CS443 -- A3

Write the queries necessary to obtain the required information. Make sure all columns you return have descriptive column headings. You must show the result of the query. For example, if the query is:

- ⇨ Show the office id, the city, and the region
- ⇨ Your query should be:

  select office, city, region
  from offices;
- ⇨ and then you need to show the following on the screen: (snapshot of your result)

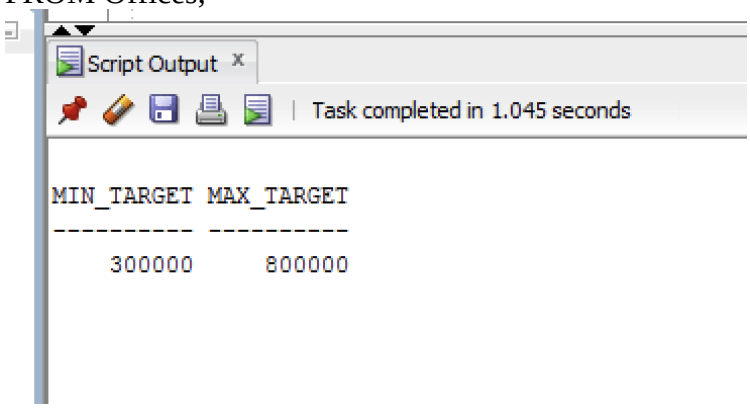
```
OFFICE CITY            REGION
------- --------------- ----------
     22 Denver          Western
     11 New York        Eastern
     12 Chicago         Eastern
     13 Atlanta         Eastern
     21 Los Angeles     Western
```

In this assignment, some questions require using "Delete", "Insert", and "Update". Thus, if a table is modified, the modification may be cascaded to other questions. To avoid that, if a question changes table X, before doing the next question, drop and recreate table X. This ensures that all questions will be executed on original unchanged tables.

The following are the questions for A3.

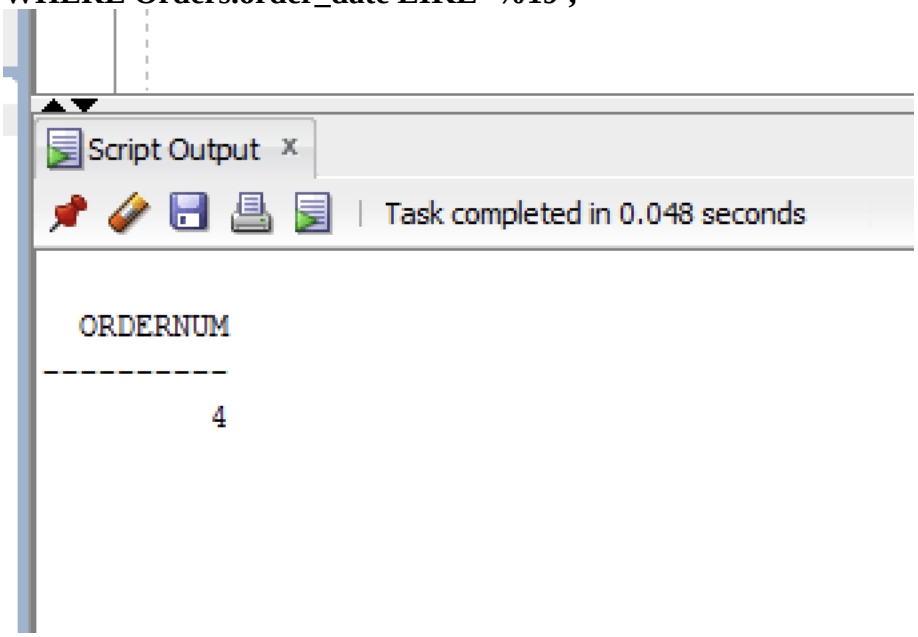
---

1) Return the Minimum and Maximum Target for all offices.
   SELECT MIN(target) AS Min_Target, MAX(target) AS Max_Target
   FROM Offices;

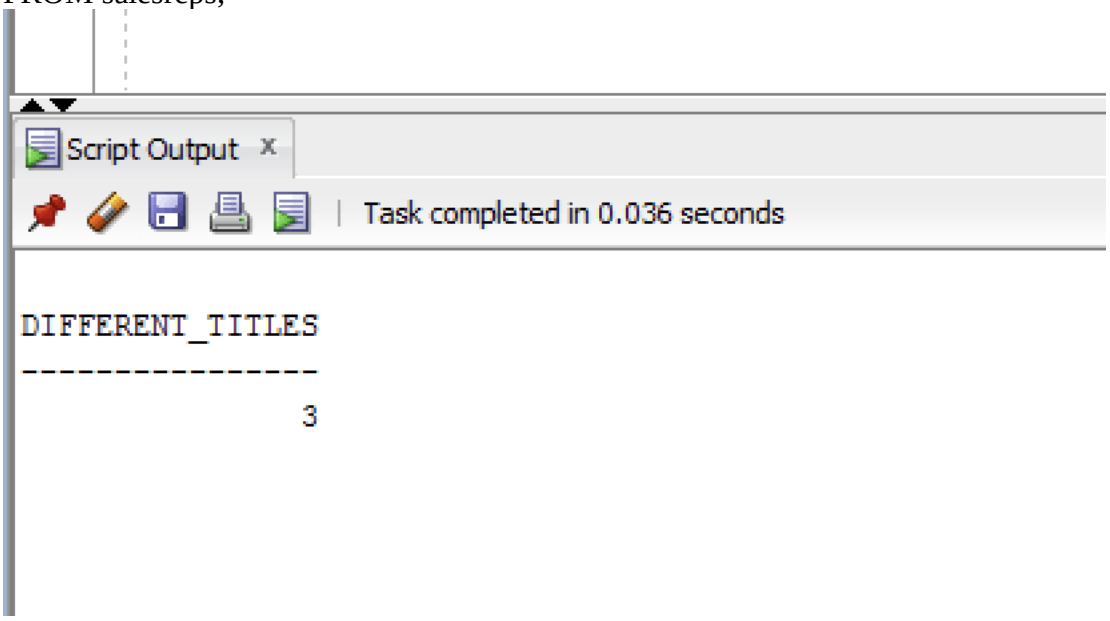


2) Determine how many orders were made in 2019. Return the number of rows that meet this condition.
   SELECT COUNT(*) AS orderNUM

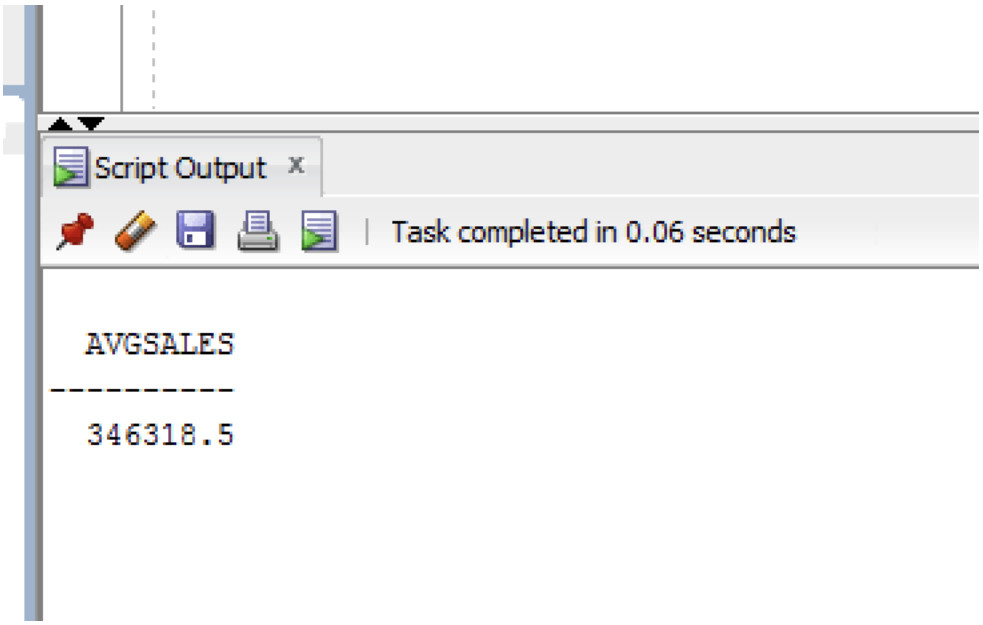
FROM Orders
WHERE Orders.order_date LIKE '%19';



3) How many different titles in the sales reps table.
SELECT COUNT(DISTINCT title) AS Different_Titles
FROM salesreps;

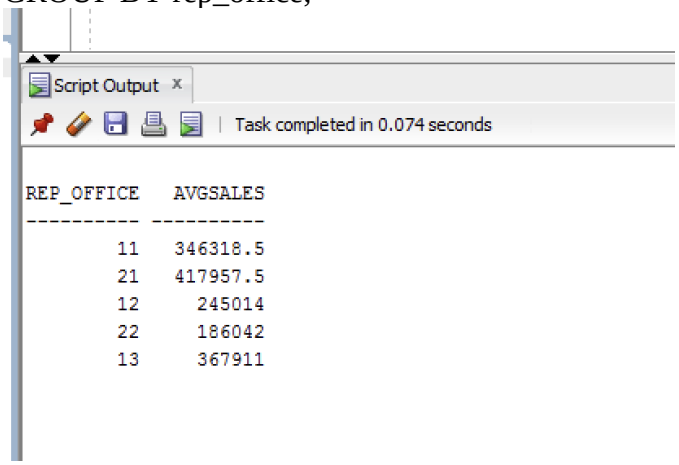


4) What is the average sales for salesreps in office 11.
SELECT AVG(sales) AS Avg_Sales_Office_11
FROM salesreps
WHERE office_id = 11;



5) What is the average sale amount for each sale reps in each office. Null should be ignored

```
SELECT SalesReps.rep_office, AVG(SalesReps.sales) AS avgSales
FROM SalesReps, Offices
WHERE SalesReps.rep_office = Offices.office
AND SalesReps.sales IS NOT NULL
GROUP BY rep_office;
```



6) For each salesrep that has made an order, list the minimum, maximum and average order amount for all their orders. Include only those orders made anytime from 2020-2021. Omit from the list any salesrep that has only made 1 order in this time frame. Sort the results by Empl_Num.

```
SELECT ord1.rep,
    MIN(ord1.amount) AS MinAmount,
    MAX(ord1.amount) AS MaxAmount,
    AVG(ord1.amount) AS AvgAmount
```

```
FROM Orders ord1
WHERE ord1.order_date BETWEEN TO_DATE('01-JAN-2020', 'DD-MON-YYYY')
AND TO_DATE('31-DEC-2021', 'DD-MON-YYYY')
GROUP BY ord1.rep
HAVING COUNT(ord1.order_num) > 1
ORDER BY ord1.rep;
```

Script Output

Task completed in 0.083 seconds

REP	MINAMOUNT	MAXAMOUNT	AVGAMOUNT
102	3750	15000	9375
108	760	45000	22880

7) Use a sub-query to list the Customer Number, Name, and Credit Limit of any customers who have exceeded their credit limit (amount > credit limit) on any order.

```
SELECT cust_num , company, credit_limit
FROM Customers
WHERE credit_limit < ANY (
SELECT orders.amount
FROM Orders
WHERE Orders.cust = Customers.cust_num);
```

Script Output

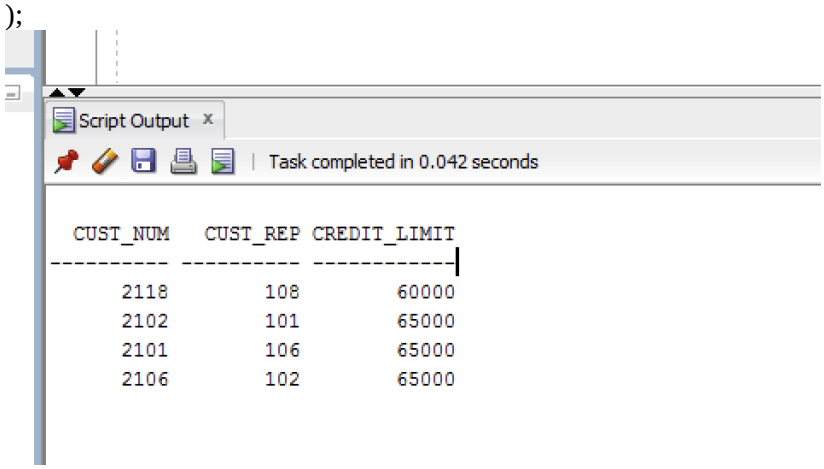
Task completed in 0.086 seconds

CUST_NUM	COMPANY	CREDIT_LIMIT
2109	Chen Associates	25000
2113	Ian and Schmidt	20000

8) Use a subquery and using the “all” keyword to find the customer number, Salesrep id, and CreditLimit of every customer whose CreditLimit is larger than the CreditLimit of all of the customers of sales rep number 109.

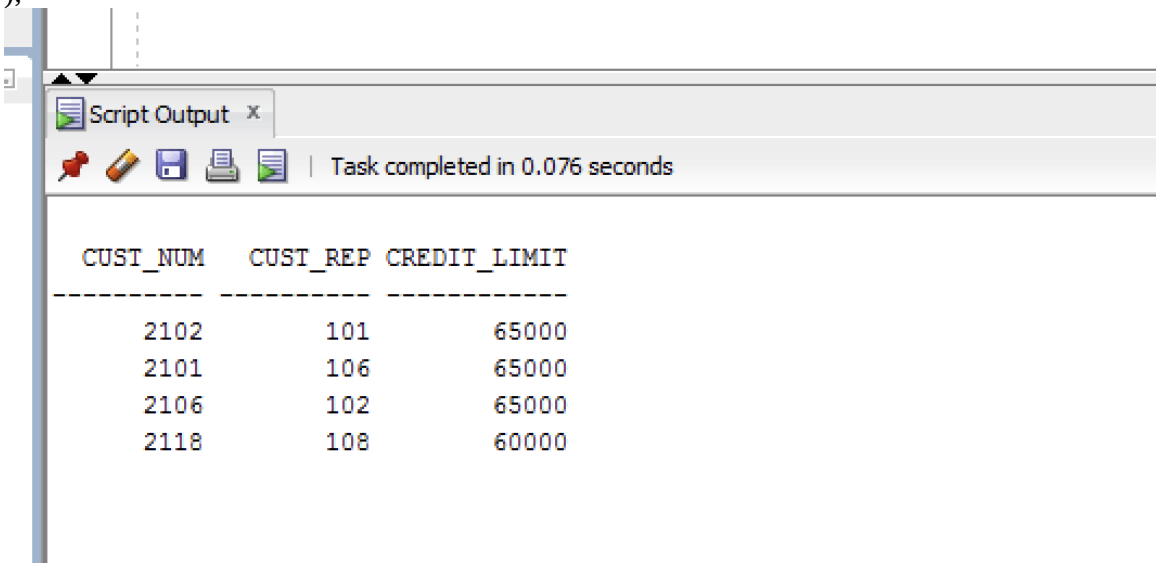
```
SELECT cust_num , Customers.cust_rep, Credit_limit
```

```
FROM Customers
WHERE Credit_limit > ALL
(
SELECT Credit_limit
FROM Customers
WHERE Customers.cust_rep = 109
);
```



9) Do question 8, still using the subquery but do not use the "all" keyword.

```
SELECT cust_num , Customers.cust_rep, Credit_limit
FROM Customers
WHERE Credit_limit >
(
SELECT MAX(Credit_limit)
FROM Customers
WHERE Customers.cust_rep = 109
);
```



10) Use sub query and "in" keyword to print the salesreps (ids) who have taken order for the companies starts with letter 'Z' or with letter 'J'. Duplicate rows are not allowed

```
SELECT DISTINCT empl_num
FROM SalesReps
WHERE empl_num IN
(
```

```
SELECT Customers.cust_rep
FROM Customers
```

Script Output

Task completed in 0.043 seconds

```
  EMPL_NUM
----------
       103
       106
       108
```

```
);
```

11) Use subquery to find the id and the name of every sales rep that represents at least one customer with a credit limit of greater than $60,000.

```
SELECT SALESREPS.EMPL_NUM, SALESREPS.NAME
FROM SALESREPS
WHERE SALESREPS.EMPL_NUM IN
(
SELECT SALESREPS.EMPL_NUM
FROM CUSTOMERS
WHERE CREDIT_LIMIT > 60000
);
```

Script Output

Task completed in 0.038 seconds

```
  EMPL_NUM NAME
---------- ---------------
       105 Bill Adams
       109 Mary Jones
       102 Sue Smith
       106 Sam Clark
       104 Bob Smith
       101 Dan Roberts
       110 Tom Synder
       108 Larry Fitch
       103 Paul Cruz
       107 Nacy Angelli
```

12) Use sub query and keyword “exists” to list the id and the name of the salesreps in which some customers have orders some products in their hiredate.

```
SELECT sr.empl_num, sr.name
FROM Salesreps sr
WHERE EXISTS (
   SELECT 1
   FROM Orders order
   WHERE order.rep = sr.empl_num
   AND order.order_date = sr.hire_date
);
```

```
Script Output
Task completed in 0.043 seconds

  EMPL_NUM NAME
---------- ---------------
       102 Sue Smith
```

13) List all the products (only Product_ID) that have never been sold.

```
SELECT PRODUCTS.PRODUCT_ID
FROM PRODUCTS
WHERE NOT EXISTS
(
SELECT ORDERS.QTY
FROM ORDERS
WHERE ORDERS.PRODUCT = PRODUCTS.PRODUCT_ID
);
```

```
-----
XK48
887P
887X
41001
41089
XK48A
887F
41672
```

14) Insert the following information into the OFFICES table:

Office: 10 City: Miami Region: Southern Manager: 106 Sales: 0

- Target should be Null. Do not use explicit Null for the target in your insert statement.
- Show that office 10 is inserted by writing (select * from offices where office = 10)
- to revise the table to its original values
- Do (delete from offices where office = 10)

```
INSERT INTO Offices (office, city, region, mgr, sales, target)
VALUES (10, 'Miami', 'Southern', 106, 0);
SELECT * FROM Offices WHERE office = 10;
DELETE FROM Offices WHERE office = 10;
```

```
Script Output
Task completed in 0.055 seconds

1 row inserted.

    OFFICE CITY            REGION           MGR     TARGET      SALES
---------- --------------- ---------- ---------- ---------- ----------
        10 Miami           Southern          106                     0

1 row deleted.
```

15) Write an insert statement to add Your Name as Empl_Num 772. Use the date the insert is done for the hire date (sysdate). Sales is zero.

- *Other columns should remain NULL. Use explicit null to make the other fields to be null;*
- *Now delete this row to make the salesreps table goes back to its original state*

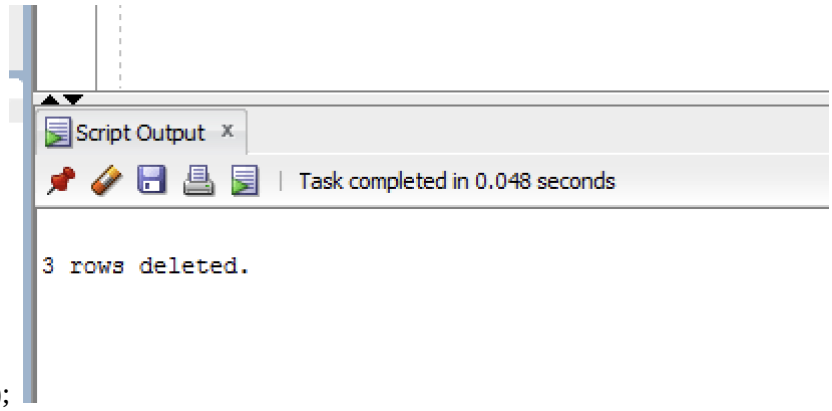
  INSERT INTO Salesreps (empl_num, name, hire_date, sales, title, rep_office, manager, quota, age)
  VALUES (772, 'ARPITA GODBOLE', SYSDATE, 0, NULL, NULL, NULL, NULL, NULL);
  DELETE FROM Salesreps WHERE empl_num = 772;

```
Script Output
Task completed in 0.073 seconds

1 row inserted.

1 row deleted.
```

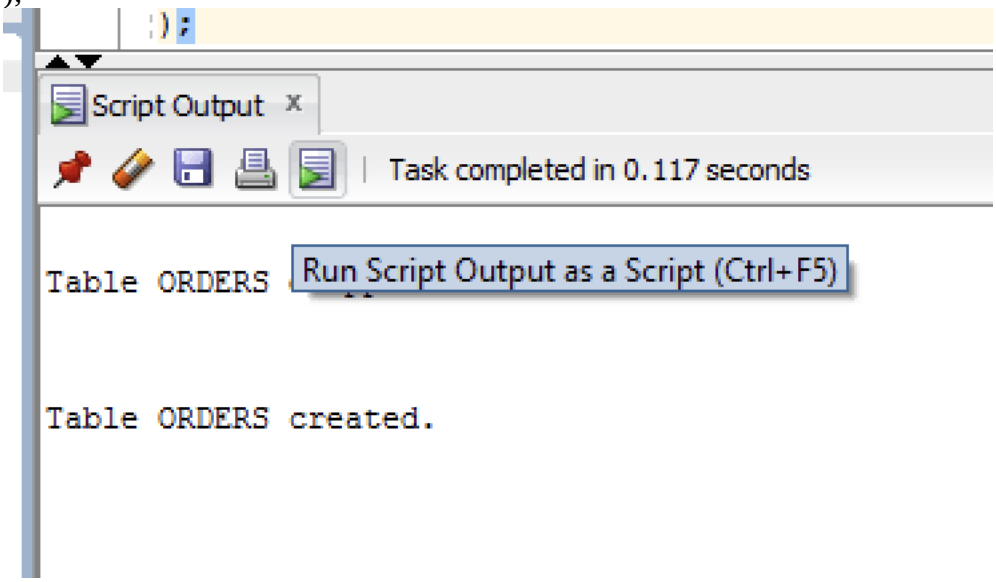
16) Use subquery to Delete all orders for employees 'Dan Roberts'.
To make the orders table back to its original state, drop the order table and recreate it with its original records
Recreate the orders table after doing the delete

```
DELETE FROM Orders
WHERE rep = (
   SELECT empl_num
```

```
    FROM Salesreps
    WHERE name = 'Dan Roberts'
```



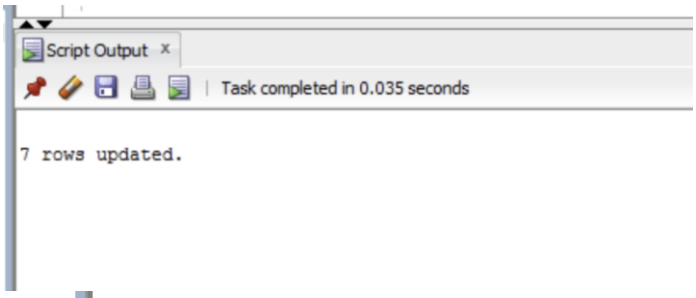

```
);
DROP TABLE Orders;
CREATE TABLE Orders (
    order_num NUMBER(6,0) PRIMARY KEY,
    order_date DATE NOT NULL,
    cust NUMBER(4,0) NOT NULL,
    rep NUMBER(3,0),
    mfr VARCHAR2(3) NOT NULL,
    product VARCHAR2(5) NOT NULL,
    qty NUMBER(4,0) NOT NULL,
    amount NUMBER(9,2) NOT NULL
);
```

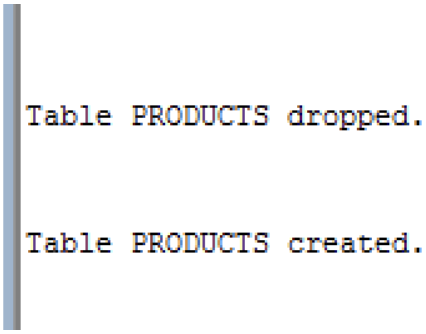


17) Lower the price of the products by 10% if they are higher the average price
    Recreate the products table after doing the update

```
UPDATE PRODUCTS
SET PRICE = PRICE * 0.9
WHERE PRODUCTS.PRICE >
(
```

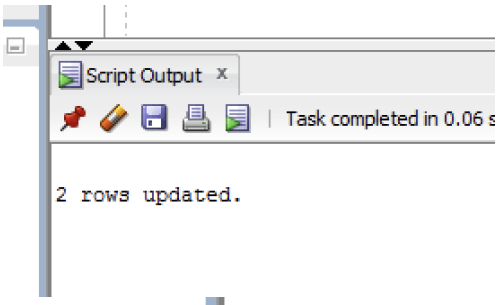
```
SELECT AVG(PRICE)
FROM PRODUCTS
);
DROP TABLE Products;
CREATE TABLE Products
(
   mfr_id VARCHAR2(3) NOT NULL,
   product_id VARCHAR2(5) NOT NULL,
   description VARCHAR2(20) NOT NULL,
   price NUMBER(9,2) NOT NULL,
   qty_on_hand NUMBER(5,0) NOT NULL
);
```

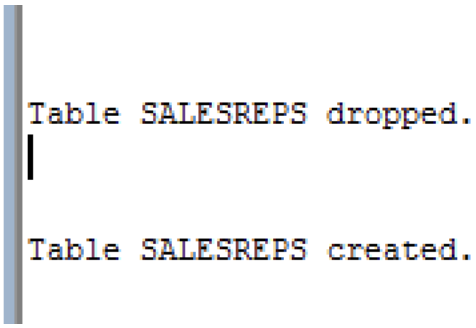




18) Set the quota of the salesreps to (average of the quota) + 1500 if they are hired in 2021.
Recreate the salesreps table after doing the update

```
UPDATE SALESREPS
SET QUOTA = 1500+
(
SELECT AVG(QUOTA)
FROM SALESREPS
)
WHERE HIRE_DATE LIKE '%21';
DROP TABLE Salesreps;

CREATE TABLE Salesreps (
   empl_num NUMBER(3,0) PRIMARY KEY,
   name VARCHAR2(15) NOT NULL,
   age NUMBER(3,0),
   rep_office NUMBER(2,0),
   title VARCHAR2(10),
   hire_date VARCHAR2(10) NOT NULL,
   manager NUMBER(3,0),
   quota NUMBER(10,2),
   sales NUMBER(10,2) NOT NULL
);
```





19) Increase customers credit limit by 25% for all customers that have 2 or more orders in which each order is more than $2250.

```
UPDATE CUSTOMERS
SET CREDIT_LIMIT = CREDIT_LIMIT * 1.25
WHERE CUST_NUM IN
(
SELECT CUSTOMERS.CUST_NUM
WHERE AMOUNT > 2250
GROUP BY CUSTOMERS.CUST_NUM
HAVING 2 <= COUNT(ORDER_NUM)
);
```

```
DROP TABLE Customers;

CREATE TABLE Customers (
   cust_num NUMBER(4,0) PRIMARY KEY,
   company VARCHAR2(20) NOT NULL,
   cust_rep NUMBER(3,0),
   credit_limit NUMBER(10,2)
);
```

```
Script Output
Task completed in 0.087 seconds

21 rows updated.

Table CUSTOMERS dropped.

Table CUSTOMERS created.
```

Recreate the customers table after doing this update

20) Increase the credit limit of any customer who has any order that exceeds their credit limit. The new credit limit should be set to their maximum order amount plus $1,000. This must be done in 1 SQL statement.
`

Recreate the customers table after doing this update

```
UPDATE Customers
SET CUSTOMERS.CREDIT_LIMIT =
(
SELECT MAX(ORDERS.AMOUNT)
FROM ORDERS
WHERE ORDERS.CUST = CUSTOMERS.CUST_NUM + 1000
)
WHERE CUSTOMERS.CREDIT_LIMIT < ANY
(
SELECT ORDERS.AMOUNT
FROM ORDERS
WHERE CUSTOMERS.CUST_NUM = ORDERS.CUST
);
DROP TABLE Customers;

CREATE TABLE Customers (
   cust_num NUMBER(4,0) PRIMARY KEY,
   company VARCHAR2(20) NOT NULL,
   cust_rep NUMBER(3,0),
   credit_limit NUMBER(10,2)
```

```
2 rows updated.
```

Script Output

Task completed in 0.083 seconds

```
Table CUSTOMERS dropped.


Table CUSTOMERS created.
```

);