

1. Consider the following schema for Order Database:

SALESMAN (*Salesman_id*, Name, City, Commission)

CUSTOMER (*Customer_id*, Cust_Name, City, Grade, Salesman_id)

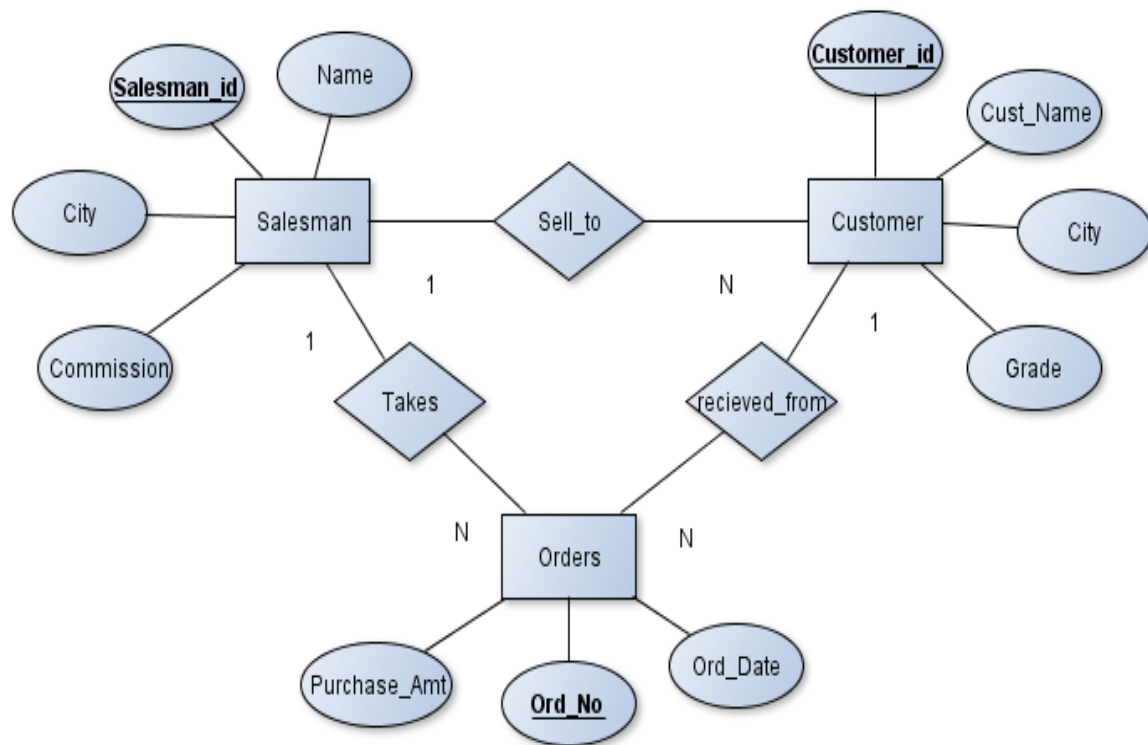
ORDERS (*Ord_No*, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

Write SQL queries to

1. Count the customers with grades above Bangalore's average.
2. Find the name and numbers of all salesmen who had more than one customer.
3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)
4. Create a view that finds the salesman who has the customer with the highest order of a day.
5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

Solution:

Entity-Relationship Diagram



Schema Diagram

Salesman

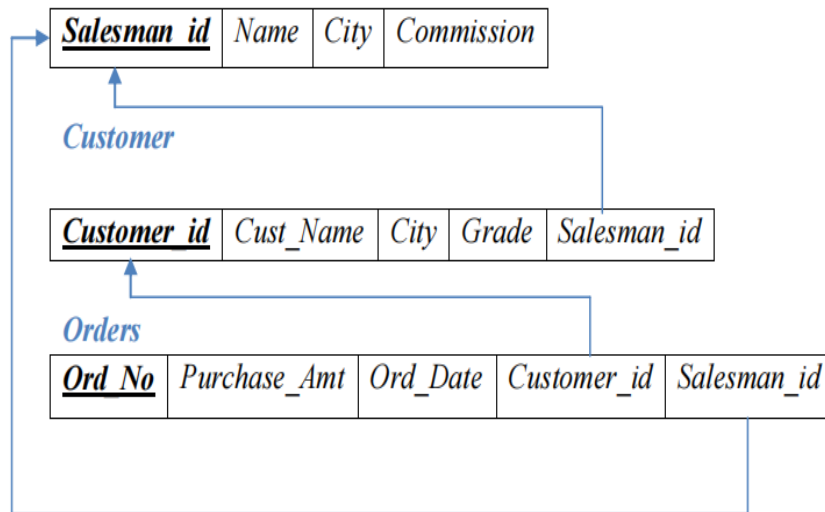


Table Creation

```
CREATE TABLE SALESMAN (  
SALESMAN_ID INT (4),  
NAME VARCHAR (20),  
CITY VARCHAR (20),  
COMMISSION VARCHAR (20),  
PRIMARY KEY (SALESMAN_ID));
```

```
CREATE TABLE CUSTOMER  
(CUSTOMER_ID INT (4),  
CUST_NAME VARCHAR (20),  
CITY VARCHAR (20),  
GRADE INT (3),  
Salesman_id int (4),  
PRIMARY KEY (CUSTOMER_ID),  
FOREIGN KEY (salesman_id)  
REFERENCES SALESMAN (SALESMAN_ID) ON DELETE SET NULL);
```

```
CREATE TABLE ORDERS  
(ORD_NO INT (5),  
PURCHASE_AMT DECIMAL (10, 2),  
ORD_DATE DATE,  
customer_id int,  
salesman_id int,  
PRIMARY KEY (ORD_NO),  
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID) ON  
DELETE CASCADE,  
FOREIGN KEY (SALESMAN_ID) REFERENCES SALESMAN (SALESMAN_ID) ON  
DELETE CASCADE);
```

Table Descriptions

DESC SALESMAN;

SQL> DESC SALESMAN;

Name	Null?	Type
SALESMAN_ID	NOT NULL	NUMBER(4)
NAME		VARCHAR2(15)
CITY		VARCHAR2(15)
COMMISSION		NUMBER(3,2)

DESC CUSTOMER;

SQL> DESC CUSTOMER ;

Name	Null?	Type
CUSTOMER_ID	NOT NULL	NUMBER(4)
CUST_NAME		VARCHAR2(15)
CITY		VARCHAR2(15)
GRADE		NUMBER(3)
SALESMAN_ID		NUMBER(4)

DESC ORDERS;

SQL> DESC ORDERS;

Name	Null?	Type
ORD_NO	NOT NULL	NUMBER(5)
PURCHASE_AMT		NUMBER(10,2)
ORD_DATE		DATE
CUSTOMER_ID		NUMBER(4)
SALESMAN_ID		NUMBER(4)

Insertion of Values to Tables

INSERT INTO SALESMAN VALUES (1000, 'JOHN', 'BANGALORE', '25%');
INSERT INTO SALESMAN VALUES (2000, 'RAVI', 'BANGALORE', '20%');
INSERT INTO SALESMAN VALUES (3000, 'KUMAR', 'MYSORE', '15%');
INSERT INTO SALESMAN VALUES (4000, 'SMITH', 'DELHI', '30%');
INSERT INTO SALESMAN VALUES (5000, 'HARSHA', 'HYDRABAD', '15%');

INSERT INTO CUSTOMER VALUES (10, 'PREETHI', 'BANGALORE', 100, 1000);
INSERT INTO CUSTOMER VALUES (11, 'VIVEK', 'MANGALORE', 300, 1000);
INSERT INTO CUSTOMER VALUES (12, 'BHASKAR', 'CHENNAI', 400, 2000);
INSERT INTO CUSTOMER VALUES (13, 'CHETHAN', 'BANGALORE', 200, 2000);
INSERT INTO CUSTOMER VALUES (14, 'MAMATHA', 'BANGALORE', 400, 3000);

INSERT INTO ORDERS VALUES (50, 5000, '04-MAY-17', 10, 1000);

```

INSERT INTO ORDERS VALUES (51, 450, '20-JAN-17', 10, 2000);
INSERT INTO ORDERS VALUES (52, 1000, '24-FEB-17', 13, 2000);
INSERT INTO ORDERS VALUES (53, 3500, '13-APR-17', 14, 3000);
INSERT INTO ORDERS VALUES (54, 550, '09-MAR-17', 12, 2000);

```

Display the records of tables

```
SELECT * FROM SALESMAN;
```

SALESMAN_ID	NAME	CITY	COMMISSION
1000	JOHN	BANGALORE	25 %
2000	RAVI	BANGALORE	20 %
3000	KUMAR	MYSORE	15 %
4000	SMITH	DELHI	30 %
5000	HARSHA	HYDRABAD	15 %

```
SELECT * FROM CUSTOMER;
```

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
10	PREETHI	BANGALORE	100	1000
11	VIVEK	MANGALORE	300	1000
12	BHASKAR	CHENNAI	400	2000
13	CHETHAN	BANGALORE	200	2000
14	MAMATHA	BANGALORE	400	3000

```
SELECT * FROM ORDERS;
```

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
50	5000	04-MAY-17	10	1000
51	450	20-JAN-17	10	2000
52	1000	24-FEB-17	13	2000
53	3500	13-APR-17	14	3000
54	550	09-MAR-17	12	2000

Queries:

1. Count the customers with grades above Bangalore's average.

```
SELECT GRADE, COUNT(DISTINCT CUSTOMER_ID)
FROM CUSTOMER1
GROUP BY GRADE
HAVING GRADE > (SELECT AVG(GRADE)
FROM CUSTOMER1
WHERE CITY='BANGALORE');
```

Output:

GRADE	COUNT(DISTINCTCUSTOMER_ID)
300	1
400	2

2. Find the ID and name of all salesmen who had more than one customer.

```
SELECT SALESMAN_ID, NAME
FROM SALESMAN A
WHERE 1 < (SELECT COUNT(*)
FROM CUSTOMER
WHERE SALESMAN_ID=A.SALESMAN_ID);
```

Output:

	SALESMAN_ID	NAME
▶	1000	JOHN
	2000	RAVI
•	NULL	NULL

3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation) along with commission details.

```
SELECT SALESMAN.SALESMAN_ID, NAME, CUST_NAME, COMMISSION
FROM SALESMAN, CUSTOMER
WHERE SALESMAN.CITY = CUSTOMER.CITY
UNION
SELECT SALESMAN_ID, NAME, 'NO MATCH', COMMISSION
FROM SALESMAN
WHERE NOT CITY = ANY
(SELECT CITY
FROM CUSTOMER)
ORDER BY 2 DESC;
```

SALESMAN_ID	NAME	CUST_NAME	COMMISSION
4000	SMITH	NO MATCH	30 %
2000	RAVI	CHETHAN	20 %
2000	RAVI	MAMATHA	20 %
2000	RAVI	PREETHI	20 %
3000	KUMAR	NO MATCH	15 %
1000	JOHN	CHETHAN	25 %
1000	JOHN	MAMATHA	25 %
1000	JOHN	PREETHI	25 %
5000	HARSHA	NO MATCH	15 %

4.

Create a view that finds the salesman who has the customer with the highest order of a day.

```
CREATE VIEW SALESMAN_HIGH_ORDER AS
SELECT B.ORD_DATE, A.SALESMAN_ID, A.NAME
FROM SALESMAN A, ORDERS B
WHERE A.SALESMAN_ID = B.SALESMAN_ID
AND B.PURCHASE_AMT=(SELECT MAX(PURCHASE_AMT)
FROM ORDERS C
WHERE C.ORD_DATE = B.ORD_DATE);
```

To display the created view table records.

Select * from SALESMAN_HIGH_ORDER;

ORD_DATE	SALESMAN_ID	NAME
04-MAY-17	1000	JOHN
20-JAN-17	2000	RAVI
24-FEB-17	2000	RAVI
13-APR-17	3000	KUMAR
09-MAR-17	2000	RAVI

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

Use ON DELETE CASCADE at the end of foreign key definitions while creating child table orders and then execute the following:

Use ON DELETE SET NULL at the end of foreign key definitions while creating child table customers and then executes the following:

```
DELETE FROM SALESMAN
WHERE SALESMAN_ID=1000;
```

```
SQL> DELETE FROM SALESMAN  
2 WHERE SALESMAN_ID=1000;
```

1 row deleted.

```
SQL> SELECT * FROM SALESMAN;
```

SALESMAN_ID	NAME	CITY	COMMISSION
2000	RAVI	BANGALORE	20 %
3000	KUMAR	MYSORE	15 %
4000	SMITH	DELHI	30 %
5000	HARSHA	HYDRABAD	15 %