FALL 2018

UNIVERSITY OF TEXAS AT DALLAS | Campbell Rd, Richardson, TX

DATABSE DESIGN (CS 6360.002) - FINAL PROJECT

PRASHUK AJMERA - PXA172730  
ABHISHEK Halugudde shivappa – AXH180008

DATABASE SYSTEM FOR TAXI SERVICE

Table of Contents

[REQUIREMENTS 2](#_Toc531889253)

[MODELLING OF REQUIREMENTS AS ER-DIAGRAM 3](#_Toc531889254)

[ASSUMPTIONS: 3](#_Toc531889255)

[MAPPING OF ERD IN RELATIONAL SCHEMA 4](#_Toc531889256)

[SQL STATEMENS FOR TABLE CREATION 6](#_Toc531889257)

[SQL STATEMENS FOR FORIGN KEY CREATION 9](#_Toc531889258)

[SQL STATEMENS FOR INSERT COMMANDS 10](#_Toc531889259)

[PL/SQL – PROCEDURES 11](#_Toc531889260)

[Procedure Code block for Book\_Taxi 11](#_Toc531889261)

[Procedure Code block for TRIP\_END 13](#_Toc531889262)

[PL/SQL – TRIGGERS 15](#_Toc531889263)

[Procedure Code block for Update\_Driver\_Rating 15](#_Toc531889264)

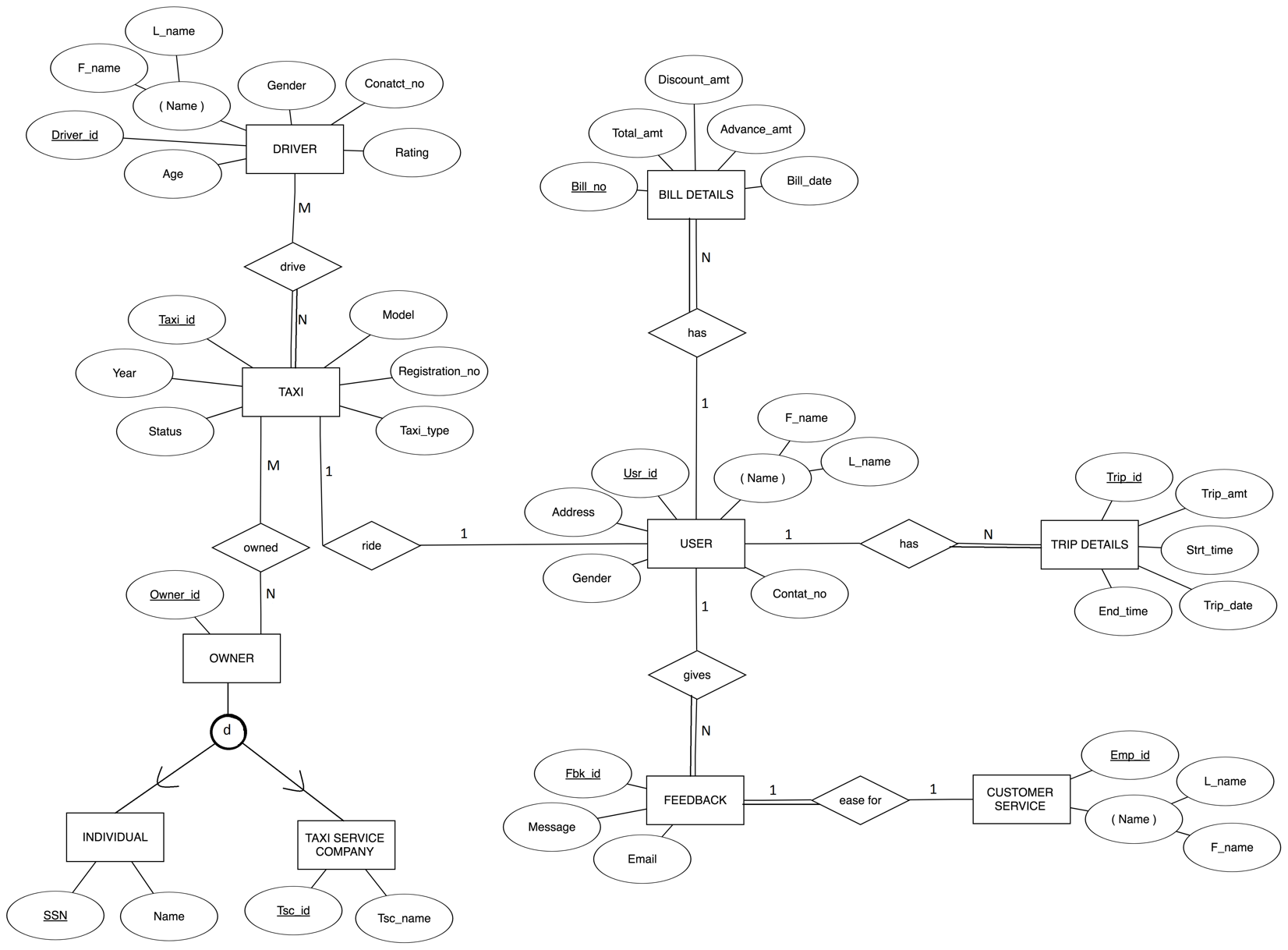
[Procedure Code block for Add\_no\_of\_cars 16](#_Toc531889265)

[NORMALIZATION OF RELATIONAL SCHEMA 17](#_Toc531889266)

# REQUIREMENTS

* The Taxi Service Database involves around three main entities Taxi, User and Trip.
* Taxi can be booked for a specific location with a specific address by a User. User has a unique User\_id, a Contact\_no and an Email.
* A Taxi Service has a number of taxis for service. Each taxi is described by Taxi\_id, Registration\_no, Model, Manufactured year and Status.
* Taxi has a parameter Taxi\_type. It can be ‘Economy’, ‘Standard’, ’SUV’, ‘Premium’ and ‘Minivan’. Taxi\_type defines the price per hour.
* A User can reserve a taxi for a number of hours/days. He can use any valid promotional code.
* A user is uniquely identified by his/her User\_id. User information consists of his name as first name, last name, address, age and contact number.
* When a user books a taxi and starts the trip by the driver the start time automatically updated by the system.
* When the trip ends, the end trip time also automatically updated in the database by the system.
* A unique bill is generated with a Bill\_no after a trip ends which has the information of user, driver, amount, date.
* The total amount and net amount are calculated based on start time, end time, taxi price per hour and promotional code if any.
* A taxi is categorized as Individual Owner and Taxi Service Company. Every taxi has a owner and he/she can give his/her car for the taxi service. Every owner has SSN and name. For the taxi service company information like tcs\_id and tsc\_name will also be there.
* A registered user will be provided with a login id and password. A customer can save his credit/debit card details for future payment.
* Partial payment can also be made at the time of booking and the balance must be paid by the user at the end of the trip.
* If user is a customer, he/she can pay through saved debit/credit card details
* A taxi can be drive by a driver. Driver has uniquely identified by the Driver\_id. Other information consists of name, gender, contact\_no, rating and age.
* After the trip over a unique trip\_id is generated for that particular trip. Along with all the necessary trip\_details such as amount, date etc.
* Users can also the give the feedback/rating for the trip they traveled into it. The feedback can be a message or rating out five for the driver who is giving trip to that user.
* Feedback can be taking by the customer service center representative. They have the information like emp\_id, name and email.

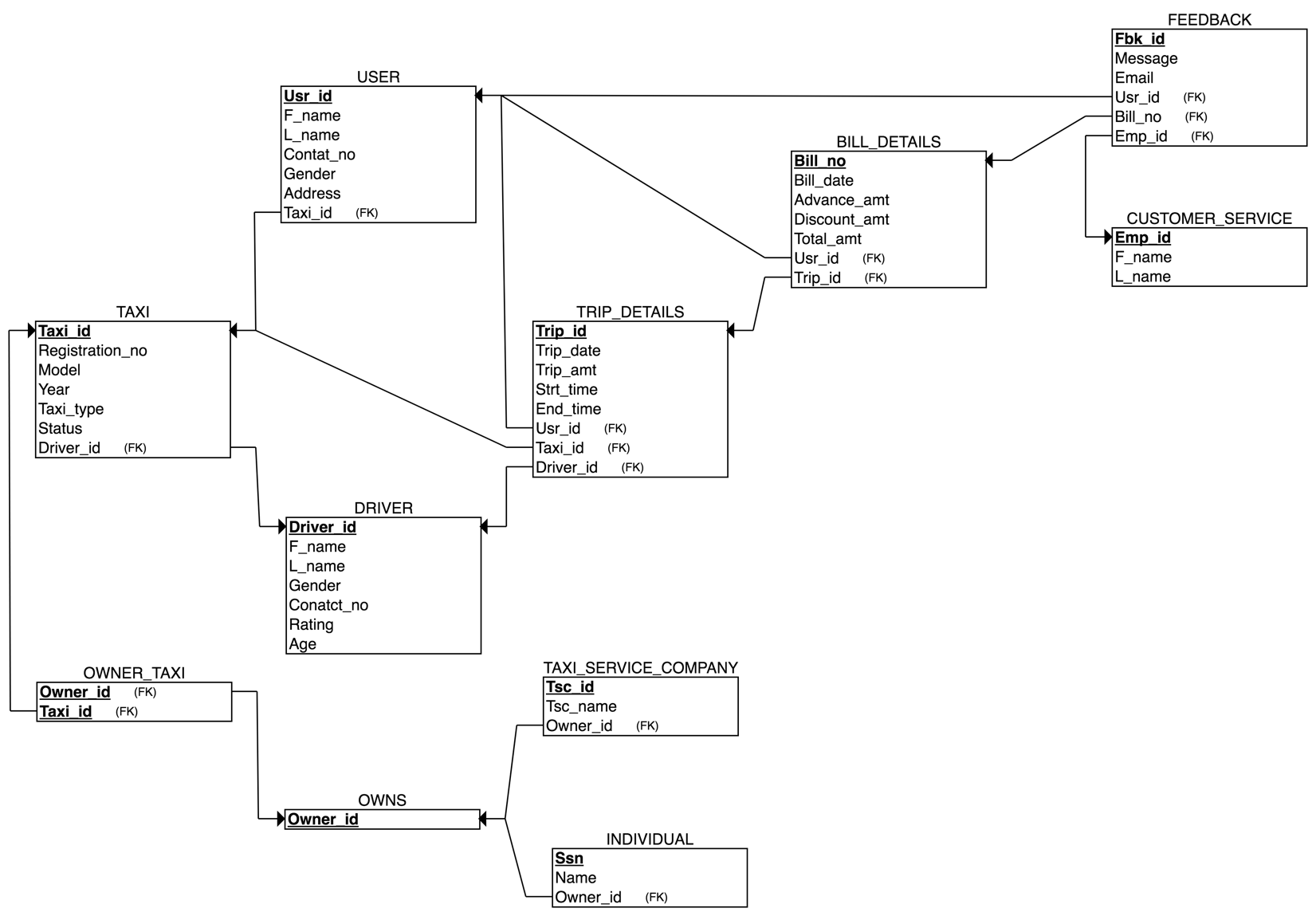
# MODELLING OF REQUIREMENTS AS ER-DIAGRAM



## ASSUMPTIONS:

* Many drivers can drive many taxis (M:N)
* Many owners can give many taxis at a time (M:N)
* One customer service representative can take one feedback at a time (1:1)
* Single user can have multiple trips details (1:N)
* Single user can have multiple bills details (1:N)
* Single user can give many feedbacks (1:N)
* Single user can ride in one taxi at a time (1:1)

# MAPPING OF ERD IN RELATIONAL SCHEMA



**TAXI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Taxi\_id | Registration\_no | Taxi\_Model | Taxi\_Year | Taxi\_type | Status | Driver\_id |

* Primary Key: Taxi\_id
* Foreign Keys: Driver\_id

**USER\_TBL**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Usr\_id | F\_name | L\_name | Contat\_no | Gender | Address | Taxi\_id |

* Primary Key: Usr\_id
* Foreign Keys: Taxi\_id

**DRIVER**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Driver\_id | F\_name | L\_name | Gender | Conatct\_no | Rating | Age |

* Primary Key: Driver\_id
* Foreign Keys: NA

**TRIP\_DETAILS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trip\_id | Trip\_date | Trip\_amt | Driver\_id | Usr\_id |

|  |  |  |
| --- | --- | --- |
| Taxi\_id | Strt\_time | End\_time |

* Primary Key: Trip\_id
* Foreign Keys: Taxi\_id, Usr\_id, Driver\_id

**BILL\_DETAILS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bill\_no | Bill\_date | Advance\_amt | Discount\_amt | Total\_amt | Usr\_id | Trip\_id |

* Primary Key: Bill\_no
* Foreign Keys: Usr\_id, Trip\_id

**CUSTOMER\_SERVICE**

|  |  |  |
| --- | --- | --- |
| Emp\_id | F\_name | L\_name |

* Primary Key: Emp\_id
* Foreign Keys: NA

**FEEDBACK**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fbk\_id | Message | Email | Emp\_id | Usr\_id | Trip\_id |

* Primary Key: Fbk\_id
* Foreign Keys: Usr\_id, Emp\_id, Trip\_id

**OWNER\_TAXI**

|  |  |
| --- | --- |
| Owner\_id | Taxi\_id |

* Primary Key: Owner\_id, Taxi\_id
* Foreign Keys: Owner\_id, Taxi\_id

**OWNS**

|  |  |
| --- | --- |
| Owner\_id | No\_Cars |

* Primary Key: Owner\_id
* Foreign Keys: NA

**INDIVIDUAL**

|  |  |  |
| --- | --- | --- |
| Ssn | Name | Owner\_id |

* Primary Key: Ssn
* Foreign Keys: Owner\_id

**TAXI\_SERVICE\_COMPANY**

|  |  |  |
| --- | --- | --- |
| Tsc\_id | Tsc\_name | Owner\_id |

* Primary Key: Tsc\_id
* Foreign Keys: Owner\_id

# SQL STATEMENS FOR TABLE CREATION

----------------------------------------------

-- Table Creation

----------------------------------------------

CREATE TABLE TAXI (

Taxi\_id integer NOT NULL,

Registration\_no VARCHAR(20),

Taxi\_Model VARCHAR(20),

Taxi\_Year DATE,

Taxi\_type VARCHAR(20),

Status VARCHAR(20),

Driver\_id integer,

PRIMARY KEY (Taxi\_id),

UNIQUE (Registration\_no)

);

CREATE TABLE USER\_TBL (

Usr\_id integer NOT NULL,

F\_name VARCHAR(20),

L\_name VARCHAR(20),

Contat\_no integer,

Gender VARCHAR(10),

Address VARCHAR(50),

Taxi\_id integer,

PRIMARY KEY (Usr\_id)

);

CREATE TABLE DRIVER (

Driver\_id integer NOT NULL,

F\_name VARCHAR(10),

L\_name VARCHAR(20),

Gender VARCHAR(10),

Conatct\_no VARCHAR(20),

Rating integer,

Age integer,

PRIMARY KEY (Driver\_id)

);

CREATE TABLE TRIP\_DETAILS (

Trip\_id integer NOT NULL,

Trip\_date DATE,

Trip\_amt decimal(10,2),

Driver\_id integer,

Usr\_id integer,

Taxi\_id integer,

Strt\_time TIMESTAMP,

End\_time TIMESTAMP,

PRIMARY KEY (Trip\_id)

);

CREATE TABLE BILL\_DETAILS (

Bill\_no integer NOT NULL,

Bill\_date DATE,

Advance\_amt decimal(10,2),

Discount\_amt decimal(10,2),

Total\_amt decimal(10,2),

Usr\_id integer,

Trip\_id integer,

PRIMARY KEY (Bill\_no),

);

CREATE TABLE CUSTOMER\_SERVICE (

Emp\_id integer NOT NULL,

F\_name VARCHAR(20),

L\_name VARCHAR(20),

PRIMARY KEY (Emp\_id)

);

CREATE TABLE FEEDBACK (

Fbk\_id integer NOT NULL,

Message VARCHAR(140),

Email VARCHAR(50),

Emp\_id integer,

Usr\_id integer,

Trip\_id integer,

PRIMARY KEY (Fbk\_id),

);

CREATE TABLE OWNS (

Owner\_id integer NOT NULL,

No\_Cars integer,

PRIMARY KEY (Owner\_id)

);

CREATE TABLE OWNER\_TAXI (

Owner\_id integer NOT NULL,

Taxi\_id integer,

PRIMARY KEY (Owner\_id, Taxi\_id)

);

CREATE TABLE INDIVIDUAL (

Ssn integer NOT NULL,

Name VARCHAR(20),

Owner\_id integer,

PRIMARY KEY (Ssn)

);

CREATE TABLE TAXI\_SERVICE\_COMPANY (

Tsc\_id integer NOT NULL,

Tsc\_name VARCHAR(20),

Owner\_id integer,

PRIMARY KEY (Tsc\_id)

);

# SQL STATEMENS FOR FORIGN KEY CREATION

----------------------------------------------

-- Foreign key creation

----------------------------------------------

ALTER TABLE TAXI ADD CONSTRAINT fketadr FOREIGN KEY (Driver\_id) REFERENCES DRIVER(Driver\_id) ON DELETE CASCADE;

ALTER TABLE USER\_TBL ADD CONSTRAINT fkusta FOREIGN KEY (Taxi\_id) REFERENCES TAXI(Taxi\_id) ON DELETE CASCADE;

ALTER TABLE TRIP\_DETAILS ADD CONSTRAINT fktddr FOREIGN KEY (Driver\_id) REFERENCES DRIVER(Driver\_id) ON DELETE CASCADE;

ALTER TABLE TRIP\_DETAILS ADD CONSTRAINT fktdusr FOREIGN KEY (Usr\_id) REFERENCES USER\_TBL(Usr\_id) ON DELETE CASCADE;

ALTER TABLE TRIP\_DETAILS ADD CONSTRAINT fktdtax FOREIGN KEY (Taxi\_id) REFERENCES TAXI(Taxi\_id) ON DELETE CASCADE;

ALTER TABLE BILL\_DETAILS ADD CONSTRAINT fkbdtd FOREIGN KEY (Trip\_id) REFERENCES TRIP\_DETAILS(Trip\_id) ON DELETE CASCADE;

ALTER TABLE BILL\_DETAILS ADD CONSTRAINT fkbdusr FOREIGN KEY (Usr\_id) REFERENCES USER\_TBL(Usr\_id) ON DELETE CASCADE;

ALTER TABLE FEEDBACK ADD CONSTRAINT fkfbemp FOREIGN KEY (Emp\_id) REFERENCES CUSTOMER\_SERVICE(Emp\_id) ON DELETE CASCADE;

ALTER TABLE FEEDBACK ADD CONSTRAINT fkfbtd FOREIGN KEY (Trip\_id) REFERENCES TRIP\_DETAILS(Trip\_id) ON DELETE CASCADE;

ALTER TABLE FEEDBACK ADD CONSTRAINT fkfbusr FOREIGN KEY (Usr\_id) REFERENCES USER\_TBL(Usr\_id) ON DELETE CASCADE;

ALTER TABLE OWNER\_TAXI ADD CONSTRAINT fkeowtax FOREIGN KEY (Taxi\_id) REFERENCES TAXI(Taxi\_id) ON DELETE CASCADE;

ALTER TABLE OWNER\_TAXI ADD CONSTRAINT fkeowowns FOREIGN KEY (Owner\_id) REFERENCES OWNS(Owner\_id) ON DELETE CASCADE;

ALTER TABLE INDIVIDUAL ADD CONSTRAINT fkeinowns FOREIGN KEY (Owner\_id) REFERENCES OWNS(Owner\_id) ON DELETE CASCADE;

ALTER TABLE TAXI\_SERVICE\_COMPANY ADD CONSTRAINT fketscowns FOREIGN KEY (Owner\_id) REFERENCES OWNS(Owner\_id) ON DELETE CASCADE;

# SQL STATEMENS FOR INSERT COMMANDS

----------------------------------------------

-- Insert Commands

----------------------------------------------

INSERT INTO TAXI VALUES(1,'KA-15R-3367','BENZE 300',to\_date('01/01/2017','mm/dd/yyyy'),'SUV','Available',1)

INSERT INTO DRIVER VALUES(1,'Abhi','Gowda','Male','4693805870',5,25);

INSERT INTO USER\_TBL VALUES(1,'USER1','LNAME','123456','Male','MCCAllum','1');

INSERT INTO TRIP\_DETAILS VALUES(1,to\_date('01/01/2017','mm/dd/yyyy'),123,1,1,1,TO\_TIMESTAMP('2017-01-01 06:14:00', 'YYYY-MM-DD HH24:MI:SS'),TO\_TIMESTAMP('2017-01-01 08:14:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO BILL\_DETAILS VALUES(1,to\_date('01/01/2017','mm/dd/yyyy'),1000.10,20.11,null,1,1);

INSERT INTO CUSTOMER\_SERVICE VALUES(1,'abhi','gowda');

INSERT INTO FEEDBACK VALUES(1,'not so good','abhi@gmail.com',1,1,1);

INSERT INTO OWNS VALUES(1,1);

INSERT INTO OWNS VALUES(2,1);

INSERT INTO OWNER\_TAXI (1,1);

INSERT INTO INDIVIDUAL VALUES(123,'abhi owner ind',1);

INSERT INTO TAXI\_SERVICE\_COMPANY VALUES (1,'abhi taxi comp',2);

INSERT INTO INDIVIDUAL values(123,'abhi owner ind',1);

INSERT INTO TAXI\_SERVICE\_COMPANY values (1,'abhi taxi comp',2);

# PL/SQL – PROCEDURES

## Procedure Code block for Book\_Taxi

----------------------------------------------

-- Procedure Creation

-- this procedure creates a use\_table entry and creates the trip and bill\_detail for the trip

-- input parameters : Name , Address, Contact, Taxi\_Model, Gender, Advance

----------------------------------------------

CREATE OR REPLACE PROCEDURE BOOK\_TAXI

( Name IN VARCHAR2,

v\_Address IN VARCHAR2,

v\_Contact IN VARCHAR2,

Taxi\_Model IN VARCHAR2,

v\_Gender IN VARCHAR2,

Advance IN decimal,

)

AS

BEGIN

DECLARE

v\_usr\_id INT :=-1;

v\_Trip\_id INT :=-1;

v\_Bill\_no INT :=-1;

v\_Taxi\_id INT :=-1;

v\_Driver\_id INT :=1;

BEGIN

select MAX(Usr\_id)+1 into v\_usr\_id from USER\_TBL ;

select MAX(Trip\_id)+1 into v\_Trip\_id from TRIP\_DETAILS ;

select MAX(Bill\_no)+1 into v\_Bill\_no from BILL\_DETAILS ;

select taxi\_id, Driver\_id into v\_Taxi\_id,v\_Driver\_id from TAXI where Status = 'Available' and Taxi\_Model = Taxi\_Model;

insert into USER\_TBL values(v\_usr\_id, SUBSTR (Name, 1, INSTR(Name,' ',1)),SUBSTR (Name, INSTR(Name,' ',1)+1,LENGTH(Name)),v\_Contact,v\_Gender,v\_Address,v\_Taxi\_id);

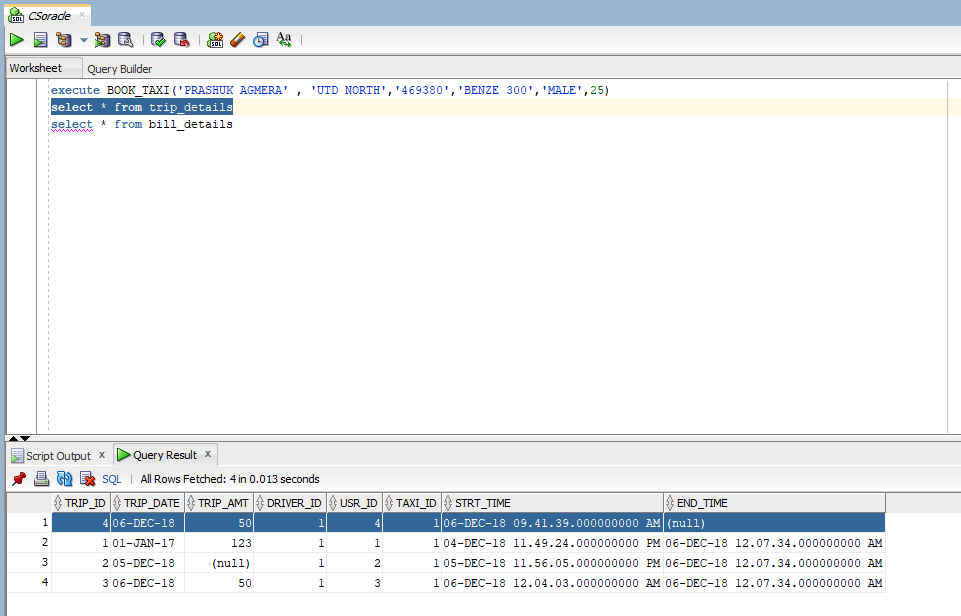
insert into TRIP\_DETAILS values(v\_Trip\_id,sysdate, 50,v\_Driver\_id,v\_usr\_id,v\_Taxi\_id,sysdate,null);

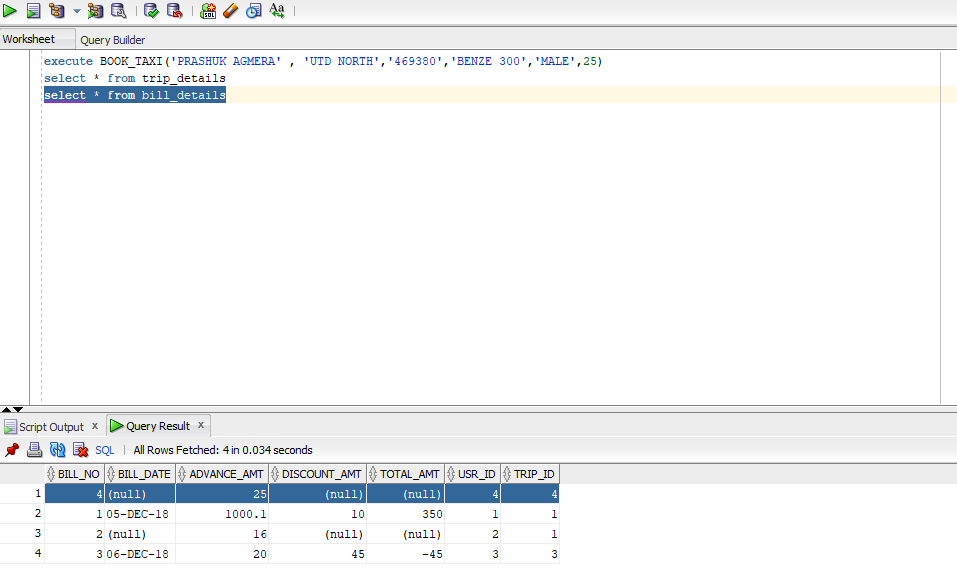
insert into BILL\_DETAILS values(v\_Bill\_no,null,Advance,null,null,v\_usr\_id,v\_Trip\_id);

END ;

END;

/





## Procedure Code block for TRIP\_END

----------------------------------------------

-- Procedure Creation

-- this procedure will calculate the final amount for the trip and update the amount attributes in trip and bill details

-- input parameters :  trip\_id, discount

----------------------------------------------

CREATE OR REPLACE PROCEDURE TRIP\_END(v\_trip IN INT , v\_discount IN Decimal )

AS

BEGIN

DECLARE

v\_total\_time INT := -1;

v\_bill\_no INT :=-1;

BEGIN

select extract(day from (sysdate - Strt\_time))\*24 + extract(hour from (sysdate - Strt\_time)) into v\_total\_time from TRIP\_DETAILS where Trip\_id = v\_trip;

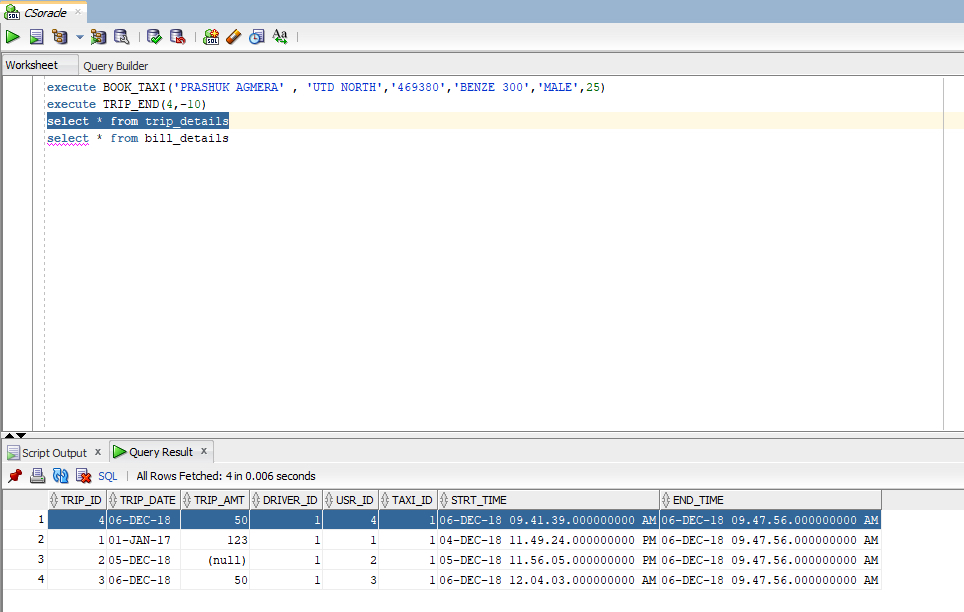
update TRIP\_DETAILS set End\_time = sysdate where Trip\_id = Trip\_id ;

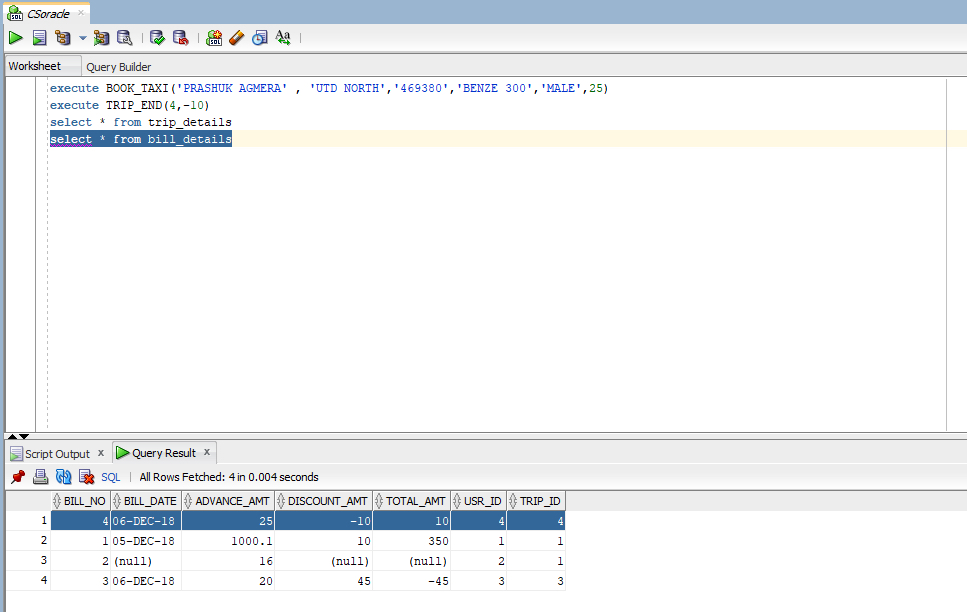
update BILL\_DETAILS set Bill\_date = sysdate , Discount\_amt = v\_discount ,Total\_amt = (v\_total\_time \* 15) - v\_discount where Trip\_id = v\_trip ;

END ;

END ;

/





# PL/SQL – TRIGGERS

## Procedure Code block for Update\_Driver\_Rating

----------------------------------------------

-- Trigger Creation

-- this trigger is called when inserted(After) to the feedback table

-- the trigger will decrease the driver rating by 1 if user feed back is bad for a driver

----------------------------------------------

CREATE OR REPLACE TRIGGER UPDATE\_DRIVER\_RATING

AFTER INSERT ON FEEDBACK

FOR EACH ROW

WHEN (NEW.Message like '%Bad Driver%' )

DECLARE

v\_driver\_id INT;

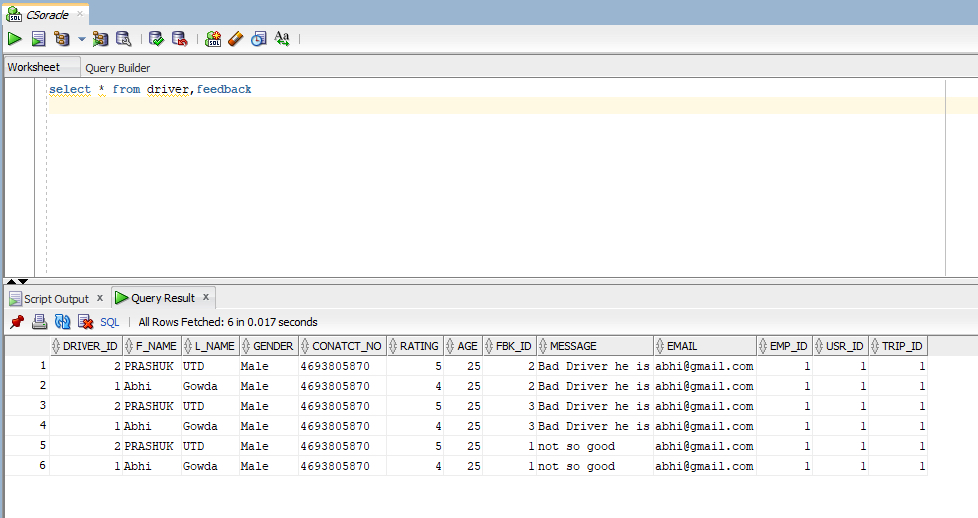
BEGIN

select driver\_id into v\_driver\_id from TRIP\_DETAILS where trip\_id = :NEW.Trip\_id;

update DRIVER set Rating = Rating -1 where driver\_id = v\_driver\_id;

END;

/



## Procedure Code block for Add\_no\_of\_cars

----------------------------------------------

-- Trigger Creation

-- this trigger is called before the INSERT OR UPDATE ON OWNS table

-- the trigger will calculate the number of cars owned by the owner and updates the no\_of\_cars columns in the OWNS table

----------------------------------------------

CREATE OR REPLACE TRIGGER ADD\_NO\_OF\_CARS

BEFORE INSERT OR UPDATE ON OWNS

FOR EACH ROW

DECLARE

v\_no\_of\_cars INT;

BEGIN

select count(Taxi\_id) into v\_no\_of\_cars from OWNER\_TAXI where Owner\_id = :NEW.Owner\_id group by Owner\_id;

:NEW.No\_Cars := v\_no\_of\_cars;

END;

/



# NORMALIZATION OF RELATIONAL SCHEMA

* TAXI

{Taxi\_id → Registration\_no, Taxi\_Model, Taxi\_Year, Taxi\_type, Status}

* USER

{Usr\_id → F\_name, L\_name, Contat\_no, Gender, Address, Taxi\_id}

* DRIVER

{Driver\_id → F\_name, L\_name, Gender, Conatct\_no, Rating, Age}

* TRIP\_DETAILS

{Trip\_id → Trip\_date, Trip\_amt, Driver\_id, Usr\_id, Taxi\_id, Strt\_time, End\_time}

* BILL\_DETAILS

{Bill\_no → Bill\_date, Advance\_amt, Discount\_amt, Total\_amt, Usr\_id, Trip\_id}

* CUSTOMER\_SERVICE

{Emp\_id → F\_name, L\_name}

* FEEDBACK

{Fbk\_id → Message, Email, Emp\_id, Usr\_id, Trip\_id}

* OWNER\_TAXI

{Owner\_id → Taxi\_id}

* OWNS

{Owner\_id → No\_Cars}

* INDIVIDUAL

{Ssn → Name, Owner\_id}

* TAXI\_SERVICE\_COMPANY

{Tsc\_id → Tsc\_name, Owner\_id}