Jaron Jon Javier

Professor Ahmad Hadaegh

CS443 - Database Management Systems

22 November, 2023

Assignment 3

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

SELECT MIN(offices.sales) AS "MIN SALES", MAX(offices.sales) AS "MAX SALES"
FROM offices;

```
SELECT MIN(offices.sales) AS "MIN SALES", MAX(offices.sales) AS "MAX SALES"
FROM offices;
```

Query Result

SQL | All Rows Fetched: 1 in 0.066 seconds

	MIN SALES	MAX SALES
1	186042	835915

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

SELECT COUNT(*) AS “Orders from 2019”
FROM ORDERS
WHERE ORDERS.ORDER_DATE LIKE ‘%19’

```
SELECT COUNT(*) AS "Orders from 2019"
FROM ORDERS
WHERE ORDERS.ORDER_DATE LIKE '%19';
```

Query Result

SQL | All Rows Fetched: 1 in 0.006 seconds

	Orders from 2019
1	4

3. How many different titles in the sales reps table

SELECT COUNT(DISTINCT salesreps.title) AS "Number of Titles"
FROM salesreps
WHERE salesreps.title IS NOT NULL;

```
SELECT COUNT(DISTINCT salesreps.title) AS "Number of Titles"
FROM salesreps
WHERE salesreps.title IS NOT NULL;
```

Query Result

All Rows Fetched: 1 in 0.012 seconds

	Number of Titles
1	3

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

SELECT AVG(salesreps.sales) AS "Average Sales in Office 22"
FROM salesreps,offices
WHERE offices.office=22;

```
SELECT AVG(salesreps.sales) AS "Average Sales in Office 22"
FROM salesreps,offices
WHERE offices.office=22;
```

Query Result

All Rows Fetched: 1 in 0.029 seconds

	Average Sales in Office 22
1	289353.2

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

SELECT AVG(salesreps.sales) AS "Average Sales in each Office"
FROM salesreps,offices
WHERE salesreps.rep_office = offices.office
GROUP BY salesreps.rep_office;

```
SELECT AVG(salesreps.sales) AS "Average Sales in each Office"
FROM salesreps,offices
WHERE salesreps.rep_office = offices.office
GROUP BY salesreps.rep_office;
```

Query Result

SQL | All Rows Fetched: 5 in 0.036 seconds

	Average Sales in each Office
1	346318.5
2	417957.5
3	245014
4	186042
5	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.

```
SELECT salesreps.empl_num, MIN(ORDERS.AMOUNT) AS "MIN AMOUNT", MAX(ORDERS.AMOUNT) AS "MAX AMOUNT", AVG(ORDERS.AMOUNT) AS "AVG AMOUNT"
FROM ORDERS,salesreps
WHERE salesreps.empl_num=ORDERS.REP AND EXTRACT(year FROM ORDERS.ORDER_DATE) BETWEEN 2020 AND 2021
GROUP BY salesreps.empl_num
HAVING COUNT(ORDERS.ORDER_NUM)>1
ORDER BY salesreps.empl_num;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.075 seconds

	EMPL_NUM	MIN AMOUNT	MAX AMOUNT	AVG AMOUNT
1	102	3750	15000	9375
2	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 customers.cust_num, customers.company, customers.credit_limit
FROM customers
WHERE customers.credit_limit < ANY
    (SELECT orders.amount
    FROM orders
    WHERE orders.cust = customers.cust_num);
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.014 seconds

	CUST_NUM	COMPANY	CREDIT_LIMIT
1	2109	Chen Associates	25000
2	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 customers.cust_num, salesreps.empl_num, customers.credit_limit
FROM customers, salesreps
WHERE customers.cust_rep=salesreps.empl_num AND customers.credit_limit > ALL
    (SELECT customers.credit_limit
    FROM customers
    WHERE customers.cust_rep=109);
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.009 seconds

	CUST_NUM	EMPL_NUM	CREDIT_LIMIT
1	2118	108	60000
2	2102	101	65000
3	2101	106	65000
4	2106	102	65000

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

```
SELECT customers.cust_num, salesreps.empl_num, customers.credit_limit
FROM customers, salesreps
WHERE customers.cust_rep=salesreps.empl_num AND customers.credit_limit >
    (SELECT MAX(customers.credit_limit)
    FROM customers
    WHERE customers.cust_rep=109);
```

Script Output | Query... | SQL | All Rows Fetched: 4 in 0.023 seconds

	CUST_NUM	EMPL_NUM	CREDIT_LIMIT
1	2102	101	65000
2	2101	106	65000
3	2106	102	65000
4	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

```
SELECT DISTINCT salesreps.empl_num
FROM salesreps
WHERE salesreps.empl_num IN
    (SELECT customers.cust_rep
    FROM customers
    WHERE customers.company LIKE 'Z%' OR customers.company LIKE 'J%');
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.013 seconds

	EMPL_NUM
1	103
2	106
3	108

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 $60,000

```
SELECT salesreps.empl_num, salesreps.name
FROM salesreps
WHERE salesreps.empl_num IN
    (SELECT customers.cust_rep
    FROM customers
    WHERE(customers.credit_limit >60000.00));
```

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.01 seconds

	EMPL_NUM	NAME
1	101	Dan Roberts
2	102	Sue Smith
3	106	Sam Clark

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 salesreps.empl_num, salesreps.name
FROM salesreps
WHERE EXISTS
    (SELECT orders.rep
    FROM orders
    WHERE salesreps.hire_date = orders.order_date);
```

Script Output | Query...

SQL | All Rows Fetched: 2 in 0.017 seconds

	EMPL_NUM	NAME
1	107	Nacy Angelli
2	102	Sue Smith

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

```
SELECT products.mfr_id, products.product_id
FROM products
WHERE NOT EXISTS (
    SELECT orders.QTY
    FROM orders
    WHERE (orders.product = products.product_id AND orders.mfr = products.mfr_id));
```

Script Output | Query Result

SQL | All Rows Fetched: 8 in 0.013 seconds

	MFR_ID	PRODUCT_ID
1	ACI	41001
2	BIC	41089
3	IMM	887F
4	IMM	887X
5	QSA	XK48A
6	QSA	XK48
7	IMM	887P
8	BIC	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, target, sales)
VALUES (10, 'Miami', 'Southern', 106, 0, 0);
DELETE FROM offices WHERE office =10;
```

Query Result | Script Output

Task completed in 0.086 seconds

```
1 row inserted.

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(name, empl_num, hire_date, sales)
VALUES ('Jaron Javier', 772, sysdate, 0);
DELETE FROM salesreps WHERE empl_num =772;
```

Query Result x | Script Output x

Task completed in 0.045 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

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 price > (SELECT AVG(price) FROM products);
```

18. Let 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 = (SELECT AVG(quota) FROM salesreps WHERE YEAR(hire_date) = 2021) + 1500
WHERE YEAR(hire_date) = 2021;
```

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

```
UPDATE customers
SET credit_limit = credit_limit * 1.25
WHERE cust_num IN (
  SELECT o.cust_num
  FROM orders o
  WHERE o.amount > 500
  GROUP BY o.cust_num
  HAVING COUNT(*) >= 3
);
```

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 credit_limit = (
  SELECT MAX(amount) + 1000
  FROM orders
  WHERE orders.cust_num = customers.cust_num
)
WHERE EXISTS (
  SELECT 1
  FROM orders
  WHERE orders.cust_num = customers.cust_num
  AND orders.amount > customers.credit_limit
);
```