Project:
Car Dealership Scenario

Rachel Armington
User24
Oracle SQL Developer: AIT732_2017
Professor: Harry Shasho
Due: May 11, 2017

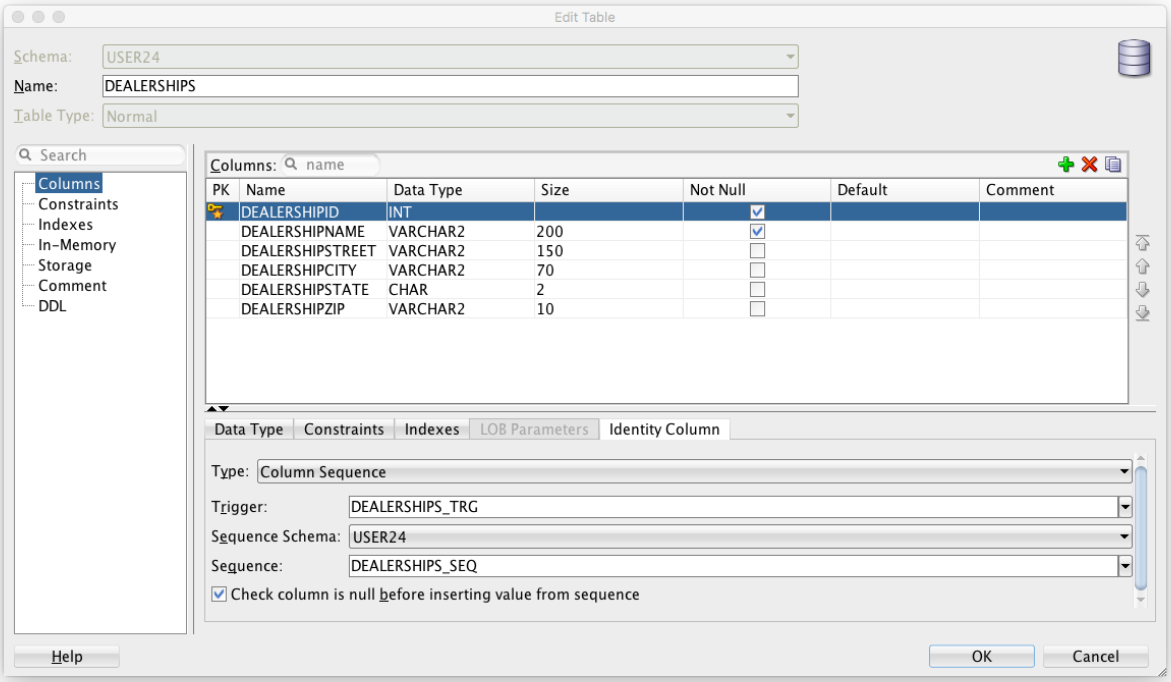
Table of Contents

Create Tables SQL Code

Dealerships Table

```
CREATE TABLE Dealerships (
dealershipId            int PRIMARY KEY NOT NULL,
dealershipName          varchar(200) NOT NULL UNIQUE,
dealershipStreet        varchar(150),
dealershipCity          varchar(70),
dealershipState         char(2),
dealershipZip           varchar(10)
);
```

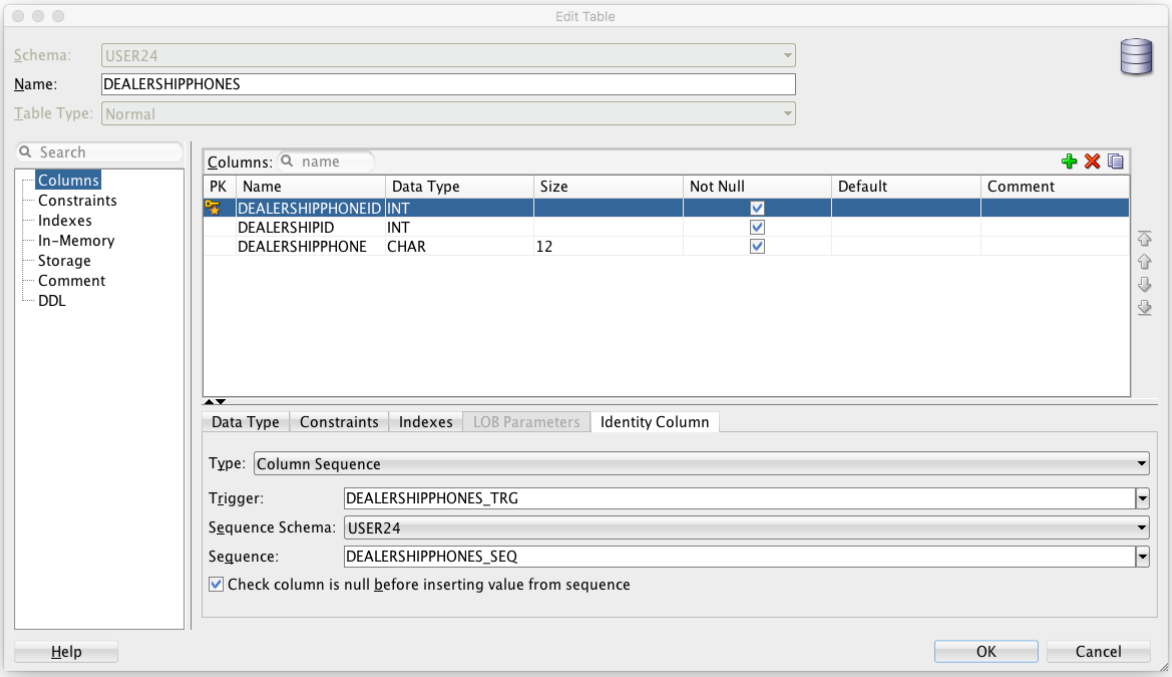
*NOTE: make dealershipId a column sequence (auto increment)



DealershipPhones Table

```
CREATE TABLE DealershipPhones (
dealershipPhoneId       int PRIMARY KEY NOT NULL,
dealershipId            int NOT NULL REFERENCES Dealerships(dealershipId),
dealershipPhone         char(12) NOT NULL
);
```

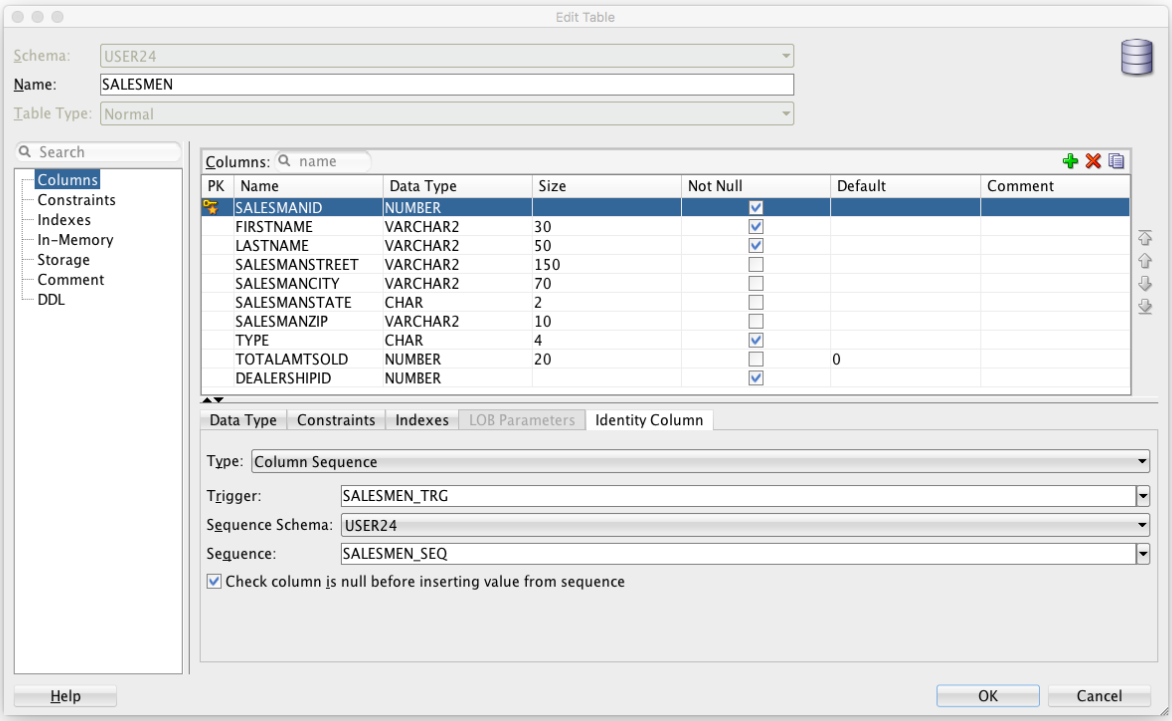
*NOTE: make dealershipPhoneId a column sequence (auto increment)



Salesmen Table

```
CREATE TABLE Salesmen (
salesmanId        int PRIMARY KEY NOT NULL,
firstName         varchar(30) NOT NULL,
lastName          varchar(50) NOT NULL,
salesmanStreet    varchar(150),
salesmanCity      varchar(70),
salesmanState     char(2),
salesmanZip       varchar(10),
type              char(4) NOT NULL CHECK (type IN ('Full', 'Part', 'Temp')),
totalAmtSold      number(20,2) DEFAULT 0,
dealershipId      int NOT NULL REFERENCES Dealerships(dealershipId)
);
```

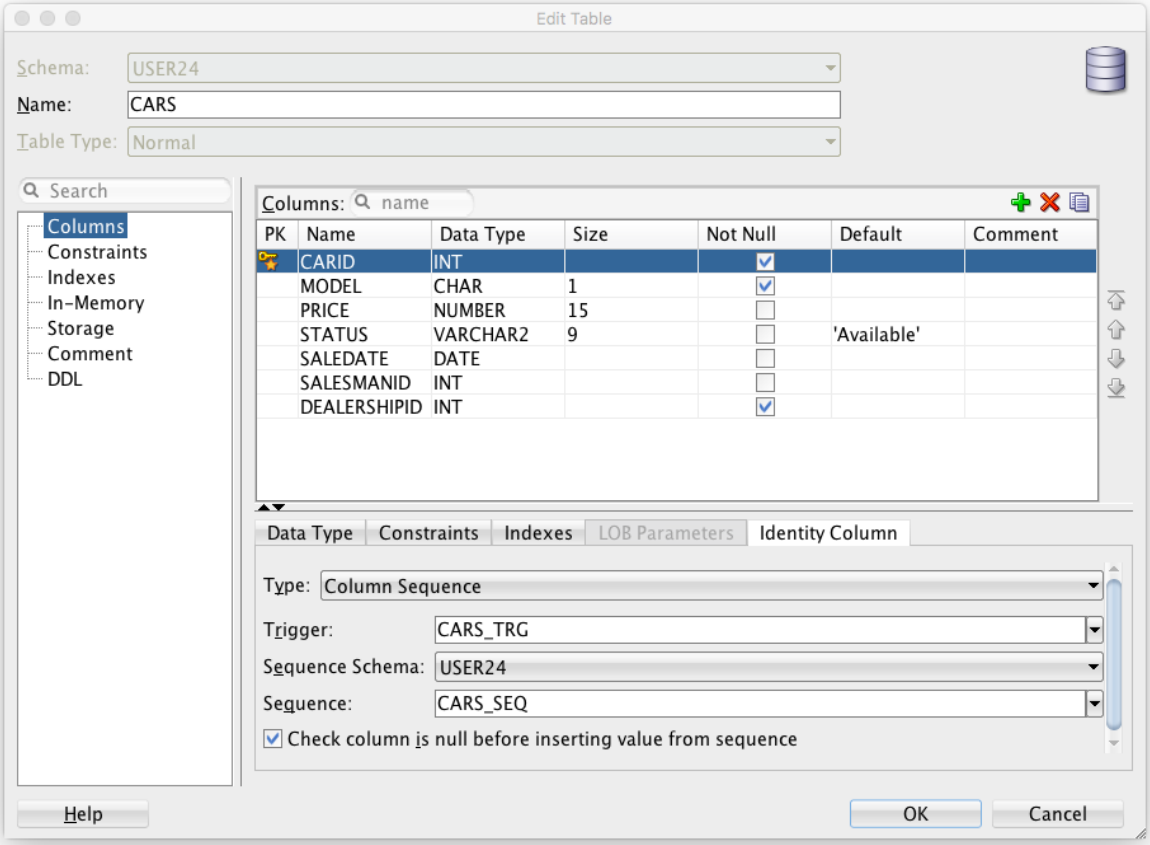
*NOTE: make salesmanId a column sequence (auto increment)



Cars Table

```
CREATE TABLE Cars (
carId              int PRIMARY KEY NOT NULL,
model              char(1) NOT NULL CHECK (model IN ('L', 'S', 'E')),
price              number(15,2) CHECK (price > 0),
status             varchar(9) DEFAULT 'Available',
saleDate           date,
salesmanId         int REFERENCES Salesmen(salesmanId),
dealershipId       int NOT NULL REFERENCES Dealerships(dealershipId)
);
```

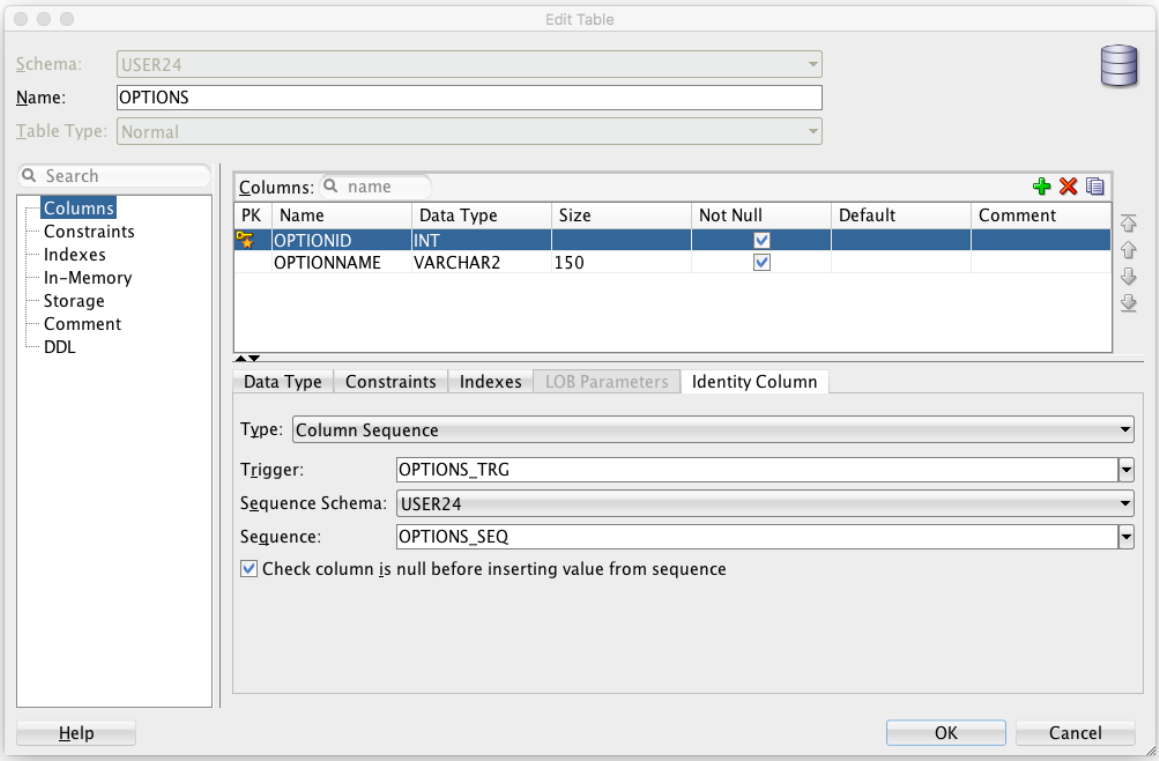
*NOTE: make carId a column sequence (auto increment)



Options Table

```
CREATE TABLE Options (
optionId    int PRIMARY KEY NOT NULL,
optionName  varchar(150) NOT NULL
);
```

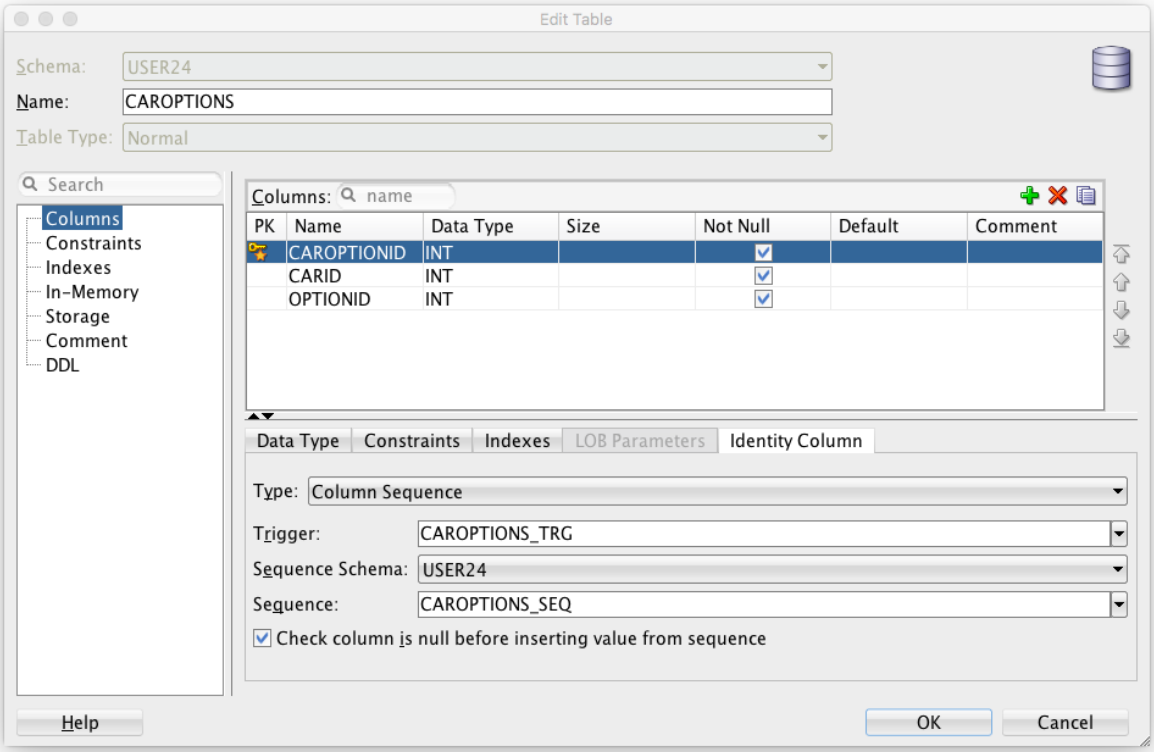
*NOTE: make optionId a column sequence (auto increment)



CarOptions Table

```
CREATE TABLE CarOptions (
carOptionId       int PRIMARY KEY NOT NULL,
carId             int NOT NULL REFERENCES Cars(carId),
optionId          int NOT NULL REFERENCES Options(optionId)
);
```

*NOTE: make carOptionId a column sequence (auto increment)



Insert Data Into Tables SQL Code

Dealerships Table

```
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'Passport BMW', '4730 Auth Place', 'Suitland',
'MD', '20746-4201');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Silver Spring', '3211 Automobile Blvd',
'Silver Spring', 'MD', '20904-4909');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Annapolis', '31 Old Mill Bottom Rd N',
'Annapolis', 'MD', '21409-5431');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Alexandria', '499 S Pickett St',
'Alexandria', 'VA', '22304-4705');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
```

```
(DEALERSHIPS_SEQ.nextval, 'BMW of Rockville', '1300 Rockville Pike',
'Rockville', 'MD', '20852-1412');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Fairfax', '8427 Lee Hwy', 'Fairfax', 'VA',
'22031-2212');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Catonsville', '6700 Baltimore National
Pike', 'Baltimore', 'MD', '21228-3909');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'Northwest BMW', '9702 Reisterstown Rd', 'Owings
Mills', 'MD', '21117-4120');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Towson', '700 Kenilworth Dr', 'Towson',
'MD', '21204-2427');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Sterling', '21710 Auto World Circle',
'Sterling', 'VA', '20166-2516');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'BMW of Bel Air', '716 Belair Rd', 'Bel Air', 'MD',
'21014-4222');
INSERT INTO Dealerships (dealershipId, dealershipName, dealershipStreet,
dealershipCity, dealershipState, dealershipZip) VALUES
(DEALERSHIPS_SEQ.nextval, 'Apple BMW of York', '1370 Roosevelt Ave', 'York',
'PA', '17404-2208');
```

AIT732_2017

Worksheet | Query Builder

```
select * from dealerships;
```

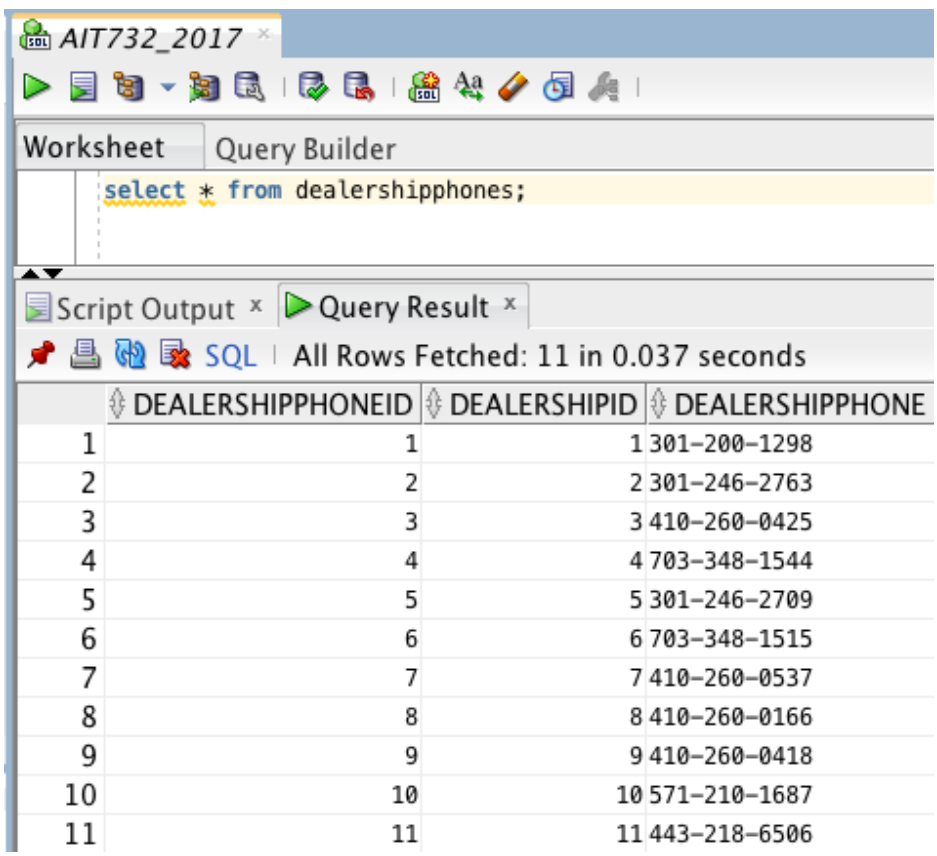
Script Output | Query Result

SQL | All Rows Fetched: 13 in 0.032 seconds

	DEALERSHIPID	DEALERSHIPNAME	DEALERSHIPSTREET	DEALERSHIPCITY	DEALERSHIPSTATE	DEALERSHIPZIP
1	1	Passport BMW	4730 Auth Place	Suitland	MD	20746-4201
2	2	BMW of Silver Spring	3211 Automobile Blvd	Silver Spring	MD	20904-4909
3	3	BMW of Annapolis	31 Old Mill Bottom Rd N	Annapolis	MD	21409-5431
4	4	BMW of Alexandria	499 S Pickett St	Alexandria	VA	22304-4705
5	5	BMW of Rockville	1300 Rockville Pike	Rockville	MD	20852-1412
6	6	BMW of Fairfax	8427 Lee Hwy	Fairfax	VA	22031-2212
7	7	BMW of Catonsville	6700 Baltimore National Pike	Baltimore	MD	21228-3909
8	8	Northwest BMW	9702 Reisterstown Rd	Owings Mills	MD	21117-4120
9	9	BMW of Towson	700 Kenilworth Dr	Towson	MD	21204-2427
10	10	BMW of Sterling	21710 Auto World Circle	Sterling	VA	20166-2516
11	11	BMW of Bel Air	716 Belair Rd	Bel Air	MD	21014-4222
12	12	Apple BMW of York	1370 Roosevelt Ave	York	PA	17404-2208
13	14	BMW of Lancaster	1530 Commerce Drive	Lancaster	PA	17601-0000

DealershipPhones Table

```
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '1', '301-200-1298');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '2’, '301-246-2763');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '3', '410-260-0425');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '4', '703-348-1544');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '5', '301-246-2709');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '6', '703-348-1515');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '7', '410-260-0537');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '8', '410-260-0166');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '9', '410-260-0418');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '10', '571-210-1687');
INSERT INTO DealershipPhones (dealershipPhoneId, dealershipId,
dealershipPhone) VALUES (DEALERSHIPPHONES_SEQ.nextval, '11', '443-218-6506');
```



	DEALERSHIPPHONEID	DEALERSHIPID	DEALERSHIPPHONE
1	1	1	301-200-1298
2	2	2	301-246-2763
3	3	3	410-260-0425
4	4	4	703-348-1544
5	5	5	301-246-2709
6	6	6	703-348-1515
7	7	7	410-260-0537
8	8	8	410-260-0166
9	9	9	410-260-0418
10	10	10	571-210-1687
11	11	11	443-218-6506

Salesmen Table

```
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Ellie', 'Mi', '2215 Briarcliff Ct', 'Tysons
Corner', 'VA', '22182', 'Part', '45199', '6');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Courtney', 'Yu', '911 Army Rd', 'Towson',
'MD', '21204', 'Full', '0', '9');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Emily', 'Sloane', '9 West Maple St',
'Alexandria', 'VA', '22301', 'Temp', '0', '4');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Amanda', 'Drew', '1604 McGuckian St',
'Annapolis', 'MD', '21401', 'Full', '0', '3');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'John', 'Maple', '1510 Baylor Ave',
'Rockville', 'MD', '20850', 'Full', '0', '5');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Josh', 'Cooney', '1611 Sanford Rd', 'Silver
Spring', 'MD', '20902', 'Part', '0', '2');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'David', 'Batlas', '4710 Huron Ave',
'Suitland', 'MD', '20746', 'Part', '0', '1');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Jimmy', 'Consolas', '1006 Magruder Ave',
'Catonsville', 'MD', '21228', 'Part', '0', '7');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Rachelle', 'McCarthy', '106 Wengate Rd',
'Owings Mills', 'MD', '21117', 'Temp', '0', '8');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Kathleen', 'Simons', '9 S Hickory Ave', 'Bel
Air', 'MD', '21014', 'Part', '0', '11');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Katie', 'Brooks', '1000 S Hoga Rd',
'Sterling', 'VA', '20164', 'Temp', '0', '10');
INSERT INTO Salesmen (salesmanId, firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, type, totalAmtSold, dealershipId)
VALUES (SALESMEN_SEQ.nextval, 'Jacob', 'Chung', '7711 Magarity Rd', 'Falls
Church', 'VA', '22043', 'Temp', '0', '6');
```

AIT732_2017

Worksheet | Query Builder

```
select * from salesmen;
```

Script Output | Query Result

SQL | All Rows Fetched: 12 in 0.022 seconds

	SALESMANID	FIRSTNAME	LASTNAME	SALESMANSTREET	SALESMANCITY	SALESMANSTATE	SALESMANZIP	TYPE	TOTALAMTSOLD	DEALERSHIPID
1	1	Ellie	Mi	2215 Briarcliff Ct	Tysons Corner	VA	22182	Part	45199	6
2	2	Courtney	Yu	911 Army Rd	Towson	MD	21204	Full	0	9
3	3	Emily	Sloane	9 West Maple St	Alexandria	VA	22301	Temp	0	4
4	4	Amanda	Drew	1604 McGuckian St	Annapolis	MD	21401	Full	0	3
5	5	John	Maple	1510 Baylor Ave	Rockville	MD	20850	Full	0	5
6	6	Josh	Cooney	1611 Sanford Rd	Silver Spring	MD	20902	Part	0	2
7	7	David	Batlas	4710 Huron Ave	Suitland	MD	20746	Part	0	1
8	8	Jimmy	Consolas	1006 Magruder Ave	Catonsville	MD	21228	Part	0	7
9	9	Rachelle	McCarthy	106 Wengate Rd	Owings Mills	MD	21117	Temp	0	8
10	10	Kathleen	Simons	9 S Hickory Ave	Bel Air	MD	21014	Part	0	11
11	11	Katie	Brooks	1000 S Hoga Rd	Sterling	VA	20164	Temp	0	10
12	14	Jacob	Chung	7711 Magarity Rd	Falls Church	VA	22043	Temp	0	6

Cars Table

```
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'L', '39500', 'Available', '', '',
'1');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'S', '49000', 'Available', '', '',
'1');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'E', '44500', 'Available', '', '',
'1');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'L', '43498', 'Available', '', '',
'2');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'S', '43809', 'Available', '', '',
'3');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'E', '43864', 'Available', '', '',
'4');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'E', '43564', 'Available', '', '',
'5');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'L', '45199', 'Sold', '11-Apr-2017',
'1', '6');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'S', '44979', 'Available', '', '',
'6');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'E', '46999', 'Available', '', '',
'7');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'L', '45814', 'Available', '', '',
'8');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'S', '46729', 'Available', '', '',
'9');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'E', '46004', 'Available', '', '',
'10');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'L', '46559', 'Available', '', '',
'10');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'S', '48909', 'Available', '', '',
'11');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'E', '47609', 'Available', '', '',
'11');
```

```
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'S', '44898', 'Available', '', '',
'7');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'L', '50002', 'Available', '', '',
'7');
INSERT INTO Cars (carId, model, price, status, saleDate, salesmanId,
dealershipId) VALUES (CARS_SEQ.nextval, 'S', '70000', 'Available', '', '',
'3');
```

AIT732_2017

Worksheet | Query Builder

```
select * from Cars;
```

Script Output | Query Result

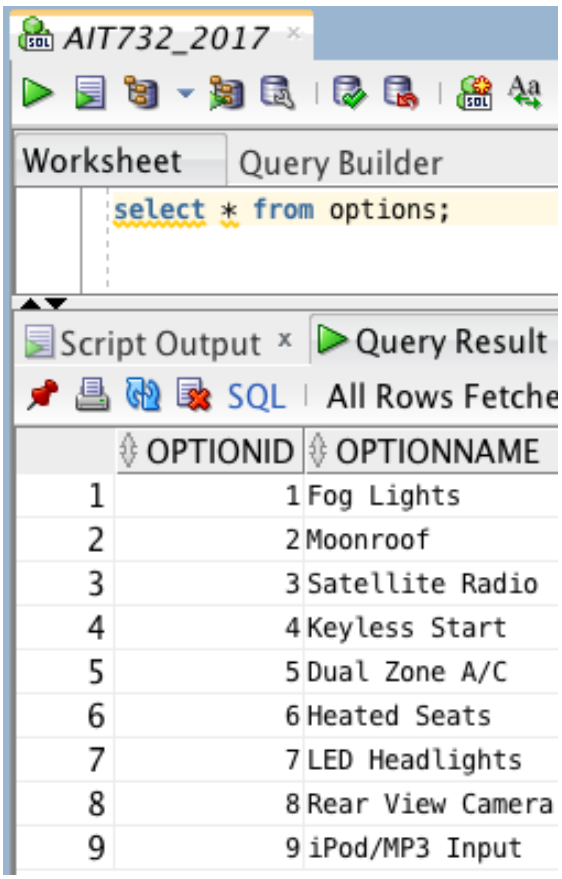
SQL | All Rows Fetched: 20 in 0.021 seconds

	CA...	MODEL	PRICE	STATUS	SALEDATE	SALESMANID	DEALERSHIPID
1	1	L	50000	Sold	24-APR-17	2	1
2	2	S	49000	Available	(null)	(null)	1
3	3	E	44500	Available	(null)	(null)	1
4	4	L	43498	Available	(null)	(null)	2
5	5	S	43809	Available	(null)	(null)	3
6	6	E	43864	Available	(null)	(null)	4
7	7	E	43564	Available	(null)	(null)	5
8	8	L	45199	Sold	11-APR-17	1	6
9	9	S	44979	Available	(null)	(null)	6
10	10	E	46999	Available	(null)	(null)	7
11	11	L	45814	Available	(null)	(null)	8
12	12	S	46729	Available	(null)	(null)	9
13	13	E	46004	Available	(null)	(null)	10
14	14	L	46559	Available	(null)	(null)	10
15	15	S	48909	Available	(null)	(null)	11
16	16	E	47609	Available	(null)	(null)	11
17	19	S	44898	Available	(null)	(null)	7
18	24	L	50002	Available	(null)	(null)	7
19	27	S	70000	AVAILABLE	(null)	(null)	3

Options Table

```
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval, 'Fog
Lights');
```

```
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval,
'Moonroof');
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval,
'Satellite Radio');
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval,
'Keyless Start');
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval, 'Dual
Zone A/C');
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval,
'Heated Seats');
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval, 'LED
Headlights');
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval, 'Rear
View Camera');
INSERT INTO Options (optionId, optionName) VALUES (OPTIONS_SEQ.nextval,
'iPod/MP3 Input');
```



CarOptions Table

```
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '1', '1');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '1', '2');
```

```
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '2', '1');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '2', '3');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '3', '1');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '4', '2');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '5', '1');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '6', '3');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '7', '3');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '1');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '2');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '3');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '4');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '5');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '6');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '7');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '8');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '8', '9');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '9', '6');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '10', '6');
INSERT INTO CarOptions (carOptionId, carID, optionId) VALUES
(CAROPTIONS_SEQ.nextval, '11', '9');
```

AIT732_2017

Worksheet | Query Builder

```
select * from caroptions;
```

Script Output | Query Result

SQL | All Rows Fetched: 21 in 0

	CAROPTIONID	CARID	OPTIONID
1	1	1	1
2	2	1	2
3	3	2	1
4	4	2	3
5	5	3	1
6	6	4	2
7	7	5	1
8	8	6	3
9	9	7	3
10	10	8	1
11	11	8	2
12	12	8	3
13	13	8	4
14	14	8	5
15	15	8	6
16	16	8	7
17	17	8	8
18	18	8	9
19	19	9	6
20	20	10	6
21	21	11	9

Requirements

Requirement 1
(Using a procedure) The system should be able to add new dealerships making sure that the name is not a duplicate. Multiple phone numbers can be added manually (without a procedure). A message should be returned with the success or failure of the action.

Code to Solve Requirement 1:
This procedure is passed in a new dealership name (varchar), dealership street (varchar), dealership city (varchar), dealership state (char), and dealership zip code (varchar), and it returns a message with the success or failure of the insert. If a duplicate dealership name is entered, it returns an error message.

```
Create or Replace Procedure insert_dealership
```

```
(v_dealershipName varchar, v_dealershipStreet varchar, v_dealershipCity
varchar, v_dealershipState char, v_dealershipZip varchar)
IS
      v_count int;      --USED TO VALIDATE INCOMING DEALERSHIP NAME
      v_ErrorCode number;     --USED FOR ERROR CHECKING
      v_ErrorMsg  varchar2(200);
      v_CurrentUser varchar2(100);

BEGIN

      /*  VALIDATE INCOMING DEALERSHIP NAME */
      select count(*)
      into v_count
      from Dealerships
      where upper(dealershipName) = upper(v_dealershipName);

      if v_count <> 0 then
            DBMS_OUTPUT.PUT_LINE('Dealership name already exists.
Insert failed.');

      else

            begin

                  insert into Dealerships (dealershipName,
dealershipStreet, dealershipCity, dealershipState, dealershipZip)
values (v_dealershipName, v_dealershipStreet, v_dealershipCity,
v_dealershipState, v_dealershipZip);

                  DBMS_OUTPUT.PUT_LINE('Inserted '  || SQL%ROWCOUNT ||
' Rows.');

                  commit;  -- TRANSACTION CONTROL

            end;

      end if;

      Exception

      When OTHERS THEN
            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            v_CurrentUser := USER;

            DBMS_OUTPUT.PUT_LINE( TO_CHAR(SYSDATE)  || v_CurrentUser
|| TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg || 'insert failed');

END;
```

```
/* TEST THE PROCEDURE */

execute insert_dealership ('BMW of Lancaster', '1530 Commerce Drive',
'Lancaster', 'PA', '17601-0000');

execute insert_dealership ('BMW of Lancaster', '', '', '', '');

select * from Dealerships;
```

Oracle SQL Developer : AIT732_2017

Connections

- Connections
  - AIT732_2017
    - Tables (Filtered)
      - CAROPTIONS
      - CARS
      - DEALERSHIPPHONES
      - DEALERSHIPS
        - DEALERSHIPID
        - DEALERSHIPNAME
        - DEALERSHIPSTREET
        - DEALERSHIPCITY
        - DEALERSHIPSTATE
        - DEALERSHIPZIP
      - DEPT
      - EMP
      - OPTIONS

Reports

- All Reports
  - Data Dictionary Reports
  - Data Modeler Reports
  - OLAP Reports
  - TimesTen Reports
  - User Defined Reports

AIT732_2017

Worksheet | Query Builder

```
execute insert_dealership ('BMW of Lancaster', '1530 Commerce Drive', 'Lancaster', 'PA', '17601-0000');

execute insert_dealership ('BMW of Lancaster', '', '', '', '');
```

Query Result | Script Output

Task completed in 0.04 seconds

```
PL/SQL procedure successfully completed.

Inserted 1 Rows.

PL/SQL procedure successfully completed.

Dealership name already exists.
```

Click on an identifier with the Command key down to perform "Go to Declaration" | Line 3 Column 64 | Insert | Modified | Unix/Mac: LF

Oracle SQL Developer : AIT732_2017

Connections

- Connections
  - AIT732_2017
    - Tables (Filtered)
      - CAROPTIONS
      - CARS
      - DEALERSHIPPHONES
      - DEALERSHIPS
        - DEALERSHIPID
        - DEALERSHIPNAME
        - DEALERSHIPSTREET
        - DEALERSHIPCITY
        - DEALERSHIPSTATE
        - DEALERSHIPZIP
      - DEPT
      - EMP
      - OPTIONS

Reports

- All Reports
  - Data Dictionary Reports
  - Data Modeler Reports
  - OLAP Reports
  - TimesTen Reports
  - User Defined Reports

AIT732_2017

Worksheet | Query Builder

```
execute insert_dealership ('BMW of Lancaster', '1530 Commerce Drive', 'Lancaster', 'PA', '17601-0000');

execute insert_dealership ('BMW of Lancaster', '', '', '', '');

select * from dealerships;
```

Script Output | Query Result

SQL | All Rows Fetched: 13 in 0.014 seconds

	DEALERSHIPID	DEALERSHIPNAME	DEALERSHIPSTREET	DEALERSHIPCITY	DEALERSHIPSTATE	DEALERSHIPZIP
1	1	Passport BMW	4730 Auth Place	Suitland	MD	20746-4201
2	2	BMW of Silver Spring	3211 Automobile Blvd	Silver Spring	MD	20904-4909
3	3	BMW of Annapolis	31 Old Mill Bottom Rd N	Annapolis	MD	21409-5431
4	4	BMW of Alexandria	499 S Pickett St	Alexandria	VA	22304-4705
5	5	BMW of Rockville	1300 Rockville Pike	Rockville	MD	20852-1412
6	6	BMW of Fairfax	8427 Lee Hwy	Fairfax	VA	22031-2212
7	7	BMW of Catonsville	6700 Baltimore National Pike	Baltimore	MD	21228-3909
8	8	Northwest BMW	9702 Reisterstown Rd	Owings Mills	MD	21117-4120
9	9	BMW of Towson	700 Kenilworth Dr	Towson	MD	21204-2427
10	10	BMW of Sterling	21710 Auto World Circle	Sterling	VA	20166-2516
11	11	BMW of Bel Air	716 Belair Rd	Bel Air	MD	21014-4222
12	12	Apple BMW of York	1370 Roosevelt Ave	York	PA	17404-2208
13	14	BMW of Lancaster	1530 Commerce Drive	Lancaster	PA	17601-0000

Click on an identifier with the Command key down to perform "Go to Declaration" | Line 5 Column 27 | Insert | Modified | Unix/Mac: LF

Requirement 2
(Using a procedure) The system should be able to add a salesman with a valid type to the database without duplicates. If there is a duplicate, a message should be shown. Total amount sold is initially zero. A message should be returned with the success or failure of the action.

Code to Solve Requirement 2:
This procedure is passed a new salesman's first name (varchar), last name (varchar), street address (varchar), city (varchar), state (char), zip code (varchar), type (char), and dealership Id (int), and it returns a message with the success or failure of the insert. If an incorrect dealership Id or type are entered, or if the salesman's name (first and last concatenated) already exists, then it returns a message.

```
Create or Replace Procedure insert_salesman
(v_firstName varchar, v_lastName varchar, v_salesmanStreet varchar,
v_salesmanCity varchar, v_salesmanState char, v_salesmanZip varchar,
v_type char, v_dealershipId int)
IS
      v_dlrshp_id int;  --USED TO VALIDATE INCOMING SALESMAN'S
DEALERSHIPID
      v_count int;            --USED TO VALIDATE SALESMAN TYPE
      v_count2 int;           --USED TO VALIDATE INCOMING SALESMAN NAME
      v_ErrorCode number;     --USED FOR ERROR CHECKING
      v_ErrorMsg  varchar2(200);
      v_CurrentUser varchar2(100);

BEGIN

      /*  VALIDATE INCOMING SALESMAN'S DEALERSHIPID, TYPE, AND NAME */

      select dealershipId
      into v_dlrshp_id
      from Dealerships
      where dealershipId = v_dealershipId;

      select count(*)
      into v_count
      from Salesmen
      where (upper(v_type) = 'FULL' OR upper(v_type) = 'PART' OR
upper(v_type) = 'TEMP');

      select count(*)
      into v_count2
      from Salesmen
      where (upper(firstName) || ' ' || upper(lastName)) =
(upper(v_firstName) || ' ' || upper(v_lastName));

      if v_count = 0 then
            DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_type) || ' is an invalid
type. Insert failed.');
```

```
      elsif v_count2 <> 0 then
            DBMS_OUTPUT.PUT_LINE('Salesman already exists. Insert
failed.');

      else

            begin

       insert into Salesmen (firstName, lastName, salesmanStreet,
salesmanCity, salesmanState, salesmanZip, "TYPE", totalAmtSold,
dealershipId) values (v_firstName, v_lastName, v_salesmanStreet,
v_salesmanCity, v_salesmanState, v_salesmanZip, v_type, '0',
v_dealershipId);

                  DBMS_OUTPUT.PUT_LINE('Inserted '  || SQL%ROWCOUNT ||
' Rows.');

                  commit;  -- TRANSACTION CONTROL

            end;

      end if;

      Exception

      When NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Invalid DealershipId for this
salesman. Insert failed.');

      When OTHERS THEN
            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            v_CurrentUser := USER;

            DBMS_OUTPUT.PUT_LINE( TO_CHAR(SYSDATE)  || v_CurrentUser
|| TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg || 'insert failed');

END;


/* TEST THE PROCEDURE */

execute insert_salesman ('Marc', 'Myers', '8001 Eastern Dr', 'Silver Spring', 'MD',
'20910', 'Full', '2');

execute insert_salesman ('Marc', 'Myers', '', '', '', '', 'Full', '2');

execute insert_salesman ('John', 'Doe', '', '', '', '', 'Contract', '2');
```

```
execute insert_salesman ('John', 'Doe', '', '', '', '', 'Temp', '200');

select * from salesmen;
```

Oracle SQL Developer : AIT732_2017

Connections

- AIT732_2017
  - Tables (Filtered)
    - CAROPTIONS
    - CARS
    - DEALERSHIPP
    - DEALERSHIPS
    - DEPT
    - EMP
    - OPTIONS
    - PRICE_UPD_L
    - RAISE_LOG
    - SALESMEN
      - SALESMAN
      - FIRSTNAM
      - LASTNAM
      - SALESMAN

Reports

- All Reports
- Data Dictionary Repo
- Data Modeler Report
- OLAP Reports
- TimesTen Reports
- User Defined Reports

AIT732_2017 | SALESMEN

AIT732_2017

Worksheet | Query Builder

```
execute insert_salesman ('Marc', 'Myers', '8001 Eastern Dr', 'Silver Spring', 'MD', '20910', 'Full', '2');

execute insert_salesman ('Marc', 'Myers', '', '', '', '', 'Full', '2');

execute insert_salesman ('John', 'Doe', '', '', '', '', 'Contract', '2');

execute insert_salesman ('John', 'Doe', '', '', '', '', 'Temp', '200');
```

Query Result | Script Output

Task completed in 0.21 seconds

```
PL/SQL procedure successfully completed.

Inserted 1 Rows.

PL/SQL procedure successfully completed.
Salesman already exists. Insert failed.

PL/SQL procedure successfully completed.

Contract is an invalid type. Insert failed.

PL/SQL procedure successfully completed.
Invalid DealershipId for this salesman. Insert failed.
```

Click on an identifier with the Command key down to perform "Go to Declaration"

Oracle SQL Developer : AIT732_2017

Connections

- AIT732_2017
  - Tables (Filtered)
    - CAROPTIONS
    - CARS
    - DEALERSHIPF
    - DEALERSHIPS
    - DEPT
    - EMP
    - OPTIONS
    - PRICE_UPD_L
    - RAISE_LOG
    - SALESMEN
      - SALESMAN
      - FIRSTNAM
      - LASTNAM
      - SALESMAN

Reports

- All Reports
- Data Dictionary Repo
- Data Modeler Report
- OLAP Reports
- TimesTen Reports
- User Defined Reports

AIT732_2017 | SALESMEN

Worksheet | Query Builder

```
execute insert_salesman ('Marc', 'Myers', '8001 Eastern Dr', 'Silver Spring', 'MD', '20910', 'Full', '2');

execute insert_salesman ('Marc', 'Myers', '', '', '', '', 'Full', '2');

execute insert_salesman ('John', 'Doe', '', '', '', '', 'Contract', '2');

execute insert_salesman ('John', 'Doe', '', '', '', '', 'Temp', '200');

select * from salesmen;
```

Script Output | Query Result

All Rows Fetched: 13 in 0.097 seconds

	SALESMA...	FIRSTNAME	LASTNAME	SALESMANSTREET	SALESMANCITY	SALESMANSTATE	SALESMANZIP	TYPE	TOTALAMTSOLD	DEALERSHIPID
1	1	Ellie	Mi	2215 Briarcliff Ct	Tysons Corner	VA	22182	Part	45199	6
2	2	Courtney	Yu	911 Army Rd	Towson	MD	21204	Full	50000	9
3	3	Emily	Sloane	9 West Maple St	Alexandria	VA	22301	Temp	0	4
4	4	Amanda	Drew	1604 McGuckian St	Annapolis	MD	21401	Full	0	3
5	5	John	Maple	1510 Baylor Ave	Rockville	MD	20850	Full	0	5
6	6	Josh	Cooney	1611 Sanford Rd	Silver Spring	MD	20902	Part	0	2
7	7	David	Batlas	4710 Huron Ave	Suitland	MD	20746	Part	0	1
8	8	Jimmy	Consolas	1006 Magruder Ave	Catonsville	MD	21228	Part	0	7
9	9	Rachelle	McCarthy	106 Wengate Rd	Owings Mills	MD	21117	Temp	0	8
10	10	Kathleen	Simons	9 S Hickory Ave	Bel Air	MD	21014	Part	0	11
11	11	Katie	Brooks	1000 S Hoga Rd	Sterling	VA	20164	Temp	0	10
12	14	Jacob	Chung	7711 Magarity Rd	Falls Church	VA	22043	Temp	0	6
13	21	Marc	Myers	8001 Eastern Dr	Silver Spring	MD	20910	Full	0	2

Click on an identifier with the Command key down to perform "Go to Declaration"

Requirement 3
(Using a procedure) The system should be able to add valid cars to the database making sure they have a valid model. A message should be returned with the success or failure of the action. Status should be initially available. Note: Options may be added manually (without a procedure if necessary).

Code to Solve Requirement 3:
This procedure is passed a new car model, price, and dealership Id, and it returns a message with the success or failure of the insert. If an incorrect model is entered or an incorrect dealership Id is entered, it returns a message. If a price less than or equal to $0 is entered, the validation trigger (trg_i_car) fires and returns an error.

```
Create or Replace Procedure insert_car
(v_model char, v_price number, v_dealershipId int)
IS
      v_dlrshp_id int;        --USED TO VALIDATE DEALERSHIPID
      v_count int;            --USED TO VALIDATE MODEL
      v_ErrorCode number;     --USED FOR ERROR CHECKING
      v_ErrorMsg  varchar2(200);
      v_CurrentUser varchar2(100);

BEGIN

      /*  VALIDATE DEALERSHIPID OF INCOMING CAR MODEL. IF INVALID, GOES
TO EXCEPTION */
      select dealershipId
      into v_dlrshp_id
      from Dealerships
      where dealershipId = v_dealershipId;

      /*  VALIDATE MODEL */
      select count(*)
      into v_count
      from Cars
      where (upper(v_model) = 'L' OR upper(v_model) = 'S' OR
upper(v_model) = 'E');

      if v_count = 0 then
            DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_model) || ' is an invalid
model. Insert failed.');

      else

            begin

            insert into Cars (model, price, status, dealershipId)
values (v_model, v_price, 'Available', v_dealershipId);
```

```
            DBMS_OUTPUT.PUT_LINE('Inserted '  || SQL%ROWCOUNT || '
Rows.');

            commit;  -- TRANSACTION CONTROL

            end;

      end if;

      Exception

      When NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Invalid DealershipId for this car.
Insert failed.');

      When OTHERS THEN
            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            v_CurrentUser := USER;

            DBMS_OUTPUT.PUT_LINE( TO_CHAR(SYSDATE)  || v_CurrentUser
|| TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg || ' Insert failed.');

END;


/* TRIGGER */

-- TRIGGER TO VALIDATE price

Create or Replace Trigger trg_i_car
Before Insert
On Cars
For Each Row
Begin

      If :new.price <=0 THEN

        raise_application_error (-20001,
                'Car price may not be less than zero. Insert failed.');

      End If;

      :new.carId := UPPER(TRIM(:new.carId));
      :new.model := UPPER(TRIM(:new.model));
      :new.status := UPPER(TRIM(:new.status));
      :new.saleDate := UPPER(TRIM(:new.saleDate));
      :new.salesmanId := UPPER(TRIM(:new.salesmanId));
      :new.dealershipId := UPPER(TRIM(:new.dealershipId));
```

```
End trg_i_car;


/* TEST THE PROCEDURE */

execute insert_car ('S', '44898', '10');

execute insert_car ('X', '44898', '7');

execute insert_car ('S', '0', '7');

execute insert_car ('S', '44898', '100');

select * from cars;
```

Oracle SQL Developer : AIT732_2017

Connections

- Connections
  - AIT732_2017
    - Tables (Filtered)
      - CAROPTIONS
      - CARS
        - CARID
        - MODEL
        - PRICE
        - STATUS
        - SALEDATE
        - SALESMAN
        - DEALERSH
      - DEALERSHIPP
      - DEALERSHIPS
      - DEPT
      - EMP

Reports

- All Reports
- Data Dictionary Repo
- Data Modeler Report
- OLAP Reports
- TimesTen Reports
- User Defined Reports

AIT732_2017

Worksheet | Query Builder

```
execute insert_car ('S', '44898', '10');

execute insert_car ('X', '44898', '7');

execute insert_car ('S', '0', '7');

execute insert_car ('S', '44898', '100');
```

Query Result | Script Output

Task completed in 0.148 seconds

```
PL/SQL procedure successfully completed.

Inserted 1 Rows.

PL/SQL procedure successfully completed.
X is an invalid model. Insert failed.

PL/SQL procedure successfully completed.
25-APR-17USER24-20001ORA-20001: Car price may not be less than zero. Insert failed.
ORA-06512: at "USER24.TRG_I_CAR", line 5
ORA-04088: error during execution of trigger 'USER24.TRG_I_CAR' Insert failed.

PL/SQL procedure successfully completed.

Invalid DealershipId for this car. Insert failed.
```

Line 7 Column 42 | Insert | Modified | Unix/Mac: LF

Oracle SQL Developer : AIT732_2017

Connections

- Connections
  - AIT732_2017
    - Tables (Filtered)
      - CAROPTIONS
      - CARS
        - CARID
        - MODEL
        - PRICE
        - STATUS
        - SALEDATE
        - SALESMAN
        - DEALERSH
      - DEALERSHIPP
      - DEALERSHIPS
      - DEPT
      - EMP

Reports

- All Reports
  - Data Dictionary Repo
  - Data Modeler Report
  - OLAP Reports
  - TimesTen Reports
  - User Defined Reports

AIT732_2017

Worksheet | Query Builder

```
execute insert_car ('S', '44898', '10');

execute insert_car ('X', '44898', '7');

execute insert_car ('S', '0', '7');

execute insert_car ('S', '44898', '100');

select * from Cars;
```

Script Output | Query Result

SQL | All Rows Fetched: 20 in 0.096 seconds

	CA...	MODEL	PRICE	STATUS	SALEDATE	SALESMANID	DEALERSHIPID
1	1	L	50000	Sold	24-APR-17	2	1
2	2	S	49000	Available	(null)	(null)	1
3	3	E	44500	Available	(null)	(null)	1
4	4	L	43498	Available	(null)	(null)	2
5	5	S	43809	Available	(null)	(null)	3
6	6	E	43864	Available	(null)	(null)	4
7	7	E	43564	Available	(null)	(null)	5
8	8	L	45199	Sold	11-APR-17	1	6
9	9	S	44979	Available	(null)	(null)	6
10	10	E	46999	Available	(null)	(null)	7
11	11	L	45814	Available	(null)	(null)	8
12	12	S	46729	Available	(null)	(null)	9
13	13	E	46004	Available	(null)	(null)	10
14	14	L	46559	Available	(null)	(null)	10
15	15	S	48909	Available	(null)	(null)	11
16	16	E	47609	Available	(null)	(null)	11
17	19	S	44898	Available	(null)	(null)	7
18	24	L	50002	Available	(null)	(null)	7
19	27	S	70000	AVAILABLE	(null)	(null)	3
20	29	S	44898	AVAILABLE	(null)	(null)	10

Line 1 Column 41 | Insert | Modified | Unix/Mac: LF

Requirement 4
(Using a procedure and trigger) Given a car ID and a salesman ID, the system should be able to record a car's sale, adding the price of the car to the correct salesman's total (via trigger) and changing the car's status to "sold".

Code to Solve Requirement 4:
This procedure is passed in a car Id and a salesman Id, and it updates the status (to "Sold"), sale date (to SYSDATE), and salesman Id (to salesman who sold the car) in the Cars table. It returns a message if the salesman Id entered is invalid. It also returns a message if the car Id entered is invalid, or if the entered car Id's status is not "Available". The trigger fires when a car's status is updated. It adds the price of the sold car to the correct salesman's total amount sold. It returns an error message if there are any errors.

```
/* PROCEDURE */

Create or Replace Procedure car_sold
(v_carId int, v_salesmanId int)

IS

      v_salesman_id int;      --USED TO VALIDATE INCOMING SALESMANID
      v_count int;      --USED TO VALIDATE INCOMING CARID
      v_ErrorCode number;
      v_ErrorMsg  Varchar2(200);
      v_CurrentUser varchar2(100);
      v_status int :=0;

BEGIN

      BEGIN --BLOCK 1

      /* validate salesmanId and carId */

      select salesmanId
      into v_salesman_id
      from Salesmen
      where salesmanId = v_salesmanId;

      select count(*)
      into v_count
      from Cars
      where carId = v_carId
      and upper(status) = upper('Available');

      if v_count != 1 then
            DBMS_OUTPUT.PUT_LINE('Invalid car Id.');
            v_status := 1;

      end if;
```

```
      EXCEPTION

            WHEN NO_DATA_FOUND or TOO_MANY_ROWS THEN
                  DBMS_OUTPUT.PUT_LINE('Invalid salesman Id.');
                  v_status := 1;

            When OTHERS THEN
            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            v_CurrentUser := USER;
            DBMS_OUTPUT.PUT_LINE( 'An error occurred, the following is
the error message');
            DBMS_OUTPUT.PUT_LINE( TO_CHAR(SYSDATE)  || v_CurrentUser
|| TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg || 'insert failed');
            v_status := 1;

      END; -- BLOCK 1 MODIFICATION

      BEGIN --BLOCK 2

      If v_status = 0 then

      /* DO UPDATE IF NO ERRORS FROM BLOCK 1 */

      UPDATE Cars
      SET status = 'Sold',
      saleDate = SYSDATE,
      salesmanId = v_salesmanId
      Where carId = v_carId;

      COMMIT;     --TRANSACTION CONTROL

      END IF;

      EXCEPTION

      WHEN OTHERS THEN
            Rollback;

            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            DBMS_OUTPUT.PUT_LINE( 'An error occurred, the following is
the error message');
            DBMS_OUTPUT.PUT_LINE( TO_CHAR(SYSDATE)    ||
TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg);

     END; -- BLOCK 2 MODIFICATION

END;
```

```
/* TRIGGER */

Create or Replace Trigger trg_update_totalAmtSold
After Update of status on Cars
For Each Row

DECLARE
      v_salesmanId int;
      v_price number(15,2);

BEGIN
      v_salesmanId := :new.salesmanId;
      v_price := :new.price;

      update Salesmen
      set totalAmtSold = totalAmtSold + v_price
      where salesmanId = v_salesmanId;

EXCEPTION
      When Others then
            raise_application_error
            (-20000,'Update of salesmen totalAmtSold failed.');

End trg_update_totalAmtSold;



/* TEST THE PROCEDURE */

execute car_sold('1','2');

select * from cars;

select * from salesmen;
```

Oracle SQL Developer : AIT732_2017

Connections

- AIT732_2017
  - Tables (Filtered)
    - CAROPTIONS
    - CARS
    - DEALERSHIPF
    - DEALERSHIPS
    - DEPT
    - EMP
    - OPTIONS
    - PRICE_UPD_L
    - RAISE_LOG
    - SALESMEN
  - Views
  - Editioning Views
  - Indexes
  - Packages

Reports

- All Reports
- Data Dictionary Repo
- Data Modeler Report
- OLAP Reports
- TimesTen Reports
- User Defined Reports

AIT732_2017 | CARS

AIT732_2017

Worksheet | Query Builder

```
execute car_sold('1','2');

select * from cars;
```

Script Output | Query Result

SQL | All Rows Fetched: 19 in 0.013 seconds

	CARID	MODEL	PRICE	STATUS	SALEDATE	SALESMANID	DEALERSHIPID
1	1	L	50000	Sold	24-APR-17	2	1
2	2	S	49000	Available	(null)	(null)	1
3	3	E	44500	Available	(null)	(null)	1
4	4	L	43498	Available	(null)	(null)	2
5	5	S	43809	Available	(null)	(null)	3
6	6	E	43864	Available	(null)	(null)	4
7	7	E	43564	Available	(null)	(null)	5
8	8	L	45199	Sold	11-APR-17	1	6
9	9	S	44979	Available	(null)	(null)	6
10	10	E	46999	Available	(null)	(null)	7
11	11	L	45814	Available	(null)	(null)	8
12	12	S	46729	Available	(null)	(null)	9
13	13	E	46004	Available	(null)	(null)	10
14	14	L	46559	Available	(null)	(null)	10
15	15	S	48909	Available	(null)	(null)	11
16	16	E	47609	Available	(null)	(null)	11
17	19	S	44898	Available	(null)	(null)	7
18	24	L	50002	Available	(null)	(null)	7
19	27	S	70000	AVAILABLE	(null)	(null)	3

Click on an identifier with the Command key down to perform "Go to Declaration"

Oracle SQL Developer : AIT732_2017

Connections

- AIT732_2017
  - Tables (Filtered)
    - CAROPTIONS
    - CARS
    - DEALERSHIPF
    - DEALERSHIPS
    - DEPT
    - EMP
    - OPTIONS
    - PRICE_UPD_L
    - RAISE_LOG
    - SALESMEN
  - Views
  - Editioning Views
  - Indexes
  - Packages

Reports

- All Reports
- Data Dictionary Repo
- Data Modeler Report
- OLAP Reports
- TimesTen Reports
- User Defined Reports

AIT732_2017 | CARS

AIT732_2017

Worksheet | Query Builder

```
execute car_sold('1','2');

select * from cars;

select * from salesmen;
```

Script Output | Query Result

SQL | All Rows Fetched: 12 in 0.015 seconds

	SALESMANID	FIRSTNAME	LASTNAME	SALESMANSTREET	SALESMANCITY	SALESMANSTATE	SALESMANZIP	TYPE	TOTALAMTSOLD	DEALERSHIPID
1	1	Ellie	Mi	2215 Briarcliff Ct	Tysons Corner	VA	22182	Part	45199	6
2	2	Courtney	Yu	911 Army Rd	Towson	MD	21204	Full	50000	9
3	3	Emily	Sloane	9 West Maple St	Alexandria	VA	22301	Temp	0	4
4	4	Amanda	Drew	1604 McGuckian St	Annapolis	MD	21401	Full	0	3
5	5	John	Maple	1510 Baylor Ave	Rockville	MD	20850	Full	0	5
6	6	Josh	Cooney	1611 Sanford Rd	Silver Spring	MD	20902	Part	0	2
7	7	David	Batlas	4710 Huron Ave	Suitland	MD	20746	Part	0	1
8	8	Jimmy	Consolas	1006 Magruder Ave	Catonsville	MD	21228	Part	0	7
9	9	Rachelle	McCarthy	106 Wengate Rd	Owings Mills	MD	21117	Temp	0	8
10	10	Kathleen	Simons	9 S Hickory Ave	Bel Air	MD	21014	Part	0	11
11	11	Katie	Brooks	1000 S Hoga Rd	Sterling	VA	20164	Temp	0	10
12	14	Jacob	Chung	7711 Magarity Rd	Falls Church	VA	22043	Temp	0	6

Click on an identifier with the Command key down to perform "Go to Declaration"

Requirement 5
(Using a procedure) Given a state, the system should be able to list all dealerships in that state.

Code to Solve Requirement 5:
This procedure is passed in a state and returns any dealerships in that state. If a state is entered in which no dealerships exist, then a message will be returned.

```
Create or Replace Procedure dealerships_loop
(v_dealershipState char)

IS

      v_count  int; --USED FOR STATE VALIDATION

      CURSOR c_DEALERSHIPS IS
      Select dealershipId, dealershipName, dealershipStreet,
      dealershipCity, dealershipState, dealershipZip
      From Dealerships
      where upper(dealershipState) = upper(v_dealershipState);

BEGIN

      select count(*)
      into v_count
      From Dealerships
      where upper(dealershipState) = upper(v_dealershipState);

      if (v_count = 0 ) then

      DBMS_OUTPUT.PUT_LINE('There are no dealerships in ' ||
TO_CHAR(v_dealershipState) || '.');

      else

            BEGIN
                  --LOOP WITH IMPLICIT VARIABLE DECLARED
                  --AUTOMATIC, OPEN FETCH, CLOSE

                  FOR v_dealerships_data IN c_DEALERSHIPS LOOP

      DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_dealerships_data.dealershipId) ||
      '     ' ||
      TO_CHAR(v_dealerships_data.dealershipName) || '     ' ||
      TO_CHAR(v_dealerships_data.dealershipStreet) || '     ' ||
      TO_CHAR(v_dealerships_data.dealershipCity) || '     ' ||
      TO_CHAR(v_dealerships_data.dealershipState) || '     ' ||
      TO_CHAR(v_dealerships_data.dealershipZip));

                  END LOOP;
```

```
            END;
      end if;
END;


/* TEST THE PROCEDURE */
execute dealerships_loop ('Md');
execute dealerships_loop ('Wy');
```

Oracle SQL Developer : AIT732_2017

Connections

- AIT732_2017
  - Tables (Filtered)
    - CAROPTIONS
    - CARS
    - DEALERSHIPF
    - DEALERSHIPS
    - DEPT
    - EMP
    - OPTIONS
    - PRICE_UPD_L
    - RAISE_LOG
    - SALESMEN
  - Views
  - Editioning Views
  - Indexes
  - Packages

Reports

- All Reports
- Data Dictionary Repo
- Data Modeler Report
- OLAP Reports
- TimesTen Reports
- User Defined Reports

AIT732_2017

Worksheet | Query Builder

```
execute dealerships_loop ('Md');

execute dealerships_loop ('Wy');
```

Query Result | Script Output

Task completed in 0.045 seconds

```
PL/SQL procedure successfully completed.

1     Passport BMW     4730 Auth Place     Suitland     MD     20746-4201
2     BMW of Silver Spring     3211 Automobile Blvd     Silver Spring     MD     20904-4909
3     BMW of Annapolis     31 Old Mill Bottom Rd N     Annapolis     MD     21409-5431
5     BMW of Rockville     1300 Rockville Pike     Rockville     MD     20852-1412
7     BMW of Catonsville     6700 Baltimore National Pike     Baltimore     MD     21228-3909
8     Northwest BMW     9702 Reisterstown Rd     Owings Mills     MD     21117-4120
9     BMW of Towson     700 Kenilworth Dr     Towson     MD     21204-2427
11     BMW of Bel Air     716 Belair Rd     Bel Air     MD     21014-4222

PL/SQL procedure successfully completed.

There are no dealerships in Wy.
```

Click on an identifier with the Command key down to perform "Go to Declaration"

Requirement 6
(Using a procedure) Given a valid option, the system should be able to list all of the cars on the lot that have that option.

Code to Solve Requirement 6:
This procedure is passed in an option name (or part of an option name) and returns all cars on the lot that have an option name like that option name. If there are no option names in the database like the option name entered, a message is returned.

```
Create or Replace Procedure cars_loop
(v_optionName varchar)

IS

      v_count  int; --USED FOR OPTION VALIDATION

      CURSOR c_CARS IS
      Select Cars.carId, Cars.model, Cars.price, Cars.status,
      Cars.saleDate, Cars.salesmanId,
      Cars.dealershipId, O.optionName
      From Cars, Options O, CarOptions CO
      where Cars.carId = CO.carId
      AND CO.optionId = O.optionId
      AND upper(optionName) like '%' || upper(v_optionName) || '%'
      ORDER BY dealershipId;

BEGIN

      select count(*)
      into v_count
      From Options
      where upper(optionName) like '%' || upper(v_optionName) || '%';

      if (v_count = 0 ) then

      DBMS_OUTPUT.PUT_LINE('There are no options like ' ||
      TO_CHAR(v_optionName) || '.');

      else

            BEGIN
            --LOOP WITH IMPLICIT VARIABLE DECLARED
            --AUTOMATIC, OPEN FETCH, CLOSE

                  FOR v_cars_data IN c_CARS LOOP

                  DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_cars_data.carId) || '
' ||
                  TO_CHAR(v_cars_data.model) || '     ' ||
```

```
                  TO_CHAR(v_cars_data.price) || '     ' ||
                  TO_CHAR(v_cars_data.status) || '     ' ||
                  TO_CHAR(v_cars_data.saleDate) || '     ' ||
                  TO_CHAR(v_cars_data.salesmanId) || '     ' ||
                  TO_CHAR(v_cars_data.dealershipId) || '     ' ||
                  TO_CHAR(v_cars_data.optionName));

                  END LOOP;

            END;

      End if;

END;



/* TEST THE PROCEDURE */

execute cars_loop ('ipod');

execute cars_loop ('wheel');
```

Oracle SQL Developer : AIT732_2017

Connections

- AIT732_2017
  - Tables (Filtered)
    - CAROPTIONS
    - CARS
    - DEALERSHIPF
    - DEALERSHIPS
    - DEPT
    - EMP
    - OPTIONS
    - PRICE_UPD_L
    - RAISE_LOG
    - SALESMEN
  - Views
  - Editioning Views
  - Indexes
  - Packages

Reports

- All Reports
- Data Dictionary Repo
- Data Modeler Report
- OLAP Reports
- TimesTen Reports
- User Defined Reports

AIT732_2017

Worksheet | Query Builder

```
execute cars_loop ('ipod');

execute cars_loop ('wheel');
```

Script Output

Task completed in 0.061 seconds

```
PL/SQL procedure successfully completed.

8     L     45199     Sold     11-APR-17     1     6     iPod/MP3 Input
11     L     45814     Available               8     iPod/MP3 Input

PL/SQL procedure successfully completed.

There are no options like wheel.
```

Click on an identifier with the Command key down to perform "Go to Declaration"

Requirement 7
(Using a procedure) Given a valid car id and a price, update the car to the new price. The price of the car cannot be zero.

Code to Solve Requirement 7:
This procedure is passed in a car Id and price and updates the price of that car Id to the new price. If an incorrect car Id is entered or if the price entered equals 0, then an error message is returned.

```
create or replace procedure update_price
(v_carId int, new_price number)
IS
      v_ErrorCode number;
      v_ErrorMsg  varchar2(200);
      v_old_price  number(7,2);
      e_invalid_price EXCEPTION;
      v_status    int := 0 ;

BEGIN

   BEGIN  --BLOCK 1  SELECT

      /*  VALIDATE PRICE  */

      select price
      into v_old_price
      from Cars
      where carId = v_carId;

      IF new_price = 0 THEN
            RAISE e_invalid_price;
      END IF;

      EXCEPTION

        WHEN NO_DATA_FOUND or TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('CarId not valid.');
            v_status := 1;
        WHEN e_invalid_price THEN
            DBMS_OUTPUT.PUT_LINE('Price not valid.');
            v_status := 1;
        WHEN OTHERS THEN
            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            DBMS_OUTPUT.PUT_LINE( 'An error occurred, the following is
the error message');
            DBMS_OUTPUT.PUT_LINE(TO_CHAR(SYSDATE)    ||
TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg);
            v_status := 1;
```

```
   END;   --END BLOCK 1 OF SELECT


   BEGIN  --BLOCK 2 MODIFICATIONS

    IF v_status = 0 then

      /* DO UPDATE IF ALL WELL  */

      UPDATE Cars set price = new_price
      where carId = v_carId;

      COMMIT;     --TRANSACTION CONTROL

    END IF;

        EXCEPTION

      WHEN OTHERS THEN
            Rollback;
            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            DBMS_OUTPUT.PUT_LINE( 'An error occurred, the following is
the error message');
            DBMS_OUTPUT.PUT_LINE(TO_CHAR(SYSDATE)    ||
TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg);

     END; -- BLOCK 2 MODIFICATION

END;


/* TEST THE PROCEDURE */

execute update_price ('1', '0');

execute update_price ('100', '50000');

execute update_price ('1', '50000');

select * from Cars;
```

Oracle SQL Developer : AIT732_2017

Connections

- Connections
  - AIT732_2017
    - Tables (Filtered
      - CAROPTION
      - CARS
      - DEALERSHIP
      - DEALERSHIP
      - DEPT
      - EMP
      - OPTIONS
      - RAISE_LOG
      - SALESMEN
    - Views
    - Editioning View
    - Indexes
    - Packages

Reports

- All Reports
  - Data Dictionary Repo
  - Data Modeler Report
  - OLAP Reports
  - TimesTen Reports
  - User Defined Report

AIT732_2017 | CARS

AIT732_2017

Worksheet | Query Builder

```
execute update_price ('1', '0');

execute update_price ('100', '50000');

execute update_price ('1', '50000');
```

Query Result | Script Output

Task completed in 0.037 seconds

```
PL/SQL procedure successfully completed.

Price not valid.

PL/SQL procedure successfully completed.

CarId not valid.

PL/SQL procedure successfully completed.
```

Click on an identifier with the Command key down to perform "Go to Declaration"

Line 5 Column 37 | Insert | Modified | Unix/Mac: LF

Oracle SQL Developer : AIT732_2017

Connections

- Connections
  - AIT732_2017
    - Tables (Filtered
      - CAROPTION
      - CARS
      - DEALERSHIP
      - DEALERSHIP
      - DEPT
      - EMP
      - OPTIONS
      - RAISE_LOG
      - SALESMEN
    - Views
    - Editioning View
    - Indexes
    - Packages

Reports

- All Reports
  - Data Dictionary Repo
  - Data Modeler Report
  - OLAP Reports
  - TimesTen Reports
  - User Defined Report

AIT732_2017 | CARS

AIT732_2017

Worksheet | Query Builder

```
execute update_price ('1', '0');

execute update_price ('100', '50000');

execute update_price ('1', '50000');

select * from Cars;
```

Script Output | Query Result

SQL | All Rows Fetched: 17 in 0.015 seconds

	CARID	MODEL	PRICE	STATUS	SALEDATE	SALESMANID	DEALERSHIPID
1	1	L	50000	Available	(null)	(null)	1
2	2	S	49000	Available	(null)	(null)	1
3	3	E	44500	Available	(null)	(null)	1
4	4	L	43498	Available	(null)	(null)	2
5	5	S	43809	Available	(null)	(null)	3
6	6	E	43864	Available	(null)	(null)	4
7	7	E	43564	Available	(null)	(null)	5
8	8	L	45199	Sold	11-APR-17	1	6
9	9	S	44979	Available	(null)	(null)	6
10	10	E	46999	Available	(null)	(null)	7
11	11	L	45814	Available	(null)	(null)	8
12	12	S	46729	Available	(null)	(null)	9
13	13	E	46004	Available	(null)	(null)	10
14	14	L	46559	Available	(null)	(null)	10
15	15	S	48909	Available	(null)	(null)	11
16	16	E	47609	Available	(null)	(null)	11
17	19	S	44898	Available	(null)	(null)	7

Click on an identifier with the Command key down to perform "Go to Declaration" | Line 7 Column 20 | Insert | Modified | Unix/Mac: LF

Requirement 8
[EXTRA CREDIT]: Write a trigger to log the information when the price of a car is updated. When the price of a car is changed, capture in a log, the old and new prices, the person who updated the price, and the date.

Code to Solve Requirement 8:
This trigger records the car Id, old car price, new car price, user, and date anytime the price of a car is updated in the Cars table.

```
create table price_upd_log
(carId int, old_price number (15,2), new_price number (15,2), the_user
varchar(150), the_date date);

CREATE OR REPLACE TRIGGER trg_upd_price
AFTER UPDATE OF price
      ON Cars
      FOR EACH ROW

DECLARE
      v_user varchar(100);
      v_ErrorCode number;
      v_ErrorMsg  varchar2(200);
BEGIN
      select user INTO v_user
      from Dual;

      INSERT INTO price_upd_log
      (carId, old_price, new_price, the_user, the_date)
      VALUES
      (:new.carId, :old.price, :new.price, v_user, SYSDATE);

EXCEPTION
   WHEN OTHERS THEN
            Rollback;
            v_ErrorCode := SQLCODE;
            v_ErrorMsg := substr(SQLERRM,1,200);
            DBMS_OUTPUT.PUT_LINE( 'An error occurred, the following is
the error message');
            DBMS_OUTPUT.PUT_LINE( TO_CHAR(SYSDATE)    ||
TO_CHAR(v_ErrorCode)  ||  v_ErrorMsg);

END;


/* TEST THE PROCEDURE */

execute update_price ('1', '50000');

select * from price_upd_log;
```

```
execute update_price ('1', '60000');

select * from price_upd_log;
```

Oracle SQL Developer : AIT732_2017

Connections

- Connections
  - AIT732_2017
    - Tables (Filtered)
      - CAROPTIONS
      - CARS
        - CARID
        - MODEL
        - PRICE
        - STATUS
        - SALEDATE
        - SALESMANID
        - DEALERSHIPID
      - DEALERSHIPPHONES
      - DEALERSHIPS
        - DEALERSHIPID
        - DEALERSHIPNAME

Reports

- All Reports
  - Data Dictionary Reports
  - Data Modeler Reports
  - OLAP Reports
  - TimesTen Reports
  - User Defined Reports

AIT732_2017

Worksheet | Query Builder

```
execute update_price ('1', '50000');

select * from price_upd_log;

execute update_price ('1', '60000');

select * from price_upd_log;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.046 seconds

	CARID	OLD_PRICE	NEW_PRICE	THE_USER	THE_DATE
1	1	50000	50000	USER24	19-APR-17
2	1	50000	60000	USER24	19-APR-17

Click on an identifier with the Command key down to perform "Go to Declaration"

Line 7 Column 29 | Insert | Modified | Unix/Mac: LF