------------------------
EMAIL                     DATE_CREATED              MEMBER_NAME
------------------------- ------------------------- -------------------------
1122 Young street, Toronto                          Silver
hr-manager01@gmail.com    oct18 2021                User04


6 rows selected.

SQL> SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
tazmat@thebe:~/CPS510/Assignments_project$
```



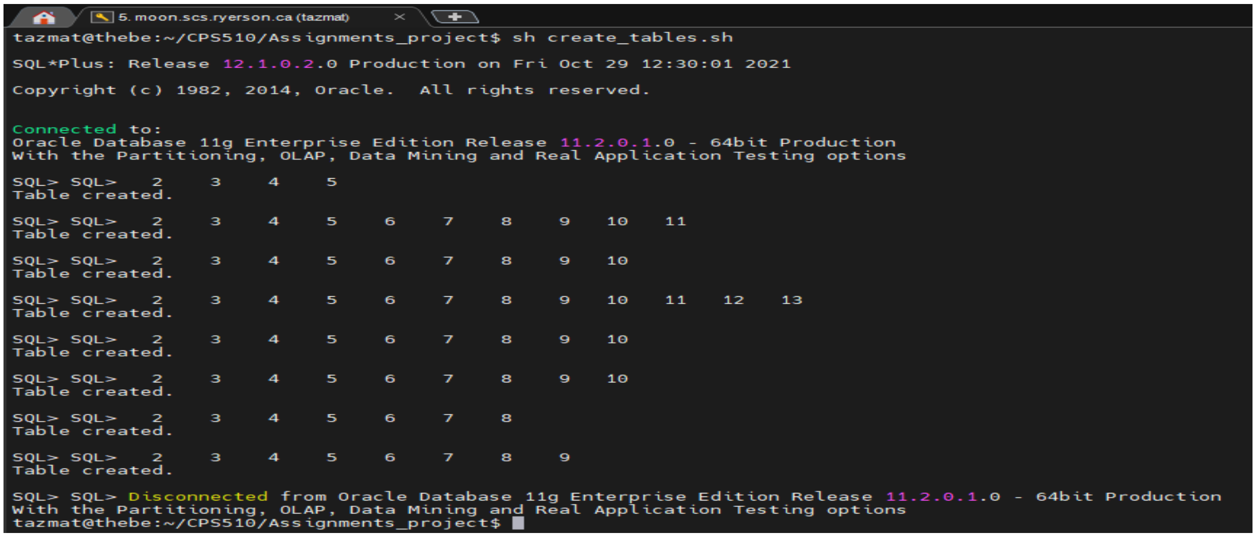
```
tazmat@thebe:~/CPS510/Assignments_project$ cat table_queries.sh
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "tazmat/08220278@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
------------------------------------------------------------
---- *********    Querries     ************* --------------
------------------------------------------------------------

-- Table Views

CREATE VIEW MEMBER_DETAILS AS
    SELECT MEMBER_ID, LOGIN_ID, MEMBER_TYPE, MEMBER_NAME
    FROM JB_MEMBERS, JB_USERS
    WHERE JB_USERS.LOGIN_ID = JB_MEMBERS.MEMBER_NAME;

CREATE VIEW MEMBER_WITH_MEMBERSHIPS AS
    SELECT MEMBER_ID, MEMBER_TYPE, MEMBER_NAME, M_TYPE
    FROM JB_MEMBERS, MEMBERSHIPS
    WHERE JB_MEMBERS.MEMBER_NAME = MEMBERSHIPS.USER_ID;

select * from member_with_memberships;

CREATE VIEW ORGANIZATIONS_WITH_DEPARTMENTS AS
    SELECT RECR_ID, DEPART_NAME, NO_EMPLOYEES
    FROM RECRUITERS, DEPARTMENTS
    WHERE RECR_ID = ORG_ID;

select * from ORGANIZATIONS_WITH_DEPARTMENTS;

CREATE VIEW posting_by_recruiter as
        SELECT job_id,job_title,job_location FROM JB_Postings
        WHERE company = 'org01';

-- Join Queries

SELECT JB_Members.member_type, JB_Members.member_name FROM JB_Members
FULL OUTER Join Connections ON JB_Members.member_id=Connections.member_id
ORDER BY JB_Members.member_name;
```

- **Drop tables**



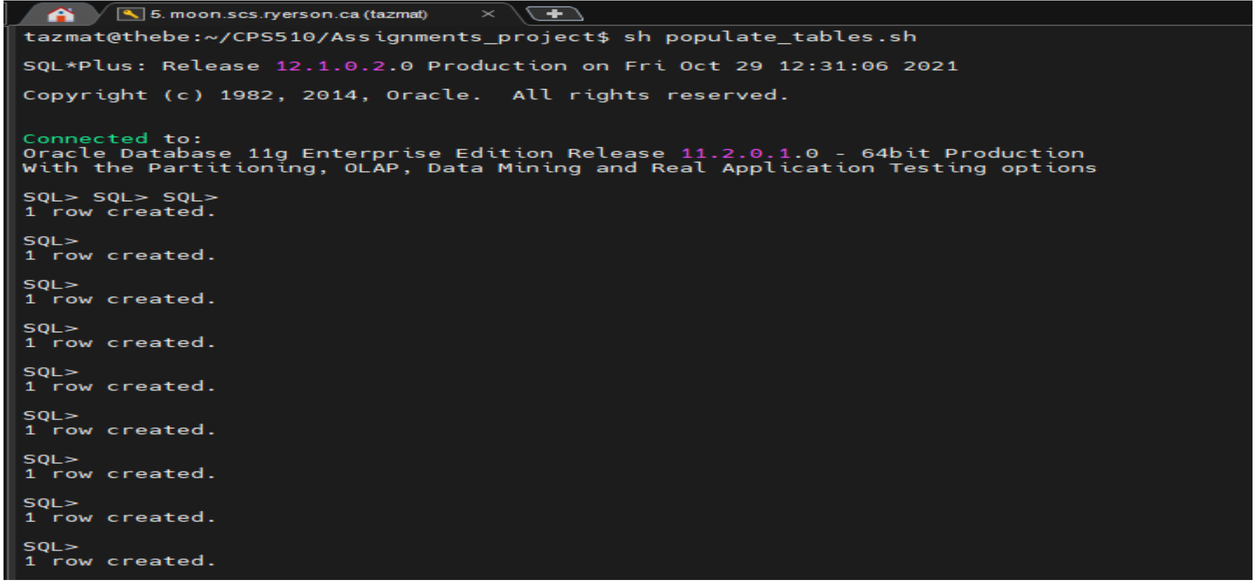
```
tazmat@thebe:~/CPS510/Assignments_project$ sh drop_tables.sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 29 12:27:54 2021

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
tazmat@thebe:~/CPS510/Assignments_project$
```

- **Create tables**



```
tazmat@thebe:~/CPS510/Assignments_project$ sh create_tables.sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 29 12:30:01 2021

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>   2    3    4    5
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9   10   11
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9   10
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9   10   11   12   13
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9   10
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9   10
Table created.

SQL> SQL>   2    3    4    5    6    7    8
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9
Table created.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
tazmat@thebe:~/CPS510/Assignments_project$
```

- **Populate tables**



```
tazmat@thebe:~/CPS510/Assignments_project$ sh populate_tables.sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 29 12:31:06 2021

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.
```

# Functional Dependencies

## -- Functional Dependencies

We have completed our database design and the next step is to normalize our DBMS design. In order to normalize our design, we first outlined the functional dependencies in the system. We normalized each table with 1NF and 2NF procedures.

We take each table and apply the normalization technique:

1.

| JB_Users | | |
|---|---|---|
| PK | Login_id | Char |
| | password | Char |

| JB_Members | | |
|---|---|---|
| PK | Member_id | Char |
| | Member_type | Char |
| | Address | Char |
| | Email | Char |
| | Date_created | Date |
| | Subscriptions | Char |
| FK | Login_id | Char |

In the table JB_Members,

Member_type, Address, Email, Date_created and Subscriptions depended on Member_id.

So, Memebr_id--→Member_type, Address,Email,Date_created, subscriptions.

But Login_id does not depend on Member_id as it belongs to another table.
This table is of the form 1NF and 2NF.

In the table JB_Users,

Password depend on Login_id.

So, key Login_id -→password.

If we look at the relationship between these two tables, each JB_Member will have one Login_id, so Member_id and login_id will have one to one relationship and vice versa. This table is of the form 1NF and 2NF.

2.

| Recruiters | | |
|---|---|---|
| PK | recr_id | Char |
| | Company_name | Char |
| | Manager_name | Char |
| FK | Member_id | Char |

In this table Recruiters recr_id is a on which all the other attributes depend on company_name and Manager_name but Member_id does not depend on recr_id and that is okay as it belongs to another table and is a primary key to another table.

Represented as Recr_id→company_name,Manager_name.

Recruiter Table has one to one relationship with JB_Members table and holds dependencies. Recr_id→Member_id and vice versa. This table is of the form 1NF and 2NF.

3.

| | **Qualifications** | |
|---|---|---|
| PK | Qualification_id | Char |
| | Edu_level | Char |
| | Experience | Number |
| | Cover_letter | Char |
| | Certi_License | Char |
| FK | JB_user_id | Char |

In this table Qualification_id is a key on which all the other attributes depend.

Represented as Qulification_id→Edu_level,Experience,Cover_leter and Certi_license.

Whereas Jb_user_id is a key to another table and acts as forign key here and does not depends on Qualification_id.

Having a foreign key makes the relationship between Qualifications table and JB_Users table and we can describe that as a one to one relationship. As you see, each JB_user_id is associated with only one Qualification_id. In other words each JB_user will have one qualification to hold. This table is of the form 1NF and 2NF.

4.

| | **HR_Department** | |
|---|---|---|
| PK | Depart_id | Char |
| | Depart_name | Char |
| | HR_Manager | Number |
| FK | Org_id | Char |

In the table Departments key Depart_id has attributes that directly depend on it.

We could represent by Depart_id→Depart_name,Hr_Manger

Whereas Org_id is a foreign key and doesn’t depend on depart_id.

The table departments hold relation with Recruiters table as one to many relationships. That means Recruiter could have many departments but one department would have only one recruiter/company. Relationship between the two tables (Recruiter and Departments) will represent in term of functional dependencies as Depart_id depend on Recr_id (Depart_id→Depart_id).

5.

| JB_Postings | | |
|---|---|---|
| PK | Job_id | Char |
| | Company | Char |
| | Salary | Number |
| | Job_title | Char |
| | Job_location | Char |
| | Job_type | Char |
| | Effect_date | Date |
| | End_date | Date |
| FK | depart_id | Char |

This Table holds functional dependency with the HR_Department table as Job_id from JB_Postings table depends on Depart_id.

It holds many to one relationship with the HR_department table as one HR will have many job postings.

If you look at the table JB_posings all the attributes depend on the Job_id key of the table.

We can represent this as Job_id →company,Salary,Job_tile,Job_location,Job_type,Effect_date and End_date.

This table is of the form 1NF and 2NF.

6.

| Connections | | |
|---|---|---|
| PK | Conn_id | Char |
| | Date_made | Date |
| | Conn_details | Char |
| FK | Member_id | Char |

This table holds the many-to-many relationship with JB_Memebrs table as each connection will have many members and many connections belong to one member.

So conn_id depends on Member_id (conn_id →member_id) and member_id depends on conn_id (member_id → conn_id).

If we look at the conn_id key all the other attribute depends on Conn_id so →date_made and Conn_details. This table is of the form 1NF and 2NF.

**Conclusion:** All the tables in our DBMS are normalized to 1NF and 2NF form.

## -- Normalization/ 3NF

In order to normalize our design, we first outlined the functional dependencies in the system. We have normalized each table with 1NF and 2NF procedure and now we will apply the 3rd NF.

In our database no table use any transitive keys to make primary keys, so our database if of form 3NF.

We take each table and apply the 3NF normalization technique:

1.

| | JB_Members | |
|---|---|---|
| PK | Member_id | Char |
| | Member_type | Char |
| | Address | Char |
| | Date_created | Char |
| | Subscriptions | Date |
| | Login_id | Char |
| FK | | Char |

All the non key attributes in the table depends on the primary key Member_id and all non-key attribute are non-transitively dependent on Member_id that makes it to 3NF.

JB_Members (Member_id, Member_type,Address,Email,Date_created,Subscriptions).

So, Member_id--→ { Member_type, Address,Date_created, subscriptions}.

This table has only one candidate key and that is the primary key (Member_id) of the table.

But Login_id does not depend on Member_id as it's belonging to another table. This table is of the form 3NF.

2.

| | JB_Users | |
|---|---|---|
| PK | Login_id | Char |
| | Password | Char |

In the table JB_Users, all the non-key attributes depends on key attribute which is Login_id depends on Password.

Also there is no transitive key in the table.

So, key Login_id -→ password.

**Note:** If we look at the relationship between these two tables, each JB_Member will have one Login_id, so Member_id and Login_id will have one to one relationship and vice versa. Both tables are functionally dependent and all primary keys are non-transitive. This table is of the form satisfies all three form 1NF, 2NF and 3NF.

3.

| | Recruiters | |
|---|---|---|
| PK | recr_id | Char |
| | Company_name | Char |
| | Manager_name | Char |
| FK | Member_id | Char |

In this table Recruiters (recr_**id**, Company_name, Manager_name) all the non-key attributes depends on recr_id and that is a primary key of the table. This table has no transitive keys as all keys are unique.

Represented as Recr_id→company_name,Manager_name.

This table is of the form 3NF.

4.

| | Qualifications | |
|---|---|---|
| PK | Qualification_id | Char |
| | Edu_level | Char |
| | Experience | Number |
| | Cover_letter | Char |
| | Certi_License | Char |
| FK | JB_user_id | Char |

In this table Qualification_id is a key on which all the other non-key attributes depends on and there is no key that is transitively dependent of any other table keys. All the keys of the table are unique.

Represented as Qualification_id→Edu_level,Expeience,Cover_leter and Certi_license.

Whereas Jb_user_id is a key to another table and acts as forign key her and does not depends on Qualification_id. This table is of the form 3NF.

5.

| HR_Department |
|---|

| | | |
|---|---|---|
| PK | Depart_id | Char |
| | Depart_name | Char |
| | HR_Manager | Number |
| FK | Org_id | Char |

All the non key attributes in the table depends on the primary key Depart_id and all non-key attribute is non-transitively dependent on Depart_id that makes it to 3NF.

All the non-key attributes are unique.
We could represent by Depart_id→Depart_name,Hr_Manger

Whereas Org_id is a foreign key and doesn't depends on depart_id. This table is of the form 3NF.

6.

| JB_Postings | | |
|---|---|---|
| PK | Job_id | Char |
| | Company | Char |
| | Salary | Number |
| | Job_title | Char |
| | Job_location | Char |
| | Job_type | Char |
| | Effect_date | Date |
| | End_date | Date |
| FK | depart_id | Char |

If you look at the table JB_posings all the non-key attributes depends on Job_id key of the table. There are no transitive relations in the table.

We can represent this as Job_id →company,Salary,Job_tile,Job_location,Job_type,Effect_date and End_date.

This table is of the form 3NF.

7.

| Connections | | |
|---|---|---|
| PK | Conn_id | Char |
| | Date_made | Date |
| | Conn_details | Char |
| FK | Member_id | Char |

If we look at the conn_id key all the other non-key attribute depends on Conn_id so → date_made and Conn_details. This table is of the form 3NF.

**Conclusion:** All the tables in our DBMS are normalized to 1NF, 2NF and 3NF form as all keys are unique.

## Normalization / BCNF

# Conversion of 3NF to BCNF Normalize form by using Algorithm

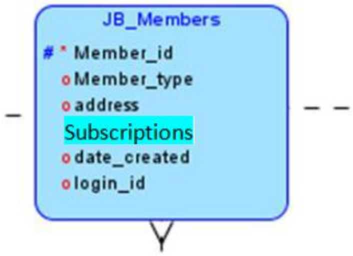
**-- Normalization/ Conversion of 3NF to BCNF with the help of Algorithm.**
**--- We used Bernstein's Algorithm to achieve this task:**

### Bernstein's Algorithm - Broken down into 4 steps:

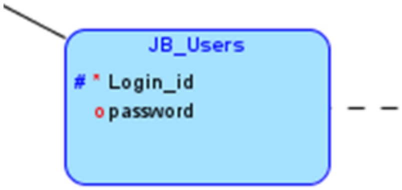
1) **Determine all the functional dependencies**
2) **a) Find and remove redundancies**
   **b) Find and remove partial dependencies**
3) **Find keys**
4) **Create tables**

### Step 1 (finding all functional dependencies)

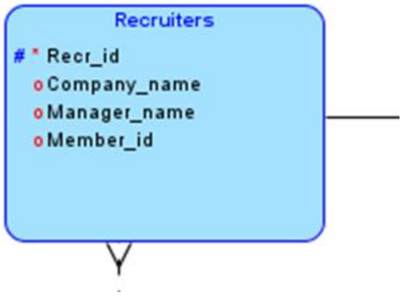
- **Member_id** → {Member_type, address, date_created, Subscriptions}



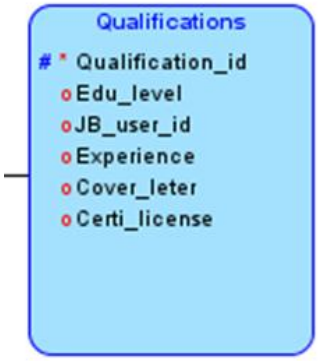
- **Login_id** → { password}



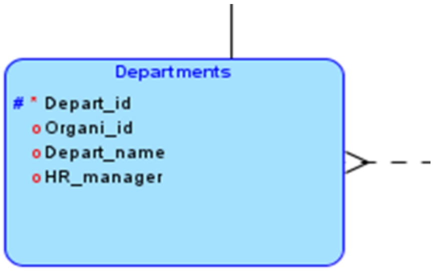
**--Recr_id** → {company_name, Manager_name}



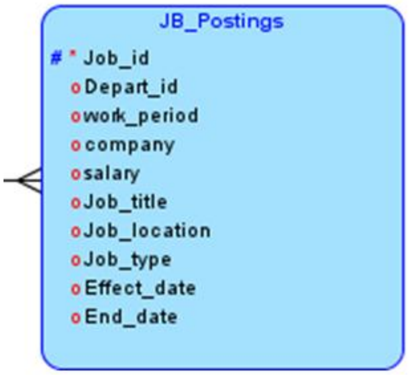
- **Qualification_id**→ {Edu_level, Experience, cover_letter, Certi_license}



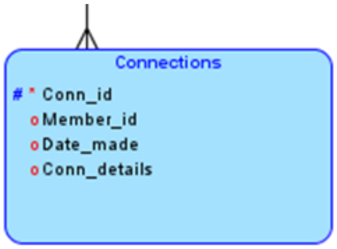
- **depart_id** → {depart_name, HR_manager}



- **job_id →** {work_period, company, salary, job_title, job_location, job_type,Effect_date, End_date}



- **Conn_id →** {Date_made, Conn_details}



**Step 2a (Break RHS and find redundancies)**

Get rid of Redundancies

- **Member_id** → {Member_type, address, date_created, Subscriptions}
    - Reduced list of FD's:
    - **Member_id**→ {Member_type}
    - **Member_id** →{address}
    - **Member_id** →{date_created}
    - **Member_id** →{Subscriptions}
    - ● **No redundancies**
- **Login_id** → { password}
    - Reduced list of FD's:
    - **Login_id** -> {password}
    - ● **No redundancies**
- **Recr_id** → {company_name, Manager_name}
    - Reduced list of FD's:
    - **Recr_id** → {company_name}
    - **Recr_id** → {Manger_name}
    - ● **No redundancies**
- **Qualification_id** → {Edu_level, experience, cover_letter, Certi_license}
    - Reduced list of FD's:
    - **Qualification_id** → {user_degree}
    - **Qualification_id** →{ experience}
    - **Qualification_id** →{cover_letter}
    - **Qualification_id** →{Certi_license}
    - ● **No redundancies**
- **depart_id** → {depart_name, HR_manager}
    - Reduced list of FD's:
    - **depart_id** → {depart_name}
    - **depart_id** → {HR_manager}
    - ● **No redundancies**
- **job_id** → {work_period, company, salary, job_title, job_location, job_type,Effect_date, End_date}
    - Reduced list of FD's:
    - **job_id** →{work_period}
    - **job_id** →{company}

- 
    - **job_id** → {salary}
    - **job_id** → {job_title}
    - **job_id** → {job_location}
    - **job_id** →{job_type}
    - **job_id** →{Effect_date}
    - **job_id** →{End_date}
    - **No redundancies**

- **Conn_id** → {money}
    - <u>Reduced list of FD's:</u>
    - **conn_id** → {Date_made}
    - **conn_id** → {conn_details}
    - **No redundancies**

**Step 2b (Minimize LHS, find and remove partial dependencies)**

- LHS is already minimized, therefore there are no partial dependencies

**Step 3(Find keys) (relational schema)**

- **Member_id**→ {Member_type, address, date_created, subscriptions}
    - Attributes on RHS but not on LHS (cannot be keys)
        - Member_type
        - Address
        - Date_created
        - Subscriptions
    - Possible Keys
        - Member_id
- **Login_id** → { password}
    - Attributes on RHS but not on LHS (cannot be keys)
        - password
    - Possible Keys
        - Login_id

- **Recr_id** → {company_name, manager_name}
    - Attributes on RHS but not on LHS (cannot be keys)
        - Company_name
        - Manger_name
    - Possible Keys

    - Recr_id

- **depart_id** → {depart_name, HR_manager}
  - Attributes on RHS but not on LHS (cannot be keys)
    - depart_name
    - HR_Manger
  - Possible Keys
    - depart_id

- **qualification_id** → {Edu_level, experience, cover_letter, Certi_license}
  - Attributes on RHS but not on LHS (cannot be keys)
    - Edu_level
    - Experience
    - Cover_letter
    - Certi_license
- Possible Keys
    - Qualification_id

- **job_id** → {work_period, company, salary, job_title, job_location, job_type,Effect_date, End_date}
  - Attributes on RHS but not on LHS (cannot be keys)
    - work_period
    - company
    - salary
    - job_title
    - job_location
    - job_type
    - Effect_date
    - End_date
- Possible Keys
  - job_id

- **Conn_id** → {Date_made, Conn_details}
  - Attributes on RHS but not on LHS (cannot be keys)
    - Date_made
    - Conn_details
- Possible Keys
    - Conn_id

**Step 4(Make tables)**

**R1** (**member_id**, member_type, address, date_created, subscriptions, login_id)
With
FD: **member_id** → { member_type, address, date_created , subscriptions }

**R2** (**Login_id**, password)
With
FD: **user_id** →{ user_password}

**R3** (**recr_id**, company_name, manager_name, member_id)
With
FD: **recr_id** →{company_name, manager_name }

**R4** (**depart_id**, depart_name, hr_manager,Org_id)
With
FD: **depart_id** →{ depart_name, hr_manager }

**R5** (**qualification_id**, edu_level, experience, cover_letter, Certi_license, user_id)
With
FD: **qualification_id** → { edu_level, experience, cover_letter, Certi_license }

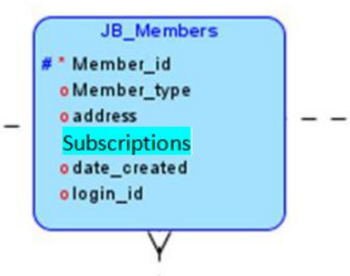
**R6** (**job_id**, work_period, company, salary, job_title, job_location, job_type, effect_date,end_date, depart_id)
With
FD: **job_id** →{, work_period, company, salary, job_title, job_location, job_type, effect_date,end_date }

**R7** (**conn_id**, date_made, conn_details, member_id)
With
FD: **conn_id** → { date_made, conn_details }

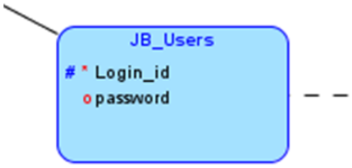
**BCNF (Boyce/Codd Normal Form)**

**Step 1 (finding all functional dependencies (List of all attributes and FDs))**

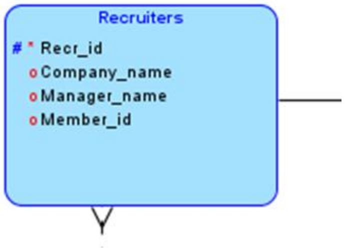
- **member_id** → { member_type, address, date_created, subscriptions, login_id }



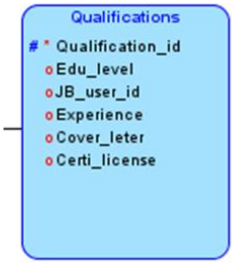
- **Login_id →** { password}



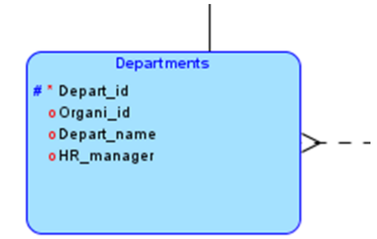
- **Recr_id →** {company_name, Manager_name, member_id}



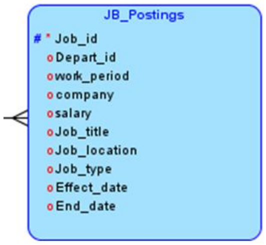
- **Qualification_id→** {Edu_level, Experience, cover_letter, Certi_license, jb_user_id}



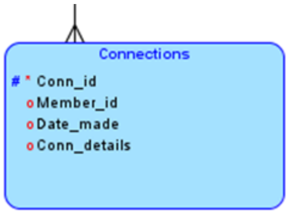
- **depart_id →** {depart name, HR_manager, Organi_id}



- **job_id** → {work_period, company, salary, job_title, job_location, job_type,Effect_date, End_date, depart_id, Depart_id}



- **Conn_id** → {Date_made, Conn_details, member_id}



**Step 2 (make sure that the left hand side are keys, if not decompose)**

Consider the relation schema

**R1** (**member_id**, member_type, address, date_created, subscriptions, login_id)
With FD
**member_id** → { member_type, address, date_created , subscriptions }
This schema has one candidate key
member_id
Therefore this schema is in BCNF

Consider the relation schema
**R2** (**Login_id**, password)
With FD

**Login_id** → {password}
This schema has one candidate key
login_id
Therefore this schema is in BCNF

Consider the relation schema
**R3** (**recr_id**, company_name, manager_name, member_id)
With FD
**recr_id** → {company_name, manager_name }
This schema has one candidate key
recr_id
Therefore this schema is in BCNF

Consider the relation schema
**R4** (**depart_id**, depart_name, hr_manager,Org_id)
With FD
**depart_id** → { depart_name, hr_manager }
This schema has one candidate key
depart_id
Therefore this schema is in BCNF

Consider the relation schema
**R5** (**qualification_id**, edu_level, experience, cover_letter, Certi_license, user_id)
With FD
**qualification_id** → { edu_level, experience, cover_letter, Certi_license }
This schema has one candidate key
qualification_id
Therefore this schema is in BCNF

Consider the relation schema
**R6** (**job_id**, work_period, company, salary, job_title, job_location, job_type, effect_date,end_date, depart_id)
With FD
**job_id** → { work_period, company, salary, job_title, job_location, job_type,effect_date,end_date}
This schema has one candidate key
job_id
Therefore this schema is in BCNF

Consider the relation schema
**R7** (**conn_id**, date_made, conn_details, member_id)
With FD
**conn_id** → { date_made, conn_details }

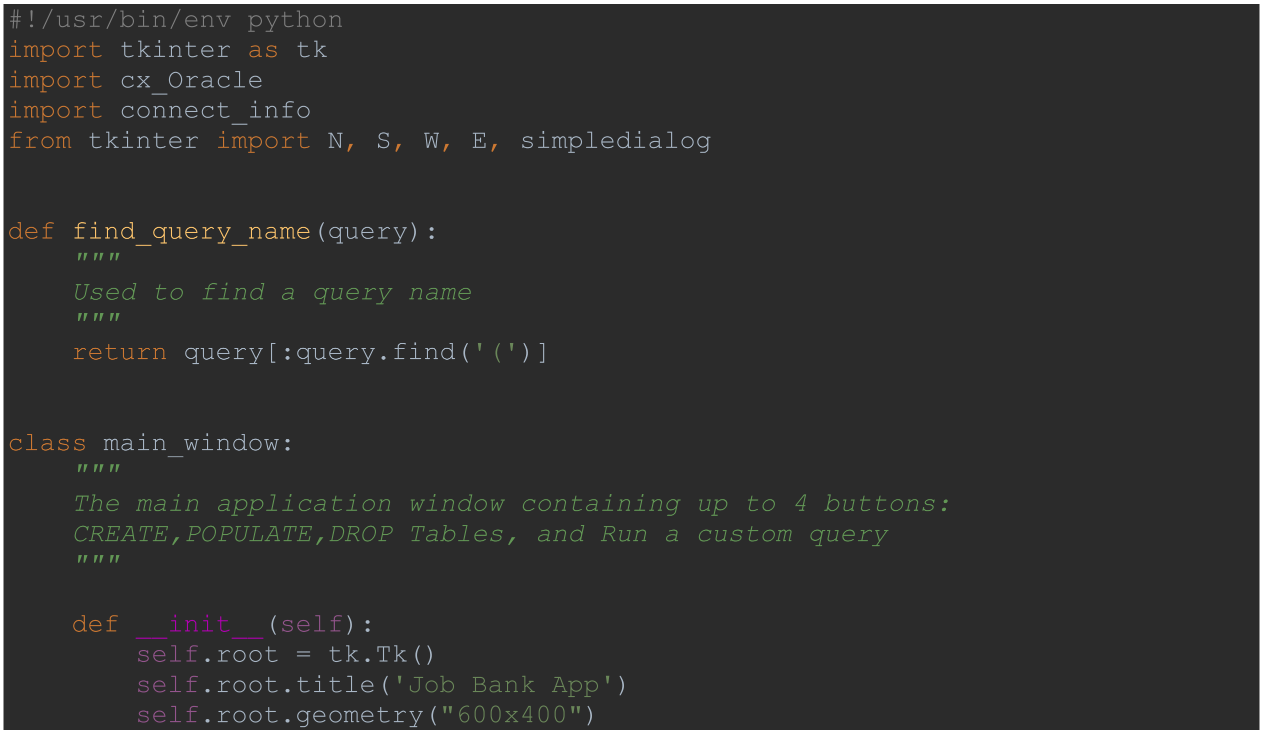
This schema has one candidate key
conn_id
Therefore this schema is in BCNF

**Step 3 (final BCNF schema for R)**

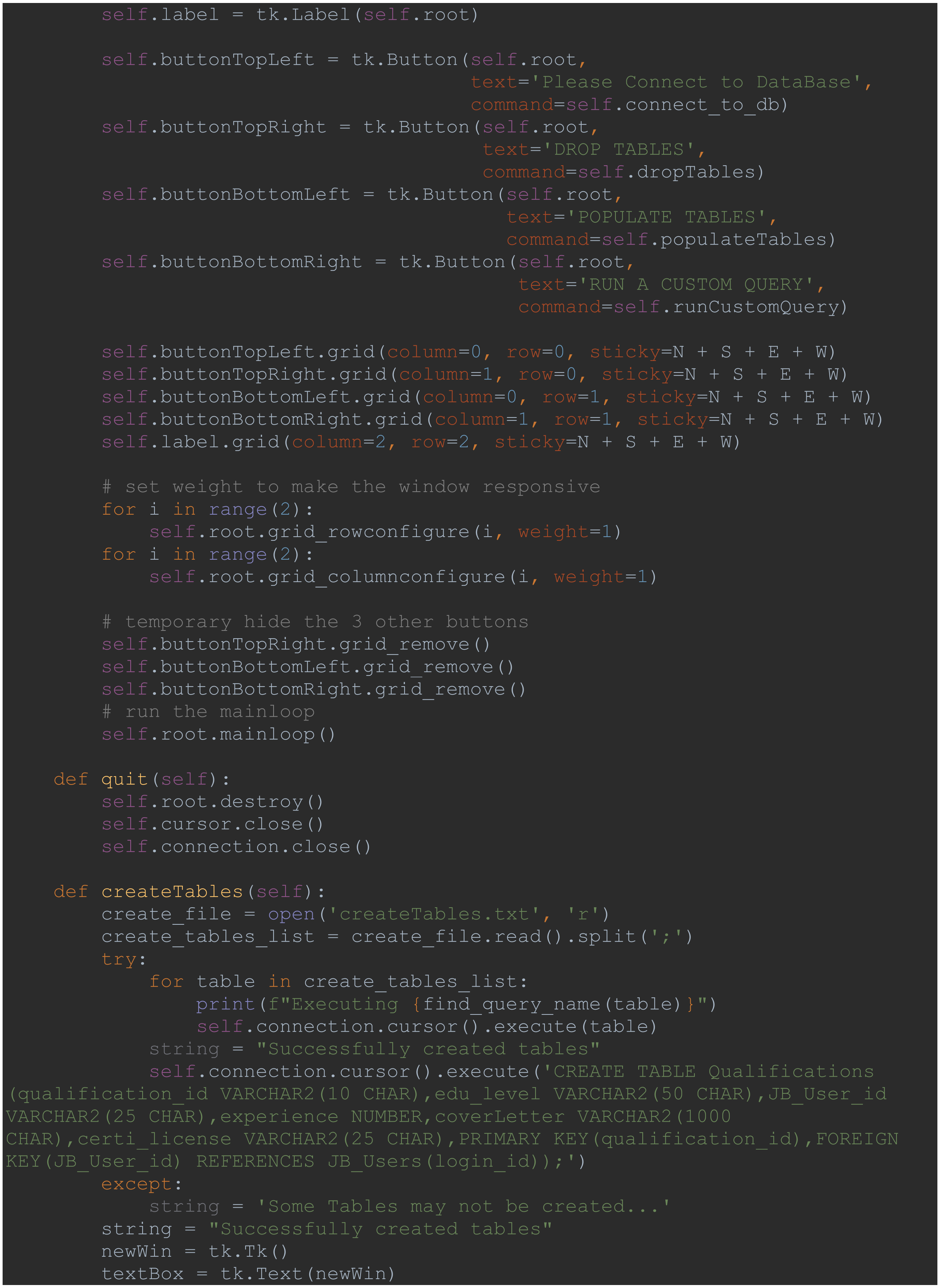
**R1** (**member_id**, member_type, address, date_created, subscriptions, login_id)
**R2** (**Login_id**, password)
**R3** (**recr_id**, company_name, manager_name, member_id)
**R4** (**depart_id**, depart_name, hr_manager,Org_id)
**R5** (**qualification_id**, edu_level, experience, cover_letter, Certi_license, user_id)
**R6** (**job_id**, work_period, company, salary, job_title, job_location, job_type, effect_date,end_date, depart_id)
**R7** (**conn_id**, date_made, conn_details, member_id)

Note: All the tables are of the form BCNF and we didn't need to combine any tables for this stage.

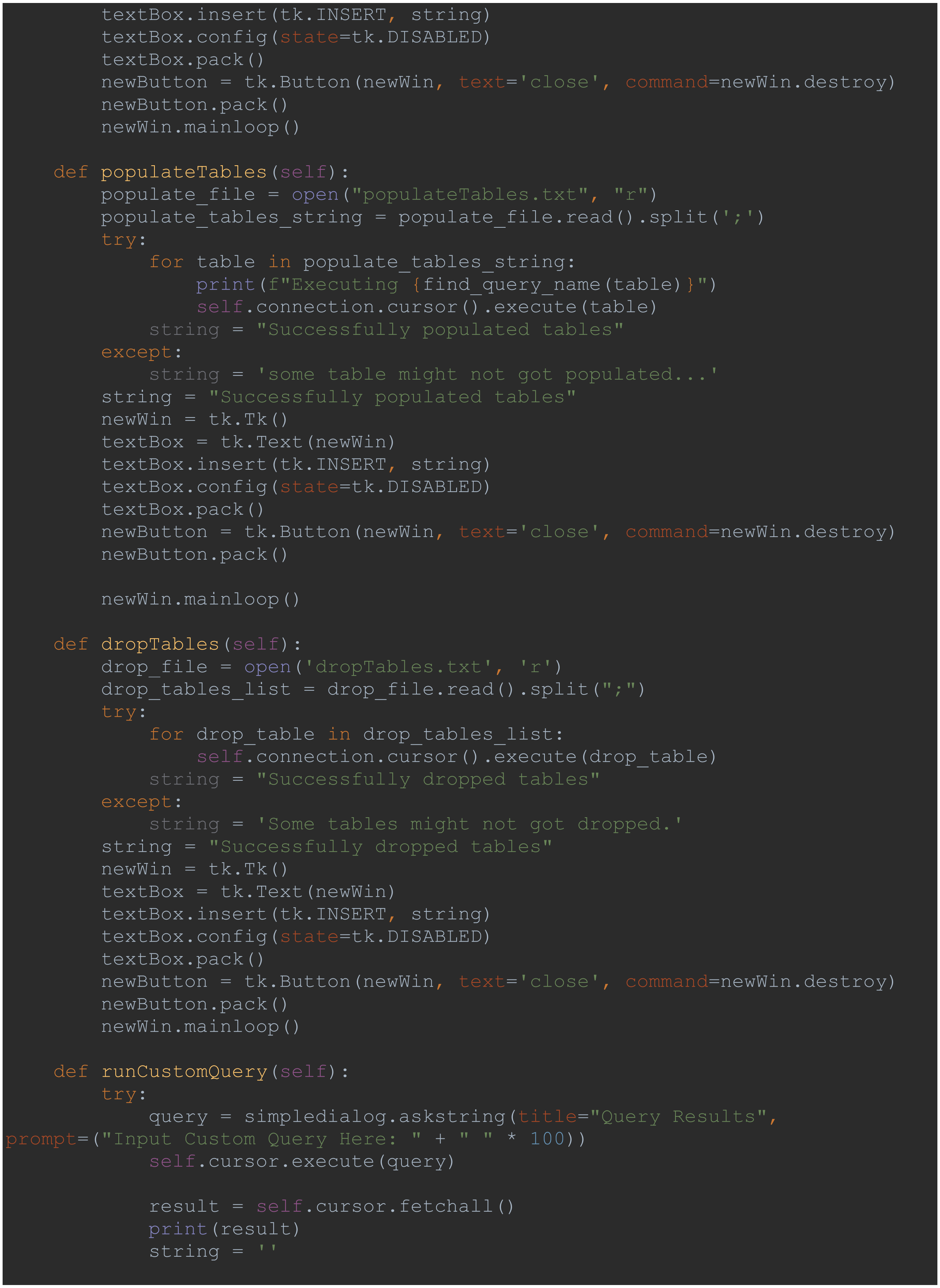
# Graphical User Interface Python Code:



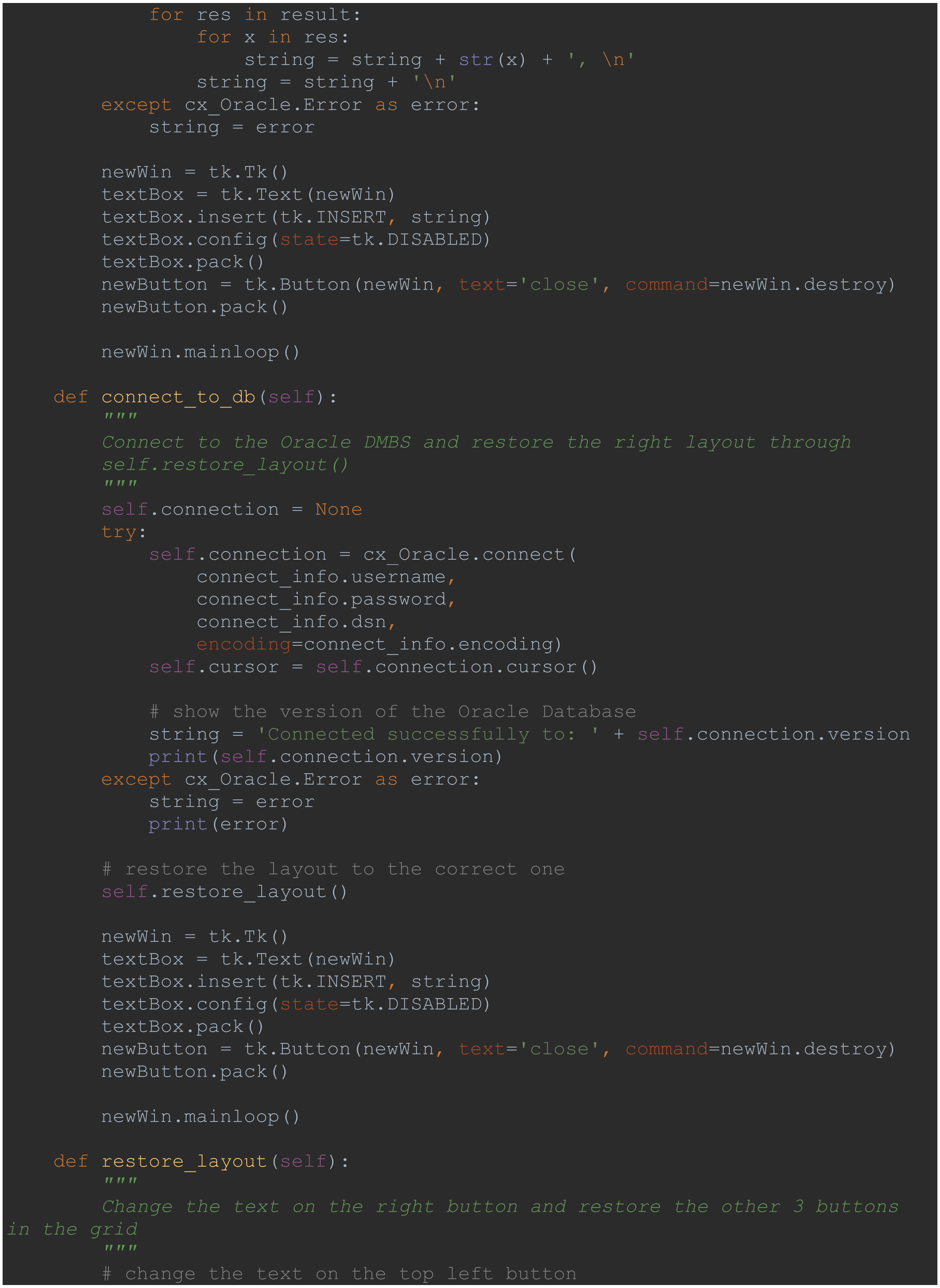
```
#!/usr/bin/env python
import tkinter as tk
import cx_Oracle
import connect_info
from tkinter import N, S, W, E, simpledialog


def find_query_name(query):
    """
    Used to find a query name
    """
    return query[:query.find('(')]


class main_window:
    """
    The main application window containing up to 4 buttons:
    CREATE,POPULATE,DROP Tables, and Run a custom query
    """

    def __init__(self):
        self.root = tk.Tk()
        self.root.title('Job Bank App')
        self.root.geometry("600x400")
```



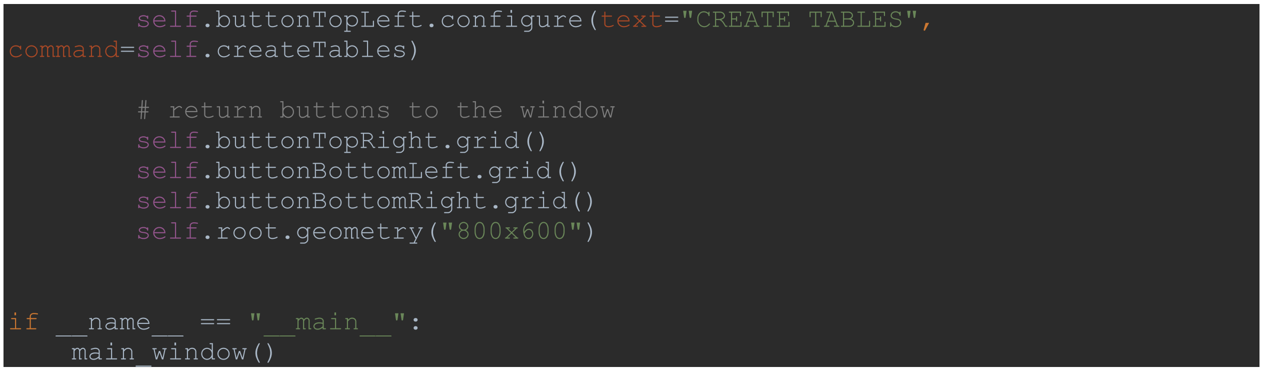
```
        self.label = tk.Label(self.root)

        self.buttonTopLeft = tk.Button(self.root,
                                       text='Please Connect to DataBase',
                                       command=self.connect_to_db)
        self.buttonTopRight = tk.Button(self.root,
                                        text='DROP TABLES',
                                        command=self.dropTables)
        self.buttonBottomLeft = tk.Button(self.root,
                                          text='POPULATE TABLES',
                                          command=self.populateTables)
        self.buttonBottomRight = tk.Button(self.root,
                                           text='RUN A CUSTOM QUERY',
                                           command=self.runCustomQuery)

        self.buttonTopLeft.grid(column=0, row=0, sticky=N + S + E + W)
        self.buttonTopRight.grid(column=1, row=0, sticky=N + S + E + W)
        self.buttonBottomLeft.grid(column=0, row=1, sticky=N + S + E + W)
        self.buttonBottomRight.grid(column=1, row=1, sticky=N + S + E + W)
        self.label.grid(column=2, row=2, sticky=N + S + E + W)

        # set weight to make the window responsive
        for i in range(2):
            self.root.grid_rowconfigure(i, weight=1)
        for i in range(2):
            self.root.grid_columnconfigure(i, weight=1)

        # temporary hide the 3 other buttons
        self.buttonTopRight.grid_remove()
        self.buttonBottomLeft.grid_remove()
        self.buttonBottomRight.grid_remove()
        # run the mainloop
        self.root.mainloop()

    def quit(self):
        self.root.destroy()
        self.cursor.close()
        self.connection.close()

    def createTables(self):
        create_file = open('createTables.txt', 'r')
        create_tables_list = create_file.read().split(';')
        try:
            for table in create_tables_list:
                print(f"Executing {find_query_name(table)}")
                self.connection.cursor().execute(table)
            string = "Successfully created tables"
            self.connection.cursor().execute('CREATE TABLE Qualifications
(qualification_id VARCHAR2(10 CHAR),edu_level VARCHAR2(50 CHAR),JB_User_id
VARCHAR2(25 CHAR),experience NUMBER,coverLetter VARCHAR2(1000
CHAR),certi_license VARCHAR2(25 CHAR),PRIMARY KEY(qualification_id),FOREIGN
KEY(JB_User_id) REFERENCES JB_Users(login_id));')
        except:
            string = 'Some Tables may not be created...'
        string = "Successfully created tables"
        newWin = tk.Tk()
        textBox = tk.Text(newWin)
```



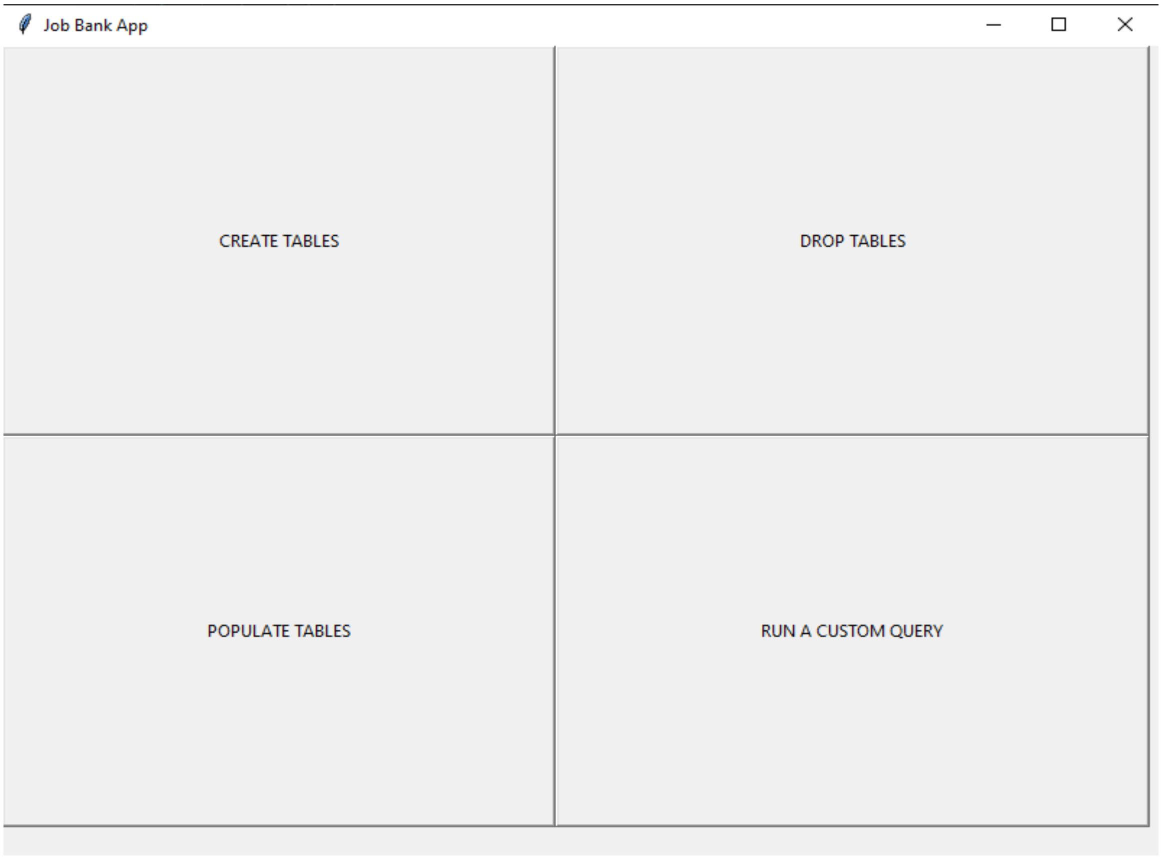
```
        textBox.insert(tk.INSERT, string)
        textBox.config(state=tk.DISABLED)
        textBox.pack()
        newButton = tk.Button(newWin, text='close', command=newWin.destroy)
        newButton.pack()
        newWin.mainloop()

    def populateTables(self):
        populate_file = open("populateTables.txt", "r")
        populate_tables_string = populate_file.read().split(';')
        try:
            for table in populate_tables_string:
                print(f"Executing {find_query_name(table)}")
                self.connection.cursor().execute(table)
            string = "Successfully populated tables"
        except:
            string = 'some table might not got populated...'
        string = "Successfully populated tables"
        newWin = tk.Tk()
        textBox = tk.Text(newWin)
        textBox.insert(tk.INSERT, string)
        textBox.config(state=tk.DISABLED)
        textBox.pack()
        newButton = tk.Button(newWin, text='close', command=newWin.destroy)
        newButton.pack()

        newWin.mainloop()

    def dropTables(self):
        drop_file = open('dropTables.txt', 'r')
        drop_tables_list = drop_file.read().split(";")
        try:
            for drop_table in drop_tables_list:
                self.connection.cursor().execute(drop_table)
            string = "Successfully dropped tables"
        except:
            string = 'Some tables might not got dropped.'
        string = "Successfully dropped tables"
        newWin = tk.Tk()
        textBox = tk.Text(newWin)
        textBox.insert(tk.INSERT, string)
        textBox.config(state=tk.DISABLED)
        textBox.pack()
        newButton = tk.Button(newWin, text='close', command=newWin.destroy)
        newButton.pack()
        newWin.mainloop()

    def runCustomQuery(self):
        try:
            query = simpledialog.askstring(title="Query Results",
prompt=("Input Custom Query Here: " + " " * 100))
            self.cursor.execute(query)

            result = self.cursor.fetchall()
            print(result)
            string = ''
```

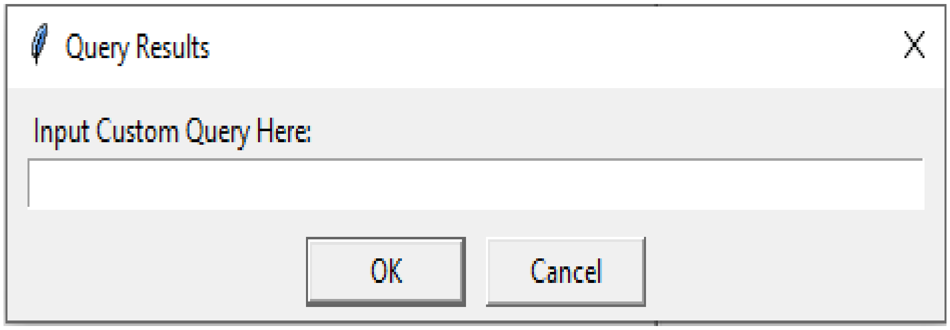


```
            for res in result:
                for x in res:
                    string = string + str(x) + ', \n'
                string = string + '\n'
        except cx_Oracle.Error as error:
            string = error

        newWin = tk.Tk()
        textBox = tk.Text(newWin)
        textBox.insert(tk.INSERT, string)
        textBox.config(state=tk.DISABLED)
        textBox.pack()
        newButton = tk.Button(newWin, text='close', command=newWin.destroy)
        newButton.pack()

        newWin.mainloop()

    def connect_to_db(self):
        """
        Connect to the Oracle DMBS and restore the right layout through
        self.restore_layout()
        """
        self.connection = None
        try:
            self.connection = cx_Oracle.connect(
                connect_info.username,
                connect_info.password,
                connect_info.dsn,
                encoding=connect_info.encoding)
            self.cursor = self.connection.cursor()

            # show the version of the Oracle Database
            string = 'Connected successfully to: ' + self.connection.version
            print(self.connection.version)
        except cx_Oracle.Error as error:
            string = error
            print(error)

        # restore the layout to the correct one
        self.restore_layout()

        newWin = tk.Tk()
        textBox = tk.Text(newWin)
        textBox.insert(tk.INSERT, string)
        textBox.config(state=tk.DISABLED)
        textBox.pack()
        newButton = tk.Button(newWin, text='close', command=newWin.destroy)
        newButton.pack()

        newWin.mainloop()

    def restore_layout(self):
        """
        Change the text on the right button and restore the other 3 buttons
in the grid
        """
        # change the text on the top left button
```



```
        self.buttonTopLeft.configure(text="CREATE TABLES",
command=self.createTables)

        # return buttons to the window
        self.buttonTopRight.grid()
        self.buttonBottomLeft.grid()
        self.buttonBottomRight.grid()
        self.root.geometry("800x600")


if __name__ == "__main__":
    main_window()
```





## Simple and Advanced Database Queries (with Relational Algebra):

```
----------------------------------------------------------
---- *********    Queries    ************* --------------
----------------------------------------------------------
```

**SQL**

```
SELECT *
FROM JB_Postings
WHERE job_location = 'Toronto';
```

**Relational Algebra**

$\sigma_{\text{Location = 'Toronto'}}$ (JB_postings)

**SQL**

```
SELECT *
FROM Qualifications
WHERE years_experience >= 5;
```

**Relational Algebra**

σ year_experience ≥ 5 (Qualifications)

**SQL**

```
SELECT *
FROM JB_Members
WHERE Subscription = 'Premium' AND member_type = 'recruiter';
```

**Relational Algebra**

σ (Subscription = ‘Premium’) AND (member_type = ‘recruiter’) (JB_Members)

**SQL**

```
SELECT *
FROM Department
ORDER BY Org_id DESC, depart_name ASC;
```

**Relational Algebra**

τ (Org_id DESC) AND (depart_name ASC) (Department)

**SQL**

```
SELECT company_name AS company
FROM JB_Recruiters
ORDER BY Manager_name;
```

**Relational Algebra**

τ company_name ASC Π Manager_name (JB_Recruiters)

**SQL**

```
SELECT job_location, COUNT(job_id) AS INTEGER_located
```

```
FROM JB_Postings
GROUP BY job_location;
```

**Relational Algebra**

**Π** job_location, job_id (𝓖 job_location )( JB_Postings)

**SQL**

```
SELECT login_id, edu_level, experience
FROM Qualifications, JB_Members
24
WHERE ((edu_level = 'University of Toronto' OR edu_level = Seneca
College')
AND experience BETWEEN 2 AND 10);
```

**Relational Algebra**

**Π** login_id, edu_level, experience (**σ** (edu_level = ‘University of Toronto) OR (edu_level = ‘Seneca College’) AND
(experience ≥ 2) AND (experience ≤ 10)) (Qualifications >< JB_Members)

**SQL**

```
SELECT subscription, login_id, edu_level, job_title
FROM JB_Postings j, Qualifications q, JB_Users a, JB_Members s
WHERE (s.subscription = ‘Gold’ AND q.edu_level = Seneca College');
```

**Relational Algebra**

**Π** subscription, login_id, edu_level, job_titile (**σ** (subscription = ‘Gold’) AND (user_degree = ‘Seneca College’) )(JB_Postings ><
Qualifications >< JB_Users >< JB_Members)

**SQL**

```
SELECT JB_Members.member_id, JB_Members.date_created, Connections.date_made,
Connections.details
FROM JB_Members
INNER JOIN Connections
ON JB_Members.member_id = Connections.member_id;
```

**Relational Algebra**

**Π** member_id, date_created, date_made, details (**σ** member_id = member_id) (JB_Members
Connections)

## **Conclusion:**

Working on this Online Job Bank Database Management System has provided us with a foundation in all the aspects of database design and implementation. Through the course of the 10 assignments, we realized how important it was to represent data in a clear and concise manner. With the theoretical knowledge and the understanding of entity relationship diagrams, relational schema design, functional dependencies, normalization, etc., we were able to turn various meaningless sets of data into a useful and accessible database.

In the technical aspect, we familiarize ourselves with the services provided by Oracle 12g and SQL. Using the tools provided, we learned how to create tables, drop tables, populate tables and create custom queries that can display query information in any format. Finally, creating a GUI using python also familiarized us with the interactions between the front-end and back-end components of the database.

This project was also crucial in exercising our teamwork and communication skills along with improving our technical abilities with project management and software development.

Overall, working on the job bank DBMS was different, challenging yet an enjoyable experience. It truly unlocked our potential to use the knowledge and skills that we have acquired in this course for the future career development.