# AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

***FACULTY OF SCIENCE & TECHNOLOGY***

Course Title

**INTRODUCTION TO DATABASE**

**Summer 2022-2023**
**Section: G**

**TITLE**

**THE JOB MANAGEMENT SYSTEM**

**Supervised By**

MD Sajid Bin Faisal

**Submitted By: Group 06**

| Name | ID |
|---|---|
| Aria Rahman | 22-49494-3 |

## TABLE OF CONTENTS

## 1.INTRODUCTION:

The  job management project is a simplified system which offers a radical platform to the employment. Initially, this system is designed to assigning jobs to the employees who are eligible for the job. To maintain the information of required processes, database management system can be reliable . For that purpose, we can utilize oracle database for SQL, draw.io for ER diagram, java for database connectivity.

## 2.CASE STUDY:

### The Job Management System

In a Job Management System, there is an Authority that acts as an intermediate between Employees and Jobs. The Authority has properties like Name, Phone Number, and a unique identifier as Authority ID. Employees are gathered by the Authority. The information of an Employee includes their ID, Name, Contact Number, Preferred Job, and Job Status.

For storing Jobs, individual Authority takes over. The Job is characterized by properties such as Account Number, Name, Salary, and Age Limit. In a Job, multiple Employees can be required, and it may have been taken by one Employee already. Meanwhile, one Job can be handled by an Employee at a time. Eventually, Employees execute the Job.

The Authority assigns Jobs to Employees. This system is solely responsible for managing Jobs and providing them to Employees.

## 3.ER DIAGRAM:

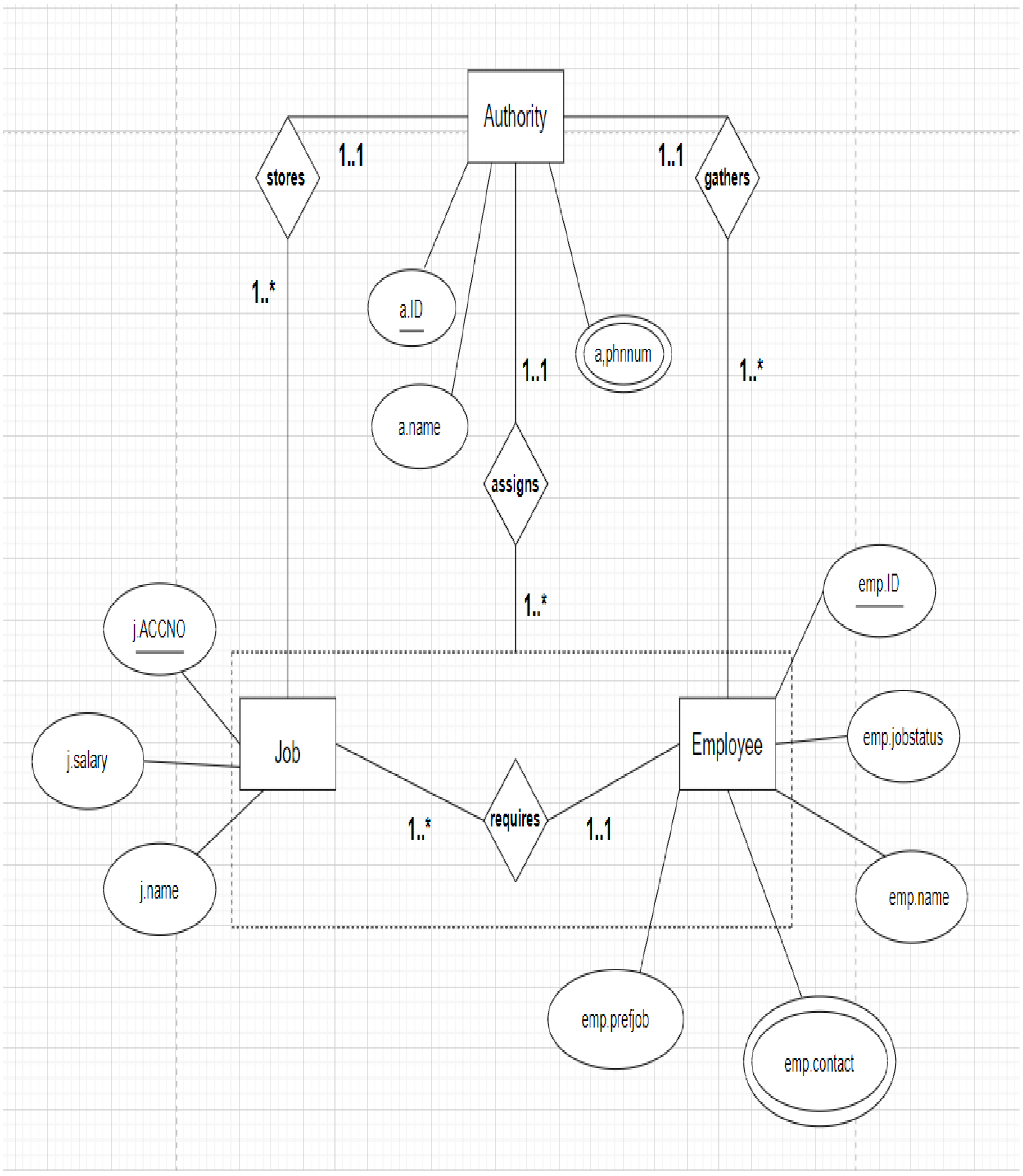


Figure 1: Er diagram for Job Management System.

## 4.NORMALIZATION:

**Relation Name:** Stores (1 .. *)

**UNF:** AID,aname,aphnnum,JACCNO,jsalary,jname.

**1NF:** Same as UNF.

**2NF:**

1. AID(primary key), aname, aphnnum.
2. JACCNO(primary key),jsalary,jname, AID(foreign key).

**3NF**: Same as 2NF.

**Relation Name:** Gathers (1 .. *)

**UNF:** AID,aname,aphnnum,EMPID,empname,empjobstatus,empcontact,empprefjob.

**1NF:** Same as UNF.

**2NF:**

1. AID(primary key), aname, aphnnum.
2. EMPID(primary key),empname, empjobstatus, empcontact, empprefjob, AID(foreign key).

**3NF**: Same as 2NF.

**Relation Name:** Requires (1 .. *)

**UNF:** EMPID,empname,empjobstatus,empcontact,empprefjob, JACCNO,jsalary,jname.

**1NF:** Same as UNF.

**2NF:**

1. JACCNO(primary key),jsalary,jname, EMPID(foreign key).
2. EMPID(primary key),empname,empjobstatus,empcontact,empprefjob.

**3NF**: Same as 2NF.

**<u>Relation Name:</u>** Assigns (1 .. *)

**UNF:**EMPID,empname,empjobstatus,empcontact,empprefjob,JACCNO,jsalary,jname,

AID,aname,aphnnum.

**1NF:** Same as UNF.

**2NF:**

1. EMPID(primary key),empname,empjobstatus,empcontact,empprefjob, JACCNO,jsalary,jname,AID(foreign key).

2. AID(primary key), aname, aphnnum.

**3NF**: Same as 2NF.

## 5.FINALIZATION:

| | | |
|---|---|---|
| 1 | AID(primary key), aname, aphnnum. | |
| 2 | JACCNO(primary key),jsalary,jname, AID(foreign key). | |
| 3 | AID(primary key), aname, aphnnum. | ✕ |
| 4 | EMPID(primary key),empname,empjobstatus,empcontact,empprefjob, AID(foreign key). | |
| 5 | JACCNO(primary key),jsalary,jname, EMPID(foreign key). | |
| 6 | EMPID(primary key),empname,empjobstatus,empcontact,empprefjob. | |
| 7 | EMPID(primary key),empname,empjobstatus,empcontact,empprefjob, JACCNO,jsalary,jname,AID(foreign key). | |
| 8 | AID(primary key), aname, aphnnum | ✕ |

## FINAL TABLES:

**1.**AID(primary key), aname, aphnnum.[**AUTHORITY**]

**2**.JACCNO(primary key),jsalary,jname, AID(foreign key).[**STORE**]

**3.**EMPID(primary key),empname,empjobstatus,empcontact,empprefjob,
AID(foreign key).[**GATHER**]

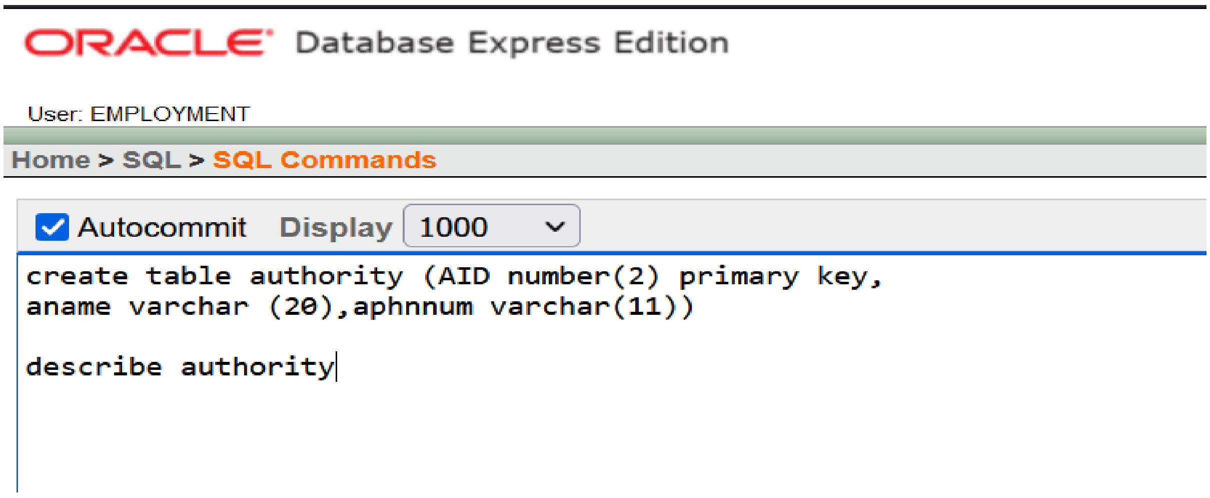
**4**.JACCNO(primary key),jsalary,jname, EMPID(foreign key).[**REQUIRE**]

**5.**EMPID(primary key),empname,empjobstatus,empcontact,empprefjob.[**EMPLOYEE**]

**6**.EMPID(primary  key),empname,empjobstatus,empcontact,empprefjob,
JACCNO,jsalary,jname,AID(foreign key).[**ASSIGN**]

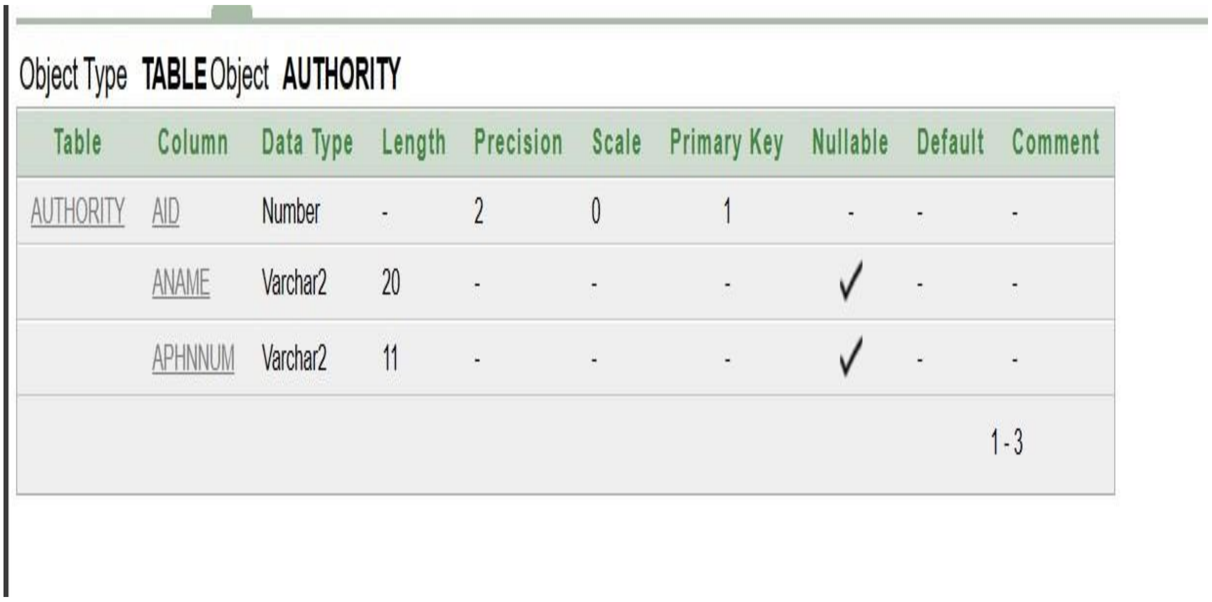
## 6.TABLE CREATION:

1.Table Name: **AUTHORITY**



```
create table authority (AID number(2) primary key,
aname varchar (20),aphnnum varchar(11))

describe authority
```

Fig 1.1: Creation of table Authority.



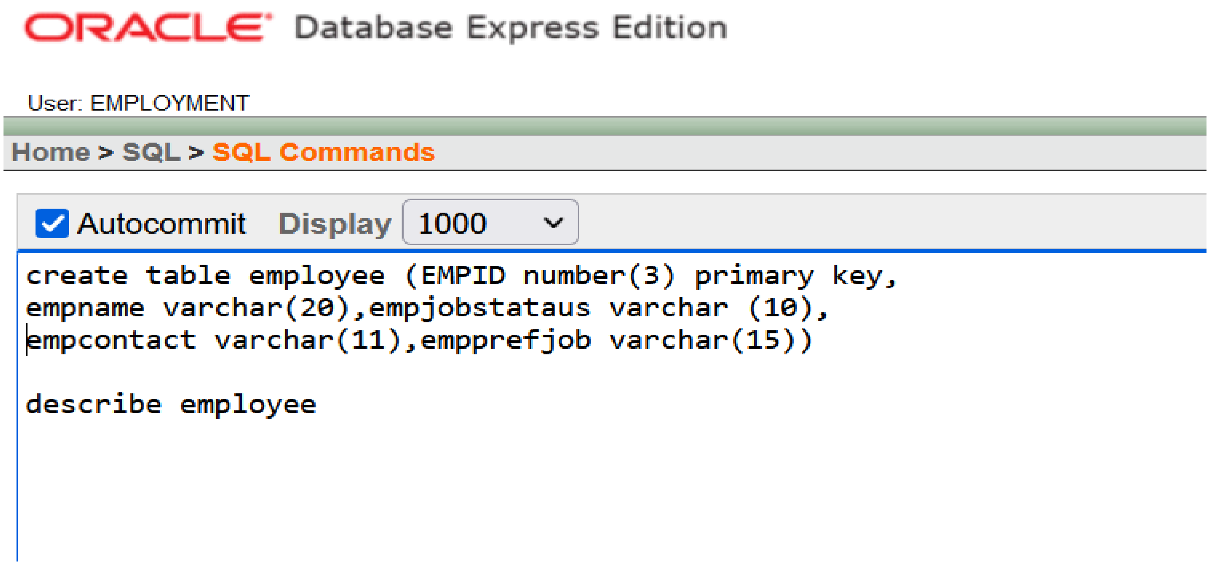
Object Type TABLE Object AUTHORITY

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| AUTHORITY | AID | Number | - | 2 | 0 | 1 | - | - | - |
| | ANAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | APHNNUM | Varchar2 | 11 | - | - | - | ✓ | - | - |

1 - 3

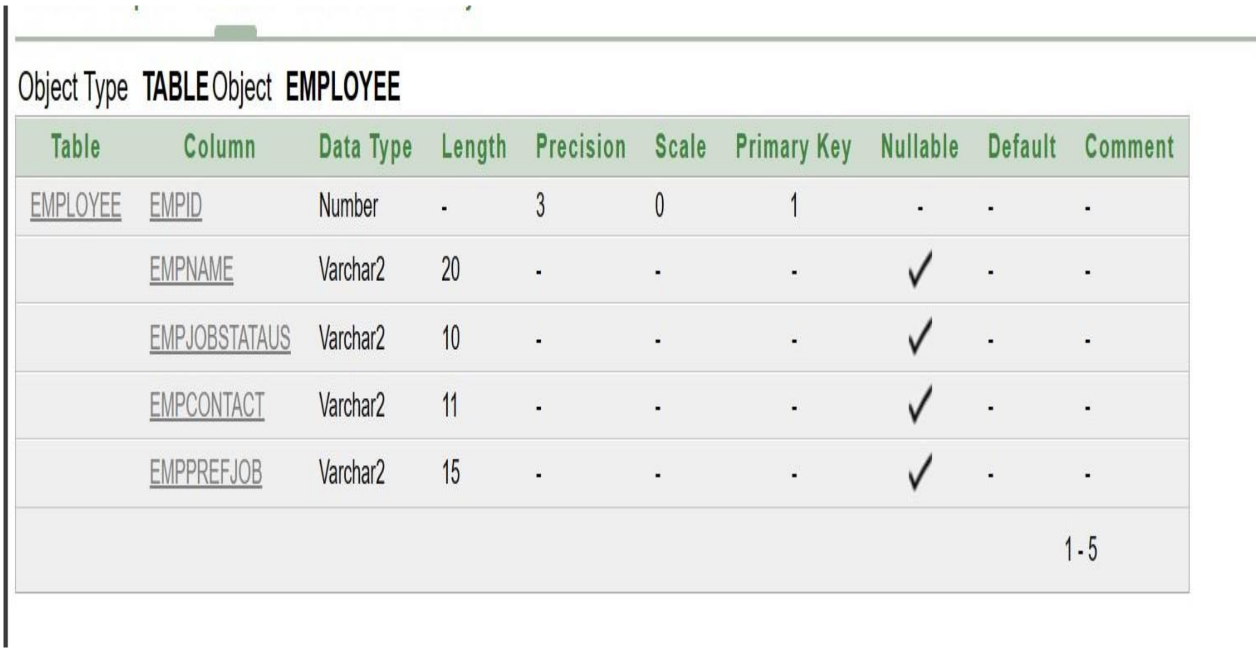
Fig 1.2: Description of table Authority.

## 2.Table Name: **EMPLOYEE**



```
create table employee (EMPID number(3) primary key,
empname varchar(20),empjobstataus varchar (10),
empcontact varchar(11),empprefjob varchar(15))

describe employee
```

Fig 2.1: Creation of table Employee.



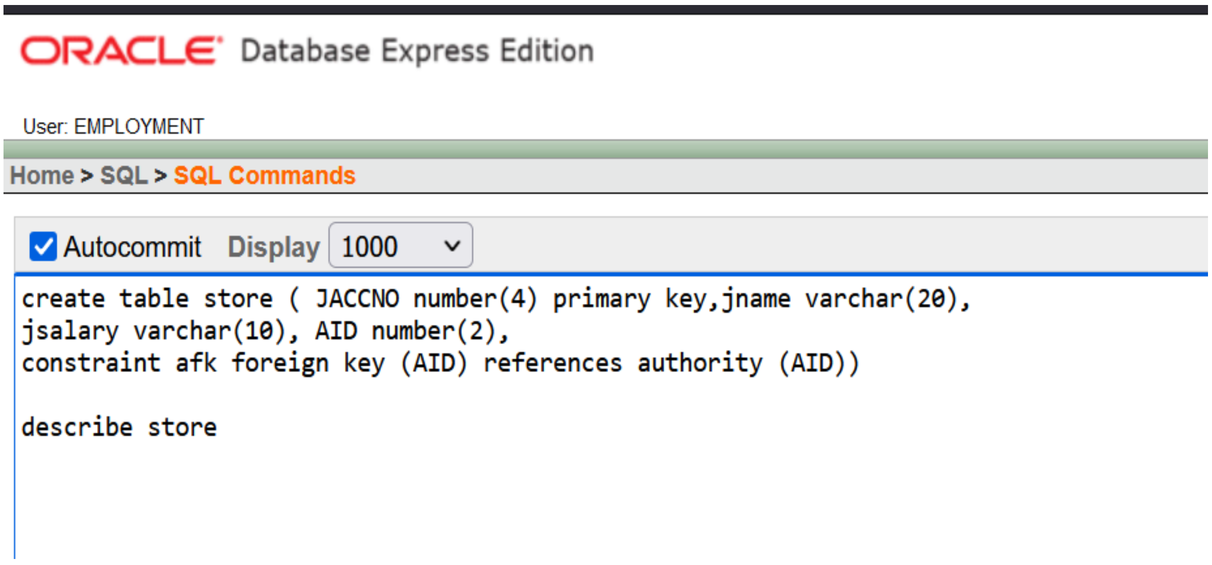
Object Type TABLE Object EMPLOYEE

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| EMPLOYEE | EMPID | Number | - | 3 | 0 | 1 | - | - | - |
| | EMPNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | EMPJOBSTATAUS | Varchar2 | 10 | - | - | - | ✓ | - | - |
| | EMPCONTACT | Varchar2 | 11 | - | - | - | ✓ | - | - |
| | EMPPREFJOB | Varchar2 | 15 | - | - | - | ✓ | - | - |

1 - 5

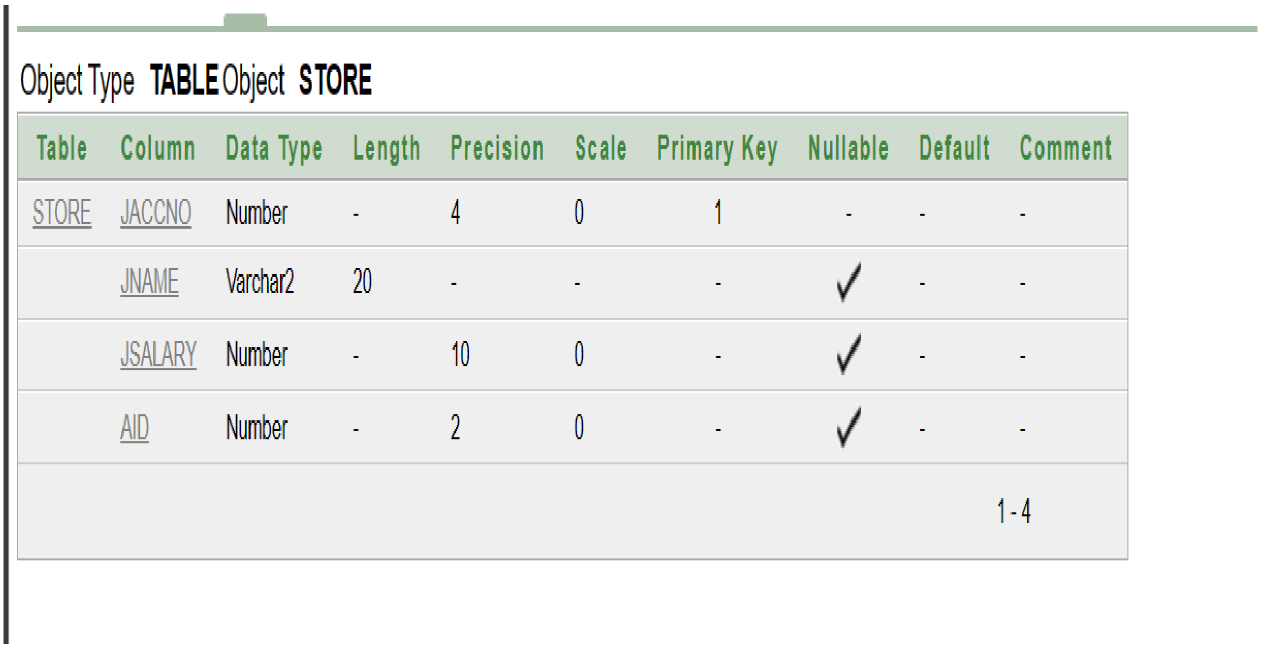
Fig 2.2: Description of table Employee .

3.Table Name: **STORE**



```
create table store ( JACCNO number(4) primary key,jname varchar(20),
jsalary varchar(10), AID number(2),
constraint afk foreign key (AID) references authority (AID))

describe store
```

Fig 3.1: Creation of table Store.



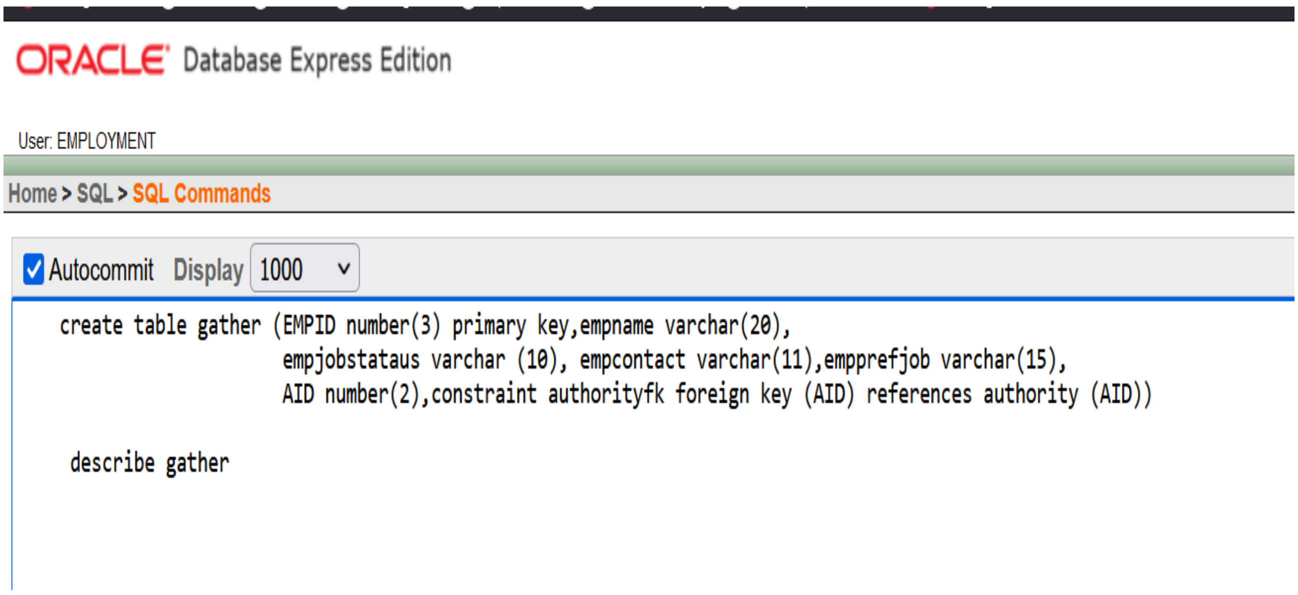
Object Type **TABLE** Object **STORE**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| STORE | JACCNO | Number | - | 4 | 0 | 1 | - | - | - |
| | JNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | JSALARY | Number | - | 10 | 0 | - | ✓ | - | - |
| | AID | Number | - | 2 | 0 | - | ✓ | - | - |

1 - 4

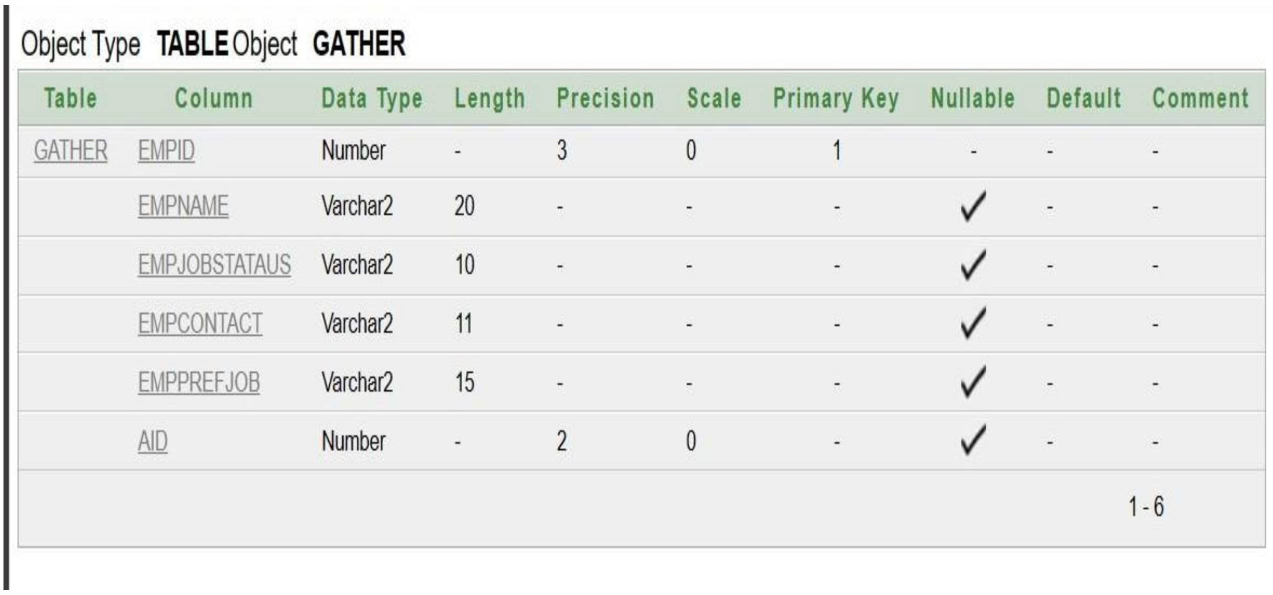
Fig 3.2: Description of table Store.

## 4.Table Name: **GATHER**



```
create table gather (EMPID number(3) primary key,empname varchar(20),
                     empjobstataus varchar (10), empcontact varchar(11),empprefjob varchar(15),
                     AID number(2),constraint authorityfk foreign key (AID) references authority (AID))

 describe gather
```

Fig 4.1: Creation of table Gather.



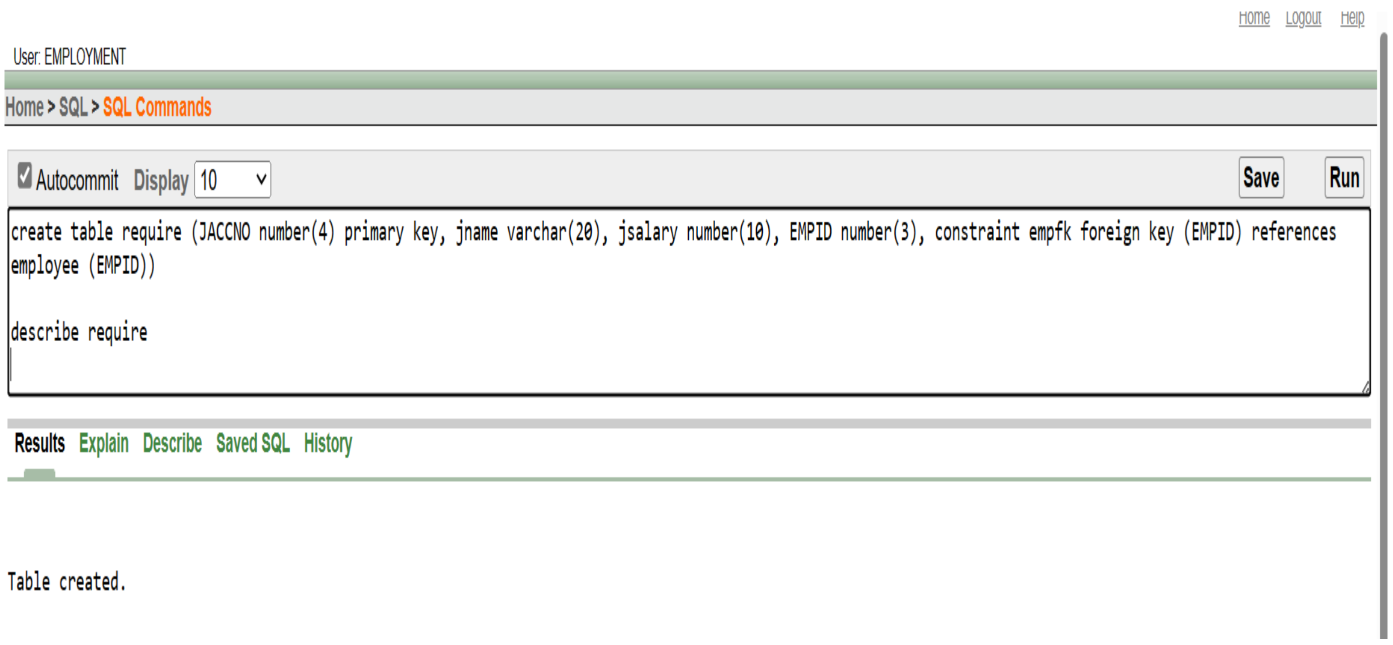
Object Type **TABLE** Object **GATHER**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| GATHER | EMPID | Number | - | 3 | 0 | 1 | - | - | - |
| | EMPNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | EMPJOBSTATAUS | Varchar2 | 10 | - | - | - | ✓ | - | - |
| | EMPCONTACT | Varchar2 | 11 | - | - | - | ✓ | - | - |
| | EMPPREFJOB | Varchar2 | 15 | - | - | - | ✓ | - | - |
| | AID | Number | - | 2 | 0 | - | ✓ | - | - |

1 - 6

Fig 4.2: Description of table Gather.

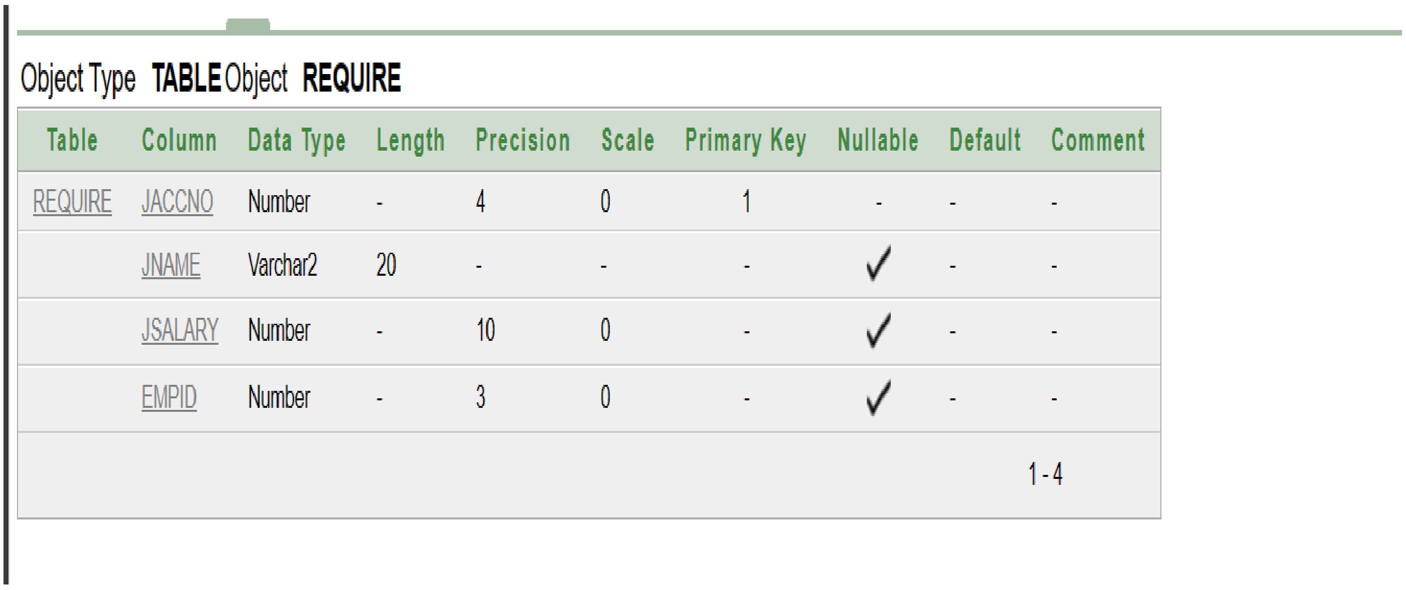
## 5.Table Name: **REQUIRE**


Home Logout Help
User: EMPLOYMENT
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
create table require (JACCNO number(4) primary key, jname varchar(20), jsalary number(10), EMPID number(3), constraint empfk foreign key (EMPID) references employee (EMPID))

describe require
```

```
Table created.
```

Fig 5.1: Creation of table Require.



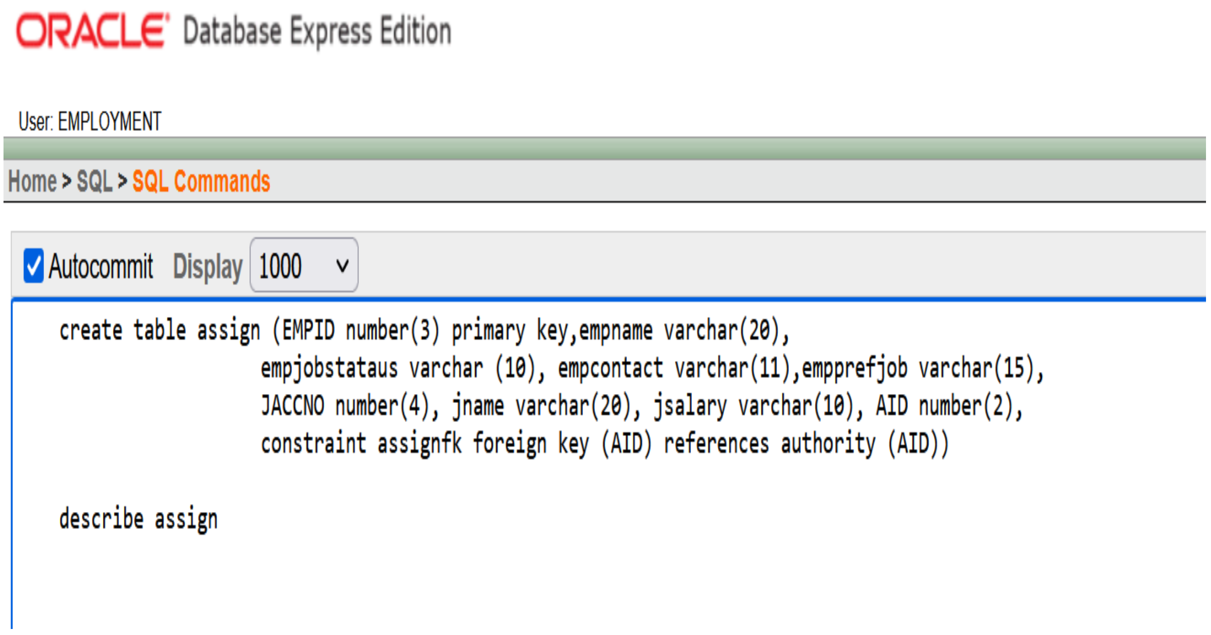
Object Type **TABLE** Object **REQUIRE**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| REQUIRE | JACCNO | Number | - | 4 | 0 | 1 | - | - | - |
| | JNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | JSALARY | Number | - | 10 | 0 | - | ✓ | - | - |
| | EMPID | Number | - | 3 | 0 | - | ✓ | - | - |

1 - 4

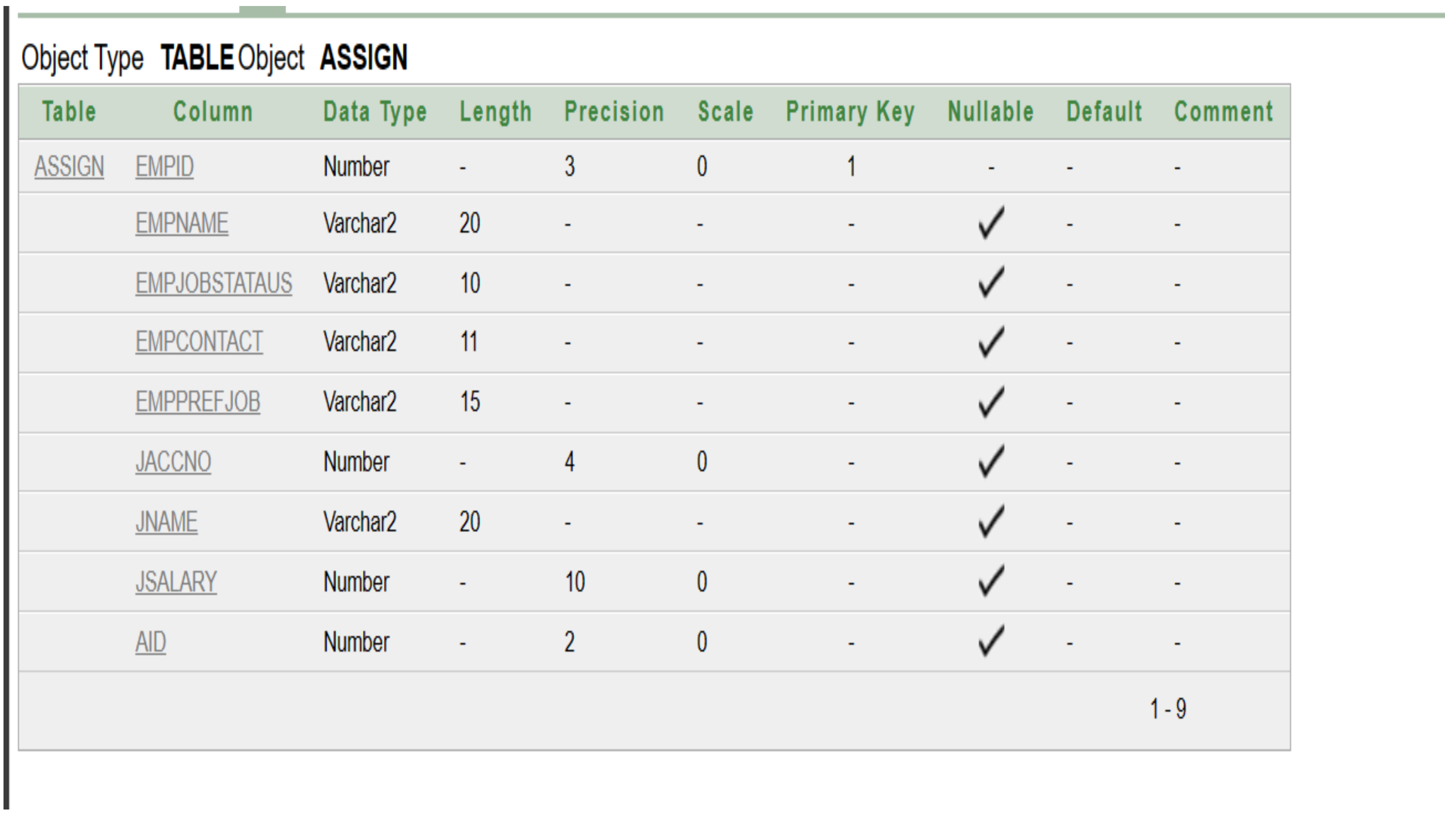
Fig 5.2: Description of table Require.

## 6.Table Name: **ASSIGN**



```
create table assign (EMPID number(3) primary key,empname varchar(20),
                     empjobstataus varchar (10), empcontact varchar(11),empprefjob varchar(15),
                     JACCNO number(4), jname varchar(20), jsalary varchar(10), AID number(2),
                     constraint assignfk foreign key (AID) references authority (AID))


describe assign
```

Fig 6.1: Creation of table Assign.



Object Type **TABLE** Object **ASSIGN**

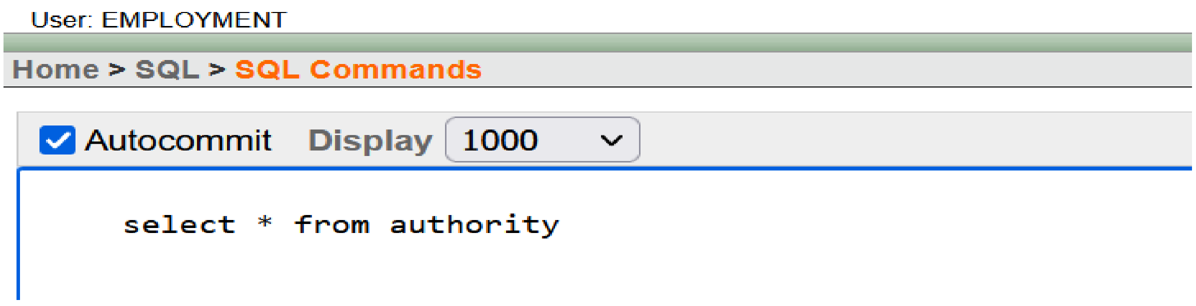
| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| ASSIGN | EMPID | Number | - | 3 | 0 | 1 | - | - | - |
| | EMPNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | EMPJOBSTATAUS | Varchar2 | 10 | - | - | - | ✓ | - | - |
| | EMPCONTACT | Varchar2 | 11 | - | - | - | ✓ | - | - |
| | EMPPREFJOB | Varchar2 | 15 | - | - | - | ✓ | - | - |
| | JACCNO | Number | - | 4 | 0 | - | ✓ | - | - |
| | JNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | JSALARY | Number | - | 10 | 0 | - | ✓ | - | - |
| | AID | Number | - | 2 | 0 | - | ✓ | - | - |

1 - 9

Fig 6.2: Description of table Assign.

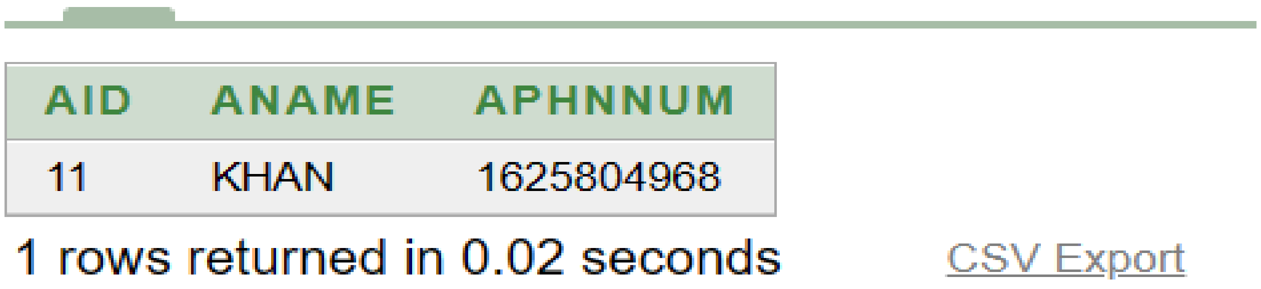
## 7.DATA INSERTION:

1.Table Name: **AUTHORITY**



```
select * from authority
```

Fig 7.1: Insertion of table Authority.

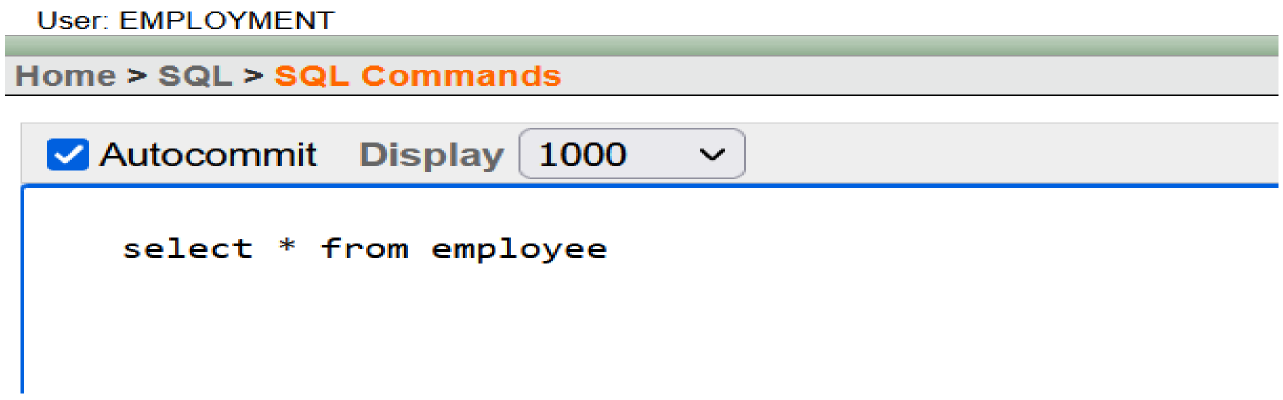


| AID | ANAME | APHNNUM |
|---|---|---|
| 11 | KHAN | 1625804968 |

1 rows returned in 0.02 seconds

Fig 7.2: Rows in table Authority .

2.Table Name: **EMPLOYEE**

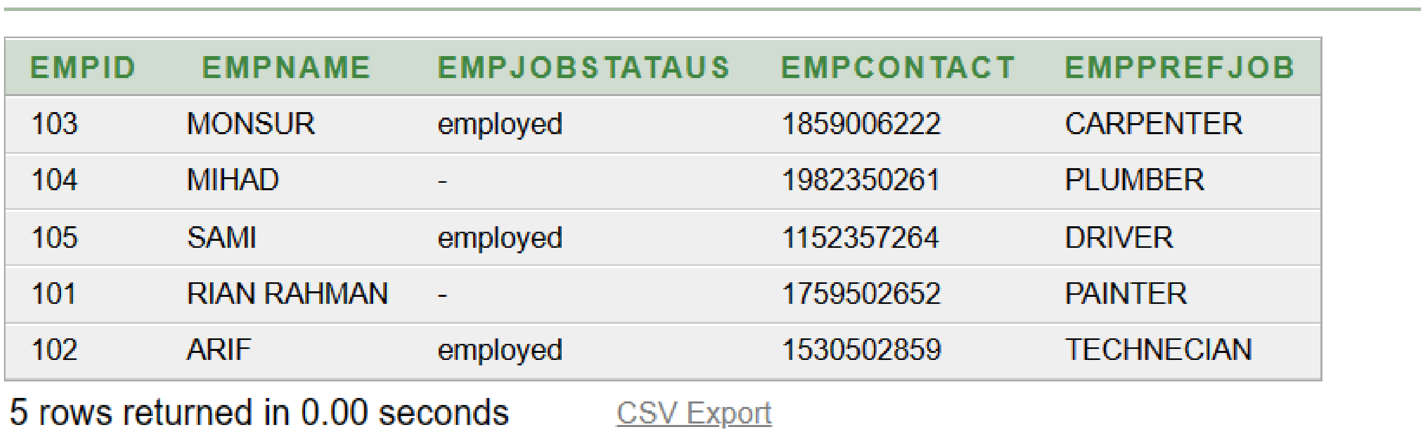


User: EMPLOYMENT

Home > SQL > SQL Commands

```
select * from employee
```

Fig 7.3: Insertion of table Employee.

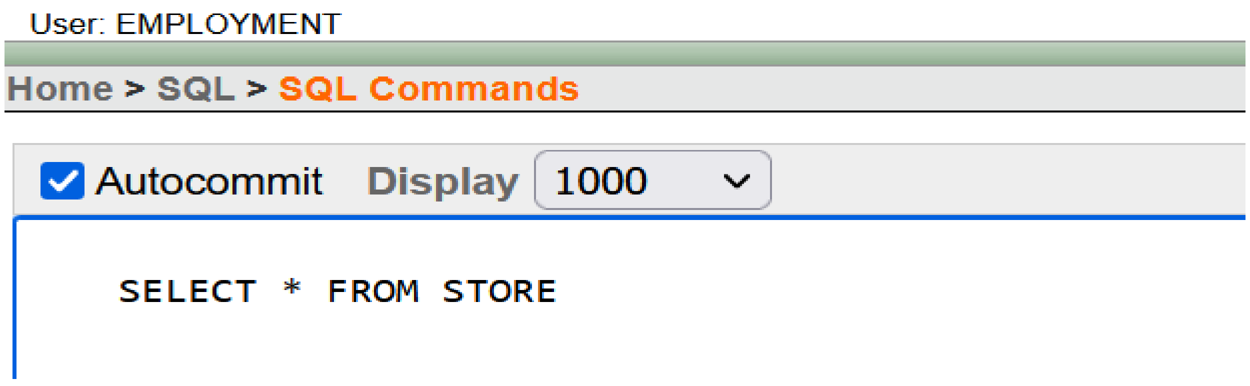


| EMPID | EMPNAME | EMPJOBSTATAUS | EMPCONTACT | EMPPREFJOB |
|---|---|---|---|---|
| 103 | MONSUR | employed | 1859006222 | CARPENTER |
| 104 | MIHAD | - | 1982350261 | PLUMBER |
| 105 | SAMI | employed | 1152357264 | DRIVER |
| 101 | RIAN RAHMAN | - | 1759502652 | PAINTER |
| 102 | ARIF | employed | 1530502859 | TECHNECIAN |

5 rows returned in 0.00 seconds

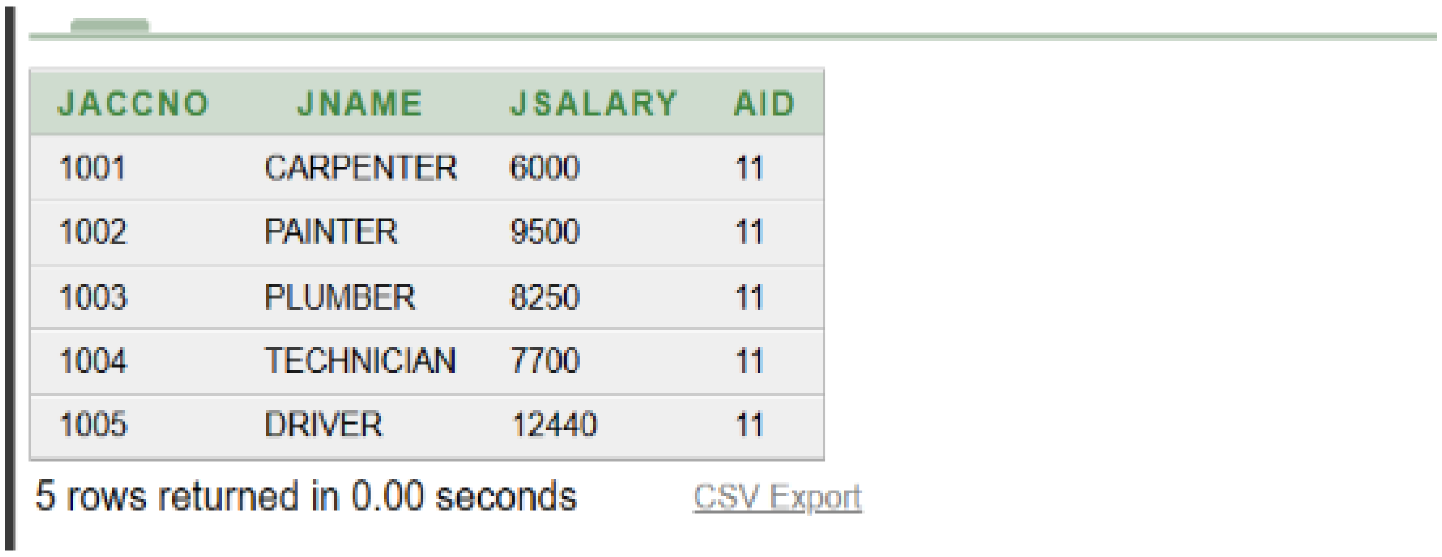
Fig 7.4: Rows in table Employee.

3.Table Name: **STORE**



```
SELECT * FROM STORE
```

Fig 7.5: Insertion of table Store.

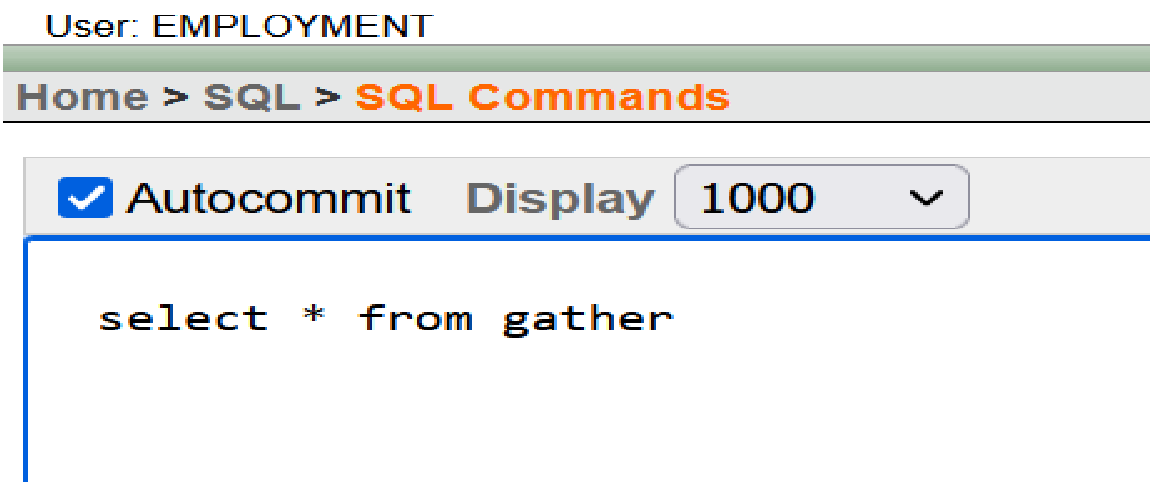


| JACCNO | JNAME | JSALARY | AID |
|---|---|---|---|
| 1001 | CARPENTER | 6000 | 11 |
| 1002 | PAINTER | 9500 | 11 |
| 1003 | PLUMBER | 8250 | 11 |
| 1004 | TECHNICIAN | 7700 | 11 |
| 1005 | DRIVER | 12440 | 11 |

5 rows returned in 0.00 seconds

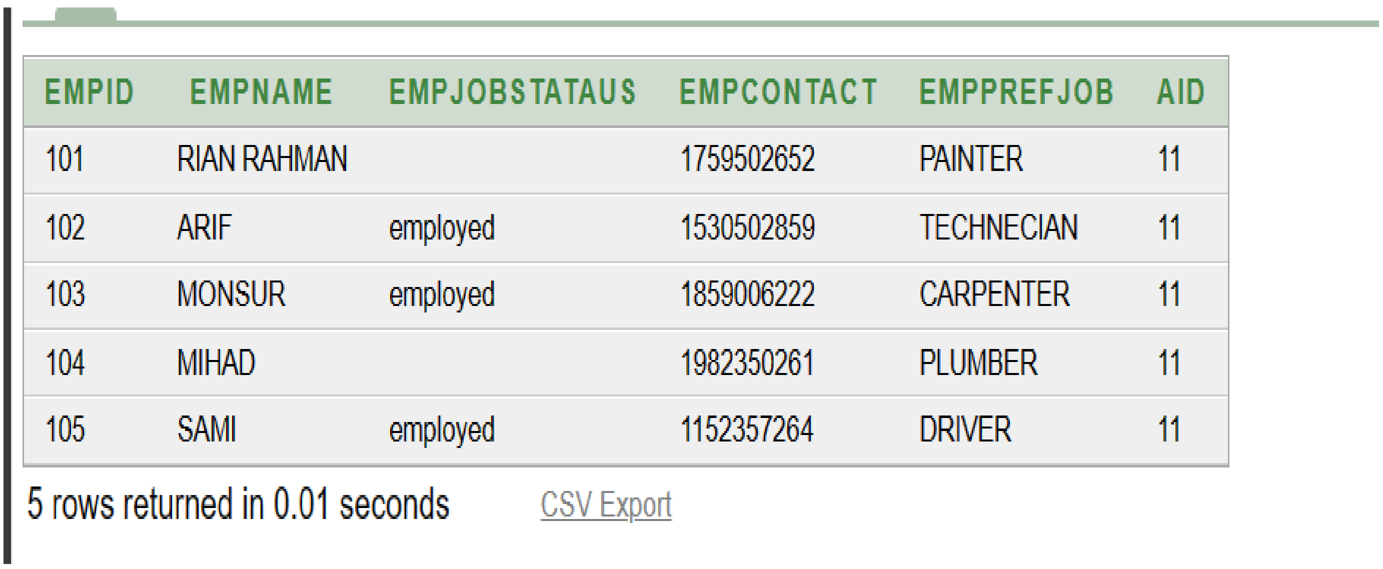
Fig 7.6: Rows in table Store.

4.Table Name: **GATHER**



```
select * from gather
```

Fig 7.7: Insertion of table Gather.

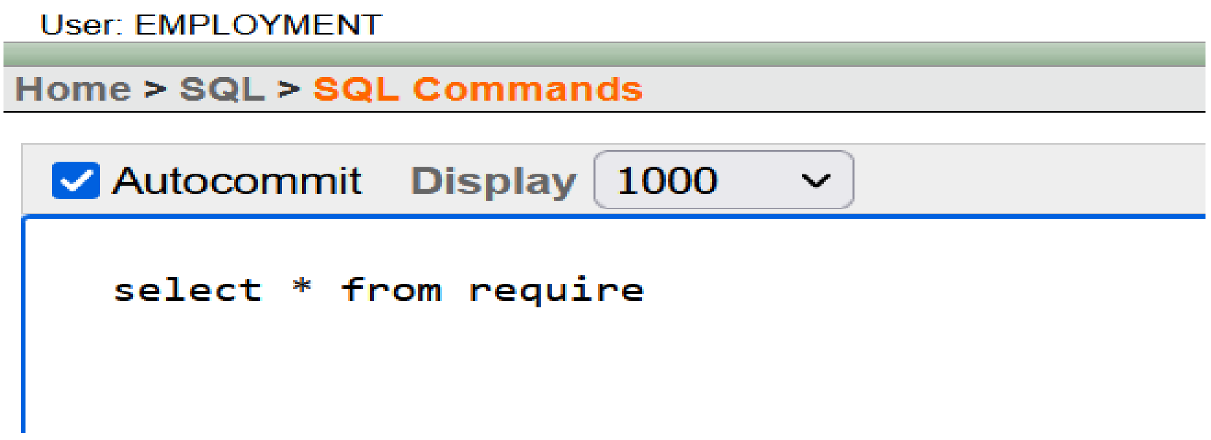


| EMPID | EMPNAME | EMPJOBSTATAUS | EMPCONTACT | EMPPREFJOB | AID |
|---|---|---|---|---|---|
| 101 | RIAN RAHMAN | | 1759502652 | PAINTER | 11 |
| 102 | ARIF | employed | 1530502859 | TECHNECIAN | 11 |
| 103 | MONSUR | employed | 1859006222 | CARPENTER | 11 |
| 104 | MIHAD | | 1982350261 | PLUMBER | 11 |
| 105 | SAMI | employed | 1152357264 | DRIVER | 11 |

5 rows returned in 0.01 seconds    CSV Export

Fig 7.8: Rows in table Gather.

5.Table Name: **REQUIRE**



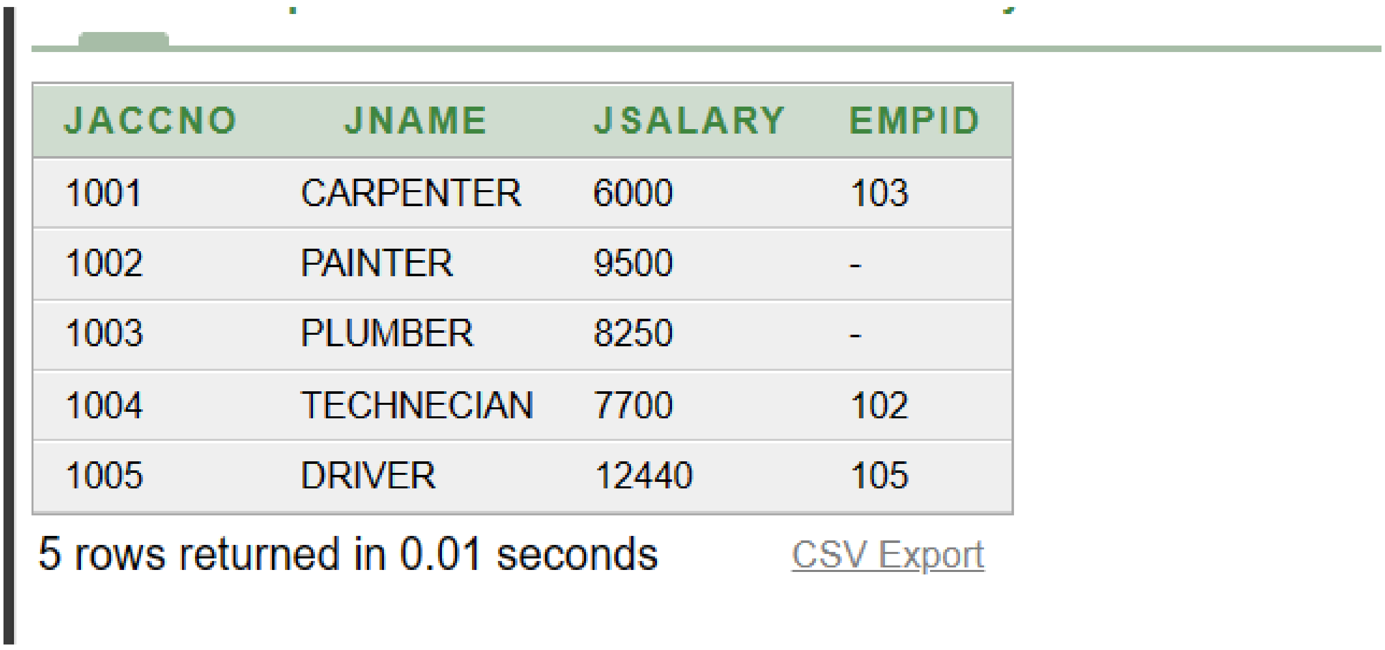
User: EMPLOYMENT

Home > SQL > SQL Commands

Autocommit [✓] Display 1000

```
select * from require
```

Fig 7.9: Insertion of table Require.

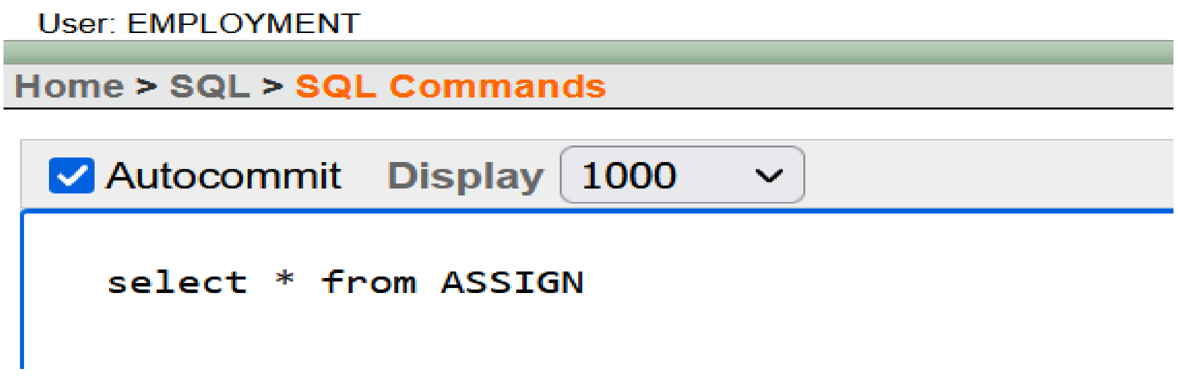


| JACCNO | JNAME | JSALARY | EMPID |
|---|---|---|---|
| 1001 | CARPENTER | 6000 | 103 |
| 1002 | PAINTER | 9500 | - |
| 1003 | PLUMBER | 8250 | - |
| 1004 | TECHNECIAN | 7700 | 102 |
| 1005 | DRIVER | 12440 | 105 |

5 rows returned in 0.01 seconds CSV Export

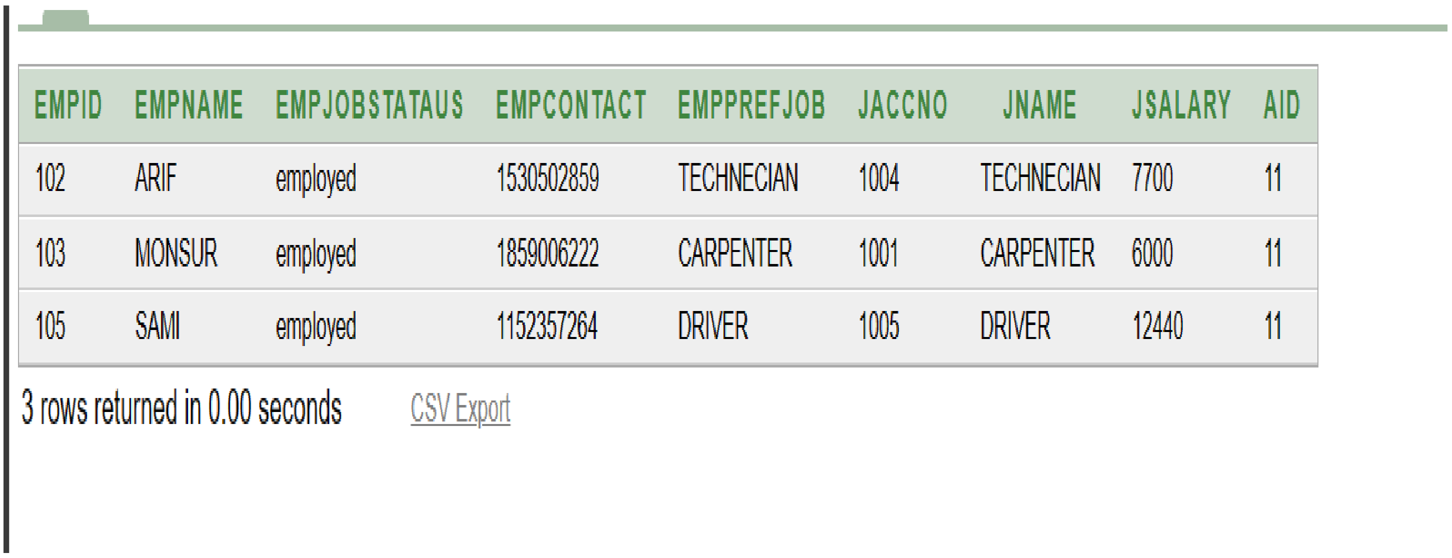
Fig 7.10: Rows in table Require.

6.Table Name: **ASSIGN**



```
select * from ASSIGN
```

Fig 7.9: Insertion of table Assign.



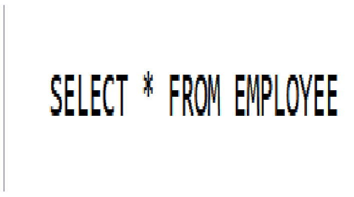
| EMPID | EMPNAME | EMPJOBSTATAUS | EMPCONTACT | EMPPREFJOB | JACCNO | JNAME | JSALARY | AID |
|---|---|---|---|---|---|---|---|---|
| 102 | ARIF | employed | 1530502859 | TECHNECIAN | 1004 | TECHNECIAN | 7700 | 11 |
| 103 | MONSUR | employed | 1859006222 | CARPENTER | 1001 | CARPENTER | 6000 | 11 |
| 105 | SAMI | employed | 1152357264 | DRIVER | 1005 | DRIVER | 12440 | 11 |

3 rows returned in 0.00 seconds CSV Export

Fig 7.10: Rows in table Assign.

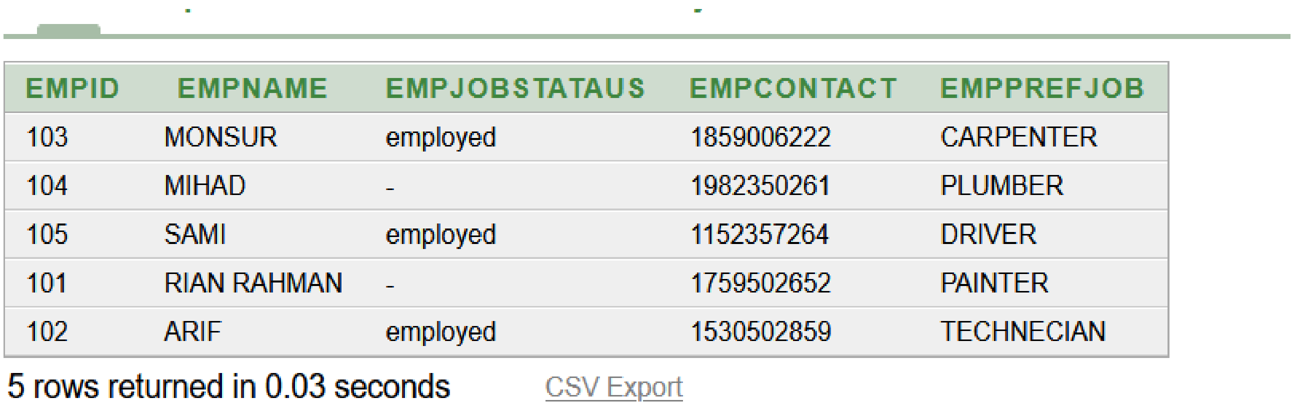
## 8.QUERY TEST:

A. **SIMPLE QUERY:** Show the details of employees.



```
SELECT * FROM EMPLOYEE
```

Figure A.1:SQL command for simple query.

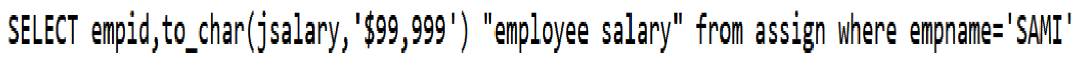


| EMPID | EMPNAME | EMPJOBSTATAUS | EMPCONTACT | EMPPREFJOB |
|---|---|---|---|---|
| 103 | MONSUR | employed | 1859006222 | CARPENTER |
| 104 | MIHAD | - | 1982350261 | PLUMBER |
| 105 | SAMI | employed | 1152357264 | DRIVER |
| 101 | RIAN RAHMAN | - | 1759502652 | PAINTER |
| 102 | ARIF | employed | 1530502859 | TECHNECIAN |

5 rows returned in 0.03 seconds CSV Export

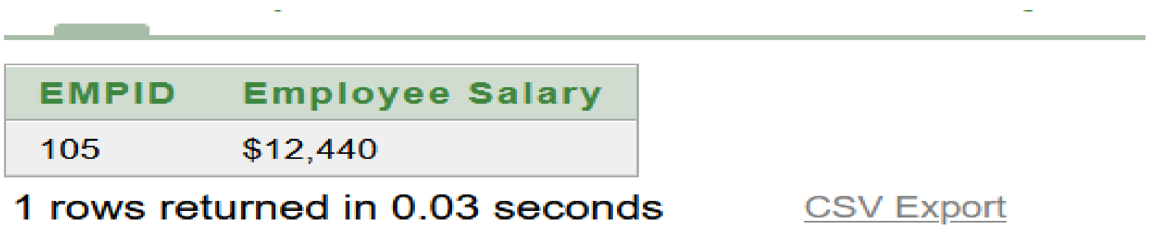
Figure A.2: Execution of  Simple query.

B. **QUERY WITH A SINGLE ROW FUNCTION:** Show the salary of 'SAMI' in '$99,999' format.



```
SELECT empid,to_char(jsalary,'$99,999') "employee salary" from assign where empname='SAMI'
```

Figure B.1:SQL command for single row function query.



| EMPID | Employee Salary |
|---|---|
| 105 | $12,440 |

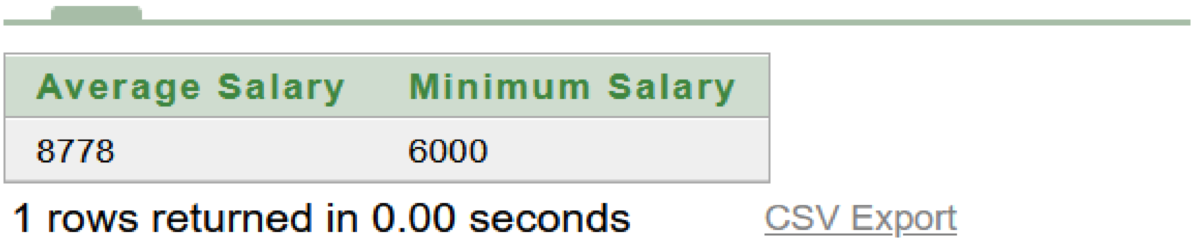
1 rows returned in 0.03 seconds CSV Export

Figure B.2: Execution of Single row function query.

C. **QUERY WITH AGGREGATE FUNCTION:** Show the average, minimum salary of all the jobs.

```
select AVG(jsalary) "average salary",min(jsalary) "minimum salary" from store
```

Figure C.1:SQL command for aggregated function query.



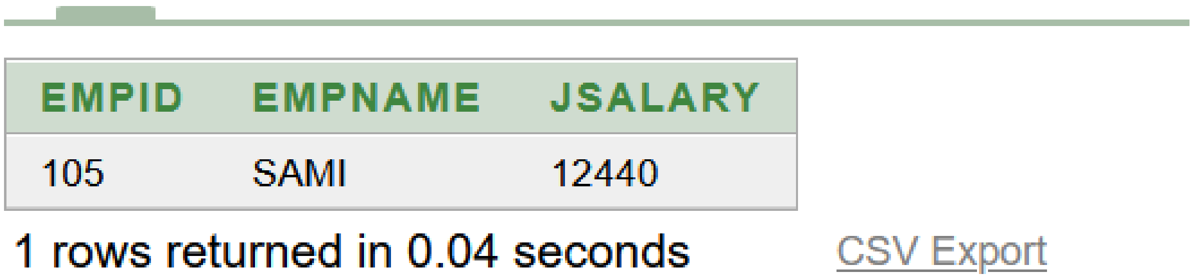
```
Average Salary    Minimum Salary
8778              6000

1 rows returned in 0.00 seconds        CSV Export
```

Figure C.2: Execution of aggregated function query.

D. **SINGLE ROW SUBQUERY:** Show the name, employee id & job of the employee who gets more salary than Arif.

```
select empid,empname,jsalary from assign where jsalary>
    (select jsalary from assign where empname='ARIF')
```

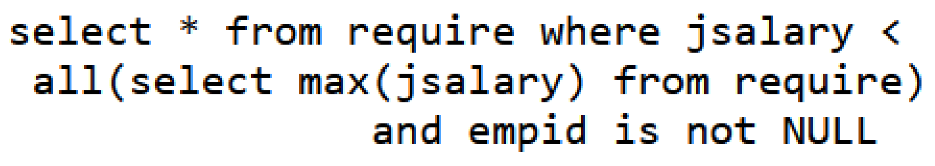
Figure D.1: SQL command for single row subquery.



```
EMPID    EMPNAME    JSALARY
105      SAMI       12440

1 rows returned in 0.04 seconds        CSV Export
```

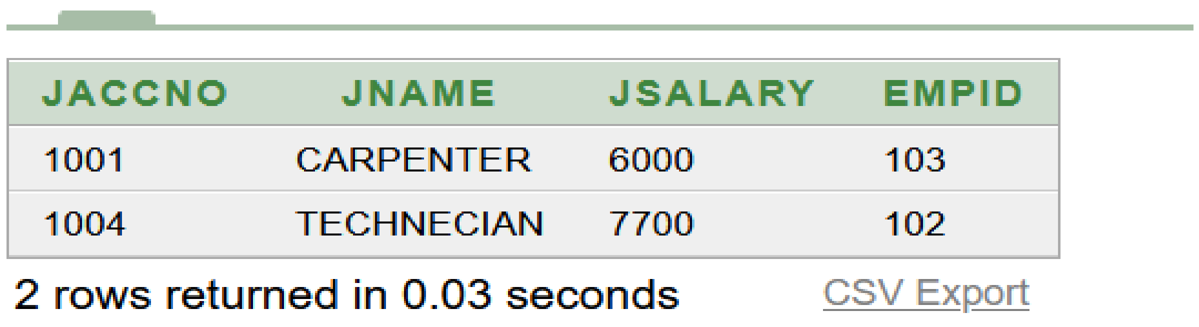
Figure D.2: Execution of single row subquery.

**E. <u>MULTIPLE ROW SUBQUERY:</u>** Show the details of a job where employees get less salary than max salary of all the employees and empid is not null.



```
select * from require where jsalary <
 all(select max(jsalary) from require)
              and empid is not NULL
```

Figure E.1: SQL command for multiple row subquery.



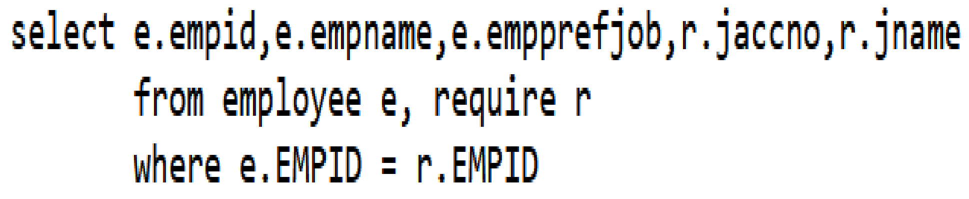
| JACCNO | JNAME | JSALARY | EMPID |
|---|---|---|---|
| 1001 | CARPENTER | 6000 | 103 |
| 1004 | TECHNECIAN | 7700 | 102 |

2 rows returned in 0.03 seconds        CSV Export

Figure E.2: Execution of multiple row subquery.

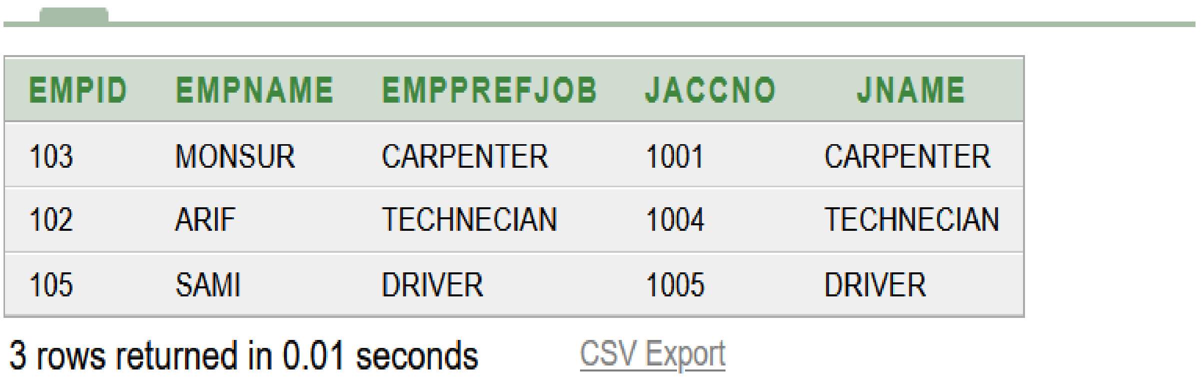
**F. <u>JOINING QUERY:</u>**

i. **EQUIJOIN :** Display employee name, preferable job, jobaccno,jobname with an equijoin of table employee and require.



```
select e.empid,e.empname,e.empprefjob,r.jaccno,r.jname
       from employee e, require r
       where e.EMPID = r.EMPID
```

Figure F.1: SQL command for equijoin query.



| EMPID | EMPNAME | EMPPREFJOB | JACCNO | JNAME |
|---|---|---|---|---|
| 103 | MONSUR | CARPENTER | 1001 | CARPENTER |
| 102 | ARIF | TECHNECIAN | 1004 | TECHNECIAN |
| 105 | SAMI | DRIVER | 1005 | DRIVER |

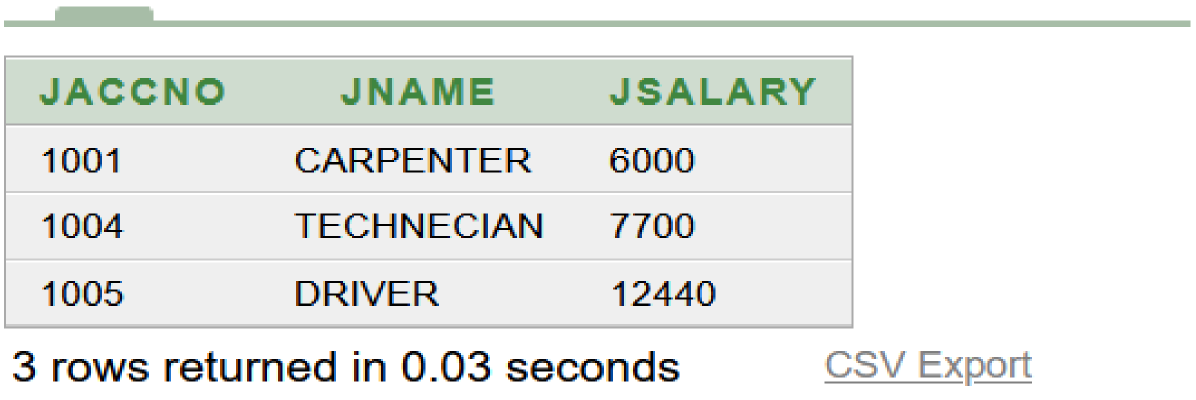
3 rows returned in 0.01 seconds CSV Export

Figure F.2: Execution of equijoin query.

ii. **SELFJOIN:** Display a self-joining in require table where empID in the employee table is equal to empID in the job table.

```
select job.jaccno,job.jname,employee.jsalary
    from require job, require employee
    where job.EMPID = employee.EMPID
```

Figure F.3: SQL command for self-joining query.



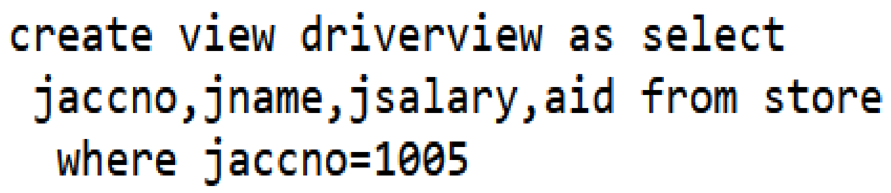
| JACCNO | JNAME | JSALARY |
|---|---|---|
| 1001 | CARPENTER | 6000 |
| 1004 | TECHNECIAN | 7700 |
| 1005 | DRIVER | 12440 |

3 rows returned in 0.03 seconds CSV Export

Figure E.2: Execution of self-joining query.

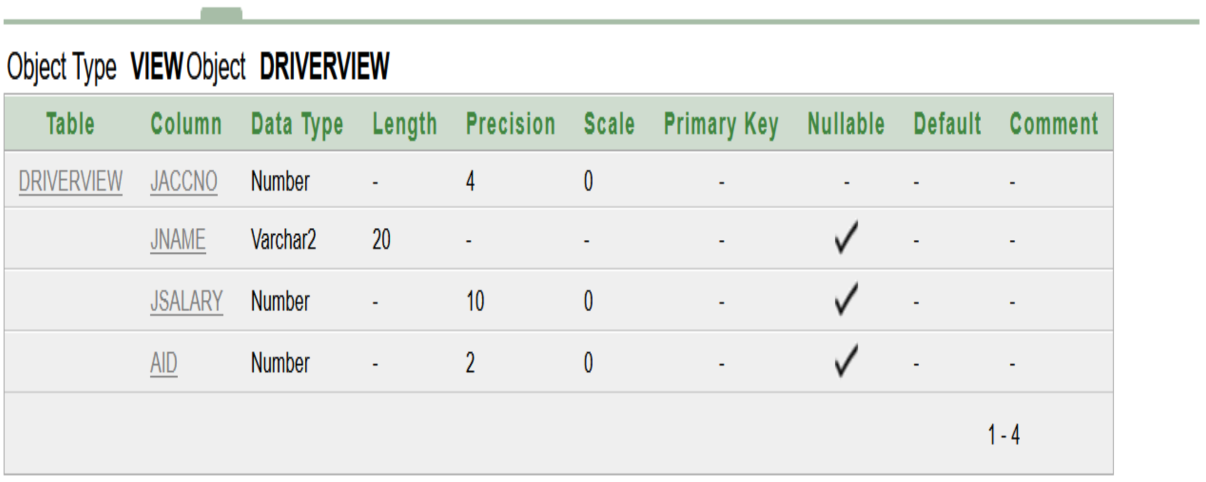
## G. <u>CREATE VIEW:</u>

a. **SIMPLE VIEW:** create a view named as driverview where the job accno 1005 will be shown over the columns job name, job account no ,authority id and job salary.



```
create view driverview as select
 jaccno,jname,jsalary,aid from store
  where jaccno=1005
```

Figure a.1: Simple view creation SQL command.

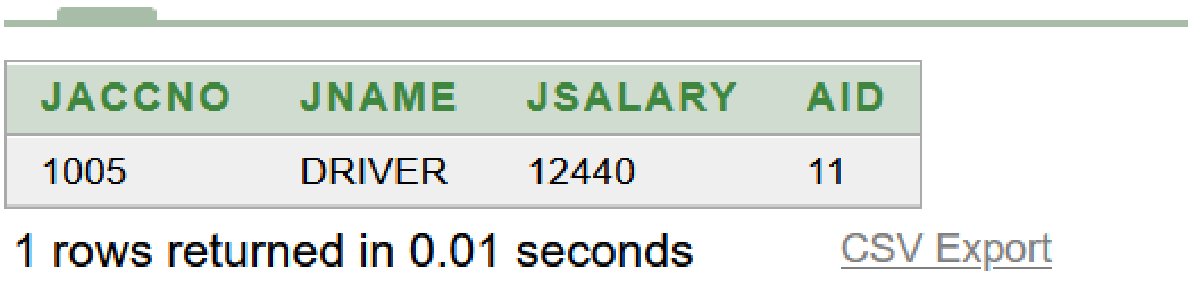


Object Type **VIEW** Object **DRIVERVIEW**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| DRIVERVIEW | JACCNO | Number | - | 4 | 0 | - | - | - | - |
|  | JNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
|  | JSALARY | Number | - | 10 | 0 | - | ✓ | - | - |
|  | AID | Number | - | 2 | 0 | - | ✓ | - | - |

1 - 4

Figure a.2: Description of the simple view.

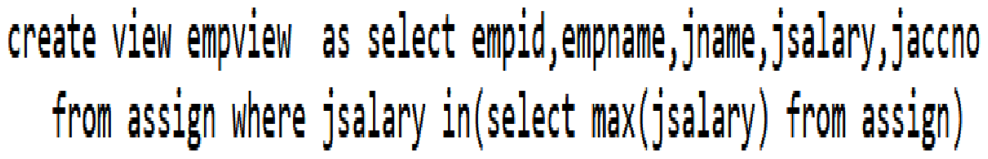


| JACCNO | JNAME | JSALARY | AID |
|---|---|---|---|
| 1005 | DRIVER | 12440 | 11 |

1 rows returned in 0.01 seconds CSV Export

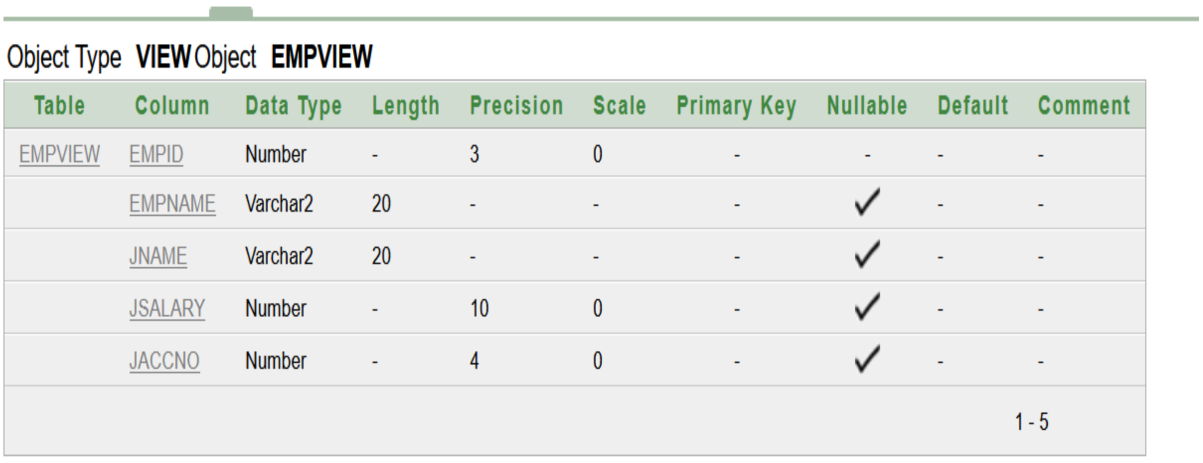
Figure a.3: Result of the simple view as a whole table.

b. **COMPLEX VIEW:** create a view named as empview where the job salary will be max salary shown over the column's employee id,employee name, job accno, job name, and job salary.



```
create view empview  as select empid,empname,jname,jsalary,jaccno
  from assign where jsalary in(select max(jsalary) from assign)
```

Figure b.1: Complex view creation SQL command.

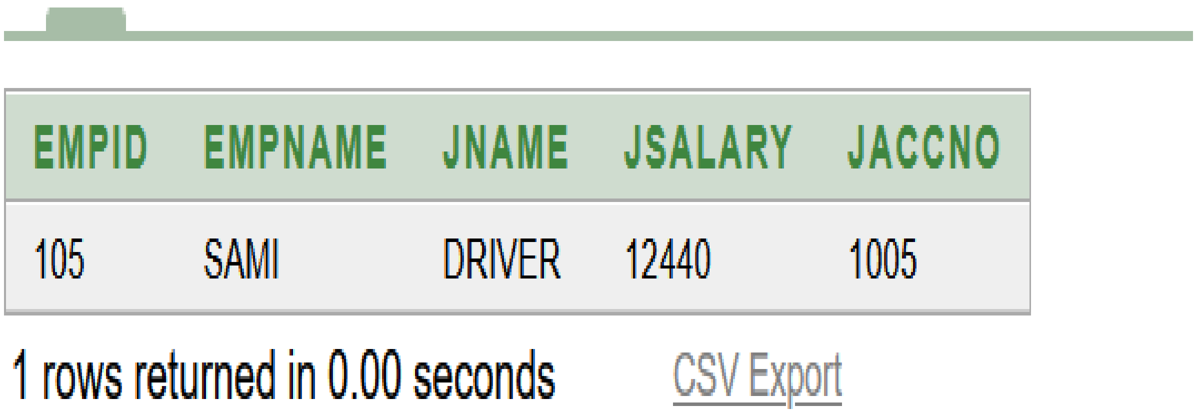


Object Type **VIEW** Object **EMPVIEW**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| EMPVIEW | EMPID | Number | - | 3 | 0 | - | - | - | - |
| | EMPNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | JNAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | JSALARY | Number | - | 10 | 0 | - | ✓ | - | - |
| | JACCNO | Number | - | 4 | 0 | - | ✓ | - | - |

1 - 5

Figure b.2: Description of the complex view.



| EMPID | EMPNAME | JNAME | JSALARY | JACCNO |
|---|---|---|---|---|
| 105 | SAMI | DRIVER | 12440 | 1005 |

1 rows returned in 0.00 seconds    CSV Export

Figure b.3: Result of the complex view as a whole table.

## 9.DATABASE CONNECTION:

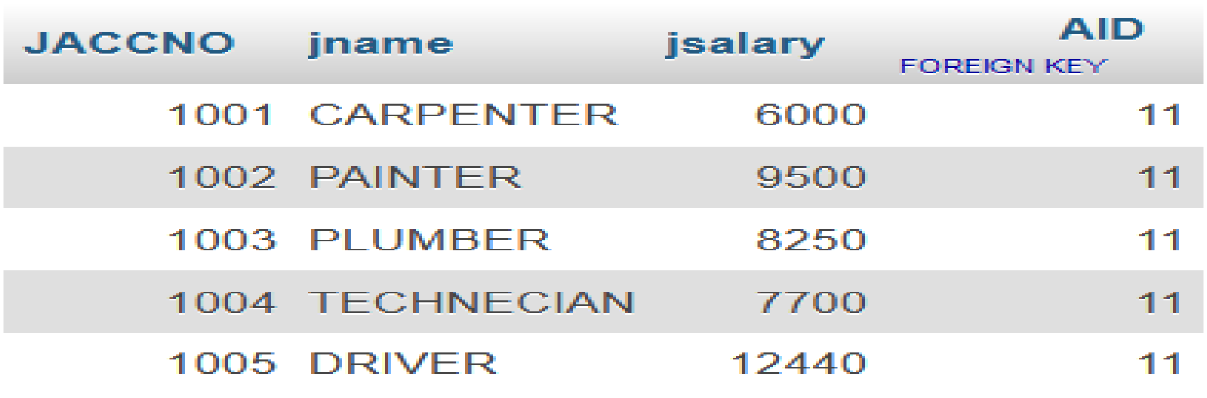
In database connection process ,we needed to follow these steps which are given below:

I. **Database connector**: Firstly, installed MySQL java connector (jar file) which is **mysql-connector-j-8.2.0.jar.**

II. **MySQL Server:** xampp server.

1) In this, we select the xampp icon control UI and then start both Apache and mysql.
2) Next, we go to the MySQL admin panel to create a database named **employment** and inside of it, create a table named **store** and insert values into the table.

### THE STORE TABLE IN MySQL:



| JACCNO | jname | jsalary | AID FOREIGN KEY |
|---|---|---|---|
| 1001 | CARPENTER | 6000 | 11 |
| 1002 | PAINTER | 9500 | 11 |
| 1003 | PLUMBER | 8250 | 11 |
| 1004 | TECHNECIAN | 7700 | 11 |
| 1005 | DRIVER | 12440 | 11 |

Fig 9.1: INSERTED ROWS IN STORE TABLE.

III. **IDE:** Set up IntelliJ IDEA by using the PROJECT STRUCTURE settings to ensure that the jar file is added in module's dependencies and to make a solid connection, also added the jar file in the library.

Now, for setting up the database connection with IDE final steps are:

1. **REGISTERING THE DRIVER:** Initially registered MySQL JBDC DRIVER.

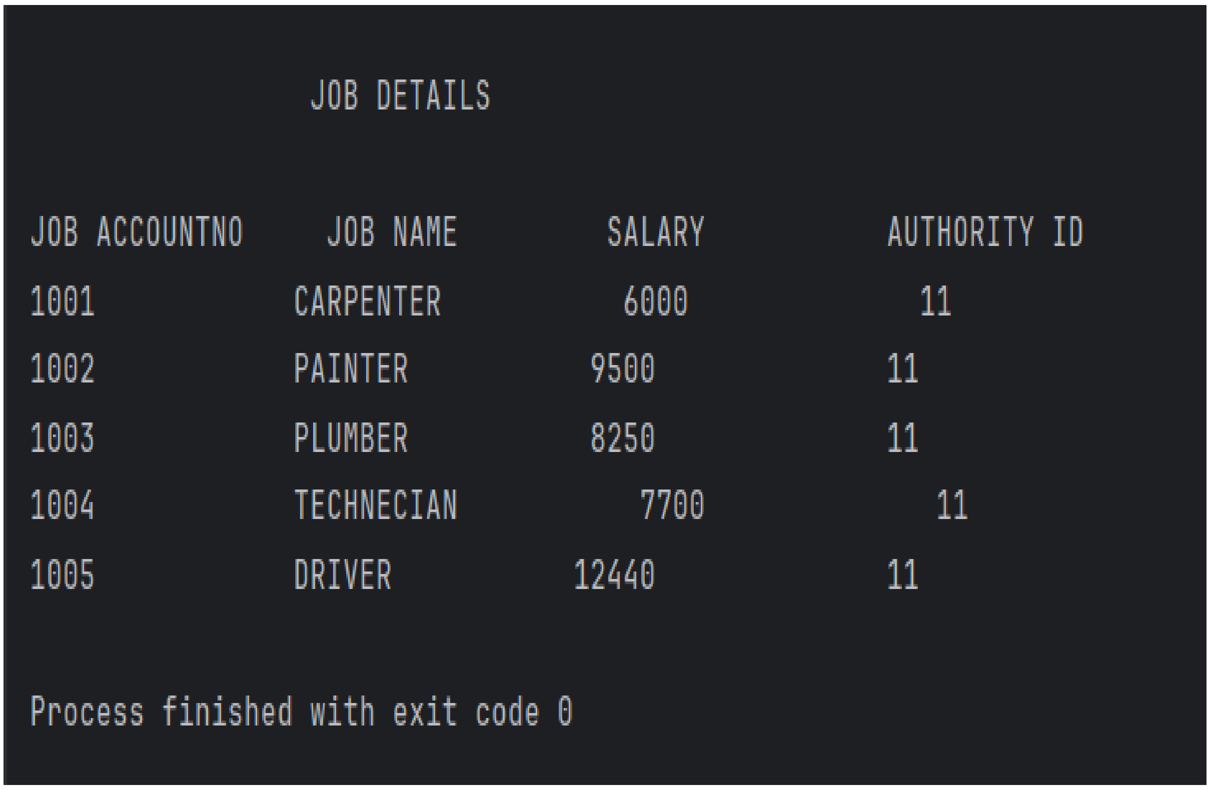
2. **CONNECTING TO THE DATABASE:** Established a connection to the MySQL database which form a secure pathway.

3. **CREATING A STATEMENT:** After securing the connection, created a statement object to execute my SQL queries on the database.

4. **EXECUTING QUERIES IN THE RESULTSET():** Executed queries by using the statement object in Resultset, which contains the data from database.

5. **CLOSING THE CONNECTION**: To prevent unnecessary resource usages , closed the connection for proper resource management.

**The output is:**



```
                 JOB DETAILS

JOB ACCOUNTNO     JOB NAME         SALARY           AUTHORITY ID
1001            CARPENTER          6000               11
1002            PAINTER          9500             11
1003            PLUMBER          8250             11
1004            TECHNECIAN           7700               11
1005            DRIVER          12440             11

Process finished with exit code 0
```

Fig 9.2: CONSOLE OUTPUT OF TABLE STORE.

## 10.CONCLUSION:

The job management system project defines a well-thought-out database management system which ensure efficiency of storing data. In the process of creating database, visual representation is provided through an ER diagram, relationships between entities have been established. Future applications may explore integrating the system with mobile platforms, enabling real-time updates and remote access for users.

The project can easily expand into domains like project management or automating tasks as it's adaptable. This report has highlighted implementation of  all requirements of users, optimized data normalization to lessen redundancy and enhance data integrity.