ADVANCED DATABASE MANAGEMENT

GROUP PROJECT REPORT

OLIVE

JOB ANALYTICS

Submitted By

Bharath Jagadeesh Upasi

Roselle Paala

Sandeep Jiyalal Kahar

Shruthi Reddy Gujjula

Subrahmanyam Mallavarapu

Contents

INTRODUCTION:

The JOB ANALYTICS system is the platform where a user can login to primarily check a host of information related to jobs and companies. The user-friendly UI helps users to easily navigate and find the information they are looking for. On this platform a user can create a free account. This platform provides users with information which otherwise users have to manually search for collect from various multiple sources. Here users have access to job availability information. They can check the number of openings/vacancies for a particular role in a particular company at a particular location. They can also get the contact information of the companies. They can learn about the minimum qualifications required to apply for a job. This platform, upon tie ups with companies, also provides information of the employees as to how many employees are working in a department. What are their education profiles and past work experience along with the skills set they possess? Using all this information we perform analysis using Spotfire and present the user with analytical information. With this feature user can also check statistics like students from which university have been successful in getting a placement. What are the job availability trends for a role in the recent years?

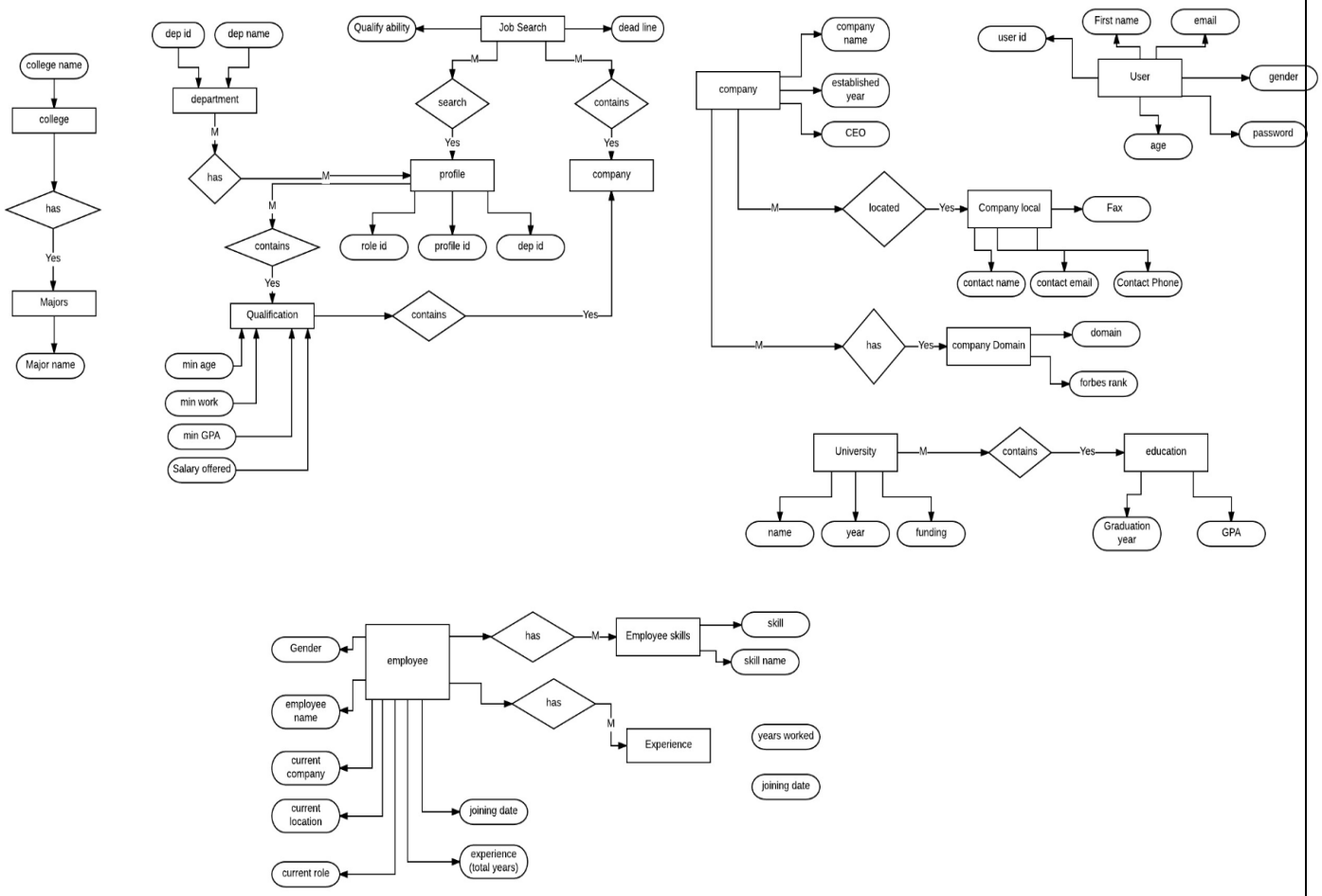
We have used Oracle Database to store all the information. We developed the Front-end UI using Bootstrap jQuery, JSP and for framework we used Springs and JDBC.

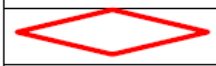
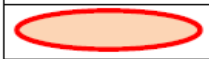
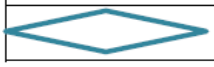
ASSUMPTION:

We have made the assumption that we have a tie up with the companies, and companies provide us with employee profiles to store in our DB.

CONCEPTUAL DESIGN:

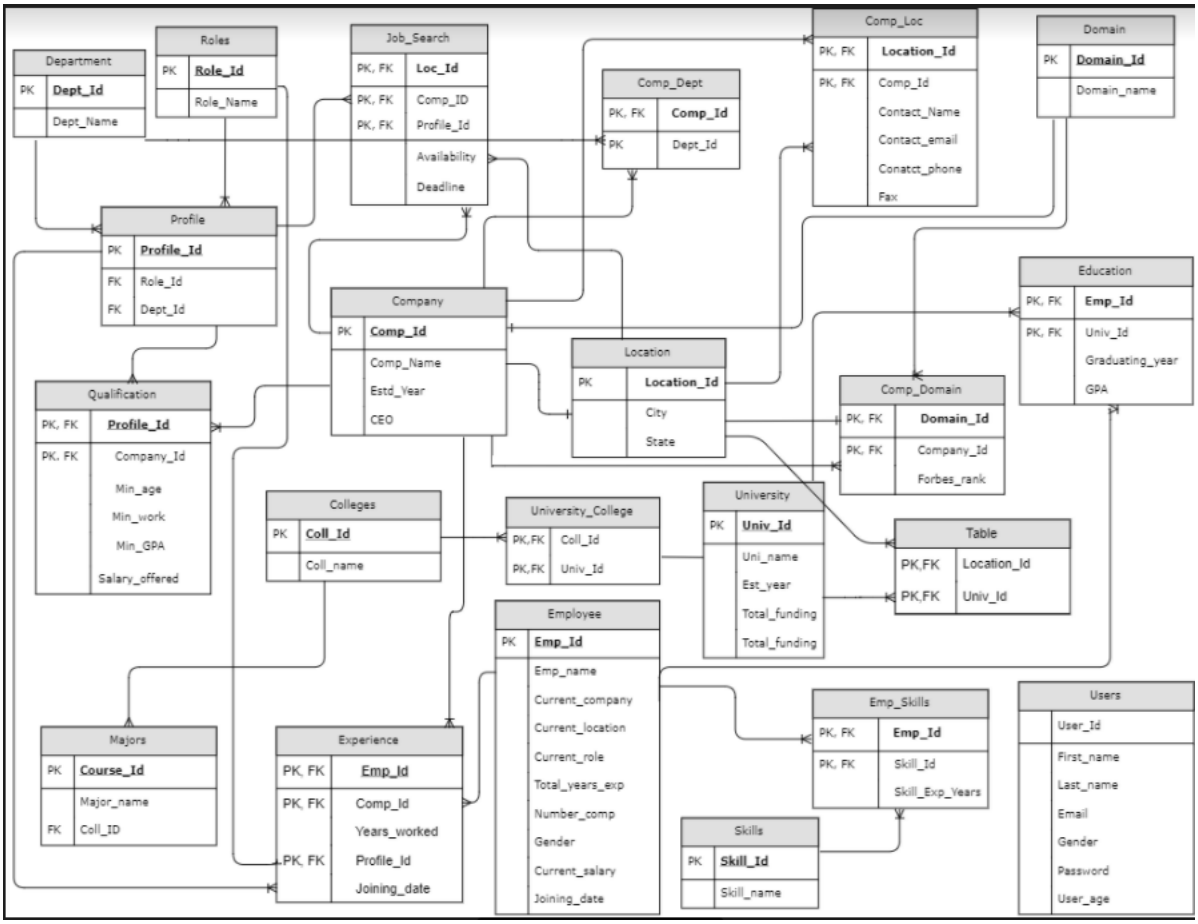
Entity-Relationship diagram or as we call it the E-R diagram is the first step of any DB project. This is the conceptual design phase of the project. Here we focus only on gathering the requirements of the DB for the business model. We list out entities, their attributes and the relationships between those entities. This diagram gives us a brief overview of what the project is about and how the DB design looks like. For example, here we have entities like company and its attributes are company name, an id for the company.



Symbols	Indication
	Relationship (m:n)
	Primary key
	Relationship (1:n)

LOGICAL DESIGN:

After the conceptual design phase next comes the Logical Design phase. Here we take the DB design forward from the E-R diagram. We convert the entities and attributes to tables and columns in the tables respectively. Relationships between tables can be of 3 types, viz, Unary, Binary and Ternary. We also have the relationship cardinality of 3 types, viz, one-one, one-many and many-many. We define all this in the logical design and draw the below logical diagram of the DB.



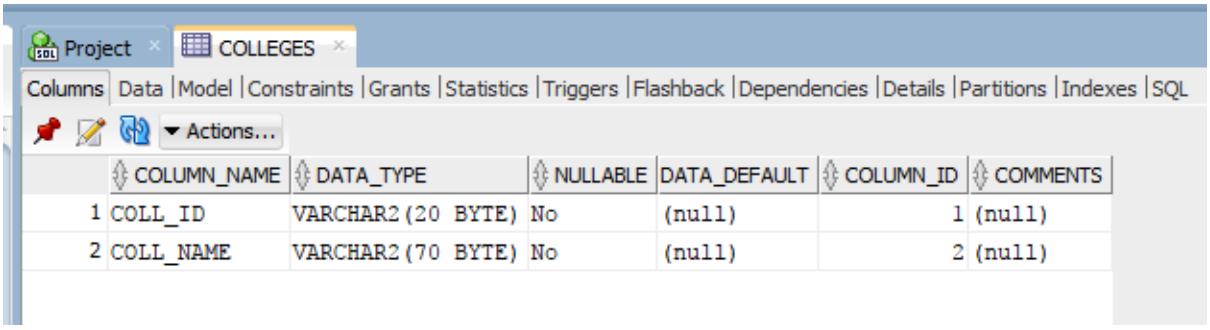
PHYSICAL DATABASE DESIGN:

Physical Design phase comes next in the DB design process. He we choose the DB, Oracle for this project, and also other file structures. We then define the data dictionary for all the tables. Data dictionary includes the data types of all the columns. Also, constraints to be imposed on the columns are defined in this phase before actually going ahead and creating the tables. Below are the DDL statements and snapshots for the tables.

Table: COLLEGES

```
CREATE TABLE COLLEGES
(COLL_ID VARCHAR2(20 BYTE) NOT NULL,
 COLL_NAME VARCHAR2(70 BYTE) NOT NULL,
 CONSTRAINT PK_COLL_ID PRIMARY KEY (COLL_ID));
```

Data Dictionary:



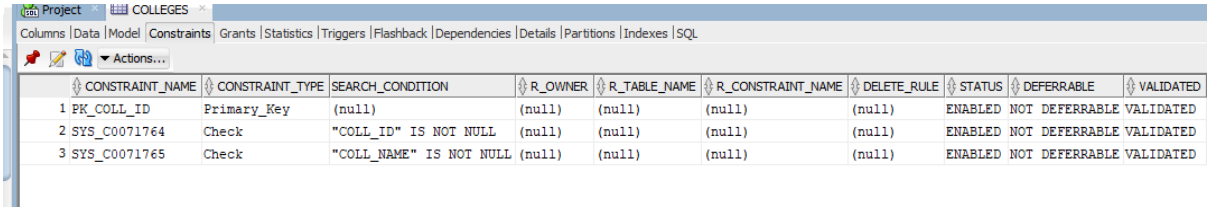
Project | COLLEGES

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COLL_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	COLL_NAME	VARCHAR2(70 BYTE)	No	(null)	2	(null)

Constraints:



Project | COLLEGES

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

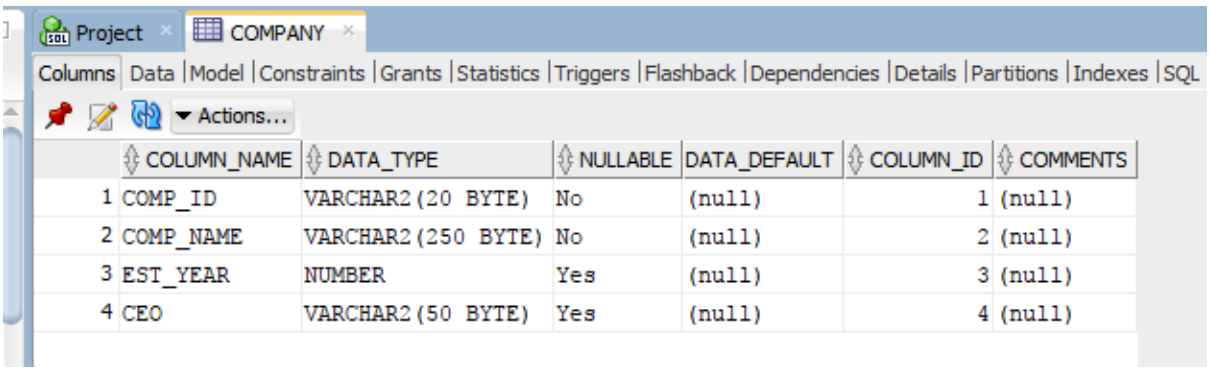
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	PK_COLL_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0071764	Check	"COLL_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
3	SYS_C0071765	Check	"COLL_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Table: COMPANY

```
CREATE TABLE COMPANY
(
  COMP_ID VARCHAR2(20 BYTE) NOT NULL
, COMP_NAME VARCHAR2(250 BYTE) NOT NULL
, EST_YEAR NUMBER
, CEO VARCHAR2(50 BYTE)
, CONSTRAINT PK_COMP_ID PRIMARY KEY (COMP_ID)
  );
```

Data Dictionary:



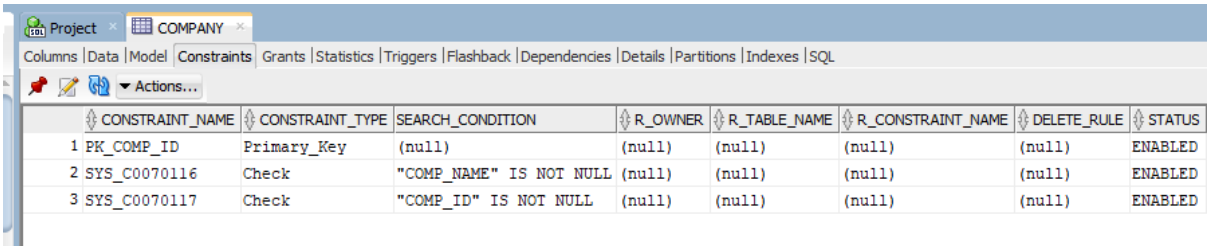
Project | COMPANY

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COMP_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	COMP_NAME	VARCHAR2(250 BYTE)	No	(null)	2	(null)
3	EST_YEAR	NUMBER	Yes	(null)	3	(null)
4	CEO	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)

Constraints:

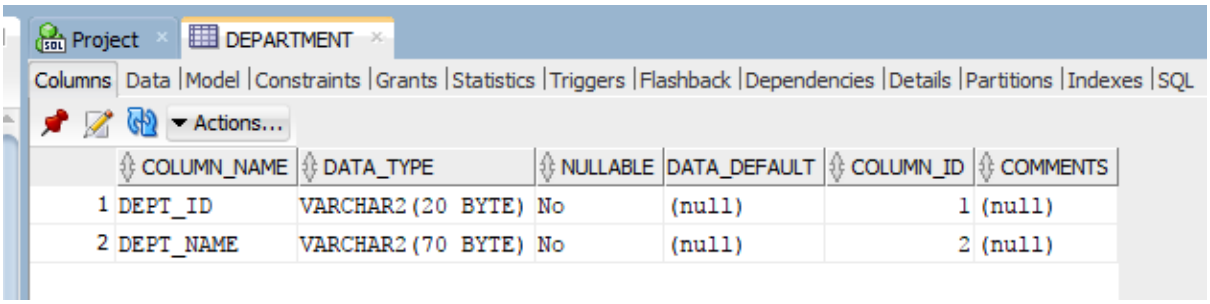


	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS
1	PK_COMP_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED
2	SYS_C0070116	Check	"COMP_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED
3	SYS_C0070117	Check	"COMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED

Table: DEPARTMENT

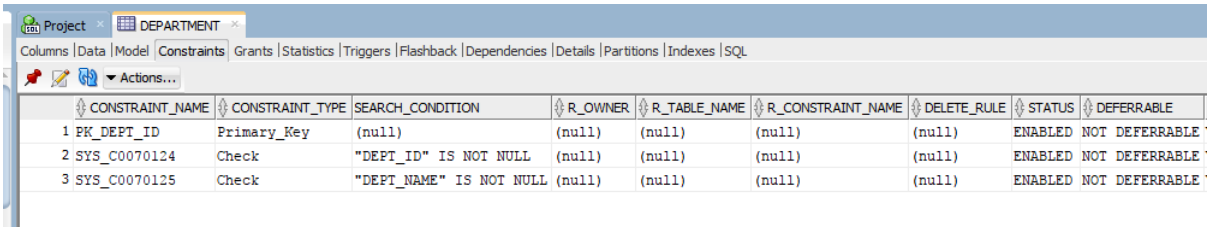
```
CREATE TABLE DEPARTMENT
(
  DEPT_ID VARCHAR2(20 BYTE) NOT NULL
, DEPT_NAME VARCHAR2(70 BYTE) NOT NULL
, CONSTRAINT PK_DEPT_ID PRIMARY KEY ( DEPT_ID )
  );
```

Data Dictionary:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	DEPT_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	DEPT_NAME	VARCHAR2(70 BYTE)	No	(null)	2	(null)

Constraints:

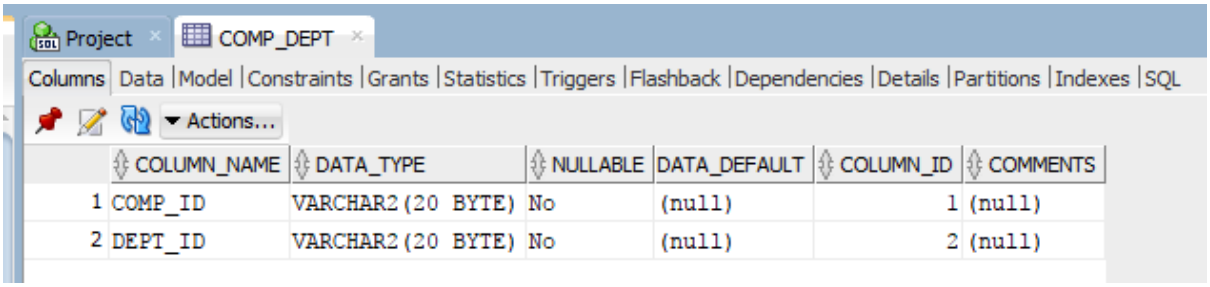


	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	PK_DEPT_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
2	SYS_C0070124	Check	"DEPT_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
3	SYS_C0070125	Check	"DEPT_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: COMP_DEPT

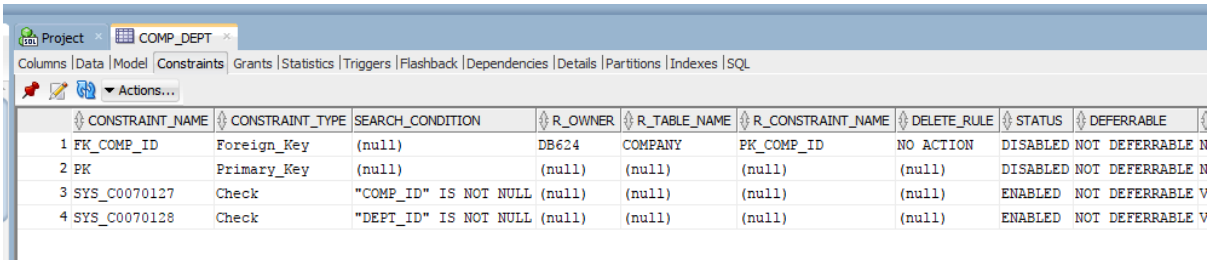
```
CREATE TABLE COMP_DEPT
(
  COMP_ID VARCHAR2(20 BYTE) NOT NULL
, DEPT_ID VARCHAR2(20 BYTE) NOT NULL
, CONSTRAINT PK PRIMARY KEY ( COMP_ID , DEPT_ID )
  );
```

Data Dictionary:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COMP_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	DEPT_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)

Constraints:

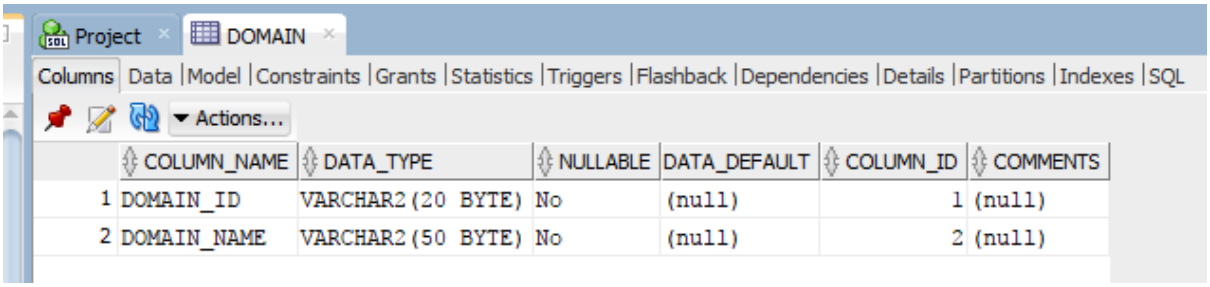


	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	FK_COMP_ID	Foreign_Key	(null)	DB624	COMPANY	PK_COMP_ID	NO ACTION	DISABLED	NOT DEFERRABLE
2	PK	Primary_Key	(null)	(null)	(null)	(null)	(null)	DISABLED	NOT DEFERRABLE
3	SYS_C0070127	Check	"COMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
4	SYS_C0070128	Check	"DEPT_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: DOMAIN

```
CREATE TABLE DOMAIN
(
  DOMAIN_ID VARCHAR2(20 BYTE) NOT NULL
, DOMAIN_NAME VARCHAR2(50 BYTE) NOT NULL
, CONSTRAINT PK_DOMAIN_ID PRIMARY KEY (DOMAIN_ID)
  );
```

Data Dictionary:



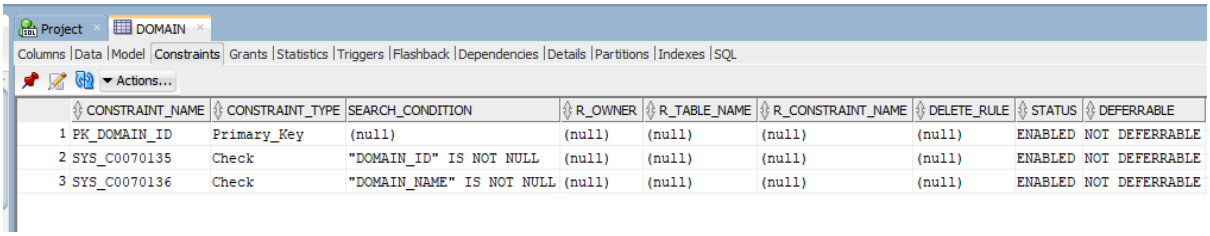
Project | DOMAIN

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	DOMAIN_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	DOMAIN_NAME	VARCHAR2(50 BYTE)	No	(null)	2	(null)

Constraints:



Project | DOMAIN

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

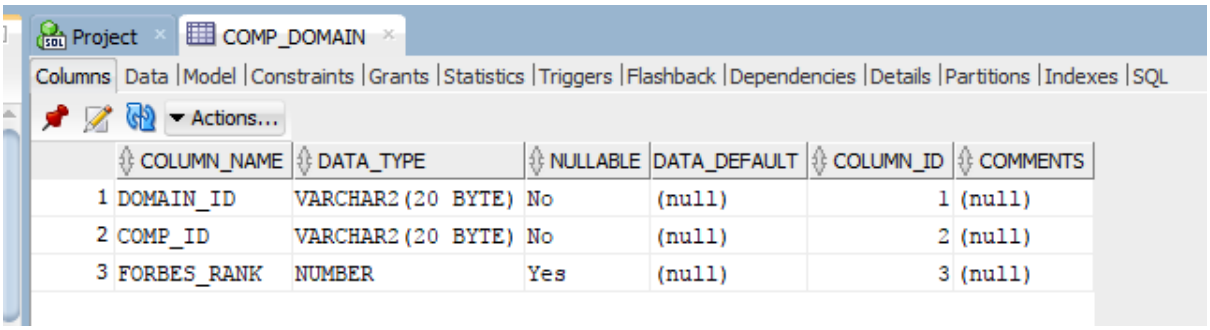
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	PK_DOMAIN_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
2	SYS_C0070135	Check	"DOMAIN_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
3	SYS_C0070136	Check	"DOMAIN_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: COMP_DOMAIN

```
CREATE TABLE COMP_DOMAIN
(
  DOMAIN_ID VARCHAR2(20 BYTE) NOT NULL
, COMP_ID VARCHAR2(20 BYTE) NOT NULL
, FORBES_RANK NUMBER
, CONSTRAINT PK_COMP_DOMAIN PRIMARY KEY (DOMAIN_ID, COMP_ID )
  );
```

Data Dictionary:



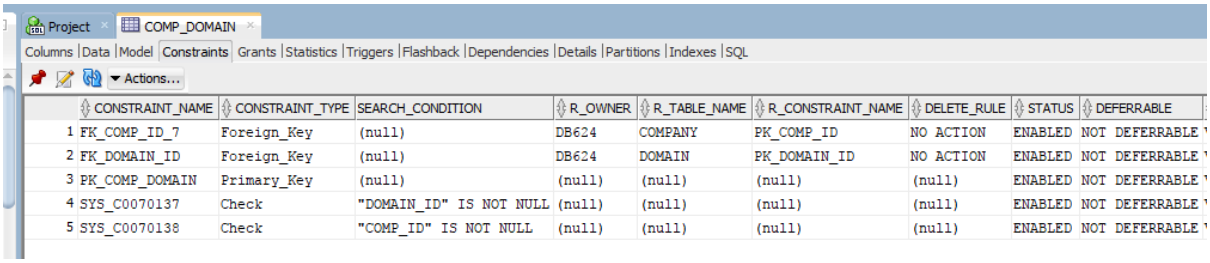
Project | COMP_DOMAIN

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	DOMAIN_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	COMP_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	FORBES_RANK	NUMBER	Yes	(null)	3	(null)

Constraints:



Project | COMP_DOMAIN

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

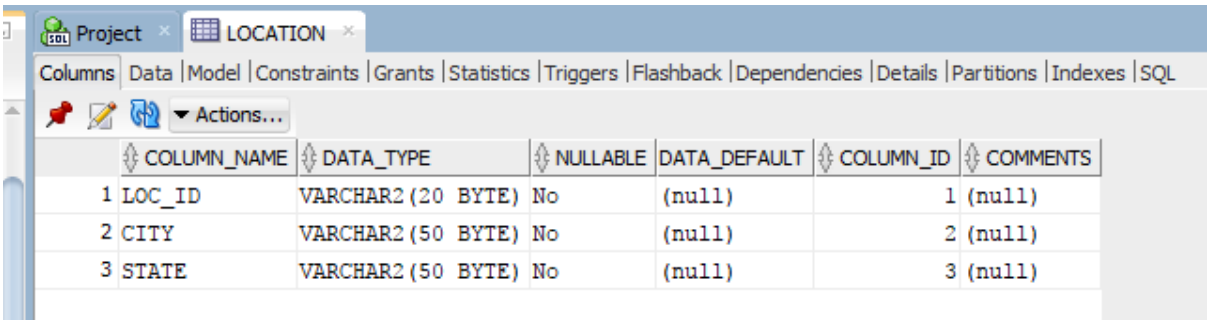
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	FK_COMP_ID_7	Foreign_Key	(null)	DB624	COMPANY	PK_COMP_ID	NO ACTION	ENABLED	NOT DEFERRABLE
2	FK_DOMAIN_ID	Foreign_Key	(null)	DB624	DOMAIN	PK_DOMAIN_ID	NO ACTION	ENABLED	NOT DEFERRABLE
3	PK_COMP_DOMAIN	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
4	SYS_C0070137	Check	"DOMAIN_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
5	SYS_C0070138	Check	"COMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: LOCATION

```
CREATE TABLE LOCATION
(
  LOC_ID VARCHAR2(20 BYTE) NOT NULL
, CITY VARCHAR2(50 BYTE) NOT NULL
, STATE VARCHAR2(50 BYTE) NOT NULL
, CONSTRAINT PK_LOC_ID PRIMARY KEY ( LOC_ID )
 );
```

Data Dictionary:



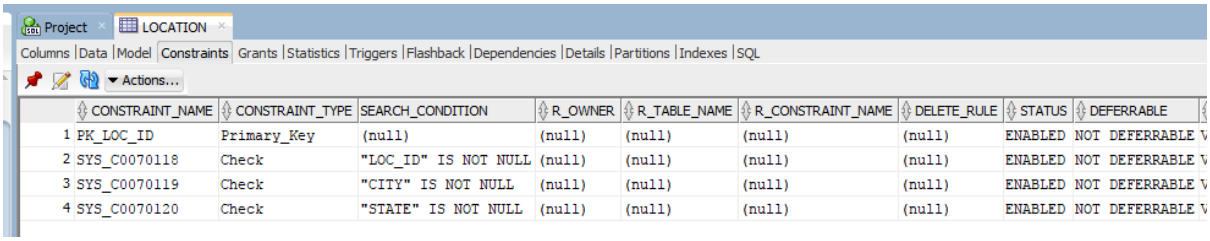
Project | LOCATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	LOC_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	CITY	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	STATE	VARCHAR2(50 BYTE)	No	(null)	3	(null)

Constraints:



Project | LOCATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

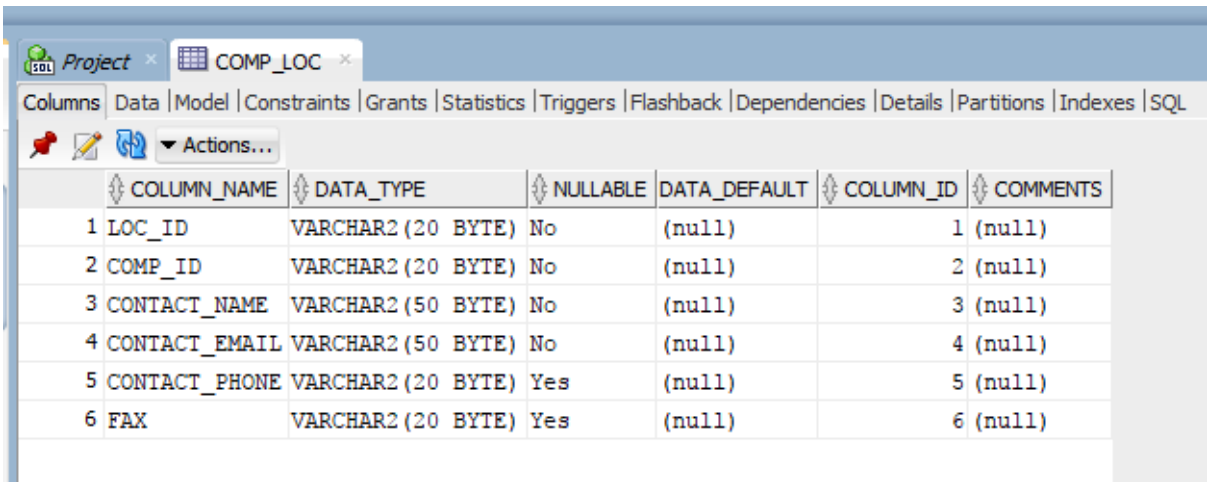
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	PK_LOC_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
2	SYS_C0070118	Check	"LOC_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
3	SYS_C0070119	Check	"CITY" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
4	SYS_C0070120	Check	"STATE" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: COMP_LOC

```
CREATE TABLE COMP_LOC
(
  LOC_ID VARCHAR2(20 BYTE) NOT NULL
, COMP_ID VARCHAR2(20 BYTE) NOT NULL
, CONTACT_NAME VARCHAR2(50 BYTE) NOT NULL
, CONTACT_EMAIL VARCHAR2(50 BYTE) NOT NULL
, CONTACT_PHONE VARCHAR2(20 BYTE)
, FAX VARCHAR2(20 BYTE)
, CONSTRAINT PK_COMP_LOC PRIMARY KEY (COMP_ID, LOC_ID)
 );
```

Data Dictionary:



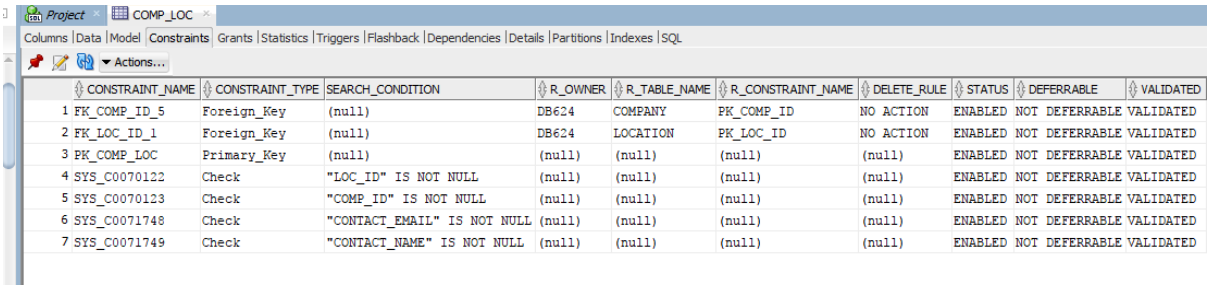
Project | COMP_LOC

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	LOC_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	COMP_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	CONTACT_NAME	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4	CONTACT_EMAIL	VARCHAR2(50 BYTE)	No	(null)	4	(null)
5	CONTACT_PHONE	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)
6	FAX	VARCHAR2(20 BYTE)	Yes	(null)	6	(null)

Constraints:



Project | COMP_LOC

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

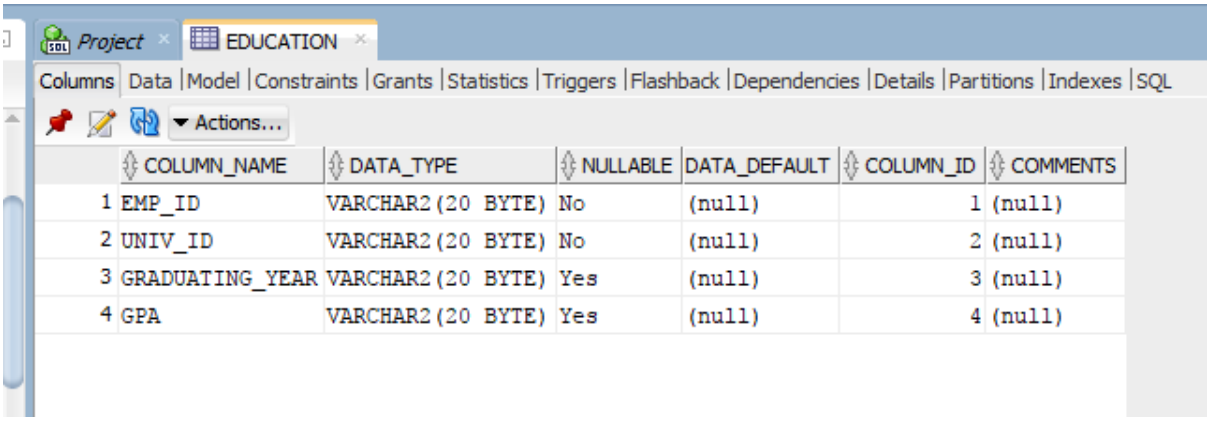
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	FK_COMP_ID_5	Foreign_Key	(null)	DB624	COMPANY	PK_COMP_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
2	FK_LOC_ID_1	Foreign_Key	(null)	DB624	LOCATION	PK_LOC_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
3	PK_COMP_LOC	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0070122	Check	"LOC_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
5	SYS_C0070123	Check	"COMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
6	SYS_C0071748	Check	"CONTACT_EMAIL" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
7	SYS_C0071749	Check	"CONTACT_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Table: EDUCATION

```
CREATE TABLE EDUCATION
(
  EMP_ID VARCHAR2(20 BYTE) NOT NULL
, UNIV_ID VARCHAR2(20 BYTE) NOT NULL
, GRADUATING_YEAR VARCHAR2(20 BYTE)
, GPA VARCHAR2(20 BYTE)
, CONSTRAINT PK_EDUCATION PRIMARY KEY ( EMP_ID , UNIV_ID )
  );
```

Data Dictionary:



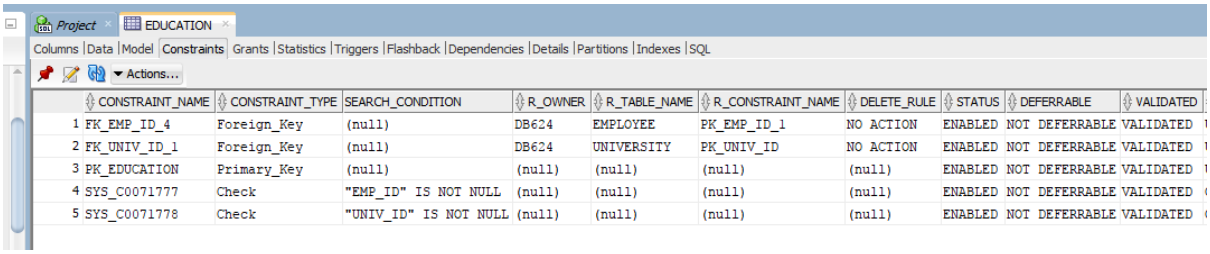
Project | EDUCATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMP_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	UNIV_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	GRADUATING_YEAR	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	GPA	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)

Constraints:



Project | EDUCATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

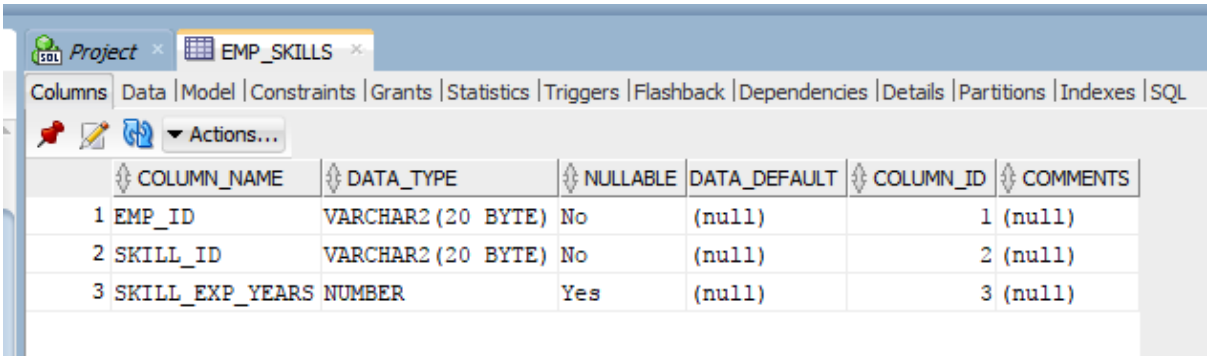
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	FK_EMP_ID_4	Foreign_Key	(null)	DB624	EMPLOYEE	PK_EMP_ID_1	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
2	FK_UNIV_ID_1	Foreign_Key	(null)	DB624	UNIVERSITY	PK_UNIV_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
3	PK_EDUCATION	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0071777	Check	"EMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
5	SYS_C0071778	Check	"UNIV_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Table: EMP_SKILLS

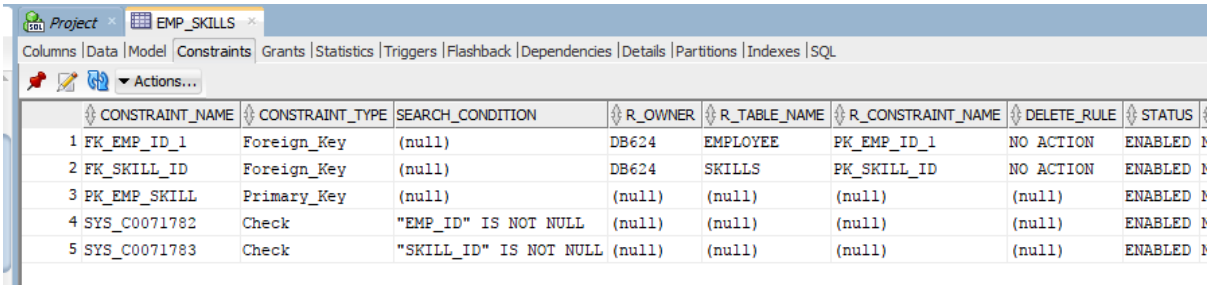
```
CREATE TABLE EMP_SKILLS
(
  EMP_ID VARCHAR2(20 BYTE) NOT NULL
, SKILL_ID VARCHAR2(20 BYTE) NOT NULL
, SKILL_EXP_YEARS NUMBER
, CONSTRAINT PK_EMP_SKILL PRIMARY KEY (SKILL_ID, EMP_ID)
 );
```

Data Dictionary:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMP_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	SKILL_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	SKILL_EXP_YEARS	NUMBER	Yes	(null)	3	(null)

Constraints:



	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS
1	FK_EMP_ID_1	Foreign_Key	(null)	DB624	EMPLOYEE	PK_EMP_ID_1	NO ACTION	ENABLED
2	FK_SKILL_ID	Foreign_Key	(null)	DB624	SKILLS	PK_SKILL_ID	NO ACTION	ENABLED
3	PK_EMP_SKILL	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED
4	SYS_C0071782	Check	"EMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED
5	SYS_C0071783	Check	"SKILL_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED

Table: EMPLOYEE

```
CREATE TABLE EMPLOYEE
(
  EMP_ID VARCHAR2(20 BYTE) NOT NULL
, EMP_NAME VARCHAR2(30 BYTE) NOT NULL
```

, CURRENT_COMPANY VARCHAR2(70 BYTE)

, CURRENT_LOCATION VARCHAR2(70 BYTE)

, CURRENT_ROLE VARCHAR2(70 BYTE)

, TOTAL_YEARS_EXP NUMBER

, NUMBER_COMP NUMBER

, GENDER VARCHAR2(20 BYTE) NOT NULL

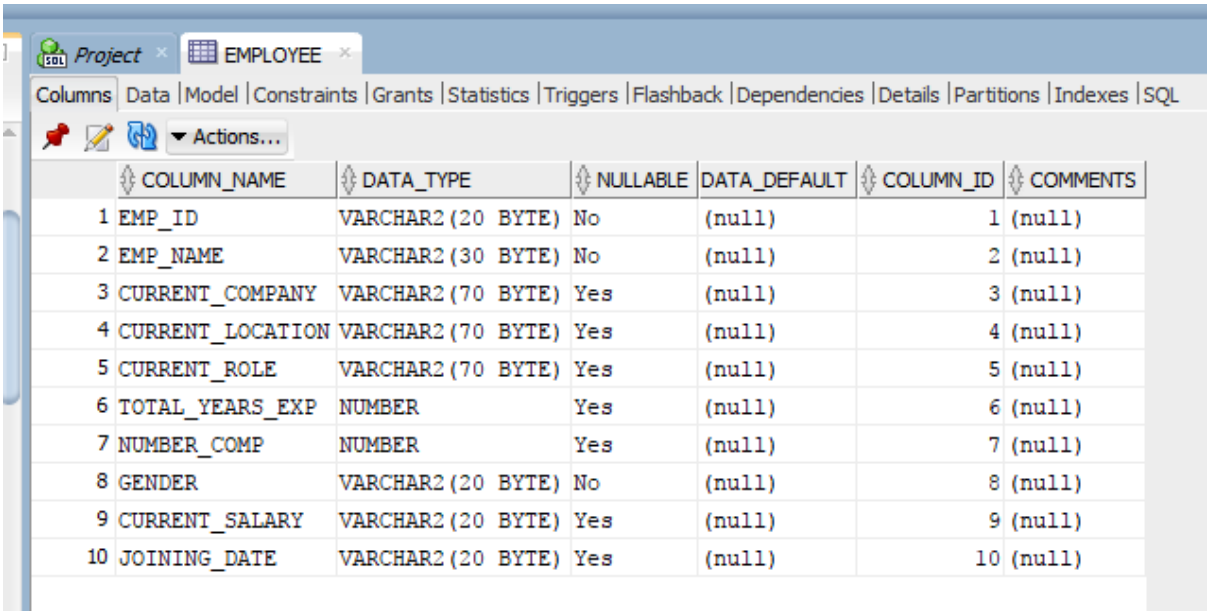
, CURRENT_SALARY VARCHAR2(20 BYTE)

, JOINING_DATE VARCHAR2(20 BYTE)

, CONSTRAINT PK_EMP_ID_1 PRIMARY KEY (EMP_ID)

);

Data Dictionary:



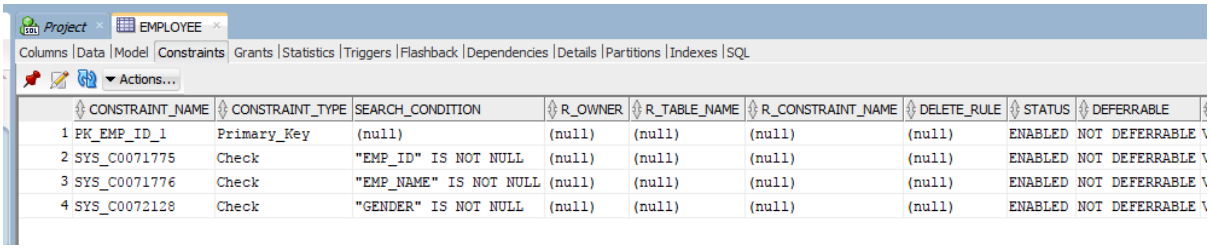
Project | EMPLOYEE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMP_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	EMP_NAME	VARCHAR2(30 BYTE)	No	(null)	2	(null)
3	CURRENT_COMPANY	VARCHAR2(70 BYTE)	Yes	(null)	3	(null)
4	CURRENT_LOCATION	VARCHAR2(70 BYTE)	Yes	(null)	4	(null)
5	CURRENT_ROLE	VARCHAR2(70 BYTE)	Yes	(null)	5	(null)
6	TOTAL_YEARS_EXP	NUMBER	Yes	(null)	6	(null)
7	NUMBER_COMP	NUMBER	Yes	(null)	7	(null)
8	GENDER	VARCHAR2(20 BYTE)	No	(null)	8	(null)
9	CURRENT_SALARY	VARCHAR2(20 BYTE)	Yes	(null)	9	(null)
10	JOINING_DATE	VARCHAR2(20 BYTE)	Yes	(null)	10	(null)

Constraints:



Project | EMPLOYEE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

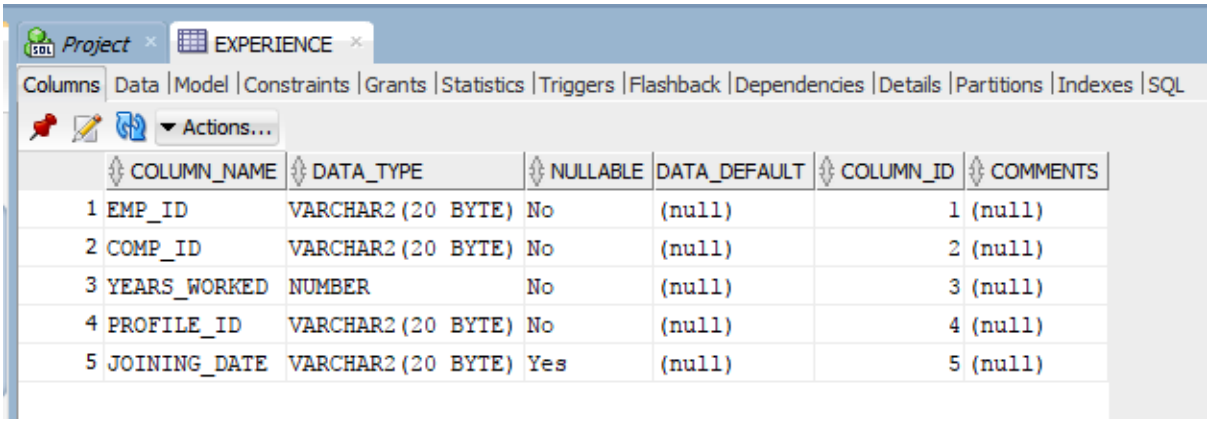
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	PK_EMP_ID_1	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
2	SYS_C0071775	Check	"EMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
3	SYS_C0071776	Check	"EMP_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
4	SYS_C0072128	Check	"GENDER" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: EXPERIENCE

```
CREATE TABLE EXPERIENCE
(
  EMP_ID VARCHAR2(20 BYTE) NOT NULL
, COMP_ID VARCHAR2(20 BYTE) NOT NULL
, YEARS_WORKED NUMBER NOT NULL
, PROFILE_ID VARCHAR2(20 BYTE) NOT NULL
, JOINING_DATE VARCHAR2(20 BYTE)
, CONSTRAINT PK_EMP_COMP_PROFILE PRIMARY KEY ( EMP_ID , COMP_ID , PROFILE_ID )
  );
```

Data Dictionary:



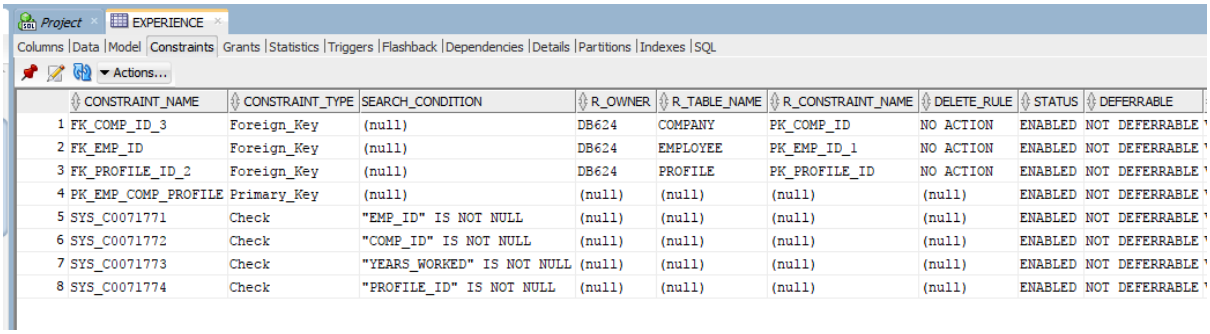
Project | EXPERIENCE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMP_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	COMP_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	YEARS_WORKED	NUMBER	No	(null)	3	(null)
4	PROFILE_ID	VARCHAR2(20 BYTE)	No	(null)	4	(null)
5	JOINING_DATE	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)

Constraints:



Project | EXPERIENCE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

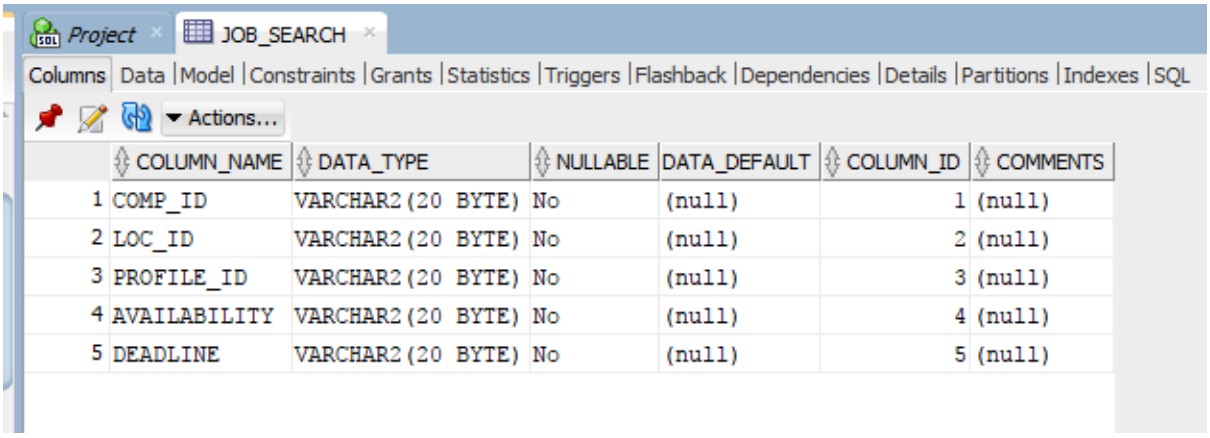
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	FK_COMP_ID_3	Foreign_Key	(null)	DB624	COMPANY	PK_COMP_ID	NO ACTION	ENABLED	NOT DEFERRABLE
2	FK_EMP_ID	Foreign_Key	(null)	DB624	EMPLOYEE	PK_EMP_ID_1	NO ACTION	ENABLED	NOT DEFERRABLE
3	FK_PROFILE_ID_2	Foreign_Key	(null)	DB624	PROFILE	PK_PROFILE_ID	NO ACTION	ENABLED	NOT DEFERRABLE
4	PK_EMP_COMP_PROFILE	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
5	SYS_C0071771	Check	"EMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
6	SYS_C0071772	Check	"COMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
7	SYS_C0071773	Check	"YEARS_WORKED" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
8	SYS_C0071774	Check	"PROFILE_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: JOB_SEARCH

```
CREATE TABLE JOB_SEARCH
(
  COMP_ID VARCHAR2(20 BYTE) NOT NULL
, LOC_ID VARCHAR2(20 BYTE) NOT NULL
, PROFILE_ID VARCHAR2(20 BYTE) NOT NULL
, AVAILABILITY VARCHAR2(20 BYTE) NOT NULL
, DEADLINE VARCHAR2(20 BYTE) NOT NULL
, CONSTRAINT PK_COMP_LOC_PROFILE PRIMARY KEY ( COMP_ID , LOC_ID , PROFILE_ID )
  );
```

Data Dictionary:



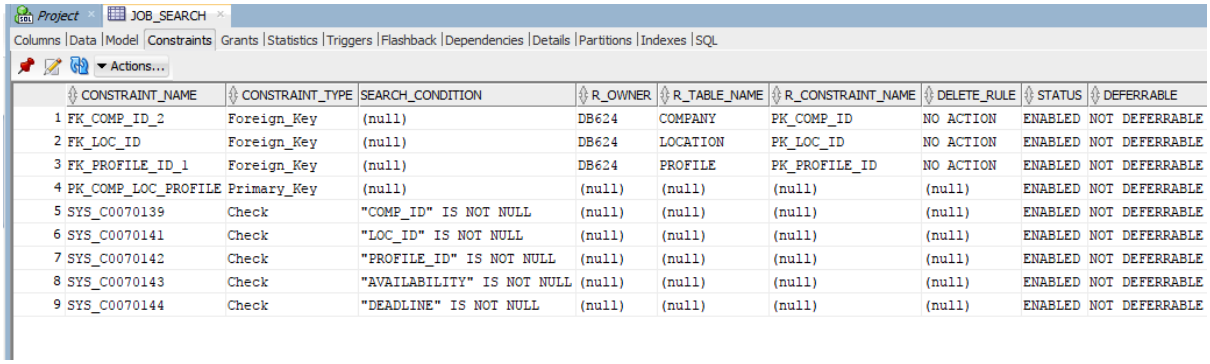
Project | JOB_SEARCH

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COMP_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	LOC_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	PROFILE_ID	VARCHAR2(20 BYTE)	No	(null)	3	(null)
4	AVAILABILITY	VARCHAR2(20 BYTE)	No	(null)	4	(null)
5	DEADLINE	VARCHAR2(20 BYTE)	No	(null)	5	(null)

Constraints:



Project | JOB_SEARCH

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

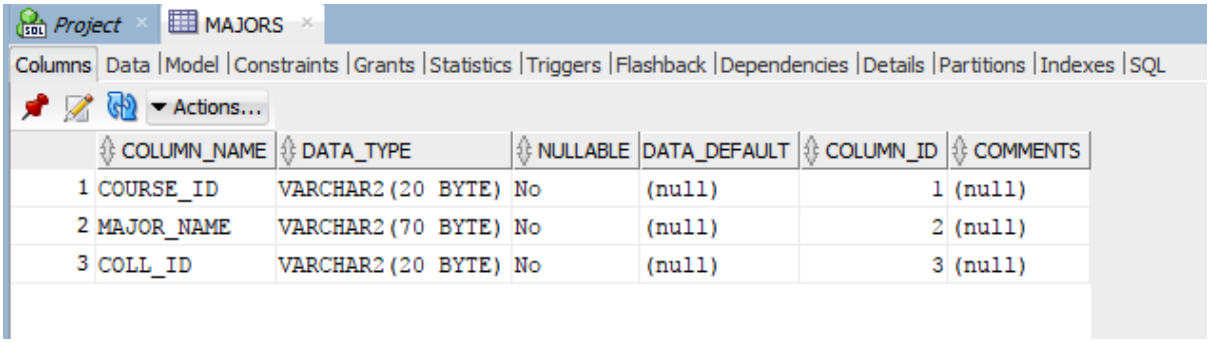
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	FK_COMP_ID_2	Foreign_Key	(null)	DB624	COMPANY	PK_COMP_ID	NO ACTION	ENABLED	NOT DEFERRABLE
2	FK_LOC_ID	Foreign_Key	(null)	DB624	LOCATION	PK_LOC_ID	NO ACTION	ENABLED	NOT DEFERRABLE
3	FK_PROFILE_ID_1	Foreign_Key	(null)	DB624	PROFILE	PK_PROFILE_ID	NO ACTION	ENABLED	NOT DEFERRABLE
4	PK_COMP_LOC_PROFILE	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
5	SYS_C0070139	Check	"COMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
6	SYS_C0070141	Check	"LOC_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
7	SYS_C0070142	Check	"PROFILE_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
8	SYS_C0070143	Check	"AVAILABILITY" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
9	SYS_C0070144	Check	"DEADLINE" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: MAJORS

```
CREATE TABLE MAJORS
(
  COURSE_ID VARCHAR2(20 BYTE) NOT NULL
, MAJOR_NAME VARCHAR2(70 BYTE) NOT NULL
, COLL_ID VARCHAR2(20 BYTE) NOT NULL
, CONSTRAINT PK_MAJOR_ID PRIMARY KEY (COURSE_ID)
  );
```

Data Dictionary:



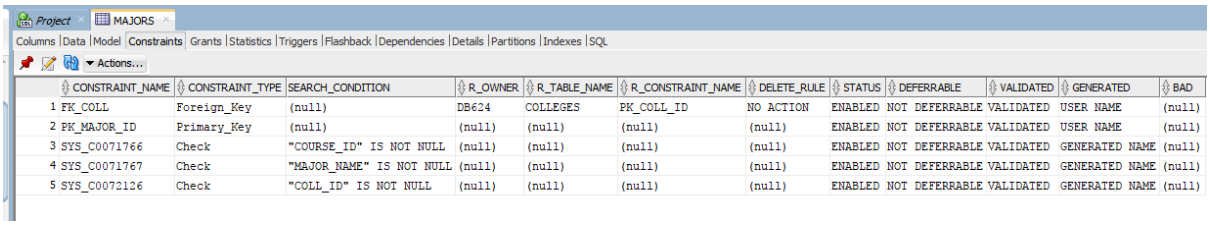
Project | MAJORS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COURSE_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	MAJOR_NAME	VARCHAR2(70 BYTE)	No	(null)	2	(null)
3	COLL_ID	VARCHAR2(20 BYTE)	No	(null)	3	(null)

Constraints:



Project | MAJORS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED	GENERATED	BAD
1	FK_COLL	Foreign_Key	(null)	DB624	COLLEGES	PK_COLL_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)
2	PK_MAJOR_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)
3	SYS_C0071766	Check	"COURSE_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)
4	SYS_C0071767	Check	"MAJOR_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)
5	SYS_C0072126	Check	"COLL_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)

Table: PROFILE

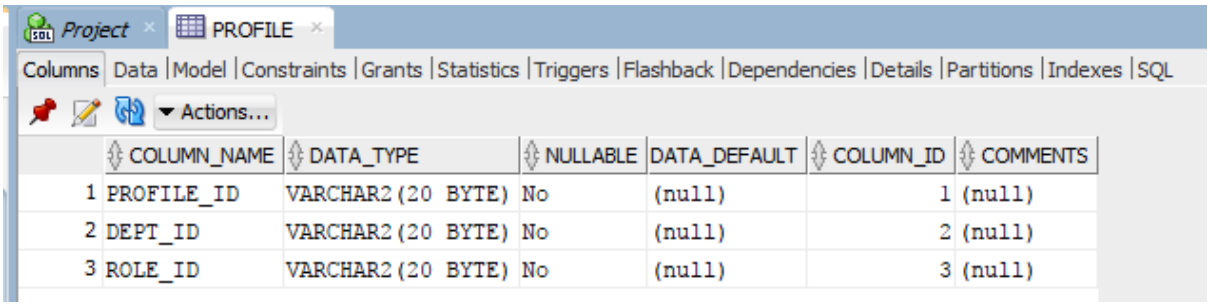
```
CREATE TABLE PROFILE
(
  PROFILE_ID VARCHAR2(20 BYTE) NOT NULL
, DEPT_ID VARCHAR2(20 BYTE) NOT NULL
```

, ROLE_ID VARCHAR2(20 BYTE) NOT NULL

, CONSTRAINT PK_PROFILE_ID PRIMARY KEY (PROFILE_ID)

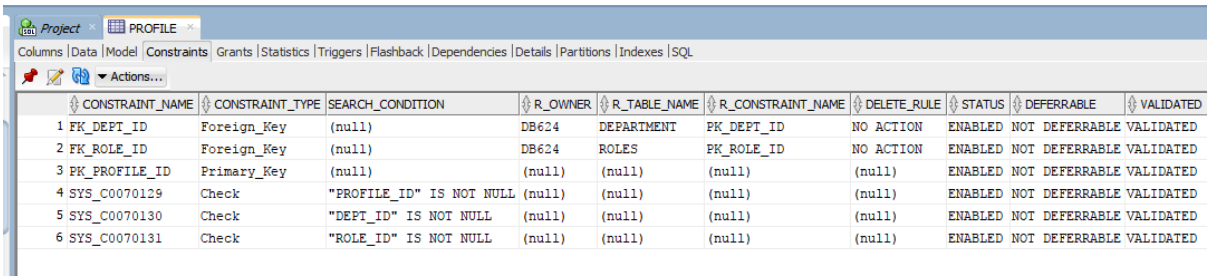
  );

Data Dictionary:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	PROFILE_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	DEPT_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	ROLE_ID	VARCHAR2(20 BYTE)	No	(null)	3	(null)

Constraints:



	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	FK_DEPT_ID	Foreign_Key	(null)	DB624	DEPARTMENT	PK_DEPT_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
2	FK_ROLE_ID	Foreign_Key	(null)	DB624	ROLES	PK_ROLE_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
3	PK_PROFILE_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0070129	Check	"PROFILE_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
5	SYS_C0070130	Check	"DEPT_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
6	SYS_C0070131	Check	"ROLE_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Table: QUALIFICATION

CREATE TABLE QUALIFICATION

(

  PROFILE_ID VARCHAR2(20 BYTE) NOT NULL

, COMP_ID VARCHAR2(20 BYTE) NOT NULL

, MIN_AGE VARCHAR2(20 BYTE)

, MIN_WORK VARCHAR2(20 BYTE)

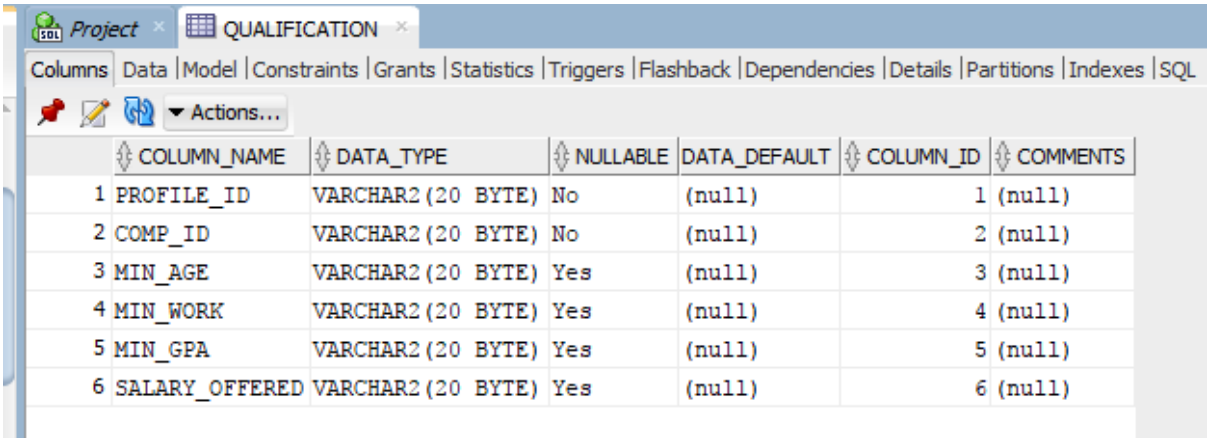
, MIN_GPA VARCHAR2(20 BYTE)

, SALARY_OFFERED VARCHAR2(20 BYTE)

, CONSTRAINT PK_COMP_PROFILE PRIMARY KEY (PROFILE_ID, COMP_ID)

  );

Data Dictionary:



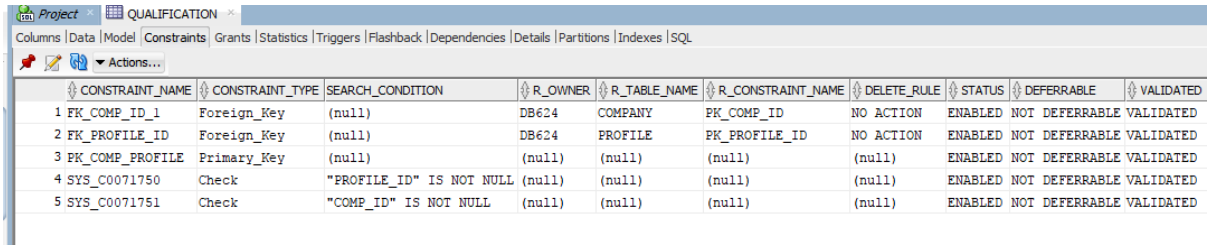
Project | QUALIFICATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	PROFILE_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	COMP_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	MIN_AGE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	MIN_WORK	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	MIN_GPA	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)
6	SALARY_OFFERED	VARCHAR2(20 BYTE)	Yes	(null)	6	(null)

Constraints:



Project | QUALIFICATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

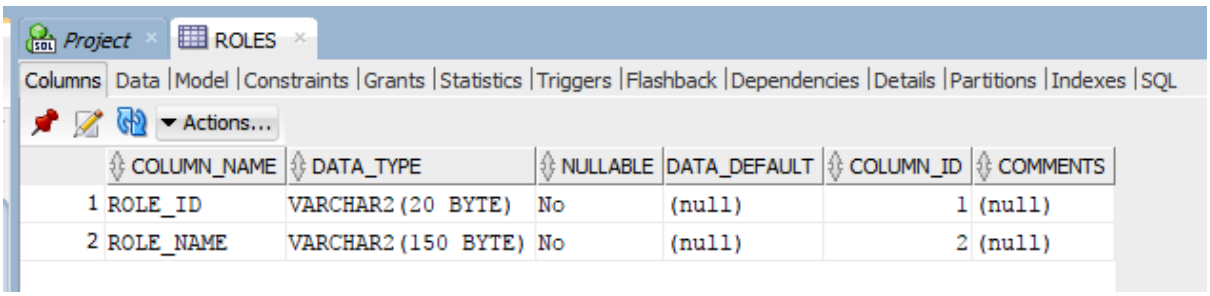
Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	FK_COMP_ID_1	Foreign_Key	(null)	DB624	COMPANY	PK_COMP_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
2	FK_PROFILE_ID	Foreign_Key	(null)	DB624	PROFILE	PK_PROFILE_ID	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
3	PK_COMP_PROFILE	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0071750	Check	"PROFILE_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
5	SYS_C0071751	Check	"COMP_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Table: ROLES

```
CREATE TABLE ROLES
(
  ROLE_ID VARCHAR2(20 BYTE) NOT NULL
, ROLE_NAME VARCHAR2(150 BYTE) NOT NULL
, CONSTRAINT PK_ROLE_ID PRIMARY KEY (ROLE_ID)
  );
```

Data Dictionary:



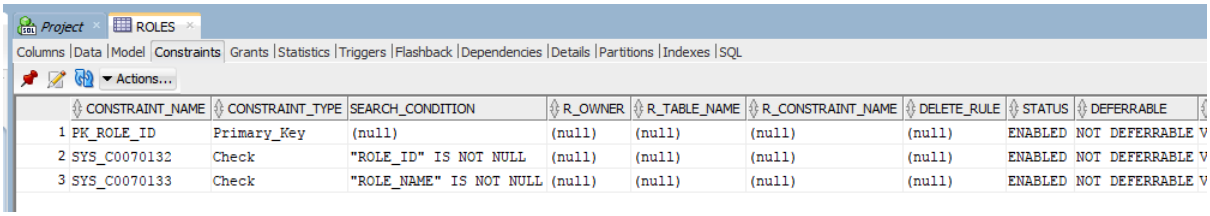
Project | ROLES

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ROLE_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	ROLE_NAME	VARCHAR2(150 BYTE)	No	(null)	2	(null)

Constraints:

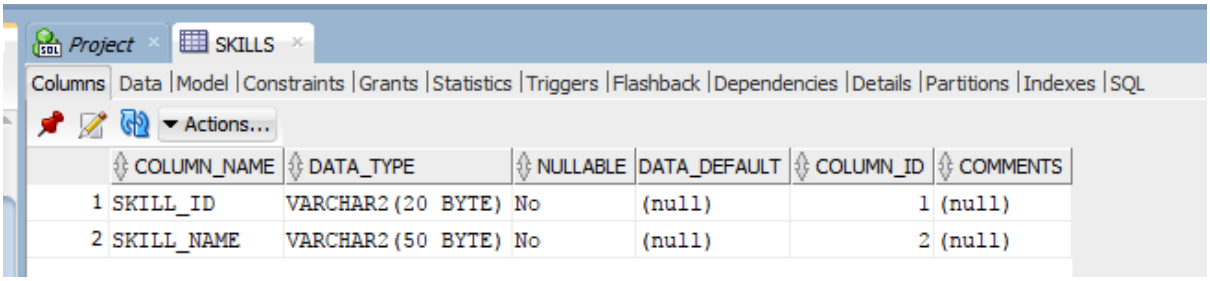


	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	PK_ROLE_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
2	SYS_C0070132	Check	"ROLE_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
3	SYS_C0070133	Check	"ROLE_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: SKILLS

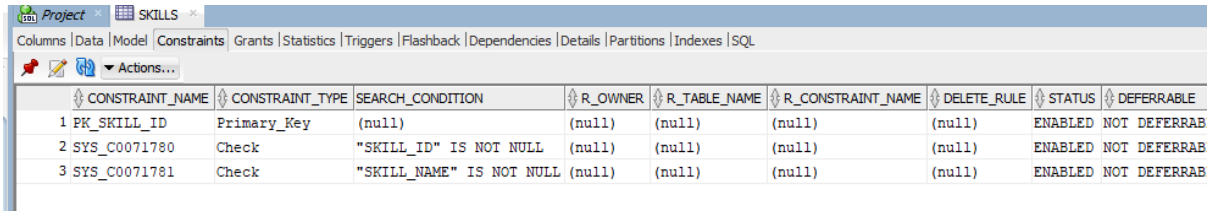
```
CREATE TABLE SKILLS
(
  SKILL_ID VARCHAR2(20 BYTE) NOT NULL
, SKILL_NAME VARCHAR2(50 BYTE) NOT NULL
, CONSTRAINT PK_SKILL_ID PRIMARY KEY (SKILL_ID)
  );
```

Data Dictionary:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SKILL_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	SKILL_NAME	VARCHAR2(50 BYTE)	No	(null)	2	(null)

Constraints:

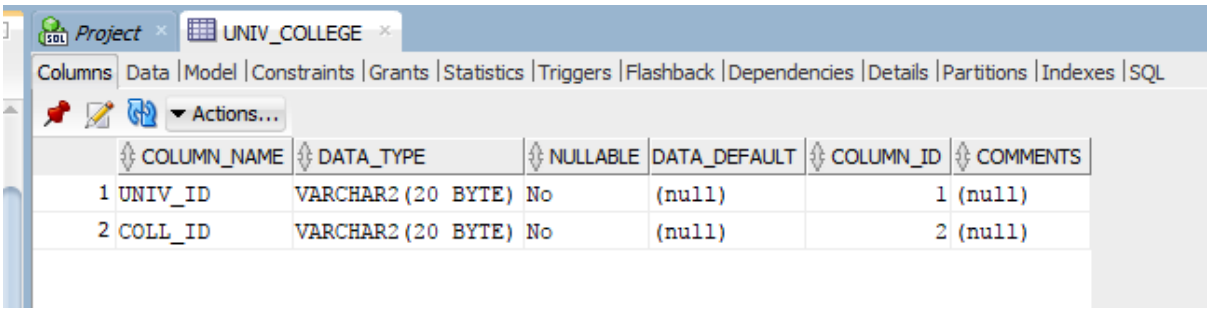


	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	PK_SKILL_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRAB
2	SYS_C0071780	Check	"SKILL_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRAB
3	SYS_C0071781	Check	"SKILL_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRAB

Table: UNIV_COLLEGES

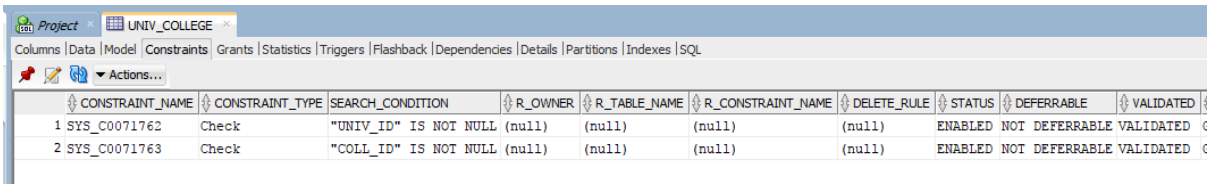
```
CREATE TABLE UNIV_COLLEGE
(
  UNIV_ID VARCHAR2(20 BYTE) NOT NULL
, COLL_ID VARCHAR2(20 BYTE) NOT NULL
);
```

Data Dictionary:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	UNIV_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	COLL_ID	VARCHAR2(20 BYTE)	No	(null)	2	(null)

Constraints:

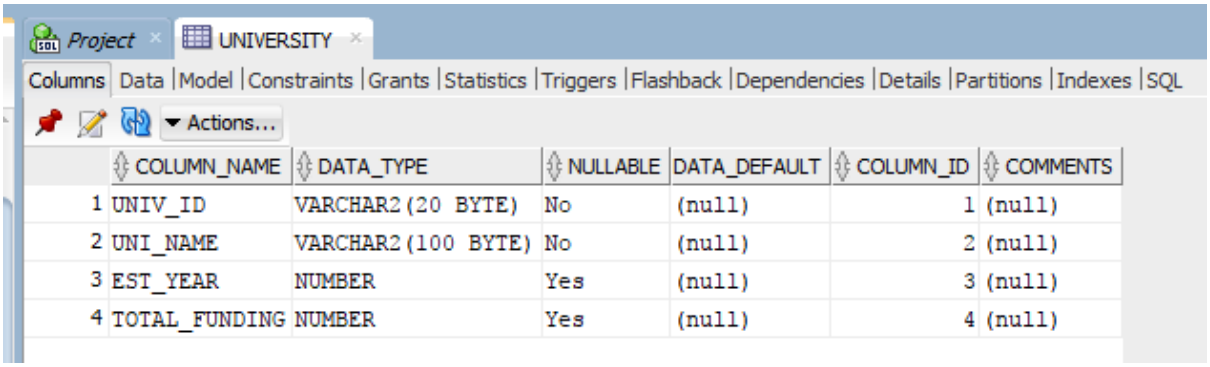


	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	SYS_C0071762	Check	"UNIV_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0071763	Check	"COLL_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Table: UNIVERSITY

```
CREATE TABLE UNIVERSITY
(
  UNIV_ID VARCHAR2(20 BYTE) NOT NULL
, UNI_NAME VARCHAR2(100 BYTE) NOT NULL
, EST_YEAR NUMBER
, TOTAL_FUNDING NUMBER
, CONSTRAINT PK_UNIV_ID PRIMARY KEY (UNIV_ID)
  );
```

Data Dictionary:



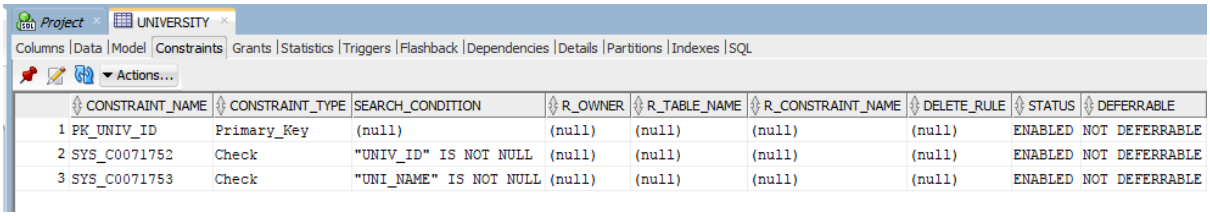
Project | UNIVERSITY

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	UNIV_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	UNI_NAME	VARCHAR2(100 BYTE)	No	(null)	2	(null)
3	EST_YEAR	NUMBER	Yes	(null)	3	(null)
4	TOTAL_FUNDING	NUMBER	Yes	(null)	4	(null)

Constraints:



Project | UNIVERSITY

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	PK_UNIV_ID	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
2	SYS_C0071752	Check	"UNIV_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
3	SYS_C0071753	Check	"UNI_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

Table: USERS

```
CREATE TABLE USERS
(
USER_ID VARCHAR2(20 BYTE) NOT NULL
, FIRST_NAME VARCHAR2(20 BYTE) NOT NULL
, EMAIL VARCHAR2(40 BYTE)
, GENDER VARCHAR2(20 BYTE)
, PASSWORD VARCHAR2(20 BYTE) NOT NULL
, LAST_NAME VARCHAR2(20 BYTE) NOT NULL
, USER_AGE NUMBER
);
```

Data Dictionary:

Project | USERS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	USER_ID	VARCHAR2(20 BYTE)	No	(null)	1	(null)
2	FIRST_NAME	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3	EMAIL	VARCHAR2(40 BYTE)	Yes	(null)	3	(null)
4	GENDER	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	PASSWORD	VARCHAR2(20 BYTE)	No	(null)	5	(null)
6	LAST_NAME	VARCHAR2(20 BYTE)	No	(null)	6	(null)
7	USER_AGE	NUMBER	Yes	(null)	7	(null)

Constraints:

Project | USERS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE
1	SYS_C0071742	Check	"USER_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
2	SYS_C0071744	Check	"FIRST_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
3	SYS_C0071745	Check	"PASSWORD" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE
4	SYS_C0071832	Check	"LAST_NAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE

DATA GENERATION AND LOADING:

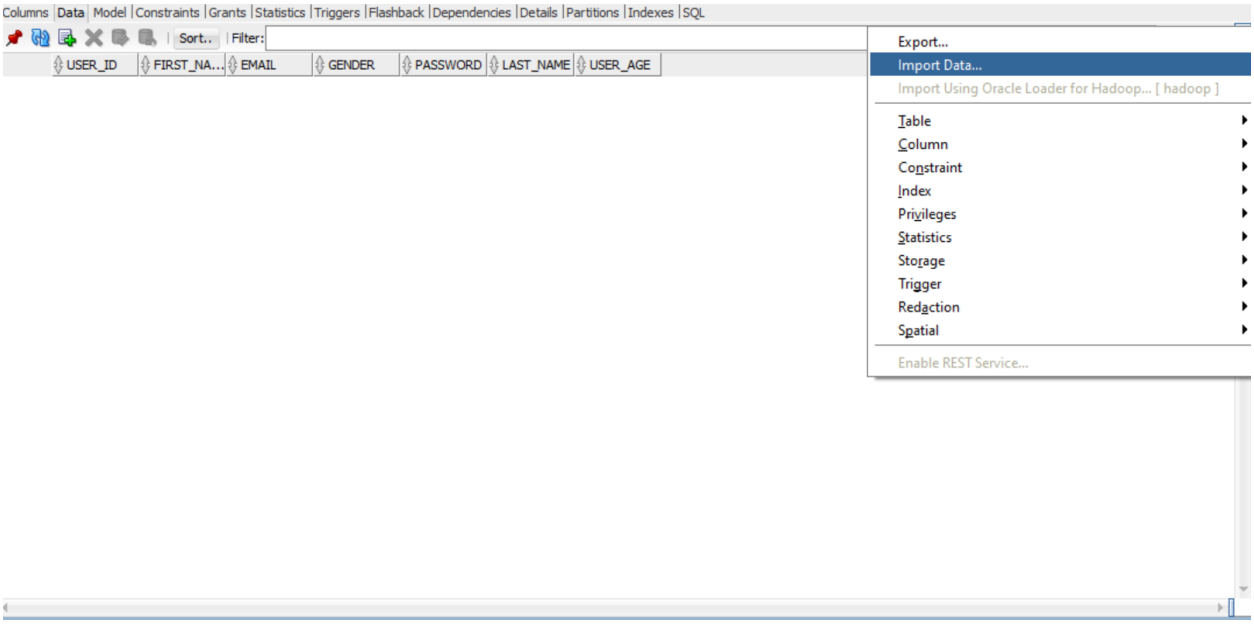
The data is created mainly from data generating website https://www.mockaroo.com/

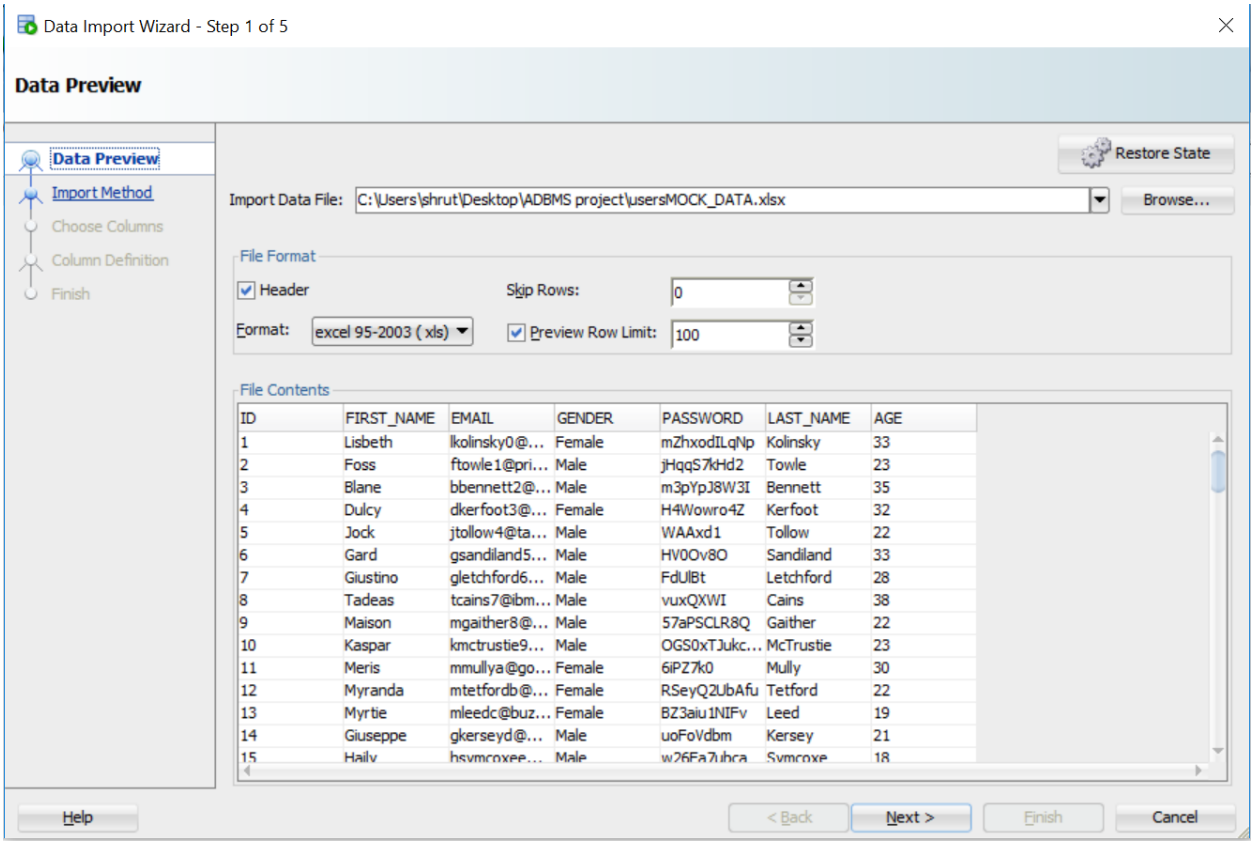
For the University table, we have considered the dataset giving the data on all major Universities in US.

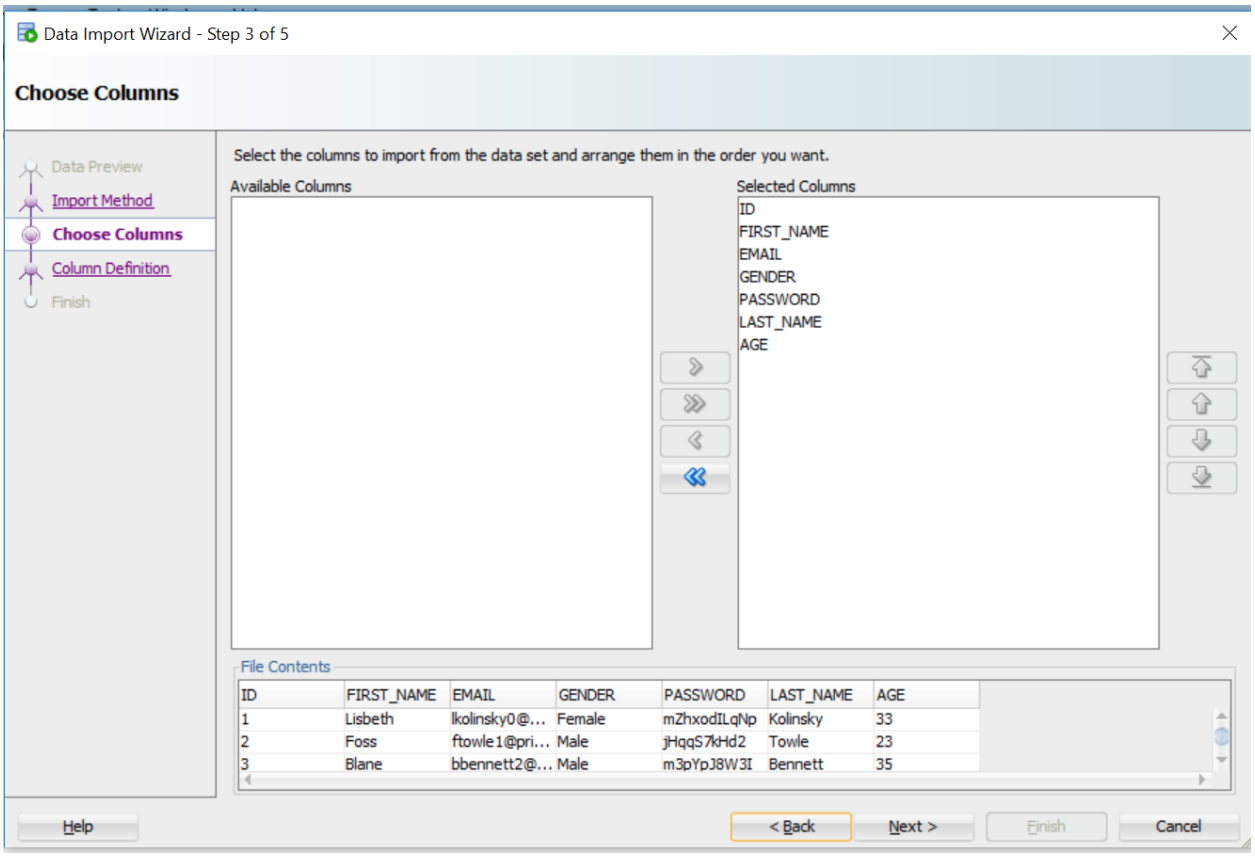
Data is loaded for the below tables:

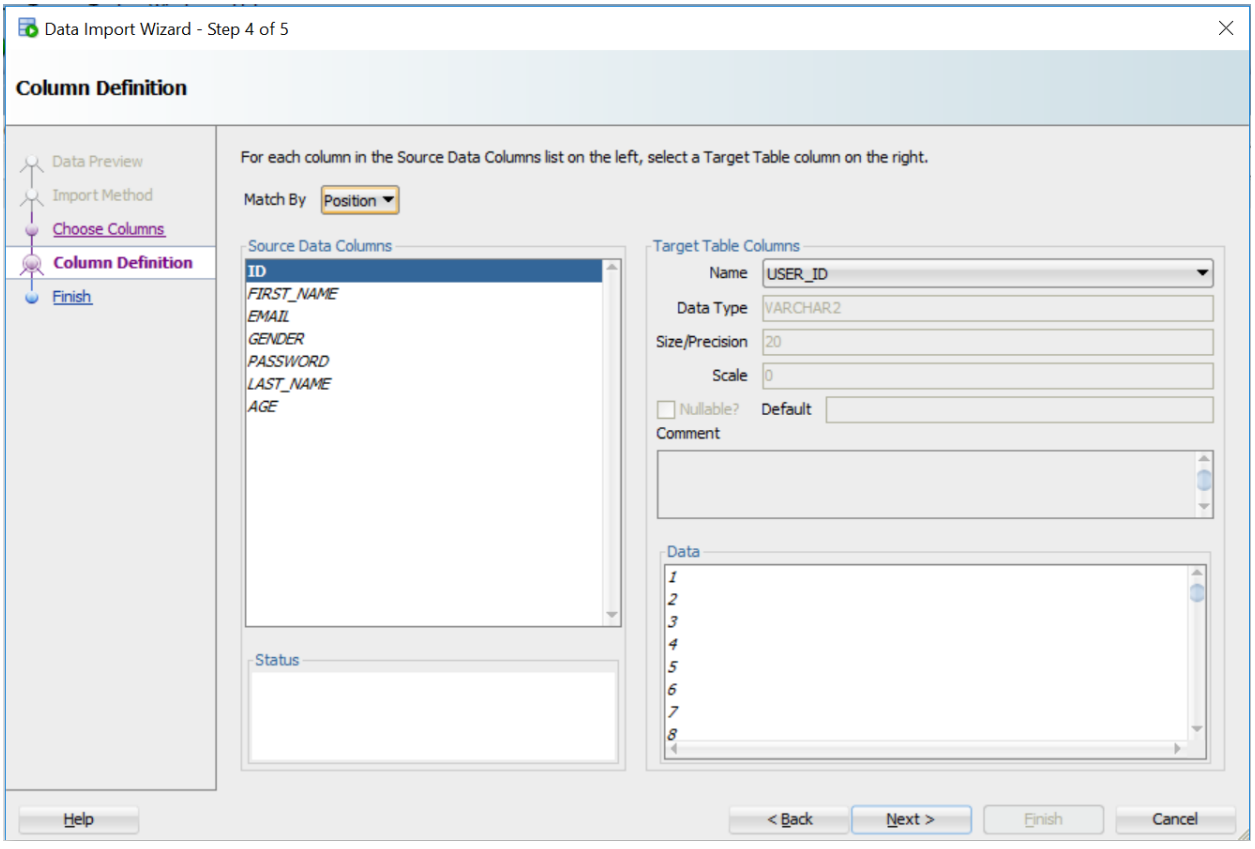
TABLE	NUMBER OF RECORDS
COLLEGES	21
COMP_DEPT	510
COMP_LOC	510
COMPANY	2946
DEPARTMENT	30
DOMAIN	25
EDUCATION	1000
EMP_SKILLS	1618
EMPLOYEE	5874
EXPERIENCE	1000
JOB_SEARCH	1000
LOCATION	4409
MAJORS	248
PROFILE	1000
QUALIFICATION	1000
ROLES	8550
SKILLS	928
UNIV_COLLEGE	566
UNIV_LOCATION	567
UNIVERSITY	4000
USERS	1000

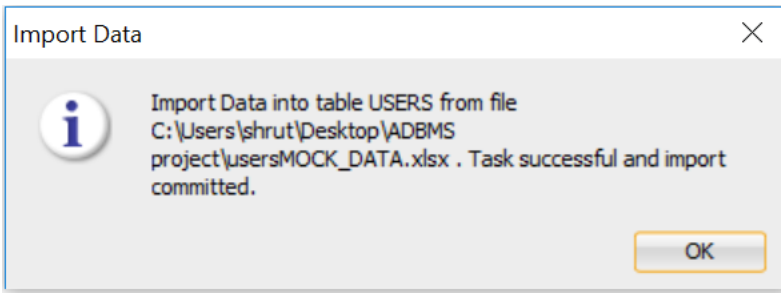
The data is loaded into DB using import wizard as shown below.






Data Import Wizard - Step 3 of 5
Choose Columns
Data Preview
Import Method
Choose Columns
Column Definition
Finish
Select the columns to import from the data set and arrange them in the order you want.
Available Columns
Selected Columns
ID
FIRST_NAME
EMAIL
GENDER
PASSWORD
LAST_NAME
AGE
File Contents
ID FIRST_NAME EMAIL GENDER PASSWORD LAST_NAME AGE
1 Lisbeth lkolinsky0@... Female mZhxodILqNp Kolinsky 33
2 Foss ftowle1@pri... Male jHqqS7kHd2 Towle 23
3 Blane bbennett2@... Male m3pYpJ8W3I Bennett 35
Help
< Back
Next >
Finish
Cancel


Data Import Wizard - Step 4 of 5
Column Definition
Data Preview
Import Method
Choose Columns
Column Definition
Finish
For each column in the Source Data Columns list on the left, select a Target Table column on the right.
Match By
Position
Source Data Columns
ID
FIRST_NAME
EMAIL
GENDER
PASSWORD
LAST_NAME
AGE
Target Table Columns
Name
USER_ID
Data Type
VARCHAR2
Size/Precision
20
Scale
0
Nullable?
Default
Comment
Data
1
2
3
4
5
6
7
8
Status
Help
< Back
Next >
Finish
Cancel



PERFORMANCE TUNING:

Indexing:

Indexes are used by queries to retrieve the data from database tables quickly. When a column is indexed in a table or cluster, a schema object is created that contains an entry for each value that appears in the indexed column(s), which produces direct fast access to the rows.

There are several types of Index structures available depending on the need:

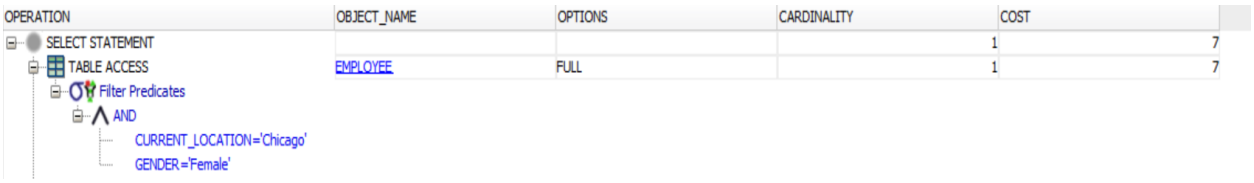
- B-tree index is in the form of a binary tree and is the default index type.
- Bitmap index has a bitmap for each distinct value indexed, and each bit position represents a row that may or may not contain the indexed value. It is best for low-cardinality columns.

B + TREE INDEXES:

a: When the details for currently employed Females in Chicago location is searched for.

Before Indexing:

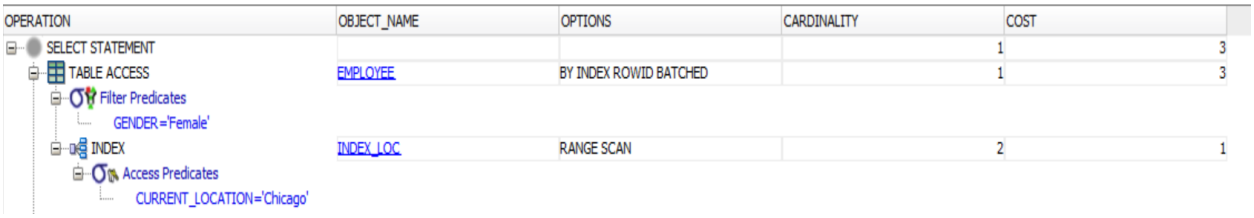
```
SELECT *
FROM EMPLOYEE
WHERE GENDER='Female' and CURRENT_LOCATION='Chicago';
```



Creating Index:

```
CREATE INDEX INDEX_LOC
 ON EMPLOYEE(CURRENT_LOCATION);
```

After Indexing:

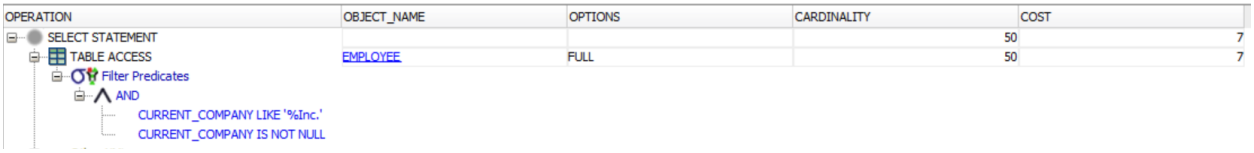


From the above explanation plan it can be clearly seen that Index is used in the query operation to search for a location. Due to the presence of the index, the cost has decreased from 7 to 3.

b: Searching for the employee details who works for Inc. companies:

Before Indexing:

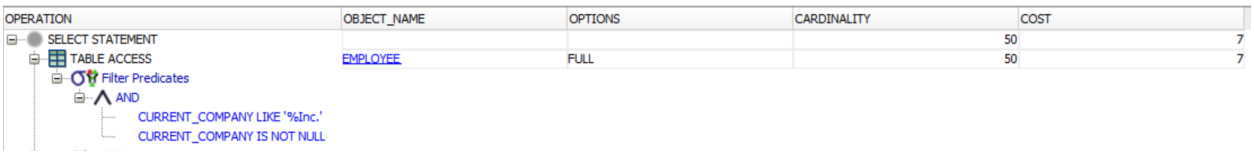
```
SELECT *
FROM EMPLOYEE
WHERE CURRENT_COMPANY like '%Inc.';
```



Creating Index:

```
CREATE INDEX INDEX_CURRENT_COMP
ON EMPLOYEE(CURRENT_COMPANY);
```

After Indexing:



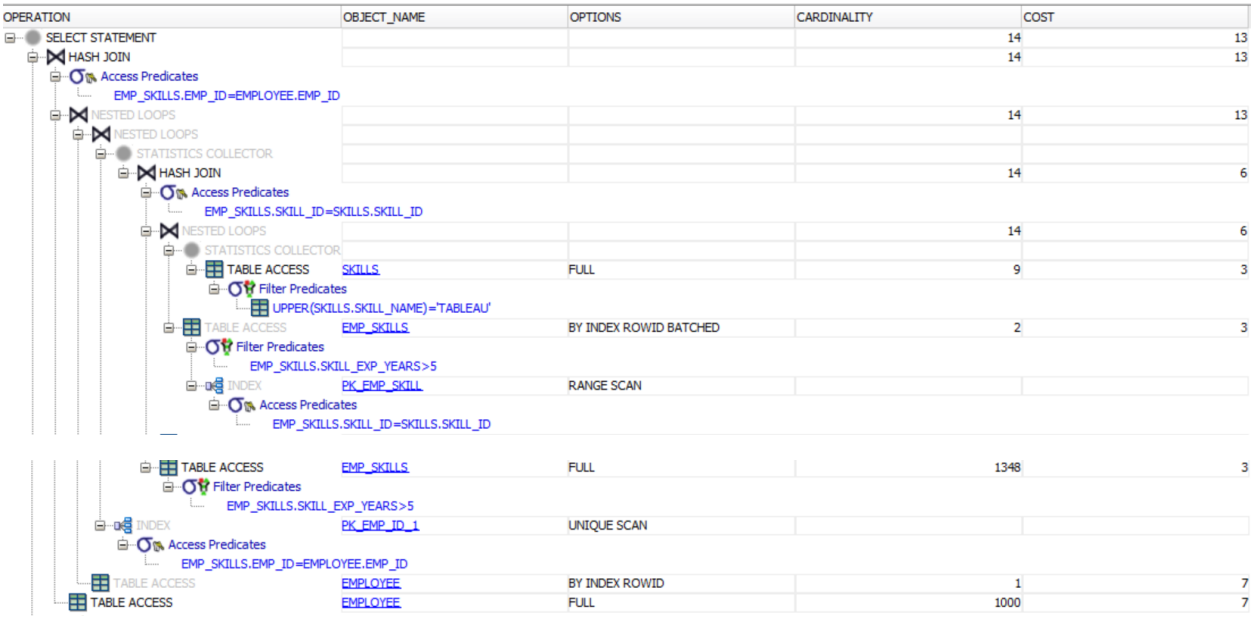
Here for the above case, when searching for a specific string in the Current Company, there is no change in the cost. The reason being B-Tree indexes search using binary algorithm, which sorts

the Current Company column based on the first letter used in the where clause, as we are searching for a specific part of text, the indexing strategy will not be of any help.

PRIMARY INDEX:

Searching for the employees who are experienced in Tableau for more than 5 years.

```
SELECT EMPLOYEE.EMP_NAME, EMP_SKILLS.SKILL_EXP_YEARS

FROM EMP_SKILLS

JOIN SKILLS ON EMP_SKILLS.SKILL_ID=SKILLS.SKILL_ID

JOIN EMPLOYEE ON EMP_SKILLS.EMP_ID=EMPLOYEE.EMP_ID

WHERE UPPER(SKILLS.SKILL_NAME) ='TABLEAU' AND
EMP_SKILLS.SKILL_EXP_YEARS >'5';
```



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	13
HASH JOIN			14	13
Access Predicates				
EMP_SKILLS.EMP_ID=EMPLOYEE.EMP_ID				
NESTED LOOPS			14	13
NESTED LOOPS				
STATISTICS COLLECTOR				
HASH JOIN			14	6
Access Predicates				
EMP_SKILLS.SKILL_ID=SKILLS.SKILL_ID				
NESTED LOOPS			14	6
STATISTICS COLLECTOR				
TABLE ACCESS	SKILLS	FULL	9	3
Filter Predicates				
UPPER(SKILLS.SKILL_NAME)='TABLEAU'				
TABLE ACCESS	EMP_SKILLS	BY INDEX ROWID BATCHED	2	3
Filter Predicates				
EMP_SKILLS.SKILL_EXP_YEARS>5				
INDEX	PK_EMP_SKILL	RANGE SCAN		
Access Predicates				
EMP_SKILLS.SKILL_ID=SKILLS.SKILL_ID				
TABLE ACCESS	EMP_SKILLS	FULL	1348	3
Filter Predicates				
EMP_SKILLS.SKILL_EXP_YEARS>5				
INDEX	PK_EMP_ID_1	UNIQUE SCAN		
Access Predicates				
EMP_SKILLS.EMP_ID=EMPLOYEE.EMP_ID				
TABLE ACCESS	EMPLOYEE	BY INDEX ROWID	1	7
TABLE ACCESS	EMPLOYEE	FULL	1000	7

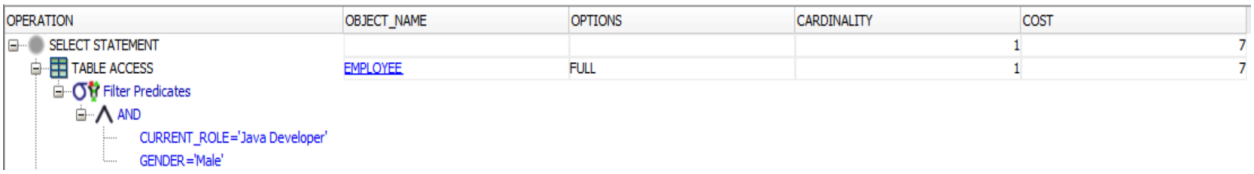
We know that there would be UNIQUE key constraints created on the Primary keys for each table. As per the execution plan seen above, for the query operation which involved joining three tables using the JOIN statement, the primary indexes for each table came into action. As a result, we have utilized our existing unique indexes and reduced the cost of the operation.

MULTI-CLOUMN INDEXING:

Users would like to search for number of Male Java Developers currently employed, so it would be wise to use a multi column indexing for faster output of results to users for providing a better customer service.

Before Indexing:

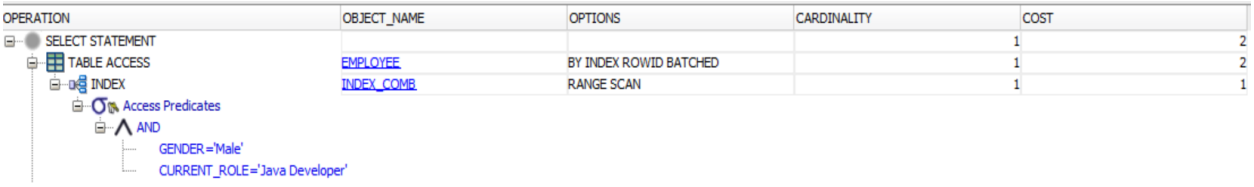
```
SELECT * FROM EMPLOYEE

WHERE GENDER='Male' AND CURRENT_ROLE = 'Java Developer';
```



Create Indexing:

```
CREATE INDEX INDEX_COMBINATION

ON EMPLOYEE (GENDER, CURRENT_ROLE);
```

After Indexing:



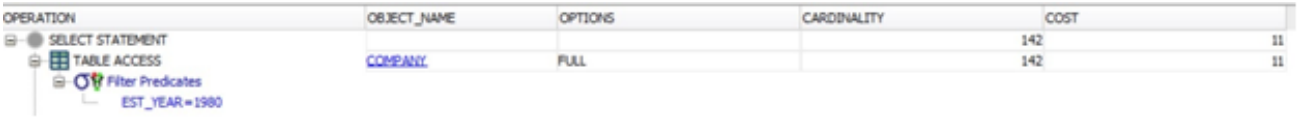
As seen from the above execution plan, the cost has dropped from 7 to 2 with the use of multi-column indexing. The execution time for the query has also decreased.

BITMAP INDEXES:

a: When users want to search for the companies that are established in a particular year. An index created on the EST_YEAR column would produce optimized results.

Before Indexing:

```
SELECT *

FROM COMPANY

WHERE EST_YEAR = 1980;
```

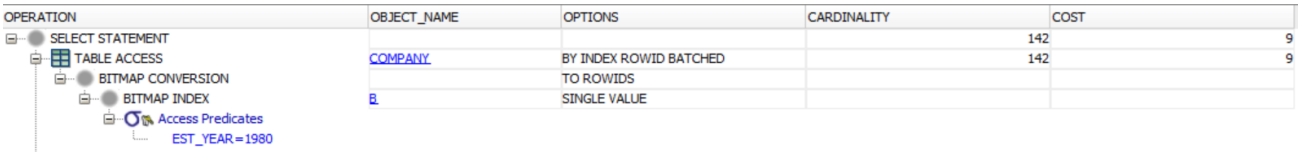


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			142	11
TABLE ACCESS	COMPANY	FULL	142	11
Filter Predicates				
EST_YEAR=1980				

Create Indexing:

```
CREATE BITMAP INDEX B
ON COMPANY(EST_YEAR);
```

After Indexing:



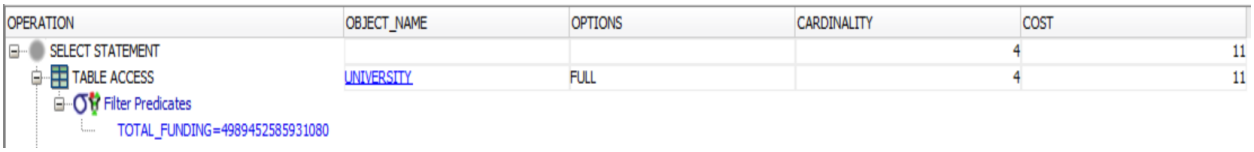
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			142	9
TABLE ACCESS	COMPANY	BY INDEX ROWID BATCHED	142	9
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	B	SINGLE VALUE		
Access Predicates				
EST_YEAR=1980				

As we can see from the execution plan, the bitmap index created on the EST_YEAR column has been called into the operation & the cost of the operation has reduced.

b: When users want to search for a university based on its total funding available. An index created on the TOTAL_FUNDING column would produce optimized results.

Before Indexing:

```
SELECT *
FROM UNIVERSITY
WHERE TOTAL_FUNDING='4989452585931080';
```

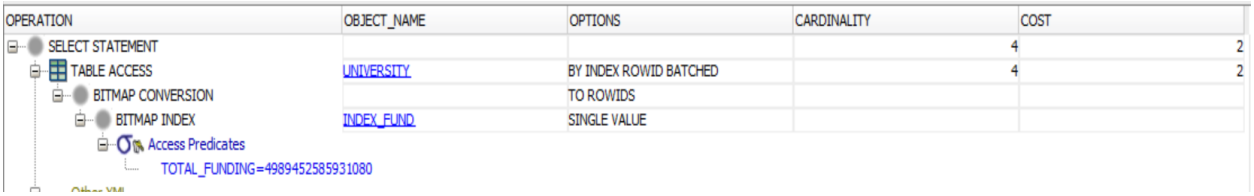


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4	11
TABLE ACCESS	UNIVERSITY	FULL	4	11
Filter Predicates				
TOTAL_FUNDING=4989452585931080				

Create Indexing:

```
CREATE BITMAP INDEX INDEX_FUND
ON UNIVERSITY(TOTAL_FUNDING);
```

After Indexing:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4	2
TABLE ACCESS	UNIVERSITY	BY INDEX ROWID BATCHED	4	2
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	INDEX_FUND	SINGLE VALUE		
Access Predicates				
TOTAL_FUNDING=4989452585931080				

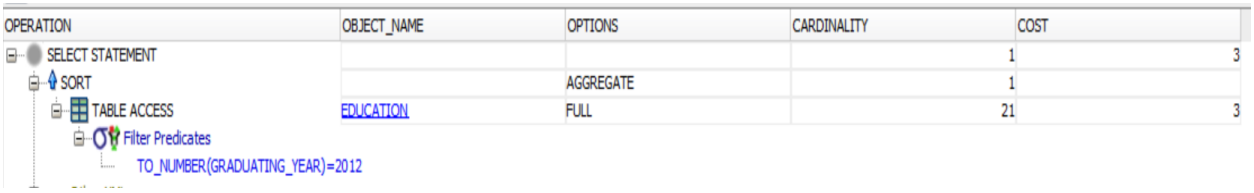
As seen from the execution plan, the bitmap index created on the TOTAL_FUNDING column has been called into the operation. The cost has decreased from 11 to 2 in this scenario.

MULTI-COLUMN BITMAP INDEX:

Users would like to search for average GPA for a particular Graduating year, so it would be wise to use a multi column indexing for faster output of results**.**

Before Indexing:

```
SELECT AVG(EDUCATION.GPA) AS AVERAGE_GPA
FROM EDUCATION
WHERE GRADUATING_YEAR=2012;
```

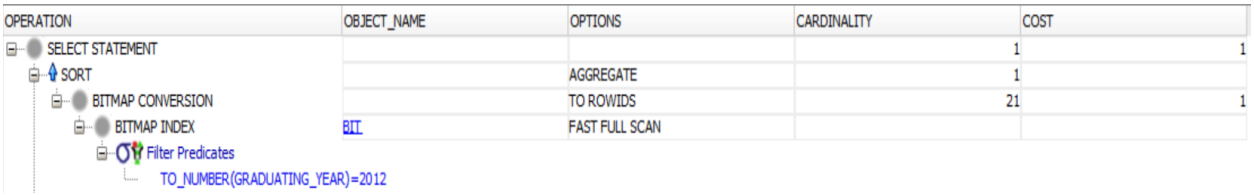


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	3
SORT		AGGREGATE	1	
TABLE ACCESS	EDUCATION	FULL	21	3
Filter Predicates				
TO_NUMBER(GRADUATING_YEAR)=2012				

Create Indexing:

```
CREATE BITMAP INDEX BIT
ON EDUCATION (GPA, GRADUATING_YEAR);
```

After Indexing:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
SORT		AGGREGATE	1	
BITMAP CONVERSION		TO ROWIDS	21	1
BITMAP INDEX	BIT	FAST FULL SCAN		
Filter Predicates				
TO_NUMBER(GRADUATING_YEAR)=2012				

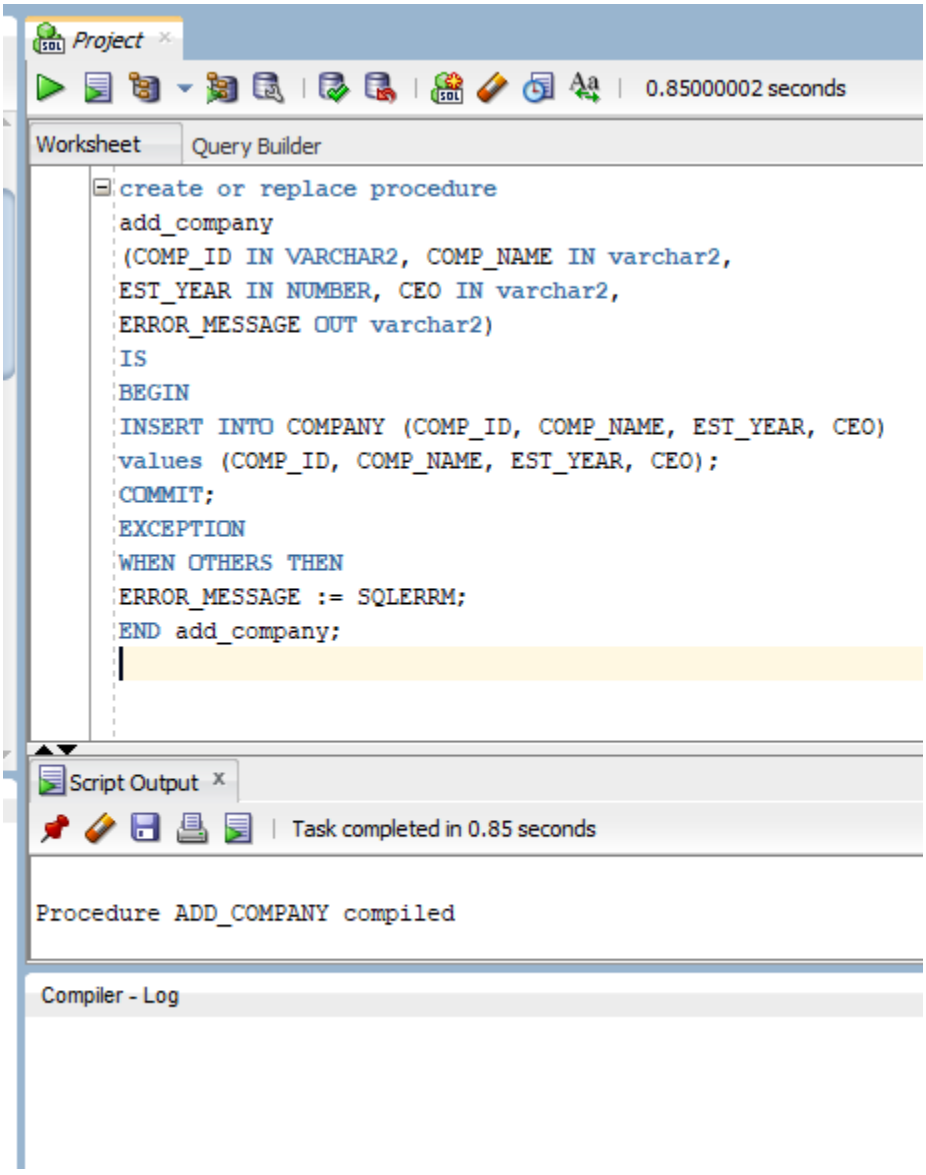
As seen from the execution plan, the bitmap index created as a combination of two columns: GPA & TOTAL_FUNDING column has been called into the operation. As a result, a fast full scan has taken place since the index contains all the columns that are needed for the query and the cost has decreased from 3 to 1 in this scenario.

DATABASE PROGRAMMING:

Stored Procedures:

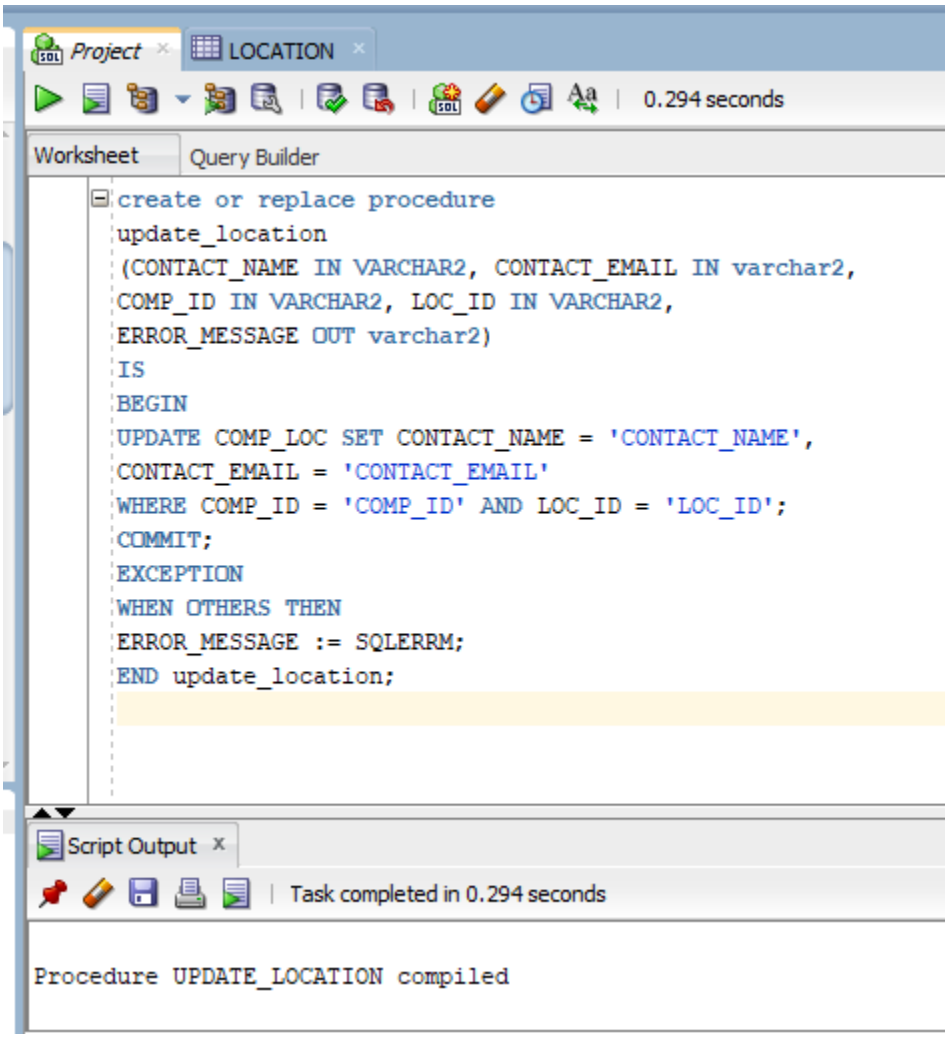
A **stored procedure** is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in the SQL developer. Procedures are compiled only once, and it can be run many times. It does not return any value.

1) **Procedure to Insert a row**
In the below **add_company** procedure, we have created a set of statements to insert the values into company table. With the help of this procedure we can insert the values by calling the procedure name and passing the arguments in it.



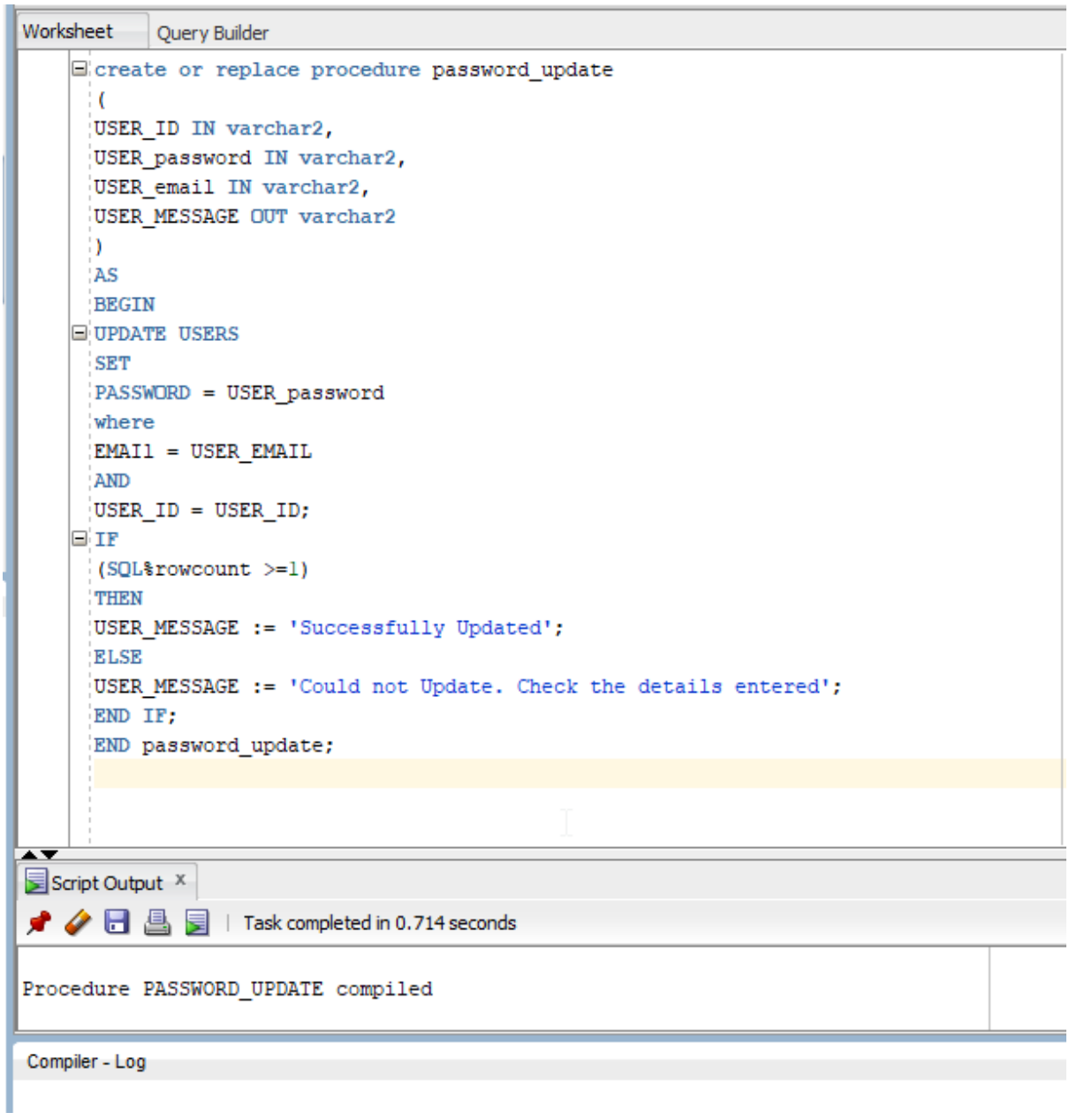
2) **Procedure to update a row**

In the below **update_location** procedure, we have written sql statements to update the contact information of a company at a particular location in the COMP_LOC table. With the help of this procedure we can update the contact by calling the procedure name and passing the appropriate arguments in it.



3) **Procedure to update user password.**

In the below **password_update** procedure, we have created a set of statements to update the password in the Users table. With the help of this procedure we can Update the password by calling the procedure name and passing the appropriate arguments in it.

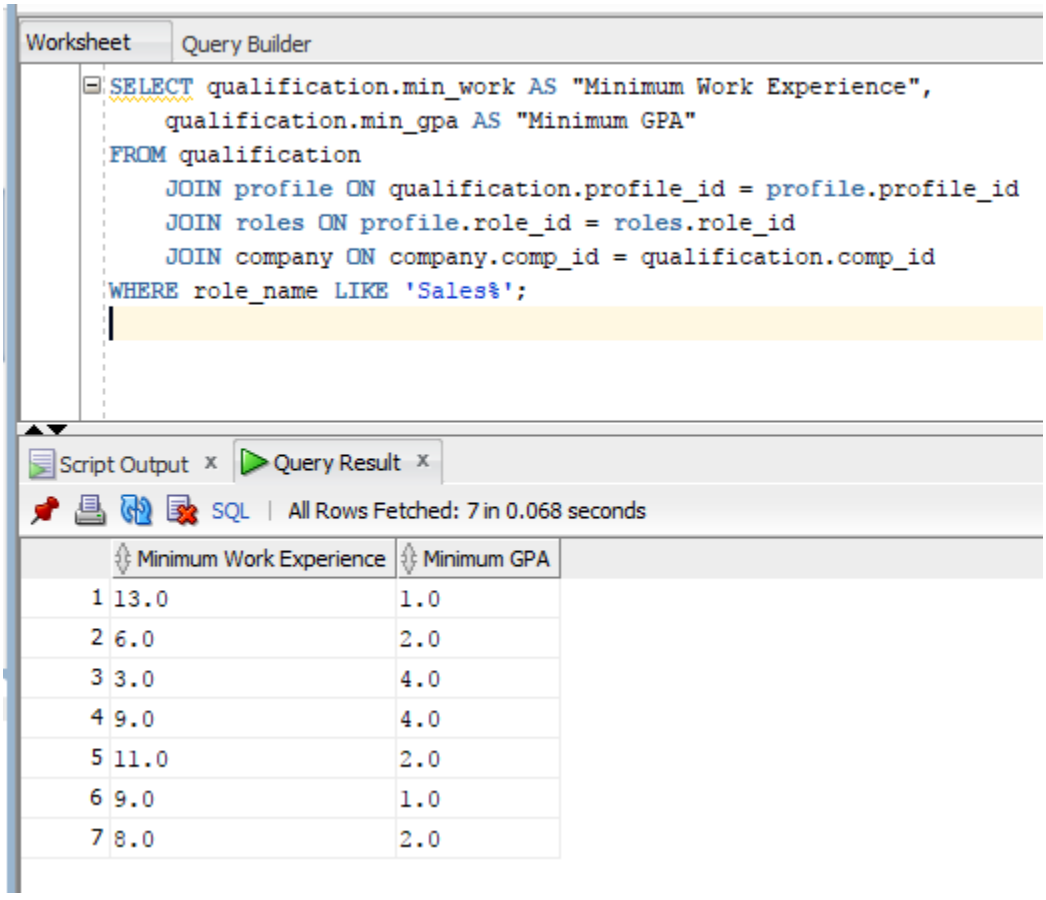


SQL QUERYING:

1) What is the Min work exp & Min GPA required for a sales role in any company.

SQL code:

```
SELECT QUALIFICATION.MIN_WORK AS "Minimum Work Experience",
   QUALIFICATION.MIN_GPA AS "Minimum Gpa"
FROM QUALIFICATION
   JOIN PROFILE ON QUALIFICATION.PROFILE_ID = PROFILE.PROFILE_ID
   JOIN ROLES ON PROFILE.ROLE_ID = ROLES.ROLE_ID
   JOIN COMPANY ON COMPANY.COMP_ID = QUALIFICATION.COMP_ID
WHERE ROLE_NAME LIKE 'Sales%';
```

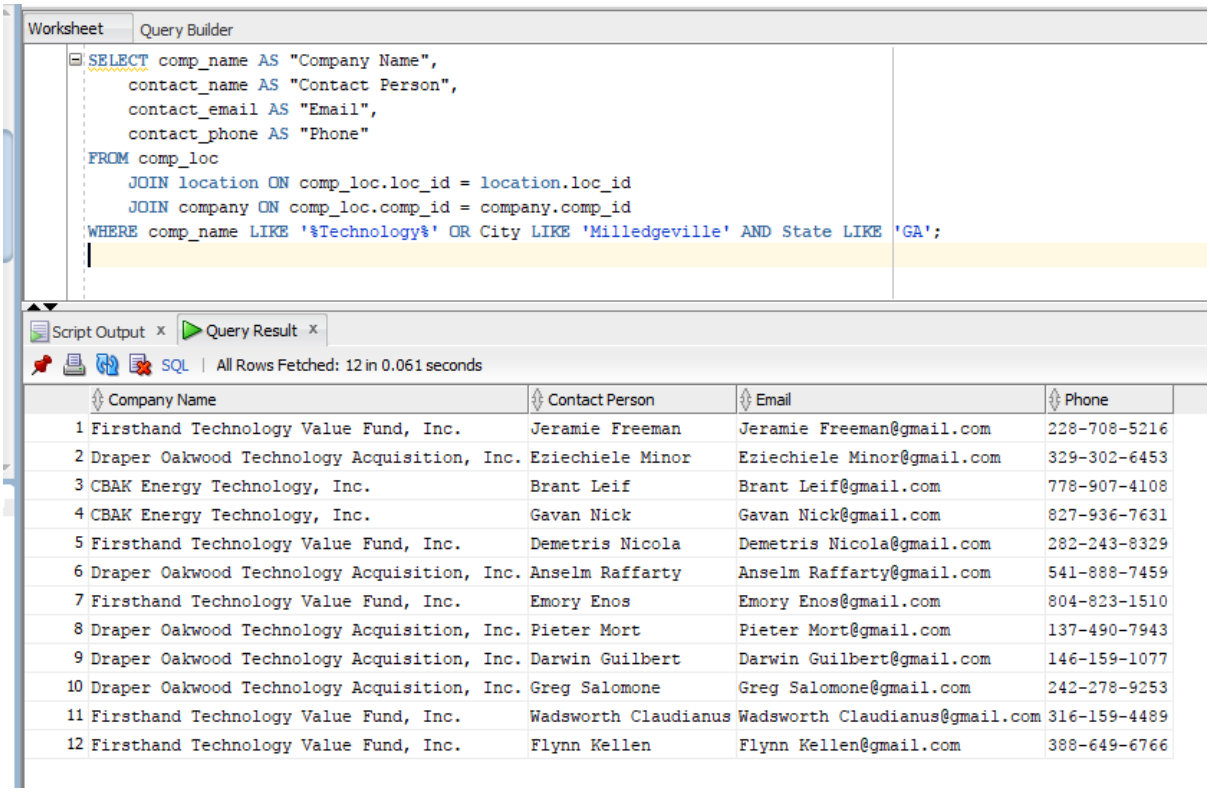


	Minimum Work Experience	Minimum GPA
1	13.0	1.0
2	6.0	2.0
3	3.0	4.0
4	9.0	4.0
5	11.0	2.0
6	9.0	1.0
7	8.0	2.0

2) What is the contact information for a technology company at Milledgeville and Georgia state.

SQL code:

```
SELECT COMP_NAME AS "Company Name",
    CONTACT_NAME AS "Contact Person",
    CONTACT_EMAIL AS "Email",
    CONTACT_PHONE AS "Phone"
FROM COMP_LOC
    JOIN LOCATION ON COMP_LOC.LOC_ID = LOCATION.LOC_ID
    JOIN COMPANY ON COMP_LOC.COMP_ID = COMPANY.COMP_ID
WHERE COMP_NAME LIKE '%Technology%' OR CITY LIKE 'milledgeville' AND STATE LIKE 'GA';
```



Worksheet | Query Builder

```
SELECT comp_name AS "Company Name",
    contact_name AS "Contact Person",
    contact_email AS "Email",
    contact_phone AS "Phone"
FROM comp_loc
    JOIN location ON comp_loc.loc_id = location.loc_id
    JOIN company ON comp_loc.comp_id = company.comp_id
WHERE comp_name LIKE '%Technology%' OR City LIKE 'Milledgeville' AND State LIKE 'GA';
```

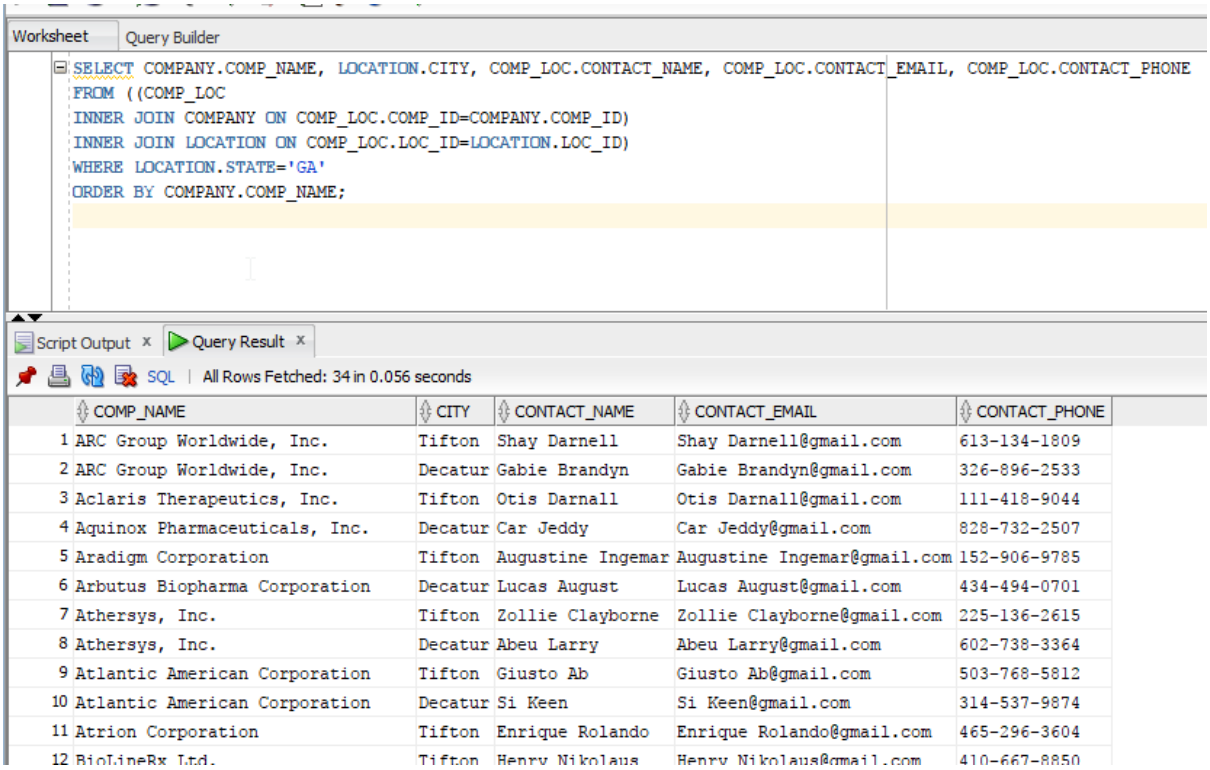
Script Output | Query Result | SQL | All Rows Fetched: 12 in 0.061 seconds

	Company Name	Contact Person	Email	Phone
1	Firsthand Technology Value Fund, Inc.	Jeramie Freeman	Jeramie Freeman@gmail.com	228-708-5216
2	Draper Oakwood Technology Acquisition, Inc.	Eziechiele Minor	Eziechiele Minor@gmail.com	329-302-6453
3	CBAK Energy Technology, Inc.	Brant Leif	Brant Leif@gmail.com	778-907-4108
4	CBAK Energy Technology, Inc.	Gavan Nick	Gavan Nick@gmail.com	827-936-7631
5	Firsthand Technology Value Fund, Inc.	Demetris Nicola	Demetris Nicola@gmail.com	282-243-8329
6	Draper Oakwood Technology Acquisition, Inc.	Anselm Raffarty	Anselm Raffarty@gmail.com	541-888-7459
7	Firsthand Technology Value Fund, Inc.	Emory Enos	Emory Enos@gmail.com	804-823-1510
8	Draper Oakwood Technology Acquisition, Inc.	Pieter Mort	Pieter Mort@gmail.com	137-490-7943
9	Draper Oakwood Technology Acquisition, Inc.	Darwin Guilbert	Darwin Guilbert@gmail.com	146-159-1077
10	Draper Oakwood Technology Acquisition, Inc.	Greg Salomone	Greg Salomone@gmail.com	242-278-9253
11	Firsthand Technology Value Fund, Inc.	Wadsworth Claudianus	Wadsworth Claudianus@gmail.com	316-159-4489
12	Firsthand Technology Value Fund, Inc.	Flynn Kellen	Flynn Kellen@gmail.com	388-649-6766

3) What is the contact information for all the companies located in various cities in Georgia.

SQL code:

```
SELECT COMPANY.COMP_NAME, LOCATION.CITY, COMP_LOC.CONTACT_NAME,
COMP_LOC.CONTACT_EMAIL, COMP_LOC.CONTACT_PHONE

FROM ((COMP_LOC

INNER JOIN COMPANY ON COMP_LOC.COMP_ID=COMPANY.COMP_ID)

INNER JOIN LOCATION ON COMP_LOC.LOC_ID=LOCATION.LOC_ID)

WHERE LOCATION.STATE='GA'

ORDER BY COMPANY.COMP_NAME;
```



Worksheet | Query Builder

```
SELECT COMPANY.COMP_NAME, LOCATION.CITY, COMP_LOC.CONTACT_NAME, COMP_LOC.CONTACT_EMAIL, COMP_LOC.CONTACT_PHONE
FROM ((COMP_LOC
INNER JOIN COMPANY ON COMP_LOC.COMP_ID=COMPANY.COMP_ID)
INNER JOIN LOCATION ON COMP_LOC.LOC_ID=LOCATION.LOC_ID)
WHERE LOCATION.STATE='GA'
ORDER BY COMPANY.COMP_NAME;
```

Script Output | Query Result

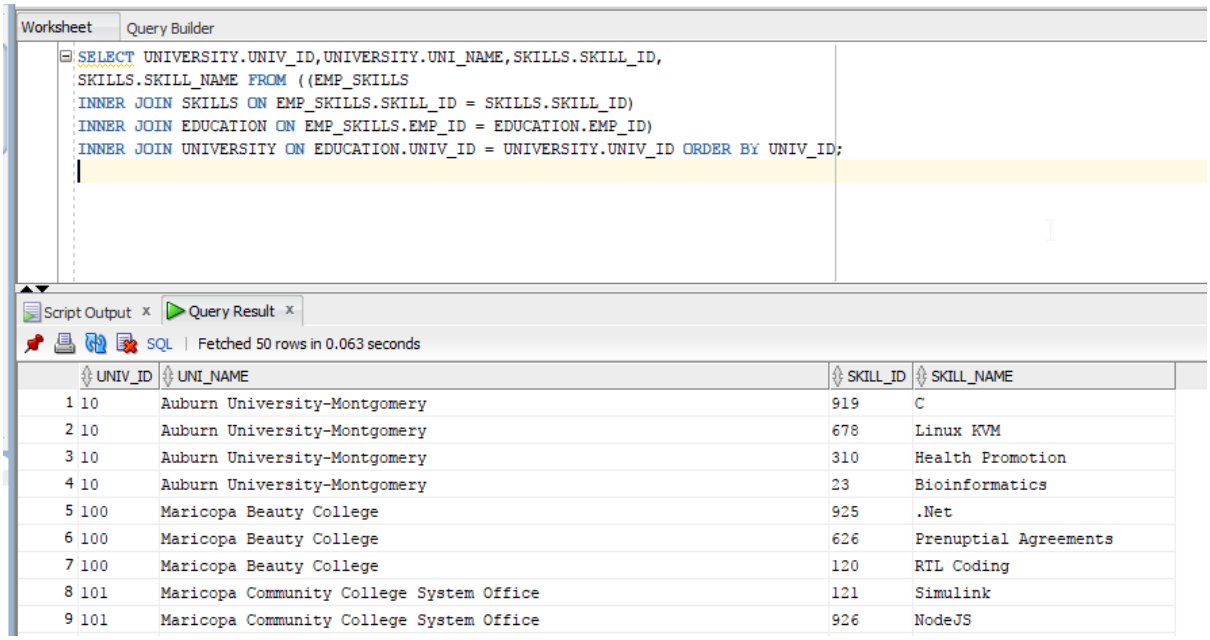
SQL | All Rows Fetched: 34 in 0.056 seconds

	COMP_NAME	CITY	CONTACT_NAME	CONTACT_EMAIL	CONTACT_PHONE
1	ARC Group Worldwide, Inc.	Tifton	Shay Darnell	Shay Darnell@gmail.com	613-134-1809
2	ARC Group Worldwide, Inc.	Decatur	Gabie Brandyn	Gabie Brandyn@gmail.com	326-896-2533
3	Aclaris Therapeutics, Inc.	Tifton	Otis Darnall	Otis Darnall@gmail.com	111-418-9044
4	Aquinox Pharmaceuticals, Inc.	Decatur	Car Jeddy	Car Jeddy@gmail.com	828-732-2507
5	Aradigm Corporation	Tifton	Augustine Ingemar	Augustine Ingemar@gmail.com	152-906-9785
6	Arbutus Biopharma Corporation	Decatur	Lucas August	Lucas August@gmail.com	434-494-0701
7	Athersys, Inc.	Tifton	Zollie Clayborne	Zollie Clayborne@gmail.com	225-136-2615
8	Athersys, Inc.	Decatur	Abeu Larry	Abeu Larry@gmail.com	602-738-3364
9	Atlantic American Corporation	Tifton	Giusto Ab	Giusto Ab@gmail.com	503-768-5812
10	Atlantic American Corporation	Decatur	Si Keen	Si Keen@gmail.com	314-537-9874
11	Atrion Corporation	Tifton	Enrique Rolando	Enrique Rolando@gmail.com	465-296-3604
12	BioLineRx Ltd.	Tifton	Henry Nikolaus	Henry Nikolaus@gmail.com	410-667-8850

4) Which skill imparted in any university has produced an employee.

SQL code:

```
SELECT UNIVERSITY.UNIV_ID,UNIVERSITY.UNI_NAME,SKILLS.SKILL_ID,
SKILLS.SKILL_NAME
FROM ((EMP_SKILLS
INNER JOIN SKILLS ON EMP_SKILLS.SKILL_ID = SKILLS.SKILL_ID)
INNER JOIN EDUCATION ON EMP_SKILLS.EMP_ID = EDUCATION.EMP_ID)
INNER JOIN UNIVERSITY ON EDUCATION.UNIV_ID = UNIVERSITY.UNIV_ID
ORDER BY UNIV_ID;
```



Worksheet | Query Builder

```
SELECT UNIVERSITY.UNIV_ID,UNIVERSITY.UNI_NAME,SKILLS.SKILL_ID,
SKILLS.SKILL_NAME FROM ((EMP_SKILLS
INNER JOIN SKILLS ON EMP_SKILLS.SKILL_ID = SKILLS.SKILL_ID)
INNER JOIN EDUCATION ON EMP_SKILLS.EMP_ID = EDUCATION.EMP_ID)
INNER JOIN UNIVERSITY ON EDUCATION.UNIV_ID = UNIVERSITY.UNIV_ID ORDER BY UNIV_ID;
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.063 seconds

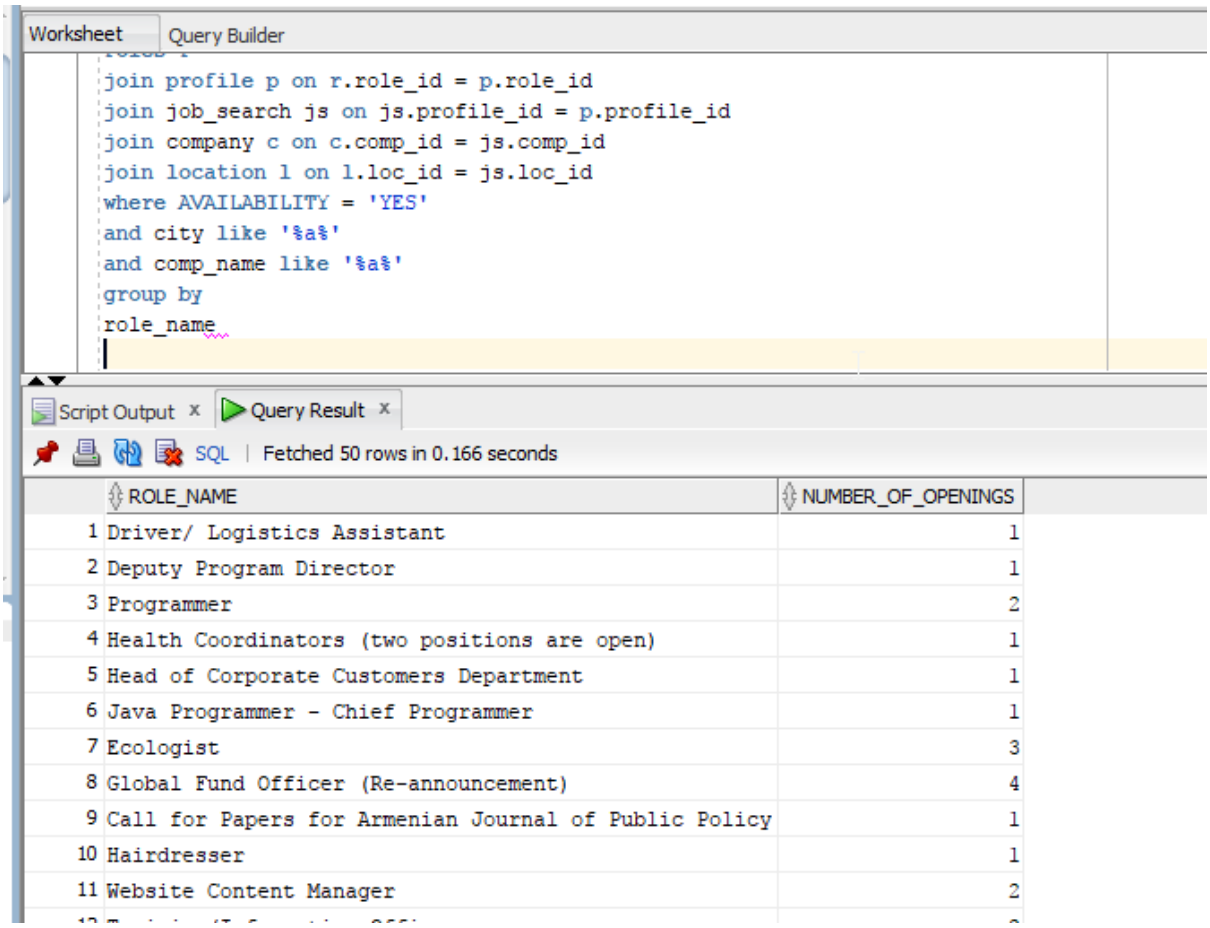
	UNIV_ID	UNI_NAME	SKILL_ID	SKILL_NAME
1	10	Auburn University-Montgomery	919	C
2	10	Auburn University-Montgomery	678	Linux KVM
3	10	Auburn University-Montgomery	310	Health Promotion
4	10	Auburn University-Montgomery	23	Bioinformatics
5	100	Maricopa Beauty College	925	.Net
6	100	Maricopa Beauty College	626	Prenuptial Agreements
7	100	Maricopa Beauty College	120	RTL Coding
8	101	Maricopa Community College System Office	121	Simulink
9	101	Maricopa Community College System Office	926	NodeJS

5) How many open positions for a particular role are available.

SQL code:

```
SELECT
ROLE_NAME,
COUNT(*) NUMBER_OF_OPENINGS
FROM
ROLES R
JOIN PROFILE P ON R.ROLE_ID = P.ROLE_ID
```

```
JOIN JOB_SEARCH JS ON JS.PROFILE_ID = P.PROFILE_ID
JOIN COMPANY C ON C.COMP_ID = JS.COMP_ID
JOIN LOCATION L ON L.LOC_ID = JS.LOC_ID
WHERE AVAILABILITY = 'YES'
AND CITY LIKE '%a%'
AND COMP_NAME LIKE '%a%'
GROUP BY
ROLE_NAME;
```

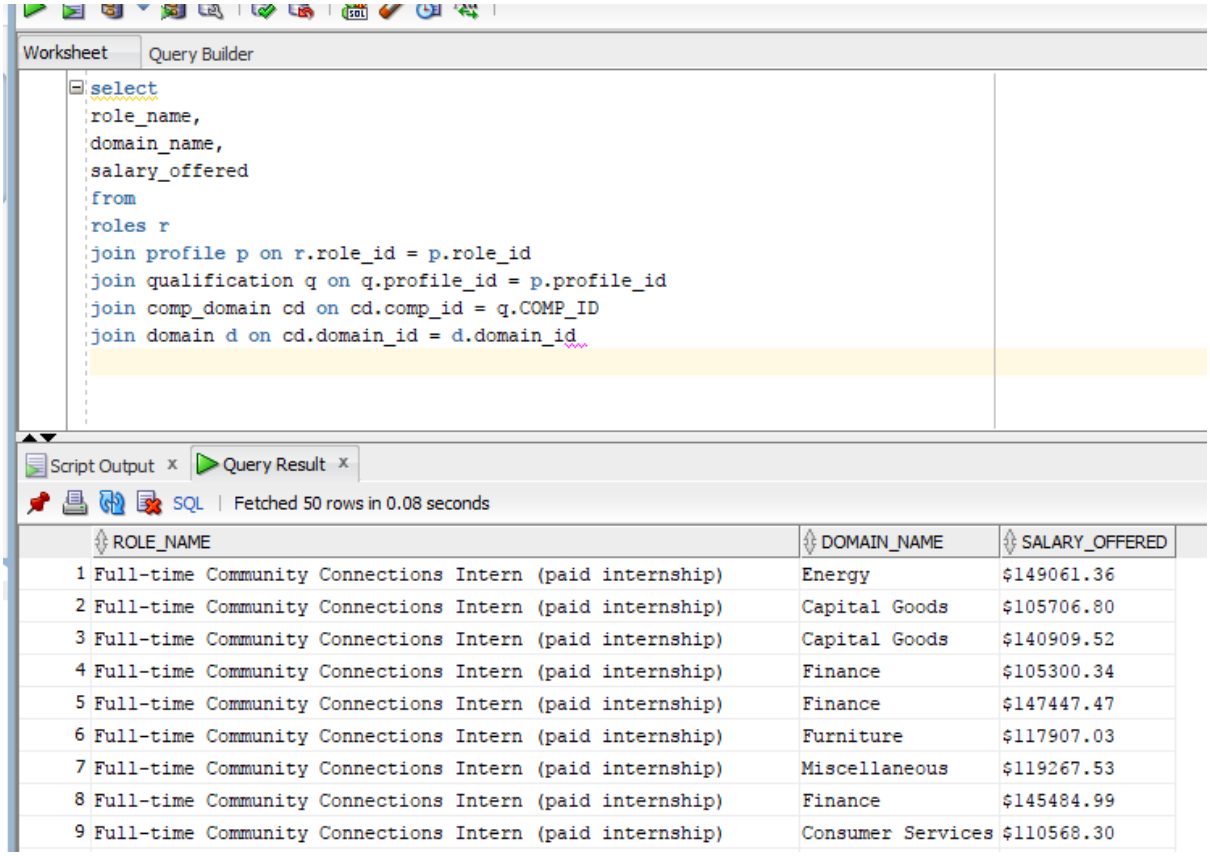


	ROLE_NAME	NUMBER_OF_OPENINGS
1	Driver/ Logistics Assistant	1
2	Deputy Program Director	1
3	Programmer	2
4	Health Coordinators (two positions are open)	1
5	Head of Corporate Customers Department	1
6	Java Programmer - Chief Programmer	1
7	Ecologist	3
8	Global Fund Officer (Re-announcement)	4
9	Call for Papers for Armenian Journal of Public Policy	1
10	Hairdresser	1
11	Website Content Manager	2

6) What is the salary offered for a particular position in a particular domain (irrespective of company /location).

SQL code:

```
SELECT
ROLE_NAME,
DOMAIN_NAME,
SALARY_OFFERED
FROM
ROLES R
JOIN PROFILE P ON R.ROLE_ID = P.ROLE_ID
JOIN QUALIFICATION Q ON Q.PROFILE_ID = P.PROFILE_ID
JOIN COMP_DOMAIN CD ON CD.COMP_ID = Q.COMP_ID
JOIN DOMAIN D ON CD.DOMAIN_ID = D.DOMAIN_ID;
```

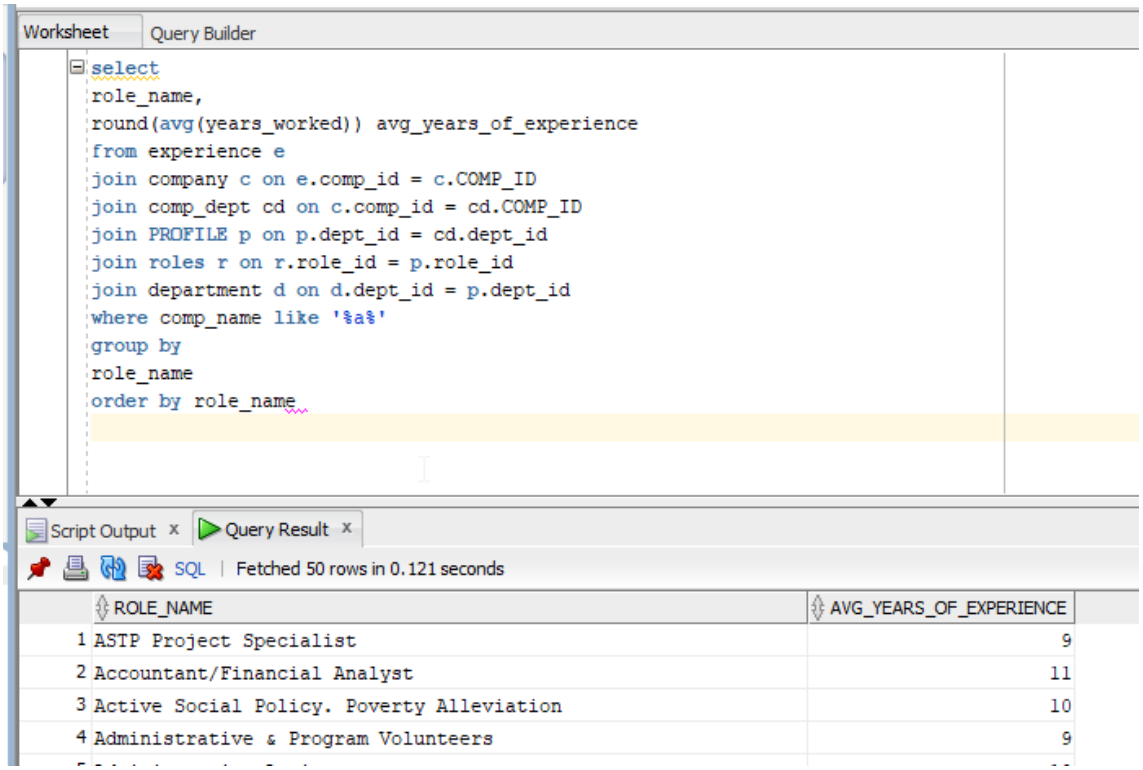


	ROLE_NAME	DOMAIN_NAME	SALARY_OFFERED
1	Full-time Community Connections Intern (paid internship)	Energy	$149061.36
2	Full-time Community Connections Intern (paid internship)	Capital Goods	$105706.80
3	Full-time Community Connections Intern (paid internship)	Capital Goods	$140909.52
4	Full-time Community Connections Intern (paid internship)	Finance	$105300.34
5	Full-time Community Connections Intern (paid internship)	Finance	$147447.47
6	Full-time Community Connections Intern (paid internship)	Furniture	$117907.03
7	Full-time Community Connections Intern (paid internship)	Miscellaneous	$119267.53
8	Full-time Community Connections Intern (paid internship)	Finance	$145484.99
9	Full-time Community Connections Intern (paid internship)	Consumer Services	$110568.30

7) Average experience in years for each role (can filter by company name or department name).

SQL code:

```
SELECT
ROLE_NAME,
ROUND(AVG(YEARS_WORKED)) AVG_YEARS_OF_EXPERIENCE
FROM EXPERIENCE E
JOIN COMPANY C ON E.COMP_ID = C.COMP_ID
JOIN COMP_DEPT CD ON C.COMP_ID = CD.COMP_ID
JOIN PROFILE P ON P.DEPT_ID = CD.DEPT_ID
JOIN ROLES R ON R.ROLE_ID = P.ROLE_ID
JOIN DEPARTMENT D ON D.DEPT_ID = P.DEPT_ID
WHERE COMP_NAME LIKE '%a%'
GROUP BY
ROLE_NAME
ORDER BY ROLE_NAME
```

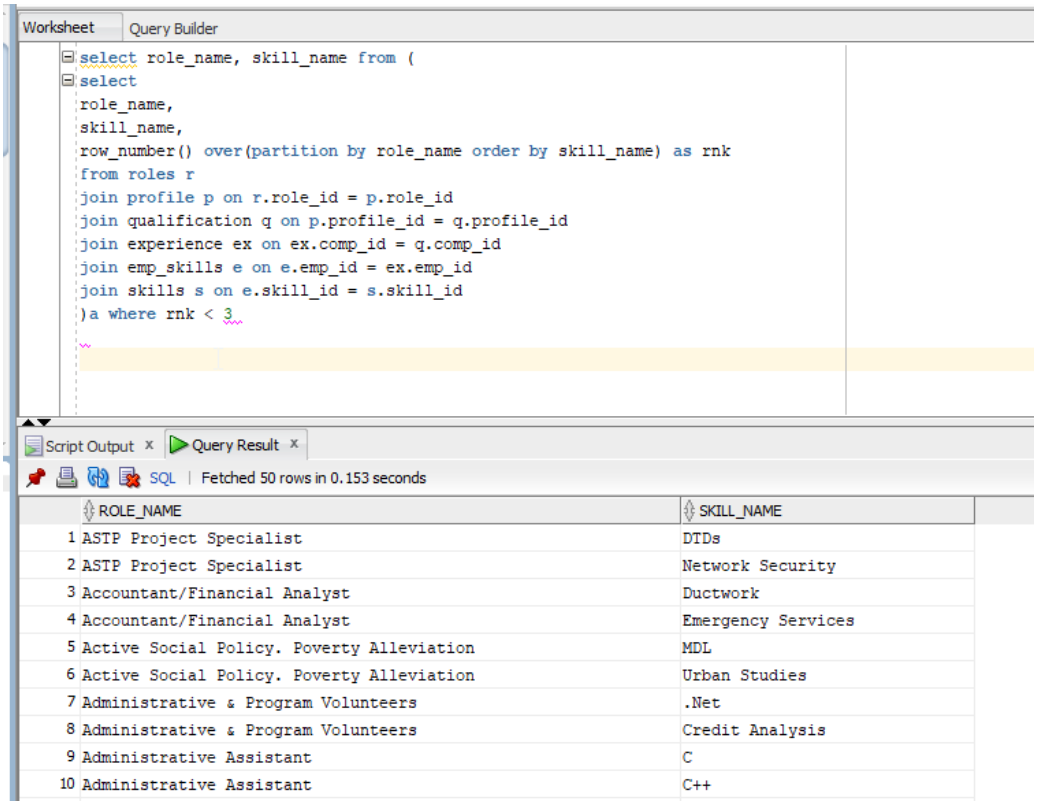


	ROLE_NAME	AVG_YEARS_OF_EXPERIENCE
1	ASTP Project Specialist	9
2	Accountant/Financial Analyst	11
3	Active Social Policy. Poverty Alleviation	10
4	Administrative & Program Volunteers	9

8) Top 2 skills for each role.

SQL code:

```
SELECT ROLE_NAME, SKILL_NAME FROM (
SELECT
ROLE_NAME,
SKILL_NAME,
ROW_NUMBER() OVER(PARTITION BY ROLE_NAME ORDER BY SKILL_NAME) AS RNK
FROM ROLES R
JOIN PROFILE P ON R.ROLE_ID = P.ROLE_ID
JOIN QUALIFICATION Q ON P.PROFILE_ID = Q.PROFILE_ID
JOIN EXPERIENCE EX ON EX.COMP_ID = Q.COMP_ID
JOIN EMP_SKILLS E ON E.EMP_ID = EX.EMP_ID
JOIN SKILLS S ON E.SKILL_ID = S.SKILL_ID
) A WHERE RNK < 3
```

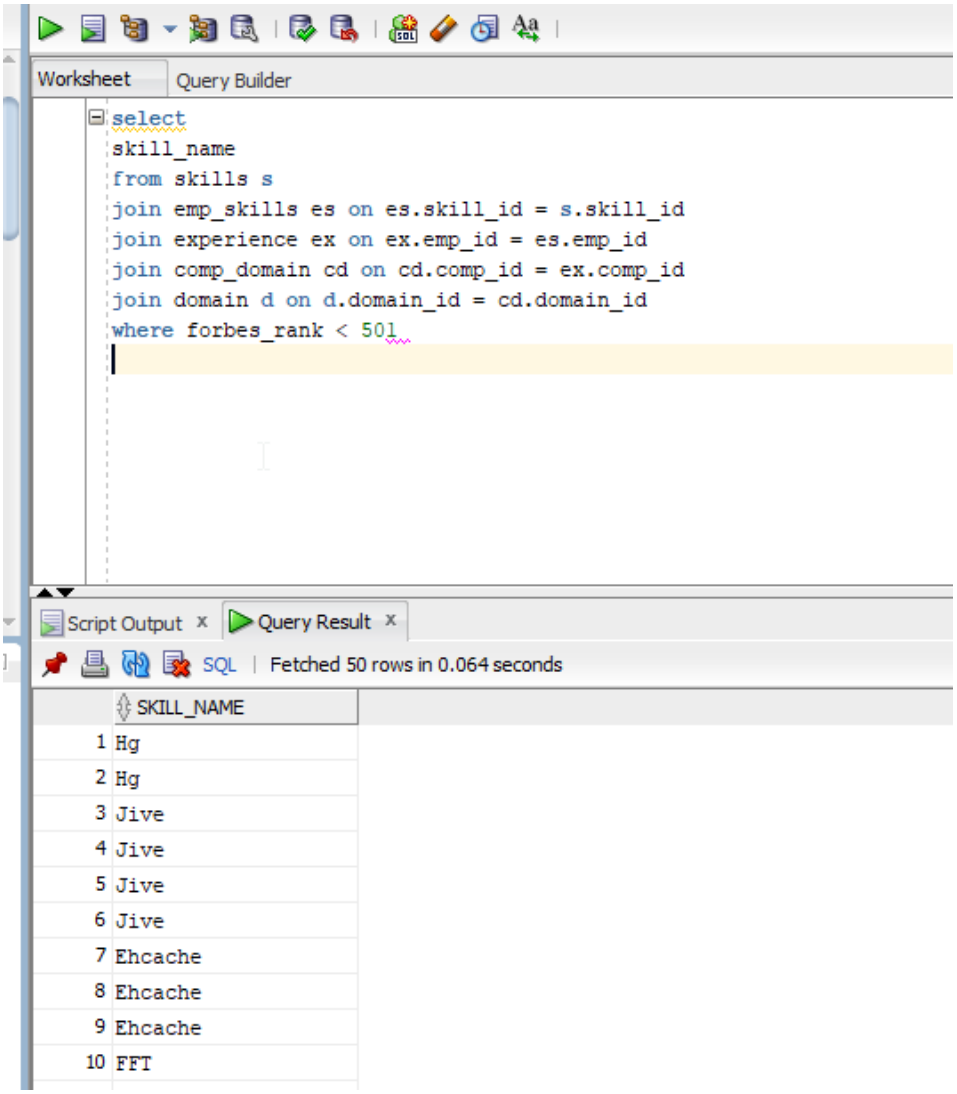


	ROLE_NAME	SKILL_NAME
1	ASTP Project Specialist	DTDs
2	ASTP Project Specialist	Network Security
3	Accountant/Financial Analyst	Ductwork
4	Accountant/Financial Analyst	Emergency Services
5	Active Social Policy. Poverty Alleviation	MDL
6	Active Social Policy. Poverty Alleviation	Urban Studies
7	Administrative & Program Volunteers	.Net
8	Administrative & Program Volunteers	Credit Analysis
9	Administrative Assistant	C
10	Administrative Assistant	C++

9) Which skills get jobs in companies that are ranked less than 501 in Forbes.

SQL code:

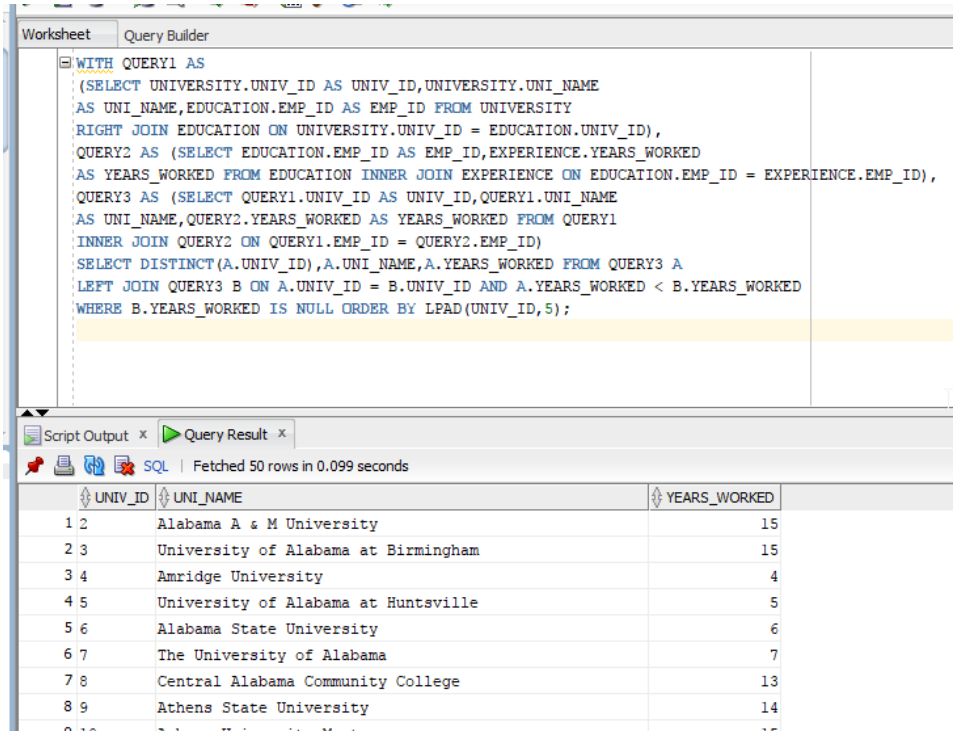
```
SELECT
SKILL_NAME
FROM SKILLS S
JOIN EMP_SKILLS ES ON ES.SKILL_ID = S.SKILL_ID
JOIN EXPERIENCE EX ON EX.EMP_ID = ES.EMP_ID
JOIN COMP_DOMAIN CD ON CD.COMP_ID = EX.COMP_ID
JOIN DOMAIN D ON D.DOMAIN_ID = CD.DOMAIN_ID
WHERE FORBES_RANK < 501
```



10) For maximum how long have the graduates FROM each university working.

SQL code:

```
WITH QUERY1 AS
(SELECT UNIVERSITY.UNIV_ID AS UNIV_ID,UNIVERSITY.UNI_NAME
AS UNI_NAME,EDUCATION.EMP_ID AS EMP_ID FROM UNIVERSITY
RIGHT JOIN EDUCATION ON UNIVERSITY.UNIV_ID = EDUCATION.UNIV_ID),
QUERY2 AS (SELECT EDUCATION.EMP_ID AS EMP_ID,EXPERIENCE.YEARS_WORKED
AS YEARS_WORKED FROM EDUCATION INNER JOIN EXPERIENCE ON EDUCATION.EMP_ID = EXPERIENCE.EMP_ID),
QUERY3 AS (SELECT QUERY1.UNIV_ID AS UNIV_ID,QUERY1.UNI_NAME
AS UNI_NAME,QUERY2.YEARS_WORKED AS YEARS_WORKED FROM QUERY1
INNER JOIN QUERY2 ON QUERY1.EMP_ID = QUERY2.EMP_ID)
SELECT DISTINCT(A.UNIV_ID),A.UNI_NAME,A.YEARS_WORKED FROM QUERY3 A
LEFT JOIN QUERY3 B ON A.UNIV_ID = B.UNIV_ID AND A.YEARS_WORKED < B.YEARS_WORKED
WHERE B.YEARS_WORKED IS NULL ORDER BY LPAD(UNIV_ID,5);
```



Worksheet | Query Builder

Script Output | Query Result

SQL | Fetched 50 rows in 0.099 seconds

	UNIV_ID	UNI_NAME	YEARS_WORKED
1	2	Alabama A & M University	15
2	3	University of Alabama at Birmingham	15
3	4	Amridge University	4
4	5	University of Alabama at Huntsville	5
5	6	Alabama State University	6
6	7	The University of Alabama	7
7	8	Central Alabama Community College	13
8	9	Athens State University	14

PERFORMANCE TUNING IN SQL

There are many effective ways in which we can write SQL queries optimize cost. Below are a few of those techniques we implemented.

1) Use actual names of columns instead of *.

```
SELECT
COMP_NAME,
COMP_ID
FROM COMPANY;
```

2) Use of JOINS instead of sub-queries. Also limit using outer JOINS

```
SELECT COMP_NAME
FROM
((COMP_DOMAIN
INNER JOIN COMPANY ON COMP_DOMAIN.COMP_ID=COMPANY.COMP_ID)
INNER JOIN DOMAIN ON COMP_DOMAIN.DOMAIN_ID=DOMAIN.DOMAIN_ID),
((COMP_LOC
INNER JOIN LOCATION ON LOCATION.LOC_ID=COMP_LOC.LOC_ID)
INNER JOIN COMPANY ON COMPANY.COMP_ID=COMP_LOC.LOC_ID)
WHERE
LOCATION.CITY='Austin' AND DOMAIN.DOMAIN_NAME='Health care' AND
COMP_DOMAIN.FORBES_RANK<='500';
```

3) Perform ORDER BY / GROUP BY on Indexed columns

```
SELECT UNIVERSITY.UNIV_ID,UNIVERSITY.UNI_NAME,SKILLS.SKILL_ID,
SKILLS.SKILL_NAME
FROM ((EMP_SKILLS
INNER JOIN SKILLS ON EMP_SKILLS.SKILL_ID = SKILLS.SKILL_ID)
INNER JOIN EDUCATION ON EMP_SKILLS.EMP_ID = EDUCATION.EMP_ID)
INNER JOIN UNIVERSITY ON EDUCATION.UNIV_ID = UNIVERSITY.UNIV_ID ORDER
BY UNIV_ID;
```

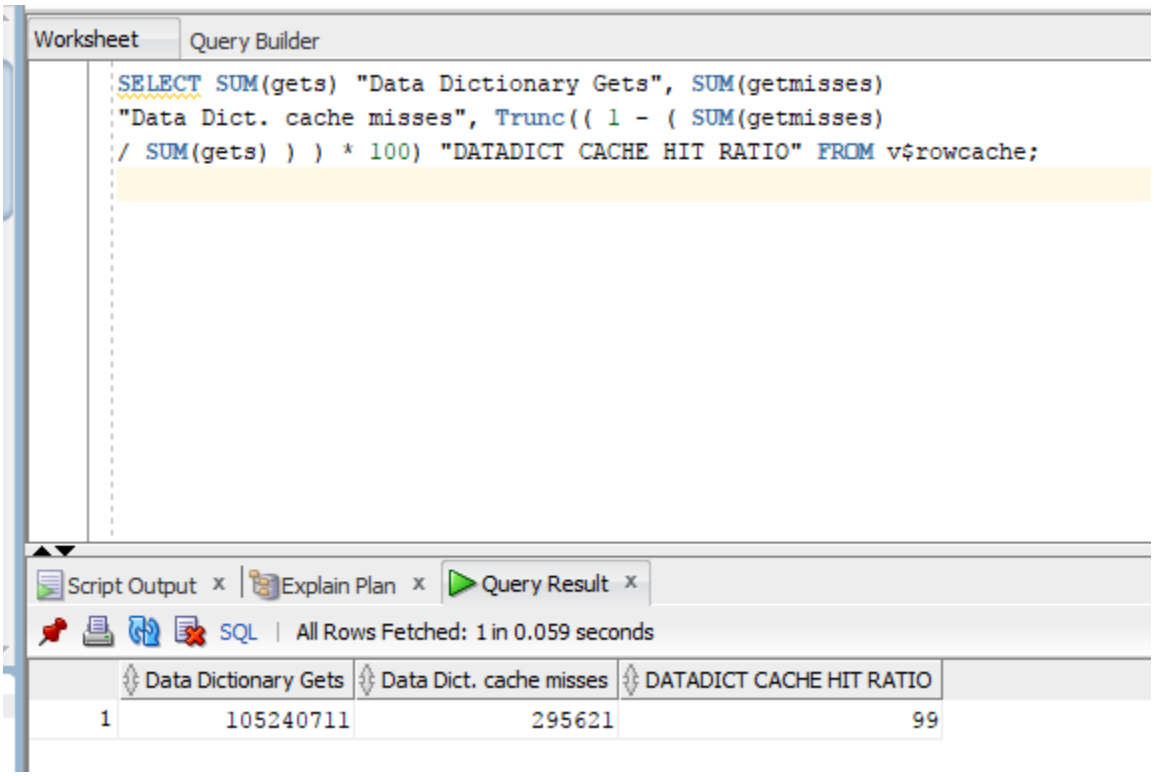
DATABASE SCRIPTING:

Database scripting involves writing SQL scripts to monitor performance of the DB. This is mainly used by DB administrators. Below are a few DB scripts we used to monitor the performance.

*These are generic scripts which are used universally.

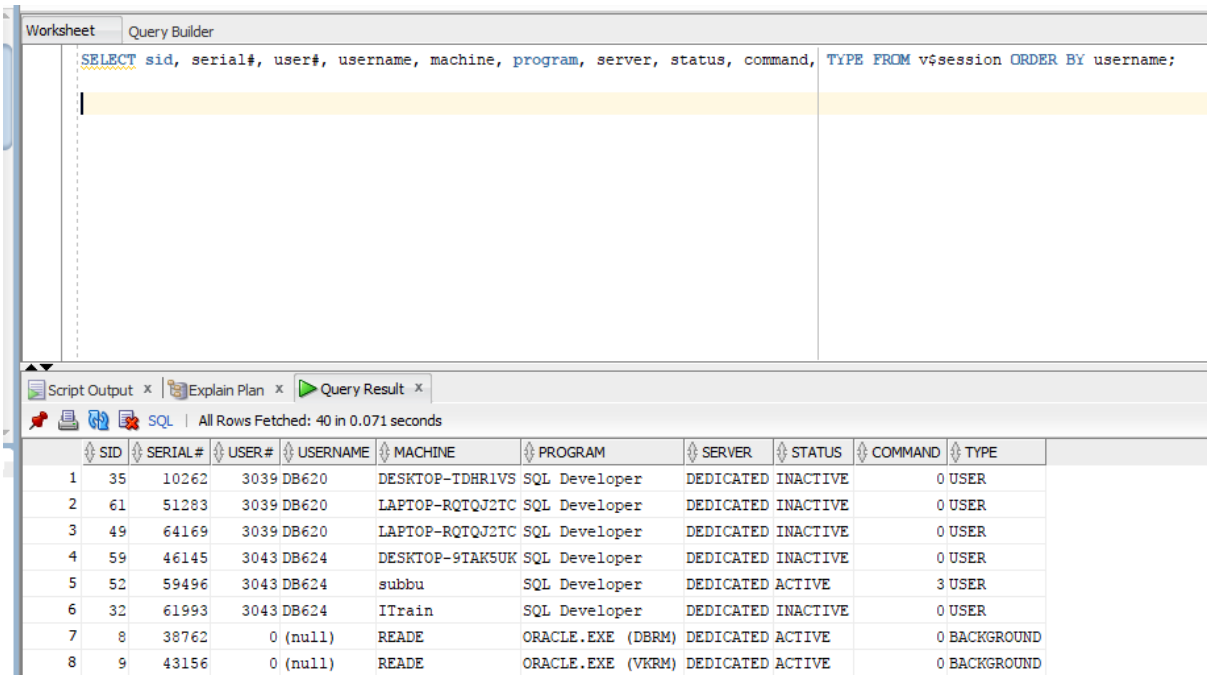
1) Data Dictionary cache hit ratio and data dictionary cache misses can be checked using the below script.

```
SELECT SUM(gets) "Data Dictionary Gets", SUM(getmisses)
"Data Dict. cache misses", Trunc(( 1 - ( SUM(getmisses)
/ SUM(gets) ) ) * 100) "DATADICT CACHE HIT RATIO" FROM v$rowcache;
```



2) We can check the status of the users in a DB, and a host of other details about those users using the below script.

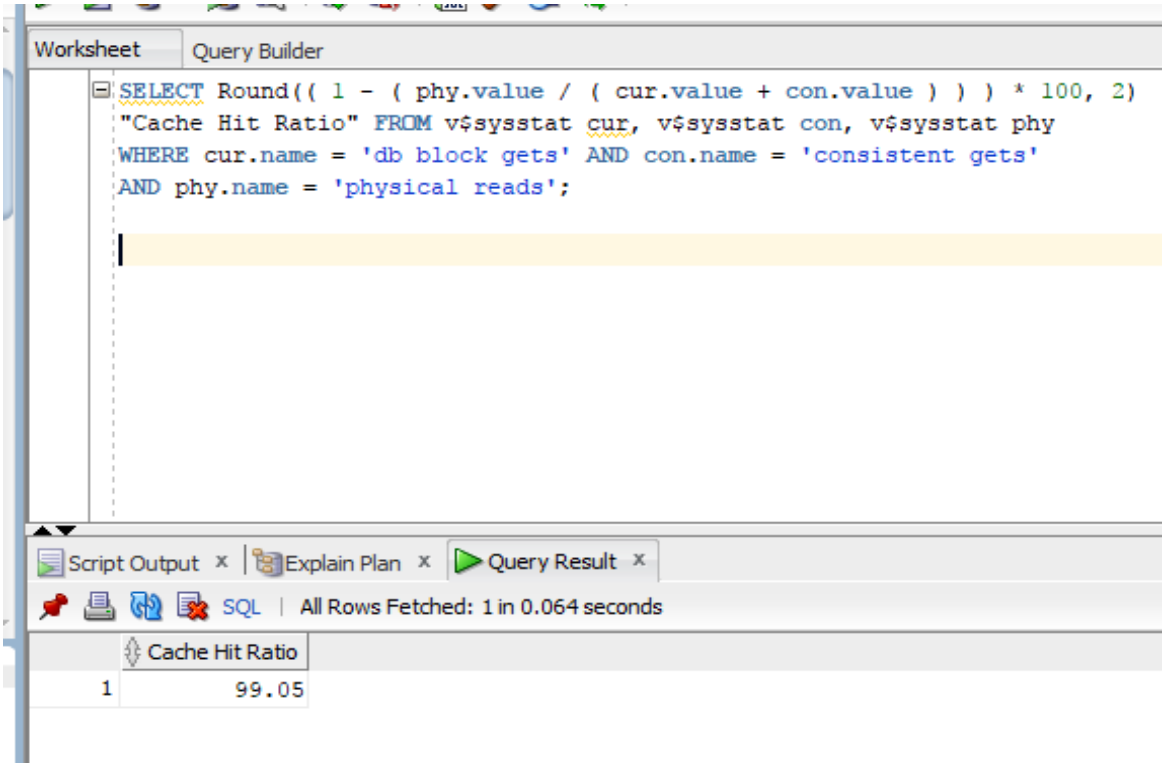
```
SELECT sid, serial#, user#, username, machine, program, server, status, command, TYPE
FROM v$session ORDER BY username;
```



	SID	SERIAL#	USER#	USERNAME	MACHINE	PROGRAM	SERVER	STATUS	COMMAND	TYPE
1	35	10262	3039	DB620	DESKTOP-TDHR1VS	SQL Developer	DEDICATED	INACTIVE	0	USER
2	61	51283	3039	DB620	LAPTOP-RQTQJ2TC	SQL Developer	DEDICATED	INACTIVE	0	USER
3	49	64169	3039	DB620	LAPTOP-RQTQJ2TC	SQL Developer	DEDICATED	INACTIVE	0	USER
4	59	46145	3043	DB624	DESKTOP-9TAK5UK	SQL Developer	DEDICATED	INACTIVE	0	USER
5	52	59496	3043	DB624	subbu	SQL Developer	DEDICATED	ACTIVE	3	USER
6	32	61993	3043	DB624	ITrain	SQL Developer	DEDICATED	INACTIVE	0	USER
7	8	38762	0	(null)	READE	ORACLE.EXE (DBRM)	DEDICATED	ACTIVE	0	BACKGROUND
8	9	43156	0	(null)	READE	ORACLE.EXE (VKRM)	DEDICATED	ACTIVE	0	BACKGROUND

3) DB buffer cache hit ratio can be checked using the below script.

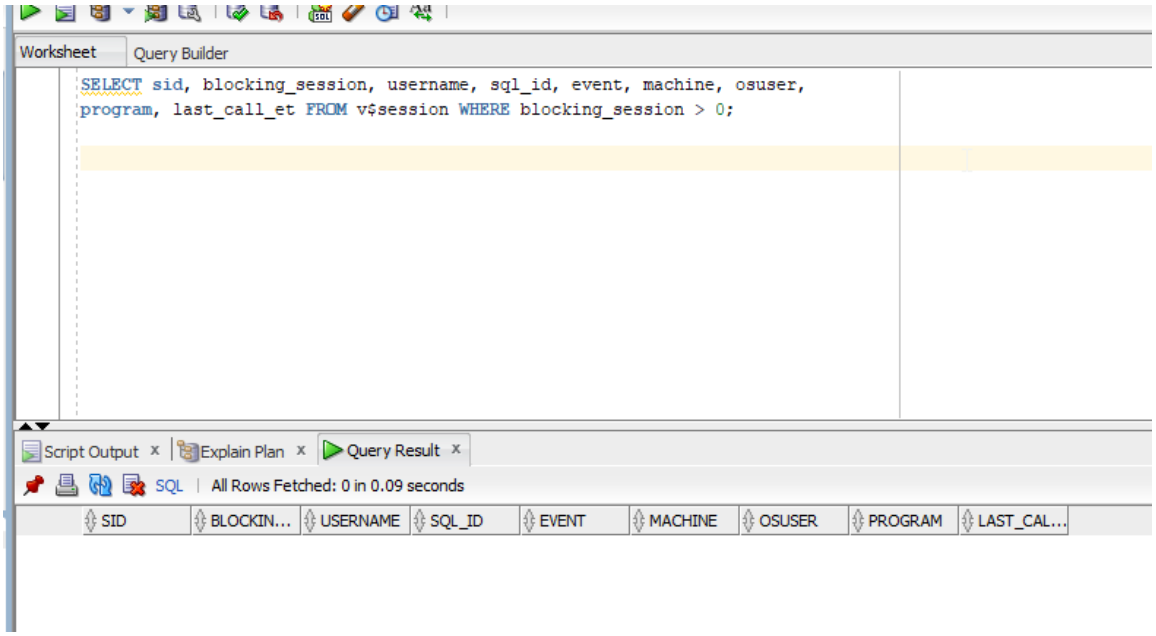
```
SELECT Round(( 1 - ( phy.value / ( cur.value + con.value ) ) ) * 100, 2)
"Cache Hit Ratio" FROM v$sysstat cur, v$sysstat con, v$sysstat phy
WHERE cur.name = 'db block gets' AND con.name = 'consistent gets'
AND phy.name = 'physical reads';
```



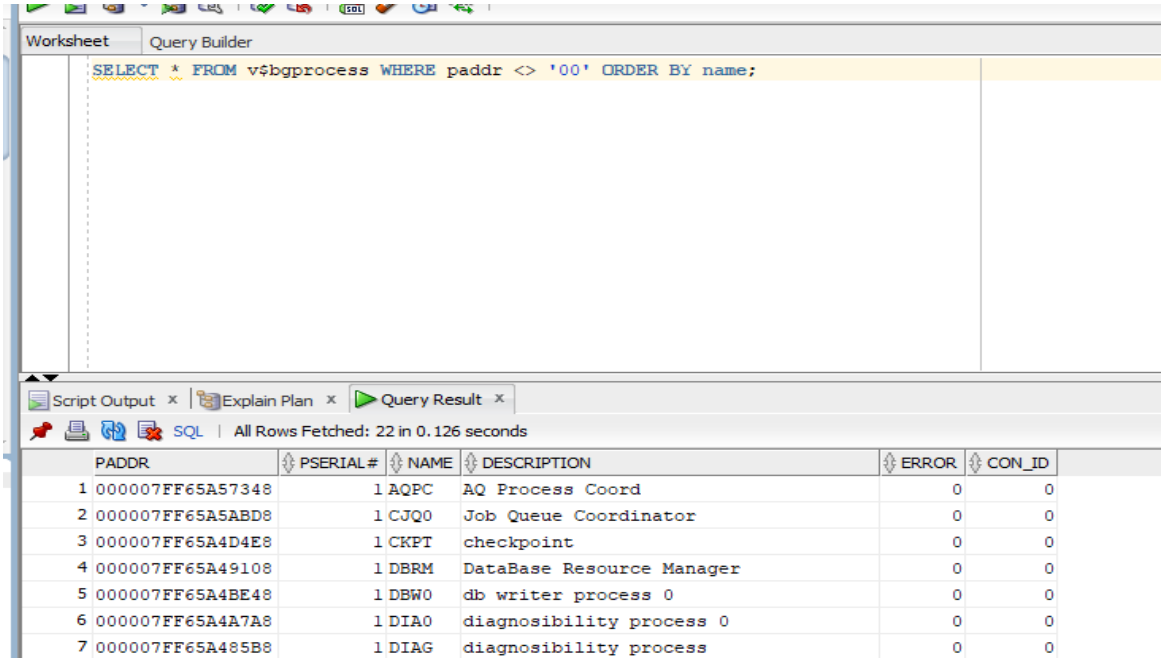
	Cache Hit Ratio
1	99.05

4) All blocking sessions and the user information of those blocking sessions can be checked using the below script.

SELECT sid, blocking_session, username, sql_id, event, machine, osuser, program, last_call_et FROM v$session WHERE blocking_session > 0;



5) A host of information about currently running background processes and their description can be check using the below script.
SELECT * FROM v$bgprocess WHERE paddr <> '00' ORDER BY name;



	PADDR	PSERIAL#	NAME	DESCRIPTION	ERROR	CON_ID
1	000007FF65A57348	1	AQPC	AQ Process Coord	0	0
2	000007FF65A5ABD8	1	CJQ0	Job Queue Coordinator	0	0
3	000007FF65A4D4E8	1	CKPT	checkpoint	0	0
4	000007FF65A49108	1	DBRM	DataBase Resource Manager	0	0
5	000007FF65A4BE48	1	DBW0	db writer process 0	0	0
6	000007FF65A4A7A8	1	DIA0	diagnosibility process 0	0	0
7	000007FF65A485B8	1	DIAG	diagnosibility process	0	0

VISUALIZATION USING SPOTFIRE:

"Data visualization or data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data, meaning "information that has been abstracted in some schematic form, including attributes or variables for the units of information".

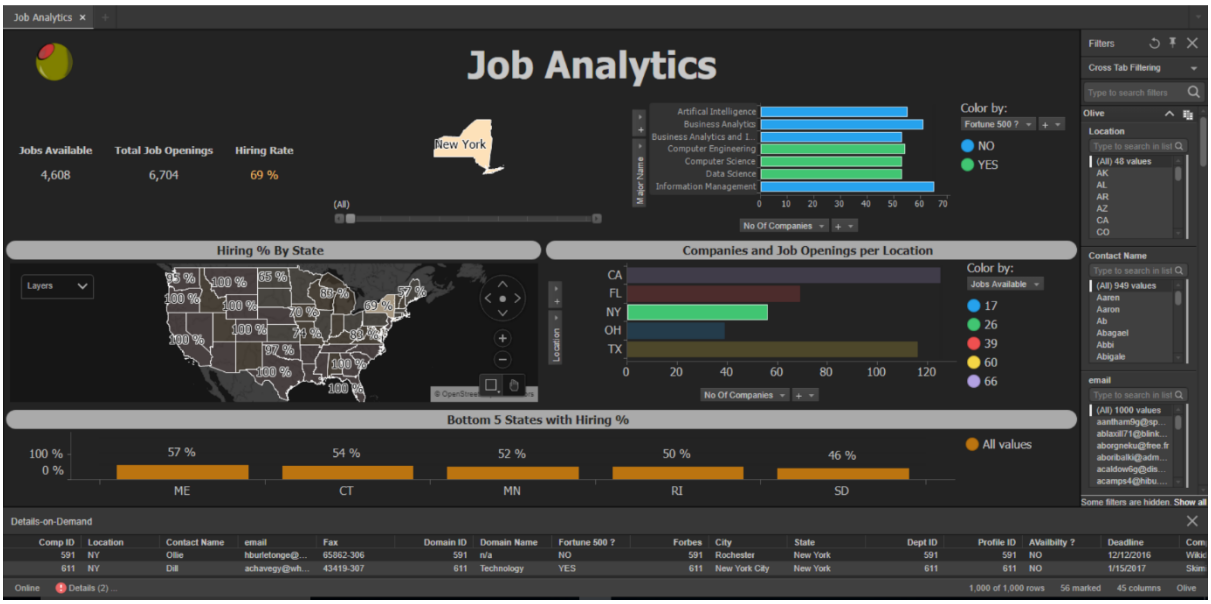
A primary goal of data visualization is to communicate information clearly and efficiently via statistical graphics, plots and information graphics. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message. Effective visualization helps users analyze and reason about data and evidence. It makes complex data more accessible, understandable and usable. Users may have particular analytical tasks, such as making comparisons or understanding causality, and the design principle of the graphic (i.e., showing comparisons or showing causality) follows the task.

The reason behind selecting Spotfire for visualization is because of the following features.

- **Data Discovery**
  Communicate findings and concepts through intuitive visualizations.
- **Data Wrangling**
  Improve and prepare data as you work on it.
- **Predictive analytics**
  Anticipate what's next.
- **Big Data Analytics**
  Turbo-charge your analytics and your business.
- **Location Analytics**
  Associate data with location for immediate understanding
- **Enterprise Scale Analytics**
  Rely on unparalleled performance, governance, flexibility.

For the project, we have chosen Location Analytics and Data discovery to display some intuitive visualizations.

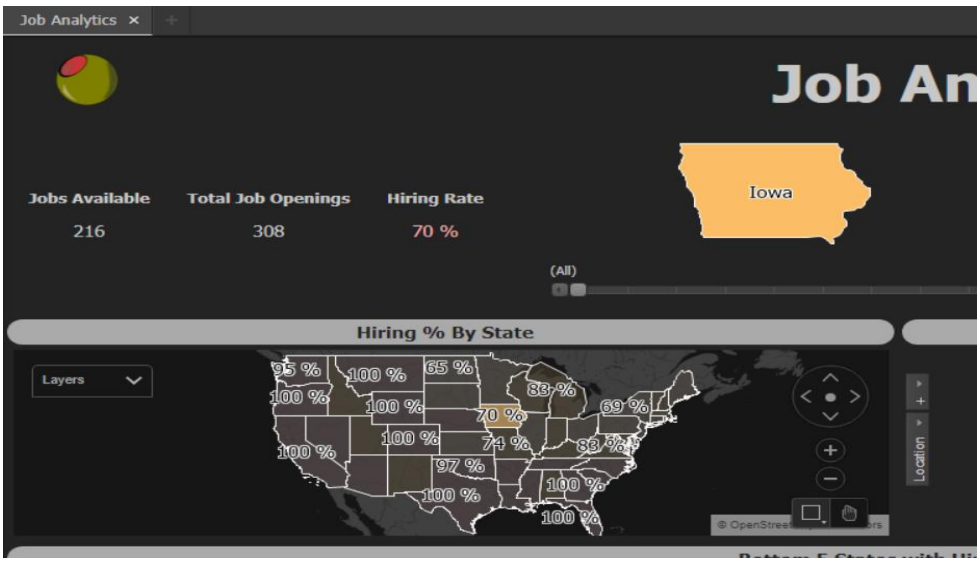
Here's how the GUI looks like when clicked on the front end to view trends and analytics.



The visualization used is dynamic and the data filters based on the filter panel given on the right side of the dashboard.

1) Data Metrics

The metrics on the left side show the availability of Jobs with respect to all the job openings. The data can be filtered state wise, city wise, company wise or any other filter selected in the filter pane on the right side. Eg. I select IOWA on the geo-analytics chart that shows 70% hiring rate.



2) Data Metric

The Metric Displayed in the Top left shows the number of Jobs with Availability filter as “Yes” and the total jobs that were available giving you are the rate of hiring.

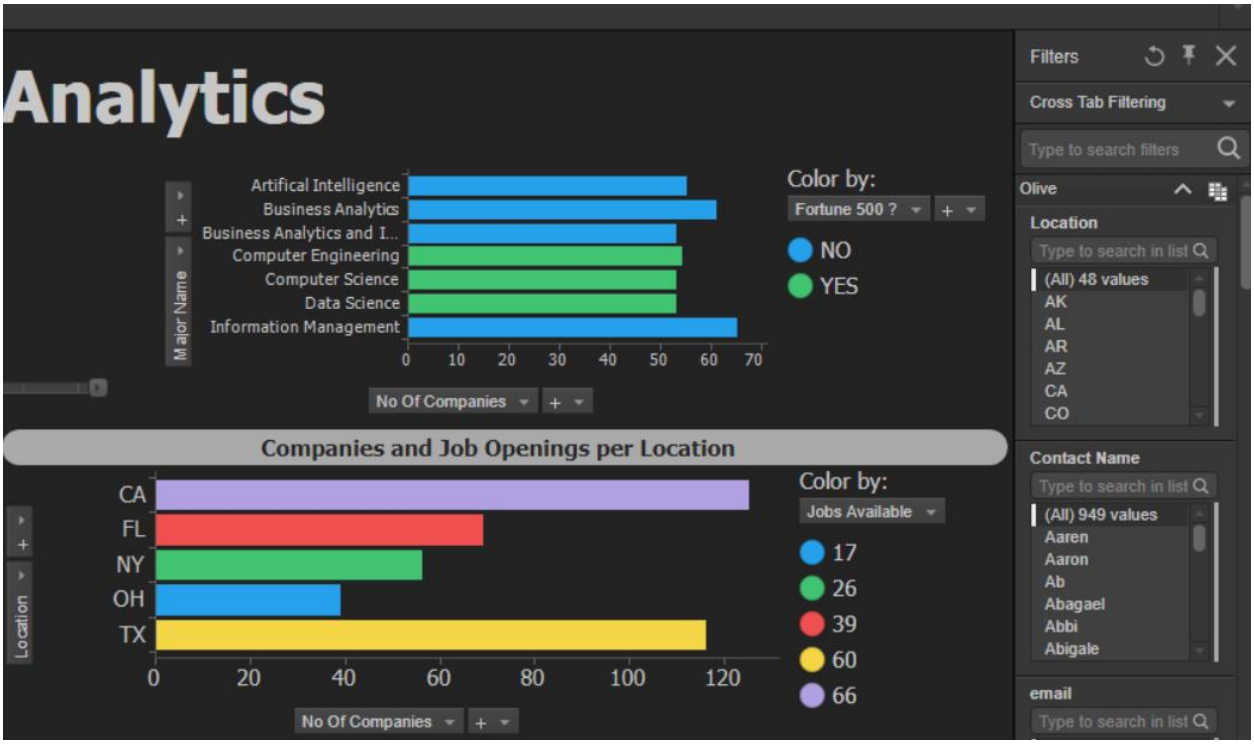
3) Companies and Skills

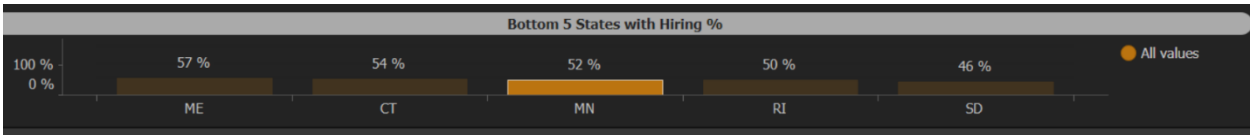
Displaying Data visualization w.r.t Skills.

The topmost chart Displays which of the skills can possibly land you a job in fortune 500 companies based on the sample data we have.

The chart below displays no. of companies per location and displays how many jobs are available in each location.

Negative Analysis gives a perspective of where the results are not favorable and needs to be worked upon. In this product, we are showing the bottom five states with hiring % which displays fewer chances of getting hired.





4) Raw Data

Every selection on the filter panel filters out the data on the visualization and helps you understand the trend for that particular set of data, need may arise to download the data set for further manual analysis, hence a table “Details on Demand” with all the fields of the database is a feature of Spotfire to display data in a tabular format.

WEB APPLICATION:

TECHNICAL STACK

Language: Java

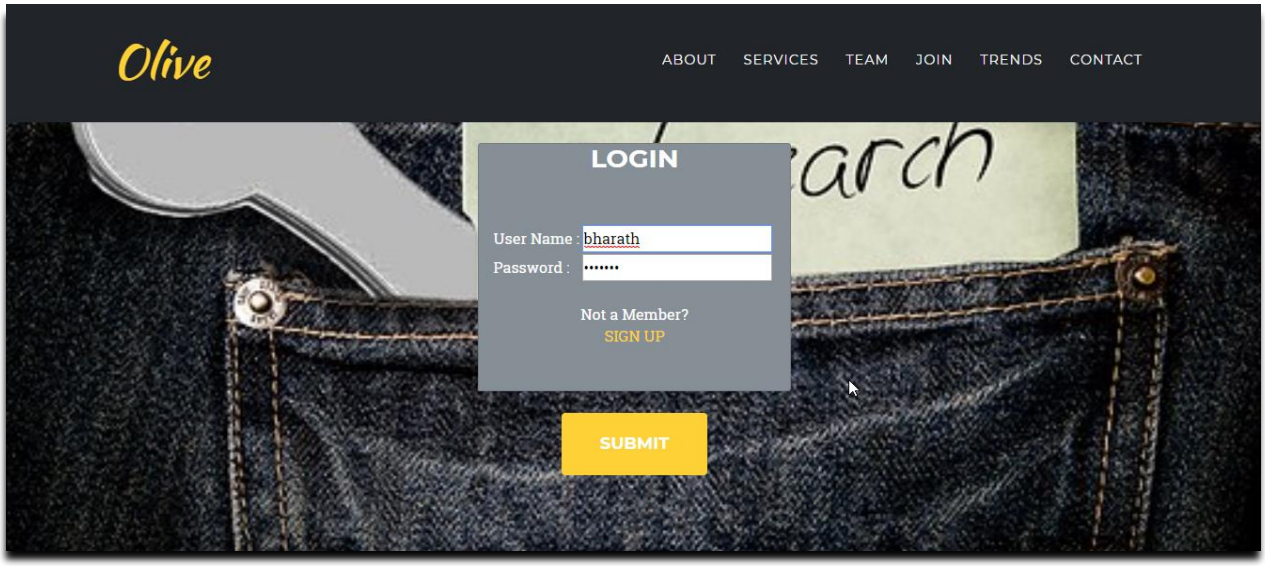
Framework: Springs, JDBC

Front End: Bootstrap, jQuery, JSP.

Database: Oracle 12c

LOGIN

User will enter his username and password for login. If the credential match with the user table data, then the user is allowed to login. If the user is not a registered user, he can Sign UP to the application using the Sing up link provided.



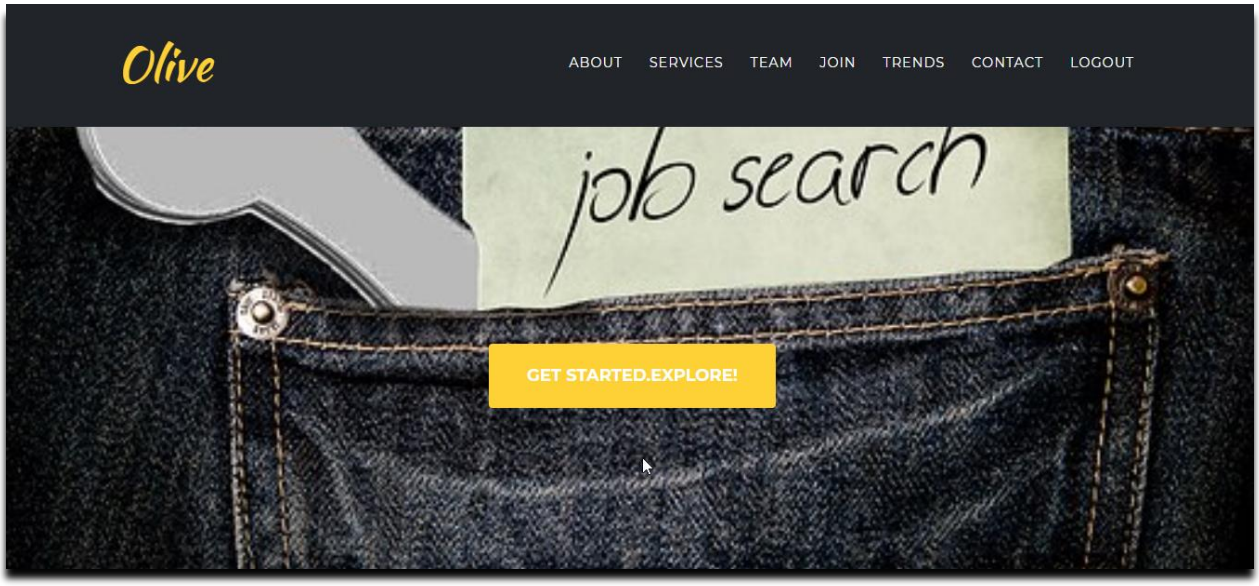
LANDING PAGE

After successful login the user is redirected to the landing screen of the system. The top right consists of the menu and the same is retained throughout the application to make the design symmetric.

It consists of the Get Started button which will take you to the core functionality of the system. This is basically provided to get to know the different functionality and features of the system.

Landing screen has the following sections:

1. About
2. Services
3. Team
4. Join
5. Trends
6. Contact
7. Log out



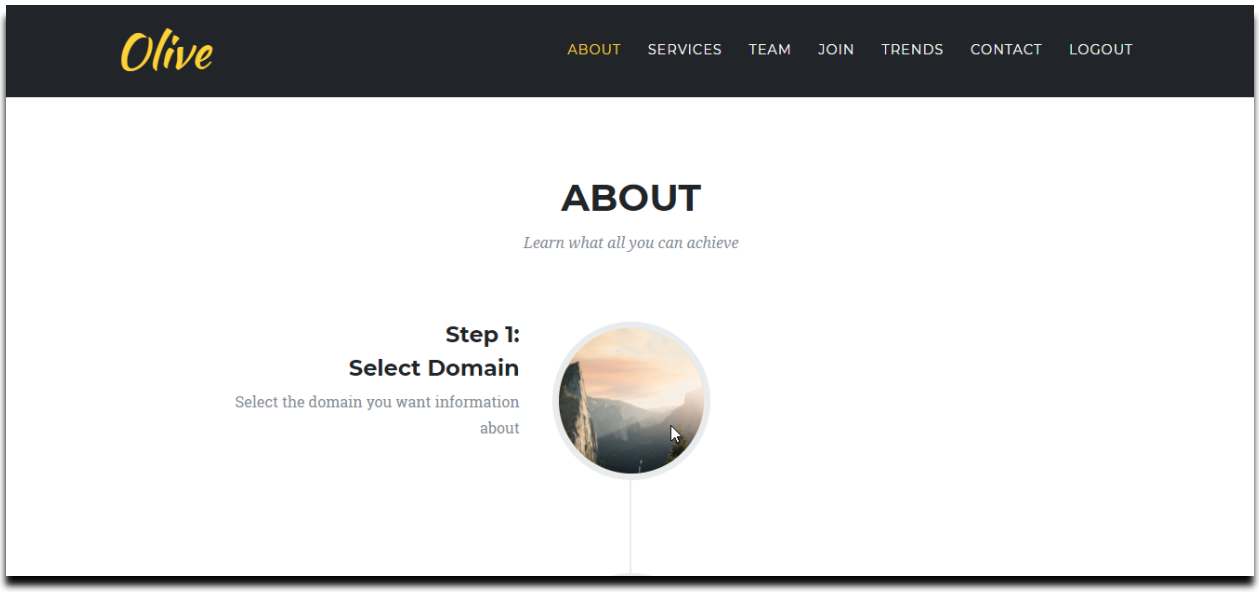
ABOUT

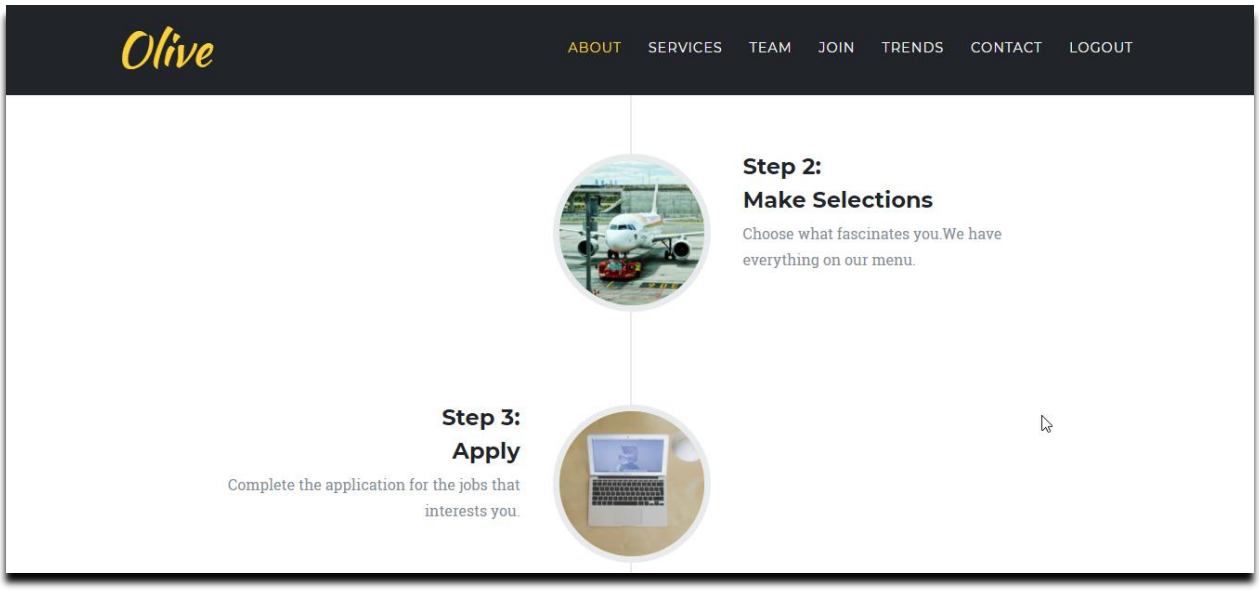
This section guided the end user regarding the flow of the application so that the end user knows the steps to be followed to effectively use the system.

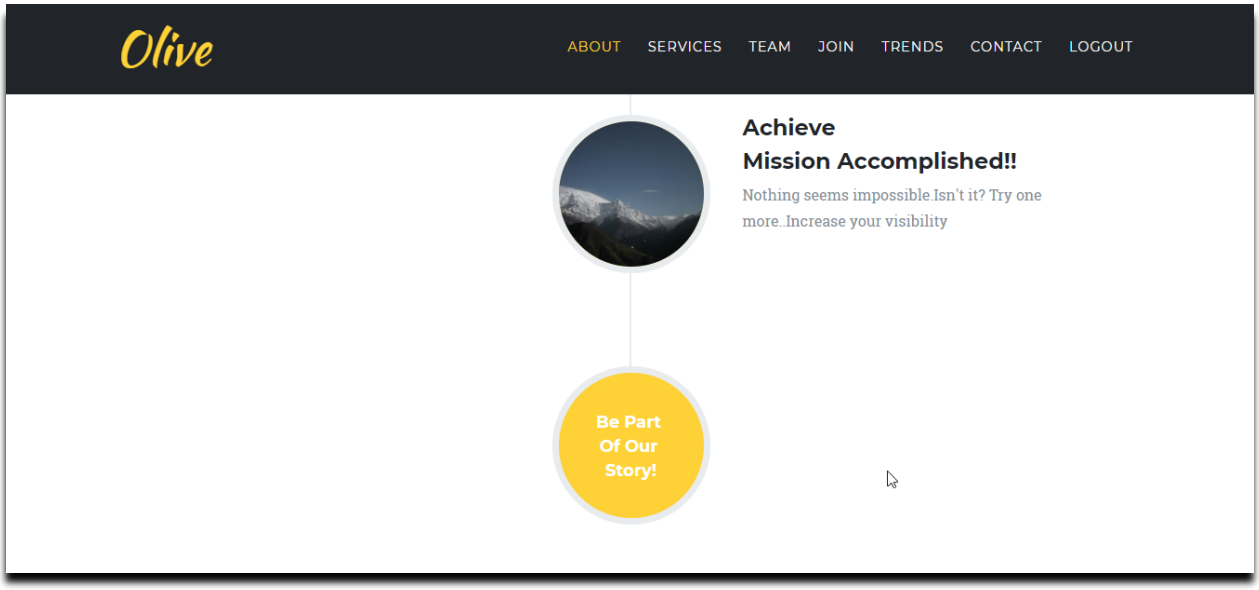
First the user registers and selects the Domain he is interested to know about.

In the second step he makes the different search criteria selection so that he effectively finds what he is in need off.

He finds the suitable jobs and can act on the same either by applying od savin the same for further reference.





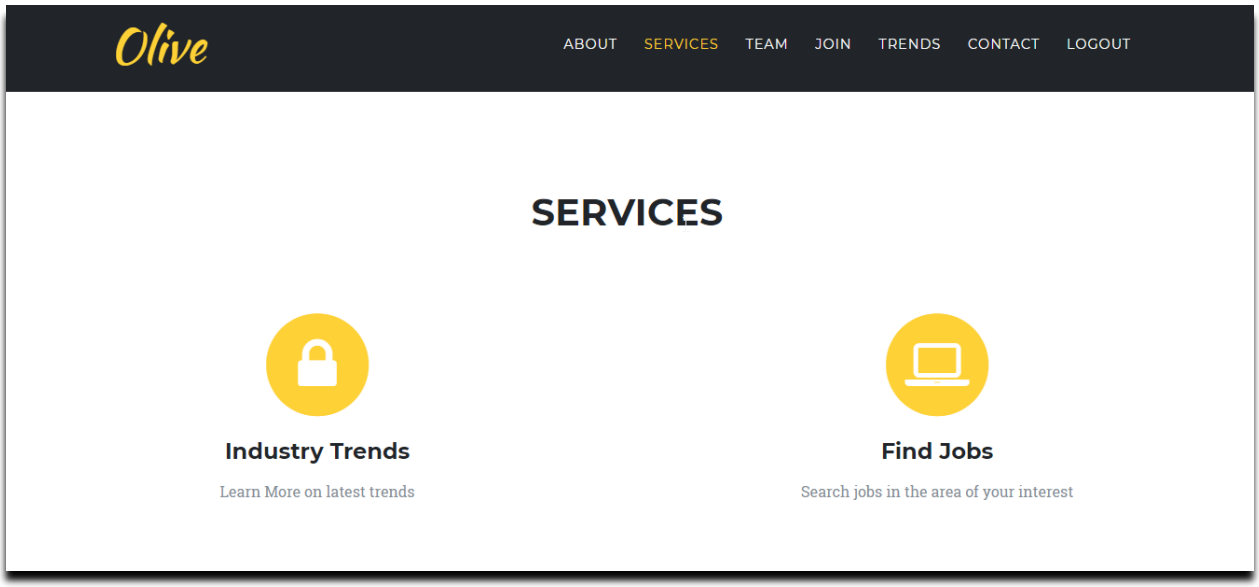


SERVICES

This screen provided the user with the features the system provides so that he can feel comfortable with the working.

The system helps the end user to know the industry trends so that he can effectively focus his learning on those areas.

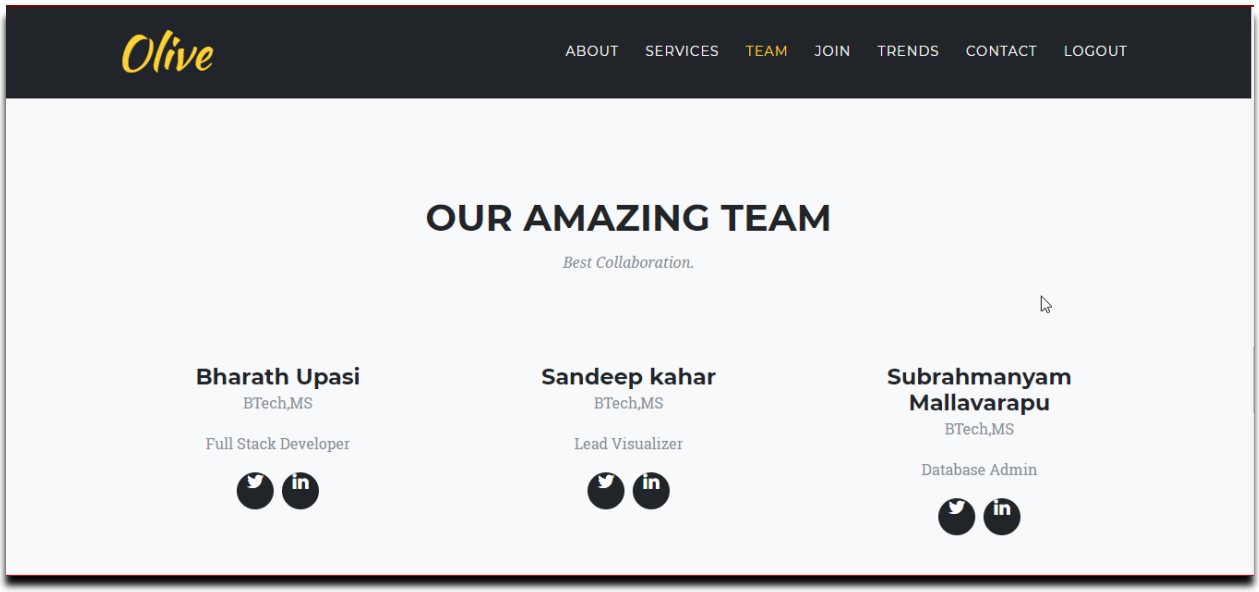
It also provides the user the list of available jobs in the same area so that the user can know the same and apply.

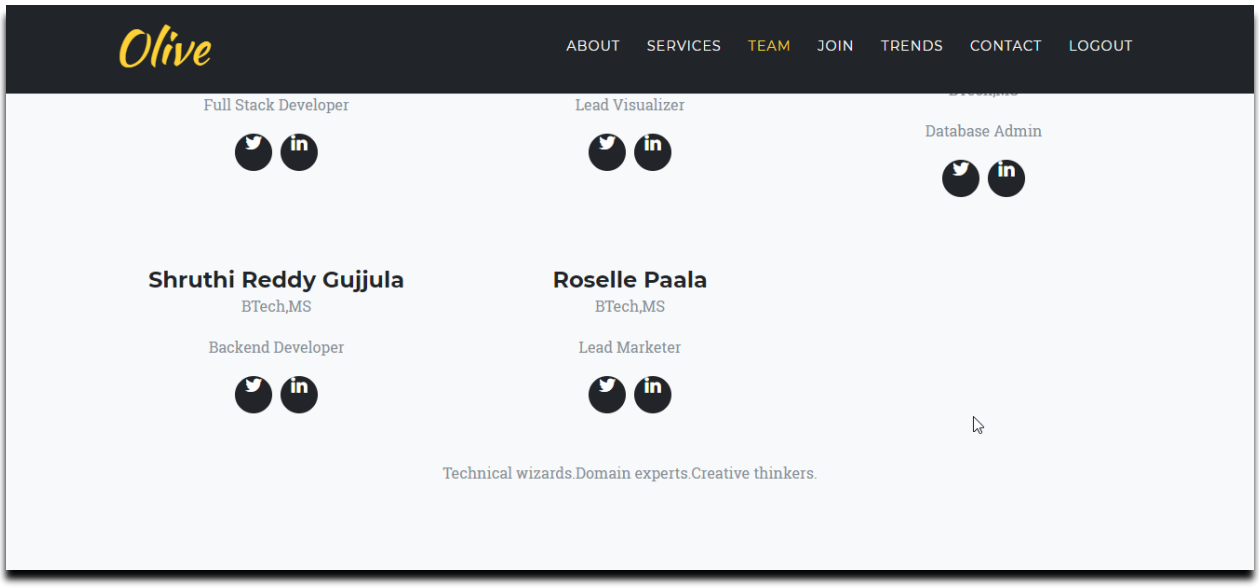


TEAM

This view basically provides the data regarding the team members of the company so that they can be familiar with people they are dealing with.

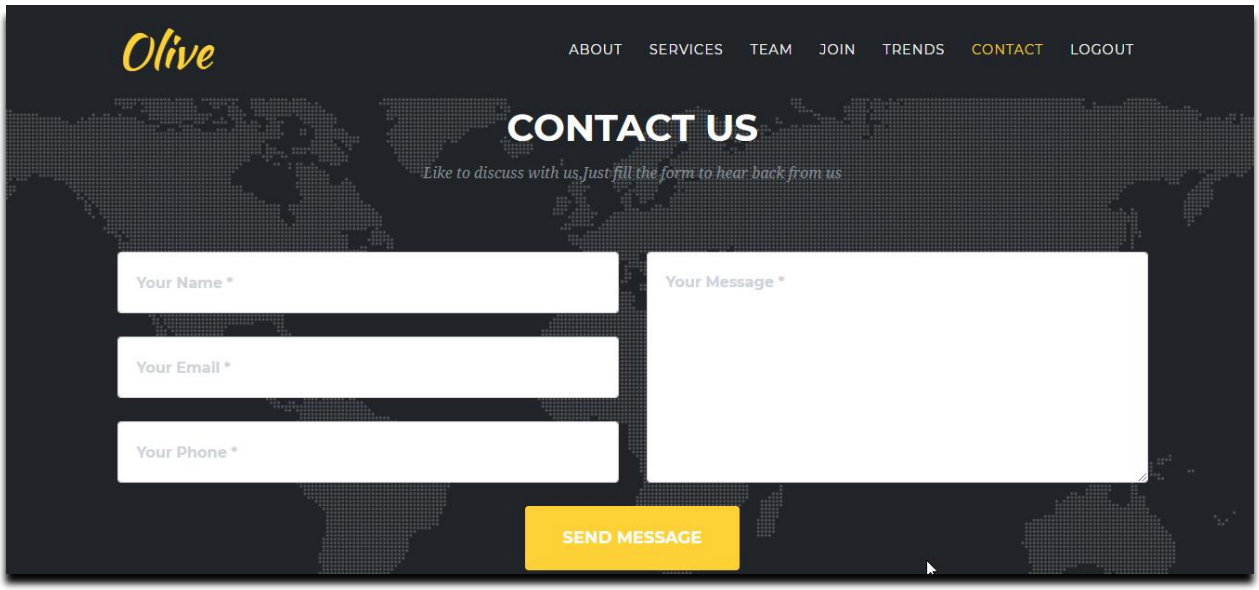
This also has links to their specific Twitter and LinkedIn accounts also to know them better.





CONTACT US:

This screen is provided for the end user so that he can use the same to contact the right persons of the company so that they can resolve their queries effectively.

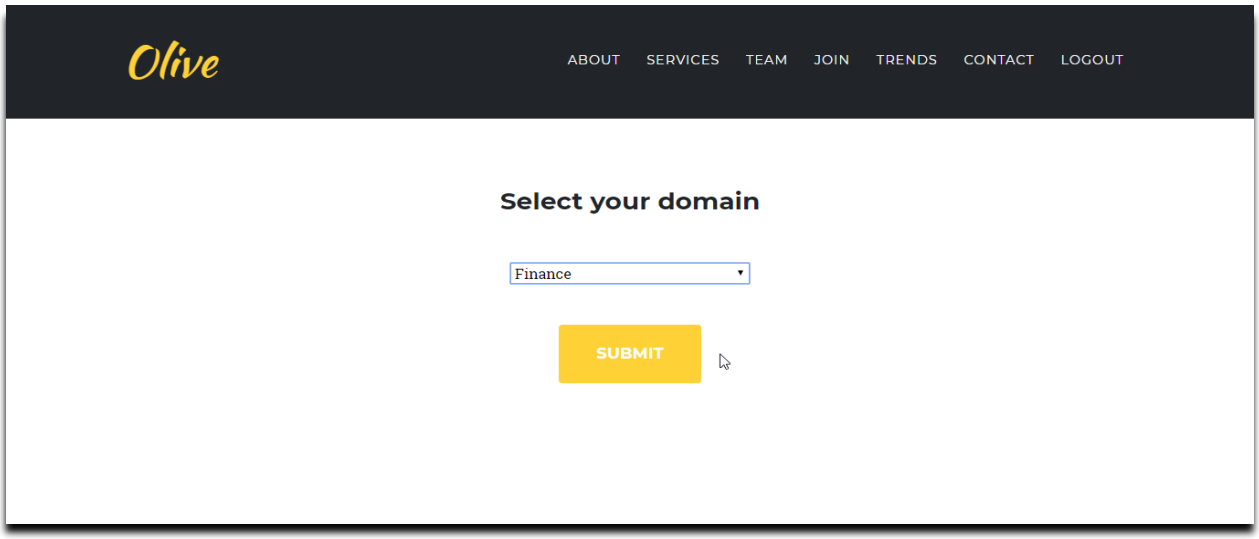


GET STARTED:

After the user click on the “Get Started.Explore!” button then the user will be directed to the screen shown below.

User needs to select the suitable domain that is his area of interest and the same will be maintained in the further part of the application.

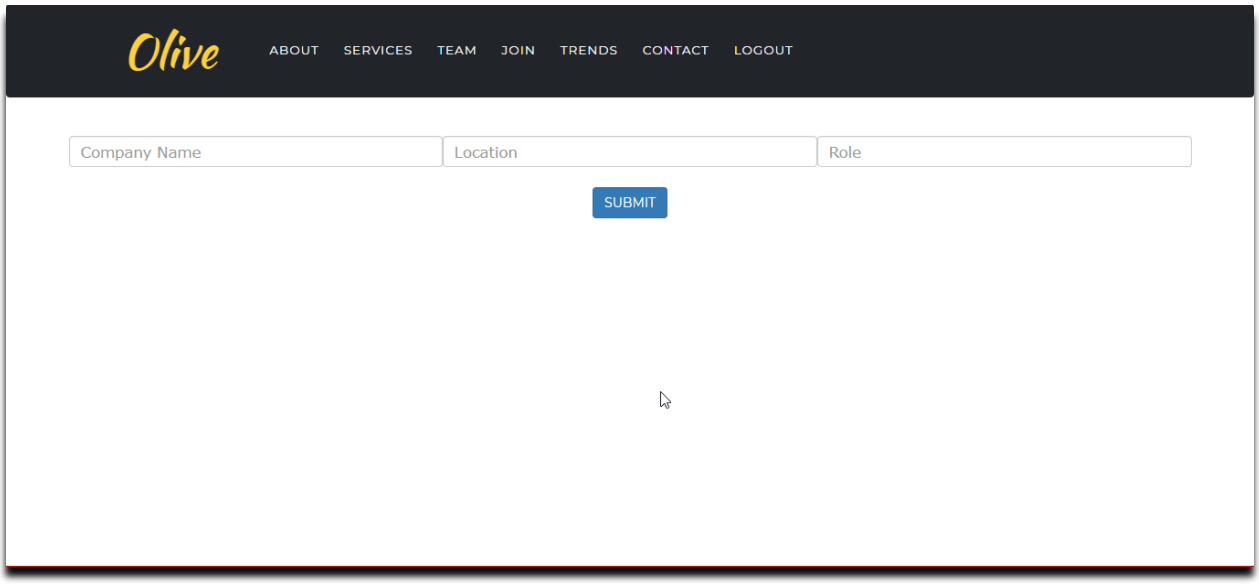
So only data relevant to that domain will be shown to the end user.



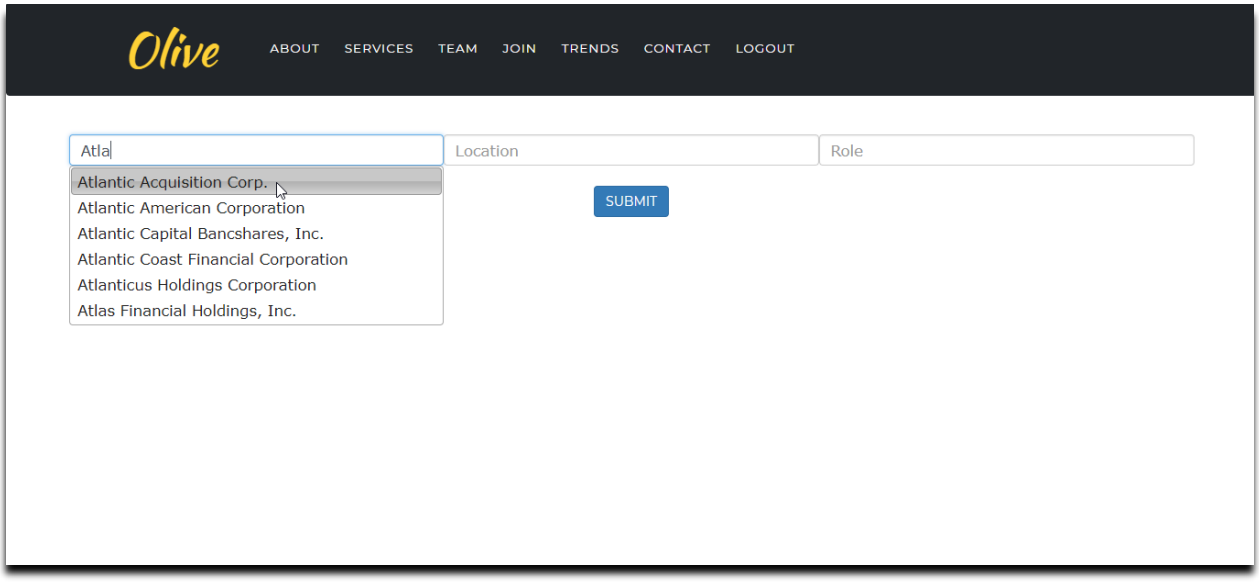
JOB SEARCH:

After suitable selection of the domain then the user will be redirected to the Job Search screen as shown below.

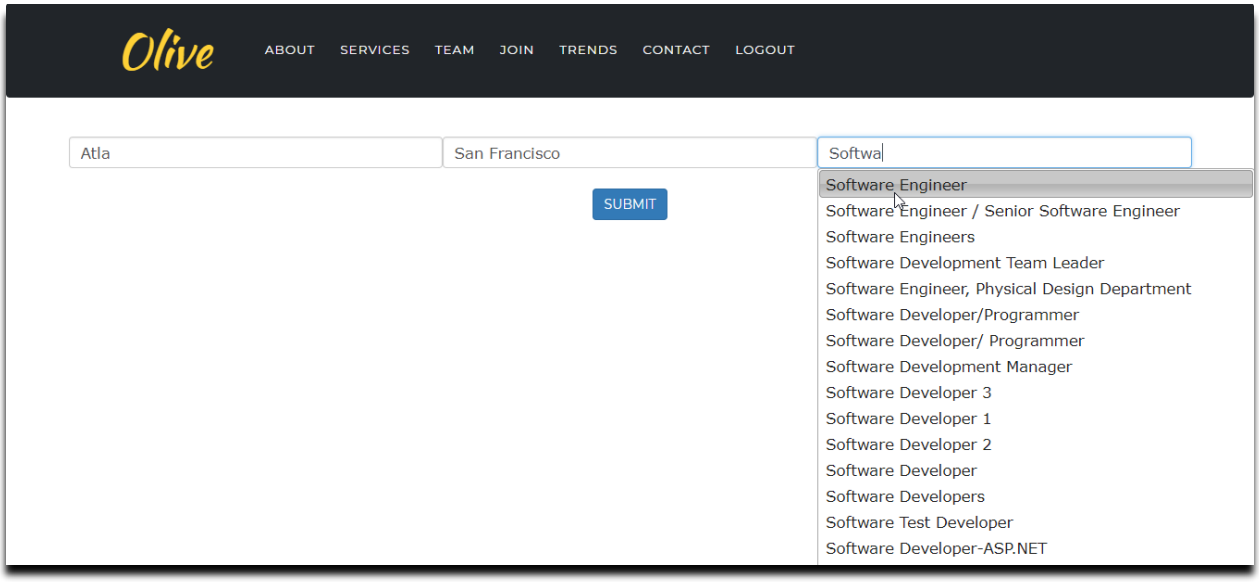
It has the following fields and selection in based on the OR criteria and the user can make any combinations of selection.



User types the first few letters of the company name then the system will automatically suggest all the company names matching that criteria and the user can select the one that interests him.



He further makes the selection of Location where e wants to work and the role in which he wishes to join using the same suggestion search as shown below.



After the selections are made and on click on submit the queries are fired to the database to get the vacancy in the company for that specific role.

If the vacancy exists, then the user has the option to provide two different action.

1.**Apply**:

The user will be directed to the company website so that he can complete the application for that that job profile and take the first step in achieving his dreams.

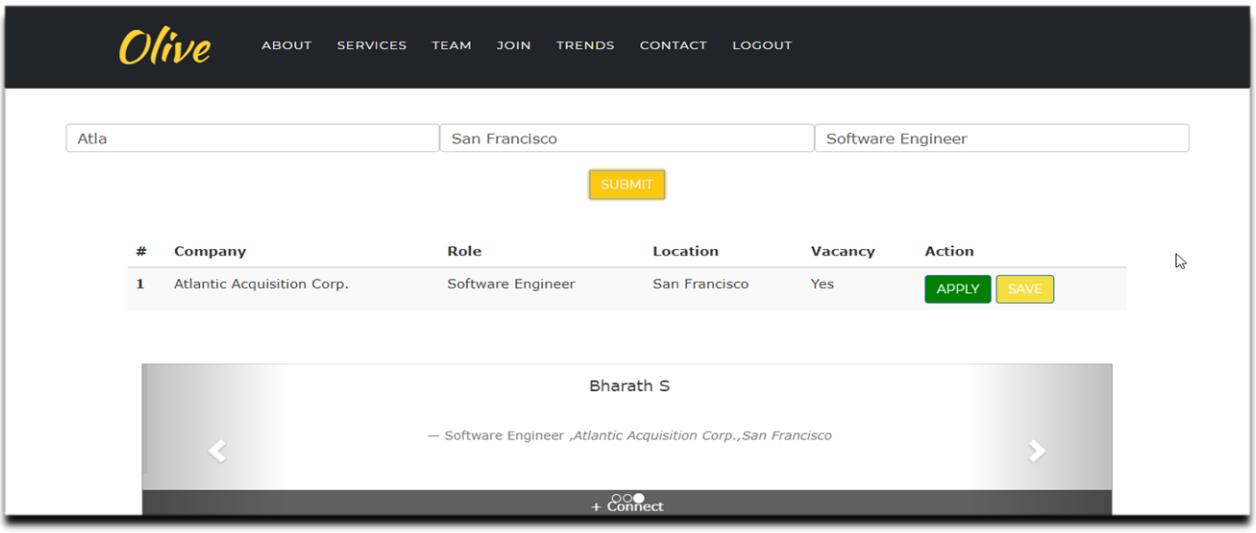
2. **Save:**

The user can also save the job profiles so that he can revisit or apply as and when he is comfortable to complete the application

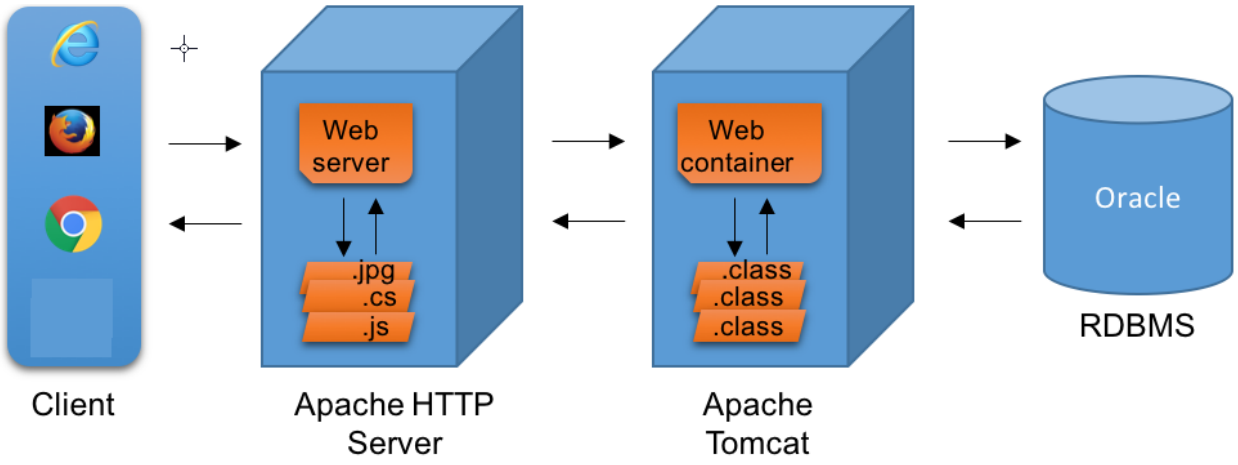
CONNECTIONS:

The screen also has the unique feature where in it shows the mutual connections the logged user can have. This is pulled on the criteria where in the employees already working in the company and the user who is applying are of the same university.

This feature allows the user to connect with the people already working on the sane profile and get to know about the company more.



DEPLOYMENT:



The user will be accessing the system through the web browser and the system is compatible with all the latest browser and is based on the concepts of responsive design.

The frequently used components will be kept in the Apache web Server and this will increase the speed pf accessing the application and will reduce the lag at the client end.

The business logic will be hosted in the application server Apache Tomcat and thus will help for remote and effective maintenance.

We have used Oracle 12c database and the table structures and all the performance points has been considered to make the system scale up for 10000 concurrent users.

FUTURE SCOPE:

The future scope of the project is below:

- Currently our database stores information about companies in the US region. In future, we can extend our database to include all the companies from various countries.
- Same applies to the universities. Currently, we have the data of the Universities in US region, we can extend this to other country universities.
- As an extension to our web application, we could create a mobile app in future.
- In this project we have implemented and displayed a few analytics than can be performed using the data, but in future we can take this forward and demonstrate a whole lot of statistics with the data available.

VIDEO URL:

The Video URL for the project is accessible at this link https://www.youtube.com/watch?v=7x1-rfIDOt0&feature=youtu.be