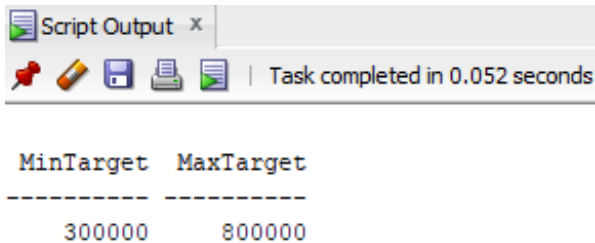


- 1) Return the Minimum and Maximum Target for all offices.

Query:

```
SELECT MIN(TARGET) AS "MinTarget", MAX(TARGET) AS "MaxTarget"
FROM OFFICES;
```

Screenshot:



The screenshot shows a 'Script Output' window with a toolbar and a status bar indicating 'Task completed in 0.052 seconds'. Below the toolbar, the query result is displayed as a table with two columns: 'MinTarget' and 'MaxTarget'. The values are 300000 and 800000 respectively.

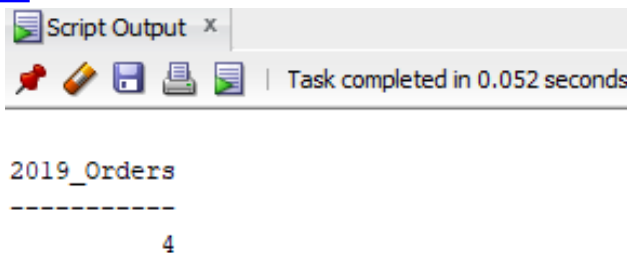
MinTarget	MaxTarget
300000	800000

- 2) Determine how many orders were made in 2019. Return the number of rows that meet this condition.

Query:

```
SELECT COUNT(ORDER_DATE) AS "2019_Orders"
FROM ORDERS
WHERE TO_CHAR(ORDER_DATE, 'YYYY') LIKE '2019'
```

Screenshot:



The screenshot shows a 'Script Output' window with a toolbar and a status bar indicating 'Task completed in 0.052 seconds'. Below the toolbar, the query result is displayed as a table with one column: '2019\_Orders'. The value is 4.

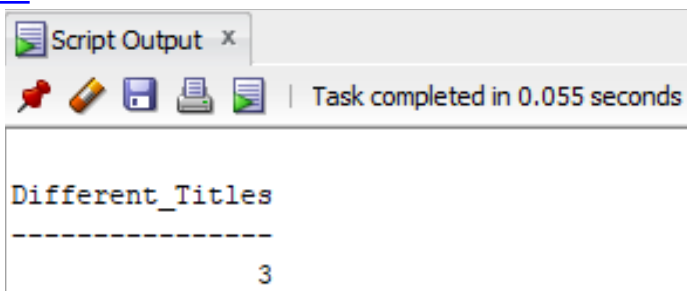
2019_Orders
4

- 3) How many different titles in the sales reps table.

Query:

```
SELECT COUNT(DISTINCT TITLE) AS "Different_Titles"
FROM SALESREPS;
```

Screenshot:



The screenshot shows a 'Script Output' window with a toolbar and a status bar indicating 'Task completed in 0.055 seconds'. Below the toolbar, the query result is displayed as a table with one column: 'Different\_Titles'. The value is 3.

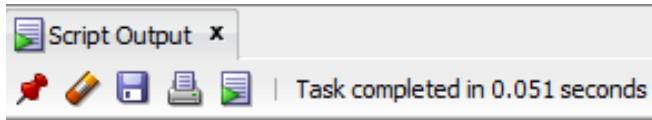
Different_Titles
3

- 4) What is the average sales for salesreps in office 22.

Query:

```
SELECT AVG(SALES) AS "Avg_Sales_Office_22"
FROM SALESREPS
WHERE REP_OFFICE = 22;
```

Screenshot:



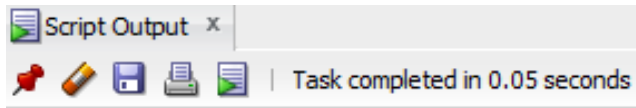
```
Avg_Sales_Office_22
-----
186042
```

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

Query:

```
SELECT REP_OFFICE, AVG(SALES) AS "Offices_Avg_Sales"
FROM SALESREPS
WHERE REP_OFFICE IS NOT NULL
GROUP BY REP_OFFICE;
```

Screenshot:



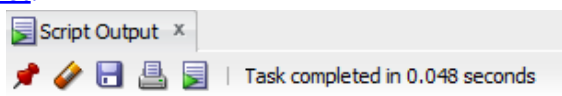
```
REP_OFFICE Offices_Avg_Sales
-----
11          346318.5
21          417957.5
12          245014
22          186042
13          367911
```

- 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.

Query:

```
SELECT EMPL_NUM, MIN(AMOUNT) AS "Order_Min", MAX(AMOUNT) AS "Order_Max",
AVG(AMOUNT) AS "Order_Avg"
FROM ORDERS, SALESREPS
WHERE REP = EMPL_NUM AND (TO_CHAR(ORDER_DATE, 'YYYY') BETWEEN 2020 AND 2021)
HAVING COUNT(ORDER_NUM) > 1
GROUP BY EMPL_NUM
ORDER BY EMPL_NUM;
```

Screenshot:



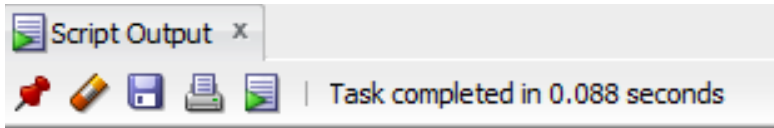
```
EMPL_NUM Order_Min Order_Max Order_Avg
-----
102      3750      15000      9375
108       760      45000     22880
```

- 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.

Query:

```
SELECT CUST_NUM, COMPANY, CREDIT_LIMIT
FROM CUSTOMERS
WHERE CREDIT_LIMIT < ANY
  (SELECT AMOUNT
   FROM ORDERS
   WHERE CUST = CUST_NUM);
```

Screenshot:



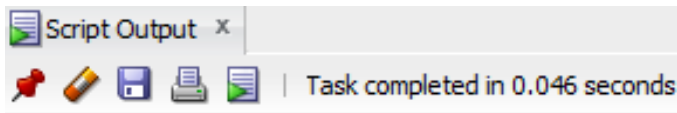
CUST_NUM	COMPANY	CREDIT_LIMIT
2109	Chen Associates	25000
2113	Ian and Schmidt	20000

- 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.

Query:

```
SELECT CUST_NUM, CUST_REP, CREDIT_LIMIT
FROM CUSTOMERS
WHERE CREDIT_LIMIT > ALL
  (SELECT CREDIT_LIMIT
   FROM CUSTOMERS
   WHERE CUST_REP = 109);
```

Screenshot:



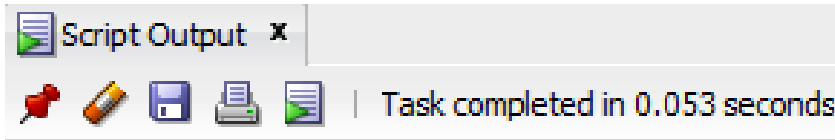
CUST_NUM	CUST_REP	CREDIT_LIMIT
2118	108	60000
2102	101	65000
2101	106	65000
2106	102	65000

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

Query:

```
SELECT CUST_NUM, CUST_REP, CREDIT_LIMIT
FROM CUSTOMERS
WHERE CREDIT_LIMIT >
      (SELECT MAX(CREDIT_LIMIT)
       FROM CUSTOMERS
       WHERE CUST_REP = 109);
```

Screenshot:



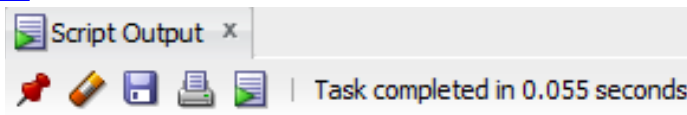
CUST_NUM	CUST_REP	CREDIT_LIMIT
2102	101	65000
2101	106	65000
2106	102	65000
2118	108	60000

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

Query:

```
SELECT DISTINCT REP
FROM ORDERS
WHERE CUST IN
      (SELECT CUST_NUM
       FROM CUSTOMERS
       WHERE COMPANY LIKE 'Z%' OR COMPANY LIKE 'J%');
```

Screenshot:



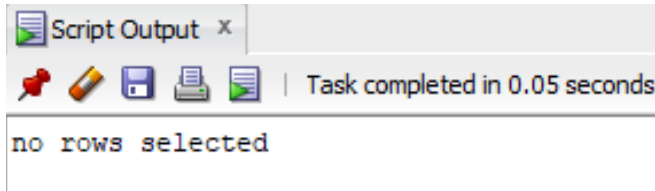
REP
108
105
103
106

11) Use sub query to find the id and the name of every sales rep that represents at least one customer with a credit limit of greater than \$600,000.

Query:

```
SELECT EMPL_NUM, NAME
FROM SALESREPS
WHERE EMPL_NUM IN
  (SELECT CUST_REP
   FROM CUSTOMERS
   WHERE CREDIT_LIMIT > 600000);
```

Screenshot:

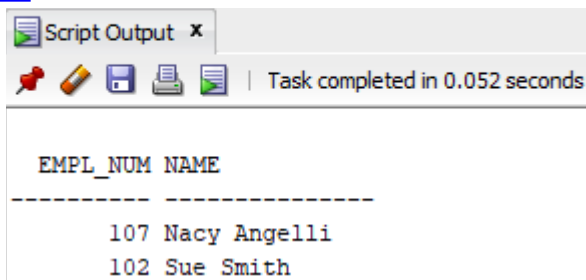


12) Use sub query and keyword “exists” to list the id and the name of the salesreps in which some customers have orders some products in their hiredate.

Query:

```
SELECT EMPL_NUM, NAME
FROM SALESREPS
WHERE EXISTS(
  (SELECT *
   FROM ORDERS
   WHERE ORDER_DATE = HIRE_DATE)
);
```

Screenshot:

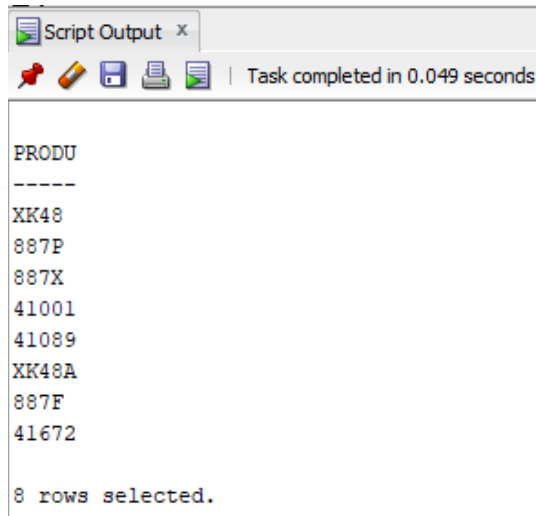


13) List all the products (only Product\_ID) that have never been sold.

Query:

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

Screenshot:



Script Output x

Task completed in 0.049 seconds

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

8 rows selected.
```

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)

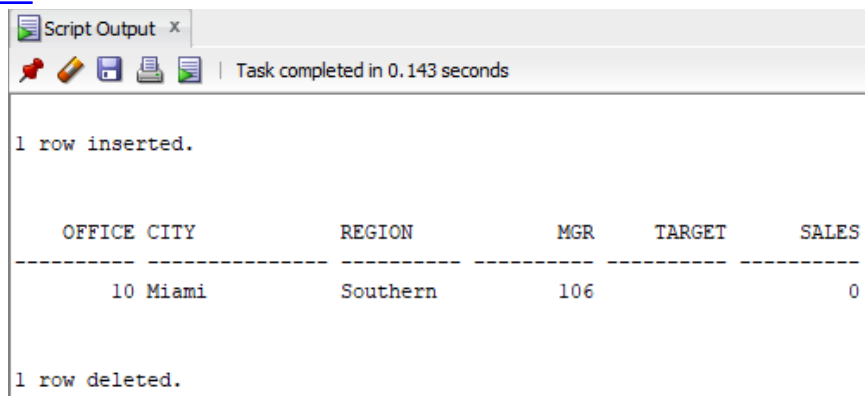
Query:

```
INSERT INTO OFFICES
VALUES (10, 'Miami', 'Southern', 106, '', 0);

SELECT *
FROM OFFICES
WHERE OFFICE = 10;

DELETE FROM OFFICES
WHERE OFFICE = 10;
```

Screenshot:



Script Output x

Task completed in 0.143 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

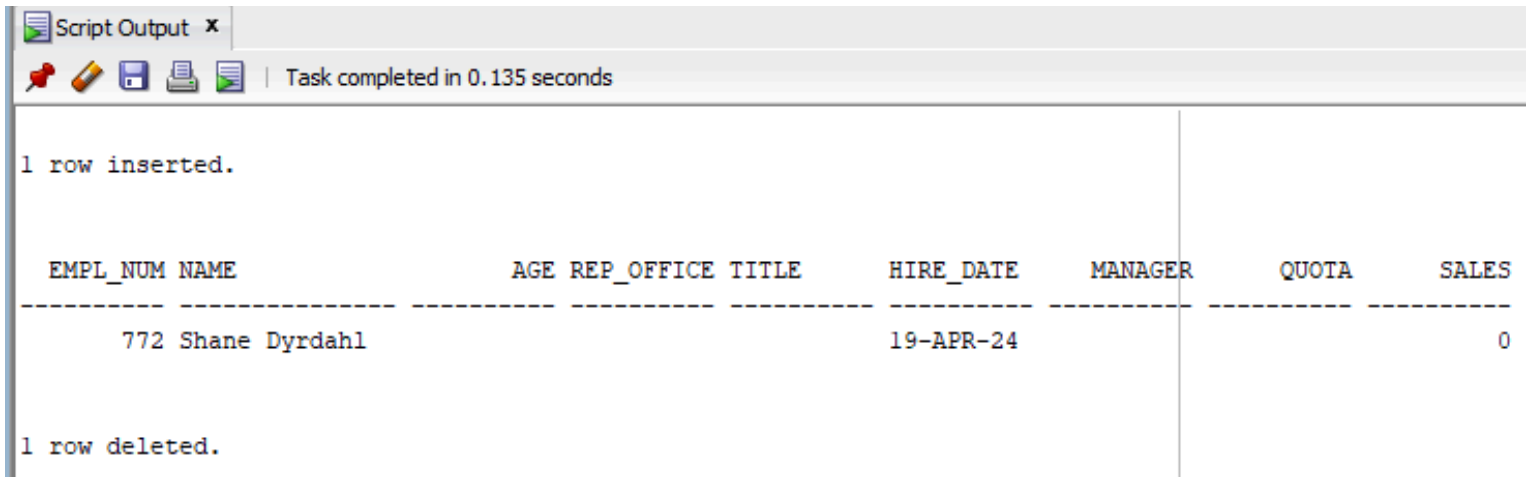
Query:

```
INSERT INTO SALESREPS (EMPL_NUM, NAME, HIRE_DATE, SALES)
VALUES (772, 'Shane Dyrdaahl', SYSDATE, 0);

SELECT *
FROM SALESREPS
WHERE NAME = 'Shane Dyrdaahl';

DELETE FROM SALESREPS
WHERE NAME = 'Shane Dyrdaahl';
```

Screenshot:



Script Output x

Task completed in 0.135 seconds

1 row inserted.

EMPL_NUM	NAME	AGE	REP_OFFICE	TITLE	HIRE_DATE	MANAGER	QUOTA	SALES
772	Shane Dyrdaahl				19-APR-24			0

1 row deleted.

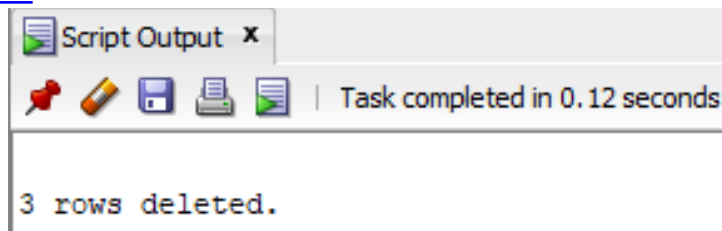
16) Use subquery to Delete all orders for employees 'Dan Roberts'.

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

Query:

```
DELETE FROM ORDERS
WHERE REP IN
(SELECT EMPL_NUM
FROM SALESREPS
WHERE NAME = 'Dan Roberts' AND EMPL_NUM = REP);
```

Screenshot:



Script Output x

Task completed in 0.12 seconds

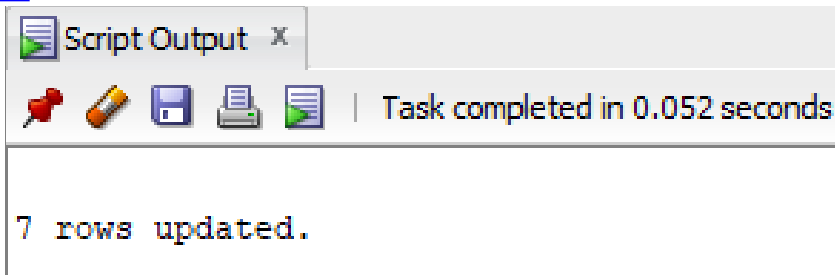
3 rows deleted.

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

Query:

```
UPDATE PRODUCTS
SET PRICE = 0.9 * PRICE
WHERE PRICE >
    (SELECT AVG(PRICE)
     FROM PRODUCTS);
```

Screenshot:

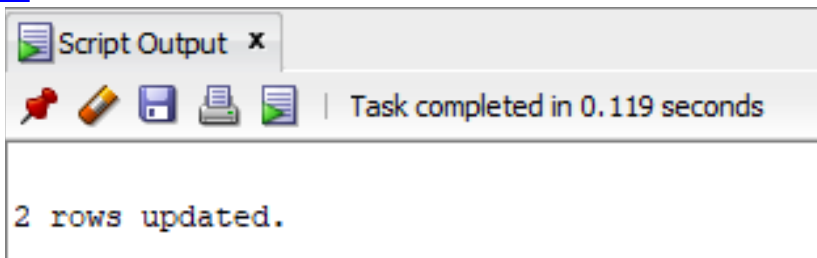


- 18) Set the quota of the salesreps to (average of the quota) + 1500 if they are hired in 2021.  
Recreate the salesreps table after doing the update

Query:

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

Screenshot:



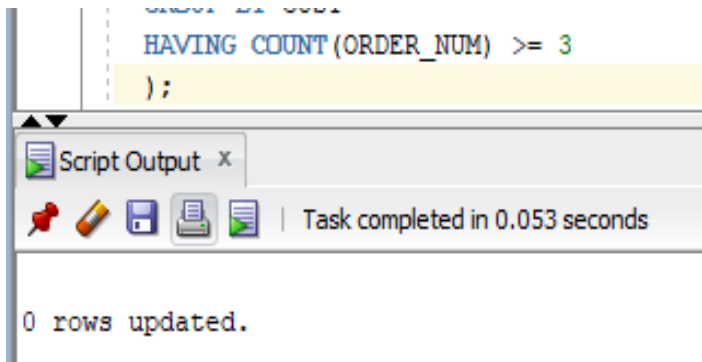


- 19) Increase customers credit limit by 25% for all customers that have 3 or more orders in which each order is more than \$122500.  
Recreate the customers table after doing this update

Query:

```
UPDATE CUSTOMERS
SET CREDIT_LIMIT = CREDIT_LIMIT * 1.25
WHERE CUST_NUM IN (
    SELECT CUST
    FROM ORDERS
    WHERE AMOUNT > 122500
    GROUP BY CUST
    HAVING COUNT(ORDER_NUM) >= 3
);
```

Screenshot:



- 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

Query:

```
UPDATE CUSTOMERS
SET CREDIT_LIMIT = (
    SELECT MAX(AMOUNT) + 1000
    FROM ORDERS
    WHERE CUST = CUST_NUM AND AMOUNT >
    CREDIT_LIMIT
)
WHERE CUST_NUM IN (
    SELECT CUST
    FROM ORDERS
    WHERE AMOUNT > CREDIT_LIMIT
);
```

Screenshot:

