DAY 2 – SQL COMMANDS

CREATE AND INSERT TABLE :

```
CREATE TABLE SALES_TABLE (SALE_ID NUMBER(10) PRIMARY KEY,PRODUCT_ID NUMBER(10),QUANTITY_SOLD NUMBER(10),SALE_DATE DATE,TOTAL_PRICE NUMBER(10,2));
INSERT INTO SALES_TABLE VALUES (1,101,5,TO_DATE('2024-01-01','YYYY-MM-DD'),2500.00);
INSERT INTO SALES_TABLE VALUES (2,102,3,TO_DATE('2024-01-02','YYYY-MM-DD'),900.00);
INSERT INTO SALES_TABLE VALUES (3,103,2,TO_DATE('2024-01-02','YYYY-MM-DD'),60.00);
INSERT INTO SALES_TABLE VALUES (4,104,4,TO_DATE('2024-01-03','YYYY-MM-DD'),80.00);
INSERT INTO SALES_TABLE VALUES (5,105,6,TO_DATE('2024-01-03','YYYY-MM-DD'),90.00);
SELECT *FROM SALES_TABLE;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	01/01/2024	2500
2	102	3	01/02/2024	900
3	103	2	01/02/2024	60
4	104	4	01/03/2024	80
5	105	6	01/03/2024	90

5 rows returned in 0.03 seconds Download

1. Retrieve all columns from the Sales table.

```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	2500
2	102	3	2024-01-02	900
3	103	2	2024-01-02	60
4	104	4	2024-01-03	80
5	105	6	2024-01-03	90

2. Retrieve sale_id and quantity_sold from sales table.

```
SELECT SALE_ID, QUANTITY_SOLD FROM SALES_TABLE ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	QUANTITY_SOLD
1	5
2	3
3	2
4	4
5	6

3. Retrieve the sale_id and sale_date from the Sales table.

```
SELECT SALE_ID, TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE FROM SALES_TABLE ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	SALE_DATE
1	2024-01-01
2	2024-01-02
3	2024-01-02
4	2024-01-03
5	2024-01-03

4. Filter the Sales table to show only sales with a total_price greater than $100.

```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE WHERE TOTAL_PRICE > 100 ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	2500
2	102	3	2024-01-02	900

5. Retrieve the sale_id and total_price from the Sales table for sales made on January 3, 2024.

```
SELECT SALE_ID, TOTAL_PRICE FROM SALES_TABLE WHERE SALE_DATE = TO_DATE('2024-01-03','YYYY-MM-DD');
```

Results Explain Describe Saved SQL History

SALE_ID	TOTAL_PRICE
4	80
5	90

6. Retrieve the sale_id, product_id, and total_price from the Sales table for sales with a quantity_sold greater than 4.

```
SELECT SALE_ID, PRODUCT_ID, TOTAL_PRICE FROM SALES_TABLE WHERE QUANTITY_SOLD > 4 ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	TOTAL_PRICE
1	101	2500
5	105	90

7. Retrieve all columns from the Sales table those sale_id are 1, 3 & 5.

```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE WHERE SALE_ID IN (1,3,5) ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	2500
3	103	2	2024-01-02	60
5	105	6	2024-01-03	90

8. Retrieve all columns from the Sales table those total_price between 90 and 1000.

```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE WHERE TOTAL_PRICE BETWEEN 90 AND 1000;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
2	102	3	2024-01-02	900
5	105	6	2024-01-03	90

9. Retrieve all columns from the Sales table those total_price not between 90 and 1000.

```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE WHERE TOTAL_PRICE NOT BETWEEN 90 AND 1000 ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	2500
3	103	2	2024-01-02	60
4	104	4	2024-01-03	80

10. Retrieve all columns from the Sales table those sale_id are not in 1, 3 & 5.

```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE WHERE SALE_ID NOT IN (1,3,5) ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
2	102	3	2024-01-02	900
4	104	4	2024-01-03	80

11. Update total_price as 500 in the Sales table those sale_id are 1, 3 & 5.

```
UPDATE SALES_TABLE SET TOTAL_PRICE = 500 WHERE SALE_ID IN (1,3,5);
```

Results Explain Describe Saved SQL History

3 row(s) updated.

```
UPDATE SALES_TABLE SET TOTAL_PRICE = 500 WHERE SALE_ID IN (1,3,5);
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	500
2	102	3	2024-01-02	900
3	103	2	2024-01-02	500
4	104	4	2024-01-03	80
5	105	6	2024-01-03	500

12. delete from the Sales table those total_price not between 90 and 1000.

```
DELETE FROM SALES_TABLE WHERE TOTAL_PRICE NOT BETWEEN 90 AND 1000;
```

Results Explain Describe Saved SQL History

1 row(s) deleted.

```
DELETE FROM SALES_TABLE WHERE TOTAL_PRICE NOT BETWEEN 90 AND 1000;
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	500
2	102	3	2024-01-02	900
3	103	2	2024-01-02	500
5	105	6	2024-01-03	500

13. Sort all the records using sale_id column in ascending order.

```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE ORDER BY SALE_ID;
```

Results Explain Describe Saved SQL History

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	500
2	102	3	2024-01-02	900
3	103	2	2024-01-02	500
5	105	6	2024-01-03	500

14. Sort all the records using sale_id column in descending order .

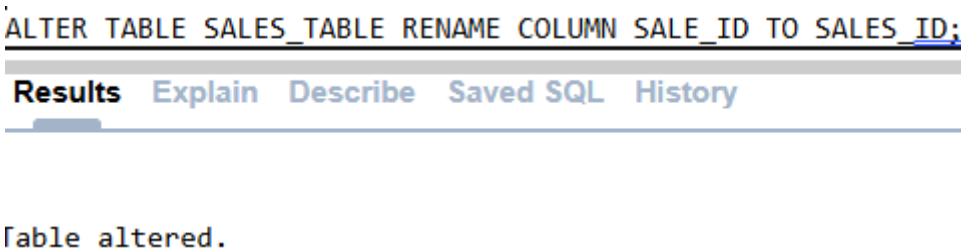
```
SELECT SALE_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE ORDER BY SALE_ID DESC;
```

Results Explain Describe Saved SQL History

Explain

SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
5	105	6	2024-01-03	500
3	103	2	2024-01-02	500
2	102	3	2024-01-02	900
1	101	5	2024-01-01	500

15. Rename the sale_id column as sales_id :

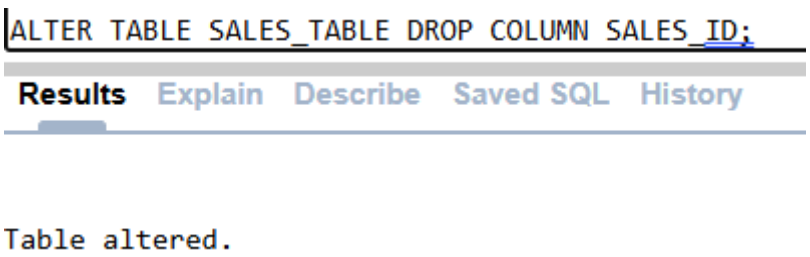



```
ALTER TABLE SALES_TABLE RENAME COLUMN SALE_ID TO SALES_ID;
SELECT SALES_ID,PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE ORDER BY SALES_ID;
```

Results Explain Describe Saved SQL History

SALES_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	500
2	102	3	2024-01-02	900
3	103	2	2024-01-02	500
5	105	6	2024-01-03	500

16. Drop the column sales_id.




```
ALTER TABLE SALES_TABLE DROP COLUMN SALES_ID;
SELECT PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM SALES_TABLE ORDER BY PRODUCT_ID;
```

Results Explain Describe Saved SQL History

PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
101	5	2024-01-01	500
102	3	2024-01-02	900
103	2	2024-01-02	500
105	6	2024-01-03	500

17. Rename the table as tbl_sales.

```
RENAME SALES_TABLE TO TBL_SALES;
```

Results Explain Describe Saved

Statement processed.

```
RENAME SALES_TABLE TO TBL_SALES;
SELECT PRODUCT_ID,QUANTITY_SOLD,TO_CHAR(SALE_DATE, 'YYYY-MM-DD') AS SALE_DATE, TOTAL_PRICE FROM TBL_SALES ORDER BY PRODUCT_ID;
```

Results Explain Describe Saved SQL History

PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
101	5	2024-01-01	500
102	3	2024-01-02	900
103	2	2024-01-02	500
105	6	2024-01-03	500

18. Drop the table.

```
DROP TABLE TBL_SALES;
```

Results Explain Describ

Table dropped.

0.06 seconds