FALL 2018

DATABASE SYSTEM FOR TAXI SERVICE

DATABSE DESIGN (CS 6360.002) - FINAL PROJECT

PRASHUK AJMERA - PXA172730
ABHISHEK HALUGUDDE SHIVAPPA – AXH180008

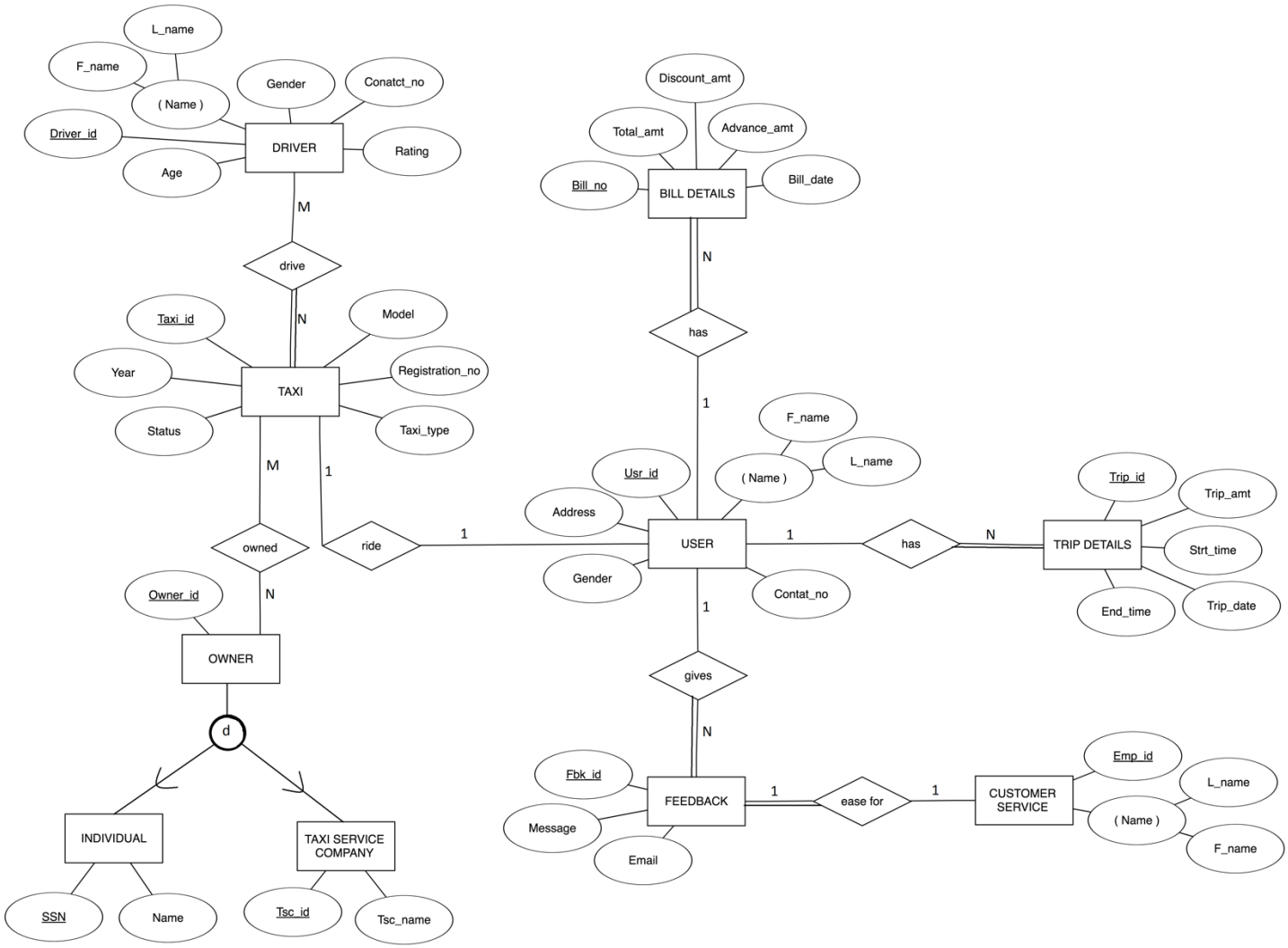
UNIVERSITY OF TEXAS AT DALLAS | Campbell Rd, Richardson, TX

Table of Contents

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.

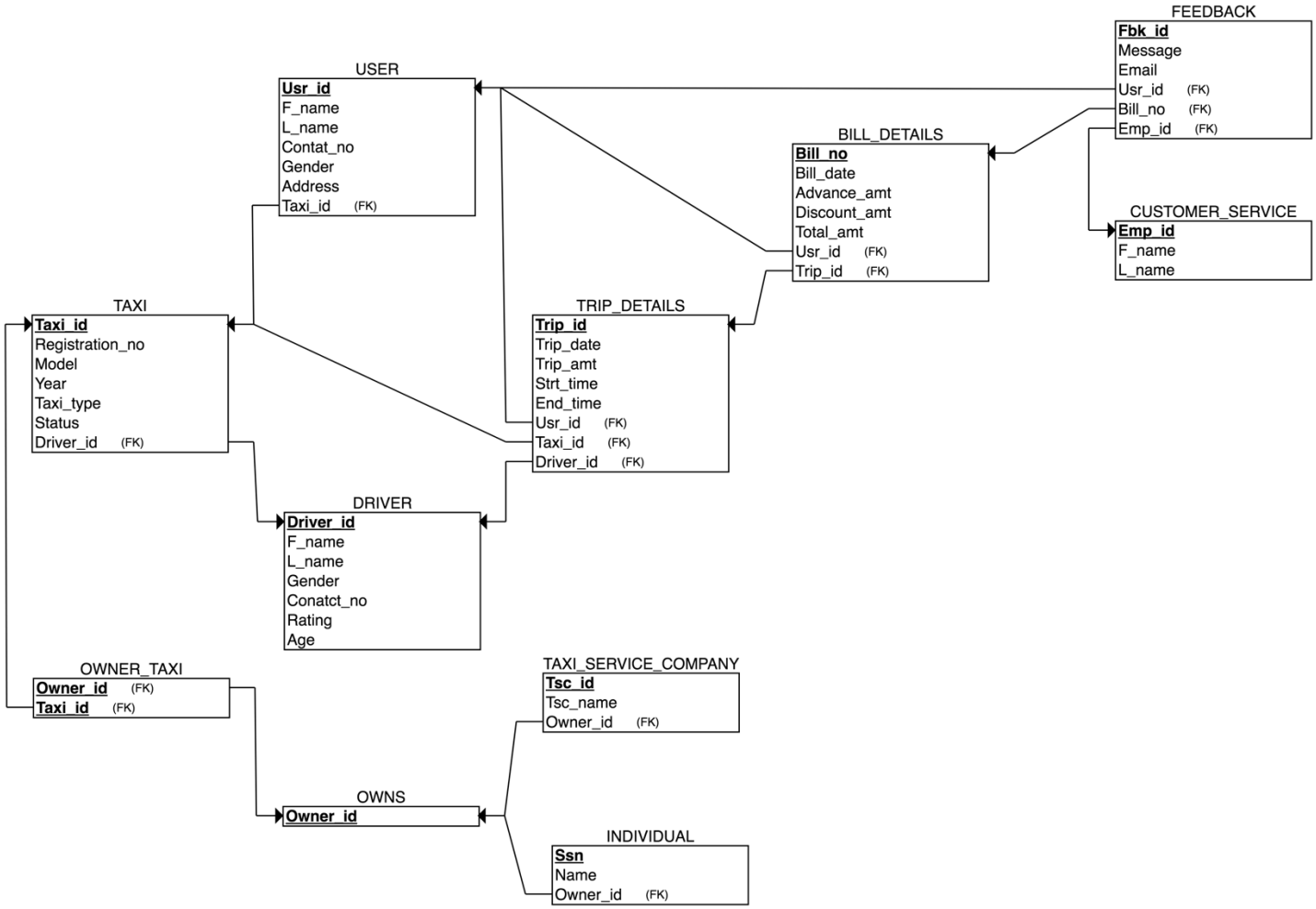
MODELLING OF REQUIREMENTS AS ER-DIAGRAM



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;
/
```

CSorade

Worksheet | Query Builder

```
execute BOOK_TAXI('PRASHUK AGMERA' , 'UTD NORTH','469380','BENZE 300','MALE',25)
select * from trip_details
select * from bill_details
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.013 seconds

	TRIP_ID	TRIP_DATE	TRIP_AMT	DRIVER_ID	USR_ID	TAXI_ID	STRT_TIME	END_TIME
1	4	06-DEC-18	50	1	4	1	06-DEC-18 09.41.39.000000000 AM	(null)
2	1	01-JAN-17	123	1	1	1	04-DEC-18 11.49.24.000000000 PM	06-DEC-18 12.07.34.000000000 AM
3	2	05-DEC-18	(null)	1	2	1	05-DEC-18 11.56.05.000000000 PM	06-DEC-18 12.07.34.000000000 AM
4	3	06-DEC-18	50	1	3	1	06-DEC-18 12.04.03.000000000 AM	06-DEC-18 12.07.34.000000000 AM

Worksheet | Query Builder

```
execute BOOK_TAXI('PRASHUK AGMERA' , 'UTD NORTH','469380','BENZE 300','MALE',25)
select * from trip_details
select * from bill_details
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.034 seconds

	BILL_NO	BILL_DATE	ADVANCE_AMT	DISCOUNT_AMT	TOTAL_AMT	USR_ID	TRIP_ID
1	4	(null)	25	(null)	(null)	4	4
2	1	05-DEC-18	1000.1	10	350	1	1
3	2	(null)	16	(null)	(null)	2	1
4	3	06-DEC-18	20	45	-45	3	3

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 ;
/
```

CSorade

Worksheet | Query Builder

```
execute BOOK_TAXI('PRASHUK AGMERA' , 'UTD NORTH','469380','BENZE 300','MALE',25)
execute TRIP_END(4,-10)
select * from trip_details
select * from bill_details
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.006 seconds

	TRIP_ID	TRIP_DATE	TRIP_AMT	DRIVER_ID	USR_ID	TAXI_ID	STRT_TIME	END_TIME
1	4	06-DEC-18	50	1	4	1	06-DEC-18 09.41.39.000000000 AM	06-DEC-18 09.47.56.000000000 AM
2	1	01-JAN-17	123	1	1	1	04-DEC-18 11.49.24.000000000 PM	06-DEC-18 09.47.56.000000000 AM
3	2	05-DEC-18	(null)	1	2	1	05-DEC-18 11.56.05.000000000 PM	06-DEC-18 09.47.56.000000000 AM
4	3	06-DEC-18	50	1	3	1	06-DEC-18 12.04.03.000000000 AM	06-DEC-18 09.47.56.000000000 AM

CSorade

Worksheet | Query Builder

```
execute BOOK_TAXI('PRASHUK AGMERA' , 'UTD NORTH','469380','BENZE 300','MALE',25)
execute TRIP_END(4,-10)
select * from trip_details
select * from bill_details
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.004 seconds

	BILL_NO	BILL_DATE	ADVANCE_AMT	DISCOUNT_AMT	TOTAL_AMT	USR_ID	TRIP_ID
1	4	06-DEC-18	25	-10	10	4	4
2	1	05-DEC-18	1000.1	10	350	1	1
3	2	(null)	16	(null)	(null)	2	1
4	3	06-DEC-18	20	45	-45	3	3

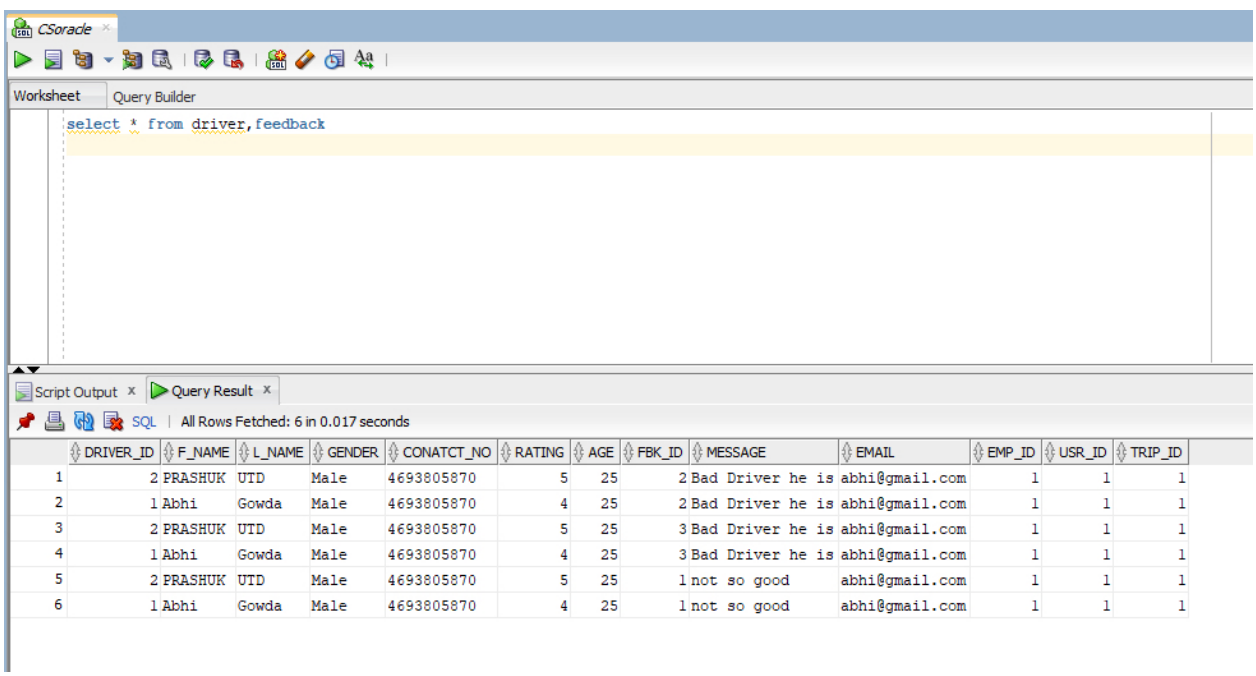
PL/SQL – TRIGGERS

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;
/
```

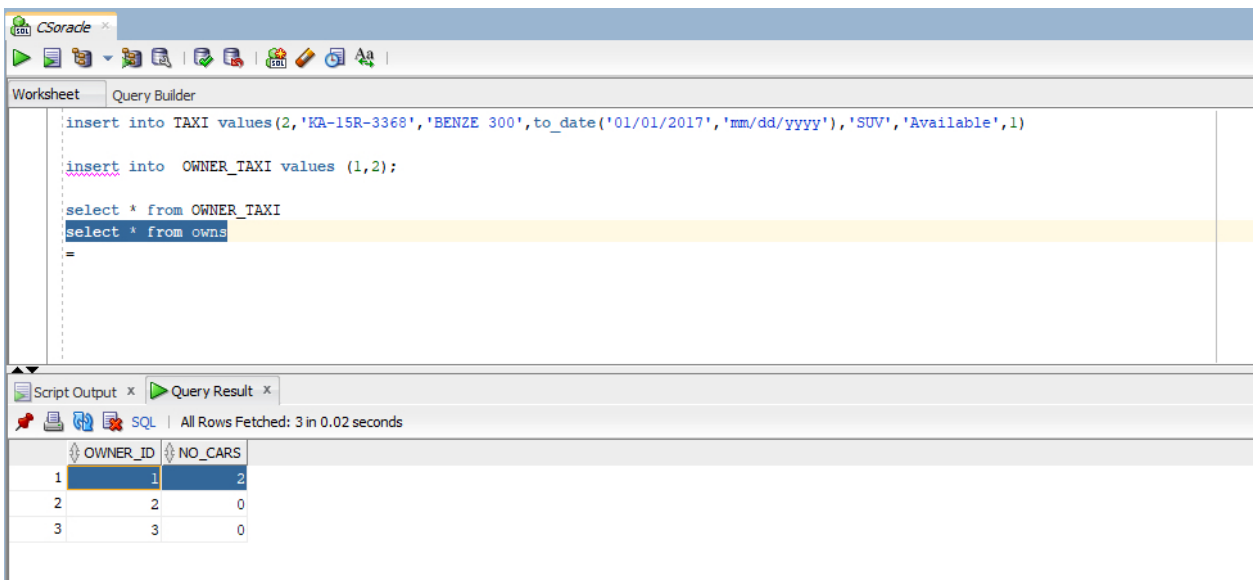


	DRIVER_ID	F_NAME	L_NAME	GENDER	CONATCT_NO	RATING	AGE	FBK_ID	MESSAGE	EMAIL	EMP_ID	USR_ID	TRIP_ID
1	2	PRASHUK	UTD	Male	4693805870	5	25	2	Bad Driver he is	abhi@gmail.com	1	1	1
2	1	Abhi	Gowda	Male	4693805870	4	25	2	Bad Driver he is	abhi@gmail.com	1	1	1
3	2	PRASHUK	UTD	Male	4693805870	5	25	3	Bad Driver he is	abhi@gmail.com	1	1	1
4	1	Abhi	Gowda	Male	4693805870	4	25	3	Bad Driver he is	abhi@gmail.com	1	1	1
5	2	PRASHUK	UTD	Male	4693805870	5	25	1	not so good	abhi@gmail.com	1	1	1
6	1	Abhi	Gowda	Male	4693805870	4	25	1	not so good	abhi@gmail.com	1	1	1

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}