SQL PROJECT

TABLE STRUCTURE

1. Sales Table : The Sales table records information about product sales, including the quantity sold, sale date, and total price for each sale. It serves as a transactional data source for analyzing sales trends.
2. Products Table The Products table contains details about products, including their names, categories, and unit prices. It provides reference data for linking product information to sales transactions.

Creating Table SALES

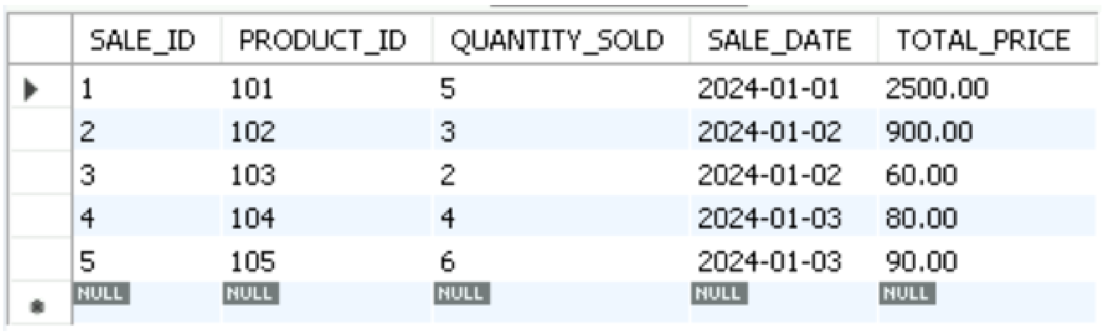
```
CREATE TABLE SALES(SALE_ID INT PRIMARY KEY, PRODUCT_ID INT,

                    QUANTITY_SOLD INT, SALE_DATE DATE,

                    TOTAL_PRICE DECIMAL(10,2))
```

Inserting Values Into Table SALES

```
INSERT INTO SALES VALUES(1, 101, 5, '2024-01-01', 2500.00);

INSERT INTO SALES VALUES(2, 102, 3, '2024-01-02', 900.00);

INSERT INTO SALES VALUES(3, 103, 2, '2024-01-02', 60.00);

INSERT INTO SALES VALUES(4, 104, 4, '2024-01-03', 80.00);

INSERT INTO SALES VALUES(5, 105, 6, '2024-01-03', 90.00);
```

Viewing SALES Table

```
SELECT * FROM SALES;
```



SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
1	101	5	2024-01-01	2500.00
2	102	3	2024-01-02	900.00
3	103	2	2024-01-02	60.00
4	104	4	2024-01-03	80.00
5	105	6	2024-01-03	90.00
NULL	NULL	NULL	NULL	NULL

Creating Table PRODUCTS

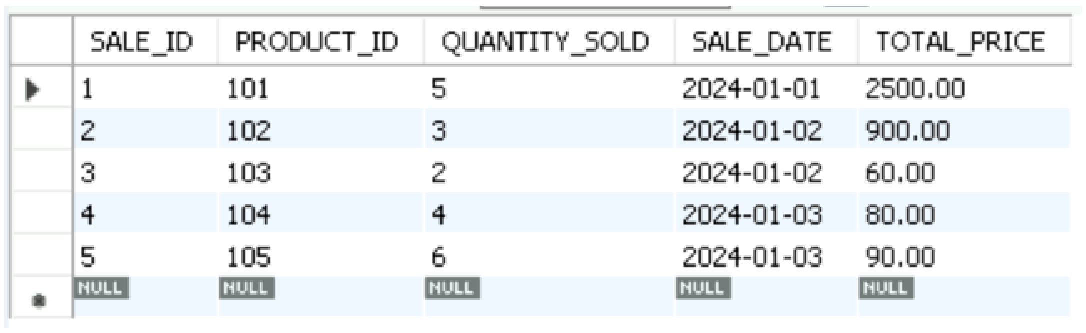
```
CREATE TABLE PRODUCTS(PRODUCT_ID INT PRIMARY KEY,
                      PRODUCT_NAME VARCHAR(100),
                      CATEGORY VARCHAR(50),
                      UNIT_PRICE DECIMAL(10,2));
```

Inserting Values Into Table PRODUCTS

```
INSERT INTO PRODUCTS VALUES(101, 'LAPTOP', 'ELECTRONICS', 500.00);

INSERT INTO PRODUCTS VALUES(102, 'SMARTPHONE', 'ELECTRONICS', 300.00);

INSERT INTO PRODUCTS VALUES(103, 'HEADPHONE', 'ELECTRONICS', 30.00);

INSERT INTO PRODUCTS VALUES(104, 'KEYBOARD', 'ELECTRONICS', 20.00);

INSERT INTO PRODUCTS VALUES(105, 'MOUSE', 'ELECTRONICS', 15.00);
```

Viewing PRODUCTS Table

```
SELECT * FROM PRODUCTS;
```



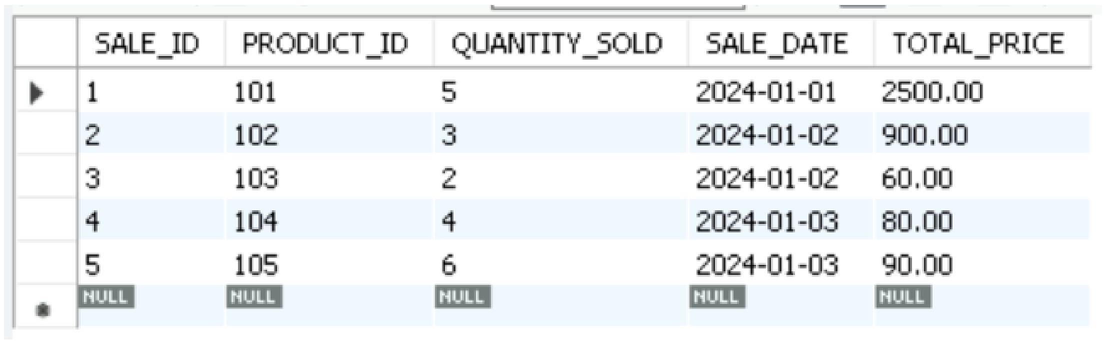
	SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
▶	1	101	5	2024-01-01	2500.00
	2	102	3	2024-01-02	900.00
	3	103	2	2024-01-02	60.00
	4	104	4	2024-01-03	80.00
	5	105	6	2024-01-03	90.00
*	NULL	NULL	NULL	NULL	NULL

QUESTIONS

Q1: Retrieve the product details (name, category, unit price) for products that have a quantity sold greater than the average quantity sold across all products.

ANS:

SELECT PRODUCT_NAME, CATEGORY, UNIT_PRICE FROM PRODUCTS WHERE PRODUCT_ID IN (SELECT PRODUCT_ID FROM SALES GROUP BY PRODUCT_ID HAVING SUM(QUANTITY_SOLD) > (SELECT AVG(TOTAL_SOLD) FROM (SELECT SUM(QUANTITY_SOLD) AS TOTAL_SOLD FROM SALES GROUP BY PRODUCT_ID) AS PRODUCT_SALES));



	SALE_ID	PRODUCT_ID	QUANTITY_SOLD	SALE_DATE	TOTAL_PRICE
▶	1	101	5	2024-01-01	2500.00
	2	102	3	2024-01-02	900.00
	3	103	2	2024-01-02	60.00
	4	104	4	2024-01-03	80.00
	5	105	6	2024-01-03	90.00
*	NULL	NULL	NULL	NULL	NULL

Q2: Add a foreign key constraint to the Sales table that references the product_id column in the Products table.

ANS:

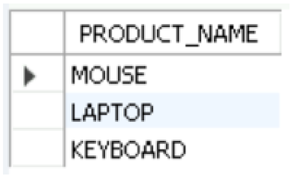
ALTER TABLE SALES ADD CONSTRAINT FOREIGN_KEY FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCTS(PRODUCT_ID);

Q3: Create a view named Top_Products that lists the top 3 products based on the total quantity sold.

ANS:

```
CREATE VIEW TOP_PRODUCTS AS SELECT P.PRODUCT_NAME FROM PRODUCTS P JOIN
SALES S ON P.PRODUCT_ID = S.PRODUCT_ID GROUP BY P.PRODUCT_ID,
P.PRODUCT_NAME ORDER BY SUM(QUANTITY_SOLD) DESC LIMIT 3;
```

```
SELECT * FROM TOP_PRODUCTS;
```



	PRODUCT_NAME
▶	MOUSE
	LAPTOP
	KEYBOARD