DEPARTMENT OF COMPUTER SCIENCES TECHNOLOGY

Name of the laboratory: Database Systems Lab
Subject code: 14CS2012

List of Exercises

#	Date	Name of the Experiment	Marks	Signature
1	18.07.2016	Creating table and setting constraint using DDL and DML commands		
2	25.07.2016	Managing Table using DML, DCL and TCL commands		
3	01.08.2016	Aggregate and Built–in Functions, Group By & Order By		
4	22.08.2016	Set Operations and Joins		
5	29.08.2016	Sub Queries and View		
6	19.09.2016	Application Development using JDBC connectivity		
7	03.10.2016	PL/SQL Triggers		
8	10.10.2016	PL/SQL- Functions and Procedures		
9	17.10.2016	Creation of other Database Objects		
10	31.10.2016	Web Application using PHP and MySQL		

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228
DATE: 11-08-16
EXPERIMENT NO-1

Video URL: https://youtu.be/1hfrKDK9kCs?list=PLRS_VYrnFL6lj53kJl8EaWDEsoE-8Epkj

AIM:

Create tables using DDL and DCL commands.

DESCRIPTION:

Data Definition Language (DDL) - These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.

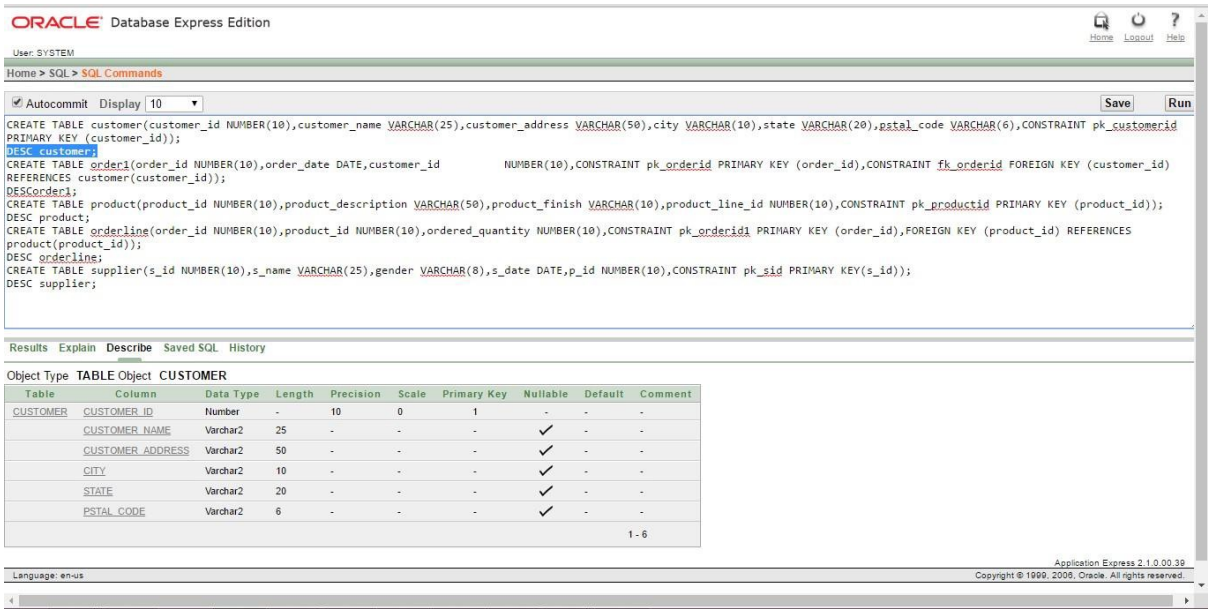
Data Control Language (DCL) - These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE.

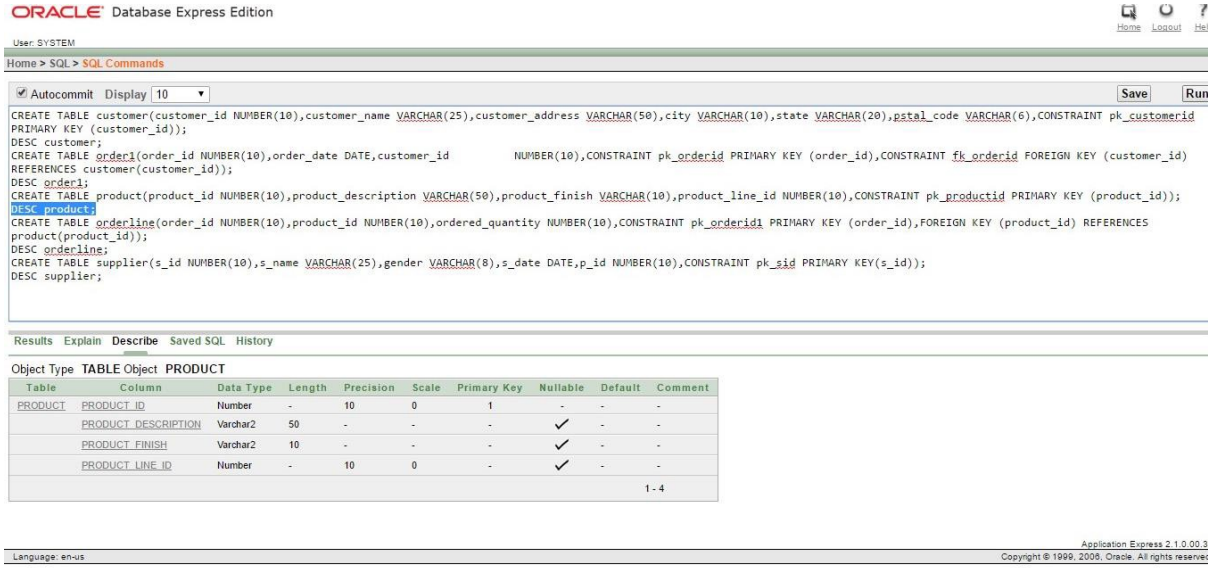
1.Create the above tables, with primary key constraints.
2.Add foreign key reference to the customer_id attribute.

```
CREATE TABLE customer(customer_id NUMBER(10),customer_name
VARCHAR(25),customer_address VARCHAR(50),city VARCHAR(10),state
VARCHAR(20),pstal_code VARCHAR(6),CONSTRAINT pk_customerid PRIMARY KEY
(customer_id));
DESC customer;
CREATE TABLE orders(order_id NUMBER(10),order_date DATE,customer_id
NUMBER(10),CONSTRAINT pk_orderid PRIMARY KEY (order_id),CONSTRAINT
fk_orderid FOREIGN KEY (customer_id) REFERENCES customer(customer_id));
DESCorders;
CREATE TABLE product(product_id NUMBER(10),product_description
VARCHAR(50),product_finish VARCHAR(10),product_line_id
NUMBER(10),CONSTRAINT pk_productid PRIMARY KEY (product_id));
DESC product;
CREATE TABLE orderline(order_id NUMBER(10),product_id
NUMBER(10),ordered_quantity
NUMBER(10),CONSTRAINT pk_orderid1 PRIMARY KEY (order_id),FOREIGN KEY
(product_id) REFERENCES product(product_id));
```

DESC orderline;
CREATE TABLE supplier(s_id NUMBER(10),s_name VARCHAR(25),gender VARCHAR(8),s_date DATE,p_id NUMBER(10),CONSTRAINT pk_sid PRIMARYKEY(s_id));

DESC supplier;





ORACLE Database Express Edition

Home Logout Help

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE TABLE customer(customer_id NUMBER(10),customer_name VARCHAR(25),customer_address VARCHAR(50),city VARCHAR(10),state VARCHAR(20),pstal_code VARCHAR(6),CONSTRAINT pk_customerid
PRIMARY KEY (customer_id));
DESC customer;
CREATE TABLE order1(order_id NUMBER(10),order_date DATE,customer_id          NUMBER(10),CONSTRAINT pk_orderid PRIMARY KEY (order_id),CONSTRAINT fk_orderid FOREIGN KEY (customer_id)
REFERENCES customer(customer_id));
DESC order1;
CREATE TABLE product(product_id NUMBER(10),product_description VARCHAR(50),product_finish VARCHAR(10),product_line_id NUMBER(10),CONSTRAINT pk_productid PRIMARY KEY (product_id));
DESC product;
CREATE TABLE orderline(order_id NUMBER(10),product_id NUMBER(10),ordered_quantity NUMBER(10),CONSTRAINT pk_orderid1 PRIMARY KEY (order_id),FOREIGN KEY (product_id) REFERENCES
product(product_id));
DESC orderline;
CREATE TABLE supplier(s_id NUMBER(10),s_name VARCHAR(25),gender VARCHAR(8),s_date DATE,p_id NUMBER(10),CONSTRAINT pk_sid PRIMARY KEY(s_id));
DESC supplier;
```

Results Explain Describe Saved SQL History

Object Type TABLE Object ORDER1

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDER1	ORDER_ID	Number	-	10	0	1	-	-	-
	ORDER_DATE	Date	7	-	-	-	✓	-	-
	CUSTOMER_ID	Number	-	10	0	-	✓	-	-

1 - 3

Application Express 2.1.0.00.39

Language: en-us

ORACLE Database Express Edition

Home Logout Help

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

CREATE TABLE customer(customer_id NUMBER(10),customer_name VARCHAR(25),customer_address VARCHAR(50),city VARCHAR(10),state VARCHAR(20),pstal_code VARCHAR(6),CONSTRAINT pk_customerid
PRIMARY KEY (customer_id));
DESC customer;
CREATE TABLE order1(order_id NUMBER(10),order_date DATE,customer_id   NUMBER(10),CONSTRAINT pk_orderid PRIMARY KEY (order_id),CONSTRAINT fk_orderid FOREIGN KEY (customer_id)
REFERENCES customer(customer_id));
DESC order1;
CREATE TABLE product(product_id NUMBER(10),product_description VARCHAR(50),product_finish VARCHAR(10),product_line_id NUMBER(10),CONSTRAINT pk_productid PRIMARY KEY (product_id));
DESC product;
CREATE TABLE orderline(order_id NUMBER(10),product_id NUMBER(10),ordered_quantity NUMBER(10),CONSTRAINT pk_orderid1 PRIMARY KEY (order_id),FOREIGN KEY (product_id) REFERENCES
product(product_id));
DESC orderline;
CREATE TABLE supplier(s_id NUMBER(10),s_name VARCHAR(25),gender VARCHAR(8),s_date DATE,p_id NUMBER(10),CONSTRAINT pk_sid PRIMARY KEY(s_id));
DESC supplier;

Results Explain Describe Saved SQL History

Object Type TABLE Object ORDERLINE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERLINE	ORDER_ID	Number	-	10	0	1	-	-	-
	PRODUCT_ID	Number	-	10	0	-	✓	-	-
	ORDERED_QUANTITY	Number	-	10	0	-	✓	-	-

1 - 3

Application Express 2.1.0.00.39

Language: en-us

Copyright © 1999, 2006, Oracle. All rights reserved.

ORACLE Database Express Edition

Home Logout Help

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

CREATE TABLE customer(customer_id NUMBER(10),customer_name VARCHAR(25),customer_address VARCHAR(50),city VARCHAR(10),state VARCHAR(20),pstal_code VARCHAR(6),CONSTRAINT pk_customerid
PRIMARY KEY (customer_id));
DESC customer;
CREATE TABLE order1(order_id NUMBER(10),order_date DATE,customer_id   NUMBER(10),CONSTRAINT pk_orderid PRIMARY KEY (order_id),CONSTRAINT fk_orderid FOREIGN KEY (customer_id)
REFERENCES customer(customer_id));
DESC order1;
CREATE TABLE product(product_id NUMBER(10),product_description VARCHAR(50),product_finish VARCHAR(10),product_line_id NUMBER(10),CONSTRAINT pk_productid PRIMARY KEY (product_id));
DESC product;
CREATE TABLE orderline(order_id NUMBER(10),product_id NUMBER(10),ordered_quantity NUMBER(10),CONSTRAINT pk_orderid1 PRIMARY KEY (order_id),FOREIGN KEY (product_id) REFERENCES
product(product_id));
DESC orderline;
CREATE TABLE supplier(s_id NUMBER(10),s_name VARCHAR(25),gender VARCHAR(8),s_date DATE,p_id NUMBER(10),CONSTRAINT pk_sid PRIMARY KEY(s_id));
DESC supplier;

Results Explain Describe Saved SQL History

Object Type TABLE Object SUPPLIER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SUPPLIER	S_ID	Number	-	10	0	1	-	-	-
	S_NAME	Varchar2	25	-	-	-	✓	-	-
	GENDER	Varchar2	8	-	-	-	✓	-	-
	S_DATE	Date	7	-	-	-	✓	-	-
	P_ID	Number	-	10	0	-	✓	-	-

1 - 5

Application Express 2.1.0.00.39

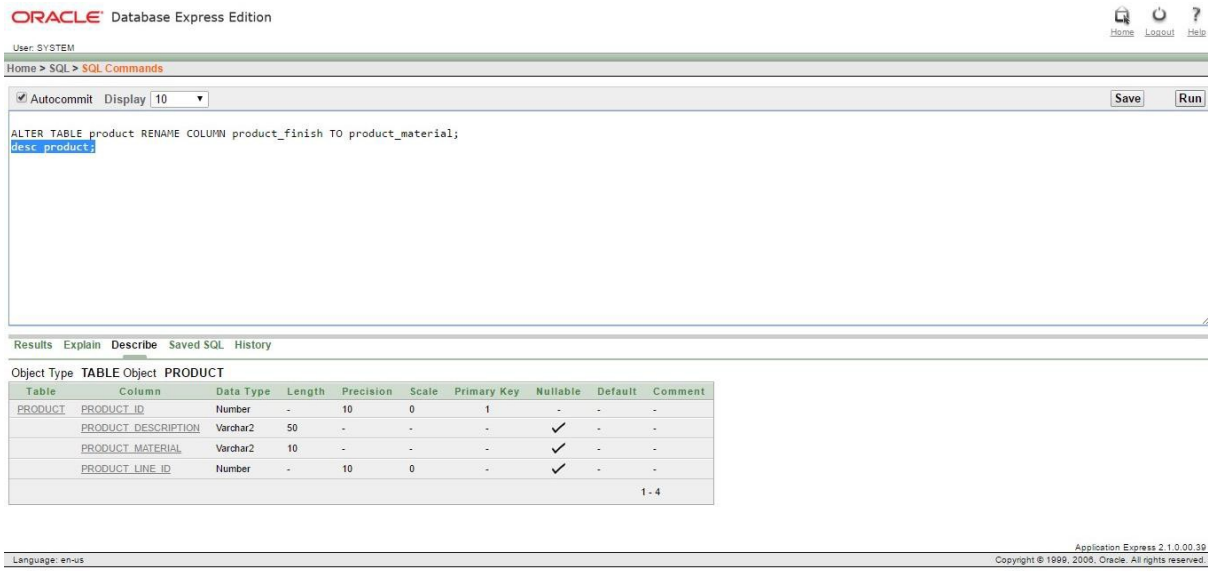
Language: en-us

Copyright © 1999, 2006, Oracle. All rights reserved.

3.Modify the attribute product_finish in the product table to product_material.

ALTER TABLE product RENAME COLUMN product_finish TO product_material;

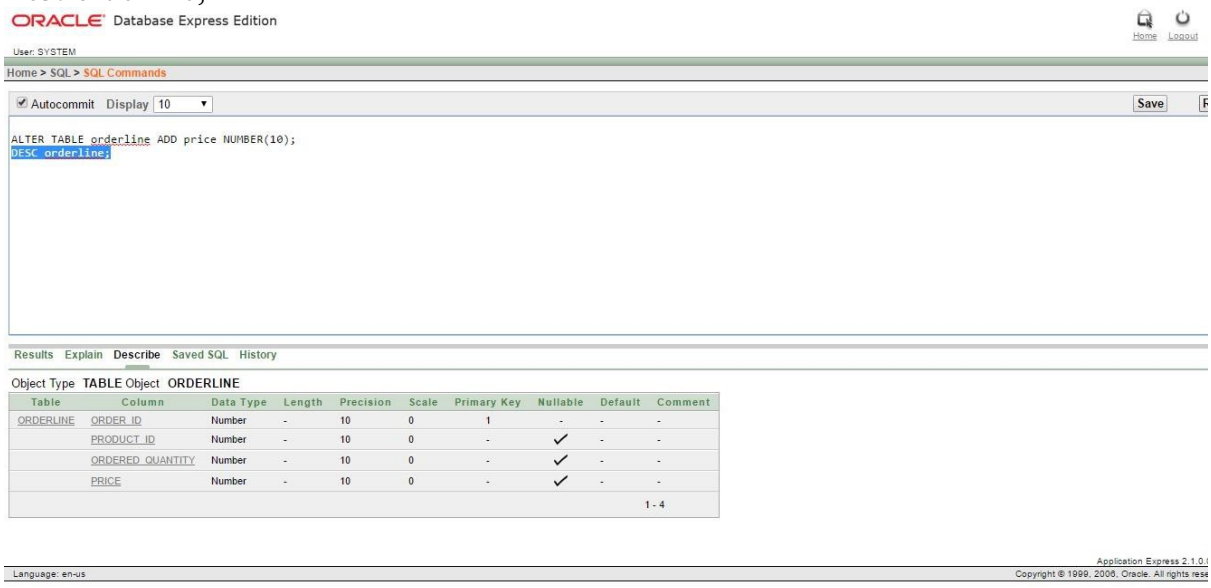
DESC product;



4.Add an attribute called price in the Orderline table

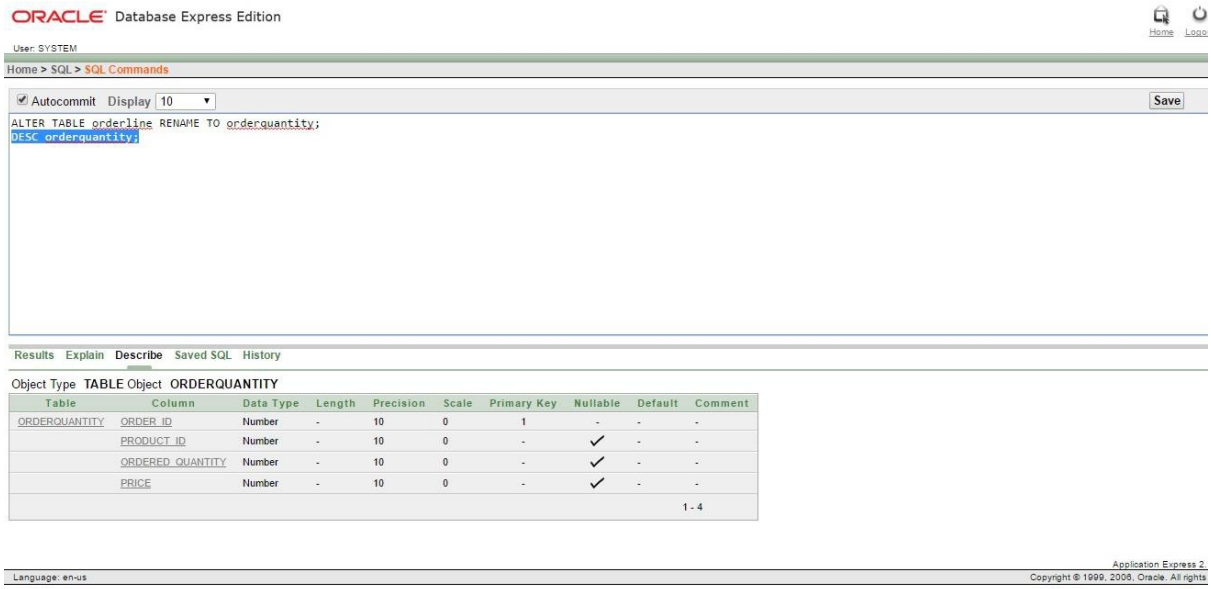
ALTER TABLE orderline ADD price NUMBER(10);

Desc orderline;



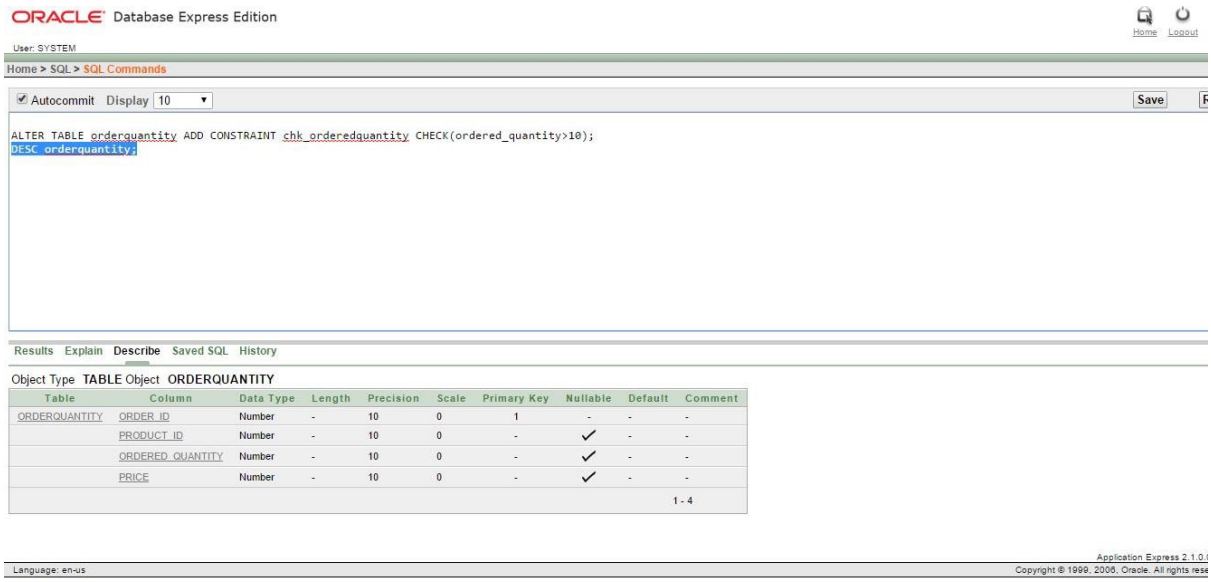
5.Rename orderline to orderquantity

ALTER TABLE orderline RENAME TO orderquantity;
DESC orderquantity;

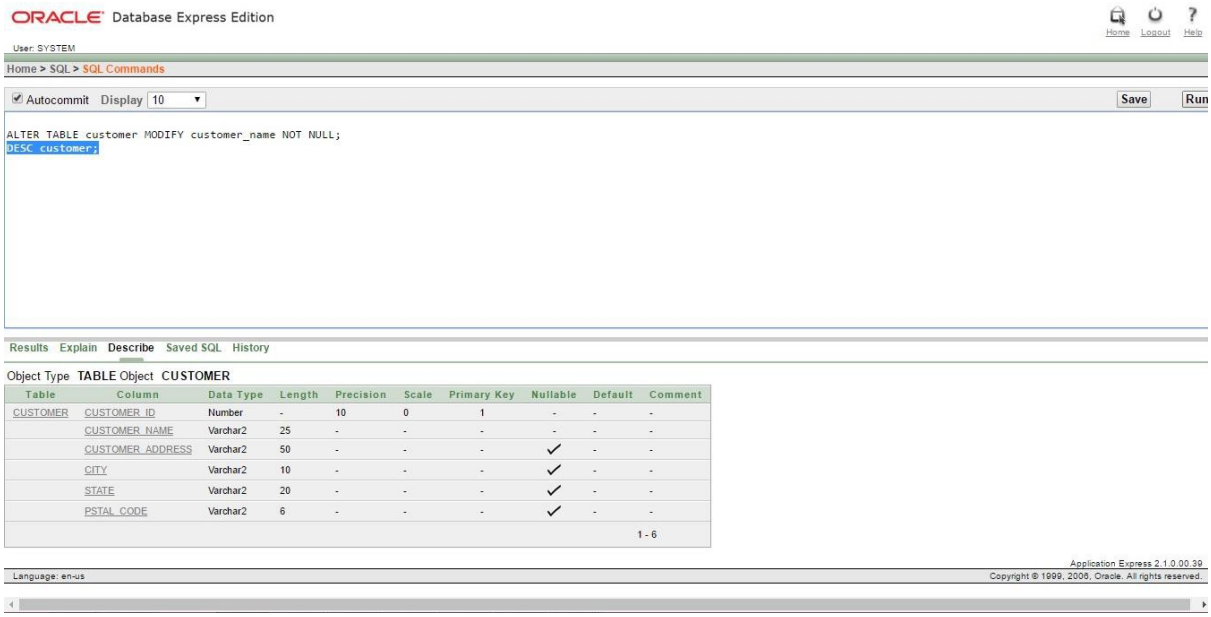


6.Add a check constraint to ordered_quantity column and check that quantity is above 10

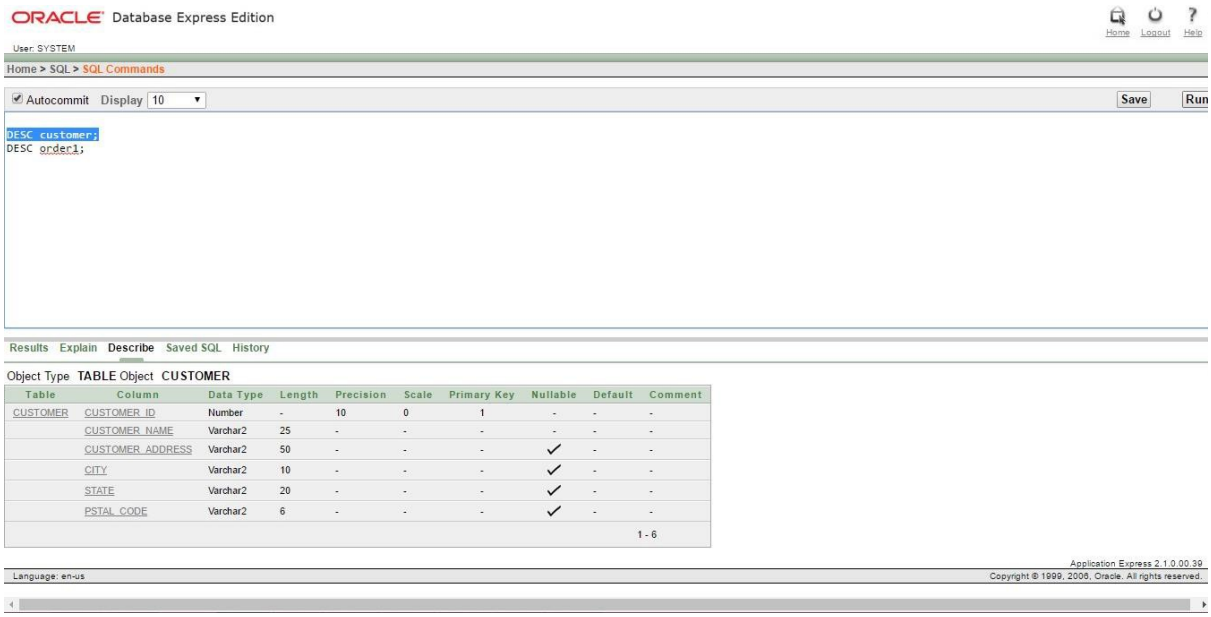
ALTER TABLE orderquantity ADD CONSTRAINT chk_orderedquantity
CHECK(ordered_quantity>10);
DESC orderquantity;

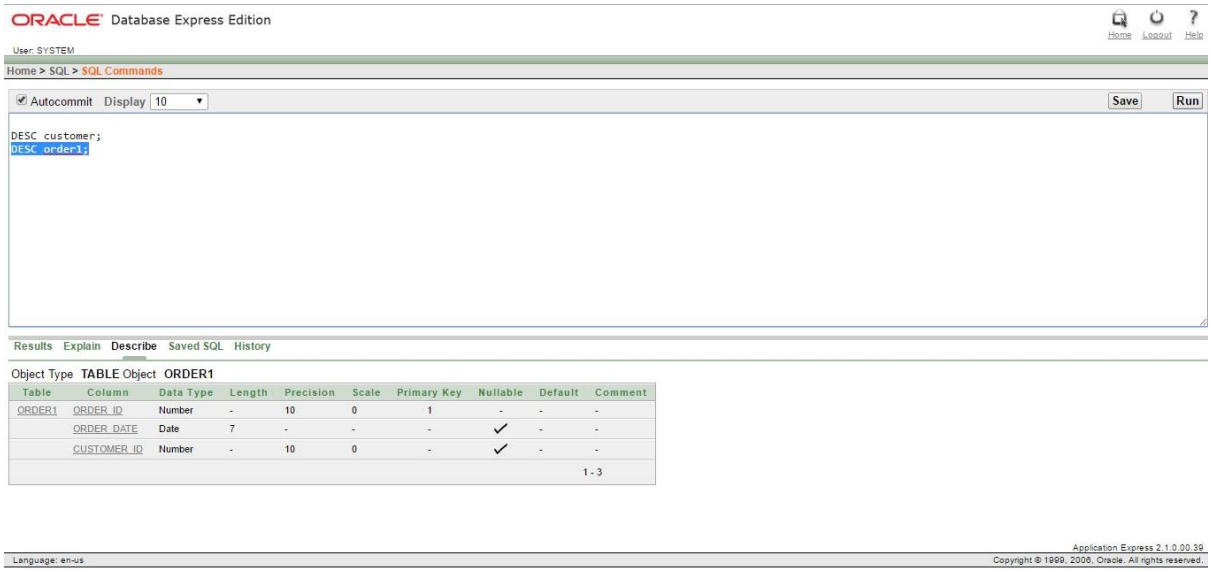


7.Add a not null constraint to customer_name attribute.
ALTER TABLE customer MODIFY customer_name NOT NULL; DESC customer;



8. the structure of the customer and order table
DESC customer;
DESC orders;





9.Insert the above values in to the corresponding table.

```
INSERT INTO customer VALUES (1, 'John Doe', '392 Sunset Blvd.', 'New York',
'NT','10059');
INSERT INTO customer VALUES (2, 'Mary Smith', '6900 Main St.', 'SanFrancis',
'CA','94032');
INSERT INTO customer VALUES (3, 'Richard Newman', '2040 Riverside Rd.', 'San Diego',
'CA','92010');
INSERT INTO customer VALUES (4, 'Cathy Cook', '4010 Speedway', 'Tucson',
'AZ','85719');
INSERT INTO orders VALUES (100,TO_DATE('01-OCT-2014','DD-MONYYYY'),1);
INSERT INTO orders VALUES (101, TO_DATE('01-OCT-2014','DD-MON-YYYY'),
2);
INSERT INTO orders VALUES (102, TO_DATE('02-OCT-2014','DD-MON-YYYY'),
3);
INSERT INTO orders VALUES (103, TO_DATE('03-OCT-2014','DD-MON-YYYY'),
2);
INSERT INTO orders VALUES (104, TO_DATE('10-OCT-2014','DD-MON-YYYY'),
1);
INSERT INTO orders VALUES (105, TO_DATE('10-OCT-2014','DD-MON-YYYY'),
4);
INSERT INTO orders VALUES (106, TO_DATE('10-OCT-2014','DD-MON-YYYY'),
2);
INSERT INTO orders VALUES (107,TO_DATE('10-OCT-2014','DD-MON-YYYY'),
1);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(1000, 'Office Desk', 'Cherry',10);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(1001, 'Manager"s Desk', 'Red Oak',10);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(2000, 'Office Chair', 'Cherry',20);
```

```
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(2001, 'Manager"s Desk', 'NaturalOak',20);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(3000, 'Book Shelf', 'NaturalAsh',30);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(3001, 'Duplex Book Shelf', 'White Ash',30);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(4000, 'Table Lamp', 'NaturalAsh',40);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(4001, 'Duplex Table Lamp', 'White Ash',40);
INSERT INTO
product(product_id,product_description,product_material,product_line_id)VALUES
(9999, 'Keyboard', 'Plastic',50);
INSERT INTO orderquantity VALUES (100, 4000, 10, 10000);
INSERT INTO orderquantity VALUES (101, 1000, 20, 15000);
INSERT INTO orderquantity VALUES (101, 2000, 20, 20000);
INSERT INTO orderquantity VALUES (102, 3000, 15, 23500); INSERT INTO
orderquantity VALUES (102, 2000, 12, 34560);
INSERT INTO orderquantity VALUES (103, 4001, 14, 1000);
INSERT INTO orderquantity VALUES (104, 2000, 10, 2000);
INSERT INTO orderquantity VALUES (105, 3001, 20, 30005);
INSERT INTO orderquantity VALUES (106, 3000, 12, 20500);
INSERT INTO orderquantity VALUES (106, 4000, 11, 30000);
INSERT INTO orderquantity VALUES (107, 4001, 5, 40100);
INSERT INTO supplier VALUES (1,'aaa','M' ,TO_DATE('11-NOV-2014','DD-
MONYYYY'),1000);
INSERT INTO supplier VALUES (2, 'bbb', 'F',TO_DATE('01-NOV-2014','DD-MONYYYY'),
2000);
INSERT INTO supplier VALUES (2, 'bbb', 'F',TO_DATE('01-NOV-2014','DD-MONYYYY'),
2001);
INSERT INTO supplier VALUES (3,'ccc','M',TO_DATE('20-OCT-2014','DD-
MONYYYY'),9999);
INSERT INTO supplier VALUES (4,'ddd','F',TO_DATE('11-NOV-2014','DD-
MONYYYY'),4001);
INSERT INTO supplier VALUES (4, 'ddd', 'F',TO_DATE('05-NOV-2014','DD-MONYYYY'),
4000);
INSERT INTO supplier VALUES (5, 'eee','M',TO_DATE('13-NOV-2014','DD-MONYYYY'),
1001);
```

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
desc Customer;
desc orderquantity;


Insert into Customer values(1,'John Doe', '392 Sunset Blvd.', 'New York', 'NT',10059);
Insert into Customer values(2, 'Mary Smith', '6900 Main St.', 'San Francisco', 'CA','94032');
Insert into Customer values(3, 'Richard Newman', '2040 Riverside Rd.', 'San Diego', 'CA','92010');
Insert into Customer values(4, 'Cathy Cook', '4010 Speedway', 'Tucson', 'AZ','85719');

select * from customer;

Insert into Orderquantity values(100, 4000, 10,1500);
Insert into Orderquantity values(101, 1000, 20,2000);
Insert into Orderquantity values(101, 2000, 20,3466);
```

Results Explain Describe Saved SQL History

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CITY	STATE	POSTAL_CODE
1	John Doe	392 Sunset Blvd.	New York	NT	10059
2	Mary Smith	6900 Main St.	San Francisco	CA	10032
3	Richard Newman	2040 Riverside Rd.	San Diego	CA	92010
4	Cathy Cook	4010 Speedway	Tucson	AZ	85719

4 rows returned in 0.00 seconds

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
Insert into Orderquantity values(100, 4000, 10,1500);
Insert into Orderquantity values(101, 1000, 20,2000);
Insert into Orderquantity values(101, 2000, 20,3466);
Insert into Orderquantity values(102, 3000, 15,1999);
Insert into Orderquantity values(102, 2000, 12,1000);
Insert into Orderquantity values(103, 4001, 14,2000);
Insert into Orderquantity values(104, 2000, 10,3477);
Insert into Orderquantity values(105, 3001, 20,3667);
Insert into Orderquantity values(106, 3000, 12,4500);
Insert into Orderquantity values(106, 4000, 11,1000);
Insert into Orderquantity values(107, 4001, 5,5670);

select * from orderquantity;


Insert into product values(1000, 'Office Desk', 'Cherry', 95.0);
```

Results Explain Describe Saved SQL History

ORDER_ID	PRODUCT_ID	ORDER_QTY	PRICE
101	1000	20	2000
102	3000	15	1999
103	4001	14	2000
105	3001	20	3667
106	3000	12	4500

5 rows returned in 0.00 seconds

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
Insert into product values(2001, 'Manager''s Desk', 'Natural Oak', 129.0);
Insert into product values(3000, 'Book Shelf', 'Natural Ash', 35.0);
Insert into product values(3001, 'Duplex Book Shelf', 'White Ash', 80.0);
Insert into product values(4000, 'Table Lamp', 'Natural Ash', 15.0);
Insert into product values(4001, 'Duplex Table Lamp', 'White Ash', 40.0);
Insert into product values(9999, 'Keyboard', 'Plastic', 20.0);

select * from product;

Insert into Orders values(100, '01-OCT-14', 1 );
Insert into Orders values( 101, '01-OCT-14', 2);
Insert into Orders values( 102, '02-OCT-14', 3);
Insert into Orders values( 103, '03-OCT-14', 2);
Insert into Orders values(104, '10-OCT-14', 1 );
Insert into Orders values(105, '10-OCT-14', 4 );
Insert into Orders values(106, '10-OCT-14', 2 );
```

Results Explain Describe Saved SQL History

PRODUCT_ID	PRODUCT_DESCRIPTION	PRODUCT_MATERIAL	PRODUCT_LINE_ID	PRODUCT_QTY
1000	Office Desk	Cherry	95	12
1001	Manager's Desk	Red Oak	199	23
2000	Office Chair	Cherry	75	32
2001	Manager's Desk	Natural Oak	129	23
3000	Book Shelf	Natural Ash	35	32
3001	Duplex Book Shelf	White Ash	80	10
4000	Table Lamp	Natural Ash	15	32
4001	Duplex Table Lamp	White Ash	40	11
9999	Keyboard	Plastic	20	12

9 rows returned in 0.00 seconds

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

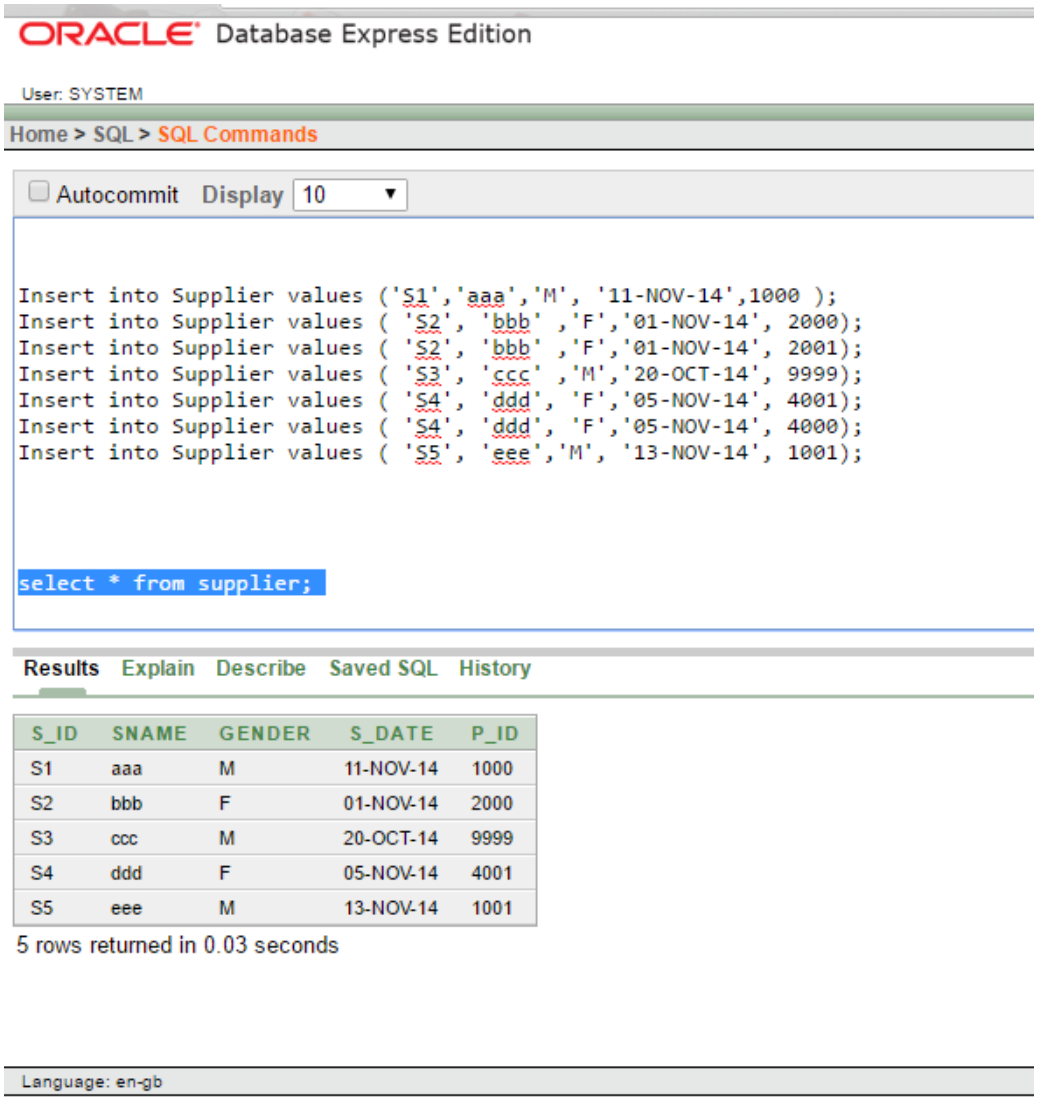
Autocommit Display 10

```
Insert into Orders values(100, '01-OCT-14', 1 );
Insert into Orders values( 101, '01-OCT-14', 2);
Insert into Orders values( 102, '02-OCT-14', 3);
Insert into Orders values( 103, '03-OCT-14', 2);
Insert into Orders values(104, '10-OCT-14', 1 );
Insert into Orders values(105, '10-OCT-14', 4 );
Insert into Orders values(106, '10-OCT-14', 2 );
Insert into Orders values( 107, '10-OCT-14', 1);

select * from orders;

Insert into Supplier values ('S1','aaa','M', '11-NOV-14',1000 );
```

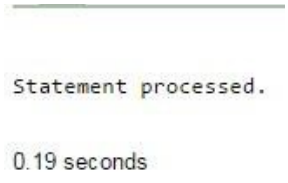
Results Explain Describe Saved SQL History

ORDER_ID	ORDER_DATE	CUSTOMERID
100	01-OCT-14	1
101	01-OCT-14	2
102	02-OCT-14	3
103	03-OCT-14	2
105	10-OCT-14	4
106	10-OCT-14	2
107	10-OCT-14	1
108	10-NOV-14	1

8 rows returned in 0.02 seconds



10.Store the data permanently in stable storage
COMMIT;



11.Copy the customer table to a new table called 'customer1'. Rename the new table as customer1.
SELECT * FROM supplier;
CREATE TABLE customer1 AS (SELECT * FROM customer);
SELECT * FROM customer1;

Results Explain Describe Saved SQL History

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CITY	STATE	PSTAL_CODE
1	John Doe	392 Sunset Blvd.	New York	NT	10059
2	Mary Smith	6900 Main St.	SanFrancis	CA	94032
3	Richard Newman	2040 Riverside Rd.	San Diego	CA	92010
4	Cathy Cook	4010 Speedway	Tucson	AZ	85719

12.Grant select privilege to user1 on Order table
13.Try to insert values into Order table from user1 and state what happen
14.Grant insert privilege on Order table to user1
15.Try to delete values from Order table from user1 and state what happens
16.Try to insert values into Order table from user1 and state what happens.
18 .revoke all privileges from user1 on Order table.

```
CREATE USER user1 IDENTIFIED BY ab;
GRANT select ON orders TO user1;
GRANT insert ON orders TO user1;
REVOKE ALL ON orders FROM user1;
```

19.Remove all data's from the customer1 table.
20.Drop the structure of the customer1 table.

```
TRUNCATE TABLE customer1;
 DROP TABLE customer1;
```

```
Table truncated.

0.56 seconds
```

no data found

```
DROP TABLE customer1;

SELECT * FROM customer WHERE city='New York';

UPDATE customer SET pstal_code='10032' WHERE customer_name='Mary Smith';

SELECT * FROM customer;
```

Results Explain Describe Saved SQL History

```
ORA-00942: table or view does not exist
```

Result:

SQL queries using DDL and DCL commands are successfully executed.

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 11-08-16

EXPERIMENT-NO 2

Video Link : https://youtu.be/IlP8IFKLGiY?list=PLRS_VYrnFL6lj53kJl8EaWDEsoE-8Epkj

AIM: To create tables using DML and TCL commands COMMANDS.

DESCRIPTION:

Data Manipulation Language (DML) - These SQL commands are used for storing, retrieving, modifying, and deleting data. These commands are SELECT, INSERT, UPDATE, and DELETE.

Transaction Control Language (TCL) - These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.

1.Display the product table and orderline table

DESC product;
DESC orderline;

Object Type **TABLE** Object **ORDERQUANTITY**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERQUANTITY	ORDER_ID	Number	-	10	0	1	-	-	-
	PRODUCT_ID	Number	-	10	0	-	✓	-	-
	ORDERED_QUANTITY	Number	-	10	0	-	✓	-	-
	PRICE	Number	-	10	0	-	✓	-	-

1 - 4

PRODUCT_ID	PRODUCT_DESCRIPTION	PRODUCT_MATERIAL	PRODUCT_LINE_ID
1000	Office Desk	Cherry	10
1001	Manager's Desk	Red Oak	10
2000	Office Chair	Cherry	20
2001	Manager's Desk	NaturalOak	20
3000	Book Shelf	NaturalAsh	30
3001	Duplex Book Shelf	White Ash	30
4000	Table Lamp	NaturalAsh	40
4001	Duplex Table Lamp	White Ash	40
9999	Keyboard	Plastic	50

2.Select the details of customer living the city 'NewYork'.

SELECT * FROM customer WHERE city='New York';

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CITY	STATE	PSTAL_CODE
1	John Doe	392 Sunset Blvd.	New York	NT	10059

3.Update the pstal_code of 'Mary Smith', to '10032'.

UPDATE customer SET pstal_code='10032' WHERE customer_name='Mary Smith';
SELECT * FROM customer;

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CITY	STATE	PSTAL_CODE
1	John Doe	392 Sunset Blvd.	New York	NT	10059
2	Mary Smith	6900 Main St.	SanFrancis	CA	10032
3	Richard Newman	2040 Riverside Rd.	San Diego	CA	92010
4	Cathy Cook	4010 Speedway	Tucson	AZ	85719

4.Display the details of the Order table.

SELECT * FROM order1;

ORDER_ID	ORDER_DATE	CUSTOMER_ID
100	01-OCT-14	1
101	01-OCT-14	2
102	02-OCT-14	3
103	03-OCT-14	2
104	10-OCT-14	1
105	10-OCT-14	4
106	10-OCT-14	2
107	10-OCT-14	1

5.Insert the record (108, '10-NOV-14', 1) into Order table.

INSERT INTO order1 VALUES (108,TO_DATE('10-NOV-14','DD-MON-YYYY'),1);
SELECT * FROM order1;

ORDER_ID	ORDER_DATE	CUSTOMER_ID
100	01-OCT-14	1
101	01-OCT-14	2
102	02-OCT-14	3
103	03-OCT-14	2
104	10-OCT-14	1
105	10-OCT-14	4
106	10-OCT-14	2
107	10-OCT-14	1
108	10-NOV-14	1

6.Remove the record from Order where order_id =’104’
DELETE FROM order1 WHERE order_id=104;

SELECT * FROM order1;

ORDER_ID	ORDER_DATE	CUSTOMER_ID
100	01-OCT-14	1
101	01-OCT-14	2
102	02-OCT-14	3
103	03-OCT-14	2
105	10-OCT-14	4
106	10-OCT-14	2
107	10-OCT-14	1
108	10-NOV-14	1

7.Display the details of customer whose name is 'Richard Newman'.
SELECT * FROM customer WHERE customer_name='Richard Newman';

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CITY	STATE	PSTAL_CODE
3	Richard Newman	2040 Riverside Rd.	San Diego	CA	92010

8.Display the details of the product whose price is greater than 100.
SELECT * FROM orderquantity WHERE price>100;

ORDER_ID	PRODUCT_ID	ORDERED_QUANTITY	PRICE
101	1000	20	15000
102	3000	15	23500
103	4001	14	1000
105	3001	20	30005
106	3000	12	20500
107	4000	11	30000

9.Display all the orders placed by the customer 2.
SELECT * FROM order1 WHERE customer_id=2;

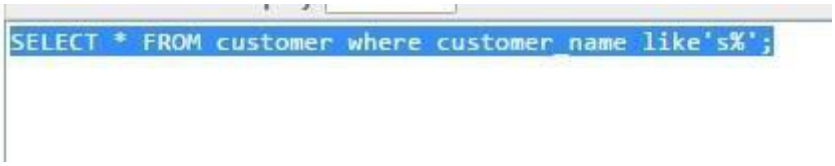
ORDER_ID	ORDER_DATE	CUSTOMER_ID
101	01-OCT-14	2
103	03-OCT-14	2
106	10-OCT-14	2

10.Display the details of Table or Chair product.
SELECT * FROM product WHERE product_description='Office Desk' OR product_description='Office Chair';

PRODUCT_ID	PRODUCT_DESCRIPTION	PRODUCT_MATERIAL	PRODUCT_LINE_ID
1000	Office Desk	Cherry	10
2000	Office Chair	Cherry	20

11.Display the details of customer whose city name ends with 'k'.
12.Display the details of the customer whose name starts with 'S'.
SELECT* FROM customer where city like '%k';
SELECT* FROM customer where name like 's%';

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CITY	STATE	PSTAL_CODE
1	John Doe	392 Sunset Blvd.	New York	NT	10059



13.Find how many orders placed for the product = 50
SELECT count(product_id) AS COUNT FROM orderquantity WHERE product_id=50;

COUNT
0

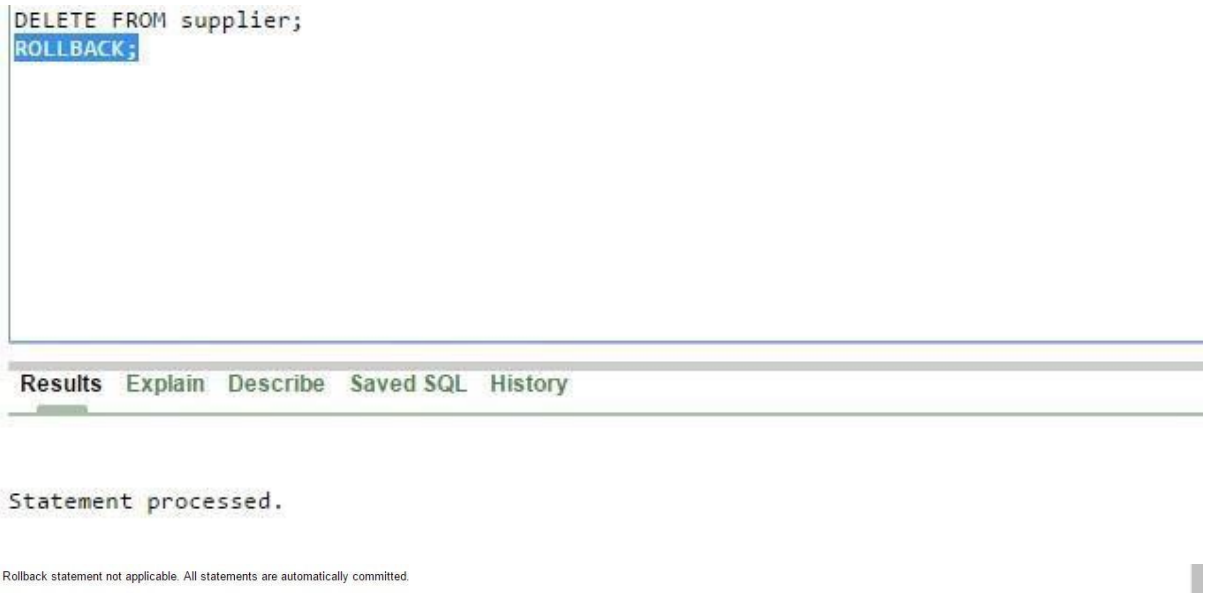
14.Give a 10% increase to all the product price and display the column with a name updated price.

UPDATE_PRICE
11000
16500
38016
1100
33005.5
33000

6 rows returned in 0.0

SELECT 0.1*price+price AS update_price
FROM orderquantity;

15. Undo the insert operation and state is it possible. If not justify 16.Undo only the delete operation.



Result: SQL queries using DML and TCL commands are successfully executed.

Database Systems Lab - 14CS2012

REGISTER NO : UR14CS228
DATE : 22-08-16
EXPERIMENT NO : 3

Video URL : https://youtu.be/nqQuIvhjIFY

AIM:

To use Aggregate and Built – in Functions, Group by and Order by functions in an database table.

DESCRIPTION:

GROUP BY: Optional section of SELECT statement used to group data based on distinct values of
specified columns.Creates a data set, containing several groups based on a condition.

ORDER BY: Used with select statement only. to view data from a table in sorted order.Rows
retrieved from the table can be sorted in either ascending or descending order.Sorting is based on
columns specified in select statement.Default sort order is ascending order

AGGREGATE FUNCTION: Functions that take a collection / set of values & returns a single value.
Used as expressions in the select statement to return summary data.

SYNTAX:

1. Display the number of customers.

```
SELECT COUNT(customer_id) FROM Customer
```

COUNT(customer_id)
7

2.What is the maximum quantity that has been ordered?

```
SELECT MAX(ordered_quantity) FROM Orderline
```

MAX(ordered_quantity)
40100

3.Display the minimum, maximum and average prices of the product.

SELECT MAX(price) AS max, min(price) AS min, avg(price) AS average FROM Product

max	min	average
50	10	27.7778

4.Display the average price of each product.

SELECT avg(price),product_description FROM Product GROUP BY product_description

avg(price)	product_description
30.0000	Book Shelf
30.0000	Duplex Boo
40.0000	Duplex Tab
50.0000	Keyboard
15.0000	Manager's
20.0000	Office Cha
10.0000	Office Des
40.0000	Table Lamp

5.Find the total quantity ordered for each product.

SELECT sum(ordered_quantity), product_id FROM Orderline GROUP BY product_id

sum(ordered_quantity)	product_id
56560	2000
20500	3000
30005	3001
10000	4000
41100	4001

6.Find out the no. of orders placed on each date.

SELECT Orders.order_date, sum(Orderline.ordered_quantity) FROM Orders,Orderline WHERE Orders.order_id=Orderline.order_id GROUP BY Orders.order_date

order_date	sum(Orderquantity.ordered_quantity)
01-OCT-14	30000
02-OCT-14	34560
03-OCT-14	1000
10-OCT-14	92605

7.Find out how many products are ordered in each order.

SELECT order_id, sum(ordered_quantity) FROM Orderline GROUP BY order_id

order_id	sum(ordered_quantity)
100	10000
101	20000
102	34560
103	1000
104	2000
105	30005
106	20500
107	40100

8.Sort the order table based on the order date.

SELECT * FROM Orders GROUP BY order_date

order_id	order_date	customer_id
100	01-OCT-14	1
102	02-OCT-14	3
103	03-OCT-14	2
104	10-OCT-14	1

9.Display the products and their quantity and sort the result on the sum of quantity ordered in descending order

SELECT product_description, sum(product_quantity)As quantity FROM Product GROUP BY product_description ORDER BY quantity DESC;

product_description	quantity
Table Lamp	50
Duplex Tab	40
Manager's	30
Keyboard	20
Office Cha	20
Book Shelf	20
Duplex Boo	10
Office Des	10

10. Display the supplier details sorted by their ratings in descending order.

SELECT supplier_details FROM Suppliers ORDER BY supplier_rating DESC;

Wood supplier
Duplex Supplier
Human resource
Keyboard Supplier
Office item Supplier

11. Display the names of all customers in uppercase.

SELECT UPPER(customer_name) FROM Customer

UPPER(customer_name)
RICHARD JOHNS
MARY JOHNS
JOSEPH JOHNS
CATHY COOK
RICHARD NEWMAN
MARY SMITH
JOHN DOE

12. Find the number of months between ordered date of order_id=103 and current date

SELECT TIMESTAMPDIFF(MONTH,order_date,curdate()) FROM Orders WHERE order_id ='103'

TIMESTAMPDIFF(MONTH,order_date,curdate())
8

13. Prefix the postal code with two zeroes

UPDATE Customer SET pstal_code=concat('00',pstal_code)

pstal_code
0010059
0010039
0010056
0085719
0092010
0094032
0010059

14. Display the substring “Newman” from “Richard Newman”.

SELECT SUBSTRING_INDEX(customer_name," ",-1) FROM Customer WHERE customer_name='Richard johns'

SUBSTRING_INDEX(customer_name," ",-1)
Johns

15. Display the number of characters in the name of customers which ends with ‘e’.

SELECT customer_name, char_length(customer_name) AS length FROM Customer WHERE customer_name LIKE '%e'

customer_name	length
John Doe	8

Result:

SQL queries using DDL and DCL commands are successfully executed.

Database Systems Lab - 14CS2012

REGISTER NO : UR14CS228
DATE : 19-09-16
EXPERIMENT NO : 4

Video URL : https://youtu.be/L2WKPtiMFrw

AIM:

To write Basic SQL queries to use Set operations and Joins.

DESCRIPTION:

Set operations allow the results of multiple queries to be combined into a single result set.Set operators include UNION, INTERSECT, and EXCEPT.

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

INNER JOIN: Returns all rows when there is at least one match in BOTH tables

LEFT JOIN: Return all rows from the left table, and the matched rows from the right table

RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table

FULL JOIN: Return all rows when there is a match in ONE of the tables

SYNTAX:

Excercise 4

1. Display the product that was ordered and delivered (Union)

SELECT product_id, product_description FROM Product UNION SELECT p_id FROM Supplier;

PRODUCT_ID

1000

1001

2000

3000

3001

4001

9999

2. Display the details of the product that are yet to be delivered. (Minus)

SELECT product_id, product_description FROM Product MINUS SELECT p_id FROM Supplier

PRODUCT_ID

3000

3001

3. List the name of the customers who has placed an order on or after 10 th OCT 2014.

SELECT Orders.customer_id,customer.customer_name,Orders.order_id FROM ORders INNER JOIN customer ON Orders.customer_id=customer.customer_id WHERE order_date > to_date('10-10-2014','DD-MM-YYYY');

CUSTOMER_ID CUSTOMER_NAME ORDER_ID

1 John Doe 108

4. Display the details of the product that are delivered within 10 days.

SELECT Supplier.p_id,Orders.order_date,Supplier.s_date FROM Supplier LEFT OUTER JOIN Orderquantity ON Supplier.p_id=Orderq fuantity.product_id LEFT OUTER JOIN Orders ON Orders.order_id=Orderquantity.order_id WHERE (TO_DATE(Supplier.s_date)-TO_DATE(Orders.order_date)) <= 10;

NO ROWS SELECTED

5. Display details of the customer who has received their ordered product.

SELECT * FROM Customer WHERE customer_id IN (SELECT customer_id FROM Orders WHERE d_time IS NOT NULL)

CUSTOMER_ID CUSTOMER_NAME

---

2 Mary Smith

2 Mary Smith

6. Display the details of the product whose quantity is greater than 10

SELECT * FROM Product, Orderquantity WHERE Product.product_id IN (SELECT product_id FROM Orderquantity WHERE ordered_quantity > 10)

PRODUCT_ID

1000

3000

4001

3001

3000

7. Display the details of order_id with Order_date and customer_name. (inner join)

SELECT order_id,order_date,customer_name FROM Orders INNER JOIN Customer ON Orders.customer_id=Customer.customer_id

ORDER_ID ORDER_DAT CUSTOMER_NAME

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

100 01-OCT-14 John Doe

101 01-OCT-14 Mary Smith

102 02-OCT-14 Richard Newman

103 03-OCT-14 Mary Smith

105 10-OCT-14 Cathy Cook

106 10-OCT-14 Mary Smith

107 10-OCT-14 John Doe

108 10-NOV-14 John Doe

8. Display the details of all the products that have been ordered and also the details of delivered product. (use left outer join)

SELECT * FROM Product LEFT OUTER JOIN Orders ON Orders.p_id=Product.p_id

RODUCT_ID PRODUCT_DESCRIPTION PRODUCT_MA PRODUCT_LINE_ID

1000 Office Desk Cherry 10

4001 Duplex Table Lamp White Ash 40

9. Display the details of all customers as well as the order details. (full outer join)

SELECT * FROM Customer FULL OUTER JOIN Orders ON Customer.customer_id = Orders.customer_id

CUSTOMER_ID CUSTOMER_NAME CUSTOMER_ADDRESS CITY

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

STATE PSTAL_ ORDER_ID ORDER_DAT CUSTOMER_ID

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

3 Richard Newman 2040 Riverside Rd. San Diego

CA 92010 102 02-OCT-14 3

10. Display the details of the Supplier who has supplied 'Office Desk'

SELECT Supplier.* FROM Supplier LEFT OUTER JOIN Product ON Supplier.p_id=Product.product_id WHERE Product.product_description='Office Desk';

S_ID S_NAME GENDER S_DATE P_ID

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

1 aaa M 11-NOV-14 1000

Result:

SQL queries using DDL and DCL commands are successfully executed.

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 26-08-16

EXPERIMENT-NO 5

Video Link : https://youtu.be/RsiWTjJedfI

AIM: To write Basic SQL queries, Sub Queries and Views.

DESCRIPTION:

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

Output:

1.Find the names of customers who have ordered for a `cherry' or ` red oak' finish.

SELECT * FROM Customer WHERE customer_id IN (SELECT customer_id FROM Orders WHERE p_id IN (SELECT product_id FROM Product WHERE product_finish='cherry' OR product_finish='red oak'))

NO ROWS SELECTED

2.Find the names of customers who have ordered for the product "Table Lamp". (use IN)

SELECT * FROM Customer WHERE customer_id IN (SELECT customer_id FROM Orders WHERE p_id IN (SELECT product_id FROM Product WHERE product_description='Table Lamp'))

NO ROWS SELECTED

3.Find the details of customers who have ordered for product id 1000. (use EXISTS)

SELECT * FROM Customer WHERE customer_id IN (SELECT customer_id FROM Orders WHERE p_id='1000')

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CITY	STATE	PSTAL
2	Mary Smith	6900 Main St.	SanFrancis	CA	10032

4.Find suppliers whose total supply quantity is greater than some supplier called S3. (use ANY)

SELECT * FROM Supplier WHERE supply_quantity > (SELECT supply_quantity FROM Supplier WHERE s_id ='S3')

NO ROWS SELECTED

5.Find the supplier with the highest rating using ALL

SELECT s_name FROM Supplier WHERE p_id= ANY (SELECT product_id FROM Orderquantity WHERE ordered_quantity= (SELECT MAX(ordered_quantity) FROM Orderquantity))

S_NAME

---

aaa

6.Display the products that are same as the product ordered by "Cathy Cook"

SELECT Product.* FROM Product WHERE product_id = ANY (SELECT p_id FROM Order WHERE customer_id = (SELECT customer_id FROM Customer WHERE customer_name ='Cathy Cook'))

PRODUCT_ID PRODUCT_DESCRIPTION

3001 Duplex Book Shelf

7.Display the supplier details whose supplied quantity is greater than the quantity of order id - 103

SELECT * FROM Supplier WHERE s_id = ANY (SELECT s_id FROM Orders WHERE ordered_quantity > (SELECT ordered_quantity FROM Orderqunantity WHERE order_id='103'))

S_ID S_NAME GENDER S_DATE P_ID

---

1 aaa M 11-NOV-14 1000

8.Display the ordered quantity of the order whose ordered date is greater than the ordered date of the order id 106

SELECT ordered_quantity FROM Orderquantity WHERE order_id = ANY (SELECT order_id FROM Orders WHERE order_date > (SELECT order_date FROM Orderquantity WHERE order_id='106'))

NO ROWS SELECTED

9.Display the customer names whose order date is equal to the order date of customer id 4.

SELECT * FROM Customer WHERE customer_id = ANY (SELECT customer_id FROM Orders WHERE order_date > ANY (SELECT order_date FROM Orders WHERE customer_id='4'))

CUSTOMER_NAME

Cathy Cook

Mary Smith

John Doe

10.Display the customer details whose rating is greater than the supplier "ccc"

SELECT * FROM Customer WHERE customer_id = ANY (SELECT rating FROM Supplirt WHERE s_name='ccc')

NO ROWS SELECTED

11.Create a view(V1) to display the customer name, order id and quantity.

CREATE OR REPLACE VIEW V1 AS (SELECT customer_name,Orders.order_id,Orderquantity.ordered_quantity FROM Customer INNER JOIN Orders ON Customer.customer_id=Orders.customer_id INNER JOIN Orderquantity ON Orderquantity.order_id=Orders.order_id);

View created.

12.Create a view(V2) to display the order-id, order-date and the date of delivery.

```
CREATE OR REPLACE VIEW V2 AS (SELECT
Orders.order_id,Orders.order_date,Supplier.s_date FROM Orders INNER JOIN
Orderquantity ON Orderquantity.order_id=Orders.order_id INNER JOIN Supplier ON
Supplier.p_id=Orderquantity.product_id);
```

View created.

13.Insert a row into the above view. Is the view updatable? If yes, is the updation reflected in the base table.

```
INSERT INTO V2 VALUES (104, TO_DATE('10-OCT-2014','DD-MON-YYYY'),
TO_DATE('15-OCT-2014','DD-MON-YYYY'));
```

View Updated

14.Create a view (V3) with customer name, order id and order date from V1 and V2.

```
CREATE OR REPLACE VIEW V3 AS (SELECT
V1.customer_name,V1.order_id,V2.order_date FROM V1 INNER JOIN V2 ON
V1.order_id=V2.order_id);
```

View created.

15.Delete a row from the above view and state what happens to the other views from which it was created.

```
DELETE FROM v3 WHERE V3.order_id=103;
DELETE FROM v3 *
```

View deleted

Result:

SQL queries using DML and TCL commands are successfully executed.

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 17-10-2016

EXPERIMENT-NO 6

Video Link : https://youtu.be/vuTSnC__AOk

AIM:
Develop an application for a company to manage its order and supply details using JDBC connectivity

DESCRIPTION:
Class.forName("oracle.jdbc.driver.OracleDriver"); //Register the Driver Class

Connection con=DriverManager.getConnection(

"jdbc:oracle:thin:@localhost:1521:xe","system","password"); // Connection

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select * from emp");

Program:

```
import java.util.Scanner;
public class Company {

    public static void main(String[] args) {

        CompanyOperations coop = new CompanyOperations();

        Scanner scan = new Scanner(System.in);
        int choice;
        System.out.println("Database Operation:");
        System.out.println("Enter 1 for New Order");
        System.out.println("Enter 2 to Modify the order
details");
        System.out.println("Enter 3 to Delete from order
table");
```

```
        System.out.println("Enter 4 to Search supply details
based on Product Id");
        System.out.println("Enter your choice of operation");
        choice = scan.nextInt();

        switch (choice) {
            case 1: {
                coop.insertQuery();
                break;
            }
            case 2: {
                coop.modificationQuery();
                break;
            }
            case 3: {
                coop.deleteQuery();
                break;
            }
            case 4: {
                coop.search1();
                break;
            }
            default: {
                System.out.println("Enter Correct Choice
Please.");
            }

        }

    }
}
```

```
import java.sql.*;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Scanner;
import java.util.logging.Formatter;
import java.util.logging.Level;
import java.util.logging.Logger;


public class CompanyOperations {


    final String JDBC_Driver =
"oracle.jdbc.driver.OracleDriver";
    final String DB_URL =
"jdbc:oracle:thin:@localhost:1521:XE";
    final String USER = "system";
    final String PASSWORD = "fish";
    Connection conn = null;
    Statement statement = null;
    Scanner scan = new Scanner(System.in);



    void insertQuery() {
        try {
            try {
                Class.forName(JDBC_Driver);//REGISTERING THE
DRIVER
            } catch (ClassNotFoundException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            }
            try {
                conn = DriverManager.getConnection(DB_URL,
USER, PASSWORD);// CONNECTION
```

```
            } catch (SQLException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            }
            int order_id;
            int customer_id;
            String sql;
            System.out.println("Enter order id");
            order_id = scan.nextInt();
            System.out.println("Enter the order date('DD-MMM-
YYYY')");
            String order_date1 = scan.next();
            SimpleDateFormat df = new SimpleDateFormat("dd-
MMM-yyyy");
            java.util.Date order_date2 =
df.parse(order_date1);
            java.sql.Date order_date = new
java.sql.Date(order_date2.getTime());
            System.out.println("Enter the customer id");
            customer_id = scan.nextInt();
            sql = "insert into
order1(order_id,order_date,customer_id)" + "values(?,?,?)";
            PreparedStatement pstatement =
conn.prepareStatement(sql);
            pstatement.setInt(1, order_id);
            pstatement.setDate(2, order_date);
            pstatement.setInt(3, customer_id);
            pstatement.executeUpdate();
            System.out.println("New Row added!");
            pstatement.close();
            conn.close();
        } catch (ParseException | SQLException ex) {

System.out.println(Arrays.toString(ex.getStackTrace()));
            System.out.println(ex.getMessage());
        }
```

```
    }


    void modificationQuery() {
        try {
            Class.forName(JDBC_Driver);
            conn = DriverManager.getConnection(DB_URL, USER,
PASSWORD);
        } catch (ClassNotFoundException ex) {
            System.out.println("Its a"+ ex.getMessage());
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }


        System.out.println("Which column would you like to
update? Press 1 for Order Date or 2 for Customer Id");
        int choice = scan.nextInt();
        System.out.println("Enter the updatable order id");
        int order_id = scan.nextInt();



        if(choice==1){
            try {
                System.out.println("Enter the new Date('DD-
MMM-YYYY')");
                String order_date1 = scan.next();
                SimpleDateFormat df = new
SimpleDateFormat("dd-MMM-yyyy");
                java.util.Date order_date2 =
df.parse(order_date1);
                java.sql.Date order_date = new
java.sql.Date(order_date2.getTime());
                String sql = "UPDATE order1 SET order_date = ?
WHERE order_id = ?";
                PreparedStatement pstatement =
conn.prepareStatement(sql);
                pstatement.setDate(1,order_date);
```

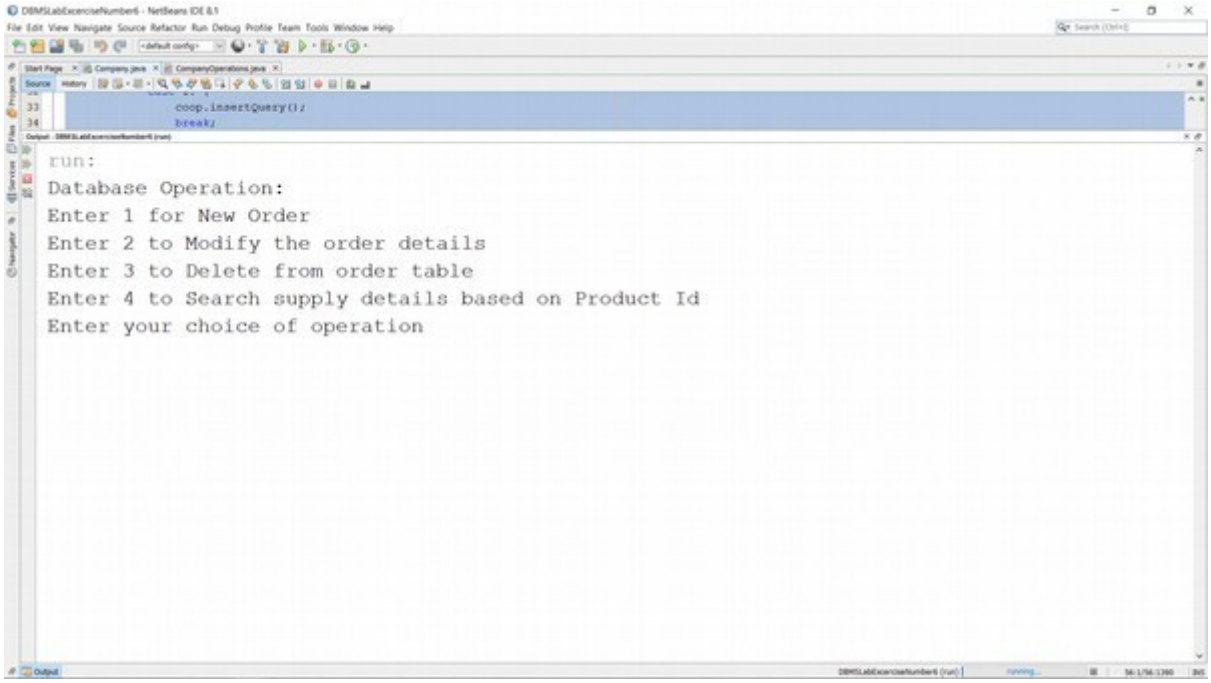
```
                pstatement.setInt(2,order_id);
                pstatement.executeUpdate();
                pstatement.close();
                conn.close();
            } catch (ParseException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            } catch (SQLException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            }
        }
        else{
            try {
                System.out.println("Enter the customer Id");
                int customer_id = scan.nextInt();
                String sql = "UPDATE order1 SET customer_id
= ? WHERE order_id = ?";
                PreparedStatement pstatement =
conn.prepareStatement(sql);
                pstatement.setInt(1,customer_id);
                pstatement.setInt(2,order_id);
                pstatement.executeUpdate();
                pstatement.close();
                conn.close();
            } catch (SQLException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            }
        }
```

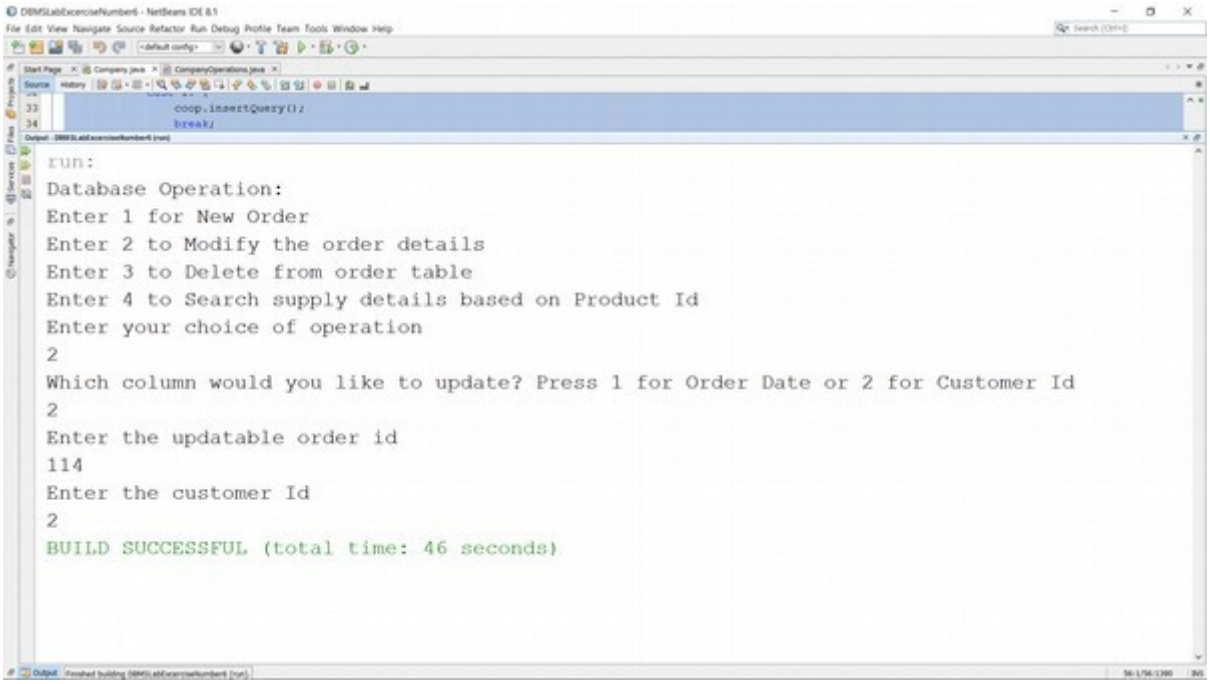
```
    }


    void deleteQuery() {
        try {
            try {
                Class.forName(JDBC_Driver);
                conn = DriverManager.getConnection(DB_URL,
USER, PASSWORD);
            } catch (ClassNotFoundException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            } catch (SQLException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            }
            System.out.println("Enter the order id to be
deleted");
            int order_id = scan.nextInt();
            String sql = "DELETE FROM order1 WHERE order_id
= ?";
            PreparedStatement pstatement =
conn.prepareStatement(sql);
            pstatement.setInt(1, order_id);
            pstatement.executeUpdate();
            pstatement.close();
            conn.close();


        } catch (SQLException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
        }
    }



    void search1(){
```

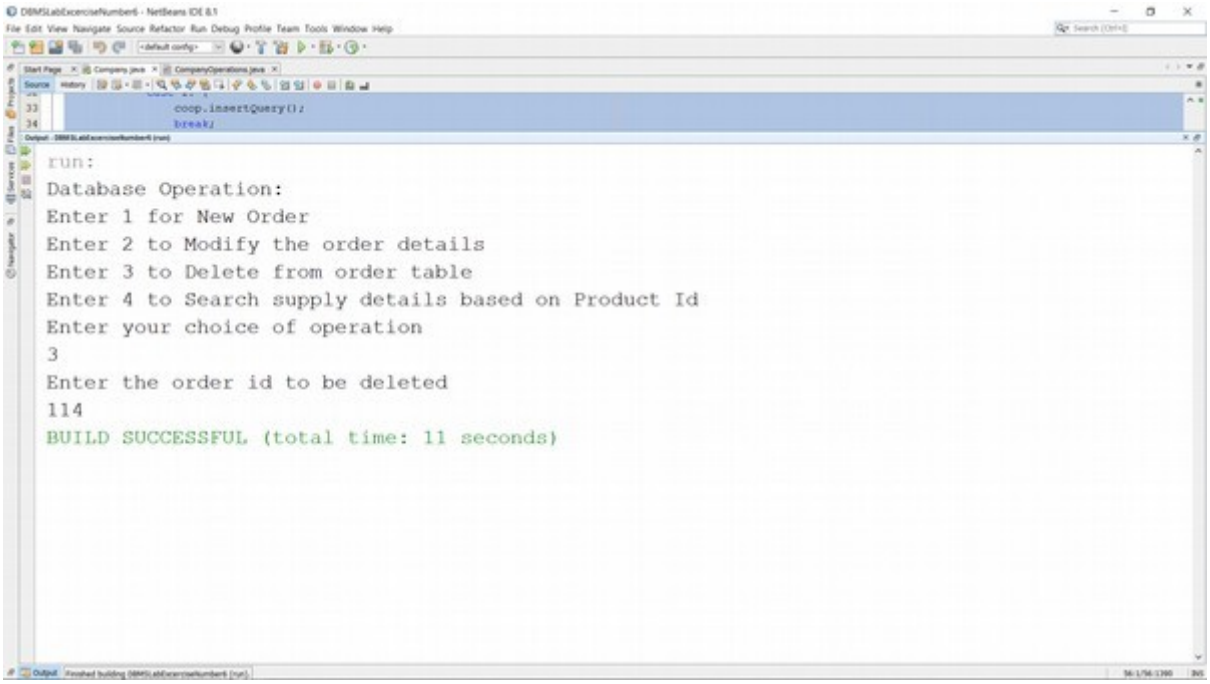
```
          try {
             int p_id1;
          try {
                 Class.forName(JDBC_Driver);
                 conn = DriverManager.getConnection(DB_URL,
USER, PASSWORD);
             } catch (ClassNotFoundException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
             } catch (SQLException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
             }
             System.out.println("Enter the Product Id");
             p_id1 = scan.nextInt();
             String p_id = Integer.toString(p_id1);


             String sql;
             sql = "SELECT * FROM supplier WHERE p_id = " +
p_id;
             Statement pstatement = conn.createStatement();
             ResultSet rs = pstatement.executeQuery(sql);
             while(rs.next()){
                 int s_id = rs.getInt("s_id");
                 String s_name = rs.getString("s_name");
                 String gender = rs.getString("gender");
                 Date s_date = rs.getDate("s_date");
                 int p_id20= rs.getInt("p_id");
                 System.out.println(s_id);
                 System.out.println(s_name);
                 System.out.println(gender);
                 System.out.println(s_date);
                 System.out.println(p_id20);
             }
```

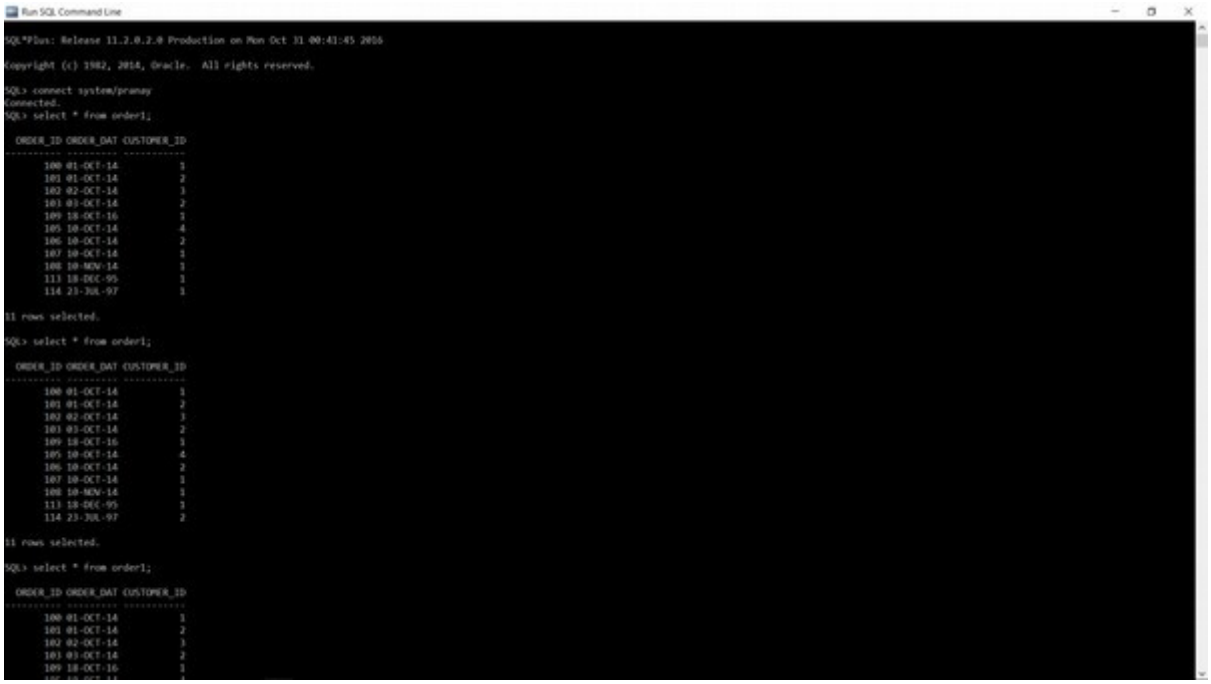
```
                rs.close();
                pstatement.close();
                conn.close();
            } catch (SQLException ex) {

Logger.getLogger(CompanyOperations.class.getName()).log(Level.
SEVERE, null, ex);
            }
        }
}
```

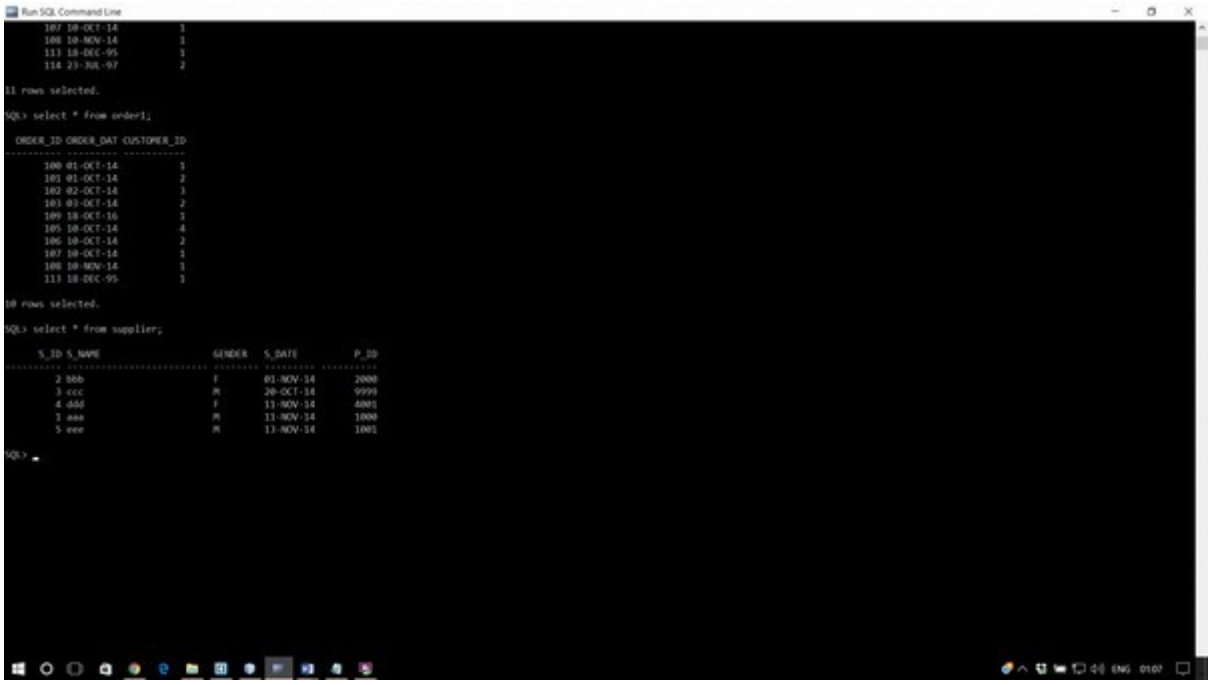
Output:




coop.insertQuery();
break;
run:
Database Operation:
Enter 1 for New Order
Enter 2 to Modify the order details
Enter 3 to Delete from order table
Enter 4 to Search supply details based on Product Id
Enter your choice of operation
2
Which column would you like to update? Press 1 for Order Date or 2 for Customer Id
2
Enter the updatable order id
114
Enter the customer Id
2
BUILD SUCCESSFUL (total time: 46 seconds)


coop.insertQuery();
break;
run:
Database Operation:
Enter 1 for New Order
Enter 2 to Modify the order details
Enter 3 to Delete from order table
Enter 4 to Search supply details based on Product Id
Enter your choice of operation
3
Enter the order id to be deleted
114
BUILD SUCCESSFUL (total time: 11 seconds)





Result:

Application Development using JDBC Connectivity was successfully completed.

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 24-10-2016

EXPERIMENT-NO 7

Video Link : https://youtu.be/VqU3vLtdLzg

AIM:

To Create Triggers based on different questions.

DESCRIPTION:

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database.

Program:

1.Create a simple Trigger that does not allow delete operations on the Supplier table

```
CREATE OR REPLACE TRIGGER trigger1
    BEFORE DELETE ON supplier
    FOR EACH ROW
   BEGIN
    IF deleting THEN
    RAISE_APPLICATION_ERROR(-20101,'Deletion is not enabled');
    END IF;
    END;
     /
delete supplier Where S_ID='ur3';
```

2. Create a Trigger that raises an User Defined Error Message and does not allow updating the order table.

```
CREATE OR REPLACE TRIGGER trigger2
    BEFORE UPDATE ON order1
    FOR EACH ROW
   BEGIN
   IF inserting OR updating THEN
   RAISE_APPLICATION_ERROR(-20101,'Updation is not allowed');
   END IF;
   END;
   /
update order1 SET customer_id=10 WHERE order_id=1000;
```

3. Create a trigger to implement referential integrity policy “on delete set NULL” in the order_quantity table. That is when an order-id is deleted in the order table, set NULL for deleted order-id in the order_quantity table.

```
 CREATE OR REPLACE TRIGGER trigger3
   BEFORE DELETE ON order1
   FOR EACH ROW
   BEGIN
   UPDATE orderquantity SET order_id=NULL WHERE
order_id=:old.ORDER_ID;
   END;
    /

delete order1 where ORDER_ID=1000;
```

4. Create a trigger to update the order table whenever the customer id is updated in the customer table.

```
CREATE OR REPLACE TRIGGER trigger4
   BEFORE UPDATE ON customer
   FOR EACH ROW
   BEGIN
```

```
   UPDATE order1 SET customer_id=:new.customer_id WHERE
customer_id=:old.customer_id;
   END;
 /
 update customer set CUSTOMER_ID=40 where
CUSTOMER_NAME='john';
```

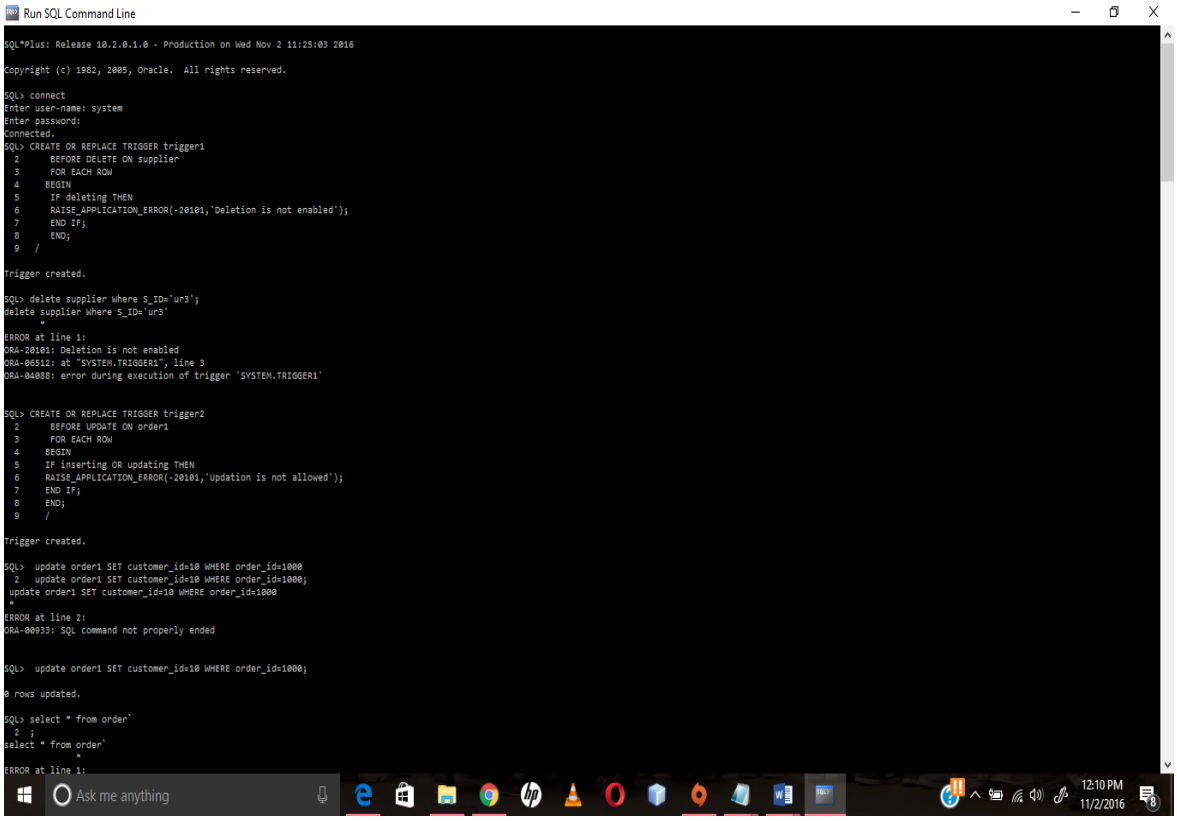
5. Create a trigger for customer table that inserts a newly inserted values in the customer_audit table. The structure of the customer_audit table is (cus_id, cus_name, city).

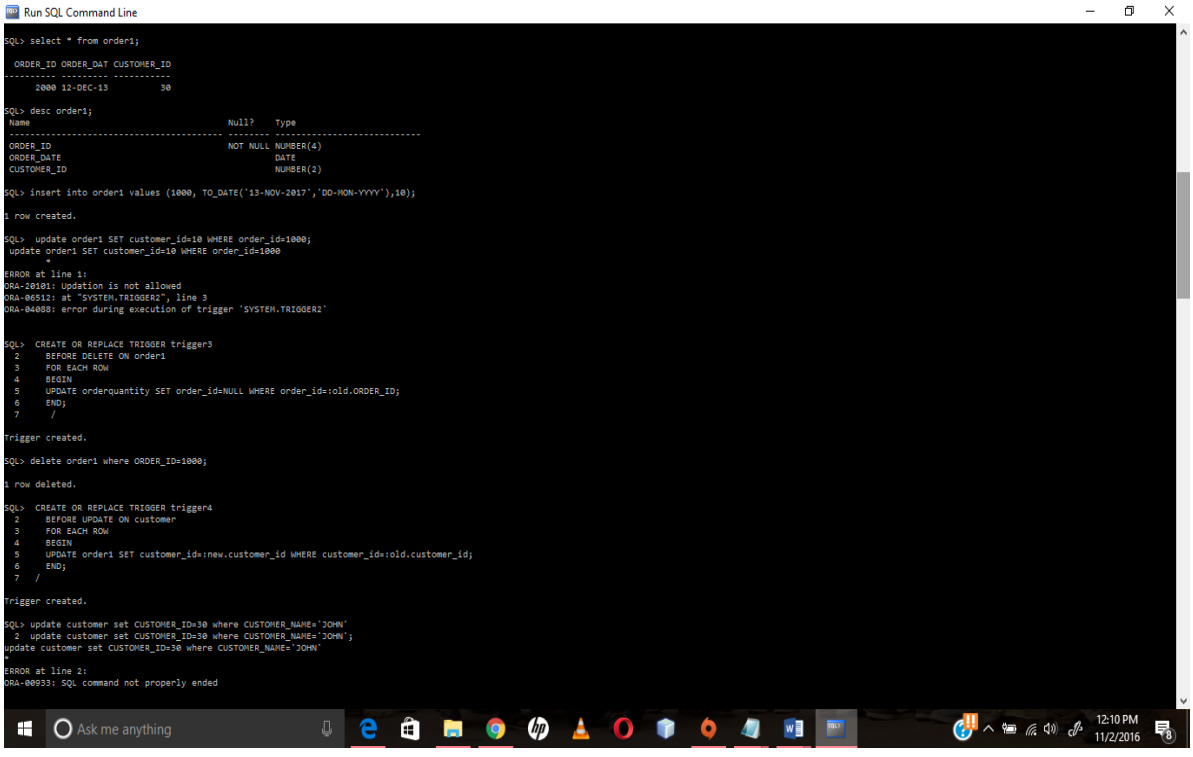
```
CREATE OR REPLACE TRIGGER trigger5
    AFTER INSERT ON customer
    FOR EACH ROW
    BEGIN
    INSERT INTO customer_audit
VALUES(:new.customer_id,:new.customer_name,:new.city);
    END;
    /
INSERT INTO customer VALUES (70, 'Reuben', 'Mulund', 'Mulund',
'Maharashtra','641114');
```

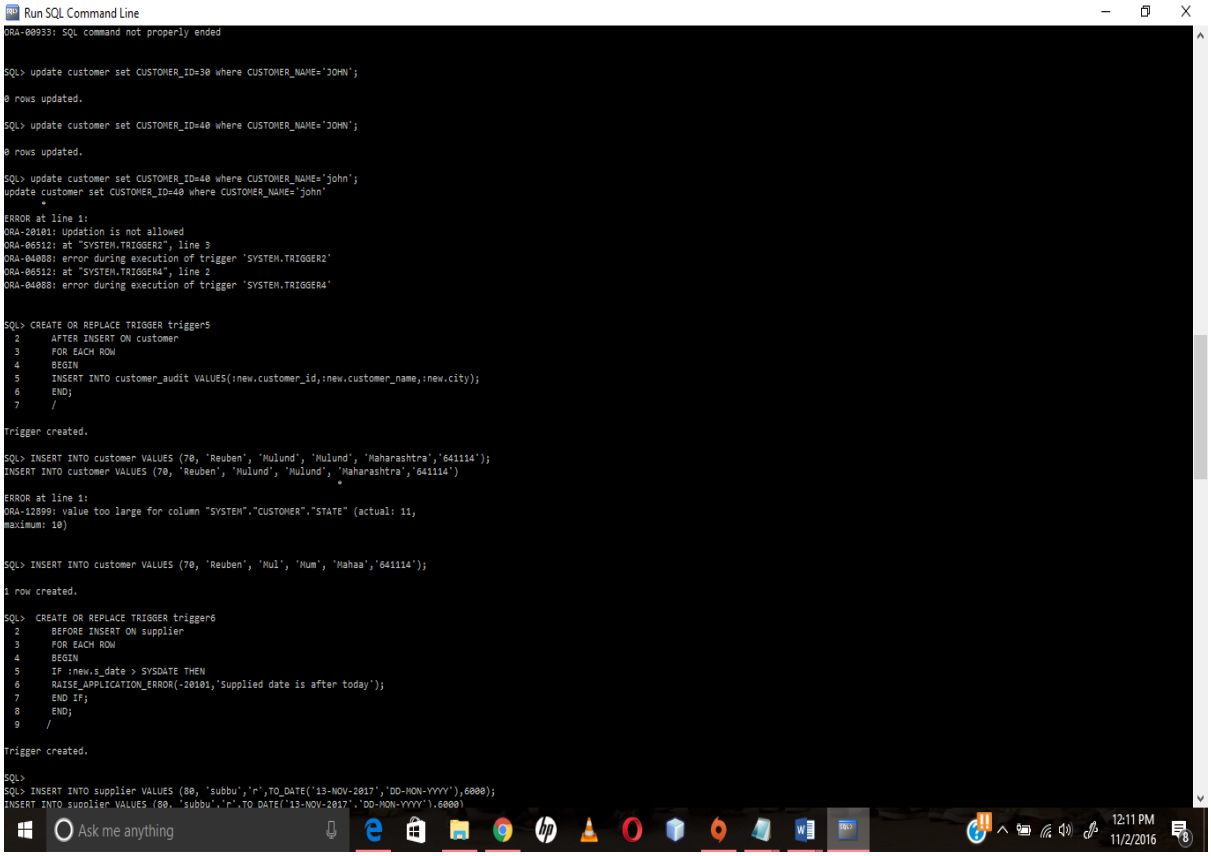
6. Create a Trigger that raises an User Defined Error Message and does not allow updating the supplier table if the supplied date is greater than the current date

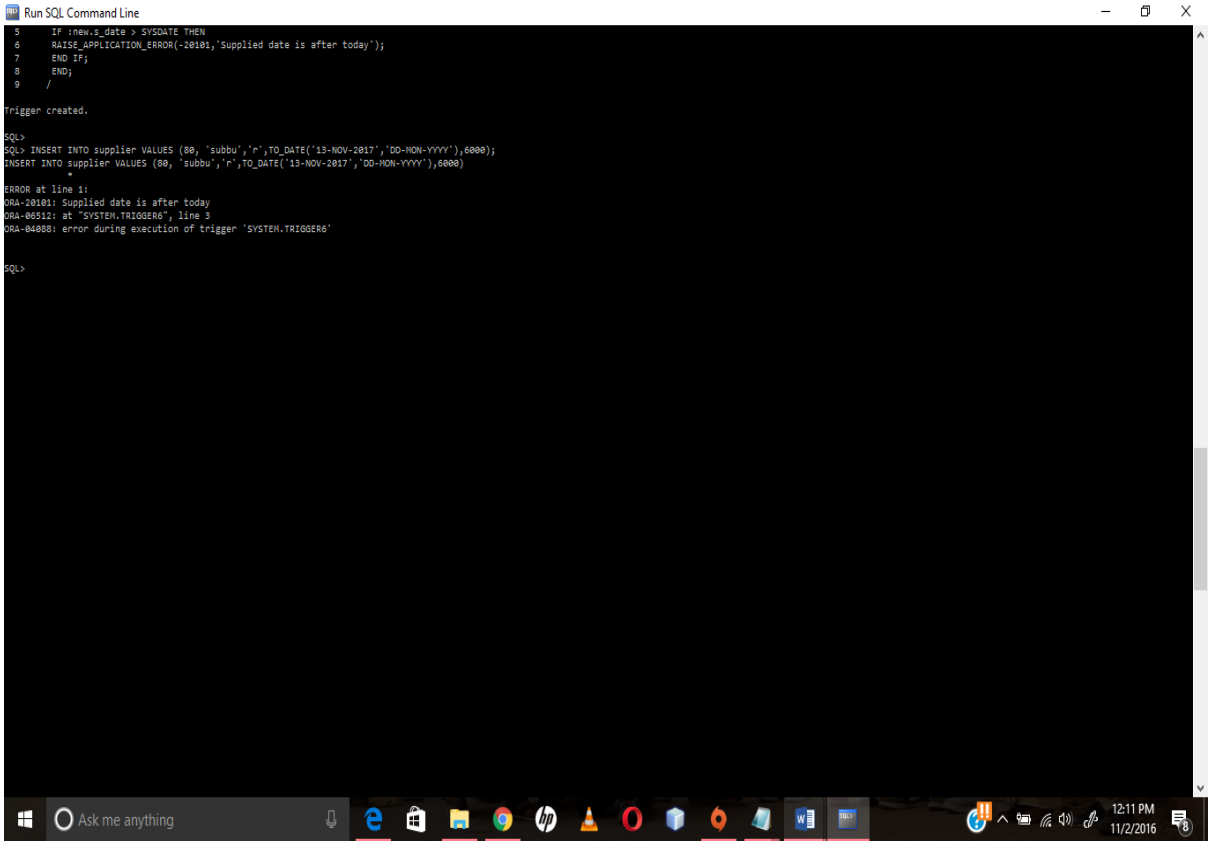
```
CREATE OR REPLACE TRIGGER trigger6
    BEFORE INSERT ON supplier
    FOR EACH ROW
    BEGIN
    IF :new.s_date > SYSDATE THEN
    RAISE_APPLICATION_ERROR(-20101,'Supplied date is after
today');
    END IF;
    END;
   /
INSERT INTO supplier VALUES (80, 'subbu','r',TO_DATE('13-NOV-
2017','DD-MON-YYYY'),6000);
```

Output:


Run SQL Command Line
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Nov 2 11:25:03 2016
Copyright (c) 1982, 2005, Oracle. All rights reserved.
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> CREATE OR REPLACE TRIGGER trigger1
2 BEFORE DELETE ON supplier
3 FOR EACH ROW
4 BEGIN
5 IF deleting THEN
6 RAISE_APPLICATION_ERROR(-20101,'Deletion is not enabled');
7 END IF;
8 END;
9 /
Trigger created.
SQL> delete supplier Where S_ID='ur3';
delete supplier Where S_ID='ur3'
*
ERROR at line 1:
ORA-20101: Deletion is not enabled
ORA-06512: at "SYSTEM.TRIGGER1", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGER1'
SQL> CREATE OR REPLACE TRIGGER trigger2
2 BEFORE UPDATE ON order1
3 FOR EACH ROW
4 BEGIN
5 IF inserting OR updating THEN
6 RAISE_APPLICATION_ERROR(-20101,'Updation is not allowed');
7 END IF;
8 END;
9 /
Trigger created.
SQL> update order1 SET customer_id=10 WHERE order_id=1000
2 update order1 SET customer_id=10 WHERE order_id=1000;
update order1 SET customer_id=10 WHERE order_id=1000
*
ERROR at line 2:
ORA-00933: SQL command not properly ended
SQL> update order1 SET customer_id=10 WHERE order_id=1000;
0 rows updated.
SQL> select * from order`
2 ;
select * from order`
*
ERROR at line 1:
Ask me anything
12:10 PM
11/2/2016


Run SQL Command Line
SQL> select * from order1;
ORDER_ID ORDER_DAT CUSTOMER_ID
2000 12-DEC-13 30
SQL> desc order1;
Name Null? Type
ORDER_ID NOT NULL NUMBER(4)
ORDER_DATE DATE
CUSTOMER_ID NUMBER(2)
SQL> insert into order1 values (1000, TO_DATE('13-NOV-2017','DD-MON-YYYY'),10);
1 row created.
SQL> update order1 SET customer_id=10 WHERE order_id=1000;
update order1 SET customer_id=10 WHERE order_id=1000
*
ERROR at line 1:
ORA-20101: Updation is not allowed
ORA-06512: at "SYSTEM.TRIGGER2", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGER2'
SQL> CREATE OR REPLACE TRIGGER trigger3
2 BEFORE DELETE ON order1
3 FOR EACH ROW
4 BEGIN
5 UPDATE orderquantity SET order_id=NULL WHERE order_id=:old.ORDER_ID;
6 END;
7 /
Trigger created.
SQL> delete order1 where ORDER_ID=1000;
1 row deleted.
SQL> CREATE OR REPLACE TRIGGER trigger4
2 BEFORE UPDATE ON customer
3 FOR EACH ROW
4 BEGIN
5 UPDATE order1 SET customer_id=:new.customer_id WHERE customer_id=:old.customer_id;
6 END;
7 /
Trigger created.
SQL> update customer set CUSTOMER_ID=30 where CUSTOMER_NAME='JOHN'
2 update customer set CUSTOMER_ID=30 where CUSTOMER_NAME='JOHN';
update customer set CUSTOMER_ID=30 where CUSTOMER_NAME='JOHN'
*
ERROR at line 2:
ORA-00933: SQL command not properly ended
Ask me anything
12:10 PM
11/2/2016


Run SQL Command Line
ORA-00933: SQL command not properly ended
SQL> update customer set CUSTOMER_ID=30 where CUSTOMER_NAME='JOHN';
0 rows updated.
SQL> update customer set CUSTOMER_ID=40 where CUSTOMER_NAME='JOHN';
0 rows updated.
SQL> update customer set CUSTOMER_ID=40 where CUSTOMER_NAME='john';
update customer set CUSTOMER_ID=40 where CUSTOMER_NAME='john'
*
ERROR at line 1:
ORA-20101: Updation is not allowed
ORA-06512: at "SYSTEM.TRIGGER2", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGER2'
ORA-06512: at "SYSTEM.TRIGGER4", line 2
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGER4'
SQL> CREATE OR REPLACE TRIGGER trigger5
2 AFTER INSERT ON customer
3 FOR EACH ROW
4 BEGIN
5 INSERT INTO customer_audit VALUES(:new.customer_id,:new.customer_name,:new.city);
6 END;
7 /
Trigger created.
SQL> INSERT INTO customer VALUES (70, 'Reuben', 'Mulund', 'Mulund', 'Maharashtra','641114');
INSERT INTO customer VALUES (70, 'Reuben', 'Mulund', 'Mulund', 'Maharashtra','641114')
*
ERROR at line 1:
ORA-12899: value too large for column "SYSTEM"."CUSTOMER"."STATE" (actual: 11,
maximum: 10)
SQL> INSERT INTO customer VALUES (70, 'Reuben', 'Mul', 'Mum', 'Mahaa','641114');
1 row created.
SQL> CREATE OR REPLACE TRIGGER trigger6
2 BEFORE INSERT ON supplier
3 FOR EACH ROW
4 BEGIN
5 IF :new.s_date > SYSDATE THEN
6 RAISE_APPLICATION_ERROR(-20101,'Supplied date is after today');
7 END IF;
8 END;
9 /
Trigger created.
SQL>
SQL> INSERT INTO supplier VALUES (80, 'subbu','r',TO_DATE('13-NOV-2017','DD-MON-YYYY'),6000);
INSERT INTO supplier VALUES (80, 'subbu','r',TO_DATE('13-NOV-2017','DD-MON-YYYY'),6000)
Ask me anything
12:11 PM
11/2/2016


Run SQL Command Line
5 IF :new.s_date > SYSDATE THEN
6 RAISE_APPLICATION_ERROR(-20101,'Supplied date is after today');
7 END IF;
8 END;
9 /
Trigger created.
SQL>
SQL> INSERT INTO supplier VALUES (80, 'subbu','r',TO_DATE('13-NOV-2017','DD-MON-YYYY'),6000);
INSERT INTO supplier VALUES (80, 'subbu','r',TO_DATE('13-NOV-2017','DD-MON-YYYY'),6000)
*
ERROR at line 1:
ORA-20101: Supplied date is after today
ORA-06512: at "SYSTEM.TRIGGER6", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGER6'
SQL>
Ask me anything
12:11 PM
11/2/2016

Result:

Triggers were successfully created and tested for all the different situations

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 31-10-16

EXPERIMENT-NO 8

Video Link : https://youtu.be/Szpl-eIcNG4

AIM:
To Create Functions and Procedures for the following requirements.

DESCRIPTION:

PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms:

- **Functions**: these subprograms return a single value, mainly used to compute and return a value.
- **Procedures**: these subprograms do not return a value directly, mainly used to perform an action.

Program:

1. Write a PL/SQL procedure that will accept the product id from the user, check if the product is supplied by the supplier and display the status.

```
create or replace procedure p1 (pid in number) is
sid supplier.S_ID%type;
begin
select S_ID into sid from supplier where P_ID=pid;
dbms_output.put_line('supplier id ' ||sid);
end;


begin
  p1(221);
end;
```

```
SQL>
SQL> create or replace procedure p1 (pid in number)
  2   is
  3  sid supplier.S_ID%type;
  4  begin
  5  select S_ID into sid from supplier where P_ID=pid;
  6  dbms_output.put_line('supplier id ' ||sid);
  7  end;
  8  /

Procedure created.

SQL> begin
  2    p1(221);
  3  end;
  4  /
supplier id ur3

PL/SQL procedure successfully completed.
```

2. Write a procedure to calculate total price for the product that has been supplied and Pass the supplier id as the argument.

```
create or replace procedure p2 (sid in varchar2)
is
pi orderquantity.price%type;
oq supplier.S_Q%type;
begin
select orderquantity.price into pi from orderquantity,supplier
where supplier.S_ID=sid and rownum=1;
select S_Q into oq from supplier where S_ID=sid;
dbms_output.put_line('total price : '||pi*oq);
end;


declare
    begin
  p2('ur3');
    end;
```

```
SQL>  create or replace procedure p2 (sid in varchar2)
  2  is
  3  pi orderquantity.price%type;
  4  oq supplier.S_Q%type;
  5  begin
  6  select orderquantity.price into pi from orderquantity,supplier where supplier.S_ID=sid and rownum=1;
  7  select S_Q into oq from supplier where S_ID=sid;
  8  dbms_output.put_line('total price : '||pi*oq);
  9  end;
 10  /

Procedure created.

SQL> declare
  2      begin
  3    p2('ur3');
  4      end;
  5  /
total price : 1000000

PL/SQL procedure successfully completed.
```

3. Write a procedure raise_quantity which increases the quantity of the ordered product. It accepts an order id and quantity to be increased. It uses the order id to find the current quantity from the ORDER table and update the quantity.

```
create or replace procedure raise_quantity(q in number,o in
varchar2,q1 out number) is

begin

select distinct(order_q) into q1 from orderquantity where
order_id=o;

dbms_output.put_line('present quantity : ' || q1);

update orderquantity set order_q=(q1+q) where order_id=o;

select distinct(order_q) into q1 from orderquantity where
order_id=o;

dbms_output.put_line('updated quantity : '|| q1);

end;


declare

q2 number;

begin

raise_quantity(100,10,q2);

end;
```

```
SQL>  create or replace procedure raise_quantity(q in number,o in varchar2,q1 out number) is
  2  begin
  3  select distinct(order_q) into q1 from orderquantity where order_id=o;
  4  dbms_output.put_line('present quantity : ' || q1);
  5  update orderquantity set order_q=(q1+q) where order_id=o;
  6  select distinct(order_q) into q1 from orderquantity where order_id=o;
  7  dbms_output.put_line('updated quantity : '|| q1);
  8  end;
  9  /

Procedure created.

SQL> declare
  2  q2 number;
  3  begin
  4  raise_quantity(100,10,q2);
  5  end;
  6  /
present quantity : 200
updated quantity : 300

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL function STATUS to return value SUPPLIED if the product number passed to it is available in the Supplier table else will return NOT SUPPLIED.

```
create or replace function status(p_id in varchar2) return
varchar2 is

a varchar2(30);

b varchar2(30);
```

```
cursor br is
select distinct(p_id)  from supplier where p_id=p_id;
begin
open br;
fetch br into b;

if p_id=b then
 a:='supplied';
else
 a:='notsupplied';
end if;
return a;
end;

declare
c varchar2(100);
begin
c:=status(21);
dbms_output.put_line(c);
end;
```

```
SQL> create or replace function status(p_id in varchar2) return varchar2 is
  2  a varchar2(30);
  3  b varchar2(30);
  4  cursor br is
  5  select distinct(p_id)  from supplier where p_id=p_id;
  6  begin
  7  open br;
  8  fetch br into b;
  9
 10  if p_id=b then
 11   a:='supplied';
 12  else
 13   a:='notsupplied';
 14  end if;
 15  return a;
 16  end;
 17  /

Function created.

SQL> declare
  2  c varchar2(10);
  3  begin
  4  c:=status(221);
  5  dbms_output.put_line(c);
  6  end;
  7  /
supplied

PL/SQL procedure successfully completed.
```

5. Write a PL/SQL function to return the product name when the product id is passed as an argument.

```
create or replace function f1(p_id in varchar2)
return varchar2 is
pd varchar2(20);
begin
select product_description into pd from product where
product_id=p_id;
return pd;
end;


declare
a varchar2(20);
begin
a:=f1(221);
dbms_output.put_line(a);
end;
```

```
SQL> create or replace function f1(p_id in varchar2)
  2  return varchar2 is
  3  pd varchar2(20);
  4  begin
  5  select product_description into pd from product where product_id=p_id;
  6  return pd;
  7  end;
  8  /

Function created.

SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:=f1(221);
  5  dbms_output.put_line(a);
  6  end;
  7  /
chair

PL/SQL procedure successfully completed.
```

6. Write a PL/SQL function to return the maximum quantity ordered by the customers.

```
create or replace function f2
return number is
q number;
begin
select max(order_q) into q from orderquantity;
return q;
```

```
end;


declare
a number;
begin
a:=f2();
dbms_output.put_line(a);
end;
```

```
SQL> create or replace function f2
  2  return number is
  3  q number;
  4  begin
  5  select max(order_q) into q from orderquantity;
  6  return q;
  7  end;
  8  /

Function created.

SQL>
SQL> declare
  2  a number;
  3  begin
  4  a:=f2();
  5  dbms_output.put_line(a);
  6  end;
  7  /
300

PL/SQL procedure successfully completed.
```

Result:

Functions and procedures were successfully created and tested for all the different situations

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 31-10-16

EXPERIMENT-NO 9

Video Link : https://youtu.be/kCpEj4YB7yU

AIM:
To create indexes and procedures for the following requirements.

DESCRIPTION:
Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

Sequence is a feature supported by some database systems to produce unique values on demand. Some DBMS like MySQL supports AUTO_INCREMENT in place of Sequence. AUTO_INCREMENT is applied on columns, it automatically increments the column value by 1 each time a new record is entered into the table. Sequence is also some what similar to AUTO_INCREMENT but its has some extra features.

Program:

1. Create Index on city column of the customer table.

```
create INDEX I1 ON customer(city);
```

2. Create index on order_id column of the order table.

```
create INDEX I2 ON order1(customer_id);
```

3.Create a composite index on product table on attributes product_id and product_description.

```
create index i3 ON product(product_id,product_description);
```

4. Create a functional index on customer table on attribute cname(upper case)

```
create index i4 ON customer(UPPER(customer_name));
```

5.Create a bit map index on the gender field of the supplier table.

```
create bitmap index i5 on supplier(gender);
```

6.Drop the last created Index.

```
Drop index i5
```

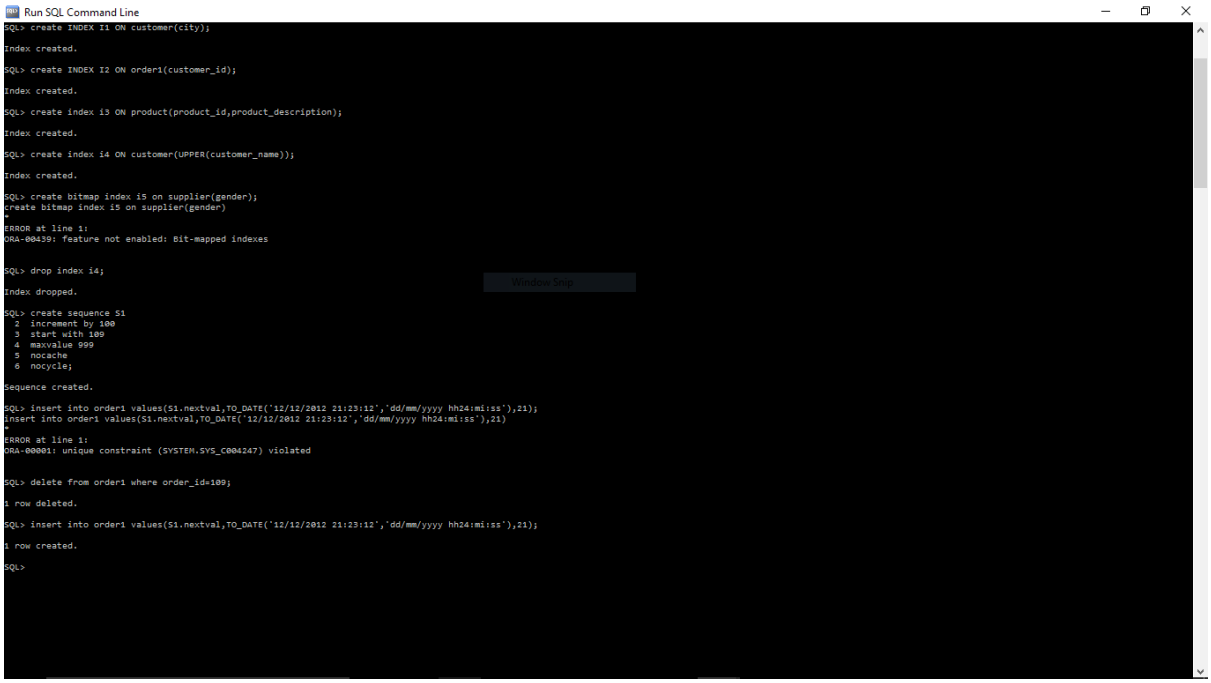
7. Create a sequence in the order-id of the order table.

```
create sequence S1
increment by 100
start with 109
maxvalue 999
nocache
nocycle;
```

8. Insert values into the order table and for order_id use the sequence number.

```
insert into order1 values(S1.nextval,TO_DATE('12/12/2012
21:23:12','dd/mm/yyyy hh24:mi:ss'),21);
```

Output:



Result:

Indexes and sequences were successfully created and tested for all the different situations

Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 31-10-2016

EXPERIMENT-NO 10

Video Link : https://youtu.be/B1bHFoHzv6s

Aim

To design a multi paged hotel reservation in php using super global variables.

Description:

In this exercise we will be creating multiple web pages using php super global variables.

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_COOKIE
- $_SESSION

Program:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Register New User</title>
       <script>
               function check_pass() {
        var1=document.getElementById("pass").value;
        var2=document.getElementById("repass").value;
        if(var1!==var2)
        {
            alert("Password Not Matched");
            return false;
        }
        else
        return true;
       }

            </script>
```

```
    </head>
    <body>
        <h1>TO RESERVE ROOM, SIGN UP</h1>
        <form  method="POST" action="ServerScript.php" onsubmit="return
check_pass();">

                Username:<br>
                <input type="text" name="username" id="username"
required><br><br>

                Email Address:<br>
                <input type="text" name="email" id="email"
required><br><br>

                Password:<br>
                <input type="password" name="pass" id="pass"
required><br><br>

                Re-type Password:<br>
                <input type="password" name="repass" id="repass"
required><br><br>

          <input type="submit" value="Register">

        </form>
        <h3>Already a user login <a href="Login.php">here</a></h3>

    </body>
</html>

//serverscript.php
<?php

$con= mysqli_connect("localhost", "root","","weblab") or
die(mysql_error());

$username=$_POST['username'];
$email=$_POST['email'];
$password= sha1($_POST['pass']);

$query="insert into users values ('$username','$email','$password')";
$result=mysqli_query($con,$query) or die(throw_error(mysqli_error($con)));

?>

//login.php

<!DOCTYPE html>
<html>
      <head>
            <title>Welcome to the login page</title>
            <link rel="stylesheet" type="text/css" href="login.css">
      </head>
      <body>
            <form action="selection.php" method="get">
                  <fieldset>
                        <legend>Login</legend>
                        <input type="text" name="name" id="loginName"
placeholder="Enter your name" required>
                        <br/>
                        <input type="submit" name="login" value="Enter"
id="loginButton">
                  </fieldset>
            </form>
            <script type="text/javascript">
                  window.onload = function(){
                        document.getElementById("loginName").value = "";
                  };
            </script>
      </body>
</html>
```

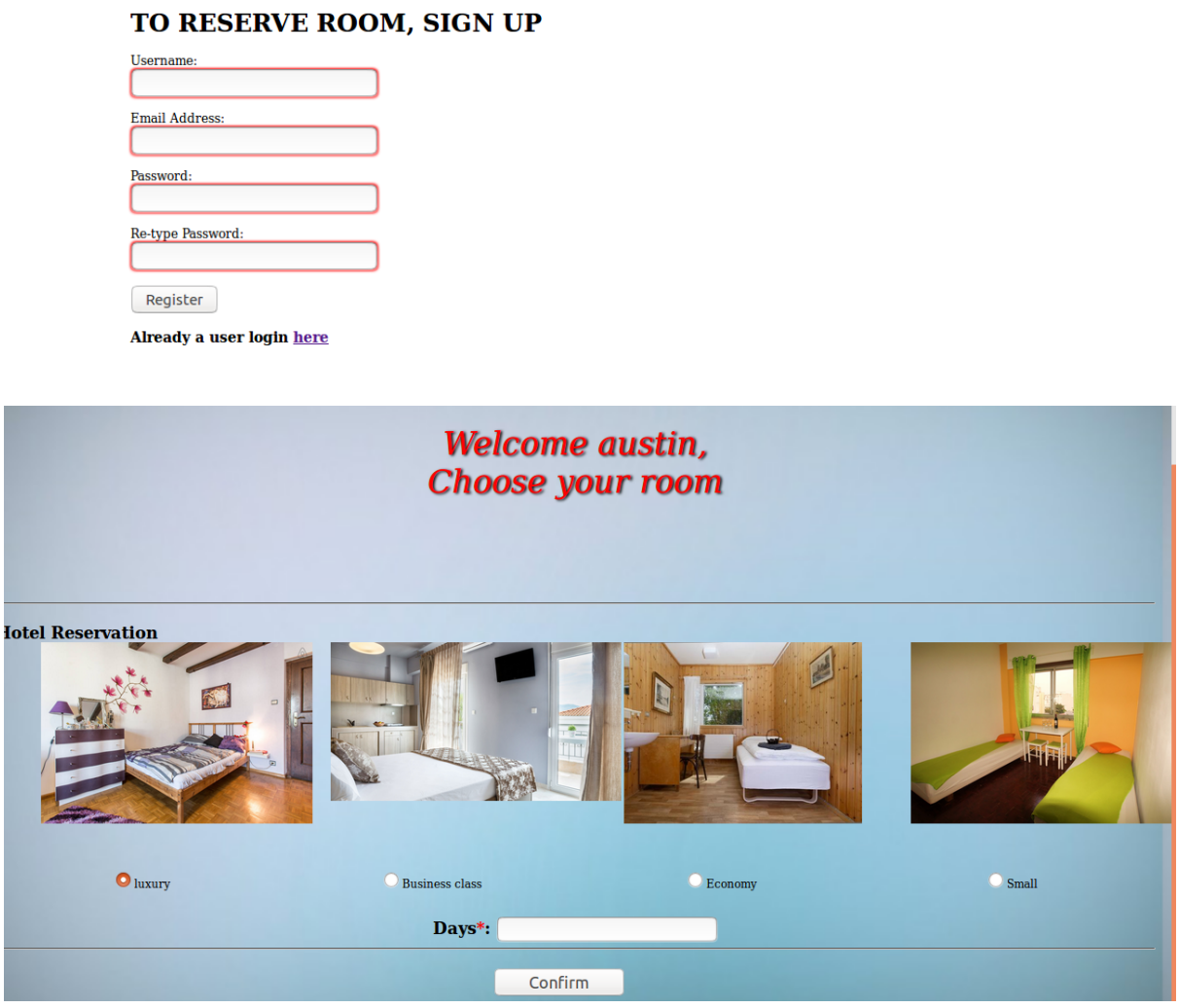
```
//validatelogin.php
<?php

 $con= mysqli_connect("localhost", "root","","weblab") or
die(mysql_error());

      if(isset($_COOKIE['username'])){
      $username=$_COOKIE['username'];
      $query="select * from users where Username='".$username."'";
      }

      else{
      $username=$_POST['username'];
      $password=$_POST['pass'];
      $en_cr=sha1($password);
      $query="select * from users where username='".$username."' and
pass='".$en_cr."'";
      }

      $result=mysqli_query($con,$query);

      if($result!=true){
      echo "Error2: ".mysql_error()."<br>";
      }

      if(mysqli_num_rows($result)>0){

          session_start();
          if(isset($_POST['remember'])){
              setcookie("username",$username,time()-3600);
          }
          echo "Login Successfull<br>";
          echo "Welcome ".$username;
          $_SESSION["name"]=$username;
          header('location:selection.php');
      }
      else{

          echo "<script>alert(\"Invalid Username or
Password\")</script><br>";
          header('location:Login.php');
        }
?>

//selection.php
<!DOCTYPE html>
<html>
      <head>
            <title>Room selection</title>
            <link rel="stylesheet" type="text/css" href="selection.css">
      </head>
      <body>
            <div id="banner">
                  <?php
                        session_start();
                        echo "<p>Welcome ".$_SESSION["name"].",<br/>Choose
your room</p>";
                  ?>
            </div>
            <hr/>
            <br/>
            <form action="details.php" method="post">
                  <span class="heading">Hotel Reservation</span></span><br>
                  <div id="options">
                        <div class="optionDetails">
                              <div class="optionImg">
                                    <img src="l.jpeg" width=""
height="80%">
                              </div>
                              <input type="radio" name="room"
value="luxury" checked>luxury
                        </div>
```

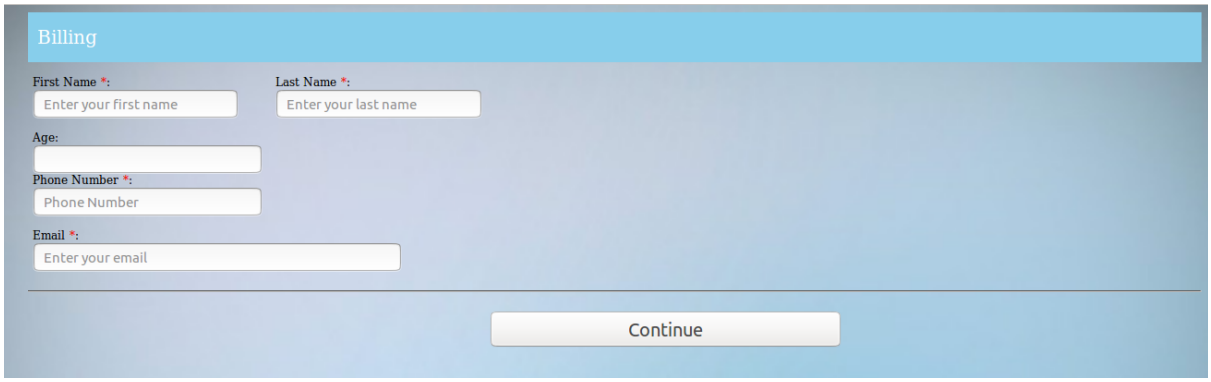
```
                        <div class="optionDetails">
                              <div class="optionImg">
                                    <img src="lmedium.jpg" width=""
height="70%">
                              </div>
                              <input type="radio" name="room"
value="Business class">Business class
                        </div>
                        <div class="optionDetails">
                              <div class="optionImg">
                                    <img src="msmall.jpg" width=""
height="80%">
                              </div>
                              <input type="radio" name="room"
value="Economy">Economy
                        </div>
                        <div class="optionDetails">
                              <div class="optionImg">
                                    <img src="small.jpeg" width=""
height="80%">
                              </div>
                              <input type="radio" name="room"
value="Small">Small
                        </div>
                  </div>

                  <div class="h">
                        <br/><br/>
                        <span class="heading">Days<span
class="asterix">*</span>:
                              <input type="text" name="days" required>
                        </span>
                  </div>
                  <hr/><br/>
                  <input type="submit" id="next" value="Confirm">
            </form>
      </body>
</html>

//details.php
<!DOCTYPE html>
<html>
<head>
      <title>Details</title>
      <link rel="stylesheet" type="text/css" href="shipping.css">
</head>
<body>
      <?php
            session_start();
            $_SESSION['days']=$_POST['days'];
            $_SESSION['room']=$_POST['room'];
      ?>
      <form method="post" action="confirm.php">
            <div>
                  <div class="header">Billing</div>
                  <div class="sections">
                        <div class="names">
                              First Name<span
class="asterix">*</span>:<br/>
                              <input type="text" name="first"
placeholder="Enter your first name" class="namesInput">
                        </div>
                        <div class="names">
                              Last Name<span class="asterix">*</span>:<br/>
                              <input type="text" name="last"
placeholder="Enter your last name" class="namesInput">
                        </div>
                        <br/><br/>
                        <div >
                              Age:<br/>
                              <input type="text" name="age" required>
                        </div>
```

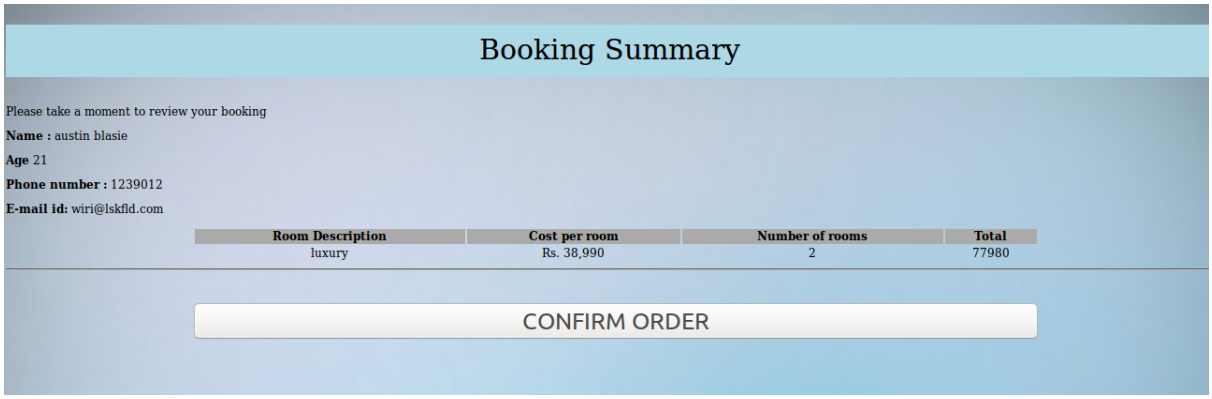
```
                        <div class="zipPhone">
                              Phone Number<span
class="asterix">*</span>:<br/>
                              <input type="text" name="number"
placeholder="Phone Number" required>
                        </div>
                        <br/><br/>
                        <div class="mail">
                              Email<span class="asterix">*</span>:<br/>
                              <input type="email" name="email"
placeholder="Enter your email" class="email" required>
                        </div>
                        <br/>
                        <hr/><br/>
                        <input type="submit" value="Continue" name="pay"
id="pay"/>
                  </div>
            </div>
      </form>
</body>
</html>

//confirm.php
<!DOCTYPE html>
<html>
<head>
      <title>Order Summary</title>
      <link rel="stylesheet" type="text/css" href="confirm.css">
</head>
<body>
      <p id="banner">Booking Summary</p>
      <p>Please take a moment to review your booking</p>
      <?php
            $con= mysqli_connect("localhost", "root","","weblab");
            session_start();
            $cost;
            $cost_num;
            $days=$_SESSION['days'];
            $room=$_SESSION['room'];
            $name=$_POST['first'];
            $age=$_POST['age'];
            $number=$_POST['number'];
            $email=$_POST['email'];
            switch($_SESSION['room']){
                  case 'luxury': $cost = 'Rs. 38,990'; $cost_num = 38990;
break;
                  case 'Business class': $cost = 'Rs. 17,590'; $cost_num =
17590; break;
                  case 'Economy': $cost = 'Rs. 12,857'; $cost_num = 12857;
break;
                  case 'Small': $cost = 'Rs. 7,499'; $cost_num = 7499;
break;
            }
            $total_cost = $cost_num*$_SESSION['days'];
            mysqli_query($con,"INSERT INTO orders
VALUES('$room','$days','$name','$age','$number','$email','$total_cost')");
            echo "<div>
            <p>
                  <span class='bold'>Name       :</span> ".
$_POST['first']." ".$_POST['last']."
            </p>
            <p>
                  <span class='bold'>Age</span> ".$_POST['age']."
            </p>
            <p>
                  <span class='bold'>Phone number :</span> ".
$_POST['number']."
            </p>
            <p>
                  <span class='bold'>E-mail id:</span> ".$_POST['email']."
            </p>
```

```
        </div>".
        "<table id='table'>
              <tr id='head'>
                    <th>Room Description</td>
                    <th>Cost per room</td>
                    <th>Number of rooms</td>
                    <th>Total</td>
              </tr>
              <tr>
                    <td>".$_SESSION['room']."</td>
                    <td>".$cost."</td>
                    <td>".$_SESSION['days']."</td>
                    <td>".$total_cost."</td>
              </tr>

        </table>"
        ?>
        <hr>
        <br/><br/>
        <a href="thankyou.php"><button>CONFIRM ORDER</button></a>
</body>
</html>
```

Output:







Result:

The webpages using php super global variables has been created successfully.