# DriveBase

SUBMITTED BY

Rama Krishna Raju Rudraraju – 50153214

INSTRUCTOR

Dr. MUTLU METE

**STEP 1:**

## One-to-One Binary Relationship:

- CUSTOMER → LICENSE
  A customer holds one license and a license belongs to one customer.

## One-to-One Unary Relationship:

- VEHICLE → VEHICLE
  A vehicle can be a backup of another vehicle of same kind.

## One-to-Many Unary Relationship:

- EMPLOYEE → EMPLOYEE
  An employee can supervise many employees but an employee is under the supervision of one employee.

## One-to-Many Binary Relationship:

- STATE → COURSE
  One state can have many courses but one course belongs to one state.
  A state can have many courses

- STATE → CUSTOMER
  One state can have many customers but one customer belongs to one state.

- STATE → LICENSE
  One state can provide many licenses but a license can be provided by one state.

- ROOM → COURSE
  One room can held many course but a course can be held in one room.

- EMPLOYEE → COURSE
  An employee can instruct many courses but a course can be thought but one Employee.

- BRANCH → EMPLOYEE
  One branch has many employees but an employee can work in one branch.

- EMPLOYEE → VEHICLE
  One employee manages many vehicles but a vehicle is managed by one employee.

## Many-to-many Binary relationship:

- CUSTOMER → COURSE
  a customer can enroll in many courses and a course can be chosen by many customers.

- CUSTOMER → VEHICLE
  a customer can practice on many vehicles and a vehicle is used by many customers.

### Intersection Data:

An intersection data can be placed between a many - many binary relation.

CUSTOMER → COURSE is a many – many relation in which the intersecting data between these entities is Progress.

## STEP 2:

### Driving school

This is a project which revolves around driving school. Here we have considered a small case and explained how a driving school operates how its functions.

This project stores the data related to:

- Customer information and their license information.
- Courses offered by driving school.
- Employee Information.
- Branch: Address or location Information.
- Details of Vehicle.
- Progress of customers in their corresponding courses.
- Minimum age requirements in different states.
- Class room information where classes will be held.

This project driving school deals with how driving school operates and how it functions. It offers various courses based on state wise, adult or teen, and based on driving license class for example class C or class B etc. Customer progress in his courses will be recorded. Driving school will maintain employee and vehicle information.

**Description of Each Entity and relations**:-

## STATE

STATE is an entity in which we store the Shortname as primary key which stores the short name of state (ex: for New York it is NY), Name to store the name of the state and MinAgeReq is the minimum age requirement of the state to get the license. It has one-to-many relationship with LICENSE, CUSTOMER and COURSE tables.

## COURSE

COURSE is an entity which stores the details of courses. It has CourseID as primary key which stores ID's of courses, CourseName, StartDate to store the starting date of the course, EndDate to store the ending date of the course, RoomID to store the room number of different courses. It has one-to-many relationship with STATE, CUSTOMER, ROOM and EMPLOYEE tables.

## CUSTOMER

CUSTOMER is an entity which stores the details of customer. It has CustomerID as primary key which stores customer ID's, and remaining attributes Name, DOB, Email, Address, Zip, State to store the corresponding details of the customer. It has one one-to-one binary relationship with License, One-to-many relationship with State, Many-to-Many relationship with COURSE and VEHICLE.

## LICENSE

LICENSE is an entity which stores the details contained in the license. It has LicenseID as primary key which stores the License number and remaining attributes Class, IssueDate, ExpiryDate, Restrictions, Address, State, Sex, and Eyes to store the corresponding details in the license. It has one one-to-one Binary relationship with CUSTOMER and one-to-many Binary relationship with STATE.

## ROOM

ROOM is an entity which stores the details of the class room. It has the RoomID as primary key which stores the room number, and remaining attributes RoomName, Capacity stores the corresponding details of room. It has only one One-to-many Relationship with COURSE.

## EMPLOYEE

EMPLOYEE is an entity which stores the details of the employee who is working in the Driving school. It has Employee ID as primary key which stores the employee number and remaining Attributes Name, DOB (Date of birth), JoiningDate, YearOfExp, and Address stores the corresponding details of the employee. It has three one-to-many Binary relationships with COURSE, VEHICLE, BRANCH and a one-to-many Unary relationships with itself**.**

## BRANCH

BRANCH is an entity which stores the details of the Driving school branch. It has BranchID as primary key which stores the Branch number of the Driving school, and remaining attributes Name, Address, Zip stores the respective information of the branch. It has one One-to-many relationship with EMPLOYEE.

## VEHICLE

VEHICLE is an entity which stores vehicle information. It has VehicleID as primary key which stores the vehicle number, and Name, Model, Manfyear (manufacturing year), Mileage stores the respective information of the vehicle. It has one one-to-one unary relationship with VEHICLE.

## STEP 3:

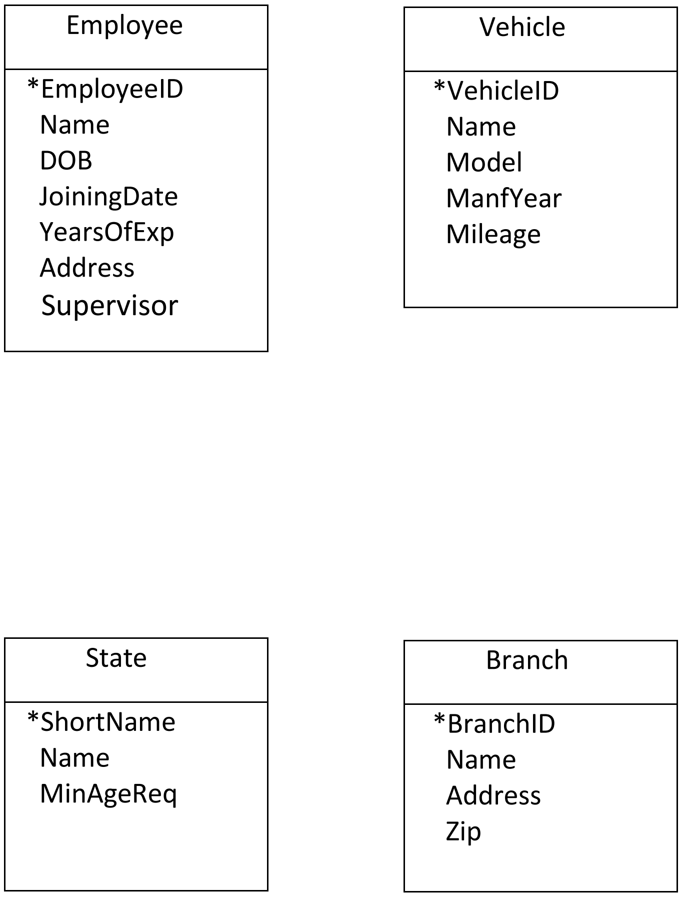
List of all entities along with their unique identifier marked with *

| Customer |
|---|
| *CustomerID |
| Name |
| DOB |
| Email |
| Address |
| Zip |
| State |

| License |
|---|
| *LicenseID |
| Class |
| IssueDate |
| ExpiryDate |
| Restrictions |
| Address |
| State |
| Sex |
| Eyes |

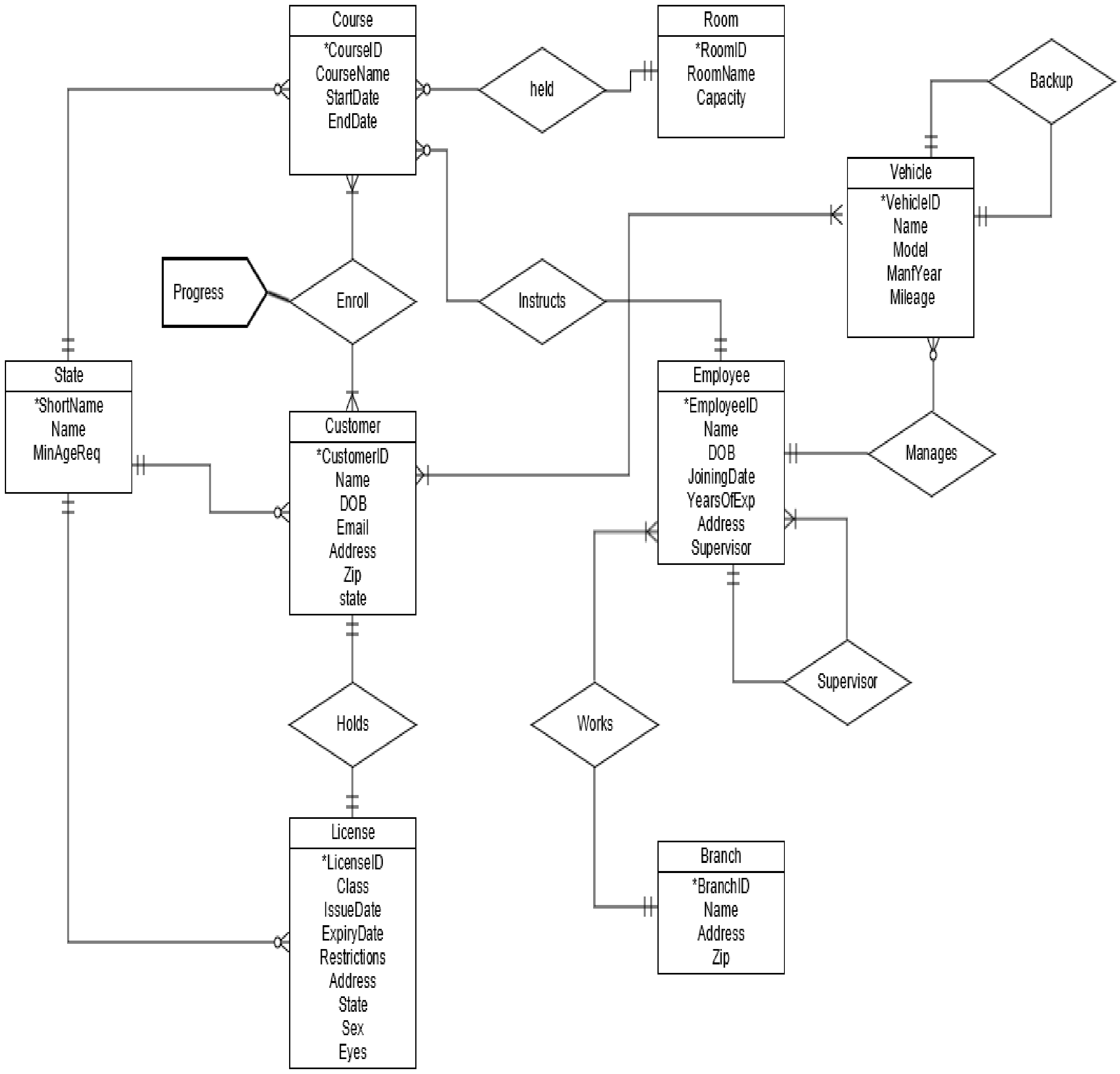
| Course |
|---|
| *CourseID |
| CourseName |
| StartDate |
| EndDate |

| Room |
|---|
| *RoomID |
| RoomName |
| Capacity |


Employee
*EmployeeID
Name
DOB
JoiningDate
YearsOfExp
Address
Supervisor
Vehicle
*VehicleID
Name
Model
ManfYear
Mileage
State
*ShortName
Name
MinAgeReq
Branch
*BranchID
Name
Address
Zip

**STEP 4:**

**ER-Diagram:**



## STEP 5:

Functional Dependencies

1) **Course:**
   CourseID → CourseName, StartDate, EndDate
2) **Customer:**
   CustomerID → Name, DOB, Email, Address, Zip, State
3) **License:**
   LicenseID → Class, IssueDate, ExpiryDate, Restrictions, Address, State, Sex, Eyes
4) **Room:**
   RoomID → RoomName, Capacity
5) **Employee:**
   EmployeeID → Name, DOB, JoningDate, YearsOfExp, Address, Supervisor
6) **State:**
   ShortName → Name, MinAgeReq
7) **Branch:**
   BranchID → Name, Address, Zip
8) **Vehicle:**
   VehicleID → Name, Model, ManfYear, Mileage

## STEP 6:

Tables from ER Diagram, before Normalization:

<u>Customer One – One License</u>

| <u>CustomerID</u> | Name | DOB | Email | Address | Zip | LicenseID | Class | IssueDate | ExpiryDate | Restriction | State | Sex | Eyes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | |

(Continuation for above table is at below)

<u>Customer Many – Many Course</u>

| CustomerID | Name | DOB | Email | Address | Zip | State | CourseID | CourseName | StartDate |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

(Continuation for above table is at below)

| EndDate | Progress |
|---|---|
| | |

Customer One – Many State

| CustomerID | Name | DOB | Email | Address | Zip | State | ShortName | Name | MinAgeReq |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

Customer Many – Many Vehicle

| CustomerID | Name | DOB | Email | Address | Zip | State | VehicleID | Name | Model | ManfYear | Mileag |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |

Course One – Many Room

| CourseID | CourseName | StartDate | EndDate | RoomID | RoomName | Capacity |
|---|---|---|---|---|---|---|
| | | | | | | |

Course One – Many Employee

| CourseID | CourseName | Start Date | End Date | EmployeeID | Name | DOB | Joining Date | YearsOf Exp | Address | Supervi |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

Course One – Many State

| <u>CourseID</u> | CourseName | StartDate | EndDate | ShortName | Name | MinAgeReq |
|---|---|---|---|---|---|---|
| | | | | | | |

License One – Many State

| <u>LicenseID</u> | Class | IssueDate | ExpiryDate | Restrictions | Address | State | Sex | Eyes |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

(Continuation for above table is at below)

| ShortName | Name | MinAgeReq |
|---|---|---|
| | | |

Employee One – many Vehicle

| <u>EmployeeID</u> | Name | DOB | Joining Date | YearsOf Exp | Address | Supervisor | VehicleID | Name | Model | ManfY ar |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

Employee One – Many Branch

| <u>EmployeeID</u> | Name | DOB | Joining Date | YearsOf Exp | Address | Supervisor | BranchID | Name | Address | Zip |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

Employee one – many Unary

| EmployeeID | Name | DOB | JoiningDate | YearsOfExp | Address | Supervisor |
|---|---|---|---|---|---|---|
| | | | | | | |

Vehicle one – one Unary

| VehicleID | Name | Model | ManfYear | Mileage | BackUpOf |
|---|---|---|---|---|---|
| | | | | | |

## STEP 7:

**Normalization**:-
All the tables above are in 0NF and we need to normalize the data.
We consider all the entities and convert them to 3NF so that it obeys 1NF, 2NF. In 3NF there will not be any transitive dependencies and none of the non key has dependency on other non-keys.

## STEP 8:

## Tables after Normalization:

Customer

| CustomerID | Name | DOB | Email | Address | Zip | State | LicenseID |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

License

| LicenseID | Class | IssueDate | ExpiryDate | Restrictions | Address | State | Sex | Eyes |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

Room

| <u>RoomID</u> | RoomName | Capacity |
|---|---|---|
| | | |

State

| <u>ShortName</u> | Name | MinAgeReq |
|---|---|---|
| | | |

Branch

| <u>BranchID</u> | Name | Address | Zip |
|---|---|---|---|
| | | | |

Employee

| <u>EmployeeID</u> | Name | DOB | JoiningDate | YearsOfExp | Address | Supervisor |
|---|---|---|---|---|---|---|
| | | | | | | |

Vehicle

| <u>VehicleID</u> | Name | Model | ManfYear | Mileage | BackUpOf |
|---|---|---|---|---|---|
| | | | | | |

Course

| <u>CourseID</u> | CourseName | StartDate | EndDate | RoomID | Instructor | CourseForState |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |

CustomerCourses

| CustomerID<br>---------------- | CourseID<br>------------ | Progress | VehicleID |
|---|---|---|---|
| | | | |

## STEP 9:

### Cascade and Delete Rule:

In the relation between Customer and Course, if we try to delete a Course then the Customer progress related to courses should be deleted.

## Step 10:

## Table Creation Scripts:

**License:**

```
create table License(

  LicenseID Number(10) NOT NULL Primary Key,

  Class Char Not Null,

  IssueDate Date Not Null,

  ExpiryDate Date Not Null,

  Restrictions varchar2(5) Null,

  Address varchar2(50) Not Null,

  State varchar2(2) Not Null references State(ShortName),
  Sex Char Not Null,

  Eyes varchar2(8) Not Null);
```

**<u>Customer:</u>**

```
Create table Customer(

  CustomerID Number(10) Not Null Primary Key,

  Name Varchar2(32) Not Null,

  DOB Date Not Null,

  Email varchar2(25) Null,

  Address varchar2(50) Not Null,

  Zip Number(5) Not Null,

  State varchar2(2) Not Null references State(ShortName),

  LicenseID Number(10) Not null references License(LicenseID));
```

**<u>Room:</u>**

```
Create table Room(

  RoomID Number Not Null Primary Key,

  RoomName varchar2(25),

  Capacity Number);
```

**<u>State:</u>**

```
Create Table State(

  ShortName varchar2(2) Not Null Primary Key,

  Name varchar2(20) Not Null,
  MinAgeReq Number);
```

**Branch:**

```
Create table Branch(
  BranchID Number Not Null Primary Key,
  Name varchar2(25) Not Null,
  Address varchar2(50) Not Null,
  Zip number(5) Not Null);
```

**Employee:**

```
Create Table Employee(
  EmployeeID Number Not Null Primary Key,
  Name varchar2(32) Not Null,
  DOB date Not Null,
  JoiningDate Date Not Null,
  YearsOfExp Number default 0,
  Address varchar2(50),
  WorkLocation Number Not Null references Branch(BranchID),
  Supervisor Number Not Null references Employee(EmployeeID));
```

**Vehicle:**

```
Create table Vehicle(
  VehicleID varchar2(25) Not Null Primary Key,
  Name varchar2(25) Not Null,
  Model varchar2(25) Not Null,
  ManfYear number Null,
  Mileage number Null,
  ManagesBy number Null references Employee(EmployeeID),
  BackUpOf varchar2(25) Null references Vehicle(VehicleID));
```

**Course:**

```
Create table Course(
  CourseID varchar2(10) Not Null Primary Key,
  CourseName varchar2(25) Not Null,
  StartDate Date Not Null,
  EndDate Date Not Null,
  RoomID Number Not Null,
  Instructor Number Not Null,
  CourseForState varchar2(2) Not Null,
  constraint fk_CourseRoom Foreign Key(RoomID) references Room(RoomID),
  constraint fk_Instructor Foreign Key(Instructor) references Employee(EmployeeID),
  constraint fk_CourseForState Foreign Key(CourseForState) references State(ShortName));
```

**CustomerCourses:**

```
Create Table CustomerCourses(
  CustomerID Number Not Null,
  CourseID varchar2(10) Not Null,
  Progress Number default 0,
  VehicleID varchar2(25) Null,
  constraint pk_CC Primary Key(CustomerID,CourseID),
  constraint fk_CustDel Foreign Key(CustomerID) references Customer(CustomerID) on delete cascade,
  constraint pk_CourseDel Foreign Key(CourseID) references Course(CourseID) on delete cascade);
```

**STEP 11:**

## License:

```
insert into License(LicenseID,Class,IssueDate,ExpiryDate,Restrictions,Address,State,Sex,Eyes) values(98734567,'C',to_date('10-27-2015','MM-DD-YYYY'),to_date('03-01-2018','MM-DD-YYYY'),'B','2231 Live Oak Street','TX','M','BLK');

insert into License(LicenseID,Class,IssueDate,ExpiryDate,Restrictions,Address,State,Sex,Eyes) values(98734568,'C',to_date('10-27-2014','MM-DD-YYYY'),to_date('03-01-2017','MM-DD-YYYY'),'B','2231 Live Oak Street','CA','M','BLK');

insert into License(LicenseID,Class,IssueDate,ExpiryDate,Restrictions,Address,State,Sex,Eyes) values(98734569,'B',to_date('10-27-2015','MM-DD-YYYY'),to_date('03-01-2018','MM-DD-YYYY'),'B','2231 Live Oak Street','TX','F','BLK');

insert into License(LicenseID,Class,IssueDate,ExpiryDate,Restrictions,Address,State,Sex,Eyes) values(98734570,'B',to_date('10-27-2013','MM-DD-YYYY'),to_date('03-01-2016','MM-DD-YYYY'),'A','2231 Live Oak Street','AZ','M','BLK');

insert into License(LicenseID,Class,IssueDate,ExpiryDate,Restrictions,Address,State,Sex,Eyes) values(98734571,'A',to_date('10-27-2014','MM-DD-YYYY'),to_date('03-01-2017','MM-DD-YYYY'),'A','2231 Live Oak Street','NY','F','BLK');
```

## Customer:

```
Insert into Customer(CustomerID,Name,DOB,Email,Address,ZIP,State,LicenseID) values(1001, 'Brahmananda Reddy A',to_date('11-25-1991','MM-DD-YYYY'),'brahma@gmail.com','2231 Live Oak Street',75428,'TX',98734567);
```

Insert into Customer(CustomerID,Name,DOB,Email,Address,ZIP,State,LicenseID) values(1002, 'Hari',to_date('10-23-1992','MM-DD-YYYY'),'hari@gmail.com','2231 Live Oak Street',75429,'CA',98734568);

Insert into Customer(CustomerID,Name,DOB,Email,Address,ZIP,State,LicenseID) values(1003, 'John',to_date('01-14-1988','MM-DD-YYYY'),'john@gmail.com','2231 Live Oak Street',75634,'TX',98734569);

Insert into Customer(CustomerID,Name,DOB,Email,Address,ZIP,State,LicenseID) values(1004, 'Venkata',to_date('03-07-1979','MM-DD-YYYY'),'venkata@gmail.com','2231 Live Oak Street',65724,'AZ',98734570);

Insert into Customer(CustomerID,Name,DOB,Email,Address,ZIP,State,LicenseID) values(1005,'Chaitanya P',to_date('10-18-1990','MM-DD-YYYY'),'chaitu@gmail.com','2231 Live Oak Street',54321,'NY',98734571);

## Room:

insert into Room(RoomID,RoomName,Capacity) values(101,'Ray Burn Conf Room',60);

insert into Room(RoomID,RoomName,Capacity) values(102,'Room 2',60);

insert into Room(RoomID,RoomName,Capacity) values(103,'Room 3',40);

insert into Room(RoomID,RoomName,Capacity) values(104,'Room 4',80);

insert into Room(RoomID,RoomName,Capacity) values(105,'Room 5',30);

## State:

Insert into State(ShortName,Name,MinAgeReq) values('TX','Texas',18);

Insert into State(ShortName,Name,MinAgeReq) values('NY','New York',18);

Insert into State(ShortName,Name,MinAgeReq) values('AZ','Arizona',18);

Insert into State(ShortName,Name,MinAgeReq) values('CA','California',18);

Insert into State(ShortName,Name,MinAgeReq) values('NJ','New Jersey',18);

## Branch:

Insert into Branch(BranchID,Name,Address,Zip) values(101,'Commerce','2231 Live Oak',75428);

Insert into Branch(BranchID,Name,Address,Zip) values(102,'California','2231 Live Oak',75429);

Insert into Branch(BranchID,Name,Address,Zip) values(103,'Dallas','2231 Live Oak',75634);

Insert into Branch(BranchID,Name,Address,Zip) values(104,'Arizona','2231 Live Oak',65724);

Insert into Branch(BranchID,Name,Address,Zip) values(105,'New York','2231 Live Oak',54321);

## Employee:

Insert into Employee(EmployeeID,Name,DOB,JoiningDate,YearsOfExp,Address,Supervisor) values(55549,'Govardhan Reddy A',to_date('11-25-1991','MM-DD-YYYY'),SYSDATE,2.4,'2231 Live Oak',55549);

Insert into Employee(EmployeeID,Name,DOB,JoiningDate,YearsOfExp,Address,Supervisor) values(55550,'Steve',to_date('11-25-1991','MM-DD-YYYY'),SYSDATE,1,'2231 Live Oak',55549);

Insert into Employee(EmployeeID,Name,DOB,JoiningDate,YearsOfExp,Address,Supervisor) values(55551,'Marc',to_date('11-25-1991','MM-DD-YYYY'),SYSDATE,1.5,'2231 Live Oak',55549);

Insert into Employee(EmployeeID,Name,DOB,JoiningDate,YearsOfExp,Address,Supervisor) values(55552,'Larry',to_date('11-25-1991','MM-DD-YYYY'),SYSDATE,0.4,'2231 Live Oak',55549);

Insert into Employee(EmployeeID,Name,DOB,JoiningDate,YearsOfExp,Address,Supervisor) values(55553,'Satya',to_date('11-25-1991','MM-DD-YYYY'),SYSDATE,1.9,'2231 Live Oak',55549);

## Vehicle:

insert into Vehicle(VehicleID,Name,Model,ManfYear,Mileage,Managesby) values('SE12345', 'Toyota','Camry',1999,22,55549);

insert into Vehicle(VehicleID,Name,Model,ManfYear,Mileage,Managesby,BackUpOf) values('SE65748','Toyota','Corolla',2000,22,55550,'SE12345');

```
insert into Vehicle(VehicleID,Name,Model,ManfYear,Mileage,Managesby) values('SE78645',
'Lexus','SUV',2013,22,55551);

insert into Vehicle(VehicleID,Name,Model,ManfYear,Mileage,Managesby) values('SE19876',
'Acura','mdx',2012,22,55552);

insert into Vehicle(VehicleID,Name,Model,ManfYear,Mileage,Managesby) values('SE26409',
'Tata','Nano',2012,22,55553);
```

## <u>Course:</u>

```
insert into
Course(CourseID,CourseName,StartDate,EndDate,RoomID,Instructor,CourseForState)
values('TX123','Car Course For Texas',to_date('05-20-2015','MM-DD-YYYY'),to_date('06-20-
2015','MM-DD-YYYY'),101,55549,'TX');

insert into
Course(CourseID,CourseName,StartDate,EndDate,RoomID,Instructor,CourseForState)
values('CA123','Car Course For California',to_date('05-20-2015','MM-DD-YYYY'),to_date('06-20-
2015','MM-DD-YYYY'),102,55550,'CA');

insert into
Course(CourseID,CourseName,StartDate,EndDate,RoomID,Instructor,CourseForState)
values('TX125','Bike Course For Texas',to_date('05-20-2015','MM-DD-YYYY'),to_date('06-20-
2015','MM-DD-YYYY'),103,55551,'TX');

insert into
Course(CourseID,CourseName,StartDate,EndDate,RoomID,Instructor,CourseForState)
values('AZ123','Bike Course For Arizona',to_date('05-20-2015','MM-DD-YYYY'),to_date('06-20-
2015','MM-DD-YYYY'),104,55552,'TX');

insert into
Course(CourseID,CourseName,StartDate,EndDate,RoomID,Instructor,CourseForState)
values('NY123','Commercial Veh NY',to_date('05-20-2015','MM-DD-YYYY'),to_date('06-20-
2015','MM-DD-YYYY'),105,55553,'NY');
```

## <u>CustomerCourses:</u>

```
insert into CustomerCourses(CustomerID,CourseID,Progress,VehicleID) values(1001,'TX123',0,'');
```

insert into CustomerCourses(CustomerID,CourseID,Progress,VehicleID) values(1002,'CA123' ,20,'SE12345');

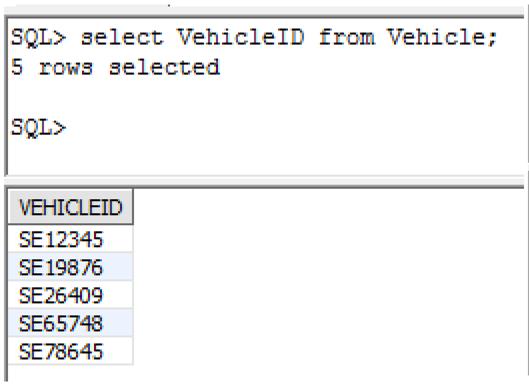
insert into CustomerCourses(CustomerID,CourseID,Progress,VehicleID) values(1003,'TX125' ,25,'SE78645');

insert into CustomerCourses(CustomerID,CourseID,Progress,VehicleID) values(1004,'AZ123' ,40,'SE19876');

insert into CustomerCourses(CustomerID,CourseID,Progress,VehicleID) values(1005,'NY123' ,60,'SE26409');

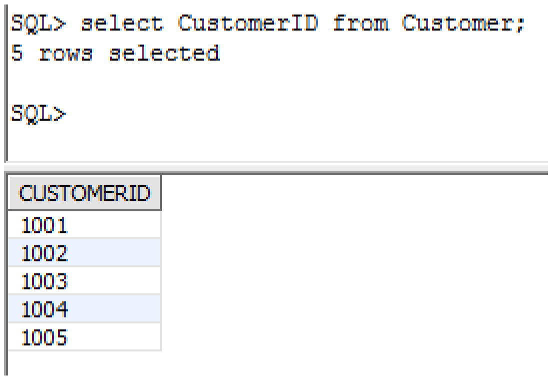
**STEP 12:**

List only primary keys for three different tables.



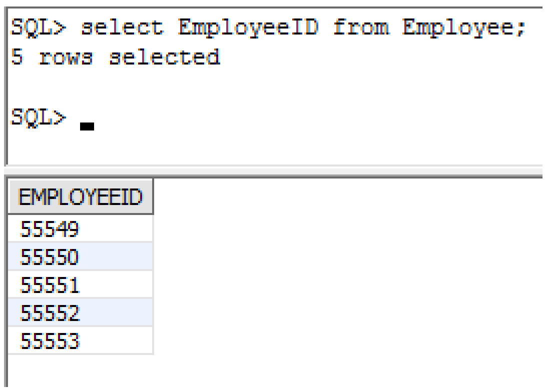
```
SQL> select VehicleID from Vehicle;
5 rows selected

SQL>
```

| VEHICLEID |
|---|
| SE12345 |
| SE19876 |
| SE26409 |
| SE65748 |
| SE78645 |



```
SQL> select CustomerID from Customer;
5 rows selected

SQL>
```

| CUSTOMERID |
|---|
| 1001 |
| 1002 |
| 1003 |
| 1004 |
| 1005 |

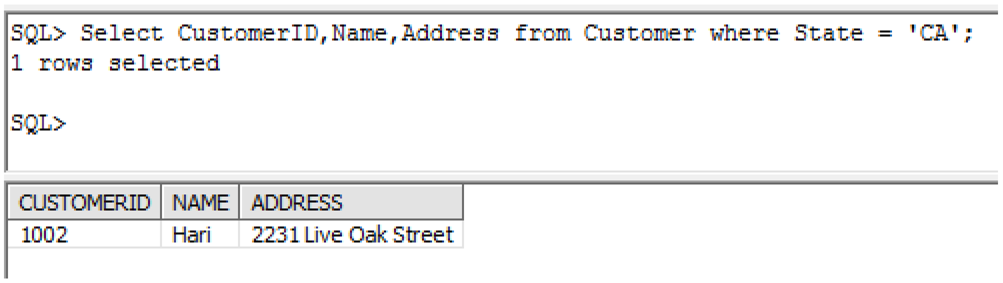


```
SQL> select EmployeeID from Employee;
5 rows selected

SQL>
```

| EMPLOYEEID |
|---|
| 55549 |
| 55550 |
| 55551 |
| 55552 |
| 55553 |

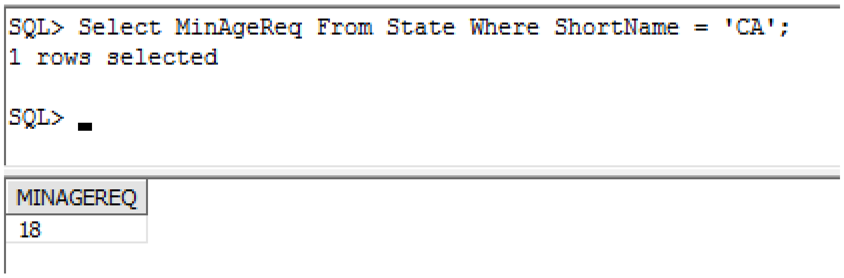
**STEP 13:**

SELECT command with WHERE statement on two different tables.



```
SQL> Select CustomerID,Name,Address from Customer where State = 'CA';
1 rows selected

SQL>
```

| CUSTOMERID | NAME | ADDRESS |
|---|---|---|
| 1002 | Hari | 2231 Live Oak Street |

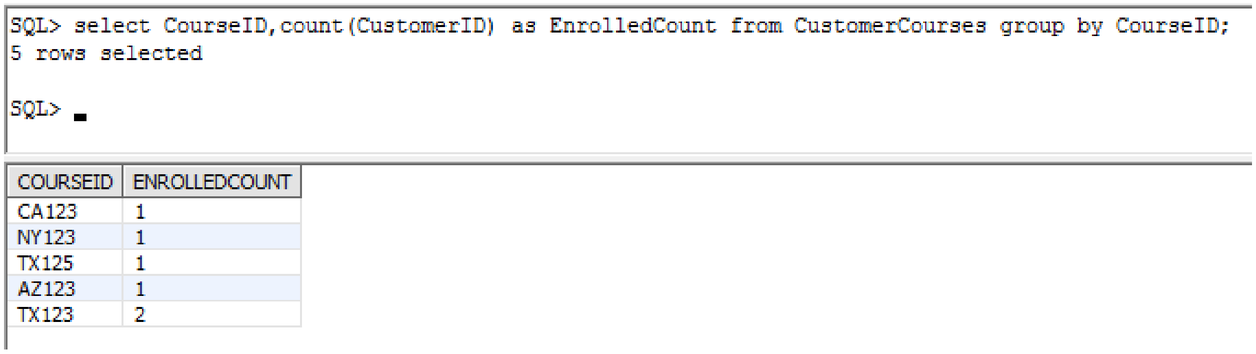


```
SQL> Select MinAgeReq From State Where ShortName = 'CA';
1 rows selected

SQL>
```

| MINAGEREQ |
|---|
| 18 |

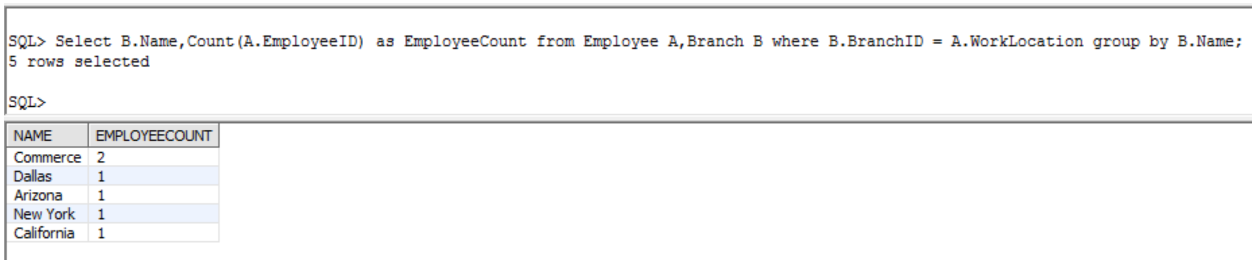
**STEP 14:**

Use of SELECT command with GROUP BY statement on two different tables



```
SQL> select CourseID,count(CustomerID) as EnrolledCount from CustomerCourses group by CourseID;
5 rows selected

SQL>
```

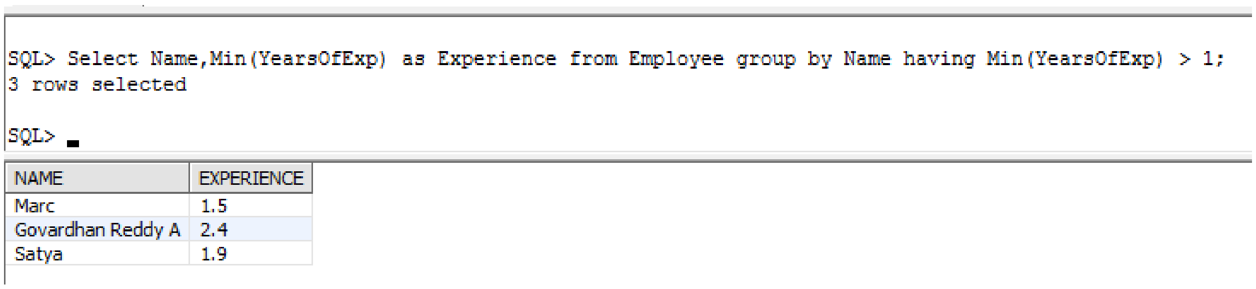
| COURSEID | ENROLLEDCOUNT |
|---|---|
| CA123 | 1 |
| NY123 | 1 |
| TX125 | 1 |
| AZ123 | 1 |
| TX123 | 2 |



```
SQL> Select B.Name,Count(A.EmployeeID) as EmployeeCount from Employee A,Branch B where B.BranchID = A.WorkLocation group by B.Name;
5 rows selected

SQL>
```

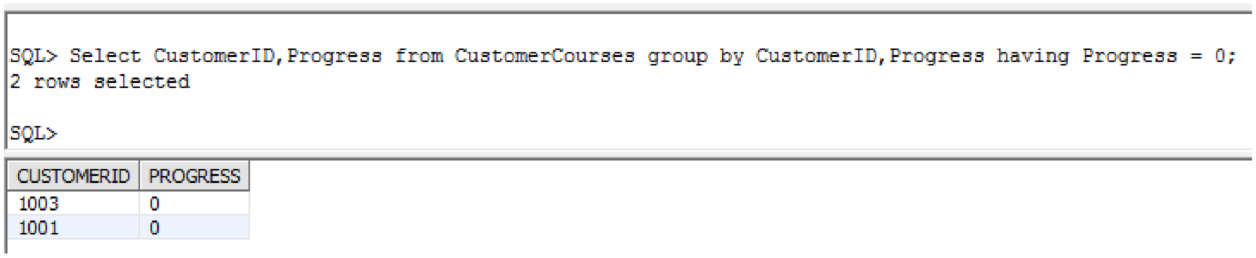
| NAME | EMPLOYEECOUNT |
|---|---|
| Commerce | 2 |
| Dallas | 1 |
| Arizona | 1 |
| New York | 1 |
| California | 1 |

Use of SELECT command with HAVING statement on two different tables



```
SQL> Select Name,Min(YearsOfExp) as Experience from Employee group by Name having Min(YearsOfExp) > 1;
3 rows selected

SQL>
```

| NAME | EXPERIENCE |
|---|---|
| Marc | 1.5 |
| Govardhan Reddy A | 2.4 |
| Satya | 1.9 |

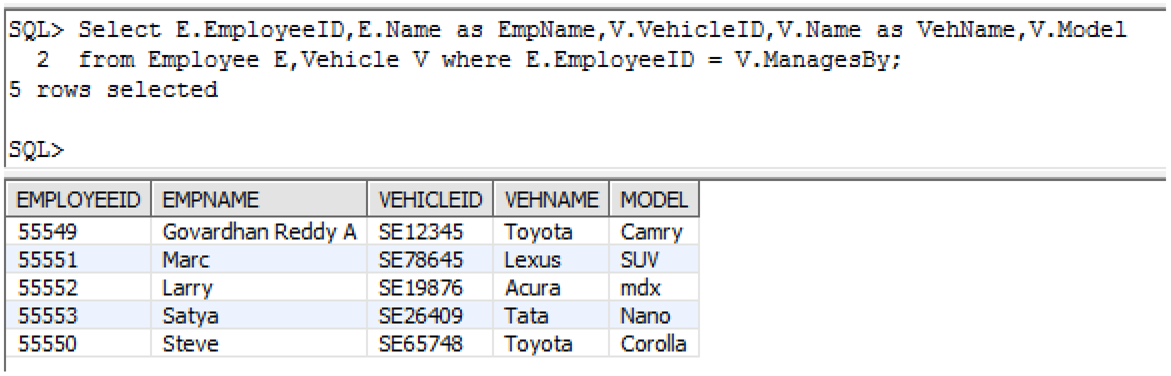


```
SQL> Select CustomerID,Progress from CustomerCourses group by CustomerID,Progress having Progress = 0;
2 rows selected

SQL>
```

| CUSTOMERID | PROGRESS |
|---|---|
| 1003 | 0 |
| 1001 | 0 |

## STEP 15:

Inner Join

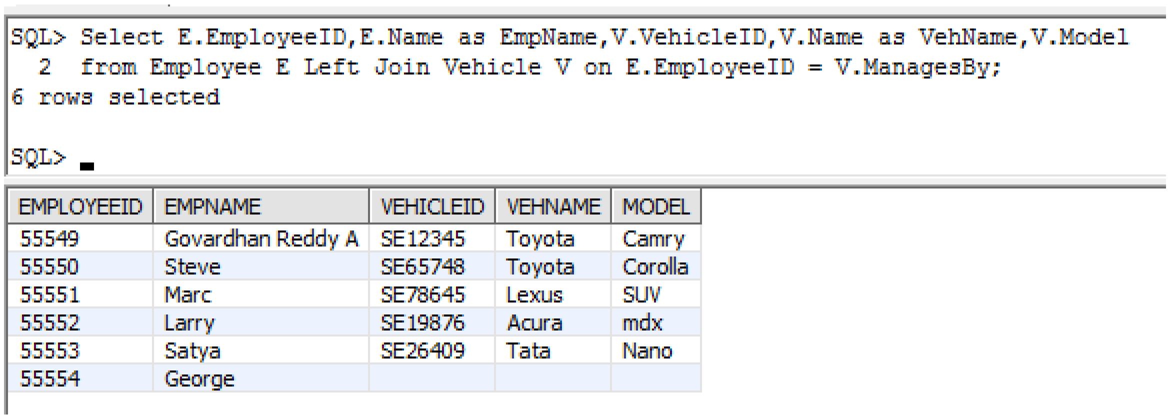


```
SQL> Select E.EmployeeID,E.Name as EmpName,V.VehicleID,V.Name as VehName,V.Model
  2  from Employee E,Vehicle V where E.EmployeeID = V.ManagesBy;
5 rows selected

SQL>
```

| EMPLOYEEID | EMPNAME | VEHICLEID | VEHNAME | MODEL |
|---|---|---|---|---|
| 55549 | Govardhan Reddy A | SE12345 | Toyota | Camry |
| 55551 | Marc | SE78645 | Lexus | SUV |
| 55552 | Larry | SE19876 | Acura | mdx |
| 55553 | Satya | SE26409 | Tata | Nano |
| 55550 | Steve | SE65748 | Toyota | Corolla |

## STEP 16:

Left Outer join

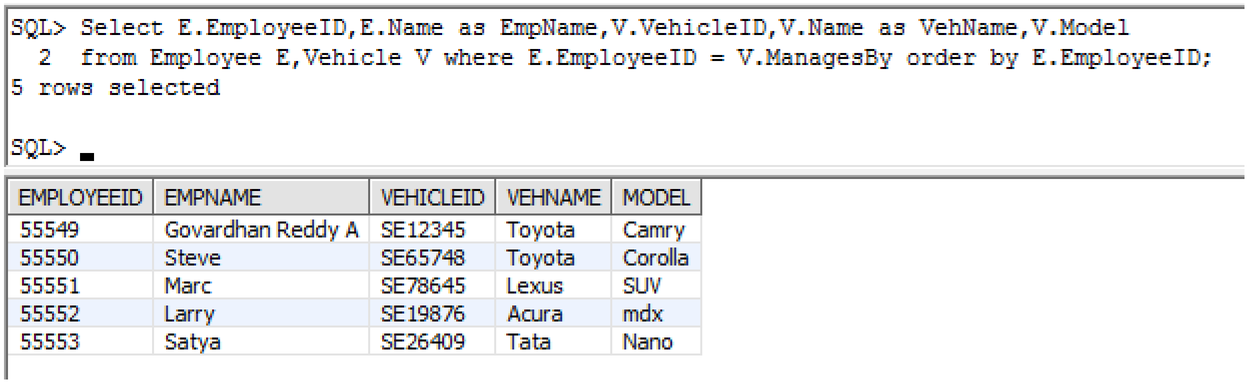


```
SQL> Select E.EmployeeID,E.Name as EmpName,V.VehicleID,V.Name as VehName,V.Model
  2  from Employee E Left Join Vehicle V on E.EmployeeID = V.ManagesBy;
6 rows selected

SQL> _
```

| EMPLOYEEID | EMPNAME | VEHICLEID | VEHNAME | MODEL |
|---|---|---|---|---|
| 55549 | Govardhan Reddy A | SE12345 | Toyota | Camry |
| 55550 | Steve | SE65748 | Toyota | Corolla |
| 55551 | Marc | SE78645 | Lexus | SUV |
| 55552 | Larry | SE19876 | Acura | mdx |
| 55553 | Satya | SE26409 | Tata | Nano |
| 55554 | George | | | |

**STEP 17:**

Use ORDER BY statement to order inner join operation according to foreign key



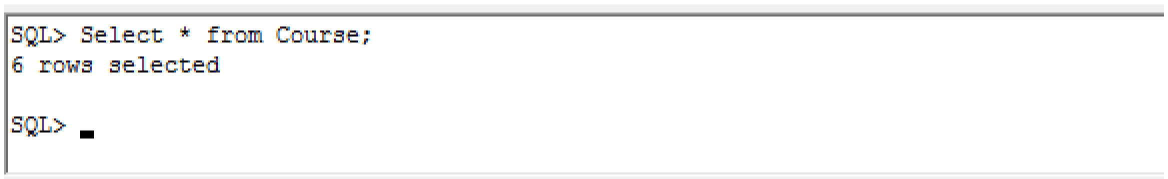
```
SQL> Select E.EmployeeID,E.Name as EmpName,V.VehicleID,V.Name as VehName,V.Model
  2  from Employee E,Vehicle V where E.EmployeeID = V.ManagesBy order by E.EmployeeID;
5 rows selected

SQL> _
```

| EMPLOYEEID | EMPNAME | VEHICLEID | VEHNAME | MODEL |
|---|---|---|---|---|
| 55549 | Govardhan Reddy A | SE12345 | Toyota | Camry |
| 55550 | Steve | SE65748 | Toyota | Corolla |
| 55551 | Marc | SE78645 | Lexus | SUV |
| 55552 | Larry | SE19876 | Acura | mdx |
| 55553 | Satya | SE26409 | Tata | Nano |

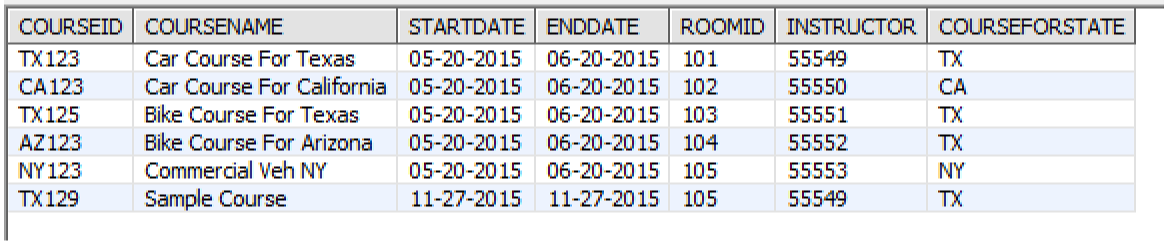
**STEP 18:**

"Cascade delete" operation over two tables.

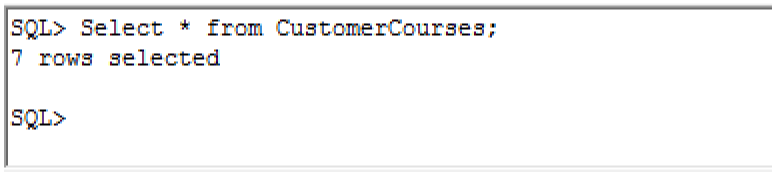
Date in Course and CustomerCourses tables before delete operation.



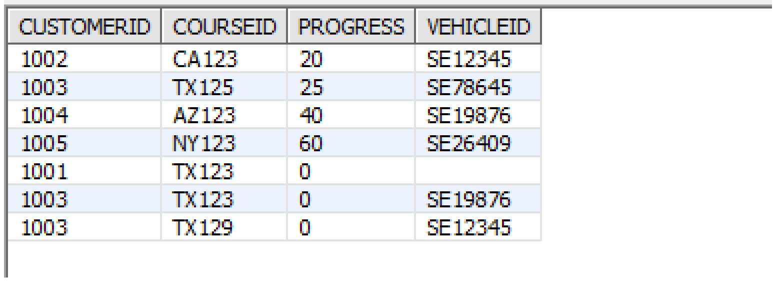
```
SQL> Select * from Course;
6 rows selected

SQL>
```



| COURSEID | COURSENAME | STARTDATE | ENDDATE | ROOMID | INSTRUCTOR | COURSEFORSTATE |
|---|---|---|---|---|---|---|
| TX123 | Car Course For Texas | 05-20-2015 | 06-20-2015 | 101 | 55549 | TX |
| CA123 | Car Course For California | 05-20-2015 | 06-20-2015 | 102 | 55550 | CA |
| TX125 | Bike Course For Texas | 05-20-2015 | 06-20-2015 | 103 | 55551 | TX |
| AZ123 | Bike Course For Arizona | 05-20-2015 | 06-20-2015 | 104 | 55552 | TX |
| NY123 | Commercial Veh NY | 05-20-2015 | 06-20-2015 | 105 | 55553 | NY |
| TX129 | Sample Course | 11-27-2015 | 11-27-2015 | 105 | 55549 | TX |

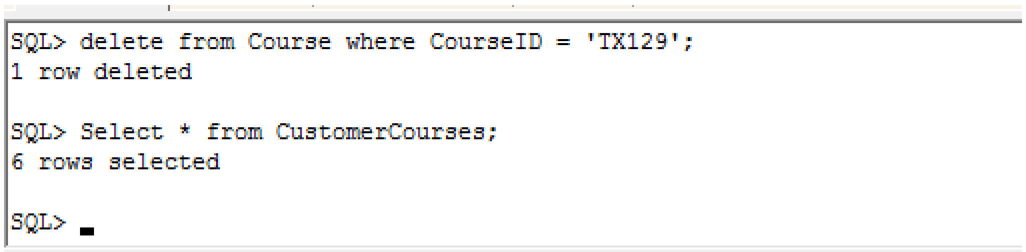


```
SQL> Select * from CustomerCourses;
7 rows selected

SQL>
```

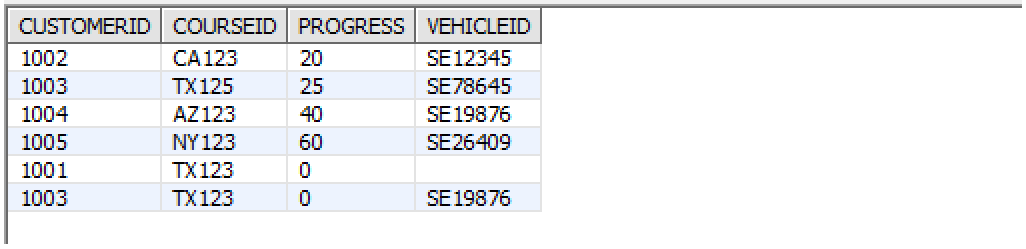


| CUSTOMERID | COURSEID | PROGRESS | VEHICLEID |
|---|---|---|---|
| 1002 | CA123 | 20 | SE12345 |
| 1003 | TX125 | 25 | SE78645 |
| 1004 | AZ123 | 40 | SE19876 |
| 1005 | NY123 | 60 | SE26409 |
| 1001 | TX123 | 0 | |
| 1003 | TX123 | 0 | SE19876 |
| 1003 | TX129 | 0 | SE12345 |

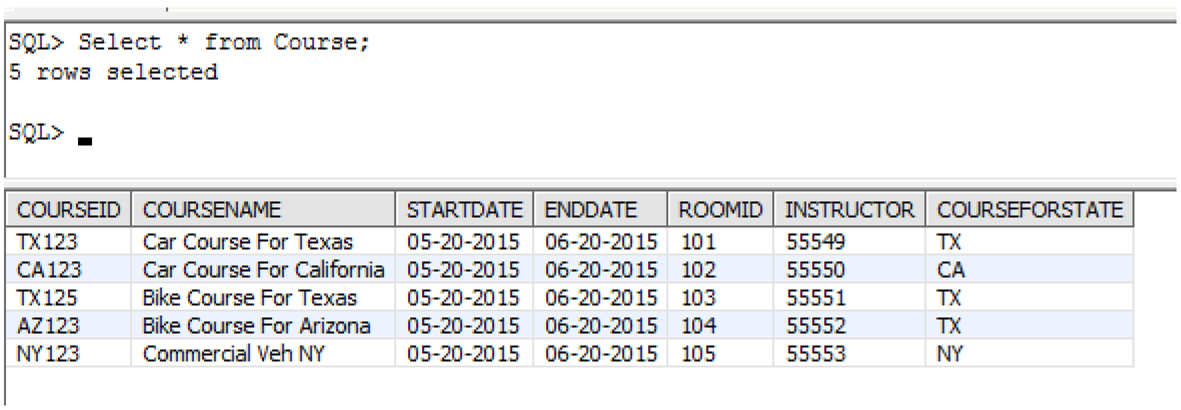
After deleting Course TX129.



```
SQL> delete from Course where CourseID = 'TX129';
1 row deleted

SQL> Select * from CustomerCourses;
6 rows selected

SQL>
```



| CUSTOMERID | COURSEID | PROGRESS | VEHICLEID |
|---|---|---|---|
| 1002 | CA123 | 20 | SE12345 |
| 1003 | TX125 | 25 | SE78645 |
| 1004 | AZ123 | 40 | SE19876 |
| 1005 | NY123 | 60 | SE26409 |
| 1001 | TX123 | 0 | |
| 1003 | TX123 | 0 | SE19876 |

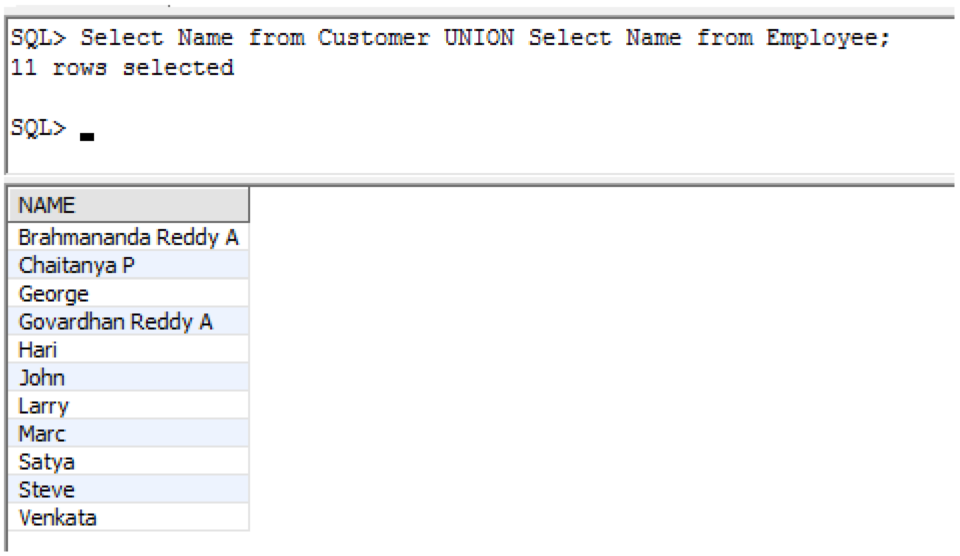


```
SQL> Select * from Course;
5 rows selected

SQL> ▬
```

| COURSEID | COURSENAME | STARTDATE | ENDDATE | ROOMID | INSTRUCTOR | COURSEFORSTATE |
|---|---|---|---|---|---|---|
| TX123 | Car Course For Texas | 05-20-2015 | 06-20-2015 | 101 | 55549 | TX |
| CA123 | Car Course For California | 05-20-2015 | 06-20-2015 | 102 | 55550 | CA |
| TX125 | Bike Course For Texas | 05-20-2015 | 06-20-2015 | 103 | 55551 | TX |
| AZ123 | Bike Course For Arizona | 05-20-2015 | 06-20-2015 | 104 | 55552 | TX |
| NY123 | Commercial Veh NY | 05-20-2015 | 06-20-2015 | 105 | 55553 | NY |

## STEP 19:

Use UNION statement



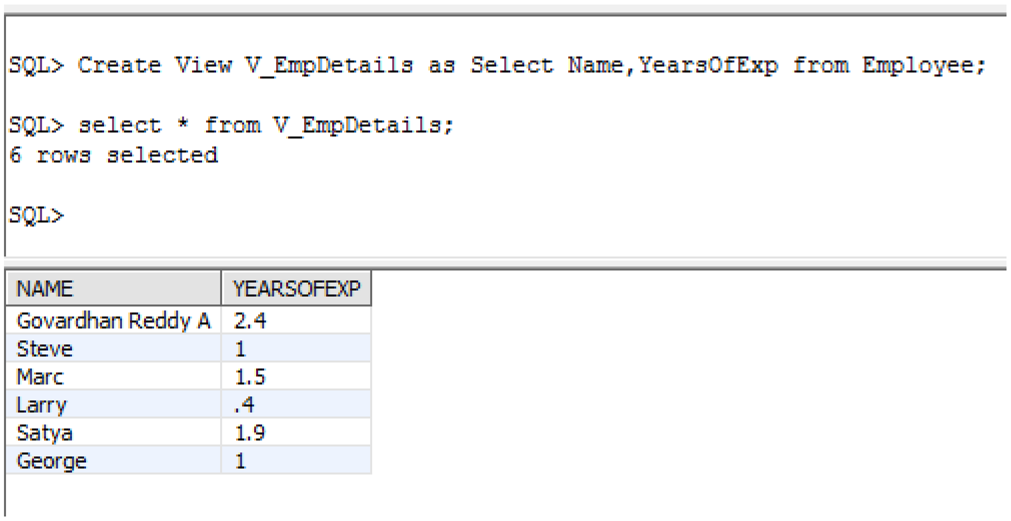
```
SQL> Select Name from Customer UNION Select Name from Employee;
11 rows selected

SQL> ▬
```

| NAME |
|---|
| Brahmananda Reddy A |
| Chaitanya P |
| George |
| Govardhan Reddy A |
| Hari |
| John |
| Larry |
| Marc |
| Satya |
| Steve |
| Venkata |

## STEP 20:

VIEW Statement.

Create View V_EmpDetails as Select  Name,YearsOfExp from Employee;

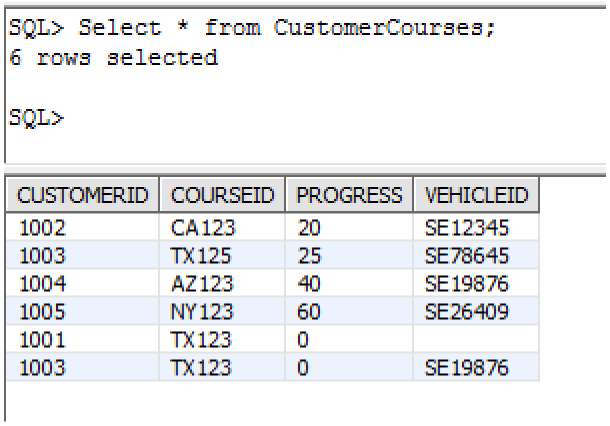

```
SQL> Create View V_EmpDetails as Select Name,YearsOfExp from Employee;

SQL> select * from V_EmpDetails;
6 rows selected

SQL>
```

| NAME | YEARSOFEXP |
|---|---|
| Govardhan Reddy A | 2.4 |
| Steve | 1 |
| Marc | 1.5 |
| Larry | .4 |
| Satya | 1.9 |
| George | 1 |

**STEP 21:**

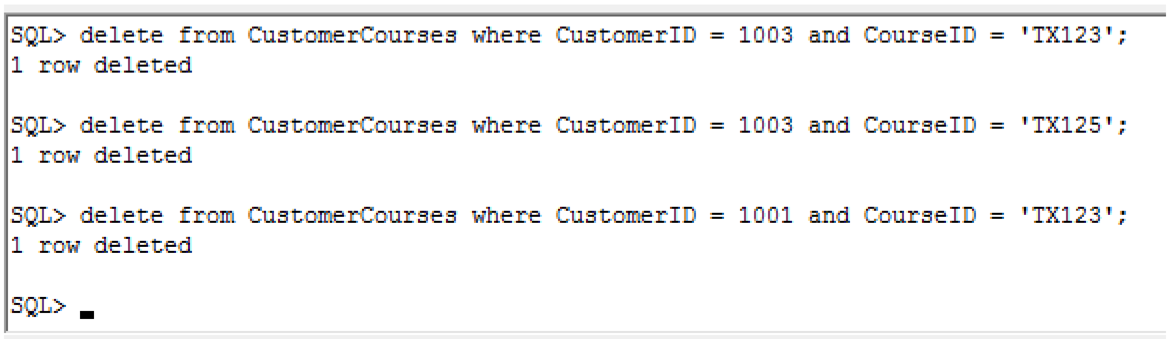
Delete three rows from a table.

Before Deletion

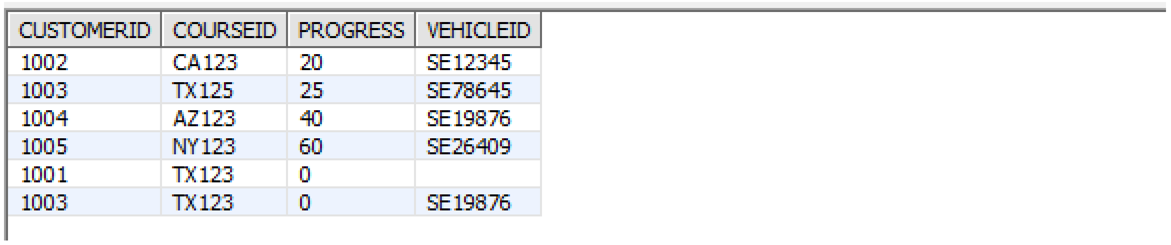

```
SQL> Select * from CustomerCourses;
6 rows selected

SQL>
```

| CUSTOMERID | COURSEID | PROGRESS | VEHICLEID |
|---|---|---|---|
| 1002 | CA123 | 20 | SE12345 |
| 1003 | TX125 | 25 | SE78645 |
| 1004 | AZ123 | 40 | SE19876 |
| 1005 | NY123 | 60 | SE26409 |
| 1001 | TX123 | 0 | |
| 1003 | TX123 | 0 | SE19876 |

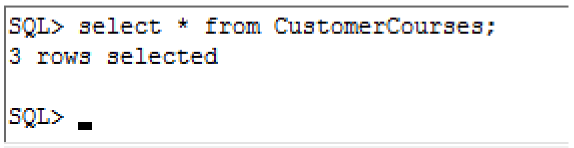
After Deletion



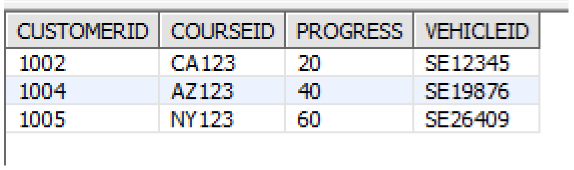
```
SQL> delete from CustomerCourses where CustomerID = 1003 and CourseID = 'TX123';
1 row deleted

SQL> delete from CustomerCourses where CustomerID = 1003 and CourseID = 'TX125';
1 row deleted

SQL> delete from CustomerCourses where CustomerID = 1001 and CourseID = 'TX123';
1 row deleted

SQL> 
```



| CUSTOMERID | COURSEID | PROGRESS | VEHICLEID |
|---|---|---|---|
| 1002 | CA123 | 20 | SE12345 |
| 1003 | TX125 | 25 | SE78645 |
| 1004 | AZ123 | 40 | SE19876 |
| 1005 | NY123 | 60 | SE26409 |
| 1001 | TX123 | 0 | |
| 1003 | TX123 | 0 | SE19876 |



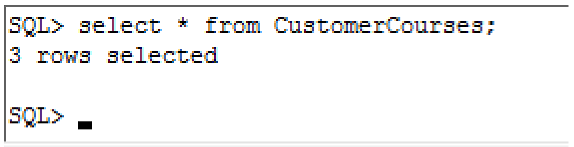
```
SQL> select * from CustomerCourses;
3 rows selected

SQL> 
```



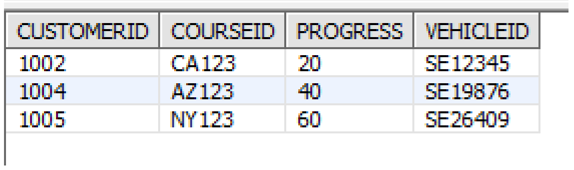
| CUSTOMERID | COURSEID | PROGRESS | VEHICLEID |
|---|---|---|---|
| 1002 | CA123 | 20 | SE12345 |
| 1004 | AZ123 | 40 | SE19876 |
| 1005 | NY123 | 60 | SE26409 |

## STEP 22:

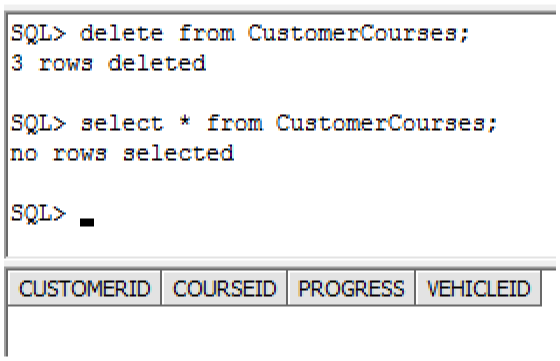
Delete all rows from a table and then delete empty table from database



```
SQL> select * from CustomerCourses;
3 rows selected

SQL> 
```

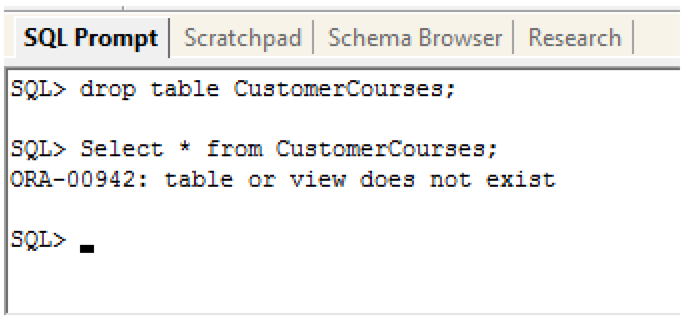


| CUSTOMERID | COURSEID | PROGRESS | VEHICLEID |
|---|---|---|---|
| 1002 | CA123 | 20 | SE12345 |
| 1004 | AZ123 | 40 | SE19876 |
| 1005 | NY123 | 60 | SE26409 |



```
SQL> delete from CustomerCourses;
3 rows deleted

SQL> select * from CustomerCourses;
no rows selected

SQL> ▂
```

| CUSTOMERID | COURSEID | PROGRESS | VEHICLEID |
|---|---|---|---|



```
SQL> drop table CustomerCourses;

SQL> Select * from CustomerCourses;
ORA-00942: table or view does not exist

SQL> ▂
```